




PROYECTO DE DAM GYMWIZZ

Aplicación de gestión de clientes para gimnasios.

Franco Ruben Milazzo - Y8886214V

19/6/2024



INDICE

1.RESUMEN.....	3
2.PALABRAS CLAVE.....	3
3.INTRODUCCION.....	4
4.OBJETIVOS.....	5
5.ANÁLISIS DEL CONTEXTO.....	6
5.1 Análisis del contexto.....	6
5.2 Innovación.....	8
6.DISEÑO.....	9
6.1. DISEÑO DE LA INTERFAZ.....	9
6.2 Modelo de datos.....	18
6.3Diagrama de Clases.....	19
6.4 Despliegue.....	20
7. PLANIFICACIÓN.....	21
7.1. DIAGRAMA DE GANTT.....	23
7.2 Definición de Recursos y Logística Necesarios para Cada Actividad:.....	24
8.IMPLEMENTACION.....	29
ENVÍO DE RUTINAS POR CORREO.....	29
GRAFICO DEL DASHBOARD.....	30
TABLA DE CLIENTES.....	31
FILTRO DE CLIENTES EN LA TABLA.....	32
ABRIR DATOS DEL CLIENTE.....	32
MÉTODO QUE HASHEA LA CONTRASEÑA.....	33
BOTON LOGIN.....	34
MÉTODO PARA VALIDAR EL LOGIN DEL USUARIO.....	34
BOTÓN DEL SOPORTE EN EL LOGIN Y REGISTRO.....	35
GENERAR CODIGO DE RECUPERACION.....	35
RECUPERAR CONTRASEÑA.....	35
REGISTRAR UN PAGO DE UN CLIENTE.....	37
BORRAR UN CLIENTE.....	38
9.PUESTA EN MARCHA, EXPLOTACIÓN.....	38
9.PUESTA EN MARCHA, EXPLOTACIÓN.....	39
Cambios de Configuración, Seguridad y Legalidad antes de la Puesta en Producción..	39
10.Prueba y control de calidad.....	41
10.1 Prueba de Registro e Inicio de Sesión.....	41
10.2 Prueba de Registro de cliente y edición de datos.....	43
11. Gestión económica o plan de empresa.....	46
Número de Empresas y Tamaño:.....	46
Aspectos Culturales, Sociales y Medioambientales:.....	46
El Producto o Servicio.....	49
12.Conclusiones y valoración personal.....	50
13.BIBLIOGRAFIA.....	51

1.RESUMEN

GymWizz es una aplicación de gestión diseñada para facilitar la administración de gimnasios resolviendo el problema de administrar eficientemente los datos de los clientes, las finanzas y la comunicación personalizada. Innovador en su enfoque, GymWizz integra un sistema de registro e inicio de sesión para administradores de gimnasios, un completo dashboard que muestra ganancias mensuales y datos relevantes, incluida una lista detallada de clientes. Destaca por su capacidad para identificar y gestionar los clientes morosos, ofreciendo una interfaz gráfica atractiva y fácil de usar.

La característica distintiva de GymWizz radica en su capacidad para enviar rutinas de entrenamiento personalizadas por correo electrónico a los nuevos clientes, basadas en sus datos personales. Esta característica busca empatizar con el cliente y añade un toque innovador al proyecto al mejorar la experiencia y fomentar su compromiso con el gimnasio. En resumen, GymWizz aborda eficazmente los desafíos de gestión de clientes en gimnasios, ofreciendo una solución integral que mejora la eficiencia operativa y la satisfacción del usuario.

2.PALABRAS CLAVE

Gestión de gimnasios, Aplicación de gestión de clientes, Administración de clientes de gimnasio, Dashboard de gimnasio, Registro de gimnasios, Finanzas de gimnasios, Rutinas de entrenamiento personalizadas,, Administración de membresías de gimnasios, Estadísticas de gimnasios, Aplicacion en JavaFx para la gestion de clientes con base de datos.

3.INTRODUCCION

En el mundo de la gestión de gimnasios, la necesidad de una herramienta integral y eficiente para administrar clientes, finanzas y comunicaciones personalizadas se vuelve cada vez más evidente. Como alguien que ha estado involucrado en el mundo de la administración de gimnasios, me encontré constantemente enfrentando desafíos significativos en la gestión diaria. Fue esta observación de las dificultades y la falta de soluciones satisfactorias lo que me inspiró a concebir y desarrollar el proyecto GymWizz.

La gestión de un gimnasio conlleva una serie de tareas complejas, desde mantener un registro de los clientes y suscripciones hasta garantizar que las finanzas estén en orden y que la comunicación con los clientes sea efectiva y personalizada. En mi experiencia diaria, noté la falta de una solución integral que abordara todas estas necesidades de manera eficiente y amigable.

La motivación detrás de la creación de GymWizz fue resolver estos problemas de gestión de manera efectiva y proporcionar a los propietarios de gimnasios una herramienta poderosa que simplifique su trabajo diario. Mis padres, que tenían un gimnasio, enfrentaron desafíos similares en la administración de su negocio. Fue esta experiencia personal la que me impulsó a buscar una solución que ayudará a todos los propietarios de gimnasios que enfrentan problemas similares.

Este proyecto no solo es una solución empresarial, sino también una respuesta a una necesidad clara y presente en el campo de la administración de gimnasios.

A lo largo de esta documentación, presentaré GymWizz, una aplicación diseñada para facilitar la gestión de gimnasios, abordando los desafíos identificados y proporcionando una solución integral que mejora la eficiencia operativa y la satisfacción del usuario. Con GymWizz, la administración de gimnasios se convierte en una tarea más fluida y gratificante, permitiendo a los propietarios de gimnasios centrarse en brindar una experiencia excepcional a sus clientes.

4.OBJETIVOS

Facilitar la administración de gimnasios:

Proporcionar a los propietarios de gimnasios una herramienta integral para gestionar de manera eficiente los datos de los clientes, las finanzas y la comunicación personalizada.

Agilizar el proceso de gestión de clientes:

Implementar un sistema de registro e inicio de sesión que permita a los administradores de gimnasios acceder fácilmente a la plataforma y administrar los datos de los clientes de manera segura.

Mejorar la toma de decisiones a través de un dashboard completo:

Desarrollar un dashboard intuitivo que ofrezca una visión general de las estadísticas clave del gimnasio, como las ganancias mensuales, los nuevos clientes y la lista de clientes, facilitando así la toma de decisiones informadas.

Identificar y gestionar clientes morosos de manera eficiente:

Integrar una función que permita identificar automáticamente a los clientes morosos y proporcionar herramientas para gestionar los pagos pendientes de manera efectiva.

Ofrecer rutinas de entrenamiento personalizadas para mejorar la experiencia del cliente:

Desarrollar un sistema automatizado que envíe rutinas de entrenamiento personalizadas por correo electrónico a los nuevos clientes, basadas en sus datos personales.

Garantizar un excelente servicio de soporte al cliente:

Incorporar un botón de soporte que permita a los usuarios enviar mensajes directamente al equipo de soporte en caso de problemas con la aplicación, asegurando una rápida resolución de problemas y una experiencia fluida para los usuarios.

5. ANÁLISIS DEL CONTEXTO

5.1 Análisis del contexto

En el sector de la gestión de gimnasios, existen varias soluciones que compiten directamente con GymWizz. A continuación, se presenta un análisis de la competencia:

- ❖ GymMaster(GymMaster Limited): Ofrece una amplia gama de funciones de gestión de gimnasios, incluida la administración de clientes, programación de clases, pagos y marketing.
 - Lo mejor: Amplia gama de funciones de gestión, incluyendo programación de clases, seguimiento de clientes y marketing.
 - Lo peor: Carece de integración con sistemas de facturación y contabilidad.
 - Precio: Desde \$99 al mes.
- ❖ Zen Planner(Zen Planner LLC): Proporciona una plataforma completa para la gestión de gimnasios, que incluye programación de clases, seguimiento de clientes, procesamiento de pagos y herramientas de marketing.
 - Lo mejor: Plataforma completa que cubre todas las necesidades de gestión de gimnasios.
 - Lo peor: Carece de ciertas funcionalidades avanzadas de análisis de datos.
 - Precio: Desde \$117 al mes.
- ❖ Mindbody(Mindbody, Inc.): Ofrece soluciones de gestión para gimnasios y estudios de fitness, incluyendo programación de clases, gestión de clientes, pagos y marketing.
 - Lo mejor: Solución de gestión completa con una amplia gama de herramientas.
 - Lo peor: Complejidad en la interfaz de usuario y precio elevado.
 - Precio: Desde \$129 al mes.

Estas empresas tienen como clientes a gimnasios de diferentes tamaños, Entrenadores personales y profesionales del fitness que gestionan sus propios clientes.

Análisis DAFO frente a la competencia:

Debilidades:

- Menos funcionalidades avanzadas de análisis de datos en comparación con algunos competidores.
- Falta de integración con gestión de horario de personal training o programación de actividades.

Fortalezas:

- Interfaz intuitiva y fácil de usar.
- Envío automático de rutinas de entrenamiento personalizadas.
- Soporte integrado para resolver problemas rápidamente.
- Precio competitivo en el mercado.

Oportunidades:

- No hay muchas aplicaciones en el mercado con un precio accesible.
- El aumento exponencial del sector fitness en los últimos años lleva a una alta demanda de este tipo de soluciones.

Amenazas:

- Competencia fuerte en el mercado de la gestión de gimnasios.
- Cambios en las preferencias del usuario y en las tecnologías pueden afectar la demanda de la aplicación.

5.2 Innovación

GymWizz ofrece varias innovaciones que cubren necesidades existentes en el mercado:

- Envío automático de rutinas de entrenamiento personalizadas por correo electrónico a los nuevos clientes, basadas en sus datos personales y preferencias de entrenamiento.
- Integración de un sistema de soporte directo dentro de la aplicación para resolver problemas de manera rápida y eficiente.
- Una interfaz gráfica intuitiva que cualquier usuario puede entender y utilizar

Mejora la eficacia, la eficiencia o ambas a la vez:

GymWizz simplifica y optimiza el proceso de administración de gimnasios, mejorando la eficiencia operativa al ofrecer un conjunto completo de herramientas en una sola plataforma.

Supone un cambio de paradigma en el sector:

La automatización del envío de rutinas de entrenamiento personalizadas por correo electrónico supone un cambio significativo en la forma en que los gimnasios se relacionan con sus clientes, ofreciendo una experiencia más personalizada y atractiva desde el primer día.

Soluciones innovadoras aplicadas:

GymWizz emplea técnicas avanzadas de procesamiento de datos y automatización para ofrecer un sistema integral de gestión de clientes de gimnasios.

Utiliza algoritmos de personalización para enviar rutinas de entrenamiento adaptadas a las necesidades individuales de cada cliente.

Implementa un sistema de soporte integrado para proporcionar una asistencia rápida y efectiva a los usuarios.

6.DISEÑO

La interfaz gráfica es fundamental en el diseño de una aplicación, ya que es el punto de contacto directo entre el usuario y la aplicación. Por este motivo, se ha desarrollado una interfaz que sea clara, intuitiva y de fácil manejo, con el propósito de garantizar una experiencia de usuario satisfactoria.

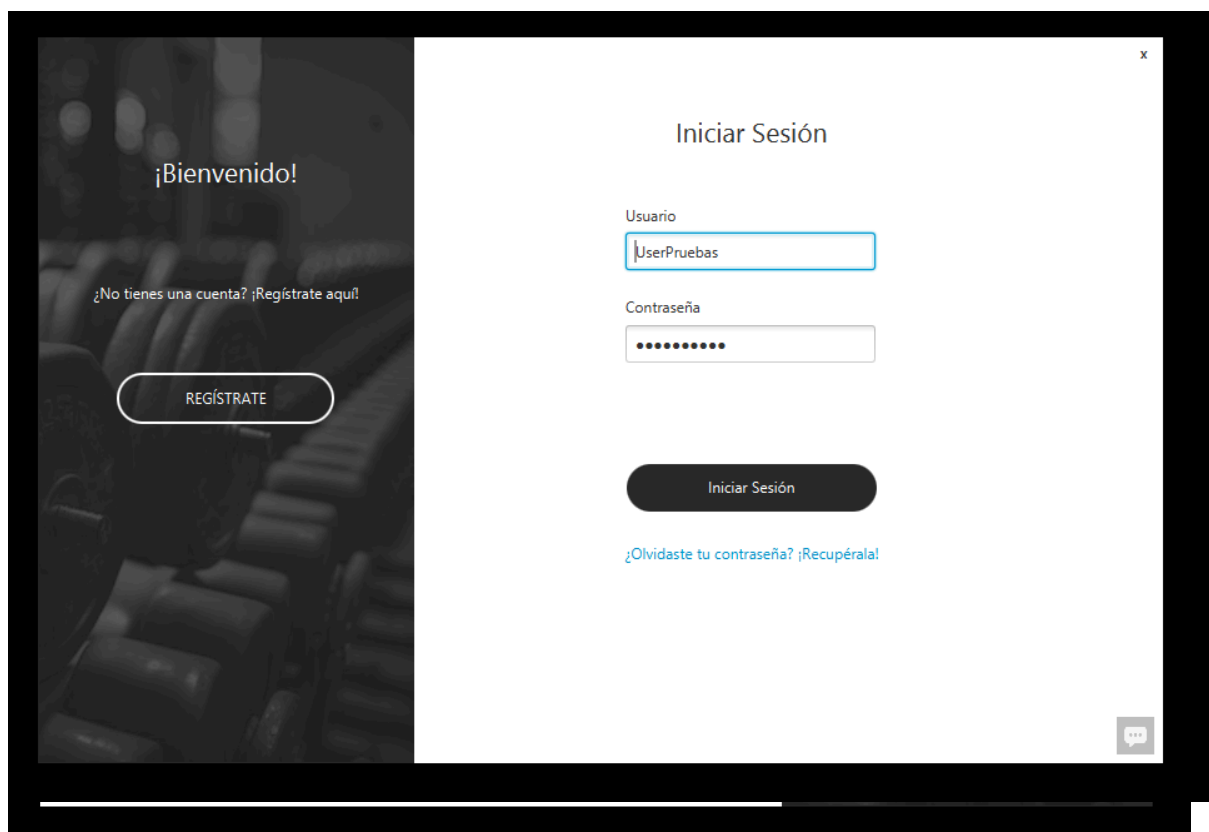
6.1. DISEÑO DE LA INTERFAZ

Login y Registro

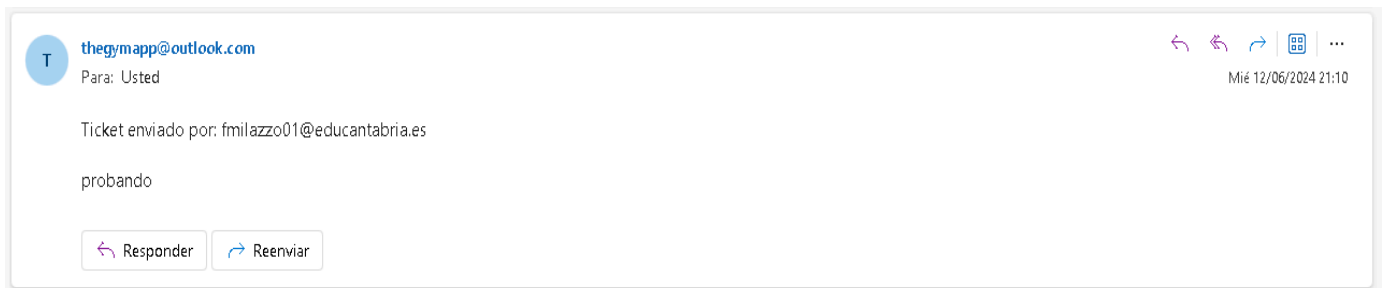
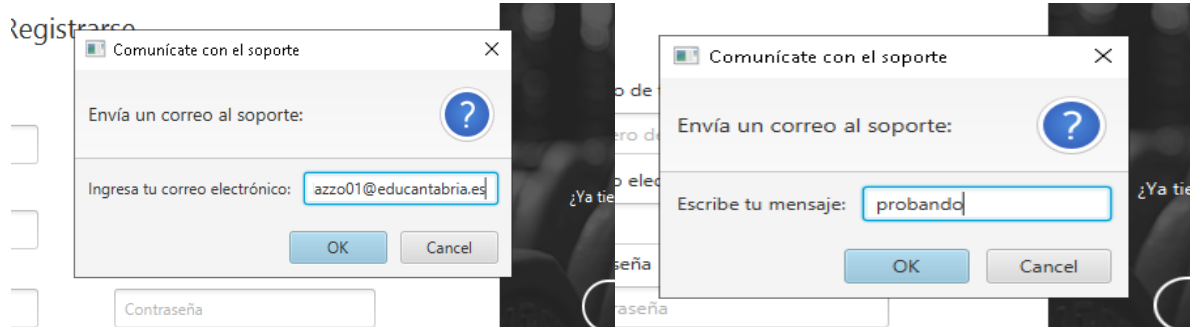
Cuando estés en la pantalla de inicio de sesión y hagas clic en el botón de registrarse, se activará una animación que moverá el panel de la izquierda hacia la derecha, revelando así el formulario de registro.

Del mismo modo, al hacer clic en el botón de inicio de sesión estando en la pantalla de registro, se aplicará una animación que devolverá el panel a su posición original, permitiendo al usuario alternar de manera fluida entre el proceso de inicio de sesión y el de registro.

El inicio de sesión cuenta con los campos de usuario y contraseña, el botón de iniciar sesión



En la pantalla de inicio de sesión y registro, también contamos con un botón de soporte. Al hacer clic en este botón, se solicitará al usuario que introduzca su correo electrónico. Esto se utilizará para identificar al remitente del ticket de soporte, ya que aún no hemos iniciado sesión. Después de ingresar el correo electrónico, se pedirá al usuario que escriba su mensaje de soporte.



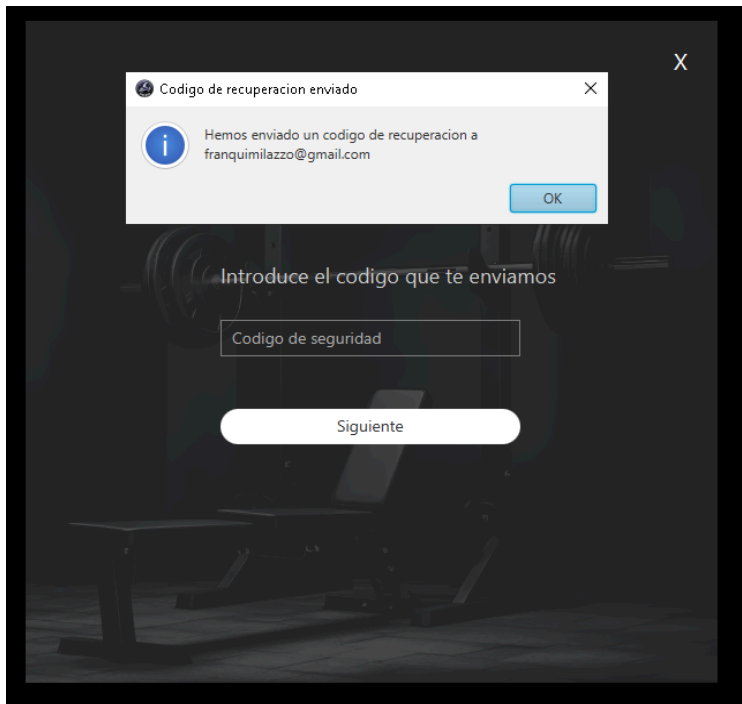
Recuperar contraseña

En caso de que el usuario olvide su contraseña, solo necesita hacer clic en el enlace de "Recuperar Contraseña". Esto abrirá una ventana donde deberá ingresar su correo electrónico para recibir un código de recuperación.

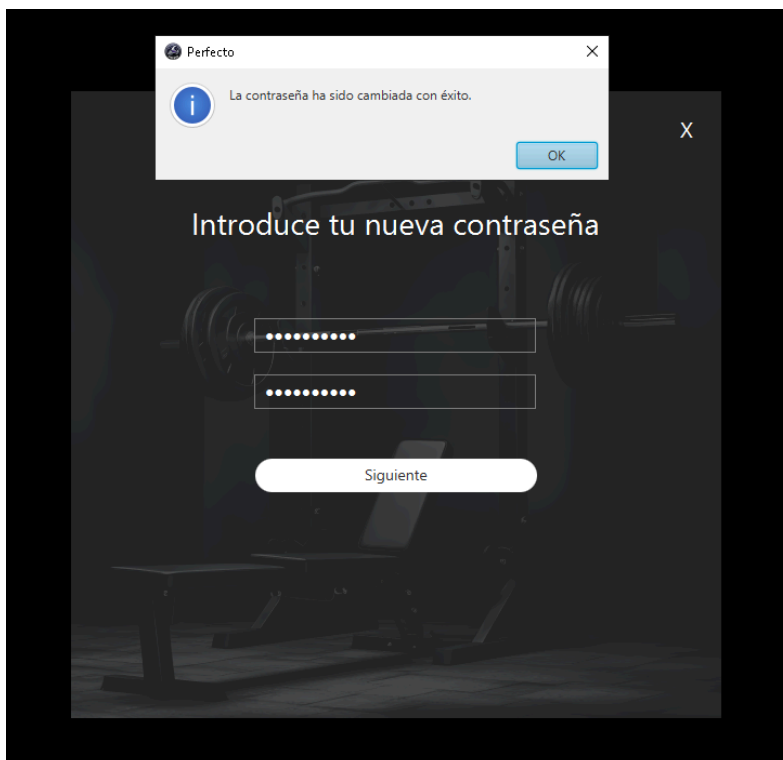


Cuando completes tu correo electrónico, recibirás un mensaje con un código de recuperación generado aleatoriamente. Este código deberás introducirlo en la ventana correspondiente para poder continuar con el proceso de recuperación de contraseña.



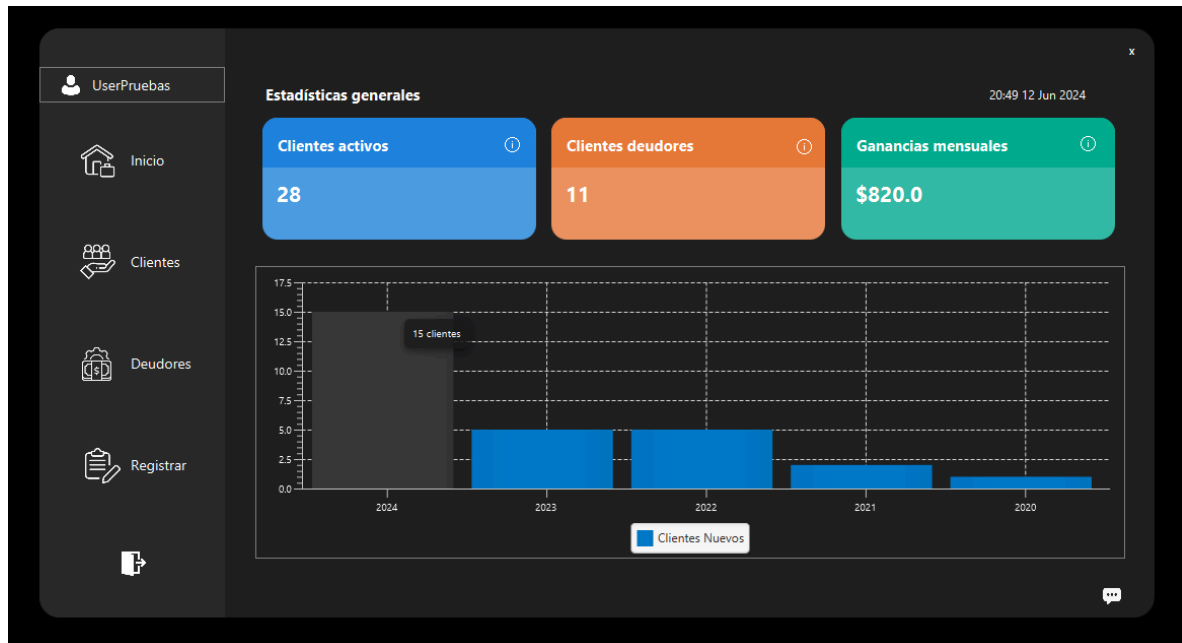


Y solo si introducimos el código correctamente nos dejara colocar una nueva contraseña con un formato seguro y la aplicación volverá automáticamente al login



Dashboard del Inicio:

Brinda al usuario una vista general y rápida del estado actual de su gimnasio. Aquí podrá ver la cantidad de clientes activos, los clientes que no han pagado su membresía y las ganancias del mes. Además, incluye un gráfico que muestra la evolución de la clientela, con la cantidad de nuevos inscritos en los últimos años.



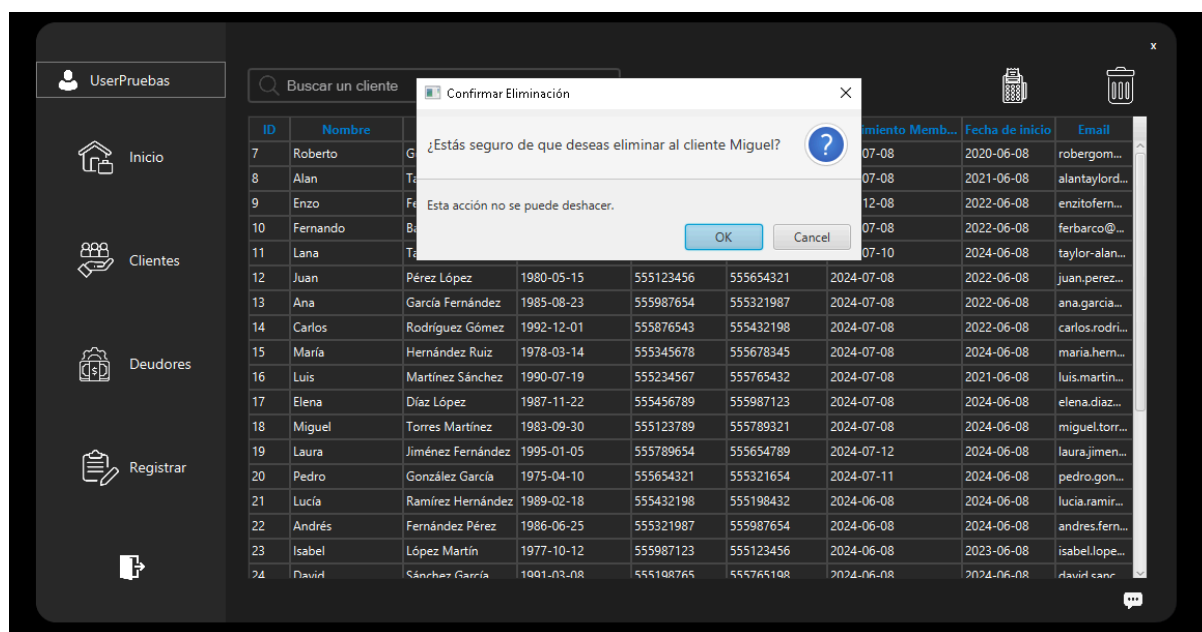
Lista de todos los clientes:

Aquí el usuario podrá visualizar una lista completa de todos los clientes del gimnasio, junto con sus datos más relevantes, tales como su nombre y la fecha de vencimiento de la membresía. Para registrar el pago de un cliente el usuario puede clickear el cliente que quiera y darle al botón con la caja registradora, ahí se desplegará un menú donde podrá seleccionar que tipo de membresía adquirió el cliente seleccionado.

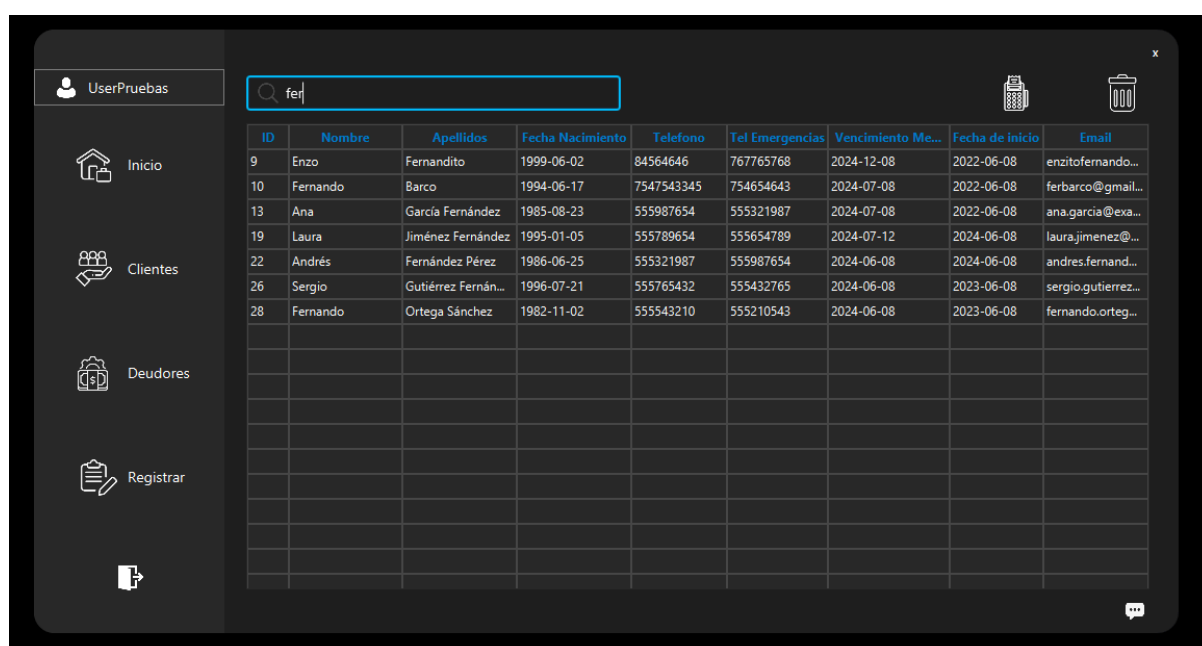
The screenshot shows a table of all clients with columns for ID, Nombre, Apellidos, Fecha Nacimiento, Telefono, Tel Emergencias, Vencimiento Memb..., Fecha de inicio, and Email. A modal titled 'Seleccionar Tipo de Pago' is open, showing options for 'Mensualidad', 'Trimestre', 'Semestre', and 'Año'.

ID	Nombre	Apellidos	Fecha Nacimiento	Telefono	Tel Emergencias	Vencimiento Memb...	Fecha de inicio	Email
7	Roberto	Gomez	1994-06-07	88888999	5566577858	2024-07-08	2020-06-08	robergom...
8	Alan	Taylor	2002-06-19	9999424535	6546546456	2024-07-08	2021-06-08	alantaylor...
9	Enzo	Fernando	1999-06-02	84564646	767765768	2024-12-08	2022-06-08	enzofern...
10	Fernando	Barco					2022-06-08	ferbarco@...
11	Lana	Taylor					2024-06-08	taylor-alan...
12	Juan	Pérez López					2022-06-08	juan.perez...
13	Ana	García Fernández					2022-06-08	ana.garcia...
14	Carlos	Rodríguez Gómez					2022-06-08	carlos.rodri...
15	María	Hernández Ruiz					2024-06-08	maria.hern...
16	Luis	Martínez Sánchez					2021-06-08	luis.martin...
17	Elena	Díaz López					2024-06-08	elena.diaz...
18	Miguel	Torres Martínez	1983-09-30	555123789	555789321	2024-07-08	2024-06-08	miguel.torr...
19	Laura	Jiménez Fernández	1995-01-05	555789654	555654789	2024-07-08	2024-06-08	laura.jimen...
20	Pedro	González García	1975-04-10	555654321	555321654	2024-07-11	2024-06-08	pedro.gon...
21	Lucía	Ramírez Hernández	1989-02-18	555432198	555198432	2024-06-08	2024-06-08	lucia.ramir...
22	Andrés	Fernández Pérez	1986-06-25	555321987	555987654	2024-06-08	2024-06-08	andres.fern...
23	Isabel	López Martín	1977-10-12	555987123	555123456	2024-06-08	2023-06-08	isabel.lope...
24	David	Sánchez García	1991-02-08	555198765	555765198	2024-06-08	2024-06-08	david.sanc...

Para eliminar un usuario, el proceso es igualmente sencillo. Basta con hacer clic en el usuario deseado y seleccionar el botón con la imagen de la papelera. A continuación, se desplegará otro menú para confirmar la acción.



Además, la lista cuenta con un práctico buscador que permite filtrar los clientes por nombre y apellidos, proporcionando así una herramienta adicional para una gestión más ágil y eficiente.



Lista de deudores:

En esta sección, se muestran únicamente los clientes cuyas membresías han vencido, lo que facilita un seguimiento más efectivo de aquellos que aún adeudan sus pagos.

ID	Nombre	Apellidos	Fecha Nacimiento	Telefono	Tel Emergen...	Vencimient...	Fecha de ...	Email
21	Lucía	Ramírez Hernández	1989-02-18	555432198	555198432	2024-06-08	2024-06-08	lucia.ramirez@exam...
22	Andrés	Fernández Pérez	1986-06-25	555321987	555987654	2024-06-08	2024-06-08	andres.fernandez@...
23	Isabel	López Martín	1977-10-12	555987123	555123456	2024-06-08	2023-06-08	isabel.lopez@exam...
24	David	Sánchez García	1991-03-08	555198765	555765198	2024-06-08	2024-06-08	david.sanchez@exa...
25	Natalia	Ruiz López	1984-12-17	555876321	555321876	2024-06-08	2023-06-08	natalia.ruiz@exampl...
26	Sergio	Gutiérrez Fernández	1996-07-21	555765432	555432765	2024-06-08	2023-06-08	sergio.gutierrez@ex...
27	Patricia	Castro Gómez	1979-05-27	555654789	555789654	2024-06-08	2023-06-08	patricia.castro@exa...
28	Fernando	Ortega Sánchez	1982-11-02	555543210	555210543	2024-06-08	2023-06-08	fernando.ortega@e...
29	Clara	Vargas Ramírez	1993-06-14	555432987	555987432	2024-06-08	2024-06-08	clara.vargas@exam...
30	Jorge	Flores Ruiz	1988-09-05	555321654	555654321	2024-06-08	2024-06-08	jorge.flores@exam...
31	Alba	Moreno García	1997-12-29	555789123	555123789	2024-06-08	2024-06-08	alba.moreno@exam...

Ficha de cliente

Al hacer doble clic sobre un cliente, ya sea en la lista de clientes normal o en la lista de deudores, se abrirá en una nueva ventana la ficha del cliente seleccionado. Aquí, el usuario tendrá la posibilidad de modificar los datos del cliente haciendo clic en el botón con el lápiz, que habilita la edición de los campos. Si desea guardar los cambios realizados, simplemente deberá hacer clic en el botón con el tilde. En caso contrario, puede cancelar la operación haciendo clic en el botón con la cruz.

UserPruebas

Inicio

Clientes

Deudores

Registrar

Nombre

Fernando

Apellidos

Barco

Número de teléfono

7547543345

Fecha de inscripción

6/8/2022

Correo electrónico

ferbarco@gmail.com

Teléfono de emergencias

754654643

Fecha de nacimiento

6/17/1994

✎

✓

✕

Lista de clientes recientes

Fecha de inicio	Email
20-06-08	robergomez@gm...
21-06-08	alantaylor@mai...
22-06-08	enzitofernando@g...
22-06-08	ferbarco@gmail.c...
24-06-08	taylor-alan2002@...
22-06-08	juan.perez@exam...
22-06-08	ana.garcia@exam...
22-06-08	carlos.rodriguez@...
24-06-08	maria.hernandez...
21-06-08	luis.martinez@exa...
24-06-08	elena.diaz@exam...
24-06-08	miguel.torres@exa...
24-06-08	laura.jimenez@exa...
24-06-08	pedro.gonzalez@e...
24-06-08	lucia.ramirez@exa...
24-06-08	andres.fernandez...
23-06-08	isabel.lopez@exa...
24-06-08	david.sanchez@ex...

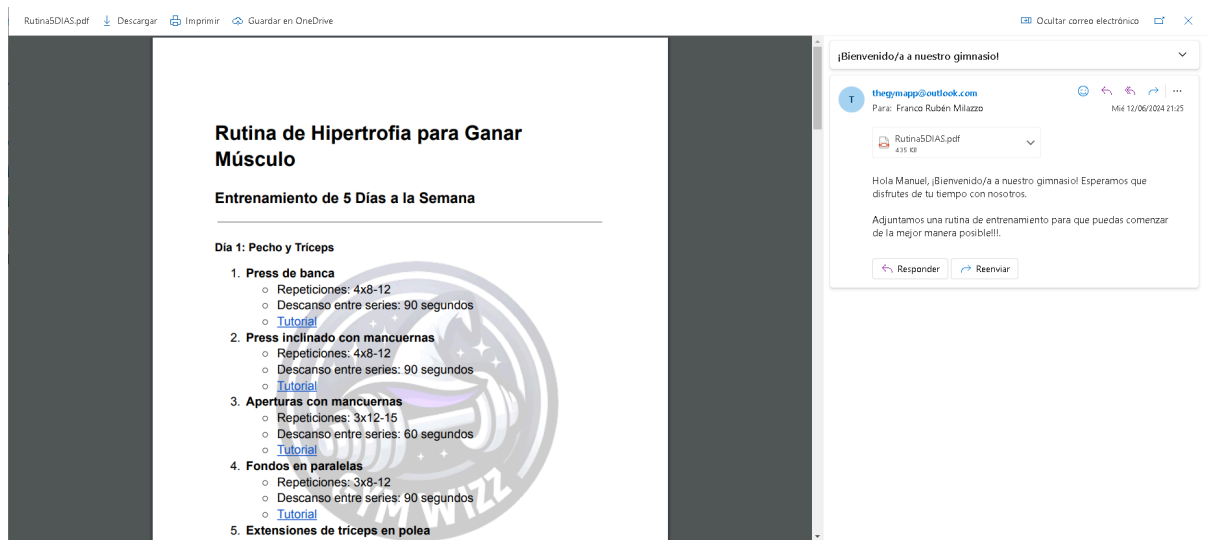
Menú de registro de un cliente:

Este menú permite al usuario agregar un nuevo cliente a la base de datos del sistema.

The screenshot shows a web application interface for registering a new client. On the left is a dark sidebar with a user profile 'UserPruebas' and four menu items: 'Inicio' (home icon), 'Clientes' (group of people icon), 'Deudores' (wallet icon), and 'Registrar' (clipboard icon). The main area is titled 'Registro de cliente' and contains a form with the following fields: 'Nombre' (Manuel), 'Correo electrónico' (manuelgonzales@example.com), 'Apellidos' (Gonzales), 'Teléfono de emergencias' (622233518), 'Número de teléfono' (622233510), 'Fecha de nacimiento' (5/28/2001 with a calendar icon), 'Genero' (Masculino dropdown), and 'Días de entrenamiento' (5 dropdown). A 'Registrar Cliente' button is at the bottom. A small chat bubble icon is in the bottom right corner.

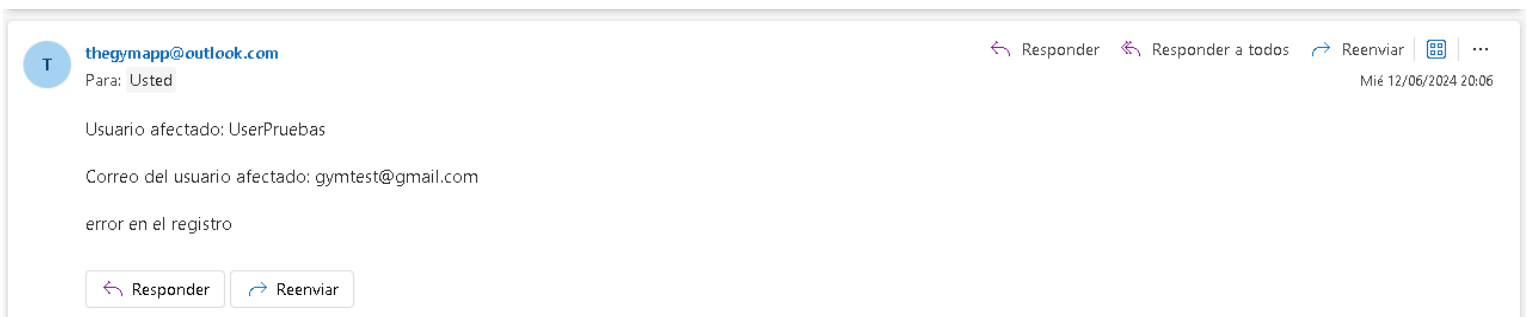
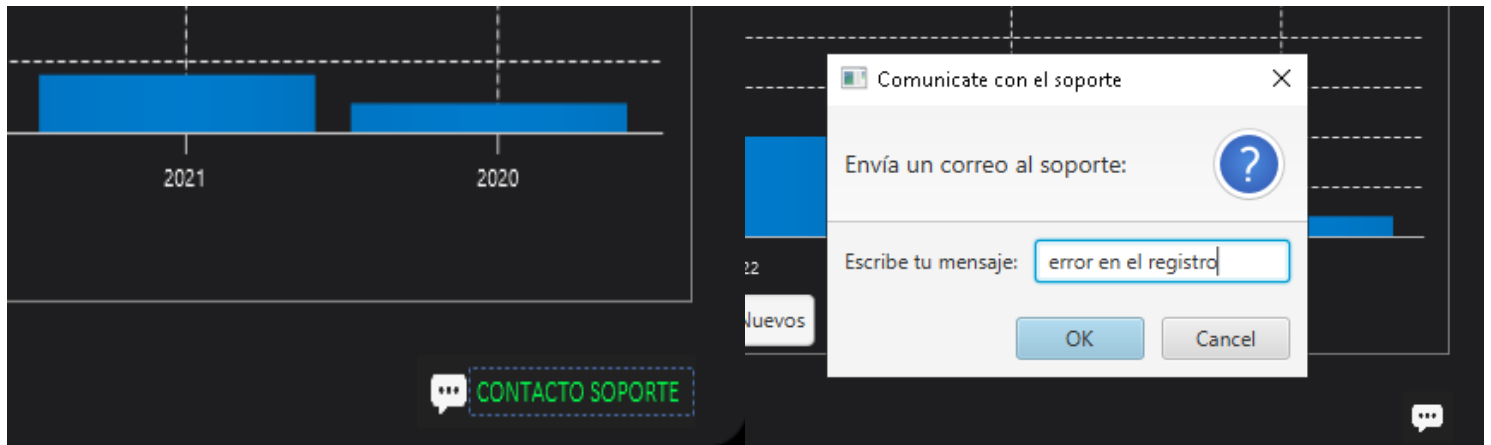
Una vez que se completen todos los datos del nuevo cliente, se le enviará automáticamente un correo de bienvenida. Este correo contendrá una rutina de entrenamiento personalizada, detallada y adaptada según los días de entrenamiento por semana y el género del cliente. Además, la rutina incluirá un video tutorial de cómo ejecutar cada ejercicio, garantizando así una experiencia completa y orientada al éxito desde el primer momento. Este enfoque personalizado no solo facilita la integración del cliente en el gimnasio, sino que también contribuye a su fidelización al proporcionar un servicio que se adapta a sus necesidades específicas desde el inicio.

de su membresía.



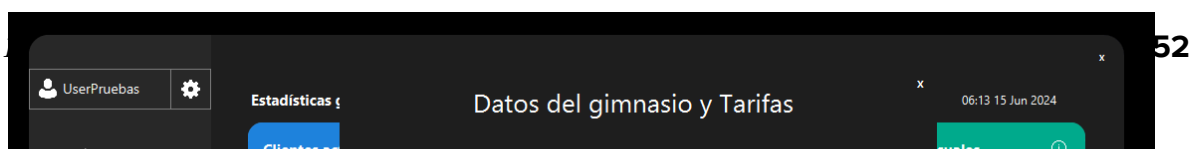
Contacto con el soporte:

En toda la aplicación principal, en la parte inferior, se encuentra un botón que al pasar el cursor por encima se despliega, permitiendo así enviar un mensaje de soporte. Este diseño garantiza que el usuario siempre tenga acceso rápido a la asistencia en caso de necesitar ayuda o tener algún problema con la aplicación.



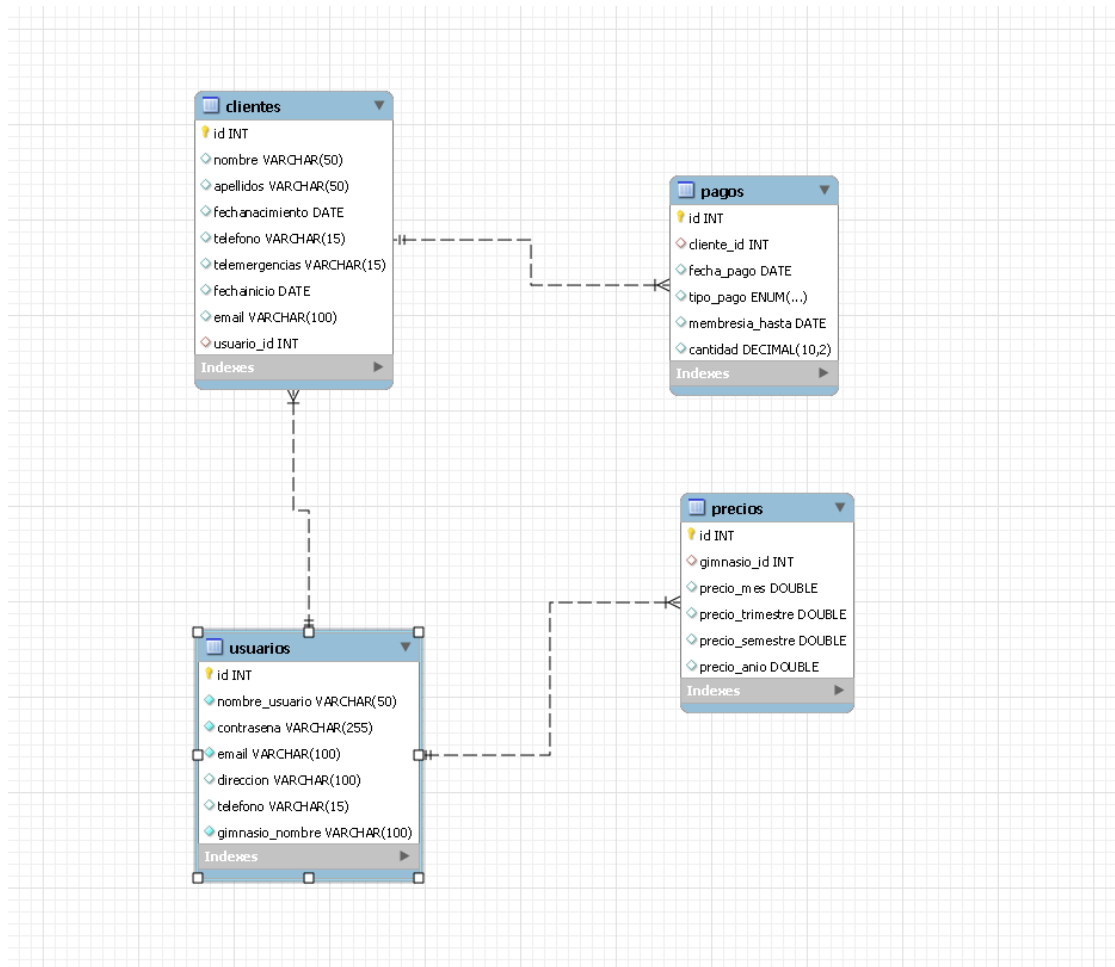
Configuración Gimnasio:

Si hacemos click a la derecha del usuario en el engranaje podemos acceder al menú para cambiar los datos del gimnasio y los precios de la membresía



6.2 Modelo de datos

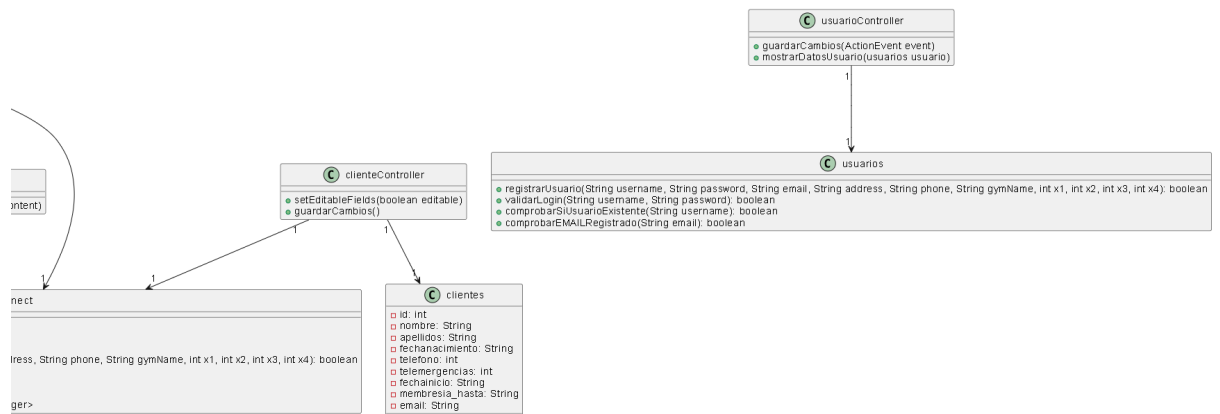
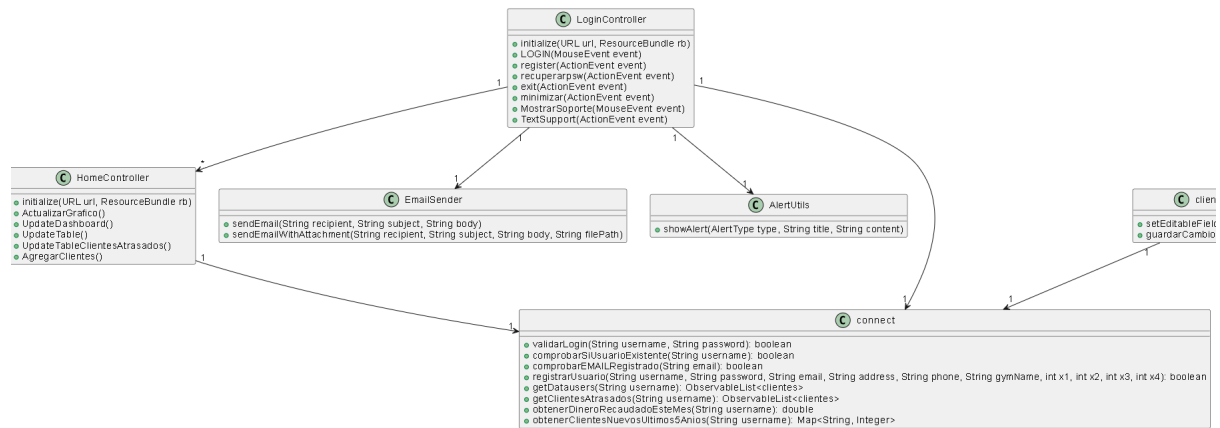
La base de datos para la aplicación GymWizz se implementará utilizando el sistema de gestión de bases de datos relacional MySQL debido a su fiabilidad, escalabilidad y amplia compatibilidad.



Utilidad en el contexto de la aplicación:

- La tabla de **usuarios** almacena los datos de los propietarios o administradores de los gimnasios, permitiendo la gestión de la información relacionada con los gimnasios.
- La tabla **clientes** gestiona la información de los clientes de los gimnasios, incluidos sus datos personales y de membresía.
- La tabla **pagos** registra los pagos realizados por los clientes, lo que facilita el seguimiento de las membresías y la gestión de la facturación.
- La tabla **precios** almacena los precios de las diferentes membresías ofrecidas por los gimnasios, permitiendo una gestión eficiente de las tarifas y promociones.

6.3 Diagrama de Clases



6.4 Despliegue

El despliegue de GymWizz se llevará a cabo en principio para países hispanohablantes. La aplicación se desarrollará utilizando JavaFX, lo que garantiza su compatibilidad con múltiples plataformas como Windows, Linux y Mac OS.

Plataforma de Despliegue:

Para alojar GymWizz, se utilizará Amazon EC2. Este servicio de cómputo en la nube de Amazon Web Services proporciona una infraestructura escalable y confiable para ejecutar aplicaciones Java.

Herramientas de Desarrollo y Despliegue:

- **NetBeans:** NetBeans es la plataforma de desarrollo principal para GymWizz. Proporciona un entorno de desarrollo integrado (IDE) altamente funcional que facilita la construcción y el despliegue de aplicaciones JavaFX.
- **Scene Builder:** Utilizado en conjunto con NetBeans, Scene Builder se emplea para diseñar la interfaz de usuario de GymWizz, lo que facilita la creación de una experiencia visualmente atractiva y funcional.

Costos:

Servidor de Aplicaciones (Amazon EC2): Los costos asociados con el despliegue en Amazon EC2 variarán según la capacidad y el tiempo de ejecución de la instancia. Sin embargo, Amazon ofrece opciones de tarifas flexibles que se adaptan a las necesidades del proyecto.

Mantenimiento: Los costos de mantenimiento estarán principalmente asociados con la gestión del servidor de aplicaciones y las actualizaciones de la aplicación GymWizz.

Escalabilidad:

Una de las ventajas de desplegar en la nube es la capacidad de escalar la aplicación según las necesidades. Si GymWizz experimenta un aumento en la demanda, se pueden agregar más recursos de manera rápida y sencilla en la plataforma.

7. PLANIFICACIÓN

La planificación tiene como objetivo trazar una ruta precisa y detallada para la implementación exitosa de la aplicación. Se ha desarrollado una planificación minuciosa que cubre desde la definición de los objetivos hasta la entrega final del proyecto. Esta planificación aborda los aspectos críticos del despliegue, asegurando una ejecución eficiente y efectiva del proceso.

1. Definir las Actividades y Tareas:

- **1.1 Investigación y Planificación Inicial**
 - 1.1.1 Investigar soluciones de gestión de gimnasios existentes.
 - 1.1.2 Definir requisitos específicos del proyecto GymWizz.
 - 1.1.3 Planificar la estructura general del proyecto.
- **1.2 Diseño de la Interfaz de Usuario (UI)**
 - 1.2.1 Crear mockups y prototipos de la interfaz de usuario.
 - 1.2.2 Recopilar retroalimentación de potenciales usuarios.
 - 1.2.3 Refinar el diseño de la interfaz de usuario basado en la retroalimentación recibida.
- **1.3 Integración de la Base de Datos MySQL**
 - 1.3.1 Diseñar la estructura de la base de datos.
 - 1.3.2 Configurar el servidor MySQL y crear la base de datos.
 - 1.3.3 Implementar la conexión entre la aplicación y la base de datos.
- **1.4 Implementación del Sistema de Registro y Login**
 - 1.4.1 Desarrollar el sistema de registro.
 - 1.4.2 Crear la funcionalidad de inicio de sesión.
- **1.5 Desarrollo del Módulo de Contacto con Soporte**
 - 1.5.1 Diseñar la interfaz de usuario para el módulo de contacto.
 - 1.5.2 Implementar el sistema de contacto con soporte.
- **1.6 Implementación del Dashboard**
 - 1.6.1 Diseñar la estructura del dashboard y los gráficos necesarios.
 - 1.6.2 Desarrollar la funcionalidad para mostrar ganancias mensuales y otros datos relevantes.
- **1.7 Creación del Menú para Agregar Nuevos Clientes**
 - 1.7.1 Diseñar el menú para agregar nuevos clientes.
 - 1.7.2 Desarrollar el menú para agregar nuevos clientes.
- **1.8 Implementación de la Funcionalidad de Registro de Pagos y Eliminación de Clientes**

- 1.8.1 Crear la interfaz para registrar pagos de clientes.
- 1.8.2 Desarrollar la funcionalidad para eliminar clientes deseados y para registrar pagos.
- **1.9 Desarrollo del Módulo de Envío de Correos Electrónicos Personalizados**
 - 1.9.1 Diseñar plantillas de correos electrónicos personalizados como el de las rutinas.
 - 1.9.2 Integrar la funcionalidad de envío de correos electrónicos basados en datos de clientes.
- **1.10 Desarrollo del Botón de Soporte**
 - 1.10.1 Diseñar la ubicación y apariencia del botón de soporte.
 - 1.10.2 Implementar la funcionalidad de soporte.
- **1.11 Funcionalidad de Edición de Clientes**
 - 1.11.1 Diseñar la interfaz para editar la información de los clientes.
 - 1.11.2 Desarrollar la funcionalidad para editar datos de clientes existentes.
- **1.12 Funcionalidad de Edición de Usuarios y Precios**
 - 1.12.1 Crear la interfaz de administración de usuarios y precios.
 - 1.12.2 Desarrollar la funcionalidad para editar usuarios y precios de membresía.
- **1.13 Pruebas y Depuración**
 - 1.13.1 Realizar pruebas de todas las funcionalidades.
 - 1.13.2 Identificar y corregir errores y fallos en el sistema.

2. Secuenciar las Actividades y Asignar Tiempos (Cronograma):

En este caso se utilizó un diagrama de Gantt que será mencionado más abajo.

3. Definir los Recursos y la Logística que Necesitas:

- Recursos Humanos:
 - Desarrolladores de Software
 - Diseñadores de Interfaz de Usuario
 - Especialistas en Base de Datos
 - Personal de Soporte Técnico

- Recursos Materiales:
 - Equipos de Desarrollo (ordenadores, software, etc.)

- Servidores para la Base de Datos
- Logística:
 - Coordinación de reuniones regulares de seguimiento del proyecto.
 - Establecimiento de canales de comunicación eficientes entre el equipo.

4. Determinar los Procedimientos de Cada Actividad:

- Definir un flujo de trabajo claro para cada etapa del desarrollo.
- Establecer protocolos de revisión y aprobación para el diseño y desarrollo de funcionalidades.

5. Identificar Riesgos y Definir un Plan de Prevención:

- Riesgos Potenciales:
 - Retrasos en el desarrollo de funcionalidades clave.
 - Problemas de compatibilidad entre diferentes sistemas.
 - Fallos de seguridad en la gestión de datos de los clientes.
- Plan de Prevención:
 - Asignación de tiempo adicional en el cronograma para posibles retrasos.
 - Realización de pruebas exhaustivas en diferentes entornos para garantizar la compatibilidad.
 - Implementación de protocolos de seguridad robustos y encriptación de datos sensibles.

6. Calcular los Costes:

- Costes Potenciales:
 - Salarios del personal involucrado en el desarrollo.
 - Gastos de infraestructura (servidores, mantenimiento, etc.).
 - Posibles costos adicionales para solucionar problemas imprevistos.

7.1. DIAGRAMA DE GANTT

En el siguiente diagrama de Gantt, se presenta la planificación detallada de las actividades necesarias para llevar a cabo la aplicación GymWizz. Este esquema proporciona una visión clara y estructurada de las tareas que se han dividido desde el inicio hasta la conclusión del proyecto.

Diagrama de Gantt GymWizz



7.2 Definición de Recursos y Logística Necesarios para Cada Actividad:

1. Investigación y Planificación Inicial

- **Recursos Materiales:**
 - Ordenador con acceso a internet para investigación.
 - Herramientas de planificación como software de gestión de proyectos.
- **Recursos Humanos:**
 - Gerente de Proyecto.
 - Desarrollador.
 - Especialista en Marketing.
- **Logística:**
 - Acceso a recursos bibliográficos y en línea para investigación.
 - Reuniones regulares de planificación del proyecto.

2. Diseño de la Interfaz de Usuario (UI)

- **Recursos Materiales:**
 - Netbeans y Scene Builder
 - Software de diseño gráfico como photoshop.
- **Recursos Humanos:**
 - Diseñador.
 - Desarrollador.
- **Logística:**
 - Sesiones de revisión y retroalimentación con el equipo de desarrollo y usuarios potenciales.

3. Integración de la Base de Datos MySQL

- **Recursos Materiales:**
 - Servidor MySQL.
 - Herramientas de gestión de bases de datos.
- **Recursos Humanos:**
 - Administrador de Base de Datos.
 - Desarrolladores.
- **Logística:**
 - Acceso a los servidores y permisos para configurar la base de datos.

4. Implementación del Sistema de Registro y Login

- **Recursos Materiales:**
 - Netbeans.
- **Recursos Humanos:**
 - Desarrolladores.
- **Logística:**
 - Coordinación con el equipo de diseño para la integración de la interfaz de usuario.

5. Desarrollo del Módulo de Contacto con Soporte

- **Recursos Materiales:**
 - Plataforma de correos como Outlook.
 - Netbeans
- **Recursos Humanos:**
 - Desarrolladores.
 - Especialista en Soporte Técnico.
- **Logística:**

- Capacitación del personal de soporte sobre el nuevo sistema de contacto.

6. Implementación del Dashboard

- **Recursos Materiales:**
 - Netbeans y SceneBuilder
- **Recursos Humanos:**
 - Desarrolladores.
- **Logística:**
 - Reuniones de planificación con el equipo para determinar los KPIs a visualizar en el dashboard.

7. Creación del Menú para Agregar Nuevos Clientes

- **Recursos Materiales:**
 - Software de desarrollo de interfaces de usuario.
- **Recursos Humanos:**
 - Desarrolladores.
- **Logística:**
 - Integración con el sistema de registro y base de datos existente.

8. Implementación de la Funcionalidad de Registro de Pagos y Eliminación de Clientes

- **Recursos Materiales:**
 - NetBeans y Scenebuilder
- **Recursos Humanos:**
 - Desarrolladores.
- **Logística:**
 - Asegurar la seguridad y precisión de los datos almacenados.

9. Desarrollo del Módulo de Envío de Correos Electrónicos Personalizados

- **Recursos Materiales:**
 - Plataforma de correos como Outlook.
- **Recursos Humanos:**
 - Desarrolladores.
 - Entrenador para diseñar las rutinas.
 - Especialista en Marketing Digital.
- **Logística:**
 - Creación de plantillas de correo electrónico personalizadas y automatización del proceso de envío.

10. Desarrollo del Botón de Soporte

- **Recursos Materiales:**
 - Plataforma de correos como Outlook.
 - Netbeans
- **Recursos Humanos:**
 - Desarrollador.
 - Especialista en Soporte Técnico.
- **Logística:**
 - Integración del sistema de soporte en la aplicación y capacitación del personal.

11. Funcionalidad de Edición de Clientes

- **Recursos Materiales:**
 - Netbeans
 - Scenebuilder
- **Recursos Humanos:**
 - Desarrollador.
- **Logística:**
 - Implementación de controles de acceso adecuados para evitar ediciones no autorizadas.

12. Funcionalidad de Edición de Usuarios y Precios

- **Recursos Materiales:**
 - Netbeans
 - Scenebuilder
- **Recursos Humanos:**
 - Desarrollador.
- **Logística:**
 - Asegurar que las actualizaciones de precios se reflejen adecuadamente en toda la plataforma.

13. Pruebas y Depuración

- **Recursos Materiales:**
 - Herramientas de pruebas automatizadas.
- **Recursos Humanos:**
 - Equipo de Pruebas QA.
 - Desarrollador.
- **Logística:**
 - Documentar y resolver cualquier problema encontrado durante las pruebas.

8.IMPLEMENTACION

En la implementación de GymWizz se utilizarán las siguientes tecnologías y herramientas: NetBeans como el Entorno de Desarrollo Integrado (IDE) principal, MySQL para la gestión de la base de datos, Scene Builder para el diseño de la interfaz de usuario y Java como el lenguaje de programación principal. Se llevarán a cabo pruebas exhaustivas utilizando las capacidades de depuración de NetBeans para garantizar la calidad y la fiabilidad del proyecto.

ENVÍO DE RUTINAS POR CORREO

Este método se encarga de enviar un correo electrónico con un archivo adjunto, que en el caso de la aplicación son rutinas de gimnasio.

```
//metodo para enviar un correo con un archivo adjunto
public static void sendEmailWithAttachment(String recipientEmail, String subject, String body, String filePath) throws IOException {
    Properties props = new Properties();
    props.put(key:"mail.smtp.auth", value: "true");
    props.put(key:"mail.smtp.starttls.enable", value: "true");
    props.put(key:"mail.smtp.host", value: "smtp.office365.com");
    props.put(key:"mail.smtp.port", value: "587");

    Session session = Session.getInstance(props, new Authenticator() {
        protected PasswordAuthentication getPasswordAuthentication() {
            return new PasswordAuthentication(USERNAME, PASSWORD);
        }
    });

    try {
        Message message = new MimeMessage(session);
        message.setFrom(new InternetAddress(ADDRESS_SENDER_EMAIL)); //remitente
        message.setRecipients(Message.RecipientType.TO, InternetAddress.parse(ADDRESS_LIST_DESTINATARIO)); //destinatario
        message.setSubject(subject);

        // Crear la parte del cuerpo del mensaje
        BodyPart messageBodyPart = new MimeBodyPart();
        messageBodyPart.setText(body); //texto del mensaje

        // Crear un mensaje multipart para adjuntar el archivo
        Multipart multipart = new MimeMultipart();
        multipart.addBodyPart(messageBodyPart);

        // Parte dos= el archivo adjunto
        messageBodyPart = new MimeBodyPart();
        InputStream inputStream = EmailSender.class.getResourceAsStream("/pdfs/" + filePath); // Leer el archivo adjunto como un InputStream
        DataSource source = new ByteArrayDataSource(inputStream, "application/pdf"); // Crear una fuente de datos para el archivo adjunto
        messageBodyPart.setDataHandler(new DataHandler(source)); // establecer el datahandler para el archivo adjunto
        messageBodyPart.setFileName(filePath); // establecer el nombre del archivo adjunto
        multipart.addBodyPart(messageBodyPart); // anadir la parte del archivo adjunto al multipart

        // Enviar todas las partes del mensaje
        message.setContent(multipart);

        Transport.send(message); // Enviar el mensaje

        System.out.println("Email enviado correctamente a " + recipientEmail);
    } catch (MessagingException | IOException e) {
        throw new RuntimeException(message: "Error al enviar el email con archivo adjunto", cause: e);
    }
}
```

Y este método determina qué rutina enviar según lo que seleccione el usuario en los combobox

```
// metodo para determinar el nombre de archivo de la rutina según la selección de los ComboBox
private String obtenerRutinaFileName() {
    String genero = ComboBoxGenero.getValue();
    String diasEntreno = ComboBoxDiasEntreno.getValue();
    // verifica si ambos valores de los combobox no son nulos
    if (genero != null && diasEntreno != null) { // si es femenino
        if (genero.equals(anObject: "Femenino")) {
            if (diasEntreno.equals(anObject: "3")) {
                return "Rutina3DIASMujer.pdf";
            } else if (diasEntreno.equals(anObject: "4")) {
                return "Rutina4DIASMujer.pdf";
            } else if (diasEntreno.equals(anObject: "5")) {
                return "Rutina5DIASMujer.pdf";
            }
        } else { // si es masculino
            if (diasEntreno.equals(anObject: "3")) {
                return "Rutina3DIAS.pdf";
            } else if (diasEntreno.equals(anObject: "4")) {
                return "Rutina4DIAS.pdf";
            } else if (diasEntreno.equals(anObject: "5")) {
                return "Rutina5DIAS.pdf";
            }
        }
    }
    // en caso de no ser ninguno de los casos devuelve una rutina default
    return "RutinaFullBody.pdf";
}
```

GRAFICO DEL DASHBOARD

Este método actualiza un gráfico en el panel de control (dashboard) con datos de la cantidad de clientes que se inscribieron en cada uno de los últimos cinco años y con un tooltip muestra la cantidad exacta al pasar el cursor por encima.

```
// metodo que añade los clientes a el grafico del dashboard
public void ActualizarGrafico() {
    chart.getData().clear(); // limpiar el grafico antes de añadir nuevos datos

    XYChart.Series<String, Double> graficox = new XYChart.Series<>(); // Crear una nueva serie de datos para el grafico
    graficox.setName("Clientes Nuevos");

    Map<String, Integer> clientesNuevosPorAño = connect.obtenerClientesNuevosUltimos5Años(); // Obtener los datos de clientes nuevos por año de la base de datos
    for (Map.Entry<String, Integer> entry : clientesNuevosPorAño.entrySet()) { // Recorrer cada entrada en el mapa de datos
        XYChart.Data<String, Double> data = new XYChart.Data<>(entry.getKey(), (double) entry.getValue()); // Crear un nuevo dato para el grafico con el año y la cantidad

        // Agregar un listener para asegurarnos de que el nodo este creado
        data.nodeProperty().addListener(new ChangeListener<Node>() {
            @Override
            public void changed(ObservableValue<? extends Node> ov, Node oldNode, Node newNode) {
                if (newNode != null) { // verificar si el nuevo nodo esta creado
                    Tooltip tooltip = new Tooltip(entry.getValue() + " clientes"); // Crear un Tooltip con el numero de clientes

                    tooltip.setShowDelay(4000, Duration.millis(4.0)); // sacar delay del tooltip

                    Tooltip.install((Node) newNode, tooltip); // poner el tooltip en el nodo del dato
                }
            }
        });

        graficox.getData().add(data); // agregar el dato a la serie
    }
    chart.getData().add(graficox); // agregar la serie de datos al grafico
}
```

TABLA DE CLIENTES

El método UpdateTable se encarga de actualizar la tabla de clientes en la interfaz con los datos obtenidos de la consulta de la base de datos getDatausers para el usuario que ha iniciado sesión y los asigna a una lista. Finalmente, actualiza los datos en la tabla con la lista de clientes, reflejando así los cambios en la interfaz de usuario.

```
// metodo que pone los datos de la consulta en la tabla
public void UpdateTable() {
    col_id.setCellValueFactory(new PropertyValueFactory<>(string: "id"));
    col_nombre.setCellValueFactory(new PropertyValueFactory<>(string: "nombre"));
    col_apellidos.setCellValueFactory(new PropertyValueFactory<>(string: "apellidos"));
    col_fechanac.setCellValueFactory(new PropertyValueFactory<>(string: "fechanacimiento"));
    col_telefono.setCellValueFactory(new PropertyValueFactory<>(string: "telefono"));
    col_teleemergencias.setCellValueFactory(new PropertyValueFactory<>(string: "teleemergencias"));
    col_fechainicio.setCellValueFactory(new PropertyValueFactory<>(string: "fechainicio"));
    col_vencimientoMembresia.setCellValueFactory(new PropertyValueFactory<>(string: "membresia_hasta"));
    col_email.setCellValueFactory(new PropertyValueFactory<>(string: "email"));

    listaClientes = connect.getDatausers(nombreUsuario: App.NombreUsuarioLogeado); // obtengo los datos de la consu
    table_clientes.setItems(col: listaClientes); //pongo la lista en la tabla
}
```

```
// metodo que obtiene datos de todos los clientes de un determinado usuario
public static ObservableList<clientes> getDatausers(String nombreUsuario) {
    Connection conn = ConnectDb();
    ObservableList<clientes> list = FXCollections.observableArrayList();
    try {
        String query = "SELECT clientes.*, MAX(pagos.membresia_hasta) AS membresia_hasta "
            + "FROM clientes "
            + "JOIN usuarios ON clientes.usuario_id = usuarios.id "
            + "LEFT JOIN pagos ON clientes.id = pagos.cliente_id "
            + "WHERE usuarios.nombre_usuario = ? "
            + "GROUP BY clientes.id";

        PreparedStatement ps = conn.prepareStatement(query);
        ps.setString(parameterIndex: 1, x: nombreUsuario); // Establecer el nombre de usuario como parametro

        ResultSet rs = ps.executeQuery();

        while (rs.next()) {
            String membresiaHastaStr = rs.getString(columnLabel: "membresia_hasta");
            LocalDate membresiaHasta = null;
            if (membresiaHastaStr != null) {
                membresiaHasta = LocalDate.parse(text: membresiaHastaStr);
            }

            list.add(new clientes(
                id: Integer.parseInt(rs.getString(columnLabel: "id")),
                nombre: rs.getString(columnLabel: "nombre"),
                apellidos: rs.getString(columnLabel: "apellidos"),
                fechanacimiento: LocalDate.parse(text: rs.getString(columnLabel: "fechanacimiento")),
                telefono: rs.getString(columnLabel: "telefono"),
                teleemergencias: rs.getString(columnLabel: "teleemergencias"),
                fechainicio: LocalDate.parse(text: rs.getString(columnLabel: "fechainicio")),
                membresia_hasta: membresiaHasta,
                email: rs.getString(columnLabel: "email")
            ));
        }
    } catch (NumberFormatException | SQLException e) {
    } finally {
        try {
            conn.close(); // Cerrar la conexion
        } catch (SQLException e) {
        }
    }
    return list;
}
```


FILTRO DE CLIENTES EN LA TABLA

Este método permite filtrar los clientes en la TableView clientes basado en el texto ingresado en un campo de búsqueda. Obtiene el texto ingresado en el campo de búsqueda y lo convierte a minúsculas y sin acentos para facilitar la comparación. Si hay coincidencia, agrega el cliente a la lista de clientes filtrados. Finalmente, actualiza la tabla con la lista de clientes filtrados, mostrando así solo los clientes que coinciden con el texto de búsqueda.

```
//METODO QUE PERMITE ENCONTRAR UN CLIENTE EN EL TABLEVIEW de clientes
@FXML
void filtrarClientes(KeyEvent event) {
    String searchText = Metodos.removeAccents(input: textfieldFiltro.getText().toLowerCase()); // Obtener el texto d

    ObservableList<clientes> clientesFiltrados = FXCollections.observableArrayList(); // Crear una nueva lista obs

    for (clientes cliente : listaClientes) { // Iterar sobre la lista de clientes original
        if (Metodos.removeAccents(input: cliente.getNombre().toLowerCase()).contains(=: searchText)
            || Metodos.removeAccents(input: cliente.getApellidos().toLowerCase()).contains(=: searchText)) { // V
            clientesFiltrados.add(=: cliente); // Agregar el cliente a la lista filtrada
        }
    }

    table_clientes.setItems(=: clientesFiltrados); // Actualizar la tabla con la lista de clientes filtrados
}
```

ABRIR DATOS DEL CLIENTE

El método maneja el evento de click en la tabla de clientes cuando se hace doble clic con el botón izquierdo del mouse, obtiene el cliente seleccionado en la tabla y abre una ventana de cliente con la información del cliente seleccionado.

```
//maneja el evento de click en al tableview
private void handleTableViewClick(MouseEvent event) {
    if (event.getButton().equals(OTHER: MouseButton.PRIMARY) && event.getClickCount() == 2) { // Verifica si se hizo doble click en el boton izquierdo
        TableView<clientes> table = (TableView<clientes>) event.getSource(); // Obtiene la TableView a partir del evento
        clientes selectedCliente = table.getSelectionModel().getSelectedItem(); // Obtiene el cliente seleccionado en la TableView
        if (selectedCliente != null) { // Verifica si se ha seleccionado un cliente
            System.out.println("Nombre del cliente: " + selectedCliente.getNombre());
            openClienteWindow(cliente:selectedCliente); // Abre la ventana de cliente con la informacion del cliente seleccionado
        }
        index = table.getSelectionModel().getSelectedIndex(); // Actualiza el indice seleccionado para que funcionen los botones de borrar y pagar
    }
}
```

TRANSICIÓN ENTRE LOGIN Y REGISTRO

Estos botones permiten alternar entre la visualización del formulario de registro y el formulario de inicio de sesión en una interfaz de usuario.

- El primer botón desplaza el panel hacia la derecha para mostrar el formulario de registro y ajusta la posición de un panel blanco. Además, muestra el formulario de registro y oculta el formulario de inicio de sesión.
- El segundo desplaza el panel de vuelta a su posición inicial para mostrar el formulario de inicio de sesión. También restaura la posición del panel blanco, muestra el formulario de inicio de sesión y oculta el formulario de registro.

```
//metodo para desplazar el panel para mostrar el registro
@FXML
void btn1(MouseEvent event) {
    TranslateTransition slide = new TranslateTransition(); // Define una transición de desplazamiento
    slide.setDuration(Duration.seconds(0.7)); //duracion de la transición
    slide.setNode(layer2); // Define el nodo que se desplazara ( el panel que se mueve)
    slide.setToX(600); // Mueve el panel hacia la derecha
    slide.play(); //inicia la animacion

    WhitePane.setTranslateX(-300); // mueve el panel blanco hacia la izquierda
    mostrarRegistro(); // Muestra el formulario de registro
    ocultarLogin(); // Oculta el formulario de inicio de sesión
}

//metodo para desplazar el panel para mostrar el login
@FXML
void btn2(MouseEvent event) {
    TranslateTransition slide = new TranslateTransition();
    slide.setDuration(Duration.seconds(0.6));
    slide.setNode(layer2);
    slide.setToX(layer2InitialX); // posicion inicial del panel
    slide.play();

    WhitePane.setTranslateX(whitePaneInitialX); // Restablece la poiscion inicial del panel blanco
    mostrarLogin();
    ocultarRegistro();
}

//metodo para mostrar el panel de inicio de sesión
@FXML
void btn3(MouseEvent event) {
    TranslateTransition slide = new TranslateTransition();
    slide.setDuration(Duration.seconds(0.6));
    slide.setNode(layer2);
    slide.setToX(layer2InitialX);
    slide.play();

    WhitePane.setTranslateX(whitePaneInitialX);
    mostrarLogin();
    ocultarRegistro();
}
```

MÉTODO QUE HASHEA LA CONTRASEÑA

Este método toma una contraseña como entrada y la hashea utilizando el algoritmo de hash SHA-256. Primero, obtiene una instancia de MessageDigest para el algoritmo SHA-256. Luego, obtiene el array de bytes del hash de la contraseña. Finalmente, convierte el array de bytes en una cadena codificada en base64 y la devuelve.

```
// metodo para hashear la contraseña usando SHA-256
private static String hashPassword(String password) throws NoSuchAlgorithmException {
    MessageDigest md = MessageDigest.getInstance("SHA-256"); // Obtener una instancia de MessageDigest para SHA-256
    byte[] hash = md.digest(password.getBytes()); // Obtener el array de bytes del hash de la contraseña
    return Base64.getEncoder().encodeToString(hash); // Convertir el array de bytes a una cadena codificada en base64 y devolverla
}
```

BOTON LOGIN

Este método obtiene el nombre de usuario y la contraseña ingresados por el usuario. Luego, valida el inicio de sesión llamando al método validarLogin de la clase connect. Si el inicio de sesión es exitoso, muestra un mensaje de inicio de sesión exitoso en la consola, almacena el nombre de usuario en una variable y redirige al usuario a la página de inicio.

```
// metodo del boton de login que valida el inicio de sesion y abre la app
@FXML
void LOGIN(MouseEvent event) throws IOException {
    String nombreUsuario = LOG_NomUsuario.getText();
    String contrasena = LOG_PasswordField.getText();

    // Validar el inicio de ses
    if (connect.validarLogin(nombreUsuario, contrasena)) {
        System.out.println("Inicio de sesion exitoso para el usuario: " + nombreUsuario);
        App.NombreUsuarioLogeado = nombreUsuario; // almaceno el nombre del usuario en la variable
    } else {
        App.SetRoot(fxml: "home", width: 1100, height: 580);
        AlertUtils.showAlert(alertType: Alert.AlertType.INFORMATION, title: "Error", message: "Nombre de usuario o contraseña incorrectos.");
    }
}
```

MÉTODO PARA VALIDAR EL LOGIN DEL USUARIO

Abre una conexión a la base de datos y ejecuta una consulta para obtener la contraseña asociada al nombre de usuario proporcionado. Luego, compara la contraseña almacenada en la base de datos con la contraseña proporcionada después de haberla hasheado. Si las contraseñas coinciden, devuelve true sino false

```
// metodo que valida el login de usuario y devuelve true si es correcto
public static boolean validarLogin(String nombreUsuario, String contrasena) {
    Connection conn = ConnectDb();
    String query = "SELECT contrasena FROM usuarios WHERE nombre_usuario = ?";

    try (PreparedStatement ps = conn.prepareStatement(sql: query)) {
        ps.setString(parameterIndex: 1, x: nombreUsuario);
        ResultSet rs = ps.executeQuery();

        if (rs.next()) {
            String hashedPasswordFromDB = rs.getString(columnLabel: "contrasena"); // Obtener la contraseña hasheada
            String hashedPassword = hashPassword(password: contrasena); // Hashear la contraseña
            // Comparar las contraseñas hasheadas
            return hashedPasswordFromDB.equals(anObject: hashedPassword); // Comparar las contraseñas hasheadas
        } else {
            return false; // Si no se encuentra un usuario con el nombre de usuario proporcionado
        }
    } catch (SQLException | NoSuchAlgorithmException e) {
        e.printStackTrace();
        return false;
    } finally {
        try {
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

BOTÓN DEL SOPORTE EN EL LOGIN Y REGISTRO

Este método permite enviar un correo electrónico al soporte técnico desde la interfaz de usuario. Abre un cuadro de diálogo para que el usuario ingrese su correo electrónico y luego otro cuadro de diálogo para que escriba su mensaje.

```
// metodo para enviar un correo de asistencia
@FXML
void TextSupport(ActionEvent event) {
    TextInputDialog emailDialog = new TextInputDialog(); // Crear un cuadro de dialogo para ingresar el correo electrónico
    emailDialog.setTitle(string: "Comunicate con el soporte");
    emailDialog.setHeaderText(string: "Envía un correo al soporte:");
    emailDialog.setContentText(string: "Ingresa tu correo electrónico:");

    Optional<String> emailResult = emailDialog.showAndWait(); // Mostrar el cuadro de dialogo para introducir el email y obtener el resultado
    emailResult.ifPresent(email -> {
        TextInputDialog messageDialog = new TextInputDialog();
        messageDialog.setTitle(string: "Comunicate con el soporte");
        messageDialog.setHeaderText(string: "Envía un correo al soporte:");
        messageDialog.setContentText(string: "Escribe tu mensaje:");

        Optional<String> messageResult = messageDialog.showAndWait(); // Mostrar el cuadro de dialogo y obtener el resultado del mensaje
        messageResult.ifPresent(message -> {
            String recipientEmail = "thegymapp@outlook.com";
            String subject = "Correo de soporte desde el login";
            String finalMessage = "Ticket enviado por: " + email + "\n\n" + message;
            EmailSender.sendEmail(recipientEmail, subject, body: finalMessage); //ENVIAR EL CORREO CON TICKET DE SOPORTE
            AlertUtils.showAlert(alertType: AlertType.CONFIRMATION, title: "Correo enviado", message: "Tu mensaje ha sido enviado al soporte.");
        });
    });
}
```

GENERAR CODIGO DE RECUPERACION

Este método utiliza una instancia de SecureRandom para generar bytes aleatorios. Luego, convierte estos bytes en una cadena codificada en base64 sin relleno y la devuelve. Los códigos generados tienen una longitud de 6 caracteres.

```
// metodo para generar codigos de recuperacion aleatorios
public static String generarCodigoRecuperacion() {
    SecureRandom random = new SecureRandom(); // Generar una instancia de SecureRandom
    byte[] bytes = new byte[6]; // Crear un array de bytes de longitud 6
    random.nextBytes(bytes); // Generar bytes aleatorios
    return Base64.getUrlEncoder().withoutPadding().encodeToString(bytes); // Codificar los bytes en base64 sin relleno
}
```

RECUPERAR CONTRASEÑA

El método obtiene el correo electrónico ingresado por el usuario. Luego, verifica si el correo electrónico está registrado en la base de datos llamando al método comprobarEMAILRegistrado de la clase connect. Si el correo electrónico está registrado, oculta el primer panel de recuperación y muestra el segundo panel de recuperación. Luego, genera un código de recuperación utilizando el método generarCodigoRecuperacion.

Posteriormente, envía un correo electrónico al correo del usuario con el código de recuperación generado, que el usuario tendrá que introducir en el siguiente panel.

```
@FXML
void Siguiente(ActionEvent event) { //metodo para ir de la primera opcion de recuperar a la segunda

    emailUsuario = JTextFieldCorreo.getText(); // obtengo el email del usuario
    if (connect.comprobarEMAILRegistrado(email: emailUsuario)) {
        PaneRecuperar1.setVisible(bln:false);
        PaneRecuperar2.setVisible(bln:true);

        String recipientEmail = emailUsuario;
        String subject = "Código de recuperación de tu cuenta de GymWizz";
        codigoRecuperacion = Metodos.generarCodigoRecuperacion(); // Generamos el codigo de recuperacion
        String body = "Recibimos una solicitud para restablecer tu contraseña.\n"
            + "Ingresa el siguiente código para restablecer la contraseña: " + codigoRecuperacion;

        EmailSender.sendEmail(recipientEmail, subject, body);
        AlertUtils.showAlert(alertType: Alert.AlertType.INFORMATION, title: "Codigo de recuperacion enviado", "Hemos enviado un codigo de recuperacion a tu correo electronico");
        System.out.println("Codigo correcto= " + codigoRecuperacion);
    } else {

        AlertUtils.showAlert(alertType: Alert.AlertType.ERROR, title: "Correo desconocido", "El correo electronico " + emailUsuario + " no existe en la base de datos");
        System.out.println("No existe ese correo en la bd");
    }
}
```

Este método verifica si el código introducido por el usuario coincide con el código de recuperación generado. Si es así, oculta el segundo panel de recuperación y muestra el tercer panel.

```
@FXML
void Siguiente2(ActionEvent event) { //metodo para ir de la segunda opcion de recuperar a la tercera y ultima

    if (JTextFieldCodigoSeg.getText().equals(anObject.codigoRecuperacion)) { //VERIFICO SI EL USUARIO INTRODUJO EL CODIGO CORRECTO

        PaneRecuperar2.setVisible(bln:false);
        PaneRecuperar3.setVisible(bln:true);

    } else {
        AlertUtils.showAlert(alertType: Alert.AlertType.ERROR, title: "Codigo equivado", message: "El codigo que has introducido no es correcto, intenta de nuevo");
    }
}
```

Este método verifica si las contraseñas introducidas por el usuario coinciden. Si coinciden y la nueva contraseña cumple con los requisitos de seguridad, se cambia la contraseña en la base de datos.

```
@FXML
void Siguientes(ActionEvent event) throws IOException {

    if (JTextFieldContrasena.getText().equals(anObject.JTextFieldConfirmarContra.getText())) {
        String contrasena = JTextFieldContrasena.getText();

        if (connect.validarContrasena(contrasena)) {
            // Si la contraseña es valida cambiarla

            connect.cambiarContrasenaPorEmail(email: emailUsuario, nuevaContrasena: contrasena);
            AlertUtils.showAlert(alertType: Alert.AlertType.INFORMATION, title: "Perfecto", message: "La contraseña ha sido cambiada con éxito.");
            App.setRoot(fxml: "login", width: 900, height: 580);
        } else {
            // Si la contraseña no es valida muestra un mensaje de error
            AlertUtils.showAlert(alertType: Alert.AlertType.ERROR, title: "Error", message: "La contraseña no cumple con los requisitos de seguridad.");
        }
    } else {
        // Si las contraseñas no coinciden muestra un mensaje de error
        AlertUtils.showAlert(alertType: Alert.AlertType.ERROR, title: "Error", message: "Las contraseñas no coinciden.");
    }
}
```

REGISTRAR UN PAGO DE UN CLIENTE

El método primero obtiene el cliente seleccionado en la tabla. Luego, muestra un cuadro de diálogo para que el usuario seleccione el tipo de pago: mensualidad, trimestre, semestre o año. Después de que el usuario selecciona el tipo de pago, calcula el monto del pago y la fecha de vencimiento de la membresía. Antes de insertar un nuevo pago, elimina los pagos anteriores para el cliente seleccionado para evitar la repetición de pagos en la tabla de deudores.

```
// BOTON PARA QUE AL CLICKEAR EL TABLEVIEW EL CLIENTE PUEDA REGISTRAR UN PAGO DEL CLIENTE
public void PagarButton() {
    clientes selectedCliente = table_clientes.getSelectionModel().getSelectedItem(); // Obtiene el cliente seleccionado
    if (selectedCliente != null) { // Verifica si se ha seleccionado un cliente en la tabla
        String selectedClienteNombre = selectedCliente.getNombre(); // Obtiene el nombre del cliente seleccionado

        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setTitle(string: "Seleccionar Tipo de Pago");
        alert.setHeaderText("Selecciona el tipo de pago para el cliente: " + selectedClienteNombre);
        alert.setContentText(string: "Elige una opción:");

        ButtonType mensualidadButton = new ButtonType(string: "Mensualidad");
        ButtonType trimestreButton = new ButtonType(string: "Trimestre");
        ButtonType semestreButton = new ButtonType(string: "Semestre");
        ButtonType anoButton = new ButtonType(string: "Año");

        alert.getButtonTypes().setAll(es: mensualidadButton, es: trimestreButton, es: semestreButton, es: anoButton);

        Optional<ButtonType> result = alert.showAndWait();
        if (result.isPresent()) {
            String selectedOption = result.get().getText();
            LocalDate fechaPago = LocalDate.now(); // Obtiene la fecha actual para el pago

            // Calcula el monto del pago dependiendo del tipo de pago seleccionado
            double montoPago = connect.calcularMontoPago(tipoPago:selectedOption, nombreUsuario: App.NombreUsuarioLogeado);

            // Calcula la fecha de vencimiento de la membresia
            LocalDate membresiaHasta = Metodos.calcularFechaVencimiento(tipoPago:selectedOption, fechaPago);

            // Antes de insertar un nuevo pago, elimina los pagos anteriores para el cliente seleccionado DE ESTE MODO NO SE REPETIRAN EN LA T
            String eliminarPagosAntiguosQuery = "DELETE FROM pagos WHERE cliente_id = ?";
            try {
                Connection conn = connect.ConnectDb();
                PreparedStatement eliminarPagosAntiguosStmt = conn.prepareStatement(sql: eliminarPagosAntiguosQuery);
                eliminarPagosAntiguosStmt.setInt(parameterIndex: 1, x: selectedCliente.getId());
                eliminarPagosAntiguosStmt.executeUpdate();
                conn.close();
            } catch (SQLException e) {
                AlertUtils.showAlert(alertType: Alert.AlertType.ERROR, title: "Error", "Error al eliminar pagos antiguos: " + e.getMessage());
                return; // Si hay un error al eliminar pagos antiguos, salir del método sin insertar el nuevo pago
            }
        }
    }
}
```

Después de eliminar los pagos antiguos, inserta el nuevo pago en la base de datos junto con la fecha de vencimiento. Finalmente, muestra una ventana emergente

confirmando que el pago ha sido registrado correctamente y actualiza la tabla de clientes atrasados, la tabla principal y el dashboard.

```
// Inserta el pago en la base de datos junto con la fecha de vencimiento
try {
    Connection conn = connect.ConnectDb();
    String sql = "INSERT INTO pagos (cliente_id, fecha_pago, tipo_pago, membresia_hasta, cantidad) VALUES (?, ?, ?, ?, ?)";
    PreparedStatement pst = conn.prepareStatement(sql);
    pst.setInt(parameterIndex: 1, x: selectedCliente.getId());
    pst.setDate(parameterIndex: 2, x: java.sql.Date.valueOf(date: fechaPago));
    pst.setString(parameterIndex: 3, x: selectedOption);
    pst.setDate(parameterIndex: 4, x: java.sql.Date.valueOf(date: membresiaHasta)); // Establece la fecha de vencimiento
    pst.setDouble(parameterIndex: 5, x: montoPago); // Establece el monto del pago
    pst.executeUpdate();
    conn.close();

    String message = "Se ha registrado el pago de tipo " + selectedOption + " para el cliente " + selectedClienteNombre;
    AlertUtils.showAlert(alertType: Alert.AlertType.INFORMATION, title: "Pago Confirmado", message);
    UpdateTableClientesAtrasados();
    UpdateTable();
    UpdateDashboard();
} catch (SQLException e) {
    e.printStackTrace();
    AlertUtils.showAlert(alertType: Alert.AlertType.ERROR, title: "Error", message: "Error al registrar el pago en la base de datos.");
}
}
```

BORRAR UN CLIENTE

El método primero obtiene el cliente seleccionado en la tabla. Luego, muestra un cuadro de diálogo de confirmación para que el usuario confirme si desea eliminar al cliente seleccionado. Si el usuario confirma la eliminación, se conecta a la base de datos y ejecuta una consulta para eliminar al cliente.

```
// metodo para borrar un cliente de la tableview
public void DeleteFromTable() {

    clientes selectedCliente = table_clientes.getSelectionModel().getSelectedItem(); // Obtiene el cliente seleccionado
    if (selectedCliente != null) { // Verifica si se ha seleccionado un cliente en la tabla
        String selectedClienteNombre = selectedCliente.getNombre(); // Obtiene el nombre del cliente seleccionado

        // mostrar dialogo de confirmacion
        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setTitle(string: "Confirmar Eliminación");
        alert.setHeaderText("¿Estás seguro de que deseas eliminar al cliente " + selectedClienteNombre + "?");
        alert.setContentText(string: "Esta acción no se puede deshacer.");

        Optional<ButtonType> result = alert.showAndWait();
        if (result.isPresent() && result.get() == ButtonType.OK) {
            // Si el usuario da a OK=
            conn = connect.ConnectDb();
            String sql = "DELETE FROM clientes WHERE id = ?";
            try {
                pst = conn.prepareStatement(sql);
                pst.setInt(parameterIndex: 1, x: selectedCliente.getId());
                pst.executeUpdate();

                AlertUtils.showAlert(alertType: Alert.AlertType.INFORMATION, title: "Éxito", message: "Cliente eliminado exitosamente");

                UpdateTable(); // Actualiza la tabla para reflejar los cambios
                UpdateTableClientesAtrasados(); // actualizo los clientes atrasados tmb
                UpdateDashboard();
            } catch (Exception e) {
                AlertUtils.showAlert(alertType: Alert.AlertType.ERROR, title: "Error", message: "Error al eliminar el cliente: " + e.getMessage());
            }
        } else {
            AlertUtils.showAlert(alertType: Alert.AlertType.WARNING, title: "Advertencia", message: "Por favor, selecciona un cliente para borrar");
        }
    }
}
```

9.PUESTA EN MARCHA, EXPLOTACIÓN

Antes de pasar el proyecto a producción, es crucial realizar pruebas exhaustivas de seguridad, configuración y cumplimiento legal. A continuación, se detallan los aspectos evaluados y las acciones tomadas:

Cambios de Configuración, Seguridad y Legalidad antes de la Puesta en Producción

1. Test de Seguridad:

- Se realizó un exhaustivo test de seguridad para identificar posibles vulnerabilidades en la aplicación GymWizz.
- Se utilizaron herramientas de escaneo de vulnerabilidades y se realizaron pruebas de penetración para evaluar la resistencia del sistema a posibles ataques.
- Se identificaron y corrigieron las siguientes vulnerabilidades:
 - Cifrado de contraseñas.
 - Fallos de seguridad en la gestión de sesiones.
 - Problemas de autenticación débil.
- Se implementaron medidas de seguridad adicionales, incluyendo:
 - Implementación de encriptación de datos sensibles y comunicaciones.
 - Reforzamiento de políticas de contraseñas y autenticación de usuarios.

2. Configuración de Seguridad:

- Se revisó y actualizó la configuración de seguridad de la infraestructura de GymWizz para garantizar la protección adecuada de los datos de los clientes y la integridad del sistema.
- Se estableció un plan de respuesta a incidentes para abordar de manera eficiente cualquier problema de seguridad que pudiera surgir en el futuro.

3. Aspectos Legales:

- Se llevó a cabo una revisión exhaustiva de la legalidad del proyecto GymWizz en relación con las regulaciones de protección de datos, privacidad y cualquier otra normativa relevante.
- Se aseguró el cumplimiento de las leyes de protección de datos, como el Reglamento General de Protección de Datos (GDPR) en la Unión Europea y leyes locales de privacidad en otras jurisdicciones.

- Se actualizó la política de privacidad y los términos de uso de GymWizz para reflejar los cambios realizados y para garantizar la transparencia y el cumplimiento normativo.

Pasos Necesarios para Llevar el Producto a Producción

Una vez completadas las pruebas de seguridad y configuración, se llevarán a cabo los siguientes pasos para llevar el producto GymWizz a producción de manera efectiva:

1. Migración de la Base de Datos a Amazon Web Services (AWS):

- Migrar la base de datos de la aplicación GymWizz desde el host gratuito actual a un entorno más escalable y robusto, utilizando Amazon Web Services (AWS) EC2.
- Realizar pruebas exhaustivas para verificar el correcto funcionamiento de la base de datos en el entorno de AWS EC2 y ajustar las configuraciones según fuera necesario para optimizar el rendimiento y la fiabilidad.

2. Monitoreo y Mantenimiento:

- Se configuraron herramientas de monitoreo y registro para supervisar el rendimiento y la disponibilidad de GymWizz en tiempo real.
- Se programaron tareas de mantenimiento regulares para aplicar actualizaciones de seguridad, parches de software y mejoras de rendimiento según sea necesario.

3. Prueba de Producción:

- Se llevaron a cabo pruebas exhaustivas en el entorno de producción para validar el correcto funcionamiento de GymWizz bajo condiciones de uso real.
- Se realizaron pruebas de carga para evaluar la capacidad del sistema para manejar la demanda esperada de usuarios.
- Se verificó la integridad de los datos y la precisión de las funcionalidades críticas antes de abrir el sistema al público.

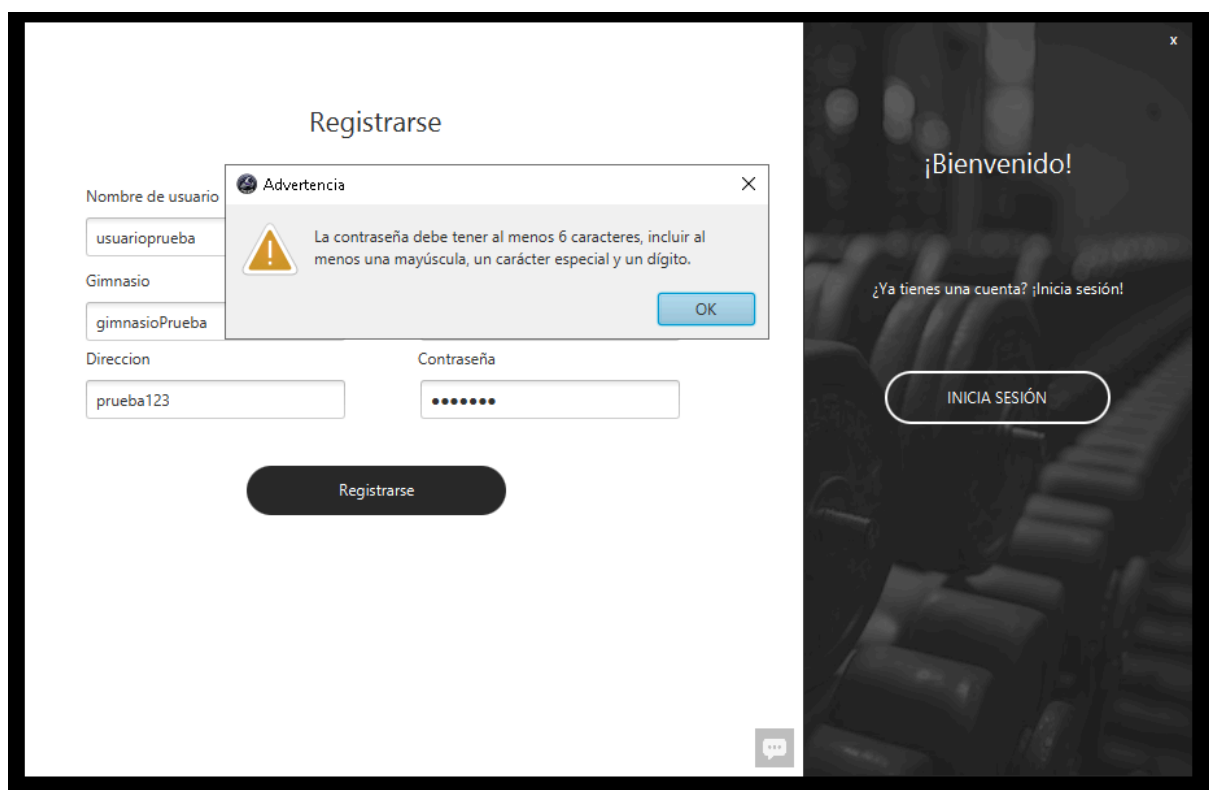
10.Prueba y control de calidad

Para garantizar que el producto final funcione correctamente y cumpla con las expectativas, se definirá un plan de pruebas exhaustivo. A continuación se describen las pruebas a realizar:

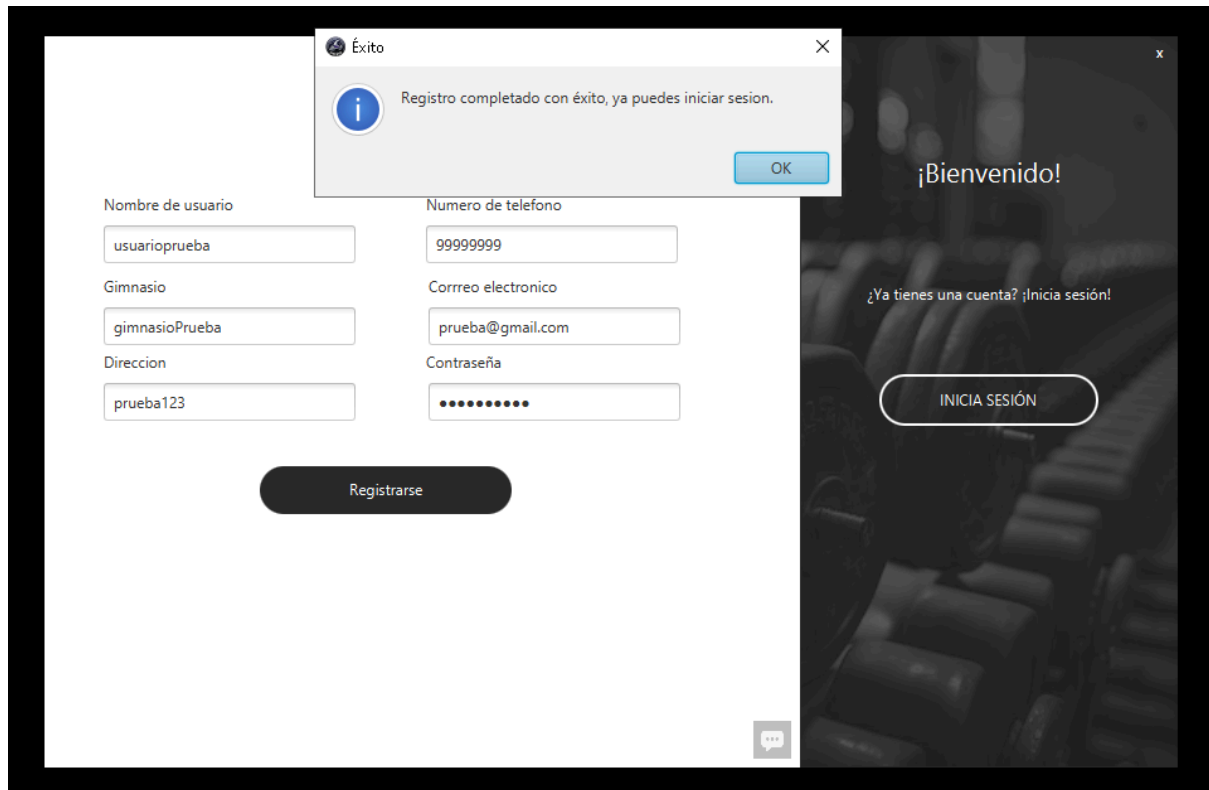
10.1 Prueba de Registro e Inicio de Sesión

Descripción: Verificar que los administradores de gimnasios puedan registrarse y acceder al sistema correctamente y que se verifique la seguridad de los datos .

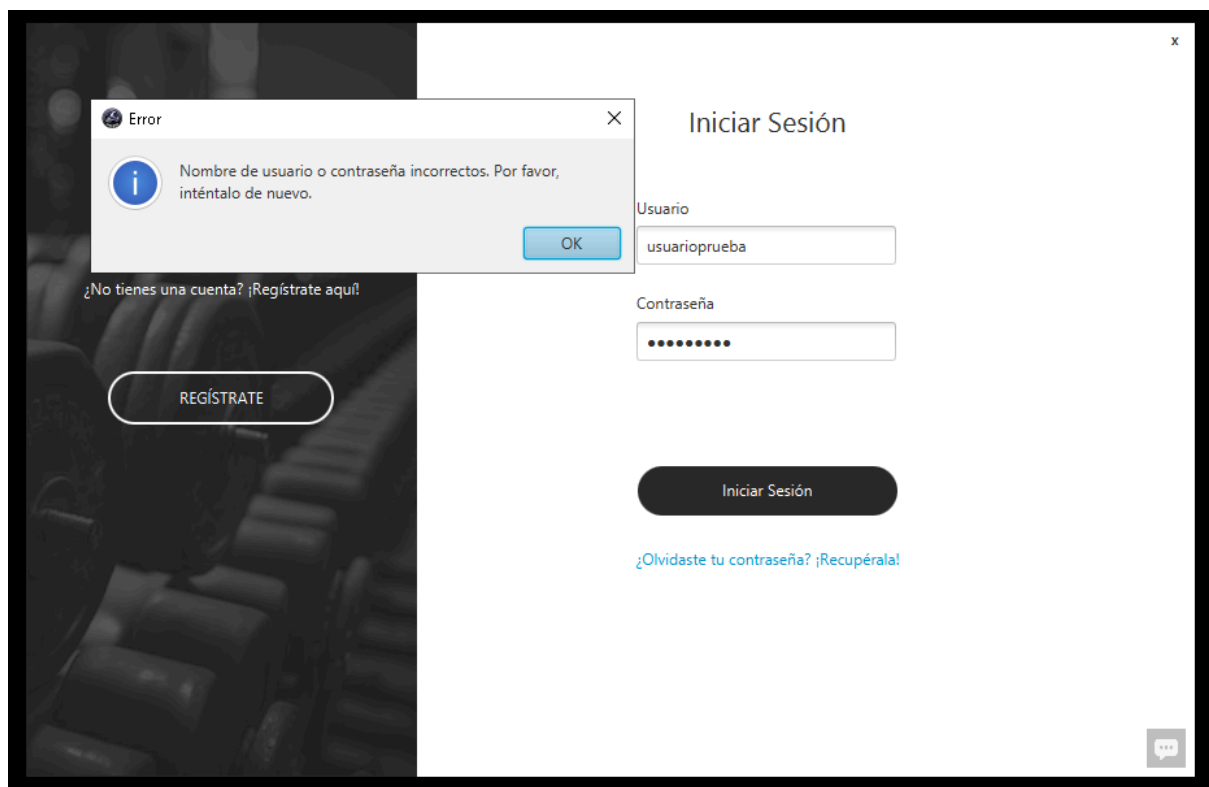
Si La contraseña no cumple con los estándares de seguridad no deja registrarse



Al utilizar una contraseña válida permite registrarse



Si la contraseña o el usuario están mal da error

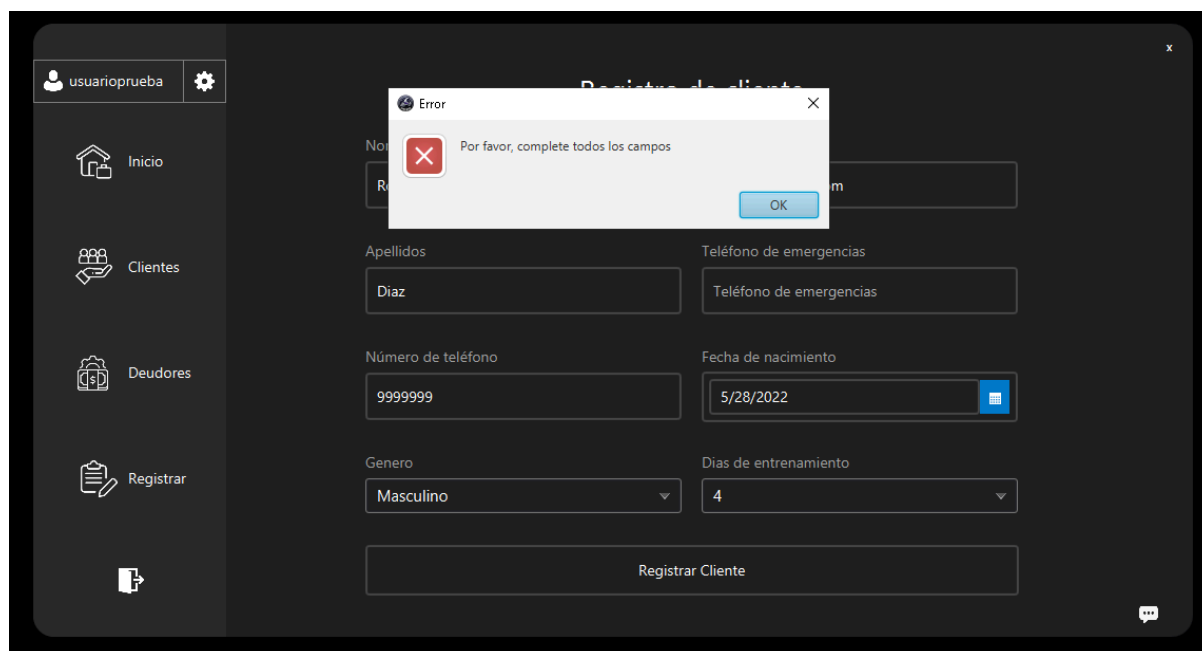


Al poner los datos correctamente deja loguearse



10.2 Prueba de Registro de cliente y edición de datos

Si hay algún dato vacío no deja registrar al cliente



Al registrarlo lo añade a la base de datos, le envía el correo de bienvenida y ya es visible en clientes

usuarioprueba

Inicio

Cientes

Deudores

Registrar

Éxito

Cliente agregado exitosamente

OK

Apellidos: Diaz

Teléfono de emergencias: 888888888

Número de teléfono: 9999999

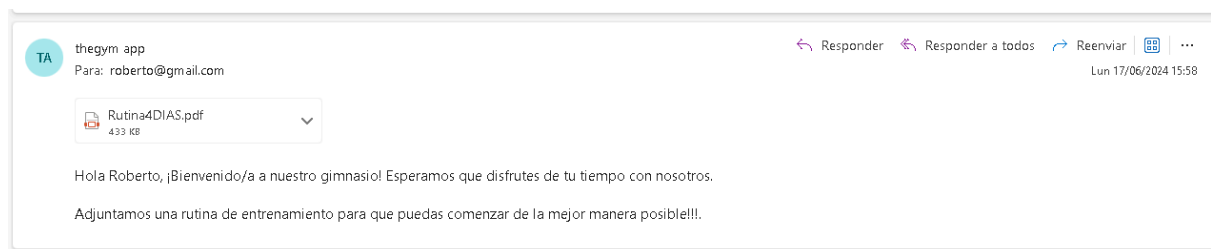
Fecha de nacimiento: 5/28/2022

Género: Masculino

Días de entrenamiento: 4

Registrar Cliente

Y ya se puede ver que el correo fue enviado al cliente



Al ir a la pestaña clientes ya se puede registrar un pago

usuarioprueba

Buscar un cliente

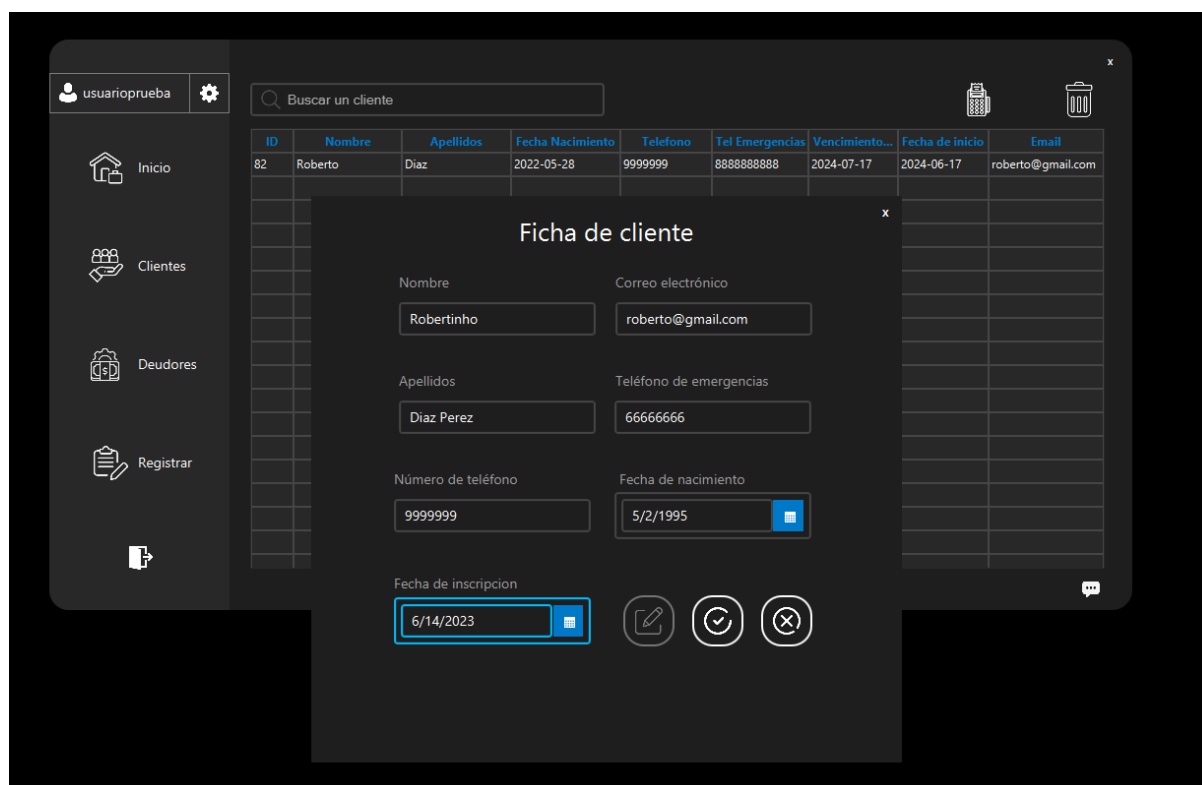
ID	Nombre	Apellidos	Fecha Nacimiento	Telefono	Tel Emergencias	Vencimiento...	Fecha de inicio	Email
82	Roberto	Diaz	2022-05-28	9999999	8888888888		2024-06-17	roberto@gmail.com

Pago Confirmado

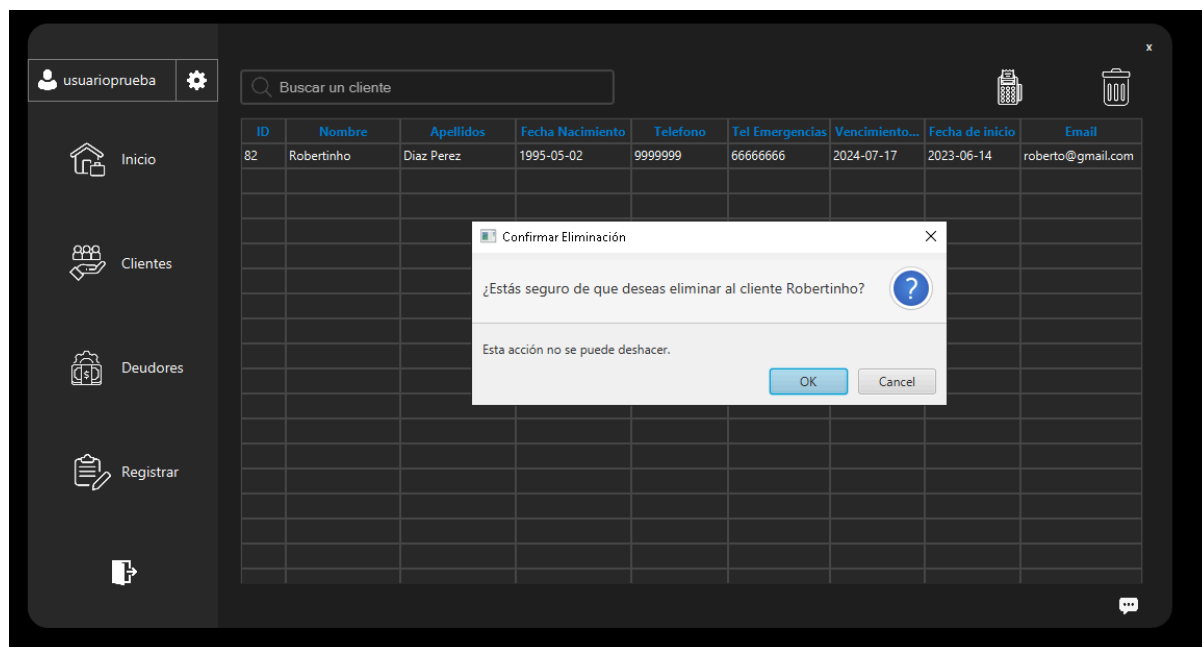
Se ha registrado el pago de tipo Mensualidad para el cliente Roberto

OK

Al hacer doble click sobre el cliente se pueden editar sus datos



Y al clicar el cliente y darle a eliminar se elimina sin problemas



11. Gestión económica o plan de empresa

El Sector Productivo

El proyecto GymWizz se encuadra en el sector productivo de las nuevas tecnologías y software en España, específicamente en el subsector del software de gestión de clientes para gimnasios y centros deportivos. Este sector se caracteriza por su dinamismo y constante evolución, impulsado por la digitalización y la necesidad de soluciones tecnológicas avanzadas para optimizar operaciones comerciales y administrativas.

Número de Empresas y Tamaño:

- **Empresas en el Sector:** Según el INE, en 2022 había aproximadamente 35,000 empresas dedicadas al desarrollo de software y servicios informáticos en España.
- **Tamaño Promedio:** La mayoría de estas empresas son PYMEs con un promedio de 10 a 50 empleados. Sin embargo, existen grandes corporaciones con plantillas superiores a los 200 trabajadores.
- **Cifra de Negocio:** El sector de software en España generó un volumen de negocio de aproximadamente 20,000 millones de euros en 2022, con una tendencia de crecimiento anual del 10%.
- **Forma Jurídica:** Predominan las sociedades limitadas (S.L.) y anónimas (S.A.), debido a la flexibilidad y ventajas fiscales que ofrecen.
- **Dimensión Geográfica:** Aunque se encuentran distribuidas por toda España, las regiones con mayor concentración son Madrid, Cataluña y la Comunidad Valenciana.
- **Estructura Organizativa:** Las empresas del sector suelen tener una estructura organizativa jerárquica, con departamentos de desarrollo, marketing, ventas, soporte técnico y administración.

Aspectos Culturales, Sociales y Medioambientales:

- **Culturales:** Existe una cultura de innovación y colaboración, con una alta adopción de metodologías ágiles (Scrum, Kanban).
- **Sociales:** La creciente demanda de soluciones tecnológicas ha generado un aumento en la formación especializada y la creación de empleo en el sector.
- **Medioambientales:** Las empresas del sector están adoptando prácticas sostenibles, como la reducción del consumo de papel mediante soluciones digitales y la implementación de políticas de trabajo remoto para disminuir la huella de carbono.

La Empresa

GymWizz se constituirá como una Sociedad de Responsabilidad Limitada (S.L.), una forma jurídica que ofrece flexibilidad y protección a los socios frente a posibles responsabilidades económicas.

Justificación de la Forma Jurídica:

- **Ventajas:**
 - **Responsabilidad Limitada:** Los socios no responden con su patrimonio personal, sólo hasta el capital aportado.
 - **Flexibilidad en la gestión:** Menos requisitos formales y administrativos en comparación con una sociedad anónima.
 - **Fiscalidad Ventajosa:** Tipo impositivo reducido para PYMEs en el Impuesto de Sociedades.
- **Inconvenientes:**
 - **Capital Mínimo:** Requiere un capital mínimo de 3,000 euros para su constitución.
 - **Limitación en la Transmisión de Participaciones:** Restricciones para la venta de participaciones sociales.

Trámites de Constitución y Puesta en Marcha:

1. **Reserva de Denominación Social:**
 - Organismo: Registro Mercantil Central.
 - Plazo: 1-2 días.
2. **Obtención del Certificado Digital:**
 - Organismo: Fábrica Nacional de Moneda y Timbre.
 - Plazo: 1-2 días.
3. **Apertura de Cuenta Bancaria y Depósito de Capital:**
 - Banco BBVA.
 - Plazo: 1 día.
4. **Escritura Pública de Constitución:**
 - Notario.
 - Plazo: 1 día.
5. **Inscripción en el Registro Mercantil:**
 - Registro Mercantil Cantabria.
 - Plazo: 5-10 días.
6. **Obtención del CIF Provisional:**
 - Agencia Estatal Tributaria.
 - Plazo: 1-2 días.

7. Alta en el Impuesto de Actividades Económicas (IAE):

- Agencia Estatal Tributaria.
- Plazo: 1-2 días.

8. Inscripción en la Seguridad Social:

- Tesorería General de la Seguridad Social.
- Plazo: 1-2 días.

9. Comunicación de Apertura del Centro de Trabajo:

- Inspección Provincial de Trabajo.
- Plazo: 1 día.

10. Registro de la Marca:

- Oficina Española de Patentes y Marcas.
- Plazo: 4-6 meses.

Organigrama de la Empresa:

- **CEO / Director General (Franco Ruben Milazzo)**
- **Desarrollador de Software (Franco Ruben Milazzo)**
- **Soporte Técnico y Atención al Cliente (Franco Ruben Milazzo)**

Convenio Colectivo Aplicable:

- Convenio Colectivo del Sector de la Industria de Tecnologías de la Información y de la Comunicación (TIC).

Plan de Prevención de Riesgos Laborales: El plan de prevención de riesgos laborales para GymWizz abordará varios aspectos clave para garantizar un entorno de trabajo seguro y saludable:

1. Riesgos Eléctricos:

- **Descripción:** Uso de equipos informáticos y electrónicos.
- **Nivel de Riesgo:** Moderado.
- **Medidas de Prevención:** Mantenimiento regular de equipos, uso de regletas con protección contra sobretensiones, y formación en buenas prácticas.

2. Riesgos Ergonómicos:

- **Descripción:** Trabajo prolongado frente al ordenador.
- **Nivel de Riesgo:** Moderado.
- **Medidas de Prevención:** Evaluación ergonómica del puesto de trabajo, sillas ajustables, monitores a la altura adecuada, y pausas regulares para evitar la fatiga.

3. Riesgos Psicosociales:

- **Descripción:** Estrés relacionado con plazos de entrega y carga de trabajo.
- **Nivel de Riesgo:** Moderado.
- **Medidas de Prevención:** Gestión adecuada de la carga de trabajo, establecimiento de plazos realistas, y creación de un entorno de trabajo positivo y de apoyo.

4. Riesgos Químicos:

- **Descripción:** Uso mínimo de productos de limpieza y mantenimiento.
- **Nivel de Riesgo:** Bajo.
- **Medidas de Prevención:** Uso de productos no tóxicos y almacenamiento seguro de los mismos.

5. Riesgos Relacionados con Incendios:

- **Descripción:** Posibilidad de incendios eléctricos.
- **Nivel de Riesgo:** Moderado.
- **Medidas de Prevención:** Instalación de extintores adecuados, salidas de emergencia señalizadas y prácticas regulares de evacuación.

6. Riesgos de Seguridad de la Información:

- **Descripción:** Protección de datos sensibles.
- **Nivel de Riesgo:** Moderado.
- **Medidas de Prevención:** Implementación de medidas de seguridad, como firewalls, encriptación y políticas de contraseñas seguras.

El Producto o Servicio

Descripción del Producto: GymWizz es una aplicación de gestión para gimnasios, diseñada para optimizar la administración de clientes, finanzas y comunicación. Las características técnicas incluyen:

- **Sistema de Registro e Inicio de Sesión:** Seguridad avanzada con datos cifrados.
- **Dashboard Interactivo:** Visualización de ganancias mensuales, estadísticas de clientes y notificaciones importantes.
- **Gestión de Clientes Morosos:** Identificación automática y gestión de deudas pendientes.
- **Envío de rutinas Personalizadas:** Algoritmo que genera y envía rutinas de entrenamiento por correo electrónico basado en los datos personales del cliente.

- **Interfaz Gráfica Atractiva y Fácil de Usar:** Diseño intuitivo para una navegación sencilla.

Tecnología Empleada:

- **JavaFX**
- **Base de Datos:** MYSQL
- **Servicios en la Nube:** AWS para hosting y almacenamiento.
- **Seguridad:** Implementación de encriptación de datos.

Necesidades que Satisface:

- **Eficiencia Administrativa:** Simplifica la gestión diaria del gimnasio.
- **Mejora en la comunicación:** Facilita la interacción personalizada con los clientes.
- **Aumento de la Retención de Clientes:** Ofrece un servicio más personalizado y eficiente.

Factores de Innovación:

- **Personalización Avanzada:** Utiliza datos personales para crear rutinas de entrenamiento individualizadas.
- **Automatización de Tareas:** Reduce la carga administrativa mediante procesos automáticos.
- **Interfaz Amigable:** Mejora la experiencia del usuario con un diseño intuitivo y atractivo.

GymWizz no sólo responde a las necesidades actuales de los gimnasios, sino que también incorpora elementos innovadores que lo diferencian de las soluciones existentes en el mercado, proporcionando un valor añadido tanto para los administradores de gimnasios como para los usuarios finales.

12.Conclusiones y valoración personal

La experiencia en las FCTs me enseñó técnicas efectivas para acelerar mi proceso de aprendizaje y asimilar nuevos conocimientos de manera más rápida y efectiva. Esto fue especialmente útil al enfrentar desafíos del desarrollo de GymWizz, permitiéndome avanzar de manera más eficiente en el proyecto. Además la experiencia profesional adquirida durante este período fue de mucha ayuda. Aprendí a adaptarme a un entorno laboral real, a trabajar de manera autónoma y a resolver problemas de manera efectiva.

13.BIBLIOGRAFIA

- **Flaticon:** Plataforma que ofrece una amplia variedad de recursos gráficos gratuitos para proyectos: <https://www.flaticon.es/>
- **JavaMail API:** Documentación oficial de la API JavaMail, utilizada para el envío y recepción de correos electrónicos en aplicaciones Java: <https://javaee.github.io/javamail/>
- **CodeAmir en YouTube:** Canal de YouTube que ofrece tutoriales y recursos sobre JavaFX, ideal para aprender a desarrollar aplicaciones gráficas en Java: <https://www.youtube.com/@codeamir/videos>
- **DiscoDuroDeRoer en YouTube:** Canal de YouTube que proporciona tutoriales de Programación de todo tipo <https://www.youtube.com/@DiscoDurodeRoer>
- **Tutorial de TableView por CodeAmir en YouTube:** Tutorial detallado en YouTube sobre TableView en JavaFX <https://www.youtube.com/watch?v=V9nDH2iBJSM&t=526s>
- **Stack Overflow:** comunidad de desarrolladores para hacer y encontrar preguntas: <https://stackoverflow.com/>