REPÚBLICA DEL ECUADOR

INSTITUTO SUPERIOR TECNOLÓGICO TENA
Tecnología, Innovación y Desarrollo

CARRERA DE TECNOLOGÍA SUPERIOR EN

DESARROLLO DE SOFTWARE

DESARROLLO DE UNA APLICACIÓN PARA LA GESTIÓN DE CLIENTES, EN EL GIMNASIO SPARTANS GYM, UBICADO EN LA CIUDAD DE ARCHIDONA, PROVINCIA DE NAPO.

Trabajo de Integración Curricular, presentado como requisito parcial para optar por el título de Tecnólogo Superior de en Desarrollo de Software.

AUTOR: MILTON ANDRÉS PAZOS CHURACU

TUTOR: ING. JUAN M. ESPÍN MONTESDEOCA

Tena - Ecuador

2024-IS

APROBACIÓN DEL TUTOR

ING. JUAN M. ESPÍN MONTESDEOCA

PROFESOR DEL INSTITUTO SUPERIOR TECNOLÓGICO TENA.

CERTIFICA:

En calidad de Tutor del Proyecto Integrador denominado: DESARROLLO DE UNA APLICACIÓN PARA LA GESTIÓN DE CLIENTES, EN EL GIMNASIO SPARTANS GYM, UBICADO EN LA CIUDAD DE ARCHIDONA, PROVINCIA DE NAPO, de autoría del señor MILTON ANDRES PAZOS CHURACU, con CC. 150114754-8 estudiante de la Carrera de Tecnología Superior el Desarrollo de Software del Instituto Superior Tecnológico Tena, CERTIFICO que se ha realizado la revisión prolija del Trabajo antes citado, cumple con los requisitos de fondo y de forma que exigen los respectivos reglamentos e instituciones.

Tena, 6 de septiembre de 2024

Ing. Juan M. Espín Montesdeoca

TUTOR DEL TIC

CERTIFICACIÓN DEL TRIBUNAL CALIFICADOR

Tena, 06 de septiembre de 2024

Los Miembros del Tribunal de Grado **ING. ITALO LARA, ING. FERNANDO NUÑEZ, ING. GONZALO GUANIPATIN** abajo firmantes, certificamos que el Trabajo de Titulación denominado: DESARROLLO DE UNA APLICACIÓN PARA LA GESTIÓN DE CLIENTES, EN EL GIMNASIO SPARTANS GYM, UBICADO EN LA CIUDAD DE ARCHIDONA, PROVINCIA DE NAPO, presentado por MILTON ANDRES PAZOS CHURACU, con CC: 150114754-8, estudiante de la Carrera de Tecnología Superior en Desarrollo de Software del Instituto Superior Tecnológico Tena, ha sido corregida y revisada; por lo que autorizamos su presentación.

Atentamente;

ING. ITALO LARA

PRESIDENTE DEL TRIBUNAL

ING. FERNANDO NUÑEZ

MIEMBRO DEL TRIBUNAL

ING. GONZALO GUANIPATIN

MIEMBRO DEL TRIBUNAL

AUTORÍA

Yo, MILTON ANDRES PAZOS CHURACU, con CC: 1501147548, declaro ser autor del presente Trabajo de Titulación denominado: DESARROLLO DE UNA APLICACIÓN PARA LA GESTIÓN DE CLIENTES, EN EL GIMNASIO SPARTANS GYM, UBICADO EN LA CIUDAD DE ARCHIDONA, PROVINCIA DE NAPO y absuelvo expresamente al Instituto Superior Tecnológico Tena, y a sus representantes jurídicos de posibles reclamos o acciones legales por el contenido de la misma.

Adicionalmente acepto y autorizo al Instituto Superior Tecnológico Tena, la publicación de mi trabajo de Titulación en el repositorio institucional- biblioteca Virtual.

AUTOR:

MILTON ANDRES PAZOS CHURACU

CÉDULA: 150114754-8

FECHA: Tena, 05 de septiembre de 2024

CARTA DE AUTORIZACIÓN POR PARTE DEL AUTOR

Yo, MILTON ANDRES PAZOS CHURACU, declaro ser autor del Trabajo de Titulación titulado: DESARROLLO DE UNA APLICACIÓN PARA LA GESTIÓN DE CLIENTES, EN EL GIMNASIO SPARTANS GYM, UBICADO EN LA CIUDAD DE ARCHIDONA, PROVINCIA DE NAPO, como requisito para la obtención del Título de: TECNÓLOGO SUPERIOR EN DESARROLLO DE SOFTWARE: autorizo al Sistema Bibliotecario del Instituto Superior Tecnológico Tena, para que con fines académicos, muestre al mundo la producción intelectual del Instituto, a través de la visualización de su contenido que constará en el Repositorio Digital Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el RDI, en las redes de información del país y del exterior, con las cuales tenga convenio el Instituto. El Instituto Superior Tecnológico Tena, no se responsabiliza por el plagió o copia del presente trabajo que realice un tercero. Para constancia de esta autorización, en la ciudad de Tena, 02 de abril de 2020, firma el autor.

AUTOR: Milton Andrés Pazos Churacu

FIRMA:
CÉDULA: 150114754-8
DIRECCIÓN: Barrio 13 de abril calle 7 transversal 18
CORREO ELECTRÓNICO: milton10andres1999@gmail.com
CELULAR: 0991252055

DATOS COMPLEMENTARIOS
TUTOR: Ing. Juan M. Espín Montesdeoca
TRIBUNAL DEL GRADO:
ING. ITALO LARA. (Presidente).
ING. FERNANDO NUÑEZ. (Miembro).
ING. GONZALO GUANIPATIN. (Miembro).

DEDICATORIA

A mi madre, por ser mi fuente de inspiración y fortaleza. Tu amor incondicional, tus sacrificios y tu fe en mí me han dado la energía para superar cada obstáculo, ha sido mi guía y mi refugio en los momentos difíciles, y gracias a ti he aprendido a luchar por mis sueños sin rendirme. A mi padrastro, por ser un pilar en mi vida, su apoyo, consejo y confianza en mis capacidades han sido fundamentales en este camino, siendo un ejemplo de dedicación y esfuerzo, y siempre te estaré agradecido por todo lo que has hecho por mí.

También a mis hermanos, por ser mis compañeros de vida, cada uno de ustedes ha aportado alegría, risas y motivación en mi vida, recordándome siempre la importancia de la familia. Gracias por estar a mi lado y por ser mi mayor motivación para seguir adelante.

A mis compañeros, por compartir este viaje académico conmigo. Juntos hemos enfrentado desafíos, hemos crecido y hemos aprendido. Su amistad y apoyo han hecho que este camino sea más llevadero y enriquecedor.

Y finalmente a mis profesores, por compartir su conocimiento y por guiarme con paciencia y dedicación, su compromiso con nuestra formación ha sido un faro en este camino, y sus enseñanzas han dejado una huella imborrable en mi vida. Gracias por creer en mí y por impulsarme a ser mejor cada día.

Este logro es tanto mío como de todos ustedes.

AGRADECIMIENTO

Quiero dedicar unas palabras de profundo agradecimiento a mi querida madre, desde mis primeros días de estudio hasta el presente, su inquebrantable apoyo y sacrificio han sido mi mayor fuente de inspiración. Gracias por tu constante aliento, por creer en mí incluso en los momentos más difíciles, y por ser mi guía amorosa en cada paso del camino educativo, tu dedicación y amor incondicional han sido el motor que impulsa mi perseverancia y éxito académico.

Agradezco también a mis seres queridos, cuyo apoyo emocional y motivación han sido fundamentales en mi travesía educativa, vuestras palabras de ánimo y presencia constante han sido un faro de esperanza en los momentos desafiantes. Cada gesto de aliento y cada momento compartido han fortalecido mi determinación y me han recordado la importancia de cultivar relaciones significativas mientras persigo mis metas académicas.

No puedo pasar por alto la contribución invaluable de mis estimados docentes y profesores, que a través de su dedicación, conocimiento y paciencia, han moldeado mi comprensión del mundo y han encendido mi pasión por el aprendizaje, gracias por desafiarme a superar mis límites, por impartirme no solo conocimientos académicos, sino también habilidades de vida que llevaré conmigo más allá del aula. Vuestra influencia perdurará en mí como un testimonio de su compromiso con la educación y el desarrollo integral de vuestros estudiantes.

ÍNDICE

1 TEMA

DESARROLLO DE UNA APLICACIÓN PARA LA GESTIÓN DE CLIENTES, EN EL GIMNASIO SPARTANS GYM, UBICADO EN LA CIUDAD DE ARCHIDONA, PROVINCIA DE NAPO.

2 RESUMEN

El presente proyecto tiene como objetivo DESARROLLO DE UNA APLICACIÓN PARA LA GESTIÓN DE CLIENTES, EN EL GIMNASIO SPARTANS GYM, UBICADO EN LA CIUDAD DE ARCHIDONA, PROVINCIA DE NAPO. La aplicación busca solucionar los problemas de eficiencia y calidad en la gestión de datos de clientes, membresías, productos y medidas físicas que actualmente se manejan mediante hojas de cálculo. La implementación de esta aplicación permitirá una administración más eficiente, mejorando la experiencia del usuario y la satisfacción del cliente. El proyecto abarca el análisis de necesidades, el diseño de prototipos, la creación de los componentes del sistema y la implementación final, garantizando un sistema integral y automatizado para el gimnasio.

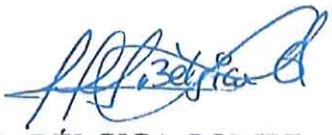
Palabras clave: Gestión de clientes, aplicación de escritorio, eficiencia administrativa, gimnasio, SPARTANS GYM, Archidona.

ABSTRACT

The objective of this project is DEVELOPMENT OF AN APPLICATION FOR CUSTOMER MANAGEMENT, IN THE SPARTANS GYM, LOCATED IN THE CITY OF ARCHIDONA, PROVINCE OF NAPO. The application seeks to solve efficiency and quality problems in the management of customer data, memberships, products and physical measurements that are currently managed through spreadsheets. The implementation of this application will allow for more efficient administration, improving user experience and customer satisfaction. The project covers needs analysis, prototype design, creation of system components and final implementation, guaranteeing a comprehensive and automated system for the gym.

Keywords: Customer management, desktop application, administrative efficiency, gym, SPARTANS GYM, Archidona.

Reviewd by:



TCHR. BÉLGICA GOMEZ

TEACHER OF THE LENGUAGE CENTER

3 FUNDAMENTACIÓN DEL TEMA

3.1 Necesidad

La gestión eficiente de los clientes en un gimnasio es fundamental para mejorar la experiencia del usuario, aumentar la retención de clientes y optimizar los procesos administrativos. En el Gimnasio SPARTANS GYM, se ha identificado la necesidad de un sistema que permita una gestión más eficiente y organizada de los datos de los clientes, como inscripciones, pagos, asistencia y servicios utilizados. Actualmente, la falta de un sistema informatizado genera retrasos, errores en la información y una administración ineficiente.

3.2 Actualidad

En la actualidad, la tecnología juega un papel crucial en la administración de cualquier negocio. La implementación de aplicaciones móviles y sistemas de gestión se ha convertido en una tendencia clave en la industria del fitness y los gimnasios. Sin embargo, el Gimnasio SPARTANS GYM no cuenta con una aplicación que facilite la gestión de clientes, lo que limita la capacidad del personal administrativo para realizar un seguimiento adecuado y eficiente de las actividades diarias. Esta carencia afecta tanto a la satisfacción del cliente como a la eficiencia operativa del gimnasio.

3.3 Importancia

La implementación de una aplicación para la gestión de clientes en SPARTANS GYM es de vital importancia para mejorar la eficiencia operativa, reducir errores administrativos y proporcionar un servicio de alta calidad a los clientes. Una gestión eficaz de los clientes permite personalizar las ofertas y servicios, mejorar la comunicación y aumentar la satisfacción y fidelidad

de los usuarios. Además, facilita el trabajo del personal administrativo, liberándolos de tareas repetitivas y permitiéndoles enfocarse en actividades de mayor valor añadido.

4 Presentación del problema profesional a responder

El Gimnasio SPARTANS GYM, ubicado en la ciudad de Archidona, Provincia de Napo, enfrenta problemas en la gestión de sus clientes, membresías y operaciones internas debido a la falta de un sistema eficiente de gestión de información. La administración actual se realiza de manera manual, lo que genera ineficiencias, errores en el manejo de datos, y dificultades en la generación de reportes y análisis.

Campo: Desarrollo de software

Área: Programación Visual

Aspecto: Sistema dinámico de escritorio utilizando BD

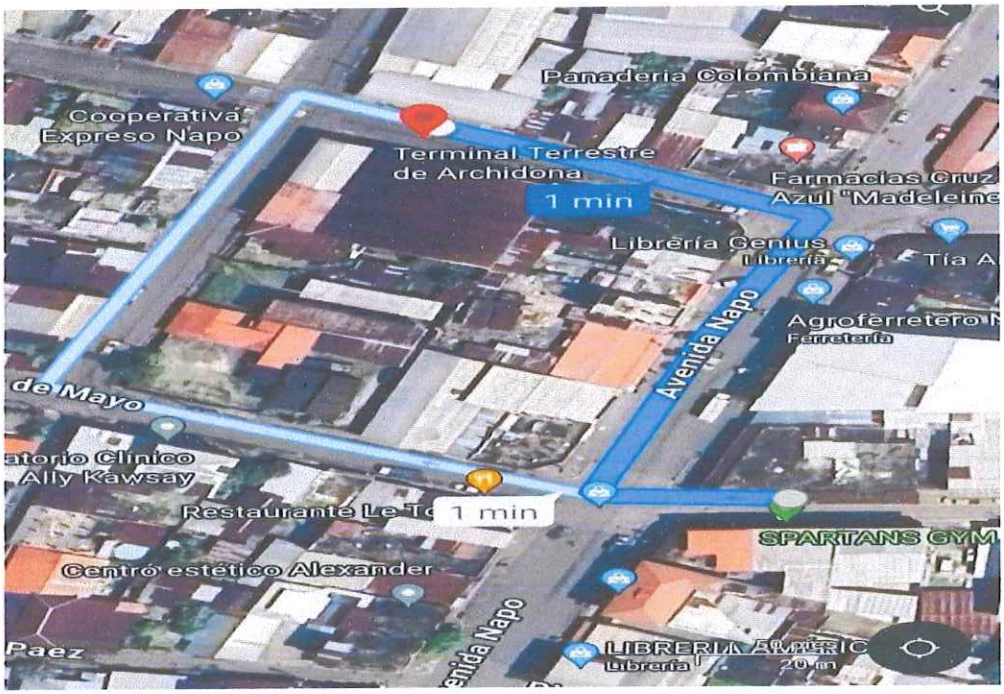
Sector: SPARTANS GYM

Línea de investigación: Tecnologías de la información y comunicación.

5 Delimitación

5.1 Delimitación Espacial

5.1.1 Figura 1: Imagen de la ubicación



Indicaciones para llegar a SPARTANS GYM una vez que haya llegado a la parada de expreso napo (Terminal Terrestre de Archidona) tiene que caminar hasta donde están los semáforos de ahí toma a mano derecha por la Avenida Napo camina hasta llegar a la esquina de ahí tiene que dirigirse a mano izquierda donde podrá ver una parada de taxis camina por la Av. 1ro de Mayo y podrá llegar a SPARTANS GYM

5.2 Delimitación Temporal

El proyecto se efectuará en el Periodo Académico 2024-IS.

5.3 Delimitación Técnica

La presente investigación tiene como alcance el diseño e implementación de un Sistema de Gestión de Clientes, Membresías, Ventas y datos del cuerpo que contempla los siguientes módulos:

- Módulo de Inicio de Sesión: Este módulo proporciona campos para el ingreso de usuario y contraseña, junto con el logo del gimnasio. También incluye dos botones: uno para iniciar sesión y otro para cancelar.

- Módulo de Registro de Clientes: Este módulo permite el registro de nuevos clientes e incluye campos para la cédula, nombre, apellido, dirección, número de teléfono y género. Además, cuenta con botones para guardar, modificar, eliminar y cancelar, así como una barra de búsqueda para facilitar la ubicación de clientes registrados.

- Módulo de Membresías: Este módulo permite gestionar las membresías de los clientes de forma integral. Ofrece campos para registrar la cédula del cliente, así como fechas de inicio y fin de la membresía, información de pago, notas adicionales y detalles sobre la membresía seleccionada (como membresía estándar o VIP). También proporciona opciones para asignar un entrenador, si es necesario. Incluye botones para guardar, modificar, eliminar y cancelar registros, junto con una barra de búsqueda para una navegación eficiente.

- Módulo de Productos: Este módulo simplifica la gestión de productos disponibles en el gimnasio. Incluye campos para el nombre del producto, su descripción, cantidad disponible y precio. Los usuarios pueden gestionar fácilmente los productos utilizando los botones de guardar, modificar, eliminar y cancelar.

- Módulo de Ventas: Este módulo agiliza el proceso de ventas de productos en el gimnasio. Permite seleccionar el producto deseado y especificar la cantidad a vender. Los usuarios pueden completar la transacción con un solo clic utilizando los botones correspondientes para vender o cancelar.
- Módulo de Reportes: Este módulo proporciona la funcionalidad para generar informes sobre diversos aspectos del gimnasio, como la asistencia de los clientes, las ventas de productos, etc.
- Módulo de Datos de Medición: Este módulo permite el registro de datos de medición física de los clientes, como altura, peso, cintura, glúteos, piernas y pecho. Incluye botones para guardar, modificar, eliminar y cancelar, así como una barra de búsqueda para facilitar la ubicación de registros previos.

5.4 Unidades de Observación

Las unidades de observación que se contemplan para este trabajo están enfocadas directamente en el desarrollo y aplicación del software.

6 Beneficiarios

6.1 Directos

- Gimnasio SPARTANSGYM

6.2 Indirectos

- Otros gimnasios locales.
- Clientes del Gimnasio SPARTANS GYM.

7 OBJETIVOS

7.1 Objetivo General

- Desarrollar una aplicación para la Gestión de Clientes, en el Gimnasio SPARTANS GYM, ubicado en la ciudad de Archidona, Provincia de Napo.

7.2 Objetivos Específico

- Analizar las necesidades del gimnasio mediante entrevistas y estudiar aplicaciones existentes.
- Diseñar la interfaz de usuario.
- Generar la codificación de los componentes principales del sistema.
- Implementar resultados de pruebas de integración y aceptación del usuario final.

8 ASIGNATURAS INTEGRADORAS

Tabla 1

Asignaturas integradoras

ASIGNATURAS INTEGRADORAS	
Asignaturas	**Resultados de Aprendizaje**
Programación Visual	• Diseñar interfaces de usuario intuitivas y atractivas utilizando herramientas y principios de diseño visual. • Implementar lógica de programación orientada a eventos para mejorar la interactividad y la experiencia del usuario. • Utilizar técnicas de depuración y pruebas para identificar y corregir errores en aplicaciones visuales. • Integrar elementos multimedia y gráficos en aplicaciones para mejorar la presentación de información. • Desarrollar habilidades para trabajar con frameworks y bibliotecas específicas para la creación de interfaces gráficas.
Base de Datos Avanzada	• Diseñar y crear modelos de bases de datos complejos que cumplan con requisitos específicos de almacenamiento y recuperación de información.

	• Optimizar el rendimiento de consultas mediante el uso de índices, vistas y técnicas de normalización. • Implementar y administrar sistemas de gestión de bases de datos relacionales y no relacionales.
Calidad de Software	• Aplicar metodologías y estándares de calidad de software para garantizar la fiabilidad y la eficiencia de los productos desarrollados. • Realizar pruebas exhaustivas de software utilizando técnicas como pruebas unitarias, de integración y de aceptación. • Identificar y corregir defectos de software mediante la aplicación de técnicas de depuración y análisis de código.

9 FUNDAMENTACIÓN TEÓRICA

9.1 Gestión de Clientes en Gimnasios

9.1.1 Importancia de la Gestión de Clientes:

La gestión de clientes es esencial para cualquier negocio, incluyendo gimnasios, ya que facilita mantener relaciones organizadas y eficientes con los usuarios, lo que aumenta su satisfacción y lealtad. Una adecuada gestión de clientes mejora la eficiencia operativa y proporciona un servicio de alta calidad (Martínez, 2018).

9.1.2 Sistemas de Información para la Gestión de Clientes:

Los sistemas de información han transformado la gestión de datos de clientes, permitiendo almacenar, organizar y analizar grandes volúmenes de información. Esto facilita la toma de decisiones y automatiza procesos administrativos, reduciendo errores y mejorando la eficiencia (García & López, 2020).

9.1.3 Desarrollo de Software en Visual Basic:

Visual Basic es un entorno de desarrollo integrado (IDE) y un lenguaje de programación orientado a eventos, desarrollado por Microsoft. Es conocido por su facilidad de uso y capacidad para crear aplicaciones de escritorio robustas. Permite construir aplicaciones rápidamente con una curva de aprendizaje relativamente baja, facilitando la creación de interfaces gráficas intuitivas y la integración con bases de datos (Fernández, 2019). Visual Basic 2019 incluye características avanzadas como IntelliSense y herramientas de colaboración en tiempo real (ElCamino.dev, 2023).

9.1.4 Bases de Datos y su Importancia en la Gestión de Clientes:

Las bases de datos son fundamentales para almacenar y gestionar la información de los clientes de manera segura y eficiente. Las bases de datos relacionales permiten organizar la

información de forma estructurada, facilitando su acceso y manejo (Pérez, 2017). MySQL Workbench 8.0 es una herramienta potente para la administración de bases de datos, proporcionando una interfaz gráfica para la configuración, administración y diseño de bases de datos MySQL.

9.1.5 Desarrollo de Software y Metodología Waterfall:

El modelo Waterfall, o en cascada, es una metodología de desarrollo de software que sigue un enfoque lineal y secuencial. Divide el proceso de desarrollo en fases distintas: análisis de requisitos, diseño, implementación, pruebas, despliegue y mantenimiento. Es ideal para proyectos con requisitos bien definidos y estables (Rodríguez, 2021)

9.1.6 Importancia del Ejercicio en Gimnasios:

Ir al gimnasio no solo mejora la salud física, sino que también contribuye al bienestar mental. La actividad física regular ayuda a prevenir enfermedades crónicas, mejora el estado de ánimo y aumenta la energía. Además, los gimnasios ofrecen un entorno social que puede ser motivador para muchas personas (Gimnasio.net, 2022).

9.1.7 Frameworks y Componentes en Visual Basic:

El uso de frameworks como Guna Framework puede mejorar significativamente el desarrollo de aplicaciones en Visual Basic. Estos frameworks proporcionan controles y componentes adicionales para la creación de interfaces de usuario modernas y eficientes, optimizando el rendimiento y facilitando el desarrollo de aplicaciones complejas (Guna Framework, 2023).

10 Marco Legal

Ley de Protección de Datos Personales:

En Ecuador, la Ley de Protección de Datos Personales establece normativas para el tratamiento y protección de datos personales, garantizando la privacidad y seguridad de la información de los clientes (Asamblea Nacional del Ecuador, 2021).

Normas de Seguridad Informática:

Las normas de seguridad informática son cruciales para proteger la integridad y confidencialidad de la información en los sistemas de gestión de clientes. Implementar políticas de seguridad y tecnologías de encriptación es vital para prevenir accesos no autorizados (Álvarez, 2020).

11 Marco Conceptual

Gestión de Clientes (CRM):

La Gestión de Relaciones con los Clientes (CRM) centraliza la información de los clientes, facilita el seguimiento de sus actividades y mejora la personalización del servicio (López, 2018).

Desarrollo de Software:

El desarrollo de software abarca concebir, diseñar, programar, documentar, probar y mantener aplicaciones y frameworks de software. Incluye fases como el análisis de requisitos, diseño del sistema, implementación, pruebas y mantenimiento (González, 2019).

Modelo Waterfall:

El modelo Waterfall sigue un enfoque lineal y secuencial en el desarrollo de software, ideal para proyectos con requisitos bien definidos y estables (Rodríguez, 2021).

Bases de Datos:

Una base de datos es un conjunto de datos organizados que facilita su acceso, gestión y actualización. Las bases de datos relacionales utilizan un modelo tabular para representar datos y sus relaciones, permitiendo consultas eficientes (Morales, 2018).

Aplicaciones de Escritorio en Visual Basic:

Visual Basic es una herramienta poderosa para el desarrollo de aplicaciones de escritorio, permitiendo crear interfaces de usuario amigables y conectarse a diversas bases de datos. Es ideal para entornos empresariales que requieren soluciones rápidas y personalizadas para la gestión de datos (Hernández, 2020).

12 METODOLOGÍA

12.1 Materiales

Para llevar a cabo el desarrollo de la aplicación para la gestión de clientes en el Gimnasio SPARTANS GYM, se utilizarán los siguientes materiales:

- Computadoras con el entorno de desarrollo Visual Basic instalado.
- Bases de datos relacionales: MySQL Workbench 8.0 para el almacenamiento de la información de los clientes y demás datos necesarios
- Herramientas de diseño gráfico: Canva para la creación de la interfaz de usuario.

- Documentación y manuales de referencia sobre Visual Basic y desarrollo de software en entornos de escritorio.
- Equipos de prueba para la validación y verificación del software desarrollado.

12.2 Aplicación de la Metodología.

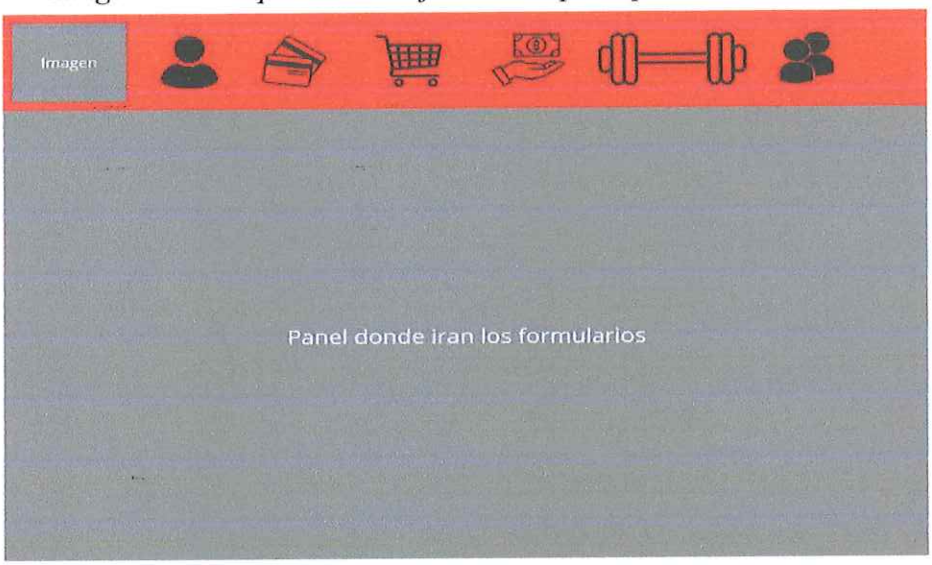
Con un enfoque mejorado en la implementación y desarrollo, se ha implementado las siguientes fases:

Fase de Requisitos: En esta fase, se recopilaron y documentaron todas las necesidades del gimnasio SPARTANS GYM. Esto incluyó entrevistas con los propietarios y el personal para entender los procesos actuales de gestión de clientes y ventas, y determinar qué funcionalidades eran necesarias en el sistema. Se definieron los módulos requeridos: Inicio de Sesión, Registro de Clientes, Membresías, Productos, Ventas, Reportes y Datos de Medición. Se elaboraron especificaciones detalladas para cada módulo, identificando los campos, botones y funcionalidades esenciales.

Fase de Diseño: En esta etapa, se definió la estructura general del sistema con un enfoque en la maquetación del formulario principal.

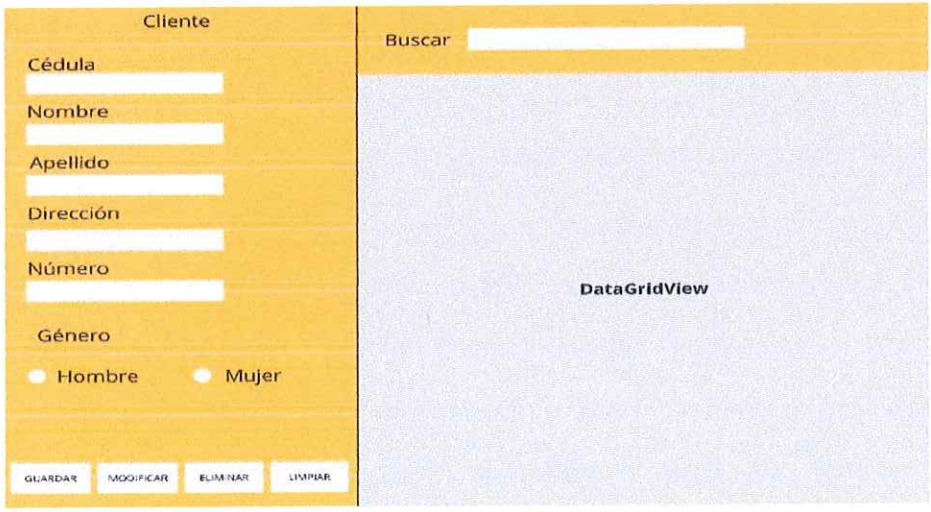
- **Maquetación del Proyecto**: Se diseñó un formulario principal que actúa como un panel central, desde el cual se accede a todos los demás formularios del sistema. Este diseño facilita la navegación y organiza los módulos de manera eficiente.

Imagen de la maquetación del formulario principal.



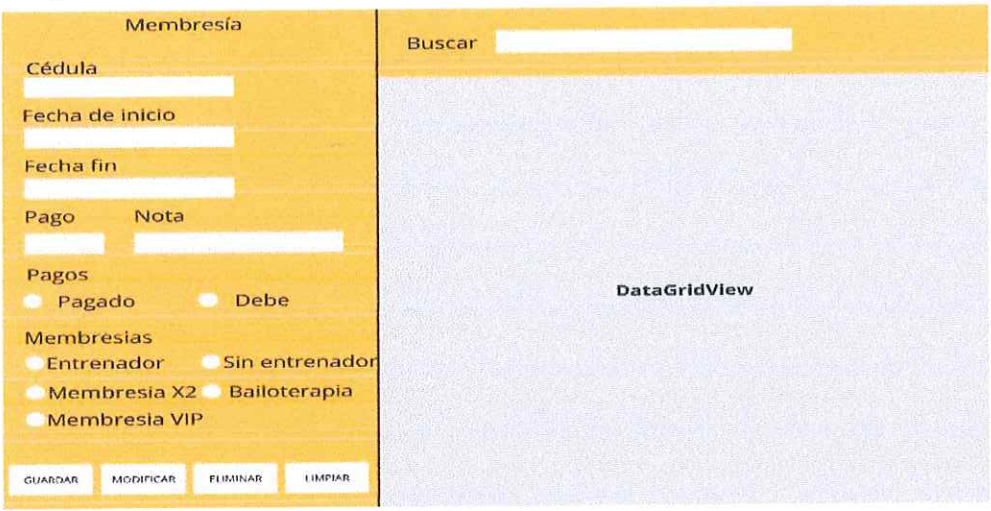
Registro de Clientes: Permite añadir la información de los clientes.

Imagen del módulo de Registro de Clientes:



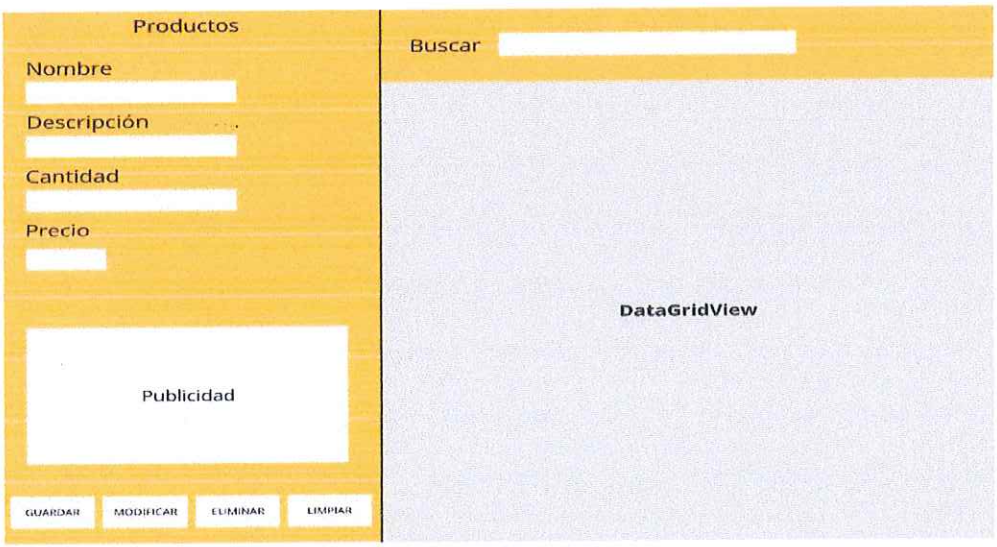
Manejo de Membresías: Facilita la gestión de suscripciones, incluyendo la adición, modificación y eliminación de membresías.

Imagen del módulo de Manejo de Membresías:



Administración de Productos: Gestiona los productos disponibles en el gimnasio.

Imagen del módulo de Administración de Productos:



Procesamiento de Ventas: Maneja las transacciones de venta.

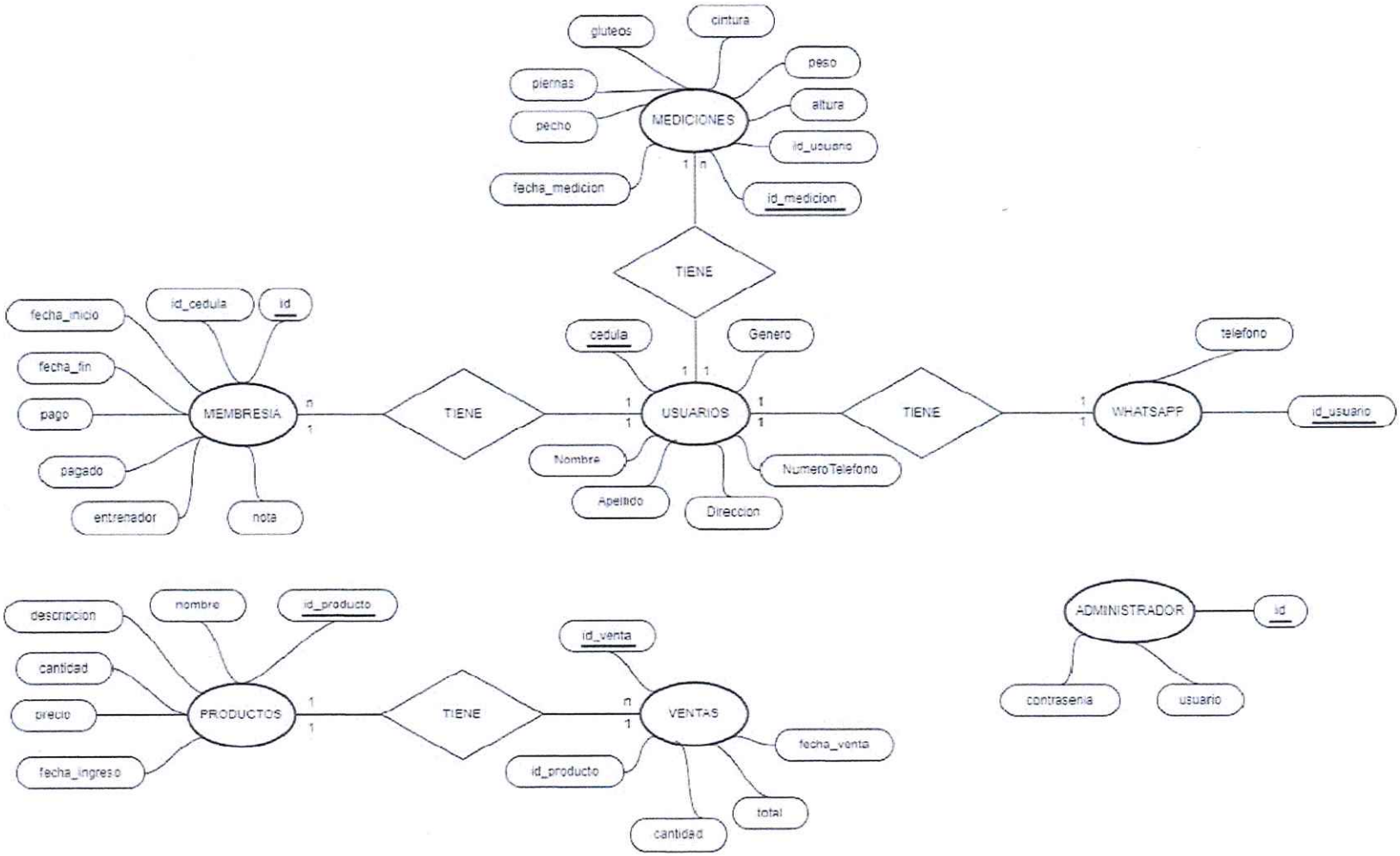
Imagen del módulo de Procesamiento de Ventas:

Reportes: Permite crear informes detallados.

Imagen del módulo de Reportes:

Base de Datos: Se creó una base de datos en MySQL, estructurada para facilitar el acceso y la gestión eficiente de la información.

Imagen del diagrama de la base de datos:



Funciones Clave:

Inicialización: Configura y carga datos al iniciar el formulario.

Guardar: Valida y almacena nuevas membresías en la base de datos.

Modificar: Actualiza la información de las membresías seleccionadas.

Eliminar: Borra membresías seleccionadas de la base de datos.

Validaciones: Asegura la integridad de los datos y ajusta campos automáticamente.

Alerta de Expiración: Notifica sobre membresías próximas a vencer.

Búsqueda: Permite buscar membresías por nombre, apellido o cédula.

Interfaz: Ajusta el formato de celdas en el DataGridView para mejorar la visualización de datos.

Fase de implementación: Durante la fase de implementación, se desarrollaron los módulos siguiendo el diseño especificado utilizando Visual Basic. Los módulos desarrollados incluyen:

Fase de Verificación: En esta fase, se llevaron a cabo pruebas unitarias exhaustivas del sistema para asegurar que todas las funcionalidades operaran correctamente y se cumplieran con los requisitos establecidos por el propietario. Estas pruebas se realizaron para verificar que cada módulo y componente del sistema funcionara de manera aislada y conforme a lo esperado.

Pruebas Unitarias: Se realizaron pruebas unitarias para asegurar que los módulos de Inicio de Sesión, Registro de Clientes, Membresías, Productos, Ventas, Reportes y Datos de Medición funcionaran correctamente. Se verificó la precisión de los datos, la integridad de las transacciones y la usabilidad de la interfaz de usuario en cada módulo.

Fase de Mantenimiento: Una vez desplegado el sistema, se estableció un plan de mantenimiento para asegurar su correcto funcionamiento a corto y largo plazo, esto incluyó la corrección de errores identificados después del lanzamiento, actualizaciones periódicas para mejorar el rendimiento y añadir nuevas funcionalidades según las necesidades del gimnasio. Se proporcionó capacitación al propietario del gimnasio para el uso efectivo del sistema y se ofreció soporte técnico continuo para resolver cualquier problema que pudiera surgir.

13 RESULTADOS

Fase de Requerimientos: Con la información que se recolectó se pudo identificar todos los procesos que se requiere automatizar por lo tanto se identificó los siguientes módulos: Inicio de Sesión, Registro de Clientes, Membresías, Productos, Ventas, Reportes y Datos de Medición. Esta información permitió el diseño para la interfaz del sistema de acuerdo con las necesidades específicas recopiladas.

Fase de Diseño: De acuerdo con la fase de requerimientos, se procedió a elaborar el diseño detallado del sistema utilizando Canva para crear diagramas de flujo, modelos de datos y prototipos de la interfaz de usuario para cada módulo del sistema, como Inicio de Sesión, Registro de Clientes, Membresías, Productos, Ventas, Reportes y Datos de Medición Física. Se definieron las estructuras de la base de datos y las interacciones entre los módulos utilizando MySQL Workbench 8.0. Además, se seleccionaron los colores, estilos y diseño de botones para asegurar una experiencia de usuario coherente y atractiva. El diseño fue iterado y refinado en colaboración con el propietario del gimnasio SPARTANS GYM para garantizar que cumpliera con todas las expectativas y necesidades identificadas en la fase de requerimientos.

Imágenes de los Módulos Diseñados: A continuación, se presentan imágenes de los diseños de los módulos mencionados, mostrando la interfaz y las funcionalidades principales de cada uno (ver Figuras 1 a 8):

Figura 1. Módulo de Inicio de Sesión.

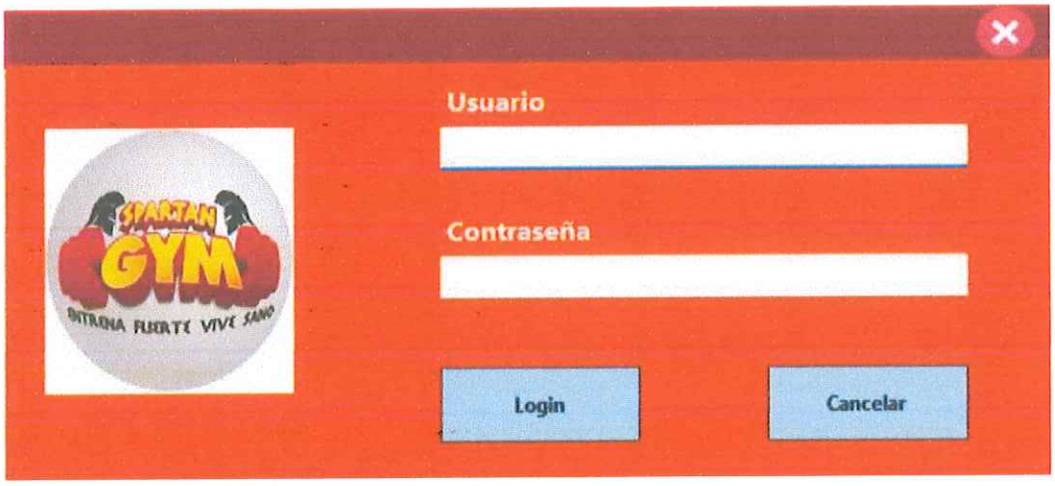


Figura 2. Módulo de Registro de Clientes.

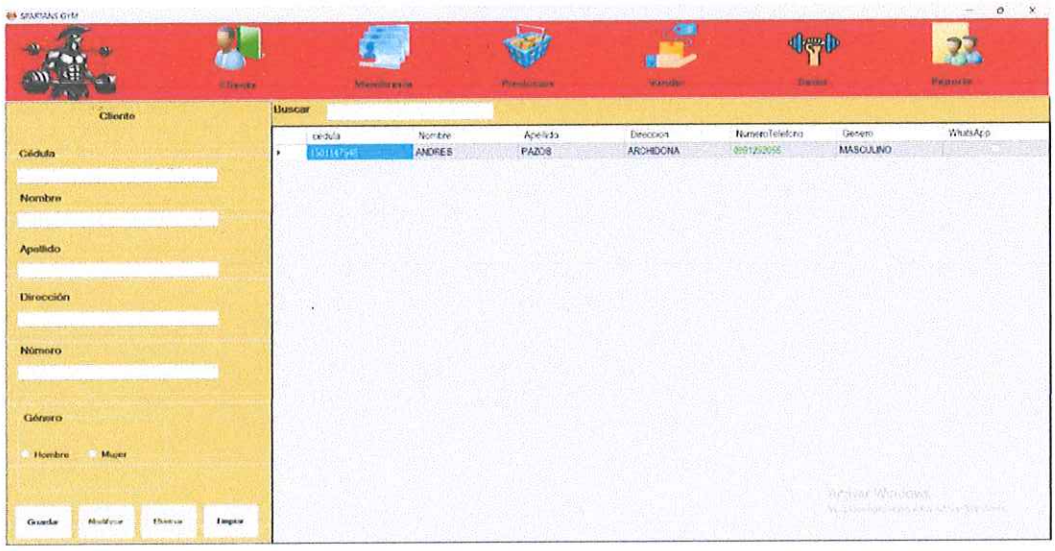


Figura 3. Módulo de Membresías.

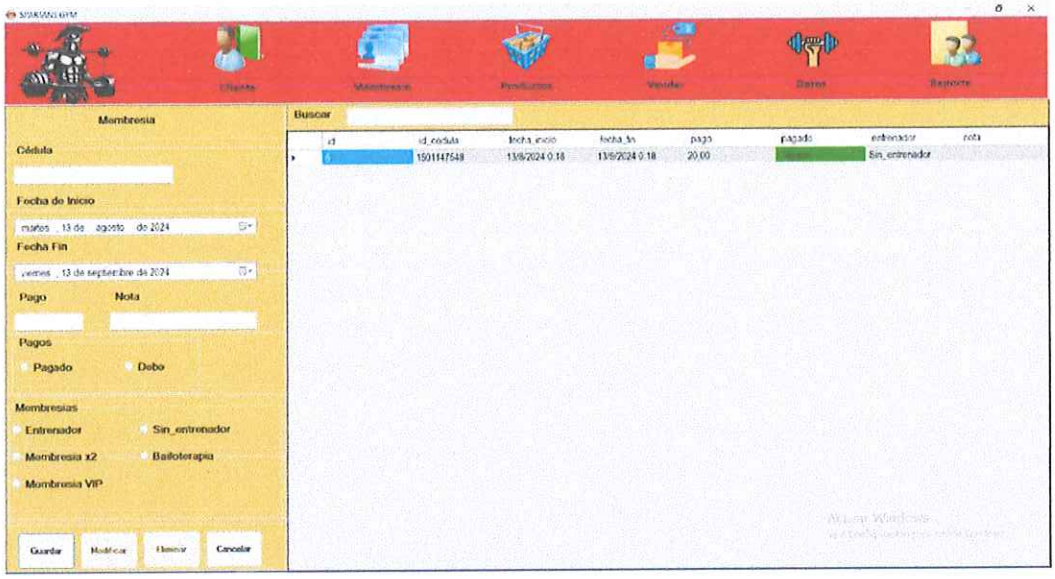


Figura 4. Módulo de Productos.

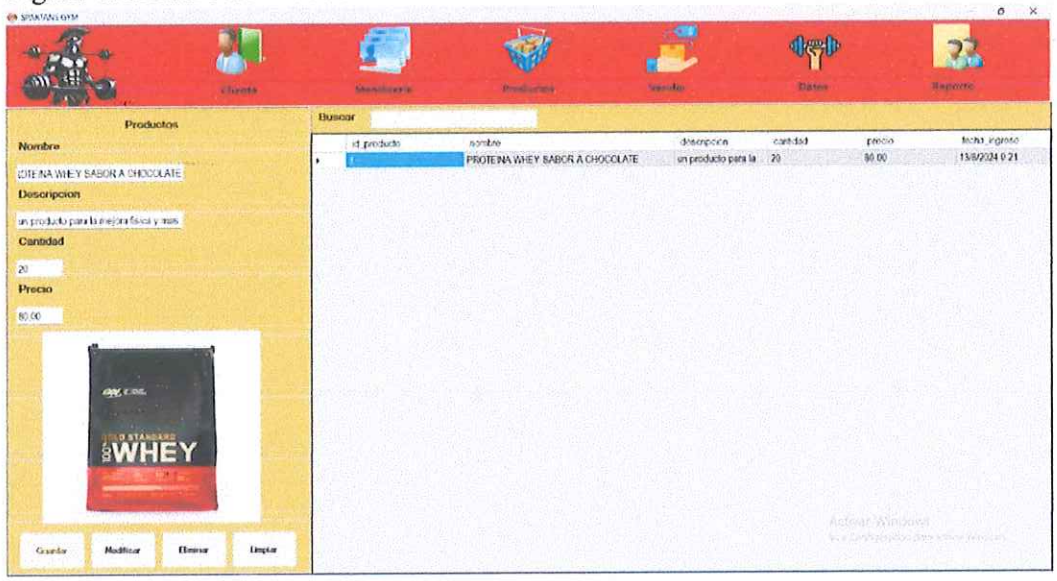


Figura 5. Módulo de Ventas.

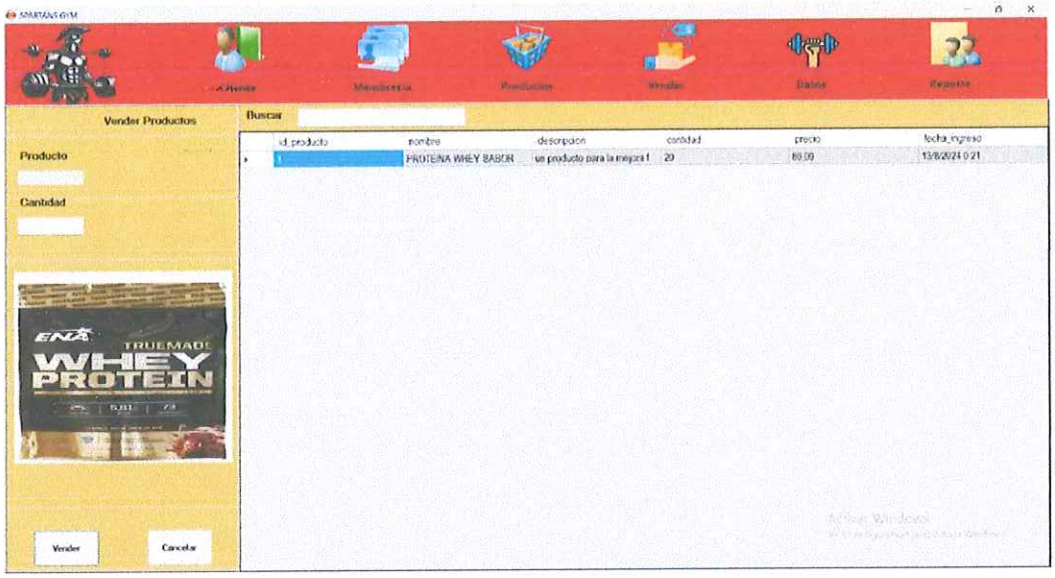
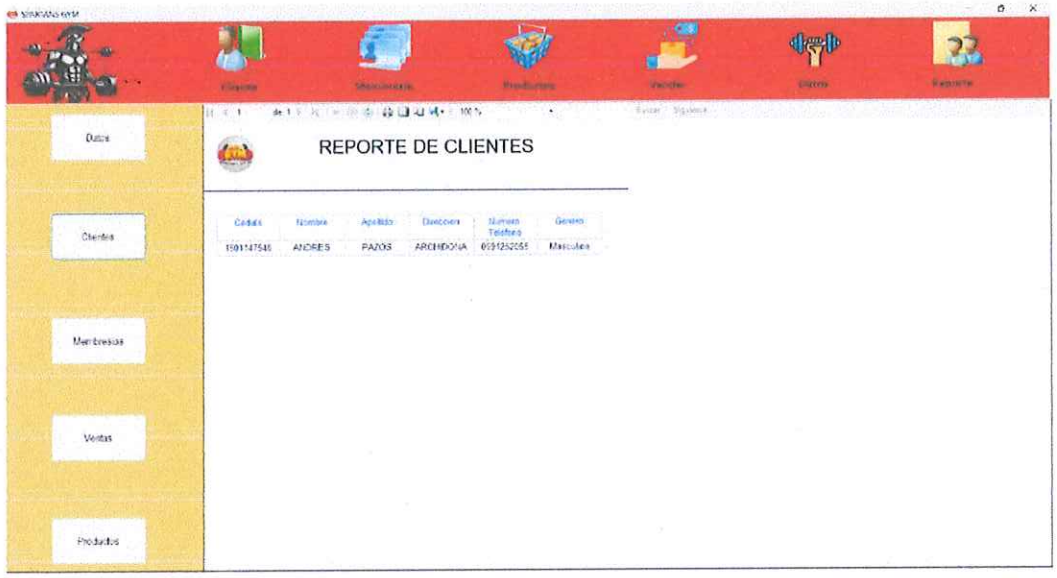


Figura 6. Módulo de Reportes.



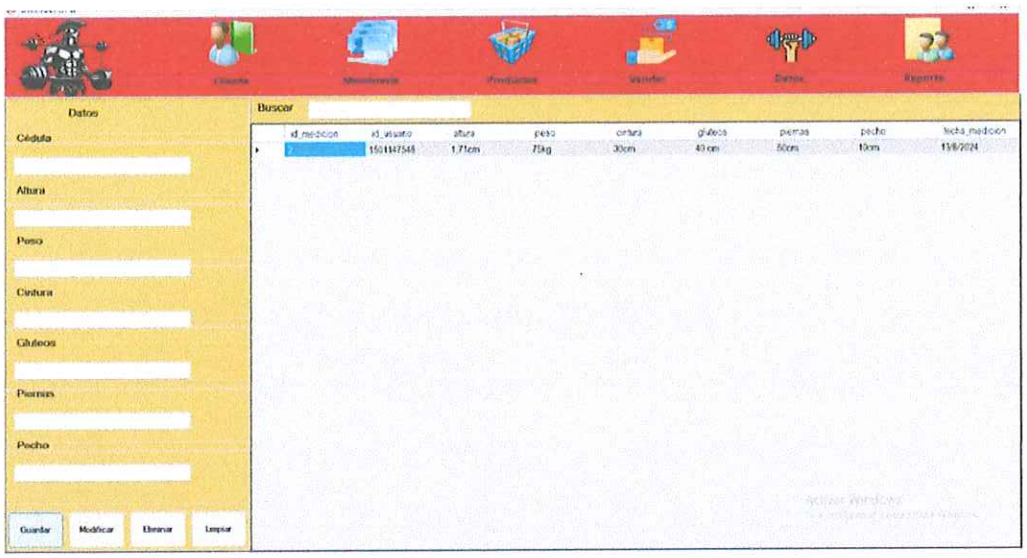
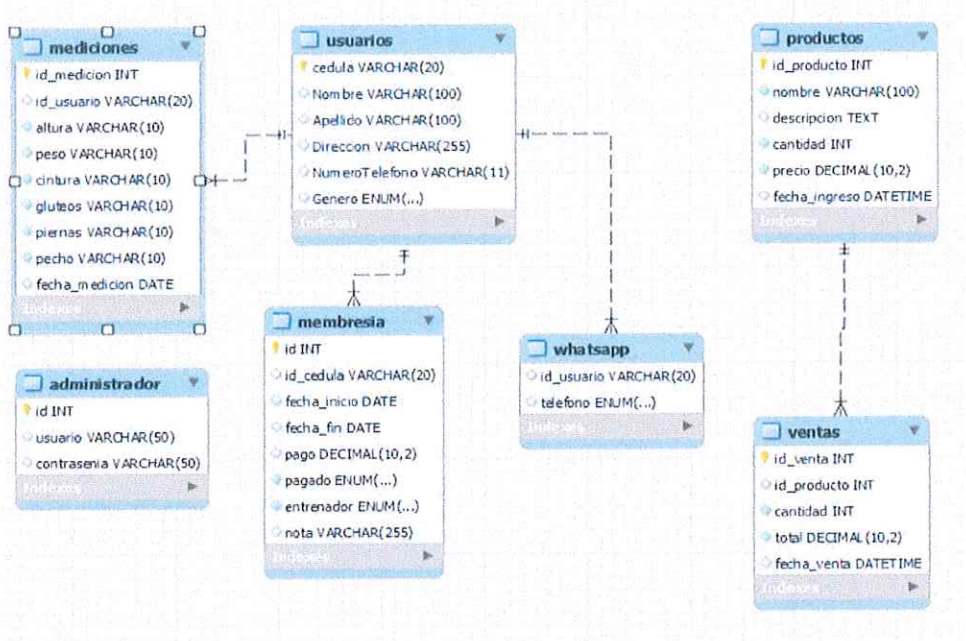


Figura 8. base de datos.



Fase de Verificación: Se realizaron pruebas unitarias exhaustivas para verificar la funcionalidad de cada módulo, asegurando que todos los componentes operaran de acuerdo con los requisitos establecidos. Se verificó la precisión de los datos, la integridad de las transacciones y la usabilidad de la interfaz.

Imágenes de las pruebas realizadas. A continuación, se presentan imágenes relacionadas con las pruebas unitarias realizadas para cada módulo del sistema (ver Figuras 1 a 5):

Figura 1. *Captura de pantalla de la prueba unitaria del módulo de Inicio de Sesión, mostrando los resultados de la verificación de datos.*

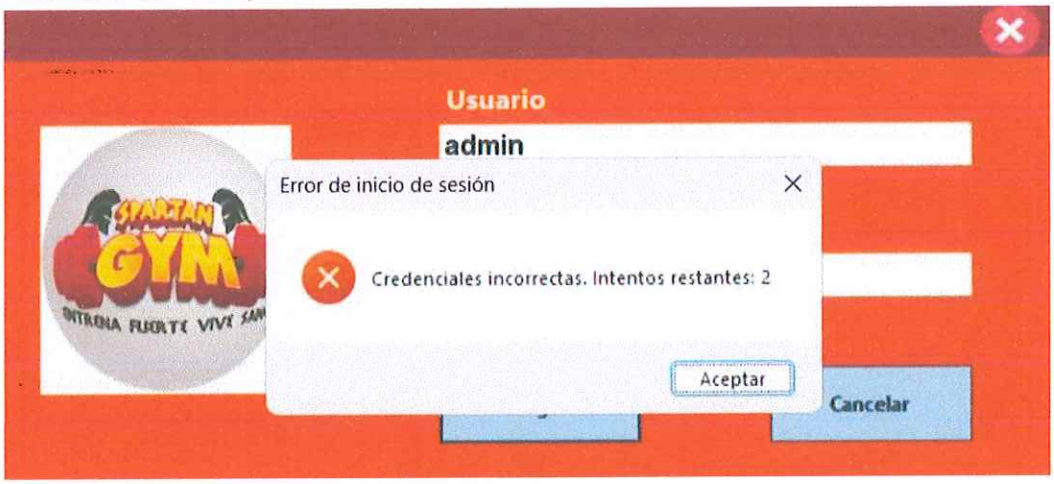


Figura 2. *Gráfico de rendimiento del módulo de Registro de Clientes, indicando el tiempo de respuesta y errores detectados.*

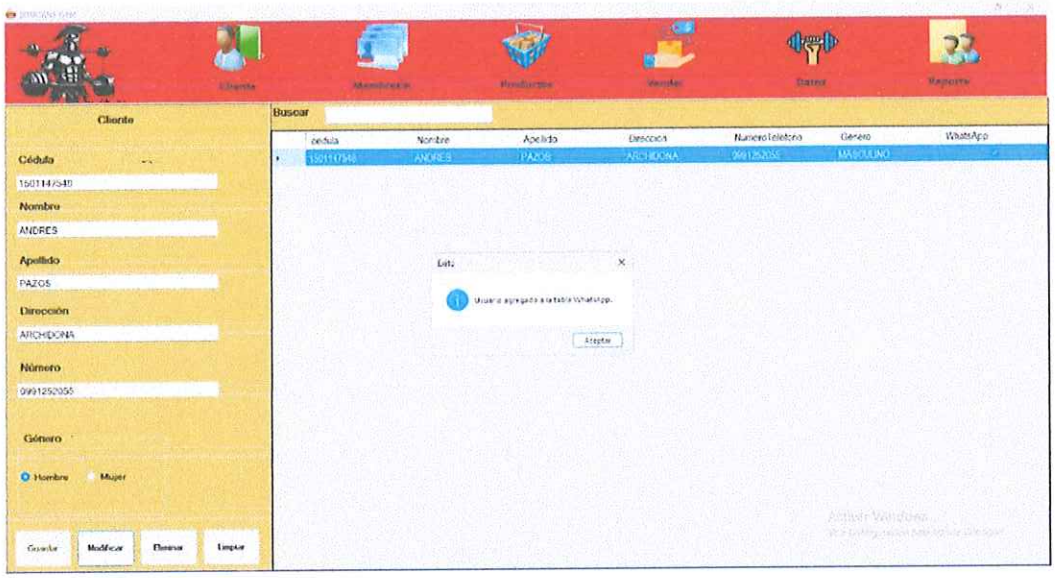


Figura 3. *Captura de pantalla de la prueba del módulo de Membresías, demostrando la integridad de las transacciones.*

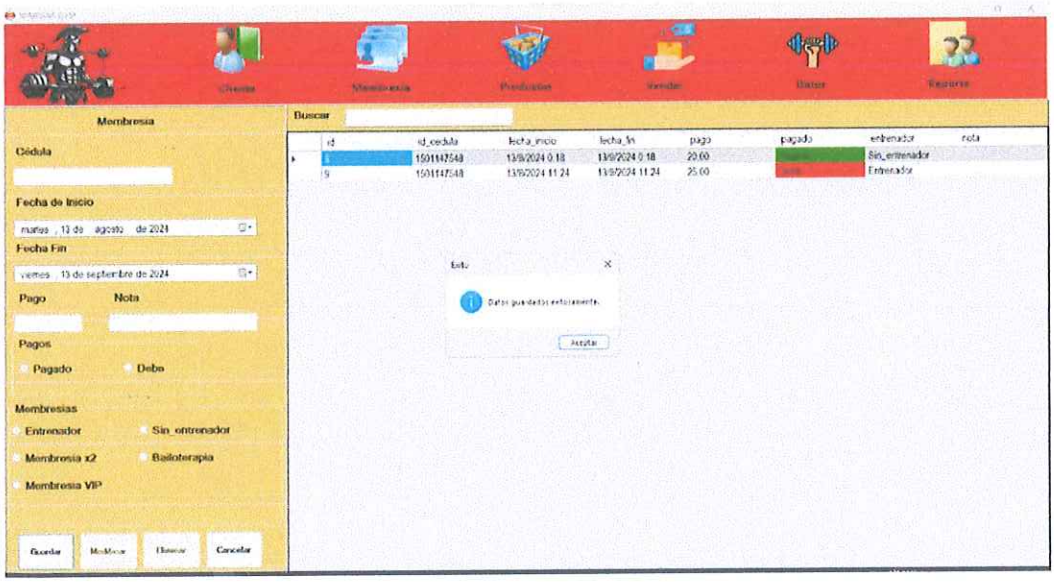


Figura 4. *Imagen del informe de prueba del módulo de Productos, destacando la precisión de los datos.*

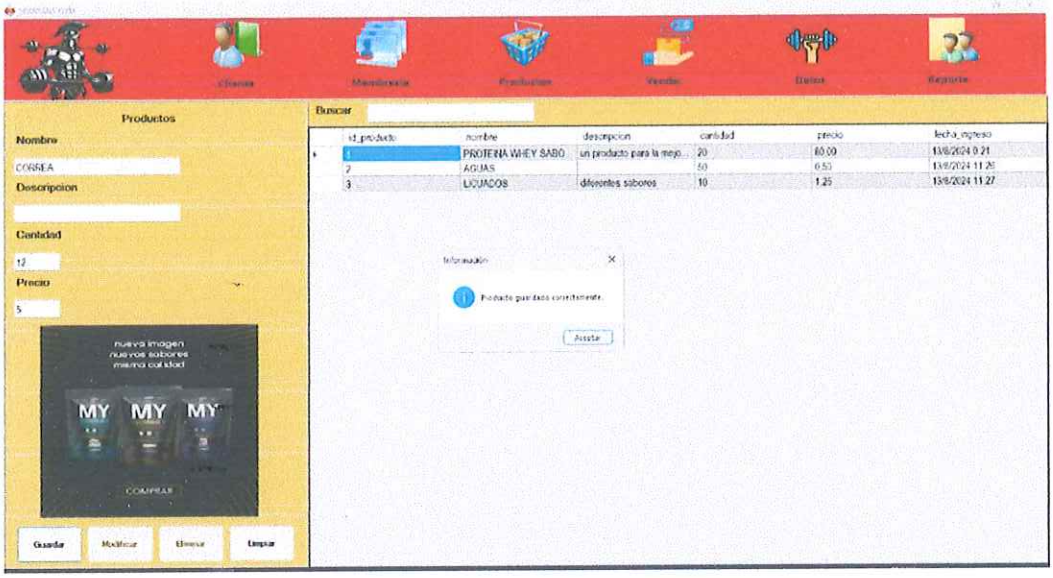
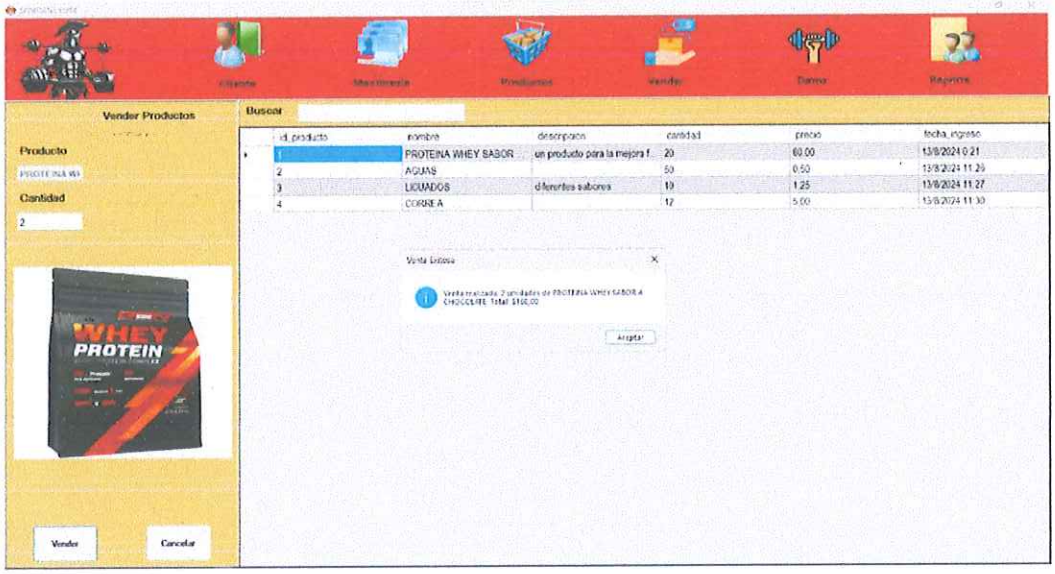


Figura 5. *Gráfico de usabilidad del módulo de Ventas, mostrando resultados de la evaluación de la interfaz de usuario.*



Fase de Mantenimiento: Después de la implementación del sistema, el propietario del gimnasio SPARTANS GYM se encargó de verificar ciertos errores que surgieron durante el uso del sistema. Los errores identificados fueron luego corregidos por el equipo de desarrollo. El propietario no realizó mejoras en la aplicación, y el foco de esta fase fue garantizar que los problemas reportados fueran solucionados de manera eficiente para asegurar el funcionamiento estable del sistema.

14 CONCLUSIONES

En conclusión, el proyecto ha avanzado de manera satisfactoria, cumpliendo con los objetivos específicos establecidos. En primer lugar, se realizó un análisis exhaustivo de las necesidades del gimnasio SPARTANS GYM mediante entrevistas y el estudio de aplicaciones existentes, lo que permitió definir los requerimientos esenciales para el sistema. El diseño de la interfaz de usuario se llevó a cabo utilizando Canva, asegurando una experiencia intuitiva y adaptada a las necesidades identificadas. La codificación de los componentes principales del sistema se realizó en Visual Basic, con la gestión de la base de datos mediante MySQL Workbench 8.0, garantizando un funcionamiento eficaz y una integración fluida de los módulos. Las pruebas de integración y aceptación del usuario final confirmaron que el sistema opera de acuerdo con los requisitos establecidos, asegurando precisión en los datos y funcionalidad en la interfaz. En la fase de mantenimiento, se gestionaron y corrigieron los errores detectados, asegurando la estabilidad del sistema. En conjunto, el proyecto ha logrado cumplir con todas las expectativas y requerimientos definidos, ofreciendo un sistema robusto y eficiente para la gestión del gimnasio.

15 RECOMENDACIONES

Implementar un plan de mantenimiento y actualización continua: Es crucial establecer un plan que incluya la integración de nuevas funcionalidades y mejoras basadas en la retroalimentación de los usuarios y los avances tecnológicos. Esto permitirá abordar áreas de mejora, mantener la relevancia del sistema en un entorno tecnológico en evolución y adaptarse a las necesidades cambiantes del gimnasio SPARTANS GYM.

Establecer un proceso regular para la revisión y actualización del sistema: Crear un calendario para revisar y actualizar el sistema periódicamente asegurará que se mantenga operativo y eficiente. Este proceso debe incluir pruebas regulares de seguridad y rendimiento, así como la incorporación de nuevas características o ajustes según las necesidades emergentes del gimnasio.

Ofrecer capacitación adicional al propietario y al personal: Proporcionar formación continua sobre el uso del sistema ayudará a garantizar una utilización más eficiente y efectiva. La capacitación debería incluir resolución de problemas, mejores prácticas y actualización sobre nuevas funcionalidades, mejorando así la experiencia del usuario y asegurando que el sistema cumpla con los objetivos del gimnasio a largo plazo.

Bibliografía

Gomez, L., & Pedro, F. (2018). Cómo implantar un SGSI según ISO/IEC 27001 y su aplicacíon de Esquema Nacional de Seguridad. Colombia: AENOR INTERNACIONAL, S.A.U.

Martínez, J. (2018). La importancia de la gestión de clientes en los negocios modernos. Editorial Gestión Empresarial.

García, L., & López, M. (2020). Sistemas de Información para la Gestión de Clientes. Tecnología y Negocios, 14(2), 45-56.

Fernández, P. (2019). Desarrollo de Software en Visual Basic. Revista de Tecnología en el Deporte, 10(1), 20-30.

Rodríguez, A. (2021). Desarrollo de Software: Fundamentos y Metodologías. Editorial Innovación y Tecnología.

Pérez, R. (2017). Bases de Datos Relacionales: Teoría y Práctica. Universidad Técnica de Quito.

Asamblea Nacional del Ecuador. (2021). Ley Orgánica de Protección de Datos Personales.

Álvarez, J. (2020). Seguridad Informática: Principios y Prácticas. Editorial Seguridad Digital.

López, M. (2018). Gestión de Relaciones con los Clientes (CRM). Revista de Administración y Negocios, 5(3), 33-44.

González, D. (2019). Desarrollo de Software: Fundamentos y Metodologías. Editorial TecnoCiencia.

Morales, S. (2018). Bases de Datos Relacionales: Conceptos y Aplicaciones. Editorial Informática Avanzada.

Hernández, J. (2020). Aplicaciones de Escritorio en Visual Basic. Universidad Tecnológica de los Andes.

ElCamino.dev. (2023). Cómo instalar Visual Studio 2019 - Guía Paso a Paso.

Guna Framework. (2023). UI/UX Controls and Components for Developers of Desktop, Reporting, Data Visualization.

ANEXO 1

ENTREVISTA REALIZADA AL PROPIETARIO

NOMBRES DEL PROPIETARIO: Jonathan Alexander Restrepo Haro

Entrevistador: Buenos días, gracias por su tiempo. Me gustaría comenzar preguntándole sobre la gestión actual de clientes en SPARTANS GYM. ¿Podría describir brevemente cómo manejan el registro y la administración de los datos de los clientes?

Propietario del Gimnasio: Buenos días. Actualmente, usamos un sistema manual para registrar a los clientes. Llevamos un libro de registros donde anotamos los detalles de cada nuevo cliente, como su nombre, dirección, número de teléfono, y el tipo de membresía que eligen. También utilizamos hojas de cálculo en Excel para llevar un seguimiento de los pagos y la asistencia de los clientes.

Entrevistador: Entiendo. ¿Cuáles son los principales desafíos que enfrentan con este sistema manual?

Propietario del Gimnasio: Uno de los mayores problemas es el tiempo que lleva registrar nuevos clientes y actualizar sus datos. Además, es fácil cometer errores cuando se introducen datos manualmente, y hemos tenido casos donde la información de los clientes se ha perdido o mezclado. También es difícil hacer un seguimiento preciso de las fechas de vencimiento de las membresías y los pagos pendientes.

Entrevistador: Parece ser bastante laborioso. ¿Cómo manejan las renovaciones de membresías y la asignación de entrenadores a los clientes?

Propietario del Gimnasio: Las renovaciones se hacen manualmente. Tenemos que revisar las hojas de cálculo para ver cuándo expiran las membresías y contactar a los clientes individualmente. Para la asignación de entrenadores, tratamos de mantener un registro separado, pero no siempre es consistente, lo que puede causar confusión.

Entrevistador: Entendido. ¿Qué tipo de mejoras esperaría ver con un nuevo sistema de gestión automatizado?

Propietario del Gimnasio: Nos gustaría tener un sistema que nos permita registrar y actualizar la información de los clientes de manera más rápida y precisa. Sería ideal tener alertas automáticas para las renovaciones de membresías y una forma más eficiente de gestionar los pagos. También nos ayudaría tener un registro centralizado de la asignación de entrenadores y la asistencia de los clientes. Además, sería excelente poder generar reportes fácilmente para analizar el rendimiento del gimnasio.

Entrevistador: Eso tiene mucho sentido. En cuanto a la venta de productos y servicios adicionales, ¿cómo se gestiona actualmente?

Propietario del Gimnasio: Para las ventas, utilizamos una caja registradora tradicional y llevamos un registro manual de los productos vendidos. Esto puede ser ineficiente y propenso a errores, especialmente cuando se trata de manejar el inventario. Sería beneficioso tener un sistema integrado que nos permita gestionar las ventas y el inventario de manera más precisa.

Entrevistador: Gracias por compartir esta información. Para terminar, ¿hay alguna funcionalidad específica que considere esencial en el nuevo sistema de gestión?

Propietario del Gimnasio: Sí, además de las funciones básicas de registro y gestión de clientes, nos gustaría tener un módulo para registrar las mediciones físicas de nuestros clientes, como peso, altura, y otras medidas corporales. Esto nos permitiría ofrecer un servicio más personalizado y hacer un seguimiento del progreso de los clientes de manera más eficiente.

Entrevistador: Perfecto. Agradezco mucho su tiempo y sus respuestas. Esta información será muy útil para diseñar un sistema que realmente satisfaga las necesidades de SPARTANS GYM.

Propietario del Gimnasio: De nada, gracias a usted. Esperamos con interés ver los resultados del nuevo sistema.

Entrevistas realizadas a 10 clientes del gimnasio **SPARTANS GYM**

¿Está interesado en un sistema que facilite el registro y actualización de sus datos personales?

- ☐ Sí
- ☐ No

¿Le gustaría recibir notificaciones automáticas sobre la renovación de su membresía?

- ☐ Sí
- ☐ No

¿Considera útil la idea de tener un seguimiento digital de sus progresos físicos (peso, altura, medidas corporales)?

- ❑ Sí
- ❑ No

¿Preferiría un sistema que gestione las ventas y el inventario de productos de manera más eficiente?

- ❑ Sí
- ❑ No

¿Cree que un sistema con reportes detallados sobre su asistencia y progreso podría mejorar su motivación y resultados?

- ❑ Sí
- ❑ No

¿Preferiría realizar pagos y renovaciones de membresías en línea, sin necesidad de visitar el gimnasio?

- ❑ Sí
- ❑ No

Gráfico 1

La gran mayoría de los clientes (90%) está interesada en un sistema que facilite el registro y actualización de sus datos personales. Esto indica una fuerte demanda por la digitalización y modernización de los procesos administrativos en SPARTANS GYM. Un pequeño porcentaje de los clientes (10%) no está interesado en la implementación de un sistema digital para el registro y actualización de datos. Esto podría deberse a una preferencia por métodos tradicionales o a una falta de familiaridad con la tecnología.

Gráfico 2

La mayoría de los clientes (60%) está interesada en recibir notificaciones automáticas sobre la renovación de su membresía. Esto sugiere que una parte significativa de los usuarios valora la conveniencia de las alertas automáticas para gestionar sus renovaciones. Un número considerable de clientes (40%) no está interesado en recibir notificaciones automáticas. Esta resistencia puede deberse a preferencias personales por métodos tradicionales de comunicación o preocupaciones sobre la frecuencia de las notificaciones.

Gráfico 3

El 100% de los encuestados considera útil la idea de tener un seguimiento digital de sus progresos físicos, lo cual indica un fuerte interés y aceptación hacia esta tecnología. Esto sugiere que los usuarios valoran la conveniencia y la capacidad de seguimiento detallado que ofrece un sistema digital.

Gráfico 4

El 70% de los encuestados indicó que no preferiría un sistema más eficiente de gestión de ventas e inventario. Esto sugiere que la mayoría está satisfecha con el sistema actual o no ve una necesidad inmediata de mejorar la eficiencia en este aspecto, es decir que solo un 30% prefiere seguir con el mismo método.

Gráfico 5

El 80% de los encuestados cree que un sistema con reportes detallados sobre su asistencia y progreso podría mejorar su motivación y resultados. Esto indica un fuerte interés y percepción positiva hacia la utilidad de la información detallada para impulsar la motivación y mejorar los resultados personales en objetivos de salud o fitness, mientras que el 20% de usuarios no consideran importante el registro de su progreso en el entrenamiento.

Gráfico 6

El 70% de los encuestados preferiría realizar pagos y renovaciones de membresías en línea, sin necesidad de visitar el gimnasio. Esto indica un fuerte interés y comodidad hacia la conveniencia

El 70% de los encuestados indicó que no preferiría un sistema más eficiente de gestión de ventas e inventario. Esto sugiere que la mayoría está satisfecha con el sistema actual o no ve una necesidad inmediata de mejorar la eficiencia en este aspecto, es decir que solo un 30% prefiere seguir con el mismo método.

Gráfico 5

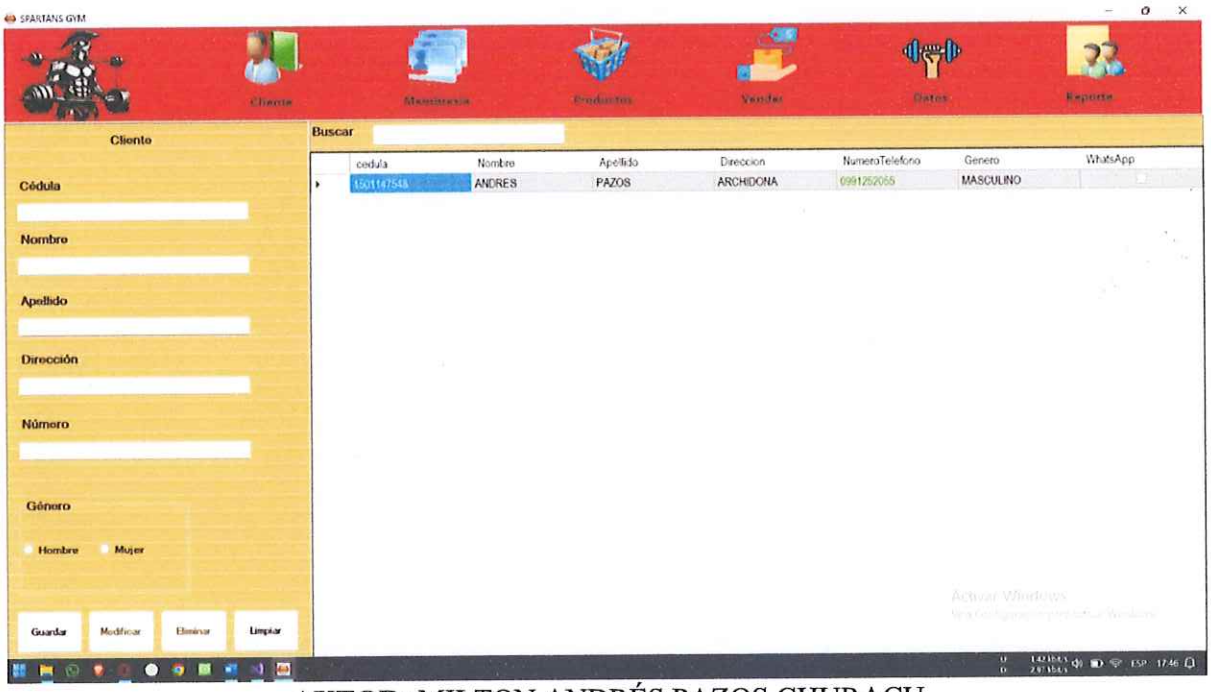
ANEXO 2

MANUAL DE USUARIO

INSTITUTO SUPERIOR TECNOLÓGICO TENA
Tecnología, Innovación y Desarrollo

CARRERA DE TECNOLOGÍA SUPERIOR EN

DESARROLLO DE SOFTWARE



AUTOR: MILTON ANDRÉS PAZOS CHURACU

Tena - Ecuador

2024-IS

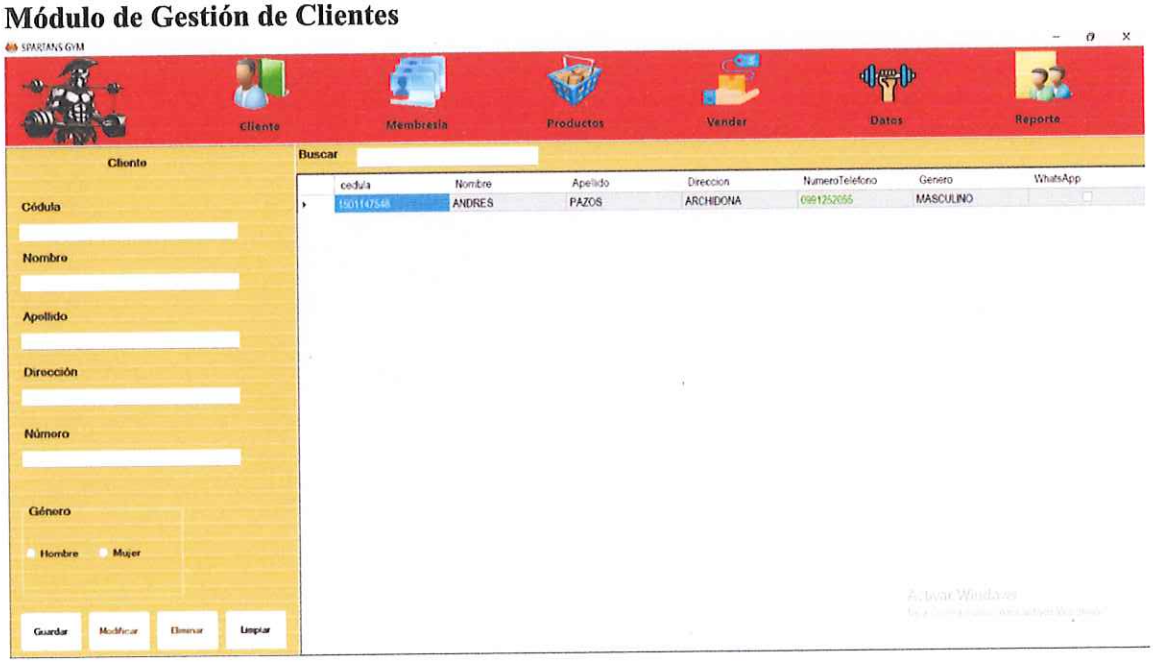
ÍNDICE

MANUAL DE USUARIO

Pantalla de Inicio de Sesión

En **Usuario** colocar "admin" porque con eso se podrá ingresar a la aplicación
En **Contraseña** colocar "admin" con eso validará el ingreso
Presionar **Login** para poder validar e ingresar

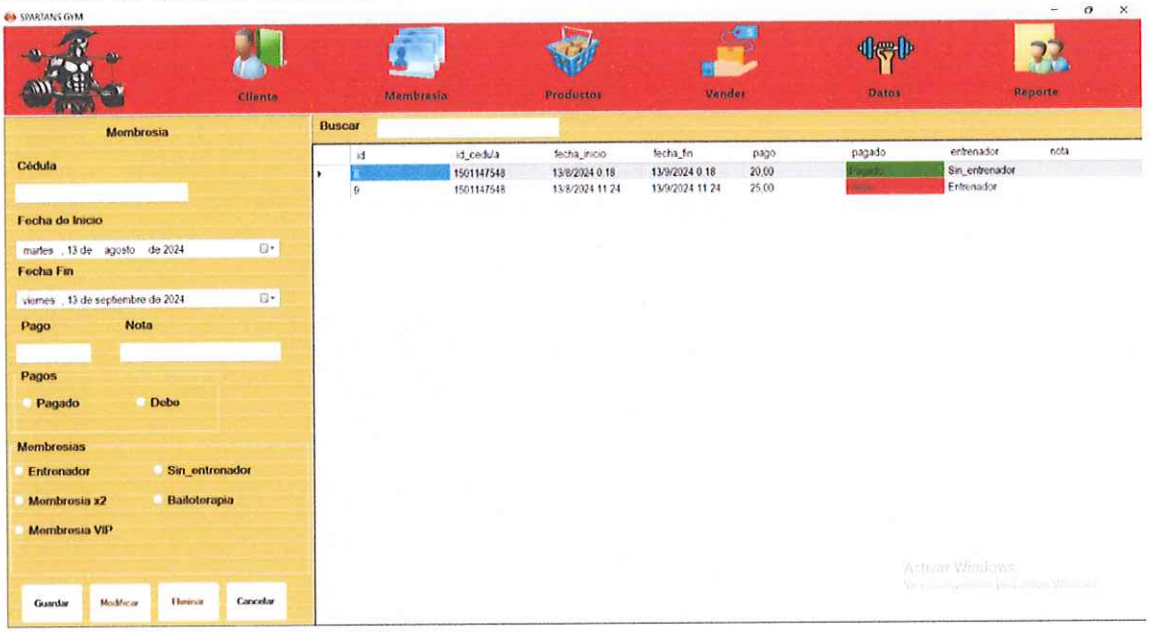
Módulo de Gestión de Clientes



Presionar la imagen de clientes en la parte superior una persona con un cuaderno atrás para el ingreso de clientes.
Cédula ingresar datos de 8 a 10 números 8 numeros

Nombre ingresar el primer nombre
Apellido ingresar el segundo apellido
Dirección ingresar la dirección de donde vive
Número ingresar el número de celular
Género seleccionar si es hombre o mujer
Botón guardar para almacenar los datos insertados en la base de datos.
Botón modificar cuando se seleccionar una fila de un cliente, inserta los datos que se requiera modificar y si desea marcar con un visto para ver si este agregado a WhatsApp y presionar modificar para guardar los datos actualizados.
Botón eliminar se selecciona la fila del cliente que desea eliminar y se borra todo relacionado con ese cliente.
Botón limpiar sirve para borrar los datos que haya en cédula, nombre, apellido, dirección, número y género.
Buscar para encontrar los clientes más rápido

Módulo de Gestión de Membresías



Presionar en la imagen membresía para ingresar al modulo
Cédula ingresar la cedula del cliente registrado.
Fecha de inicio seleccionar la fecha de ingreso por defecto viene la fecha actual.
Fecha fin seleccionar la fecha cuando finaliza la suscripción por defecto se coloca un mes de acuerdo con la fecha de inicio.
Pago ingresar el valor que se pagó.
Nota ingresar algo importante o lo que se debe o lo que se canceló.
Pagos que contiene pagado y debe seleccionar una opción
Membresías contiene sin entrenador, entrenador, membresía x2, membresía VIP y bailoterapia seleccionar una opción.

Botón guardar para almacenar los datos insertados en la base de datos.
Botón modificar cuando se seleccionar una fila de membresía, inserta los datos que se requiera modificar y presionar modificar para guardar los datos actualizados.
Botón eliminar se selecciona la fila de membresía que desea eliminar y se borrará.
Botón limpiar sirve para borrar los datos que estén.
Buscar para encontrar las membresías más rápido

Módulo de Ventas

Presionar en la imagen de la mano con un sobre se abrirá el módulo de ventas.
Productos ingresar el nombre del producto para eso tendrá que seleccionar una fila de la tabla y se coloca automáticamente el producto.
Cantidad ingresar una cantidad que se adquiere para la compra.
Botón Vender para aceptar la compra.
Botón Cancelar para borrar los datos.
Buscar los productos mas rápido.

Módulo de Productos

Presionar la imagen de la canasta para ingresar al módulo de productos.

Nombre ingresar el nombre del producto que dé sea guardar.

Descripción colocar una breve explicación sobre el producto.

Cantidad colocar la cantidad que dispone en stock.

Precio colocar el precio que este establecido.

Botón guardar para almacenar los datos insertados en la base de datos.

Botón modificar cuando se seleccionar una fila de productos, inserta los datos que se requiera modificar y presionar modificar para guardar los datos actualizados.

Botón eliminar se selecciona la fila de productos que desea eliminar y se borrará.
Botón limpiar sirve para borrar los datos que estén.
Buscar para encontrar los productos más rápido.

Módulo de Reportes

Presionar la imagen de dos personas y una carpeta atrás en la parte derecha superior y ingresa al módulo de reportes.

Botón datos le aparecerá un listado de todos datos que se hayan insertado.

Botón clientes le aparecerá un listado de todos los clientes registrados.

Boto membresías le aparecerá un listado de todas las membresías registrados.

Botón ventas le aparecerá un listado de todas las ventas realizadas.

Botón productos le aparecerá un listado de todos los productos registrados.

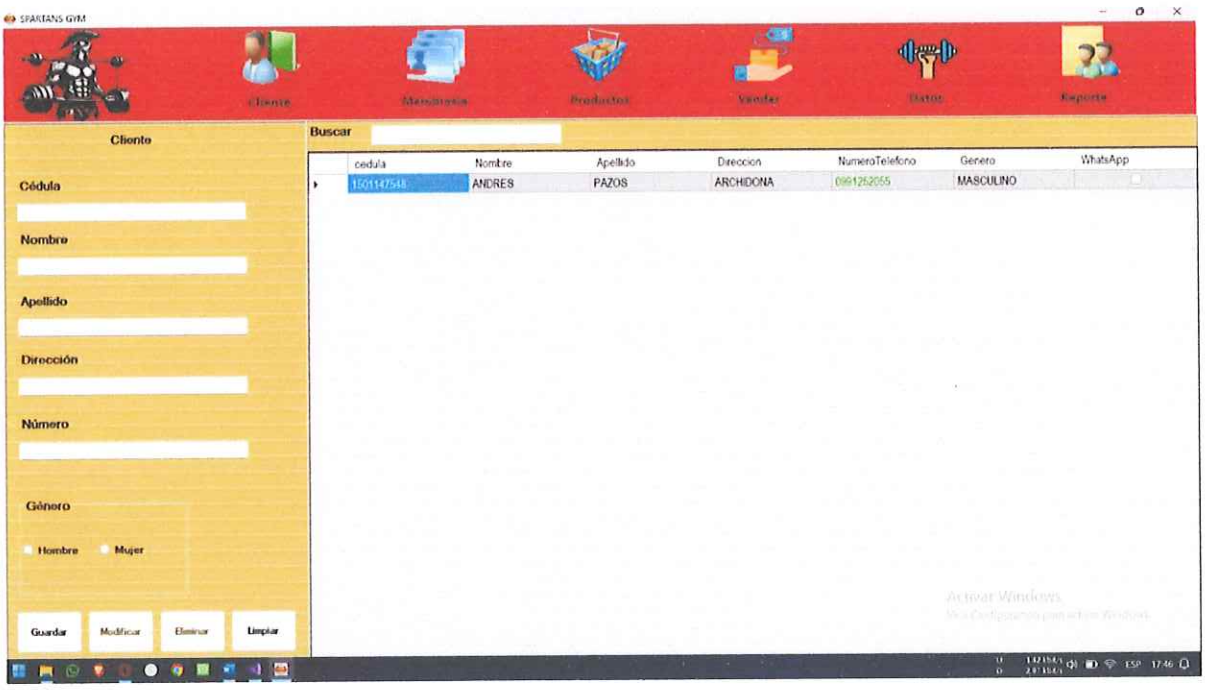
ANEXO 3

MANUAL TÉCNICO DEL CÓDIGO

INSTITUTO SUPERIOR TECNOLÓGICO TENA
Tecnología, Innovación y Desarrollo

CARRERA DE TECNOLOGÍA SUPERIOR EN

DESARROLLO DE SOFTWARE



AUTOR: MILTON ANDRÉS PAZOS CHURACU

Tena - Ecuador

2024-IS

ÍNDICE

Manual Técnico para el Sistema de Login

Código del Sistema de Login

Importaciones

vb

```
Imports MySql.Data.MySqlClient
```

Descripción: Importa la biblioteca MySql.Data.MySqlClient, que permite conectar y realizar operaciones en una base de datos MySQL desde la aplicación.

Definiciones de Clase y Variables

vb

```
Public Class login
    Dim connectionString As String = "server=localhost;database=gym;user id=root;password=root;"
    Dim intentosRestantes As Integer = 3
```

connectionString: Cadena de conexión a la base de datos MySQL, que especifica el servidor, la base de datos, el ID de usuario y la contraseña.

intentosRestantes: Número de intentos de inicio de sesión permitidos antes de cerrar la aplicación.

Métodos de Eventos

Cerrar el Formulario

vb

```
Private Sub Button2_Click(sender As Object, e As EventArgs) Handles btncancelar.Click
    Me.Close()
End Sub
```

Descripción: Evento que se activa cuando se hace clic en el botón btncancelar. Cierra el formulario de inicio de sesión.

Cerrar el Formulario al Hacer Clic en la Imagen

vb

```
Private Sub PictureBox2_Click(sender As Object, e As EventArgs) Handles pinturacerrar.Click
    Me.Close()
End Sub
```

Descripción: Evento que se activa cuando se hace clic en la imagen pinturacerrar. También cierra el formulario de inicio de sesión.

Iniciar Sesión

vb

```
Private Sub btnlogin_Click(sender As Object, e As EventArgs) Handles btnlogin.Click
    IniciarSesion()
End Sub
```

Descripción: Evento que se activa cuando se hace clic en el botón btnlogin. Llama al método IniciarSesion para procesar el inicio de sesión.

Manejo de Presión de Teclas en el Campo de Contraseña

vb

```
Private Sub txtcontrasenia_KeyPress(sender As Object, e As KeyPressEventArgs) Handles txtcontrasenia.KeyPress
    If e.KeyChar = Convert.ToChar(Keys.Enter) Then
        IniciarSesion()
    End If
End Sub
```

Descripción: Evento que se activa cuando se presiona una tecla en el campo txtcontrasenia. Si la tecla presionada es Enter, llama al método IniciarSesion.

Métodos de Función

Iniciar Sesión

vb

```
Private Sub IniciarSesion()
    Dim usuario As String = txtusuario.Text.Trim()
    Dim contrasenia As String = txtcontrasenia.Text.Trim()

    If ValidarCredenciales(usuario, contrasenia) Then
        ' Credenciales válidas, abrir la aplicación principal o el formulario deseado
        MessageBox.Show("Inicio de sesión exitoso", "Éxito", MessageBoxButtons.OK, MessageBoxIcon.Information)

        ' Instanciar y mostrar el formulario principal
        Dim formPrincipal As New formprincipal()
        Me.Hide() ' Ocultar el formulario de inicio de sesión
```

```
            formPrincipal.ShowDialog()

            ' Cerrar la aplicación cuando se cierre el formulario principal
            Me.Close()
        Else
            ' Credenciales inválidas, decrementar intentos restantes
            intentosRestantes -= 1

            If intentosRestantes > 0 Then
                MessageBox.Show($"Credenciales incorrectas. Intentos restantes: {intentosRestantes}", "Error de inicio de sesión", MessageBoxButtons.OK, MessageBoxIcon.Error)
            Else
                ' Agotados los intentos, cerrar la aplicación
                MessageBox.Show("Demasiados intentos fallidos. La aplicación se cerrará.", "Error de inicio de sesión", MessageBoxButtons.OK, MessageBoxIcon.Error)
                Me.Close()
            End If
```

End If

End Sub

Descripción: Maneja el proceso de inicio de sesión:

Obtiene el nombre de usuario y la contraseña de los campos de texto.

Llama a ValidarCredenciales para verificar si las credenciales son correctas.

Muestra un mensaje de éxito si las credenciales son válidas, oculta el formulario de inicio de sesión y muestra el formulario principal.

Si las credenciales son incorrectas, decrementa el número de intentos restantes y muestra un mensaje de error. Si se agotan los intentos, cierra la aplicación.

Validar Credenciales

vb

```
Private Function ValidarCredenciales(usuario As String, contrasenia As String) As Boolean
    Try
        Using conn As New MySqlConnection(connectionString)
            conn.Open()
```

```
            Dim query As String = "SELECT COUNT(*) FROM administrador WHERE
usuario = @usuario AND contrasenia = @contrasenia"
            Using command As New MySqlCommand(query, conn)
                command.Parameters.AddWithValue("@usuario", usuario)
                command.Parameters.AddWithValue("@contrasenia", contrasenia)

                Dim count As Integer = Convert.ToInt32(command.ExecuteScalar())

                Return count > 0
            End Using
        End Using
    Catch ex As Exception
        MessageBox.Show("Error de conexión a la base de datos: " & ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
        Return False
    End Try
End Function
```

Descripción: Verifica las credenciales del usuario en la base de datos:

Establece una conexión a la base de datos utilizando MySqlConnection.

Ejecuta una consulta SQL para contar el número de registros en la tabla administrador que coinciden con el nombre de usuario y la contraseña proporcionados.

Retorna True si el número de registros es mayor que 0, indicando credenciales válidas, y False en caso contrario.

Si ocurre un error de conexión, muestra un mensaje de error y retorna False.

16 Manual Técnico para el Formulario Principal

Código del Formulario Principal

Definiciones de Clase y Variables

vb

```
Public Class formprincipal

    Private formMembresia As membresia

    Private nuevoFormulario As Form
```

formMembresia: Variable para almacenar una instancia del formulario membresia.

nuevoFormulario: Variable para almacenar una instancia de un formulario genérico.

Eventos del Formulario

Carga del Formulario

vb

```
Private Sub formprincipal_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    My.Settings.AlertaMostrada = False
    My.Settings.Save()
End Sub
```

Descripción: Evento que se activa cuando se carga el formulario principal. Inicializa la configuración AlertaMostrada a False y guarda los cambios en las configuraciones del usuario.

Clic en PictureBox1

vb

```
Private Sub PictureBox1_Click(sender As Object, e As EventArgs) Handles PictureBox1.Click
    CerrarFormularioAnteriorEnPanel()
    Dim formUsuarios As New formusuarios()
    formUsuarios.TopLevel = False
    Panel2.Controls.Add(formUsuarios)
```

```
    formUsuarios.Dock = DockStyle.Fill
    formUsuarios.Show()
End Sub
```

Descripción: Evento que se activa cuando se hace clic en PictureBox1. Cierra cualquier formulario anterior en Panel2, crea una nueva instancia de formusuarios, la agrega al panel, la configura para que se ajuste al tamaño del panel y la muestra.

Clic en PictureBox3

vb

```
Private Sub PictureBox3_Click(sender As Object, e As EventArgs) Handles PictureBox3.Click
    CerrarFormularioAnteriorEnPanel()
    Dim formproductos As New FormProductos()
    formproductos.TopLevel = False
    Panel2.Controls.Add(formproductos)
    formproductos.Dock = DockStyle.Fill
    formproductos.Show()
End Sub
```

Descripción: Evento que se activa cuando se hace clic en PictureBox3. Similar al evento anterior, cierra el formulario anterior, crea una instancia de FormProductos, la agrega al panel y la muestra.

Clic en PictureBox4

vb

```
Private Sub PictureBox4_Click(sender As Object, e As EventArgs) Handles PictureBox4.Click
    CerrarFormularioAnteriorEnPanel()
    Dim formMembresia As New membresia()
    formMembresia.TopLevel = False
    Panel2.Controls.Add(formMembresia)
    formMembresia.Dock = DockStyle.Fill
    formMembresia.Show()
End Sub
```

Descripción: Evento que se activa cuando se hace clic en PictureBox4. Cierra el formulario anterior, crea una instancia del formulario membresia, la agrega al panel y la muestra.

Paint del Panel2

vb

```
Private Sub Panel2_Paint(sender As Object, e As PaintEventArgs) Handles Panel2.Paint
    MostrarImagenEnPanel()
End Sub
```

Descripción: Evento que se activa cuando se pinta Panel2. Llama al método MostrarImagenEnPanel para dibujar una imagen en el panel.

Métodos del Formulario

Mostrar Imagen en Panel

vb

```
Private Sub MostrarImagenEnPanel()
    Dim imagenRecursos As Image = GYMESPARTA.My.Resources.espartano
    Dim graphics As Graphics = Panel2.CreateGraphics()
    Dim rectDestino As New Rectangle(0, 0, Panel2.Width, Panel2.Height)
    graphics.DrawImage(imagenRecursos, rectDestino)
    graphics.Dispose()
```

End Sub

Descripción: Dibuja una imagen en Panel2. Carga la imagen desde los recursos de la aplicación, crea un objeto Graphics para el panel, define un rectángulo que cubre todo el panel y dibuja la imagen en ese rectángulo. Finalmente, libera los recursos del objeto Graphics.

Cerrar Formulario Anterior en Panel

vb

```
Private Sub CerrarFormularioAnteriorEnPanel()
    ' Verificar si hay un formulario anterior en Panel2 y cerrarlo si es necesario
    If Panel2.Controls.Count > 0 Then
        Dim formularioAnterior As Form = TryCast(Panel2.Controls(0), Form)
        If Not IsNothing(formularioAnterior) AndAlso Not formularioAnterior.IsDisposed Then
            formularioAnterior.Close()
        End If
    End If
End Sub
```

Descripción: Verifica si hay algún formulario en Panel2 y, si es así, lo cierra. Primero, comprueba si el panel tiene controles. Si es así, intenta convertir el primer control en un formulario y, si el formulario no está desechado, lo cierra.

Clic en PictureBox5

vb

```
Private Sub PictureBox5_Click(sender As Object, e As EventArgs) Handles PictureBox5.Click
    CerrarFormularioAnteriorEnPanel()
    Dim reportesfrom As New reportesfrom()
    reportesfrom.TopLevel = False
    Panel2.Controls.Add(reportesfrom)
    reportesfrom.Dock = DockStyle.Fill
    reportesfrom.Show()
End Sub
```

Descripción: Evento que se activa cuando se hace clic en PictureBox5. Similar a los eventos anteriores, cierra el formulario anterior, crea una instancia del formulario reportesfrom, la agrega al panel y la muestra.

Clic en PictureBox7

vb

```
Private Sub PictureBox7_Click(sender As Object, e As EventArgs) Handles PictureBox7.Click
    CerrarFormularioAnteriorEnPanel()
    Dim venderproductos As New venderproductos()
    venderproductos.TopLevel = False
    Panel2.Controls.Add(venderproductos)
    venderproductos.Dock = DockStyle.Fill
    venderproductos.Show()
End Sub
```

Descripción: Evento que se activa cuando se hace clic en PictureBox7. Cierra el formulario anterior, crea una instancia del formulario venderproductos, la agrega al panel y la muestra.

Clic en PictureBox6

vb

```
Private Sub PictureBox6_Click(sender As Object, e As EventArgs) Handles
PictureBox6.Click

    CerrarFormularioAnteriorEnPanel()

    Dim datos As New datos()

    datos.TopLevel = False

    Panel2.Controls.Add(datos)

    datos.Dock = DockStyle.Fill

    datos.Show()

End Sub
```

Descripción: Evento que se activa cuando se hace clic en PictureBox6. Cierra el formulario anterior, crea una instancia del formulario datos, la agrega al panel y la muestra.

17 Explicación del Código Usuarios

Importación de Librerías y Declaración de Variables

```
Imports MySql.Data.MySqlClient



Public Class formusuarios
```

Dim connectionString As String = "server=localhost;database=gym;user id=root;password=root;"

Dim usuariosBindingSource As New BindingSource()

Se importa la biblioteca MySql.Data.MySqlClient para trabajar con MySQL.

Se declara una variable connectionString con la cadena de conexión a la base de datos.

Se crea una instancia de BindingSource para gestionar el enlace de datos en el DataGridView.

Método CargarDatosUsuarios

```
Private Sub CargarDatosUsuarios()
    Try
        Using connection As New MySqlConnection(connectionString)
            connection.Open()
            Dim query As String = "SELECT * FROM usuarios"
            Dim adapter As New MySqlDataAdapter(query, connection)
            Dim dataSet As New DataSet()
            adapter.Fill(dataSet, "UsuariosData")

            Dim usuariosDataTable As DataTable = dataSet.Tables("UsuariosData")
```

```
        ' Agregar una nueva columna de CheckBox a la DataTable
        Dim checkBoxColumn As New DataColumn("WhatsApp", GetType(Boolean))
        checkBoxColumn.DefaultValue = False
        usuariosDataTable.Columns.Add(checkBoxColumn)

        ' Convertir los datos a mayúsculas
        For Each row As DataRow In usuariosDataTable.Rows
            For Each col As DataColumn In usuariosDataTable.Columns
                row(col) = row(col).ToString().ToUpper()
            Next
        Next

        usuariosBindingSource.DataSource = usuariosDataTable
        dgvusuarios.DataSource = usuariosBindingSource
    End Using
Catch ex As Exception
```

MessageBox.Show("Error al cargar los datos de usuarios. Detalles: " & ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

End Try

End Sub

Conecta a la base de datos, ejecuta una consulta para seleccionar todos los usuarios y carga los datos en un DataSet.

Agrega una columna WhatsApp al DataTable para marcar si el usuario está en WhatsApp.

Convierte todos los datos en el DataTable a mayúsculas.

Enlaza el DataTable al BindingSource y el DataGridView para mostrar los datos.

Método btncancelar_Click

vb

Private Sub btncancelar_Click(sender As Object, e As EventArgs) Handles btncancelar.Click

LimpiarControles()

HabilitarEdicion(False)

End Sub

Limpia los controles del formulario y deshabilita el modo de edición cuando se hace clic en el botón de cancelar.

Método dgvusuarios_CellClick

vb

```
Private Sub dgvusuarios_CellClick(sender As Object, e As DataGridViewCellEventArgs) Handles dgvusuarios.CellClick
    If e.RowIndex >= 0 Then
        Dim cedula As String = dgvusuarios.Rows(e.RowIndex).Cells("cedula").Value.ToString()
        CargarDatosSeleccionados(cedula)
    End If
    btnmodificar.Enabled = True
    btnguardar.Enabled = False
    btneliminar.Enabled = True
End Sub
```

Carga los datos del usuario seleccionado en los controles cuando se hace clic en una fila del DataGridView.

Habilita y deshabilita los botones según la acción requerida (modificar, guardar, eliminar).

Método CargarDatosSeleccionados

vb

```
Private Sub CargarDatosSeleccionados(cedula As String)
    Try
        Using connection As New MySqlConnection(connectionString)
            connection.Open()
            Dim query As String = "SELECT * FROM usuarios WHERE cedula = @cedula"
            Using command As New MySqlCommand(query, connection)
                command.Parameters.AddWithValue("@cedula", cedula)

                Using reader As MySqlDataReader = command.ExecuteReader()
                    If reader.Read() Then
                        txtcedula.Text = reader("cedula").ToString()
                        txtnombre.Text = reader("nombre").ToString()
                        txtapellido.Text = reader("apellido").ToString()
```

```
                txtdireccion.Text = reader("direccion").ToString()
                txtnumero.Text = reader("numerotelefono").ToString()

                Dim genero As String = reader("genero").ToString()
                If genero = "Masculino" Then
                    RadioButton1.Checked = True
                ElseIf genero = "Femenino" Then
                    RadioButton2.Checked = True
                End If
            End If
        End Using
    End Using
End Using
Catch ex As Exception
    MessageBox.Show("Error al cargar los datos seleccionados. Detalles: " & ex.Message,
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
End Try
```

End Sub

Obtiene y muestra los datos del usuario seleccionado en los controles del formulario.

Método CamposEstanCompletos

vb

```
Private Function CamposEstanCompletos() As Boolean
    If String.IsNullOrEmpty(txtcedula.Text) OrElse
       String.IsNullOrEmpty(txtnombre.Text) OrElse
       String.IsNullOrEmpty(txtapellido.Text) OrElse
       String.IsNullOrEmpty(txtdireccion.Text) OrElse
       String.IsNullOrEmpty(txtnumero.Text) OrElse
       (Not RadioButton1.Checked AndAlso Not RadioButton2.Checked) Then
        Return False
    End If
    Return True
End Function
```

Verifica si todos los campos obligatorios del formulario están completos.

Método btnguardar_Click

vb

```
Private Sub btnguardar_Click(sender As Object, e As EventArgs) Handles btnguardar.Click
    Try
        If Not CamposEstanCompletos() Then
            MessageBox.Show("Por favor complete todos los campos obligatorios.", "Campos Incompletos", MessageBoxButtons.OK, MessageBoxIcon.Warning)
            Return
        End If

        Dim cedula As String = txtcedula.Text
        Dim nombre As String = txtnombre.Text
        Dim apellido As String = txtapellido.Text
        Dim direccion As String = txtdireccion.Text
        Dim numeroTelefono As String = txtnumero.Text
        Dim genero As String = If(RadioButton1.Checked, "Masculino", "Femenino")
```

```
            Using connection As New MySqlConnection(connectionString)
                connection.Open()
                Dim query As String = "INSERT INTO usuarios (cedula, Nombre, Apellido,
Direccion, NumeroTelefono, Genero) VALUES (@cedula, @nombre, @apellido, @direccion,
@numeroTelefono, @genero)"
                Using command As New MySqlCommand(query, connection)
                    command.Parameters.AddWithValue("@cedula", cedula)
                    command.Parameters.AddWithValue("@nombre", nombre)
                    command.Parameters.AddWithValue("@apellido", apellido)
                    command.Parameters.AddWithValue("@direccion", direccion)
                    command.Parameters.AddWithValue("@numeroTelefono", numeroTelefono)
                    command.Parameters.AddWithValue("@genero", genero)
                    command.ExecuteNonQuery()
                End Using
            End Using

            LimpiarControles()
```

CargarDatosUsuarios()

MessageBox.Show("Datos guardados exitosamente.", "Éxito", MessageBoxButtons.OK, MessageBoxIcon.Information)

Catch ex As Exception

MessageBox.Show("Error al guardar los datos. Detalles: " & ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

End Try

End Sub

Valida que todos los campos estén completos antes de insertar un nuevo registro en la base de datos.

Limpia los controles, actualiza la vista de datos y muestra un mensaje de éxito.

Método btnmodificar_Click

vb

Private Sub btnmodificar_Click(sender As Object, e As EventArgs) Handles btnmodificar.Click

Try

If Not CamposEstanCompletos() Then

```
            MessageBox.Show("Por favor complete todos los campos obligatorios.", "Campos
Incompletos", MessageBoxButtons.OK, MessageBoxIcon.Warning)
            Return
        End If

        Dim cedula As String =
dgvusuarios.SelectedRows(0).Cells("cedula").Value.ToString()
        Dim nuevaCedula As String = txtcedula.Text
        Dim nombre As String = txtnombre.Text
        Dim apellido As String = txtapellido.Text
        Dim direccion As String = txtdireccion.Text
        Dim numeroTelefono As String = txtnumero.Text
        Dim genero As String = If(RadioButton1.Checked, "Masculino", "Femenino")

        Using connection As New MySqlConnection(connectionString)
            connection.Open()

            Dim queryDisableFK As String = "SET FOREIGN_KEY_CHECKS=0"
```

```
            Using commandDisableFK As New MySqlCommand(queryDisableFK, connection)
                commandDisableFK.ExecuteNonQuery()
            End Using

            Dim queryWhatsapp As String = "UPDATE whatsapp SET id_usuario =
@nuevaCedula WHERE id_usuario = @cedula"
            Using commandWhatsapp As New MySqlCommand(queryWhatsapp, connection)
                commandWhatsapp.Parameters.AddWithValue("@cedula", cedula)
                commandWhatsapp.Parameters.AddWithValue("@nuevaCedula", nuevaCedula)
                commandWhatsapp.ExecuteNonQuery()
            End Using

            Dim queryUpdate As String = "UPDATE usuarios SET cedula = @nuevaCedula,
Nombre = @nombre, Apellido = @apellido, Direccion = @direccion, NumeroTelefono =
@numeroTelefono, Genero = @genero WHERE cedula = @cedula"
            Using command As New MySqlCommand(queryUpdate, connection)
                command.Parameters.AddWithValue("@cedula", cedula)
                command.Parameters.AddWithValue("@nuevaCedula", nuevaCedula)
```

```
            command.Parameters.AddWithValue("@nombre", nombre)
            command.Parameters.AddWithValue("@apellido", apellido)
            command.Parameters.AddWithValue("@direccion", direccion)
            command.Parameters.AddWithValue("@numeroTelefono", numeroTelefono)
            command.Parameters.AddWithValue("@genero", genero)
            command.ExecuteNonQuery()
        End Using

        Dim queryEnableFK As String = "SET FOREIGN_KEY_CHECKS=1"
        Using commandEnableFK As New MySqlCommand(queryEnableFK, connection)
            commandEnableFK.ExecuteNonQuery()
        End Using
    End Using

    LimpiarControles()
    CargarDatosUsuarios()
```

MessageBox.Show("Datos modificados exitosamente.", "Éxito", MessageBoxButtons.OK, MessageBoxIcon.Information)

Catch ex As Exception

MessageBox.Show("Error al modificar los datos. Detalles: " & ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

End Try

End Sub

Valida los campos, obtiene el identificador del usuario seleccionado y actualiza los datos del usuario en la base de datos.

Maneja las restricciones de clave foránea y actualiza la vista de datos.

Método btneliminar_Click

Private Sub btneliminar_Click(sender As Object, e As EventArgs) Handles btneliminar.Click

Try

Dim cedula As String = dgvusuarios.SelectedRows(0).Cells("cedula").Value.ToString()

Using connection As New MySqlConnection(connectionString)

```
            connection.Open()

            Dim queryDisableFK As String = "SET FOREIGN_KEY_CHECKS=0"
            Using commandDisableFK As New MySqlCommand(queryDisableFK, connection)
                commandDisableFK.ExecuteNonQuery()
            End Using

            Dim queryDeleteWhatsapp As String = "DELETE FROM whatsapp WHERE
id_usuario = @cedula"
            Using commandDeleteWhatsapp As New MySqlCommand(queryDeleteWhatsapp,
connection)
                commandDeleteWhatsapp.Parameters.AddWithValue("@cedula", cedula)
                commandDeleteWhatsapp.ExecuteNonQuery()
            End Using

            Dim queryDelete As String = "DELETE FROM usuarios WHERE cedula =
@cedula"
            Using commandDelete As New MySqlCommand(queryDelete, connection)
```

```
                commandDelete.Parameters.AddWithValue("@cedula", cedula)
                commandDelete.ExecuteNonQuery()
            End Using

            Dim queryEnableFK As String = "SET FOREIGN_KEY_CHECKS=1"
            Using commandEnableFK As New MySqlCommand(queryEnableFK, connection)
                commandEnableFK.ExecuteNonQuery()
            End Using
        End Using

        LimpiarControles()
        CargarDatosUsuarios()
        MessageBox.Show("Datos eliminados exitosamente.", "Éxito",
MessageBoxButtons.OK, MessageBoxIcon.Information)
    Catch ex As Exception
        MessageBox.Show("Error al eliminar los datos. Detalles: " & ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
    End Try
```

End Sub

Elimina el usuario seleccionado de la base de datos, maneja las restricciones de clave foránea y actualiza la vista de datos.

Método LimpiarControles

```
Private Sub LimpiarControles()
    txtcedula.Clear()
    txtnombre.Clear()
    txtapellido.Clear()
    txtdireccion.Clear()
    txtnumero.Clear()
    RadioButton1.Checked = False
    RadioButton2.Checked = False
End Sub
```

Limpia todos los campos del formulario.

Método HabilitarEdicion

```
Private Sub HabilitarEdicion(habilitar As Boolean)
    btnguardar.Enabled = Not habilitar
```

```
        btnmodificar.Enabled = habilitar
        btneliminar.Enabled = habilitar
    End Sub
```

Habilita o deshabilita los botones de guardar, modificar y eliminar según el estado de edición.

Método BuscarUsuario

```
    Private Sub BuscarUsuario(terminoBusqueda As String)
        Try
            Dim usuariosDataTable As DataTable

            Using connection As New MySqlConnection(connectionString)
                connection.Open()
                ' Consulta SQL para buscar por nombre, apellido o cédula en la tabla usuarios
                Dim query As String = "SELECT * FROM usuarios WHERE CONCAT(Nombre, '
', Apellido, ' ', cedula) LIKE @terminoBusqueda"

                Using adapter As New MySqlDataAdapter(query, connection)
```

```
                adapter.SelectCommand.Parameters.AddWithValue("@terminoBusqueda", "%"
& terminoBusqueda & "%")

                Dim dataSet As New DataSet()
                adapter.Fill(dataSet, "UsuarioData")
                dgvusuarios.DataSource = dataSet.Tables("UsuarioData")

                ' Verificar y mostrar la columna "WhatsApp" si está presente en el DataGridView
                If dgvusuarios.Columns.Contains("WhatsApp") AndAlso
dgvusuarios.Columns("WhatsApp") IsNot Nothing Then
                    dgvusuarios.Columns("WhatsApp").Visible = True
                Else
                    ' Si la columna se ha ocultado anteriormente, mostrarla nuevamente
                    usuariosDataTable = dataSet.Tables("UsuarioData")
                    usuariosDataTable.Columns.Add("WhatsApp", GetType(Boolean))
                    usuariosDataTable.Columns("WhatsApp").SetOrdinal(3) ' Mostrar la columna
en la posición deseada
                    dgvusuarios.Columns("WhatsApp").Visible = True
```

```
            End If
        End Using
    End Using
Catch ex As Exception
    MessageBox.Show("Error al buscar usuarios. Detalles: " & ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
End Try
End Sub
```

Realiza una búsqueda de usuarios en la base de datos y actualiza el DataGridView con los resultados. Muestra la columna "WhatsApp" si está presente.

18 Manual Técnico: Módulo de Membresías

1. Importación de Librerías

```
Imports MySql.Data.MySqlClient
```

Este código importa la biblioteca MySQL Connector para .NET, permitiendo la conexión y manipulación de bases de datos MySQL desde Visual Basic.

2. Declaración de Variables y Configuración Inicial

vb

Dim connectionString As String = "server=localhost;database=gym;user id=root;password=root;"

Private alertaMostrada As Boolean = False

connectionString: Cadena de conexión para acceder a la base de datos MySQL.

alertaMostrada: Variable para rastrear si se ha mostrado una alerta de membresías cercanas.

3. Método membresia_Load

```
Private Sub membresia_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    dtpfin.Value = dtpinicio.Value.AddMonths(1)
    CargarDatosMembresia()
    Timer1.Start()
    btnmodificar.Enabled = False
    btneliminar.Enabled = False
    AddHandler dgvusuarios.DataBindingComplete, AddressOf AlternarColorFilas
End Sub
```

membresia_Load: Inicializa el formulario cargando datos, configurando la fecha de fin como un mes después de la fecha de inicio, iniciando el temporizador y deshabilitando botones de modificar y eliminar. También alterna el color de las filas en el DataGridView.

4. Método FormularioAbierto

```
Private Function FormularioAbierto() As Boolean
    Return Application.OpenForms.Count > 0
End Function
```

FormularioAbierto: Verifica si algún formulario está actualmente abierto.

5. Método btncancelar_Click

```
Private Sub btncancelar_Click(sender As Object, e As EventArgs) Handles btncancelar.Click
    LimpiarControles()
    btnguardar.Enabled = True
    txtcedula.Enabled = True
    btnmodificar.Enabled = False
    btneliminar.Enabled = False
End Sub
```

btncancelar_Click: Limpia los controles del formulario y habilita el botón de guardar mientras desactiva los botones de modificar y eliminar.

6. Método dtpinicio_ValueChanged

```
Private Sub dtpinicio_ValueChanged(sender As Object, e As EventArgs) Handles dtpinicio.ValueChanged
    dtpfin.Value = dtpinicio.Value.AddMonths(1)
End Sub
```

dtpinicio_ValueChanged: Actualiza la fecha de fin cuando cambia la fecha de inicio.

7. Método CargarDatosMembresia

```
Private Sub CargarDatosMembresia()
    Try
        Using connection As New MySqlConnection(connectionString)
            connection.Open()
            Dim query As String = "SELECT * FROM membresia"
            Using adapter As New MySqlDataAdapter(query, connection)
                Dim dataset As New DataSet()
                adapter.Fill(dataset, "MembresiaData")
                dgvusuarios.DataSource = dataset.Tables("MembresiaData")
```

End Using

End Using

Catch ex As Exception

MessageBox.Show("Error al cargar datos de membresía. Detalles: " & ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

End Try

End Sub

CargarDatosMembresia: Carga los datos de la membresía desde la base de datos y los muestra en el DataGridView.

8. Método btnguardar_Click

Private Sub btnguardar_Click(sender As Object, e As EventArgs) Handles btnguardar.Click

If String.IsNullOrWhiteSpace(txtcedula.Text) OrElse String.IsNullOrWhiteSpace(txtpago.Text) Then

MessageBox.Show("Por favor, complete todos los campos requeridos.", "Error de validación", MessageBoxButtons.OK, MessageBoxIcon.Error)

Return

End If

```
        Dim pago As Decimal
        If Not Decimal.TryParse(txtpago.Text, pago) Then
            MessageBox.Show("El campo de pago debe contener un valor numérico válido.",
"Error de validación", MessageBoxButtons.OK, MessageBoxIcon.Error)
            Return
        End If

        Dim entrenador As String = ""
        If Rdbentrenador.Checked Then
            entrenador = "Entrenador"
        ElseIf Rdbsinentrenador.Checked Then
            entrenador = "Sin_entrenador"
        ElseIf RadioButtonMEMBRESIA_GIMNASIO_X2_PERSONAS.Checked Then
            entrenador = "MEMBRESIA_GIMNASIO_X2_PERSONAS"
        ElseIf RadioButtonMEMBRESIA_BAILOTERAPIA.Checked Then
            entrenador = "MEMBRESIA_BAILOTERAPIA"
        ElseIf RadioButtonMEMBRESIA_VIP.Checked Then
```

```
            entrenador = "MEMBRESIA_VIP"
        End If

        Try
            Using connection As New MySqlConnection(connectionString)
                connection.Open()
                Dim query As String = "INSERT INTO membresia (id_cedula, fecha_inicio, fecha_fin, pago, pagado, entrenador, nota) VALUES (@cedula, @inicio, @fin, @pago, @pagado, @entrenador, @nota)"
                Using command As New MySqlCommand(query, connection)
                    command.Parameters.AddWithValue("@cedula", txtcedula.Text)
                    command.Parameters.AddWithValue("@inicio", dtpinicio.Value)
                    command.Parameters.AddWithValue("@fin", dtpfin.Value)
                    command.Parameters.AddWithValue("@pago", pago)
                    command.Parameters.AddWithValue("@pagado", If(Rdbpagado.Checked, "Pagado", "Debe"))
                    command.Parameters.AddWithValue("@entrenador", entrenador)
                    command.Parameters.AddWithValue("@nota", txtnota.Text)
```

```
            command.ExecuteNonQuery()
        End Using
    End Using

    LimpiarControles()
    CargarDatosMembresia()
    MessageBox.Show("Datos guardados exitosamente.", "Éxito",
MessageBoxButtons.OK, MessageBoxIcon.Information)
Catch ex As Exception
    MessageBox.Show("Error al guardar los datos. Detalles: " & ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
End Try
End Sub
```

btnguardar_Click: Valida los campos, determina el tipo de membresía seleccionado, y guarda los datos en la base de datos. Luego limpia los controles y actualiza el DataGridView.

9. Método btnmodificar_Click

```
Private Sub btnmodificar_Click(sender As Object, e As EventArgs) Handles
btnmodificar.Click
```

```
Try
    If dgvusuarios.SelectedRows.Count = 0 Then
        MessageBox.Show("Por favor, seleccione una fila para modificar.", "Error de validación", MessageBoxButtons.OK, MessageBoxIcon.Error)
        Return
    End If

    Dim id As Integer = Convert.ToInt32(dgvusuarios.SelectedRows(0).Cells("id").Value)

    Dim pago As Decimal
    If Not Decimal.TryParse(txtpago.Text, pago) Then
        MessageBox.Show("El campo de pago debe contener un valor numérico válido.", "Error de validación", MessageBoxButtons.OK, MessageBoxIcon.Error)
        Return
    End If

    Dim entrenador As String = ""
    If Rdbentrenador.Checked Then
```

```
            entrenador = "Entrenador"
        ElseIf Rdbsinentrenador.Checked Then
            entrenador = "Sin_entrenador"
        ElseIf RadioButtonMEMBRESIA_GIMNASIO_X2_PERSONAS.Checked Then
            entrenador = "MEMBRESIA_GIMNASIO_X2_PERSONAS"
        ElseIf RadioButtonMEMBRESIA_BAILOTERAPIA.Checked Then
            entrenador = "MEMBRESIA_BAILOTERAPIA"
        ElseIf RadioButtonMEMBRESIA_VIP.Checked Then
            entrenador = "MEMBRESIA_VIP"
        End If

        Using connection As New MySqlConnection(connectionString)
            connection.Open()
            Dim query As String = "UPDATE membresia SET id_cedula = @cedula,
fecha_inicio = @inicio, fecha_fin = @fin, pago = @pago, pagado = @pagado, entrenador =
@entrenador, nota = @nota WHERE id = @id"
            Using command As New MySqlCommand(query, connection)
                command.Parameters.AddWithValue("@id", id)
```

```
                command.Parameters.AddWithValue("@cedula", txtcedula.Text)
                command.Parameters.AddWithValue("@inicio", dtpinicio.Value)
                command.Parameters.AddWithValue("@fin", dtpfin.Value)
                command.Parameters.AddWithValue("@pago", pago)
                command.Parameters.AddWithValue("@pagado",          If(Rdbpagado.Checked,
"Pagado", "Debe"))
                command.Parameters.AddWithValue("@entrenador", entrenador)
                command.Parameters.AddWithValue("@nota", txtnota.Text)
                command.ExecuteNonQuery()
            End Using
        End Using

        LimpiarControles()
        CargarDatosMembresia()
        txtcedula.Enabled = True
        MessageBox.Show("Registro          modificado          exitosamente.",          "Éxito",
MessageBoxButtons.OK, MessageBoxIcon.Information)
    Catch ex As Exception
```

MessageBox.Show("Error al modificar los datos. Detalles: " & ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

End Try

End Sub

btnmodificar_Click: Modifica un registro seleccionado en el DataGridView con los datos ingresados en el formulario. Valida los campos y actualiza la base de datos.

10. Método btneliminar_Click

Private Sub btneliminar_Click(sender As Object, e As EventArgs) Handles btneliminar.Click

Try

If dgvusuarios.SelectedRows.Count = 0 Then

MessageBox.Show("Por favor, seleccione una fila para eliminar.", "Error de validación", MessageBoxButtons.OK, MessageBoxIcon.Error)

Return

End If

Dim id As Integer = Convert.ToInt32(dgvusuarios.SelectedRows(0).Cells("id").Value)

```
            Using connection As New MySqlConnection(connectionString)
                connection.Open()
                Dim query As String = "DELETE FROM membresia WHERE id = @id"
                Using command As New MySqlCommand(query, connection)
                    command.Parameters.AddWithValue("@id", id)
                    command.ExecuteNonQuery()
                End Using
            End Using

            LimpiarControles()
            CargarDatosMembresia()
            MessageBox.Show("Registro eliminado exitosamente.", "Éxito",
MessageBoxButtons.OK, MessageBoxIcon.Information)
        Catch ex As Exception
            MessageBox.Show("Error al eliminar los datos. Detalles: " & ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub
```

btneliminar_Click: Elimina el registro seleccionado del DataGridView y actualiza la base de datos.

11. Método AlternarColorFilas

```
Private Sub AlternarColorFilas(sender As Object, e As DataGridViewBindingCompleteEventArgs)
    For Each row As DataGridViewRow In dgvusuarios.Rows
        If row.Index Mod 2 = 0 Then
            row.DefaultCellStyle.BackColor = Color.AliceBlue
        Else
            row.DefaultCellStyle.BackColor = Color.White
        End If
    Next
End Sub
```

AlternarColorFilas: Alterna el color de fondo de las filas en el DataGridView para mejorar la legibilidad.

12. Método Timer1_Tick

```
Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
    Dim fechaActual As Date = Date.Now
```

```
        For Each row As DataGridViewRow In dgvusuarios.Rows
            If row.Cells("fecha_fin").Value IsNot DBNull.Value AndAlso
Convert.ToDateTime(row.Cells("fecha_fin").Value) < fechaActual.AddDays(3) Then
                If Not alertaMostrada Then
                    MessageBox.Show("¡Alerta! Existen membresías próximas a vencer.", "Alerta de
Membresía", MessageBoxButtons.OK, MessageBoxIcon.Warning)
                    alertaMostrada = True
                End If
            End If
        Next
    End Sub
```

Timer1_Tick: Verifica periódicamente si hay membresías próximas a vencer y muestra una alerta si es necesario.

13. Método LimpiarControles

vb

```
    Private Sub LimpiarControles()
        txtcedula.Clear()
```

```
        dtpinicio.Value = Date.Now

        dtpfin.Value = Date.Now.AddMonths(1)

        txtpago.Clear()

        txtnota.Clear()

        Rdbpagado.Checked = False

        Rdbnopagado.Checked = False

        Rdbentrenador.Checked = False

        Rdbsinentrenador.Checked = False

    End Sub
```

LimpiarControles: Limpia los controles del formulario para preparar el formulario para una nueva entrada.

19 Documentación Técnica: Formulario FormProductos

Importaciones

```
Imports MySql.Data.MySqlClient
```

Importaciones: Incluye la biblioteca para manejar conexiones y comandos MySQL.

Variables y Configuración Inicial

```
Private imagenes() As Image
```

```
Private imagenIndex As Integer = 0

Private WithEvents timer As New System.Windows.Forms.Timer()

Private connectionString As String = "server=localhost;database=gym;user id=root;password=root;"

Private idProductoSeleccionado As Integer = 0
```

imagenes: Array para almacenar imágenes para el panel de anuncios.

imagenIndex: Índice actual de la imagen que se está mostrando en el panel de anuncios.

timer: Temporizador para cambiar imágenes en el panel.

connectionString: Cadena de conexión para la base de datos MySQL.

idProductoSeleccionado: ID del producto actualmente seleccionado en la DataGridView.

Método btnGuardar_Click

```
Private Sub btnGuardar_Click(sender As Object, e As EventArgs) Handles btnguardar.Click
    If String.IsNullOrWhiteSpace(txtnombre.Text) Then
        MessageBox.Show("Ingrese un nombre para el producto.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        Return
    End If
```

```
        Try
            Using connection As New MySqlConnection(connectionString)
                connection.Open()
                Dim query As String = "INSERT INTO productos (nombre, descripcion, cantidad,
precio, fecha_ingreso) " &
                                      "VALUES (@nombre, @descripcion, @cantidad, @precio, NOW())"
                Using command As New MySqlCommand(query, connection)
                    command.Parameters.AddWithValue("@nombre", txtnombre.Text)
                    command.Parameters.AddWithValue("@descripcion",
If(String.IsNullOrWhiteSpace(txtdescripcion.Text), DBNull.Value, txtdescripcion.Text))
                    command.Parameters.AddWithValue("@cantidad",
Convert.ToInt32(txtcantidad.Text))
                    command.Parameters.AddWithValue("@precio",
Convert.ToDecimal(txtprecio.Text))
                    command.ExecuteNonQuery()
                    MessageBox.Show("Producto guardado correctamente.", "Información",
MessageBoxButtons.OK, MessageBoxIcon.Information)
                    LimpiarCampos()
```

```
                CargarProductos()
            End Using
        End Using
    Catch ex As Exception
        MessageBox.Show("Error al guardar el producto: " & ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
    End Try
End Sub
```

btnGuardar_Click: Inserta un nuevo producto en la base de datos y actualiza la DataGridView. Incluye validación para asegurar que se ingrese un nombre.

Método LimpiarCampos

```
Private Sub LimpiarCampos()
    txtnombre.Clear()
    txtdescripcion.Clear()
    txtcantidad.Clear()
    txtprecio.Clear()
End Sub
```

LimpiarCampos: Limpia los campos de entrada del formulario después de una operación.

Método CargarProductos

```
Private Sub CargarProductos()
    Try
        Using connection As New MySqlConnection(connectionString)
            connection.Open()
            Dim query As String = "SELECT id_producto, nombre, descripcion, cantidad, precio, fecha_ingreso FROM productos"
            Using adapter As New MySqlDataAdapter(query, connection)
                Dim dtProductos As New DataTable()
                adapter.Fill(dtProductos)
                DataGridView1.DataSource = dtProductos
            End Using
        End Using
    Catch ex As Exception
        MessageBox.Show("Error al cargar los productos: " & ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
    End Try
End Sub
```

CargarProductos: Carga todos los productos desde la base de datos y los muestra en la DataGridView.

Evento FormProductos_Load

Private Sub FormProductos_Load(sender As Object, e As EventArgs) Handles MyBase.Load

```
imagenes = New Image() {
   GYMESPARTA.My.Resources.proteina,
   GYMESPARTA.My.Resources.proteina2,
   GYMESPARTA.My.Resources.proteina3,
   GYMESPARTA.My.Resources.proteina4,
   GYMESPARTA.My.Resources.proteina5
}
timer.Interval = 3000
timer.Start()
btnmodificar.Enabled = False
btneliminar.Enabled = False
CargarProductos()
AddHandler DataGridView1.DataBindingComplete, AddressOf AlternarColorFilas
```

End Sub

FormProductos_Load: Inicializa el formulario cargando imágenes para el panel de anuncios, configurando el temporizador y cargando productos en la DataGridView. También establece el estado de los botones y agrega un manejador para alternar el color de las filas.

Método DataGridView1_SelectionChanged

Private Sub DataGridView1_SelectionChanged(sender As Object, e As EventArgs) Handles DataGridView1.SelectionChanged

LlenarTextBoxDesdeDataGridView()

End Sub

DataGridView1_SelectionChanged: Llena los campos del formulario con los datos del producto seleccionado en la DataGridView.

Método LlenarTextBoxDesdeDataGridView

Private Sub LlenarTextBoxDesdeDataGridView()

If DataGridView1.SelectedRows.Count > 0 Then

idProductoSeleccionado = Convert.ToInt32(DataGridView1.SelectedRows(0).Cells("id_producto").Value)

txtnombre.Text = DataGridView1.SelectedRows(0).Cells("nombre").Value.ToString()

```
            txtdescripcion.Text =
DataGridView1.SelectedRows(0).Cells("descripcion").Value.ToString()
            txtcantidad.Text =
DataGridView1.SelectedRows(0).Cells("cantidad").Value.ToString()
            txtprecio.Text = DataGridView1.SelectedRows(0).Cells("precio").Value.ToString()
            btnmodificar.Enabled = True
            btnguardar.Enabled = False
            btneliminar.Enabled = True
        End If
    End Sub
```

LlenarTextBoxDesdeDataGridView: Rellena los campos del formulario con la información del producto seleccionado y habilita/deshabilita los botones correspondientes.

Método btnmodificar_Click

```
    Private Sub btnmodificar_Click(sender As Object, e As EventArgs) Handles
btnmodificar.Click
        Try
            If idProductoSeleccionado > 0 Then
                Using connection As New MySqlConnection(connectionString)
```

```
            connection.Open()

            Dim query As String = "UPDATE productos SET nombre = @nombre,
descripcion = @descripcion, cantidad = @cantidad, precio = @precio WHERE id_producto =
@id_producto"

            Using cmd As New MySqlCommand(query, connection)

                cmd.Parameters.AddWithValue("@nombre", txtnombre.Text)

                cmd.Parameters.AddWithValue("@descripcion", txtdescripcion.Text)

                cmd.Parameters.AddWithValue("@cantidad",
Convert.ToInt32(txtcantidad.Text))

                cmd.Parameters.AddWithValue("@precio",
Convert.ToDecimal(txtprecio.Text))

                cmd.Parameters.AddWithValue("@id_producto", idProductoSeleccionado)

                cmd.ExecuteNonQuery()

                MessageBox.Show("Producto modificado con éxito.", "Éxito",
MessageBoxButtons.OK, MessageBoxIcon.Information)

                btnguardar.Enabled = False

                CargarProductos()

            End Using
```

```
            End Using
        Else
            MessageBox.Show("Selecciona un producto para modificar.", "Advertencia",
MessageBoxButtons.OK, MessageBoxIcon.Warning)
        End If
    Catch ex As Exception
        MessageBox.Show("Error al modificar el producto: " & ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
    End Try
End Sub
```

btnmodificar_Click: Modifica el producto seleccionado en la base de datos utilizando el ID del producto. Muestra un mensaje de éxito o advertencia según corresponda.

Método btncancelar_Click

```
Private Sub btncancelar_Click(sender As Object, e As EventArgs) Handles
btncancelar.Click
    LimpiarCampos()
    btneliminar.Enabled = False
    btnguardar.Enabled = True
```

btnmodificar.Enabled = False

End Sub

btncancelar_Click: Limpia los campos del formulario y restablece el estado de los botones.

Métodos de Validación de Entrada de Texto

Private Sub txtnombre_KeyPress(sender As Object, e As KeyPressEventArgs) Handles txtnombre.KeyPress

If Not Char.IsLetterOrDigit(e.KeyChar) AndAlso Not Char.IsWhiteSpace(e.KeyChar) AndAlso Not Char.IsControl(e.KeyChar) Then

e.Handled = True

End If

e.KeyChar = Char.ToUpper(e.KeyChar)

End Sub

Private Sub txtcantidad_KeyPress(sender As Object, e As KeyPressEventArgs) Handles txtcantidad.KeyPress

If Not Char.IsDigit(e.KeyChar) AndAlso Not Char.IsControl(e.KeyChar) Then

e.Handled = True

End If

```
        If e.KeyChar = ControlChars.Back Then
            Return
        End If
    End Sub

    Private Sub txtprecio_KeyPress(sender As Object, e As KeyPressEventArgs) Handles txtprecio.KeyPress
        If Not Char.IsDigit(e.KeyChar) AndAlso Not Char.IsControl(e.KeyChar) AndAlso e.KeyChar <> ","c Then
            e.Handled = True
        End If
        If e.KeyChar = ControlChars.Back Then
            Return
        End If
    End Sub
```

txtnombre_KeyPress: Permite solo letras, dígitos, espacios y caracteres de control en el campo txtnombre. Convierte las letras a mayúsculas.

txtcantidad_KeyPress: Permite solo dígitos y caracteres de control en el campo txtcantidad.

txtprecio_KeyPress: Permite dígitos, comas y caracteres de control en el campo txtprecio.

Método AlternarColorFilas

```
Private Sub AlternarColorFilas(sender As Object, e As DataGridViewBindingCompleteEventArgs)
    For i As Integer = 0 To DataGridView1.Rows.Count - 1
        If i Mod 2 = 0 Then
            DataGridView1.Rows(i).DefaultCellStyle.BackColor = Color.LightGray
        Else
            DataGridView1.Rows(i).DefaultCellStyle.BackColor = Color.White
        End If
    Next
End Sub
```

AlternarColorFilas: Alterna el color de las filas en la DataGridView para mejorar la legibilidad.

Métodos del Temporizador y Panel de Anuncios

```
Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles timer.Tick
    CambiarImagenEnPanel()
End Sub
```

```
Private Sub CambiarImagenEnPanel()
    imagenIndex = (imagenIndex + 1) Mod imagenes.Length
    panelanuncios.Invalidate()
End Sub

Private Sub panelanuncios_Paint(sender As Object, e As PaintEventArgs) Handles panelanuncios.Paint
    If imagenes.Length > 0 Then
        Dim imagenActual As Image = imagenes(imagenIndex)
        Dim graphics As Graphics = e.Graphics
        Dim rectDestino As New Rectangle(0, 0, panelanuncios.Width, panelanuncios.Height)
        graphics.DrawImage(imagenActual, rectDestino)
    End If
End Sub
```

Timer1_Tick: Evento del temporizador que cambia la imagen en el panel de anuncios cada 3 segundos.

CambiarImagenEnPanel: Actualiza el índice de la imagen y redibuja el panel.

panelanuncios_Paint: Dibuja la imagen actual en el panel de anuncios.

Método txtbuscar_TextChanged

Private Sub txtbuscar_TextChanged(sender As Object, e As EventArgs) Handles txtbuscar.TextChanged

BuscarProductos(txtbuscar.Text)

End Sub

txtbuscar_TextChanged: Llama al método de búsqueda cuando cambia el texto en el campo txtbuscar.

Método BuscarProductos

Private Sub BuscarProductos(criterio As String)

Try

Using connection As New MySqlConnection(connectionString)

connection.Open()

Dim query As String = "SELECT id_producto, nombre, descripcion, cantidad, precio, fecha_ingreso " &

"FROM productos " &

"WHERE nombre LIKE @criterio OR descripcion LIKE @criterio"

Using adapter As New MySqlDataAdapter(query, connection)

```
                adapter.SelectCommand.Parameters.AddWithValue("@criterio", "%" & criterio
& "%")

                Dim dtProductos As New DataTable()

                adapter.Fill(dtProductos)

                DataGridView1.DataSource = dtProductos

            End Using

        End Using

    Catch ex As Exception

        MessageBox.Show("Error al buscar los productos: " & ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)

    End Try

End Sub
```

BuscarProductos: Busca productos en la base de datos que coincidan con el criterio proporcionado y actualiza la DataGridView.

20 CODIGO PARA VENTA DE PRODUCTOS

Declaraciones e Inicialización

```
Imports MySql.Data.MySqlClient
```

```
Public Class venderproductos


    Private imagenes() As Image

    Private imagenIndex As Integer = 0

    Private WithEvents timer As New System.Windows.Forms.Timer()

    Dim connectionString As String = "server=localhost;database=gym;user
id=root;password=root;"

    Dim idProductoSeleccionado As Integer = -1 ' Almacena el ID del producto seleccionado
```

imagenes(): Array que almacena las imágenes que se mostrarán en un panel de anuncios.

imagenIndex: Índice actual de la imagen mostrada.

timer: Temporizador que controla el cambio de imágenes cada 3 segundos.

connectionString: Cadena de conexión a la base de datos MySQL.

idProductoSeleccionado: Almacena el ID del producto seleccionado en la DataGridView.

Método venderproductos_Load

```
Private Sub venderproductos_Load(sender As Object, e As EventArgs) Handles
MyBase.Load

    imagenes = New Image() {

        GYMESPARTA.My.Resources.proteina,
```

```
            GYMESPARTA.My.Resources.proteina2,
            GYMESPARTA.My.Resources.proteina3,
            GYMESPARTA.My.Resources.proteina4,
            GYMESPARTA.My.Resources.proteina5
        }

        Timer1.Interval = 3000
        Timer1.Start()

        CargarProductos()
        txtnombredelproducto.Enabled = False
        AddHandler DataGridView1.DataBindingComplete, AddressOf AlternarColorFilas
    End Sub
```

imagenes: Carga las imágenes desde los recursos del proyecto.

Timer1.Interval: Configura el temporizador para cambiar la imagen cada 3 segundos.

CargarProductos: Carga los productos desde la base de datos y los muestra en la DataGridView.

AddHandler: Asigna un método (AlternarColorFilas) que se ejecutará cuando los datos se completen en la DataGridView.

Método CargarProductos

```
Private Sub CargarProductos()
    Try
        Using connection As New MySqlConnection(connectionString)
            connection.Open()
            Dim query As String = "SELECT id_producto, nombre, descripcion, cantidad, precio, fecha_ingreso FROM productos"
            Using adapter As New MySqlDataAdapter(query, connection)
                Dim dtProductos As New DataTable()
                adapter.Fill(dtProductos)
                DataGridView1.DataSource = dtProductos
            End Using
        End Using
    Catch ex As Exception
        MessageBox.Show("Error al cargar los productos: " & ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
```

End Try

End Sub

CargarProductos: Realiza una consulta a la base de datos para obtener todos los productos y los muestra en la DataGridView.

Método DataGridView1_CellClick

Private Sub DataGridView1_CellClick(sender As Object, e As DataGridViewCellEventArgs) Handles DataGridView1.CellClick

If e.RowIndex >= 0 Then

idProductoSeleccionado = Convert.ToInt32(DataGridView1.Rows(e.RowIndex).Cells("id_producto").Value)

txtnombredelproducto.Text = DataGridView1.Rows(e.RowIndex).Cells("nombre").Value.ToString()

End If

End Sub

DataGridView1_CellClick: Al hacer clic en una celda, se selecciona el ID del producto y se muestra su nombre en el campo txtnombredelproducto.

Método btnvender_Click

Private Sub btnvender_Click(sender As Object, e As EventArgs) Handles btnvender.Click

```
        If idProductoSeleccionado <> -1 Then
            Dim cantidadAVender As Integer = Convert.ToInt32(txtCantidadAVender.Text)
            RealizarVenta(idProductoSeleccionado, cantidadAVender)
            CargarProductos()
            LimpiarControles()
        Else
            MessageBox.Show("Selecciona un producto antes de vender.", "Aviso",
MessageBoxButtons.OK, MessageBoxIcon.Information)
        End If
    End Sub
```

btnvender_Click: Inicia el proceso de venta si se ha seleccionado un producto. Realiza la venta, actualiza la DataGridView y limpia los controles.

Método RealizarVenta

```
    Private Sub RealizarVenta(idProducto As Integer, cantidadAVender As Integer)
        Try
            Using connection As New MySqlConnection(connectionString)
                connection.Open()
```

```
            Dim updateQuery As String = "UPDATE productos SET cantidad = cantidad -
@cantidad WHERE id_producto = @id_producto"
            Using cmd As New MySqlCommand(updateQuery, connection)
                cmd.Parameters.AddWithValue("@cantidad", cantidadAVender)
                cmd.Parameters.AddWithValue("@id_producto", idProducto)
                cmd.ExecuteNonQuery()
            End Using

            Dim productoQuery As String = "SELECT nombre, precio FROM productos
WHERE id_producto = @id_producto"
            Using cmdProducto As New MySqlCommand(productoQuery, connection)
                cmdProducto.Parameters.AddWithValue("@id_producto", idProducto)
                Dim reader As MySqlDataReader = cmdProducto.ExecuteReader()

                If reader.Read() Then
                    Dim nombreProducto As String = reader("nombre").ToString()
                    Dim precioProducto As Decimal = Convert.ToDecimal(reader("precio"))
                    reader.Close()
```

```
                Dim totalVenta As Decimal = cantidadAVender * precioProducto

                Dim insertQuery As String = "INSERT INTO ventas (id_producto, cantidad,
total) VALUES (@id_producto, @cantidad, @total)"
                Using cmdInsert As New MySqlCommand(insertQuery, connection)
                    cmdInsert.Parameters.AddWithValue("@id_producto", idProducto)
                    cmdInsert.Parameters.AddWithValue("@cantidad", cantidadAVender)
                    cmdInsert.Parameters.AddWithValue("@total", totalVenta)
                    cmdInsert.ExecuteNonQuery()
                End Using

                MessageBox.Show($"Venta realizada: {cantidadAVender} unidades de
{nombreProducto}. Total: {totalVenta:C}", "Venta Exitosa", MessageBoxButtons.OK,
MessageBoxIcon.Information)
            End If
        End Using
    End Using
```

```
        Catch ex As Exception
            MessageBox.Show("Error al realizar la venta: " & ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub
```

RealizarVenta: Actualiza la cantidad de producto en la base de datos, calcula el total de la venta, y registra la venta en la tabla correspondiente. Muestra un mensaje confirmando la venta exitosa.

Métodos de Utilidad

```
    Private Sub LimpiarControles()
        idProductoSeleccionado = -1
        txtnombredelproducto.Clear()
        txtCantidadAVender.Clear()
    End Sub


    Private Sub txtCantidadAVender_KeyPress(sender As Object, e As KeyPressEventArgs)
Handles txtCantidadAVender.KeyPress
```

```
        If Not Char.IsDigit(e.KeyChar) AndAlso Not Char.IsControl(e.KeyChar) AndAlso
e.KeyChar <> ","c Then
            e.Handled = True
        End If
        If e.KeyChar = ControlChars.Back Then
            Return
        End If
    End Sub

    Private Sub AlternarColorFilas(sender As Object, e As
DataGridViewBindingCompleteEventArgs)
        For i As Integer = 0 To DataGridView1.Rows.Count - 1
            If i Mod 2 = 0 Then
                DataGridView1.Rows(i).DefaultCellStyle.BackColor = Color.LightGray
            Else
                DataGridView1.Rows(i).DefaultCellStyle.BackColor = Color.White
            End If
        Next
```

```
End Sub

Private Sub CambiarImagenEnPanel()
    imagenIndex = (imagenIndex + 1) Mod imagenes.Length
    panelanuncios.Invalidate()
End Sub

Private Sub panelanuncios_Paint(sender As Object, e As PaintEventArgs) Handles panelanuncios.Paint
    If imagenes.Length > 0 Then
        Dim imagenActual As Image = imagenes(imagenIndex)
        Dim graphics As Graphics = e.Graphics
        Dim rectDestino As New Rectangle(0, 0, panelanuncios.Width, panelanuncios.Height)
        graphics.DrawImage(imagenActual, rectDestino)
    End If
End Sub
```

```
Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
    CambiarImagenEnPanel()
End Sub

Private Sub txtbuscar_TextChanged(sender As Object, e As EventArgs) Handles txtbuscar.TextChanged
    BuscarProductos(txtbuscar.Text)
End Sub

Private Sub BuscarProductos(criterio As String)
    Try
        Using connection As New MySqlConnection(connectionString)
            connection.Open()
            Dim query As String = "SELECT id_producto, nombre, descripcion, cantidad, precio, fecha_ingreso " &
                                  "FROM productos " &
                                  "WHERE nombre LIKE @criterio OR descripcion LIKE @criterio"
            Using adapter As New MySqlDataAdapter(query, connection)
```

```
                adapter.SelectCommand.Parameters.AddWithValue("@criterio", "%" & criterio
& "%")

                Dim dtProductos As New DataTable()

                adapter.Fill(dtProductos)

                DataGridView1.DataSource = dtProductos

            End Using

        End Using

    Catch ex As Exception

        MessageBox.Show("Error al buscar los productos: " & ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)

    End Try

End Sub
```

LimpiarControles: Limpia los campos de texto y reinicia la selección del producto.

txtCantidadAVender_KeyPress: Valida la entrada del usuario, permitiendo solo números y la tecla de retroceso.

AlternarColorFilas: Alterna el color de las filas en la DataGridView para mejorar la visualización.

CambiarImagenEnPanel: Cambia la imagen mostrada en el panel cada vez que se ejecuta.

panelanuncios_Paint: Dibuja la imagen en el panel.

BuscarProductos: Filtra los productos en la DataGridView basándose en el criterio de búsqueda ingresado por el usuario.

21 CODIGO PARA DATOS DE MEDICIÓN

```
Imports MySql.Data.MySqlClient


Public Class datos


    ' Cadena de conexión a la base de datos MySQL
    Dim connectionString As String = "server=localhost;database=gym;user id=root;password=root;"
    Dim idMedicionSeleccionada As Integer = 0


    ' Cargar mediciones al cargar el formulario
    Private Sub datos_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        CargarMediciones()
        AddHandler DataGridView1.DataBindingComplete, AddressOf AlternarColorFilas
```

```
End Sub


' Método para limpiar los campos de texto
Private Sub LimpiarCampos()
    txtcedula.Clear()
    txtaltura.Clear()
    txtpeso.Clear()
    txtcintura.Clear()
    txtgluteos.Clear()
    txtpiernas.Clear()
    txtpecho.Clear()
    txtcedula.Enabled = True
End Sub


' Método para cargar las mediciones desde la base de datos al DataGridView
Private Sub CargarMediciones()
    Try
```

```
            Using connection As New MySqlConnection(connectionString)
                connection.Open()
                Dim query As String = "SELECT * FROM mediciones"
                Using adapter As New MySqlDataAdapter(query, connection)
                    Dim dataset As New DataSet()
                    adapter.Fill(dataset, "MedicionesData")
                    DataGridView1.DataSource = dataset.Tables("MedicionesData")
                End Using
            End Using
        Catch ex As Exception
            MessageBox.Show("Error al cargar datos de mediciones. Detalles: " & ex.Message,
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

    ' Evento para guardar una nueva medición en la base de datos
    Private Sub btnguardar_Click(sender As Object, e As EventArgs) Handles
btnguardar.Click
```

```
' Validar que todos los campos estén llenos
If String.IsNullOrEmpty(txtcedula.Text) OrElse
   String.IsNullOrEmpty(txtaltura.Text) OrElse
   String.IsNullOrEmpty(txtpeso.Text) OrElse
   String.IsNullOrEmpty(txtcintura.Text) OrElse
   String.IsNullOrEmpty(txtgluteos.Text) OrElse
   String.IsNullOrEmpty(txtpiernas.Text) OrElse
   String.IsNullOrEmpty(txtpecho.Text) Then
    MessageBox.Show("Por favor, completa todos los campos.", "Error de validación",
MessageBoxButtons.OK, MessageBoxIcon.Error)
    Return
End If

Try
    Using connection As New MySqlConnection(connectionString)
        connection.Open()
```

```
' Consulta SQL para insertar una nueva medición
Dim insertQuery As String = "INSERT INTO mediciones (id_usuario, altura, peso, cintura, gluteos, piernas, pecho, fecha_medicion) VALUES (@id_usuario, @altura, @peso, @cintura, @gluteos, @piernas, @pecho, CURRENT_DATE)"

Using cmd As New MySqlCommand(insertQuery, connection)
    ' Asignar parámetros para la inserción
    cmd.Parameters.AddWithValue("@id_usuario", txtcedula.Text)
    cmd.Parameters.AddWithValue("@altura", txtaltura.Text)
    cmd.Parameters.AddWithValue("@peso", txtpeso.Text)
    cmd.Parameters.AddWithValue("@cintura", txtcintura.Text)
    cmd.Parameters.AddWithValue("@gluteos", txtgluteos.Text)
    cmd.Parameters.AddWithValue("@piernas", txtpiernas.Text)
    cmd.Parameters.AddWithValue("@pecho", txtpecho.Text)

    ' Ejecutar la inserción
    cmd.ExecuteNonQuery()
```

```
                MessageBox.Show("Medición guardada correctamente.", "Éxito",
MessageBoxButtons.OK, MessageBoxIcon.Information)

                ' Limpiar los campos después de guardar
                LimpiarCampos()

                ' Volver a cargar las mediciones en la DataGridView
                CargarMediciones()
            End Using
        End Using
    Catch ex As Exception
        MessageBox.Show("Error al guardar la medición: " & ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
    End Try
End Sub

' Método para actualizar una fila en la base de datos
Private Sub ActualizarFila(idFila As Integer)
```

```
Try
    Using connection As New MySqlConnection(connectionString)
        connection.Open()

        ' Consulta SQL para actualizar la fila
        Dim updateQuery As String = "UPDATE mediciones SET id_usuario =
@id_usuario, altura = @altura, peso = @peso, cintura = @cintura, gluteos = @gluteos, piernas =
@piernas, pecho = @pecho WHERE id_medicion = @id_medicion"

        Using cmd As New MySqlCommand(updateQuery, connection)
            ' Asignar parámetros
            cmd.Parameters.AddWithValue("@id_usuario", txtcedula.Text)
            cmd.Parameters.AddWithValue("@altura", txtaltura.Text)
            cmd.Parameters.AddWithValue("@peso", txtpeso.Text)
            cmd.Parameters.AddWithValue("@cintura", txtcintura.Text)
            cmd.Parameters.AddWithValue("@gluteos", txtgluteos.Text)
            cmd.Parameters.AddWithValue("@piernas", txtpiernas.Text)
            cmd.Parameters.AddWithValue("@pecho", txtpecho.Text)
```

```
                cmd.Parameters.AddWithValue("@id_medicion", idFila)

                ' Ejecutar la actualización
                cmd.ExecuteNonQuery()

                MessageBox.Show("Fila modificada con éxito.", "Modificación Exitosa",
MessageBoxButtons.OK, MessageBoxIcon.Information)
            End Using
        End Using
    Catch ex As Exception
        MessageBox.Show("Error al modificar la fila: " & ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
    End Try
End Sub

' Evento para modificar una fila seleccionada
Private Sub btnmodificar_Click(sender As Object, e As EventArgs) Handles
btnmodificar.Click
```

```
' Verificar si se ha seleccionado una fila en la DataGridView
If DataGridView1.SelectedRows.Count > 0 Then
    ' Obtener el ID de la fila seleccionada
    Dim idFilaSeleccionada As Integer = Convert.ToInt32(DataGridView1.SelectedRows(0).Cells("id_medicion").Value)

    ' Mostrar un mensaje para confirmar la modificación
    Dim result As DialogResult = MessageBox.Show("¿Deseas modificar la fila seleccionada?", "Confirmar Modificación", MessageBoxButtons.YesNo, MessageBoxIcon.Question)

    If result = DialogResult.Yes Then
        ' Realizar la actualización en la base de datos
        ActualizarFila(idFilaSeleccionada)

        ' Limpiar campos después de la modificación
        LimpiarCampos()
```

```
            ' Recargar las filas actualizadas en la DataGridView
            CargarMediciones()
        End If
    Else
        MessageBox.Show("Selecciona una fila antes de intentar modificar.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
    End If
End Sub

' Método para alternar el color de las filas en el DataGridView
Private Sub AlternarColorFilas(sender As Object, e As
DataGridViewBindingCompleteEventArgs)
    For i As Integer = 0 To DataGridView1.Rows.Count - 1
        If i Mod 2 = 0 Then
            DataGridView1.Rows(i).DefaultCellStyle.BackColor = Color.LightGray ' Color
para filas pares
        Else
```

```
                DataGridView1.Rows(i).DefaultCellStyle.BackColor = Color.White ' Color para
filas impares
            End If
        Next
    End Sub

    ' Evento para mostrar los datos de la fila seleccionada en los TextBox al hacer clic en una
celda
    Private Sub DataGridView1_CellClick(sender As Object, e As
DataGridViewCellEventArgs) Handles DataGridView1.CellClick
        MostrarDatosSeleccionados()
    End Sub

    ' Método para mostrar los datos de la fila seleccionada en los TextBox
    Private Sub MostrarDatosSeleccionados()
        If DataGridView1.SelectedRows.Count > 0 Then
            Dim selectedRow As DataGridViewRow = DataGridView1.SelectedRows(0)
```

```
            txtcedula.Text = selectedRow.Cells("id_usuario").Value.ToString()
            txtaltura.Text = selectedRow.Cells("altura").Value.ToString()
            txtpeso.Text = selectedRow.Cells("peso").Value.ToString()
            txtcintura.Text = selectedRow.Cells("cintura").Value.ToString()
            txtgluteos.Text = selectedRow.Cells("gluteos").Value.ToString()
            txtpiernas.Text = selectedRow.Cells("piernas").Value.ToString()
            txtpecho.Text = selectedRow.Cells("pecho").Value.ToString()

            txtcedula.Enabled = False
            idMedicionSeleccionada =
Convert.ToInt32(selectedRow.Cells("id_medicion").Value)
        End If
    End Sub

    ' Evento para cancelar la operación y limpiar los campos
    Private Sub btncancelar_Click(sender As Object, e As EventArgs) Handles
btncancelar.Click
        LimpiarCampos()
```

```
End Sub

' Evento para eliminar una fila seleccionada

Private Sub btneliminar_Click(sender As Object, e As EventArgs) Handles
btneliminar.Click

    If DataGridView1.SelectedRows.Count > 0 Then

        Dim idMedicionSeleccionada As Integer =
Convert.ToInt32(DataGridView1.SelectedRows(0).Cells("id_medicion").Value)


        Dim result As DialogResult = MessageBox.Show("¿Estás seguro de que deseas
eliminar esta medición?", "Confirmar Eliminación", MessageBoxButtons.YesNo,
MessageBoxIcon.Warning)


        If result = DialogResult.Yes Then

            EliminarFila(idMedicionSeleccionada)

            LimpiarCampos()

            CargarMediciones()

        End If
```

```
            Else
                MessageBox.Show("Selecciona una fila antes de intentar eliminar.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            End If
        End Sub

        ' Método para eliminar una fila en la base de datos
        Private Sub EliminarFila(idFila As Integer)
            Try
                Using connection As New MySqlConnection(connectionString)
                    connection.Open()

                    ' Consulta SQL para eliminar la fila
                    Dim deleteQuery As String = "DELETE FROM mediciones WHERE
id_medicion = @id_medicion"

                    Using cmd As New MySqlCommand(deleteQuery, connection)
                        ' Asignar parámetro de eliminación
```

```
                cmd.Parameters.AddWithValue("@id_medicion", idFila)

                ' Ejecutar la eliminación
                cmd.ExecuteNonQuery()

                MessageBox.Show("Fila eliminada con éxito.", "Eliminación Exitosa",
MessageBoxButtons.OK, MessageBoxIcon.Information)
            End Using
        End Using
    Catch ex As Exception
        MessageBox.Show("Error al eliminar la fila: " & ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
    End Try
End Sub

' Evento para buscar mediciones en la base de datos
Private Sub txtbuscar_TextChanged(sender As Object, e As EventArgs) Handles
txtbuscar.TextChanged
```

```
        BuscarMediciones(txtbuscar.Text)
    End Sub

    ' Método para buscar mediciones en la base de datos según la búsqueda del usuario
    Private Sub BuscarMediciones(filtro As String)
        Try
            Using connection As New MySqlConnection(connectionString)
                connection.Open()

                ' Consulta SQL para buscar mediciones
                Dim searchQuery As String = "SELECT mediciones.id_medicion, mediciones.id_usuario, mediciones.altura, mediciones.peso, mediciones.cintura, mediciones.gluteos, mediciones.piernas, mediciones.pecho " &
                                            "FROM mediciones " &
                                            "INNER JOIN usuarios ON mediciones.id_usuario = usuarios.id_usuario " &
                                            "WHERE usuarios.nombre LIKE @filtro OR usuarios.apellido LIKE @filtro OR usuarios.cedula LIKE @filtro"
```

```
                Using adapter As New MySqlDataAdapter(searchQuery, connection)
                    adapter.SelectCommand.Parameters.AddWithValue("@filtro", "%" & filtro &
"%")

                    Dim dataset As New DataSet()
                    adapter.Fill(dataset, "MedicionesData")
                    DataGridView1.DataSource = dataset.Tables("MedicionesData")
                End Using
            End Using
        Catch ex As Exception
            MessageBox.Show("Error al buscar mediciones: " & ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
    End Sub

End Class
```

22 CÓDIGO DE REPORTES

Método CerrarFormularioAnteriorEnPanel

Este método se encarga de verificar si hay un formulario cargado previamente en Panel9, y si lo hay, lo cierra antes de cargar un nuevo formulario. Esto asegura que solo un formulario esté abierto en el panel a la vez.

```
Private Sub CerrarFormularioAnteriorEnPanel()
    ' Verificar si hay un formulario anterior en Panel9 y cerrarlo si es necesario
    If Panel9.Controls.Count > 0 Then
        Dim formularioAnterior As Form = TryCast(Panel9.Controls(0), Form)
        If Not IsNothing(formularioAnterior) AndAlso Not formularioAnterior.IsDisposed Then
            formularioAnterior.Close()
        End If
    End If
End Sub
```

2. Botones para Cargar Formularios

Cada botón en el formulario principal (reportesfrom) está asociado a un evento Click, que, cuando se activa, llama al método CerrarFormularioAnteriorEnPanel y luego carga el nuevo formulario deseado dentro de Panel9.

Ejemplo del botón btnproductos:

```
Private Sub Button5_Click(sender As Object, e As EventArgs) Handles btnproductos.Click
    CerrarFormularioAnteriorEnPanel()
    Dim resporproductos As New resporproductos()
    resporproductos.TopLevel = False
    Panel9.Controls.Add(resporproductos)
    resporproductos.Dock = DockStyle.Fill
    resporproductos.Show()
End Sub
```

3. Explicación General del Código

Eventos **Click**: Cada botón (btnproductos, btnusuarios, btnmembresias, btnventas, btndatos) está diseñado para cargar un formulario diferente en Panel9 cuando se hace clic en él.

Propiedades del Formulario: Los formularios cargados (resporproductos, reporteusuario, reportemembresia, reportventas, reportedatos) tienen TopLevel establecido en False para que se comporten como controles hijos de Panel9, y se establece su Dock en DockStyle.Fill para que ocupen todo el espacio disponible en el panel.

Código Completo con Comentarios

A continuación se muestra el código completo con comentarios explicativos:

vb

```
Public Class reportesfrom

  Private Sub reportesfrom_Load(sender As Object, e As EventArgs) Handles MyBase.Load

    ' Código que se ejecuta cuando el formulario principal se carga

  End Sub


  ' Método para cerrar cualquier formulario que esté abierto en Panel9

  Private Sub CerrarFormularioAnteriorEnPanel()

    ' Verificar si hay un formulario anterior en Panel9 y cerrarlo si es necesario

    If Panel9.Controls.Count > 0 Then

      Dim formularioAnterior As Form = TryCast(Panel9.Controls(0), Form)

      If Not IsNothing(formularioAnterior) AndAlso Not formularioAnterior.IsDisposed Then

        formularioAnterior.Close()

      End If

    End If

  End Sub
```

```
' Evento Click del botón btnproductos
Private Sub Button5_Click(sender As Object, e As EventArgs) Handles btnproductos.Click
    CerrarFormularioAnteriorEnPanel()
    Dim resporproductos As New resporproductos()
    resporproductos.TopLevel = False
    Panel9.Controls.Add(resporproductos)
    resporproductos.Dock = DockStyle.Fill
    resporproductos.Show()
End Sub

' Evento Click del botón btnusuarios
Private Sub Button2_Click(sender As Object, e As EventArgs) Handles btnusuarios.Click
    CerrarFormularioAnteriorEnPanel()
    Dim reporteusuario As New reporteusuario()
    reporteusuario.TopLevel = False
    Panel9.Controls.Add(reporteusuario)
    reporteusuario.Dock = DockStyle.Fill
```

```
    reporteusuario.Show()
End Sub

' Evento Click del botón btnmembresias
Private Sub Button3_Click(sender As Object, e As EventArgs) Handles btnmembresias.Click
    CerrarFormularioAnteriorEnPanel()
    Dim reportemembresia As New reportemembresia()
    reportemembresia.TopLevel = False
    Panel9.Controls.Add(reportemembresia)
    reportemembresia.Dock = DockStyle.Fill
    reportemembresia.Show()
End Sub

' Evento Click del botón btnventas
Private Sub Button4_Click(sender As Object, e As EventArgs) Handles btnventas.Click
    CerrarFormularioAnteriorEnPanel()
    Dim reportventas As New reportventas()
```

```
        reportventas.TopLevel = False
        Panel9.Controls.Add(reportventas)
        reportventas.Dock = DockStyle.Fill
        reportventas.Show()
    End Sub

    ' Evento Click del botón btndatos
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles btndatos.Click
        CerrarFormularioAnteriorEnPanel()
        Dim reportedatos As New reportedatos()
        reportedatos.TopLevel = False
        Panel9.Controls.Add(reportedatos)
        reportedatos.Dock = DockStyle.Fill
        reportedatos.Show()
    End Sub

End Class
```

23 CÓDIGO DE DATOS REPORTES

1. Importaciones

Las importaciones en la parte superior del código permiten el uso de funcionalidades específicas de MySQL y de los informes de Microsoft:

vb

```
Imports MySql.Data.MySqlClient

Imports Microsoft.Reporting.WinForms
```

2. Evento reportedatos_Load

Este evento se ejecuta cuando el formulario reportedatos se carga por primera vez. Llama al método CargarInforme para cargar todos los datos de la tabla mediciones y los muestra en el ReportViewer.

vb

```
Private Sub reportedatos_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    CargarInforme() ' Cargar todos los datos al iniciar
    Me.ReportViewer1.RefreshReport()
End Sub
```

3. Método CargarInforme

Este método es responsable de conectar con la base de datos, ejecutar la consulta SQL, y cargar los datos en el ReportViewer.

Pasos principales:

Conexión a la base de datos:

Se establece una cadena de conexión (cadenaConn) que contiene la información necesaria para conectarse a la base de datos MySQL.

vb

Dim cadenaConn As String = "server=localhost;database=gym;user id=root;password=root;"

Consulta SQL:

Se define una consulta SQL (querymediciones) que selecciona todos los registros de la tabla mediciones.

vb

Dim querymediciones As String = "SELECT * FROM mediciones"

Ejecución de la consulta:

Se utiliza un MySqlCommand para ejecutar la consulta y un MySqlDataAdapter para llenar un DataTable con los resultados.

vb

```
Dim cmd As New MySqlCommand(querymediciones, conn)
Dim medicionesda As New MySqlDataAdapter(cmd)
Dim medicionesdt As New DataTable()
medicionesda.Fill(medicionesdt)
```

Configuración del **ReportViewer**:

Se limpia cualquier fuente de datos existente en el ReportViewer, se crea una nueva ReportDataSource con el DataTable (medicionesdt), y se agrega como fuente de datos del ReportViewer.

```
ReportViewer1.LocalReport.DataSources.Clear()
Dim rp As New ReportDataSource("DataSet1", medicionesdt)
ReportViewer1.LocalReport.DataSources.Add(rp)
```

Manejo de errores:

En caso de que ocurra un error (por ejemplo, si no se puede conectar a la base de datos), se muestra un mensaje al usuario con información sobre el error.

```
        Catch ex As Exception
            MessageBox.Show(ex.Message & vbCrLf & "Error en la ejecución, no se conectó al servicio de MySQL", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
```

Código Completo con Comentarios

A continuación se muestra el código completo con comentarios explicativos:

```
Imports MySql.Data.MySqlClient
Imports Microsoft.Reporting.WinForms

Public Class reportedatos
    ' Evento que se ejecuta cuando el formulario se carga
    Private Sub reportedatos_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        CargarInforme() ' Cargar todos los datos al iniciar
        Me.ReportViewer1.RefreshReport()
    End Sub

    ' Método para cargar todos los datos de la tabla 'mediciones' en el ReportViewer
```

```
Private Sub CargarInforme()

    ' Cadena de conexión a la base de datos MySQL

    Dim cadenaConn As String = "server=localhost;database=gym;user
id=root;password=root;"


    ' Consulta SQL para seleccionar todos los datos de la tabla 'mediciones'

    Dim querymediciones As String = "SELECT * FROM mediciones"


    ' Uso de la conexión dentro de un bloque Using para asegurar su cierre adecuado

    Using conn As New MySqlConnection(cadenaConn)

      Try

        conn.Open()


        ' Crear un comando para ejecutar la consulta

        Dim cmd As New MySqlCommand(querymediciones, conn)


        ' Crear un adaptador y un DataTable para almacenar los datos de mediciones

        Dim medicionesda As New MySqlDataAdapter(cmd)
```

```
                Dim medicionesdt As New DataTable()
                medicionesda.Fill(medicionesdt)

                ' Limpiar las fuentes de datos existentes del ReportViewer
                ReportViewer1.LocalReport.DataSources.Clear()

                ' Agregar el DataTable como fuente de datos
                Dim rp As New ReportDataSource("DataSet1", medicionesdt)
                ReportViewer1.LocalReport.DataSources.Add(rp)

                ' Refrescar el informe para que se muestren los datos
                ReportViewer1.RefreshReport()
            Catch ex As Exception
                ' Manejo de errores: Mostrar mensaje de error si la conexión o la consulta fallan
                MessageBox.Show(ex.Message & vbCrLf & "Error en la ejecución, no se conectó al servicio de MySQL", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            End Try
        End Using
```

```
    End Sub

End Class
```

24 CÓDIGO PARA REPORTE MEMBRESIA

1. Importaciones

Las importaciones permiten el uso de funcionalidades específicas de MySQL y de los informes de Microsoft:

```
Imports MySql.Data.MySqlClient

Imports Microsoft.Reporting.WinForms
```

2. Evento reportemembresia_Load

Este evento se ejecuta cuando el formulario reportemembresia se carga por primera vez. Llama al método CargarInformeMembresia para cargar los datos de la tabla membresia y los muestra en el ReportViewer.

```
Private Sub reportemembresia_Load(sender As Object, e As EventArgs) Handles MyBase.Load

    CargarInformeMembresia()

    Me.ReportViewer1.RefreshReport()

End Sub
```

3. Método CargarInformeMembresia

Este método es responsable de conectarse a la base de datos, ejecutar la consulta SQL, y cargar los datos en el ReportViewer.

Pasos principales:

Conexión a la base de datos:

Se define una cadena de conexión (cadenaConn) que contiene la información para conectarse a la base de datos MySQL.

```
Dim cadenaConn As String = "server=localhost;database=gym;user id=root;password=root;"
```

Consulta SQL:

Se define una consulta SQL (queryMembresia) para seleccionar todos los registros de la tabla membresia.

```
Dim queryMembresia As String = "Select * from membresia"
```

Ejecución de la consulta:

Se crea un MySqlConnection y se abre la conexión a la base de datos.

Se utiliza un MySqlDataAdapter para llenar un DataTable con los resultados de la consulta.

```
Dim conn As New MySqlConnection(cadenaConn)

conn.Open()

Dim membresiada As New MySqlDataAdapter(queryMembresia, conn)

Dim membresiadt As New DataTable()
```

membresiada.Fill(membresiadt)

Configuración del **ReportViewer**:

Se limpia cualquier fuente de datos existente en el ReportViewer, se crea una nueva ReportDataSource con el DataTable (membresiadt), y se agrega como fuente de datos del ReportViewer.

ReportViewer1.LocalReport.DataSources.Clear()

Dim rp As New ReportDataSource("DataSet1", membresiadt)

ReportViewer1.LocalReport.DataSources.Add(rp)

Manejo de errores:

Si ocurre un error (por ejemplo, si no se puede conectar a la base de datos), se muestra un mensaje al usuario.

Catch ex As Exception

MessageBox.Show(ex.Message & vbCrLf & "Error en la ejecución, no se conectó al servicio de MySQL", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

Cierre de la conexión:

Se asegura que la conexión a la base de datos se cierra en el bloque Finally, independientemente de si ocurrió un error o no.

Finally

conn.Close()

```
End Try
```

Código Completo con Comentarios

A continuación se muestra el código completo con comentarios explicativos:

```
Imports MySql.Data.MySqlClient
Imports Microsoft.Reporting.WinForms


Public Class reportemembresia
    ' Evento que se ejecuta cuando el formulario se carga
    Private Sub reportemembresia_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        CargarInformeMembresia()
        Me.ReportViewer1.RefreshReport()
    End Sub


    ' Método para cargar todos los datos de la tabla 'membresia' en el ReportViewer
    Private Sub CargarInformeMembresia()
        ' Cadena de conexión a la base de datos MySQL
```

```
        Dim cadenaConn As String = "server=localhost;database=gym;user
id=root;password=root;"

        ' Consulta SQL para seleccionar todos los datos de la tabla 'membresia'
        Dim queryMembresia As String = "Select * from membresia"
        Dim conn As New MySqlConnection(cadenaConn)

        Try
            conn.Open()

            ' Crear un adaptador y un DataTable para almacenar los datos de membresía
            Dim membresiada As New MySqlDataAdapter(queryMembresia, conn)
            Dim membresiadt As New DataTable()
            membresiada.Fill(membresiadt)

            ' Limpiar las fuentes de datos existentes del ReportViewer
            ReportViewer1.LocalReport.DataSources.Clear()
```

```
            ' Agregar el DataTable como fuente de datos
            Dim rp As New ReportDataSource("DataSet1", membresiadt)
            ReportViewer1.LocalReport.DataSources.Add(rp)

            ' Refrescar el informe para que se muestren los datos
            ReportViewer1.RefreshReport()
        Catch ex As Exception
            ' Manejo de errores: Mostrar mensaje de error si la conexión o la consulta fallan
            MessageBox.Show(ex.Message & vbCrLf & "Error en la ejecución, no se conectó al servicio de MySQL", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        Finally
            ' Asegurarse de cerrar la conexión a la base de datos
            conn.Close()
        End Try
    End Sub
End Class
```

25 CÓDIGO PARA REPORTE USUARIOS

1. Importaciones

Estas importaciones permiten el uso de funcionalidades para conectar con MySQL y para trabajar con informes de Microsoft.

```
Imports MySql.Data.MySqlClient

Imports Microsoft.Reporting.WinForms
```

2. Evento reporteusuario_Load

Este evento se dispara cuando el formulario reporteusuario se carga por primera vez. Llama al método CargarInformeMembresia (debería ser renombrado para reflejar que está cargando datos de usuarios) para cargar los datos y actualizar el ReportViewer.

```
Private Sub reporteusuario_Load(sender As Object, e As EventArgs) Handles MyBase.Load

    CargarInformeMembresia()

    Me.ReportViewer1.RefreshReport()

End Sub
```

3. Método CargarInformeMembresia

Este método se encarga de conectarse a la base de datos, ejecutar una consulta para obtener datos de la tabla usuarios, y cargar esos datos en el ReportViewer.

Pasos principales:

Conexión a la base de datos:

Se define la cadena de conexión (cadenaConn) para conectar con la base de datos MySQL.

Dim cadenaConn As String = "server=localhost;database=gym;user id=root;password=root;"

Consulta SQL:

La consulta SQL (queryMembresia) está diseñada para seleccionar todos los registros de la tabla usuarios.

Dim queryMembresia As String = "Select * from usuarios"

Ejecución de la consulta:

Se abre la conexión a la base de datos y se utiliza un MySqlDataAdapter para llenar un DataTable con los resultados de la consulta.

Dim conn As New MySqlConnection(cadenaConn)

conn.Open()

Dim membresiada As New MySqlDataAdapter(queryMembresia, conn)

Dim membresiadt As New DataTable()

membresiada.Fill(membresiadt)

Configuración del **ReportViewer**:

Se limpia cualquier fuente de datos existente en el ReportViewer, se agrega una nueva ReportDataSource con el DataTable (membresiadt), y se configura como la fuente de datos del ReportViewer.

ReportViewer1.LocalReport.DataSources.Clear()

Dim rp As New ReportDataSource("DataSet1", membresiadt)

ReportViewer1.LocalReport.DataSources.Add(rp)

Manejo de errores:

Si ocurre un error durante la conexión o la consulta, se muestra un mensaje de error al usuario.

Catch ex As Exception

MessageBox.Show(ex.Message & vbCrLf & "Error en la ejecución, no se conectó al servicio de MySQL", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

Cierre de la conexión:

Se asegura que la conexión a la base de datos se cierra en el bloque Finally, independientemente de si ocurrió un error.

Finally

conn.Close()

End Try

Código Completo con Comentarios

Aquí está el código completo con comentarios para mayor claridad:

Imports MySql.Data.MySqlClient

Imports Microsoft.Reporting.WinForms

```
Public Class reporteusuario

    ' Evento que se ejecuta cuando el formulario se carga

    Private Sub reporteusuario_Load(sender As Object, e As EventArgs) Handles MyBase.Load

        CargarInformeUsuario() ' Llamar al método para cargar los datos de usuarios

        Me.ReportViewer1.RefreshReport() ' Actualizar el ReportViewer

    End Sub


    ' Método para cargar todos los datos de la tabla 'usuarios' en el ReportViewer

    Private Sub CargarInformeUsuario()

        ' Cadena de conexión a la base de datos MySQL

        Dim cadenaConn As String = "server=localhost;database=gym;user
id=root;password=root;"


        ' Consulta SQL para seleccionar todos los datos de la tabla 'usuarios'

        Dim queryUsuarios As String = "Select * from usuarios"

        Dim conn As New MySqlConnection(cadenaConn)
```

```
Try
    conn.Open()

    ' Crear un adaptador y un DataTable para almacenar los datos de usuarios
    Dim usuariosda As New MySqlDataAdapter(queryUsuarios, conn)
    Dim usuariosdt As New DataTable()
    usuariosda.Fill(usuariosdt)

    ' Limpiar las fuentes de datos existentes del ReportViewer
    ReportViewer1.LocalReport.DataSources.Clear()

    ' Agregar el DataTable como fuente de datos
    Dim rp As New ReportDataSource("DataSet1", usuariosdt)
    ReportViewer1.LocalReport.DataSources.Add(rp)

    ' Refrescar el informe para que se muestren los datos
    ReportViewer1.RefreshReport()
```

```
        Catch ex As Exception
            ' Manejo de errores: Mostrar mensaje de error si la conexión o la consulta fallan
            MessageBox.Show(ex.Message & vbCrLf & "Error en la ejecución, no se conectó al servicio de MySQL", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        Finally
            ' Asegurarse de cerrar la conexión a la base de datos
            conn.Close()
        End Try
    End Sub
End Class
```

26 CÓDIGO PARA REPORTES DE VENTAS

1. Importaciones

Estas importaciones son necesarias para conectarse a la base de datos MySQL y para trabajar con el ReportViewer.

```
Imports MySql.Data.MySqlClient
Imports Microsoft.Reporting.WinForms
```

2. Evento reportventas_Load

Este evento se ejecuta cuando el formulario reportventas se carga por primera vez. Llama al método CargarInformeMembresia (debería ser renombrado para reflejar que está cargando datos de ventas) para cargar los datos y actualizar el ReportViewer.

```
Private Sub reportventas_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    CargarInformeVentas() ' Llamar al método para cargar los datos de ventas
    Me.ReportViewer1.RefreshReport() ' Actualizar el ReportViewer
End Sub
```

3. Método CargarInformeMembresia

Este método se encarga de conectarse a la base de datos, ejecutar una consulta para obtener datos de la tabla ventas, y cargar esos datos en el ReportViewer.

Pasos principales:

Conexión a la base de datos:

Se define la cadena de conexión (cadenaConn) para conectar con la base de datos MySQL.

```
Dim cadenaConn As String = "server=localhost;database=gym;user id=root;password=root;"
```

Consulta SQL:

La consulta SQL (queryventas) está diseñada para seleccionar todos los registros de la tabla ventas.

```
Dim queryventas As String = "Select * from ventas"
```

Ejecución de la consulta:

Se abre la conexión a la base de datos y se utiliza un MySqlDataAdapter para llenar un DataTable con los resultados de la consulta.

```
Dim conn As New MySqlConnection(cadenaConn)

conn.Open()

Dim ventasda As New MySqlDataAdapter(queryventas, conn)

Dim ventasdt As New DataTable()

ventasda.Fill(ventasdt)
```

Configuración del **ReportViewer**:

Se limpia cualquier fuente de datos existente en el ReportViewer, se agrega una nueva ReportDataSource con el DataTable (ventasdt), y se configura como la fuente de datos del ReportViewer.

```
ReportViewer1.LocalReport.DataSources.Clear()

Dim rp As New ReportDataSource("DataSet1", ventasdt)

ReportViewer1.LocalReport.DataSources.Add(rp)
```

Manejo de errores:

Si ocurre un error durante la conexión o la consulta, se muestra un mensaje de error al usuario.

Catch ex As Exception

MessageBox.Show(ex.Message & vbCrLf & "Error en la ejecución, no se conectó al servicio de MySQL", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)

Cierre de la conexión:

Se asegura que la conexión a la base de datos se cierra en el bloque Finally, independientemente de si ocurrió un error.

Finally

conn.Close()

End Try

Código Completo con Comentarios

Aquí está el código completo con comentarios actualizados:

Imports MySql.Data.MySqlClient

Imports Microsoft.Reporting.WinForms

Public Class reportventas

' Evento que se ejecuta cuando el formulario se carga

Private Sub reportventas_Load(sender As Object, e As EventArgs) Handles MyBase.Load

CargarInformeVentas() ' Llamar al método para cargar los datos de ventas

```
        Me.ReportViewer1.RefreshReport() ' Actualizar el ReportViewer
    End Sub

    ' Método para cargar todos los datos de la tabla 'ventas' en el ReportViewer
    Private Sub CargarInformeVentas()
        ' Cadena de conexión a la base de datos MySQL
        Dim cadenaConn As String = "server=localhost;database=gym;user
id=root;password=root;"

        ' Consulta SQL para seleccionar todos los datos de la tabla 'ventas'
        Dim queryventas As String = "Select * from ventas"
        Dim conn As New MySqlConnection(cadenaConn)

        Try
            conn.Open()

            ' Crear un adaptador y un DataTable para almacenar los datos de ventas
            Dim ventasda As New MySqlDataAdapter(queryventas, conn)
```

```
            Dim ventasdt As New DataTable()
            ventasda.Fill(ventasdt)

            ' Limpiar las fuentes de datos existentes del ReportViewer
            ReportViewer1.LocalReport.DataSources.Clear()

            ' Agregar el DataTable como fuente de datos
            Dim rp As New ReportDataSource("DataSet1", ventasdt)
            ReportViewer1.LocalReport.DataSources.Add(rp)

            ' Refrescar el informe para que se muestren los datos
            ReportViewer1.RefreshReport()
        Catch ex As Exception
            ' Manejo de errores: Mostrar mensaje de error si la conexión o la consulta fallan
            MessageBox.Show(ex.Message & vbCrLf & "Error en la ejecución, no se conectó al servicio de MySQL", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        Finally
            ' Asegurarse de cerrar la conexión a la base de datos
```

```
            conn.Close()
        End Try
    End Sub
End Class
```

27 CÓDIGO PARA REPORTE DE PRODUCTOS

1. Importaciones

Estas importaciones son necesarias para conectarse a la base de datos MySQL y para trabajar con el ReportViewer.

```
Imports MySql.Data.MySqlClient
Imports Microsoft.Reporting.WinForms
```

2. Evento resporproductos_Load

Este evento se ejecuta cuando el formulario resporproductos se carga por primera vez. Llama al método CargarInformeProductos (debería ser renombrado para reflejar que está cargando datos de productos) para cargar los datos y actualizar el ReportViewer.

```
Private Sub resporproductos_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    CargarInformeProductos() ' Llamar al método para cargar los datos de productos
    Me.ReportViewer1.RefreshReport() ' Actualizar el ReportViewer
```

End Sub

3. Método CargarInformeMembresia

Este método se encarga de conectarse a la base de datos, ejecutar una consulta para obtener datos de la tabla productos, y cargar esos datos en el ReportViewer.

Pasos principales:

Conexión a la base de datos:

Se define la cadena de conexión (cadenaConn) para conectar con la base de datos MySQL.

```
Dim cadenaConn As String = "server=localhost;database=gym;user id=root;password=root;"
```

Consulta SQL:

La consulta SQL (queryproductos) está diseñada para seleccionar todos los registros de la tabla productos.

```
Dim queryproductos As String = "Select * from productos"
```

Ejecución de la consulta:

Se abre la conexión a la base de datos y se utiliza un MySqlDataAdapter para llenar un DataTable con los resultados de la consulta.

```
Dim conn As New MySqlConnection(cadenaConn)

conn.Open()

Dim productosda As New MySqlDataAdapter(queryproductos, conn)
```

```
Dim productosdt As New DataTable()

productosda.Fill(productosdt)
```

Configuración del **ReportViewer**:

Se limpia cualquier fuente de datos existente en el ReportViewer, se agrega una nueva ReportDataSource con el DataTable (productosdt), y se configura como la fuente de datos del ReportViewer.

```
ReportViewer1.LocalReport.DataSources.Clear()

Dim rp As New ReportDataSource("DataSet1", productosdt)

ReportViewer1.LocalReport.DataSources.Add(rp)
```

Manejo de errores:

Si ocurre un error durante la conexión o la consulta, se muestra un mensaje de error al usuario.

```
Catch ex As Exception

    MessageBox.Show(ex.Message & vbCrLf & "Error en la ejecución, no se conectó al servicio de MySQL", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
```

Cierre de la conexión:

Se asegura que la conexión a la base de datos se cierra en el bloque Finally, independientemente de si ocurrió un error.

```
Finally
    conn.Close()
End Try
```

Código Completo con Comentarios

Aquí está el código completo con comentarios actualizados:

```
Imports MySql.Data.MySqlClient
Imports Microsoft.Reporting.WinForms

Public Class resporproductos
    ' Evento que se ejecuta cuando el formulario se carga
    Private Sub resporproductos_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        CargarInformeProductos() ' Llamar al método para cargar los datos de productos
        Me.ReportViewer1.RefreshReport() ' Actualizar el ReportViewer
    End Sub

    ' Método para cargar todos los datos de la tabla 'productos' en el ReportViewer
    Private Sub CargarInformeProductos()
```

```
' Cadena de conexión a la base de datos MySQL
Dim cadenaConn As String = "server=localhost;database=gym;user id=root;password=root;"

' Consulta SQL para seleccionar todos los datos de la tabla 'productos'
Dim queryproductos As String = "Select * from productos"
Dim conn As New MySqlConnection(cadenaConn)

Try
    conn.Open()

    ' Crear un adaptador y un DataTable para almacenar los datos de productos
    Dim productosda As New MySqlDataAdapter(queryproductos, conn)
    Dim productosdt As New DataTable()
    productosda.Fill(productosdt)

    ' Limpiar las fuentes de datos existentes del ReportViewer
    ReportViewer1.LocalReport.DataSources.Clear()
```

```
            ' Agregar el DataTable como fuente de datos
            Dim rp As New ReportDataSource("DataSet1", productosdt)
            ReportViewer1.LocalReport.DataSources.Add(rp)

            ' Refrescar el informe para que se muestren los datos
            ReportViewer1.RefreshReport()
        Catch ex As Exception
            ' Manejo de errores: Mostrar mensaje de error si la conexión o la consulta fallan
            MessageBox.Show(ex.Message & vbCrLf & "Error en la ejecución, no se conectó al servicio de MySQL", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        Finally
            ' Asegurarse de cerrar la conexión a la base de datos
            conn.Close()
        End Try
    End Sub
End Class
```