

¡BIENVENIDOS A GOATYM! 🦸

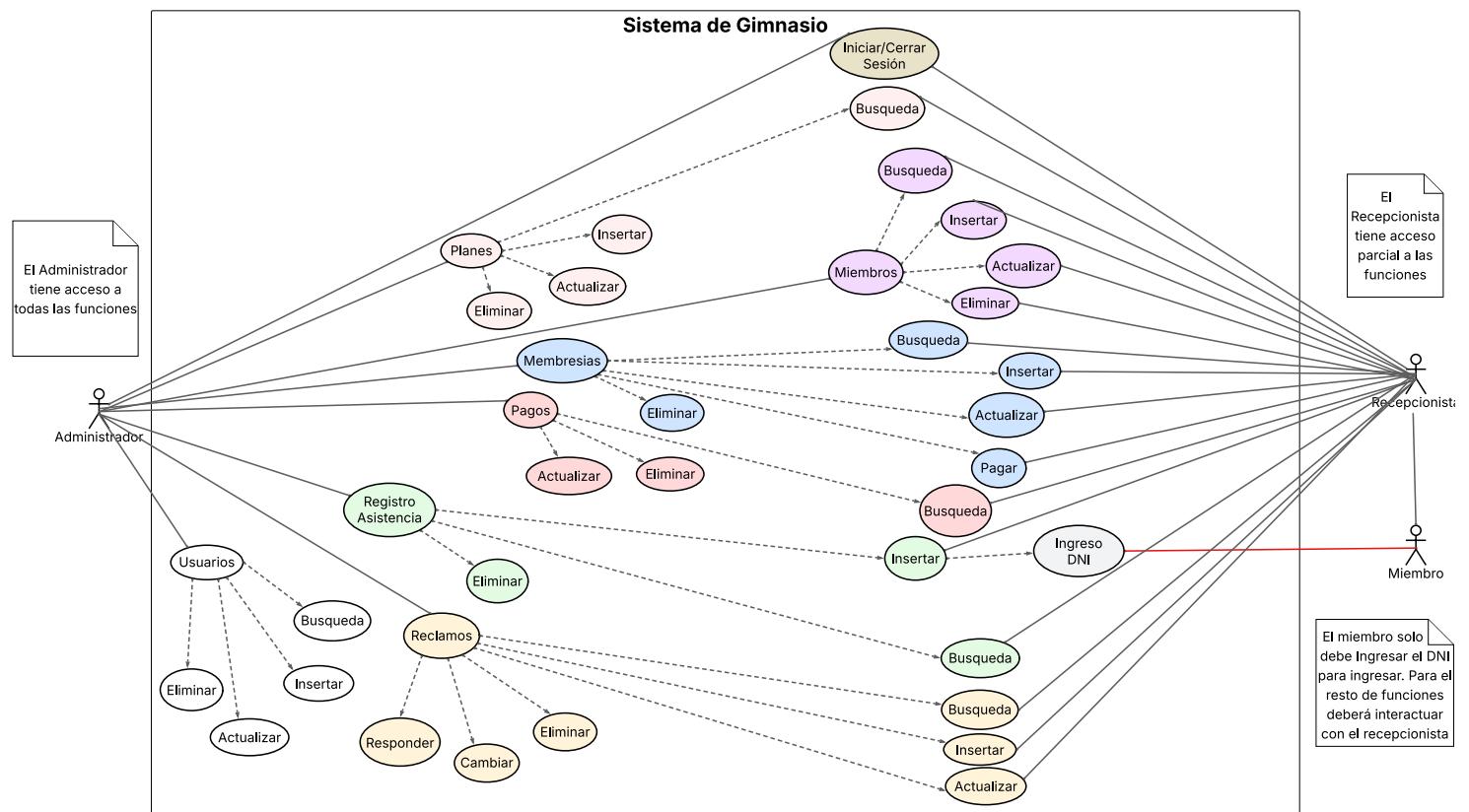
EQUIPO DE DESARROLLO: PROGOAT

- Casas Alan Fernando - EISI 1244
- Diaz Carlos Emanuel - EISI 1225
- Gianello Ramiro Valentín - EISI 1333

Función Principal

Goatym es un sistema diseñado para gestión de miembros, planes de membresía, pagos, control de asistencia y seguimiento de reclamos para un gimnasio. Además, permite la creación de usuarios con dos roles principales: Administrador y Recepcionista.

Funciones detalladas del sistema



Gestión de Planes

Función	Roles
Visualizar un listado completo de los planes. Este listado incluye: nombre del plan, descripción, duración, precio, fecha de creación y fecha de última modificación. Los planes se ordenan por la fecha de última modificación (más reciente primero) y se muestra la cantidad total de registros.	Administrador, Recepcionista

Función	Roles
Realizar búsquedas de planes por nombre, precio o duración.	Administrador, Recepcionista
Insertar nuevos planes en el sistema, ingresando atributos obligatorios (nombre, duración, precio) y optionales (descripción).	Administrador
Actualizar los datos de planes existentes.	Administrador
Eliminar planes. Esta acción no se permitirá si el plan está asociado a alguna membresía activa.	Administrador

Gestión de miembros

Función	Roles
Visualizar un listado completo de los miembros. Este listado incluye: DNI, nombre, apellido, género, teléfono, email, fecha de registro y fecha de última modificación. Los miembros se ordenan por la fecha de última modificación (más reciente primero) y se muestra la cantidad total de registros.	Administrador, Recepcionista
Realizar búsquedas de miembros por nombre o DNI.	Administrador, Recepcionista
Insertar nuevos miembros en el sistema, ingresando datos obligatorios (DNI, nombre y apellido) y optionales (género, teléfono, email). Posteriormente, es posible asignar y procesar el pago de una membresía para el miembro registrado. Estas operaciones pueden cancelarse en cualquier momento durante el proceso.	Administrador, Recepcionista
Modificar los datos de miembros existentes.	Administrador, Recepcionista
Eliminar miembros. Esta acción no se permitirá si el miembro tiene una membresía asignada.	Administrador, Recepcionista

Gestión de Membresías

Funcion	Roles
Visualizar un listado completo de las membresías. Este listado incluye: DNI, nombre y apellido del miembro; nombre, precio y duración del plan; fecha de inicio, fecha de fin y estado de la membresía; fecha de registro y fecha de última	Administrador, Recepcionista

Función	Roles
modificación de la membresía. Las membresías se ordenan por la fecha de última modificación (más reciente primero) y se muestra la cantidad total de registros.	
Realizar búsquedas de membresías por DNI del miembro, tipo de plan, o estado (activas/inactivas).	Administrador, Recepcionista
Insertar nuevas membresias al sistema, seleccionando el miembro y plan.	Administrador, Recepcionista
Confirmar el pago de una membresía, seleccionando el método de pago (obligatorio) y añadiendo opcionalmente el número de comprobante y notas adicionales. Luego del pago, el sistema cambiara el estado de la membresía a activo y actualizara la fecha de vencimiento. Cuando esta llegue a su fin la membresía volverá a estar inactiva.	Administrador, Recepcionista
Actualizar la membresía de un miembro, permitiendo el cambio de plan.	Administrador, Recepcionista
Eliminar la membresia de un miembro.	Administrador

Gestión de Pagos

Función	Roles
Visualizar un listado de todos los pagos registrados. Este listado detalla: nombre, apellido y DNI del miembro; nombre del plan, monto pagado, método de pago, número de comprobante, fecha de pago y usuario encargado del registro. Los pagos se ordenan por fecha de pago (más reciente primero) y se muestra la cantidad total de registros, así como el total de ingresos.	Administrador, Recepcionista
Realizar búsquedas de pagos utilizando filtros por rango de fechas (inicio y fin), DNI del miembro, plan, monto o método de pago.	Administrador, Recepcionista
Actualizar los datos de un pago existente.	Administrador
Eliminar registros de pagos	Administrador

Gestión de Asistencia

Función	Roles
<p>Registrar la asistencia mediante el ingreso del DNI del miembro. El sistema validará el DNI:</p> <ul style="list-style-type: none"> - Si el DNI no se encuentra, se informará. - Si el DNI se encuentra pero la membresía está vencida, el ingreso será fallido y no se permitirá el acceso. - Si el DNI se encuentra y la membresía está activa, el ingreso será exitoso y se permitirá el acceso. En los casos de DNI encontrado (ingreso fallido o exitoso), se mostrará un resumen del estado de las membresías del miembro, incluyendo: DNI, apellido, nombre, plan, duración, fecha de inicio, fecha de fin, estado y días restantes. 	Administrador, Recepcionista
<p>Visualizar el registro de asistencia, que es un listado de todos los intentos de ingreso (exitosos y fallidos). Este listado incluye: DNI, nombre y apellido del miembro, fecha y resultado del ingreso. Los registros se ordenan por fecha de ingreso (más reciente primero) y se muestra la cantidad total de ingresos.</p>	Administrador, Recepcionista
<p>Realizar búsquedas en el registro de asistencia por DNI del miembro o por rango de fechas.</p>	Administrador, Recepcionista
<p>Eliminar registro de asistencia</p>	Administrador

Gestión de Reclamos

Función	Roles
<p>Visualizar un listado completo de los reclamos. Este listado incluye: tipo de reclamo, descripción, fecha de envío, estado (pendiente/resuelto), respuesta, fecha de respuesta y DNI del miembro (si aplica). Los reclamos se ordenan por fecha de envío (más reciente primero) y se muestra la cantidad total de reclamos.</p>	Administrador, Recepcionista
<p>Realizar búsquedas de reclamos filtrando por estado (por defecto "pendiente", o "resuelto")</p>	Administrador, Recepcionista
<p>Registrar un nuevo reclamo, ingresando tipo y descripción (obligatorios), y DNI del miembro (opcional).</p>	Administrador, Recepcionista
<p>Actualizar los datos de un reclamo existente.</p>	Administrador, Recepcionista
<p>Eliminar un reclamo</p>	Administrador

Función	Roles
Cambiar el estado de un reclamo (de pendiente a resuelto, o viceversa).	Administrador
Responder un reclamo, detallando la resolución del mismo en el sistema.	Administrador

Gestión de Usuarios (Administradores y Recepcionistas)

Función	Roles
<p>Iniciar sesión en el sistema utilizando nombre de usuario y contraseña.</p> <ul style="list-style-type: none"> - Un inicio de sesión exitoso redirige a una pantalla de bienvenida. - Rol Administrador: Acceso completo a todas las funcionalidades del sistema. Las pestañas visibles son: Planes, Miembros, Membresías, Pagos, Asistencia, Registro de Asistencias, Reclamos y Usuarios. También se muestra el nombre del usuario logueado con la opción de "Cerrar Sesión". - Rol Recepcionista: Acceso parcial a las funcionalidades, sin acceso a la pestaña "Usuarios". 	Administrador, Recepcionista
Cerrar la sesión actual del usuario en el sistema.	Administrador, Recepcionista
Visualizar un listado completo de los usuarios del sistema. Este listado incluye: nombre de usuario, hash de contraseña, nombre completo, email, rol (administrador o recepcionista), fecha de creación y fecha de última modificación. Los usuarios se ordenan por la fecha de última modificación (más reciente primero) y se muestra la cantidad total de usuarios.	Administrador
Realizar búsquedas de usuarios por nombre	Administrador
Insertar nuevos usuarios en el sistema, ingresando datos obligatorios (nombre de usuario, contraseña, nombre completo, rol) y opcionales (email)	Administrador
Modificar los datos de usuarios existentes, incluyendo la opción de cambiar la contraseña	Administrador
Eliminar usuarios del sistema.	Administrador

Alcance y límites

1. Es una aplicación de escritorio diseñada para la gestión simplificada de un gimnasio.
2. El acceso de los miembros se registra mediante su DNI; el sistema no está diseñado para lectores de huellas digitales u otros métodos de identificación biométrica.

3. El sistema está concebido para operar en un único puesto de check-in y para la gestión de una sola sucursal de gimnasio.
4. El sistema registra únicamente los pagos correspondientes a membresías; no incluye una gestión financiera integral (ej. otros ingresos, egresos, contabilidad).
5. Los roles definidos en el sistema son: Administrador, Recepcionista y Miembro. No se contemplan roles adicionales como entrenadores u otro personal.
6. El sistema no incluye funcionalidades para la gestión de inventario, ni para procesos de compra o venta de productos.
7. Las operaciones en los listados (ej. selección para edición o eliminación) se realizan de forma individual (registro por registro); no se admite selección múltiple.
8. El sistema no cuenta con la funcionalidad de exportar listados a PDF u otros formatos de archivo.

Namespace Gimnasio.Datos

Classes

[ConexionBase](#)

Clase base para la gestión de la conexión y operaciones con la base de datos MySQL. Proporciona métodos genéricos para ejecutar consultas y comandos SQL.

Beneficios de utilizar [ExecuteQuery\(string, Dictionary<string, object>\)](#) y [ExecuteNonQuery\(string, Dictionary<string, object>\)](#):

- Centralizan y simplifican el acceso a datos, evitando duplicación de código.
- Permiten el uso de parámetros, ayudando a prevenir inyecciones SQL.
- Manejan automáticamente la apertura y cierre de conexiones.
- Facilitan el mantenimiento y la escalabilidad del acceso a la base de datos.

Todas las operaciones de acceso a datos están envueltas en bloques Try...Catch.

- Si ocurre una excepción durante la ejecución de una consulta o comando SQL, el error se registra en el archivo de log mediante [Log\(string, Exception\)](#).
- Tras registrar el error, la excepción se propaga nuevamente mediante Throw New Exception(ex.Message).

[DAsistencia](#)

Clase de acceso a datos para la gestión de asistencias. Hereda de [ConexionBase](#) y utiliza la entidad [Asistencia](#). Proporciona métodos CRUD y de búsqueda para la tabla asistencia y la vista `vista_asistencia`.

La vista consolida la información relevante de los registros de asistencias, permitiendo consultar en una sola consulta datos de la asistencia, el miembro, la membresía y el plan asociado. Realiza LEFT JOIN entre la asistencia y las demás tablas, permitiendo obtener la información de asistencia incluso si los datos de miembro, membresía o plan no están presentes.

```
VIEW `vista_asistencia` AS
  SELECT
    `a`.`id_asistencia` AS `id_asistencia`,
    `a`.`id_miembro` AS `id_miembro`,
    `a`.`id_membresia_valida` AS `id_membresia`,
    `m`.`dni` AS `dni_miembro`,
    `m`.`nombre` AS `nombre_miembro`,
    `m`.`apellido` AS `apellido_miembro`,
    `a`.`fecha_hora_checkin` AS `fecha_ingreso`,
    `a`.`resultado` AS `resultado`,
    `pm`.`nombre_plan` AS `nombre_plan_membresia`
```

```

FROM
  (((`asistencia` `a`
    LEFT JOIN `miembros` `m` ON ((`a`.`id_miembro` = `m`.`id_miembro`)))
    LEFT JOIN `membresias_miembro` `mm` ON ((`a`.`id_membresia_valida` =
= `mm`.`id_membresia`)))
      LEFT JOIN `planes_membresia` `pm` ON ((`mm`.`id_plan` = `pm`.`id_plan`)))
    ORDER BY `a`.`fecha_hora_checkin` DESC

```

Los diccionarios se utilizan para asociar los parametros de la consulta con los parametros del metodo

DMembresias

Clase de acceso a datos para la gestión de membresías en el sistema de gimnasio. Hereda de [ConexionBase](#) y utiliza la entidad [Membresias](#). Proporciona métodos CRUD y de búsqueda para la tabla membresias_miembro y la vista vista_membresias.

La vista_membresias es una vista SQL que consolida información relevante de las membresías, uniendo datos de las tablas membresias_miembro, miembros y planes_membresia.

```

VIEW `vista_membresias` AS
  SELECT
    `mm`.`id_membresia` AS `id_membresia`,
    `mm`.`id_miembro` AS `id_miembro`,
    `mm`.`id_plan` AS `id_plan`,
    `m`.`dni` AS `dni_miembro`,
    `m`.`apellido` AS `apellido_miembro`,
    `m`.`nombre` AS `nombre_miembro`,
    `p`.`nombre_plan` AS `nombre_plan`,
    `p`.`precio` AS `precio_plan`,
    `p`.`duracion_dias` AS `duracion_dias_plan`,
    `mm`.`fecha_inicio` AS `fecha_inicio`,
    `mm`.`fecha_fin` AS `fecha_fin`,
    `mm`.`estado_membresia` AS `estado_membresia`,
    `mm`.`fecha_registro` AS `fecha_registro`,
    `mm`.`ultima_modificacion` AS `ultima_modificacion`

  FROM
    ((`membresias_miembro` `mm`
      JOIN `miembros` `m` ON ((`mm`.`id_miembro` = `m`.`id_miembro`)))
        JOIN `planes_membresia` `p` ON ((`mm`.`id_plan` = `p`.`id_plan`)))
      ORDER BY `mm`.`ultima_modificacion` DESC

```

Los diccionarios se utilizan para asociar los parametros de la consulta con los parametros del metodo

DMiembros

Clase de acceso a datos para la gestión de miembros en el sistema de gimnasio. Hereda de [ConexionBase](#) y utiliza la entidad [Miembros](#). Proporciona métodos CRUD y de búsqueda para la tabla miembros.

Los diccionarios se utilizan para asociar los parametros de la consulta con los parametros del metodo

[DPagos](#)

Clase de acceso a datos para la gestión de pagos en el sistema de gimnasio. Hereda de [ConexionBase](#) y utiliza la entidad [Pagos](#). Proporciona métodos CRUD y de búsqueda para la tabla pagos y la vista vista_pagos.

La vista consolida la información relevante de los registros de pagos, Permite consultar en una sola consulta los datos del pago, el miembro, el plan, el usuario que registró el pago. Realiza LEFT JOIN entre pagos y las demás tablas, permitiendo obtener la información de pagos incluso si los datos de miembro, membresía, plan o usuario no están presentes.

```
VIEW `vista_pagos` AS
  SELECT
    `p`.`id_pago` AS `id_pago`,
    `p`.`id_membresia` AS `id_membresia`,
    `p`.`id_usuario_registro` AS `id_usuario_registro`,
    `m`.`apellido` AS `apellido_miembro`,
    `m`.`nombre` AS `nombre_miembro`,
    `m`.`dni` AS `dni_miembro`,
    `pm`.`nombre_plan` AS `nombre_plan`,
    `p`.`monto_pagado` AS `monto`,
    `p`.`metodo_pago` AS `metodo`,
    `p`.`numero_comprobante` AS `comprobante`,
    `p`.`notas` AS `notas`,
    `p`.`fecha_pago` AS `fecha_pago`,
    `us`.`nombre_completo` AS `nombre_usuario`
  FROM
    ((((`pagos` `p`
      LEFT JOIN `membresias_miembro` `mm` ON ((`p`.`id_membresia` =
      `mm`.`id_membresia`)))
      LEFT JOIN `miembros` `m` ON ((`mm`.`id_miembro` = `m`.`id_miembro`)))
      LEFT JOIN `planes_membresia` `pm` ON ((`mm`.`id_plan` = `pm`.`id_plan`)))
      LEFT JOIN `usuarios_sistema` `us` ON ((`p`.`id_usuario_registro` =
      `us`.`id_usuario`)))
    ORDER BY `p`.`fecha_pago` DESC
```

Los diccionarios se utilizan para asociar los parametros de la consulta con los parametros del metodo

[DPlanes](#)

Clase de acceso a datos para la gestión de planes de membresía. Hereda de [ConexionBase](#) y utiliza la entidad [Planes](#). Proporciona métodos CRUD y de búsqueda para la tabla `planes_membresia`.

Los diccionarios se utilizan para asociar los parámetros de la consulta con los parámetros del método

[DReclamos](#)

Clase de acceso a datos para la gestión de reclamos en el sistema de gimnasio. Hereda de [ConexionBase](#) y utiliza la entidad [Reclamos](#). Proporciona métodos CRUD y de búsqueda para la tabla `reclamos` y la vista `vista_reclamos`.

La vista consolida la información relevante de los reclamos, permitiendo consultar en una sola consulta datos del reclamo y el miembro que lo realizó. Realiza LEFT JOIN entre la `reclamos` y `miembros`, permitiendo obtener la información de reclamo incluso si los datos de miembro no están presentes.

```
VIEW `vista_reclamos` AS
SELECT
    `r`.`id_reclamos` AS `id_reclamos`,
    `r`.`tipo` AS `tipo`,
    `r`.`descripcion` AS `descripcion`,
    `r`.`fecha_envio` AS `fecha_envio`,
    `r`.`estado` AS `estado`,
    `r`.`respuesta` AS `respuesta`,
    `r`.`fecha_respuesta` AS `fecha_respuesta`,
    `m`.`dni` AS `dni_miembro`
FROM
    (`reclamos` `r`
    LEFT JOIN `miembros` `m` On ((`r`.`id_miembro` = `m`.`id_miembro`)))
ORDER BY `r`.`fecha_envio` DESC
```

Los diccionarios se utilizan para asociar los parámetros de la consulta con los parámetros del método

[DUuarios](#)

Clase de acceso a datos para la gestión de usuarios en el sistema de gimnasio. Hereda de [ConexionBase](#) y utiliza la entidad [Usuarios](#). Proporciona métodos CRUD y de búsqueda para la tabla `usuarios_sistema` y la vista `vista_usuarios`.

La vista consolida la información relevante de los registros de usuarios, permitiendo consultar en una sola consulta datos de los usuarios y sus roles. Realiza JOIN entre la tabla de usuarios y la tabla de roles, permitiendo obtener la información de usuario junto con su rol asociado.

```
VIEW `vista_usuarios` AS
SELECT
```

```
`u`.`id_usuario` AS `id_usuario`,
`u`.`username` AS `username`,
`u`.`password_hash` AS `password_hash`,
`u`.`nombre_completo` AS `nombre_completo`,
`u`.`email` AS `email`,
`r`.`nombre_rol` AS `nombre_rol`,
`u`.`fecha_creacion` AS `fecha_creacion`,
`u`.`ultima_modificacion` AS `ultima_modificacion`

FROM
(`usuarios_sistema` `u`
JOIN `roles` `r` ON ((`u`.`id_rol` = `r`.`id_rol`)))
ORDER BY `u`.`ultima_modificacion` DESC
```

Los diccionarios se utilizan para asociar los parámetros de la consulta con los parámetros del método.

Class ConexionBase

Namespace: [Gimnasio.Datos](#)

Clase base para la gestión de la conexión y operaciones con la base de datos MySQL. Proporciona métodos genéricos para ejecutar consultas y comandos SQL.

Beneficios de utilizar [ExecuteQuery\(string, Dictionary<string, object>\)](#) y [ExecuteNonQuery\(string, Dictionary<string, object>\)](#):

- Centralizan y simplifican el acceso a datos, evitando duplicación de código.
- Permiten el uso de parámetros, ayudando a prevenir inyecciones SQL.
- Manejan automáticamente la apertura y cierre de conexiones.
- Facilitan el mantenimiento y la escalabilidad del acceso a la base de datos.

Todas las operaciones de acceso a datos están envueltas en bloques Try...Catch.

- Si ocurre una excepción durante la ejecución de una consulta o comando SQL, el error se registra en el archivo de log mediante [Log\(string, Exception\)](#).
- Tras registrar el error, la excepción se propaga nuevamente mediante Throw New Exception(ex.Message).

```
public class ConexionBase
```

Inheritance

[object](#) ↗ ← ConexionBase

Derived

[DAistencia](#), [DMembresias](#), [DMIembros](#), [DPagos](#), [DPlanes](#), [DReclamos](#), [DUuarios](#)

Fields

connectionString

Cadena de conexión a la base de datos MySQL.

```
private string connectionString
```

Field Value

[string](#)

Methods

ExecuteNonQuery(string, Dictionary<string, object>)

Ejecuta un comando SQL sobre la base de datos MySQL que no retorna resultados (por ejemplo, INSERT, UPDATE o DELETE). Proceso:

1. Abre una conexión a la base de datos utilizando la cadena de conexión definida.
2. Crea un comando SQL con la instrucción proporcionada y agrega los parámetros especificados (si existen).
3. Ejecuta el comando mediante MySqlCommand.ExecuteNonQuery para realizar la operación solicitada.
4. No retorna ningún valor, ya que está orientado a operaciones que modifican datos pero no devuelven resultados.

```
public void ExecuteNonQuery(string query, Dictionary<string, object> parameters)
```

Parameters

query [string](#)

Comando SQL a ejecutar (INSERT, UPDATE, DELETE, etc.).

parameters [Dictionary](#)<[string](#), [object](#)>

Diccionario de parámetros para el comando (puede ser Nothing).

ExecuteQuery(string, Dictionary<string, object>)

Ejecuta una consulta SQL sobre la base de datos MySQL y retorna los resultados en un DataTable.

Proceso:

1. Abre una conexión a la base de datos utilizando la cadena de conexión definida.
2. Crea un comando SQL con la consulta proporcionada y agrega los parámetros especificados (si existen).
3. Utiliza un MySqlDataAdapter para ejecutar la consulta y llenar un DataTable con los resultados.
4. Retorna el DataTable con los datos obtenidos.

```
public DataTable ExecuteQuery(string query, Dictionary<string, object> parameters)
```

Parameters

query [string](#)

Consulta SQL a ejecutar.

parameters [Dictionary](#)<[string](#), [object](#)>

Diccionario de parámetros para la consulta (puede ser Nothing).

Returns

[DataTable](#)

DataTable con los resultados de la consulta.

Class DAsistencia

Namespace: [Gimnasio.Datos](#)

Clase de acceso a datos para la gestión de asistencias. Hereda de [ConexionBase](#) y utiliza la entidad [Asistencia](#). Proporciona métodos CRUD y de búsqueda para la tabla asistencia y la vista vista_asistencia.

La vista consolida la información relevante de los registros de asistencias, permitiendo consultar en una sola consulta datos de la asistencia, el miembro, la membresía y el plan asociado. Realiza LEFT JOIN entre la asistencia y las demás tablas, permitiendo obtener la información de asistencia incluso si los datos de miembro, membresía o plan no están presentes.

```
VIEW `vista_asistencia` AS
  SELECT
    `a`.`id_asistencia` AS `id_asistencia`,
    `a`.`id_miembro` AS `id_miembro`,
    `a`.`id_membresia_valida` AS `id_membresia`,
    `m`.`dni` AS `dni_miembro`,
    `m`.`nombre` AS `nombre_miembro`,
    `m`.`apellido` AS `apellido_miembro`,
    `a`.`fecha_hora_checkin` AS `fecha_ingreso`,
    `a`.`resultado` AS `resultado`,
    `pm`.`nombre_plan` AS `nombre_plan_membresia`
  FROM
    (((`asistencia` `a`
      LEFT JOIN `miembros` `m` ON ((`a`.`id_miembro` = `m`.`id_miembro`)))
      LEFT JOIN `membresias_miembro` `mm` ON ((`a`.`id_membresia_valida` =
      `mm`.`id_membresia`)))
      LEFT JOIN `planes_membresia` `pm` ON ((`mm`.`id_plan` = `pm`.`id_plan`)))
  ORDER BY `a`.`fecha_hora_checkin` DESC
```

Los diccionarios se utilizan para asociar los parametros de la consulta con los parametros del metodo

```
public class DAsistencia : ConexionBase
```

Inheritance

[object](#) ↗ ← [ConexionBase](#) ← DAsistencia

Methods

Eliminar(uint)

Recibe el id de la asistencia a eliminar y ejecuta una sentencia SQL (DELETE) que elimina el registro de asistencia correspondiente.

```
public void Eliminar(uint id)
```

Parameters

[id uint](#)

Id único de la asistencia a eliminar.

Listar()

Realiza una consulta SQL (SELECT) que obtiene todos los registros de la vista_asistencia.

```
public DataTable Listar()
```

Returns

[DataTable](#)

DataTable con los datos de las asistencias.

ListarPorDNI(string)

Realiza una consulta SQL (SELECT) sobre la vista_asistencia para obtener los registros de asistencias cuyo DNI del miembro coincida parcial o totalmente con el valor proporcionado.

```
public DataTable ListarPorDNI(string dni)
```

Parameters

[dni string](#)

DNI o parte del DNI del miembro a buscar.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ListarPorFecha(DateTime, DateTime)

Realiza una consulta SQL (SELECT) sobre la vista_asistencia para obtener los registros de asistencias cuya fecha de ingreso se encuentre dentro del rango proporcionado.

```
public DataTable ListarPorFecha(DateTime fechaInicio, DateTime fechaFin)
```

Parameters

[fechaInicio](#) [DateTime](#)

Fecha de inicio del rango.

[fechaFin](#) [DateTime](#)

Fecha de fin del rango.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

RegistrarAsistencia(Asistencia)

Recibe una instancia de Asistencia y ejecuta una sentencia SQL (INSERT) que inserta un nuevo registro de asistencia con los datos proporcionados. Si el id_miembro o id_membresia_valida son nulos, se insertará NULL en la base de datos

```
public void RegistrarAsistencia(Asistencia asistencia)
```

Parameters

asistencia [Asistencia](#)

Instancia de Asistencia con los datos a insertar.

Class DMembresias

Namespace: [Gimnasio.Datos](#)

Clase de acceso a datos para la gestión de membresías en el sistema de gimnasio. Hereda de [ConexionBase](#) y utiliza la entidad [Membresias](#). Proporciona métodos CRUD y de búsqueda para la tabla membresias_miembro y la vista vista_membresias.

La vista_membresias es una vista SQL que consolida información relevante de las membresías, uniendo datos de las tablas membresias_miembro, miembros y planes_membresia.

```
VIEW `vista_membresias` AS
SELECT
    `mm`.`id_membresia` AS `id_membresia`,
    `mm`.`id_miembro` AS `id_miembro`,
    `mm`.`id_plan` AS `id_plan`,
    `m`.`dni` AS `dni_miembro`,
    `m`.`apellido` AS `apellido_miembro`,
    `m`.`nombre` AS `nombre_miembro`,
    `p`.`nombre_plan` AS `nombre_plan`,
    `p`.`precio` AS `precio_plan`,
    `p`.`duracion_dias` AS `duracion_dias_plan`,
    `mm`.`fecha_inicio` AS `fecha_inicio`,
    `mm`.`fecha_fin` AS `fecha_fin`,
    `mm`.`estado_membresia` AS `estado_membresia`,
    `mm`.`fecha_registro` AS `fecha_registro`,
    `mm`.`ultima_modificacion` AS `ultima_modificacion`
FROM
    ((`membresias_miembro` `mm`
JOIN `miembros` `m` ON ((`mm`.`id_miembro` = `m`.`id_miembro`)))
JOIN `planes_membresia` `p` ON ((`mm`.`id_plan` = `p`.`id_plan`)))
ORDER BY `mm`.`ultima_modificacion` DESC
```

Los diccionarios se utilizan para asociar los parametros de la consulta con los parametros del metodo

```
public class DMembresias : ConexionBase
```

Inheritance

[Object](#) ↗ ← [ConexionBase](#) ← DMembresias

Methods

Actualizar(Membresias)

Recibe una instancia de Membresia y ejecuta una sentencia SQL (UPDATE) que actualiza el plan de un registro de membresia existente que corresponde al id de la instancia.

```
public void Actualizar(Membresias membresia)
```

Parameters

membresia [Membresias](#)

Instancia de [Membresias](#) con los datos actualizados.

ActualizarAEstadoInactiva()

Ejecuta una sentencia SQL (UPDATE) que actualiza el estado de un registro de membresia existente a inactiva si la fecha de vencimiento es menor que la fecha actual.

```
public void ActualizarAEstadoInactiva()
```

ActualizarEstadoYFechas(Membresias)

Recibe una instancia de Membresia y ejecuta una sentencia SQL (UPDATE) que actualiza las fechas de un registro de membresia existente que corresponde al id de la instancia.

```
public void ActualizarEstadoYFechas(Membresias membresia)
```

Parameters

membresia [Membresias](#)

Instancia de [Membresias](#) con los datos a actualizar.

Eliminar(uint)

Recibe el id de la membresía a eliminar y ejecuta una sentencia SQL (DELETE) que elimina el registro de membresía correspondiente.

```
public void Eliminar(uint id)
```

Parameters

id [uint](#)

Id único de la membresía a eliminar.

Insertar(Membresias)

Recibe una instancia de Membresia y ejecuta una sentencia SQL (INSERT) que inserta un nuevo registro de membresía con los datos proporcionados.

```
public void Insertar(Membresias membresia)
```

Parameters

membresia [Membresias](#)

Instancia de [Membresias](#) a insertar.

Listar()

Realiza una consulta SQL (SELECT) que obtiene todas las membresías de la vista `vista_membresias`.

```
public DataTable Listar()
```

Returns

[DataTable](#)

DataTable con los datos de las membresías.

ListarPorDni(string)

Recibe un dni de miembro y ejecuta una sentencia SQL (SELECT) que obtiene los registros de la membresías correspondientes al miembro. Permite buscar por parte del DNI utilizando la cláusula LIKE.

```
public DataTable ListarPorDni(string dni)
```

Parameters

dni string ↗

DNI o parte del DNI del miembro a buscar.

Returns

DataTable ↗

DataTable con los resultados de la búsqueda.

ListarPorEstado(string)

Recibe el estado(Activa, Inactiva) y ejecuta una sentencia SQL (SELECT) que obtiene los registros de membresías que tienen ese estado.

```
public DataTable ListarPorEstado(string estado)
```

Parameters

estado string ↗

Estado de la membresía.

Returns

DataTable ↗

DataTable con los resultados de la búsqueda.

ListarPorNombrePlan(string)

Recibe el nombre de un plan y ejecuta una sentencia SQL (SELECT) que obtiene los registros de membresías que tienen ese plan. Permite buscar por parte del nombre utilizando la cláusula LIKE.

```
public DataTable ListarPorNombrePlan(string nombrePlan)
```

Parameters

nombrePlan [string](#)

Nombre o parte del nombre del plan a buscar.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ObtenerDuracionPorMembresia(uint)

Recibe un id de membresía y ejecuta una sentencia SQL (SELECT) que obtiene la duración del plan de la membresía correspondiente. Luego, la convierte a entero.

```
public uint ObtenerDuracionPorMembresia(uint idMembresia)
```

Parameters

idMembresia [uint](#)

Identificador único de la membresía.

Returns

[uint](#)

Duración en días del plan.

ObtenerIdMembresia(Membresias)

Recibe una instancia de Membresia y ejecuta una sentencia SQL (SELECT) que obtiene el id de la membresía correspondiente a la instancia. Luego lo convierte a entero.

```
public uint ObtenerIdMembresia(Membresias membresia)
```

Parameters

membresia [Membresias](#)

Instancia de [Membresias](#) para la cual se busca el ID.

Returns

[uint](#)

Identificador único de la membresía.

ObtenerMembresiaMasReciente(uint)

Recibe un id de Miembro y ejecuta una sentencia SQL (SELECT) que obtiene la membresía más reciente del miembro. La consulta ordena por fecha de fin en orden descendente y limita el resultado a 1.

```
public DataTable ObtenerMembresiaMasReciente(uint idMiembro)
```

Parameters

idMiembro [uint](#)

Identificador único del miembro.

Returns

[DataTable](#)

DataTable con los datos de la membresía más reciente.

ObtenerPorDni(string)

Recibe un dni de miembro y ejecuta una sentencia SQL (SELECT) que obtiene los registros de la membresías correspondientes al miembro.

```
public DataTable ObtenerPorDni(string dni)
```

Parameters

dni [string](#)

DNI del miembro.

Returns

[DataTable](#)

DataTable con los datos de la membresía encontrada.

VerificarExistenciaDeMiembroYPlan(Membresias)

Recibe una instancia de membresia y ejecuta una sentencia SQL (SELECT COUNT *) que cuenta la cantidad total de registros de membresia con el mismo id_miembro y id_plan. Si el resultado es mayor a 0, significa que ya existe una membresía activa con ese plan para ese miembro.

```
public bool VerificarExistenciaDeMiembroYPlan(Membresias membresia)
```

Parameters

membresia [Membresias](#)

Instancia de [Membresias](#) a verificar.

Returns

[bool](#)

True si existe, False en caso contrario.

Class DMiembros

Namespace: [Gimnasio.Datos](#)

Clase de acceso a datos para la gestión de miembros en el sistema de gimnasio. Hereda de [ConexionBase](#) y utiliza la entidad [Miembros](#). Proporciona métodos CRUD y de búsqueda para la tabla miembros.

Los diccionarios se utilizan para asociar los parametros de la consulta con los parametros del metodo

```
public class DMiembros : ConexionBase
```

Inheritance

[object](#) ← [ConexionBase](#) ← DMiembros

Methods

Actualizar(Miembros)

Recibe una instancia de Miembros y ejecuta una sentencia SQL (UPDATE) que actualiza los campos de un registro de miembro existente que corresponde al id de la instancia.

```
public void Actualizar(Miembros Obj)
```

Parameters

Obj [Miembros](#)

Instancia de [Miembros](#) con los datos actualizados.

Eliminar(uint)

Recibe el id del miembro a eliminar y ejecuta una sentencia SQL (DELETE) que elimina el registro de miembro correspondiente. Si el miembro tiene membresías asociadas y existe una restricción de clave foránea, captura la excepción y lanza un mensaje específico.

```
public void Eliminar(uint id)
```

Parameters

`id uint` ↗

Identificador único del miembro a eliminar.

Insertar(Miembros)

Utiliza los datos proporcionados en la instancia de Miembros para ejecutar una sentencia SQL(INSERT) para insertar un nuevo miembro. Los valores nulos se almacenan como NULL en la base de datos. Si el DNI esta duplicado, lanza una excepción.

```
public void Insertar(Miembros Obj)
```

Parameters

`Obj Miembros`

Instancia de [Miembros](#) a insertar.

Listar()

Ejecuta una consulta SQL (SELECT) que obtiene todos los miembros ordenados por la fecha de última modificación

```
public DataTable Listar()
```

Returns

[DataTable](#) ↗

DataTable con los datos de los miembros.

ListarPorDni(string)

Recibe el Dni o parte del Dni del miembro a buscar y ejecuta una sentencia SQL (SELECT) que busca coincidencias en la base de datos. Utiliza la cláusula LIKE para permitir coincidencias parciales. Los resultados se ordenan por la fecha de última modificación.

```
public DataTable ListarPorDni(string dni)
```

Parameters

dni [string](#)

DNI o parte del DNI del miembro a buscar.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ListarPorNombre(string)

Recibe el nombre o parte del nombre/apellido del miembro a buscar y ejecuta una sentencia SQL (SELECT) que busca coincidencias en la base de datos. Utiliza la cláusula LIKE para permitir coincidencias parciales. Los resultados se ordenan por la fecha de última modificación.

```
public DataTable ListarPorNombre(string nombre)
```

Parameters

nombre [string](#)

Nombre o parte del nombre/apellido del miembro a buscar.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ObtenerPorDni(string)

Recibe el Dni del miembro a buscar y ejecuta una sentencia SQL (SELECT) que busca coincidencias exactas en la base de datos.

```
public DataTable ObtenerPorDni(string dni)
```

Parameters

dni [string](#)

DNI del miembro a buscar.

Returns

[DataTable](#)

DataTable con los datos del miembro encontrado.

Class DPagos

Namespace: [Gimnasio.Datos](#)

Clase de acceso a datos para la gestión de pagos en el sistema de gimnasio. Hereda de [ConexionBase](#) y utiliza la entidad [Pagos](#). Proporciona métodos CRUD y de búsqueda para la tabla pagos y la vista vista_pagos.

La vista consolida la información relevante de los registros de pagos. Permite consultar en una sola consulta los datos del pago, el miembro, el plan, el usuario que registró el pago. Realiza LEFT JOIN entre pagos y las demás tablas, permitiendo obtener la información de pagos incluso si los datos de miembro, membresía, plan o usuario no están presentes.

```
VIEW `vista_pagos` AS
SELECT
    `p`.`id_pago` AS `id_pago`,
    `p`.`id_membresia` AS `id_membresia`,
    `p`.`id_usuario_registro` AS `id_usuario_registro`,
    `m`.`apellido` AS `apellido_miembro`,
    `m`.`nombre` AS `nombre_miembro`,
    `m`.`dni` AS `dni_miembro`,
    `pm`.`nombre_plan` AS `nombre_plan`,
    `p`.`monto_pagado` AS `monto`,
    `p`.`metodo_pago` AS `metodo`,
    `p`.`numero_comprobante` AS `comprobante`,
    `p`.`notas` AS `notas`,
    `p`.`fecha_pago` AS `fecha_pago`,
    `us`.`nombre_completo` AS `nombre_usuario`
FROM
    (((`pagos` `p`
    LEFT JOIN `membresias_miembro` `mm` ON ((`p`.`id_membresia` = `mm`.`id_membresia`)))
    LEFT JOIN `miembros` `m` ON ((`mm`.`id_miembro` = `m`.`id_miembro`)))
    LEFT JOIN `planes_membresia` `pm` ON ((`mm`.`id_plan` = `pm`.`id_plan`)))
    LEFT JOIN `usuarios_sistema` `us` ON ((`p`.`id_usuario_registro` =
    `us`.`id_usuario`)))
ORDER BY `p`.`fecha_pago` DESC
```

Los diccionarios se utilizan para asociar los parametros de la consulta con los parametros del metodo

```
public class DPagos : ConexionBase
```

Inheritance

```
object ↵ ConexionBase ↵ DPagos
```

Methods

Actualizar(Pagos)

Recibe una instancia de Pago y ejecuta una sentencia SQL (UPDATE) que actualiza los datos de un registro de pago existente que corresponde al id de la instancia.

```
public void Actualizar(Pagos pago)
```

Parameters

[pago](#) [Pagos](#)

Instancia de [Pagos](#) con los datos actualizados.

Eliminar(uint)

Recibe el id del pago a eliminar y ejecuta una sentencia SQL (DELETE) que elimina el registro de pago correspondiente.

```
public void Eliminar(uint id)
```

Parameters

[id](#) [uint](#)

Identificador único del pago a eliminar.

Insertar(Pagos)

Recibe una instancia de Pagos y ejecuta una sentencia SQL (INSERT) que inserta un nuevo registro de pago con los datos proporcionados. Si los datos son nulos, se insertará NULL en la base de datos.

```
public void Insertar(Pagos pago)
```

Parameters

pago [Pagos](#)

Instancia de [Pagos](#) a insertar.

Listar()

Realiza una consulta SQL(SELECT) que Obtiene todos los pagos desde la vista_pagos.

```
public DataTable Listar()
```

Returns

[DataTable](#)

DataTable con los datos de los pagos.

ListarPorDni(string)

Recibe un dni de miembro y ejecuta una sentencia SQL (SELECT) que obtiene los registros de pagos correspondientes al miembro. Permite buscar por parte del DNI utilizando la cláusula LIKE.

```
public DataTable ListarPorDni(string dni)
```

Parameters

dni [string](#)

DNI o parte del DNI del miembro a buscar.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ListarPorFecha(DateTime, DateTime)

Realiza una consulta SQL (SELECT) sobre la vista_pagos para obtener los registros cuya fecha de pago se encuentre dentro del rango proporcionado.

```
public DataTable ListarPorFecha(DateTime fechaInicio, DateTime fechaFin)
```

Parameters

fechaInicio [DateTime](#)

Fecha de inicio del rango.

fechaFin [DateTime](#)

Fecha de fin del rango.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ListarPorMetodoPago(string)

Recibe el metodo de pago y ejecuta una sentencia SQL (SELECT) que obtiene los registros de pago que tienen ese metodo.

```
public DataTable ListarPorMetodoPago(string metodoPago)
```

Parameters

metodoPago [string](#)

Método de pago a buscar.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ListarPorMontos(decimal, decimal)

Realiza una consulta SQL (SELECT) sobre la vista_pagos para obtener los registros cuyo monto se encuentre dentro del rango proporcionado.

```
public DataTable ListarPorMontos(decimal montoMin, decimal montoMax)
```

Parameters

montoMin [decimal](#)

Monto mínimo.

montoMax [decimal](#)

Monto máximo.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ListarPorNombrePlan(string)

Recibe el nombre de un plan y ejecuta una sentencia SQL (SELECT) que obtiene los registros de pagos correspondientes a ese plan. Permite buscar por parte del nombre utilizando la cláusula LIKE.

```
public DataTable ListarPorNombrePlan(string nombre)
```

Parameters

nombre [string](#)

Nombre o parte del nombre del plan a buscar.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

Class DPlanes

Namespace: [Gimnasio.Datos](#)

Clase de acceso a datos para la gestión de planes de membresía. Hereda de [ConexionBase](#) y utiliza la entidad [Planes](#). Proporciona métodos CRUD y de búsqueda para la tabla `planes_membresia`.

Los diccionarios se utilizan para asociar los parametros de la consulta con los parametros del metodo

```
public class DPlanes : ConexionBase
```

Inheritance

[object](#) ← [ConexionBase](#) ← DPlanes

Methods

Actualizar(Planes)

Recibe una instancia de Planes y ejecuta una sentencia SQL (UPDATE) que actualiza los campos de un registro de plan existente que corresponde al id de la instancia. Los valores nulos se almacenan como NULL en la base de datos.

```
public void Actualizar(Planes Obj)
```

Parameters

`Obj` [Planes](#)

Instancia de [Planes](#) con los datos actualizados.

Eliminar(uint)

Recibe el id del plan a eliminar y ejecuta una sentencia SQL (DELETE) que elimina el registro de plan correspondiente. Si el plan tiene membresías asociadas y existe una restricción de clave foránea, captura la excepción y lanza un mensaje específico.

```
public void Eliminar(uint id)
```

Parameters

[id uint](#)

Identificador único del plan a eliminar.

Exceptions

[Exception](#)

Se lanza si el plan tiene membresías asociadas o por errores de la base de datos.

Insertar(Planes)

Utiliza los datos proporcionados en la instancia de Planes recibida para ejecutar una sentencia SQL(INSERT), la cual inserta el nuevo plan. Los valores nulos se almacenan como NULL en la base de datos. Se lanza una excepción si el nombre del plan ya existe.

```
public void Insertar(Planes Obj)
```

Parameters

[Obj Planes](#)

Instancia de [Planes](#) a insertar.

Exceptions

[Exception](#)

Se lanza si el nombre del plan ya existe o por errores de la base de datos.

Listar()

Ejecuta una consulta SQL (SELECT) que obtiene todos los planes de la base de datos ordenados por la última modificación.

```
public DataTable Listar()
```

Returns

[DataTable](#)

DataTable con los datos de los planes.

ListarPorDuracion(uint)

Recibe la duración en días del plan a buscar y ejecuta una sentencia SQL (SELECT) que busca coincidencias en la base de datos. Utiliza la cláusula WHERE para filtrar por duración exacta.

```
public DataTable ListarPorDuracion(uint duracion)
```

Parameters

[duracion uint](#)

Duración en días del plan.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ListarPorNombre(string)

Recibe el nombre o parte del nombre del plan a buscar y ejecuta una sentencia SQL (SELECT) que busca coincidencias en la base de datos. Utiliza la cláusula LIKE para permitir coincidencias parciales.

```
public DataTable ListarPorNombre(string nombre)
```

Parameters

[nombre string](#)

Nombre o parte del nombre del plan a buscar.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ListarPorPrecio(decimal)

Recibe el precio del plan a buscar y ejecuta una sentencia SQL (SELECT) que busca coincidencias en la base de datos. Utiliza la cláusula WHERE para filtrar por precio exacto.

```
public DataTable ListarPorPrecio(decimal precio)
```

Parameters

[precio decimal](#)

Precio del plan.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

Class DReclamos

Namespace: [Gimnasio.Datos](#)

Clase de acceso a datos para la gestión de reclamos en el sistema de gimnasio. Hereda de [ConexionBase](#) y utiliza la entidad [Reclamos](#). Proporciona métodos CRUD y de búsqueda para la tabla reclamos y la vista vista_reclamos.

La vista consolida la información relevante de los reclamos, permitiendo consultar en una sola consulta datos del reclamo y el miembro que lo realizó. Realiza LEFT JOIN entre la reclamos y miembros, permitiendo obtener la información de reclamo incluso si los datos de miembro no están presentes.

```
VIEW `vista_reclamos` AS
SELECT
    `r`.`id_reclamos` AS `id_reclamos`,
    `r`.`tipo` AS `tipo`,
    `r`.`descripcion` AS `descripcion`,
    `r`.`fecha_envio` AS `fecha_envio`,
    `r`.`estado` AS `estado`,
    `r`.`respuesta` AS `respuesta`,
    `r`.`fecha_respuesta` AS `fecha_respuesta`,
    `m`.`dni` AS `dni_miembro`
FROM
    (`reclamos` `r`
    LEFT JOIN `miembros` `m` On ((`r`.`id_miembro` = `m`.`id_miembro`)))
ORDER BY `r`.`fecha_envio` DESC
```

Los diccionarios se utilizan para asociar los parámetros de la consulta con los parámetros del método

```
public class DReclamos : ConexionBase
```

Inheritance

[object](#) ← [ConexionBase](#) ← DReclamos

Methods

Actualizar(Reclamos)

Recibe una instancia de Reclamos y ejecuta una sentencia SQL (UPDATE) que actualiza los datos de un registro de reclamo existente que corresponde al id de la instancia.

```
public void Actualizar(Reclamos Obj)
```

Parameters

Obj [Reclamos](#)

Instancia de [Reclamos](#) con los datos actualizados.

ActualizarElEstadoAPendiente(uint)

Recibe un id de reclamo y ejecuta una sentencia SQL (UPDATE) que actualiza el estado a pendiente de un registro de reclamo que corresponde al id.

```
public void ActualizarElEstadoAPendiente(uint id)
```

Parameters

id [uint](#)

Identificador único del reclamo.

ActualizarElEstadoAResuelto(uint)

Recibe un id de reclamo y ejecuta una sentencia SQL (UPDATE) que actualiza el estado a resuelto de un registro de reclamo que corresponde al id.

```
public void ActualizarElEstadoAResuelto(uint id)
```

Parameters

id [uint](#)

Identificador único del reclamo.

ActualizarRespuesta(Reclamos)

Recibe una instancia de Reclamos y ejecuta una sentencia SQL (UPDATE) que actualiza el campo de respuesta de un registro de reclamo existente que corresponde al id de la instancia.

```
public void ActualizarRespuesta(Reclamos Obj)
```

Parameters

Obj [Reclamos](#)

Instancia de [Reclamos](#) con la respuesta actualizada.

Eliminar(uint)

Recibe el id del reclamo a eliminar y ejecuta una sentencia SQL (DELETE) que elimina el registro de reclamo correspondiente.

```
public void Eliminar(uint id)
```

Parameters

id uint

Identificador único del reclamo a eliminar.

Insertar(Reclamos)

Recibe una instancia de Reclamos y ejecuta una sentencia SQL (INSERT) que inserta un nuevo registro de reclamos con los datos proporcionados. Si id de miembro es nulo, se insertará NULL en la base de datos Utiliza los datos de la entidad [Reclamos](#).

```
public void Insertar(Reclamos Obj)
```

Parameters

Obj [Reclamos](#)

Instancia de [Reclamos](#) a insertar.

Listar()

Realiza una consulta SQL (SELECT) que obtiene todos los registros de la vista_reclamos.

```
public DataTable Listar()
```

Returns

[DataTable](#)

DataTable con los datos de los reclamos.

ListarPorEstado(string)

Recibe el estado y ejecuta una sentencia SQL (SELECT) que obtiene los registros de reclamos que tienen ese estado.

```
public DataTable ListarPorEstado(string Estado)
```

Parameters

Estado [string](#)

Estado del reclamo ("pendiente", "resuelto").

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

Class DUsuarios

Namespace: [Gimnasio.Datos](#)

Clase de acceso a datos para la gestión de usuarios en el sistema de gimnasio. Hereda de [ConexionBase](#) y utiliza la entidad [Usuarios](#). Proporciona métodos CRUD y de búsqueda para la tabla usuarios_sistema y la vista_usuarios.

La vista consolida la información relevante de los registros de usuarios, permitiendo consultar en una sola consulta datos de los usuarios y sus roles. Realiza JOIN entre la tabla de usuarios y la tabla de roles, permitiendo obtener la información de usuario junto con su rol asociado.

```
VIEW `vista_usuarios` AS
  SELECT
    `u`.`id_usuario` AS `id_usuario`,
    `u`.`username` AS `username`,
    `u`.`password_hash` AS `password_hash`,
    `u`.`nombre_completo` AS `nombre_completo`,
    `u`.`email` AS `email`,
    `r`.`nombre_rol` AS `nombre_rol`,
    `u`.`fecha_creacion` AS `fecha_creacion`,
    `u`.`ultima_modificacion` AS `ultima_modificacion`
  FROM
    (`usuarios_sistema` `u`
     JOIN `roles` `r` ON ((`u`.`id_rol` = `r`.`id_rol`)))
  ORDER BY `u`.`ultima_modificacion` DESC
```

Los diccionarios se utilizan para asociar los parámetros de la consulta con los parámetros del método.

```
public class DUsuarios : ConexionBase
```

Inheritance

[object](#) ↗ ← [ConexionBase](#) ← DUsuarios

Methods

Actualizar(Usuarios)

Recibe una instancia de usuarios y ejecuta una sentencia SQL (UPDATE) que actualiza los datos de un registro de usuario existente que corresponde al id de la instancia.

```
public void Actualizar(Usuarios Obj)
```

Parameters

Obj [Usuarios](#)

Instancia de [Usuarios](#) con los datos actualizados.

Eliminar(uint)

Recibe el id del usuario a eliminar y ejecuta una sentencia SQL (DELETE) que elimina el registro de usuario correspondiente.

```
public void Eliminar(uint id)
```

Parameters

id [uint](#)

Identificador único del usuario a eliminar.

Insertar(Usuarios)

Recibe una instancia de Usuarios y ejecuta una sentencia SQL (INSERT) que inserta un nuevo registro de usuarios con los datos proporcionados.

```
public void Insertar(Usuarios Obj)
```

Parameters

Obj [Usuarios](#)

Instancia de [Usuarios](#) a insertar.

Listar()

Realiza una consulta SQL (SELECT) que obtiene todos los registros de la vista_usuarios.

```
public DataTable Listar()
```

Returns

[DataTable](#)

DataTable con los datos de los usuarios.

ListarPorNombre(string)

Recibe el nombre de un usuario y ejecuta una sentencia SQL (SELECT) que obtiene los registros de usuario que tienen ese nombre. Permite buscar por parte del nombre utilizando la cláusula LIKE.

```
public DataTable ListarPorNombre(string nombre)
```

Parameters

nombre [string](#)

Nombre o parte del nombre del usuario a buscar.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ObtenerPorUsername(string)

Recibe un username y ejecuta una sentencia SQL (SELECT) que obtiene el registro de usuario correspondiente. Luego convierte el DataTable en una instancia de Usuarios o retorna Nothing en caso de no encontrar.

```
public Usuarios ObtenerPorUsername(string username)
```

Parameters

username [string](#)

Nombre de usuario a buscar.

Returns

[Usuarios](#)

Instancia de [Usuarios](#) si se encuentra el usuario, Nothing en caso contrario.

Namespace Gimnasio.Entidades

Classes

Asistencia

Representa un registro de asistencia de un miembro en el sistema del gimnasio. Contiene información sobre el miembro, la fecha y hora de ingreso, el resultado y la membresía válida asociada.

Membresias

Representa una membresía de un miembro en el sistema del gimnasio. Contiene información sobre el plan, fechas, estado y días restantes de la membresía.

Miembros

Representa un miembro del gimnasio. Contiene información personal y de contacto, así como fechas de registro y última modificación.

Pagos

Representa un pago realizado por una membresía en el sistema del gimnasio. Contiene información sobre el monto, método, usuario que registró el pago y otros detalles relevantes.

Planes

Representa un plan de membresía dentro del sistema del gimnasio. Contiene información relevante como nombre, descripción, duración, precio y fechas de creación y modificación.

Reclamos

Representa un reclamo realizado por un miembro en el sistema del gimnasio. Contiene información sobre el tipo, descripción, estado, respuesta y fechas asociadas al reclamo.

Usuarios

Representa un usuario del sistema del gimnasio. Contiene información de autenticación, datos personales, rol y fechas de registro y modificación.

Class Asistencia

Namespace: [Gimnasio.Entidades](#)

Representa un registro de asistencia de un miembro en el sistema del gimnasio. Contiene información sobre el miembro, la fecha y hora de ingreso, el resultado y la membresía válida asociada.

```
public class Asistencia
```

Inheritance

[object](#) ← Asistencia

Fields

_fechaHoraCheckin

Fecha y hora en que se registró la asistencia.

```
private DateTime _fechaHoraCheckin
```

Field Value

[DateTime](#)

_idAsistencia

Identificador único del registro de asistencia.

```
private ulong _idAsistencia
```

Field Value

[ulong](#)

_idMembresiaValida

Identificador de la membresía válida utilizada para el ingreso (puede ser nulo).

```
private uint? _idMembresiaValida
```

Field Value

[uint](#)?

_idMiembro

Identificador del miembro que realizó el check-in (puede ser nulo).

```
private uint? _idMiembro
```

Field Value

[uint](#)?

_resultado

Resultado del intento de asistencia ('Exitoso', 'Fallido_DNI_NoEncontrado', 'Fallido_Membresia_Inactiva', 'Fallido_No_Hay_Membresia', 'Fallido_Otro')

```
private string _resultado
```

Field Value

[string](#)?

Properties

FechaHoraCheckin

Obtiene o establece la fecha y hora del check-in.

```
public DateTime FechaHoraCheckin { get; set; }
```

Property Value

[DateTime](#)

IdAsistencia

Obtiene o establece el identificador único del registro de asistencia.

```
public ulong IdAsistencia { get; set; }
```

Property Value

[ulong](#)

IdMembresiaValida

Obtiene o establece el identificador de la membresía válida asociada al registro de asistencia.

```
public uint? IdMembresiaValida { get; set; }
```

Property Value

[uint](#)?

IdMiembro

Obtiene o establece el identificador del miembro que realizó el check-in.

```
public uint? IdMiembro { get; set; }
```

Property Value

[uint](#)?

Resultado

Obtiene o establece el resultado del registro de asistencia.

```
public string Resultado { get; set; }
```

Property Value

[string](#) ↗

Class Membresias

Namespace: [Gimnasio.Entidades](#)

Representa una membresía de un miembro en el sistema del gimnasio. Contiene información sobre el plan, fechas, estado y días restantes de la membresía.

```
public class Membresias
```

Inheritance

[object](#) ← Membresias

Fields

_diasRestantes

Días restantes hasta la finalización de la membresía.

```
private uint _diasRestantes
```

Field Value

[uint](#)

_estadoMembresia

Estado actual de la membresía (activa, inactiva).

```
private string _estadoMembresia
```

Field Value

[string](#)

_fechaFin

Fecha de finalización de la membresía.

```
private DateTime _fechaFin
```

Field Value

[DateTime](#)

_fechainicio

Fecha de inicio de la membresía.

```
private DateTime _fechaInicio
```

Field Value

[DateTime](#)

_fechaRegistro

Fecha de registro de la membresía en el sistema.

```
private DateTime _fechaRegistro
```

Field Value

[DateTime](#)

_idMembresia

Identificador único de la membresía.

```
private uint _idMembresia
```

Field Value

[uint](#)

_idMiembro

Identificador del miembro asociado a la membresía.

`private uint _idMiembro`

Field Value

[uint](#)

_idPlan

Identificador del plan asociado a la membresía.

`private uint _idPlan`

Field Value

[uint](#)

_ultimaModificacion

Fecha de la última modificación de la membresía.

`private DateTime _ultimaModificacion`

Field Value

[DateTime](#)

Properties

DiasRestantes

Obtiene o establece los días restantes hasta la finalización de la membresía.

```
public uint DiasRestantes { get; set; }
```

Property Value

[uint](#)

EstadoMembresia

Obtiene o establece el estado actual de la membresía.

```
public string EstadoMembresia { get; set; }
```

Property Value

[string](#)

FechaFin

Obtiene o establece la fecha de finalización de la membresía.

```
public DateTime FechaFin { get; set; }
```

Property Value

[DateTime](#)

Fechainicio

Obtiene o establece la fecha de inicio de la membresía.

```
public DateTime FechaInicio { get; set; }
```

Property Value

[DateTime](#)

FechaRegistro

Obtiene o establece la fecha de registro de la membresía.

```
public DateTime FechaRegistro { get; set; }
```

Property Value

[DateTime](#)

IdMembresia

Obtiene o establece el identificador único de la membresía.

```
public uint IdMembresia { get; set; }
```

Property Value

[uint](#)

IdMiembro

Obtiene o establece el identificador del miembro asociado.

```
public uint IdMiembro { get; set; }
```

Property Value

[uint](#)

IdPlan

Obtiene o establece el identificador del plan asociado.

```
public uint IdPlan { get; set; }
```

Property Value

[uint](#)

UltimaModificacion

Obtiene o establece la fecha de la última modificación de la membresía.

```
public DateTime UltimaModificacion { get; set; }
```

Property Value

[DateTime](#)

Class Miembros

Namespace: [Gimnasio.Entidades](#)

Representa un miembro del gimnasio. Contiene información personal y de contacto, así como fechas de registro y última modificación.

```
public class Miembros
```

Inheritance

[object](#) ← Miembros

Fields

_apellido

Apellido del miembro.

```
private string _apellido
```

Field Value

[string](#)

_dni

Documento Nacional de Identidad del miembro.

```
private string _dni
```

Field Value

[string](#)

_email

Correo electrónico del miembro.

```
private string _email
```

Field Value

[string ↗](#)

_fechaRegistro

Fecha de registro del miembro en el sistema.

```
private DateTime _fechaRegistro
```

Field Value

[DateTime ↗](#)

_genero

Género del miembro ('Masculino', 'Femenino', 'Otro', 'Prefiero no decir')

```
private string _genero
```

Field Value

[string ↗](#)

_idMiembro

Identificador único del miembro.

```
private uint _idMiembro
```

Field Value

[uint](#)

_nombre

Nombre del miembro.

```
private string _nombre
```

Field Value

[string](#)

_telefono

Teléfono de contacto del miembro.

```
private string _telefono
```

Field Value

[string](#)

_ultimaModificacion

Fecha de la última modificación de los datos del miembro.

```
private DateTime _ultimaModificacion
```

Field Value

[DateTime](#)

Properties

Apellido

Obtiene o establece el apellido del miembro.

```
public string Apellido { get; set; }
```

Property Value

[string](#)

Dni

Obtiene o establece el DNI del miembro.

```
public string Dni { get; set; }
```

Property Value

[string](#)

Email

Obtiene o establece el correo electrónico del miembro.

```
public string Email { get; set; }
```

Property Value

[string](#)

FechaRegistro

Obtiene o establece la fecha de registro del miembro.

```
public DateTime FechaRegistro { get; set; }
```

Property Value

[DateTime](#)

Genero

Obtiene o establece el género del miembro.

```
public string Genero { get; set; }
```

Property Value

[string](#)

IdMiembro

Obtiene o establece el identificador único del miembro.

```
public uint IdMiembro { get; set; }
```

Property Value

[uint](#)

Nombre

Obtiene o establece el nombre del miembro.

```
public string Nombre { get; set; }
```

Property Value

[string](#)

Telefono

Obtiene o establece el teléfono del miembro.

```
public string Telefono { get; set; }
```

Property Value

[string](#)

UltimaModificacion

Obtiene o establece la fecha de la última modificación de los datos del miembro.

```
public DateTime UltimaModificacion { get; set; }
```

Property Value

[DateTime](#)

Class Pagos

Namespace: [Gimnasio.Entidades](#)

Representa un pago realizado por una membresía en el sistema del gimnasio. Contiene información sobre el monto, método, usuario que registró el pago y otros detalles relevantes.

```
public class Pagos
```

Inheritance

[object](#) ← Pagos

Fields

_fechaPago

Fecha y hora en que se realizó el pago.

```
private DateTime _fechaPago
```

Field Value

[DateTime](#)

_idMembresia

Identificador de la membresía asociada al pago.

```
private uint? _idMembresia
```

Field Value

[uint](#)?

_idPago

Identificador único del pago.

```
private uint _idPago
```

Field Value

[uint](#) ↗

_idUsuarioRegistro

Identificador del usuario que registró el pago.

```
private uint? _idUsuarioRegistro
```

Field Value

[uint](#) ↗?

_metodoPago

Método de pago utilizado ('Efectivo', 'Tarjeta Débito', 'Tarjeta Crédito', 'Transferencia Bancaria', 'Mercado Pago', 'Otro').

```
private string _metodoPago
```

Field Value

[string](#) ↗

_montoPagado

Monto pagado.

```
private decimal _montoPagado
```

Field Value

[decimal](#) ↗

_notas

Notas adicionales sobre el pago.

`private string _notas`

Field Value

[string](#) ↗

_numeroComprobante

Número de comprobante del pago (si corresponde).

`private string _numeroComprobante`

Field Value

[string](#) ↗

Properties

FechaPago

Obtiene o establece la fecha y hora en que se realizó el pago.

`public DateTime FechaPago { get; set; }`

Property Value

[DateTime](#) ↗

IdMembresia

Obtiene o establece el identificador de la membresía asociada al pago.

```
public uint? IdMembresia { get; set; }
```

Property Value

[uint](#)?

IdPago

Obtiene o establece el identificador único del pago.

```
public uint IdPago { get; set; }
```

Property Value

[uint](#)?

IdUsuarioRegistro

Obtiene o establece el identificador del usuario que registró el pago.

```
public uint? IdUsuarioRegistro { get; set; }
```

Property Value

[uint](#)?

MetodoPago

Obtiene o establece el método de pago utilizado.

```
public string MetodoPago { get; set; }
```

Property Value

[string](#) ↗

MontoPagado

Obtiene o establece el monto pagado.

```
public decimal MontoPagado { get; set; }
```

Property Value

[decimal](#) ↗

Notas

Obtiene o establece notas adicionales sobre el pago.

```
public string Notas { get; set; }
```

Property Value

[string](#) ↗

NumeroComprobante

Obtiene o establece el número de comprobante del pago.

```
public string NumeroComprobante { get; set; }
```

Property Value

[string](#) ↗

Class Planes

Namespace: [Gimnasio.Entidades](#)

Representa un plan de membresía dentro del sistema del gimnasio. Contiene información relevante como nombre, descripción, duración, precio y fechas de creación y modificación.

```
public class Planes
```

Inheritance

[object](#) ← Planes

Fields

_Descripcion

Descripción del plan.

```
private string _Descripcion
```

Field Value

[string](#)

_DuracionDias

Duración del plan en días.

```
private uint _DuracionDias
```

Field Value

[uint](#)

_FechaCreacion

Fecha de creación del plan.

```
private DateTime _FechaCreacion
```

Field Value

[DateTime](#)

_IdPlan

Identificador único del plan.

```
private uint _IdPlan
```

Field Value

[uint](#)

_NombrePlan

Nombre del plan.

```
private string _NombrePlan
```

Field Value

[string](#)

_Precio

Precio del plan.

```
private decimal _Precio
```

Field Value

[decimal](#)

_UltimaModificacion

Fecha de la última modificación del plan.

```
private DateTime _UltimaModificacion
```

Field Value

[DateTime](#)

Properties

Descripcion

Obtiene o establece la descripción del plan.

```
public string Descripcion { get; set; }
```

Property Value

[string](#)

DuracionDias

Obtiene o establece la duración del plan en días.

```
public uint DuracionDias { get; set; }
```

Property Value

[uint](#)

FechaCreacion

Obtiene o establece la fecha de creación del plan.

```
public DateTime FechaCreacion { get; set; }
```

Property Value

[DateTime](#)

IdPlan

Obtiene o establece el identificador único del plan.

```
public uint IdPlan { get; set; }
```

Property Value

[uint](#)

NombrePlan

Obtiene o establece el nombre del plan.

```
public string NombrePlan { get; set; }
```

Property Value

[string](#)

Precio

Obtiene o establece el precio del plan.

```
public decimal Precio { get; set; }
```

Property Value

[decimal](#) ↴

UltimaModificacion

Obtiene o establece la fecha de la última modificación del plan.

```
public DateTime UltimaModificacion { get; set; }
```

Property Value

[DateTime](#) ↴

Class Reclamos

Namespace: [Gimnasio.Entidades](#)

Representa un reclamo realizado por un miembro en el sistema del gimnasio. Contiene información sobre el tipo, descripción, estado, respuesta y fechas asociadas al reclamo.

```
public class Reclamos
```

Inheritance

[object](#) ← Reclamos

Fields

_descripcion

Descripción detallada del reclamo.

```
private string _descripcion
```

Field Value

[string](#)

_estado

Estado actual del reclamo (pendiente, resuelto).

```
private string _estado
```

Field Value

[string](#)

_fechaEnvio

Fecha en que se envió el reclamo.

```
private DateTime _fechaEnvio
```

Field Value

[DateTime](#)

_fechaRespuesta

Fecha en que se respondió el reclamo.

```
private DateTime _fechaRespuesta
```

Field Value

[DateTime](#)

_idMiembro

Identificador del miembro que realizó el reclamo (puede ser nulo).

```
private uint? _idMiembro
```

Field Value

[uint](#)?

_idReclamos

Identificador único del reclamo.

```
private uint _idReclamos
```

Field Value

[uint](#)

_respuesta

Respuesta dada al reclamo.

```
private string _respuesta
```

Field Value

[string](#)

_tipo

Tipo de reclamo ('sugerencia', 'reclamo').

```
private string _tipo
```

Field Value

[string](#)

Properties

Descripcion

Obtiene o establece la descripción del reclamo.

```
public string Descripcion { get; set; }
```

Property Value

[string](#)

Estado

Obtiene o establece el estado actual del reclamo.

```
public string Estado { get; set; }
```

Property Value

[string](#)

FechaEnvio

Obtiene o establece la fecha de envío del reclamo.

```
public DateTime FechaEnvio { get; set; }
```

Property Value

[DateTime](#)

FechaRespuesta

Obtiene o establece la fecha de respuesta del reclamo.

```
public DateTime FechaRespuesta { get; set; }
```

Property Value

[DateTime](#)

IdMiembro

Obtiene o establece el identificador del miembro que realizó el reclamo.

```
public uint? IdMiembro { get; set; }
```

Property Value

[uint](#)?

IdReclamos

Obtiene o establece el identificador único del reclamo.

```
public uint IdReclamos { get; set; }
```

Property Value

[uint](#)

Respuesta

Obtiene o establece la respuesta dada al reclamo.

```
public string Respuesta { get; set; }
```

Property Value

[string](#)

Tipo

Obtiene o establece el tipo de reclamo.

```
public string Tipo { get; set; }
```

Property Value

[string](#)

Class Usuarios

Namespace: [Gimnasio.Entidades](#)

Representa un usuario del sistema del gimnasio. Contiene información de autenticación, datos personales, rol y fechas de registro y modificación.

```
public class Usuarios
```

Inheritance

[object](#) ← Usuarios

Fields

_email

Correo electrónico del usuario.

```
private string _email
```

Field Value

[string](#)

_fechaCreacion

Fecha de creación del usuario en el sistema.

```
private DateTime _fechaCreacion
```

Field Value

[DateTime](#)

_idRol

Identificador del rol asignado al usuario.

```
private uint _idRol
```

Field Value

[uint](#)

_idUsuario

Identificador único del usuario.

```
private uint _idUsuario
```

Field Value

[uint](#)

_nombreCompleto

Nombre completo del usuario.

```
private string _nombreCompleto
```

Field Value

[string](#)

_passwordHash

Hash de la contraseña del usuario.

```
private string _passwordHash
```

Field Value

[string](#)

_ultimaModificacion

Fecha de la última modificación de los datos del usuario.

```
private DateTime _ultimaModificacion
```

Field Value

[DateTime](#)

_username

Nombre de usuario utilizado para iniciar sesión.

```
private string _username
```

Field Value

[string](#)

Properties

Email

Obtiene o establece el correo electrónico del usuario.

```
public string Email { get; set; }
```

Property Value

[string](#)

FechaCreacion

Obtiene o establece la fecha de creación del usuario.

```
public DateTime FechaCreacion { get; set; }
```

Property Value

[DateTime](#)

IdRol

Obtiene o establece el identificador del rol asignado al usuario.

```
public uint IdRol { get; set; }
```

Property Value

[uint](#)

IdUsuario

Obtiene o establece el identificador único del usuario.

```
public uint IdUsuario { get; set; }
```

Property Value

[uint](#)

NombreCompleto

Obtiene o establece el nombre completo del usuario.

```
public string NombreCompleto { get; set; }
```

Property Value

[string](#)

PasswordHash

Obtiene o establece el hash de la contraseña.

```
public string PasswordHash { get; set; }
```

Property Value

[string](#)

UltimaModificacion

Obtiene o establece la fecha de la última modificación de los datos del usuario.

```
public DateTime UltimaModificacion { get; set; }
```

Property Value

[DateTime](#)

Username

Obtiene o establece el nombre de usuario.

```
public string Username { get; set; }
```

Property Value

[string](#)

Namespace Gimnasio.Errores

Classes

[ManejarErrores](#)

Clase utilitaria para el registro de errores en el sistema. Permite guardar mensajes de error y excepciones en un archivo de log ubicado en la carpeta Logs. Ademas permite mostrar mensajes de error al usuario mediante cuadros de diálogo.

Class ManejarErrores

Namespace: [Gimnasio.Errores](#)

Clase utilitaria para el registro de errores en el sistema. Permite guardar mensajes de error y excepciones en un archivo de log ubicado en la carpeta Logs. Ademas permite mostrar mensajes de error al usuario mediante cuadros de diálogo.

```
public class ManejarErrores
```

Inheritance

[object](#) ← ManejarErrores

Fields

logFilePath

Ruta completa del archivo de log donde se almacenan los errores.

```
private static readonly string logFilePath
```

Field Value

[string](#)

Methods

Log(string, Exception)

1. Verifica si el directorio de logs existe; si no, lo crea automáticamente.
2. Abre (o crea) el archivo log.txt en modo adjuntar dentro de la carpeta Logs de la aplicación.
3. Escribe una entrada de error que incluye:
 - Fecha y hora del registro.
 - Mensaje personalizado que describe el contexto del error.
 - Mensaje de la excepción capturada.
 - Stack trace de la excepción.
4. Separa cada registro con una línea divisoria.

5. Si ocurre una excepción de E/S durante el proceso de log, la omite silenciosamente para evitar errores adicionales. Este método es estático y puede ser llamado desde cualquier parte del sistema para registrar errores técnicos o de negocio.

```
public static void Log(string message, Exception ex)
```

Parameters

message [string](#)

Mensaje personalizado que describe el contexto del error.

ex [Exception](#)

Excepción capturada que contiene detalles del error.

Mostrar(string, Exception)

1. Registra la excepción recibida en el archivo log.txt utilizando [Log\(string, Exception\)](#) con el mensaje "Capa Presentación".
2. Muestra un cuadro de diálogo al usuario con un mensaje personalizado y el mensaje de la excepción, usando MsgBox en modo crítico.
3. Si ocurre una excepción de E/S durante el proceso, la omite silenciosamente para evitar errores adicionales. Este método asegura que todos los errores sean registrados y notificados al usuario de forma clara.

```
public static void Mostrar(string mensajeUsuario, Exception ex)
```

Parameters

mensajeUsuario [string](#)

Mensaje personalizado que se mostrará al usuario.

ex [Exception](#)

Excepción capturada que será registrada y cuyo mensaje se mostrará al usuario.

Namespace Gimnasio.Negocio

Clases

[NAsistencia](#)

Lógica de negocio para la gestión de asistencias en el sistema de gimnasio. Interactúa con la capa de datos [DAsistencia](#) y la entidad [Asistencia](#). Todas las operaciones de la capa de negocio están envueltas en bloques Try...Catch.

Si ocurre una excepción, se registra el error utilizando [Log\(string, Exception\)](#) en un log.txt Luego, la excepción se propaga nuevamente mediante Throw New Exception(ex.Message).

[NMembresias](#)

Lógica de negocio para la gestión de membresías en el sistema de gimnasio. Interactúa con la capa de datos [dMembresias](#) y la entidad [Membresias](#). Todas las operaciones de la capa de negocio están envueltas en bloques Try...Catch.

Si ocurre una excepción, se registra el error utilizando [Log\(string, Exception\)](#) en un log.txt Luego, la excepción se propaga nuevamente mediante Throw New Exception(ex.Message).

[NMiembros](#)

Lógica de negocio para la gestión de miembros en el sistema de gimnasio. Interactúa con la capa de datos [dMiembros](#) y la entidad [Miembros](#). Todas las operaciones de la capa de negocio están envueltas en bloques Try...Catch.

Si ocurre una excepción, se registra el error utilizando [Log\(string, Exception\)](#) en un log.txt Luego, la excepción se propaga nuevamente mediante Throw New Exception(ex.Message).

[NPagos](#)

Lógica de negocio para la gestión de pagos en el sistema de gimnasio. Interactúa con la capa de datos [dPagos](#) y la entidad [Pagos](#). Todas las operaciones de la capa de negocio están envueltas en bloques Try...Catch.

Si ocurre una excepción, se registra el error utilizando [Log\(string, Exception\)](#) en un log.txt Luego, la excepción se propaga nuevamente mediante Throw New Exception(ex.Message).

[NPlanes](#)

Lógica de negocio para la gestión de planes en el sistema de gimnasio. Interactúa con la capa de datos [dPlanes](#) y la entidad [Planes](#). Todas las operaciones de la capa de negocio están envueltas en bloques Try...Catch.

Si ocurre una excepción, se registra el error utilizando [Log\(string, Exception\)](#) en un log.txt Luego, la excepción se propaga nuevamente mediante Throw New Exception(ex.Message) para que pueda ser gestionada en la interfaz de usuario.

[NReclamos](#)

Lógica de negocio para la gestión de reclamos en el sistema de gimnasio. Interactúa con la capa de datos [dReclamos](#) y la entidad [Reclamos](#). Todas las operaciones de la capa de negocio están envueltas en bloques Try...Catch.

Si ocurre una excepción, se registra el error utilizando [Log\(string, Exception\)](#) en un log.txt Luego, la excepción se propaga nuevamente mediante Throw New Exception(ex.Message) para que pueda ser gestionada en la interfaz de usuario.

[N Usuarios](#)

Lógica de negocio para la gestión de usuarios en el sistema de gimnasio. Interactúa con la capa de datos [dUsuarios](#) y la entidad [Usuarios](#). Todas las operaciones de la capa de negocio están envueltas en bloques Try...Catch.

Si ocurre una excepción, se registra el error utilizando [Log\(string, Exception\)](#) en un log.txt Luego, la excepción se propaga nuevamente mediante Throw New Exception(ex.Message) para que pueda ser gestionada en la interfaz de usuario.

Class NAistencia

Namespace: [Gimnasio.Negocio](#)

Lógica de negocio para la gestión de asistencias en el sistema de gimnasio. Interactúa con la capa de datos [DAsistencia](#) y la entidad [Asistencia](#). Todas las operaciones de la capa de negocio están envueltas en bloques Try...Catch.

Si ocurre una excepción, se registra el error utilizando [Log\(string, Exception\)](#) en un log.txt Luego, la excepción se propaga nuevamente mediante Throw New Exception(ex.Message).

```
public class NAistencia
```

Inheritance

[object](#) ← NAistencia

Fields

dAsistencias

Instancia de la capa de datos para asistencias.

```
private DAsistencia dAsistencias
```

Field Value

[DAsistencia](#)

Methods

Eliminar(uint)

Elimina un registro de asistencia del sistema según su identificador utilizando la capa de datos [Eliminar\(uint\)](#).

```
public void Eliminar(uint id)
```

Parameters

id [uint](#)

Identificador único de la asistencia a eliminar.

Listar()

Obtiene la lista de todas las asistencias registradas.

```
public DataTable Listar()
```

Returns

[DataTable](#)

DataTable con los datos de las asistencias.

ListarPorDNI(string)

Busca asistencias por DNI del miembro utilizando la capa de datos [ListarPorDNI\(string\)](#).

```
public DataTable ListarPorDNI(string dni)
```

Parameters

dni [string](#)

DNI del miembro a buscar.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ListarPorFecha(DateTime, DateTime)

Busca asistencias por rango de fechas utilizando la capa de datos [ListarPorFecha\(DateTime, DateTime\)](#).

```
public DataTable ListarPorFecha(DateTime fechaInicio, DateTime fechaFin)
```

Parameters

fechaInicio [DateTime](#)

Fecha de inicio del rango.

fechaFin [DateTime](#)

Fecha de fin del rango.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

RegistrarIngresoPorDNI(string)

Registra el ingreso de un miembro al gimnasio utilizando su DNI. Proceso:

1. Busca el miembro por su DNI mediante [ObtenerPorDni\(string\)](#).
 - Si no existe, registra la asistencia como "Fallido_DNI_NoEncontrado" y retorna ese resultado.
2. Si el miembro existe, obtiene su membresía más reciente con [ObtenerMembresiaMasReciente\(uint\)](#).
 - Si no tiene membresía, registra la asistencia como "Fallido_No_Hay_Membresia" y retorna ese resultado.
 - Si la membresía está inactiva, registra la asistencia como "Fallido_Membresia_Inactiva" y retorna ese resultado.
 - Si la membresía está activa, registra la asistencia como "Exitoso" y retorna ese resultado.
3. En todos los casos, se crea un registro de asistencia en la base de datos mediante [RegistrarAsistencia\(Asistencia\)](#), incluyendo el resultado, la fecha y hora del intento, el id del miembro (si corresponde) y el id de la membresía válida (si corresponde).

```
public string RegistrarIngresoPorDNI(string dni)
```

Parameters

dni [string](#)

DNI del miembro a registrar el ingreso.

Returns

[string](#)

"Exito" si la membresía está activa, "Fallido_Membresia_Inactiva" si la membresía está inactiva, "Fallido_DNI_NoEncontrado" si el DNI no existe, "Fallido_No_Hay_Membresia" si no tiene membresía, "Fallido_Otro" para otros casos.

Class NMembresias

Namespace: [Gimnasio.Negocio](#)

Lógica de negocio para la gestión de membresías en el sistema de gimnasio. Interactúa con la capa de datos [dMembresias](#) y la entidad [Membresias](#). Todas las operaciones de la capa de negocio están envueltas en bloques Try...Catch.

Si ocurre una excepción, se registra el error utilizando [Log\(string, Exception\)](#) en un log.txt Luego, la excepción se propaga nuevamente mediante Throw New Exception(ex.Message).

```
public class NMembresias
```

Inheritance

[object](#) ← NMembresias

Fields

dMembresias

Instancia de la capa de datos para membresías.

```
private DMembresias dMembresias
```

Field Value

[DMembresias](#)

Methods

ActualizaAEstadoInactiva()

Actualiza el estado de las membresías vencidas a "Inactiva" utilizando [ActualizarAEstadoInactiva\(\)](#).

```
public void ActualizaAEstadoInactiva()
```

Actualizar(Membresias)

Actualiza los datos de una membresía existente. Valida que no exista con [VerificarExistenciaDeMiembroYPlan\(Membresias\)](#). Si no existe, actualiza la membresía en la base de datos con [Actualizar\(Membresias\)](#).

```
public void Actualizar(Membresias membresia)
```

Parameters

membresia [Membresias](#)

Instancia de [Membresias](#) con los datos actualizados.

Eliminar(uint)

Elimina una membresía según su id con [Eliminar\(uint\)](#).

```
public void Eliminar(uint id)
```

Parameters

id [uint](#)

Identificador único de la membresía a eliminar.

Insertar(Membresias)

Inserta una nueva membresía en el sistema. Valida que no exista una membresía activa o pendiente de pago para el mismo miembro y plan con [VerificarExistenciaDeMiembroYPlan\(Membresias\)](#). Si no existe, establece la fecha de inicio y fin de la membresía y la inserta en la bd con [Insertar\(Membresias\)](#).

```
public void Insertar(Membresias membresia)
```

Parameters

membresia [Membresias](#)

Instancia de [Membresias](#) a insertar.

Listar()

Obtiene la lista de todas las membresías registradas con [Listar\(\)](#). Antes de listar, actualiza el estado de las membresías vencidas a inactivas mediante [ActualizaAEstadoinactiva\(\)](#).

```
public DataTable Listar()
```

Returns

[DataTable](#)

DataTable con los datos de las membresías.

ListarPorDni(string)

Busca membresías por DNI del miembro con [ListarPorDni\(string\)](#).

```
public DataTable ListarPorDni(string dni)
```

Parameters

dni [string](#)

DNI o parte del DNI del miembro a buscar.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ListarPorEstado(string)

Busca membresías por estado con [ListarPorEstado\(string\)](#).

```
public DataTable ListarPorEstado(string estado)
```

Parameters

estado [string](#)

Estado de la membresía (por ejemplo: "Activa", "Inactiva").

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ListarPorNombrePlan(string)

Busca membresías por nombre de plan con [ListarPorNombrePlan\(string\)](#).

```
public DataTable ListarPorNombrePlan(string nombre)
```

Parameters

nombre [string](#)

Nombre o parte del nombre del plan a buscar.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

Exceptions

[Exception](#)

Propaga excepciones de la capa de datos.

ObtenerIdMembresia(Membresias)

Obtiene el id de una membresía específica con [ObtenerIdMembresia\(Membresias\)](#).

```
public uint ObtenerIdMembresia(Membresias membresia)
```

Parameters

membresia [Membresias](#)

Instancia de [Membresias](#) para la cual se busca el ID.

Returns

[uint](#)

Identificador único de la membresía.

ObtenerMembresiaMasReciente(uint)

Obtiene la membresía más reciente de un miembro con [ObtenerMembresiaMasReciente\(uint\)](#).

```
public DataTable ObtenerMembresiaMasReciente(uint idMiembro)
```

Parameters

idMiembro [uint](#)

Identificador único del miembro.

Returns

[DataTable](#)

DataTable con los datos de la membresía más reciente.

ObtenerPorDni(string)

Obtiene una membresía por el DNI del miembro con [ObtenerPorDni\(string\)](#).

```
public DataTable ObtenerPorDni(string dni)
```

Parameters

dni [string](#)

DNI del miembro.

Returns

[DataTable ↗](#)

DataTable con los datos de la membresía encontrada.

Class NMiembros

Namespace: [Gimasio.Negocio](#)

Lógica de negocio para la gestión de miembros en el sistema de gimnasio. Interactúa con la capa de datos [dMiembros](#) y la entidad [Miembros](#). Todas las operaciones de la capa de negocio están envueltas en bloques Try...Catch.

Si ocurre una excepción, se registra el error utilizando [Log\(string, Exception\)](#) en un log.txt Luego, la excepción se propaga nuevamente mediante Throw New Exception(ex.Message).

```
public class NMiembros
```

Inheritance

[object](#) ← NMiembros

Fields

dMiembros

Instancia de la capa de datos para miembros.

```
private DMiembros dMiembros
```

Field Value

[DMiembros](#)

Methods

Actualizar(Miembros)

Valida que los campos sean correctos con [ValidarCampos\(Miembros\)](#). Actualiza los datos de un miembro existente en la base de datos con [Insertar\(Miembros\)](#).

```
public void Actualizar(Miembros Obj)
```

Parameters

Obj [Miembros](#)

Instancia de [Miembros](#) con los datos actualizados.

Eliminar(uint)

Elimina un miembro del sistema según su id con [Eliminar\(uint\)](#).

```
public void Eliminar(uint id)
```

Parameters

id [uint](#)

Identificador único del miembro a eliminar.

Insertar(Miembros)

Obtiene un miembro por su ID con [ObtenerPorDni\(string\)](#). Valida que no exista un miembro con el mismo DNI y que los campos sean correctos con [ValidarCampos\(Miembros\)](#). Por ultimo, inserta el miembro en la base de datos con [Insertar\(Miembros\)](#).

```
public void Insertar(Miembros Obj)
```

Parameters

Obj [Miembros](#)

Instancia de [Miembros](#) a insertar.

Listar()

Obtiene la lista de todos los miembros registrados con [Listar\(\)](#).

```
public DataTable Listar()
```

Returns

[DataTable ↗](#)

DataTable con los datos de los miembros.

ListarPorDni(string)

Realiza una validación y busca miembros por DNI utilizando la capa de datos [ListarPorDni\(string\)](#).

```
public DataTable ListarPorDni(string dni)
```

Parameters

[dni string ↗](#)

DNI o parte del DNI del miembro a buscar.

Returns

[DataTable ↗](#)

DataTable con los resultados de la búsqueda.

ListarPorNombre(string)

Realiza una validación y busca miembros por nombre utilizando la capa de datos [ListarPorNombre\(string\)](#).

```
public DataTable ListarPorNombre(string nombre)
```

Parameters

[nombre string ↗](#)

Nombre o parte del nombre del miembro a buscar.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ObtenerPorDni(string)

Realiza una validación y obtiene un miembro por su DNI utilizando la capa de datos [ObtenerPorDni\(string\)](#).

```
public DataTable ObtenerPorDni(string dni)
```

Parameters

[dni string](#)

DNI del miembro a buscar.

Returns

[DataTable](#)

DataTable con los datos del miembro encontrado.

ValidarCampos(Miembros)

Valida los campos de la entidad [Miembros](#) antes de realizar operaciones de inserción o actualización.

```
private void ValidarCampos(Miembros Obj)
```

Parameters

[Obj Miembros](#)

Instancia de [Miembros](#) a validar.

Class NPagos

Namespace: [Gimnasio.Negocio](#)

Lógica de negocio para la gestión de pagos en el sistema de gimnasio. Interactúa con la capa de datos [dPagos](#) y la entidad [Pagos](#). Todas las operaciones de la capa de negocio están envueltas en bloques Try...Catch.

Si ocurre una excepción, se registra el error utilizando [Log\(string, Exception\)](#) en un log.txt Luego, la excepción se propaga nuevamente mediante Throw New Exception(ex.Message).

```
public class NPagos
```

Inheritance

[object](#) ← NPagos

Fields

dPagos

Instancia de la capa de datos para pagos.

```
private DPagos dPagos
```

Field Value

[DPagos](#)

Methods

Actualizar(Pagos)

Actualiza un pago existente en el sistema.

```
public void Actualizar(Pagos pago)
```

Parameters

pago [Pagos](#)

Instancia de [Pagos](#) con los datos actualizados.

Eliminar(uint)

Elimina un pago del sistema según su id con [Eliminar\(uint\)](#).

```
public void Eliminar(uint id)
```

Parameters

[id uint](#)

Identificador único del pago a eliminar.

Insertar(Pagos)

1. Valida los campos de la entidad [Pagos](#) recibida como parámetro.
2. Inserta el pago en la base de datos utilizando [Insertar\(Pagos\)](#).
3. Actualiza la membresía asociada al pago:
 - o Obtiene la duración de la membresía con [ObtenerDuracionPorMembresia\(uint\)](#).
 - o Establece la fecha de inicio como la fecha actual y la fecha de fin sumando la duración.
 - o Cambia el estado de la membresía a "Activa".
 - o Actualiza estos datos en la base de datos mediante [ActualizarEstadoYFechas\(Membresias\)](#).

```
public void Insertar(Pagos pago)
```

Parameters

[pago Pagos](#)

Instancia de [Pagos](#) a insertar.

Listar()

Obtiene la lista de todos los pagos registrados con [Listar\(\)](#).

```
public DataTable Listar()
```

Returns

[DataTable](#)

DataTable con los datos de los pagos.

ListarPorDni(string)

Realiza una validación y busca pagos por DNI del miembro utilizando la capa de datos [ListarPorDni\(string\)](#).

```
public DataTable ListarPorDni(string dni)
```

Parameters

dni [string](#)

DNI o parte del DNI del miembro a buscar.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ListarPorFecha(DateTime, DateTime)

Busca pagos por rango de fechas utilizando la capa de datos [ListarPorFecha\(DateTime, DateTime\)](#).

```
public DataTable ListarPorFecha(DateTime fechaInicio, DateTime fechaFin)
```

Parameters

fechaInicio [DateTime](#)

Fecha de inicio del rango.

fechaFin [DateTime](#)

Fecha de fin del rango.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ListarPorMetodoPago(string)

Busca pagos por método de pago utilizando la capa de datos [ListarPorMetodoPago\(string\)](#).

```
public DataTable ListarPorMetodoPago(string metodo)
```

Parameters

metodo [string](#)

Método de pago a buscar.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ListarPorMontos(decimal, decimal)

Realiza una validación y Busca pagos por rango de montos utilizando la capa de datos [ListarPorMontos\(decimal, decimal\)](#).

```
public DataTable ListarPorMontos(decimal montoMin, decimal montoMax)
```

Parameters

montoMin [decimal](#)

Monto mínimo.

montoMax [decimal](#)

Monto máximo.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ListarPorNombrePlan(string)

Realiza una validación y Busca pagos por nombre de plan utilizando la capa de datos [ListarPorNombrePlan\(string\)](#).

```
public DataTable ListarPorNombrePlan(string nombre)
```

Parameters

nombre [string](#)

Nombre o parte del nombre del plan a buscar.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ValidarCampos(Pagos)

Valida los campos de la entidad [Pagos](#) antes de realizar operaciones de inserción.

```
private void ValidarCampos(Pagos Obj)
```

Parameters

Obj Pagos

Instancia de [Pagos](#) a validar.

Class NPlanes

Namespace: [Gimnasio.Negocio](#)

Lógica de negocio para la gestión de planes en el sistema de gimnasio. Interactúa con la capa de datos [dPlanes](#) y la entidad [Planes](#). Todas las operaciones de la capa de negocio están envueltas en bloques Try...Catch.

Si ocurre una excepción, se registra el error utilizando [Log\(string, Exception\)](#) en un log.txt Luego, la excepción se propaga nuevamente mediante Throw New Exception(ex.Message) para que pueda ser gestionada en la interfaz de usuario.

```
public class NPlanes
```

Inheritance

[object](#) ← NPlanes

Fields

dPlanes

Instancia de la capa de datos para planes.

```
private DPlanes dPlanes
```

Field Value

[DPlanes](#)

Methods

Actualizar(Planes)

1. Valida los campos de la entidad [Planes](#) con [ValidarCampos\(Planes\)](#):
2. Actualiza el plan en la base de datos mediante [Actualizar\(Planes\)](#).

```
public void Actualizar(Planes Obj)
```

Parameters

Obj [Planes](#)

Instancia de [Planes](#) con los datos actualizados.

Eliminar(uint)

Elimina un plan según su id con [Eliminar\(uint\)](#).

```
public void Eliminar(uint id)
```

Parameters

id [uint](#)

Identificador único del plan a eliminar.

Insertar(Planes)

1. Verifica que no exista ya un plan con el mismo nombre utilizando [ListarPorNombre\(string\)](#).
 - Si existe, lanza una excepción indicando que el plan ya está registrado.
2. Valida los campos de la entidad [Planes](#) con [ValidarCampos\(Planes\)](#):
3. Inserta el plan en la base de datos mediante [Insertar\(Planes\)](#).

```
public void Insertar(Planes Obj)
```

Parameters

Obj [Planes](#)

Instancia de [Planes](#) a insertar.

Listar()

Obtiene la lista de todos los planes registrados con [Listar\(\)](#).

```
public DataTable Listar()
```

Returns

[DataTable](#)

DataTable con los datos de los planes.

ListarPorDuracion(uint)

Realiza una validación y busca planes por duración utilizando la capa de datos [ListarPorDuracion\(uint\)](#).

```
public DataTable ListarPorDuracion(uint duracion)
```

Parameters

[duracion](#) [uint](#)

Duración en días del plan.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ListarPorNombre(string)

Realiza una validación y busca planes por nombre utilizando la capa de datos [ListarPorNombre\(string\)](#).

```
public DataTable ListarPorNombre(string nombre)
```

Parameters

[nombre](#) [string](#)

Nombre o parte del nombre del plan a buscar.

Returns

[DataTable](#) ↗

DataTable con los resultados de la búsqueda.

ListarPorPrecio(decimal)

Realiza una validación y busca planes por precio utilizando la capa de datos [ListarPorPrecio\(decimal\)](#).

```
public DataTable ListarPorPrecio(decimal precio)
```

Parameters

[precio](#) [decimal](#) ↗

Precio del plan.

Returns

[DataTable](#) ↗

DataTable con los resultados de la búsqueda.

ValidarCampos(Planes)

Valida los campos de la entidad [Planes](#) antes de realizar operaciones de inserción o actualización.

```
private void ValidarCampos(Planes Obj)
```

Parameters

[Obj](#) [Planes](#)

Instancia de [Planes](#) a validar.

Class NReclamos

Namespace: [Gimnasio.Negocio](#)

Lógica de negocio para la gestión de reclamos en el sistema de gimnasio. Interactúa con la capa de datos [dReclamos](#) y la entidad [Reclamos](#). Todas las operaciones de la capa de negocio están envueltas en bloques Try...Catch.

Si ocurre una excepción, se registra el error utilizando [Log\(string, Exception\)](#) en un log.txt Luego, la excepción se propaga nuevamente mediante Throw New Exception(ex.Message) para que pueda ser gestionada en la interfaz de usuario.

```
public class NReclamos
```

Inheritance

[object](#) ← NReclamos

Fields

dReclamos

Instancia de la capa de datos para reclamos.

```
private DReclamos dReclamos
```

Field Value

[DReclamos](#)

Methods

Actualizar(Reclamos)

Valida los campos y actualiza los datos de un reclamo existente con [Actualizar\(Reclamos\)](#)

```
public void Actualizar(Reclamos Obj)
```

Parameters

Obj [Reclamos](#)

Instancia de [Reclamos](#) con los datos actualizados.

ActualizarElEstadoAPendiente(uint)

Cambia el estado de un reclamo a "pendiente" utilizando [ActualizarElEstadoAPendiente\(uint\)](#).

```
public void ActualizarElEstadoAPendiente(uint id)
```

Parameters

id [uint](#)

Identificador único del reclamo.

ActualizarElEstadoAResuelto(uint)

Cambia el estado de un reclamo a "resuelto" utilizando [ActualizarElEstadoAResuelto\(uint\)](#).

```
public void ActualizarElEstadoAResuelto(uint id)
```

Parameters

id [uint](#)

Identificador único del reclamo.

ActualizarRespuesta(Reclamos)

Actualiza la respuesta de un reclamo utilizando [ActualizarRespuesta\(Reclamos\)](#).

```
public void ActualizarRespuesta(Reclamos Obj)
```

Parameters

Obj [Reclamos](#)

Instancia de [Reclamos](#) con la respuesta actualizada.

Eliminar(uint)

Elimina un reclamo del sistema según su id con [Eliminar\(uint\)](#).

```
public void Eliminar(uint id)
```

Parameters

[id uint](#)

Identificador único del reclamo a eliminar.

Insertar(Reclamos)

Valida los campos e inserta un nuevo reclamo en el sistema con [Insertar\(Reclamos\)](#).

```
public void Insertar(Reclamos Obj)
```

Parameters

[Obj Reclamos](#)

Instancia de [Reclamos](#) a insertar.

Listar()

Obtiene la lista de todos los reclamos registrados con [Listar\(\)](#).

```
public DataTable Listar()
```

Returns

[DataTable](#)

DataTable con los datos de los reclamos.

ListarPorEstado(string)

Busca reclamos por estado utilizando la capa de datos [ListarPorEstado\(string\)](#).

```
public DataTable ListarPorEstado(string estado)
```

Parameters

estado [string](#)

Estado del reclamo ("pendiente", "resuelto").

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ValidarCampos(Reclamos)

Valida los campos de la entidad [Reclamos](#) antes de realizar operaciones de inserción o actualización.

```
private void ValidarCampos(Reclamos Obj)
```

Parameters

Obj [Reclamos](#)

Instancia de [Reclamos](#) a validar.

Class NUsuarios

Namespace: [Gimnasio.Negocio](#)

Lógica de negocio para la gestión de usuarios en el sistema de gimnasio. Interactúa con la capa de datos [dUsuarios](#) y la entidad [Usuarios](#). Todas las operaciones de la capa de negocio están envueltas en bloques Try...Catch.

Si ocurre una excepción, se registra el error utilizando [Log\(string, Exception\)](#) en un log.txt Luego, la excepción se propaga nuevamente mediante Throw New Exception(ex.Message) para que pueda ser gestionada en la interfaz de usuario.

```
public class NUsuarios
```

Inheritance

[object](#) ← NUsuarios

Fields

dUsuarios

Instancia de la capa de datos para usuarios.

```
private DUsuarios dUsuarios
```

Field Value

[DUsuarios](#)

Methods

Actualizar(Usuarios)

1. Valida los campos de la entidad [Usuarios](#) recibida como parámetro mediante [Validar Campos\(Usuarios\)](#):
2. Genera el hash SHA256 de la nueva contraseña antes de almacenarla, utilizando el método privado [GenerarHash\(string\)](#).
3. Actualiza el usuario en la base de datos mediante [Actualizar\(Usuarios\)](#).

```
public void Actualizar(Usuarios Obj)
```

Parameters

Obj [Usuarios](#)

Instancia de [Usuarios](#) con los datos actualizados.

Eliminar(uint)

Elimina un usuario del sistema según su id con [Eliminar\(uint\)](#).

```
public void Eliminar(uint id)
```

Parameters

id [uint](#)

Identificador único del usuario a eliminar.

GenerarHash(string)

Genera un hash SHA256 para la contraseña proporcionada (utiliza System.Security.Cryptography)

```
private string GenerarHash(string password)
```

Parameters

password [string](#)

Contraseña en texto plano.

Returns

[string](#)

Hash de la contraseña en formato Base64.

Insertar(Usuarios)

1. Verifica si el nombre de usuario ya está registrado utilizando [ObtenerPorUsername\(string\)](#).
 - o Si el usuario existe, lanza una excepción indicando que el username ya está registrado.
2. Valida los campos de la entidad [Usuarios](#) con [ValidarCampos\(Usuarios\)](#).
3. Genera el hash SHA256 de la contraseña antes de almacenarla, utilizando el método privado [GenerarHash\(string\)](#).
4. Inserta el usuario en la base de datos mediante [Insertar\(Usuarios\)](#).

```
public void Insertar(Usuarios Obj)
```

Parameters

Obj [Usuarios](#)

Instancia de [Usuarios](#) a insertar.

Listar()

Obtiene la lista de todos los usuarios registrado con [Listar\(\)](#).

```
public DataTable Listar()
```

Returns

[DataTable](#)

DataTable con los datos de los usuarios.

ListarPorNombre(string)

Realiza una validación y busca usuarios por nombre utilizando la capa de datos [ListarPorNombre\(string\)](#).

```
public DataTable ListarPorNombre(string nombre)
```

Parameters

nombre [string](#)

Nombre o parte del nombre del usuario a buscar.

Returns

[DataTable](#)

DataTable con los resultados de la búsqueda.

ValidarCampos(Usuarios)

Valida los campos de la entidad [Usuarios](#) antes de realizar operaciones de inserción o actualización.

```
private void ValidarCampos(Usuarios Obj)
```

Parameters

[Obj Usuarios](#)

Instancia de [Usuarios](#) a validar.

ValidarCredenciales(string, string)

1. Busca el usuario por su nombre de usuario utilizando [ObtenerPorUsername\(string\)](#).
 - Si no existe, retorna Nothing.
2. Si el usuario existe, genera el hash SHA256 de la contraseña ingresada mediante [GenerarHash\(string\)](#).
3. Compara el hash generado con el hash almacenado en la base de datos.
 - Si coinciden, retorna la instancia de [Usuarios](#) correspondiente.
 - Si no coinciden, retorna Nothing.

```
public Usuarios ValidarCredenciales(string username, string password)
```

Parameters

[username string](#)

Nombre de usuario.

[password string](#)

Contraseña en texto plano.

Returns

[Usuarios](#)

Instancia de [Usuarios](#) si las credenciales son válidas, Nothing en caso contrario.

Namespace Gimnasio.Presentacion

Classes

[FrmAsistencias](#)

Formulario para el registro de asistencias de miembros mediante el ingreso de DNI. Permite registrar el ingreso, mostrar el estado de las membresías y los días restantes. Utiliza la clase [NAsistencia](#) para la lógica de negocio de asistencias y [NMembresias](#) para la consulta de membresías. Todas las operaciones de esta capa están envueltas en bloques Try...Catch. El manejo de errores se realiza a través de [Mostrar\(string, Exception\)](#) que permite guardar el error en un log.txt y a su vez mostrar un mensaje al usuario.

[FrmLogin](#)

Formulario de inicio de sesión para el sistema de gimnasio. Permite al usuario ingresar sus credenciales y acceder al sistema principal. Todas las operaciones de esta capa están envueltas en bloques Try...Catch. El manejo de errores se realiza a través de [Mostrar\(string, Exception\)](#) que permite guardar el error en un log.txt y a su vez mostrar un mensaje al usuario.

[FrmMembresias](#)

Formulario para la gestión de membresías en el sistema del gimnasio. Permite listar, buscar, agregar, actualizar, eliminar membresías y registrar pagos. Utiliza la clase [NMembresias](#) para la lógica de negocio y la clase [Membresias](#) como entidad. Todas las operaciones de esta capa están envueltas en bloques Try...Catch. El manejo de errores se realiza a través de [Mostrar\(string, Exception\)](#) que permite guardar el error en un log.txt y a su vez mostrar un mensaje al usuario.

[FrmMiembros](#)

Formulario para la gestión de miembros en el sistema del gimnasio. Permite listar, buscar, agregar, actualizar y eliminar miembros. Utiliza la clase [NMiembros](#) para la lógica de negocio y la clase [Miembros](#) como entidad. Todas las operaciones de esta capa están envueltas en bloques Try...Catch. El manejo de errores se realiza a través de [Mostrar\(string, Exception\)](#) que permite guardar el error en un log.txt y a su vez mostrar un mensaje al usuario.

[FrmPagos](#)

Formulario para la gestión y consulta de pagos en el sistema del gimnasio. Permite listar, buscar, actualizar y eliminar pagos. Utiliza la clase [NPagos](#) para la lógica de negocio y la clase [Pagos](#) como entidad. Todas las operaciones de esta capa están envueltas en bloques Try...Catch. El manejo de errores se realiza a través de [Mostrar\(string, Exception\)](#) que permite guardar el error en un log.txt y a su vez mostrar un mensaje al usuario.

[FrmPlanes](#)

Formulario para la gestión de planes en el sistema de gimnasio. Permite listar, buscar, insertar, actualizar y eliminar planes. Utiliza la clase [NPlanes](#) para la lógica de negocio y la clase [Planes](#) como

entidad. El acceso a las operaciones de mantenimiento depende del rol del usuario ([Usuarios](#)). Todas las operaciones de esta capa están envueltas en bloques Try...Catch. El manejo de errores se realiza a través de [Mostrar\(string, Exception\)](#) que permite guardar el error en un log.txt y a su vez mostrar un mensaje al usuario.

[FrmPrincipal](#)

Formulario principal de la aplicación del gimnasio. Gestiona la navegación entre los diferentes módulos y controla el acceso según el rol del usuario. Utiliza la clase [Usuarios](#) para identificar al usuario logueado y su rol. Llama a formularios secundarios como [FrmPlanes](#), [FrmMiembros](#), [FrmMembresias](#), [FrmPagos](#), [FrmRegistroAsistencias](#), [FrmReclamos](#) y [FrmUsuarios](#). Todas las operaciones de esta capa están envueltas en bloques Try...Catch. El manejo de errores se realiza a través de [Mostrar\(string, Exception\)](#) que permite guardar el error en un log.txt y a su vez mostrar un mensaje al usuario.

[FrmReclamos](#)

Formulario para la gestión de reclamos en el sistema del gimnasio. Permite listar, buscar, agregar, actualizar, responder, cambiar estado y eliminar reclamos. Utiliza la clase [nReclamos](#) para la lógica de negocio y la clase [Reclamos](#) como entidad. Todas las operaciones de esta capa están envueltas en bloques Try...Catch. El manejo de errores se realiza a través de [Mostrar\(string, Exception\)](#) que permite guardar el error en un log.txt y a su vez mostrar un mensaje al usuario.

[FrmRegistroAsistencias](#)

Formulario para la gestión y consulta de asistencias de los miembros en el sistema del gimnasio. Permite listar, buscar por DNI o fecha, y eliminar registros de asistencia. Utiliza la clase [NAsistencia](#) para la lógica de negocio. Todas las operaciones de esta capa están envueltas en bloques Try...Catch. El manejo de errores se realiza a través de [Mostrar\(string, Exception\)](#) que permite guardar el error en un log.txt y a su vez mostrar un mensaje al usuario.

[FrmUsuarios](#)

Formulario para la gestión de usuarios del sistema del gimnasio. Permite listar, buscar, agregar, actualizar y eliminar usuarios. Utiliza la clase [nUsuarios](#) para la lógica de negocio y la clase [Usuarios](#) como entidad. Todas las operaciones de esta capa están envueltas en bloques Try...Catch. El manejo de errores se realiza a través de [Mostrar\(string, Exception\)](#) que permite guardar el error en un log.txt y a su vez mostrar un mensaje al usuario.

Class FrmAsistencias

Namespace: [Gimnasio.Presentacion](#)

Formulario para el registro de asistencias de miembros mediante el ingreso de DNI. Permite registrar el ingreso, mostrar el estado de las membresías y los días restantes. Utiliza la clase [NAsistencia](#) para la lógica de negocio de asistencias y [NMembresias](#) para la consulta de membresías. Todas las operaciones de esta capa están envueltas en bloques Try...Catch. El manejo de errores se realiza a través de [Mostrar\(string, Exception\)](#) que permite guardar el error en un log.txt y a su vez mostrar un mensaje al usuario.

```
[DesignerGenerated]
public class FrmAsistencias : Form, IOleControl.Interface, IOleObject.Interface,
IOleInPlaceObject.Interface, IOleInPlaceActiveObject.Interface, IOleWindow.Interface,
IVIEWObject2.Interface, IVIEWObject.Interface, IPersistStreamInit.Interface,
IPersistPropertyBag.Interface, IPersistStorage.Interface, IPersist.Interface,
IQuickActivate.Interface, ISupportOleDropSource, IDropTarget, ISynchronizeInvoke,
IWin32Window, IBindableComponent, IKeyboardToolTip, IHandle<HWND>, IArrangedElement,
IComponent, IDisposable, IContainerControl
```

Inheritance

```
object ↳ ← MarshalByRefObject ↳ ← Component ↳ ← Control ↳ ← ScrollableControl ↳ ←
ContainerControl ↳ ← Form ↳ ← FrmAsistencias
```

Constructors

FrmAsistencias(FrmRegistroAsistencias)

Constructor del formulario [FrmAsistencias](#).

- Inicializa los componentes visuales del formulario llamando a [InitializeComponent\(\)](#).
- Recibe una instancia de [FrmRegistroAsistencias](#) como parámetro y la asigna al campo privado [frmRegistro](#).

```
public FrmAsistencias(FrmRegistroAsistencias frmRegistro)
```

Parameters

[frmRegistro](#) [FrmRegistroAsistencias](#)

Instancia de [FrmRegistroAsistencias](#) que permite la interacción entre formularios.

Fields

components

```
private IContainer components
```

Field Value

[IContainer](#)

frmRegistro

Referencia al formulario [FrmRegistroAsistencias](#) desde el formulario de asistencias.

```
private FrmRegistroAsistencias frmRegistro
```

Field Value

[FrmRegistroAsistencias](#)

nAsistencias

Instancia de la capa de negocio para asistencias.

```
private NAsistencia nAsistencias
```

Field Value

[NAsistencia](#)

Properties

Label2

```
internal virtual Label Label2 { get; set; }
```

Property Value

[Label](#) ↗

dgvListado

```
internal virtual DataGridView dgvListado { get; set; }
```

Property Value

[DataGridView](#) ↗

labelResultado

```
internal virtual Label labelResultado { get; set; }
```

Property Value

[Label](#) ↗

tbDNI

```
internal virtual TextBox tbDNI { get; set; }
```

Property Value

[TextBox](#) ↗

Methods

Dispose(bool)

Disposes of the resources (other than memory) used by the [Form](#).

```
protected override void Dispose(bool disposing)
```

Parameters

disposing [bool](#)

[true](#) to release both managed and unmanaged resources; [false](#) to release only unmanaged resources.

InitializeComponent()

```
private void InitializeComponent()
```

frmAsistencias_Load(object, EventArgs)

Evento que se ejecuta al cargar el formulario. Oculta el DataGriedView.

```
private void frmAsistencias_Load(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

tbDNI_KeyPress(object, KeyPressEventArgs)

Evento que se ejecuta al presionar una tecla en el campo de texto del DNI.

- Si la tecla presionada es Enter, intenta registrar la asistencia del miembro mediante [RegistrarIngresoPorDNI\(string\)](#).
- Llama a [ActualizarDgv\(\)](#) para actualizar el listado general de asistencias.
- Si el resultado es "Exitoso" o "Fallido_Membresia_Inactiva":

- Muestra el DataGridView con el estado de las membresías del miembro, calculando y mostrando los días restantes de cada plan.
- Muestra un mensaje de bienvenida o advertencia según el resultado.
- Si el resultado es "Fallido_DNI_NoEncontrado", informa que el DNI no fue encontrado y solicita reingreso.
- Si el resultado es "Fallido_No_Hay_Membresia", informa que no posee membresías activas y sugiere inscribirse a un plan.
- En cualquier otro caso, lanza una excepción indicando un error desconocido.
- Limpia el campo de texto del DNI tras cada intento.

```
private void tbDNI_KeyPress(object sender, KeyPressEventArgs e)
```

Parameters

sender [object](#)

e [KeyPressEventArgs](#)

Class FrmLogin

Namespace: [Gimnasio.Presentacion](#)

Formulario de inicio de sesión para el sistema de gimnasio. Permite al usuario ingresar sus credenciales y acceder al sistema principal. Todas las operaciones de esta capa están envueltas en bloques Try...Catch. El manejo de errores se realiza a través de [Mostrar\(string, Exception\)](#) que permite guardar el error en un log.txt y a su vez mostrar un mensaje al usuario.

```
[DesignerGenerated]
public class FrmLogin : Form, IOleControl.Interface, IOleObject.Interface,
IOleInPlaceObject.Interface, IOleInPlaceActiveObject.Interface, IOleWindow.Interface,
IViewObject2.Interface, IVViewObject.Interface, IPersistStreamInit.Interface,
IPersistPropertyBag.Interface, IPersistStorage.Interface, IPersist.Interface,
IQuickActivate.Interface, ISupportOleDropSource, IDropTarget, ISynchronizeInvoke,
IWin32Window, IBindableComponent, IKeyboardToolTip, IHandle<HWND>, IArrangedElement,
IComponent, IDisposable, IContainerControl
```

Inheritance

```
object ← MarshalByRefObject ← Component ← Control ← ScrollableControl ←
ContainerControl ← Form ← FrmLogin
```

Fields

components

```
private IContainer components
```

Field Value

[IContainer](#)

Properties

PictureBox1

```
internal virtual PictureBox PictureBox1 { get; set; }
```

Property Value

[PictureBox](#)

btIniciarSesion

```
internal virtual Button btIniciarSesion { get; set; }
```

Property Value

[Button](#)

lblContraseña

```
internal virtual Label lblContraseña { get; set; }
```

Property Value

[Label](#)

lblUsuario

```
internal virtual Label lblUsuario { get; set; }
```

Property Value

[Label](#)

pbMostrarContraseña

```
internal virtual PictureBox pbMostrarContraseña { get; set; }
```

Property Value

[PictureBox](#)

tbContraseña

```
internal virtual TextBox tbContraseña { get; set; }
```

Property Value

[TextBox](#)

tbUsuario

```
internal virtual TextBox tbUsuario { get; set; }
```

Property Value

[TextBox](#)

Methods

Dispose(bool)

Disposes of the resources (other than memory) used by the [Form](#).

```
protected override void Dispose(bool disposing)
```

Parameters

disposing [bool](#)

`true` to release both managed and unmanaged resources; `false` to release only unmanaged resources.

Formato()

Establece el formato inicial de los controles del formulario. Limpia los campos de usuario y contraseña y oculta la contraseña.

```
public void Formato()
```

InitializeComponent()

```
private void InitializeComponent()
```

btIniciarSesion_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Iniciar Sesión" en el formulario de login.

- Obtiene el nombre de usuario y la contraseña ingresados en los campos.
- Valida que ambos campos no estén vacíos; si alguno está vacío, lanza una excepción y muestra un mensaje de error.
- Instancia la capa de negocio [NUsuarios](#) y llama a [ValidarCredenciales\(string, string\)](#) para verificar las credenciales.
 - Si las credenciales son correctas (el método devuelve un objeto [Usuarios](#) distinto de Nothing):
 - Oculta el formulario de login.
 - Crea y muestra el formulario principal [FrmPrincipal](#), pasando el usuario autenticado como parámetro.
 - Si las credenciales son incorrectas, lanza una excepción y muestra un mensaje de error.

```
private void btIniciarSesion_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

frmLogin_Load(object, EventArgs)

Evento que se ejecuta al cargar el formulario. Inicializa el formato de los controles

```
private void frmLogin_Load(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

pbMostrarContraseña_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el ícono para mostrar u ocultar la contraseña en el formulario de login.

- Verifica el estado actual del campo de contraseña:
 - Si la contraseña está oculta, la muestra y cambia el ícono a "ojito abierto".
 - Si la contraseña está visible, la oculta y cambia el ícono a "ojito cerrado".

```
private void pbMostrarContraseña_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

Class FrmMembresias

Namespace: [Gimnasio.Presentacion](#)

Formulario para la gestión de membresías en el sistema del gimnasio. Permite listar, buscar, agregar, actualizar, eliminar membresías y registrar pagos. Utiliza la clase [NMembresias](#) para la lógica de negocio y la clase [Membresias](#) como entidad. Todas las operaciones de esta capa están envueltas en bloques Try...Catch. El manejo de errores se realiza a través de [Mostrar\(string, Exception\)](#) que permite guardar el error en un log.txt y a su vez mostrar un mensaje al usuario.

```
[DesignerGenerated]  
public class FrmMembresias : Form, IOleControl.Interface, IOleObject.Interface,  
IOleInPlaceObject.Interface, IOleInPlaceActiveObject.Interface, IOleWindow.Interface,  
IVIEWObject2.Interface, IVIEWObject.Interface, IPersistStreamInit.Interface,  
IPersistPropertyBag.Interface, IPersistStorage.Interface, IPersist.Interface,  
IQuickActivate.Interface, ISupportOleDropSource, IDropTarget, ISynchronizeInvoke,  
IWin32Window, IBindableComponent, IKeyboardToolTip, IHandle<HWND>, IArrangedElement,  
IComponent, IDisposable, IContainerControl
```

Inheritance

```
object ↳ ← MarshalByRefObject ↳ ← Component ↳ ← Control ↳ ← ScrollableControl ↳ ←  
ContainerControl ↳ ← Form ↳ ← FrmMembresias
```

Constructors

FrmMembresias(Usuarios)

Constructor que inicializa el formulario. Si es recepcionista oculta el botón de eliminar.

```
public FrmMembresias(Usuarios usuario)
```

Parameters

usuario [Usuarios](#)

Instancia de [Usuarios](#) que representa al usuario logueado.

Fields

components

`private IContainer components`

Field Value

[IContainer](#)

esNuevo

Indica si la operación actual es de inserción (True) o actualización (False).

`private bool esNuevo`

Field Value

[bool](#)

nMembresias

Instancia de la capa de negocio para membresías.

`private NMembresias nMembresias`

Field Value

[NMembresias](#)

usuarioActual

Usuario actualmente logueado, utilizado para registrar pagos.

`private Usuarios usuarioActual`

Field Value

[Usuarios](#)

Properties

BtnPagar

```
internal virtual Button BtnPagar { get; set; }
```

Property Value

[Button](#)

Button1

```
internal virtual Button Button1 { get; set; }
```

Property Value

[Button](#)

Button2

```
internal virtual Button Button2 { get; set; }
```

Property Value

[Button](#)

btnActualizar

```
internal virtual Button btnActualizar { get; set; }
```

Property Value

[Button ↗](#)

btnCancelar

```
internal virtual Button btnCancelar { get; set; }
```

Property Value

[Button ↗](#)

btnCancelarPago

```
internal virtual Button btnCancelarPago { get; set; }
```

Property Value

[Button ↗](#)

btnEliminar

```
internal virtual Button btnEliminar { get; set; }
```

Property Value

[Button ↗](#)

btnGuardar

```
internal virtual Button btnGuardar { get; set; }
```

Property Value

[Button](#)

btnGuardarPago

```
internal virtual Button btnGuardarPago { get; set; }
```

Property Value

[Button](#)

btnInsertar

```
internal virtual Button btnInsertar { get; set; }
```

Property Value

[Button](#)

cbBuscarOpcionPlan

```
internal virtual ComboBox cbBuscarOpcionPlan { get; set; }
```

Property Value

[ComboBox](#)

cbMetodo

```
internal virtual ComboBox cbMetodo { get; set; }
```

Property Value

[ComboBox](#)

cbOpcionBuscar

```
internal virtual ComboBox cbOpcionBuscar { get; set; }
```

Property Value

[ComboBox](#)

cbOpcionBuscarMiembro

```
internal virtual ComboBox cbOpcionBuscarMiembro { get; set; }
```

Property Value

[ComboBox](#)

dgvListado

```
internal virtual DataGridView dgvListado { get; set; }
```

Property Value

[DataGridView](#)

dgvMiembro

```
internal virtual DataGridView dgvMiembro { get; set; }
```

Property Value

[DataGridView](#)

dgvPlan

```
internal virtual DataGridView dgvPlan { get; set; }
```

Property Value

[DataGridView](#)

lblComprobante

```
internal virtual Label lblComprobante { get; set; }
```

Property Value

[Label](#)

lblDatosIngreso

```
internal virtual Label lblDatosIngreso { get; set; }
```

Property Value

[Label](#)

lblMetodo

```
internal virtual Label lblMetodo { get; set; }
```

Property Value

[Label](#)

lblMiembro

```
internal virtual Label lblMiembro { get; set; }
```

Property Value

[Label](#) ↗

lblMonto

```
internal virtual Label lblMonto { get; set; }
```

Property Value

[Label](#) ↗

lblNotas

```
internal virtual Label lblNotas { get; set; }
```

Property Value

[Label](#) ↗

lblPagoIngreso

```
internal virtual Label lblPagoIngreso { get; set; }
```

Property Value

[Label](#) ↗

lblPlan

```
internal virtual Label lblPlan { get; set; }
```

Property Value

[Label](#) ↗

lblTotal

```
internal virtual Label lblTotal { get; set; }
```

Property Value

[Label](#) ↗

panelDatosIngreso

```
internal virtual Panel panelDatosIngreso { get; set; }
```

Property Value

[Panel](#) ↗

panelListado

```
internal virtual Panel panelListado { get; set; }
```

Property Value

[Panel](#) ↗

panelPagoIngreso

```
internal virtual Panel panelPagoIngreso { get; set; }
```

Property Value

[Panel](#)

pbReiniciar

```
internal virtual PictureBox pbReiniciar { get; set; }
```

Property Value

[PictureBox](#)

pbReiniciarMiembro

```
internal virtual PictureBox pbReiniciarMiembro { get; set; }
```

Property Value

[PictureBox](#)

pbReiniciarPlan

```
internal virtual PictureBox pbReiniciarPlan { get; set; }
```

Property Value

[PictureBox](#)

tbBuscar

```
internal virtual TextBox tbBuscar { get; set; }
```

Property Value

[TextBox](#)

tbBuscarMiembro

```
internal virtual TextBox tbBuscarMiembro { get; set; }
```

Property Value

[TextBox](#)

tbBuscarPlan

```
internal virtual TextBox tbBuscarPlan { get; set; }
```

Property Value

[TextBox](#)

tbComprobante

```
internal virtual TextBox tbComprobante { get; set; }
```

Property Value

[TextBox](#)

tblDmembresia

```
internal virtual TextBox tbIDmembresia { get; set; }
```

Property Value

[TextBox](#)

tbIDmiembro

```
internal virtual TextBox tbIDmiembro { get; set; }
```

Property Value

[TextBox](#)

tbIDpago

```
internal virtual TextBox tbIDpago { get; set; }
```

Property Value

[TextBox](#)

tbIDplan

```
internal virtual TextBox tbIDplan { get; set; }
```

Property Value

[TextBox](#)

tbMonto

```
internal virtual TextBox tbMonto { get; set; }
```

Property Value

[TextBox](#)

tbNotas

```
internal virtual TextBox tbNotas { get; set; }
```

Property Value

[TextBox](#)

Methods

ActualizarDgv()

Actualiza el DataGridView con la lista de membresías completa obtenida de la capa de negocio [Listar\(\)](#). También actualiza la etiqueta que muestra el total de membresias.

```
public void ActualizarDgv()
```

ActualizarDgv(DataTable)

Actualiza el DataGridView con un DataTable específico de membresía, por ejemplo, tras una búsqueda o filtrado. También actualiza la etiqueta con el total de membresias.

```
public void ActualizarDgv(DataTable listado)
```

Parameters

listado [DataTable](#)

es un DataTable que contiene la lista de membresías.

BtnPagar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Pagar".

- Verifica si hay una fila seleccionada en el DataGridView de membresías.
 - Si no hay ninguna selección, lanza una excepción y muestra un mensaje de advertencia.
- Obtiene el estado de la membresía seleccionada:
 - Si el estado es "Activa", lanza una excepción e informa que no se puede registrar un pago para una membresía activa.
 - Si el estado es "Inactiva":
 - Llama a [HabilitarIngresoPago\(\)](#) para mostrar el panel de registro de pago.
 - Precarga el ID de la membresía y el monto correspondiente en los campos de pago.
 - Establece el campo de monto como solo lectura y selecciona el método de pago predeterminado.

```
private void BtnPagar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

Dispose(bool)

Disposes of the resources (other than memory) used by the [Form](#).

```
protected override void Dispose(bool disposing)
```

Parameters

disposing [bool](#)

[true](#) to release both managed and unmanaged resources; [false](#) to release only unmanaged resources.

HabilitarIngreso()

- Hace visible el panel de ingreso de datos y lo ajusta para ocupar todo el espacio disponible.

- Deshabilita el panel de listado de membresías para evitar la interacción con el listado mientras se está ingresando o editando una membresía.
- Habilita y limpia los campos de búsqueda de miembro y plan, y muestra el botón de reinicio de miembro.
- Inicializa el DataGridView de miembros con la lista completa de miembros, configurando la visibilidad y los encabezados de las columnas.
- Inicializa el DataGridView de planes con la lista completa de planes, configurando la visibilidad y los encabezados de las columnas.
- Establece la opción predeterminada de búsqueda de miembro en "DNI" (índice 1) y de plan en "Nombre" (índice 0).

```
public void HabilitarIngreso()
```

HabilitarIngresoConMiembro(string)

- Llama a [HabilitarIngreso\(\)](#) para preparar la interfaz de ingreso de membresías.
- Asigna el valor del DNI recibido al campo de búsqueda de miembro.
- Deshabilita la edición del campo de búsqueda de miembro y del ComboBox de opciones de búsqueda de miembro, para evitar que el usuario cambie el miembro preseleccionado.
- Oculta el botón de reinicio de miembro, ya que no es necesario reiniciar la búsqueda en este contexto.
- Establece la variable esNuevo en True para indicar que la operación es de inserción de una nueva membresía.
- Cambia el texto del label para indicar al usuario que debe seleccionar un plan para agregar la membresía.

```
public void HabilitarIngresoConMiembro(string dni)
```

Parameters

dni [string](#)

DNI del miembro a preseleccionar en el formulario de ingreso de membresía.

HabilitarIngresoPago()

- Hace visible el panel de ingreso de pago panelPagoIngreso y lo ajusta para ocupar todo el espacio disponible del formulario.

- Oculta el panel de ingreso de datos y el de listado de membresía.
- Limpia el campo de monto, el campo de comprobante y el campo de notas.
- Establece el método de pago predeterminado seleccionando el primer elemento del ComboBox.

```
public void HabilitarIngresoPago()
```

HabilitarListado()

Habilita la vista de listado de membresías y oculta los paneles de ingreso y pago.

```
public void HabilitarListado()
```

InitializeComponent()

```
private void InitializeComponent()
```

btnActualizar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Actualizar".

- Verifica si hay una fila seleccionada en el DataGridView.
 - Si hay una selección:
 - Si la membresía seleccionada tiene estado "Activa", lanza una excepción e informa que no se puede actualizar una membresía activa.
 - Si la membresía es Inactiva:
 - Cambia el modo a edición estableciendo esNuevo en False.
 - Llama a [HabilitarIngreso\(\)](#) para mostrar el panel de ingreso.
 - Carga los datos de la membresía seleccionada en los campos de entrada.
 - Deshabilita la edición del miembro y oculta el botón de reinicio de miembro.
 - Cambia el texto del label para indicar el contexto de edición.
 - Si no hay ninguna fila seleccionada, lanza una excepción informando que no se ha seleccionado ningún miembro.

```
private void btnActualizar_Click(object sender, EventArgs e)
```

Parameters

sender object

e EventArgs

btnCancelarPago_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Cancelar Pago". Vuelve a la vista de listado de membresías.

```
private void btnCancelarPago_Click(object sender, EventArgs e)
```

Parameters

sender object

e EventArgs

btnCancelar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Cancelar". Vuelve a la vista de listado de membresías.

```
private void btnCancelar_Click(object sender, EventArgs e)
```

Parameters

sender object

e EventArgs

btnEliminar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Eliminar".

- Verifica si hay una fila seleccionada en el DataGridView de membresías.
 - Si no hay ninguna selección, lanza una excepción y muestra un mensaje de advertencia al usuario.
- Si hay una membresía seleccionada:
 - Obtiene el ID de la membresía a eliminar.

- Solicita confirmación al usuario mediante un cuadro de diálogo.
 - Si el usuario confirma la eliminación:
 - Llama a [Eliminar\(uint\)](#) para eliminar la membresía de la base de datos.
 - Actualiza el listado de membresías llamando a [ActualizarDgv\(\)](#). Muestra un mensaje de éxito al usuario.

```
private void btnEliminar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnGuardarPago_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Guardar Pago".

- Crea un nuevo objeto [Pagos](#) y le asigna los valores ingresados en el formulario.
- Instancia la capa de negocio [NPagos](#) y llama a [Insertar\(Pagos\)](#) para registrar el pago en la base de datos.
- Muestra un mensaje de éxito al usuario.
- Actualiza el listado de membresías para reflejar el nuevo estado.
- Vuelve a la vista de listado de membresías.

```
private void btnGuardarPago_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnGuardar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Guardar".

- Valida que haya un miembro y un plan seleccionados en los DataGridView correspondientes. Si no hay selección, lanza una excepción y muestra un mensaje.

- Crea un objeto [Membresias](#) y asigna los valores seleccionados de miembro y plan.
- Si la operación es de inserción (esNuevo = True):
 - Llama a [Insertar\(Membresias\)](#) para agregar la nueva membresía. Muestra un mensaje de éxito.
 - Pregunta al usuario si desea registrar un pago para la nueva membresía mediante un cuadro de diálogo.
 - Si el usuario acepta, habilita el panel de ingreso de pago.
 - Obtiene el ID de la membresía recién insertada usando [ObtenerIdMembresia\(Membresias\)](#), precarga el monto y método de pago, y detiene la ejecución para esperar el registro del pago.
- Si la operación es de actualización (esNuevo = False):
 - Asigna el ID de la membresía desde el campo correspondiente.
 - Llama a [Actualizar\(Membresias\)](#) para modificar la membresía existente. Muestra un mensaje de éxito.
- Finalmente, actualiza el listado de membresías y vuelve a la vista de listado.

```
private void btnGuardar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnInsertar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Insertar".

- Llama al método [HabilitarIngreso\(\)](#) para mostrar el panel de ingreso de datos.
- Establece la variable esNuevo en True para indicar que la operación es de alta.
- Cambia el texto del label para informar al usuario el contexto de la operación.

```
private void btnInsertar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

cb_SelectedIndexChanged(object, EventArgs)

Evento que se ejecuta al cambiar la opción seleccionada en el ComboBox de búsqueda de membresías.

- Al cambiar la opción, limpia el campo de búsqueda y actualiza el listado de membresías mostrando todos los registros mediante [ActualizarDgv\(\)](#).
- Si la opción seleccionada es "Activa" (índice 2):
 - Filtra y muestra solo las membresías activas utilizando [ListarPorEstado\(string\)](#) con el parámetro "Activa".
 - Deshabilita el campo de búsqueda.
- Si la opción seleccionada es "Inactiva" (índice 3):
 - Filtra y muestra solo las membresías inactivas utilizando [ListarPorEstado\(string\)](#) con el parámetro "Inactiva".
 - Deshabilita el campo de búsqueda.
- Para cualquier otra opción (por ejemplo, búsqueda por DNI o por nombre de plan):
 - Habilita el campo de búsqueda, para permitir la búsqueda dinámica.

```
private void cb_SelectedIndexChanged(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

frmMembresias_Load(object, EventArgs)

Evento que se ejecuta al cargar el formulario. Inicializa el listado de membresías con [Listar\(\)](#), configura las columnas del DataGridView y establece una opción de búsqueda del ComboBox.

```
private void frmMembresias_Load(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

pbReiniciarMiembro_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón de reinicio pbReiniciarMiembro. Limpia el campo de búsqueda de miembros tbBuscarMiembro y actualiza el DataGridView dgvMiembro mostrando la lista completa de miembros, utilizando el método [Listar\(\)](#).

```
private void pbReiniciarMiembro_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

pbReiniciarPlan_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón de reinicio pbReiniciarPlan. Limpia el campo de búsqueda de planes tbBuscarPlan y actualiza el DataGridView dgvPlan mostrando la lista completa de planes, utilizando el método [Listar\(\)](#).

```
private void pbReiniciarPlan_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

pbReiniciar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón de reinicio en la gestión de membresías. Según la opción seleccionada en el ComboBox cbOpcionBuscar:

- Si la opción es "DNI" o "Nombre de Plan" (índices 0 o 1), limpia el campo de búsqueda y actualiza el listado de membresías mostrando todos los registros.
- Si la opción es "Activa" o "Inactiva" (índices 2 o 3), restablece la opción del ComboBox a "DNI" (índice 0) y actualiza el listado mostrando todos los registros.

```
private void pbReiniciar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

tbBuscarMembresias_TextChanged(object, EventArgs)

Evento que se ejecuta cada vez que cambia el texto en el campo de búsqueda de membresías.

- Según la opción seleccionada en el ComboBox:
 - Si la opción es índice 0, realiza una búsqueda de membresías filtrando por el DNI del miembro usando [ListarPorDni\(string\)](#).
 - Si la opción es índice 1, realiza una búsqueda de membresías filtrando por el nombre del plan usando [ListarPorNombrePlan\(string\)](#).
- El resultado de la búsqueda se muestra en el DataGridView de membresías mediante el método [ActualizarDgv\(\)](#).

```
private void tbBuscarMembresias_TextChanged(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

tbBuscarMiembro_TextChanged(object, EventArgs)

Evento que se ejecuta cada vez que cambia el texto en el campo de búsqueda de miembros tbBuscarMiembro en el panel de ingreso de membresías. Según la opción seleccionada en el ComboBox cbOpcionBuscarMiembro:

- Si la opción es "Nombre" (índice 0), busca miembros utilizando [ListarPorNombre\(string\)](#).
- Si la opción es "DNI" (índice 1), busca miembros cuyo utilizando [ObtenerPorDni\(string\)](#). El resultado de la búsqueda se muestra en el DataGridView dgvMiembro.

```
private void tbBuscarMiembro_TextChanged(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

tbBuscarPlan_TextChanged(object, EventArgs)

Evento que se ejecuta cada vez que cambia el texto en el campo de búsqueda de planes.

- Según la opción seleccionada en el ComboBox:
 - Si la opción es "Nombre" (índice 0), realiza una búsqueda de planes filtrando por el nombre usando [ListarPorNombre\(string\)](#).
- El resultado de la búsqueda se muestra en el DataGridView de planes.

```
private void tbBuscarPlan_TextChanged(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

Class FrmMiembros

Namespace: [Gimnasio.Presentacion](#)

Formulario para la gestión de miembros en el sistema del gimnasio. Permite listar, buscar, agregar, actualizar y eliminar miembros. Utiliza la clase [NMiembros](#) para la lógica de negocio y la clase [Miembros](#) como entidad. Todas las operaciones de esta capa están envueltas en bloques Try...Catch. El manejo de errores se realiza a través de [Mostrar\(string, Exception\)](#), que permite guardar el error en un log.txt y a su vez mostrar un mensaje al usuario.

```
[DesignerGenerated]  
public class FrmMiembros : Form, IOleControl.Interface, IOleObject.Interface,  
IOleInPlaceObject.Interface, IOleInPlaceActiveObject.Interface, IOleWindow.Interface,  
IVIEWObject2.Interface, IVIEWObject.Interface, IPersistStreamInit.Interface,  
IPersistPropertyBag.Interface, IPersistStorage.Interface, IPersist.Interface,  
IQuickActivate.Interface, ISupportOleDropSource, IDropTarget, ISynchronizeInvoke,  
IWin32Window, IBindableComponent, IKeyboardToolTip, IHandle<HWND>, IArrangedElement,  
IComponent, IDisposable, IContainerControl
```

Inheritance

```
object ↳ ← MarshalByRefObject ↳ ← Component ↳ ← Control ↳ ← ScrollableControl ↳ ←  
ContainerControl ↳ ← Form ↳ ← FrmMiembros
```

Constructors

FrmMiembros(FrmPrincipal)

Este constructor se utiliza para inicializar el formulario y establecer la referencia al formulario principal.

```
public FrmMiembros(FrmPrincipal frmPrincipal)
```

Parameters

frmPrincipal [FrmPrincipal](#)

Instancia de [FrmPrincipal](#).

Fields

components

`private IContainer components`

Field Value

[IContainer](#)

esNuevo

Indica si la operación actual es de inserción (True) o actualización (False).

`private bool esNuevo`

Field Value

[bool](#)

frmPrincipal

Referencia al formulario principal para cambiar a la pestaña de membresías y acceder al usuario actual.

`private FrmPrincipal frmPrincipal`

Field Value

[FrmPrincipal](#)

nMiembros

Instancia de la capa de negocio para miembros.

`private NMiembros nMiembros`

Field Value

[NMiembros](#)

Properties

Label1

```
internal virtual Label Label1 { get; set; }
```

Property Value

[Label](#) ↗

Label2

```
internal virtual Label Label2 { get; set; }
```

Property Value

[Label](#) ↗

Label3

```
internal virtual Label Label3 { get; set; }
```

Property Value

[Label](#) ↗

Label5

```
internal virtual Label Label5 { get; set; }
```

Property Value

[Label](#) ↗

Label6

```
internal virtual Label Label6 { get; set; }
```

Property Value

[Label](#) ↗

btnActualizar

```
internal virtual Button btnActualizar { get; set; }
```

Property Value

[Button](#) ↗

btnCancelar

```
internal virtual Button btnCancelar { get; set; }
```

Property Value

[Button](#) ↗

btnEliminar

```
internal virtual Button btnEliminar { get; set; }
```

Property Value

[Button](#)

btnGuardar

```
internal virtual Button btnGuardar { get; set; }
```

Property Value

[Button](#)

btnInsertar

```
internal virtual Button btnInsertar { get; set; }
```

Property Value

[Button](#)

cbGenero

```
internal virtual ComboBox cbGenero { get; set; }
```

Property Value

[ComboBox](#)

cbOpcionBuscar

```
internal virtual ComboBox cbOpcionBuscar { get; set; }
```

Property Value

[ComboBox](#)

dgvListado

```
internal virtual DataGridView dgvListado { get; set; }
```

Property Value

[DataGridView](#)

labelTelefono

```
internal virtual Label labelTelefono { get; set; }
```

Property Value

[Label](#)

lblDatosIngreso

```
internal virtual Label lblDatosIngreso { get; set; }
```

Property Value

[Label](#)

lblTotal

```
internal virtual Label lblTotal { get; set; }
```

Property Value

[Label](#)

panelDatosIngreso

```
internal virtual Panel panelDatosIngreso { get; set; }
```

Property Value

[Panel](#)

panelListado

```
internal virtual Panel panelListado { get; set; }
```

Property Value

[Panel](#)

pbReiniciar

```
internal virtual PictureBox pbReiniciar { get; set; }
```

Property Value

[PictureBox](#)

tbApellido

```
internal virtual TextBox tbApellido { get; set; }
```

Property Value

[TextBox](#)

tbBuscar

```
internal virtual TextBox tbBuscar { get; set; }
```

Property Value

[TextBox](#)

tbDni

```
internal virtual TextBox tbDni { get; set; }
```

Property Value

[TextBox](#)

tbEmail

```
internal virtual TextBox tbEmail { get; set; }
```

Property Value

[TextBox](#)

tbID

```
internal virtual TextBox tbID { get; set; }
```

Property Value

[TextBox](#)

tbNombre

```
internal virtual TextBox tbNombre { get; set; }
```

Property Value

[TextBox](#)

tbTelefono

```
internal virtual TextBox tbTelefono { get; set; }
```

Property Value

[TextBox](#)

Methods

ActualizarDgv()

Actualiza el DataGridView con la lista completa de miembros obtenida desde la capa de negocio con [Listar\(\)](#). También actualiza la etiqueta con el total de miembros.

```
public void ActualizarDgv()
```

ActualizarDgv(DataTable)

Actualiza el DataGridView con un DataTable específico de miembros, por ejemplo, tras una búsqueda o filtrado. También actualiza la etiqueta con el total de miembros.

```
public void ActualizarDgv(DataTable listado)
```

Parameters

listado [DataTable](#)

es un DataTable con la información de los miembros a mostrar.

Dispose(bool)

Disposes of the resources (other than memory) used by the [Form](#).

```
protected override void Dispose(bool disposing)
```

Parameters

disposing [bool](#)

[true](#) to release both managed and unmanaged resources; [false](#) to release only unmanaged resources.

HabilitarIngreso()

- Hace visible el panel de ingreso de datos y lo ajusta para ocupar todo el espacio disponible.
- Deshabilita el panel de listado de miembros para evitar la interacción con el listado mientras se está ingresando o editando un miembro.
- Limpia los campos de entrada para asegurar que no queden datos residuales de operaciones anteriores.

```
public void HabilitarIngreso()
```

HabilitarListado()

Habilita el panel de listado de miembros y oculta el panel de ingreso de datos.

```
public void HabilitarListado()
```

InitializeComponent()

```
private void InitializeComponent()
```

btnActualizar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Actualizar".

- Verifica si hay una fila seleccionada en el DataGridView.
 - Si hay una selección:
 - Cambia el modo a edición estableciendo esNuevo en False.
 - Llama a [HabilitarIngreso\(\)](#) para mostrar el panel de ingreso.
 - Carga los datos del miembro seleccionado en los campos de entrada.
 - Cambia el texto del label a "Actualizar Miembro" para indicar el contexto de edición.
 - Si no hay selección, lanza una excepción indicando que no se ha seleccionado ningún miembro.

```
private void btnActualizar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnCancelar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Cancelar". Vuelve a la vista de listado de miembros.

```
private void btnCancelar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnEliminar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Eliminar".

- Verifica si hay una fila seleccionada en el DataGridView.
 - Si hay una selección:
 - Obtiene el identificador del miembro id_miembro de la fila seleccionada.
 - Solicita confirmación al usuario mediante un cuadro de diálogo.
 - Si el usuario confirma, invoca [Eliminar\(uint\)](#) para eliminar el miembro.

- Actualiza el listado de miembros llamando a [ActualizarDgv\(\)](#) y muestra un mensaje de éxito.
- Si no hay ninguna fila seleccionada, lanza una excepción con un mensaje.

```
private void btnEliminar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnGuardar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Guardar".

- Valida que los campos obligatorios (Dni, Nombre, Apellido) no estén vacíos.
- Si la validación falla, lanza una excepcion con el mensaje.
- Si las validaciones son correctas:
 - Crea una instancia de [Miembros](#) y carga los datos desde los campos de entrada.
 - Si esNuevo es True:
 - Utiliza [Insertar\(Miembros\)](#) para agregar el miembro y muestra un mensaje de éxito.
 - Ofrece la opción de agregar una membresía al nuevo miembro. Si el usuario acepta, abre el formulario de membresías con [MostrarFormulario\(Form\)](#) y llama con el DNI a [HabilitarIngresoConMiembro\(string\)](#).
 - Si esNuevo esFalse:
 - Asigna el IdMiembro y utiliza [Actualizar\(Miembros\)](#) para modificar el miembro existente, mostrando un mensaje de éxito.
 - Actualiza el listado de miembros llamando a [ActualizarDgv\(\)](#) y vuelve a la vista de listado con [HabilitarListado\(\)](#).

```
private void btnGuardar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnInsertar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Insertar".

- Llama al método [HabilitarIngreso\(\)](#) para mostrar el panel de ingreso de datos.
- Establece la variable esNuevo en True para indicar que la operación es de alta.
- Cambia el texto del label a "Agregar Miembro" para informar al usuario el contexto de la operación.

```
private void btnInsertar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

cb_SelectedIndexChanged(object, EventArgs)

Al cambiar de opción del ComboBox limpia el campo de búsqueda y actualiza al listado completo de miembros

```
private void cb_SelectedIndexChanged(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

frmMiembros_Load(object, EventArgs)

Evento que se ejecuta al cargar el formulario

- Llama a [ActualizarDgv\(\)](#) para cargar y mostrar la lista completa de miembros en el DataGridView.
- Configura la visibilidad y los encabezados de las columnas del DataGridView para mostrar los datos relevantes de los miembros
- Establece la opción predeterminada del ComboBox de búsqueda en la primera opción (índice 0).

```
private void frmMiembros_Load(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

pbReiniciar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Reiniciar". Limpia el campo de búsqueda y actualiza al listado completo de miembros.

```
private void pbReiniciar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

tbBuscar_TextChanged(object, EventArgs)

Evento que se ejecuta cada vez que cambia el texto en el campo de búsqueda ([tbBuscar](#)).

- Si la opción seleccionada es "Nombre" (índice 0), busca miembros por nombre utilizando [ListarPorNombre\(string\)](#) y actualiza el listado llamando a [ActualizarDgv\(DataTable\)](#).
- Si la opción seleccionada es "DNI" (índice 1), busca miembros por DNI utilizando [ListarPorDni\(string\)](#) y actualiza el listado llamando a [ActualizarDgv\(DataTable\)](#).

```
private void tbBuscar_TextChanged(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

Class FrmPagos

Namespace: [Gimnasio.Presentacion](#)

Formulario para la gestión y consulta de pagos en el sistema del gimnasio. Permite listar, buscar, actualizar y eliminar pagos. Utiliza la clase [NPagos](#) para la lógica de negocio y la clase [Pagos](#) como entidad. Todas las operaciones de esta capa están envueltas en bloques Try...Catch. El manejo de errores se realiza a través de [Mostrar\(string, Exception\)](#) que permite guardar el error en un log.txt y a su vez mostrar un mensaje al usuario.

```
[DesignerGenerated]  
public class FrmPagos : Form, IOleControl.Interface, IOleObject.Interface,  
IOleInPlaceObject.Interface, IOleInPlaceActiveObject.Interface, IOleWindow.Interface,  
IVIEWObject2.Interface, IVIEWObject.Interface, IPersistStreamInit.Interface,  
IPersistPropertyBag.Interface, IPersistStorage.Interface, IPersist.Interface,  
IQuickActivate.Interface, ISupportOleDropSource, IDropTarget, ISynchronizeInvoke,  
IWin32Window, IBindableComponent, IKeyboardToolTip, IHandle<HWND>, IArrangedElement,  
IComponent, IDisposable, IContainerControl
```

Inheritance

```
object ↳ ← MarshalByRefObject ↳ ← Component ↳ ← Control ↳ ← ScrollableControl ↳ ←  
ContainerControl ↳ ← Form ↳ ← FrmPagos
```

Constructors

FrmPagos(Usuarios)

Constructor del formulario de pagos. Si es recepcionista oculta el botón de Actualizar y Eliminar.

```
public FrmPagos(Usuarios usuario)
```

Parameters

usuario [Usuarios](#)

Instancia de [Usuarios](#) que representa al usuario logueado.

Fields

components

```
private.IContainer components
```

Field Value

[.IContainer](#)

nPagos

Instancia de la capa de negocio para pagos.

```
private NPagos nPagos
```

Field Value

[NPagos](#)

Properties

BtnBuscarFecha

```
internal virtual Button BtnBuscarFecha { get; set; }
```

Property Value

[Button](#)

Button1

```
internal virtual Button Button1 { get; set; }
```

Property Value

[Button](#) ↗

Button2

```
internal virtual Button Button2 { get; set; }
```

Property Value

[Button](#) ↗

Button3

```
internal virtual Button Button3 { get; set; }
```

Property Value

[Button](#) ↗

Label4

```
internal virtual Label Label4 { get; set; }
```

Property Value

[Label](#) ↗

Label5

```
internal virtual Label Labels5 { get; set; }
```

Property Value

[Label](#)

PanelFecha

```
internal virtual Panel PanelFecha { get; set; }
```

Property Value

[Panel](#)

PanelMonto

```
internal virtual Panel PanelMonto { get; set; }
```

Property Value

[Panel](#)

btnActualizar

```
internal virtual Button btnActualizar { get; set; }
```

Property Value

[Button](#)

btnBuscarMonto

```
internal virtual Button btnBuscarMonto { get; set; }
```

Property Value

[Button](#)

btnCancelarPago

```
internal virtual Button btnCancelarPago { get; set; }
```

Property Value

[Button ↗](#)

btnEliminar

```
internal virtual Button btnEliminar { get; set; }
```

Property Value

[Button ↗](#)

btnGuardarPago

```
internal virtual Button btnGuardarPago { get; set; }
```

Property Value

[Button ↗](#)

cbMetodo

```
internal virtual ComboBox cbMetodo { get; set; }
```

Property Value

[ComboBox ↗](#)

cbOpcionBuscar

```
internal virtual ComboBox cbOpcionBuscar { get; set; }
```

Property Value

[ComboBox](#)

dgvListado

```
internal virtual DataGridView dgvListado { get; set; }
```

Property Value

[DataGridView](#)

dtpFechaFin

```
internal virtual DateTimePicker dtpFechaFin { get; set; }
```

Property Value

[DateTimePicker](#)

dtpFechainicio

```
internal virtual DateTimePicker dtpFechainicio { get; set; }
```

Property Value

[DateTimePicker](#)

lblIngresosTotales

```
internal virtual Label lbIngresosTotales { get; set; }
```

Property Value

[Label](#) ↗

lblComprobante

```
internal virtual Label lblComprobante { get; set; }
```

Property Value

[Label](#) ↗

lblDesde

```
internal virtual Label lblDesde { get; set; }
```

Property Value

[Label](#) ↗

lblFin

```
internal virtual Label lblFin { get; set; }
```

Property Value

[Label](#) ↗

lblHasta

```
internal virtual Label lblHasta { get; set; }
```

Property Value

[Label](#) ↗

lblInicio

```
internal virtual Label lblInicio { get; set; }
```

Property Value

[Label](#) ↗

lblMetodo

```
internal virtual Label lblMetodo { get; set; }
```

Property Value

[Label](#) ↗

lblMonto

```
internal virtual Label lblMonto { get; set; }
```

Property Value

[Label](#) ↗

lblNotas

```
internal virtual Label lblNotas { get; set; }
```

Property Value

[Label](#)

lblPagoIngreso

```
internal virtual Label lblPagoIngreso { get; set; }
```

Property Value

[Label](#)

lblTotal

```
internal virtual Label lblTotal { get; set; }
```

Property Value

[Label](#)

panelEdicionPago

```
internal virtual Panel panelEdicionPago { get; set; }
```

Property Value

[Panel](#)

panelListado

```
internal virtual Panel panelListado { get; set; }
```

Property Value

[Panel](#) ↗

pbReiniciar

```
internal virtual PictureBox pbReiniciar { get; set; }
```

Property Value

[PictureBox](#) ↗

tbBuscar

```
internal virtual TextBox tbBuscar { get; set; }
```

Property Value

[TextBox](#) ↗

tbComprobante

```
internal virtual TextBox tbComprobante { get; set; }
```

Property Value

[TextBox](#) ↗

tblDpago

```
internal virtual TextBox tbIDpago { get; set; }
```

Property Value

[TextBox](#)

tbMonto

```
internal virtual TextBox tbMonto { get; set; }
```

Property Value

[TextBox](#)

tbMontoFinal

```
internal virtual TextBox tbMontoFinal { get; set; }
```

Property Value

[TextBox](#)

tbMontoInicial

```
internal virtual TextBox tbMontoInicial { get; set; }
```

Property Value

[TextBox](#)

tbNotas

```
internal virtual TextBox tbNotas { get; set; }
```

Property Value

[TextBox](#)

Methods

ActualizarDgv()

Actualiza el DataGridView con la lista completa de pagos obtenida desde la capa de negocio mediante [Listar\(\)](#). Actualiza la etiqueta `lblTotal` con la cantidad de pagos listados y calcula el total de ingresos sumando los valores de la columna "monto" de cada fila, mostrando el resultado en `lblIngresosTotales`.

```
public void ActualizarDgv()
```

ActualizarDgv(DataTable)

Actualiza el DataGridView con una lista específica de pagos proporcionada en el parámetro `listado`. Actualiza la etiqueta `lblTotal` con la cantidad de pagos listados y calcula el total de ingresos sumando los valores de la columna "monto" de cada fila, mostrando el resultado en `lblIngresosTotales`.

```
public void ActualizarDgv(DataTable listado)
```

Parameters

`listado` [DataTable](#)

DataTable que contiene la lista de pagos a mostrar en el DataGridView.

Dispose(bool)

Disposes of the resources (other than memory) used by the [Form](#).

```
protected override void Dispose(bool disposing)
```

Parameters

`disposing bool`

`true` to release both managed and unmanaged resources; `false` to release only unmanaged resources.

HabilitarEdicionPago()

- Hace visible el panel de edición de pagos y lo ajusta para ocupar todo el espacio disponible del formulario.
- Deshabilita el panel de listado de pagos para evitar la interacción con el listado mientras se edita un pago.
- Limpia los campos de monto, comprobante y notas.
- Establece el método de pago predeterminado seleccionando el primer elemento del ComboBox.

`public void HabilitarEdicionPago()`

HabilitarListado()

Habilita el panel de listado de pagos. Oculta el panel de edición de pagos.

`public void HabilitarListado()`

InitializeComponent()

`private void InitializeComponent()`

btnActualizar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Actualizar" en el formulario de pagos.

- Verifica si hay una fila seleccionada en el DataGridView.
 - Si hay una selección:
 - Llama a [HabilitarEdicionPago\(\)](#) para mostrar el panel de edición.

- Carga en los campos de edición los datos del pago seleccionado.
- Si no hay ninguna fila seleccionada, lanza una excepción informando que no se ha seleccionado ningún pago.

```
private void btnActualizar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnBuscarFecha_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón para buscar pagos por fecha en el formulario de pagos.

- Obtiene las fechas seleccionadas en los controles dtpFechalInicio y dtpFechaFin.
- Ajusta la fecha de fin para incluir todo el día seleccionado (agregando un día y restando un tick).
- Valida que la fecha de inicio no sea mayor que la fecha de fin; si lo es, lanza una excepción.
- Si las fechas son válidas, utiliza el método [ListarPorFecha\(DateTime, DateTime\)](#) para filtrar los pagos dentro del rango especificado.
- Actualiza el DataGridView con los resultados filtrados mediante [ActualizarDgv\(\)](#).

```
private void btnBuscarFecha_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnBuscarMonto_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón para buscar pagos por monto (btnBuscarMonto). Permite filtrar los pagos mostrados en el DataGridView según un rango de montos especificado por el usuario.

- Valida que ambos campos de monto (tbMontolInicial y tbMontoFinal) no estén vacíos. Verifica tambien que sean numéricos positivos.
- Comprueba que el monto inicial no sea mayor que el monto final.

- Si todas las validaciones son correctas, invoca [ListarPorMontos\(decimal, decimal\)](#) pasando los valores convertidos a decimal y actualiza el DataGridView con los resultados filtrados.

```
private void btnBuscarMonto_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnCancelarPago_Click(object, EventArgs)

```
private void btnCancelarPago_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnEliminar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Eliminar" (btnEliminar).

- Verifica si hay una fila seleccionada en el DataGridView.
- Si hay una selección, obtiene el identificador del pago ("id_pago") de la fila seleccionada.
- Solicita confirmación al usuario mediante un cuadro de diálogo antes de proceder con la eliminación.
- Si el usuario confirma, invoca [Eliminar\(uint\)](#) para eliminar el pago y actualiza el listado de pago.
- Muestra un mensaje de éxito si la eliminación se realiza correctamente.
- Si no hay ninguna fila seleccionada, lanza una excepción.

```
private void btnEliminar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnGuardarPago_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Guardar" en el formulario de edición de pagos.

- Valida que el campo monto no este vacío y que sea un número válido, sino lanza una excepción.
- Si las validaciones son correctas, crea un nuevo objeto Pagos con los datos ingresados y lo actualiza en la base de datos.
- Muestra un mensaje de éxito y actualiza el listado de pagos.

```
private void btnGuardarPago_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

cb_SelectedIndexChanged(object, EventArgs)

Evento que se ejecuta al cambiar la opción seleccionada en el ComboBox de búsqueda (cbOpcionBuscar).

- Limpia los campos de búsqueda y restablece los controles de fecha y monto a sus valores iniciales.
- Actualiza el DataGridView con la lista completa de pagos.
- Según la opción elegida:
 - Opción 0 (por fecha): muestra el panel de fechas y oculta los demás campos de búsqueda.
 - Opción 1 (por DNI) y 2 (por nombre de plan): habilita el campo de texto para búsqueda y oculta los paneles de fecha y monto.
 - Opción 3 (por monto): muestra el panel de montos y oculta los demás campos.
 - Opciones 4 a 10 (por método de pago): oculta todos los campos de búsqueda y filtra automáticamente los pagos por el método seleccionado con [ListarPorMetodoPago\(string\)](#).

```
private void cb_SelectedIndexChanged(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

dtpFechaFin_ValueChanged(object, EventArgs)

Evento que se ejecuta al cambiar el valor del DateTimePicker para la fecha de fin. Le da un formato corto a la fecha para que sea visible.

```
private void dtpFechaFin_ValueChanged(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

dtpFechainicio_ValueChanged(object, EventArgs)

Evento que se ejecuta al cambiar el valor del DateTimePicker para la fecha de inicio. Le da un formato corto a la fecha para que sea visible.

```
private void dtpFechainicio_ValueChanged(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

frmPagos_Load(object, EventArgs)

Evento que se ejecuta al cargar el formulario FrmPagos. Inicializa el listado de pagos, configura la visibilidad y los encabezados de las columnas del DataGridView dgvListado, y establece la opción predeterminada del ComboBox de búsqueda. También inicializa los controles de selección de fecha dtpFechainicio y dtpFechaFin con un formato personalizado vacío.

```
private void frmPagos_Load(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

pbReiniciar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Reiniciar" (pbReiniciar) en el formulario de pagos.

- Restablece los campos de búsqueda y filtros según la opción seleccionada en el ComboBox:
 - Opción 0 (por fecha): reinicia los controles de fecha al valor actual y les aplica un formato vacío y actualiza el listado completo de pagos.
 - Opción 1 o 2 (por DNI o por nombre de plan): limpia el campo de búsqueda y actualiza el listado completo de pagos.
 - Opción 3 (por monto): limpia los campos de monto inicial y final y actualiza el listado completo de pagos.
 - Opciones 4 a 10 (por método de pago): actualiza el listado completo de pagos y restablece la opción seleccionada en el ComboBox a la opción predeterminada (0).

```
private void pbReiniciar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

tbBuscar_TextChanged(object, EventArgs)

Evento que se ejecuta al modificar el texto en el campo de búsqueda (tbBuscar). Su función es filtrar dinámicamente los pagos mostrados en el DataGridView según el criterio seleccionado en el ComboBox de búsqueda.

- Si la opción seleccionada es 1 (DNI), invoca [ListarPorDni\(string\)](#) pasando el texto ingresado y actualiza el listado.

- Si la opción seleccionada es 2 (Nombre plan), invoca [ListarPorNombrePlan\(string\)](#) con el texto ingresado y actualiza el listado.

```
private void tbBuscar_TextChanged(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

Class FrmPlanes

Namespace: [Gimnasio.Presentacion](#)

Formulario para la gestión de planes en el sistema de gimnasio. Permite listar, buscar, insertar, actualizar y eliminar planes. Utiliza la clase [NPlanes](#) para la lógica de negocio y la clase [Planes](#) como entidad. El acceso a las operaciones de mantenimiento depende del rol del usuario ([Usuarios](#)). Todas las operaciones de esta capa están envueltas en bloques Try...Catch. El manejo de errores se realiza a través de [Mostrar\(string, Exception\)](#) que permite guardar el error en un log.txt y a su vez mostrar un mensaje al usuario.

```
[DesignerGenerated]
public class FrmPlanes : Form, IOleControl.Interface, IOleObject.Interface,
IOleInPlaceObject.Interface, IOleInPlaceActiveObject.Interface, IOleWindow.Interface,
IVIEWObject2.Interface, IVIEWObject.Interface, IPersistStreamInit.Interface,
IPersistPropertyBag.Interface, IPersistStorage.Interface, IPersist.Interface,
IQuickActivate.Interface, ISupportOleDropSource, IDropTarget, ISynchronizeInvoke,
IWin32Window, IBindableComponent, IKeyboardToolTip, IHandle<HWND>, IArrangedElement,
IComponent, IDisposable, IContainerControl
```

Inheritance

```
object ↳ ← MarshalByRefObject ↳ ← Component ↳ ← Control ↳ ← ScrollableControl ↳ ←
ContainerControl ↳ ← Form ↳ ← FrmPlanes
```

Constructors

FrmPlanes(Usuarios)

Constructor del formulario.

- Inicializa los componentes visuales.
- Si el usuario tiene el rol de Recepcionista (IdRol = 2), deshabilita y oculta los botones de insertar, actualizar y eliminar.

```
public FrmPlanes(Usuarios usuario)
```

Parameters

usuario [Usuarios](#)

Instancia de [Usuarios](#) que representa al usuario logueado.

Fields

components

```
private.IContainer components
```

Field Value

[.IContainer](#)

esNuevo

Indica si la operación actual es de inserción (True) o actualización (False).

```
private bool esNuevo
```

Field Value

[bool](#)

nPlanes

Instancia de la capa de negocio para planes.

```
private readonly NPlanes nPlanes
```

Field Value

[NPlanes](#)

Properties

btnActualizar

```
internal virtual Button btnActualizar { get; set; }
```

Property Value

[Button ↗](#)

btnCancelar

```
internal virtual Button btnCancelar { get; set; }
```

Property Value

[Button ↗](#)

btnEliminar

```
internal virtual Button btnEliminar { get; set; }
```

Property Value

[Button ↗](#)

btnGuardar

```
internal virtual Button btnGuardar { get; set; }
```

Property Value

[Button ↗](#)

btnInsertar

```
internal virtual Button btnInsertar { get; set; }
```

Property Value

[Button](#)

cbOpcionBuscar

```
internal virtual ComboBox cbOpcionBuscar { get; set; }
```

Property Value

[ComboBox](#)

dgvListado

```
internal virtual DataGridView dgvListado { get; set; }
```

Property Value

[DataGridView](#)

lblDatosIngreso

```
internal virtual Label lblDatosIngreso { get; set; }
```

Property Value

[Label](#)

lblDescripcion

```
internal virtual Label lblDescripcion { get; set; }
```

Property Value

[Label](#) ↗

lblDuracion

```
internal virtual Label lblDuracion { get; set; }
```

Property Value

[Label](#) ↗

lblNombre

```
internal virtual Label lblNombre { get; set; }
```

Property Value

[Label](#) ↗

lblPrecio

```
internal virtual Label lblPrecio { get; set; }
```

Property Value

[Label](#) ↗

lblTotal

```
internal virtual Label lblTotal { get; set; }
```

Property Value

[Label](#) ↗

panelDatosIngreso

```
internal virtual Panel panelDatosIngreso { get; set; }
```

Property Value

[Panel](#) ↗

panelListado

```
internal virtual Panel panelListado { get; set; }
```

Property Value

[Panel](#) ↗

pbReiniciar

```
internal virtual PictureBox pbReiniciar { get; set; }
```

Property Value

[PictureBox](#) ↗

tbBuscar

```
internal virtual TextBox tbBuscar { get; set; }
```

Property Value

[TextBox](#)

tbDescripcion

```
internal virtual TextBox tbDescripcion { get; set; }
```

Property Value

[TextBox](#)

tbDuracion

```
internal virtual TextBox tbDuracion { get; set; }
```

Property Value

[TextBox](#)

tbID

```
internal virtual TextBox tbID { get; set; }
```

Property Value

[TextBox](#)

tbNombre

```
internal virtual TextBox tbNombre { get; set; }
```

Property Value

[TextBox](#)

tbPrecio

```
internal virtual TextBox tbPrecio { get; set; }
```

Property Value

[TextBox](#)

Methods

ActualizarDgv()

Actualiza el DataGridView con la lista de planes completa obtenida de la capa de negocio con [Listar\(\)](#). También actualiza la etiqueta que muestra el total de planes.

```
public void ActualizarDgv()
```

ActualizarDgv(DataTable)

Actualiza el DataGridView con un DataTable específico de planes, por ejemplo, tras una búsqueda o filtrado. También actualiza la etiqueta con el total de miembros.

```
public void ActualizarDgv(DataTable listado)
```

Parameters

listado [DataTable](#)

es un DataTable que contiene la lista de planes.

Dispose(bool)

Disposes of the resources (other than memory) used by the [Form](#).

```
protected override void Dispose(bool disposing)
```

Parameters

disposing [bool](#)

[true](#) to release both managed and unmanaged resources; [false](#) to release only unmanaged resources.

HabilitarIngreso()

- Hace visible el panel de ingreso de datos y lo ajusta para ocupar todo el espacio disponible.
- Deshabilita el panel de listado de planes para evitar la interacción con el listado mientras se está ingresando o editando un plan.
- Limpia los campos de entrada para asegurar que no queden datos residuales de operaciones anteriores.

```
public void HabilitarIngreso()
```

HabilitarListado()

Habilita la vista de listado de planes y oculta el panel de ingreso de datos.

```
public void HabilitarListado()
```

InitializeComponent()

```
private void InitializeComponent()
```

btnActualizar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en "Actualizar".

- Verifica si hay una fila seleccionada en el DataGridView.
 - Si hay una selección:
 - Cambia el modo a edición estableciendo esNuevo en False.
 - Llama a [HabilitarIngreso\(\)](#) para mostrar el panel de ingreso.
 - Carga los datos del plan seleccionado en los campos de entrada
 - Cambia el texto del label a "Actualizar Plan" para indicar el contexto de edición.
 - Si no hay selección, lanza una excepción indicando que no se ha seleccionado ningún plan.

```
private void btnActualizar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnCancelar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Cancelar". Vuelve a la vista de listado de planes.

```
private void btnCancelar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnEliminar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en "Eliminar".

- Verifica si hay una fila seleccionada en el DataGridView. Si hay una selección:
 - Obtiene el id_plan de la fila seleccionada.
 - Solicita confirmación al usuario mediante un cuadro de diálogo.
 - Si el usuario confirma, invoca [Eliminar\(uint\)](#) para eliminar el plan.
 - Actualiza el listado de planes llamando a [ActualizarDgv\(\)](#) y muestra un mensaje de éxito.

- Si no hay selección, lanza una excepción indicando que no se ha seleccionado ningún plan.

```
private void btnEliminar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnGuardar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en "Guardar".

- Valida que los campos obligatorios (Nombre, Duracion, Precio) no estén vacíos. Verifica que la duración y el precio sea un valor numérico y positivo.
- Si hay algún error en la validación, lanza una excepción con un mensaje descriptivo.
- Si las validaciones son correctas:
 - Crea una instancia de [Planes](#) y carga los datos desde los campos de entrada.
 - Si esNuevo es True, utiliza [Insertar\(Planes\)](#) para agregar el plan y muestra un mensaje de éxito.
 - Si esNuevo es False, utiliza [Actualizar\(Planes\)](#) para modificar el plan existente y muestra un mensaje de éxito.
 - Actualiza el listado de planes llamando a [ActualizarDgv\(\)](#) y vuelve a la vista de listado con [HabilitarListado\(\)](#).

```
private void btnGuardar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnInsertar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Insertar".

- Habilita el panel de ingreso con [HabilitarIngreso\(\)](#).
- Establece la variable esNuevo en True para indicar que la operación es de alta.

- Cambia el texto del label a "Agregar Plan" para informar al usuario el contexto de la operación.

```
private void btnInsertar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

cb_SelectedIndexChanged(object, EventArgs)

Evento que se ejecuta al cambiar la opción seleccionada en el ComboBox de búsqueda. Se encarga de:

- Limpiar el campo de búsqueda
- Actualiza el DataGridView con el listado completo de planes llamando a [ActualizarDgv\(\)](#).

```
private void cb_SelectedIndexChanged(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

frmPlanes_Load(object, EventArgs)

Evento que se ejecuta al cargar el formulario. Actualiza el listado de planes con [ActualizarDgv\(\)](#), configura las columnas del DataGridView y establece una opción de búsqueda del ComboBox.

```
private void frmPlanes_Load(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

pbReiniciar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Reiniciar". Actualiza el listado de planes y limpia el campo de búsqueda.

```
private void pbReiniciar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

tbBuscar_KeyDown(object, KeyEventArgs)

Evento que se ejecuta al presionar una tecla en el campo de búsqueda.

- Si se presiona la tecla Enter y la opción seleccionada es 1 (búsqueda por duración):
 - Valida que el texto ingresado sea numérico y positivo.
 - Si la validación es correcta, invoca [ListarPorDuracion\(uint\)](#) y actualiza el DataGridView con los resultados.
 - Si la validación falla, lanza una excepción con un mensaje descriptivo.
- Si se presiona la tecla Enter y la opción seleccionada es 2 (búsqueda por precio):
 - Valida que el texto ingresado sea numérico y positivo.
 - Si la validación es correcta, invoca [ListarPorPrecio\(decimal\)](#) y actualiza el DataGridView con los resultados.
 - Si la validación falla, lanza una excepción con un mensaje descriptivo.

```
private void tbBuscar_KeyDown(object sender, KeyEventArgs e)
```

Parameters

sender [object](#)

e [KeyEventArgs](#)

tbBuscar_TextChanged(object, EventArgs)

Evento que se ejecuta al modificar el texto del campo de búsqueda.

- Si la opción seleccionada en el ComboBox es 0 (búsqueda por nombre), invoca [ListarPorNombre\(string\)](#) pasando el texto ingresado y actualiza el listado de planes llamando a [ActualizarDgv\(DataTable\)](#).

```
private void tbBuscar_TextChanged(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

Class FrmPrincipal

Namespace: [Gimnasio.Presentacion](#)

Formulario principal de la aplicación del gimnasio. Gestiona la navegación entre los diferentes módulos y controla el acceso según el rol del usuario. Utiliza la clase [Usuarios](#) para identificar al usuario logueado y su rol. Llama a formularios secundarios como [FrmPlanes](#), [FrmMiembros](#), [FrmMembresias](#), [FrmPagos](#), [FrmRegistroAsistencias](#), [FrmReclamos](#) y [FrmUsuarios](#). Todas las operaciones de esta capa están envueltas en bloques Try...Catch. El manejo de errores se realiza a través de [Mostrar\(string, Exception\)](#) que permite guardar el error en un log.txt y a su vez mostrar un mensaje al usuario.

```
[DesignerGenerated]  
public class FrmPrincipal : Form, IOleControl.Interface, IOleObject.Interface,  
IOleInPlaceObject.Interface, IOleInPlaceActiveObject.Interface, IOleWindow.Interface,  
IVIEWObject2.Interface, IVIEWObject.Interface, IPersistStreamInit.Interface,  
IPersistPropertyBag.Interface, IPersistStorage.Interface, IPersist.Interface,  
IQuickActivate.Interface, ISupportOleDropSource, IDropTarget, ISynchronizeInvoke,  
IWin32Window, IBindableComponent, IKeyboardToolTip, IHandle<HWND>, IArrangedElement,  
IComponent, IDisposable, IContainerControl
```

Inheritance

```
object ↳ ← MarshalByRefObject ↳ ← Component ↳ ← Control ↳ ← ScrollableControl ↳ ←  
ContainerControl ↳ ← Form ↳ ← FrmPrincipal
```

Constructors

FrmPrincipal(Usuarios)

Constructor del formulario principal.

- Inicializa los componentes visuales del formulario.
- Asigna el usuario actualmente logueado recibido como parámetro.
- Muestra el nombre completo del usuario en el botón btnUsuarioLogueado.
- Si el usuario tiene el rol de recepcionista IdRol = 2:
 - Oculta y deshabilita el apartado de usuarios.

```
public FrmPrincipal(Usuarios usuario)
```

Parameters

usuario [Usuarios](#)

Instancia de [Usuarios](#) que representa al usuario logueado.

Fields

components

```
private IContainer components
```

Field Value

[IContainer](#) ↗

usuarioActual

Usuario actualmente logueado en la aplicación.

```
private Usuarios usuarioActual
```

Field Value

[Usuarios](#)

Properties

PanelDeFormularios

```
internal virtual Panel PanelDeFormularios { get; set; }
```

Property Value

[Panel](#) ↗

PictureBox1

```
internal virtual PictureBox PictureBox1 { get; set; }
```

Property Value

[PictureBox](#)

ToolStrip1

```
internal virtual ToolStrip ToolStrip1 { get; set; }
```

Property Value

[ToolStrip](#)

ToolStripSeparator1

```
internal virtual ToolStripSeparator ToolStripSeparator1 { get; set; }
```

Property Value

[ToolStripSeparator](#)

ToolStripSeparator2

```
internal virtual ToolStripSeparator ToolStripSeparator2 { get; set; }
```

Property Value

[ToolStripSeparator](#)

ToolStripSeparator3

```
internal virtual ToolStripSeparator ToolStripSeparator3 { get; set; }
```

Property Value

[ToolStripSeparator](#) ↗

ToolStripSeparator4

```
internal virtual ToolStripSeparator ToolStripSeparator4 { get; set; }
```

Property Value

[ToolStripSeparator](#) ↗

ToolStripSeparator5

```
internal virtual ToolStripSeparator ToolStripSeparator5 { get; set; }
```

Property Value

[ToolStripSeparator](#) ↗

ToolStripSeparator6

```
internal virtual ToolStripSeparator ToolStripSeparator6 { get; set; }
```

Property Value

[ToolStripSeparator](#) ↗

ToolStripSeparator7

```
internal virtual ToolStripSeparator ToolStripSeparator7 { get; set; }
```

Property Value

[ToolStripSeparator](#)

ToolStripSeparator8

```
internal virtual ToolStripSeparator ToolStripSeparator8 { get; set; }
```

Property Value

[ToolStripSeparator](#)

btRegistroAsistencias

```
internal virtual ToolStripButton btRegistroAsistencias { get; set; }
```

Property Value

[ToolStripButton](#)

btnMembresias

```
internal virtual ToolStripButton btnMembresias { get; set; }
```

Property Value

[ToolStripButton](#)

btnMiembros

```
internal virtual ToolStripButton btnMiembros { get; set; }
```

Property Value

[ToolStripButton](#)

btnPagos

```
internal virtual ToolStripButton btnPagos { get; set; }
```

Property Value

[ToolStripButton](#)

btnPlanes

```
internal virtual ToolStripButton btnPlanes { get; set; }
```

Property Value

[ToolStripButton](#)

btnReclamos

```
internal virtual ToolStripButton btnReclamos { get; set; }
```

Property Value

[ToolStripButton](#)

btnUsuarioLogueado

```
internal virtual ToolStripDropDownButton btnUsuarioLogueado { get; set; }
```

Property Value

[ToolStripDropDownButton](#)

btnUsuarios

```
internal virtual ToolStripButton btnUsuarios { get; set; }
```

Property Value

[ToolStripButton](#)

miCerrarSesión

```
internal virtual ToolStripMenuItem miCerrarSesión { get; set; }
```

Property Value

[ToolStripMenuItem](#)

Methods

Dispose(bool)

Disposes of the resources (other than memory) used by the [Form](#).

```
protected override void Dispose(bool disposing)
```

Parameters

disposing [bool](#)

`true` to release both managed and unmanaged resources; `false` to release only unmanaged resources.

InitializeComponent()

```
private void InitializeComponent()
```

MostrarFormulario(Form)

Muestra un formulario secundario dentro del panel principal de la aplicación.

- Si ya existe un formulario cargado en el panel y no es una instancia de [FrmRegistroAsistencias](#), lo elimina y libera sus recursos para evitar fugas de memoria.
- Limpia el panel de cualquier control previo.
- Configura el formulario recibido como parámetro para que no sea de nivel superior, sin bordes y ajustado al tamaño del panel principal.
- Agrega el formulario al panel y lo muestra.
- Permite mantener la instancia de [FrmRegistroAsistencias](#) viva entre cambios de formularios, evitando su destrucción.

```
public void MostrarFormulario(Form formulario)
```

Parameters

`formulario` [Form](#)

Instancia de [Form](#) a mostrar en el panel principal.

btRegistroAsistencias_Click(object, EventArgs)

Evento que abre el formulario de registro de asistencias [FrmRegistroAsistencias](#).

```
private void btRegistroAsistencias_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnMembresias_Click(object, EventArgs)

Evento que abre el formulario de membresías [FrmMembresias](#).

```
private void btnMembresias_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnMiembros_Click(object, EventArgs)

Evento que abre el formulario de miembros [FrmMiembros](#).

```
private void btnMiembros_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnPagos_Click(object, EventArgs)

Evento que abre el formulario de pagos [FrmPagos](#).

```
private void btnPagos_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnPlanes_Click(object, EventArgs)

Evento que abre el formulario de planes [FrmPlanes](#).

```
private void btnPlanes_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnReclamos_Click(object, EventArgs)

Evento que abre el formulario de reclamos [FrmReclamos](#).

```
private void btnReclamos_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnUsuarios_Click(object, EventArgs)

Evento que abre el formulario de usuarios [FrmUsuarios](#). Solo visible para usuarios con rol de administrador.

```
private void btnUsuarios_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

frmPrincipal_FormClosed(object, FormClosedEventArgs)

```
private void frmPrincipal_FormClosed(object sender, FormClosedEventArgs e)
```

Parameters

sender [object](#)

e [FormClosedEventArgs](#)

frmPrincipal_Load(object, EventArgs)

Evento que se ejecuta al cargar el formulario principal. Muestra un mensaje de bienvenida al usuario con su nombre y rol. Actualiza el estado de las membresías utilizando [ActualizaAEstadolnactiva\(\)](#).

```
private void frmPrincipal_Load(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

miCerrarSesion_Click(object, EventArgs)

- Se ejecuta al hacer clic en el elemento de menú "Cerrar Sesión".
- Solicita confirmación al usuario mediante un cuadro de diálogo.
- Si el usuario confirma la acción:
 - Oculta el formulario principal.
 - Muestra el formulario de login.
 - Restablece los campos del formulario de login llamando a [Formato\(\)](#).
 - Libera los recursos del formulario principal.

```
private void miCerrarSesion_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

obtenerUsuarioActual()

Devuelve el usuario actualmente logueado.

```
public Usuarios obtenerUsuarioActual()
```

Returns

[Usuarios](#)

Instancia de [Usuarios](#) correspondiente al usuario actual.

Class FrmReclamos

Namespace: [Gimnasio.Presentacion](#)

Formulario para la gestión de reclamos en el sistema del gimnasio. Permite listar, buscar, agregar, actualizar, responder, cambiar estado y eliminar reclamos. Utiliza la clase [nReclamos](#) para la lógica de negocio y la clase [Reclamos](#) como entidad. Todas las operaciones de esta capa están envueltas en bloques Try...Catch. El manejo de errores se realiza a través de [Mostrar\(string, Exception\)](#) que permite guardar el error en un log.txt y a su vez mostrar un mensaje al usuario.

```
[DesignerGenerated]  
public class FrmReclamos : Form, IOleControl.Interface, IOleObject.Interface,  
IOleInPlaceObject.Interface, IOleInPlaceActiveObject.Interface, IOleWindow.Interface,  
IVIEWObject2.Interface, IVIEWObject.Interface, IPersistStreamInit.Interface,  
IPersistPropertyBag.Interface, IPersistStorage.Interface, IPersist.Interface,  
IQuickActivate.Interface, ISupportOleDropSource, IDropTarget, ISynchronizeInvoke,  
IWin32Window, IBindableComponent, IKeyboardToolTip, IHandle<HWND>, IArrangedElement,  
IComponent, IDisposable, IContainerControl
```

Inheritance

```
object ← MarshalByRefObject ← Component ← Control ← ScrollableControl ←  
ContainerControl ← Form ← FrmReclamos
```

Constructors

FrmReclamos(Usuarios)

Constructor del formulario de reclamos. Inicializa los componentes del formulario. Si el rol del usuario es 2 (receptionista), oculta los botones de eliminar, cambiar estado y responder.

```
public FrmReclamos(Usuarios usuario)
```

Parameters

usuario [Usuarios](#)

Instancia de [Usuarios](#) que representa al usuario logueado.

Fields

components

```
private.IContainer components
```

Field Value

[.IContainer](#)

esNuevo

Indica si la operación actual es de inserción ([True](#)) o actualización ([False](#)).

```
private bool esNuevo
```

Field Value

[bool](#)

nReclamos

Instancia de la capa de negocio para reclamos.

```
private NReclamos nReclamos
```

Field Value

[NReclamos](#)

Properties

Label1

```
internal virtual Label Label1 { get; set; }
```

Property Value

[Label ↗](#)

Label2

```
internal virtual Label Label2 { get; set; }
```

Property Value

[Label ↗](#)

Label3

```
internal virtual Label Label3 { get; set; }
```

Property Value

[Label ↗](#)

Label4

```
internal virtual Label Label4 { get; set; }
```

Property Value

[Label ↗](#)

Label5

```
internal virtual Label Label5 { get; set; }
```

Property Value

[Label](#)

TbRespuesta

```
internal virtual TextBox TbRespuesta { get; set; }
```

Property Value

[TextBox](#)

btnActualizar

```
internal virtual Button btnActualizar { get; set; }
```

Property Value

[Button](#)

btnCambiarEstado

```
internal virtual Button btnCambiarEstado { get; set; }
```

Property Value

[Button](#)

btnCancelar

```
internal virtual Button btnCancelar { get; set; }
```

Property Value

[Button](#)

btnEliminar

```
internal virtual Button btnEliminar { get; set; }
```

Property Value

[Button ↗](#)

btnGuardar

```
internal virtual Button btnGuardar { get; set; }
```

Property Value

[Button ↗](#)

btnInsertar

```
internal virtual Button btnInsertar { get; set; }
```

Property Value

[Button ↗](#)

btnResponder

```
internal virtual Button btnResponder { get; set; }
```

Property Value

[Button ↗](#)

cbOpcionBuscar

```
internal virtual ComboBox cbOpcionBuscar { get; set; }
```

Property Value

[ComboBox](#)

cbTipo

```
internal virtual ComboBox cbTipo { get; set; }
```

Property Value

[ComboBox](#)

dgvListado

```
internal virtual DataGridView dgvListado { get; set; }
```

Property Value

[DataGridView](#)

lblDatosIngreso

```
internal virtual Label lblDatosIngreso { get; set; }
```

Property Value

[Label](#)

lblTotal

```
internal virtual Label lblTotal { get; set; }
```

Property Value

[Label](#)

panelDatosIngreso

```
internal virtual Panel panelDatosIngreso { get; set; }
```

Property Value

[Panel](#)

panelListado

```
internal virtual Panel panelListado { get; set; }
```

Property Value

[Panel](#)

panelRespuestaIngreso

```
internal virtual Panel panelRespuestaIngreso { get; set; }
```

Property Value

[Panel](#)

tbCancelarRespuesta

```
internal virtual Button tbCancelarRespuesta { get; set; }
```

Property Value

[Button](#)

tbDNI

```
internal virtual TextBox tbDNI { get; set; }
```

Property Value

[TextBox](#)

tbDescripcion

```
internal virtual TextBox tbDescripcion { get; set; }
```

Property Value

[TextBox](#)

tbGuardarRespuesta

```
internal virtual Button tbGuardarRespuesta { get; set; }
```

Property Value

[Button](#)

tbID

```
internal virtual TextBox tbID { get; set; }
```

Property Value

[TextBox](#)

Methods

ActualizarDgv()

Actualiza el DataGridView con la lista de reclamos obtenida desde la capa de negocio mediante [Listar\(\)](#).

- Obtiene todos los reclamos registrados en el sistema y los asigna como origen de datos del DataGridView.
- Actualiza la etiqueta `lblTotal` mostrando la cantidad total de reclamos listados.

```
public void ActualizarDgv()
```

ActualizarDgv(DataTable)

Actualiza el DataGridView con una lista específica de reclamos proporcionada como parámetro.

- Asigna el DataTable recibido `Listado` como origen de datos del DataGridView.
- Actualiza la etiqueta `lblTotal` mostrando la cantidad total de reclamos listados en el DataGridView.

```
public void ActualizarDgv(DataTable Listado)
```

Parameters

`Listado` [DataTable](#)

DataTable que contiene la lista de reclamos a mostrar en el DataGridView.

Dispose(bool)

Disposes of the resources (other than memory) used by the [Form](#).

```
protected override void Dispose(bool disposing)
```

Parameters

disposing [bool](#)

[true](#) to release both managed and unmanaged resources; [false](#) to release only unmanaged resources.

HabilitarIngreso()

Muestra el panel de datos de ingreso y lo ajusta para ocupar todo el espacio disponible. Deshabilita el panel de listado para evitar la interacción mientras se realiza el ingreso o edición. Selecciona la primera opción del ComboBox de tipo de reclamo. Limpia los campos de descripción, DNI y respuesta para asegurar que no contengan datos previos.

```
public void HabilitarIngreso()
```

HabilitarListado()

Habilita la vista de listado de reclamos y oculta los paneles de ingreso y respuesta.

```
public void HabilitarListado()
```

HabilitarRespuesta()

Muestra el panel de respuesta y lo ajusta para ocupar todo el espacio disponible en el formulario. Deshabilita el panel de listado para evitar la interacción con la lista de reclamos mientras se ingresa la respuesta. Limpia el campo de texto de respuesta para asegurar que no contenga datos previos.

```
public void HabilitarRespuesta()
```

InitializeComponent()

```
private void InitializeComponent()
```

btnActualizar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Actualizar".

- Verifica si hay una fila seleccionada en el DataGridView de reclamos.
- Si hay selección, habilita el panel de ingreso mediante [HabilitarIngreso\(\)](#) para editar el reclamo.
- Establece la variable esNuevo en False para indicar que la operación es una actualización.
- Carga en los controles de ingreso los datos del reclamo seleccionado (ID, tipo y descripción).
- Cambia el texto de la etiqueta lblDatosIngreso a "Actualizar Reclamo" para informar al usuario.
- Si no hay selección, lanza una excepción.

```
private void btnActualizar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnCambiarEstado_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Cambiar Estado".

- Verifica si hay una fila seleccionada en el DataGridView de reclamos.
- Si hay selección, obtiene el ID y el estado actual del reclamo seleccionado.
- Si el estado es "pendiente", utiliza [ActualizarElEstadoAResuelto\(uint\)](#) para cambiarlo a "resuelto" y muestra un mensaje de confirmación.
- Si el estado es "resuelto", utiliza [ActualizarElEstadoAPendiente\(uint\)](#) para cambiarlo a "pendiente" y muestra un mensaje de confirmación.
- Actualiza el listado de reclamos en el DataGridView mediante [ActualizarDgv\(\)](#).
- Si no hay selección, lanza una excepción.

```
private void btnCambiarEstado_Click(object sender, EventArgs e)
```

Parameters

sender object

e EventArgs

btnCancelar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Cancelar". Vuelve a la vista de listado de reclamos.

```
private void btnCancelar_Click(object sender, EventArgs e)
```

Parameters

sender object

e EventArgs

btnEliminar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Eliminar".

- Verifica si hay una fila seleccionada en el DataGridView de reclamos.
- Si hay selección, obtiene el ID del reclamo seleccionado y solicita confirmación al usuario mediante un cuadro de diálogo.
- Si el usuario confirma, utiliza [Eliminar\(uint\)](#) para eliminar el reclamo de la base de datos.
- Actualiza el listado de reclamos en el DataGridView mediante [ActualizarDgv\(\)](#) y muestra un mensaje de éxito.
- Si no hay selección, lanza una excepción.

```
private void btnEliminar_Click(object sender, EventArgs e)
```

Parameters

sender object

e EventArgs

btnGuardar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Guardar".

- Valida que los campos obligatorios de tipo y descripción estén completos; si no, lanza una excepción.
- Crea una instancia de [Reclamos](#) y asigna los valores de los controles del formulario.
- Si se ingresó un DNI, busca el miembro correspondiente usando [ObtenerPorDni\(string\)](#) y asigna su ID al reclamo;
- Si no se encuentra, lanza una excepción. Y si no se ingreso un DNI asigna Nothing al IdMiembro
- Si la operación es de inserción, utiliza [Insertar\(Reclamos\)](#) para agregar el reclamo y muestra un mensaje de éxito.
- Si la operación es de actualización, asigna el ID del reclamo y utiliza [Actualizar\(Reclamos\)](#), mostrando un mensaje de éxito.
- Actualiza el listado de reclamos mediante [ActualizarDgv\(\)](#) y vuelve a la vista de listado con [HabilitarListado\(\)](#).

```
private void btnGuardar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnInsertar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Insertar".

- Habilita el panel de ingreso para permitir la carga de un nuevo reclamo mediante [HabilitarIngreso\(\)](#).
- Cambia el texto de la etiqueta [lblDatosIngreso](#) a "Nuevo reclamo" para indicar la acción al usuario.
- Establece la variable [esNuevo](#) en True para señalar que la siguiente operación será una inserción.

```
private void btnInsertar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnResponder_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Responder".

- Verifica si hay una fila seleccionada en el DataGridView de reclamos.
- Si hay selección, habilita el panel de respuesta mediante [HabilitarRespuesta\(\)](#) para permitir ingresar o modificar la respuesta.
- Carga en los controles de respuesta el ID y la respuesta actual del reclamo seleccionado.
- Si no hay selección, lanza una excepción

```
private void btnResponder_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

cb_SelectedIndexChanged(object, EventArgs)

Evento que se ejecuta al cambiar la opción seleccionada en el ComboBox de búsqueda de reclamos.

- Filtra el listado de reclamos mostrado en el DataGridView según el estado seleccionado en el ComboBox (todos, pendiente o resuelto).
- Utiliza una estructura Select Case para determinar el filtro:
 - Opción 0: muestra todos los reclamos llamando a [ActualizarDgv\(\)](#).
 - Opción 1: muestra solo los reclamos pendientes llamando a [ListarPorEstado\(string\)](#) con el parámetro "pendiente".
 - Opción 2: muestra solo los reclamos resueltos llamando a [ListarPorEstado\(string\)](#) con el parámetro "resuelto".
- Actualiza el DataGridView con el resultado correspondiente.

```
private void cb_SelectedIndexChanged(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

frmReclamos_Load(object, EventArgs)

Evento que se ejecuta al cargar el formulario. Inicializa el listado de reclamos, configura las columnas del DataGridView y selecciona la 1º opción del ComboBox.

```
private void frmReclamos_Load(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

tbCancelarRespuesta_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Cancelar Respuesta". Vuelve a la vista de listado de reclamos.

```
private void tbCancelarRespuesta_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

tbGuardarRespuesta_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Guardar Respuesta".

- Valida que el campo de respuesta no esté vacío; si no lanza una excepción
- Crea una instancia de [Reclamos](#) y asigna la respuesta y el ID del reclamo seleccionado.
- Utiliza [ActualizarRespuesta\(Reclamos\)](#) para actualizar la respuesta del reclamo en la base de datos.
- Muestra un mensaje de éxito al usuario.
- Actualiza el listado de reclamos mediante [ActualizarDgv\(\)](#) y vuelve a la vista de listado con [Habilitar Listado\(\)](#).

```
private void tbGuardarRespuesta_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

Class FrmRegistroAsistencias

Namespace: [Gimnasio.Presentacion](#)

Formulario para la gestión y consulta de asistencias de los miembros en el sistema del gimnasio. Permite listar, buscar por DNI o fecha, y eliminar registros de asistencia. Utiliza la clase [NAsistencia](#) para la lógica de negocio. Todas las operaciones de esta capa están envueltas en bloques Try...Catch. El manejo de errores se realiza a través de [Mostrar\(string, Exception\)](#), que permite guardar el error en un log.txt y a su vez mostrar un mensaje al usuario.

```
[DesignerGenerated]
public class FrmRegistroAsistencias : Form, IOleControl.Interface, IOleObject.Interface,
IOleInPlaceObject.Interface, IOleInPlaceActiveObject.Interface, IOleWindow.Interface,
IVIEWObject2.Interface, IVIEWObject.Interface, IPersistStreamInit.Interface,
IPersistPropertyBag.Interface, IPersistStorage.Interface, IPersist.Interface,
IQuickActivate.Interface, ISupportOleDropSource, IDropTarget, ISynchronizeInvoke,
IWin32Window, IBindableComponent, IKeyboardToolTip, IHandle<HWND>, IArrangedElement,
IComponent, IDisposable, IContainerControl
```

Inheritance

```
object ↳ ← MarshalByRefObject ↳ ← Component ↳ ← Control ↳ ← ScrollableControl ↳ ←
ContainerControl ↳ ← Form ↳ ← FrmRegistroAsistencias
```

Constructors

FrmRegistroAsistencias(Usuarios)

Constructor del formulario de asistencias. Si el usuario es recepcionista, oculta el botón de eliminar.

```
public FrmRegistroAsistencias(Usuarios usuario)
```

Parameters

usuario [Usuarios](#)

Instancia de [Usuarios](#) que representa al usuario logueado.

Fields

components

```
private.IContainer components
```

Field Value

[.IContainer](#)

nAsistencias

Instancia de la capa de negocio para asistencias.

```
private NAsistencia nAsistencias
```

Field Value

[NAsistencia](#)

Properties

BtnBuscarFecha

```
internal virtual Button BtnBuscarFecha { get; set; }
```

Property Value

[Button](#)

Label1

```
internal virtual Label Label1 { get; set; }
```

Property Value

[Label](#) ↗

Label2

```
internal virtual Label Label2 { get; set; }
```

Property Value

[Label](#) ↗

Label3

```
internal virtual Label Label3 { get; set; }
```

Property Value

[Label](#) ↗

Label4

```
internal virtual Label Label4 { get; set; }
```

Property Value

[Label](#) ↗

PanelFecha

```
internal virtual Panel PanelFecha { get; set; }
```

Property Value

[Panel](#)

btnBuscar

```
internal virtual Button btnBuscar { get; set; }
```

Property Value

[Button](#)

btnEliminar

```
internal virtual Button btnEliminar { get; set; }
```

Property Value

[Button](#)

btnInsertar

```
internal virtual Button btnInsertar { get; set; }
```

Property Value

[Button](#)

cbOpcionBuscar

```
internal virtual ComboBox cbOpcionBuscar { get; set; }
```

Property Value

[ComboBox](#)

dgvListado

```
internal virtual DataGridView dgvListado { get; set; }
```

Property Value

[DataGridView](#)

dtpFechaFin

```
internal virtual DateTimePicker dtpFechaFin { get; set; }
```

Property Value

[DateTimePicker](#)

dtpFechalnicio

```
internal virtual DateTimePicker dtpFechalnicio { get; set; }
```

Property Value

[DateTimePicker](#)

lblTotal

```
internal virtual Label lblTotal { get; set; }
```

Property Value

[Label](#)

pbReiniciar

```
internal virtual PictureBox pbReiniciar { get; set; }
```

Property Value

[PictureBox](#)

tbBuscar

```
internal virtual TextBox tbBuscar { get; set; }
```

Property Value

[TextBox](#)

Methods

ActualizarDgv()

Actualiza el DataGridView con la lista de asistencias obtenida desde la capa de negocio mediante [Listar\(\)](#).

- Obtiene todos los registros de asistencias y los asigna como origen de datos del DataGridView.
- Actualiza la etiqueta lblTotal mostrando la cantidad total de asistencias listadas.

```
public void ActualizarDgv()
```

ActualizarDgv(DataTable)

Actualiza el DataGridView con una lista específica de asistencias proporcionada como parámetro.

- Asigna el DataTable recibido como origen de datos del DataGridView.
- Actualiza la etiqueta lblTotal mostrando la cantidad total de asistencias listadas en el DataGridView.

```
public void ActualizarDgv(DataTable Listado)
```

Parameters

Dispose(bool)

Disposes of the resources (other than memory) used by the [Form](#).

```
protected override void Dispose(bool disposing)
```

Parameters

disposing [bool](#)

[true](#) to release both managed and unmanaged resources; [false](#) to release only unmanaged resources.

InitializeComponent()

```
private void InitializeComponent()
```

btnBuscar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón para buscar asistencias por fecha.

- Obtiene las fechas seleccionadas en los controles DateTimePicker para inicio y fin.
- Valida que la fecha de inicio no sea mayor que la fecha de fin; si lo es, lanza una excepción.
- Llama a [ListarPorFecha\(DateTime, DateTime\)](#) para obtener las asistencias dentro del rango de fechas especificado.
- Actualiza el DataGridView con los resultados llamando a [ActualizarDgy\(\)](#).

```
private void btnBuscar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnEliminar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Eliminar".

- Verifica si hay una fila seleccionada en el DataGridView de asistencias.
- Si hay selección, obtiene el ID de la asistencia seleccionada y solicita confirmación al usuario mediante un cuadro de diálogo.
- Si el usuario confirma, utiliza [Eliminar\(uint\)](#) para eliminar el registro de asistencia de la base de datos.
- Actualiza el listado de asistencias en el DataGridView mediante [ActualizarDgv\(\)](#) y muestra un mensaje de éxito.
- Si no hay selección, lanza una excepción indicando que no se ha seleccionado ninguna asistencia para eliminar.

```
private void btnEliminar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnInsertar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Insertar" en el formulario de registro de asistencias.

- Crea una nueva instancia del formulario [FrmAsistencias](#), pasando como parámetro la instancia actual de [FrmRegistroAsistencias](#) para permitir la comunicación entre formularios.
- Muestra el formulario de registro de asistencias para permitir el ingreso de una nueva asistencia mediante el DNI del miembro.

```
private void btnInsertar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

cb_SelectedIndexChanged(object, EventArgs)

Evento que se ejecuta al cambiar la opción seleccionada en el ComboBox de búsqueda de asistencias.

- Actualiza el DataGridView mostrando todas las asistencias mediante [ActualizarDgv\(\)](#).
- Utiliza una estructura Select Case para determinar el tipo de búsqueda:
 - Opción 0: habilita el campo de texto para buscar por DNI y oculta el panel de búsqueda por fecha.
 - Opción 1: oculta y deshabilita el campo de texto de búsqueda y muestra el panel para búsqueda por rango de fechas.
- Permite alternar entre búsqueda por DNI o por fechas según la selección del usuario.

```
private void cb_SelectedIndexChanged(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

dtpFechaFin_ValueChanged(object, EventArgs)

Evento que se ejecuta al cambiar el valor del DateTimePicker para la fecha de fin. Cambia el formato a corto para mostrar solo la fecha.

```
private void dtpFechaFin_ValueChanged(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

dtpFechainicio_ValueChanged(object, EventArgs)

Evento que se ejecuta al cambiar el valor del DateTimePicker para la fecha de inicio. Cambia el formato a corto para mostrar solo la fecha.

```
private void dtpFechaInicio_ValueChanged(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

frmRegistroAsistencias_Load(object, EventArgs)

Evento que se ejecuta al cargar el formulario de registro de asistencias.

- Inicializa el listado de asistencias en el DataGridView llamando a [ActualizarDgy\(\)](#).
- Configura la visibilidad y los encabezados de las columnas del DataGridView para mostrar solo la información relevante y con títulos descriptivos.
- Establece la opción predeterminada del ComboBox de búsqueda en la segunda opción (índice 1).
- Configura los controles DateTimePicker para fecha de inicio y fin con un formato personalizado vacío

```
private void frmRegistroAsistencias_Load(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

pbReiniciar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Reiniciar".

- Si la opción seleccionada en el ComboBox es búsqueda por DNI (índice 0), limpia el campo de búsqueda y actualiza el DataGridView mostrando todas las asistencias.
- Si la opción seleccionada es búsqueda por fecha (índice 1), restablece los DateTimePicker de fecha de inicio y fin a la fecha actual, limpia sus formatos y actualiza el DataGridView mostrando todas las asistencias.

```
private void pbReiniciar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

tbBuscar_TextChanged(object, EventArgs)

Evento que se ejecuta al cambiar el texto en el campo de búsqueda de asistencias.

- Si la opción seleccionada en el ComboBox es la de búsqueda por DNI (índice 0), filtra las asistencias utilizando [ListarPorDNI\(string\)](#) con el texto ingresado.
- Actualiza el DataGridView con los resultados de la búsqueda llamando a [ActualizarDgy\(\)](#).

```
private void tbBuscar_TextChanged(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

Class FrmUsuarios

Namespace: [Gimnasio.Presentacion](#)

Formulario para la gestión de usuarios del sistema del gimnasio. Permite listar, buscar, agregar, actualizar y eliminar usuarios. Utiliza la clase [nUsuarios](#) para la lógica de negocio y la clase [Usuarios](#) como entidad. Todas las operaciones de esta capa están envueltas en bloques Try...Catch. El manejo de errores se realiza a través de [Mostrar\(string, Exception\)](#) que permite guardar el error en un log.txt y a su vez mostrar un mensaje al usuario.

```
[DesignerGenerated]  
public class FrmUsuarios : Form, IOleControl.Interface, IOleObject.Interface,  
IOleInPlaceObject.Interface, IOleInPlaceActiveObject.Interface, IOleWindow.Interface,  
IVIEWObject2.Interface, IVIEWObject.Interface, IPersistStreamInit.Interface,  
IPersistPropertyBag.Interface, IPersistStorage.Interface, IPersist.Interface,  
IQuickActivate.Interface, ISupportOleDropSource, IDropTarget, ISynchronizeInvoke,  
IWin32Window, IBindableComponent, IKeyboardToolTip, IHandle<HWND>, IArrangedElement,  
IComponent, IDisposable, IContainerControl
```

Inheritance

```
object ← MarshalByRefObject ← Component ← Control ← ScrollableControl ←  
ContainerControl ← Form ← FrmUsuarios
```

Fields

components

```
private IContainer components
```

Field Value

[IContainer](#)

esNuevo

Indica si la operación actual es de inserción ([True](#)) o actualización ([False](#)).

```
private bool esNuevo
```

Field Value

[bool ↗](#)

nUsuarios

Instancia de la capa de negocio para usuarios.

```
private NUsuarios nUsuarios
```

Field Value

[NUsuarios](#)

Properties

Label1

```
internal virtual Label Label1 { get; set; }
```

Property Value

[Label ↗](#)

Label2

```
internal virtual Label Label2 { get; set; }
```

Property Value

[Label ↗](#)

Label3

```
internal virtual Label Label3 { get; set; }
```

Property Value

[Label ↗](#)

Label6

```
internal virtual Label Label6 { get; set; }
```

Property Value

[Label ↗](#)

btnActualizar

```
internal virtual Button btnActualizar { get; set; }
```

Property Value

[Button ↗](#)

btnCancelar

```
internal virtual Button btnCancelar { get; set; }
```

Property Value

[Button ↗](#)

btnEliminar

```
internal virtual Button btnEliminar { get; set; }
```

Property Value

[Button ↗](#)

btnGuardar

```
internal virtual Button btnGuardar { get; set; }
```

Property Value

[Button ↗](#)

btnInsertar

```
internal virtual Button btnInsertar { get; set; }
```

Property Value

[Button ↗](#)

cbOpcionBuscar

```
internal virtual ComboBox cbOpcionBuscar { get; set; }
```

Property Value

[ComboBox ↗](#)

cbRol

```
internal virtual ComboBox cbRol { get; set; }
```

Property Value

[ComboBox](#)

dgvListado

```
internal virtual DataGridView dgvListado { get; set; }
```

Property Value

[DataGridView](#)

labelTelefono

```
internal virtual Label labelTelefono { get; set; }
```

Property Value

[Label](#)

lblDatosIngreso

```
internal virtual Label lblDatosIngreso { get; set; }
```

Property Value

[Label](#)

lblTotal

```
internal virtual Label lblTotal { get; set; }
```

Property Value

[Label](#)

panelDatosIngreso

```
internal virtual Panel panelDatosIngreso { get; set; }
```

Property Value

[Panel](#)

panelListado

```
internal virtual Panel panelListado { get; set; }
```

Property Value

[Panel](#)

pbMostrarContraseña

```
internal virtual PictureBox pbMostrarContraseña { get; set; }
```

Property Value

[PictureBox](#)

pbReiniciar

```
internal virtual PictureBox pbReiniciar { get; set; }
```

Property Value

[PictureBox](#)

tbBuscar

```
internal virtual TextBox tbBuscar { get; set; }
```

Property Value

[TextBox](#)

tbContraseña

```
internal virtual TextBox tbContraseña { get; set; }
```

Property Value

[TextBox](#)

tbEmail

```
internal virtual TextBox tbEmail { get; set; }
```

Property Value

[TextBox](#)

tbID

```
internal virtual TextBox tbID { get; set; }
```

Property Value

[TextBox](#)

tbNombreCompleto

```
internal virtual TextBox tbNombreCompleto { get; set; }
```

Property Value

[TextBox](#)

tbNombreUsuario

```
internal virtual TextBox tbNombreUsuario { get; set; }
```

Property Value

[TextBox](#)

Methods

ActualizarDgv()

Actualiza el DataGridView con la lista de usuarios obtenida desde la capa de negocio mediante [Listar\(\)](#).

- Obtiene todos los registros de usuarios y los asigna como origen de datos del DataGridView.
- Actualiza la etiqueta lblTotal mostrando la cantidad total de usuarios listados.

```
public void ActualizarDgv()
```

ActualizarDgv(DataTable)

Actualiza el DataGridView con una lista específica de usuarios proporcionada como parámetro.

- Asigna el DataTable recibido como origen de datos del DataGridView.
- Actualiza la etiqueta lblTotal mostrando la cantidad total de usuarios listados en el DataGridView.

```
public void ActualizarDgv(DataTable Listado)
```

Parameters

Listado [DataTable](#)

Dispose(bool)

Disposes of the resources (other than memory) used by the [Form](#).

```
protected override void Dispose(bool disposing)
```

Parameters

disposing [bool](#)

[true](#) to release both managed and unmanaged resources; [false](#) to release only unmanaged resources.

HabilitarIngreso()

- Muestra el panel de ingreso ([panelDatosIngreso](#)) y lo ajusta para ocupar todo el espacio disponible en el formulario.
- Deshabilita el panel de listado para evitar la interacción mientras se realiza el ingreso o edición.
- Limpia los campos de nombre de usuario y contraseña para asegurar que no contengan datos previos.

```
public void HabilitarIngreso()
```

HabilitarListado()

Habilita la vista de listado de usuarios y oculta el panel de ingreso.

```
public void HabilitarListado()
```

InitializeComponent()

```
private void InitializeComponent()
```

btnActualizar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Actualizar" en el formulario de usuarios.

- Verifica si hay una fila seleccionada en el DataGridView de usuarios.
- Si hay selección, prepara el formulario para la edición de un usuario existente:
 - Establece la variable esNuevo en False para indicar que la siguiente operación será una actualización.
 - Habilita el panel de ingreso de datos mediante [HabilitarIngreso\(\)](#).
 - Cambia el texto de la etiqueta lblDatosIngreso a "Actualizar Usuario".
 - Carga en los controles de ingreso los datos del usuario seleccionado
- Si no hay selección, lanza una excepción indicando que no se seleccionó ningún usuario para actualizar.

```
private void btnActualizar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnCancelar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Cancelar". Vuelve a la vista de listado de usuarios.

```
private void btnCancelar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnEliminar_Click(object, EventArgs)

```
private void btnEliminar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnGuardar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Guardar" en el formulario de usuarios.

- Valida que los campos obligatorios estén completos; si no lanza una excepción.
- Crea una instancia de [Usuarios](#) y asigna los valores de los controles del formulario.
- Si la operación es de inserción, utiliza [Insertar\(Usuarios\)](#) para agregar el usuario y muestra un mensaje de éxito.
- Si la operación es de actualización, asigna el ID del usuario y utiliza [Actualizar\(Usuarios\)](#) para modificar el usuario, mostrando un mensaje de éxito.
- Actualiza el listado de usuarios mediante [ActualizarDgv\(\)](#) y vuelve a la vista de listado con [Habilitar Listado\(\)](#).

```
private void btnGuardar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

btnInsertar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón "Insertar" en el formulario de usuarios.

- Habilita el panel de ingreso de datos mediante [HabilitarIngreso\(\)](#) para permitir la carga de un nuevo usuario.
- Establece la variable esNuevo en True para indicar que la siguiente operación será una inserción.
- Cambia el texto de la etiqueta lblDatosIngreso a "Agregar Usuario" para informar al usuario sobre la acción actual.

```
private void btnInsertar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

frmUsuarios_Load(object, EventArgs)

Evento que se ejecuta al cargar el formulario. Inicializa el listado de usuarios y configura las columnas del DataGridView. Selecciona una opción por defecto del comboBox.

```
private void frmUsuarios_Load(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

pbMostrarContraseña_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el ícono para mostrar u ocultar la contraseña. Cambia la visibilidad del campo de contraseña.

```
private void pbMostrarContraseña_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

pbReiniciar_Click(object, EventArgs)

Evento que se ejecuta al hacer clic en el botón o ícono de "Reiniciar" en el formulario de usuarios.

- Limpia el campo de búsqueda de usuarios ([tbBuscar](#)), eliminando cualquier texto ingresado por el usuario.
- Actualiza el listado usuarios llamando a [ActualizarDgv\(\)](#), mostrando nuevamente la lista completa de usuarios sin filtros.

```
private void pbReiniciar_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

tbBuscar_TextChanged(object, EventArgs)

Evento que se ejecuta al cambiar el texto en el campo de búsqueda de usuarios.

- Si la opción seleccionada en el ComboBox de búsqueda es la de buscar por nombre (índice 0), filtra los usuarios utilizando [ListarPorNombre\(string\)](#) con el texto ingresado.
- Actualiza el DataGridView con los resultados llamando a [ActualizarDgv\(\)](#).

```
private void tbBuscar_TextChanged(object sender, EventArgs e)
```

Parameters

sender [object](#)

e [EventArgs](#)

Namespace Gimnasio.Presentacion.My Classes

[MyProject](#)

Class MyProject

Namespace: [Gimnasio.Presentacion.My](#)

```
[HideModuleName]  
internal class MyProject
```

MyProject

Fields

m_AppObjectProvider

```
private static readonly MyProject.ThreadSafeObjectProvider<MyApplication>  
m_AppObjectProvider
```

Field Value

[MyProject.ThreadSafeObjectProvider](#)<MyApplication>

m_ComputerObjectProvider

```
private static readonly MyProject.ThreadSafeObjectProvider<MyComputer>  
m_ComputerObjectProvider
```

Field Value

[MyProject.ThreadSafeObjectProvider](#)<MyComputer>

m_MyFormsObjectProvider

```
private static MyProject.ThreadSafeObjectProvider<MyProject.MyForms> m_MyFormsObjectProvider
```

Field Value

[MyProject.ThreadSafeObjectProvider<MyProject.MyForms>](#)

m_MyWebServicesObjectProvider

```
private static readonly MyProject.ThreadSafeObjectProvider<MyProject.MyWebServices>
m_MyWebServicesObjectProvider
```

Field Value

[MyProject.ThreadSafeObjectProvider<MyProject.MyWebServices>](#)

m_UserObjectProvider

```
private static readonly MyProject.ThreadSafeObjectProvider<User> m_UserObjectProvider
```

Field Value

[MyProject.ThreadSafeObjectProvider<User>](#)

Properties

Application

```
internal static MyApplication Application { get; }
```

Property Value

MyApplication

Computer

```
internal static MyComputer Computer { get; }
```

Property Value

MyComputer

Forms

```
internal static MyProject.MyForms Forms { get; }
```

Property Value

[MyProject.MyForms](#)

User

```
internal static User User { get; }
```

Property Value

[User](#)

WebServices

```
internal static MyProject.MyWebServices WebServices { get; }
```

Property Value

[MyProject.MyWebServices](#)

Namespace Gimnasio.Presentacion.My.Resources

Classes

[Resources](#)

Clase de recurso fuertemente tipado, para buscar cadenas traducidas, etc.

Class Resources

Namespace: [Gimnasio.Presentacion.My.Resources](#)

Clase de recurso fuertemente tipado, para buscar cadenas traducidas, etc.

```
[HideModuleName]  
internal class Resources
```

Resources

Fields

resourceCulture

```
private static CultureInfo resourceCulture
```

Field Value

[CultureInfo](#)

resourceMan

```
private static ResourceManager resourceMan
```

Field Value

[ResourceManager](#)

Properties

Culture

Reemplaza la propiedad CurrentUICulture del subproceso actual para todas las búsquedas de recursos mediante esta clase de recurso fuertemente tipado.

```
internal static CultureInfo Culture { get; set; }
```

Property Value

[CultureInfo](#)

ResourceManager

Devuelve la instancia de ResourceManager almacenada en caché utilizada por esta clase.

```
internal static ResourceManager ResourceManager { get; }
```

Property Value

[ResourceManager](#)

casa

Busca un recurso adaptado de tipo System.Drawing.Bitmap.

```
internal static Bitmap casa { get; }
```

Property Value

[Bitmap](#)

ojoContraseña

Busca un recurso adaptado de tipo System.Drawing.Bitmap.

```
internal static Bitmap ojoContraseña { get; }
```

Property Value

[Bitmap](#)

ojos_abiertos

Busca un recurso adaptado de tipo System.Drawing.Bitmap.

```
internal static Bitmap ojos_abiertos { get; }
```

Property Value

[Bitmap](#)

ojos_cerrados

Busca un recurso adaptado de tipo System.Drawing.Bitmap.

```
internal static Bitmap ojos_cerrados { get; }
```

Property Value

[Bitmap](#)

reiniciar

Busca un recurso adaptado de tipo System.Drawing.Bitmap.

```
internal static Bitmap reiniciar { get; }
```

Property Value

[Bitmap](#)

usuario

Busca un recurso adaptado de tipo System.Drawing.Bitmap.

```
internal static Bitmap usuario { get; }
```

Property Value

[Bitmap](#) ↗

