

The background is a dark navy blue. In the top-left corner, there are two overlapping geometric shapes: a blue parallelogram and a light green parallelogram. In the top-right corner, there is a grey, 3D-rendered circuit board pattern. In the bottom-left corner, there is a circular inset showing a detailed, high-magnification view of a printed circuit board (PCB) with various electronic components and solder points. The text 'TERRAZA PILATES' is centered in the middle-right portion of the image.

TERRAZA PILATES

Índice

[Introducción](#)

[Objetivo](#)

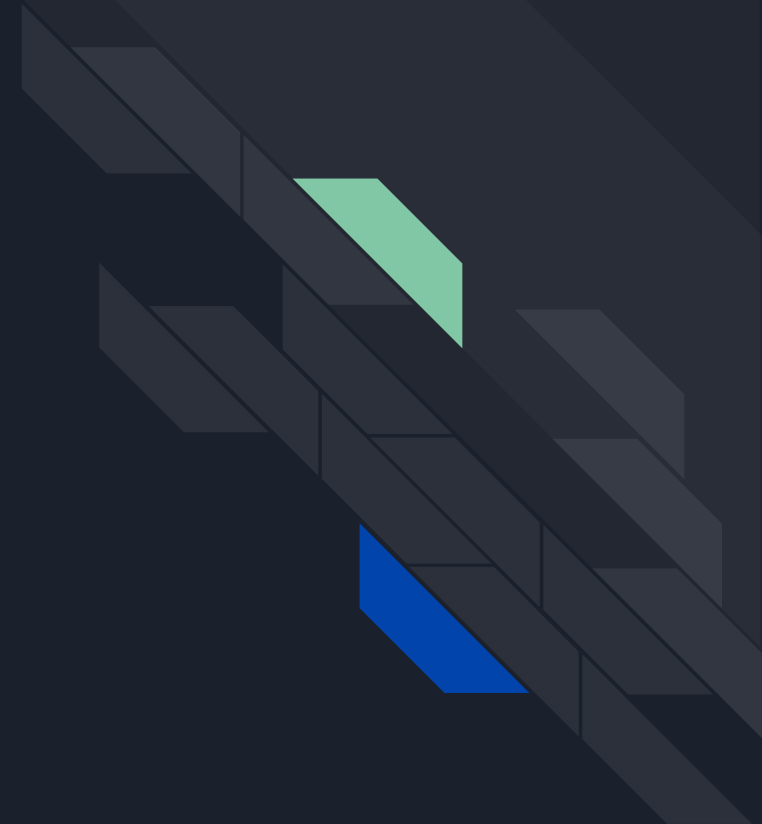
[Situación Problemática](#)

[Modelo de Negocio](#)

[Diagrama Relacional](#)

[Scripts](#)

[A Futuro](#)





Introducción


En un mundo enfocado en la salud y el bienestar, Terraza Pilates emerge como un gimnasio único que combina la serenidad al aire libre con la efectividad del método Pilates. Más que un simple gimnasio, es un destino de bienestar personalizado.

Este informe proporciona una descripción detallada del modelo de negocio relacionado con la creación de una base de datos para un gimnasio. La base de datos se ha diseñado para gestionar la información de clientes, clases, instructores, asistencia, equipamiento, evaluaciones físicas y más. El objetivo principal es optimizar la gestión y operación del gimnasio, mejorar la experiencia del cliente y facilitar el seguimiento de su progreso físico.

Esta base de datos también se convierte en una fuente de datos estratégicos para impulsar el crecimiento de Terraza Pilates. Es el corazón del negocio, conectando miembros con su camino hacia la salud y el bienestar, fortaleciendo la comunidad y consolidando su posición en la industria del fitness y el Pilates.



Objetivo del proyecto



El proyecto tiene como objetivo principal implementar y mantener una base de datos eficiente para mejorar la experiencia del cliente en Terraza Pilates, optimizar la gestión operativa, y utilizar datos estratégicos para el crecimiento del gimnasio. Esto se logrará mediante la personalización de servicios, una programación eficaz de clases, la gestión de información relevante y la toma de decisiones informadas para la expansión del negocio.



Situación Problemática

01

La base de datos de Terraza Pilates se creó debido a la necesidad de gestionar de manera eficiente la información crítica relacionada con clientes, instructores y horarios. La alta demanda y la saturación de horarios requerían un sistema que permitiera una programación optimizada de clases y reservas anticipadas.

02

La base de datos se implementó para mejorar la experiencia del cliente al personalizar planes de ejercicio y proporcionar un seguimiento más efectivo. Esto se traduce en una mayor retención de clientes, ya que se pueden abordar de manera proactiva las necesidades individuales.

03

La base de datos se convierte en una fuente valiosa de datos para tomar decisiones estratégicas que impulsen el crecimiento de Terraza Pilates. Ofrece información sobre la demanda de clases, la eficiencia del personal y el rendimiento general del gimnasio, lo que ayuda a la dirección a planificar su expansión y mejora continua.

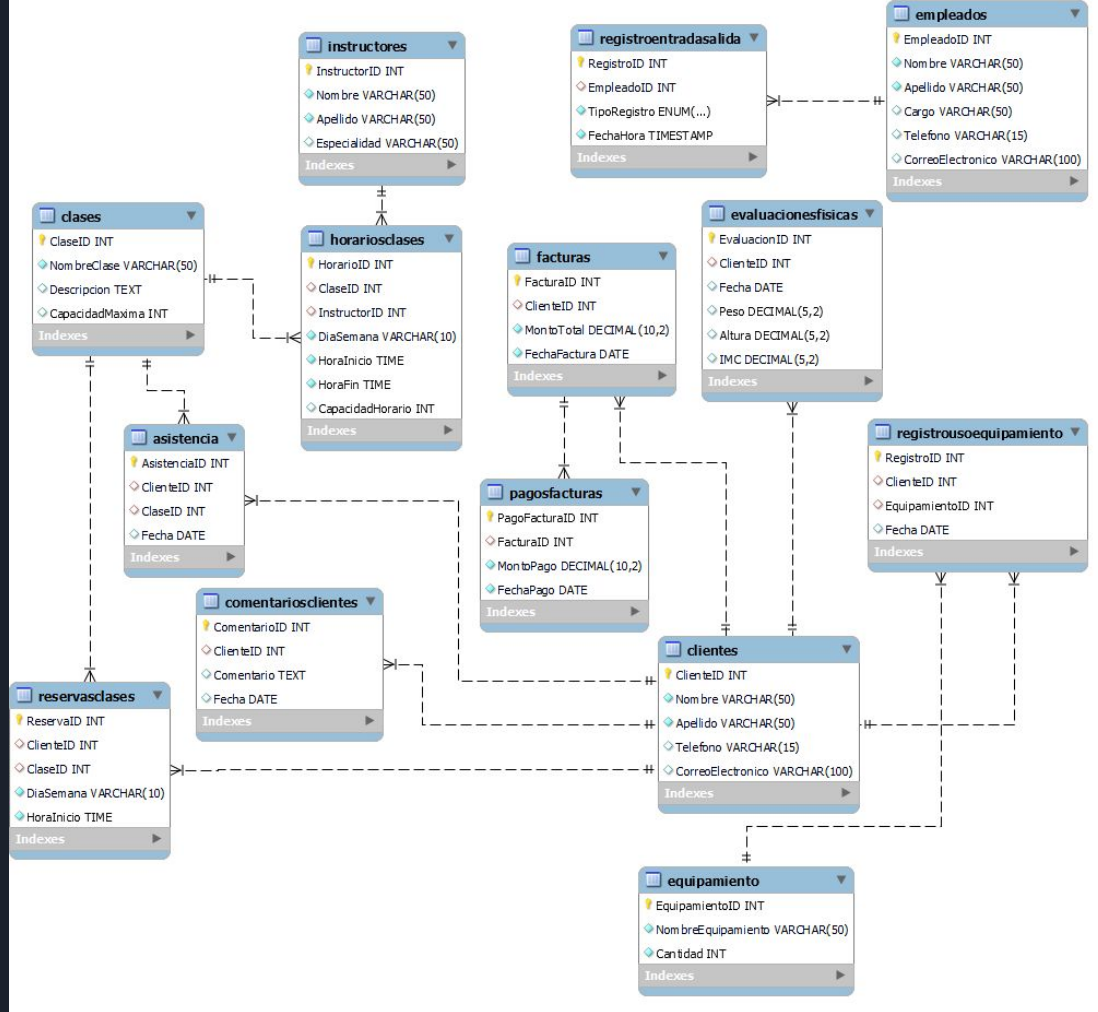


Modelo de Negocio

El modelo de negocio de Terraza Pilates se centra en ofrecer una experiencia integral de bienestar a través de clases de Pilates en un entorno al aire libre. Ofrecen suscripciones y membresías para acceder a clases personalizadas. La base de datos gestiona información de clientes y preferencias. Instructores altamente calificados brindan atención individualizada. La ubicación al aire libre añade un valor distintivo. Se enfocan en el bienestar integral y promueven la retención de clientes mediante programas de fidelización. Utilizan datos para tomar decisiones estratégicas y asegurar el crecimiento del gimnasio. En resumen, Terraza Pilates se distingue por su atención personalizada, ambiente único y compromiso con el bienestar integral de sus clientes, respaldado por una gestión eficiente y estrategias de retención.



DIAGRAMA RELACIONAL





DESCRIPCIÓN DE TABLAS

Tabla Empleados:			
Columna	Tipo	Clave Primaria	No Nulo
EmpleadoID	INT	Sí	Sí
Nombre	VARCHAR(50)	No	Sí
Apellido	VARCHAR(50)	No	Sí
Cargo	VARCHAR(50)	No	No
Telefono	VARCHAR(15)	No	No
CorreoElectronico	VARCHAR(100)	No	No
Tabla RegistroEntradaSalida:			
Columna	Tipo	Clave Primaria	No Nulo
RegistroID	INT	Sí	Sí
EmpleadoID	INT	No	No
TipoRegistro	ENUM('Entrada', 'Salida')	No	Sí
FechaHora	TIMESTAMP	No	Sí



Tabla Clientes:			
Columna	Tipo	Clave Primaria	No Nulo
CienteID	INT	Sí	Sí
Nombre	VARCHAR(50)	No	Sí
Apellido	VARCHAR(50)	No	Sí
Telefono	VARCHAR(15)	No	No
CorreoElectronico	VARCHAR(100)	No	No
Tabla Clases:			
Columna	Tipo	Clave Primaria	No Nulo
ClaseID	INT	Sí	Sí
NombreClase	VARCHAR(50)	No	Sí
Descripcion	TEXT	No	No
CapacidadMaxima	INT	No	No
Tabla Instructores:			
Columna	Tipo	Clave Primaria	No Nulo
InstructorID	INT	Sí	Sí
Nombre	VARCHAR(50)	No	Sí
Apellido	VARCHAR(50)	No	Sí
Especialidad	VARCHAR(50)	No	No




Tabla Asistencia:			
Columna	Tipo	Clave Primaria	No Nulo
AsistencialID	INT	Sí	Sí
ClienteID	INT	No	No
ClaseID	INT	No	No
Fecha	DATE	No	No
Tabla HorariosClases:			
Columna	Tipo	Clave Primaria	No Nulo
HorarioID	INT	Sí	Sí
ClaseID	INT	No	No
InstructorID	INT	No	No
DiaSemana	VARCHAR(10)	No	Sí
HoraInicio	TIME	No	Sí
HoraFin	TIME	No	Sí
CapacidadHorario	INT	No	No
Tabla ReservasClases:			
Columna	Tipo	Clave Primaria	No Nulo
ReservaID	INT	Sí	Sí
ClienteID	INT	No	No
ClaseID	INT	No	No
DiaSemana	VARCHAR(10)	No	Sí
HoraInicio	TIME	No	Sí



Tabla Equipamiento:			
Columna	Tipo	Clave Primaria	No Nulo
EquipamientoID	INT	Sí	Sí
NombreEquipamiento	VARCHAR(50)	No	Sí
Cantidad	INT	No	Sí
Tabla RegistroUsoEquipamiento:			
Columna	Tipo	Clave Primaria	No Nulo
RegistroID	INT	Sí	Sí
ClienteID	INT	No	No
EquipamientoID	INT	No	No
Fecha	DATE	No	No
Tabla EvaluacionesFisicas:			
Columna	Tipo	Clave Primaria	No Nulo
EvaluacionID	INT	Sí	Sí
ClienteID	INT	No	No
Fecha	DATE	No	No
Peso	DECIMAL(5,2)	No	No
Altura	DECIMAL(5,2)	No	No
IMC	DECIMAL(5,2)	No	No



Tabla ComentariosClientes:			
Columna	Tipo	Clave Primaria	No Nulo
ComentariolD	INT	Sí	Sí
ClienteID	INT	No	No
Comentario	TEXT	No	Sí
Fecha	DATE	No	No
Tabla Facturas:			
Columna	Tipo	Clave Primaria	No Nulo
FacturalID	INT	Sí	Sí
ClienteID	INT	No	No
MontoTotal	DECIMAL(10,2)	No	Sí
FechaFactura	DATE	No	Sí
Tabla PagosFacturas:			
Columna	Tipo	Clave Primaria	No Nulo
PagoFacturalID	INT	Sí	Sí
FacturalID	INT	No	No
MontoPago	DECIMAL(10,2)	No	Sí
FechaPago	DATE	No	Sí



En este Script, primero te
encontrarás con la creación
de las 14 Tablas

Estas tablas permiten una
gestión integral de la
membresía, clases,
equipamiento, empleados y
más en el gimnasio, lo que
facilita la administración y
mejora la experiencia tanto
de los clientes como del
personal.



Cientes:

Almacena información de los clientes, como nombre, apellido, teléfono y correo electrónico. Se utiliza para gestionar la membresía y la comunicación con los clientes.



```
1 CREATE TABLE IF NOT EXISTS Cientes (  
2     ClienteID INT AUTO_INCREMENT PRIMARY KEY,  
3     Nombre VARCHAR(50) NOT NULL,  
4     Apellido VARCHAR(50) NOT NULL,  
5     Telefono VARCHAR(15),  
6     CorreoElectronico VARCHAR(100)  
7 );
```



Tablas

Clases: Registra detalles de las clases ofrecidas, como nombre, descripción y capacidad máxima de participantes.

```
1 CREATE TABLE IF NOT EXISTS Clases (  
2     ClaseID INT AUTO_INCREMENT PRIMARY KEY,  
3     NombreClase VARCHAR(50) NOT NULL,  
4     Descripcion TEXT,  
5     CapacidadMaxima INT DEFAULT NULL  
6 );
```



Tablas

Instructores:
Guarda
información sobre
los instructores,
incluyendo
nombre, apellido y
especialidad.



```
1 CREATE TABLE IF NOT EXISTS Instructores (  
2     InstructorID INT AUTO_INCREMENT PRIMARY KEY,  
3     Nombre VARCHAR(50) NOT NULL,  
4     Apellido VARCHAR(50) NOT NULL,  
5     Especialidad VARCHAR(50)  
6 );
```




Tablas

Asistencia: Registra la asistencia de los clientes a las clases, con referencias a la tabla de Clientes y Clases para seguimiento.



```
1 CREATE TABLE IF NOT EXISTS Asistencia (  
2     AsistenciaID INT AUTO_INCREMENT PRIMARY KEY,  
3     ClienteID INT,  
4     ClaseID INT,  
5     Fecha DATE,  
6     FOREIGN KEY (ClienteID) REFERENCES Clientes(ClienteID),  
7     FOREIGN KEY (ClaseID) REFERENCES Clases(ClaseID)  
8 );
```



Tablas

Reservas de Clases: Registra las reservas de clases por parte de los clientes, con referencias a Clientes y Clases.



```
1 CREATE TABLE IF NOT EXISTS ReservasClases (  
2     ReservaID INT AUTO_INCREMENT PRIMARY KEY,  
3     ClienteID INT DEFAULT NULL,  
4     ClaseID INT DEFAULT NULL,  
5     DiaSemana VARCHAR(10) NOT NULL,  
6     HoraInicio TIME NOT NULL,  
7     FOREIGN KEY (ClienteID) REFERENCES Clientes(ClienteID),  
8     FOREIGN KEY (ClaseID) REFERENCES Clases(ClaseID)  
9 );
```



Horarios de Clases:
Almacena
información sobre los
horarios de las clases,
incluyendo el día de
la semana, hora de
inicio y finalización, y
capacidad del
horario. Tiene
referencias a Clases e
Instructores.



```
1 CREATE TABLE IF NOT EXISTS HorariosClases (  
2     HorarioID INT AUTO_INCREMENT PRIMARY KEY,  
3     ClaseID INT DEFAULT NULL,  
4     InstructorID INT DEFAULT NULL,  
5     DiaSemana VARCHAR(10) NOT NULL,  
6     HoraInicio TIME NOT NULL,  
7     HoraFin TIME NOT NULL,  
8     CapacidadHorario INT DEFAULT 0,  
9     FOREIGN KEY (ClaseID) REFERENCES Clases(ClaseID),  
10    FOREIGN KEY (InstructorID) REFERENCES Instructores(InstructorID)  
11 );
```



Tablas

Equipamiento:
Contiene
información sobre el
equipamiento
disponible en el
gimnasio, incluyendo
el nombre y la
cantidad disponible.



```
1 CREATE TABLE IF NOT EXISTS Equipamiento (  
2     EquipamientoID INT AUTO_INCREMENT PRIMARY KEY,  
3     NombreEquipamiento VARCHAR(50) NOT NULL,  
4     Cantidad INT NOT NULL  
5 );
```



Tablas

Registro de Uso de Equipamiento:
Registra cuándo y qué equipamiento utiliza cada cliente.



```
1 CREATE TABLE IF NOT EXISTS RegistroUsoEquipamiento (  
2     RegistroID INT AUTO_INCREMENT PRIMARY KEY,  
3     ClienteID INT,  
4     EquipamientoID INT,  
5     Fecha DATE,  
6     FOREIGN KEY (ClienteID) REFERENCES Clientes(ClienteID),  
7     FOREIGN KEY (EquipamientoID) REFERENCES Equipamiento(EquipamientoID)  
8 );
```



Tablas

Evaluaciones Físicas:
Registra las
evaluaciones físicas
de los clientes,
incluyendo peso,
altura e índice de
masa corporal (IMC).



```
1 CREATE TABLE IF NOT EXISTS EvaluacionesFisicas (  
2     EvaluacionID INT AUTO_INCREMENT PRIMARY KEY,  
3     ClienteID INT,  
4     Fecha DATE,  
5     Peso DECIMAL(5, 2),  
6     Altura DECIMAL(5, 2),  
7     IMC DECIMAL(5, 2),  
8     FOREIGN KEY (ClienteID) REFERENCES Clientes(ClienteID)  
9 );
```



Tablas

Comentarios de los Clientes:
Permite a los clientes dejar comentarios sobre su experiencia en el gimnasio.



```
1 CREATE TABLE IF NOT EXISTS ComentariosClientes (  
2     ComentarioID INT AUTO_INCREMENT PRIMARY KEY,  
3     ClienteID INT,  
4     Comentario TEXT,  
5     Fecha DATE,  
6     FOREIGN KEY (ClienteID) REFERENCES Clientes(ClienteID)  
7 );
```



Tablas

Facturas: Registra las facturas emitidas a los clientes, incluyendo el monto total y la fecha de emisión.



```
1 CREATE TABLE IF NOT EXISTS Facturas (  
2     FacturaID INT AUTO_INCREMENT PRIMARY KEY,  
3     ClienteID INT,  
4     MontoTotal DECIMAL(10, 2) NOT NULL,  
5     FechaFactura DATE NOT NULL,  
6     FOREIGN KEY (ClienteID) REFERENCES Clientes(ClienteID)  
7 );
```




Pagos de Facturas:
Registra los pagos
realizados por los
clientes en relación
con las facturas.



```
1 CREATE TABLE IF NOT EXISTS PagosFacturas (  
2     PagoFacturaID INT AUTO_INCREMENT PRIMARY KEY,  
3     FacturaID INT,  
4     MontoPago DECIMAL(10, 2) NOT NULL,  
5     FechaPago DATE NOT NULL,  
6     FOREIGN KEY (FacturaID) REFERENCES Facturas(FacturaID)  
7 );
```



Tablas

Empleados: Almacena información sobre los empleados del gimnasio, incluyendo nombre, apellido, cargo, teléfono y correo electrónico.



```
1 CREATE TABLE IF NOT EXISTS Empleados (  
2     EmpleadoID INT AUTO_INCREMENT PRIMARY KEY,  
3     Nombre VARCHAR(50) NOT NULL,  
4     Apellido VARCHAR(50) NOT NULL,  
5     Cargo VARCHAR(50),  
6     Telefono VARCHAR(15),  
7     CorreoElectronico VARCHAR(100)  
8 );
```



Tablas

Registro de Entradas/Salidas de Empleados: Registra las entradas y salidas de los empleados, proporcionando un registro de tiempo y asistencia.



```
1 CREATE TABLE IF NOT EXISTS RegistroEntradaSalida (  
2     RegistroID INT AUTO_INCREMENT PRIMARY KEY,  
3     EmpleadoID INT,  
4     TipoRegistro ENUM('Entrada', 'Salida') NOT NULL,  
5     FechaHora TIMESTAMP NOT NULL,  
6     FOREIGN KEY (EmpleadoID) REFERENCES Empleados(EmpleadoID)  
7 );
```



En este script se encuentran todas las VISTAS


En resumen, estas vistas simplifican y agregan datos clave de varias tablas para brindar una visión más completa y accesible de la información relevante en el contexto de los clientes, instructores, finanzas, empleados, evaluaciones físicas y horarios de clases en el gimnasio. Esto facilita consultas y análisis específicos sin tener que acceder directamente a múltiples tablas.

VistaClientesReservas: Esta vista combina datos de las tablas "Clientes" y "ReservasClases". Proporciona información sobre los clientes y sus reservas de clases, incluyendo el día y la hora de la reserva. Facilita el seguimiento de las reservas de clases por parte de los clientes.

```
1 CREATE VIEW VistaClientesReservas AS
2 SELECT
3     c.ClienteID,
4     c.Nombre AS NombreCliente,
5     c.Apellido AS ApellidoCliente,
6     c.Telefono,
7     c.CorreoElectronico,
8     r.ReservaID,
9     r.ClaseID,
10    r.DiaSemana AS DiaReserva,
11    r.HoraInicio AS HoraReserva
12 FROM
13     Clientes c
14 LEFT JOIN
15     ReservasClases r ON c.ClienteID = r.ClienteID;
```

VistaInstructoresClases:
Combina datos de las tablas "Instructores" y "HorariosClases". Ofrece una visión de los instructores y las clases que imparten, incluyendo el día, la hora de inicio y finalización. Ayuda a gestionar la asignación de instructores a clases.

```
1 CREATE VIEW VistaInstructoresClases AS
2 SELECT
3     i.InstructorID,
4     i.Nombre AS NombreInstructor,
5     i.Apellido AS ApellidoInstructor,
6     i.Especialidad,
7     h.ClaseID,
8     h.DiaSemana,
9     h.HoraInicio,
10    h.HoraFin
11 FROM
12     Instructores i
13 LEFT JOIN
14     HorariosClases h ON i.InstructorID = h.InstructorID;
```




```
1 CREATE VIEW VistaFacturas AS
2 SELECT
3     f.FacturaID,
4     f.ClienteID,
5     c.Nombre AS NombreCliente,
6     c.Apellido AS ApellidoCliente,
7     f.MontoTotal,
8     f.FechaFactura,
9     p.PagoFacturaID,
10    p.MontoPago,
11    p.FechaPago
12 FROM
13     Facturas f
14 LEFT JOIN
15     Clientes c ON f.ClienteID = c.ClienteID
16 LEFT JOIN
17     PagosFacturas p ON f.FacturaID = p.FacturaID;
```

VistaFacturas: Esta vista fusiona datos de las tablas "Facturas", "Clientes" y "PagosFacturas". Muestra información sobre las facturas emitidas a los clientes, los pagos realizados y el estado de las cuentas. Simplifica el seguimiento financiero.

VistaEmpleadosEntradaSalida:
Combina datos de las tablas
"Empleados" y
"RegistroEntradaSalida".
Proporciona un registro de las
entradas y salidas de los
empleados, lo que facilita el
seguimiento de la asistencia y
las horas trabajadas.

```
1 CREATE VIEW VistaEmpleadosEntradaSalida AS
2 SELECT
3     e.EmpleadoID,
4     e.Nombre AS NombreEmpleado,
5     e.Apellido AS ApellidoEmpleado,
6     e.Cargo,
7     r.TipoRegistro,
8     r.FechaHora
9 FROM
10     Empleados e
11 LEFT JOIN
12     RegistroEntradaSalida r ON e.EmpleadoID = r.EmpleadoID;
```

```
1 CREATE VIEW VistaEvaluacionesFisicas AS
2 SELECT
3     ef.EvaluacionID,
4     ef.ClienteID,
5     c.Nombre AS NombreCliente,
6     c.Apellido AS ApellidoCliente,
7     ef.Fecha,
8     ef.Peso,
9     ef.Altura,
10    ef.IMC
11 FROM
12     EvaluacionesFisicas ef
13 LEFT JOIN
14     Clientes c ON ef.ClienteID = c.ClienteID;
```

VistaEvaluacionesFisicas: Fusiona datos de las tablas "EvaluacionesFisicas" y "Clientes". Ofrece una visión de las evaluaciones físicas de los clientes, incluyendo peso, altura e índice de masa corporal (IMC). Facilita el seguimiento de la evolución física de los clientes.

VistaHorariosClases: Combina datos de las tablas "HorariosClases", "Clases" e "Instructores". Muestra información sobre los horarios de las clases, incluyendo el nombre de la clase, el instructor asignado, el día y la hora. Ayuda en la programación y gestión de clases.


```
1 CREATE VIEW VistaHorariosClases AS
2 SELECT
3     h.HorarioID,
4     c.NombreClase,
5     i.Nombre AS NombreInstructor,
6     h.DiaSemana,
7     h.HoraInicio,
8     h.HoraFin,
9     h.CapacidadHorario
10 FROM
11     HorariosClases h
12 LEFT JOIN
13     Clases c ON h.ClaseID = c.ClaseID
14 LEFT JOIN
15     Instructores i ON h.InstructorID = i.InstructorID;
```



Function


```
1 CREATE FUNCTION AgregarEmpleado(  
2     p_Nombre VARCHAR(50),  
3     p_Apellido VARCHAR(50),  
4     p_Cargo VARCHAR(50),  
5     p_Telefono VARCHAR(15),  
6     p_CorreoElectronico VARCHAR(100)  
7 )  
8 RETURNS INT  
9 NO SQL  
10 BEGIN  
11     INSERT INTO Empleados (Nombre, Apellido, Cargo, Telefono, CorreoElectronico)  
12     VALUES (p_Nombre, p_Apellido, p_Cargo, p_Telefono, p_CorreoElectronico);  
13     RETURN LAST_INSERT_ID();  
14 END;
```

La función "AgregarEmpleado" permite agregar un nuevo empleado a la base de datos del gimnasio. Toma información como nombre, apellido, cargo, teléfono y correo electrónico como entrada, la inserta en la tabla "Empleados" y devuelve el ID único del nuevo empleado. Simplifica la tarea de registro de empleados de manera eficiente.



En este script se encuentran todos los Stored Procedures

Estos stored procedures son procedimientos almacenados que realizan operaciones específicas en la base de datos del gimnasio. Aquí tienes una breve explicación de lo que hace cada uno de ellos:





Stored Procedures

InsertarNuevoCliente:

Este procedimiento permite ingresar un nuevo cliente en la base de datos, tomando como entrada el nombre, apellido, teléfono y correo electrónico del cliente. Realiza una inserción en la tabla "Clientes" con esta información.

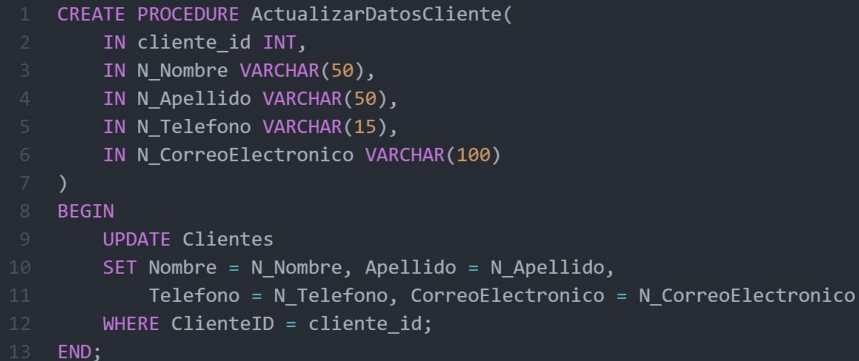
```
1 CREATE PROCEDURE InsertarNuevoCliente(  
2     IN Nombre VARCHAR(50),  
3     IN Apellido VARCHAR(50),  
4     IN Telefono VARCHAR(15),  
5     IN CorreoElectronico VARCHAR(100)  
6 )  
7 BEGIN  
8     INSERT INTO Clientes (Nombre, Apellido, Telefono, CorreoElectronico)  
9     VALUES (Nombre, Apellido, Telefono, CorreoElectronico);  
10 END;
```



Stored Procedures

ActualizarDatosCliente:

Permite actualizar la información de un cliente existente. Toma como entrada el ID del cliente y los nuevos datos de nombre, apellido, teléfono y correo electrónico. Actualiza los registros correspondientes en la tabla "Clientes".



```
1 CREATE PROCEDURE ActualizarDatosCliente(  
2     IN cliente_id INT,  
3     IN N_Nombre VARCHAR(50),  
4     IN N_Apellido VARCHAR(50),  
5     IN N_Telefono VARCHAR(15),  
6     IN N_CorreoElectronico VARCHAR(100)  
7 )  
8 BEGIN  
9     UPDATE Clientes  
10    SET Nombre = N_Nombre, Apellido = N_Apellido,  
11        Telefono = N_Telefono, CorreoElectronico = N_CorreoElectronico  
12    WHERE ClienteID = cliente_id;  
13 END;
```



Stored Procedures

EliminarCliente:

Este procedimiento permite eliminar un cliente de la base de datos. Toma como entrada el ID del cliente y elimina el registro correspondiente de la tabla "Clientes".

```
1 CREATE PROCEDURE EliminarCliente(  
2     IN cliente_id INT  
3 )  
4 BEGIN  
5     DELETE FROM Clientes WHERE ClienteID = cliente_id;  
6 END;
```

Stored

```
1 CREATE PROCEDURE RegistrarReservaClase(
2     IN cliente_id INT,
3     IN clase_id INT,
4     IN dia_semana VARCHAR(10),
5     IN hora_inicio TIME
6 )
7 BEGIN
8     DECLARE capacidad_actual INT;
9     DECLARE capacidad_maxima INT;
10
11     -- Obtener la capacidad máxima de la clase
12     SELECT CapacidadMaxima INTO capacidad_maxima
13     FROM Clases
14     WHERE ClaseID = clase_id;
15
16     -- Obtener la capacidad actual de la clase en el horario específico
17     SELECT COUNT(*) INTO capacidad_actual
18     FROM ReservasClases rc
19     JOIN HorariosClases hc ON rc.ClaseID = hc.ClaseID
20     WHERE rc.DiaSemana = dia_semana
21     AND rc.HoraInicio = hora_inicio;
22
23     -- Verificar si hay espacio disponible
24     IF capacidad_actual < capacidad_maxima THEN
25         -- Realizar la reserva
26         INSERT INTO ReservasClases (ClienteID, ClaseID, DiaSemana, HoraInicio)
27         VALUES (cliente_id, clase_id, dia_semana, hora_inicio);
28
29         SELECT 'Reserva exitosa.' AS Resultado;
30     ELSE
31         SELECT 'No hay espacio disponible en esta clase.' AS Resultado;
32     END IF;
33 END;
```

RegistrarReservaClase:

Registra la reserva de una clase por parte de un cliente. Toma como entrada el ID del cliente, el ID de la clase, el día de la semana y la hora de inicio. Verifica la capacidad de la clase en ese horario y realiza la reserva si hay espacio disponible.



Stored Procedures

EliminarReservaClase:

Permite a un cliente cancelar una reserva de clase. Toma como entrada el ID del cliente, el día de la semana y la hora de inicio. Verifica si el cliente tiene una reserva en ese horario y, si es así, elimina la reserva.

```
1 CREATE PROCEDURE EliminarReservaClase(
2     IN cliente_id INT,
3     IN dia_semana VARCHAR(10),
4     IN hora_inicio TIME
5 )
6 BEGIN
7     DECLARE reserva_id INT;
8     DECLARE clase_id INT;
9
10    -- Verificar si el cliente tiene una reserva en el horario especificado
11    SELECT rc.ReservaID, rc.ClaseID
12    INTO reserva_id, clase_id
13    FROM ReservasClases rc
14    JOIN HorariosClases hc ON rc.ClaseID = hc.ClaseID
15    WHERE rc.ClienteID = cliente_id AND rc.DiaSemana = dia_semana AND rc.HoraInicio = hora_inicio;
16
17    IF reserva_id IS NOT NULL THEN
18        -- Eliminar la reserva
19        DELETE FROM ReservasClases WHERE ReservaID = reserva_id;
20
21        SELECT 'Reserva eliminada exitosamente.' AS Resultado;
22    ELSE
23        SELECT 'No se encontró una reserva para el cliente en el horario especificado.' AS Resultado;
24    END IF;
25 END;
```



En este script se encuentran los Triggers

Estos triggers son fragmentos de código que se ejecutan automáticamente en respuesta a ciertos eventos en la base de datos. Aquí está una breve explicación de lo que hacen cada uno de estos triggers:



Triggers

ActualizarCapacidadHorarioDespués
DeReserva:

Desencadenado después de la
inserción de una nueva reserva en la
tabla "ReservasClases".

Incrementa la capacidad disponible
en la tabla "HorariosClases" para una
clase específica, día y hora después de
que un cliente realice una reserva.



```
1 CREATE TRIGGER ActualizarCapacidadHorarioDespuesDeReserva
2 AFTER INSERT ON ReservasClases
3 FOR EACH ROW
4 BEGIN
5     UPDATE HorariosClases
6     SET CapacidadHorario = CapacidadHorario + 1
7     WHERE ClaseID = NEW.ClaseID AND DiaSemana = NEW.DiaSemana AND HoraInicio = NEW.HoraInicio;
8 END;
```



Triggers

ActualizarCapacidadHorarioDespuésDeEliminarReserva:
Desencadenado después de eliminar una reserva de la tabla "ReservasClases".
Decrementa la capacidad disponible en la tabla "HorariosClases" para una clase específica, día y hora después de que un cliente elimine una reserva.



```
1 CREATE TRIGGER ActualizarCapacidadHorarioDespuesDeEliminarReserva
2 AFTER DELETE ON ReservasClases
3 FOR EACH ROW
4 BEGIN
5     UPDATE HorariosClases
6     SET CapacidadHorario = CapacidadHorario - 1
7     WHERE ClaseID = OLD.ClaseID AND DiaSemana = OLD.DiaSemana AND HoraInicio = OLD.HoraInicio;
8 END;
```



Triggers

ActualizarFechaMontoFacturaDespuésDePago:
Desencadenado después de insertar un nuevo registro de pago en la tabla "PagosFacturas".
Actualiza automáticamente la fecha de factura a la fecha actual y recalcula el monto total en la tabla "Facturas" para una factura específica, asegurando que refleje los pagos realizados por el cliente.



```
1 CREATE TRIGGER ActualizarFechaMontoFacturaDespuesDePago
2 AFTER INSERT ON PagosFacturas
3 FOR EACH ROW
4 BEGIN
5     DECLARE factura_id INT;
6     SELECT FacturaID INTO factura_id FROM Facturas WHERE FacturaID = NEW.FacturaID;
7     UPDATE Facturas
8     SET FechaFactura = NOW(),
9         MontoTotal = (SELECT SUM(MontoPago) FROM PagosFacturas WHERE FacturaID = factura_id)
10    WHERE FacturaID = factura_id;
11 END;
```



Triggers

ActualizarIMCDespuésDeActualizarEvaluaciónFísica:
Desencadenado después de actualizar una evaluación física en la tabla "EvaluacionesFisicas".
Calcula automáticamente el índice de masa corporal (IMC) basado en las nuevas mediciones de peso y altura del cliente y actualiza el valor en la tabla "EvaluacionesFisicas".



```
1 CREATE TRIGGER ActualizarIMCDespuesDeActualizarEvaluacionFisica
2 AFTER UPDATE ON EvaluacionesFisicas
3 FOR EACH ROW
4 BEGIN
5     UPDATE EvaluacionesFisicas
6     SET IMC = ROUND(NEW.Peso / ((NEW.Altura / 100) * (NEW.Altura / 100)), 2)
7     WHERE EvaluacionID = NEW.EvaluacionID;
8 END;
```



Triggers

RegistrarEntradaSalidaEmpleado:
Desencadenado antes de insertar
un registro de entrada o salida en
la tabla "RegistroEntradaSalida".
Registra automáticamente la
fecha y hora de entrada o salida
de un empleado en la tabla
"RegistroEntradaSalida" cuando
se inserta un nuevo registro.

```
1 CREATE TRIGGER RegistrarEntradaSalidaEmpleado
2 BEFORE INSERT ON RegistroEntradaSalida
3 FOR EACH ROW
4 BEGIN
5     INSERT INTO RegistroEntradaSalida (EmpleadoID, TipoRegistro, FechaHora)
6     VALUES (NEW.EmpleadoID, NEW.TipoRegistro, NOW());
7 END;
```



Inserts

```
('Luis', 'Ramirez', 'Gerente', '555-1111', 'luis@example.com'),
('Sofia', 'Gonzalez', 'Recepcionista', '555-2222', 'sofia@example.com');

-- Insertar datos en la tabla de RegistroEntradaSalida
INSERT INTO RegistroEntradaSalida (EmpleadoID, TipoRegistro, FechaHora)
VALUES
    (1, 'Entrada', '2023-10-05 08:00:00'),
    (2, 'Entrada', '2023-10-06 09:00:00'),
    (1, 'Salida', '2023-10-05 17:00:00');

-- Insertar datos en la tabla ReservasClases
INSERT INTO reservasclases (ClienteID, ClaseID, DiaSemana, HoraInicio)
VALUES
    (1, 1, 'Lunes', '09:00:00'),
    (2, 3, 'Martes', '17:30:00'),
    (3, 2, 'Miércoles', '19:00:00');
```



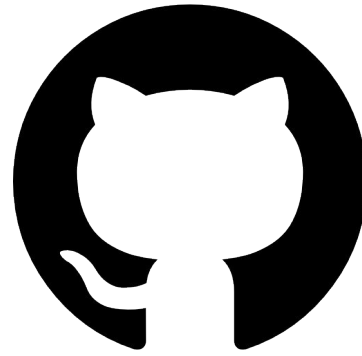

A Futuro

El futuro de la base de datos de Terraza Pilates se orienta hacia una mejora significativa de la experiencia del cliente, mediante la personalización de recomendaciones basadas en su historial y objetivos. También se enfoca en la integración con aplicaciones móviles y plataformas en línea para facilitar reservas y seguimiento de progresos. La automatización de procesos administrativos y la actualización en tiempo real de horarios se volverán fundamentales. Además, se busca implementar análisis predictivos para anticipar las necesidades de los clientes y ajustar la programación en consecuencia. La adaptación de la base de datos para incorporar nuevos servicios y gestionar recursos es otro aspecto clave. La seguridad de los datos se reforzará para cumplir con regulaciones de privacidad. Se planea realizar análisis más profundos sobre tendencias y rendimiento mediante informes y analíticas avanzadas. Por último, se diseñará la base de datos con escalabilidad en mente, permitiendo que crezca con el negocio. En resumen, el futuro de la base de datos de Terraza Pilates se enfoca en mejorar la experiencia del cliente, la eficiencia operativa y la toma de decisiones basada en datos, con un enfoque en la adaptabilidad para el éxito continuo.



¡Gracias!

La base de datos de Terraza Pilates es esencial para el éxito del negocio, ya que garantiza la gestión eficiente de todas las operaciones relacionadas con el estudio de Pilates. Proporciona a los clientes un servicio personalizado y seguro, al tiempo que facilita la administración interna y el análisis de datos



GitHub



Mockaroo



MySQL