Instituto Tecnológico de Costa Rica

Unidad desconcentrada de Computación

Sistema de gestión de gimnasio fastfitness

Bases de Datos I

Profesor:

Leonardo Víquez

Integrantes:

Jonathan Sancho Loaiza

José Andrés Arrieta Alfaro

Angie Melisa Aguilar Alemán

I Semestre del 2025

Introducción

En muchos gimnasios, la gestión del día a día suele hacerse de forma manual, lo que puede generar confusión, errores y pérdida de tiempo. Llevar el control de clientes, clases, pagos y reportes usando hojas de cálculo, agendas físicas o apuntes desorganizados termina siendo poco práctico, especialmente cuando el gimnasio empieza a crecer. Esto afecta directamente la experiencia del cliente y complica el trabajo del personal administrativo.

Por eso, en este proyecto se desarrolló una aplicación web pensada específicamente para resolver ese tipo de problemas. Utilizando tecnologías modernas como Next.js y React.js para el frontend, junto con Express.js y SQL Server para el backend y la base de datos, se construyó una solución que permite automatizar y simplificar los procesos clave del gimnasio.

La plataforma permite registrar y administrar fácilmente a los clientes, asignarles clases o entrenadores, llevar un control detallado de las sesiones a las que asisten y registrar los pagos de forma clara y ordenada. Además, incluye un sistema de reportes que le da a los administradores una vista completa de todo lo que ocurre en el gimnasio: cuántos clientes hay activos, cuántos pagos se han recibido, qué clases tienen mayor participación, entre otros datos útiles para tomar decisiones.

Con esta herramienta, el gimnasio puede olvidarse del caos de los registros manuales y dar un paso hacia una gestión más eficiente, moderna y profesional.

Componentes

Modelo relacional

Relaciones de la tabla persona:

La tabla padre persona se encarga de heredar sus atributos a las tablas hijo administrador, cliente y entrenador, de manera que se modularice el uso de sus atributos y de esta manera evitar redundancia y repetición de datos, además se le aplica su debida normalización a la tabla para tener atributos como género, teléfonos y distritos de manera separada.

Relaciones de la tabla cliente:

Cada cliente tiene asociado 1 membresía (que se podía actualizar o renovar) y que cada una de sus membresías tiene un historial de pago donde se incluyera la forma de pago y el tipo de membresías, adema de esto, cada cliente puede inscribirse a las distintas sesiones programadas que se den en el gimnasio, además de poder poner a los clientes presentes o ausentes mediante la tabla asistencia de clientes.

Relaciones de la tabla *sesion_programada* y sesión:

En el sistema se pueden crear distintas clases y a partir de ellas se pueden crear varias instancias o sesiones de estas, en dichas sesiones se incluyen su clase, su grupo y el horario que se imparte, para que de esta manera se puedan inscribir clientes que cuenten con una membresía no vencida y con un número limitado de cupo, además de asignar el respectivo entrenador a la sesión.

Relaciones de la tabla entrenador

Cada entrenador puede dar n sesiones, por medio de la tabla *entrenador_sesion*, se pueden registrar cual entrenador va a dar que sesión, evitando errores de que a 1 entrenador se le asigne una clase que ya está asignada por otro.

Relaciones de la tabla administradores y maquina

Cada uno de los administradores pueden generar distintas sesiones de revisión a las maquinas para hacerles observaciones y cambiar su estado, además de ver el historial de revisiones de las maquinas mediante la tabla *revisión_maquina.*

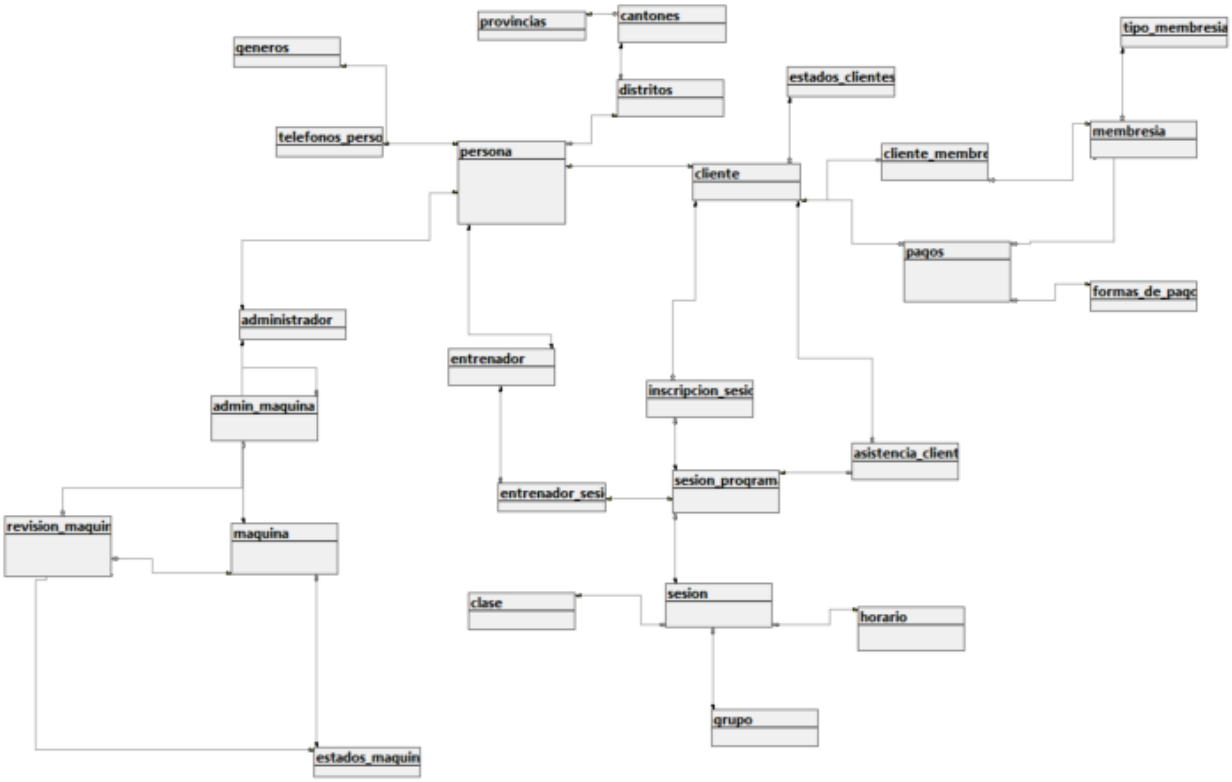

Figura [1]: Modelo relacional de la base de datos

Creación propia

Reglas

Se crean 5 reglas asociados a 5 diferentes tipos de datos

1. **ReglaCedulaNumerica:** esta regla aplica una restricción aplicando una expresión regular, de manera que se se valide un patron para la cedula de la persona.

```
CREATE RULE ReglaCedulaNumerica AS
    @cedula LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]';
GO
```

2. **ReglaCorreo:** esta regla aplica una validación de correo del formato algo@algo.algo, para evitar el uso de correos inválidos

```
CREATE RULE ReglaCorreo AS
    @correo LIKE '_%@__%.__%';
GO
```

3. **ReglaTelefono:** esta regla valida el uso de teléfonos validos para costa rica, empezando con 2 y continuando hasta obtener 8 digitos del 0 al 9

```
CREATE RULE ReglaTelefono AS
    @telefono LIKE '[2-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]';
GO
```

4. **ReglaNombreYApellidos:** esta regla valida el solo uso de letras mayúsculas y minúsculas, evitando el uso de números y caracteres especiales en el nombre.

```
CREATE RULE ReglaNombreYApellidos AS
    @nombre NOT LIKE '%[^a-zA-ZáéíóúÁÉÍÓÚñÑ ]%';
GO
```

5. **ReglaEdad:** validación simple de edades entre 5 y 120.

```
CREATE RULE ReglaEdad AS
    @edad BETWEEN 5 AND 120;
GO
```

Vistas

Se crean un total de 9 vistas, todas y cada una con un propósito distinto

1. **Vista_clientes:** esta vista muestra información detallada de todos los clientes, incluyendo nombre, apellidos, cedula, correo, teléfono, fecha de registro, fecha de expiración de membresia, tipo de membresia y su estado
2. **Vista_clientes_sesion:** esta vista muestra información de las sesiones programadas en las que están inscritos los clientes, incluyendo su nombre, cedula, nombre de clase, numero de grupo, día, fecha de sesión, hora de inicio, hora de fin, entrenador y su respectiva asistencia
3. **Vista_detalles_sesion_programadas:** muestra información de sesiones disponibles, incluyendo su clase, descripción, numero de grupo, día, hora de inicio y hora de fin, esta vista fue sumamente útil para poder realizar la inscripción de los clientes.
4. **Vista_sesiones_sin_entrenador:** muestra información como nombre, grupo, hora de inicio y fin además de la fecha de las sesiones que aun no se les ha asignado un entrenador
5. **Vista_entrenador_sesiones_totales:** muestra información de la cantidad de sesiones asignada a cada uno de los entrenadores

6. **Vista_admin_maquina:** muestra información como nombre, id_maquina, modelo, marca, estado, última revisión y cantidad de maquinas por cada uno de los administradores, permitiendo ver la información relevante sobre la administración del inventario de las maquinas
7. **Vista_revision_maquina:** muestra información de la revisión de las maquinas, como el estado más reciente, observaciones y cedula del administrador
8. **Vista_historial_pagos_clientes:** muestra un historial detallado de todos los pagos relacionados a los tipos de membresia de cada uno del cliente incluyendo la forma de pago, monto y fecha
9. **Vista_total_clases_por_sesion:** muestra el total de sesiones programadas por cada una de las clases disponibles, incluyendo el nombre y su descripción

Índices

Se crearon 3 indices en las las columnas de la tabla persona, ya que esta contiene los atributos mas frecuentemente accedidos, fueron creados en la columna **nombre**, **apellido1 y apellido 2**, los 3 de tipo NONCLUSTERED, ya que el motor de base de datos crea automáticamente un índice en las llaves primarias de las tablas.

Triggers

Se crearon 3 indices para mantener un control de las operaciones crud realizadas en la base de datos.

1. **trigger_set_fecha_expiracion_membresia:** este trigger se encarga de determinar automáticamente la fecha de expiración de una membresia nueva, renovada o actualizada
2. **trigger_actualizar_estado_cliente:** se encarga de actualizar el estado de cliente a inactivos a los clientes cuya membresia esta vencida o no tengan una aun
3. **trigger_recalcular_cant_maquinas_admin:** se encarga de mantener actualizado la cantidad de maquinas que lleva a cargo cada uno de los administradores.

Procedimientos almacenados

Se crean un total de 16 procedimientos almacenados, todos transaccionales, esto para poder ejecutar todo tipo de operación sobre todas y cada una de las relaciones de la base de datos.

1. **insertar_cliente:** Registra un nuevo cliente en la base de datos. Inserta su información personal y lo enlaza con las entidades correspondientes (persona, membresía u otros datos asociados).
2. **actualizar_persona:** Actualiza los datos de una persona seleccionada (nombre, contacto, etc.). Gracias al ON UPDATE CASCADE en la tabla

persona, los cambios se propagan automáticamente a las tablas relacionadas.

3. **eliminar_persona:** Elimina completamente del sistema a una persona, eliminando todas sus relaciones
4. **actualizar_membresia_cliente:** Actualiza la información de la membresía del cliente (fecha de vencimiento, estado, tipo de membresia), respetando el historial y las reglas de negocio
5. **renovar_membresia:** Renueva una membresía existente sin crear una nueva. Solo actualiza fechas de vencimiento y vigencia, manteniendo el ID de la membresía y dejando registro de pago
6. **registrar_pago_membresia:** Registra un nuevo pago asociado a la membresía del cliente. El pago puede actualizar automáticamente el estado de la membresía si se encontraba inactiva.
7. **crear_clase:** Crea una nueva clase en el sistema (ej: Zumba, Spinning, Pesas Funcionales), registrando su nombre, duración, cupo estimado y otros datos necesarios.
8. **crear_sesion_programada:** Genera una sesión específica a partir de una clase, asignándole fecha, hora, y eventualmente capacidad máxima o entrenador.
9. **asignar_entrenador_a_sesion_programada:** Asocia un entrenador a una sesión previamente creada. Asegura que la sesión cuente con personal a cargo y puede ser utilizado desde una interfaz administrativa o mediante verificación automatizada.

10. **inscribir_cliente_a_sesion_programada:** Inscribe a un cliente en una sesión. Antes de registrar la inscripción, se valida que la sesión tenga cupo, que el cliente esté activo, y que no esté inscrito ya.
11. **registrar_asistencia_cliente:** Registra la asistencia de un cliente a una sesión en la que está inscrito. Esta operación también puede incluir validaciones de hora, fecha y estado de la sesión.
12. **obtener_estadisticas_por_fecha:** Devuelve estadísticas del gimnasio en una fecha específica. Incluye métricas como número de clientes activos, sesiones realizadas, pagos, etc.
13. **agregar_maquina:** Registra una nueva máquina (bicicleta, caminadora, banco de pesas, etc.), incluyendo su tipo, marca, estado inicial y fecha de ingreso al inventario.
14. **revisar_maquina:** Actualiza la información de mantenimiento o revisión de una máquina. Permite registrar observaciones, técnicos responsables y cambiar el estado de la máquina (ej. funcional, en reparación, fuera de servicio).
15. **eliminar_clase:** Elimina una clase y toda su información ligada en sus tablas relacionadas, tales como sesión y inscripciones
16. **eliminar_sesion_programada:** Elimina una sesión programada, incluyendo las personas inscritas, tipo de clase, horarios, esto es útil en clase de que se quiera cancelar una clase

Cursores

Se crearon 2 cursores enfocados a las tablas donde puede haber campos nulos debido a su falta de auditoría.

1. **cursor_maquinas_vencidas:** Recorre todas las máquinas cuya fecha de revisión ha expirado. Se usa para detectar las que requieren mantenimiento preventivo o revisión urgente.
2. **cursor_sesiones_sin_entrenador:** Recorre las sesiones programadas que aún no tienen un entrenador asignado. Permite mantener el control de la planificación semanal y evitar sesiones sin responsable.

Consultas de grado avanzado de procesamiento

Consulta avanzada 1: Ranking de clientes por número de clases inscritas de manera descendente, utilizando funciones de agregación como COUNT, RANK, aplicando ORDER BY Y GROUP BY, se logra una consulta que muestra un ranking de los clientes con más clases inscritas históricamente.

Consulta avanzada 2: Clientes que tienen membresia proxima a vencer, utilizando funciones de fechas como DATEDIFF y GETDATE, se logra obtener una consulta que nos dice las membresías que están próximas a vencer.

Consulta avanzada 3: Promedio de matriculados por grupo y detección de sobrecupo, mediante el uso de CASE se logra una consulta bastante elegante donde se logra determinar las sesiones con sobrecupo, llenas o disponibles.

Consulta avanzada 4: distribución de genero por estado de clientes, utilizando funciones de agregación como SUM combinado con los condicionales CASE y la función COUNT, se logra determinar la distribución por genero de los clientes.

Consulta avanzada 5: Cuenta las cantidades de sesiones por mes: Mediante JOINS y funciones de agregación se logra determinar la cantidad de sesiones por mes.