INSTITUTO TECNOLÓGICO DE LAS AMÉRICAS

Las Americas Institute of Technology

Participantes:

Keury Ramirez **2023-1101**

Victor Sánchez **2023-1146**

Erick de la rosa **2023-1172**

Carrera: Desarrollo de software.

Materia: BD Avanzada.

Docente: Daniel Arturo Sánchez De Oleo.

Tema: Proyecto Final.

Fecha:

24/7/2024

Tabla de contenido.

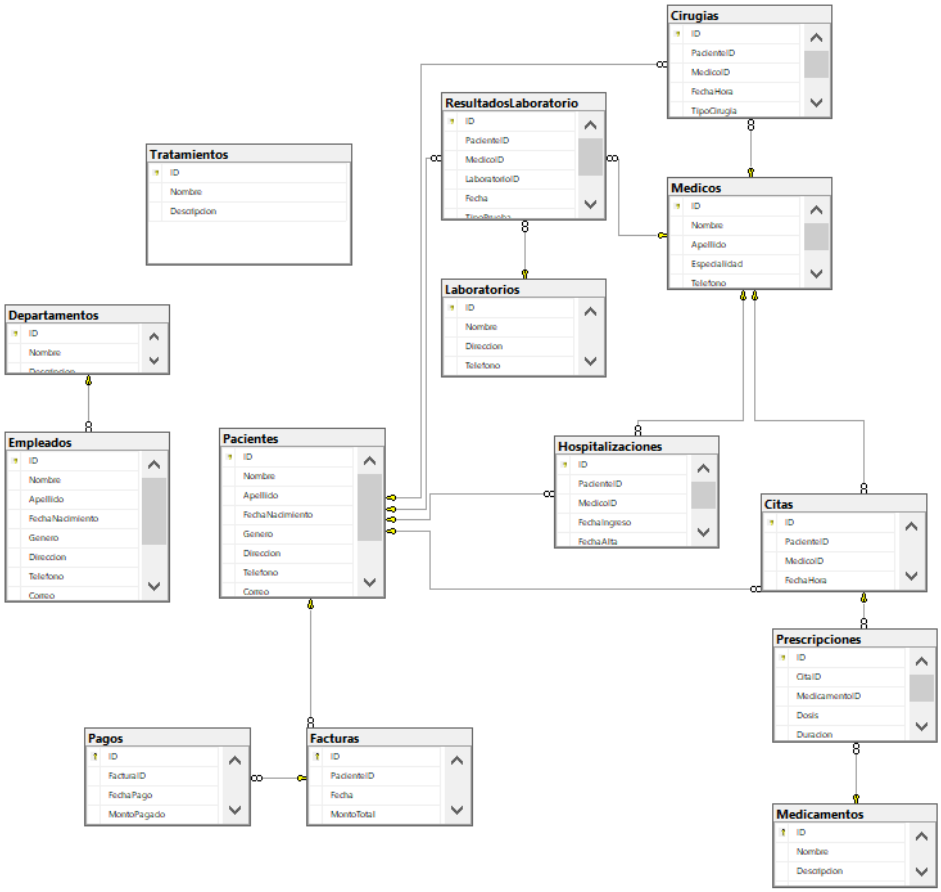
MEDICORE

Medicore es un sistema de gestión integral para hospitales, diseñado para manejar eficientemente la información relacionada con pacientes, médicos, citas, tratamientos, y administración hospitalaria. Este sistema incluye diversas tablas que permiten almacenar y gestionar datos críticos, como información personal de pacientes, detalles de médicos y sus especialidades, programación de citas, tratamientos y medicamentos, departamentos del hospital, y registros de empleados. Además, **Medicore** abarca funcionalidades para hospitalizaciones, cirugías, resultados de laboratorio, facturación y pagos, proporcionando una solución **completa** y **robusta** para la gestión hospitalaria.

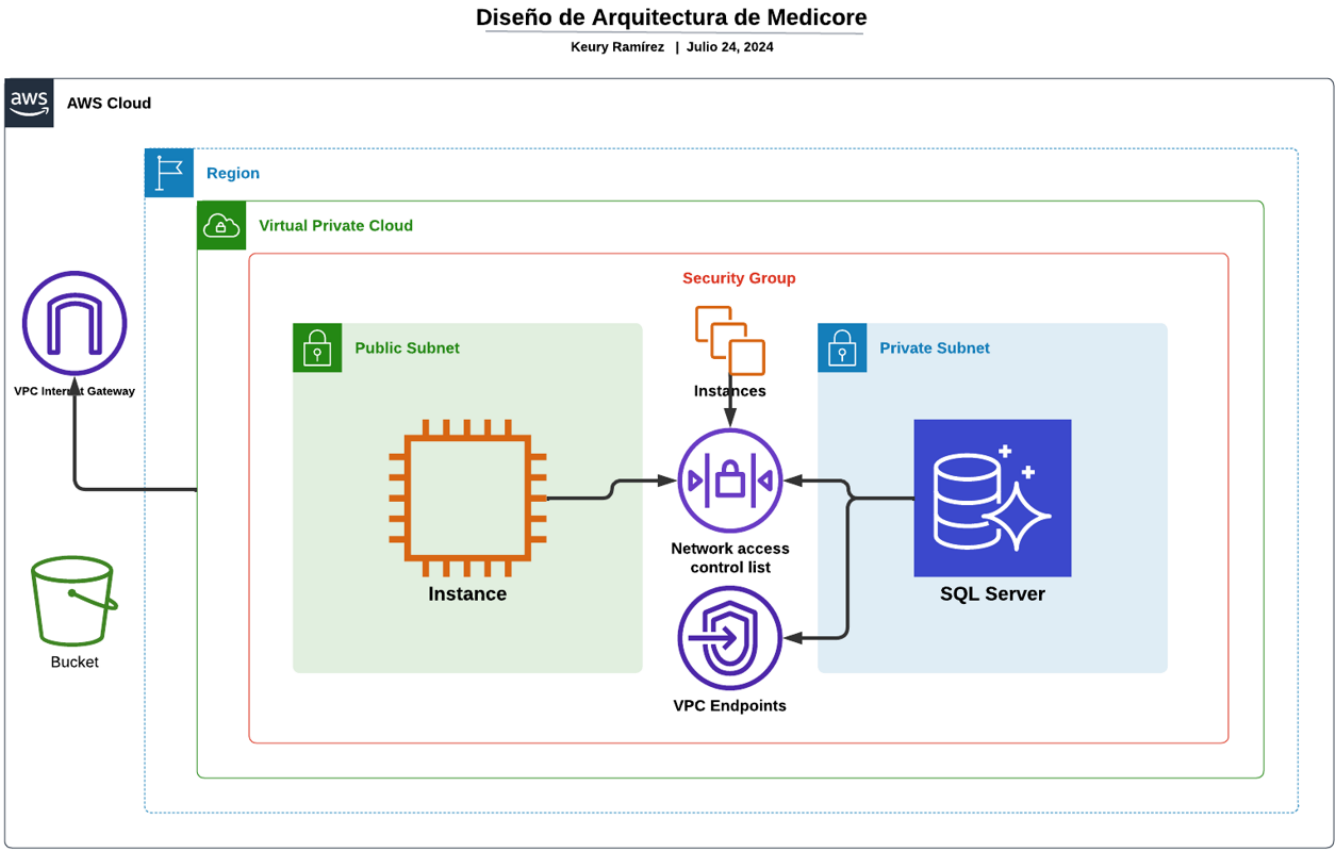
1.0 Descripción de las tablas - SQL Server.

- **Pacientes:** Almacena información personal de los pacientes.
- **Médicos:** Almacena información de los médicos y su especialidad.
- **Citas:** Registra las citas entre pacientes y médicos.
- **Tratamientos:** Define los diferentes tratamientos disponibles.
- **Medicamentos:** Almacena información sobre los medicamentos.
- **Prescripciones:** Relaciona las citas con los medicamentos prescritos.
- **Departamentos:** Almacena los diferentes departamentos del hospital.
- **Empleados:** Contiene información de los empleados del hospital.
- **Hospitalizaciones:** Registra los detalles de las hospitalizaciones de los pacientes.
- **Cirugías:** Almacena detalles sobre las cirugías realizadas.
- **Laboratorios:** Contiene información sobre los laboratorios asociados.
- **Resultados_Laboratorio:** Almacena los resultados de las pruebas de laboratorio.
- **Facturas:** Registra las facturas generadas para los pacientes.
- **Pagos:** Registra los pagos realizados por los pacientes.

1.1 Estructura de la Base de Datos.



1.2 Arquitectura de la Base de Datos.



1.3 Desarrollo de las tablas.

Cada tabla de la base de datos **Medicore** está diseñada para almacenar datos específicos. A continuación se describen cada tabla y sus campos:

Pacientes.

- ID (INT, IDENTITY, PRIMARY KEY)
- Nombre (VARCHAR(100), NOT NULL)
- Apellido (VARCHAR(100), NOT NULL)
- FechaNacimiento (DATE, NOT NULL)
- Genero (ENUM('M', 'F'), NOT NULL)
- Direccion (VARCHAR(255))
- Telefono (VARCHAR(20))
- Correo (VARCHAR(100))
- SeguroMedico (VARCHAR(100))
- NumeroSeguroSocial (VARCHAR(20))

Médicos.

- ID (INT, IDENTITY, PRIMARY KEY)
- Nombre (VARCHAR(100), NOT NULL)
- Apellido (VARCHAR(100), NOT NULL)
- Especialidad (VARCHAR(100), NOT NULL)
- Telefono (VARCHAR(20))
- Correo (VARCHAR(100))

Citas.

- ID (INT, IDENTITY, PRIMARY KEY)
- PacienteID (INT, NOT NULL, FOREIGN KEY)
- MedicoID (INT, NOT NULL, FOREIGN KEY)
- FechaHora (DATETIME, NOT NULL)
- Motivo (TEXT)

Tratamientos.

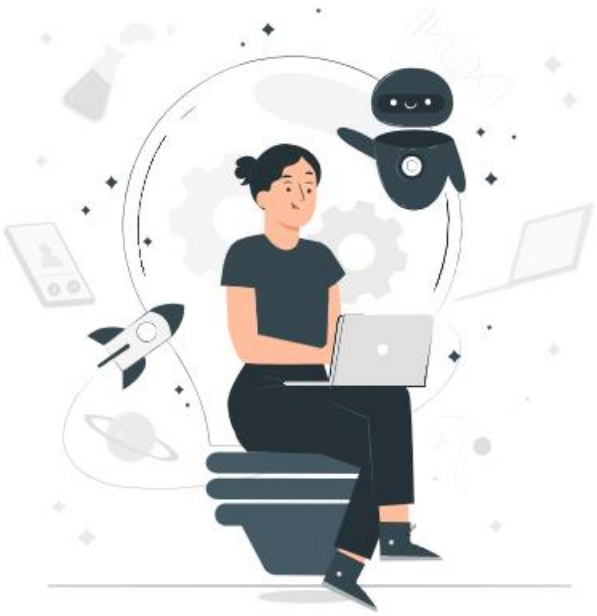
- ID (INT, IDENTITY, PRIMARY KEY)
- Nombre (VARCHAR(100), NOT NULL)
- Descripcion (TEXT)

Medicamentos

- ID (INT, IDENTITY, PRIMARY KEY)
- Nombre (VARCHAR(100), NOT NULL)
- Descripcion (TEXT)
- Precio (DECIMAL(10, 2), NOT NULL)

Prescripciones.

- ID (INT, IDENTITY, PRIMARY KEY)
- CitaID (INT, NOT NULL, FOREIGN KEY)
- MedicamentoID (INT, NOT NULL, FOREIGN KEY)
- Dosis (VARCHAR(100), NOT NULL)
- Duracion (INT, NOT NULL) -- en días
- Instrucciones (TEXT)



Departamentos.

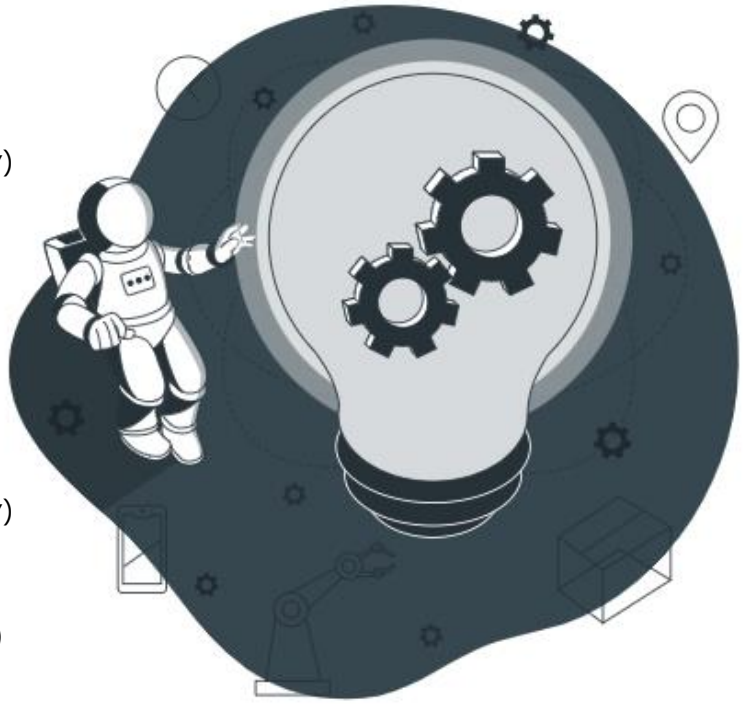
- ID (INT, IDENTITY, PRIMARY KEY)
- Nombre (VARCHAR(100), NOT NULL)
- Descripcion (TEXT)

Empleados.

- ID (INT, IDENTITY, PRIMARY KEY)
- Nombre (VARCHAR(100), NOT NULL)
- Apellido (VARCHAR(100), NOT NULL)
- FechaNacimiento (DATE, NOT NULL)
- Genero (ENUM('M', 'F'), NOT NULL)
- Direccion (VARCHAR(255))
- Telefono (VARCHAR(20))
- Correo (VARCHAR(100))
- DepartamentoID (INT, NOT NULL, FOREIGN KEY)
- Cargo (VARCHAR(100), NOT NULL)

Hospitalizaciones.

- ID (INT, IDENTITY, PRIMARY KEY)
- PacienteID (INT, NOT NULL, FOREIGN KEY)
- MedicoID (INT, NOT NULL, FOREIGN KEY)
- FechaIngreso (DATETIME, NOT NULL)
- FechaAlta (DATETIME)
- Motivo (TEXT)



Cirugías.

- ID (INT, IDENTITY, PRIMARY KEY)
- PacienteID (INT, NOT NULL, FOREIGN KEY)
- MedicoID (INT, NOT NULL, FOREIGN KEY)
- FechaHora (DATETIME, NOT NULL)
- TipoCirugia (VARCHAR(100), NOT NULL)
- Descripcion (TEXT)

Laboratorios.

- ID (INT, IDENTITY, PRIMARY KEY)
- Nombre (VARCHAR(100), NOT NULL)
- Direccion (VARCHAR(255))
- Telefono (VARCHAR(20))
- Correo (VARCHAR(100))

Resultados_Laboratorio.

- ID (INT, IDENTITY, PRIMARY KEY)
- PacienteID (INT, NOT NULL, FOREIGN KEY)
- MedicoID (INT, NOT NULL, FOREIGN KEY)
- LaboratorioID (INT, NOT NULL, FOREIGN KEY)
- Fecha (DATETIME, NOT NULL)
- TipoPrueba (VARCHAR(100), NOT NULL)
- Resultados (TEXT)

Facturas.

- ID (INT, IDENTITY, PRIMARY KEY)
- PacienteID (INT, NOT NULL, FOREIGN KEY)
- Fecha (DATETIME, NOT NULL)
- MontoTotal (DECIMAL(10, 2), NOT NULL)
- Descripcion (TEXT)

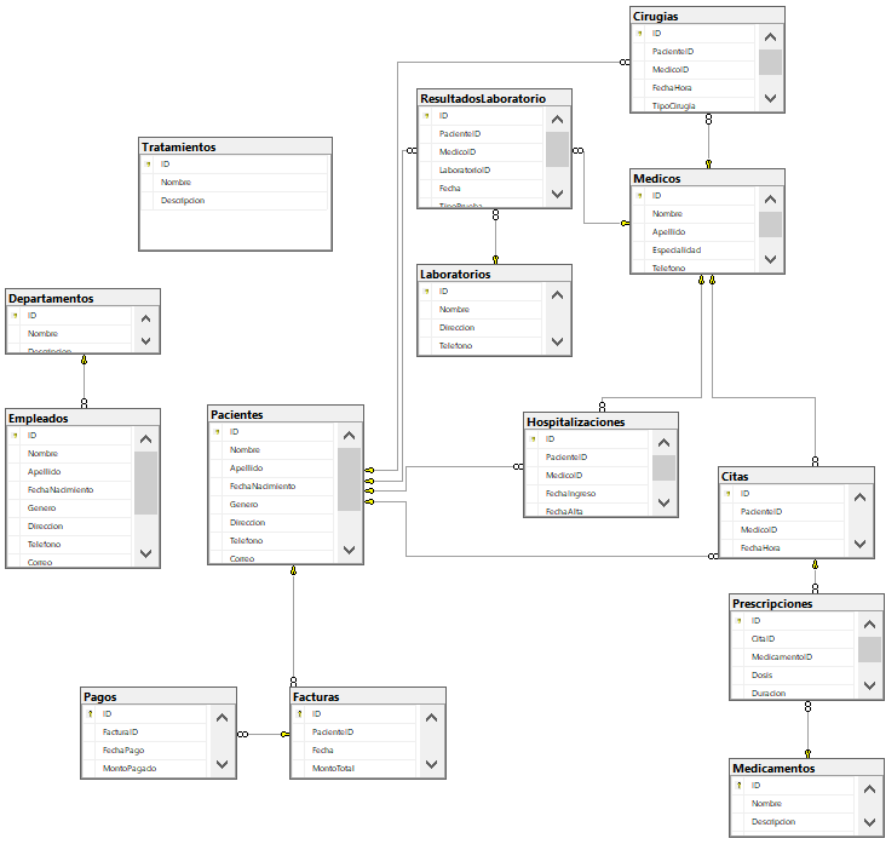
Pagos.

- ID (INT, IDENTITY, PRIMARY KEY)
- FacturaID (INT, NOT NULL, FOREIGN KEY)
- FechaPago (DATETIME, NOT NULL)
- MontoPagado (DECIMAL(10, 2), NOT NULL)
- MetodoPago (VARCHAR(100), NOT NULL)

1.4 Relación entre Tablas.

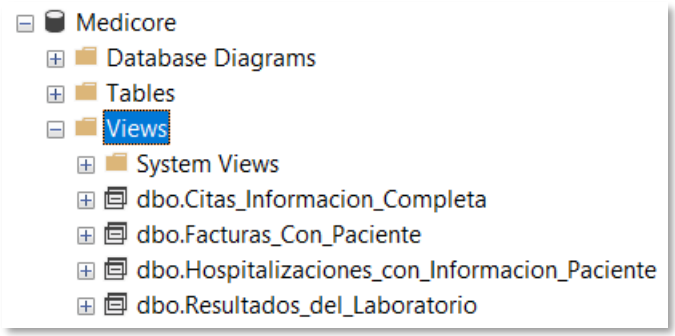
Las relaciones entre las tablas son esenciales para mantener la integridad referencial en la base de datos. A continuación se describe cómo se relacionan las tablas entre sí:

- La tabla **Citas** se relaciona con Pacientes y Médicos a través de las claves foráneas PacienteID y MedicoID.
- La tabla **Prescripciones** se relaciona con Citas y Medicamentos a través de las claves foráneas CitaID y MedicamentoID.
- La tabla **Empleados** se relaciona con Departamentos a través de la clave foránea DepartamentoID.
- La tabla **Hospitalizaciones** se relaciona con Pacientes y Medicos a través de las claves foráneas PacienteID y MedicoID.
- La tabla **Cirugías** se relaciona con Pacientes y Medicos a través de las claves foráneas PacienteID y MedicoID.
- La tabla **Resultados_Laboratorio** se relaciona con Pacientes, Médicos y Laboratorios a través de las claves foráneas PacienteID, MedicoID y LaboratorioID.
- La tabla **Facturas** se relaciona con Pacientes a través de la clave foránea PacienteID.
- La tabla **Pagos** se relaciona con Facturas a través de la clave foránea FacturaID.

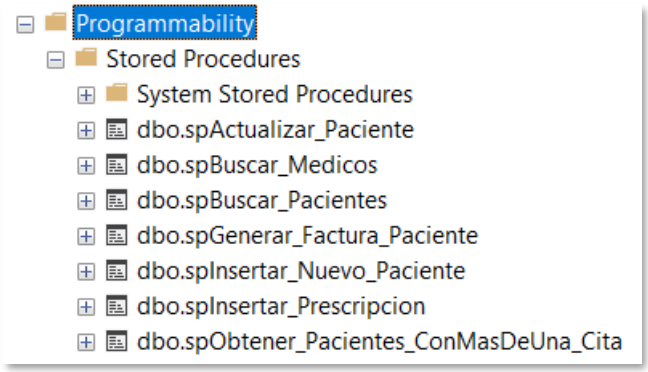


1.5 Vistas y Procedimientos En Medicore.

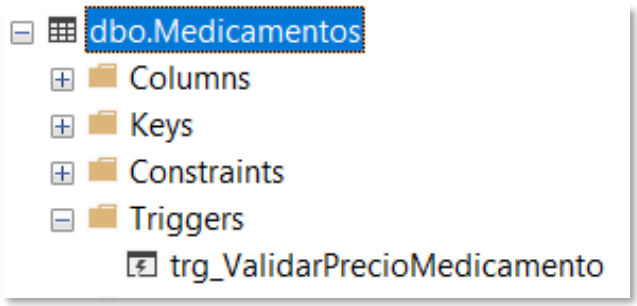
❖ **Vistas de Medicore.**

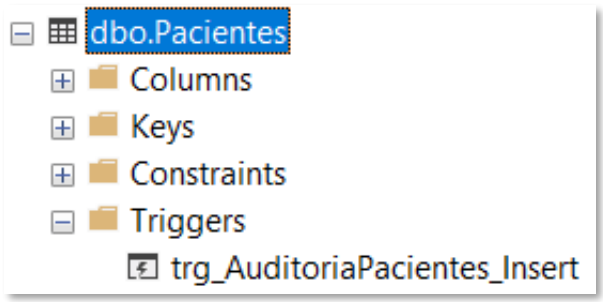


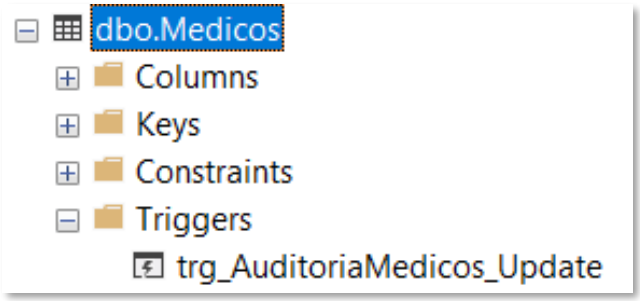
❖ **Procedimientos de Medicore.**



1.6 Triggers En Medicore.







1.7 Seguridad y Control de Acceso.

Roles y Permisos:

- **Administrador:** Acceso total a todas las tablas y operaciones.
- Doctor: Acceso a las tablas relacionadas con pacientes, citas, tratamientos y resultados de laboratorio.
- **Enfermero:** Acceso a las tablas relacionadas con pacientes y hospitalizaciones.
- **Recepcionista:** Acceso a las tablas de citas y pacientes para gestionar las reservas de citas.

Medidas de Seguridad:

- **Encriptación de Datos Sensibles:** Los datos como números de seguro social y direcciones de correo deben ser encriptados.
- **Control de Acceso Basado en Roles (RBAC):** Implementación de permisos basados en roles para asegurar que los usuarios solo puedan acceder a la información necesaria para sus funciones.

1.8 Mantenimiento y Backup.

Planes de Mantenimiento:

- **Re-indexación:** Reindexar las tablas periódicamente para mantener el rendimiento.
- **Actualización de Estadísticas:** Mantener actualizadas las estadísticas de la base de datos para optimizar las consultas.

Estrategias de Backup y Recuperación.

- **Backup Diario:** Realizar copias de seguridad diarias de la base de datos.
- **Recuperación ante Desastres:** Tener un plan de recuperación documentado y probado para restaurar la base de datos en caso de fallo.

1.9 Ejemplo de consultas avanzadas.

-- Obtener todos los pacientes.

```
SELECT * FROM Pacientes;
```

-- Obtener todos los pacientes masculinos

```
SELECT * FROM Pacientes
WHERE Genero = 'M';
```

-- Obtener los pacientes que tienen más de una cita.

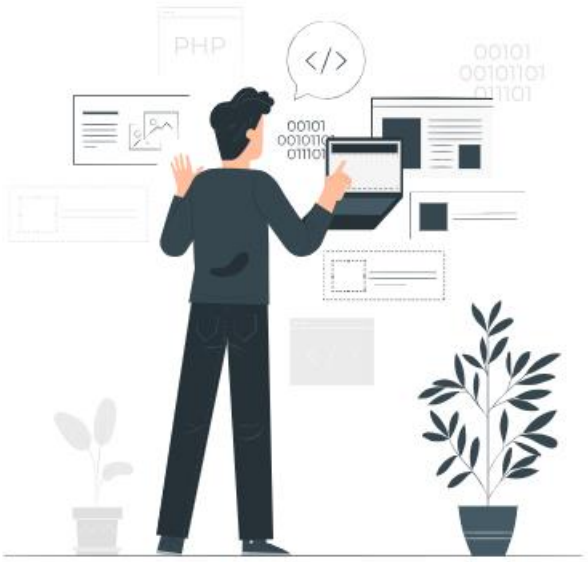
```
SELECT p.ID, p.Nombre, p.Apellido
FROM Pacientes p
JOIN Citas c ON p.ID = c.PacienteID
GROUP BY p.ID, p.Nombre, p.Apellido
HAVING COUNT(c.ID) > 1; --Todos tienen solo 1 cita, para confirmar cambiar > por =.
```

-- Obtener todas las citas con información del paciente y del médico.

```
SELECT c.ID,
        p.Nombre AS PacienteNombre,
        p.Apellido AS PacienteApellido,
    m.Nombre AS MedicoNombre,
        m.Apellido AS MedicoApellido,
    c.FechaHora,
        c.Motivo
FROM Citas c
JOIN Pacientes p ON c.PacienteID = p.ID
JOIN Medicos m ON c.MedicoID = m.ID;
```

-- Obtener los pacientes con sus últimos resultados de laboratorio.

```
SELECT p.ID, p.Nombre,
        p.Apellido,
        rl.TipoPrueba,
        rl.Resultados,
        rl.Fecha
FROM Pacientes p
JOIN ResultadosLaboratorio rl
ON p.ID = rl.PacienteID
WHERE rl.Fecha = (SELECT MAX(Fecha)
FROM ResultadosLaboratorio
WHERE PacienteID = p.ID);
```



1.10 Casos de Uso.

Insertar Nuevo Paciente.
Descripción: Permite agregar un nuevo paciente a la base de datos con todos sus detalles, como nombre, apellido, fecha de nacimiento, género, dirección, teléfono, correo electrónico, seguro médico y número de seguro social.

Actualizar Información de un Paciente.
Descripción: Permite modificar la información de un paciente existente, como su dirección o número de teléfono, cuando haya cambios en sus datos personales.

Eliminar un Paciente.
Descripción: Permite eliminar un paciente de la base de datos. Esto también eliminará cualquier referencia a ese paciente en otras tablas, dependiendo de cómo se maneje la integridad referencial.

Buscar Pacientes por Nombre.
Descripción: Permite buscar pacientes cuyo nombre o apellido contengan una cadena específica. Es útil para encontrar rápidamente pacientes cuando solo se tiene información parcial.

Buscar Médicos por Nombre.
Descripción: Permite buscar médicos cuyo nombre o apellido contengan una cadena específica. Facilita encontrar médicos cuando solo se tiene información parcial.

Obtener Pacientes con Más de Una Cita.
Descripción: Muestra una lista de pacientes que tienen más de una cita registrada en la base de datos. Esto puede ayudar a identificar pacientes que requieren atención continua o seguimiento frecuente.

Obtener Médicos con Hospitalizaciones
Descripción: Muestra una lista de médicos que han tenido pacientes hospitalizados. Esto ayuda a identificar a los médicos que están involucrados en hospitalizaciones.

Verificar el Precio de Medicamentos.
Descripción: Asegura que el precio de un medicamento no sea menor que 0 al insertar o actualizar datos de medicamentos. Esto evita errores de precio en la base de datos.

Auditar Cambios en Pacientes y Médicos.
Descripción: Registra cambios en los datos de pacientes y médicos, así como eliminaciones, para mantener un historial de auditoría. Esto es útil para el seguimiento y la seguridad de los datos.

Registrar Resultados de Laboratorio.
Descripción: Permite ingresar y almacenar los resultados de pruebas de laboratorio realizadas a los pacientes, junto con la información del laboratorio y el médico que solicitó la prueba.

Generar y Registrar Facturas.
Descripción: Permite generar facturas para los pacientes y registrar los pagos realizados. Incluye detalles del monto total, descripción y fecha de emisión de la factura.

Actualizar Fechas de Alta en Hospitalizaciones.
Descripción: Actualiza automáticamente la fecha de alta para los pacientes cuando se registra una hospitalización, garantizando que la información esté siempre actualizada.

1.11 Flujos de Trabajo.

Proceso de Admisión de Pacientes:

- Recepción registra al paciente en la tabla Pacientes.
- Se programa una cita en la tabla Citas.
- El médico consulta al paciente y prescribe tratamientos registrados en Prescripciones.

Proceso de Facturación:

- Una vez completado el tratamiento, se genera una factura en la tabla Facturas.
- El pago es registrado en la tabla Pagos.

1.12 Apéndices.

- **PacienteID:** Identificador único de cada paciente.
- **MedicoID:** Identificador único de cada médico.
- **CitaID:** Identificador único de cada cita.
- **Prescripción:** Instrucción de un médico para el tratamiento de un paciente.

2.0 MongoDB - MEDICORE.

Visión General del Proyecto:

El sistema está diseñado para gestionar información de pacientes, médicos, citas, hospitalizaciones, cirugías, resultados de laboratorio, facturas y prescripciones. La base de datos en MongoDB almacena esta información de manera estructurada para facilitar el acceso y la gestión eficiente de datos médicos.

2.1 Arquitectura de la Base de Datos.

Modelo de Datos.

La base de datos está compuesta por las siguientes colecciones:

- **Pacientes:** Información personal y médica de los pacientes.
- **Medicos:** Datos de los médicos, incluyendo especialidades y contactos.
- **Citas:** Información sobre citas programadas entre pacientes y médicos.
- **Hospitalizaciones:** Detalles de hospitalizaciones de pacientes.
- **Cirugias:** Información sobre cirugías realizadas.
- **Resultados_Laboratorios:** Resultados de pruebas de laboratorio.
- **Facturas:** Facturación relacionada con pacientes y servicios médicos.
- **Prescripciones:** Medicamentos recetados durante citas médicas.

Diagrama de Entidad-Relación.

Nota: Aunque MongoDB no utiliza diagramas ER tradicionales, un diagrama simple puede ayudar a entender la estructura de colecciones y sus relaciones.

2.2 Estrategias de Diseño.

Desnormalización

Para mejorar el rendimiento de las consultas y simplificar el acceso a datos relacionados, se han incluido documentos embebidos en lugar de referencias en algunas colecciones. Por ejemplo, en la colección de Citas, se embebe la información del paciente y del médico en cada documento de cita.

Indexación

Los siguientes índices se han creado para optimizar las consultas:

- **Pacientes:** Índice en el campo **Correo** para búsquedas rápidas por correo electrónico.
- **Citas:** Índice en el campo **FechaHora** para consultas basadas en la fecha y hora de la cita.

2.3 Rendimiento.

Optimización.

Se han implementado índices en campos clave y se han utilizado estrategias de desnormalización para mejorar el rendimiento de las consultas.

Análisis de Rendimiento.

Se utiliza el perfilador de MongoDB y herramientas de monitoreo para identificar y resolver problemas de rendimiento.

2.4 Manejo de Errores.

Errores Comunes.

- **Error de conexión:** Asegúrate de que el servidor MongoDB esté en funcionamiento y la red esté correctamente configurada.
- **Error de índice:** Verifica que los índices estén correctamente definidos y actualizados.

Manejo de Excepciones.

Implementa manejo de errores en el código de aplicación para capturar y manejar excepciones relacionadas con la base de datos de manera adecuada.

2.5 Seguridad y Acceso.

Controles de Acceso.

Los roles y permisos están configurados para asegurar que solo los usuarios autorizados puedan acceder o modificar datos sensibles. MongoDB utiliza autenticación basada en usuario para controlar el acceso a la base de datos.

Autenticación y Autorización.

Se recomienda utilizar autenticación de usuario con roles específicos para operaciones de lectura y escritura. Las credenciales de acceso deben ser manejadas de forma segura.

2.6 Ejemplos de Consultas.

Obtener todos los pacientes:

```
db.Pacientes.find({})
```

Obtener citas de un paciente específico:

```
db.Citas.find({"Paciente.Correo": "juan@example.com"})
```

2.7 Consultas Avanzadas.

Obtener resultados de laboratorio para un paciente específico en un rango de fechas:

```
db.ResultadosLaboratorios.find({ "Paciente.Correo": "juan@example.com", "Fecha": { "$gte": ISODate("2023-01-01T00:00:00Z"), "$lte": ISODate("2023-12-31T23:59:59Z") } })
```

Actualizar el número de teléfono de un paciente:

```
db.Pacientes.updateOne( { "Correo": "juan@example.com" },
{ $set: { "Telefono": "987654321" } } )
```

Eliminar una cita específica:

```
db.Citas.deleteOne({ "FechaHora": ISODate("2023-07-18T10:00:00Z") })
```

2.8 Mantenimiento y Copias de Seguridad.

Backup.

Se realizan copias de seguridad diarias. Las copias de seguridad se almacenan en un entorno seguro y se verifican regularmente.

Restauración.

Las copias de seguridad se pueden restaurar utilizando mongorestore. Este procedimiento se realiza en un entorno de pruebas antes de aplicar la restauración en producción.

Monitoreo y Mantenimiento.

El monitoreo continuo se realiza utilizando herramientas como MongoDB Atlas o herramientas de terceros para asegurar el rendimiento y la disponibilidad de la base de datos.

2.9 Escalabilidad.

Estrategias de Escalado.

La base de datos se puede escalar verticalmente agregando recursos al servidor o horizontalmente mediante el sharding (si se requiere en el futuro).

2.10 Glosario.

- **MongoDB:** Base de datos NoSQL orientada a documentos.
- **Colección:** Conjunto de documentos en MongoDB, similar a una tabla en SQL.
- **Documento:** Unidad básica de datos en MongoDB, similar a una fila en SQL.

2.11 Apéndices.

Script de Backup:

```
mongodump --db mi_base_de_datos --out /ruta/del/backup
```

Script de Restauración:

```
mongorestore --db mi_base_de_datos /ruta/del/backup/mi_base_de_datos
```

3.0 REDIS – MEDICORE.

Descripción General

Esta aplicación de consola está diseñada para interactuar con una base de datos en Redis. Redis es una base de datos en memoria que utiliza estructuras de datos clave-valor. En este caso, la aplicación organiza la información en hashes, que son colecciones de pares clave-valor almacenados en Redis.

3.1 Características Principales.

Base de Datos en Redis:

- Utiliza RedisDB como sistema de gestión de base de datos en memoria.
- Almacena datos en estructuras de tipo hash, que permiten una organización eficiente de datos estructurados.

Estructura de Datos:

- Hashes: Los datos se organizan en hashes, con cada hash representando una entidad de las tablas (por ejemplo, pacientes, medicos, citas, etc.).
- Claves: Los hashes están organizados por identificador único (ID), facilitando la recuperación específica de datos.

Aplicación de Consola en C#:

- La aplicación está construida en C# y utiliza la biblioteca StackExchange.Redis para interactuar con Redis.
- Proporciona una interfaz de usuario simple y directa en la consola para la administración y consulta de datos.

Menú Interactivo:

- Menú Principal: Presenta un menú de selección con opciones numeradas para acceder a diferentes tablas de la base de datos.
- Encabezado Personalizado: Incluye un encabezado visualmente atractivo hecho con tipografía ASCII para mejorar la experiencia del usuario.

Operaciones Soportadas:

- Visualización de Datos: Permite al usuario seleccionar una tabla, ingresar un identificador de hash y ver los datos correspondientes.
- Interactividad: El usuario puede navegar por el menú y ver los detalles de las entidades almacenadas en Redis.

3.2 Uso de la Aplicación.

Ejecución:

- La aplicación se ejecuta desde la consola de comandos.
- Al iniciar, muestra el menú principal con opciones numeradas para diferentes tablas.

Selección de Tabla:

- El usuario elige una tabla ingresando el número correspondiente (1-8).
- Después de seleccionar la tabla, se solicita al usuario que ingrese el identificador del hash para recuperar los datos.

Visualización de Datos:

- La aplicación muestra los datos del hash seleccionado en un formato legible.
- Si no se encuentran datos para la clave proporcionada, se muestra un mensaje informativo.

Salir de la Aplicación:

- La opción 0 en el menú permite al usuario salir de la aplicación.

3.3 Ejemplo de Uso.

```
-------------------------------------
       Bienvenido a Medicore,
  Seleccione una de nuestras tablas.
-------------------------------------
  1. Pacientes
  2. Médicos
  3. Citas
  4. Hospitalizaciones
  5. Cirugías
  6. Resultados de Laboratorio
  7. Facturas
  8. Prescripciones
  0. Salir
-------------------------------------

Opción (Ej.1): 5
Ingresa el identificador (Ej.1): 1

Datos para cirugias:1
Paciente.Nombre: Juan
Paciente.Apellido: Pérez
Paciente.FechaNacimiento: 1980-01-01
Paciente.Genero: M
Paciente.Direccion: Calle Falsa 123
Paciente.Telefono: 123456789
Paciente.Correo: juan@example.com
Paciente.SeguroMedico: Seguro XYZ
Paciente.NumeroSeguroSocial: 123-45-6789
Medico.Nombre: Dr. Ana
Medico.Apellido: López
Medico.Especialidad: Cardiología
Medico.Telefono: 987654321
Medico.Correo: ana@example.com
FechaHora: 2023-07-20T09:00:00Z
TipoCirugia: Apendicectomía
Descripcion: Extirpación del apéndice

Presiona una tecla para continuar...
```

3.4 Requisitos.

- **Redis:** Asegúrate de que el servidor Redis esté en funcionamiento y accesible desde la aplicación.
- **Bibliotecas:** La aplicación requiere la biblioteca StackExchange.Redis. Instálala a través de NuGet en tu proyecto C#.

4.0 Conclusión General de Medicore.

La documentación proporciona un análisis exhaustivo del sistema MEDICORE, diseñado para optimizar la gestión hospitalaria mediante la integración de diversas tecnologías de bases de datos. En SQL Server, el sistema estructura la información hospitalaria en una serie de tablas interrelacionadas que abarcan desde los datos personales de pacientes y médicos hasta la gestión de citas, tratamientos, y resultados de laboratorio, garantizando una administración eficiente y una integridad referencial robusta. Por otro lado, MongoDB se presenta como una solución flexible con un modelo de datos desnormalizado, lo que permite una optimización de consultas mediante la desnormalización de datos y la implementación de índices, además de facilitar una gestión dinámica de errores y una escalabilidad efectiva. Finalmente, Redis contribuye con su arquitectura en memoria que organiza la información en hashes, proporcionando un acceso ultrarrápido a los datos y una administración eficiente a través de una interfaz en consola construida en C#. La documentación también detalla aspectos esenciales como la seguridad, el mantenimiento, y las estrategias de respaldo, abordando desde la encriptación de datos hasta la recuperación ante desastres. En conjunto, MEDICORE combina estas tecnologías avanzadas para ofrecer una solución integral y escalable, optimizando la gestión de datos hospitalarios y asegurando una operación continua y fiable.