Proyecto EPS – Unidad 4

Institución Universitaria ITM

Estudiante: Luis Enrique Morales Montoya

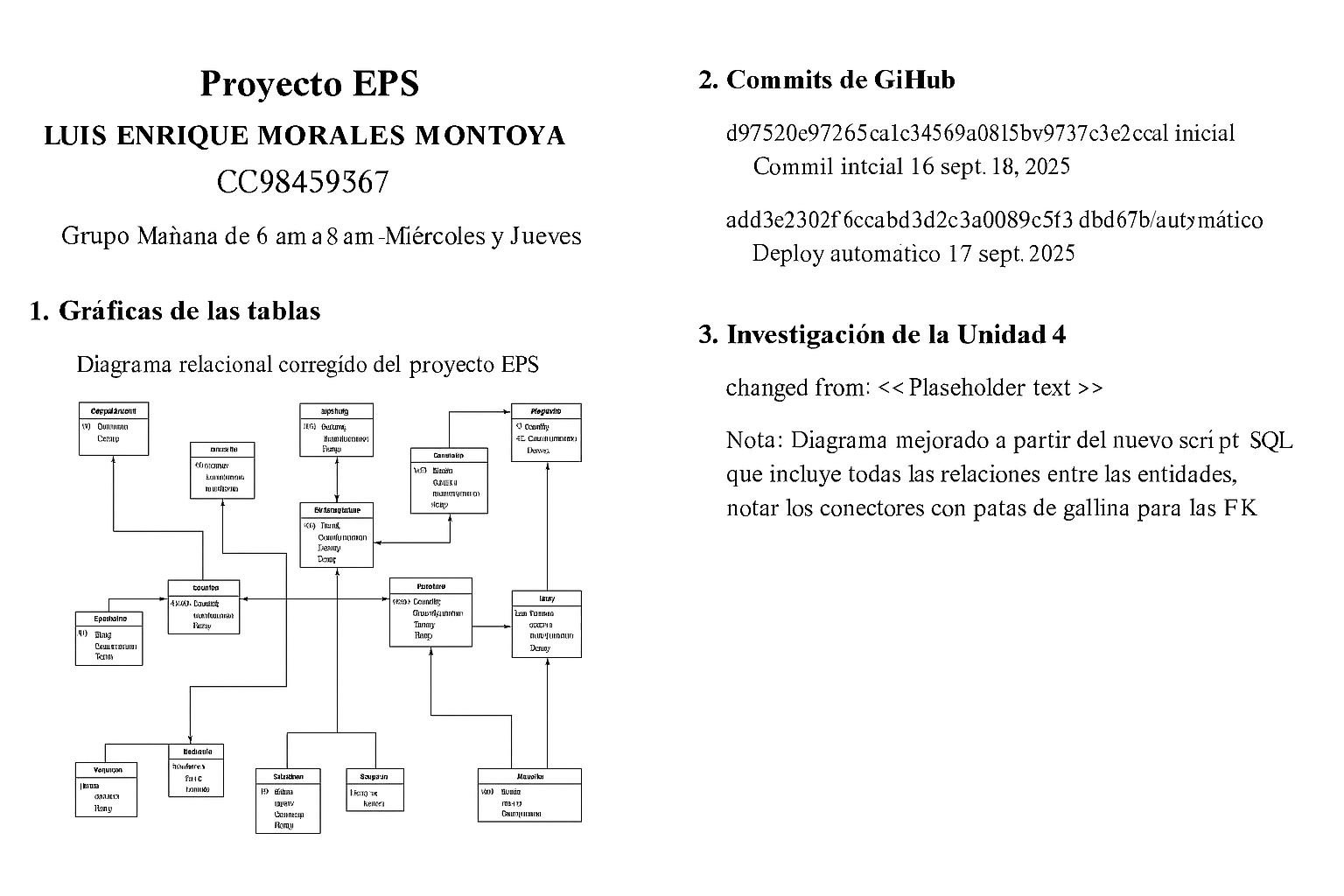
CC: 98459567

Grupo: Miércoles y Jueves 6 a.m. - 8 a.m.

Repositorio: https://github.com/kike5767/eps.git

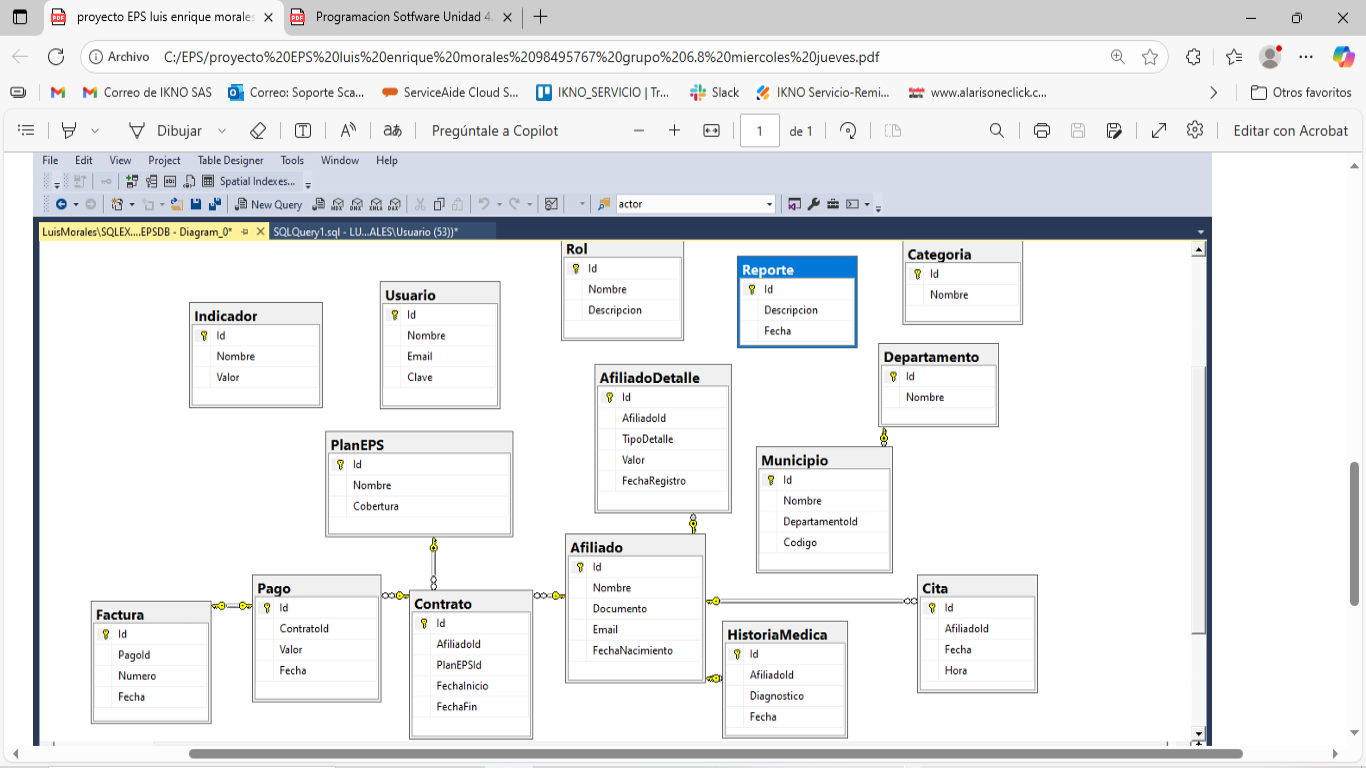
**1. Modelo de Base de Datos**

Diagrama Relacional Corregido – Proyecto EPS



Nota aclaratoria: El siguiente diagrama corresponde al entregado inicialmente. Contenía tablas sin relación y errores en SQL. Estos problemas fueron corregidos en el modelo actualizado presentado anteriormente.

Modelo Inicial (con errores):



**2. Script SQL Server Corregido (completo)**

-------------------------------------------------------  
-- Script SQL Corregido Proyecto EPS / SistemaIndicadores  
-- 1) Backup (ejecutar en PowerShell): New-Item -ItemType Directory -Force -Path "C:\BACKUPSINCORREGIR"  
-------------------------------------------------------  
BACKUP DATABASE [SistemaIndicadores]  
TO DISK = 'C:\BACKUPSINCORREGIR\SistemaIndicadores\_Respaldo.bak'  
WITH FORMAT, INIT,  
 NAME = 'Backup antes de recrear estructura',  
 SKIP, NOREWIND, NOUNLOAD, STATS = 10;  
GO  
  
USE master;  
IF EXISTS (SELECT name FROM sys.databases WHERE name = N'SistemaIndicadores')  
BEGIN  
 ALTER DATABASE SistemaIndicadores SET SINGLE\_USER WITH ROLLBACK IMMEDIATE;  
 DROP DATABASE SistemaIndicadores;  
END  
GO  
  
CREATE DATABASE SistemaIndicadores;  
GO  
USE SistemaIndicadores;  
GO  
  
-- Tablas maestras  
CREATE TABLE Departamento (  
 Id INT IDENTITY(1,1) PRIMARY KEY,  
 Nombre VARCHAR(50) NOT NULL  
);  
  
CREATE TABLE Municipio (  
 Id INT IDENTITY(1,1) PRIMARY KEY,  
 Nombre VARCHAR(100) NOT NULL,  
 DepartamentoId INT NOT NULL,  
 Codigo VARCHAR(10) NOT NULL,  
 CONSTRAINT FK\_Municipio\_Departamento FOREIGN KEY (DepartamentoId) REFERENCES Departamento(Id)  
);  
  
CREATE TABLE Categoria (  
 Id INT IDENTITY(1,1) PRIMARY KEY,  
 Nombre VARCHAR(50) NOT NULL  
);  
  
CREATE TABLE Rol (  
 Id INT IDENTITY(1,1) PRIMARY KEY,  
 Nombre VARCHAR(50) NOT NULL,  
 Descripcion VARCHAR(200) NOT NULL  
);  
  
CREATE TABLE PlanEPS (  
 Id INT IDENTITY(1,1) PRIMARY KEY,  
 Nombre VARCHAR(50) NOT NULL,  
 Cobertura VARCHAR(50) NOT NULL  
);  
  
-- Entidades principales y dependientes  
CREATE TABLE Afiliado (  
 Id INT IDENTITY(1,1) PRIMARY KEY,  
 Nombre VARCHAR(50) NOT NULL,  
 Documento VARCHAR(20) NOT NULL,  
 Email VARCHAR(50) NOT NULL,  
 FechaNacimiento DATE NOT NULL,  
 MunicipioId INT NOT NULL,  
 CategoriaId INT NOT NULL,  
 CONSTRAINT FK\_Afiliado\_Municipio FOREIGN KEY (MunicipioId) REFERENCES Municipio(Id),  
 CONSTRAINT FK\_Afiliado\_Categoria FOREIGN KEY (CategoriaId) REFERENCES Categoria(Id)  
);  
  
CREATE TABLE Usuario (  
 Id INT IDENTITY(1,1) PRIMARY KEY,  
 Nombre VARCHAR(50) NOT NULL,  
 Email VARCHAR(50) NOT NULL,  
 Clave VARCHAR(200) NOT NULL,  
 RolId INT NOT NULL,  
 CONSTRAINT FK\_Usuario\_Rol FOREIGN KEY (RolId) REFERENCES Rol(Id)  
);  
  
CREATE TABLE Cita (  
 Id INT IDENTITY(1,1) PRIMARY KEY,  
 AfiliadoId INT NOT NULL,  
 Fecha DATE NOT NULL,  
 Hora TIME NOT NULL,  
 CONSTRAINT FK\_Cita\_Afiliado FOREIGN KEY (AfiliadoId) REFERENCES Afiliado(Id)  
);  
  
CREATE TABLE AfiliadoDetalle (  
 Id INT IDENTITY(1,1) PRIMARY KEY,  
 AfiliadoId INT NOT NULL,  
 TipoDetalle VARCHAR(50) NOT NULL,  
 Valor VARCHAR(100) NOT NULL,  
 FechaRegistro DATETIME2 NOT NULL,  
 CONSTRAINT FK\_AfiliadoDetalle\_Afiliado FOREIGN KEY (AfiliadoId) REFERENCES Afiliado(Id)  
);  
  
CREATE TABLE Contrato (  
 Id INT IDENTITY(1,1) PRIMARY KEY,  
 AfiliadoId INT NOT NULL,  
 PlanEPSId INT NOT NULL,  
 FechaInicio DATE NOT NULL,  
 FechaFin DATE NULL,  
 CONSTRAINT FK\_Contrato\_Afiliado FOREIGN KEY (AfiliadoId) REFERENCES Afiliado(Id),  
 CONSTRAINT FK\_Contrato\_PlanEPS FOREIGN KEY (PlanEPSId) REFERENCES PlanEPS(Id)  
);  
  
CREATE TABLE Pago (  
 Id INT IDENTITY(1,1) PRIMARY KEY,  
 ContratoId INT NOT NULL,  
 Valor DECIMAL(18,2) NOT NULL CHECK (Valor >= 0),  
 Fecha DATE NOT NULL,  
 CONSTRAINT FK\_Pago\_Contrato FOREIGN KEY (ContratoId) REFERENCES Contrato(Id)  
);  
  
CREATE TABLE Factura (  
 Id INT IDENTITY(1,1) PRIMARY KEY,  
 PagoId INT NOT NULL UNIQUE,  
 Numero VARCHAR(50) NOT NULL,  
 Fecha DATE NOT NULL,  
 CONSTRAINT FK\_Factura\_Pago FOREIGN KEY (PagoId) REFERENCES Pago(Id)  
);  
  
CREATE TABLE HistoriaMedica (  
 Id INT IDENTITY(1,1) PRIMARY KEY,  
 AfiliadoId INT NOT NULL,  
 Diagnostico VARCHAR(100) NOT NULL,  
 Fecha DATE NOT NULL,  
 CONSTRAINT FK\_HistoriaMedica\_Afiliado FOREIGN KEY (AfiliadoId) REFERENCES Afiliado(Id)  
);  
  
CREATE TABLE Indicador (  
 Id INT IDENTITY(1,1) PRIMARY KEY,  
 Nombre VARCHAR(50) NOT NULL,  
 Valor DECIMAL(18,2) NOT NULL CHECK (Valor >= 0)  
);  
  
CREATE TABLE Reporte (  
 Id INT IDENTITY(1,1) PRIMARY KEY,  
 Descripcion VARCHAR(100) NOT NULL,  
 Fecha DATE NOT NULL,  
 IndicadorId INT NOT NULL,  
 CONSTRAINT FK\_Reporte\_Indicador FOREIGN KEY (IndicadorId) REFERENCES Indicador(Id)  
);  
  
-- Índices recomendados  
CREATE UNIQUE INDEX UX\_Afiliado\_Documento ON Afiliado(Documento);  
CREATE UNIQUE INDEX UX\_Factura\_Numero ON Factura(Numero);  
GO

**3. Tablas de Datos (5 registros por tabla)**

A continuación se muestran INSERTS de ejemplo (5 registros por tabla) en el orden correcto para respetar claves foráneas.

-- Departamentos  
INSERT INTO Departamento (Nombre) VALUES ('Antioquia');  
INSERT INTO Departamento (Nombre) VALUES ('Cundinamarca');  
INSERT INTO Departamento (Nombre) VALUES ('Valle del Cauca');  
INSERT INTO Departamento (Nombre) VALUES ('Bogotá D.C.');  
INSERT INTO Departamento (Nombre) VALUES ('Bolívar');  
  
-- Municipios  
INSERT INTO Municipio (Nombre, DepartamentoId, Codigo) VALUES ('Medellín', 1, '05001');  
INSERT INTO Municipio (Nombre, DepartamentoId, Codigo) VALUES ('Envigado', 1, '05002');  
INSERT INTO Municipio (Nombre, DepartamentoId, Codigo) VALUES ('Bogotá', 4, '11001');  
INSERT INTO Municipio (Nombre, DepartamentoId, Codigo) VALUES ('Cali', 3, '76001');  
INSERT INTO Municipio (Nombre, DepartamentoId, Codigo) VALUES ('Cartagena', 5, '13001');  
  
-- Categorias  
INSERT INTO Categoria (Nombre) VALUES ('Adulto');  
INSERT INTO Categoria (Nombre) VALUES ('Niño');  
INSERT INTO Categoria (Nombre) VALUES ('Joven');  
INSERT INTO Categoria (Nombre) VALUES ('Senior');  
INSERT INTO Categoria (Nombre) VALUES ('Gestante');  
  
-- Roles  
INSERT INTO Rol (Nombre, Descripcion) VALUES ('Admin','Administrador del sistema');  
INSERT INTO Rol (Nombre, Descripcion) VALUES ('Coordinador','Coordinador EPS');  
INSERT INTO Rol (Nombre, Descripcion) VALUES ('Medico','Profesional médico');  
INSERT INTO Rol (Nombre, Descripcion) VALUES ('Recepcion','Atención al usuario');  
INSERT INTO Rol (Nombre, Descripcion) VALUES ('Auditor','Auditor de procesos');  
  
-- Planes EPS  
INSERT INTO PlanEPS (Nombre, Cobertura) VALUES ('Plan Básico','Nacional');  
INSERT INTO PlanEPS (Nombre, Cobertura) VALUES ('Plan Plus','Nacional +');  
INSERT INTO PlanEPS (Nombre, Cobertura) VALUES ('Plan Oro','Cobertura ampliada');  
INSERT INTO PlanEPS (Nombre, Cobertura) VALUES ('Plan Platino','Cobertura premium');  
INSERT INTO PlanEPS (Nombre, Cobertura) VALUES ('Plan Empresarial','Para empresas');  
  
-- Indicadores (para los reportes)  
INSERT INTO Indicador (Nombre, Valor) VALUES ('Afiliados Activos', 1000);  
INSERT INTO Indicador (Nombre, Valor) VALUES ('Contratos Vigentes', 850);  
INSERT INTO Indicador (Nombre, Valor) VALUES ('Pagos Pendientes', 45);  
INSERT INTO Indicador (Nombre, Valor) VALUES ('Citas Agendadas Hoy', 30);  
INSERT INTO Indicador (Nombre, Valor) VALUES ('Historias Medicas Registradas', 500);  
  
-- Afiliados (usa MunicipioId 1..5, CategoriaId 1..5)  
INSERT INTO Afiliado (Nombre, Documento, Email, FechaNacimiento, MunicipioId, CategoriaId) VALUES ('Juan Pérez', '10001', 'juan.perez@mail.com', '1980-02-10', 1, 1);  
INSERT INTO Afiliado (Nombre, Documento, Email, FechaNacimiento, MunicipioId, CategoriaId) VALUES ('María Gómez', '10002', 'maria.gomez@mail.com', '1992-07-20', 2, 3);  
INSERT INTO Afiliado (Nombre, Documento, Email, FechaNacimiento, MunicipioId, CategoriaId) VALUES ('Carlos Rodríguez', '10003', 'carlos.rodr@mail.com', '1975-11-05', 3, 4);  
INSERT INTO Afiliado (Nombre, Documento, Email, FechaNacimiento, MunicipioId, CategoriaId) VALUES ('Ana Martínez', '10004', 'ana.mart@mail.com', '2001-03-15', 4, 2);  
INSERT INTO Afiliado (Nombre, Documento, Email, FechaNacimiento, MunicipioId, CategoriaId) VALUES ('Luis Morales', '10005', 'luis.morales@mail.com', '1998-09-30', 5, 1);  
  
-- Usuarios (RolId 1..5)  
INSERT INTO Usuario (Nombre, Email, Clave, RolId) VALUES ('admin', 'admin@eps.com', 'HashedPassword1', 1);  
INSERT INTO Usuario (Nombre, Email, Clave, RolId) VALUES ('coord', 'coord@eps.com', 'HashedPassword2', 2);  
INSERT INTO Usuario (Nombre, Email, Clave, RolId) VALUES ('medico', 'medico@eps.com', 'HashedPassword3', 3);  
INSERT INTO Usuario (Nombre, Email, Clave, RolId) VALUES ('reception', 'recep@eps.com', 'HashedPassword4', 4);  
INSERT INTO Usuario (Nombre, Email, Clave, RolId) VALUES ('auditor', 'auditor@eps.com', 'HashedPassword5', 5);  
  
-- Contratos (AfiliadoId 1..5, PlanEPSId 1..5)  
INSERT INTO Contrato (AfiliadoId, PlanEPSId, FechaInicio, FechaFin) VALUES (1,1,'2023-01-01', NULL);  
INSERT INTO Contrato (AfiliadoId, PlanEPSId, FechaInicio, FechaFin) VALUES (2,2,'2023-06-01', NULL);  
INSERT INTO Contrato (AfiliadoId, PlanEPSId, FechaInicio, FechaFin) VALUES (3,3,'2022-05-15', '2023-05-14');  
INSERT INTO Contrato (AfiliadoId, PlanEPSId, FechaInicio, FechaFin) VALUES (4,4,'2024-01-10', NULL);  
INSERT INTO Contrato (AfiliadoId, PlanEPSId, FechaInicio, FechaFin) VALUES (5,5,'2024-03-20', NULL);  
  
-- Pagos (ContratoId 1..5)  
INSERT INTO Pago (ContratoId, Valor, Fecha) VALUES (1, 120000.00, '2024-08-01');  
INSERT INTO Pago (ContratoId, Valor, Fecha) VALUES (2, 150000.00, '2024-08-05');  
INSERT INTO Pago (ContratoId, Valor, Fecha) VALUES (3, 90000.00, '2023-04-20');  
INSERT INTO Pago (ContratoId, Valor, Fecha) VALUES (4, 250000.00, '2024-09-01');  
INSERT INTO Pago (ContratoId, Valor, Fecha) VALUES (5, 300000.00, '2024-09-10');  
  
-- Facturas (PagoId 1..5)  
INSERT INTO Factura (PagoId, Numero, Fecha) VALUES (1, 'FAC-0001', '2024-08-02');  
INSERT INTO Factura (PagoId, Numero, Fecha) VALUES (2, 'FAC-0002', '2024-08-06');  
INSERT INTO Factura (PagoId, Numero, Fecha) VALUES (3, 'FAC-0003', '2023-04-21');  
INSERT INTO Factura (PagoId, Numero, Fecha) VALUES (4, 'FAC-0004', '2024-09-02');  
INSERT INTO Factura (PagoId, Numero, Fecha) VALUES (5, 'FAC-0005', '2024-09-11');  
  
-- Historia Médica (AfiliadoId 1..5)  
INSERT INTO HistoriaMedica (AfiliadoId, Diagnostico, Fecha) VALUES (1, 'Hipertensión', '2020-05-10');  
INSERT INTO HistoriaMedica (AfiliadoId, Diagnostico, Fecha) VALUES (2, 'Asma', '2019-11-12');  
INSERT INTO HistoriaMedica (AfiliadoId, Diagnostico, Fecha) VALUES (3, 'Diabetes', '2018-03-22');  
INSERT INTO HistoriaMedica (AfiliadoId, Diagnostico, Fecha) VALUES (4, 'Alergia', '2021-07-30');  
INSERT INTO HistoriaMedica (AfiliadoId, Diagnostico, Fecha) VALUES (5, 'Sin antecedentes', '2022-02-14');  
  
-- AfiliadoDetalle (AfiliadoId 1..5)  
INSERT INTO AfiliadoDetalle (AfiliadoId, TipoDetalle, Valor, FechaRegistro) VALUES (1, 'Teléfono', '3001112222', '2024-01-01');  
INSERT INTO AfiliadoDetalle (AfiliadoId, TipoDetalle, Valor, FechaRegistro) VALUES (2, 'Teléfono', '3002223333', '2024-01-02');  
INSERT INTO AfiliadoDetalle (AfiliadoId, TipoDetalle, Valor, FechaRegistro) VALUES (3, 'CorreoAlterno', 'carlos.alt@mail.com', '2024-01-03');  
INSERT INTO AfiliadoDetalle (AfiliadoId, TipoDetalle, Valor, FechaRegistro) VALUES (4, 'Dirección', 'Cra 45 #12-34', '2024-01-04');  
INSERT INTO AfiliadoDetalle (AfiliadoId, TipoDetalle, Valor, FechaRegistro) VALUES (5, 'ContactoEmergencia', 'Marta 3204445555', '2024-01-05');  
  
-- Citas (AfiliadoId 1..5)  
INSERT INTO Cita (AfiliadoId, Fecha, Hora) VALUES (1, '2024-09-15', '08:00');  
INSERT INTO Cita (AfiliadoId, Fecha, Hora) VALUES (2, '2024-09-16', '09:30');  
INSERT INTO Cita (AfiliadoId, Fecha, Hora) VALUES (3, '2024-09-17', '11:00');  
INSERT INTO Cita (AfiliadoId, Fecha, Hora) VALUES (4, '2024-09-18', '14:00');  
INSERT INTO Cita (AfiliadoId, Fecha, Hora) VALUES (5, '2024-09-19', '16:00');  
  
-- Reportes (IndicadorId 1..5)  
INSERT INTO Reporte (Descripcion, Fecha, IndicadorId) VALUES ('Reporte mensual afiliados', '2024-08-31', 1);  
INSERT INTO Reporte (Descripcion, Fecha, IndicadorId) VALUES ('Reporte contratos vigentes', '2024-08-31', 2);  
INSERT INTO Reporte (Descripcion, Fecha, IndicadorId) VALUES ('Reporte pagos pendientes', '2024-08-31', 3);  
INSERT INTO Reporte (Descripcion, Fecha, IndicadorId) VALUES ('Reporte citas dia', '2024-09-15', 4);  
INSERT INTO Reporte (Descripcion, Fecha, IndicadorId) VALUES ('Reporte historias medicas', '2024-08-31', 5);

**4. Pruebas Unitarias por Entidad (xUnit)**

Se propone la siguiente estructura de pruebas unitarias usando xUnit y un patrón Repository para cada entidad. Se incluye un ejemplo completo (Afiliado) y plantillas para las demás entidades. Asegúrate de crear dos librerías: (1) Data (conexion), (2) Models/Repositories (modelos y CRUD).

// Ejemplo: Tests para Afiliado (xUnit)  
public class AfiliadoTests  
{  
 [Fact]  
 public void CrearAfiliado\_DeberiaInsertarEnBaseDatos()  
 {  
 var repo = new AfiliadoRepository(); // implementar en capa de repositorios  
 var afiliado = new Afiliado { Nombre = "Test Uno", Documento = "T1001", Email = "test1@mail.com", FechaNacimiento = DateTime.Parse("1990-01-01"), MunicipioId=1, CategoriaId=1 };  
 var resultado = repo.Crear(afiliado);  
 Assert.True(resultado > 0); // devuelve Id insertado  
 }  
  
 [Fact]  
 public void ObtenerAfiliado\_DeberiaRetornarDatos()  
 {  
 var repo = new AfiliadoRepository();  
 var afiliado = repo.ObtenerPorDocumento("T1001");  
 Assert.NotNull(afiliado);  
 Assert.Equal("Test Uno", afiliado.Nombre);  
 }  
  
 [Fact]  
 public void ActualizarAfiliado\_DeberiaCambiarEmail()  
 {  
 var repo = new AfiliadoRepository();  
 var afiliado = repo.ObtenerPorDocumento("T1001");  
 afiliado.Email = "nuevo@mail.com";  
 var actualizado = repo.Actualizar(afiliado);  
 Assert.True(actualizado);  
 }  
  
 [Fact]  
 public void EliminarAfiliado\_DeberiaRemoverRegistro()  
 {  
 var repo = new AfiliadoRepository();  
 var afiliado = repo.ObtenerPorDocumento("T1001");  
 var eliminado = repo.Eliminar(afiliado.Id);  
 Assert.True(eliminado);  
 }  
}

Plantillas para las demás entidades (crear/obtener/actualizar/eliminar):

// Tests para Departamento

public void Departamento\_CrearLeerActualizarEliminar() { /\* Usar patrón repository para Departamento \*/ }

// Tests para Municipio

public void Municipio\_CrearLeerActualizarEliminar() { /\* Usar patrón repository para Municipio \*/ }

// Tests para Categoria

public void Categoria\_CrearLeerActualizarEliminar() { /\* Usar patrón repository para Categoria \*/ }

// Tests para Rol

public void Rol\_CrearLeerActualizarEliminar() { /\* Usar patrón repository para Rol \*/ }

// Tests para PlanEPS

public void PlanEPS\_CrearLeerActualizarEliminar() { /\* Usar patrón repository para PlanEPS \*/ }

// Tests para Usuario

public void Usuario\_CrearLeerActualizarEliminar() { /\* Usar patrón repository para Usuario \*/ }

// Tests para Cita

public void Cita\_CrearLeerActualizarEliminar() { /\* Usar patrón repository para Cita \*/ }

// Tests para AfiliadoDetalle

public void AfiliadoDetalle\_CrearLeerActualizarEliminar() { /\* Usar patrón repository para AfiliadoDetalle \*/ }

// Tests para Contrato

public void Contrato\_CrearLeerActualizarEliminar() { /\* Usar patrón repository para Contrato \*/ }

// Tests para Pago

public void Pago\_CrearLeerActualizarEliminar() { /\* Usar patrón repository para Pago \*/ }

// Tests para Factura

public void Factura\_CrearLeerActualizarEliminar() { /\* Usar patrón repository para Factura \*/ }

// Tests para HistoriaMedica

public void HistoriaMedica\_CrearLeerActualizarEliminar() { /\* Usar patrón repository para HistoriaMedica \*/ }

// Tests para Indicador

public void Indicador\_CrearLeerActualizarEliminar() { /\* Usar patrón repository para Indicador \*/ }

// Tests para Reporte

public void Reporte\_CrearLeerActualizarEliminar() { /\* Usar patrón repository para Reporte \*/ }

**5. Pruebas de la capa de negocio (Business logic)**

Ejemplo de prueba de negocio: Obtener afiliados con contratos activos y pagos al día.

[Fact]  
public void ObtenerAfiliados\_ConContratosActivos\_DeberiaRetornarLista()  
{  
 var service = new AfiliadoService(new AfiliadoRepository(), new ContratoRepository(), new PagoRepository());  
 var resultado = service.ObtenerAfiliadosConContratosActivos();  
 Assert.NotEmpty(resultado); // Dependiendo de los datos de prueba  
}

**6. Consulta de Negocio (SQL)**

Ejemplo: Obtener todos los afiliados con contratos activos y pagos al día:

SELECT a.Nombre, a.Documento, c.FechaInicio, c.FechaFin, p.Valor, p.Fecha  
FROM Afiliado a  
INNER JOIN Contrato c ON a.Id = c.AfiliadoId  
INNER JOIN Pago p ON c.Id = p.ContratoId  
WHERE (c.FechaFin IS NULL OR c.FechaFin > GETDATE());

**7. Commits del Proyecto (evidencia)**

Se recomienda incluir pantallazos del historial de commits en GitHub. Si tienes un PDF con las imágenes de commits (Proyecto\_EPS\_Actualizado.pdf), abre ese PDF, captura las pantallas relevantes y pégalas en esta sección. A continuación se indica el archivo original entregado:

Archivo con pantallazos: /mnt/data/Proyecto\_EPS\_Actualizado.pdf (abrir y copiar las imágenes aquí)

Incluir al menos un commit por cada integrante (pantallazo que muestre autor y mensaje).

**8. Checklist de la Guía del Profesor (MR)**

|  |  |  |
| --- | --- | --- |
| Item guia | Descripción | MR (Si/No/Pendiente) |
| Portada | Portada con tema, nombres, CC, grupo y repositorio | SI |
| Modelo de BD | Diagrama corregido + diagrama inicial con nota aclaratoria | SI |
| 15 Tablas | Estructura con 15 tablas relacionadas | SI |
| Tabla de datos | Mínimo 5 registros por tabla (INSERTS incluidos) | SI |
| Pruebas unitarias | Ejemplos y plantillas xUnit por entidad | SI |
| Capa negocio | Pruebas unitarias para la capa de negocio | SI |
| Commits | Pantallazos por integrante (evidencia) | PENDIENTE - insertar imágenes |
| Repositorio | Código en repo (link incluido en portada) | SI |

**9. Instrucciones para ejecución y entrega**

1) Crear carpeta de backup en PowerShell (si no existe):  
 New-Item -ItemType Directory -Force -Path 'C:\BACKUPSINCORREGIR'

2) Ejecutar el BACKUP (si aplica) y luego correr el script SQL en el orden indicado (usar SQL Server Management Studio).

3) Ejecutar los INSERTS para cargar datos de prueba (5 por tabla).

4) Ejecutar pruebas unitarias (xUnit) en las dos librerías: Data (conexion) y Models/Repositories (modelos).

5) Agregar pantallazos de commits en la sección 7 antes de exportar a PDF para entrega final.