Estela Sarai Renderos Guzman studt.estela@gmail.com

Estado de cuenta tarjeta de crédito

El presente proyecto tiene el objetivó de realizar funcionalidades básicas dentro de un sistema bancario para transacciones de cuenta de crédito de las cuales incluye las siguientes:

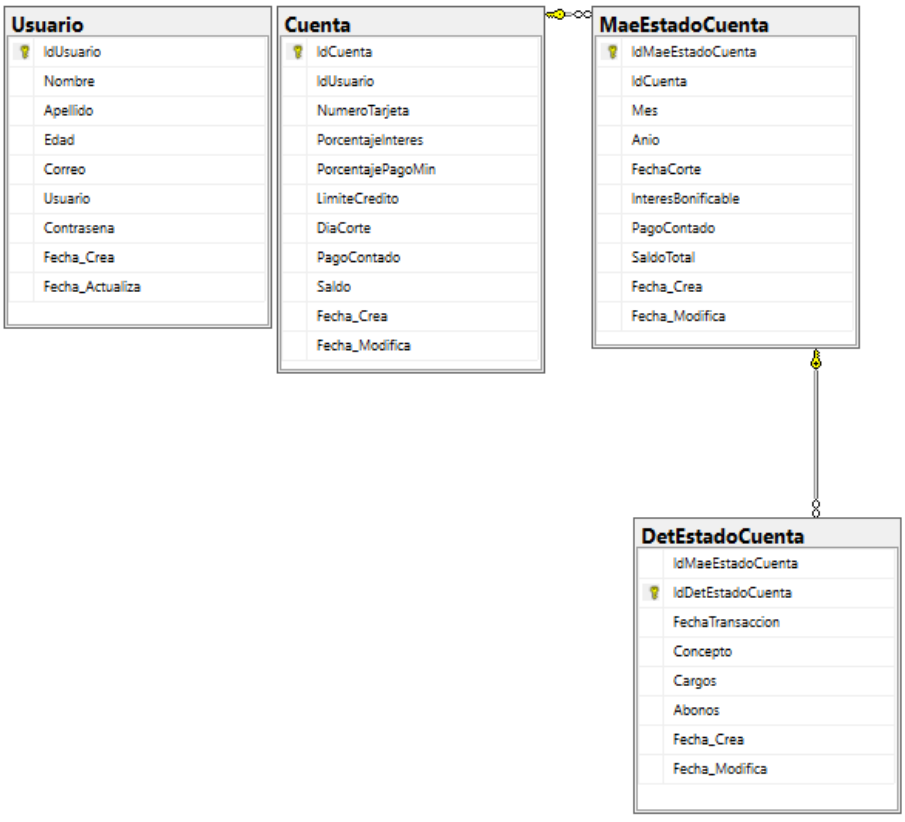
- Aperturas de cuentas
- Realizar pagos
- Realizar compras
- Consultar transacciones

Arquitectura de Desarrollo

Base de datos

Herramienta	Versión
SQL Server	2019
MSSMS	18.9.1
SQL Source Control	

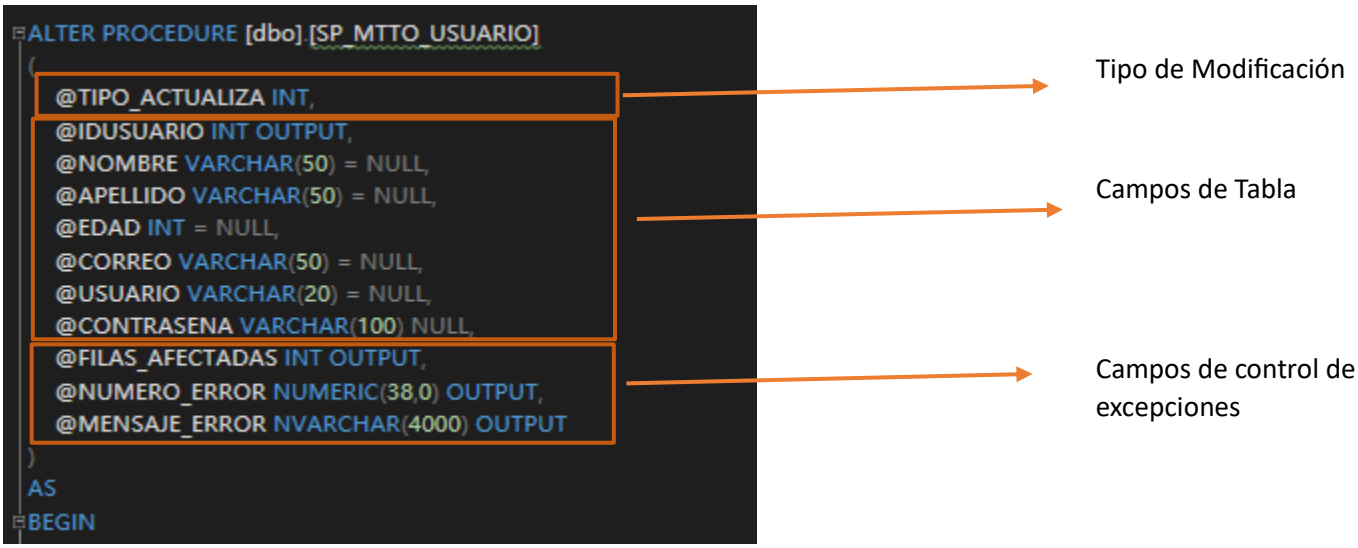
Diagrama



Procedimientos Almacenados

Se realizo un procedimiento para el mantenimiento de cada una de las tablas, el cual tiene la siguiente estructura:

Los parámetros se dividen en tres partes, el primero es el que determina que acción se realizara (Agregar, Actualizar o Eliminar), los últimos 3 parámetros son los que nos permitirán controlar una excepción y saber el número de filas que fueron afectadas y los parámetros de en medio se tendrán los campos a modificar o bien aquellos que nos permitan hacerlo.




```
BEGIN TRY
   IF @TIPO_ACTUALIZA=1 --Adicionar
   BEGIN
      SELECT @IDUSUARIO=ISNULL(MAX(IdUsuario),0)+1
      FROM Usuario

      INSERT INTO Usuario([IdUsuario], [Nombre], [Apellido], [Edad], [Correo], [Usuario], [Contrasena], [Fecha_Crea] [Fecha_Actualiza])
       VALUES( @IDUSUARIO, @NOMBRE, @APELLIDO, @EDAD, @CORREO, @USUARIO, @CONTRASENA , GETDATE(),GETDATE())

   END ELSE
   IF @TIPO_ACTUALIZA=2 --Actualizar
   BEGIN
   UPDATE Usuario SET
   Nombre = @NOMBRE,
   Apellido = @APELLIDO,
   Edad = @EDAD,
   Correo = @CORREO,
   Usuario = @USUARIO,
   Contrasena = @CONTRASENA,
   Fecha_Actualiza = GETDATE()
   WHERE IdUsuario = @IDUSUARIO

   END ELSE
   IF @TIPO_ACTUALIZA=3 --Eliminar
   BEGIN
      DELETE FROM Usuario
      WHERE IdUsuario = @IDUSUARIO
   END

   SELECT @FILAS_AFECTADAS=@@ROWCOUNT

END TRY

BEGIN CATCH

   SELECT @NUMERO_ERROR=ERROR_NUMBER(), @MENSAJE_ERROR=ERROR_MESSAGE(), @FILAS_AFECTADAS = 0
END CATCH;
```

Se realizo un procedimiento que permitirá devolver las consulta en base al tipo seleccionado en un combo dentro de la aplicación.

```
ALTER PROCEDURE [dbo].[SP_DATA_ESTADO_CUENTA]
(
    @TIPO_CONSULTA INT,
    @IDUSUARIO INT,
    @IDCUENTA INT OUTPUT,
    @MES INT = NULL
)
AS
BEGIN
```

```
IF @TIPO_CONSULTA=1 -- Estado de Cuentas Del Usuario
BEGIN
        SELECT DISTINCT
                A.Nombre +' ' + A.Apellido AS Nombre,
                A.Usuario,
                A.Correo,
                B.Fecha_Crea AS FechaApertura,
                B.NumeroTarjeta AS NumeroTarjeta,
                B.LimiteCredito,
                B.PagoContado * B.PorcentajePagoMin AS PagoMinimo,
                B.LimiteCredito - B.PagoContado AS SaldoDisponible,
                B.PagoContado * B.PorcentajeInteres AS InteresBonificable,
                B.PagoContado,
                B.Saldo AS SaldoActual
        FROM Usuario A WITH(NOLOCK)
        INNER JOIN Cuenta B WITH(NOLOCK) ON B.IdUsuario = A.IdUsuario
        LEFT JOIN MaeEstadoCuenta C WITH(NOLOCK) ON C.IdCuenta = B.IdCuenta
        LEFT JOIN DetEstadoCuenta D WITH(NOLOCK) ON D.IdMaeEstadoCuenta = C.IdMaeEstadoCuenta
        WHERE A.IdUsuario = @IDUSUARIO

END ELSE
```

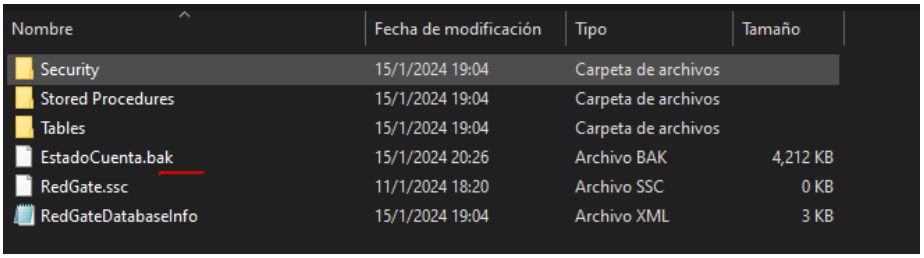
```
IF @TIPO_CONSULTA=2 -- Encabezado de estado de cuenta por mes
BEGIN
	SELECT DISTINCT
		D.Nombre +' ' + D.Apellido AS Nombre,
		D.Usuario,
		D.Correo,
		C.Fecha_Crea AS FechaApertura,
		A.FechaCorte,
		C.NumeroTarjeta AS NumeroTarjeta,
		C.LimiteCredito,
		C.PagoContado * C.PorcentajePagoMin AS PagoMinimo,
		C.LimiteCredito - C.PagoContado AS SaldoDisponible,
		A.InteresBonificable, -- MES
		A.PagoContado,
		A.SaldoTotal AS SaldoActual -- MES
	FROM MaeEstadoCuenta A WITH(NOLOCK)
	INNER JOIN DetEstadoCuenta B WITH(NOLOCK) ON A.IdMaeEstadoCuenta = B.IdMaeEstadoCuenta
	INNER JOIN Cuenta C WITH(NOLOCK) ON A.IdCuenta = C.IdCuenta
	INNER JOIN Usuario D WITH(NOLOCK) ON C.IdUsuario = D.IdUsuario
	WHERE C.IdCuenta = @IDCUENTA
	AND A.Mes = @MES

END ELSE
IF @TIPO_CONSULTA=3 -- Detalle de compras de la cuenta segun el mes
BEGIN
	SELECT
		B.FechaTransaccion,
		B.Concepto,
		ISNULL(B.Cargos,0) Cargos,
		ISNULL(B.Abonos,0) Abonos
	FROM MaeEstadoCuenta A WITH(NOLOCK)
	INNER JOIN DetEstadoCuenta B WITH(NOLOCK) ON A.IdMaeEstadoCuenta = B.IdMaeEstadoCuenta
	INNER JOIN Cuenta C WITH(NOLOCK) ON A.IdCuenta = C.IdCuenta
	WHERE C.IdCuenta = @IDCUENTA
	AND A.Mes = @IDCUENTA
END ELSE
SELECT 'No existe la Consulta'
```

Triggers

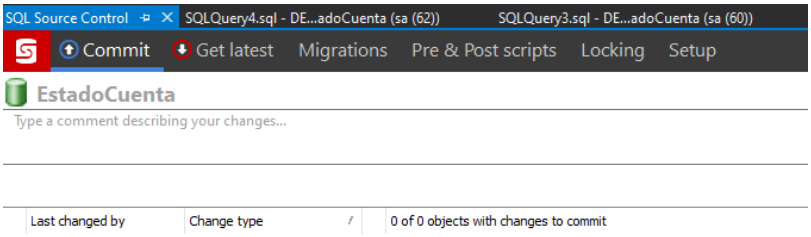
Se crearon tres Triggers para la tabla de DetEstadoCuenta, para el control de las transacciones, ya sea cuando se haga una compra o un pago, se hará la actualización de los montos del MaeEstadoCuenta y Cuenta al que corresponde.

```
ALTER TRIGGER [dbo] [ADD_ESTADO_CUENTA] ON [dbo] [DetEstadoCuenta] AFTER INSERT
AS
	DECLARE @IDMAE_ESTADO_CUENTA INT, @IDCUENTA INT, @INTERES_BONIFICABLE DECIMAL(18,4), @PAGO_CONTADO DECIMAL(14,4), @SALDO_TOTAL DECIMAL(18,4)
	BEGIN
		SELECT @IDMAE_ESTADO_CUENTA = IdMaeEstadoCuenta FROM Inserted
		SELECT
			@IDCUENTA = C.IdCuenta,
			@PAGO_CONTADO = SUM(A.Cargos) - SUM(A.Abonos), -- Sub total sin intereses del mes / Pago al contado
			@INTERES_BONIFICABLE = ((SUM(A.Cargos) - SUM(A.Abonos)) * C.PorcentajeInteres), -- Interes Bonificable
			@SALDO_TOTAL = (SUM(A.Cargos) - SUM(A.Abonos)) + ((SUM(A.Cargos) - SUM(A.Abonos)) * C.PorcentajeInteres)
		FROM dbo.DetEstadoCuenta A
		INNER JOIN MaeEstadoCuenta B WITH(NOLOCK) ON A.IdMaeEstadoCuenta = B.IdMaeEstadoCuenta
		INNER JOIN Cuenta C WITH(NOLOCK) ON B.IdCuenta = C.IdCuenta
		WHERE B.IdMaeEstadoCuenta = @IDMAE_ESTADO_CUENTA
		GROUP BY A.IdMaeEstadoCuenta,
			C.IdCuenta,
			C.PorcentajeInteres

		UPDATE MaeEstadoCuenta SET
		InteresBonificable = @INTERES_BONIFICABLE,
		PagoContado = @PAGO_CONTADO,
		SaldoTotal = @SALDO_TOTAL,
		Fecha_Modifica = GETDATE()
		WHERE IdMaeEstadoCuenta = @IDMAE_ESTADO_CUENTA
		AND IdCuenta = @IDCUENTA

		SELECT
		@SALDO_TOTAL = SUM(SaldoTotal),
		@PAGO_CONTADO = SUM(PagoContado)
		FROM MaeEstadoCuenta WITH(NOLOCK)
		WHERE IdCuenta = @IDCUENTA

		UPDATE Cuenta SET Saldo = @SALDO_TOTAL, PagoContado = @PAGO_CONTADO ,Fecha_Modifica = GETDATE()
		WHERE IdCuenta = @IDCUENTA
	END
```

El Backup se encuentro dentro de la carpeta EstadoCuentaDB



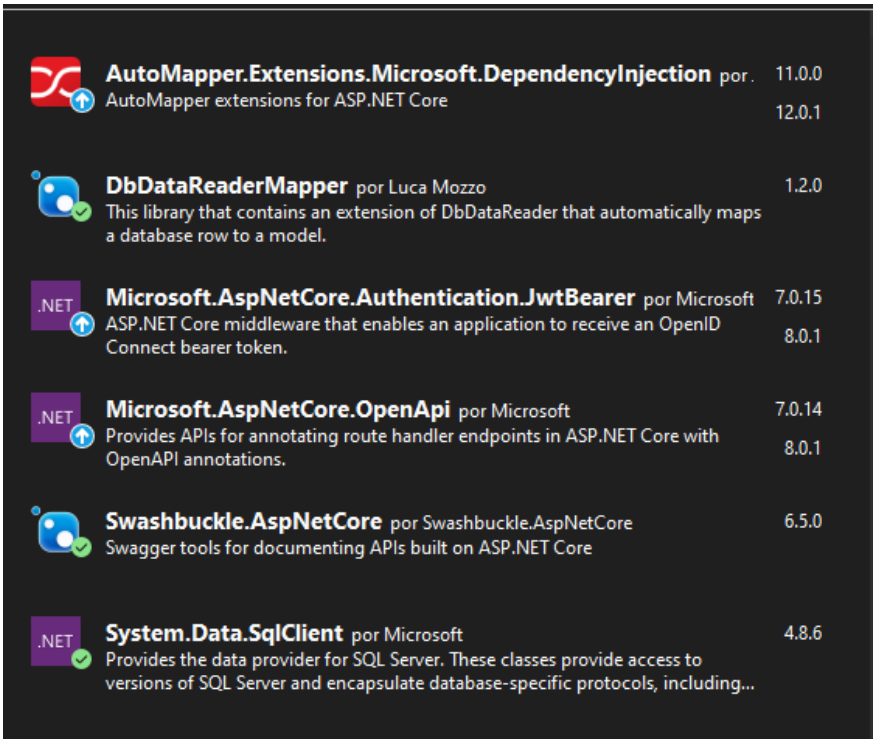
Para llevar el seguimiento de los cambios con Git dentro de la base se utilizó SQL Source Control



API Rest

Herramienta	Versión
Visual Studio	2022
.Net	7
Asp.net	

Extensiones



Archivos Compartidos

Respuesta: nos permite controlar las respuestas a las consultas, creando campos para control de excepciones y devolución de objectos, las filas posibles afectadas y si la excusión del procedimiento fue exitosa.

```
namespace Cuentas.Options.Shared
{
    85 referencias
    public class Respuesta
    {
        35 referencias
        public decimal ErrorCode { get; set; }
        23 referencias
        public string ErrorMessage { get; set; }
        19 referencias
        public string ErrorSource { get; set; }
        25 referencias
        public bool Result { get; set; }
        23 referencias
        public object CodeHelper { get; set; }
        18 referencias
        public int RowsAffected { get; set; }
        24 referencias
        public object Data { get; set; }

        10 referencias
        public Respuesta()
        {
            ErrorCode = 0;
            ErrorSource = "";
            ErrorMessage = "";
            RowsAffected = 0;
            CodeHelper = 0;
            Result = false;
        }
    }
}
```

Parámetro: se utiliza como objecto genérico para agregar los parámetros a los procedimientos almacenados

```
namespace Cuentas.Options.Shared
{
    73 referencias
    public class Parametro
    {
        61 referencias
        public string ParameterName { get; set; }
        56 referencias
        public object Value { get; set; }
        56 referencias
        public System.Data.DbType DbType { get; set; }
        19 referencias
        public System.Data.ParameterDirection Direction { get; set; } = System.Data.ParameterDirection.Input;
        7 referencias
        public int Size { get; set; }
    }
}
```

ControlData: Clase genérica que nos permitirá realizar consultas y ejecuciones a la DB, devolviendo un Objecto Respuesta hacia los repositorios de las tablas.

```
4 referencias
public async Task<object> ExecCmd(CommandType xCommandType,
                                  string xQry,
                                  bool xRowsAfected = true,
                                  List<Parametro> xParametros = null)
{
    object sqlrows = 0;

    try
    {
        if (objConnection.State != ConnectionState.Open)
            await objConnection.OpenAsync();

        objCommand = dataFactory.CreateCommand();
        objCommand.Connection = objConnection;
        objCommand.CommandType = xCommandType;
        objCommand.CommandText = xQry;
        objCommand.CommandTimeout = 0;

        if (xParametros != null)
        {
            foreach (Parametro p in xParametros)
            {
                if (p.ParameterName != "")
                {
                    objParameter = dataFactory.CreateParameter();
                    objParameter.DbType = p.DbType;
                    objParameter.ParameterName = p.ParameterName;
                    objParameter.Value = p.Value;
                    objParameter.Direction = p.Direction;
                    objParameter.Size = p.Size;

                    objCommand.Parameters.Add(objParameter);
                }
            }
        }

        if (xRowsAfected)
        {
            sqlrows = await objCommand.ExecuteNonQueryAsync();
        }
        else
        {
            sqlrows = await objCommand.ExecuteScalarAsync();
        }
    }
    catch (System.Exception)
    {
        throw;
    }
    finally
    {
        objConnection.Close();
    }
```

BaseRepository : Es una esquema genérico de repositorios para utilizar la clase ControlData

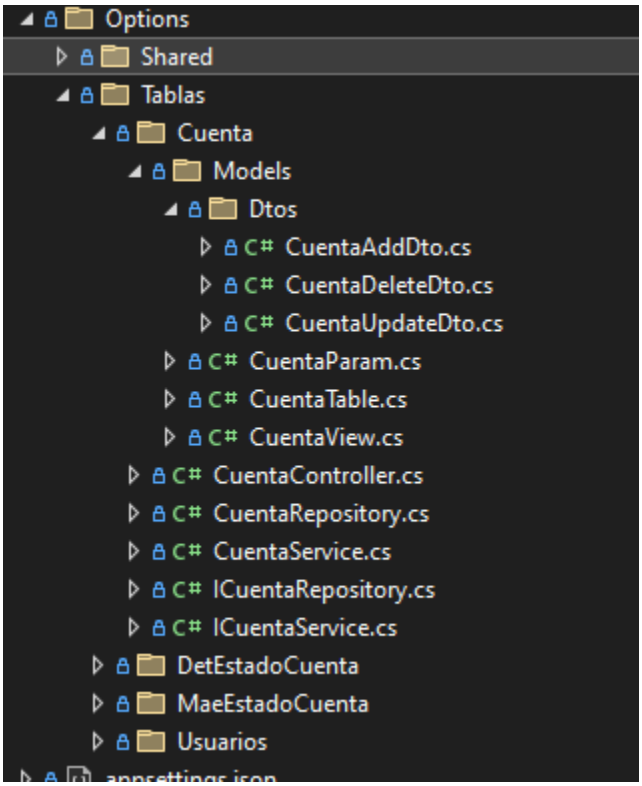
```
using AutoMapper;

namespace Cuentas.Options.Shared
{
    9 referencias
    public abstract class BaseRepository<TEntity> where TEntity : BaseEntity
    {
        public ControlData objData;
        public IMapper _mapper;
        4 referencias
        protected BaseRepository(string connectionString, string providerName)
        {
            objData = new ControlData(connectionString, providerName);
        }
    }
}
```

ApplicationMapper: Clase control de mappeo para las tablas a Dtos.

```
namespace Cuentas.Options.Shared
{
    1 referencia
    public class ApplicationMapper : Profile
    {
        0 referencias
        public ApplicationMapper() {

            CreateMap<CuentaTable,CuentaAddDto>().ReverseMap();
            CreateMap<CuentaTable,CuentaUpdateDto>().ReverseMap();
            CreateMap<CuentaTable, CuentaDeleteDto>().ReverseMap();

            CreateMap<UsuarioTable, UsuarioAddDto>().ReverseMap();
            CreateMap<UsuarioTable, UsuarioUpdateDto>().ReverseMap();
            CreateMap<UsuarioTable, UsuarioDeleteDto>().ReverseMap();

            CreateMap<MaeEstadoCuentaTable, MaeEstadoCuentaAddDto>().ReverseMap();
            CreateMap<MaeEstadoCuentaTable, MaeEstadoCuentaUpdateDto>().ReverseMap();
            CreateMap<MaeEstadoCuentaTable, MaeEstadoCuentaDeleteDto>().ReverseMap();

            CreateMap<DetEstadoCuentaTable, DetEstadoCuentaAddDto>().ReverseMap();
            CreateMap<DetEstadoCuentaTable, DetEstadoCuentaUpdateDto>().ReverseMap();
            CreateMap<DetEstadoCuentaTable, DetEstadoCuentaDeleteDto>().ReverseMap();
        }
    }
}
```

Tablas

Para este proyecto se hizo uso del patrón de diseño repositorio, Dtos y Automapper, los que nos permitirán llevar un mayor control y fácil manejo de nuestra información.



Los controladores son los que nos permiten controlar las peticiones Https y enviar información a través de las mismas

```
[HttpPost]
[ProducesResponseType(StatusCodes.Status401Unauthorized)]
0 referencias
public async Task<IActionResult> Post(CuentaAddDto Data)
{
    var resultado = await _service.CreateAsync(Data);
    if (resultado.ErrorCode == 0)
    {
        return StatusCode(201, resultado);
    }
    else
    {
        return BadRequest(resultado);
    }
}
```

Los servicios tienen la función desglosar los parámetros y hacer el mapeo de DTOs a Tablas correspondientes

```
namespace Cuentas.Services
{
    2 referencias
    public class CuentaService : ICuentaService
    {
        public readonly ICuentaRepository _repo;
        public readonly IMapper _mapper;
        0 referencias
        public CuentaService(ICuentaRepository repo, IMapper mapper)
        {
            _repo = repo;
            _mapper = mapper;
        }

        2 referencias
        public async Task<Respuesta> GetDataAsync(CuentaParam Parametros)
        {
            var parametro = new List<Parametro>();
            parametro.Add(new Parametro() { ParameterName = "TIPO_CONSULTA", Value = 1, DbType = System.Data.DbType.Int32 });
            parametro.Add(new Parametro() { ParameterName = "IDUSUARIO", Value = Parametros.IdUsuario, DbType = System.Data.DbType.Int32 });
            parametro.Add(new Parametro() { ParameterName = "IDCUENTA", Value = Parametros.IdCuenta, DbType = System.Data.DbType.Int32 });
            parametro.Add(new Parametro() { ParameterName = "MES", Value = Parametros.Mes, DbType = System.Data.DbType.Int32 });
            return await _repo.GetDataAsync(parametro);
        }
        2 referencias
        public async Task<Respuesta> CreateAsync(CuentaAddDto Data)
        {
            var CuentaTable = _mapper.Map<CuentaTable>(Data);
            return await _repo.CreateAsync(CuentaTable);
        }

        2 referencias
        public async Task<Respuesta> UpdateAsync(CuentaUpdateDto Data)
        {
            var CuentaTable = _mapper.Map<CuentaTable>(Data);
            return await _repo.UpdateAsync(CuentaTable);
        }
        2 referencias
        public async Task<Respuesta> DeleteAsync(CuentaDeleteDto Data)
        {
            var CuentaTable = _mapper.Map<CuentaTable>(Data);
            return await _repo.DeleteAsync(CuentaTable);
        }
    }
```

En los repositorios se hará la petición a la base de datos por medio de la clase ControlData y realizar la correcta conversión de vista o respuesta, a los servicios.

```
public async Task<Respuesta>  GetDataAsync(List<Parametro> Parametros)
{
    Respuesta objResultado = new Respuesta();

    try
    {
        var reader = await objData.GetDataReader( CommandType.StoredProcedure, "SP_DATA_ESTADO_CUENTA", Parametros);

        List<CuentaView> ListCuenta = new List<CuentaView>();
        while (await reader.ReadAsync())
        {
            var response = reader.MapToObject<CuentaView>();
            if (response != null)
            {
                ListCuenta.Add(response);
            }
        }

        reader.Close();
        reader = null;
        objResultado.Data = ListCuenta;
        objResultado.Result = true;
        objResultado.RowsAffected = ListCuenta.Count;
        objResultado.CodeHelper = 0;
        objResultado.ErrorCode = 0;
        objResultado.ErrorMessage = "";
        objResultado.ErrorSource = "";

    }
    catch (System.Exception e)
    {
        objResultado.Data = null;
        objResultado.Result = false;
        objResultado.CodeHelper = 0;
        objResultado.ErrorCode = -1;
        objResultado.ErrorMessage = e.Message;
        objResultado.ErrorSource += $"[{e.Source}]";
    }
    finally
    {
        objData.objConnection.Close();
    }

    return objResultado;
}
```

Program.cs | UsuarioService.cs | CuentaService.cs | CuentaRepository.cs | CuentaController.cs

Cuentas

```
1  using Microsoft.AspNetCore.Authentication.JwtBearer;
2  using Microsoft.IdentityModel.Tokens;
3  using System.Text;
4  using AutoMapper;
5  using Cuentas.Repositories;
6  using Cuentas.Services;
7  using Microsoft.Extensions.Options;
8  using Microsoft.OpenApi.Models;
9
10 var builder = WebApplication.CreateBuilder(args);
11
12 #region "Repositorios"
13
14 builder.Services.AddScoped<ICuentaRepository, CuentaRepository>();
15 builder.Services.AddScoped<IUsuarioRepository, UsuarioRepository>();
16 builder.Services.AddScoped<IMaeEstadoCuentaRepository, MaeEstadoCuentaRepository>();
17 builder.Services.AddScoped<IDetEstadoCuentaRepository, DetEstadoCuentaRepository>();
18
19 #endregion
20
21 #region "Servicios"
22
23 builder.Services.AddScoped<ICuentaService, CuentaService>();
24 builder.Services.AddScoped<IUsuarioService, UsuarioService>();
25 builder.Services.AddScoped<IMaeEstadoCuentaService, MaeEstadoCuentaService>();
26 builder.Services.AddScoped<IDetEstadoCuentaService, DetEstadoCuentaService>();
27 #endregion
28
```

Se realizo la autorización por medio de JWT y autenticación por medio la encriptación MD5

```
1 referencia
private string Encrypt(string contrasena) {

    MD5CryptoServiceProvider x = new MD5CryptoServiceProvider();
    byte[] data = System.Text.Encoding.UTF8.GetBytes(contrasena);
    data = x.ComputeHash(data);
    string resp = "";
    for (int i = 0; i < data.Length; i++)
        resp += data[i].ToString("x2").ToLower();
    return resp;
}
```

```
1 referencia
private string GenerarToken(UsuarioLogin table) {

    var ManejarToken = new JwtSecurityTokenHandler();
    var key = Encoding.ASCII.GetBytes(_token);

    var tokenDescriptor = new SecurityTokenDescriptor
    {
        Subject = new ClaimsIdentity(new Claim[] {
            new Claim(ClaimTypes.Name,table.Usuario)
        }),
        Expires = DateTime.UtcNow.AddDays(1),
        SigningCredentials = new(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)
    };

    var Token = ManejarToken.CreateToken(tokenDescriptor);

    return ManejarToken.WriteToken(Token);
}
```

```
2 referencias
public async Task<Respuesta> LoginAsync(UsuarioLogin Data)
{
    Respuesta Respuesta = new Respuesta();

    List<Parametro> Parametro = new List<Parametro>();
    try
    {
        Parametro.Add(new Parametro() { ParameterName = "Usuario", Value = Data.Usuario, DbType = System.Data.DbType.String });

        Respuesta = await _repo.GetAllDataAsync(Parametro);

        if (Respuesta.Result == false)
        {
            Respuesta.Data = null;
            Respuesta.Result = false;
            Respuesta.RowsAffected = 0;
            Respuesta.CodeHelper = 0;
            Respuesta.ErrorCode = -1;
            Respuesta.ErrorMessage = "El usuario no existe";
            Respuesta.ErrorSource = "Login()";

            return Respuesta;
        }

        var UsuarioRespuesta = (List<UsuarioView>)Respuesta.Data;

        Data.Contrasena = EnCrypt(Data.Contrasena);

        if (Data.Contrasena != UsuarioRespuesta[0].Contrasena)
        {
            Respuesta.Data = null;
            Respuesta.Result = false;
            Respuesta.RowsAffected = 0;
            Respuesta.CodeHelper = 0;
            Respuesta.ErrorCode = -1;
            Respuesta.ErrorMessage = "El usuario o la contraseña son incorectos";
            Respuesta.ErrorSource = "";

            return Respuesta;
        }
```

```
            }

            var token = GenerarToken(Data);
            if (!string.IsNullOrEmpty(token))
            {
                Data.Token = token;
                Respuesta.Data = Data;
                Respuesta.Result = true;
                Respuesta.RowsAffected = 0;
                Respuesta.CodeHelper = 0;
                Respuesta.ErrorCode = 0;
                Respuesta.ErrorMessage = "";
                Respuesta.ErrorSource = "";
            }else {
                Respuesta.Data = Data;
                Respuesta.Result = false;
                Respuesta.RowsAffected = 0;
                Respuesta.CodeHelper = 0;
                Respuesta.ErrorCode = -1;
                Respuesta.ErrorMessage = "No se pudo generar el token";
                Respuesta.ErrorSource = "Login()";
            }

            return Respuesta;

        }
        catch (System.Exception e)
        {
            Respuesta.Data = "";
            Respuesta.Result = false;
            Respuesta.RowsAffected = 0;
            Respuesta.CodeHelper = 0;
            Respuesta.ErrorCode = -1;
            Respuesta.ErrorMessage = e.Message;
            Respuesta.ErrorSource += $"[{e.Source}]";
        }

        return Respuesta;

    }
```

configuración para autorización por medio de SecurityRequirement

```
builder.Services.AddAuthentication(x =>
{
    x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
}).AddJwtBearer(x =>
{
    x.RequireHttpsMetadata = false;
    x.SaveToken = true;
    x.TokenValidationParameters = new TokenValidationParameters
    {
        ValidateIssuerSigningKey = true,
        IssuerSigningKey = new SymmetricSecurityKey(Encoding.ASCII.GetBytes(key)),
        ValidateIssuer = false,
        ValidateAudience = false
    };
});
///// Autenticacion con Bearer
builder.Services.AddSwaggerGen(options =>
{
    options.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme
    {
        Description =
        "Autenticación JWT usando el esquema Bearer. \r\n\r\n " +
        "Ingrese la palabra 'Bearer' seguida de un [espacio] y despues su token en el campo de abajo \r\n\r\n" +
        "Ejemplo: \"Bearer tkdknkdllskd\"",
        Name = "Authorization",
        In = ParameterLocation.Header,
        Scheme = "Bearer"
    });
    options.AddSecurityRequirement(new OpenApiSecurityRequirement()
    {
        {
            new OpenApiSecurityScheme
            {
                Reference = new OpenApiReference
                            {
                                Type = ReferenceType.SecurityScheme,
                                Id = "Bearer"
                            },
                Scheme = "oauth2",
                Name = "Bearer",
                In = ParameterLocation.Header
            },
            new List<string>()
        }
    });
```

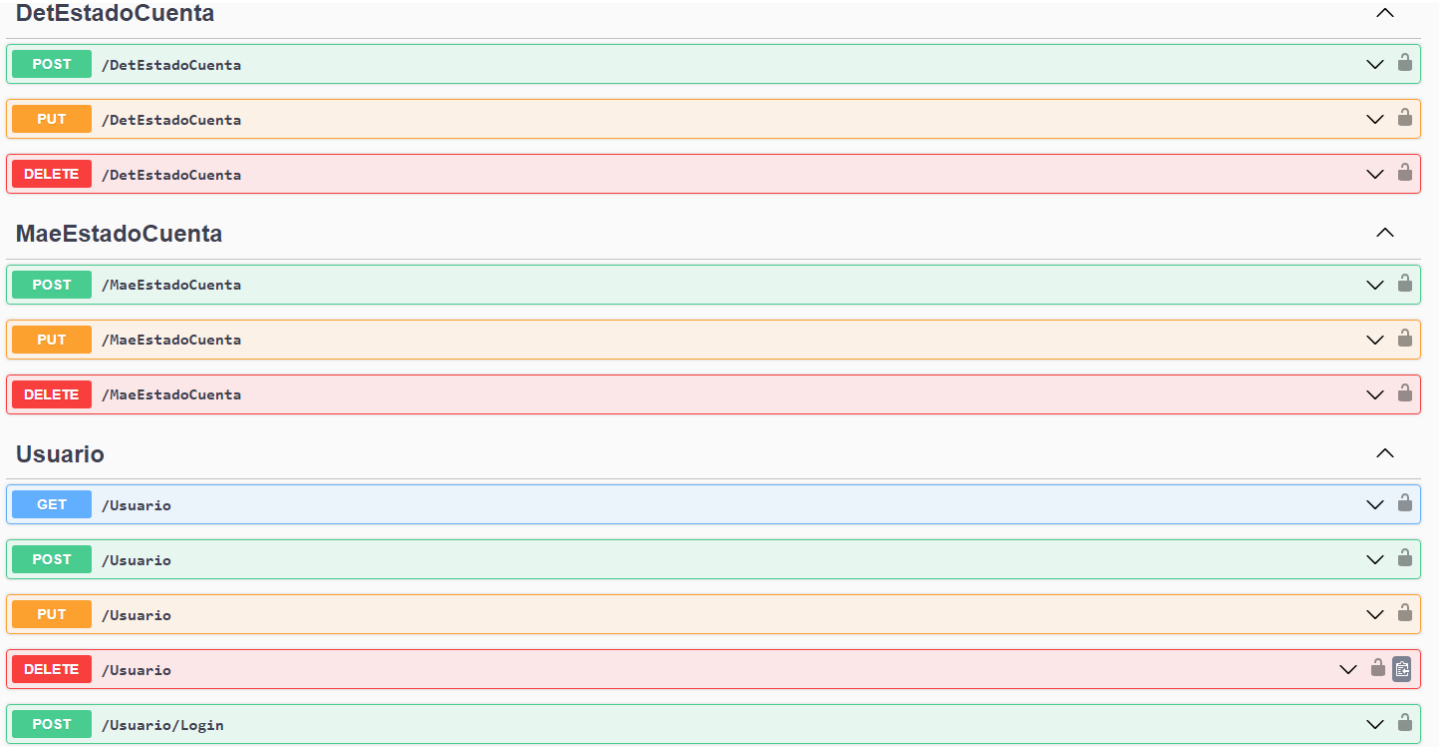
Cuentas 1.0 OAS3

https://localhost:7168/swagger/v1/swagger.json

Authorize

Cuenta

- GET /Cuenta
- POST /Cuenta
- PUT /Cuenta
- DELETE /Cuenta



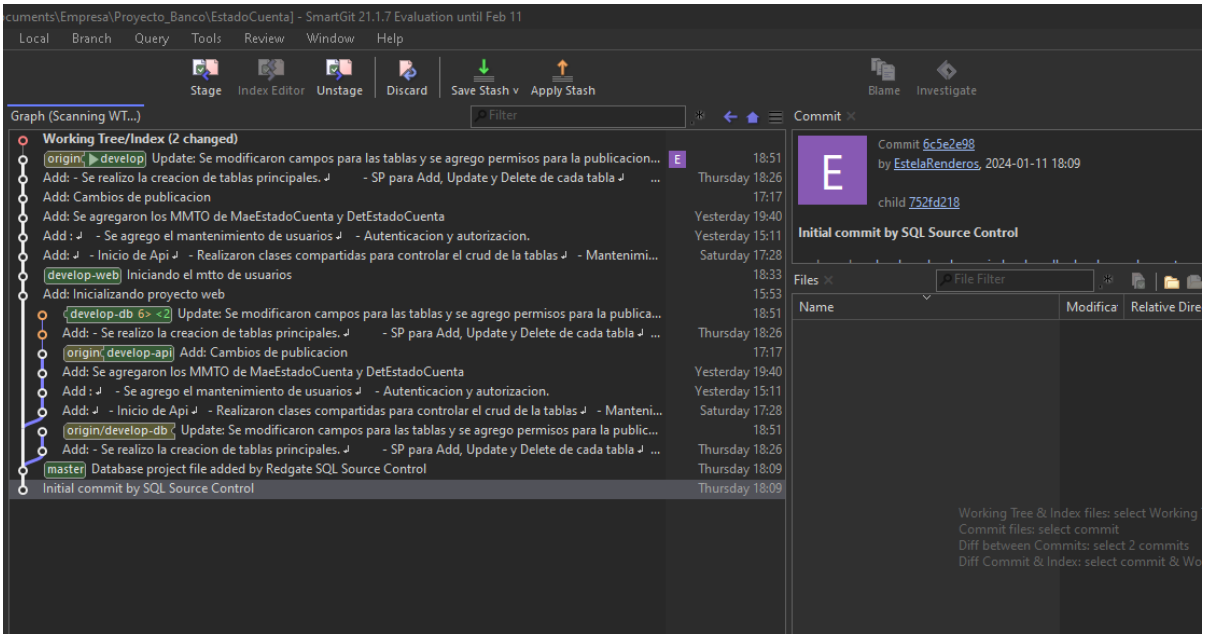
Control de versiones

Herramienta	**versión**
Git	2.39.1
Github	
SmartGit	

Ramas

- ✓ Develop : Principal donde se encuentra todo el proyecto
- ✓ Develop-api: solución Api en ASP .Net
- ✓ Develop-web: Solucion ASP MVC
- ✓ Develop-db: Base de datos en SQL Server

Link de Github : https://github.com/EstelaST/EstadoCuenta/tree/develop


cuments\Empresa\Proyecto_Banco\EstadoCuenta] - SmartGit 21.1.7 Evaluation until Feb 11
Local Branch Query Tools Review Window Help
Stage Index Editor Unstage Discard Save Stash Apply Stash Blame Investigate
Graph (Scanning WT...)
Filter
Working Tree/Index (2 changed)
origin/develop Update: Se modificaron campos para las tablas y se agrego permisos para la publicacion... E 18:51
Add: - Se realizo la creacion de tablas principales. ↲ - SP para Add, Update y Delete de cada tabla ↲ ... Thursday 18:26
Add: Cambios de publicacion 17:17
Add: Se agregaron los MMTO de MaeEstadoCuenta y DetEstadoCuenta Yesterday 19:40
Add : ↲ - Se agrego el mantenimiento de usuarios ↲ - Autenticacion y autorizacion. Yesterday 15:11
Add: ↲ - Inicio de Api ↲ - Realizaron clases compartidas para controlar el crud de la tablas ↲ - Mantenimi... Saturday 17:28
develop-web Iniciando el mtto de usuarios 18:33
Add: Inicializando proyecto web 15:53
develop-db 6> <2 Update: Se modificaron campos para las tablas y se agrego permisos para la publica... 18:51
Add: - Se realizo la creacion de tablas principales. ↲ - SP para Add, Update y Delete de cada tabla ↲ ... Thursday 18:26
origin/develop-api Add: Cambios de publicacion 17:17
Add: Se agregaron los MMTO de MaeEstadoCuenta y DetEstadoCuenta Yesterday 19:40
Add : ↲ - Se agrego el mantenimiento de usuarios ↲ - Autenticacion y autorizacion. Yesterday 15:11
Add: ↲ - Inicio de Api ↲ - Realizaron clases compartidas para controlar el crud de la tablas ↲ - Manteni... Saturday 17:28
origin/develop-db Update: Se modificaron campos para las tablas y se agrego permisos para la public... 18:51
Add: - Se realizo la creacion de tablas principales. ↲ - SP para Add, Update y Delete de cada tabla ↲ ... Thursday 18:26
master Database project file added by Redgate SQL Source Control Thursday 18:09
Initial commit by SQL Source Control Thursday 18:09
Commit
E
Commit 6c5e2e98
by EstelaRenderos, 2024-01-11 18:09
child 752fd218
Initial commit by SQL Source Control
Files
File Filter
Name
Modifica
Relative Dire
Working Tree & Index files: select Working
Commit files: select commit
Diff between Commits: select 2 commits
Diff Commit & Index: select commit & Wo