Api
QualaTest

- Tipo de proyecto: APIRets.
- Tipo de API: Sistema N Capas, REST apoyado en el estándar HTTP.
- **Entidades:**
    1. **Sucursales:**
        a. Codigo
        b. Nombre
        c. Descripcion
        d. Direccion
        e. Identificacion
        f. FechaCreacion
        g. IdMoneda
    2. **Moneda:**
        a. ID
        b. Nombre
- **Tablas base de datos:**
    1. **TBL_SUCURSAL:**
    2. **Moneda:**

- **Conexión a la base de datos:**
    1. "ConnectionStrings": {
    2. "conexionDB": "Data Source= ins-dllo-test-01.public.33e082952ab4.database.windows.net,3342; Initial Catalog= TestDB; User Id=prueba;Password=pruebaconcepto ; Connect Timeout = 30; "
    3. }Conexión con la base de datos “TestDb” por medio del inyectable de la cadena de conexión obtenida del appsetings.json .
    4. Interacción con la base de datos por medio de consultas Stored Procedures.

- **Explicación del funcionamiento de la API:**
    1. Arquitectura Implementada: Se trabajo con la sistema NCapas, que consta de una solución con cuatro proyectos divididos para optimización de la API y mejorar la estabilidad de la misma.

2. Capa Datos: tiene dos proyectos, AppContexto que consta de dos subcarpetas. La subcarpeta Interfaces: donde se declaran las firmas de los métodos, producto, factura y detalles factura individualmente, la subcarpeta Repositorio contiene las implementaciones de las firmas de la subcarpeta AppContexto , es donde se realizan las peticiones por medio de la cadena inyectada desde archivo appsetings.json (referenciado), las consultas se realizan a través de procedimientos almacenados creados anteriormente en la base de datos con la ayuda del paquete Dapper para mapear los parámetros a la consulta.
   En la capa Datos se encuentra un proyecto classLib con el nombre de entidades donde se crearon las entidades de la API, DetalleFactura, Factura y Producto.

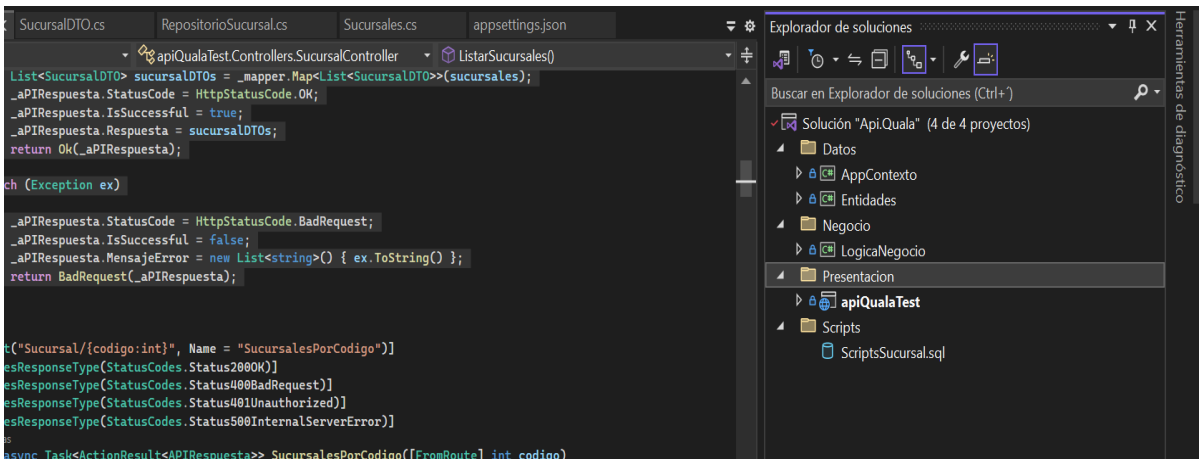
3. La capa Negocio contiene toda la lógica de negocio con el cual se cumple con la separación cliente servidor, consta de dos subclases, Interfaces y Negocio que están dentro de un proyecto Classlib llamado LogicaNegocio.

4. La capa presentación contiene el proyecto WebApi con tres controladores para la separación de responsabilidades que se encargara de la comunicación con la aplicación del frontend y la lógica del backend , SucursalController

```
[HttpGet("ListarSucursales")]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status401Unauthorized)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
public async Task<ActionResult<APIRespuesta>> ListarSucursales()
{
    try
    {
        List<Sucursales> sucursales = await _negocioSucursal.ListarSucursales();
        if (sucursales == null)
        {
            _aPIRespuesta.StatusCode = HttpStatusCode.BadRequest;
            _aPIRespuesta.IsSuccessful = false;
            _aPIRespuesta.MensajeError = new List<string> { "BD no retorno data" };
            return BadRequest(_aPIRespuesta);
        }
        List<SucursalDTO> sucursalDTOs = _mapper.Map<List<SucursalDTO>>(sucursales);
        _aPIRespuesta.StatusCode = HttpStatusCode.OK;
        _aPIRespuesta.IsSuccessful = true;
        _aPIRespuesta.Respuesta = sucursalDTOs;
        return Ok(_aPIRespuesta);
    }
    catch (Exception ex)
    {
        _aPIRespuesta.StatusCode = HttpStatusCode.BadRequest;
        _aPIRespuesta.IsSuccessful = false;
        _aPIRespuesta.MensajeError = new List<string>() { ex.ToString() };
        return BadRequest(_aPIRespuesta);
    }
```

5. En la clase starup en la configuración de servicios se trabajó con el AddSingleton para permitir la inyección por medio de dependencias en clases en las diferentes

capas de la API.



6.

- Paquetes usados en la API:
  1. Dapper v2.0.123
  2. Swashbuckle.AspNetCore v6.2.3
  3. Newtonsoft.Json v13.01
  4. System.Runtime v4.3.1
  5. Microsoft.Extensions.Configuration.Abstractions v5.0.0

- Swagger ,Link GitHub y YouTube:
  1. https://github.com/JoanAndresRG/ApiQualaTest.git GitHub Backend