



Tema

EUREKA BANK SOAP JAVA

**Lindsay Domenique Barrionuevo Ordoñez, Joel
Alessandro Rivera López, Leonardo Javier
Yaranga Suquillo**

Tutor

Ing. Eduardo Mauricio Campaña Ortega

MIS.MDU.CCNA.CCIA.

PhD. (c) Ingeniería de Software

PhD. (c) Seguridad Información

Fecha

03/12/2025

ÍNDICE DE CONTENIDOS

1. INTRODUCCIÓN	7
2. OBJETIVOS	7
2.1 Objetivo General	7
2.2 Objetivos Específicos	7
3. MARCO TEORICO	8
3.1 Web Services RESTful y su Arquitectura	8
3.2 Patrón de Arquitectura Modelo-Vista-Controlador (MVC)	8
3.3 Base de Datos en la Nube: Google Cloud SQL Server	9
3.4 Payara y Java para Servicios SOAP	9
4. DESARROLLO DEL APLICATIVO	10
4.1 INSTALACIÓN DE HERRAMIENTAS.....	10
4.1.1 HERRAMIENTAS NECESARIAS.....	10
4.1.2 CREACIÓN DE BD en la NUBE	11
4.2 APlicación SERVIDOR.....	16
4.2.1 MODELO MVC	17
4.2.2 CONEXIÓN A BASE DE DATOS	18
4.2.4 CODIFICACIÓN DE CLASES	21
4.2.5 CREACIÓN DEL SERVICIO WEB	43
4.3 CREACIÓN DEL PROYECTO CONSOLA	46
4.3.1 ESTRUCTURA MVC.....	47
4.3.2 CODIFICACIÓN DE CLASES	48
4.3.4 EJECUCIÓN.....	61
4.4 CREACIÓN DEL PROYECTO CLIENTE ESCRITORIO	62
4.4.1 CREACIÓN DE LA CAPA DE SERVICIO	63
4.4.4 CREACIÓN DE LA VISTA	71
4.5 CREACIÓN DEL PROYECTO WEB	139
4.5.1 CREACIÓN ENTORNO.....	139
4.5.2 CREACIÓN CAPA SERVICIO	140

4.5.3 CREACIÓN CAPA VISTA	144
4.5.4 EJECUCION.....	171
4.6 CREACIÓN DEL PROYECTO MÓVIL.....	174
4.6.1 CREACIÓN PROYECTO MÓVIL.....	174
4.6.2 CREACIÓN CAPA MODELO.....	177
4.6.3. CREACIÓN CAPA SERVICIO	187
4.6.4 Creación Capa Vista.....	193
4.6.4 EJECUCIÓN.....	206
5. CONCLUSIONES	210
6. RECOMENDACIONES	211

ÍNDICE DE IMÁGENES

Figura 1 MySQL WorkBench.....	10
Figura 2 Estructura inicial del proyecto.....	11
Figura 3 Pantalla de Cloud SQL.....	11
Figura 4 Ingreso Gmail.....	11
Figura 5 Configuración de Cloud SQL.....	12
Figura 6 Configuración de Cloud SQL.....	13
Figura 7 Tablas de la base de datos.....	13
Figura 8 Configuración de Network	14
Figura 9 Creación Base de datos y carga	15
Figura 10 Nuevo proyecto	16
Figura 11 Aplicación tipo web	16
Figura 12 Estructura de servidor	17
Figura 13 Creación de los paquetes	17
Figura 14 Estructura de las carpetas	17
Figura 15 Creación de la clase AccesoDB	18
Figura 16 Codificación de la conexión de la base de datos.....	18
Figura 17 Creación de la clase movimiento	21
Figura 18 Configuración de nombre.....	43
Figura 19 Codificación de los servicios web Movimientos.....	43
Figura 20 Implementación de etiquetas	43
Figura 21 Codificación del servicio web Deposito.....	44
Figura 22 Despliegue de la aplicación	44
Figura 23 Inicio del servidor	45
Figura 24 Verificación de los servicios web creados	45
Figura 25 Creación Dotnet Consola.....	46
Figura 26 Creación proyecto estructura general.....	47
Figura 27 Creación paquete ec.edu.monster.controlador.....	47
Figura 28 Creación paquete ec.edu.monster.servicio.....	47
Figura 29 Creación paquete ec.edu.monster.modelo	48

Figura 30 Estructura proyecto cliente	48
Figura 31 Creación clase Movimiento	49
Figura 32 Codificación clase Movimiento	49
Figura 33 Creación clase EurekaService	49
Figura 34 Codificación clase EurekaService.....	50
Figura 35 Creación clase ConsoleView	51
Figura 36 Codificación clase WSEurekaClient	51
Figura 37 Ejecución clase WSEurekaClient.....	61
Figura 38 Ejecución clase WSEurekaClient.....	61
Figura 39 Creación del proyecto cliente.....	62
Figura 40 Creación del cliente del servicio web	62
Figura 41 Estructura inicial del proyecto.....	63
Figura 42 Creación de la clase EurekaService	63
Figura 43 Llamada a los métodos del servicio web.....	63
Figura 44 Creación de las vistas	71
Figura 45 Prueba a la interfaz gráfica principal.....	71
Figura 46 Prueba a la interfaz gráfica de gestión de cuentas	72
Figura 47 Prueba a la interfaz gráfica de consulta de movimientos bancarios	72
Figura 48 Creación directorio 02.CLIENTE WEB	139
Figura 49 Creación Estructura Web	139
Figura 50 Ejecución Vista Principal.....	171
Figura 51 Ejecución Vista Consultar Clientes	172
Figura 52 Ejecución Vista Consultar Movimientos.....	172
Figura 53 Ejecución Vista Realizar Depósitos.....	173
Figura 54 Ejecución Vista Realizar Transferencia.....	173
Figura 55 Ejecución Vista Retiro.....	174
Figura 56 Creación Nuevo Proyecto desde File.....	174
Figura 57 Creación Nuevo Proyecto Selección Herramienta	174
Figura 58 Configuración el nombre de la solución.....	175
Figura 59 Configuración Nuevo Proyecto	175
Figura 60 Estructura proyecto.....	176

Figura 61 Creación nueva clase en paquete modelo	177
Figura 62 Codificación clase MovimientoData.....	177
Figura 63 Creación clase CuentaData.....	178
Figura 64 Codificación clase CuentaData	179
Figura 65 Creación modelos de carga de las vistas.....	179
Figura 66 Modelo para Carga de Movimientos.....	180
Figura 67 Carga de datos en Cuentas.....	184
Figura 68 Creación clase EurekaBankService	187
Figura 69 Instancia de clase EurekaBankService.....	187
Figura 70 Modificación clase AppShell.....	192
Figura 71 Creación archivos en carpeta Views.....	193
Figura 72 Ejecución de Pantalla de Inicio de Sesión	206
Figura 73 Ejecución Pantalla Principal.....	207
Figura 74 Ejecución Pantalla Depósitos	208
Figura 75 Ejecución Pantalla Consultas de una cuenta en específico.....	209
Figura 76 Ejecución Pantalla Retiro.....	209
Figura 77 Ejecución Pantalla Transferencia	210

ÍNDICE DE TABLAS

TABLA 1 Clase Acceso Base de Datos	19
TABLA 2 Codificación clase Movimiento	22
TABLA 3 Codificación Clase EurekaService.....	25
TABLA 4 Codificación Clase WSEurekaClient.....	52
TABLA 5 Clase llamada a servicios web.....	64
TABLA 6 ConsultaView	73
TABLA 7 MainView	89
TABLA 8 DepositoView	105
TABLA 9 Codificación App.py	140
TABLA 10 Codificación Vista Principal	144
TABLA 11 Codificación realizarDepósitos.....	147
TABLA 12 Codificación clase MovimientoData	177
TABLA 13 Codificación clase MovementsViewModel	180
TABLA 14 Codificación clase AccountsViewModel	184
TABLA 15 Codificación clase EurekaBankService	187
TABLA 16 Codificación clase AppShell.....	193
TABLA 17 Codificación archivo AccountsScreen.xaml	194
TABLA 18 Codificación archivo LoginScreen.xaml.....	196
TABLA 19 Codificación archivo MovementsScreen.xaml.....	199
TABLA 20 Codificación archivo NewMovementScreen.xaml	203

1. INTRODUCCIÓN

La **API Web RESTful (Representational State Transfer)** constituye una tecnología fundamental para la interoperabilidad de aplicaciones modernas. Su diseño se basa en el **intercambio de datos en formato JSON** (principalmente), empleando métodos HTTP estándar para establecer un protocolo que dicta la manera en que los recursos deben ser accedidos y las operaciones deben ser invocadas. Una de las mayores fortalezas de REST reside en su **independencia de lenguaje de programación y plataforma**, cualidad que lo posiciona como una solución óptima para entornos que exigen alta velocidad, escalabilidad y una comunicación fluida entre sistemas.

Dentro del ámbito de la arquitectura de *software*, la implementación de las API REST está usualmente documentada mediante especificaciones como **OpenAPI** o **Swagger**. Estos lenguajes son esenciales porque detallan la interfaz completa del servicio, especificando los *endpoints* disponibles, los parámetros requeridos y las estructuras de datos manejadas, garantizando así una interacción precisa y rigurosamente definida entre los componentes cliente y servidor.

En el ecosistema **.NET**, el desarrollo y consumo de API se simplifica mediante *frameworks* robustos como **ASP.NET Core Web API** y el lenguaje de programación **C#**. Estas herramientas, junto con el *framework* **.NET Core**, proveen el soporte necesario para manejar las especificaciones REST y JSON, facilitando su despliegue en entornos de servidor como **Kestrel** (integrado) o **IIS (Internet Information Services)**.

Para la gestión y persistencia de los datos en esta implementación, se ha seleccionado una base de datos **SQL Server**. Esta elección, común en entornos .NET, combina la robustez y funcionalidad avanzada de **SQL Server** con la escalabilidad y flexibilidad de la nube (o de un *hosting* empresarial), asegurando una alta disponibilidad y acceso seguro.

2. OBJETIVOS

2.1 Objetivo General

El propósito fundamental de este trabajo es la elaboración de un manual técnico exhaustivo que detalle las fases de diseño y desarrollo de un Conversor de Unidades con capacidades para gestionar conversiones de longitud y peso. La solución arquitectónica se basará en una API Web RESTful implementada en el lenguaje C# (utilizando ASP.NET Core), la cual será desplegada en el servidor de aplicaciones Kestrel/IIS. Para la persistencia de datos, se empleará SQL Server en un entorno de nube. Finalmente, se documentará la creación de interfaces de cliente que permitan acceder a la API desde una variedad de entornos, incluyendo aplicaciones de escritorio, consola, web y móvil.

2.2 Objetivos Específicos

- Se describirá de forma exhaustiva el proceso de diseño y desarrollo de la API Web RESTful, detallando el uso específico del lenguaje C# con ASP.NET Core Web API, el

- despliegue en el servidor Kestrel/IIS, y la estrategia de gestión de datos persistentes mediante SQL Server (utilizando Entity Framework Core).
- Se proporcionarán directrices metodológicas para el desarrollo de aplicaciones cliente destinadas a diversas plataformas (como escritorio, consola, web y móvil). Estas guías enfatizarán la integración exitosa y el consumo eficiente de la API REST central, detallando el manejo del formato JSON.
 - Se procederá a la clarificación de los principios fundamentales inherentes a la arquitectura REST, el formato de datos JSON, la especificación OpenAPI/Swagger y las tecnologías específicas de .NET utilizadas en el proyecto. El objetivo es asegurar que el lector obtenga una comprensión profunda de su importancia estratégica dentro del desarrollo de soluciones de software modernas.

3. MARCO TEORICO

3.1 Web Services RESTful y su Arquitectura

Una API Web RESTful es un servicio web que adhiere a los principios arquitectónicos REST (Representational State Transfer). Esta API utiliza los métodos HTTP estándar (GET, POST, PUT, DELETE) para intercambiar mensajes que representan el estado de los recursos, típicamente estructurados en formato JSON (JavaScript Object Notation). Las principales características de REST incluyen:

- **Estandarización y Simplicidad:** Utiliza los protocolos y verbos existentes de HTTP, lo que simplifica su diseño e implementación. El formato JSON, ligero y legible, asegura una alta interoperabilidad entre diferentes clientes y servidores.
- **Protocolo de Transporte Único:** Opera exclusivamente sobre HTTP/HTTPS, aprovechando la infraestructura web existente para la transmisión de datos.
- **Contrato de Interfaz Documentado:** Las APIs RESTful suelen utilizar estándares como Swagger para describir de manera explícita los *endpoints*, los parámetros de entrada y salida, y los modelos de datos.
- **Escalabilidad y Flexibilidad:** Promueve la arquitectura *sin estado* (*stateless*), lo que facilita la escalabilidad horizontal y permite la implementación de seguridad moderna (como tokens JWT) y el versionado de la API.

3.2 Patrón de Arquitectura Modelo-Vista-Controlador (MVC)

El Patrón Modelo-Vista-Controlador (MVC) organiza las aplicaciones en tres componentes principales, cuyo objetivo es aislar la lógica de negocio de la interfaz de usuario.

- **Modelo:** Este componente representa la lógica de negocio y los datos de la aplicación. En el contexto de una API Web RESTful, el Modelo incluye las clases que definen las entidades o recursos (utilizando Entity Framework Core para la persistencia en SQL

Server) y el código que ejecuta las reglas de negocio (como la lógica de las conversiones de unidades). Este componente opera independientemente del protocolo de comunicación.

- **Vista:** Se encarga de la presentación de los datos al usuario final. A diferencia de las aplicaciones tradicionales, en una arquitectura RESTful, la Vista es la aplicación cliente separada (web, móvil, escritorio). Su función es consumir los datos de la API (generalmente en formato JSON), interpretar la respuesta y renderizar la interfaz gráfica correspondiente.
- **Controlador:** Actúa como el intermediario, gestionando las solicitudes HTTP entrantes (las peticiones REST). En ASP.NET Core Web API, los Controladores reciben la solicitud, invocan las operaciones necesarias en el Modelo para procesar los datos o la lógica, y finalmente envían la respuesta serializada, comúnmente en formato JSON, de vuelta al cliente.

3.3 Base de Datos en la Nube: Google Cloud SQL Server

La solución utilizará una base de datos **SQL Server** gestionada o alojada en la nube, la cual se presenta como un servicio de gestión robusto que amalgama la funcionalidad inherente de SQL Server con las ventajas de una infraestructura en la nube. Estas características clave son:

- **Disponibilidad Garantizada:** El servicio asegura una alta disponibilidad (High Availability) mediante la replicación automática de los datos y mecanismos de *failover*, lo que permite una recuperación eficiente y continua ante cualquier eventualidad o fallo.
- **Capacidad de Crecimiento:** Ofrece una notable escalabilidad, facilitando el ajuste dinámico de los recursos de la base de datos para responder de manera óptima a las demandas y al crecimiento de la API REST.
- **Protección de Datos:** Se implementa un esquema robusto de seguridad que incluye el control de acceso basado en identidades, la obligatoriedad de conexiones cifradas (SSL/TLS) y avanzados sistemas de monitoreo y auditoría.
- **Compatibilidad Sencilla:** Presenta una excelente facilidad de integración con la plataforma .NET, siendo la base de datos nativa y accediéndose eficientemente a través de Entity Framework Core (ORM) o del *driver* ADO.NET.

Dentro del marco de este proyecto, SQL Server tendrá la función de centralizar y almacenar toda la información requerida para las operaciones del conversor de unidades. Esto incluye la persistencia de los parámetros de configuración, las unidades de medida manejadas y un registro histórico de las conversiones realizadas.

3.4 Payara y Java para Servicios SOAP

La solución se fundamenta en el uso de C# y .NET Core para el desarrollo de la API. El servidor de aplicaciones Kestrel, junto con IIS (Internet Information Services), representa el entorno de *hosting* ideal. Kestrel es un servidor web ligero integrado en .NET Core, mientras que IIS es la opción robusta de Microsoft para entornos productivos. Ambos cumplen con las especificaciones de .NET Core y son plataformas de alto rendimiento, idóneas para alojar API RESTful de alta demanda, gracias a su soporte nativo para ASP.NET Core Web API.

En el marco de esta implementación, el desarrollo de la API REST se llevará a cabo utilizando ASP.NET Core Web API y el lenguaje C#. Los componentes principales y sus funciones son los siguientes:

- **ASP.NET Core Web API:** Será el *framework* principal empleado para la definición de los *endpoints* REST, encargándose de la serialización/deserialización de datos en formato JSON y de la documentación (por ejemplo, mediante OpenAPI/Swagger).
- **SQL Server:** Se encargará del almacenamiento persistente de los datos requeridos por la API. El acceso a esta base de datos se realizará mediante Entity Framework Core (ORM), lo que asegura un mapeo objeto-relacional eficiente y un rendimiento optimizado.
- **Eficiencia de Despliegue:** La arquitectura permite que la API sea empaquetada y desplegada de manera sencilla en el servidor Kestrel o IIS, asegurando un entorno de ejecución de alta velocidad y garantizando la conectividad segura con la base de datos SQL Server.

4. DESARROLLO DEL APLICATIVO

4.1 INSTALACIÓN DE HERRAMIENTAS

4.1.1 HERRAMIENTAS NECESARIAS

Para la elaboración del proyecto, en primer lugar, se debe tener dbeaver.

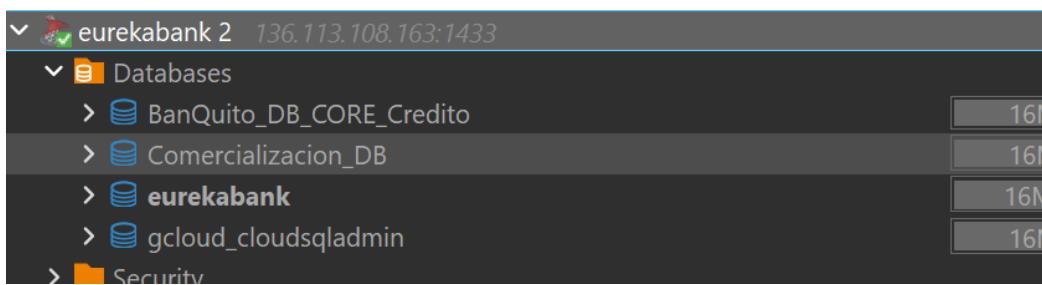


Figura 1 MySQL WorkBench

📁 01.SERVIDOR	2/6/2024 13:10	Carpeta de archivos
📁 02. CLIENTE CONSOLA	2/6/2024 16:05	Carpeta de archivos
📁 02. CLIENTE ESCRITORIO	2/6/2024 15:17	Carpeta de archivos
📁 02. CLIENTE WEB	8/6/2024 17:20	Carpeta de archivos
📁 02. CLIENTE MÓVIL	9/6/2024 19:34	Carpeta de archivos
📁 03. BDD	9/6/2024 19:34	Carpeta de archivos
📁 04. DOCUMENTACIÓN	9/6/2024 19:34	Carpeta de archivos

Figura 2 Estructura inicial del proyecto

4.1.2 CREACIÓN DE BD en la NUBE

A continuación, se describe el paso a paso para crear una base de datos MySQL en Google Cloud Platform (GCP) y configurarla para que sea utilizada por el servicio SOAP

Figura 3 Pantalla de Cloud SQL

Figura 4 Ingreso Gmail

Se crea un Proyecto en Google Cloud. En la parte superior de la consola, haz clic en el menú desplegable de proyectos y selecciona Nuevo Proyecto. Primero se inserta la información de la cuenta que incluye el país de uso y el sistema de facturación. Posteriormente se crea la instancia de MySQL dentro de Google Cloud.

Elige una edición de Cloud SQL

Una edición de Cloud SQL determina las características fundamentales de tu instancia. Elige la mejor opción según tu presupuesto y necesidades de rendimiento. [Más información](#)

Resumen	
Edición de Cloud SQL	Enterprise
Región	us-central1 (Iowa)
Versión de la base de datos	MySQL 8.0
Tipo de máquina	db-custom-8-32768
CPU virtuales	8 vCPU
RAM	32 GB
Caché de datos	Inhabilitado
Almacenamiento	250 GB SSD
Conexiones	IP pública
Copia de seguridad	Automatizada
Disponibilidad	Varias zonas (con alta disponibilidad)
Recuperación de un momento determinado	Habilitada
Capacidad de procesamiento de la red (MB/s) <small>(?)</small>	2,000 de 2,000
IOPS <small>(?)</small>	Lectura: 13,500 de 15,000 Escritura: 13,500 de 15,000
Capacidad de procesamiento del disco (MB/s) <small>(?)</small>	Lectura: 120.0 de 800.0 Escritura: 120.0 de 800.0

Precio estimado (sin descuentos)

Estos elementos representan únicamente los recursos de procesamiento, memoria y almacenamiento de Cloud SQL y reflejan la configuración que estableciste para tu instancia

Figura 5 Configuración de Cloud SQL

Creamos la instancia que servirán para las bases de datos de mariaDB

Descripción general Editar Importar Exportar

Todas las instancias > sql-server-db-gr05

sql-server-db-gr05

SQL Server 2022 Standard

Aprende los conceptos básicos de Cloud SQL

Figura 6 Configuración de Cloud SQL

Creada la base de datos se comprueba la correcta creación

Bases de datos

Todas las instancias > sql-server-db-gr05

sql-server-db-gr05

SQL Server 2022 Standard

Intercalación SQL_Latin1_General_CI_AS
predeterminada

Crear base de datos

Nombre	Intercalación	
BanQuito_DB_CORE_Credito	SQL_Latin1_General_CI_AS	
Comercializacion_DB	SQL_Latin1_General_CI_AS	
eurekabank	SQL_Latin1_General_CI_AS	

Figura 7 Tablas de la base de datos

Una vez creada se agrega una ip accesible para el público.

Todas las instancias > sql-server-db-gr05

✓ **sql-server-db-gr05**

SQL Server 2022 Standard

Resumen **Redes** Seguridad Pruebas de conectividad

Elige cómo quieres que se conecte la fuente a esta instancia y, luego, define las redes que estarán autorizadas para conectarse.[Más información](#)

Puedes usar el proxy de Cloud SQL para aumentar la seguridad con cualquiera de las opciones.[Más información](#)

Asignación de IP de la instancia

IP privada
Asigna una dirección IP de VPC interna alojada en Google. Se requieren APIs y permisos adicionales.[Más información](#)

IP pública
Asigna una dirección IP externa a la que se puede acceder a través de Internet. Se requiere el uso de una red autorizada o el proxy de Cloud SQL para conectarse a esta instancia.[Más información](#)

Redes autorizadas

Puedes especificar rangos de CIDR para permitir que las direcciones IP de esos rangos accedan a tu instancia.[Más información](#)

⚠️ Si permites 0.0.0.0/0, se abrirá tu instancia a todos los clientes IPv4, incluidos los no deseados. Usa rangos de IP más estrechos o el proxy de autenticación de Cloud SQL para mejorar la seguridad. Si debes permitir 0.0.0.0/0, asegúrate de que las contraseñas sean seguras.[Más información](#)

▼ Global (0.0.0.0/0)	
Añadir otra red	

Figura 8 Configuración de Network

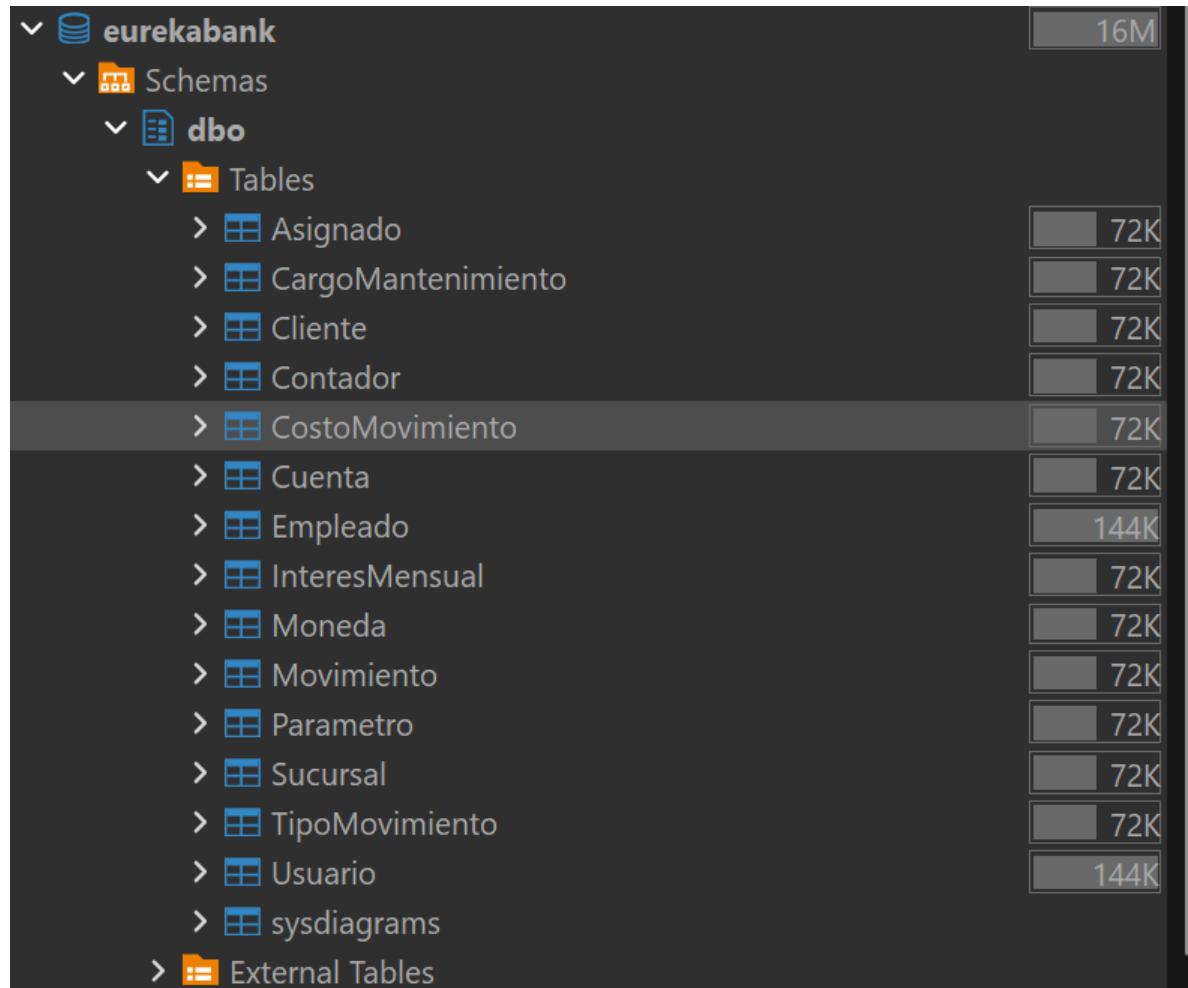


Figura 9 Creación Base de datos y carga

4.2 APLICACIÓN SERVIDOR

El primer paso esencial consiste en la creación de una nueva solución. Para ello, se selecciona la opción de proyecto Web API dentro de la categoría .NET (o ASP.NET Core en versiones modernas de Visual Studio). Este proceso se inicia seleccionando la opción Archivo > Nuevo > Proyecto y eligiendo la plantilla ASP.NET Core Web API. Esta selección configura automáticamente el entorno de desarrollo en C# con la estructura fundamental para exponer recursos RESTful.

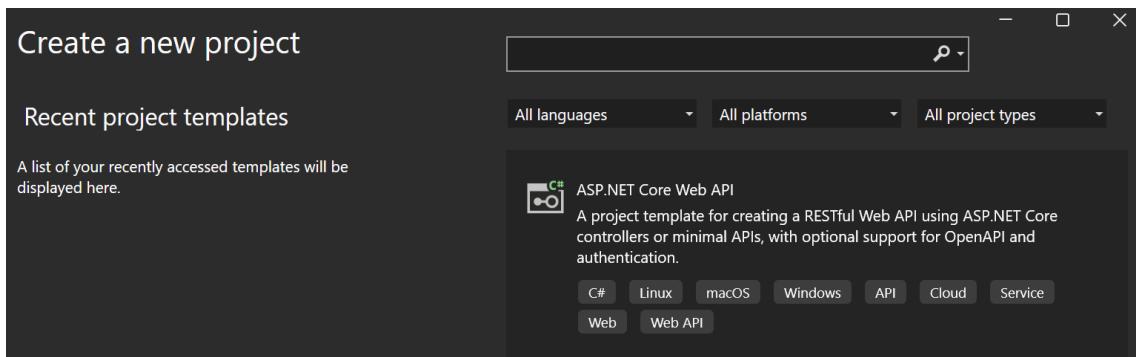


Figura 10 Nuevo proyecto

Posteriormente, seleccionar la opción Java Web > Web Application.

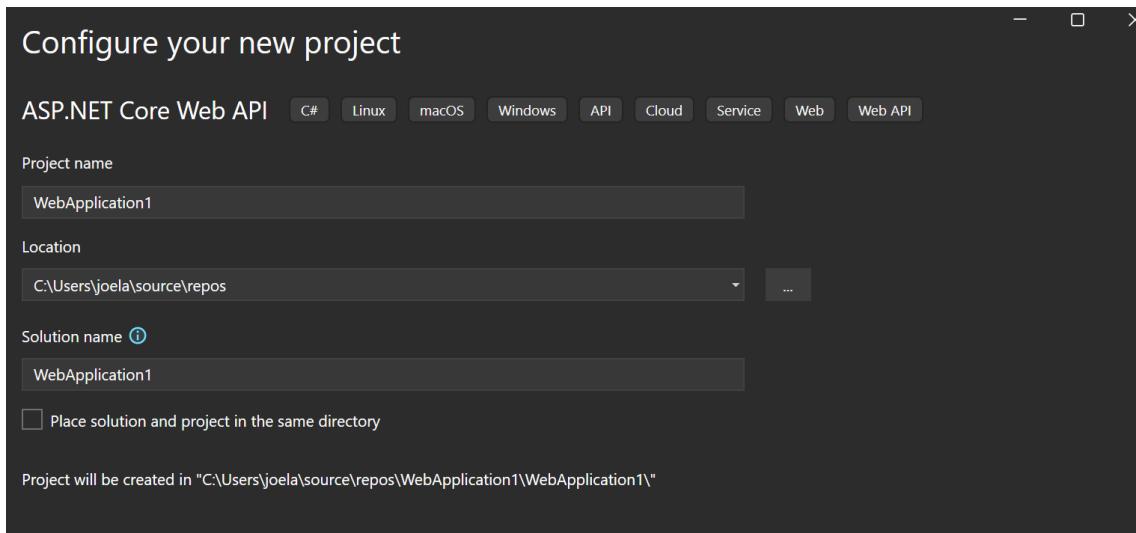


Figura 11 Aplicación tipo web

Se verifica la estructura del proyecto Servidor

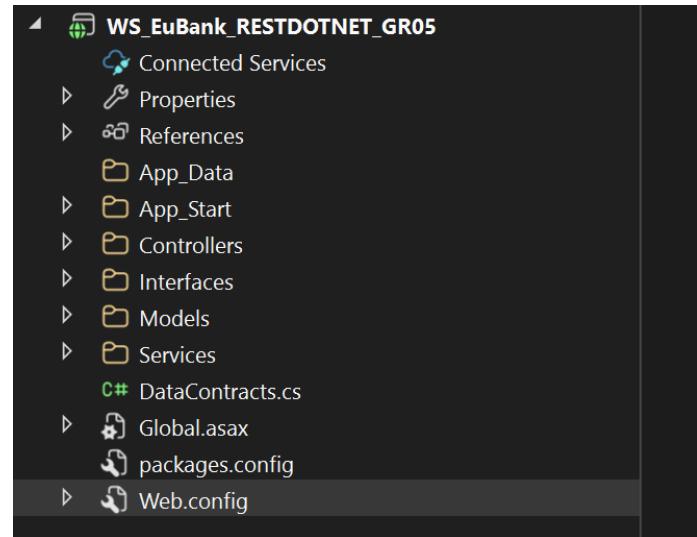


Figura 12 Estructura de servidor

4.2.1 MODELO MVC

Se crea las carpetas necesarias para el entorno de trabajo.



Figura 13 Creación de los paquetes

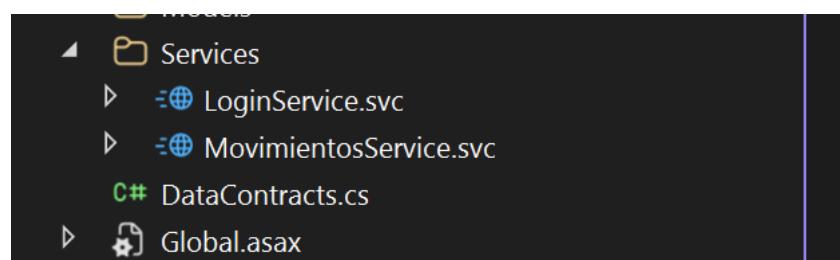


Figura 14 Estructura de las carpetas

4.2.2 CONEXIÓN A BASE DE DATOS

Se crea la conexión con la base de datos con el uso de una librería JDBC que sería la conexión con el paquete que se crea con las credenciales para la conexión.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <configSections>
        <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkId=237468 -->
        <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" />
    </configSections>
    <appSettings>
        <add key="aspnet:UseTaskFriendlySynchronizationContext" value="true" />
    </appSettings>
    <connectionStrings>
        <add name="EurekaBankDB" connectionString="Data Source=136.113.108.1433;Initial Catalog=eurekabank;User ID=sqlserver;Password=BaRiYan&quot;" />
    </connectionStrings>
    <system.web>
        <compilation debug="true" targetFramework="4.7.2" />
        <httpRuntime targetFramework="4.7.2" />
    </system.web>
    <system.serviceModel>
        <behaviors>
            <serviceBehaviors>
                <behavior>
```

Figura 15 Creación de la clase AccesoDB

```
1   using System.Data.Entity;
2
3   namespace WS_EuBank_SOAPDOTNET_GR05
4   {
5
6       public class EurekaBankContext : DbContext
7   {
8       public EurekaBankContext() : base("EurekaBankDB") { }
9
10      public DbSet<Cuenta> Cuentas { get; set; }
11      public DbSet<Movimiento> Movimientos { get; set; }
12      public DbSet<Usuario> Usuarios { get; set; }
13      public DbSet<Cliente> Clientes { get; set; }
14      public DbSet<TipoMovimiento> TiposMovimiento { get; set; }
15
16      protected override void OnModelCreating(DbModelBuilder modelBuilder)
17      {
18          modelBuilder.Entity<Cuenta>().HasKey(c => c.ChrCuenCodigo);
19          modelBuilder.Entity<Movimiento>().HasKey(m => new { m.ChrCuenCodigo, m.IntMoviNumero });
20          modelBuilder.Entity<Usuario>().HasKey(u => u.ChrEmplCodigo);
21          modelBuilder.Entity<Cliente>().HasKey(cl => cl.ChrClieCodigo);
22          modelBuilder.Entity<TipoMovimiento>().HasKey(tm => tm.ChrTipoCodigo);
23      }
24  }
```

Figura 16 Codificación de la conexión de la base de datos

Tabla 1 Clase Acceso Base de Datos

```
using System.Data.Entity;

namespace WS_EuBank_SOAPDOTNET_GR05
{
    public class EurekaBankContext : DbContext
    {
        public EurekaBankContext() : base("EurekaBankDB") { }

        public DbSet<Cuenta> Cuentas { get; set; }
        public DbSet<Movimiento> Movimientos { get; set; }
        public DbSet<Usuario> Usuarios { get; set; }
        public DbSet<Cliente> Clientes { get; set; }
        public DbSet<TipoMovimiento> TiposMovimiento { get; set; }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Cuenta>().HasKey(c => c.ChrCuenCodigo);
            modelBuilder.Entity<Movimiento>().HasKey(m => new { m.ChrCuenCodigo, m.IntMoviNumero });
            modelBuilder.Entity<Usuario>().HasKey(u => u.ChrEmplCodigo);
            modelBuilder.Entity<Cliente>().HasKey(cl => cl.ChrClieCodigo);
        }
    }
}
```

```
modelBuilder.Entity<TipoMovimiento>().HasKey(tm => tm.ChrTipoCodigo);  
}  
}  
}
```

4.2.4 CODIFICACIÓN DE CLASES

Habiendo obtenido la conexión se crea una nueva clase dentro del paquete modelo llamada Movimiento.

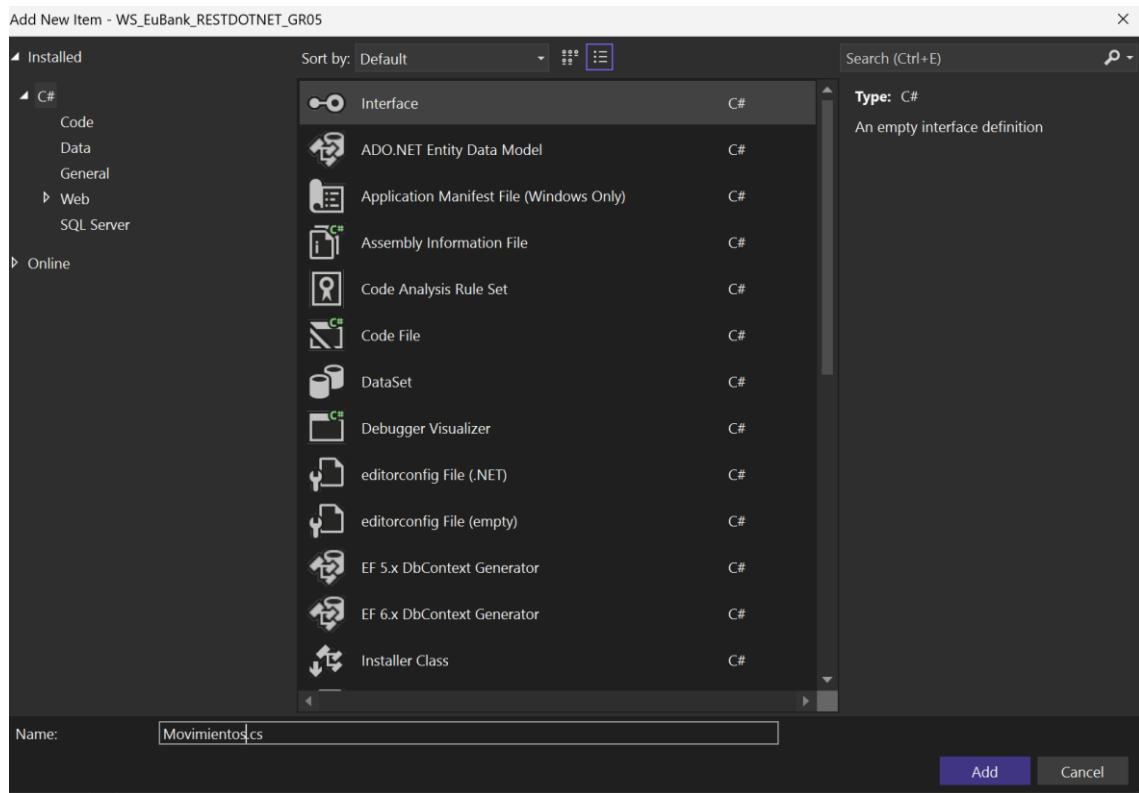


Figura 17 Creación de la clase movimiento

TABLA 2 Codificación clase Movimiento

```
using System;  
  
using System.ComponentModel.DataAnnotations;  
  
using System.ComponentModel.DataAnnotations.Schema;  
  
  
namespace WS_EuBank_SOAPDOTNET_GR05  
{  
    [Table("Movimiento")]  
    public class Movimiento  
    {  
        [Key]  
        [Column("chr_cuencodigo", Order = 0)]  
        public string ChrCuenCodigo { get; set; }  
    }  
}
```

[Key]

```
[Column("int_movinumero", Order = 1)]
```

```
public int IntMoviNumero { get; set; }
```

```
[Column("dtt_movifecha")]
```

```
public DateTime DttMoviFecha { get; set; }
```

```
[Column("chr_emplcodigo")]
```

```
public string ChrEmplCodigo { get; set; }
```

```
[Column("chr_tipocodigo")]
```

```
public string ChrTipoCodigo { get; set; }
```

```
[Column("dec_moviimporte")]

public decimal DecMovilImporte { get; set; }

[Column("chr_cuenreferencia")]

public string ChrCuenReferencia { get; set; }

public virtual Cuenta Cuenta { get; set; }

}
```

TABLA 3 Codificación Clase EurekaService

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace WS_EuBank_SOAPDOTNET_GR05
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu to change the class name "MovimientosService" in code, svc and config file together.

    // NOTE: In order to launch WCF Test Client for testing this service, please select MovimientosService.svc or MovimientosService.svc.cs at the Solution Explorer and start
    // debugging.

    public class MovimientosService : IMovimientosService
```

```
private const string EmpleadoDefault = "0001";  
  
private const string TipoDeposito = "003";  
  
private const string TipoRetiro = "004";  
  
private const string TipoTransferenciaSalida = "009";  
  
private const string TipoTransferencialIngreso = "008";  
  
public CuentaData GetDatosCuenta(string cuenta)  
{  
    using (var context = new EurekaBankContext())  
    {  
        var cuentaData = (from c in context.Cuentas  
                         join cl in context.Clientes on c.ChrClieCodigo equals cl.ChrClieCodigo  
                         where c.ChrCtaCodigo == cuenta  
                         select new CuentaData {  
                             CtaCodigo = c.ChrCtaCodigo,  
                             ClieNombre = cl.ChrClieNombre,  
                             ClieApellido = cl.ChrClieApellido,  
                             CtaSaldo = c.DblCtaSaldo  
                         }).Single();  
        return cuentaData;  
    }  
}
```

```
where c.ChrCuenCodigo == cuenta

select new CuentaData

{
    Código = c.ChrCuenCodigo,
    Moneda = c.ChrMoneCodigo,
    Saldo = c.DecCuenSaldo,
    Estado = c.VchCuenEstado,
    NombreCliente = cl.VchClieNombre + " " + cl.VchCliePaterno + " " + cl.VchClieMaterno,
    EmailCliente = cl.VchClieEmail,
    TelefonoCliente = cl.VchClieTelefono
}).FirstOrDefault();

return cuentaData;
}
```

```
}

public List<MovimientoData> GetMovimientos(string cuenta)

{
    using (var context = new EurekaBankContext())
    {
        var query = context.Movimientos.AsQueryable();

        if (!string.IsNullOrWhiteSpace(cuenta) && cuenta != "?")
        {
            query = query.Where(m => m.ChrCuenCodigo == cuenta);
        }

        return query.OrderBy(m => m.DttMoviFecha).ThenBy(m => m.IntMoviNumero)
            .Select(m => new MovimientoData
```

```
{  
  
    Numero = m.IntMoviNumero,  
  
    Fecha = m.DttMoviFecha,  
  
    Tipo = m.ChrTipoCodigo,  
  
    Importe = m.DecMovilImporte,  
  
    Referencia = m.ChrCuenReferencia  
  
}).ToList();  
  
}  
  
}  
  
  
public ResultadoOperacion ProcesarMovimiento(string tipo, string cuenta, string cuentaDestino, decimal monto)  
  
{  
  
    if (monto <= 0) return new ResultadoOperacion(-1, "El monto debe ser mayor a cero.");
```

```
switch (tipo.ToUpper())  
{  
    case "DEPOSITO":  
  
        return Deposito(cuenta, monto) ? new ResultadoOperacion(1, "Depósito realizado exitosamente.") : new ResultadoOperacion(-1, "Error al realizar el depósito.");  
  
    case "RETIRO":  
  
        return Retiro(cuenta, monto) ? new ResultadoOperacion(1, "Retiro realizado exitosamente.") : new ResultadoOperacion(-1, "Error al realizar el retiro. Verifique saldo suficiente.");  
  
    case "TRANSFERENCIA":  
  
        if (string.IsNullOrEmpty(cuentaDestino)) return new ResultadoOperacion(-1, "Cuenta destino requerida para transferencia.");  
  
        return Transferencia(cuenta, cuentaDestino, monto) ? new ResultadoOperacion(1, "Transferencia realizada exitosamente.") : new ResultadoOperacion(-1, "Error al realizar la transferencia. Verifique cuentas y saldo.");  
  
    default:  
  
        return new ResultadoOperacion(-1, "Tipo de movimiento no válido.");  
}
```

```
}

private bool Deposito(string cuenta, decimal monto)
{
    using (var context = new EurekaBankContext())
    {
        using (var transaction = context.Database.BeginTransaction())
        {
            try
            {
                var c = context.Cuentas.FirstOrDefault(cu => cu.ChrCuenCodigo == cuenta);
                if (c == null) return false;

                int nextNum = c.IntCuenContMov + 1;
```

```
c.DecCuenSaldo += monto;

c.IntCuenContMov = nextNum;

var mov = new Movimiento

{

    ChrCuenCodigo = cuenta,

    IntMoviNumero = nextNum,

    DttMoviFecha = DateTime.Now.Date,

    ChrEmplCodigo = EmpleadoDefault,

    ChrTipoCodigo = TipoDeposito,

    DecMovilImporte = monto

};

context.Movimientos.Add(mov);

context.SaveChanges();
```

```
        transaction.Commit();

        return true;
    }

    catch
    {
        transaction.Rollback();

        return false;
    }
}

}

private bool Retiro(string cuenta, decimal monto)
```

```
{  
  
    using (var context = new EurekaBankContext())  
  
    {  
  
        using (var transaction = context.Database.BeginTransaction())  
  
        {  
  
            try  
  
            {  
  
                var c = context.Cuentas.FirstOrDefault(cu => cu.ChrCuenCodigo == cuenta);  
  
                if (c == null || c.DecCuenSaldo < monto) return false;  
  
                int nextNum = c.IntCuenContMov + 1;  
  
                c.DecCuenSaldo -= monto;  
  
                c.IntCuenContMov = nextNum;  
  
                var mov = new Movimiento
```

```
{  
  
    ChrCuenCodigo = cuenta,  
  
    IntMoviNumero = nextNum,  
  
    DttMoviFecha = DateTime.Now.Date,  
  
    ChrEmplCodigo = EmpleadoDefault,  
  
    ChrTipoCodigo = TipoRetiro,  
  
    DecMovilImporte = monto  
  
};  
  
context.Movimientos.Add(mov);  
  
context.SaveChanges();  
  
transaction.Commit();  
  
return true;  
  
}
```

```
        catch

        {

            transaction.Rollback();

            return false;

        }

    }

}

}

private bool Transferencia(string cuentaOrigen, string cuentaDestino, decimal monto)

{

    if (cuentaOrigen == cuentaDestino) return false;

    using (var context = new EurekaBankContext())
```

```
{  
  
    using (var transaction = context.Database.BeginTransaction())  
  
    {  
  
        try  
  
        {  
  
            var origen = context.Cuentas.FirstOrDefault(cu => cu.ChrCuenCodigo == cuentaOrigen);  
  
            var destino = context.Cuentas.FirstOrDefault(cu => cu.ChrCuenCodigo == cuentaDestino);  
  
            if (origen == null || destino == null || origen.DecCuenSaldo < monto || destino.VchCuenEstado != "ACTIVO") return false;  
  
            int nextNumOri = origen.IntCuenContMov + 1;  
  
            int nextNumDest = destino.IntCuenContMov + 1;  
  
            origen.DecCuenSaldo -= monto;  
        }  
    }  
}
```

```
origen.IntCuenContMov = nextNumOri;

destino.DecCuenSaldo += monto;

destino.IntCuenContMov = nextNumDest;

var movOri = new Movimiento

{

    ChrCuenCodigo = cuentaOrigen,

    IntMoviNumero = nextNumOri,

    DttMoviFecha = DateTime.Now.Date,

    ChrEmplCodigo = EmpleadoDefault,

    ChrTipoCodigo = TipoTransferenciaSalida,

    DecMovilImporte = monto,

    ChrCuenReferencia = cuentaDestino
```

```
};

var movDest = new Movimiento

{

    ChrCuenCodigo = cuentaDestino,

    IntMoviNumero = nextNumDest,

    DttMoviFecha = DateTime.Now.Date,

    ChrEmplCodigo = EmpleadoDefault,

    ChrTipoCodigo = TipoTransferencialIngreso,

    DecMovilImporte = monto,

    ChrCuenReferencia = cuentaOrigen

};

context.Movimientos.Add(movOri);

context.Movimientos.Add(movDest);
```

```
        context.SaveChanges();

        transaction.Commit();

        return true;
    }

    catch
    {
        transaction.Rollback();

        return false;
    }
}
```

```
public List<CuentaData> GetDatosCuentas()

{

    using (var context = new EurekaBankContext())

    {

        var cuentasData = (from c in context.Cuentas

                           join cl in context.Clientes on c.ChrClienCodigo equals cl.ChrClieCodigo

                           select new CuentaData

                           {

                               Codigo = c.ChrCuenCodigo,

                               Moneda = c.ChrMoneCodigo,

                               Saldo = c.DecCuenSaldo,

                               Estado = c.VchCuenEstado,

                               NombreCliente = cl.VchClieNombre + " " + cl.VchCliePaterno + " " + cl.VchClieMaterno,

```

```
EmailCliente = cl.VchClieEmail,  
  
    TelefonoCliente = cl.VchClieTelefono  
  
}).ToList();  
  
return cuentasData;  
  
}  
  
}  
  
  
public List<string> GetTiposMovimiento()  
  
{  
  
    return new List<string> { "DEPOSITO", "RETIRO", "TRANSFERENCIA" };  
  
}  
  
}  
  
}
```

4.2.5 CREACIÓN DEL SERVICIO WEB

El siguiente paso es crear el servicio web, creamos los servicios necesarios para el sistema, es decir tanto de login como el de movimientos

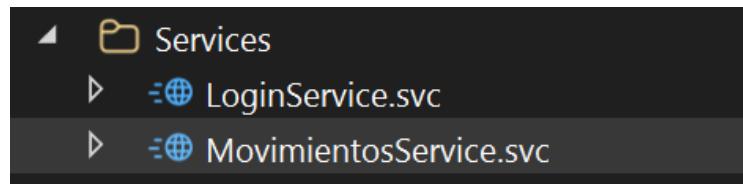


Figura 18 Configuración de nombre

```
1  using System.Collections.Generic;
2  using System.Linq;
3  using System.Web.Http;
4
5  namespace WS_EuBank_SOAPDOTNET_GR05.Controllers
6  {
7      // References
8      public class MovimientosController : ApiController
9      {
10          [HttpGet]
11          [Route("api/cuentas")]
12          // References
13          public IHttpActionResult GetDatosCuentas()
14          {
15              using (var context = new EurekaBankContext())
16              {
17                  var cuentasData = (from c in context.Cuentas
18                      join cl in context.Clientes on c.ChrClieCodigo equals cl.ChrClieCodigo
19                      select new CuentaData
20                      {
21                          Codigo = c.ChrCuenCodigo,
22                          Moneda = c.ChrMonCodigo,
23                          Saldo = c.DecCuenSaldo,
24                          Estado = c.VchCuenEstado,
25                          NombreCliente = cl.VchClieNombre + " " + cl.VchCliePaterno + " " + cl.VchClieMaterno,
26                          EmailCliente = cl.VchClieEmail,
27                          TelefonoCliente = cl.VchClieTelefono
28                      }).ToList();
29
30                  return Ok(cuentasData);
31
32          }
33          [HttpGet]
34          [Route("api/cuentas/{cuenta}")]
35          // References
36          public IHttpActionResult GetDatosCuenta(string cuenta)
37          {
38              using (var context = new EurekaBankContext())
39              {
```

Figura 19 Codificación de los servicios web Movimientos

Luego se agrega el tag de retorno y el web result para retornar la lista de movimientos.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Runtime.Serialization;
5  using System.ServiceModel;
6  using System.Text;
7
8  namespace WS_EuBank_SOAPDOTNET_GR05
9  {
10     // NOTE: You can use the "Rename" command on the "Refactor" menu to change the class name "MovimientosService" in code, svc and config file to
11     // NOTE: In order to launch WCF Test Client for testing this service, please select MovimientosService.svc or MovimientosService.svc.cs at the
12     // reference
13     public class MovimientosService : IMovimientosService
14     {
15         private const string EmpleadoDefecto = "0001";
16         private const string TipoDeposito = "003";
17         private const string TipoRetiro = "004";
18         private const string TipoTransferenciaSalida = "009";
19         private const string TipoTransferenciaIngreso = "008";
20
21         public CuentaData GetDatosCuenta(string cuenta)
22         {
23             using (var context = new EurekaBankContext())
24             {
25                 var cuentaData = (from c in context.Cuentas
26                     join cl in context.Clientes on c.ChrClieCodigo equals cl.ChrClieCodigo
27                     where c.ChrCuenCodigo == cuenta
28                     select new CuentaData
29                     {
30                         Codigo = c.ChrCuenCodigo,
31                         Moneda = c.ChrMonCodigo,
32                         Saldo = c.DecCuenSaldo,
33                         Estado = c.VchCuenEstado,
34                         NombreCliente = cl.VchClieNombre + " " + cl.VchCliePaterno + " " + cl.VchClieMaterno,
35                         EmailCliente = cl.VchClieEmail,
36                         TelefonoCliente = cl.VchClieTelefono
37                     }).FirstOrDefault();
```

Figura 20 Implementación de etiquetas

Después se procede a agregar el servicio de depósito similar al anterior servicio implementado.

Luego se procede a codificar el nuevo servicio web.

```
80      private bool Deposito(string cuenta, decimal monto)
81      {
82          using (var context = new EurekaBankContext())
83          {
84              using (var transaction = context.Database.BeginTransaction())
85              {
86                  try
87                  {
88                      var c = context.Cuentas.FirstOrDefault(cu => cu.ChrCuenCodigo == cuenta);
89                      if (c == null) return false;
90                      int nextNum = c.IntCuenContMov + 1;
91                      c.DecCuenSaldo += monto;
92                      c.IntCuenContMov = nextNum;
93                      var mov = new Movimiento
94                      {
95                          ChrCuenCodigo = cuenta,
96                          IntMovNumero = nextNum,
97                          DttMovFecha = DateTime.Now.Date,
98                          ChrEmplCodigo = EmpleadoDefault,
99                          ChrTipoCodigo = TipoDeposito,
100                         DecMovImporte = monto
101                     };
102                     context.Movimientos.Add(mov);
103                     context.SaveChanges();
104                     transaction.Commit();
105                     return true;
106                 }
107             catch
108             {
109                 transaction.Rollback();
110             }
111         }
112     }
```

Figura 21 Codificación del servicio web Deposito

Al final se agregar todos los tags necesarios. Para posteriormente realizar la ejecución la ejecución del servidor se debe limpiar y construir previamente a la ejecución.

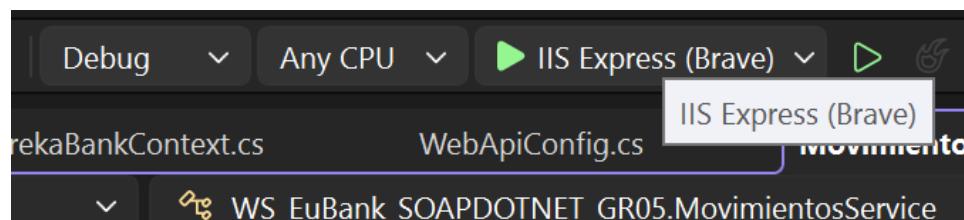


Figura 22 Despliegue de la aplicación

El servidor se levanta junto al servidor de la aplicación, se muestra el siguiente mensaje si todo sale bien.

The screenshot shows the Visual Studio IDE. The code editor displays the `MovimientosService.svc.cs` file, which contains C# code for handling bank movements. The Diagnostic Tools window is open on the right, showing a timeline from 10s to 0s. It includes sections for Events, Process Memory (MB), and CPU (% of all processors). The memory usage chart shows a peak at approximately 203 MB.

```

76     }
77   }
78 }
79
80   private bool Deposito(string cuenta, decimal monto)
81   {
82     using (var context = new EurekaBankContext())
83     {
84       using (var transaction = context.Database.BeginTransaction())
85       {
86         try
87         {
88           var c = context.Cuentas.FirstOrDefault(cu => cu.ChrCuenCodigo == cuenta);
89           if (c == null) return false;
90           int nextNum = c.IntCuenContMov + 1;
91           c.DecCuenSaldo -= monto;
92           c.IntCuenContMov = nextNum;
93           var mov = new Movimiento
94           {
95             ChrCuenCodigo = cuenta,
96             IntMovinNumero = nextNum,
97             DttMovFecha = DateTime.Now.Date,
98             ChrEmplCodigo = EmpleadoDefault,
99             ChrTipoCodigo = TipoDeposito,
100            DecMovImporte = monto
101          };
102          context.Movimientos.Add(mov);
103          context.SaveChanges();
104          transaction.Commit();
105          return true;
106        }
107      catch
108      {
109        transaction.Rollback();
110        return false;
111      }
112    }
113  }
114
115 }
```

Figura 23 Inicio del servidor

Se verifica si se levantó correctamente el servidor.

The screenshot shows a web browser window with the URL `localhost:9090`. The page displays a directory listing of files and folders in the root directory, including `App_Data`, `App_Start`, `bin`, `Controllers`, `DataContracts.cs`, `Global.asax`, `Global.asax.cs`, `Interfaces`, `Models`, `obj`, `packages.config`, `Properties`, `Services`, `Web.config`, `Web.Debug.config`, `Web.Release.config`, `WS_EuBank_RESTDOTNET_GR05.csproj`, and `WS_EuBank_RESTDOTNET_GR05.csproj.user`.

Timestamp	File/Folder
11/8/2025 4:07 PM	<dir> App_Data
12/5/2025 2:46 PM	<dir> App_Start
12/7/2025 11:48 AM	<dir> bin
12/5/2025 2:46 PM	<dir> Controllers
11/9/2025 10:52 AM	1886 DataContracts.cs
11/8/2025 9:02 PM	115 Global.asax
11/8/2025 9:03 PM	340 Global.asax.cs
12/5/2025 2:46 PM	<dir> Interfaces
12/5/2025 2:46 PM	<dir> Models
12/5/2025 2:46 PM	<dir> obj
11/8/2025 9:02 PM	2412 packages.config
12/5/2025 2:46 PM	<dir> Properties
12/5/2025 2:46 PM	<dir> Services
12/7/2025 11:40 AM	3990 Web.config
11/8/2025 4:07 PM	1300 Web.Debug.config
11/8/2025 4:07 PM	1361 Web.Release.config
11/12/2025 2:12 AM	15245 WS_EuBank_RESTDOTNET_GR05.csproj
11/8/2025 4:07 PM	1458 WS_EuBank_RESTDOTNET_GR05.csproj.user

Figura 24 Verificación de los servicios web creados

4.3 CREACIÓN DEL PROYECTO CONSOLA

Se crea una aplicación consola.

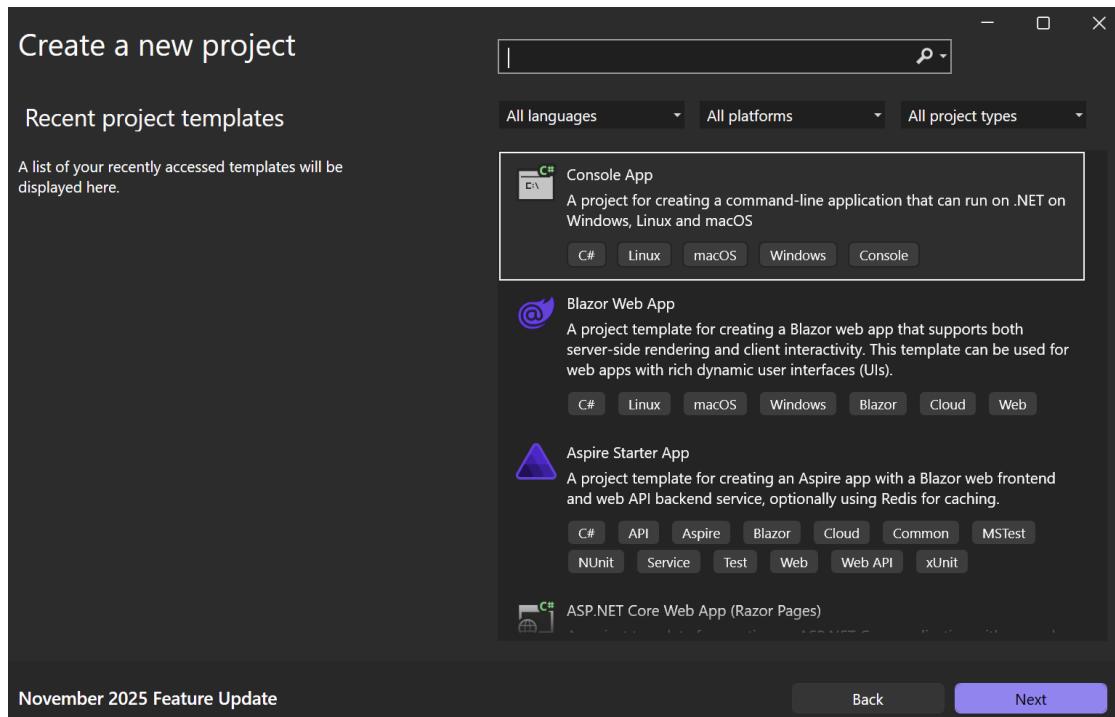


Figura 25 Creación Dotnet Consola

Realizado el servidor se avanza a la aplicación cliente de tipo Consola.

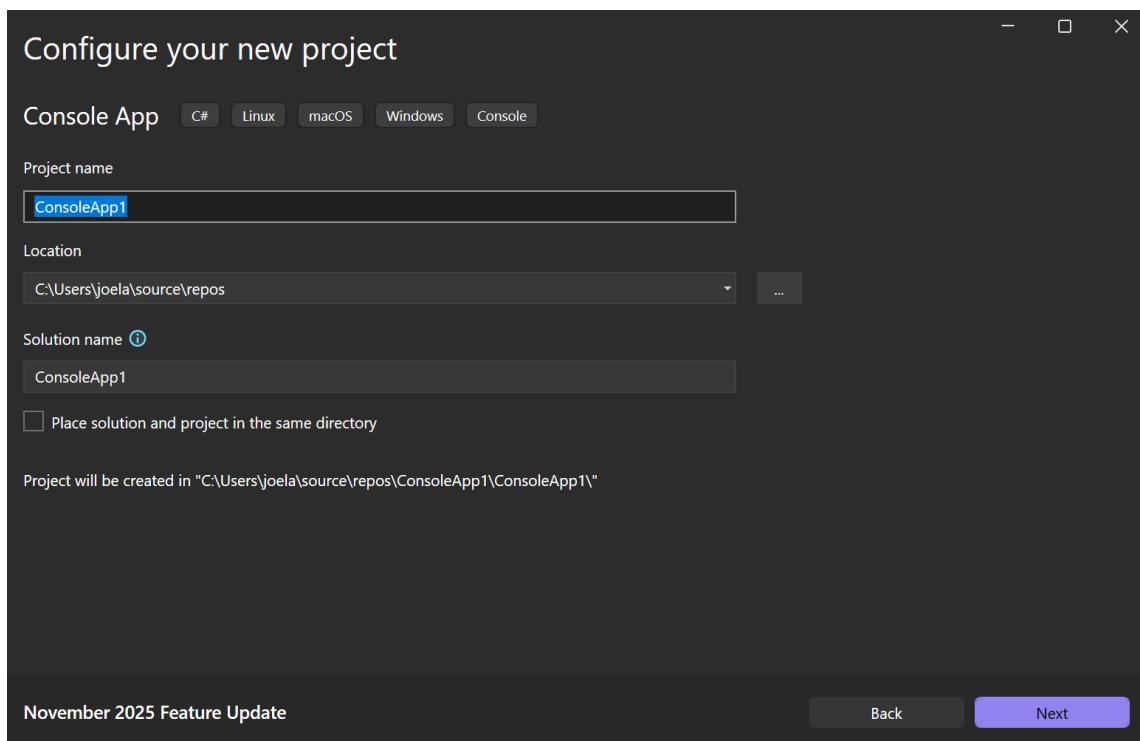


Figura 26 Creación proyecto estructura general

4.3.1 ESTRUCTURA MVC

CREACIÓN DE PAQUETE CONTROLADOR

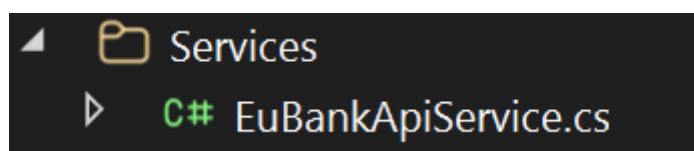


Figura 27 Creación paquete ec.edu.monster.controlador

CREACIÓN DE PACKAGE SERVICIO

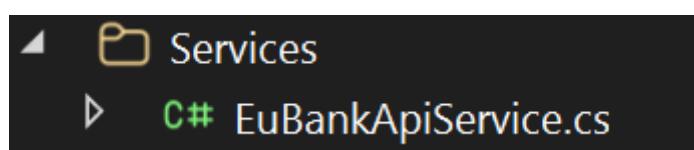


Figura 28 Creación paquete ec.edu.monster.servicio

CREACIÓN DE PACKAGE MODELO

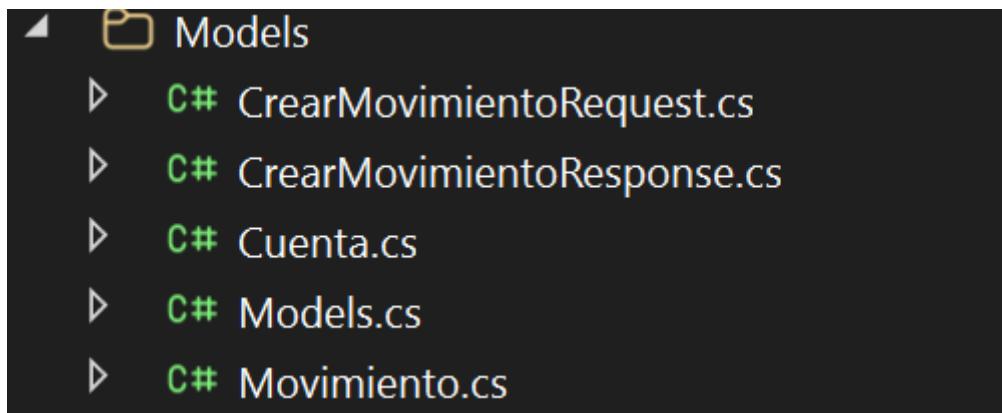


Figura 29 Creación paquete ec.edu.monster.modelo

PROYECTO ESTRUCTURADO

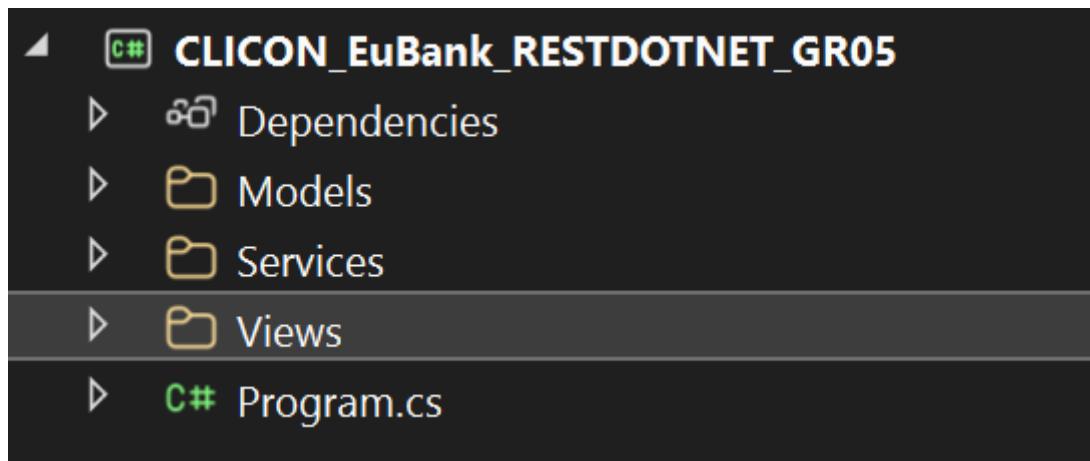


Figura 30 Estructura proyecto cliente

4.3.2 CODIFICACIÓN DE CLASES

Habiendo obtenido la conexión se crea una nueva clase dentro del paquete modelo llamada Movimiento.

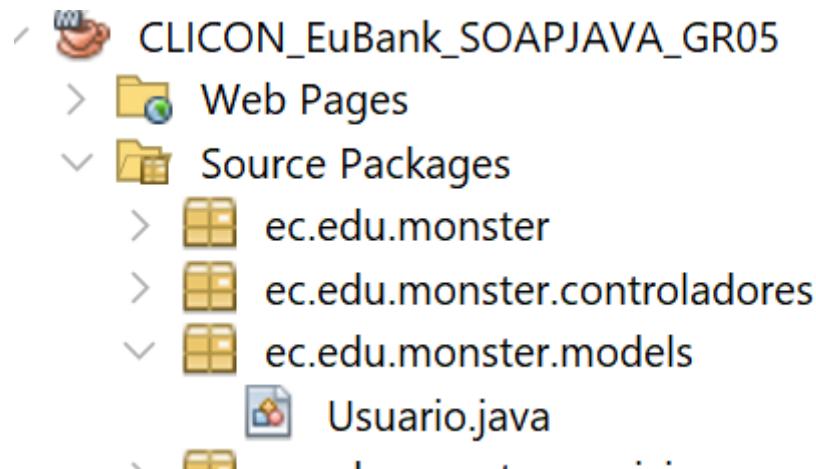


Figura 31 Creación clase Movimiento

Se procede a codificar la nueva clase llamada Movimiento la codificación es la misma que en el proyecto Cliente Escritorio.

```
1  namespace CLICON_EuBank_RESTDOTNET_GR05.Models;
2
3  public class Movimiento
4  {
5      public string? CodigoCuenta { get; set; }
6      public string? NombreCliente { get; set; }
7      public string? Numero { get; set; }
8      public DateTime Fecha { get; set; }
9      public string? Tipo { get; set; }
10     public string? Referencia { get; set; }
11     public decimal Importe { get; set; }
12     public decimal SaldoActual { get; set; }
13 }
```

Figura 32 Codificación clase Movimiento

Se procede a implementar los servicios.

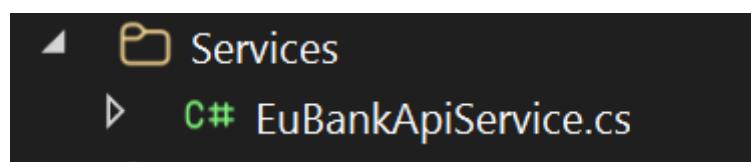


Figura 33 Creación clase EurekaService

```
1      using System.Net.Http.Json;
2      using System.Text.Json;
3      using System.Xml.Serialization;
4      using System.IO;
5      using CLIWEB_EuBank_RESTDOTNET_GR05.Models;
6      using System.Net.Http;
7      using System.Threading.Tasks;
8
9      namespace CLIWEB_EuBank_RESTDOTNET_GR05.Services
10     {
11         2 references
12         public class EuBank ApiService
13         {
14             private readonly HttpClient _httpClient;
15             private readonly string _baseUrl = "http://localhost:62278/api"; // Adjust the base URL as needed
16
17             1 reference
18             public EuBank ApiService(HttpClient httpClient)
19             {
20                 _httpClient = httpClient;
21             }
22
23             1 reference
24             public async Task<bool> LoginAsync(LoginRequest request)
25             {
26                 try
27                 {
28                     var response = await _httpClient.PostAsJsonAsync($"{_baseUrl}/login", request);
29                     if (response.IsSuccessStatusCode)
30                     {
31                         var result = await response.Content.ReadFromJsonAsync<LoginResponse>();
32                         return result?.success ?? false;
33                     }
34                 }
35                 catch
36                 {
37                     return false;
38                 }
39             }
40         }
41     }
42 }
```

Figura 34 Codificación clase EurekaService

CREACIÓN DE LA VISTA

Se crea una clase principal donde se maneja todos los movimientos de EurekaBank.

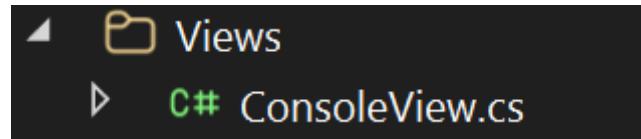


Figura 35 Creación clase ConsoleView

A continuación se codifica lo siguiente

```
1  using CLIWEB_EuBank_RESTDOTNET_GR05.Models;
2  using System;
3
4  namespace CLICON_EuBank_RESTDOTNET_GR05.Views;
5
6  public class ConsoleView
7  {
8      public void ShowWelcome()
9      {
10         Console.ForegroundColor = ConsoleColor.Blue;
11         Console.WriteLine("-----");
12         Console.WriteLine("Cliente Consola EuBank - GR05");
13         Console.WriteLine("Sistema Bancario Avanzado");
14         Console.WriteLine("IMPULSADO POR - REST");
15         Console.WriteLine("-----");
16         Console.ResetColor();
17     }
18
19     public (string? usuario, string? clave) GetLoginCredentials()
20     {
21         Console.ForegroundColor = ConsoleColor.Magenta;
22         Console.WriteLine("-----");
23         Console.Write("Usuario: ");
24         Console.ResetColor();
25         string? usuario = Console.ReadLine();
26         Console.ForegroundColor = ConsoleColor.Magenta;
27         Console.Write("Clave: ");
28         Console.ResetColor();
29         string? clave = Console.ReadLine();
30         Console.ForegroundColor = ConsoleColor.Magenta;
31         Console.WriteLine("-----");
32         Console.ResetColor();
33         return (usuario, clave);
34     }
35
```

Figura 36 Codificación clase WSEurekaClient

TABLA 4 Codificación Clase WSEurekaClient

```
using CLIWEB_EuBank_RESTDOTNET_GR05.Models;
using System;

namespace CLICON_EuBank_RESTDOTNET_GR05.Views;

public class ConsoleView
{
    public void ShowWelcome()
    {
        Console.ForegroundColor = ConsoleColor.Blue;
        Console.WriteLine("=====");
        Console.WriteLine("Cliente Consola EuBank - GR05");
        Console.WriteLine("Sistema Bancario Avanzado");
        Console.WriteLine("IMPULSADO POR - REST");
        Console.WriteLine("=====");
        Console.ResetColor();
    }

    public (string? usuario, string? clave) GetLoginCredentials()
    {
        Console.ForegroundColor = ConsoleColor.Magenta;
        Console.WriteLine("-----");
        Console.Write("Usuario: ");
        Console.ResetColor();
        string? usuario = Console.ReadLine();
```

```
Console.ForegroundColor = ConsoleColor.Magenta;
Console.Write("Clave: ");
Console.ResetColor();
string? clave = Console.ReadLine();
Console.ForegroundColor = ConsoleColor.Magenta;
Console.WriteLine("-----");
Console.ResetColor();
return (usuario, clave);
}

public void ShowLoginSuccess(string user)
{
    Console.ForegroundColor = ConsoleColor.Green;
    Console.WriteLine($"Bienvenido, {user}!");
    Console.WriteLine("=====");
    Console.ResetColor();
}

public void ShowLoginError(string message)
{
    Console.ForegroundColor = ConsoleColor.Red;
    Console.WriteLine($"Error: {message}");
    Console.ResetColor();
}

public void ShowError(string message)
{
    Console.ForegroundColor = ConsoleColor.Red;
    Console.WriteLine($"Error: {message}");
    Console.ResetColor();
}
```

```
}

public int ShowDashboardMenu()
{
    Console.WriteLine();
    Console.ForegroundColor = ConsoleColor.Magenta;
    Console.WriteLine("==> Dashboard <==");
    Console.WriteLine("=====");
    Console.ResetColor();
    Console.WriteLine("1. Ver Cuentas");
    Console.WriteLine("2. Ver Movimientos");
    Console.WriteLine("3. Crear Movimiento");
    Console.WriteLine("4. Salir");
    Console.WriteLine("=====");
    Console.ForegroundColor = ConsoleColor.Magenta;
    Console.Write("Opcion: ");
    Console.ResetColor();
    string? input = Console.ReadLine();
    return int.TryParse(input, out int opcion) ? opcion : 0;
}

public void DisplayCuentas(List<Account> cuentas)
{
    Console.WriteLine();
    if (cuentas.Any())
    {
        Console.ForegroundColor = ConsoleColor.Cyan;
        Console.WriteLine("Lista de Cuentas:");
        Console.WriteLine("=====");
        Console.ResetColor();
    }
}
```

```

Console.ForegroundColor = ConsoleColor.Yellow;
string separator = new string('-', 112);
Console.WriteLine(separator);
Console.WriteLine("Codigo | Cliente | Email | Telefono | Moneda | Saldo | Estado");
Console.WriteLine(separator);
Console.ResetColor();
foreach (var c in cuentas)
{
    Console.WriteLine(
        $"'{c.codigo,-7}' | " +
        $"'{Truncate(c.nombreCliente, 28),-28}' | " +
        $"'{Truncate(c.emailCliente, 23),-23}' | " +
        $"'{Truncate(c.telefonoCliente, 11),-11}' | " +
        $"'{c.moneda,-6}' | " +
        $"'{c.saldo,8:F2}' | " +
        $"'{c.estado,-7}'");
}
Console.ForegroundColor = ConsoleColor.Yellow;
Console.WriteLine(separator);
Console.ResetColor();
}
else
{
    Console.ForegroundColor = ConsoleColor.Red;
    Console.WriteLine("No hay cuentas disponibles");
    Console.ResetColor();
}
Console.WriteLine();
}
private static string Truncate(string? value, int maxLength)

```

```

{
    if (string.IsNullOrEmpty(value))
    {
        return "";
    }

    return value.Length <= maxLength ? value : value.Substring(0, maxLength);
}

public string? GetMovimientoFilter()
{
    Console.ForegroundColor = ConsoleColor.Magenta;
    Console.Write("Filtrar por cuenta (dejar vacío para todas): ");
    Console.ResetColor();
    return Console.ReadLine();
}

public void DisplayMovimientos(List<Movement> movimientos)
{
    Console.WriteLine();
    if (movimientos.Any())
    {
        Console.ForegroundColor = ConsoleColor.Cyan;
        Console.WriteLine("Lista de Movimientos:");
        Console.WriteLine("=====");
        Console.ResetColor();
        Console.ForegroundColor = ConsoleColor.Yellow;
        string separator = new string('-', 112);
        Console.WriteLine(separator);
        Console.WriteLine("Cuenta           | Numero | Fecha | Tipo | Referencia | Importe | Saldo Actual");

```

```
Console.WriteLine(separator);
Console.ResetColor();
var sortedMovimientos = movimientos.OrderByDescending(m => m.numero).ToList();
foreach (var m in sortedMovimientos)
{
    string cuenta = $"{m.codigoCuenta} - {m.nombreCliente}";

    Console.WriteLine(
        ${Truncate(cuenta, 28),-28} | " +
        ${m.numero,-6} | " +
        ${m.fecha:dd/MM/yyyy} | " +
        ${Truncate(m.tipo, 14),-14} | " +
        ${Truncate(m.referencia ?? "-", 10),-10} | " +
        ${m.importe,8:F2} | " +
        ${m.saldoActual,10:F2}");
}
Console.ForegroundColor = ConsoleColor.Yellow;
Console.WriteLine(separator);
Console.ResetColor();
}
else
{
    Console.ForegroundColor = ConsoleColor.Red;
    Console.WriteLine("No hay movimientos");
    Console.ResetColor();
}
Console.WriteLine();
}

public MovementRequest? GetCrearMovimientoData(List<string> tipos)
```

```
{  
    Console.WriteLine();  
    Console.ForegroundColor = ConsoleColor.Cyan;  
    Console.WriteLine("Crear Nuevo Movimiento");  
    Console.WriteLine("=====");  
    Console.ResetColor();  
    Console.WriteLine("Tipos disponibles:");  
    for (int i = 0; i < tipos.Count; i++)  
    {  
        Console.WriteLine($"{i + 1}. {tipos[i]}");  
    }  
  
    Console.ForegroundColor = ConsoleColor.Magenta;  
    Console.Write("Seleccione tipo (numero): ");  
    Console.ResetColor();  
    if (!int.TryParse(Console.ReadLine(), out int tipoIndex) || tipoIndex < 1 || tipoIndex > tipos.Count)  
    {  
        Console.ForegroundColor = ConsoleColor.Red;  
        Console.WriteLine("Seleccion invalida");  
        Console.ResetColor();  
        return null;  
    }  
  
    string tipo = tipos[tipoIndex - 1];  
  
    Console.ForegroundColor = ConsoleColor.Magenta;  
    Console.Write("Cuenta Origen: ");  
    Console.ResetColor();  
    string? origen = Console.ReadLine();  
    if (string.IsNullOrEmpty(origen))
```

```
{  
    Console.ForegroundColor = ConsoleColor.Red;  
    Console.WriteLine("Cuenta origen requerida");  
    Console.ResetColor();  
    return null;  
}  
  
string? destino = null;  
if (tipo == "TRANSFERENCIA")  
{  
    Console.ForegroundColor = ConsoleColor.Magenta;  
    Console.Write("Cuenta Destino: ");  
    Console.ResetColor();  
    destino = Console.ReadLine();  
    if (string.IsNullOrEmpty(destino))  
    {  
        Console.ForegroundColor = ConsoleColor.Red;  
        Console.WriteLine("Cuenta destino requerida para transferencias");  
        Console.ResetColor();  
        return null;  
    }  
}  
  
Console.ForegroundColor = ConsoleColor.Magenta;  
Console.Write("Monto: ");  
Console.ResetColor();  
if (!decimal.TryParse(Console.ReadLine(), out decimal importe) || importe <= 0)  
{  
    Console.ForegroundColor = ConsoleColor.Red;  
    Console.WriteLine("Monto invalido");  
}
```

```
Console.ResetColor();
return null;
}

return new MovementRequest { tipo = tipo, cuentaOrigen = origen, cuentaDestino = destino, importe = importe };
}

public void ShowMessage(string message)
{
    Console.WriteLine(message);
}
}
```

4.3.4 EJECUCIÓN

Para la ejecución el servidor debe estar corriendo.

```

C:\Users\joela\Documents\01> + - x

=====
== Dashboard ==
=====
1. Ver Cuentas
2. Ver Movimientos
3. Crear Movimiento
4. Salir
=====
Opcion: 1

Lista de Cuentas:
=====

Código | Cliente | Email | Teléfono | Moneda | Saldo | Estado
-----+-----+-----+-----+-----+-----+-----+
00100001 | ALAN ALBERTO ARANDA LUNA | a.aranda@hotmail.com | 834-67125 | 01 | 6900.00 | ACTIVO
00100002 | ALAN ALBERTO ARANDA LUNA | a.aranda@hotmail.com | 834-67125 | 02 | 4500.00 | ACTIVO
00200001 | ROSA LIZET FLORES CHAFLOQUE | r.florez@hotmail.com | 966-87567 | 01 | 7000.00 | ACTIVO
00200002 | ERIC GUSTAVO CORONEL CASTILL | gcoronel@viabcp.com | 9666-4457 | 01 | 6800.00 | ACTIVO
00200003 | EDGAR RAFAEL CHAVEZ CANALES | e.chavez@gmail.com | 999-96673 | 02 | 6000.00 | ACTIVO
00300001 | GABRIEL ALEJANDRO GONZALES G | g.gonzales@yahoo.es | 945-56782 | 01 | 0.00 | CANCELADO
=====

===== Dashboard =====
=====
1. Ver Cuentas
2. Ver Movimientos
3. Crear Movimiento

```

Figura 37 Ejecución clase WSEurekaClient

```

C:\Users\joela\Documents\01> + - x

Lista de Movimientos:
=====

Cuenta | Número | Fecha | Tipo | Referencia | Importe | Saldo Actual
-----+-----+-----+-----+-----+-----+-----+
00200001 - ROSA LIZET FLORES | 15 | 19/02/2008 | Retiro | - | -1000.00 | 7000.00
00200001 - ROSA LIZET FLORES | 14 | 13/02/2008 | Depósito | - | 2000.00 | 8000.00
00200001 - ROSA LIZET FLORES | 13 | 08/02/2008 | Retiro | - | -4000.00 | 6000.00
00200001 - ROSA LIZET FLORES | 12 | 04/02/2008 | Depósito | - | 2000.00 | 10000.00
00200001 - ROSA LIZET FLORES | 11 | 30/01/2008 | Retiro | - | -3000.00 | 8000.00
00200001 - ROSA LIZET FLORES | 10 | 27/01/2008 | Retiro | - | -1000.00 | 11000.00
00200001 - ROSA LIZET FLORES | 9 | 23/01/2008 | Depósito | - | 7000.00 | 12000.00
00200001 - ROSA LIZET FLORES | 8 | 21/01/2008 | Retiro | - | -3000.00 | 5000.00
00100001 - ALAN ALBERTO ARAN | 7 | 15/03/2008 | Depósito | - | 1000.00 | 6900.00
00200001 - ROSA LIZET FLORES | 7 | 19/01/2008 | Depósito | - | 2000.00 | 8000.00
00200003 - EDGAR RAFAEL CHAV | 6 | 11/03/2008 | Retiro | - | -500.00 | 6000.00
00100001 - ALAN ALBERTO ARAN | 6 | 03/03/2008 | Retiro | - | -800.00 | 5900.00
00200001 - ROSA LIZET FLORES | 6 | 15/01/2008 | Retiro | - | -4000.00 | 6000.00
00100001 - ALAN ALBERTO ARAN | 5 | 25/02/2008 | Retiro | - | -500.00 | 6700.00
00200003 - EDGAR RAFAEL CHAV | 5 | 25/02/2008 | Depósito | - | 3500.00 | 6500.00
00200001 - ROSA LIZET FLORES | 5 | 13/01/2008 | Depósito | - | 2000.00 | 10000.00
00100002 - ALAN ALBERTO ARAN | 4 | 08/03/2008 | Depósito | - | 1500.00 | 4500.00
00100001 - ALAN ALBERTO ARAN | 4 | 14/02/2008 | Depósito | - | 2000.00 | 7200.00
00200003 - EDGAR RAFAEL CHAV | 4 | 09/02/2008 | Retiro | - | -500.00 | 3000.00
00200001 - ROSA LIZET FLORES | 4 | 11/01/2008 | Depósito | - | 1000.00 | 8000.00
00200002 - ERIC GUSTAVO CORO | 3 | 06/03/2008 | Retiro | - | -1200.00 | 6800.00
00100002 - ALAN ALBERTO ARAN | 3 | 13/02/2008 | Depósito | - | 2200.00 | 3000.00
00300001 - GABRIEL ALEJANDRO | 3 | 25/01/2008 | Cancelar Cuent | - | -7000.00 | 0.00
00200003 - EDGAR RAFAEL CHAV | 3 | 20/01/2008 | Retiro | - | -500.00 | 3500.00
=====
```

Figura 38 Ejecución clase WSEurekaClient

4.4 CREACIÓN DEL PROYECTO CLIENTE ESCRITORIO

Se crea con la misma lógica que la de consola

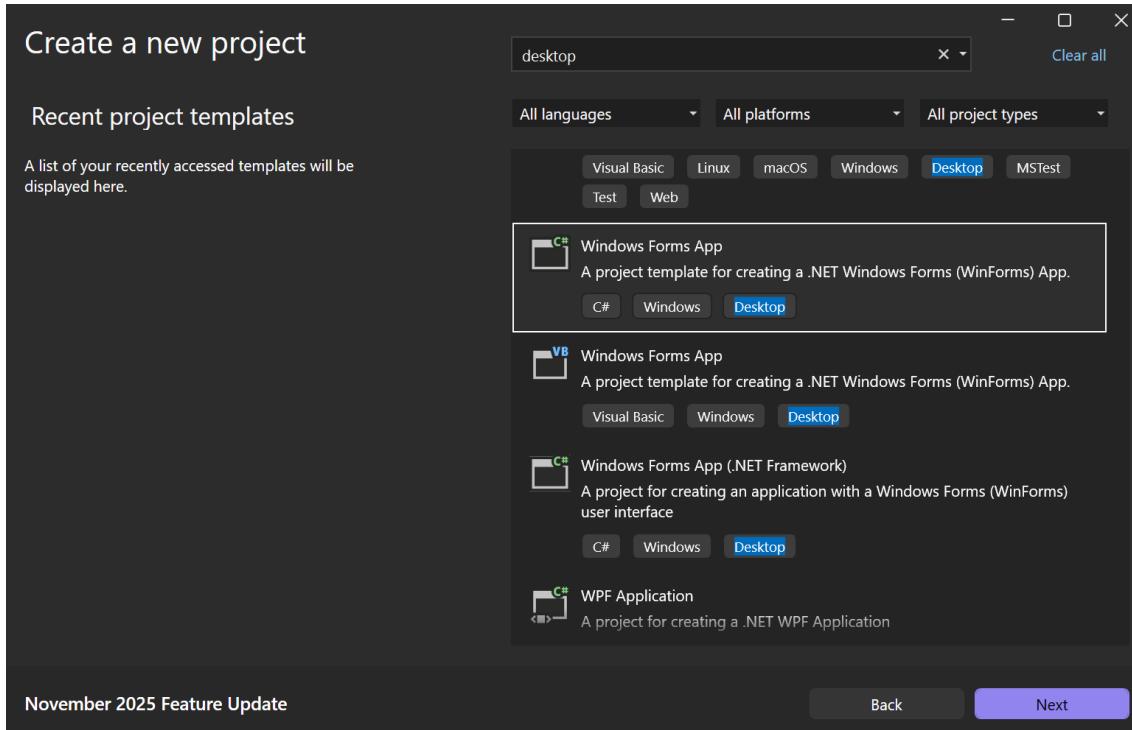


Figura 39 Creación del proyecto cliente

Se crea un cliente de servicio web.

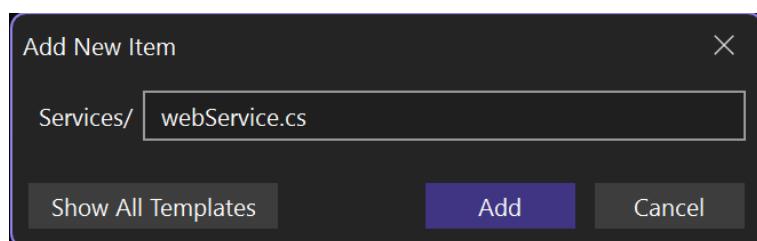


Figura 40 Creación del cliente del servicio web

Tomar en cuenta que el proyecto servidor debe estar desplegado.

Si todo va bien se muestra la siguiente estructura.

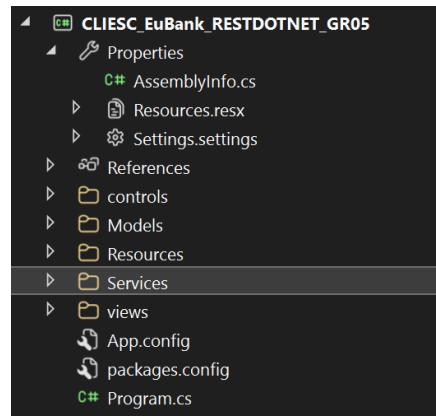


Figura 41 Estructura inicial del proyecto

4.4.1 CREACIÓN DE LA CAPA DE SERVICIO

Se crea la capa de servicio para llamar los métodos de nuestro Servidor SOAP.

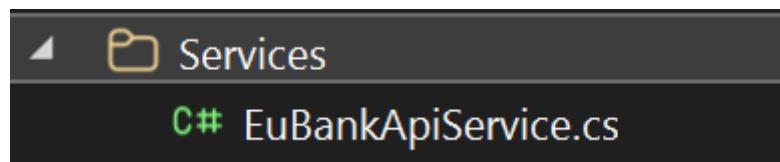


Figura 42 Creación de la clase EurekaService

Una vez creada la clase se procede a agregar las operaciones del servicio web dentro del servicio creado anteriormente:

```

1  using System.Net.Http;
2  using System.Text.Json;
3  using System.Net.Http;
4  using System.Threading.Tasks;
5  using CLIEBC_EuBank_RESTDOTNET_GR05.Models;
6  using System.Collections.Generic;
7  using System;
8
9  namespace CLIEBC_EuBank_RESTDOTNET_GR05.Services
10 {
11     // 9 references
12     public class EuBankApiService
13     {
14         // private readonly HttpClient _httpClient;
15         // private readonly string _baseUrl = "http://localhost:9090/api"; // Adjust the base URL as needed
16
17         // 4 references
18         public EuBankApiService(HttpClient httpClient)
19         {
20             _httpClient = httpClient;
21         }
22
23         // 1 reference
24         public async Task<bool> LoginAsync(LoginRequest request)
25         {
26             try
27             {
28                 var response = await _httpClient.PostAsJsonAsync($"{_baseUrl}/login", request);
29                 if (response.IsSuccessStatusCode)
30                 {
31                     var result = await response.Content.ReadFromJsonAsync<LoginResponse>();
32                     return result != null && result.success;
33                 }
34                 return false;
35             }
36             catch
37             {
38                 return false;
39             }
40         }
41     }
42 }

```

Figura 43 Llamada a los métodos del servicio web

TABLA 5 Clase llamada a servicios web

```
using System.Net.Http.Json;
using System.Text.Json;
using System.Net.Http;
using System.Threading.Tasks;
using CLIWEB_EuBank_RESTDOTNET_GR05.Models;
using System.Collections.Generic;
using System;

namespace CLIESC_EuBank_RESTDOTNET_GR05.Services
{
    public class EuBank ApiService
    {
        private readonly HttpClient _httpClient;
        private readonly string _baseUrl = "http://localhost:9090/api"; // Adjust the base URL as needed

        public EuBank ApiService(HttpClient httpClient)
        {
            _httpClient = httpClient;
        }
    }
}
```

```
public async Task<bool> LoginAsync(LoginRequest request)
{
    try
    {
        var response = await _httpClient.PostAsJsonAsync($"{_baseUrl}/login", request);
        if (response.IsSuccessStatusCode)
        {
            var result = await response.Content.ReadFromJsonAsync<LoginResponse>();
            return result != null && result.success;
        }
        return false;
    }
    catch
    {
        return false;
    }
}

public async Task<List<Account>> GetAllAccountsAsync()
{
    try
```

```
{  
    var response = await _httpClient.GetAsync($"{_baseUrl}/cuentas");  
    if (response.IsSuccessStatusCode)  
    {  
        return await response.Content.ReadFromJsonAsync<List<Account>>();  
    }  
    return null;  
}  
catch  
{  
    return null;  
}  
}  
  
public async Task<Account> GetAccountDetailsAsync(string accountCode)  
{  
    try  
    {  
        var response = await _httpClient.GetAsync($"{_baseUrl}/cuentas/{accountCode}");  
        if (response.IsSuccessStatusCode)  
        {  
            return await response.Content.ReadFromJsonAsync<Account>();  
        }  
        return null;  
    }  
    catch (Exception ex)  
    {  
        // Handle exception  
    }  
}
```

```
        return await response.Content.ReadFromJsonAsync<Account>();
    }
    return null;
}
catch
{
    return null;
}
}

public async Task<List<Movement>> GetMovementsAsync(string accountCode = null)
{
    try
    {
        var url = $"{_baseUrl}/movimientos";
        if (!string.IsNullOrEmpty(accountCode))
        {
            url += "?cuenta={accountCode}";
        }
        var response = await _httpClient.GetAsync(url);
        if (response.IsSuccessStatusCode)
```

```
{  
    return await response.Content.ReadFromJsonAsync<List<Movement>>();  
}  
return null;  
}  
catch  
{  
    return null;  
}  
}  
  
public async Task<bool> ProcessMovementAsync(MovementRequest request)  
{  
    try  
    {  
        Console.WriteLine($"[PROCESS] POST {_baseUrl}/movimientos");  
        Console.WriteLine($"[PROCESS] Request: {JsonSerializer.Serialize(request)}");  
        var response = await _httpClient.PostAsJsonAsync($"{_baseUrl}/movimientos", request);  
        Console.WriteLine($"[PROCESS] Status: {response.StatusCode}");  
        if (response.IsSuccessStatusCode)  
    }  
}
```

```
        var content = await response.Content.ReadAsStringAsync();
        Console.WriteLine($"[PROCESS] Response: {content}");
    }
    else
    {
        var error = await response.Content.ReadAsStringAsync();
        Console.WriteLine($"[PROCESS] Error Response: {error}");
    }
    return response.IsSuccessStatusCode;
}
catch (Exception ex)
{
    Console.WriteLine($"[PROCESS] Error: {ex.Message}");
    return false;
}
}

public async Task<List<string>> GetMovementTypesAsync()
{
    try
    {
```

```
var response = await _httpClient.GetAsync($"{_baseUrl}/tiposmovimiento");
if (response.IsSuccessStatusCode)
{
    return await response.Content.ReadFromJsonAsync<List<string>>();
}
return null;
}
catch
{
    return null;
}
}
}
```

4.4.4 CREACIÓN DE LA VISTA

Para la creación de la vista es necesario realizar 4 formularios para el manejo de todo el proyecto

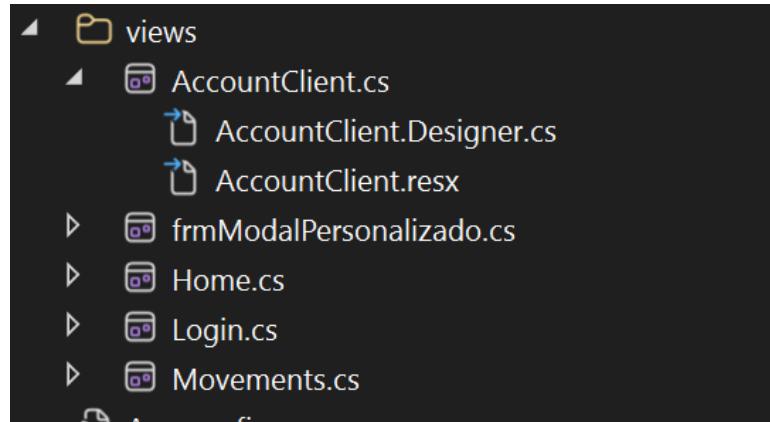


Figura 44 Creación de las vistas

Se diseña primero el login



Figura 45 Prueba a la interfaz gráfica principal

The screenshot shows the 'Gestión de Cuentas' (Account Management) section of the Eureka Bank application. On the left, there's a sidebar with 'Cuentas de Clientes' and 'Movimientos' buttons. The main area displays a table of client accounts with columns for Código, Cliente, and Email. At the bottom left is a 'Cerrar Sesión' button.

Código	Cliente	Email
00100001	ALAN ALBERTO ARANDA LUNA	a.aranda@hotmail.com
00100002	ALAN ALBERTO ARANDA LUNA	a.aranda@hotmail.com
00200001	ROSA LIZET FLORES CHAFLOQUE	r.florez@hotmail.com
00200002	ERIC GUSTAVO CORONEL CASTILLO	gcoronel@viabcp.com
00200003	EDGAR RAFAEL CHAVEZ CANALES	e.chavez@gmail.com
00300001	GABRIEL ALEJANDRO GONZALES GARCIA	g.gonzales@yahoo.es

Cerrar Sesión

Figura 46 Prueba a la interfaz gráfica de gestión de cuentas

The screenshot shows the 'Movimientos Bancarios' (Banking Movements) section of the Eureka Bank application. The sidebar has 'Cuentas de Clientes' and 'Movimientos' buttons. The main area displays a table of banking movements with columns for Número, Fecha, Tipo, Importe, Referencia, and Detalle. A '+ Crear' button is located at the top right.

Número	Fecha	Tipo	Importe	Referencia	Detalle
7	15/03/2008 00:00	Deposito	1,000.00	O	
6	11/03/2008 00:00	Retiro	-500.00	O	
4	08/03/2008 00:00	Deposito	1,500.00	O	
3	06/03/2008 00:00	Retiro	-1,200.00	O	
6	03/03/2008 00:00	Retiro	-800.00	O	
5	25/02/2008 00:00	Retiro	-500.00	O	
5	25/02/2008 00:00	Deposito	3,500.00	O	
15	19/02/2008 00:00	Retiro	-1,000.00	O	
4	14/02/2008 00:00	Deposito	2,000.00	O	
14	13/02/2008 00:00	Deposito	2,000.00	O	
3	13/02/2008 00:00	Deposito	2,200.00	O	
4	09/02/2008 00:00	Retiro	-500.00	O	

Cerrar Sesión

Figura 47 Prueba a la interfaz gráfica de consulta de movimientos bancarios

TABLA 6 ConsultaView

```
package ec.edu.monster.vista;

import ec.edu.monster.servicios.EuBankService;
import ec.edu.monster.ws.eurekabank.DatosCuenta;
import ec.edu.monster.ws.eurekabank.MovimientoData;
import java.text.DecimalFormat;
import java.text.DecimalFormatSymbols;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
```

```
*  
* @author Dome  
*/  
  
public class Movements extends javax.swing.JPanel {  
  
    private EuBankService euBankService = new EuBankService();  
  
    private final DecimalFormat decimalFormat;  
  
    public Movements() {  
        initComponents();  
  
        DecimalFormatSymbols symbols = new DecimalFormatSymbols(Locale.US);  
  
        this.decimalFormat = new DecimalFormat("#,##0.00", symbols);  
  
        cargarDatosMovimientos();  
    }  
  
    // </editor-fold>
```

```
public void setEuBankService(EuBankService euBankService) {  
    this.euBankService = euBankService;  
    // Ahora que tenemos el servicio, cargamos los datos  
}  
  
@SuppressWarnings("unchecked")  
// <editor-fold defaultstate="collapsed" desc="Generated Code">  
private void initComponents() {  
  
    jPanel1 = new javax.swing.JPanel();  
    modernPanel2 = new ec.edu.monster.vista.ModernPanel();  
    lblimagesulli = new javax.swing.JLabel();  
    lblTitle = new javax.swing.JLabel();  
    lblTitle1 = new javax.swing.JLabel();  
    btn_crear = new ec.edu.monster.vista.ModernButton();  
    tableMovements = new ec.edu.monster.vista.ModernTable();
```

```
setBackground(new java.awt.Color(255, 255, 255));
setPreferredSize(new java.awt.Dimension(680, 570));
setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

jPanel1.setBackground(new java.awt.Color(255, 255, 255));

modernPanel2.setBackground(new java.awt.Color(219, 240, 245));
modernPanel2.setBorderColor(new java.awt.Color(255, 255, 255));
modernPanel2.setPanelBackground(new java.awt.Color(235, 247, 248));

lblimagesulli.setIcon(new javax.swing.ImageIcon(getClass().getResource("/SulliGeneral.png"))); // NOI18N
lblimagesulli.setVerticalAlignment(javax.swing.SwingConstants.TOP);

lblTitle.setFont(new java.awt.Font("Britannic Bold", 0, 24)); // NOI18N
lblTitle.setForeground(new java.awt.Color(140, 201, 221));
```

```
lblTitle.setText("Movimientos Bancarios");

lblTitle1.setFont(new java.awt.Font("Roboto", 0, 12)); // NOI18N
lblTitle1.setForeground(new java.awt.Color(102, 102, 102));
lblTitle1.setText("Revisa los movimientos realizados entre las cuentas bancarias");

btn_crear.setBackground(new java.awt.Color(153, 214, 234));
btn_crear.setBorder(null);
btn_crear.setBorderColor(new java.awt.Color(219, 240, 245));
btn_crear.setBorderThickness(0);
btn_crear.setHoverColor(new java.awt.Color(87, 175, 204));
btn_crear.setText("+ Crear");
btn_crear.setFont(new java.awt.Font("Roboto", 1, 14)); // NOI18N
btn_crear.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        btn_crearMouseClicked(evt);
    }
});
```

```
};

btn_crear.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        btn_crearActionPerformed(evt);

    }

});

javax.swing.GroupLayout modernPanel2Layout = new javax.swing.GroupLayout(modernPanel2);

modernPanel2.setLayout(modernPanel2Layout);

modernPanel2Layout.setHorizontalGroup

    modernPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

    .addGroup(modernPanel2Layout.createSequentialGroup()

        .addContainerGap()

        .addComponent(lblimagesulli)

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
.addGroup(modernPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(lblTitle1)
    .addComponent(lblTitle))
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 28, Short.MAX_VALUE)
    .addComponent(btn_crear, javax.swing.GroupLayout.PREFERRED_SIZE, 102, javax.swing.GroupLayout.PREFERRED_SIZE)
    .addGap(20, 20, 20))
);

modernPanel2Layout.setVerticalGroup(
    modernPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(lblimagesulli, javax.swing.GroupLayout.Alignment.TRAILING)
    .addGroup(modernPanel2Layout.createSequentialGroup()
        .addGap(23, 23, 23)
        .addGroup(modernPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
            .addComponent(btn_crear, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addGroup(modernPanel2Layout.createSequentialGroup()
                .addComponent(lblTitle)
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(lblTitle1))

.addGap(16, 16, 16))

);

tableMovements.setBackground(new java.awt.Color(255, 255, 255));

tableMovements.setBorder(null);

tableMovements.setBorderRadius(0);

tableMovements.setContainerBackground(new java.awt.Color(255, 255, 255));

tableMovements.setEvenRowColor(new java.awt.Color(242, 250, 253));

tableMovements.setHeaderBackground(new java.awt.Color(153, 214, 234));

tableMovements.setHoverRowColor(new java.awt.Color(204, 255, 255));

tableMovements.setFont(new java.awt.Font("Roboto", 0, 14)); // NOI18N

tableMovements.setName(""); // NOI18N

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
```

```
jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
            .addGap(30, 30, 30)
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                .addComponent(tableMovements, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(modernPanel2, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addGap(42, Short.MAX_VALUE))
    );
}

jPanel1Layout.setVerticalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
            .addGap(21, 21, 21)
            .addComponent(modernPanel2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(21, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE)
            .addGap(21, javax.swing.GroupLayout.PREFERRED_SIZE))
    );
}
```

```
.addGap(30, 30, 30)

.addComponent(tableMovements, javax.swing.GroupLayout.PREFERRED_SIZE, 382, javax.swing.GroupLayout.PREFERRED_SIZE)

.addContainerGap(33, Short.MAX_VALUE)

);

add(jPanel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 680, 570));

}// </editor-fold>

private void cargarDatosMovimientos() {

if (this.euBankService == null) {

System.err.println("Error: EuBankService no fue injectado en el panel Movements.");

return;

}

String[] columnNames = {"Cuenta", "Nro", "Fecha", "Tipo", "Importe", "Saldo"};

DefaultTableModel model = new DefaultTableModel(columnNames, 0) {
```

```
@Override  
  
public boolean isCellEditable(int row, int column) { return false; }  
  
};  
  
try {  
  
    // Lógica para traer TODOS los movimientos (igual que en consola)  
  
    List<DatosCuenta> cuentas = this.euBankService.traerCuentasConClientes();  
  
    List<MovimientoData> todosMovimientos = new ArrayList<>();  
  
    for (DatosCuenta cuenta : cuentas) {  
  
        todosMovimientos.addAll(this.euBankService.traerMovimientos(cuenta.getCodigo()));  
  
    }  
  
    // Ordena por fecha y número  
  
    todosMovimientos.sort((m1, m2) -> {  
  
        int resFecha = m2.getFecha().toGregorianCalendar().compareTo(m1.getFecha().toGregorianCalendar());  
  
        if (resFecha != 0) return resFecha;  
    }  
}
```

```
return Integer.compare(m2.getNumero(), m1.getNumero());  
});  
  
// Llena la tabla  
for (MovimientoData mov : todosMovimientos) {  
    Object[] row = new Object[] {  
        mov.getCodigoCuenta(),  
        mov.getNumero(),  
        mov.getFecha().toString().substring(0, 10), // Acorta la fecha  
        mov.getTipo(),  
        decimalFormat.format(mov.getImporte()),  
        decimalFormat.format(mov.getSaldoActual())  
    };  
    model.addRow(row);  
}
```

```
tableMovements.getTable().setModel(model);

} catch (Exception ex) {
    JOptionPane.showMessageDialog(this, "Error al cargar movimientos: " + ex.getMessage(), "Error de Conexión", JOptionPane.ERROR_MESSAGE);
}

}

private void btn_crearMouseClicked(java.awt.event.MouseEvent evt) {

}

private void btn_crearActionPerformed(java.awt.event.ActionEvent evt) {
    // 1. Obtener la ventana padre (el JFrame que contiene este JPanel)
    java.awt.Window parentWindow = javax.swing.SwingUtilities.getWindowAncestor(this);

    // 2. Crear el JDialog modal, pasándole el padre y 'true'
}
```

```
// ¡Aquí usamos el nuevo constructor!

Modal modal = new Modal((java.awt.Frame) parentWindow, true);

// 3. Mostrar el modal. El código se PAUSARÁ aquí

// hasta que el usuario cierre el diálogo.

modal.setVisible(true);

// 4. El código se reanuda AQUÍ cuando el modal se cierra.

String tipoMovimiento = modal.getSeleccion();

// 5. Verificar si el usuario presionó "Crear" (tipoMovimiento no será null)

if (tipoMovimiento != null) {

    // El usuario NO canceló

    String origen = modal.getCuentaOrigen();

    String destino = modal.getCuentaDestino();
```

```
// Imprimir en consola (para probar)

System.out.println("--- Nuevo Movimiento ---");

System.out.println("Tipo: " + tipoMovimiento);

System.out.println("Origen: " + origen);

System.out.println("Destino: " + destino);

// TODO: Aquí es donde agregas los datos a tu 'tableMovements'

// Por ejemplo:

// javax.swing.table.DefaultTableModel model = (javax.swing.table.DefaultTableModel) tableMovements.getModel();

// model.addRow(new Object[]{"ID_NUEVO", tipoMovimiento, origen, destino});

}

} else {

// El usuario presionó "Cancelar"

System.out.println("Operación cancelada.");

}

}
```

```
// Variables declaration - do not modify

private ec.edu.monster.vista.ModernButton btn_crear;

private javax.swing.JPanel jPanel1;

private javax.swing.JLabel lblTitle;

private javax.swing.JLabel lblTitle1;

private javax.swing.JLabel lblimagesulli;

private ec.edu.monster.vista.ModernPanel modernPanel2;

private ec.edu.monster.vista.ModernTable tableMovements;

// End of variables declaration

}
```

TABLA 7 MainView

```
package ec.edu.monster.vista;

import ec.edu.monster.servicios.EuBankService;
import java.awt.BorderLayout;
import java.awt.Color;

/**
 *
 * @author Dome
 */

public class Home extends javax.swing.JFrame {
    private final EuBankService euBankService = new EuBankService();
    int xMouse, yMouse;
    public Home() {
```

```
initComponents();  
  
Clients c = new Clients();  
c.setSize(680, 570);  
c.setLocation(0, 0);  
this.setLocationRelativeTo(null);  
  
Contenedor.removeAll();  
Contenedor.add(c, BorderLayout.CENTER);  
Contenedor.revalidate();  
Contenedor.repaint();  
}  
  
@SuppressWarnings("unchecked")  
// <editor-fold defaultstate="collapsed" desc="Generated Code">  
private void initComponents() {
```

```
Background = new javax.swing.JPanel();

header = new javax.swing.JPanel();

btn_exit = new ec.edu.monster.vista.ModernButton();

Menu = new javax.swing.JPanel();

lblLogoEu = new javax.swing.JLabel();

btn_clients = new ec.edu.monster.vista.ModernButton();

btn_movements = new ec.edu.monster.vista.ModernButton();

btn_exitsession = new ec.edu.monster.vista.ModernButton();

Contenedor = new javax.swing.JPanel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));

 setLocationByPlatform(true);

setUndecorated(true);

setResizable(false);
```

```
setSize(new java.awt.Dimension(850, 640));

Background.setBackground(new java.awt.Color(255, 255, 255));
Background.setToolTipText("");
Background.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
Background.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

header.setBackground(new java.awt.Color(153, 214, 234));
header.addMouseMotionListener(new java.awt.event.MouseMotionAdapter() {
    public void mouseDragged(java.awt.event.MouseEvent evt) {
        headerMouseDragged(evt);
    }
});
header.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mousePressed(java.awt.event.MouseEvent evt) {
        headerMousePressed(evt);
    }
});
```

```
}

});

btn_exit.setBackground(new java.awt.Color(153, 214, 234));
btn_exit.setBorder(null);
btn_exit.setBorderColor(new java.awt.Color(153, 214, 234));
btn_exit.setBorderRadius(0);
btn_exit.setBorderThickness(0);
btn_exit.setHoverColor(new java.awt.Color(255, 51, 51));
btn_exit.setPressedColor(new java.awt.Color(153, 0, 0));
btn_exit.setText("X");
btn_exit.setFont(new java.awt.Font("Roboto", 1, 20)); // NOI18N
btn_exit.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        btn_exitMouseClicked(evt);
    }
})
```

```
});

javax.swing.GroupLayout headerLayout = new javax.swing.GroupLayout(header);

header.setLayout(headerLayout);

headerLayout.setHorizontalGroup(
    headerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, headerLayout.createSequentialGroup()
        .addGroupGap(0, 825, Short.MAX_VALUE)
        .addComponent(btn_exit, javax.swing.GroupLayout.PREFERRED_SIZE, 45, javax.swing.GroupLayout.PREFERRED_SIZE)
    )
);

headerLayout.setVerticalGroup(
    headerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(btn_exit, javax.swing.GroupLayout.DEFAULT_SIZE, 30, Short.MAX_VALUE)
);

Background.add(header, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 870, 30));
```

```
Menu.setBackground(new java.awt.Color(242, 252, 255));

lblLogoEu.setIcon(new javax.swing.ImageIcon(getClass().getResource("/EuBank2.png"))); // NOI18N

btn_clients.setBackground(new java.awt.Color(153, 214, 234));
btn_clients.setBorder(null);
btn_clients.setBorderColor(new java.awt.Color(153, 214, 234));
btn_clients.setBorderRadius(0);
btn_clients.setBorderThickness(0);
btn_clients.setHoverColor(new java.awt.Color(87, 175, 204));
btn_clients.setText("Cuentas de Clientes");
btn_clients.setFont(new java.awt.Font("Roboto", 1, 16)); // NOI18N
btn_clients.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btn_clientsActionPerformed(evt);
    }
});
```

```
}

});

btn_movements.setBackground(new java.awt.Color(153, 214, 234));

btn_movements.setBorder(null);

btn_movements.setBorderColor(new java.awt.Color(153, 214, 234));

btn_movements.setBorderRadius(0);

btn_movements.setBorderThickness(0);

btn_movements.setHoverColor(new java.awt.Color(87, 175, 204));

btn_movements.setText("Movimientos");

btn_movements.setFont(new java.awt.Font("Roboto", 1, 16)); // NOI18N

btn_movements.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        btn_movementsActionPerformed(evt);

    }

});
```

```
btn_exitsession.setBackground(new java.awt.Color(153, 214, 234));  
btn_exitsession.setBorder(null);  
btn_exitsession.setBorderColor(new java.awt.Color(153, 214, 234));  
btn_exitsession.setBorderRadius(0);  
btn_exitsession.setBorderThickness(0);  
btn_exitsession.setHoverColor(new java.awt.Color(87, 175, 204));  
btn_exitsession.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ingresar.png"))); // NOI18N  
btn_exitsession.setText("Cerrar Sesión");  
btn_exitsession.setFont(new java.awt.Font("Roboto", 1, 18)); // NOI18N  
btn_exitsession.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);  
btn_exitsession.addMouseListener(new java.awt.event.MouseAdapter() {  
    public void mouseClicked(java.awt.event.MouseEvent evt) {  
        btn_exitsessionMouseClicked(evt);  
    }  
});
```

```
javax.swing.GroupLayout MenuLayout = new javax.swing.GroupLayout(Menu);

Menu.setLayout(MenuLayout);

MenuLayout.setHorizontalGroup(
    MenuLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(btn_clients, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(btn_movements, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(btn_exitsession, javax.swing.GroupLayout.DEFAULT_SIZE, 190, Short.MAX_VALUE)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, MenuLayout.createSequentialGroup()
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(lblLogoEu)
        .addContainerGap())
);
MenuLayout.setVerticalGroup(
    MenuLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(MenuLayout.createSequentialGroup()
```

```
.addGap(22, 22, 22)

.addComponent(lblLogoEu, javax.swing.GroupLayout.PREFERRED_SIZE, 83, javax.swing.GroupLayout.PREFERRED_SIZE)

.addGap(28, 28, 28)

.addComponent(btn_clients, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE) javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

.addGap(18, 18, 18)

.addComponent(btn_movements, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE) javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 255, Short.MAX_VALUE)

.addComponent(btn_exitsession, javax.swing.GroupLayout.PREFERRED_SIZE, 67, javax.swing.GroupLayout.PREFERRED_SIZE)

.addGap(17, 17, 17))

);

Background.add(Menu, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 30, 190, 570));

Contenedor.setBackground(new java.awt.Color(255, 255, 255));
```

```
javax.swing.GroupLayout ContenedorLayout = new javax.swing.GroupLayout(Contenedor);
Contenedor.setLayout(ContenedorLayout);
ContenedorLayout.setHorizontalGroup(
    ContenedorLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 680, Short.MAX_VALUE)
);
ContenedorLayout.setVerticalGroup(
    ContenedorLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 570, Short.MAX_VALUE)
);
Background.add(Contenedor, new org.netbeans.lib.awtextra.AbsoluteConstraints(190, 30, 680, 570));

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
```

```
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(Background, javax.swing.GroupLayout.PREFERRED_SIZE, 864, javax.swing.GroupLayout.PREFERRED_SIZE)
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(Background, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
    );
    pack();
}// </editor-fold>

private void headerMousePressed(java.awt.event.MouseEvent evt) {
    xMouse = evt.getX();
    yMouse = evt.getY();
}

}
```

```
private void headerMouseDragged(java.awt.event.MouseEvent evt) {  
    int x = evt.getXOnScreen();  
    int y = evt.getYOnScreen();  
    this.setLocation(x - xMouse, y - yMouse);  
}  
  
private void btn_exitMouseClicked(java.awt.event.MouseEvent evt) {  
    System.exit(0);  
}  
  
private void btn_exitsessionMouseClicked(java.awt.event.MouseEvent evt) {  
    System.exit(0);  
}  
  
private void btn_clientsActionPerformed(java.awt.event.ActionEvent evt) {
```

```
Clients c = new Clients();

c.setSize(680, 570);

c.setLocation(0, 0);

Contenedor.removeAll();

Contenedor.add(c, BorderLayout.CENTER);

Contenedor.revalidate();

Contenedor.repaint();

}

private void btn_movementsActionPerformed(java.awt.event.ActionEvent evt) {

Movements c = new Movements();

c.setSize(680, 570);

c.setLocation(0, 0);

Contenedor.removeAll();
```

```
Contenedor.add(c, BorderLayout.CENTER);

Contenedor.revalidate();

Contenedor.repaint();

}

public static void main(String args[]) {

    java.awt.EventQueue.invokeLater(new Runnable() {

        public void run() {

            new Home().setVisible(true);

        }

    });

}

// Variables declaration - do not modify

private javax.swing.JPanel Background;
```

```
private javax.swing.JPanel Contenedor;  
private javax.swing.JPanel Menu;  
private ec.edu.monster.vista.ModernButton btn_clients;  
private ec.edu.monster.vista.ModernButton btn_exit;  
private ec.edu.monster.vista.ModernButton btn_exitsession;  
private ec.edu.monster.vista.ModernButton btn_movements;  
private javax.swing.JPanel header;  
private javax.swing.JLabel lblLogoEu;  
// End of variables declaration  
}
```

TABLA 8 DepositoView

```
namespace CLIESC_EuBank_SOAPDOTNET_GR05.views
```

```
{  
  
partial class Login  
  
{  
  
/// <summary>  
  
/// Required designer variable.  
  
/// </summary>  
  
private System.ComponentModel.IContainer components = null;  
  
  
/// <summary>  
  
/// Clean up any resources being used.  
  
/// </summary>  
  
/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>  
  
protected override void Dispose(bool disposing)  
  
{  
  
if (disposing && (components != null))  
  
{
```

```
components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

    System.ComponentModel.ComponentResourceManager resources = new System.ComponentModel.ComponentResourceManager(typeof(Login));

    this.header = new System.Windows.Forms.Panel();

    this.min_button = new System.Windows.Forms.Button();


```

```
this.btnExit = new System.Windows.Forms.Button();

this.pictureBox4 = new System.Windows.Forms.PictureBox();

this.lblHeader = new System.Windows.Forms.Label();

this.lbl_tech = new System.Windows.Forms.Label();

this.lblnames = new System.Windows.Forms.Label();

this.dotnet_image = new System.Windows.Forms.PictureBox();

this.image_back = new System.Windows.Forms.PictureBox();

this.pnl_credenciales = new CLIESC_ConUni_SOAPDOTNET_GR05.Controls.LDPanelRound();

this.pictureBox1 = new System.Windows.Forms.PictureBox();

this.btnExit_Ingresar = new System.Windows.Forms.Button();

this.txb_password = new System.Windows.Forms.TextBox();

this.tbx_user = new System.Windows.Forms.TextBox();

this.lblPass = new System.Windows.Forms.Label();

this.lblUser = new System.Windows.Forms.Label();

this.header.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox4)).BeginInit();
```

```
((System.ComponentModel.ISupportInitialize)(this.dotnet_image)).BeginInit();  
((System.ComponentModel.ISupportInitialize)(this.image_back)).BeginInit();  
  
this.pnl_credenciales.SuspendLayout();  
  
((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();  
  
this.SuspendLayout();  
  
//  
  
// header  
  
//  
  
this.header.BackColor = System.Drawing.Color.MediumVioletRed;  
  
this.header.Controls.Add(this.min_button);  
  
this.header.Controls.Add(this.btn_exit);  
  
this.header.Controls.Add(this.pictureBox4);  
  
this.header.Controls.Add(this.lblHeader);  
  
this.header.Dock = System.Windows.Forms.DockStyle.Top;  
  
this.header.Location = new System.Drawing.Point(0, 0);  
  
this.header.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);
```

```
this.header.Name = "header";

this.header.Size = new System.Drawing.Size(705, 37);

this.header.TabIndex = 2;

this.header.MouseDown += new System.Windows.Forms.MouseEventHandler(this.header_MouseDown);

// 

// min_button

//

this.min_button.BackColor = System.Drawing.Color.Transparent;

this.min_button.BackgroundImage = ((System.Drawing.Image)(resources.GetObject("min_button.BackgroundImage")));

this.min_button.BackgroundImageLayout = System.Windows.Forms.ImageLayout.Zoom;

this.min_button.Cursor = System.Windows.Forms.Cursors.Hand;

this.min_button.FlatAppearance.BorderSize = 0;

this.min_button.FlatAppearance.MouseOverBackColor = System.Drawing.Color.FromArgb(((int)(((byte)(147)))), ((int)(((byte)(15)))), ((int)((byte)(98))));

this.min_button.FlatStyle = System.Windows.Forms.FlatStyle.Flat;

this.min_button.ForeColor = System.Drawing.Color.Transparent;

this.min_button.Location = new System.Drawing.Point(644, 8);
```

```
this.min_button.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);
this.min_button.Name = "min_button";
this.min_button.Size = new System.Drawing.Size(19, 20);
this.min_button.TabIndex = 10;
this.min_button.UseVisualStyleBackColor = true;
this.min_button.Click += new System.EventHandler(this.min_button_Click);
//
// btn_exit
//
this.btn_exit.BackColor = System.Drawing.Color.Transparent;
this.btn_exit.BackgroundImage = ((System.Drawing.Image)(resources.GetObject("btn_exit.BackgroundImage")));
this.btn_exit.BackgroundImageLayout = System.Windows.Forms.ImageLayout.Zoom;
this.btn_exit.Cursor = System.Windows.Forms.Cursors.Hand;
this.btn_exit.FlatAppearance.BorderSize = 0;
this.btn_exit.FlatAppearance.MouseOverBackColor = System.Drawing.Color.FromArgb(((int)(((byte)(147)))), ((int)(((byte)(15)))), ((int)((byte)(98)))));
this.btn_exit.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
```

```
this.btnExit.ForeColor = System.Drawing.Color.Transparent;

this.btnExit.Location = new System.Drawing.Point(677, 8);

this.btnExit.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);

this.btnExit.Name = "btn_exit";

this.btnExit.Size = new System.Drawing.Size(19, 20);

this.btnExit.TabIndex = 9;

this.btnExit.UseVisualStyleBackColor = true;

this.btnExit.Click += new System.EventHandler(this.btnExit_Click);

// 

// pictureBox4

//

this.pictureBox4.Image = ((System.Drawing.Image)(resources.GetObject("pictureBox4.Image")));

this.pictureBox4.Location = new System.Drawing.Point(2, 0);

this.pictureBox4.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);

this.pictureBox4.Name = "pictureBox4";

this.pictureBox4.Size = new System.Drawing.Size(31, 31);
```

```
this.pictureBox4.SizeMode = System.Windows.Forms.PictureBoxSizeMode.Zoom;

this.pictureBox4.TabIndex = 8;

this.pictureBox4.TabStop = false;

//  
// lblHeader  
//  
this.lblHeader.AutoSize = true;  
this.lblHeader.Font = new System.Drawing.Font("Leelawadee UI", 13.8F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));  
this.lblHeader.ForeColor = System.Drawing.Color.White;  
this.lblHeader.Location = new System.Drawing.Point(38, 3);  
this.lblHeader.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);  
this.lblHeader.Name = "lblHeader";  
this.lblHeader.Size = new System.Drawing.Size(151, 25);  
this.lblHeader.TabIndex = 5;  
this.lblHeader.Text = "Inicio de Sesión";  
//
```

```
// lbl_tech

//
this.lbl_tech.AutoSize = true;

this.lbl_tech.Font = new System.Drawing.Font("Microsoft Sans Serif", 16.2F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));

this.lbl_tech.ForeColor = System.Drawing.Color.FromArgb(((int)((byte)(81))), ((int)((byte)(43))), ((int)((byte)(212))));

this.lbl_tech.Location = new System.Drawing.Point(150, 72);

this.lbl_tech.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);

this.lbl_tech.Name = "lbl_tech";

this.lbl_tech.Size = new System.Drawing.Size(89, 26);

this.lbl_tech.TabIndex = 18;

this.lbl_tech.Text = "+ REST";

//
// lbnames

//
this.lbnames.AutoSize = true;
```

```
this.lblnames.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.lblnames.ForeColor = System.Drawing.Color.PaleVioletRed;

this.lblnames.Location = new System.Drawing.Point(88, 119);

this.lblnames.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);

this.lblnames.Name = "lblnames";

this.lblnames.Size = new System.Drawing.Size(152, 60);

this.lblnames.TabIndex = 19;

this.lblnames.Text = "Barrionuevo Lindsay\r\nRivera Joel\r\nYaranga Leonardo";

this.lblnames.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;

// 

// dotnet_image

//

this.dotnet_image.Image = global::CLIESC_EuBank_SOAPDOTNET_GR05.Properties.Resources.dotnet;

this.dotnet_image.Location = new System.Drawing.Point(103, 65);

this.dotnet_image.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);
```

```
this.dotnet_image.Name = "dotnet_image";

this.dotnet_image.Size = new System.Drawing.Size(43, 40);

this.dotnet_image.SizeMode = System.Windows.Forms.PictureBoxSizeMode.Zoom;

this.dotnet_image.TabIndex = 17;

this.dotnet_image.TabStop = false;

// 

// image_back

//

this.image_back.Image = global::CLIESC_EuBank_SOAPDOTNET_GR05.Properties.Resources.sulliLogo;

this.image_back.Location = new System.Drawing.Point(42, 185);

this.image_back.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);

this.image_back.Name = "image_back";

this.image_back.Size = new System.Drawing.Size(250, 277);

this.image_back.SizeMode = System.Windows.Forms.PictureBoxSizeMode.Zoom;

this.image_back.TabIndex = 16;

this.image_back.TabStop = false;
```

```
//  
// pnl_credenciales  
  
this.pnl_credenciales.BackColor = System.Drawing.Color.LavenderBlush;  
this.pnl_credenciales.BorderColor = System.Drawing.Color.MediumVioletRed;  
this.pnl_credenciales.BorderThickness = 1;  
this.pnl_credenciales.Controls.Add(this.pictureBox1);  
this.pnl_credenciales.Controls.Add(this.btn_Ingresar);  
this.pnl_credenciales.Controls.Add(this.txb_password);  
this.pnl_credenciales.Controls.Add(this.tbx_user);  
this.pnl_credenciales.Controls.Add(this.lblPass);  
this.pnl_credenciales.Controls.Add(this.lblUser);  
this.pnl_credenciales.CornerRadius = 10;  
this.pnl_credenciales.Location = new System.Drawing.Point(343, 60);  
this.pnl_credenciales.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);  
this.pnl_credenciales.Name = "pnl_credenciales";
```

```
this.pnl_credenciales.Size = new System.Drawing.Size(304, 401);

this.pnl_credenciales.TabIndex = 15;

// 

// pictureBox1

// 

this.pictureBox1.Image = global::CLIESC_EuBank_SOAPDOTNET_GR05.Properties.Resources.EB_REST_DOTNET2;

this.pictureBox1.Location = new System.Drawing.Point(60, 5);

this.pictureBox1.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);

this.pictureBox1.Name = "pictureBox1";

this.pictureBox1.Size = new System.Drawing.Size(190, 71);

this.pictureBox1.SizeMode = System.Windows.Forms.PictureBoxSizeModeSizeMode.Zoom;

this.pictureBox1.TabIndex = 13;

this.pictureBox1.TabStop = false;

// 

// btn_Ingresar

// 
```

```
this.btn_Ingresar.Anchor = System.Windows.Forms.AnchorStyles.None;  
  
this.btn_Ingresar.BackColor = System.Drawing.Color.MediumVioletRed;  
  
this.btn_Ingresar.Cursor = System.Windows.Forms.Cursors.Hand;  
  
this.btn_Ingresar.FlatAppearance.BorderSize = 0;  
  
this.btn_Ingresar.FlatAppearance.MouseOverBackColor = System.Drawing.Color.FromArgb(((int)(((byte)(147)))), ((int)((byte)(15))), ((int)((byte)(98))));  
  
this.btn_Ingresar.FlatStyle = System.Windows.Forms.FlatStyle.Flat;  
  
this.btn_Ingresar.Font = new System.Drawing.Font("Leelawadee UI", 13.8F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));  
  
this.btn_Ingresar.ForeColor = System.Drawing.Color.White;  
  
this.btn_Ingresar.Image = global::CLIESC_EuBank_SOAPDOTNET_GR05.Properties.Resources.ingresar;  
  
this.btn_Ingresar.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;  
  
this.btn_Ingresar.Location = new System.Drawing.Point(52, 318);  
  
this.btn_Ingresar.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);  
  
this.btn_Ingresar.Name = "btn_Ingresar";  
  
this.btn_Ingresar.Size = new System.Drawing.Size(199, 49);  
  
this.btn_Ingresar.TabIndex = 12;
```

```
this.btn_Ingresar.Text = "Iniciar Sesión";
this.btn_Ingresar.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
this.btn_Ingresar.UseVisualStyleBackColor = false;
this.btn_Ingresar.Click += new System.EventHandler(this.btn_Ingresar_Click);
// 
// txb_password
//
this.txb_password.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
this.txb_password.Font = new System.Drawing.Font("Leelawadee UI", 13.8F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.txb_password.Location = new System.Drawing.Point(35, 249);
this.txb_password.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);
this.txb_password.Name = "txb_password";
this.txb_password.Size = new System.Drawing.Size(237, 32);
this.txb_password.TabIndex = 5;
this.txb_password.UseSystemPasswordChar = true;
```

```
this.txb_password.WordWrap = false;  
  
//  
  
// tbx_user  
  
//  
  
this.tbx_user.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;  
  
this.tbx_user.Font = new System.Drawing.Font("Leelawadee UI", 16.2F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));  
  
this.tbx_user.Location = new System.Drawing.Point(34, 149);  
  
this.tbx_user.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);  
  
this.tbx_user.Name = "tbx_user";  
  
this.tbx_user.Size = new System.Drawing.Size(238, 36);  
  
this.tbx_user.TabIndex = 4;  
  
//  
  
// lblPass  
  
//  
  
this.lblPass.AutoSize = true;
```

```
this.lblPass.Font = new System.Drawing.Font("Microsoft Sans Serif", 19.8F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.lblPass.ForeColor = System.Drawing.Color.MediumVioletRed;

this.lblPass.Location = new System.Drawing.Point(65, 207);

this.lblPass.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);

this.lblPass.Name = "lblPass";

this.lblPass.Size = new System.Drawing.Size(165, 31);

this.lblPass.TabIndex = 3;

this.lblPass.Text = "Contraseña";

// 

// lblUser

//

this.lblUser.AutoSize = true;

this.lblUser.Font = new System.Drawing.Font("Microsoft Sans Serif", 19.8F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.lblUser.ForeColor = System.Drawing.Color.MediumVioletRed;

this.lblUser.Location = new System.Drawing.Point(87, 106);
```

```
this.lblUser.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
this.lblUser.Name = "lblUser";
this.lblUser.Size = new System.Drawing.Size(115, 31);
this.lblUser.TabIndex = 2;
this.lblUser.Text = "Usuario";
this.lblUser.Click += new System.EventHandler(this.lblUser_Click);
//
// Login
//
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.BackColor = System.Drawing.Color.White;
this.ClientSize = new System.Drawing.Size(705, 488);
this.Controls.Add(this.lbl_tech);
this.Controls.Add(this.dotnet_image);
this.Controls.Add(this.lblnames);
```

```
this.Controls.Add(this.image_back);

this.Controls.Add(this.pnl_credenciales);

this.Controls.Add(this.header);

this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;

this.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);

this.Name = "Login";

this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

this.Text = "Form1";

this.header.ResumeLayout(false);

this.header.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox4)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.dotnet_image)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.image_back)).EndInit();

this.pnl_credenciales.ResumeLayout(false);

this.pnl_credenciales.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
```

```
this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Panel header;
private System.Windows.Forms.Button min_button;
private System.Windows.Forms.Button btn_exit;
private System.Windows.Forms.PictureBox pictureBox4;
private System.Windows.Forms.Label lblHeader;
private CLIESC_ConUni_SOAPDOTNET_GR05.Controls.LDPanelRound pnl_credenciales;
private System.Windows.Forms.Button btn_Ingresar;
private System.Windows.Forms.TextBox txb_password;
private System.Windows.Forms.TextBox tbx_user;
```

```
private System.Windows.Forms.Label lblPass;  
private System.Windows.Forms.Label lblUser;  
private System.Windows.Forms.PictureBox image_back;  
private System.Windows.Forms.PictureBox dotnet_image;  
private System.Windows.Forms.Label lbl_tech;  
private System.Windows.Forms.Label lblnames;  
private System.Windows.Forms.PictureBox pictureBox1;  
}  
}  
  
namespace CLIESC_EuBank_SOAPDOTNET_GR05.views  
{  
    partial class Movements  
    {  
        /// <summary>  
        /// Required designer variable.  
    }
```

```
/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>
/// Clean up any resources being used.
/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{
    if (disposing && (components != null))

    {
        components.Dispose();
    }

    base.Dispose(disposing);
}
```

```
#region Windows Form Designer generated code

/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    System.Windows.Forms.DataGridViewCellStyle dataGridViewCellStyle1 = new System.Windows.Forms.DataGridViewCellStyle();
    System.Windows.Forms.DataGridViewCellStyle dataGridViewCellStyle2 = new System.Windows.Forms.DataGridViewCellStyle();
    System.Windows.Forms.DataGridViewCellStyle dataGridViewCellStyle3 = new System.Windows.Forms.DataGridViewCellStyle();
    this.GW_Movements = new System.Windows.Forms.DataGridView();
    this.IdPanelRound1 = new CLIESC_ConUni_SOAPDOTNET_GR05.Controls.LDPanelRound();
    this.btn_Ingresar = new System.Windows.Forms.Button();
    this.lblsubtitle = new System.Windows.Forms.Label();
    this.lblTitle = new System.Windows.Forms.Label();
}
```

```
this.pictureBox1 = new System.Windows.Forms.PictureBox();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();

this.flowLayoutPanel1.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();

this.SuspendLayout();

// 

// GW_Movements

// 

this.GW_Movements.AutoSizeColumnsMode = System.Windows.Forms.DataGridViewAutoSizeColumnsMode.AllCells;

this.GW_Movements.AutoSizeRowsMode = System.Windows.Forms.DataGridViewAutoSizeRowsMode.AllCells;

this.GW_Movements.BackgroundColor = System.Drawing.Color.GhostWhite;

this.GW_Movements.BorderStyle = System.Windows.Forms.BorderStyle.None;

this.GW_Movements.ColumnHeadersBorderStyle = System.Windows.Forms.DataGridViewHeaderBorderStyle.None;

dataGridViewCellStyle1.Alignment = System.Windows.Forms.DataGridViewContentAlignment.MiddleLeft;

dataGridViewCellStyle1.BackColor = System.Drawing.Color.MediumVioletRed;
```

```
dataGridViewCellStyle1.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.8F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));

dataGridViewCellStyle1.ForeColor = System.Drawing.Color.White;

dataGridViewCellStyle1.SelectionBackColor = System.Drawing.Color.DeepPink;

dataGridViewCellStyle1.SelectionForeColor = System.Drawing.Color.White;

dataGridViewCellStyle1.WrapMode = System.Windows.Forms.DataGridViewTriState.True;

this.GW_Movements.ColumnHeadersDefaultCellStyle = dataGridViewCellStyle1;

this.GW_Movements.ColumnHeadersHeightSizeMode = System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;

this.GW_Movements.EnableHeadersVisualStyles = false;

this.GW_Movements.GridColor = System.Drawing.Color.LavenderBlush;

this.GW_Movements.Location = new System.Drawing.Point(24, 136);

this.GW_Movements.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);

this.GW_Movements.Name = "GW_Movements";

this.GW_Movements.RowHeadersBorderStyle = System.Windows.Forms.DataGridViewHeaderBorderStyle.Single;

dataGridViewCellStyle2.Alignment = System.Windows.Forms.DataGridViewContentAlignment.MiddleLeft;

dataGridViewCellStyle2.BackColor = System.Drawing.Color.LavenderBlush;
```

```
dataGridViewCellStyle2.Font = new System.Drawing.Font("Microsoft Sans Serif", 7.8F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

dataGridViewCellStyle2.ForeColor = System.Drawing.Color.Black;

dataGridViewCellStyle2.SelectionBackColor = System.Drawing.Color.PaleVioletRed;

dataGridViewCellStyle2.SelectionForeColor = System.Drawing.SystemColors.HighlightText;

dataGridViewCellStyle2.WrapMode = System.Windows.Forms.DataGridViewAutoSizeTriState.True;

this.GW_Movements.RowHeadersDefaultCellStyle = dataGridViewCellStyle2;

this.GW_Movements.RowHeadersVisible = false;

this.GW_Movements.RowHeadersWidth = 51;

dataGridViewCellStyle3.BackColor = System.Drawing.Color.LavenderBlush;

dataGridViewCellStyle3.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

dataGridViewCellStyle3.ForeColor = System.Drawing.Color.Black;

dataGridViewCellStyle3.SelectionBackColor = System.Drawing.Color.PaleVioletRed;

dataGridViewCellStyle3.SelectionForeColor = System.Drawing.Color.White;

this.GW_Movements.RowsDefaultCellStyle = dataGridViewCellStyle3;

this.GW_Movements.RowTemplate.Height = 24;
```

```
this.GW_Movements.Size = new System.Drawing.Size(604, 324);

this.GW_Movements.TabIndex = 4;

this.GW_Movements.CellContentClick += new System.Windows.Forms.DataGridViewCellEventHandler(this.GW_Movements_CellContentClick);

// 

// IdPanelRound1

//

this.IdPanelRound1.BackColor = System.Drawing.Color.LavenderBlush;

this.IdPanelRound1.BorderColor = System.Drawing.Color.Transparent;

this.IdPanelRound1.BorderThickness = 0;

this.IdPanelRound1.Controls.Add(this.btn_Ingresar);

this.IdPanelRound1.Controls.Add(this.lblsubtitle);

this.IdPanelRound1.Controls.Add(this.lblTitle);

this.IdPanelRound1.Controls.Add(this.pictureBox1);

this.IdPanelRound1.CornerRadius = 10;

this.IdPanelRound1.Location = new System.Drawing.Point(24, 27);

this.IdPanelRound1.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);
```

```
this.IdPanelRound1.Name = "IdPanelRound1";
this.IdPanelRound1.Size = new System.Drawing.Size(604, 81);
this.IdPanelRound1.TabIndex = 1;
//
// btn_Ingresar
//
this.btn_Ingresar.Anchor = System.Windows.Forms.AnchorStyles.None;
this.btn_Ingresar.BackColor = System.Drawing.Color.MediumVioletRed;
this.btn_Ingresar.Cursor = System.Windows.Forms.Cursors.Hand;
this.btn_Ingresar.FlatAppearance.BorderSize = 0;
this.btn_Ingresar.FlatAppearance.MouseOverBackColor = System.Drawing.Color.FromArgb(((int)(((byte)(147)))), ((int)((byte)(15))), ((int)((byte)(98)))));
this.btn_Ingresar.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
this.btn_Ingresar.Font = new System.Drawing.Font("Leelawadee UI", 12F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.btn_Ingresar.ForeColor = System.Drawing.Color.White;
this.btn_Ingresar.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
```

```
this.btn_Ingresar.Location = new System.Drawing.Point(507, 16);

this.btn_Ingresar.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);

this.btn_Ingresar.Name = "btn_Ingresar";

this.btn_Ingresar.Size = new System.Drawing.Size(76, 49);

this.btn_Ingresar.TabIndex = 13;

this.btn_Ingresar.Text = "+ Crear";

this.btn_Ingresar.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;

this.btn_Ingresar.UseVisualStyleBackColor = false;

this.btn_Ingresar.Click += new System.EventHandler(this.btn_Ingresar_Click);

// 

// lblsubtitle

//

this.lblsubtitle.AutoSize = true;

this.lblsubtitle.Font = new System.Drawing.Font("Microsoft Sans Serif", 10.2F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));

this.lblsubtitle.ForeColor = System.Drawing.Color.Gray;
```

```
this.lblsubtitle.Location = new System.Drawing.Point(89, 50);

this.lblsubtitle.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);

this.lblsubtitle.Name = "lblsubtitle";

this.lblsubtitle.Size = new System.Drawing.Size(403, 17);

this.lblsubtitle.TabIndex = 19;

this.lblsubtitle.Text = "Revisa los movimientos realizados entre las cuentas bancarias";

// 

// lblTitle

//

this.lblTitle.AutoSize = true;

this.lblTitle.Font = new System.Drawing.Font("Microsoft Sans Serif", 16.2F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));

this.lblTitle.ForeColor = System.Drawing.Color.MediumVioletRed;

this.lblTitle.Location = new System.Drawing.Point(88, 16);

this.lblTitle.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);

this.lblTitle.Name = "lblTitle";
```

```
this.lblTitle.Size = new System.Drawing.Size(238, 26);

this.lblTitle.TabIndex = 3;

this.lblTitle.Text = "Movimientos Bancarios";

// 

// pictureBox1

//

this.pictureBox1.Image = global::CLIESC_EuBank_SOAPDOTNET_GR05.Properties.Resources.SulliGeneral;

this.pictureBox1.Location = new System.Drawing.Point(10, 2);

this.pictureBox1.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);

this.pictureBox1.Name = "pictureBox1";

this.pictureBox1.Size = new System.Drawing.Size(74, 76);

this.pictureBox1.SizeMode = System.Windows.Forms.PictureBoxSizeMode.Zoom;

this.pictureBox1.TabIndex = 0;

this.pictureBox1.TabStop = false;

// 

// Movements
```

```
//  
  
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);  
  
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;  
  
this.BackColor = System.Drawing.Color.White;  
  
this.ClientSize = new System.Drawing.Size(652, 492);  
  
this.Controls.Add(this.GW_Movements);  
  
this.Controls.Add(this.IdPanelRound1);  
  
this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;  
  
this.Margin = new System.Windows.Forms.Padding(2, 2, 2, 2);  
  
this.Name = "Movements";  
  
this.Text = "Movements";  
  
((System.ComponentModel.ISupportInitialize)(this.GW_Movements)).EndInit();  
  
this.IdPanelRound1.ResumeLayout(false);  
  
this.IdPanelRound1.PerformLayout();  
  
((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();  
  
this.ResumeLayout(false);
```

```
}

#endregion

private CLIESC_ConUni_SOAPDOTNET_GR05.Controls.LDPanelRound ldPanelRound1;
private System.Windows.Forms.Label lblsubtitle;
private System.Windows.Forms.Label lblTitle;
private System.Windows.Forms.PictureBox pictureBox1;
private System.Windows.Forms.DataGridView GW_Movements;
private System.Windows.Forms.Button btn_Ingresar;

}
```

4.5 CREACIÓN DEL PROYECTO WEB

Como tercer cliente, se crea el proyecto web usando Visual Studio, para ello debemos realizar un proceso similar al que ya teníamos anteriormente

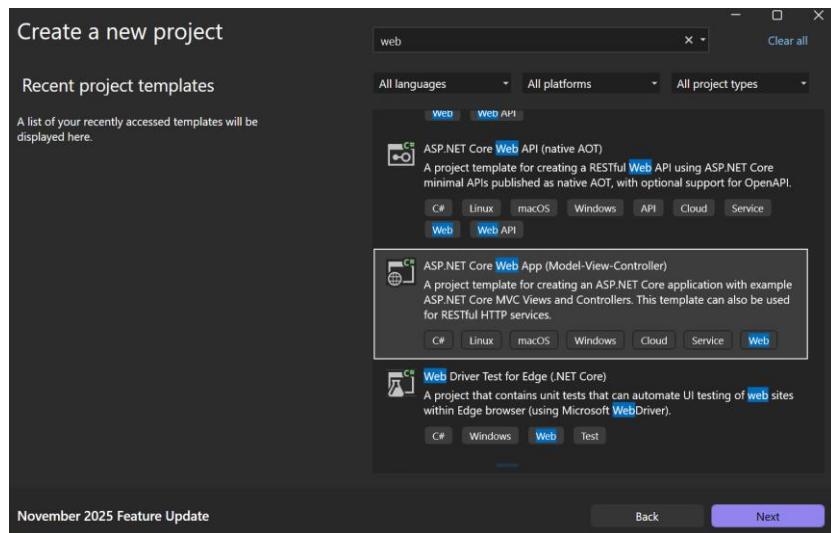


Figura 48 Creación directorio 02.CLIENTE WEB

4.5.1 CREACIÓN ENTORNO

Se debe preparar el entorno creando las carpetas necesarias y la instalación de librerías necesarias.

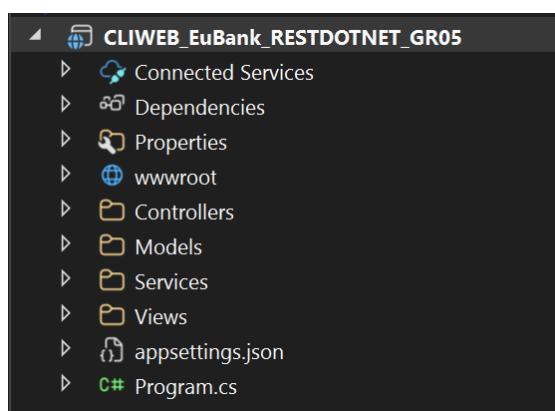


Figura 49 Creación Estructura Web

4.5.2 CREACIÓN CAPA SERVICIO

TABLA 9 Codificación App.py

```
using CLIWEB_EuBank_RESTDOTNET_GR05.Models;
using System.Net.Http.Json;
using System.Text.Json;
using System.Xml.Serialization;
using System.IO;

namespace CLIWEB_EuBank_RESTDOTNET_GR05.Services
{
    public class EuBank ApiService
    {
        private readonly HttpClient _httpClient;
        private readonly string _baseUrl = "http://localhost:62278/api"; // Adjust the base URL
        as needed

        public EuBank ApiService(HttpClient httpClient)
        {
            _httpClient = httpClient;
        }

        public async Task<bool> LoginAsync(LoginRequest request)
        {
            try
            {
                var response = await _httpClient.PostAsJsonAsync($"{_baseUrl}/login", request);
                if (response.IsSuccessStatusCode)
                {
                    var result = await response.Content.ReadFromJsonAsync<LoginResponse>();
                    return result?.success ?? false;
                }
            }
            return false;
        }
    }
}
```

```
        }

    catch
    {
        return false;
    }
}

public async Task<List<Account>?> GetAllAccountsAsync()
{
    try
    {
        var response = await _httpClient.GetAsync($"{_baseUrl}/cuentas");
        if (response.IsSuccessStatusCode)
        {
            return await response.Content.ReadFromJsonAsync<List<Account>>();
        }
        return null;
    }
    catch
    {
        return null;
    }
}

public async Task<Account?> GetAccountDetailsAsync(string accountCode)
{
    try
    {
        var response = await _httpClient.GetAsync($"{_baseUrl}/cuentas/{accountCode}");
        if (response.IsSuccessStatusCode)
        {
            return await response.Content.ReadFromJsonAsync<Account>();
        }
        return null;
    }
}
```

```
        }

    catch
    {
        return null;
    }
}

public async Task<List<Movement>?> GetMovementsAsync(string? accountCode = null)
{
    try
    {
        var url = $"{_baseUrl}/movimientos";
        if (!string.IsNullOrEmpty(accountCode))
        {
            url += $"?cuenta={accountCode}";
        }

        var response = await _httpClient.GetAsync(url);
        if (response.IsSuccessStatusCode)
        {
            return await response.Content.ReadFromJsonAsync<List<Movement>>();
        }
        return null;
    }
    catch
    {
        return null;
    }
}

public async Task<bool> ProcessMovementAsync(MovementRequest request)
{
    try
    {
        Console.WriteLine($"[PROCESS] POST {_baseUrl}/movimientos");
    }
}
```

```

        Console.WriteLine($"[PROCESS] Request: {JsonSerializer.Serialize(request)}");
        var response = await _httpClient.PostAsJsonAsync($"{_baseUrl}/movimientos",
request);
        Console.WriteLine($"[PROCESS] Status: {response.StatusCode}");
        if (response.IsSuccessStatusCode)
        {
            var content = await response.Content.ReadAsStringAsync();
            Console.WriteLine($"[PROCESS] Response: {content}");
        }
        else
        {
            var error = await response.Content.ReadAsStringAsync();
            Console.WriteLine($"[PROCESS] Error Response: {error}");
        }
        return response.IsSuccessStatusCode;
    }
    catch (Exception ex)
    {
        Console.WriteLine($"[PROCESS] Error: {ex.Message}");
        return false;
    }
}

public async Task<List<string>?> GetMovementTypesAsync()
{
    try
    {
        var response = await _httpClient.GetAsync($"{_baseUrl}/tiposmovimiento");
        if (response.IsSuccessStatusCode)
        {
            return await response.Content.ReadFromJsonAsync<List<string>>();
        }
        return null;
    }
    catch

```

```

    {
        return null;
    }
}
}
}

```

En la clase anterior se maneja todas las rutas del aplicativo y el link WSDL el cual contiene todos los métodos.

4.5.3 CREACIÓN CAPA VISTA

Se procede a crear cada una de las interfaces necesarias para el funcionamiento.

TABLA 10 Codificación Vista Principal

```

@{
    ViewData["Title"] = "Login - EuBank";
    Layout = null;
}

<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"]</title>
    <script src="https://cdn.tailwindcss.com"></script>
    <script>
        tailwind.config = {
            theme: {
                extend: {
                    colors: {
                        'eubank-primary': '#FF1493',

```

```

        'eubank-secondary': '#FF69B4',
    }
}
}
}

</script>
<style>

.form-icon {
    position: absolute;
    left: 12px;
    top: 50%;
    transform: translateY(-50%);
    color: #6b7280;
    z-index: 10;
}

.input-with-icon {
    position: relative;
}

.input-with-icon input {
    padding-left: 40px;
}

</style>
</head>
<body class="min-h-screen flex font-sans bg-gray-100">
    <div class="flex-1 flex items-center justify-center p-8">
        <div class="max-w-6xl w-full bg-white rounded-xl shadow-2xl overflow-hidden">
            <div class="flex flex-col md:flex-row">

                <div class="w-full md:w-1/2 flex items-center justify-center p-12 bg-gradient-to-r from-eubank-primary to-eubank-secondary">
                    <div class="flex flex-col items-center justify-center space-y-6 text-center">
                        
                        <div class="text-center">
                            <h3 class="text-3xl font-bold text-white mb-2">Bienvenido a EuBank</h3>

```

```

        <p class="text-purple-200 text-sm">Tu banco de confianza - Cliente REST
.NET</p>

        </div>
        </div>
        </div>

<div class="w-full md:w-1/2 p-8 md:p-12">
    <div class="text-center mb-8">
        
        <p class="text-gray-600">Sistema de Gestión Bancaria</p>
    </div>

    @if (ViewBag.Error != null)
    {
        <div class="bg-red-100 border border-red-400 text-red-700 px-4 py-3 rounded-
lg mb-6" role="alert">
            <strong class="font-bold">Error:</strong>
            <span class="block sm:inline">@ViewBag.Error</span>
        </div>
    }

<form method="post" asp-action="Login" class="space-y-6">
    <div class="input-with-icon">
        <span class="form-icon">👤 </span>
        <input type="text"
            name="usuario"
            placeholder="Usuario"
            required
            class="w-full px-4 py-3 border border-gray-300 rounded-lg focus:ring-2
            focus:ring-eubank-primary focus:border-transparent transition duration-200" />
    </div>

    <div class="input-with-icon">
        <span class="form-icon">🔒 </span>

```

```

<input type="password"
       name="clave"
       placeholder="Contraseña"
       required
       class="w-full px-4 py-3 border border-gray-300 rounded-lg focus:ring-2
focus:ring-eubank-primary focus:border-transparent transition duration-200" />
</div>

<button type="submit"
        class="w-full bg-gradient-to-r from-eubank-primary to-eubank-secondary
text-white font-bold py-3 px-4 rounded-lg hover:shadow-lg transform hover:-translate-y-0.5
transition duration-200">
    Iniciar Sesión
</button>
</form>

<div class="text-center mt-6">
    <small class="text-gray-500">© 2025 EuBank - REST.NET Group 05</small>
</div>
</div>

</div>
</div>
</div>
</body>
</html>

```

TABLA 11 Codificación realizarDepósitos

```

@{
    ViewData["Title"] = "Dashboard - EuBank";
}

```

```
<script src="https://cdn.tailwindcss.com"></script>
<script>
tailwind.config = {
  theme: {
    extend: {
      colors: {
        'eubank-primary': '#FF1493',
        'eubank-secondary': '#FF69B4',
      }
    }
  }
}
</script>

<style>
.loading-spinner {
  border: 4px solid #f3f3f3;
  border-top: 4px solid #667eea;
  border-radius: 50%;
  width: 40px;
  height: 40px;
  animation: spin 1s linear infinite;
  margin: 0 auto 10px;
}

@@keyframes spin {
  0% { transform: rotate(0deg); }
  100% { transform: rotate(360deg); }
}
</style>

<div class="mb-6">
  <div class="border-b border-gray-200">
    <nav class="-mb-px flex space-x-8">
```

```

        <button class="tab-button active border-b-2 border-eubank-secondary py-2 px-1 text-sm font-medium text-eubank-secondary">
            data-tab="cuentas">
                 Cuentas de Clientes
        </button>
        <button class="tab-button border-b-2 border-transparent py-2 px-1 text-sm font-medium text-gray-500 hover:text-gray-700 hover:border-gray-300">
            data-tab="movimientos">
                 Movimientos
        </button>
    </nav>
</div>
</div>

<div class="tab-content">
    <!-- Tab de Cuentas -->
    <div id="cuentas" class="tab-pane active">
        <div class="bg-white rounded-xl shadow-lg overflow-hidden">
            <div class="bg-gradient-to-r from-eubank-primary to-eubank-secondary text-white px-6 py-4">
                <h5 class="text-lg font-bold mb-0">Lista de Cuentas</h5>
            </div>
            <div class="p-6">
                <!-- Banner decorativo con imagen -->
                <div class="mb-6 bg-gradient-to-r from-purple-50 to-purple-100 rounded-lg p-4 flex items-center justify-between shadow-sm">
                    <div class="flex items-center space-x-4">
                        
                        <div>
                            <h6 class="text-lg font-semibold text-eubank-primary">Gestión de Cuentas</h6>
                            <p class="text-sm text-gray-600">Administra y consulta las cuentas bancarias</p>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>

```

```

<div class="hidden sm:block">
    <svg class="w-12 h-12 text-eubank-primary opacity-20" fill="currentColor" viewBox="0 0 20 20">
        <path d="M4 4a2 2 0 00-2v1h16V6a2 2 0 00-2-2H4z"></path>
        <path fill-rule="evenodd" d="M18 9H2v5a2 2 0 002h12a2 2 0 002-2V9zM4 13a1 1 0 01-1h1a1 1 0 110 2H5a1 1 0 01-1zm5-1a1 1 0 100 2h1a1 1 0 100-2H9z" clip-rule="evenodd"></path>
    </svg>
</div>
</div>

<div class="mb-6">
    <div class="flex flex-col sm:flex-row gap-4 items-start sm:items-center justify-between">
        <div class="flex-1">
            <label for="filtroEstado" class="block text-sm font-medium text-gray-700 mb-2">Filtrar por Estado:</label>
            <select id="filtroEstado" class="w-full max-w-xs border border-gray-300 rounded-lg px-3 py-2 focus:ring-2 focus:ring-eubank-primary focus:border-transparent">
                <option value="">-- Todos --</option>
                <option value="ACTIVO">Activo</option>
                <option value="INACTIVO">Inactivo</option>
            </select>
        </div>
        <div class="flex-1">
            <label for="filtroMoneda" class="block text-sm font-medium text-gray-700 mb-2">Filtrar por Moneda:</label>
            <select id="filtroMoneda" class="w-full max-w-xs border border-gray-300 rounded-lg px-3 py-2 focus:ring-2 focus:ring-eubank-primary focus:border-transparent">
                <option value="">-- Todas --</option>
            </select>
        </div>
        <div class="flex-1">
            <label for="busquedaCliente" class="block text-sm font-medium text-gray-700 mb-2">Buscar por Cliente:</label>
            <input type="text" id="busquedaCliente" placeholder="Nombre del cliente" class="w-full max-w-xs border border-gray-300 rounded-lg px-3 py-2 focus:ring-2 focus:ring-eubank-primary focus:border-transparent">
        </div>
    </div>
</div>

```

```

        </div>
    </div>
</div>

<div id="cuentasLoading" class="text-center py-10">
    <div class="loading-spinner"></div>
    <p class="text-eubank-primary font-medium">Cargando cuentas...</p>
</div>

<div id="cuentasContent" class="hidden">
    <div class="overflow-x-auto">
        <table class="min-w-full divide-y divide-gray-200">
            <thead class="bg-gray-50">
                <tr>
                    <th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Código</th>
                    <th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Cliente</th>
                    <th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Email</th>
                    <th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Teléfono</th>
                    <th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Moneda</th>
                    <th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Saldo</th>
                    <th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Estado</th>
                </tr>
            </thead>
            <tbody id="cuentasTableBody" class="bg-white divide-y divide-gray-200">
                </tbody>
            </table>
        </div>
    </div>
    <div id="cuentasError" class="bg-red-100 border border-red-400 text-red-700 px-4 py-3 rounded-lg hidden"></div>
</div>
</div>

```

```

</div>

<!-- Tab de Movimientos -->
<div id="movimientos" class="tab-pane hidden">
    <div class="bg-white rounded-xl shadow-lg overflow-hidden">
        <div class="bg-gradient-to-r from-eubank-primary to-eubank-secondary text-white px-6 py-4">
            <h5 class="text-lg font-bold mb-0">Movimientos Bancarios</h5>
            </div>
            <div class="p-6">
                <!-- Banner decorativo con imagen -->
                <div class="mb-6 bg-gradient-to-r from-purple-50 to-purple-100 rounded-lg p-4 flex items-center justify-between shadow-sm">
                    <div class="flex items-center space-x-4">
                        
                        <div>
                            <h6 class="text-lg font-semibold text-eubank-primary">Movimientos Bancario</h6>
                            <p class="text-sm text-gray-600">Revisa los movimientos realizados entre las cuentas bancarias.</p>
                        </div>
                    </div>
                </div>
                <div class="hidden sm:block">
                    <svg class="w-12 h-12 text-eubank-primary opacity-20" fill="currentColor" viewBox="0 0 20 20">
                        <path d="M4 4a2 2 0 0 2 2v1h16V6a2 2 0 0 2-2H4z"></path>
                        <path fill-rule="evenodd" d="M18 9H2v5a2 2 0 0 2 2h12a2 2 0 0 2-2V9zM4 13a1 1 0 0 1-1h1a1 1 0 1 110 2H5a1 1 0 0 1-1zm5-1a1 1 0 100 2h1a1 1 0 100-2H9z" clip-rule="evenodd"></path>
                    </svg>
                </div>
            </div>
            <div class="mb-6">
                <div class="flex flex-col sm:flex-row gap-4 items-start sm:items-center justify-between">
                    <div class="flex-1">

```

```

        <label for="selectCuenta" class="block text-sm font-medium text-gray-700 mb-2">Filtrar por Cuenta (opcional):</label>
        <select id="selectCuenta" class="w-full max-w-xs border border-gray-300 rounded-lg px-3 py-2 focus:ring-2 focus:ring-eubank-primary focus:border-transparent">
            <option value="">-- Todas las cuentas --</option>
        </select>
    </div>
    <div class="flex-shrink-0">
        <button id="btnCrearMovimiento"
            class="bg-gradient-to-r from-eubank-primary to-eubank-secondary text-white px-6 py-2 rounded-lg hover:shadow-lg transform hover:-translate-y-0.5 transition duration-200 font-medium inline-flex items-center">
            <svg class="w-5 h-5 mr-2" fill="none" stroke="currentColor" viewBox="0 0 24 24">
                <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 6v6m0 0v6m0-6h6m-6 0H6"></path>
            </svg>
            Crear
        </button>
    </div>
</div>

<div id="movimientosLoading" class="text-center py-10">
    <div class="loading-spinner"></div>
    <p class="text-eubank-primary font-medium">Cargando movimientos...</p>
</div>

<div id="movimientosContent" class="hidden">
    <div class="mb-4">
        <h6 class="text-lg font-semibold text-gray-800" id="movimientosTitle">Todos los Movimientos</h6>
    </div>
    <div class="overflow-x-auto">
        <table class="min-w-full divide-y divide-gray-200">
            <thead class="bg-gray-50">
                <tr>

```

```

        <th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Cuenta</th>
        <th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Número</th>
        <th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Fecha</th>
        <th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Tipo</th>
        <th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Referencia</th>
        <th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Importe</th>
        <th class="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Saldo Actual</th>
    </tr>
</thead>
<tbody id="movimientosTableBody" class="bg-white divide-y divide-gray-200">
    </tbody>
</table>
</div>
</div>

<div id="movimientosEmpty" class="bg-blue-100 border border-blue-400 text-blue-700 px-4 py-3 rounded-lg hidden">
    No hay movimientos disponibles.
</div>

<div id="movimientosError" class="bg-red-100 border border-red-400 text-red-700 px-4 py-3 rounded-lg hidden"></div>
</div>
</div>
</div>

<!-- Modal Crear Movimiento -->
<div id="modalCrearMovimiento" class="fixed inset-0 bg-gray-600 bg-opacity-50 overflow-y-auto h-full w-full hidden z-50">

```

```

<div class="relative top-20 mx-auto p-5 border w-11/12 max-w-[120vh] shadow-lg rounded-md bg-white">

  <div class="mt-3">
    <!-- Header -->
    <div class="bg-gradient-to-r from-eubank-primary to-eubank-secondary text-white px-6 py-4 rounded-t-md">
      <h3 class="text-lg font-bold">Crear Nuevo Movimiento</h3>
    </div>

    <!-- Formulario -->
    <div class="px-6 py-4">
      <div class="flex gap-6">
        <!-- Formulario a la izquierda -->
        <div class="flex-1">
          <form id="formCrearMovimiento">
            <div class="space-y-4">
              <!-- Tipo de Movimiento -->
              <div>
                <label for="tipoMovimiento" class="block text-sm font-medium text-gray-700 mb-2">Tipo de Movimiento:</label>
                <select id="tipoMovimiento" name="tipo" required
                      class="w-full border border-gray-300 rounded-lg px-3 py-2 focus:ring-2 focus:ring-eubank-primary focus:border-transparent">
                  <option value="">-- Seleccionar Tipo --</option>
                </select>
              </div>

              <!-- Cuenta Origen -->
              <div>
                <label for="cuentaOrigen" class="block text-sm font-medium text-gray-700 mb-2">Cuenta Origen:</label>
                <input type="text" id="cuentaOrigen" name="cuentaOrigen" required
                      placeholder="Ej: 00100001"
                      class="w-full border border-gray-300 rounded-lg px-3 py-2 focus:ring-2 focus:ring-eubank-primary focus:border-transparent">
              </div>
            </div>
          </form>
        </div>
      </div>
    </div>
  </div>
</div>

```

```

<!-- Cuenta Destino (solo para transferencias) -->
<div id="divCuentaDestino" class="hidden">
    <label for="cuentaDestino" class="block text-sm font-medium text-gray-700 mb-2">Cuenta Destino:</label>
    <input type="text" id="cuentaDestino" name="cuentaDestino"
        placeholder="Ej: 00200001"
        class="w-full border border-gray-300 rounded-lg px-3 py-2
        focus:ring-2 focus:ring-eubank-primary focus:border-transparent">
</div>

<!-- Monto -->
<div>
    <label for="monto" class="block text-sm font-medium text-gray-700 mb-2">Monto:</label>
    <input type="number" id="monto" name="importe" required
        step="0.01" min="0.01"
        placeholder="0.00"
        class="w-full border border-gray-300 rounded-lg px-3 py-2
        focus:ring-2 focus:ring-eubank-primary focus:border-transparent">
</div>
</div>

<!-- Mensaje de error -->
<div id="modalError" class="mt-4 bg-red-100 border border-red-400 text-red-700 px-4 py-3 rounded-lg hidden"></div>

<!-- Botones -->
<div class="flex justify-end space-x-3 mt-6">
    <button type="button" id="btnCancelarModal"
        class="px-4 py-2 bg-gray-300 text-gray-800 rounded-lg hover:bg-gray-400 transition duration-200">
        Cancelar
    </button>
    <button type="submit">

```



```

cargarCuentas();

// Manejar cambio de pestañas
$('.tab-button').click(function() {
    const tabName = $(this).data('tab');

    // Actualizar botones
    $('.tab-button').removeClass('active border-eubank-secondary text-eubank-secondary').addClass('border-transparent text-gray-500');
    $(this).addClass('active border-eubank-secondary text-eubank-secondary');

    // Mostrar contenido de pestaña
    $('.tab-pane').addClass('hidden');
    $('#' + tabName).removeClass('hidden');

    // Si es la pestaña de movimientos, cargar todos los movimientos
    if (tabName === 'movimientos') {
        cargarTodosMovimientos();
    }
});

// Evento para abrir modal de crear movimiento
$('#btnCrearMovimiento').click(function() {
    $('#modalCrearMovimiento').removeClass('hidden');
});

// Evento para cerrar modal
$('#btnCancelarModal').click(function() {
    cerrarModal();
});

// Cerrar modal al hacer clic fuera
$('#modalCrearMovimiento').click(function(e) {
    if (e.target === this) {
        cerrarModal();
    }
});

```

```

    }

});

// Evento para cambio de tipo de movimiento
$('#tipoMovimiento').change(function() {
    const tipo = $(this).val();

    // Mapeo de tipos de movimiento a imágenes
    const imagenesMovimiento = {
        'DEPOSITO': { img: '/images/Deposito.png', texto: 'Depósito' },
        'RETIRO': { img: '/images/Retiro.png', texto: 'Retiro' },
        'TRANSFERENCIA': { img: '/images/Transferencia.png', texto: 'Transferencia' },
    };

    if (tipo === 'TRANSFERENCIA') {
        $('#divCuentaDestino').removeClass('hidden');
        $('#cuentaDestino').attr('required', true);
    } else {
        $('#divCuentaDestino').addClass('hidden');
        $('#cuentaDestino').removeAttr('required').val("");
    }

    // Mostrar/ocultar imagen según el tipo seleccionado
    if (tipo && imagenesMovimiento[tipo]) {
        $('#imgTipoMovimiento').attr('src', imagenesMovimiento[tipo].img);
        $('#textoTipoMovimiento').text(imagenesMovimiento[tipo].texto);
        $('#imagenTipoMovimiento').removeClass('hidden');
    } else {
        $('#imagenTipoMovimiento').addClass('hidden');
    }
});

// Manejar envío del formulario de creación de movimiento
$('#formCrearMovimiento').submit(function(e) {

```

```

e.preventDefault();

const formData = {
    tipo: $('#tipoMovimiento').val(),
    cuentaOrigen: $('#cuentaOrigen').val(),
    cuentaDestino: $('#cuentaDestino').val(),
    importe: parseFloat($('#monto').val())
};

// Validación básica
if (!formData.tipo || !formData.cuentaOrigen || !formData.importe) {
    mostrarErrorModal('Por favor complete todos los campos requeridos.');
    return;
}

if ((formData.tipo === 'TRANSFERENCIA') && !formData.cuentaDestino) {
    mostrarErrorModal('Para transferencias se requiere la cuenta destino.');
    return;
}

// Enviar al servidor
crearMovimiento(formData);
});

});

// Event listeners for account filters
$('#filtroEstado, #filtroMoneda').on('change', function() {
    filtrarCuentas();
});

$('#busquedaCliente').on('keyup', function() {
    filtrarCuentas();
});

function verificarSesion() {

```

```

$.ajax({
    url: '@Url.Action("TestService", "Dashboard")',
    type: 'GET',
    success: function(data) {
        console.log('Test del servicio:', data);
    },
    error: function(xhr, status, error) {
        console.error('Error en test del servicio:', error);
    }
});

}

function cargarTiposMovimiento() {
    console.log('Cargando tipos de movimiento...');
    $.ajax({
        url: '@Url.Action("GetTiposMovimiento", "Dashboard")',
        type: 'GET',
        success: function(data) {
            console.log('Tipos de movimiento obtenidos:', data);
            tiposMovimiento = data;
            llenarSelectTiposMovimiento(data);
        },
        error: function(xhr, status, error) {
            console.error('Error al cargar tipos de movimiento:', error);
        }
    });
}

function llenarSelectTiposMovimiento(tipos) {
    const select = $('#tipoMovimiento');
    select.empty();
    select.append('<option value="">-- Seleccionar Tipo --</option>');
    tipos.forEach(function(tipo) {

```

```

        select.append(`<option value="${tipo}">${tipo}</option>`);

    });

}

function cargarCuentas() {
    console.log('Iniciando carga de cuentas...');
    $.ajax({
        url: '@Url.Action("GetCuentas", "Dashboard")',
        type: 'GET',
        success: function(data) {
            console.log('Respuesta del servidor:', data);
            if (data.error) {
                mostrarErrorCuentas(data.error);
            } else {
                cuentasData = data;
                mostrarCuentas(data);
                llenarSelectCuentas(data);
                llenarFiltroMoneda(data);
            }
        },
        error: function(xhr, status, error) {
            console.error('Error AJAX:', xhr.status, xhr.responseText, error);
            mostrarErrorCuentas('Error al cargar las cuentas: ' + error);
        }
    });
}

function mostrarCuentas(cuentas) {
    $('#cuentasLoading').addClass('hidden');
    $('#cuentasContent').removeClass('hidden');

    const tbody = $('#cuentasTableBody');
    tbody.empty();

```

```

if (cuentas.length === 0) {
    tbody.append(`<tr><td colspan="7" class="px-6 py-4 text-center text-gray-500">No
hay cuentas disponibles</td></tr>`);

    return;
}

cuentas.forEach(function(cuenta) {
    const estadoBadge = cuenta.estado === 'ACTIVO'
        ? '<span class="inline-flex items-center px-2.5 py-0.5 rounded-full text-xs font-
medium bg-green-100 text-green-800">Activo</span>'
        : '<span class="inline-flex items-center px-2.5 py-0.5 rounded-full text-xs font-
medium bg-red-100 text-red-800">Inactivo</span>';

    const row = `
        <tr class="hover:bg-gray-50">
            <td class="px-6 py-4 whitespace nowrap text-sm font-medium text-gray-
900">${cuenta.codigo}</td>
            <td class="px-6 py-4 whitespace nowrap text-sm text-gray-
900">${cuenta.nombreCliente}</td>
            <td class="px-6 py-4 whitespace nowrap text-sm text-gray-
500">${cuenta.emailCliente}</td>
            <td class="px-6 py-4 whitespace nowrap text-sm text-gray-
500">${cuenta.telefonoCliente}</td>
            <td class="px-6 py-4 whitespace nowrap text-sm text-gray-
900">${cuenta.moneda}</td>
            <td class="px-6 py-4 whitespace nowrap text-sm font-medium text-gray-
900">${formatearMoneda(cuenta.saldo, cuenta.moneda)}</td>
            <td class="px-6 py-4 whitespace nowrap">${estadoBadge}</td>
        </tr>
    `;
    tbody.append(row);
});

function mostrarErrorCuentas(mensaje) {
    $('#cuentasLoading').addClass('hidden');
    $('#cuentasError').text(mensaje).removeClass('hidden');
}

```

```

}

function llenarSelectCuentas(cuentas) {
    const select = $('#selectCuenta');
    select.empty();
    select.append('<option value="">-- Seleccione una cuenta --</option>');

    cuentas.forEach(function(cuenta) {
        select.append(`<option value="${cuenta.codigo}">${cuenta.codigo} - ${cuenta.nombreCliente}</option>`);
    });
}

function llenarFiltroMoneda(cuentas) {
    const select = $('#filtroMoneda');
    select.empty();
    select.append('<option value="">-- Todas --</option>');
    const monedas = [...new Set(cuentas.map(c => c.moneda))];
    monedas.forEach(function(moneda) {
        select.append(`<option value="${moneda}">${moneda}</option>`);
    });
}

function filtrarCuentas() {
    let filtered = cuentasData.slice();
    const estado = $('#filtroEstado').val();
    if (estado) {
        filtered = filtered.filter(c => c.estado === estado);
    }
    const moneda = $('#filtroMoneda').val();
    if (moneda) {
        filtered = filtered.filter(c => c.moneda === moneda);
    }
    const busqueda = $('#busquedaCliente').val().toLowerCase();
    if (busqueda) {

```

```

        filtered = filtered.filter(c => c.nombreCliente.toLowerCase().includes(busqueda));
    }
    mostrarCuentas(filtered);
}

// Evento cuando se selecciona una cuenta
$('#selectCuenta').change(function() {
    const codigoCuenta = $(this).val();

    if (codigoCuenta) {
        // Filtrar movimientos por cuenta específica
        cargarMovimientos(codigoCuenta);
    } else {
        // Mostrar todos los movimientos
        cargarTodosMovimientos();
    }
});

function cargarMovimientos(codigoCuenta) {
    cargarMovimientosCuentaEspecifica(codigoCuenta);
}

function cargarMovimientosCuentaEspecifica(codigoCuenta) {
    $('#movimientosEmpty').addClass('hidden');
    $('#movimientosContent').addClass('hidden');
    $('#movimientosError').addClass('hidden');
    $('#movimientosLoading').removeClass('hidden');

    $.ajax({
        url: '@Url.Action("GetMovimientos", "Dashboard")',
        type: 'GET',
        data: { cuenta: codigoCuenta },
        success: function(data) {
            if (data.error) {

```

```

        mostrarErrorMovimientos(data.error);
    } else {
        mostrarMovimientos(data, codigoCuenta, false);
    }
},
error: function(xhr, status, error) {
    mostrarErrorMovimientos('Error al cargar los movimientos: ' + error);
}
});
}

function cargarTodosMovimientos() {
    $('#movimientosEmpty').addClass('hidden');
    $('#movimientosContent').addClass('hidden');
    $('#movimientosError').addClass('hidden');
    $('#movimientosLoading').removeClass('hidden');

    $.ajax({
        url: '@Url.Action("GetTodosMovimientos", "Dashboard")',
        type: 'GET',
        success: function(data) {
            if (data.error) {
                mostrarErrorMovimientos(data.error);
            } else {
                mostrarMovimientos(data, null, true);
            }
        },
        error: function(xhr, status, error) {
            mostrarErrorMovimientos('Error al cargar todos los movimientos: ' + error);
        }
    });
}

function mostrarMovimientos(movimientos, codigoCuenta, todosLosMovimientos =
false) {

```

```

$('#movimientosLoading').addClass('hidden');
$('#movimientosContent').removeClass('hidden');

// Actualizar título
const titulo = todosLosMovimientos
    ? 'Todos los Movimientos'
    : `Movimientos de la Cuenta ${codigoCuenta}`;
$('#movimientosTitle').text(titulo);

const tbody = $('#movimientosTableBody');
tbody.empty();

if (movimientos.length === 0) {
    const mensaje = todosLosMovimientos
        ? 'No hay movimientos en ninguna cuenta'
        : `No hay movimientos para la cuenta ${codigoCuenta}`;
    tbody.append(`<tr><td colspan="7" class="px-6 py-4 text-center text-gray-500">${mensaje}</td></tr>`);
    return;
}

movimientos.forEach(function(mov) {
    const tipoBadge = obtenerBadgeTipo(mov.tipo);
    const fecha = new Date(mov.fecha).toLocaleDateString('es-ES');

    // Buscar información de la cuenta
    let cuentaInfo = "";
    if (todosLosMovimientos) {
        const cuenta = cuentasData.find(c => c.codigo === mov.codigoCuenta);
        cuentaInfo = cuenta ? `${cuenta.codigo} - ${cuenta.nombreCliente}` : mov.codigoCuenta;
    }
    else {
        // Cuando se filtra por cuenta específica, mostrar la info de esa cuenta
        const cuenta = cuentasData.find(c => c.codigo === codigoCuenta);
    }
})

```

```

        cuentaInfo = cuenta ? `${cuenta.codigo} - ${cuenta.nombreCliente}` :
codigoCuenta;
    }

const row = `

<tr class="hover:bg-gray-50">
    <td class="px-6 py-4 whitespace nowrap text-sm text-gray-900">${cuentaInfo}</td>
    <td class="px-6 py-4 whitespace nowrap text-sm font-medium text-gray-900">${mov.numero}</td>
    <td class="px-6 py-4 whitespace nowrap text-sm text-gray-900">${fecha}</td>
    <td class="px-6 py-4 whitespace nowrap">${tipoBadge}</td>
    <td class="px-6 py-4 whitespace nowrap text-sm text-gray-500">${mov.referencia || '-'}</td>
    <td class="px-6 py-4 whitespace nowrap text-sm font-medium text-gray-900">${formatearMoneda(mov.importe, 'USD')}</td>
    <td class="px-6 py-4 whitespace nowrap text-sm font-semibold text-green-600">${formatearMoneda(mov.saldoActual, 'USD')}</td>
</tr>
`;

tbody.append(row);
});

}

function mostrarErrorMovimientos(mensaje) {
    $('#movimientosLoading').addClass('hidden');
    $('#movimientosError').text(mensaje).removeClass('hidden');
}

function obtenerBadgeTipo(tipo) {
    const tiposMovimiento = {
        'Apertura de Cuenta': { texto: 'Apertura de Cuenta', color: 'bg-blue-100 text-blue-800' },
        'Deposito': { texto: 'Depósito', color: 'bg-green-100 text-green-800' },
        'Retiro': { texto: 'Retiro', color: 'bg-orange-100 text-orange-800' },
        'Cancelar Cuenta': { texto: 'Cancelar Cuenta', color: 'bg-red-100 text-red-800' },
        'Transferencia': { texto: 'Transferencia', color: 'bg-indigo-100 text-indigo-800' }
    }
}

```

```
};

const tipoInfo = tiposMovimiento[tipo] || { texto: tipo, color: 'bg-gray-100 text-gray-800' };

return `<span class="inline-flex items-center px-2.5 py-0.5 rounded-full text-xs font-medium ${tipoInfo.color}">${tipoInfo.texto}</span>`;

}

function formatearMoneda(cantidad, moneda) {
    let simbolo = '$'; // default
    if (moneda === '01') simbolo = 'S/';
    else if (moneda === '02') simbolo = '$';
    else if (moneda === 'USD') simbolo = '$';

    return simbolo + '' + Number(cantidad).toFixed(2);
}

function cerrarModal() {
    $('#modalCrearMovimiento').addClass('hidden');
    $('#formCrearMovimiento')[0].reset();
    $('#divCuentaDestino').addClass('hidden');
    $('#modalError').addClass('hidden');
    $('#imagenTipoMovimiento').addClass('hidden');
}

function mostrarErrorModal(mensaje) {
    $('#modalError').text(mensaje).removeClass('hidden');
}

function crearMovimiento(datos) {
    console.log('Creando movimiento:', datos);

    $.ajax({
        url: '@Url.Action("CrearMovimiento", "Dashboard")',

```

```

        type: 'POST',
        contentType: 'application/json',
        data: JSON.stringify(datos),
        success: function(response) {
            console.log('Respuesta del servidor:', response);
            if (response.success) {
                cerrarModal();
                // Refrescar automáticamente la vista de movimientos
                refrescarMovimientosActuales();
                alert('Movimiento creado exitosamente');
            } else {
                mostrarErrorModal(response.message || 'Error al crear el movimiento');
            }
        },
        error: function(xhr, status, error) {
            console.error('Error al crear movimiento:', error);
            mostrarErrorModal('Error al conectar con el servidor: ' + error);
        }
    });
}

// Función para refrescar la vista actual de movimientos
function refrescarMovimientosActuales() {
    const cuentaSeleccionada = $('#selectCuenta').val();

    if (cuentaSeleccionada) {
        // Si hay una cuenta seleccionada, recargar movimientos de esa cuenta
        cargarMovimientosCuentaEspecifica(cuentaSeleccionada);
    } else {
        // Si no hay cuenta seleccionada, recargar todos los movimientos
        cargarTodosMovimientos();
    }
}

```

```
}
```

```
</html>
```

4.5.4 EJECUCION

Para la ejecución del aplicativo web se procede a realizar lo siguiente.

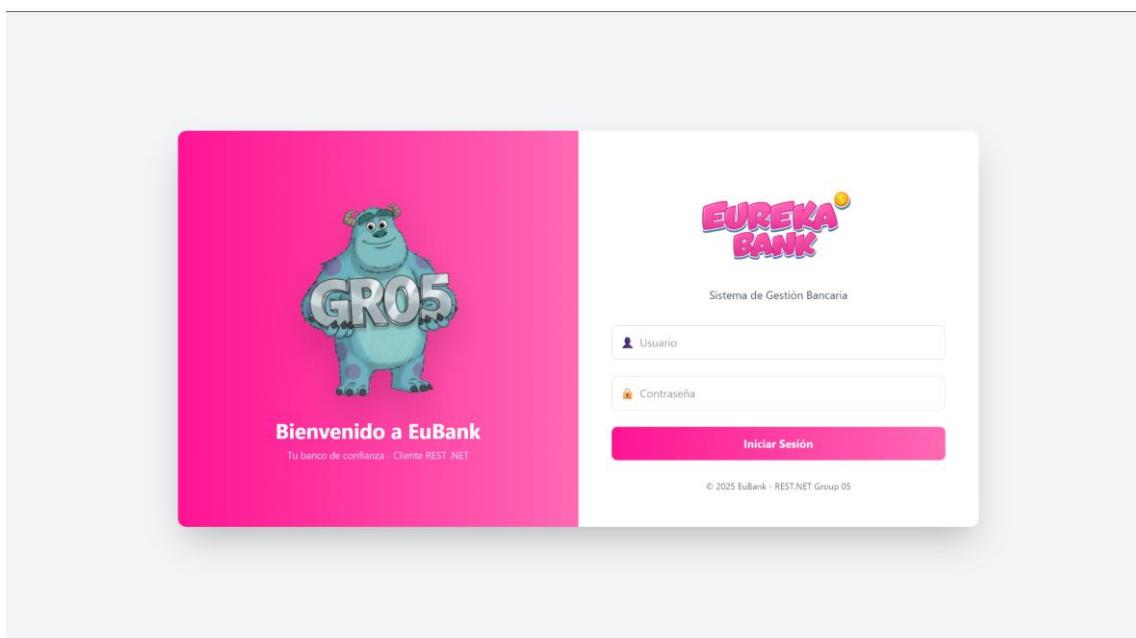


Figura 50 Ejecución Vista Principal

The screenshot shows a web-based application interface for managing bank accounts. At the top, there's a navigation bar with links for 'Panel de Control', 'Cuentas de Clientes' (selected), and 'Movimientos'. The main title is 'Lista de Cuentas' (List of Accounts). Below the title is a sub-section titled 'Gestión de Cuentas' with the subtitle 'Administra y consulta las cuentas bancarias'. There are three filter dropdowns: 'Filtrar por Estado' (Todos), 'Filtrar por Moneda' (Todas), and 'Buscar por Cliente' (empty input field). A table below lists account details:

CÓDIGO	CLIENTE	EMAIL	TELÉFONO	MONEDA	SALDO	ESTADO
00100001	ALAN ALBERTO ARANDA LUNA	a.aranda@hotmail.com	834-67125	01	\$/ 6900.00	Activo
00100002	ALAN ALBERTO ARANDA LUNA	a.aranda@hotmail.com	834-67125	02	\$ 4500.00	Activo
00200001	ROSA LIZET FLORES CHAFLOQUE	r.flerez@hotmail.com	966-67567	01	\$/ 7000.00	Activo
00200002	ERIC GUSTAVO CORONEL CASTILLO	g.coronel@vialbcp.com	9666-4457	01	\$/ 6800.00	Activo
00200003	EDGAR RAFAEL CHAVEZ CANALES	e.chavez@gmail.com	999-96673	02	\$ 6000.00	Activo
00300001	GABRIEL ALEJANDRO GONZALES GARCIA	g.gonzales@yahoo.es	945-56782	01	\$/ 0.00	Inactivo

Figura 51 Ejecución Vista Consultar Clientes

The screenshot shows a web-based application interface for viewing bank movements. At the top, there's a navigation bar with links for 'Panel de Control', 'Cuentas de Clientes' (selected), and 'Movimientos'. The main title is 'Movimientos Bancarios' (Bank Movements). Below the title is a sub-section titled 'Movimientos Bancario' with the subtitle 'Revisa los movimientos realizados entre las cuentas bancarias'. There's a dropdown filter 'Filtrar por Cuenta (opcional)' with the option '-- Seleccione una cuenta --' and a '+ Crear' button. A table below lists all movements:

CUENTA	NÚMERO	FECHA	TIPO	REFERENCIA	IMPORTE	SALDO ACTUAL
00100001 - ALAN ALBERTO ARANDA LUNA	7	15/3/2008	Depósito	-	\$ 1000.00	\$ 6900.00
00200003 - EDGAR RAFAEL CHAVEZ CANALES	6	11/3/2008	Retiro	-	\$ -500.00	\$ 6000.00
00100002 - ALAN ALBERTO ARANDA LUNA	4	8/3/2008	Depósito	-	\$ 1500.00	\$ 4500.00
00200002 - ERIC GUSTAVO CORONEL CASTILLO	3	6/3/2008	Retiro	-	\$ -1200.00	\$ 6800.00
00100001 - ALAN ALBERTO ARANDA LUNA	6	3/3/2008	Retiro	-	\$ -800.00	\$ 5900.00
00100001 - ALAN ALBERTO ARANDA LUNA	5	25/2/2008	Retiro	-	\$ -500.00	\$ 6700.00

Figura 52 Ejecución Vista Consultar Movimientos

Crear Nuevo Movimiento

Tipo de Movimiento:

DEPOSITO

Cuenta Origen:

Ej: 00100001

Monto:

0.00

Cancelar **Crear Movimiento** **Depósito**



Figura 53 Ejecución Vista Realizar Depósitos

Crear Nuevo Movimiento

Tipo de Movimiento:

TRANSFERENCIA

Cuenta Origen:

Ej: 00100001

Cuenta Destino:

Ej: 00200001

Monto:

0.00

Cancelar **Crear Movimiento** **Transferencia**



Figura 54 Ejecución Vista Realizar Transferencia



Figura 55 Ejecución Vista Retiro

4.6 CREACIÓN DEL PROYECTO MÓVIL

4.6.1 CREACIÓN PROYECTO MÓVIL

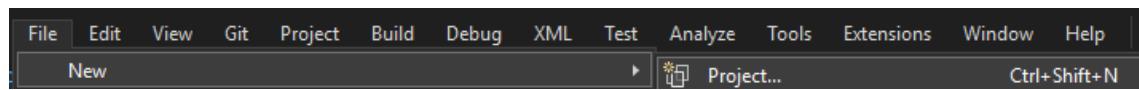


Figura 56 Creación Nuevo Proyecto desde File



Figura 57 Creación Nuevo Proyecto Selección Herramienta

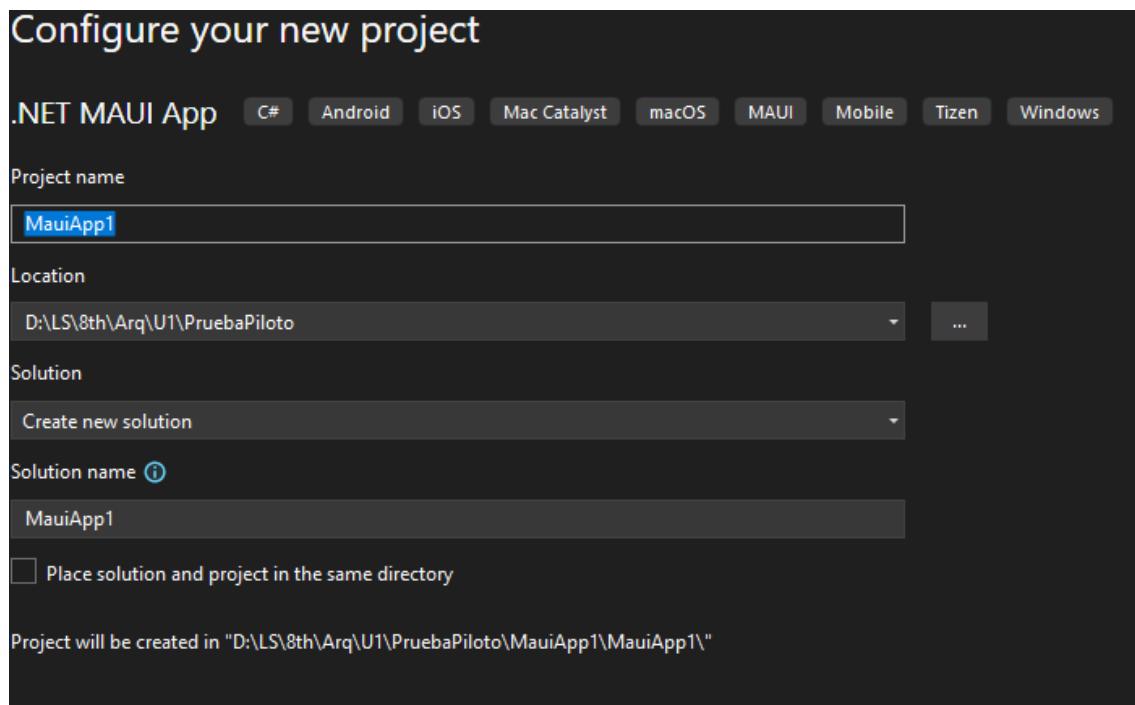


Figura 58 Configuración el nombre de la solución

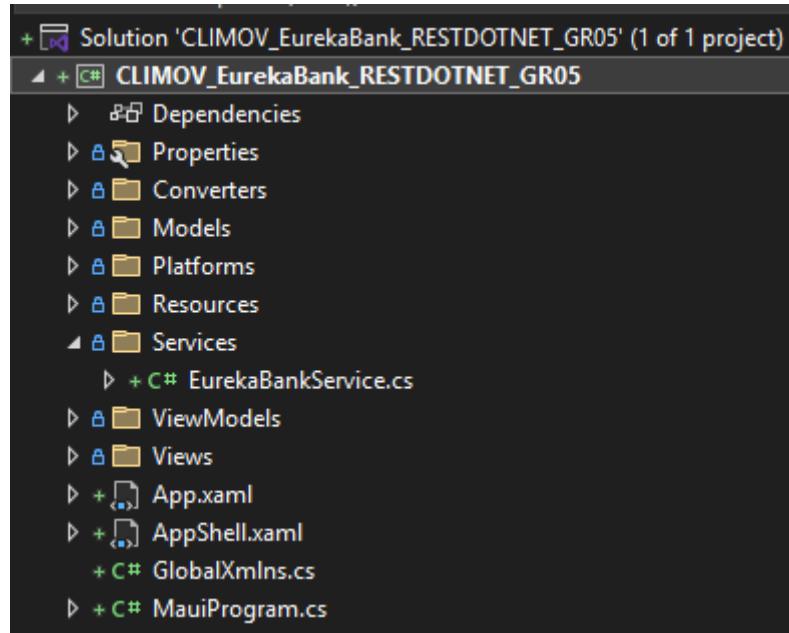


Figura 59 Configuración Nuevo Proyecto

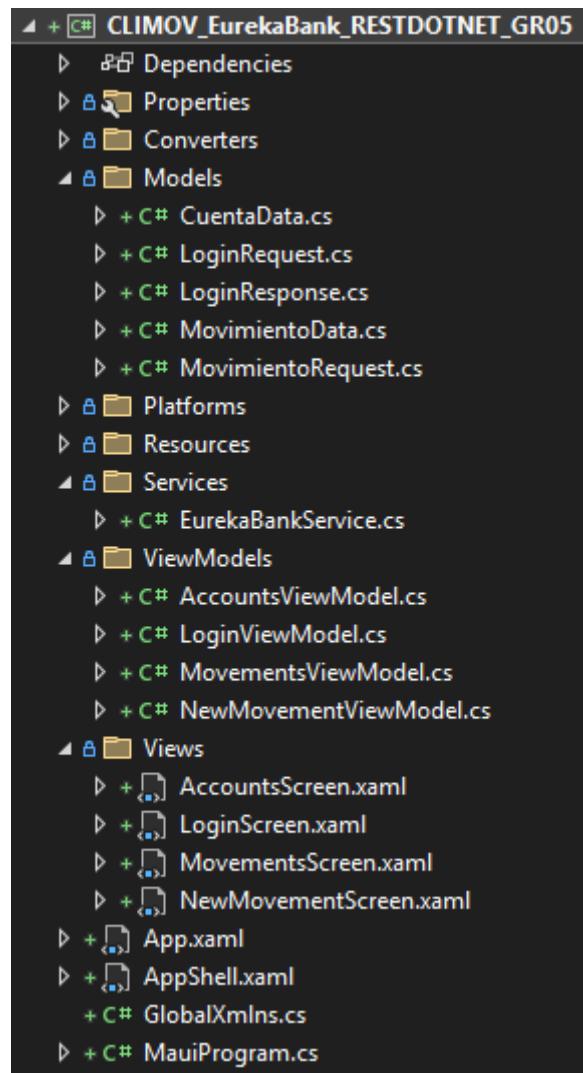


Figura 60 Estructura proyecto

4.6.2 CREACIÓN CAPA MODELO

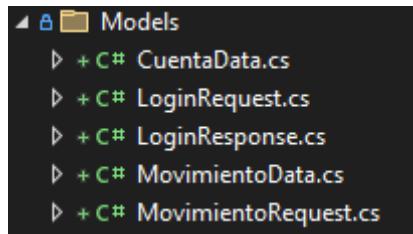


Figura 61 Creación nueva clase en paquete modelo

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CLIMOV_EurekaBank_RESTDOTNET_GR05.Models
{
    public class MovimientoData
    {
        public int Numero { get; set; }
        public DateTime Fecha { get; set; }
        public string Tipo { get; set; }
        public decimal Importe { get; set; }
        public string Referencia { get; set; }
        public string CodigoCuenta { get; set; }
        public string NombreCliente { get; set; }
        public decimal SaldoActual { get; set; }
    }
}
```

Figura 62 Codificación clase MovimientoData

TABLA 12 Codificación clase MovimientoData

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace CLIMOV_EurekaBank_RESTDOTNET_GR05.Models
{
    public class MovimientoData
    {
        public int Numero { get; set; }
        public DateTime Fecha { get; set; }
        public string Tipo { get; set; }
        public decimal Importe { get; set; }
        public string Referencia { get; set; }
        public string CodigoCuenta { get; set; }
        public string NombreCliente { get; set; }
        public decimal SaldoActual { get; set; }
    }
}
```

Luego se pasa a realizar la creación de la carpeta Models, parte de nuestro patrón MVC.

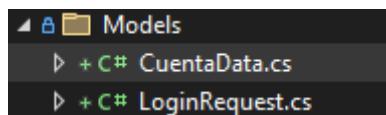


Figura 63 Creación clase CuentaData

```
2     using System.Collections.Generic;
3     using System.Linq;
4     using System.Text;
5     using System.Threading.Tasks;
6
7     namespace CLIMOV_EurekaBank_RESTDOTNET_GR05.Models
8     {
9         public class CuentaData
10        {
11            public string Código { get; set; }
12            public string Moneda { get; set; }
13            public decimal Saldo { get; set; }
14            public string Estado { get; set; }
15            public string NombreCliente { get; set; }
16            public string EmailCliente { get; set; }
17            public string TelefonoCliente { get; set; }
18        }
19    }
```

Figura 64 Codificación clase CuentaData

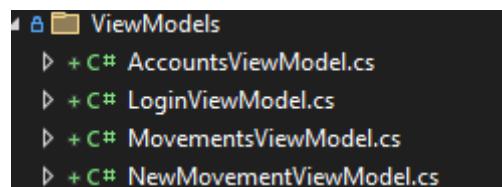
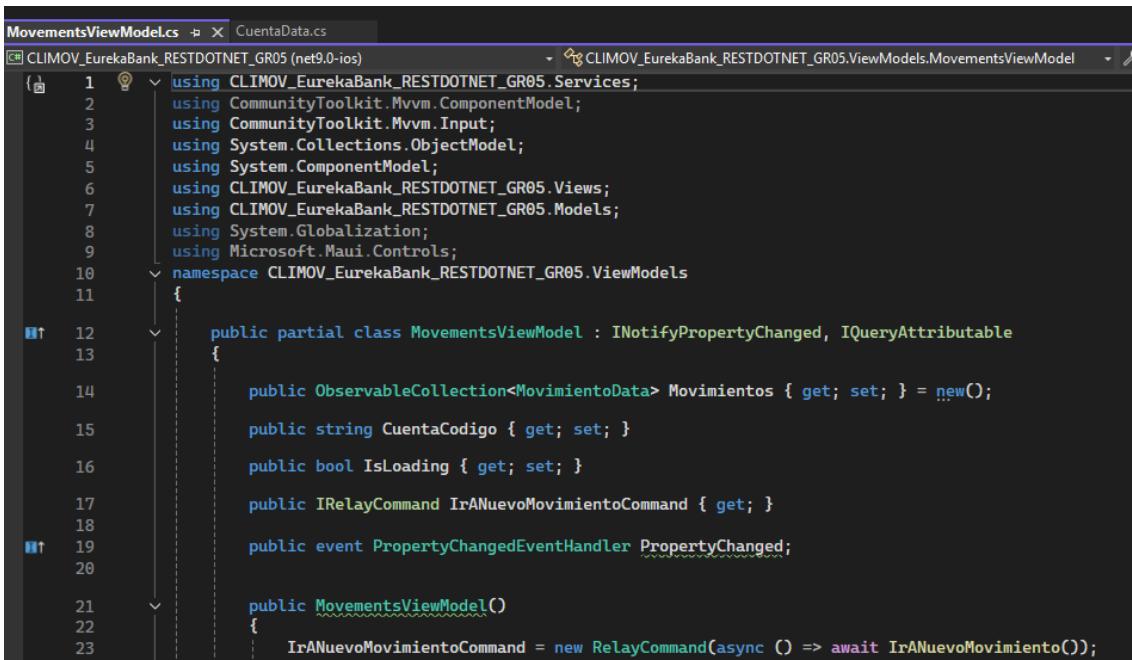


Figura 65 Creación modelos de carga de las vistas

Se crea el modelo para la carga de los movimientos



```
1  using CLIMOV_EurekaBank_RESTDOTNET_GR05.Services;
2  using CommunityToolkit.Mvvm.ComponentModel;
3  using CommunityToolkit.Mvvm.Input;
4  using System.Collections.ObjectModel;
5  using System.ComponentModel;
6  using CLIMOV_EurekaBank_RESTDOTNET_GR05.Views;
7  using CLIMOV_EurekaBank_RESTDOTNET_GR05.Models;
8  using System.Globalization;
9  using Microsoft.Maui.Controls;
10 using namespace CLIMOV_EurekaBank_RESTDOTNET_GR05.ViewModels;
11 {
12     public partial class MovementsViewModel : INotifyPropertyChanged, IQueryAttributable
13     {
14         public ObservableCollection<MovimientoData> Movimientos { get; set; } = new();
15         public string CuentaCodigo { get; set; }
16         public bool IsLoading { get; set; }
17         public IRelayCommand IrANuevoMovimientoCommand { get; }
18         public event PropertyChangedEventHandler PropertyChanged;
19
20         public MovementsViewModel()
21         {
22             IrANuevoMovimientoCommand = new RelayCommand(async () => await IrANuevoMovimiento());
23         }
24     }
25 }
```

Figura 66 Modelo para Carga de Movimientos

TABLA 13 Codificación clase MovementsViewModel

```
using CLIMOV_EurekaBank_RESTDOTNET_GR05.Services;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using System.Collections.ObjectModel;
using System.ComponentModel;
using CLIMOV_EurekaBank_RESTDOTNET_GR05.Views;
using CLIMOV_EurekaBank_RESTDOTNET_GR05.Models;
using System.Globalization;
using Microsoft.Maui.Controls;
namespace CLIMOV_EurekaBank_RESTDOTNET_GR05.ViewModels
{
    public partial class MovementsViewModel : INotifyPropertyChanged, IQueryAttributable
    {
        public ObservableCollection<MovimientoData> Movimientos { get; set; } = new();
        public string CuentaCodigo { get; set; }
        public bool IsLoading { get; set; }
    }
}
```

```
public IRelayCommand IrANuevoMovimientoCommand { get; }

public event PropertyChangedEventHandler PropertyChanged;

public MovementsViewModel()
{
    IrANuevoMovimientoCommand = new RelayCommand(async () => await
IrANuevoMovimiento());
}

private async Task IrANuevoMovimiento()
{
    if (string.IsNullOrEmpty(CuentaCodigo))
        return;

    // Navegar a la vista NewMovementScreen y pasar el código de cuenta
    //await
Shell.Current.GoToAsync($"//{nameof(NewMovementScreen)}?cuenta={CuentaCodigo}",
true);
    await
Shell.Current.GoToAsync($"{nameof(NewMovementScreen)}?cuenta={CuentaCodigo}",
true);
}

public async void ApplyQueryAttributes(IDictionary<string, object> query)
{
    if (query.TryGetValue("cuenta", out var cuenta))
    {
        CuentaCodigo = cuenta.ToString();
        OnPropertyChanged(nameof(CuentaCodigo));
        await CargarMovimientosAsync();
    }
    else
    {
        // Si ya tenías la cuenta cargada, recarga igual por si hubo cambios
    }
}
```

```

        if (!string.IsNullOrEmpty(CuentaCodigo))
            await CargarMovimientosAsync();
    }

}

private async Task CargarMovimientosAsync()
{
    try
    {
        IsLoading = true;
        OnPropertyChanged(nameof(IsLoading));

        Movimientos.Clear();

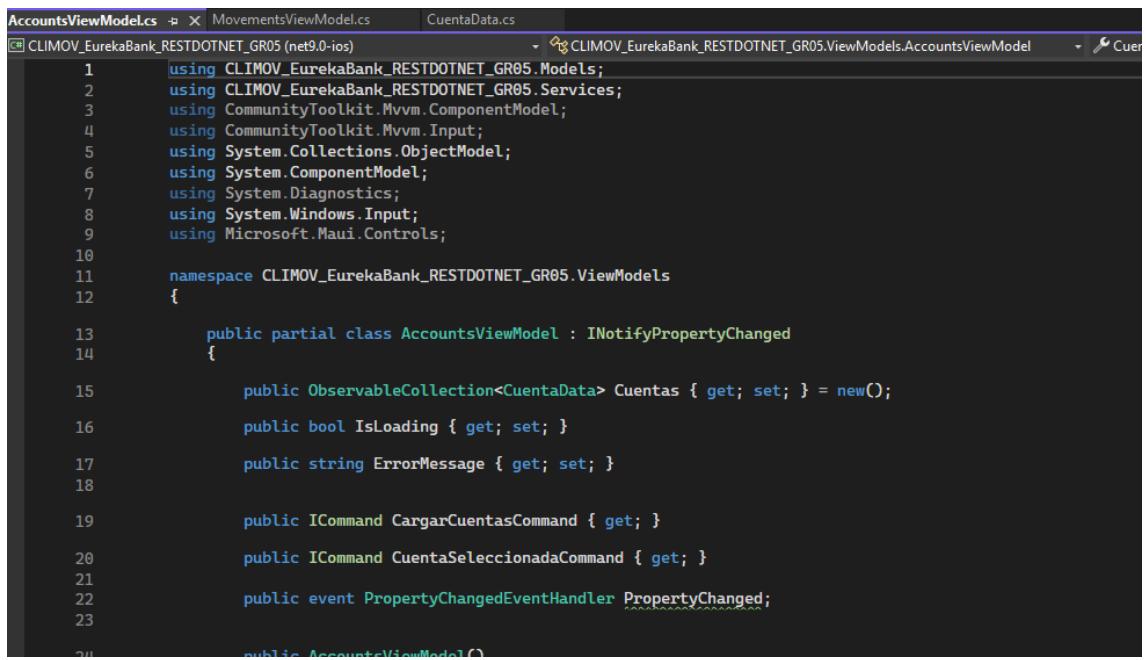
        var movimientos = await
EurekaBankService.ObtenerMovimientosAsync(CuentaCodigo);

        if (movimientos != null && movimientos.Any())
        {
            foreach (var mov in movimientos)
                Movimientos.Add(mov);
        }
        else
        {
            await Application.Current.MainPage.DisplayAlert(
                "Aviso",
                "No se encontraron movimientos para esta cuenta.",
                "OK"
            );
        }
    }
    catch (Exception ex)
    {
        await Application.Current.MainPage.DisplayAlert(
            "Error",

```

```
$"No se pudieron cargar los movimientos.\nDetalles: {ex.Message}",  
    "OK"  
);  
}  
finally  
{  
    IsLoading = false;  
    OnPropertyChanged(nameof(IsLoading));  
}  
}  
  
private void OnPropertyChanged(string propertyName)  
    => PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));  
}  
}
```

Se crea el modelo para la carga de datos en la vista de cuentas



```
1  using CLIMOV_EurekaBank_RESTDOTNET_GR05.Models;
2  using CLIMOV_EurekaBank_RESTDOTNET_GR05.Services;
3  using CommunityToolkit.Mvvm.ComponentModel;
4  using CommunityToolkit.Mvvm.Input;
5  using System.Collections.ObjectModel;
6  using System.ComponentModel;
7  using System.Diagnostics;
8  using System.Windows.Input;
9  using Microsoft.Maui.Controls;
10
11 namespace CLIMOV_EurekaBank_RESTDOTNET_GR05.ViewModels
12 {
13     public partial class AccountsViewModel : INotifyPropertyChanged
14     {
15         public ObservableCollection<CuentaData> Cuentas { get; set; } = new();
16         public bool IsLoading { get; set; }
17         public string ErrorMessage { get; set; }
18
19         public ICommand CargarCuentasCommand { get; }
20         public ICommand CuentaSeleccionadaCommand { get; }
21
22         public event PropertyChangedEventHandler PropertyChanged;
23
24         public AccountsViewModel()
25     }
26
27 }
```

Figura 67 Carga de datos en Cuentas

TABLA 14 Codificación clase AccountsViewModel

```
using CLIMOV_EurekaBank_RESTDOTNET_GR05.Models;
using CLIMOV_EurekaBank_RESTDOTNET_GR05.Services;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Diagnostics;
using System.Windows.Input;
using Microsoft.Maui.Controls;

namespace CLIMOV_EurekaBank_RESTDOTNET_GR05.ViewModels
{
    public partial class AccountsViewModel : INotifyPropertyChanged
    {
        public ObservableCollection<CuentaData> Cuentas { get; set; } = new();
```

```

public bool IsLoading { get; set; }

public string ErrorMessage { get; set; }

public ICommand CargarCuentasCommand { get; }

public ICommand CuentaSeleccionadaCommand { get; }

public event PropertyChangedEventHandler PropertyChanged;

public AccountsViewModel()
{
    CargarCuentasCommand = new Command(async () => await CargarCuentasAsync());
    CuentaSeleccionadaCommand = new Command<CuentaData>(async (cuenta) =>
    await OnCuentaSeleccionada(cuenta));

    _ = CargarCuentasAsync(); // Carga automática al iniciar
}

private async Task OnCuentaSeleccionada(CuentaData cuentaSeleccionada)
{
    if (cuentaSeleccionada == null)
        return;

    Console.WriteLine($"⌚ EUREKABANK Cuenta seleccionada:
{cuentaSeleccionada.Codigo}");

    await
Shell.Current.GoToAsync($"movimientos?cuenta={cuentaSeleccionada.Codigo}");
}

private async Task CargarCuentasAsync()
{
    try
    {
        IsLoading = true;
        ErrorMessage = string.Empty;
        OnPropertyChanged(nameof(IsLoading));
        OnPropertyChanged(nameof(ErrorMessage));
    }
}

```

```

// ◊ Llamar al nuevo método del servicio
var cuentas = await EurekaBankService.ObtenerCuentasAsync();

Cuentas.Clear();

if (cuentas == null || cuentas.Count == 0)
{
    ErrorMessage = "No se encontraron cuentas o ocurrió un error al obtenerlas.";
    OnPropertyChanged(nameof(ErrorMessage));
    return;
}

// ◊ Agregar las cuentas a la colección observable
foreach (var cuenta in cuentas)
{
    Cuentas.Add(cuenta);
}

catch (Exception ex)
{
    // ◊ Capturar cualquier excepción inesperada
    ErrorMessage = $"Error al cargar las cuentas: {ex.Message}";
    OnPropertyChanged(nameof(ErrorMessage));
}

finally
{
    IsLoading = false;
    OnPropertyChanged(nameof(IsLoading));
}
}

private void OnPropertyChanged(string propName)
=> PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propName));
}

```



4.6.3. CREACIÓN CAPA SERVICIO

En esta clase se crea la conexión entre el Cliente Móvil y el servidor utilizando la librería HttpClient para el intercambio de datos.

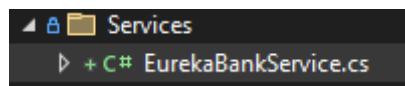


Figura 68 Creación clase EurekaBankService

```
using CLIMOV_EurekaBank_RESTDOTNET_GR05.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CLIMOV_EurekaBank_RESTDOTNET_GR05.Services
{
    public class EurekaBankService
    {
        private static readonly HttpClient _httpClient = new HttpClient
        {
            BaseAddress = new Uri("http://192.168.137.1:62279")
        };
        private static readonly JsonSerializerOptions _jsonOptions = new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true
        };

        // ● LOGIN
        public static async Task<bool success, string message> IniciarSesionAsync(string usuario, string clave)
        {
            try
            {
                var request = new LoginRequest { usuario = usuario, clave = clave };
                string json = JsonSerializer.Serialize(request);
                var response = await _httpClient.PostAsJsonAsync("api/Login", json);
                var result = await response.Content.ReadAsStringAsync();
                return JsonConvert.DeserializeObject<bool>(result);
            }
            catch (Exception ex)
            {
                return false;
            }
        }
    }
}
```

Figura 69 Instancia de clase EurekaBankService

TABLA 15 Codificación clase EurekaBankService

```
using CLIMOV_EurekaBank_RESTDOTNET_GR05.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```

using System.Text.Json;
using System.Threading.Tasks;

namespace CLIMOV_EurekaBank_RESTDOTNET_GR05.Services
{
    public class EurekaBankService
    {
        private static readonly HttpClient _httpClient = new HttpClient
        {
            BaseAddress = new Uri("http://192.168.137.1:62279")
        };
        private static readonly JsonSerializerOptions _jsonOptions = new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true
        };

        // 🔒 LOGIN -----
        public static async Task<(bool success, string message)> IniciarSesionAsync(string usuario, string clave)
        {
            try
            {
                var request = new LoginRequest { usuario = usuario, clave = clave };
                string json = JsonSerializer.Serialize(request);
                var content = new StringContent(json, Encoding.UTF8, "application/json");

                HttpResponseMessage response = await _httpClient.PostAsync("/api/login", content);

                if (!response.IsSuccessStatusCode)
                    return (false, "Error al conectar con el servidor");

                string responseContent = await response.Content.ReadAsStringAsync();
                var loginResponse = JsonSerializer.Deserialize<LoginResponse>(responseContent, _jsonOptions);
            }
        }
    }
}

```

```

        return loginResponse?.success == true
            ? (true, "Inicio de sesión exitoso ✅")
            : (false, "Usuario o clave incorrectos ❌");
    }

    catch (Exception ex)
    {
        return (false, $"Error: {ex.Message}");
    }
}

// 📋 OBTENER CUENTAS -----
public static async Task<List<CuentaData>> ObtenerCuentasAsync()
{
    try
    {
        HttpResponseMessage response = await _httpClient.GetAsync("/api/cuentas");

        if (!response.IsSuccessStatusCode)
            return new List<CuentaData>();

        string content = await response.Content.ReadAsStringAsync();
        var cuentas = JsonSerializer.Deserialize<List<CuentaData>>(content, _jsonOptions);

        return cuentas ?? new List<CuentaData>();
    }
    catch
    {
        return new List<CuentaData>();
    }
}

// 📋 OBTENER CUENTA POR CÓDIGO -----
public static async Task<CuentaData?> ObtenerCuentaAsync(string codigoCuenta)

```

```

{
    try
    {
        HttpResponseMessage response = await _httpClient.GetAsync($""/api/cuentas/{codigoCuenta}");
        if (!response.IsSuccessStatusCode)
            return null;

        string content = await response.Content.ReadAsStringAsync();
        var cuenta = JsonSerializer.Deserialize<CuentaData>(content, _jsonOptions);

        return cuenta;
    }
    catch
    {
        return null;
    }
}

// ⚙ OBTENER MOVIMIENTOS -----
public static async Task<List<MovimientoData>> ObtenerMovimientosAsync(string? cuenta = null)
{
    try
    {
        string url = string.IsNullOrEmpty(cuenta)
            ? "/api/movimientos"
            : $""/api/movimientos?cuenta={cuenta}";

        HttpResponseMessage response = await _httpClient.GetAsync(url);

        if (!response.IsSuccessStatusCode)
            return new List<MovimientoData>();

        string content = await response.Content.ReadAsStringAsync();
    }
}

```

```

        var movimientos = JsonSerializer.Deserialize<List<MovimientoData>>(content,
_jsonOptions);

        return movimientos ?? new List<MovimientoData>();
    }
    catch
    {
        return new List<MovimientoData>();
    }
}

// ⚙ PROCESAR MOVIMIENTO -----
public static async Task<(bool success, string message)>
ProcesarMovimientoAsync(MovimientoRequest request)
{
    try
    {
        string json = JsonSerializer.Serialize(request);
        var content = new StringContent(json, Encoding.UTF8, "application/json");

        HttpResponseMessage response = await
_httpClient.PostAsync("/api/movimientos", content);

        if (response.IsSuccessStatusCode)
            return (true, "Movimiento procesado correctamente ✅");

        string error = await response.Content.ReadAsStringAsync();
        return (false, $"Error: {error}");
    }
    catch (Exception ex)
    {
        return (false, $"Excepción: {ex.Message}");
    }
}

```

```

// ⚙ LISTAR TIPOS DE MOVIMIENTO -----
public static async Task<List<string>> ObtenerTiposMovimientoAsync()
{
    try
    {
        HttpResponseMessage response = await
_httpClient.GetAsync("/api/tiposmovimiento");

        if (!response.IsSuccessStatusCode)
            return new List<string>();

        string content = await response.Content.ReadAsStringAsync();
        var tipos = JsonSerializer.Deserialize<List<string>>(content, _jsonOptions);

        return tipos ?? new List<string>();
    }
    catch
    {
        return new List<string>();
    }
}

}

```

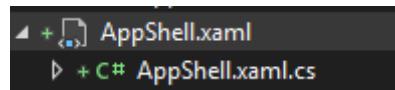


Figura 70 Modificación clase AppShell

TABLA 16 Codificación clase AppShell

```
<?xml version="1.0" encoding="UTF-8" ?>
<Shell
    x:Class="CLIMOV_EurekaBank_RESTDOTNET_GR05.AppShell"
    xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:views="clr-namespace:CLIMOV_EurekaBank_RESTDOTNET_GR05.Views"
    xmlns:local="clr-namespace:CLIMOV_EurekaBank_RESTDOTNET_GR05"
    Title="CLIMOV_EurekaBank_RESTDOTNET_GR05">

    <!-- Ruta principal: Login -->
    <ShellContent
        Route="Login"
        ContentTemplate="{DataTemplate views:LoginScreen}" />

    <ShellContent
        Route="Cuentas"
        ContentTemplate="{DataTemplate
views:AccountsScreen}" />

</Shell>
```

4.6.4 Creación Capa Vista

Aquí se programa la lógica de las interfaces donde el usuario podrá interactuar.

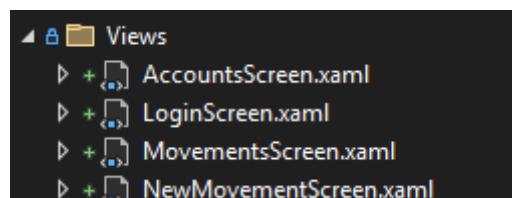


Figura 71 Creación archivos en carpeta Views

TABLA 17 Codificación archivo AccountsScreen.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:viewmodels="clr-
namespace:CLIMOV_EurekaBank_RESTDOTNET_GR05.ViewModels"
    x:Class="CLIMOV_EurekaBank_RESTDOTNET_GR05.Views.AccountsScreen"
    Title="Mis Cuentas"
    BackgroundColor="{StaticResource BackgroundColor}">

    <ContentPage.BindingContext>
        <viewmodels:AccountsViewModel />
    </ContentPage.BindingContext>

    <ScrollView>
        <VerticalStackLayout Padding="20" Spacing="15">

            <!-- Encabezado -->
            <HorizontalStackLayout HorizontalOptions="Center" Margin="0,10,0,20">
                <Image Source="sulli_general.png"
                    WidthRequest="60"
                    HeightRequest="60"
                    Aspect="AspectFit" />
                <Label Text="Mis Cuentas"
                    VerticalOptions="Center"
                    FontSize="24"
                    FontAttributes="Bold"
                    TextColor="{StaticResource AccentColor}"
                    Margin="10,0,0,0"/>
            </HorizontalStackLayout>

            <!-- Indicador de carga -->
            <ActivityIndicator IsRunning="{Binding IsLoading}">
```

```

        IsVisible="{Binding isLoading}"
        Color="{StaticResource SecondaryColor}" />

    <!-- Mensaje de error -->
    <Label Text="{Binding ErrorMessage}"
        TextColor="{StaticResource ErrorColor}"
        FontAttributes="Bold"
        IsVisible="{Binding ErrorMessage, Converter={StaticResource NullToBoolConverter}}"
        HorizontalTextAlignment="Center"
        Margin="0,5,0,10" />

    <!-- Lista de cuentas -->
    <CollectionView ItemsSource="{Binding Cuentas}">
        <CollectionView.ItemTemplate>
            <DataTemplate>
                <Frame Margin="8,6"
                    Padding="15"
                    BorderColor="{StaticResource BorderColor}"
                    BackgroundColor="{StaticResource CardBackgroundColor}"
                    CornerRadius="15"
                    >
                    <Frame.GestureRecognizers>
                        <TapGestureRecognizer
                            Command="{Binding Source={RelativeSource AncestorType={x:Type viewmodels:AccountsViewModel}}, Path=CuentaSeleccionadaCommand}"
                            CommandParameter="{Binding .}" />
                    </Frame.GestureRecognizers>

                    <VerticalStackLayout>
                        <Label Text="{Binding NombreCliente}"
                            FontAttributes="Bold"
                            FontSize="18"
                            TextColor="{StaticResource TextColor}" />
                
```

```

<Label Text="{Binding Código}"
       FontSize="14"
       TextColor="{StaticResource PlaceholderColor}" />
<Label Text="{Binding Moneda}"
       FontSize="14"
       TextColor="{StaticResource PlaceholderColor}" />
<Label Text="{Binding Saldo, StringFormat='Saldo: {0:C}'}"
       FontAttributes="Bold"
       FontSize="16"
       TextColor="{StaticResource SecondaryColor}" />
</VerticalStackLayout>
</Frame>
</DataTemplate>
</CollectionView.ItemTemplate>
</CollectionView>

</VerticalStackLayout>
</ScrollView>
</ContentPage>

```

TABLA 18 Codificación archivo LoginScreen.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
              xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
              xmlns:viewmodels="clr-
namespace:CLIMOV_EurekaBank_RESTDOTNET_GR05.ViewModels"
              x:Class="CLIMOV_EurekaBank_RESTDOTNET_GR05.Views.LoginScreen"
              Title="Iniciar Sesión"
              BackgroundColor="{StaticResource BackgroundColor}">

<ContentPage.BindingContext>

```

```

<viewmodels:LoginViewModel />
</ContentPage.BindingContext>

<!-- ScrollView para permitir desplazamiento -->
<ScrollView>
    <!-- Layout principal del formulario -->
    <VerticalStackLayout
        Spacing="20"
        Padding="40,60"
        HorizontalOptions="Fill"
        VerticalOptions="CenterAndExpand"
        BackgroundColor="{StaticResource CardBackgroundColor}"
        Margin="20"
        MinimumWidthRequest="360"
        MaximumWidthRequest="480">

        <!-- LOGO PRINCIPAL -->
        <Image Source="sulli_logo.png"
            HeightRequest="120"
            WidthRequest="120"
            HorizontalOptions="Center"
            Margin="0,0,0,8" />

        <!-- LOGO SECUNDARIO + TÍTULO -->
        <VerticalStackLayout Spacing="6" HorizontalOptions="Center">
            <Image Source="rest_dotnet.png"
                HeightRequest="160"
                Aspect="AspectFit"
                HorizontalOptions="Center" />
            <Label Text="Sistema de Gestión Bancaria"
                FontSize="18"
                FontAttributes="Bold"
                TextColor="{StaticResource PrimaryColor}"
                HorizontalTextAlignment="Center" />
    
```

```
</VerticalStackLayout>

<!-- USUARIO -->
<Entry Placeholder="Usuario"
      Text="{Binding Usuario}"
      HeightRequest="50"
      FontSize="16"
      BackgroundColor="{StaticResource InputBackgroundColor}"
      TextColor="{StaticResource EntryTextColor}"
      PlaceholderColor="{StaticResource PlaceholderColor}"
      ClearButtonVisibility="WhileEditing" />

<!-- CONTRASEÑA -->
<Entry Placeholder="Contraseña"
      Text="{Binding Clave}"
      IsPassword="True"
      HeightRequest="50"
      FontSize="16"
      BackgroundColor="{StaticResource InputBackgroundColor}"
      TextColor="{StaticResource EntryTextColor}"
      PlaceholderColor="{StaticResource PlaceholderColor}"
      ClearButtonVisibility="WhileEditing" />

<!-- BOTÓN LOGIN -->
<Button Text="Iniciar Sesión"
       Command="{Binding LoginCommand}"
       HeightRequest="55"
       CornerRadius="12"
       FontSize="18"
       FontAttributes="Bold"
       BackgroundColor="{StaticResource PrimaryColor}"
       TextColor="{StaticResource ButtonTextColor}"
       Margin="0,10,0,0" />
```

```

<!-- INDICADOR DE CARGA -->
<ActivityIndicator IsRunning="{Binding IsLoading}"
    IsVisible="{Binding IsLoading}"
    Color="{StaticResource PrimaryColor}"
    HeightRequest="30"
    Margin="0,10,0,0" />

<!-- MENSAJE DE ERROR -->
<Label Text="{Binding Mensaje}"
    FontSize="14"
    HorizontalTextAlignment="Center"
    Margin="0,10,0,0">
    <Label.Triggers>
        <DataTrigger TargetType="Label" Binding="{Binding EsExitoso}" Value="True">
            <Setter Property="TextColor" Value="{StaticResource SuccessColor}" />
        </DataTrigger>
        <DataTrigger TargetType="Label" Binding="{Binding EsExitoso}" Value="False">
            <Setter Property="TextColor" Value="{StaticResource ErrorColor}" />
        </DataTrigger>
    </Label.Triggers>
</Label>

</VerticalStackLayout>
</ScrollView>
</ContentPage>

```

TABLA 19 Codificación archivo MovementsScreen.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui">

```

```

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
xmlns:viewmodels="clr-
namespace:CLIMOV_EurekaBank_RESTDOTNET_GR05.ViewModels"
x:Class="CLIMOV_EurekaBank_RESTDOTNET_GR05.Views.MovementsScreen"
Title="Movimientos"
BackgroundColor="{StaticResource BackgroundColor}">

<ContentPage.BindingContext>
    <viewmodels:MovementsViewModel />
</ContentPage.BindingContext>

<Grid RowDefinitions="Auto,*">

    <!-- ENCABEZADO -->
    <VerticalStackLayout Grid.Row="0"
        Spacing="6"
        Padding="0,25,0,10"
        HorizontalOptions="Center"
        VerticalOptions="Center">
        <Image Source="sulli_general.png"
            HeightRequest="55"
            Aspect="AspectFit"
            HorizontalOptions="Center" />
        <Label Text="Movimientos de Cuenta"
            FontSize="22"
            FontAttributes="Bold"
            TextColor="{StaticResource SecondaryColor}"
            HorizontalTextAlignment="Center" />
        <Label Text="{Binding CuentaCodigo}"
            FontAttributes="Bold"
            FontSize="18"
            TextColor="{StaticResource TextColor}"
            HorizontalTextAlignment="Center" />
    </VerticalStackLayout>

```

```

<!-- CONTENIDO PRINCIPAL SCROLLABLE -->
<Grid Grid.Row="1">
    <ScrollView>
        <VerticalStackLayout Padding="20" Spacing="12">

            <!-- Indicador de carga -->
            <ActivityIndicator IsRunning="{Binding IsLoading}"
                IsVisible="{Binding IsLoading}"
                Color="{StaticResource SecondaryColor}"
                HeightRequest="40"
                HorizontalOptions="Center" />

            <!-- Mensaje de error -->
            <Label Text="{Binding ErrorMessage}"
                TextColor="{StaticResource ErrorColor}"
                FontSize="14"
                HorizontalTextAlignment="Center"
                Margin="0,5,0,10"
                IsVisible="{Binding ErrorMessage,
                    Converter={StaticResource NullToBoolConverter}}" />

            <!-- LISTA DE MOVIMIENTOS -->
            <CollectionView ItemsSource="{Binding Movimientos}">
                <CollectionView.ItemsLayout>
                    <GridItemsLayout Orientation="Vertical" Span="1" />
                </CollectionView.ItemsLayout>

                <CollectionView.ItemTemplate>
                    <DataTemplate>
                        <Frame Margin="8,6"
                            Padding="16"
                            CornerRadius="16"

                            BorderColor="{StaticResource BorderColor}"
                            BackgroundColor="{StaticResource CardBackgroundColor}">

```

```

<VerticalStackLayout Spacing="4">
    <Label Text="{Binding Tipo}"
        FontAttributes="Bold"
        FontSize="18"
        TextColor="{StaticResource AccentColor}" />

    <Label Text="{Binding Importe, StringFormat='Importe: {0:C}'}"
        FontAttributes="Bold"
        FontSize="16"
        TextColor="{Binding Importe, Converter={StaticResource ImporteToColorConverter}}" />

    <Label Text="{Binding SaldoActual, StringFormat='Saldo actual: {0:C}'}"
        FontSize="14"
        TextColor="{StaticResource SecondaryColor}" />

    <Label Text="{Binding Fecha, StringFormat='Fecha: {0:dd/MM/yyyy HH:mm}'}"
        FontSize="13"
        TextColor="{StaticResource PlaceholderColor}" />

    <Label Text="{Binding Referencia}"
        FontSize="13"
        FontAttributes="Italic"
        TextColor="{StaticResource TextColor}" />
</VerticalStackLayout>
</Frame>
</DataTemplate>
</CollectionView.ItemTemplate>
</CollectionView>

</VerticalStackLayout>
</ScrollView>

```

```

<!-- BOTÓN FLOTANTE -->
<Button Text="+" 
    Command="{Binding IrANuevoMovimientoCommand}" 
    CornerRadius="35" 
    WidthRequest="70" 
    HeightRequest="70" 
    FontSize="28" 
    FontAttributes="Bold" 
    TextColor="White" 
    BackgroundColor="{StaticResource AccentColor}" 
    HorizontalOptions="End" 
    VerticalOptions="End" 
    Margin="0,0,15,15" 
/>
</Grid>
</ContentPage>

```

TABLA 20 Codificación archivo NewMovementScreen.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui" 
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" 
    xmlns:viewmodels="clr-
    namespace:CLIMOV_EurekaBank_RESTDOTNET_GR05.ViewModels" 
    x:Class="CLIMOV_EurekaBank_RESTDOTNET_GR05.Views.NewMovementScreen" 
    Title="Nuevo Movimiento" 
    BackgroundColor="{StaticResource BackgroundColor}">

<ContentPage.BindingContext>

```

```

<viewmodels:NewMovementViewModel />
</ContentPage.BindingContext>

<ScrollView>
    <VerticalStackLayout Padding="25" Spacing="16">

        <!-- IMAGEN DEL TIPO DE MOVIMIENTO -->
        <Image Source="{Binding ImagenMovimiento}"
            HeightRequest="120"
            Aspect="AspectFit"
            HorizontalOptions="Center"
            Margin="0,10,0,20" />

        <!-- TÍTULO -->
        <Label Text="Registrar Movimiento"
            FontSize="22"
            FontAttributes="Bold"
            TextColor="{StaticResource SecondaryColor}"
            HorizontalTextAlignment="Center" />

        <!-- INDICADOR DE CARGA -->
        <ActivityIndicator IsRunning="{Binding IsLoading}"
            IsVisible="{Binding IsLoading}"
            Color="{StaticResource SecondaryColor}"
            HeightRequest="35"
            HorizontalOptions="Center" />

        <!-- CUENTA ORIGEN -->
        <Label Text="Cuenta Origen:" FontAttributes="Bold" />
        <Entry Text="{Binding CuentaOrigen}" IsReadOnly="True" />

        <!-- TIPO DE MOVIMIENTO -->
        <Label Text="Tipo de Movimiento:" FontAttributes="Bold" />
        <Picker ItemsSource="{Binding TiposMovimiento}" />
    
```

```

        SelectedItem="{Binding TipoSeleccionado}"
        Title="Selecciona el tipo"
        BackgroundColor="{StaticResource InputBackgroundColor}"
        TextColor="{StaticResource TextColor}"/>

        <!-- SOLO SI ES TRANSFERENCIA -->
        <VerticalStackLayout IsVisible="{Binding MostrarCuentaDestino}">
            <Label Text="Cuenta Destino:" FontAttributes="Bold" />
            <Entry Placeholder="Ej: 00100002"
                Text="{Binding CuentaDestino}" />
        </VerticalStackLayout>

        <!-- IMPORTE -->
        <Label Text="Importe:" FontAttributes="Bold" />
        <Entry Placeholder="0.00"
            Keyboard="Numeric"
            Text="{Binding Importe}" />

        <!-- MENSAJE DE ERROR / INFORMACIÓN -->
        <Label Text="{Binding Mensaje}"
            TextColor="{StaticResource ErrorColor}"
            FontSize="14"
            HorizontalTextAlignment="Center"
            Margin="0,5,0,10"
            IsVisible="{Binding Mensaje, Converter={StaticResource NullToBoolConverter}}"/>

        <!-- BOTONES -->
        <Button Text="Registrar"
            Command="{Binding RegistrarMovimientoCommand}"
            BackgroundColor="{StaticResource AccentColor}"
            TextColor="White"
            CornerRadius="12"
            FontAttributes="Bold"
            Margin="0,10,0,0" />

```

```
<Button Text="Cancelar"  
       BackgroundColor="{StaticResource InputBackgroundColor}"  
       TextColor="{StaticResource PlaceholderColor}"  
       CornerRadius="12"  
       Command="{Binding CancelarCommand}" />  
</VerticalStackLayout>  
</ScrollView>  
</ContentPage>
```

4.6.4 EJECUCIÓN

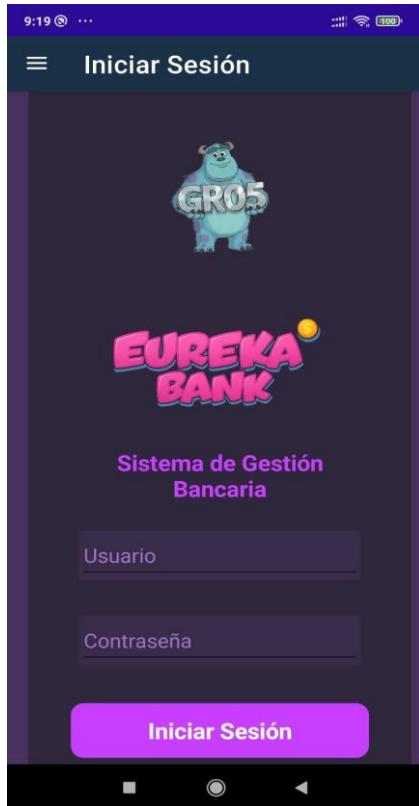


Figura 72 Ejecución de Pantalla de Inicio de Sesión

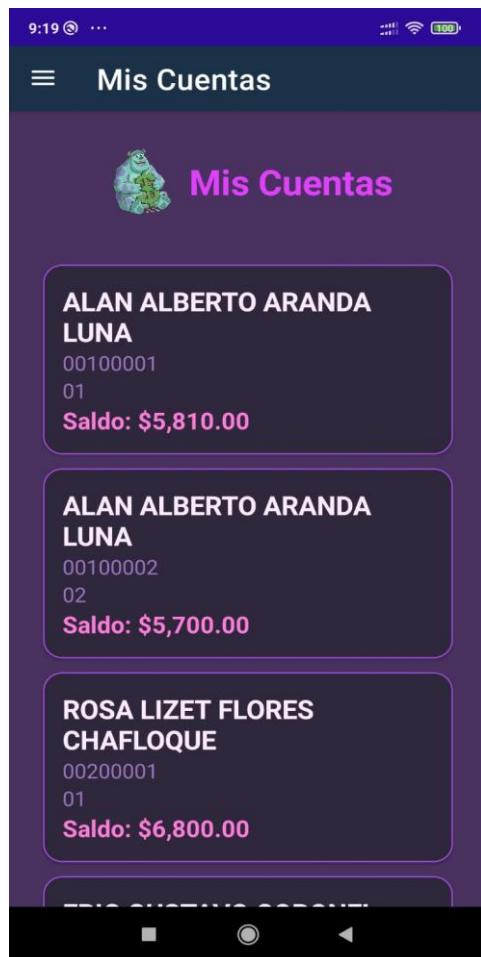


Figura 73 Ejecución Pantalla Principal



Figura 74 Ejecución Pantalla Depósitos



Figura 75 Ejecución Pantalla Consultas de una cuenta en específico



Figura 76 Ejecución Pantalla Retiro



Figura 77 Ejecución Pantalla Transferencia

5. CONCLUSIONES

Las API Web implementadas bajo el paradigma REST (Representational State Transfer) en el lenguaje C# y desplegadas en servidores de alto rendimiento como Kestrel/IIS, representan una solución de elevada confiabilidad y solidez para contextos empresariales modernos. Su valor fundamental radica en la capacidad de garantizar la interoperabilidad, velocidad y escalabilidad en las transacciones de datos. Al adherirse a estándares abiertos como el formato JSON para el intercambio de datos y utilizando la especificación OpenAPI/Swagger para la documentación, estas API facilitan una integración efectiva a través de diversas plataformas y lenguajes, manteniendo la compatibilidad en sistemas heterogéneos.

La utilización de *frameworks* especializados como ASP.NET Core Web API optimiza notablemente el proceso de desarrollo y despliegue, permitiendo que las API satisfagan las rigurosas demandas de aplicaciones críticas en sectores como el financiero, sanitario y logístico. Esto consolida a RESTful .NET como una elección arquitectónica estratégica para sistemas que requieren altos niveles de cumplimiento normativo, rendimiento y estabilidad operativa, aprovechando la eficiencia del entorno .NET Core.

Por otra parte, la adopción del patrón de diseño MVC (Modelo-Vista-Controlador), en conjunto con las capacidades de las API REST y la plataforma ASP.NET Core, posibilita la creación de un diseño arquitectónico modular y escalable. Este enfoque asegura una estricta separación de responsabilidades (lógica de negocio, presentación de datos y procesamiento de solicitudes).

Dicha modularidad no solo incrementa la mantenibilidad del sistema a largo plazo, sino que también minimiza el impacto de futuras adaptaciones o modificaciones, contribuyendo así a un ciclo de vida de la aplicación más eficiente y sostenible.

6. RECOMENDACIONES

- Se recomienda encarecidamente la utilización de servidores Kestrel o IIS en conjunto con la última versión de .NET Core. Es esencial configurar el entorno para maximizar la seguridad (mediante el uso de *tokens* JWT, por ejemplo) y el rendimiento de la API Web RESTful, asegurando que la plataforma esté alineada con las mejores prácticas de *hosting* empresarial.
- Es fundamental adoptar y explotar eficientemente las capacidades del *framework* ASP.NET Core Web API para la definición e implementación de *endpoints* RESTful. Esto incluye el uso correcto de Entity Framework Core para el acceso optimizado a la base de datos SQL Server y la implementación de la documentación a través de OpenAPI/Swagger.
- Proveer formación técnica continua a los desarrolladores en conceptos arquitectónicos clave, como los principios de diseño REST, la serialización JSON, la seguridad de las API, y la implementación rigurosa del patrón MVC (Modelo-Vista-Controlador), garantizando así una base de código consistente, segura y escalable.