



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	8
2. OBJETIVOS	8
2.1 Objetivo General	8
2.2 Objetivos Específicos.....	8
3. MARCO TEORICO.....	9
3.1 Web Services SOAP y su Arquitectura.....	9
3.2 Patrón de Arquitectura Modelo-Vista-Controlador (MVC).....	9
3.3 Base de Datos en la Nube: Google Cloud MySQL.....	9
3.4 Payara y Java para Servicios SOAP	10
4. DESARROLLO DEL APlicATIVO	11
4.1 INSTALACIÓN DE HERRAMIENTAS.....	11
4.1.1 HERRAMIENTAS NECESARIAS.....	11
4.1.2 CREACIÓN DE BD en la NUBE	11
4.2 APlicACIÓN SERVIDOR.....	16
4.2.1 MODELO MVC	18
4.2.2 CONEXIÓN A BASE DE DATOS	19
4.2.3 PRUEBA DE CONEXIÓN	23
4.2.4 CODIFICACIÓN DE CLASES	27
4.2.5 CREACIÓN DEL SERVICIO WEB	66
4.3 CREACIÓN DEL PROYECTO CONSOLA	69
4.3.1 ESTRUCTURA MVC.....	72
4.3.2 CODIFICACIÓN DE CLASES	75
4.3.3 CREACIÓN DEL CONTROLADOR.....	76
4.3.4 EJECUCIÓN.....	92
4.4 CREACIÓN DEL PROYECTO CLIENTE ESCRITORIO	93
4.4.1 CREACIÓN DE LA CAPA DE SERVICIO	95
4.4.2 CREACIÓN DE LA CAPA DE PRUEBAS	104
4.4.3 CREACIÓN DEL CONTROLADOR.....	107

4.4.4 CREACIÓN DE LA VISTA	112
4.5 CREACIÓN DEL PROYECTO WEB	159
4.5.1 CREACIÓN ENTORNO.....	159
4.5.2 CREACIÓN CAPA SERVICIO	160
4.5.3 CREACIÓN CAPA VISTA	162
4.5.4 EJECUCION.....	189
4.6 CREACIÓN DEL PROYECTO MÓVIL.....	192
4.6.1 CREACIÓN PROYECTO MÓVIL.....	192
4.6.2 CREACIÓN CAPA MODELO.....	194
4.6.3. CREACIÓN CAPA SERVICIO	203
4.6.4 Creación Capa Vista.....	215
4.6.4 EJECUCIÓN.....	235
5. CONCLUSIONES	238
6. RECOMENDACIONES	239

ÍNDICE DE IMÁGENES

Figura 1 MySQL WorkBench.....	11
Figura 2 Estructura inicial del proyecto.....	11
Figura 3 Pantalla de Cloud SQL.....	12
Figura 4 Ingreso Gmail.....	12
Figura 5 Configuración de Cloud SQL	13
Figura 6 Configuración de Cloud SQL	13
Figura 7 Tablas de la base de datos.....	14
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	17
Figura 12 Configuración Frameworks	17
Figura 13 Estructura de servidor	18
Figura 14 Creación de los paquetes	18
Figura 15 Estructura de los paquetes.....	18
Figura 16 Creación de la clase AccesoDB	19
Figura 17 Codificación de la conexión de la base de datos.....	19
Figura 18 Creación de la clase de pruebas para la conexión	23
Figura 19 Codificación clase de prueba.....	23
Figura 20 Comprobación del Driver	25
Figura 21 Bibliotecas del proyecto	25
Figura 22 Librería del conector	25
Figura 23 Conector integrado en el proyecto	26
Figura 24 Resultado de la prueba de conexión a la base	26
Figura 25 Creación de la clase movimiento	27
Figura 26 Creación Del servicio	66
Figura 27 Configuración de nombre.....	66
Figura 28 Codificación de los servicios web Movimientos.....	67
Figura 29 Implementación de etiquetas	67
Figura 30 Codificación del servicio web Depósito.....	68

Figura 31 Adición de tags	68
Figura 32 Despliegue de la aplicación	68
Figura 33 Inicio del servidor	69
Figura 34 Verificación de los servicios web creados	69
Figura 35 Creación Java Application Consola	70
Figura 36 Creación proyecto estructura general.....	70
Figura 37 Creación Web Service Client	71
Figura 38 Servicios WSDL generados en el cliente.....	71
Figura 39 Creación paquete ec.edu.monster.controlador	72
Figura 40 Creación paquete ec.edu.monster.servicio.....	73
Figura 41 Creación paquete ec.edu.monster.db.....	73
Figura 42 Creación paquete ec.edu.monster.modelo	74
Figura 43 Estructura proyecto cliente	74
Figura 44 Creación clase Movimiento	75
Figura 45 Codificación clase Movimiento	75
Figura 46 Creación clase EurekaService	76
Figura 47 Codificación clase EurekaService.....	76
Figura 48 Creación clase EurekaController	76
Figura 49 Codificación clase EurekaController.....	77
Figura 50 Creación clase WSEurekaClient.....	82
Figura 51 Codificación clase WSEurekaClient	82
Figura 52 Ejecución clase WSEurekaClient.....	92
Figura 53 Ejecución clase WSEurekaClient.....	93
Figura 54 Creación del proyecto cliente.....	94
Figura 55 Creación del cliente del servicio web	94
Figura 56 Estructura inicial del proyecto.....	95
Figura 57 Creación de la clase EurekaService	95
Figura 58 Llamada a los métodos del servicio web.....	95
Figura 59 Generación de la capa de pruebas	104
Figura 60 Clases de pruebas.....	104
Figura 61 Creación del controlador.....	107

Figura 62 Creación de las vistas	112
Figura 63 Prueba a la interfaz gráfica principal.....	112
Figura 64 Prueba a la interfaz gráfica de gestión de cuentas	113
Figura 65 Prueba a la interfaz gráfica de consulta de movimientos bancarios	113
Figura 66 Creación directorio 02.CLIENTE WEB	159
Figura 67 Creación Estructura Web	159
Figura 68 Ejecución Vista Inicio de Sesión	189
Figura 69 Ejecución Vista Principal.....	189
Figura 70 Ejecución Vista Consultar Movimientosv	190
Figura 71 Ejecución Vista Realizar Depósitos.....	190
Figura 72 Ejecución Vista Realizar Transferencia.....	191
Figura 73 Ejecución Vista Retiro.....	191
Figura 74 Creación Nuevo Proyecto.....	192
Figura 75 Creación Empty Activity	192
Figura 76 Configuración Nuevo Proyecto	192
Figura 77 Estructura proyecto.....	193
Figura 78 Creación nueva clase en paquete modelo	194
Figura 79 Codificación clase MovimientoData.....	194
Figura 80 Creación clase DatosCuenta	195
Figura 81 Codificación clase DatosCuenta	196
Figura 82 Creación clases de MovementsViewModel	198
Figura 83 Modelos de Carga de Datos	199
Figura 84 Modelo de Vista Carga Cuentas	201
Figura 85 Creación clase EurekaBankService	203
Figura 86 Instancias de clase EurekaBankService	204
Figura 87 Modificación clase MainActivity.....	213
Figura 88 Creación archivos en la carpeta screens	215
Figura 89 Ejecución Pantalla Inicio de Sesión Móvil	235
Figura 90 Ejecución Pantalla Principal Móvil.....	236
Figura 91 Ejecución Pantalla Depósitos	236
Figura 92 Ejecución Pantalla Consultas de una cuenta en específico.....	237

Figura 93 Ejecución Pantalla Retiro.....	237
Figura 94 Ejecución Pantalla Transferencia	238

ÍNDICE DE TABLAS

TABLA 1 Clase Acceso Base de Datos	20
TABLA 2 Clase de prueba de conexión	24
TABLA 3 Codificación clase Movimiento	28
TABLA 4 Codificación Clase EurekaService	37
TABLA 5 Codificación controlador	78
TABLA 6 Codificación Clase WSEurekaClient	83
TABLA 7 Clase llamada a servicios web	96
TABLA 8 Clase llamada a servicios web	105
TABLA 9 Codificación controlador	108
TABLA 10 ConsultaView	114
TABLA 11 MainView	130
TABLA 12 DepositoView	146
TABLA 13 Codificación App.py	160
TABLA 14 Codificación Vista Principal	162
TABLA 15 Codificación realizarDepósitos	165
TABLA 16 Codificación clase MovimientoData	194
TABLA 17 Codificación clase DatosCuenta	197
TABLA 18 Codificación clase MovementsViewModel	200
TABLA 19 Codificación clase AccountViewModel	202
TABLA 20 Codificación clase EurekaBankService	204
TABLA 21 Codificación clase MainActivity	214
TABLA 22 Codificación archivo AccountsScreen	215
TABLA 23 Codificación archivo LoginScreen	219
TABLA 24 Codificación archivo MovementScreen	224
TABLA 25 Codificación archivo NuevoMovimientoModal	229

1. INTRODUCCIÓN

El **Web Service SOAP (Simple Object Access Protocol)** se establece como una tecnología esencial para asegurar la interoperabilidad entre distintas aplicaciones. Su arquitectura se fundamenta en el **intercambio de mensajes en formato XML**, definiendo un protocolo estandarizado que regula la estructura de los mensajes y la forma en que los servicios deben ser invocados. La principal ventaja de SOAP radica en su **total independencia de la plataforma y el lenguaje de programación**, lo que lo convierte en una opción idónea para entornos empresariales donde la seguridad rigurosa y la comunicación efectiva entre sistemas diversos son cruciales.

En el marco de la arquitectura de *software*, la operatividad de los servicios SOAP depende inherentemente del **WSDL (Web Services Description Language)**. Este lenguaje es fundamental, ya que ofrece una descripción detallada de la interfaz del servicio, incluyendo la especificación de los métodos accesibles, los parámetros necesarios y los tipos de datos involucrados, lo cual garantiza una interacción bien definida y precisa entre los módulos cliente y servidor.

Para el desarrollo y la utilización de servicios SOAP dentro del ambiente Java, se emplean **frameworks** robustos como **Jakarta EE** (previamente Java EE) o librerías específicas como **Apache CXF** y **JAX-WS (Java API for XML Web Services)**. Estas herramientas facilitan el proceso de desarrollo al proporcionar el soporte requerido para las especificaciones SOAP y WSDL, permitiendo su despliegue en servidores de aplicaciones como Apache Tomcat o **Payara**.

En cuanto a la gestión y persistencia de la información de esta solución, se ha optado por utilizar una base de datos **Google Cloud MySQL**. Esta decisión aprovecha la reconocida estabilidad del sistema MySQL junto con los beneficios de la computación en la nube, ofreciendo características como escalabilidad dinámica, flexibilidad operativa, alta disponibilidad y un acceso seguro a distancia.

2. OBJETIVOS

2.1 Objetivo General

El objetivo de este trabajo es **documentar, mediante la creación de un manual técnico**, las etapas de diseño e implementación de un **Conversor de Unidades** capaz de manejar conversiones de longitud y peso. La solución se construirá sobre una arquitectura de **Servicio Web SOAP** desarrollado en Java, el cual será alojado en el servidor de aplicaciones **Payara**. Para la persistencia de datos, se empleará **Google Cloud MySQL** en un entorno de nube. Adicionalmente, se incluirá el desarrollo de clientes de prueba que permitan acceder al servicio desde diversas plataformas, tales como **escritorio, consola, web y móvil**.

2.2 Objetivos Específicos

- **Detalle del Servicio:** Describir minuciosamente el proceso de diseño e implementación del servicio SOAP, especificando el uso del lenguaje Java, el servidor de aplicaciones

Payara y la gestión de datos mediante Google Cloud MySQL. Se incluirá la presentación de casos prácticos que ilustren la funcionalidad de conversión de unidades.

- Guías de Integración de Clientes: Ofrecer directrices metodológicas para el desarrollo de las aplicaciones cliente en las diversas plataformas (escritorio, consola, web y móvil), haciendo hincapié en la integración efectiva y el consumo del servicio SOAP.
- Fundamentación Teórica: Clarificar los fundamentos conceptuales esenciales relacionados con la arquitectura SOAP, el estándar WSDL y las tecnologías implementadas, con el propósito de que el lector adquiera una comprensión profunda de su relevancia estratégica en el desarrollo de soluciones de software contemporáneas.

3. MARCO TEORICO

3.1 Web Services SOAP y su Arquitectura

Un Web Service SOAP es un servicio web que utiliza el protocolo SOAP (Simple Object Access Protocol) para intercambiar mensajes estructurados en formato XML entre aplicaciones. Las principales características de SOAP incluyen:

- Estandarización: Los mensajes SOAP están bien definidos por un formato XML estándar, lo que asegura interoperabilidad entre plataformas y lenguajes de programación.
- Independencia de transporte: SOAP puede utilizar diversos protocolos, como HTTP, SMTP o TCP, para la transmisión de mensajes.
- Contrato explícito: Los servicios SOAP utilizan WSDL (Web Services Description Language) para describir las operaciones disponibles, los parámetros aceptados y los tipos de datos utilizados.
- Extensibilidad: Soporta características avanzadas como WS-Security para seguridad, transacciones distribuidas y manejo detallado de errores.

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

El patrón Modelo-Vista-Controlador (MVC) organiza las aplicaciones en tres componentes principales:

- Modelo: Representa los datos y la lógica de negocio. En el caso de un servicio SOAP, el modelo incluye las operaciones definidas en el WSDL y su implementación en el servidor.
- Vista: Presenta los datos al usuario final. En aplicaciones SOAP, puede ser una interfaz gráfica o web que consume los datos enviados por el servicio.
- Controlador: Gestiona las solicitudes entrantes, procesa las operaciones en el modelo y actualiza la vista en consecuencia

3.3 Base de Datos en la Nube: Google Cloud MySQL

Google Cloud MySQL se presenta como un servicio de gestión de bases de datos completamente administrado que amalgama la funcionalidad inherente de MySQL con las ventajas inherentes de una infraestructura en la nube. Estas características clave son:

- **Disponibilidad Garantizada:** El servicio asegura una **alta disponibilidad** mediante la replicación automática de los datos, 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 aplicación.
- **Protección de Datos:** Se implementa un esquema robusto de **seguridad** que incluye el control de acceso basado en la identidad, la obligatoriedad de conexiones vía SSL (Secure Sockets Layer) y avanzados sistemas de monitoreo.
- **Compatibilidad Sencilla:** Presenta una excelente **facilidad de integración**, dado que es compatible con las herramientas estándar de MySQL y puede integrarse sin esfuerzo con aplicaciones desarrolladas en Java, típicamente a través del *driver JDBC*.

Dentro del marco de este proyecto, **Google Cloud MySQL** 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

Payara es un servidor de aplicaciones de alto rendimiento que cumple con las especificaciones completas de Jakarta EE (previamente conocido como Java EE). Su robustez lo convierte en una plataforma idónea para el hosting de servicios SOAP, gracias a su soporte nativo para tecnologías clave como JAX-WS (Java API for XML Web Services) y su capacidad probada para gestionar aplicaciones empresariales de alta complejidad.

En el marco de esta implementación, el desarrollo del servicio SOAP se llevará a cabo utilizando JAX-WS sobre la plataforma Payara. Los componentes y sus funciones son los siguientes:

- **JAX-WS:** Será el framework principal empleado para la definición del servicio SOAP, encargándose de la generación automática del documento WSDL (Web Services Description Language).
- **Google Cloud MySQL:** Se encargará del almacenamiento persistente de los datos requeridos por el servicio. El acceso a esta base de datos se realizará mediante el conector JDBC, asegurando un rendimiento y una conectividad optimizados.

- Eficiencia de Despliegue: La arquitectura permite que los servicios SOAP sean empaquetados y desplegados de manera sencilla en el servidor Payara, garantizando la conectividad y el acceso seguro a la base de datos alojada en Google Cloud.

4. DESARROLLO DEL APLICATIVO

4.1 INSTALACIÓN DE HERRAMIENTAS

4.1.1 HERRAMIENTAS NECESARIAS

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

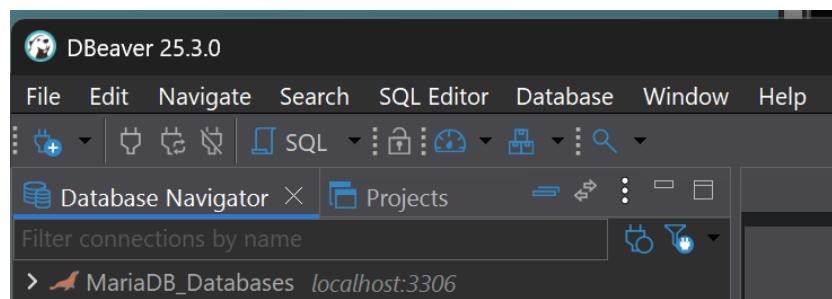


Figura 1 MySQL WorkBench

📁 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

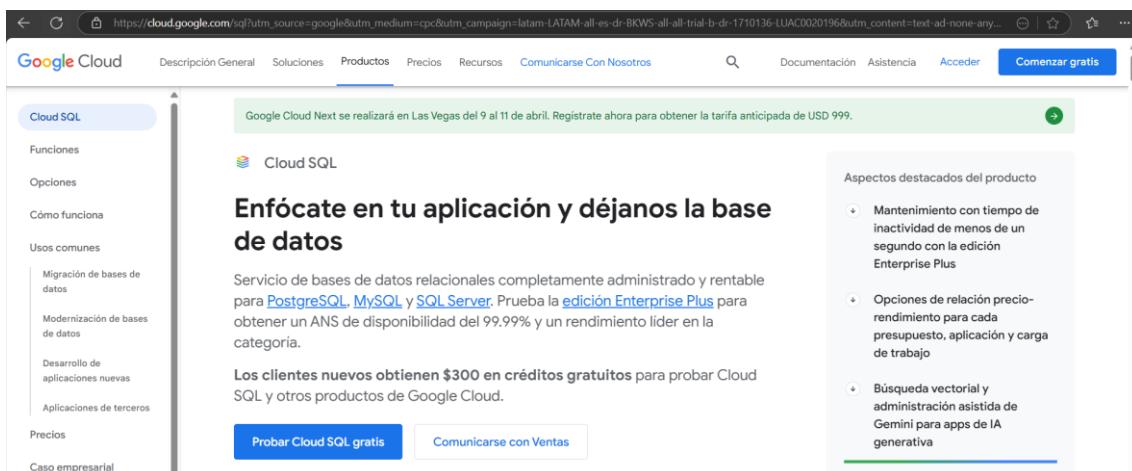


Figura 3 Pantalla de Cloud SQL

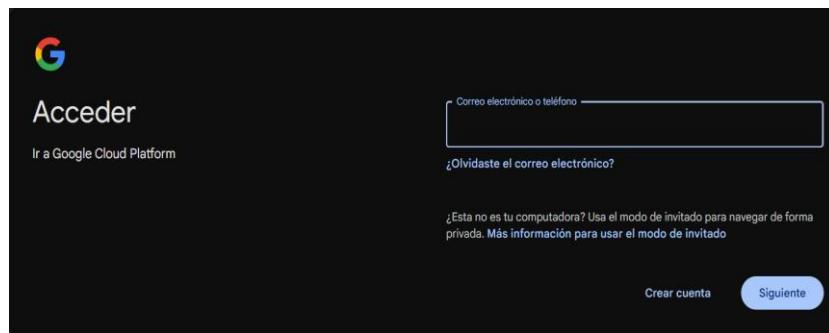


Figura 4 Ingreso Gmail

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.

[← Crea una instancia de MySQL](#)

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](#)

<input type="radio"/> Enterprise Plus	<ul style="list-style-type: none"> ANS de disponibilidad del 99.99% Tiempo de inactividad por mantenimiento planificado en menos de un segundo Escalación vertical de instancias con tiempo de inactividad casi nulo Máquinas con rendimiento optimizado Hasta 35 días para la ventana de recuperación de un momento determinado Capacidad de procesamiento hasta 3 veces mayor con caché de datos Recuperación ante desastres avanzada con cambio fácil Compatibilidad con el grupo de conexiones administrado
<input checked="" type="radio"/> Enterprise	<ul style="list-style-type: none"> ANS de disponibilidad del 99.95% Menos de 60 segundos de tiempo de inactividad por mantenimiento planificado Máquinas de uso general Hasta 7 días para la ventana de recuperación de un momento determinado

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)	2,000 de 2,000
IOPS	Lectura: 13,500 de 15,000 Escritura: 13,500 de 15,000
Capacidad de procesamiento del disco (MB/s)	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

Google Cloud My First Project Buscar (/) rec

Instancias [+ Crear instancia](#) [Migrar base de datos](#)

A partir del 1 de febrero de 2025, todas las instancias que ejecutan versiones de la comunidad con 1 cobrará la asistencia extendida a partir del 1 de mayo de 2025. Actualiza las instancias que ejecutan [información](#)

Filtro Escribir el nombre o valor de la propiedad

Estado	ID de instancia	Problemas	Edición de Cloud SQL	Tipo
<input checked="" type="checkbox"/>	maria-db-gr05		Enterprise	MySQL 8.4

Figura 6 Configuración de Cloud SQL

Creada la base de datos se comprueba la correcta creación

Nombre ↑	Intercalación	Grupo de caracteres	Tipo	⋮
banquito_db	utf8mb4_0900_ai_ci	utf8mb4	Usuario	⋮
comer_electro_db	utf8mb4_0900_ai_ci	utf8mb4	Usuario	⋮
eurekabank	utf8mb4_0900_ai_ci	utf8mb4	Usuario	⋮

Figura 7 Tablas de la base de datos

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

Todas las instancias > maria-db-gr05

maria-db-gr05

MySQL 8.4

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](#)

▼ Acceso universal (0.0.0.0/0) trash

Figura 8 Configuración de Network

eurekabank	
Tables	
Events	Asignado 48K
	CargoMantenimiento 32K
	Cliente 16K
	Contador 16K
	CostoMovimiento 32K
	Cuenta 80K
	Empleado 16K
	InteresMensual 32K
	LOGSESSION 32K
	Modulo 16K
	Moneda 16K
	Movimiento 64K
	Parametro 16K
	Permiso 32K
	Sucursal 16K
	TipoMovimiento 16K
	Usuario 32K

Figura 9 Creación Base de datos y carga

4.2 APLICACIÓN SERVIDOR

El primer paso, consiste en crear un nuevo proyecto. Dentro de la categoría Java Web, se selecciona Web Application. Para ello, se crea entonces un proyecto seleccionando la opción File > New Project.

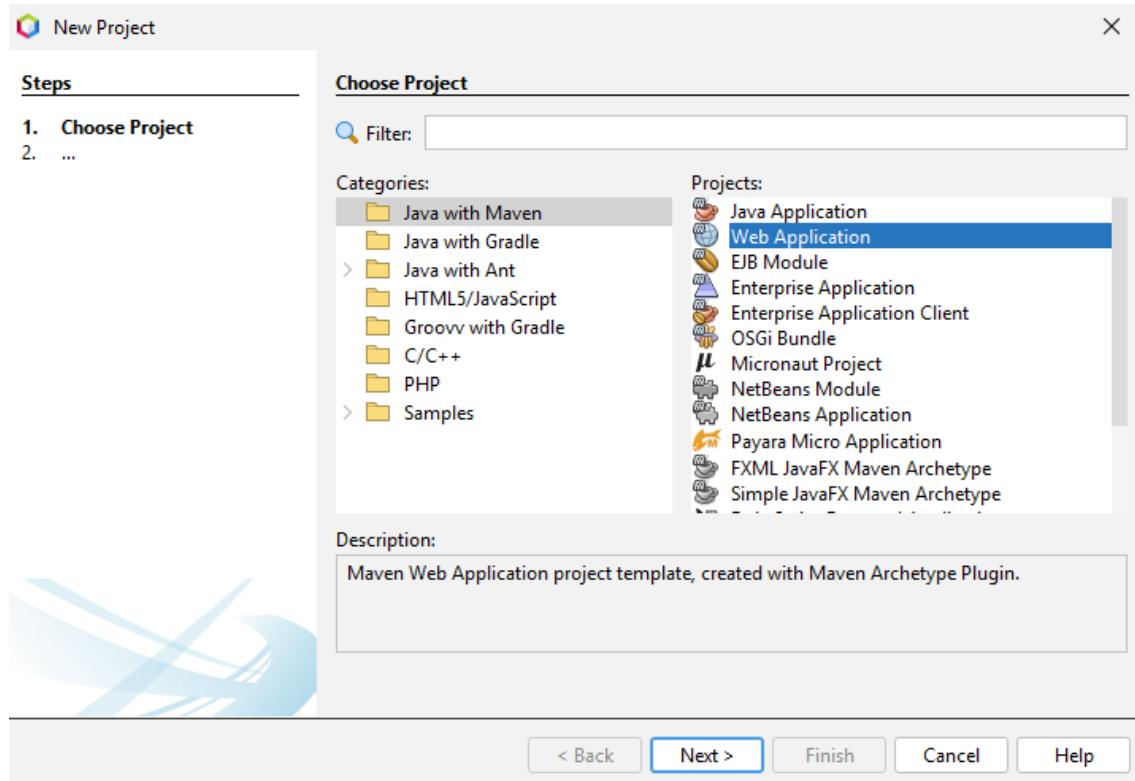


Figura 10 Nuevo proyecto

Posteriormente, seleccionar la opción Java Web > Web Application.

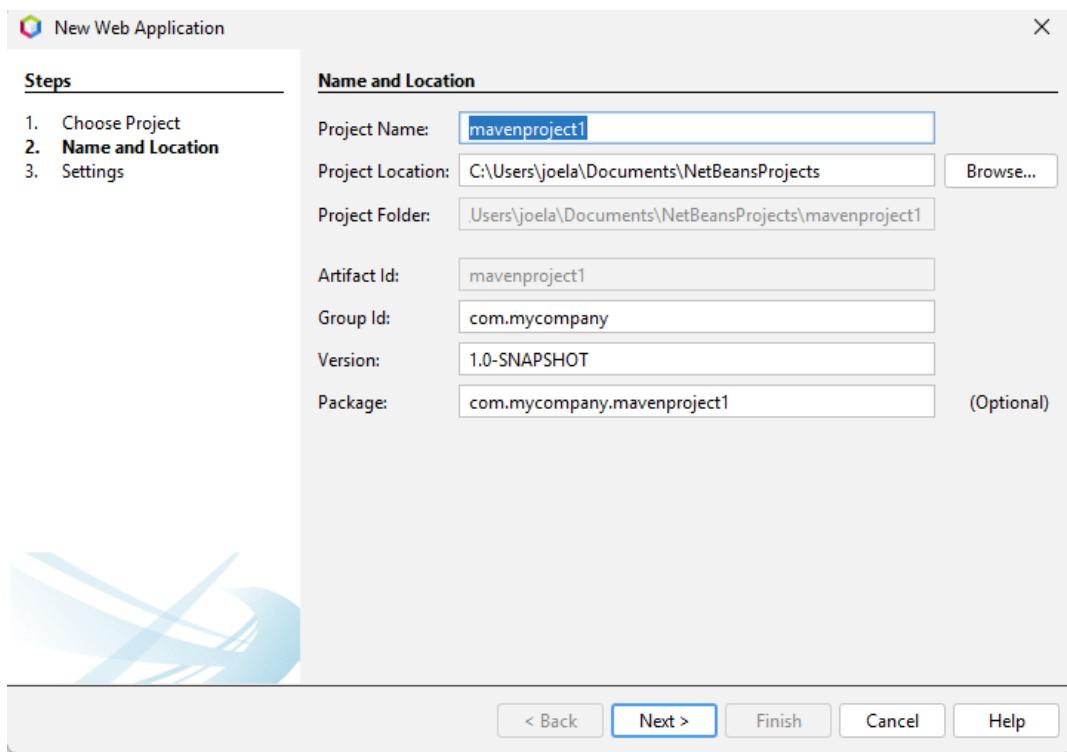


Figura 11 Aplicación tipo web

Se procede a configurar el nombre y la ubicación de la aplicación servidor.

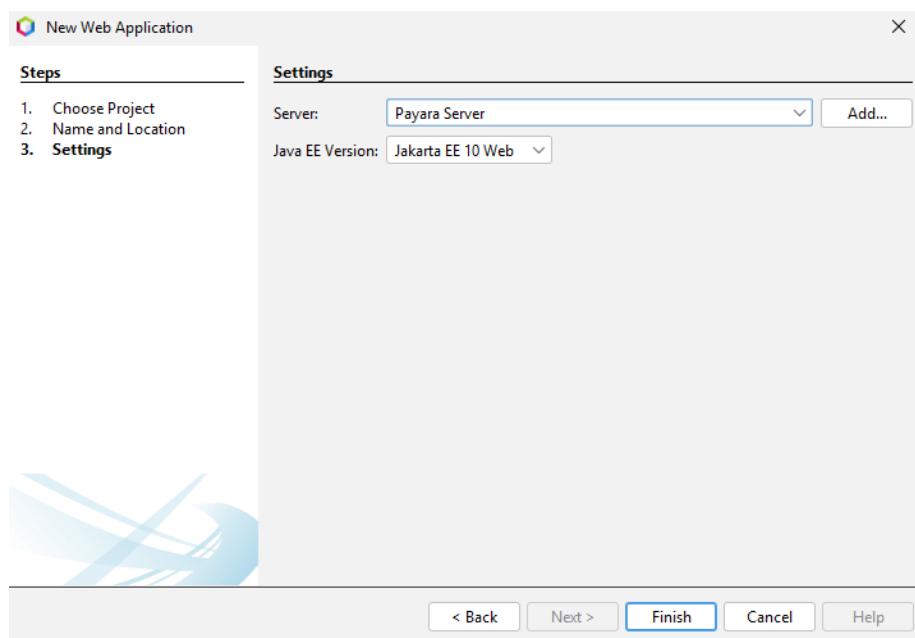


Figura 12 Configuración Frameworks

Se verifica la estructura del proyecto Servidor



Figura 13 Estructura de servidor

4.2.1 MODELO MVC

Se crea las carpetas necesarias para el entorno de trabajo.

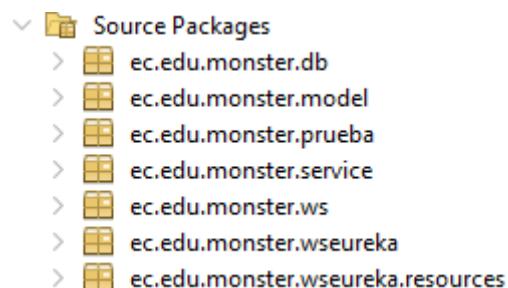


Figura 14 Creación de los paquetes

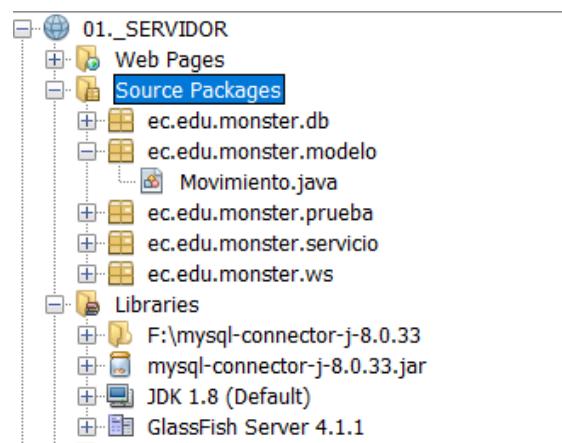


Figura 15 Estructura de los paquetes

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.



Figura 16 Creación de la clase AccesoDB

```
1  package ec.edu.monster.db;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.SQLException;
6
7  public class AccesoDB {
8
9      private AccesoDB() {
10
11
12      public static Connection getConnection() throws SQLException {
13          Connection cn = null;
14
15          try {
16              String driver = "org.mariadb.jdbc.Driver";
17              String url = "jdbc:mysql://34.135.112.222:3306/eurekabank";
18              String user = "grupo-05";
19              String pass = "BaRiYal23.";
20
21              Class.forName(driver);
22              cn = DriverManager.getConnection(url, user, pass);
23
24          } catch (ClassNotFoundException e) {
25              throw new SQLException("ERROR: no se encuentra el driver JDBC.");
26          } catch (SQLException e) {
27              throw new SQLException("ERROR al conectar: " + e.getMessage());
28          }
29
30          return cn;
31      }
32
33      public static Connection getConnectionModern() throws SQLException {
34          Connection cn = null;
35
36          try {
37
38              cn = DriverManager.getConnection(
39                  "jdbc:mysql://34.135.112.222:3306/eurekabank",
40                  "grupo-05",
41                  "BaRiYal23."
42              );
43
44      }
45  }
```

Figura 17 Codificación de la conexión de la base de datos

TABLA 1 Clase Acceso Base de Datos

```
package ec.edu.monster.db;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class AccesoDB {

    private AccesoDB() {
    }

    public static Connection getConnection() throws SQLException {
        Connection cn = null;

        try {
            String driver = "org.mariadb.jdbc.Driver";
            String url = "jdbc:mysql://34.135.112.222:3306/eurekabank";
            String user = "grupo-05";
            String pass = "BaRiYa123.";

```

```
Class.forName(driver);
cn = DriverManager.getConnection(url, user, pass);

} catch (ClassNotFoundException e) {
    throw new SQLException("ERROR: no se encuentra el driver JDBC.");
} catch (SQLException e) {
    throw new SQLException("ERROR al conectar: " + e.getMessage());
}

return cn;
}

public static Connection getConnectionModern() throws SQLException {
    Connection cn = null;

    try {

        cn = DriverManager.getConnection(
            "jdbc:mysql://34.135.112.222:3306/eurekabank",
            "grupo-05",
```

```
        "BaRiYa123."  
    );  
  
} catch (SQLException e) {  
    throw new SQLException("ERROR al conectar: " + e.getMessage());  
}  
  
return cn;  
}  
}
```

4.2.3 PRUEBA DE CONEXIÓN

Se crea una clase de pruebas para comprobar que la codificación se realizó con éxito.

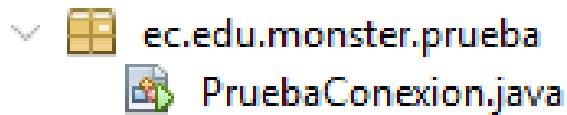


Figura 18 Creación de la clase de pruebas para la conexión

Luego se codifica la clase como si fuera una clase principal.

```
1 package ec.edu.monster.prueba;
2
3
4 import ec.edu.monster.db.AccesoDB;
5 import java.sql.Connection;
6
7 /**
8 *
9 * @author leona
10 */
11 public class PruebaConexion {
12
13     public static void main(String[] args) {
14         try (Connection cn = AccesoDB.getConnection()) {
15             System.out.println("✔ Conexión exitosa a la base de datos!");
16         } catch (Exception e) {
17             e.printStackTrace();
18         }
19     }
20 }
21
22 }
```

Figura 19 Codificación clase de prueba

TABLA 2 Clase de prueba de conexión

```
package ec.edu.monster.prueba;

import ec.edu.monster.db.AccesoDB;

import java.sql.Connection;

/**
 *
 * @author leona
 */

public class PruebaConexion {

    public static void main(String[] args) {
        try (Connection cn = AccesoDB.getConnection()) {
            System.out.println("☑ Conexión exitosa a la base de datos!");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
}
```

Para comprobar la funcionalidad de la conexión a la base de datos se procede a ejecutar el archivo PruebaConexionBD.java, en caso de estar todo correcto se espera el mensaje "ok".

```
run:  
Ok  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figura 20 Comprobación del Driver

Nota se debe agregar la librería de la conexión MySQL

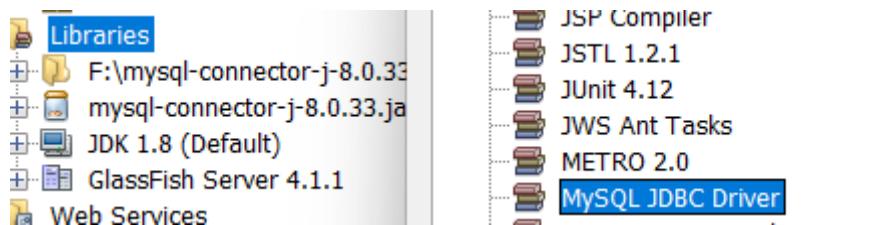


Figura 21 Bibliotecas del proyecto

Se importan todas las librerías necesarias que ofrece la librería

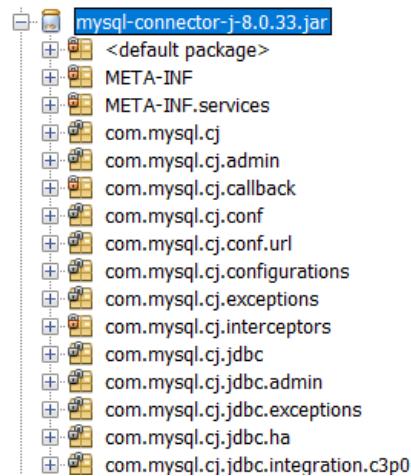


Figura 22 Librería del conector

```

    ...
    */
    package ec.edu.monster.prueba;

    import ec.edu.monster.db.AccesoDB;
    import java.sql.Connection;

```

Figura 23 Conector integrado en el proyecto

Con todo lo anterior se comprueba una conexión exitosa.

The screenshot shows an IDE interface with two tabs open: 'PruebaConexion.java' and 'pom.xml [WS_EurekaBank_SOAPJAVA_GR05]'. The code in 'PruebaConexion.java' is as follows:

```

1 package ec.edu.monster.prueba;
2
3 import ec.edu.monster.db.AccesoDB;
4 import java.sql.Connection;
5
6 /**
7  * @author leona
8  */
9 public class PruebaConexion {
10
11     public static void main(String[] args) {
12         try (Connection cn = AccesoDB.getConnection()) {
13             System.out.println("✓ Conexión exitosa a la base de datos!");
14         } catch (Exception e) {
15             e.printStackTrace();
16         }
17     }
18 }
19
20
21 }

```

The 'Output' tab displays the build log for 'WS_EurekaBank_SOAPJAVA_GR05'. It shows the following messages:

- Building WS_EurekaBank_SOAPJAVA_GR05 1.0-SNAPSHOT from pom.xml
- [war]
- The POM for org.glassfish.metro:webservices-rt:jari2.3 is invalid, transitive dependencies (if any) will not be resolved. 'dependencyManagement.dependencies.dependency.systemPath' for com.sun:tools:jar must specify an absolute path!
- resources:3.3.1:resources (default-resources) @ WS_EurekaBank_SOAPJAVA_GR05 ---
- Copying 1 resource from src\main\resources to target\classes
- compiler:3.8.1:compile (default-compile) @ WS_EurekaBank_SOAPJAVA_GR05 ---
- Nothing to compile - all classes are up to date
- exec:3.1.0:exec (default-cli) @ WS_EurekaBank_SOAPJAVA_GR05 ---
- ? Conexión exitosa a la base de datos!

Figura 24 Resultado de la prueba de conexión a la base

4.2.4 CODIFICACIÓN DE CLASES

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

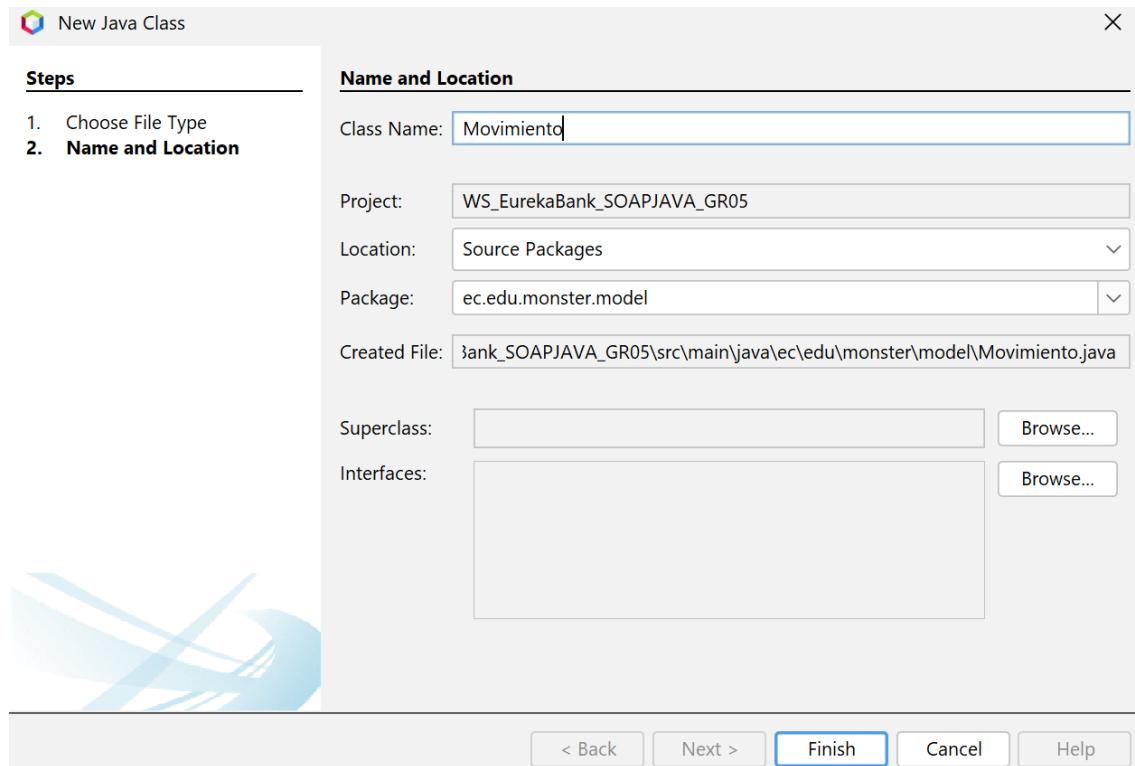


Figura 25 Creación de la clase movimiento

TABLA 3 Codificación clase Movimiento

```
package ec.edu.monster.model;

import jakarta.xml.bind.annotation.*;
import java.math.BigDecimal;
import java.util.Date;

/**
 *
 * @author leona
 */
@XmlRootElement(name = "MovimientoData")
@XmlAccessorType(XmlAccessType.FIELD)
```

```
public class MovimientoData {  
  
    @XmlElement(name = "CodigoCuenta")  
  
    private String codigoCuenta;  
  
    @XmlElement(name = "Numero")  
  
    private int numero;  
  
    @XmlElement(name = "Fecha")  
  
    private Date fecha;  
  
    @XmlElement(name = "Tipo")  
  
    private String tipo; // Ej: "Transferencia Entrada"
```

```
@XmlElement(name = "Importe")
private BigDecimal importe;

XmlElement(name = "Referencia")
private String referencia; // opcional

XmlElement(name = "SaldoActual")
private BigDecimal saldoActual;

XmlElement(name = "NombreCliente")
private String nombreCliente;
```

```
// Constructor vacío

public MovimientoData() {

}

// Getters y Setters

public String getCodigoCuenta() {

    return codigoCuenta;
}

public void setCodigoCuenta(String codigoCuenta) {

    this.codigoCuenta = codigoCuenta;
}
```

```
public int getNumero() {  
  
    return numero;  
  
}  
  
  
public void setNumero(int numero) {  
  
    this.numero = numero;  
  
}  
  
  
public Date getFecha() {  
  
    return fecha;  
  
}  
  
  
public void setFecha(Date fecha) {
```

```
this.fecha = fecha;  
}  
  
public String getTipo() {  
    return tipo;  
}  
  
public void setTipo(String tipo) {  
    this.tipo = tipo;  
}  
  
public BigDecimal getImporte() {  
    return importe;
```

```
}
```

```
public void setImporte(BigDecimal importe) {
```

```
    this.importe = importe;
```

```
}
```

```
public String getReferencia() {
```

```
    return referencia;
```

```
}
```

```
public void setReferencia(String referencia) {
```

```
    this.referencia = referencia;
```

```
}
```

```
public BigDecimal getSaldoActual() {  
  
    return saldoActual;  
  
}  
  
  
public void setSaldoActual(BigDecimal saldoActual) {  
  
    this.saldoActual = saldoActual;  
  
}  
  
  
public String getNombreCliente() {  
  
    return nombreCliente;  
  
}
```

```
public void setNombreCliente(String nombreCliente) {  
  
    this.nombreCliente = nombreCliente;  
  
}  
  
}
```

TABLA 4 Codificación Clase EurekaService

```
/*
* Copyright (c) 1996, 2020, Oracle and/or its affiliates. All rights reserved.
*
* ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
```

*
*
*
*
*
*
*
*
*
*
*/

```
package java.sql;

import java.util.Iterator;
import java.util.NoSuchElementException;
import java.util.concurrent.atomic.AtomicReferenceFieldUpdater;

/**
 * <P>An exception that provides information on a database access
 * error or other errors.
 *
 * <P>Each {@code SQLException} provides several kinds of information:
 * <UL>
 *   <LI>a string describing the error. This is used as the Java Exception
```

- * message, available via the method {@code getMessage}.
- * a "SQLstate" string, which follows either the XOPEN SQLstate conventions
- * or the SQL:2003 conventions.
- * The values of the SQLState string are described in the appropriate spec.
- * The {@code DatabaseMetaData} method {@code getSQLStateType}
- * can be used to discover whether the driver returns the XOPEN type or
- * the SQL:2003 type.
- * an integer error code that is specific to each vendor. Normally this will
- * be the actual error code returned by the underlying database.
- * a chain to a next Exception. This can be used to provide additional
- * error information.
- * the causal relationship, if any for this {@code SQLException}.
- *

```
*  
  
* @since 1.1  
  
*/  
  
public class SQLEception extends java.lang.Exception  
  
    implements Iterable<Throwable> {  
  
    /**  
     * Constructs a {@code SQLEception} object with a given  
     * {@code reason}, {@code SQLState} and  
     * {@code vendorCode}.  
     *  
     * The {@code cause} is not initialized, and may subsequently be  
     * initialized by a call to the  
    }
```

```
* {@link Throwable#initCause(java.lang.Throwable)} method.  
  
*  
  
* @param reason a description of the exception  
  
* @param SQLState an XOPEN or SQL:2003 code identifying the exception  
  
* @param vendorCode a database vendor-specific exception code  
  
*/  
  
public SQLException(String reason, String SQLState, int vendorCode) {  
  
    super(reason);  
  
    this.SQLState = SQLState;  
  
    this.vendorCode = vendorCode;  
  
    if (!(this instanceof SQLWarning)) {  
  
        if (DriverManager.getLogWriter() != null) {  
  
            DriverManager.println("SQLState(" + SQLState +
```

```
        ") vendor code(" + vendorCode + ")");
    printStackTrace(DriverManager.getLogWriter());
}
}

/**
 * Constructs a {@code SQLException} object with a given
 * {@code reason} and {@code SQLState}.
 *
 * The {@code cause} is not initialized, and may subsequently be
 * initialized by a call to the

```

```
* {@link Throwable#initCause(java.lang.Throwable)} method. The vendor code  
  
* is initialized to 0.  
  
*  
  
* @param reason a description of the exception  
  
* @param SQLState an XOPEN or SQL:2003 code identifying the exception  
  
*/  
  
public SQLException(String reason, String SQLState) {  
  
    super(reason);  
  
    this.SQLState = SQLState;  
  
    this.vendorCode = 0;  
  
    if (!(this instanceof SQLWarning)) {  
  
        if (DriverManager.getLogWriter() != null) {  
  
            printStackTrace(DriverManager.getLogWriter());
```

```
        DriverManager.println("SQLException: SQLState(" + SQLState + ")");
    }

}

}

/**  

 * Constructs a {@code SQLException} object with a given  

 * {@code reason}. The {@code SQLState} is initialized to  

 * {@code null} and the vendor code is initialized to 0.  

 *  

 * The {@code cause} is not initialized, and may subsequently be  

 * initialized by a call to the  

 * {@link Throwable#initCause(java.lang.Throwable)} method.
```

```
* @param reason a description of the exception

*/
public SQLException(String reason) {
    super(reason);
    this.SQLState = null;
    this.vendorCode = 0;
    if (!(this instanceof SQLWarning)) {
        if (DriverManager.getLogWriter() != null) {
            printStackTrace(DriverManager.getLogWriter());
        }
    }
}
```

```
/**  
  
 * Constructs a {@code SQLException} object.  
  
 * The {@code reason}, {@code SQLState} are initialized  
 * to {@code null} and the vendor code is initialized to 0.  
  
 *  
  
 * The {@code cause} is not initialized, and may subsequently be  
 * initialized by a call to the  
  
 * {@link Throwable#initCause(java.lang.Throwable)} method.  
  
 */  
  
public SQLException() {  
  
    super();
```

```
this.SQLState = null;

this.vendorCode = 0;

if (!(this instanceof SQLWarning)) {

    if (DriverManager.getLogWriter() != null) {

        printStackTrace(DriverManager.getLogWriter());

    }
}

}

/**

 * Constructs a {@code SQLException} object with a given

 * {@code cause}.

 * The {@code SQLState} is initialized
```

```
* to {@code null} and the vendor code is initialized to 0.  
  
* The {@code reason} is initialized to {@code null} if  
  
* {@code cause==null} or to {@code cause.toString()} if  
  
* {@code cause!=null}.  
  
*  
  
* @param cause the underlying reason for this {@code SQLException}  
  
* (which is saved for later retrieval by the {@code getCause()} method);  
  
* may be null indicating the cause is non-existent or unknown.  
  
* @since 1.6  
  
*/  
  
public SQLException(Throwable cause) {  
  
    super(cause);
```

```
if (!(this instanceof SQLWarning)) {  
  
    if (DriverManager.getLogWriter() != null) {  
  
        printStackTrace(DriverManager.getLogWriter());  
  
    }  
}  
  
}  
  
/**  
  
 * Constructs a {@code SQLException} object with a given  
  
 * {@code reason} and {@code cause}.  
  
 * The {@code SQLState} is initialized to {@code null}  
  
 * and the vendor code is initialized to 0.  
  
 *
```

```
* @param reason a description of the exception.  
  
* @param cause the underlying reason for this {@code SQLException}  
  
* (which is saved for later retrieval by the {@code getCause()} method);  
  
* may be null indicating the cause is non-existent or unknown.  
  
* @since 1.6  
  
*/  
  
public SQLException(String reason, Throwable cause) {  
  
    super(reason,cause);  
  
    if (!(this instanceof SQLWarning)) {  
  
        if (DriverManager.getLogWriter() != null) {  
  
            printStackTrace(DriverManager.getLogWriter());  
  
        }  
    }  
}
```

```
}

}

/**  
 * Constructs a {@code SQLException} object with a given  
 * {@code reason}, {@code SQLState} and {@code cause}.  
 *  
 * The vendor code is initialized to 0.  
 *  
 * @param reason a description of the exception.  
 * @param sqlState an XOPEN or SQL:2003 code identifying the exception  
 * @param cause the underlying reason for this {@code SQLException}  
 * (which is saved for later retrieval by the  
 * {@code getCause()} method); may be null indicating
```

```
*   the cause is non-existent or unknown.

* @since 1.6

*/

public SQLException(String reason, String sqlState, Throwable cause) {

    super(reason,cause);

    this.sqlState = sqlState;

    this.vendorCode = 0;

    if (!(this instanceof SQLWarning)) {

        if (DriverManager.getLogWriter() != null) {

            printStackTrace(DriverManager.getLogWriter());

            DriverManager.println("SQLState(" + SQLState + ")");

        }

    }

}
```

```
}

}

/**  
 * Constructs a {@code SQLException} object with a given  
 * {@code reason}, {@code SQLState}, {@code vendorCode}  
 * and {@code cause}.  
 *  
 * @param reason a description of the exception  
 * @param sqlState an XOPEN or SQL:2003 code identifying the exception  
 * @param vendorCode a database vendor-specific exception code  
 * @param cause the underlying reason for this {@code SQLException}  
 * (which is saved for later retrieval by the {@code getCause()} method);
```

```
* may be null indicating the cause is non-existent or unknown.

* @since 1.6

*/

public SQLException(String reason, String sqlState, int vendorCode, Throwable cause) {

super(reason,cause);

this.sqlState = sqlState;

this.vendorCode = vendorCode;

if (!(this instanceof SQLWarning)) {

if (DriverManager.getLogWriter() != null) {

DriverManager.println("SQLState(" + SQLState +

") vendor code(" + vendorCode + ")");

printStackTrace(DriverManager.getLogWriter());
}
```

```
}

}

}

/**

 * Retrieves the SQLState for this {@code SQLException} object.

 *

 * @return the SQLState value

 */

public String getSQLState() {

    return (SQLState);

}
```

```
/**  
  
 * Retrieves the vendor-specific exception code  
  
 * for this {@code SQLException} object.  
  
 *  
  
 * @return the vendor's error code  
  
 */  
  
public int getErrorCode()  
  
    return (vendorCode);  
  
}  
  
/**  
  
 * Retrieves the exception chained to this  
  
 * {@code SQLException} object by setNextException(SQLException ex).  
*/
```

```
* @return the next {@code SQLException} object in the chain;  
*      {@code null} if there are none  
* @see #setNextException  
*/  
  
public SQLException getNextException() {  
  
    return (next);  
}  
  
/**  
 * Adds an {@code SQLException} object to the end of the chain.  
 *  
 * @param ex the new exception that will be added to the end of
```

```
*      the {@code SQLException} chain

* @see #getNextException

*/
public void setNextException(SQLException ex) {

    SQLException current = this;

    for(;;) {

        SQLException next=current.next;

        if (next != null) {

            current = next;

            continue;

        }
    }
}
```

```
if (nextUpdater.compareAndSet(current,null,ex)) {  
  
    return;  
  
}  
  
current=current.next;  
  
}  
  
}  
  
/**  
  
 * Returns an iterator over the chained SQLExceptions. The iterator will  
  
 * be used to iterate over each SQLException and its underlying cause  
  
 * (if any).  
  
 *  
  
 * @return an iterator over the chained SQLExceptions and causes in the proper
```

```
* order

*
* @since 1.6

*/
public Iterator<Throwable> iterator() {

    return new Iterator<Throwable>() {

        SQLException firstException = SQLException.this;

        SQLException nextException = firstException.getNextException();

        Throwable cause = firstException.getCause();

        public boolean hasNext() {
```

```
if(firstException != null || nextException != null || cause != null)

    return true;

return false;

}

public Throwable next() {

    Throwable throwable = null;

    if(firstException != null){

        throwable = firstException;

        firstException = null;

    }

    else if(cause != null){

        throwable = cause;

    }

}
```

```
cause = cause.getCause();

}

else if(nextException != null){

    throwable = nextException;

    cause = nextException.getCause();

    nextException = nextException.getNextException();

}

else

    throw new NoSuchElementException();

return throwable;

}

public void remove() {
```

```
        throw new UnsupportedOperationException();

    }

};

}

/** * @serial */ private String SQLState;

/**
```

```
* @serial  
  
*/  
  
private int vendorCode;  
  
/**  
 * @serial  
 */  
  
private volatile SQLException next;  
  
  
private static final AtomicReferenceFieldUpdater<SQLException,SQLException> nextUpdater =  
    AtomicReferenceFieldUpdater.newUpdater(SQLException.class,SQLException.class,"next");  
  
private static final long serialVersionUID = 2135244094396331484L;  
  
}
```

4.2.5 CREACIÓN DEL SERVICIO WEB

El siguiente paso es crear el servicio web.

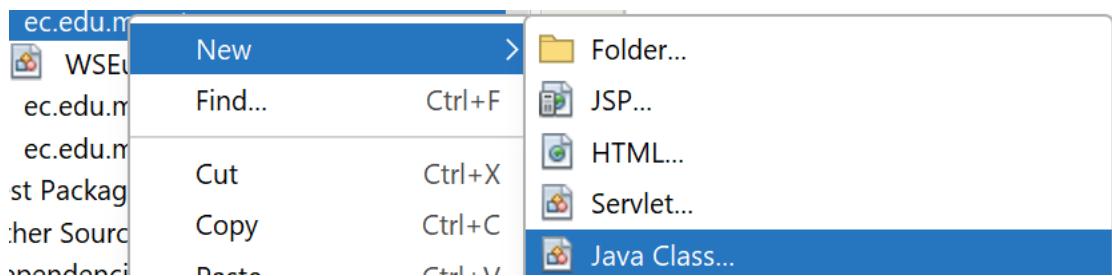


Figura 26 Creación Del servicio

Una vez dentro del menú de creación del servicio web se proporciona un nombre y se da clic en finalizar.

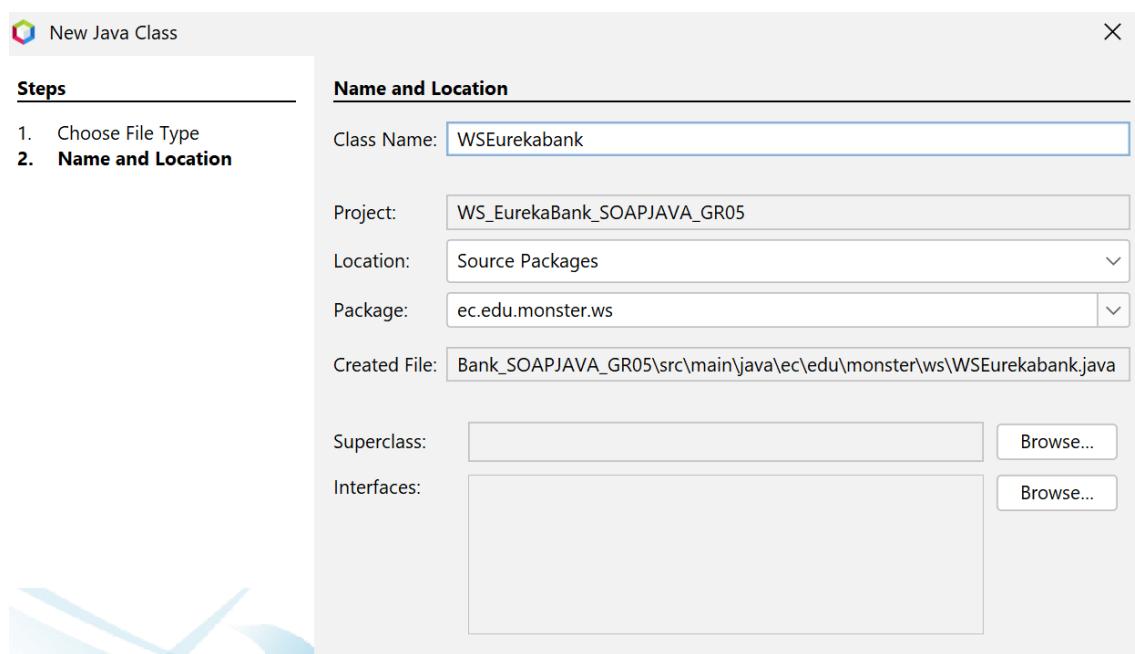


Figura 27 Configuración de nombre

```

17 @WebService(serviceName = "WSEurekaBank")
18 public class WSEurekaBank {
19
20     private static final String DEFAULT_COD_EMPLEADO = "0001";
21     private final EurekaService eurekaService = new EurekaService();
22     private final Login loginService = new Login();
23
24     /**
25      * 1. Traer movimientos de una cuenta
26      */
27     @WebMethod(operationName = "traerMovimientos")
28     @WebResult(name = "MovimientoData")
29     public List<MovimientoData> traerMovimientos(@WebParam(name = "cuenta") String cuenta) {
30         try {
31             return eurekaService.leerMovimientos(cuenta);
32         } catch (Exception e) {
33             System.err.println("Error en traerMovimientos: " + e.getMessage());
34             return new ArrayList<>();
35         }
36     }
37
38     /**
39      * 2. Registrar movimiento: Depósito, Retiro o Transferencia
40      */
41     @WebMethod(operationName = "regMovimiento")
42     @WebResult(name = "resultado")
43     public ResultadoOperacion regMovimiento(
44         @WebParam(name = "cuentaOrigen") String cuentaOrigen,
45         @WebParam(name = "cuentaDestino") String cuentaDestino,
46         @WebParam(name = "tipo") String tipo,
47         @WebParam(name = "importe") double importe) {
48

```

Figura 28 Codificación de los servicios web Movimientos

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

```

@WebMethod(operationName = "tipoMovimientosPermitidos")
@WebResult(name = "tipoMovimiento")

```

Figura 29 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.

```

41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
    @WebMethod(operationName = "regMovimiento")
    @WebResult(name = "resultado")
    public ResultadoOperacion regMovimiento(
        @WebParam(name = "cuentaOrigen") String cuentaOrigen,
        @WebParam(name = "cuentaDestino") String cuentaDestino,
        @WebParam(name = "tipo") String tipo,
        @WebParam(name = "importe") double importe) {

        try {
            // Validaciones básicas
            if (cuentaOrigen == null || cuentaOrigen.trim().isEmpty()) {
                return ResultadoOperacion.error("Cuenta origen es requerida");
            }
            if (importe <= 0) {
                return ResultadoOperacion.error("El importe debe ser mayor a 0");
            }
            if (tipo == null || tipo.trim().isEmpty()) {
                return ResultadoOperacion.error("Tipo de movimiento es requerido");
            }

            String tipoUpper = tipo.toUpperCase();

            switch (tipoUpper) {
                case "DEPOSITO":
                    if (cuentaDestino != null && !cuentaDestino.trim().isEmpty()) {
                        return ResultadoOperacion.error("Depósito no debe tener cuenta destino");
                    }
                    eurekaService.registrarDeposito(cuentaOrigen, importe, DEFAULT_COD_EMPLEADO);
                    return ResultadoOperacion.exito();
                case "RETIRO":
                    if (cuentaDestino != null && !cuentaDestino.trim().isEmpty()) {
                        return ResultadoOperacion.error("Retiro no debe tener cuenta destino");
                    }
            }
        }
    }
}

```

Figura 30 Codificación del servicio web Depósito

Al final se agregar todos los tags necesarios.

```

@WebMethod(operationName = "regMovimiento")
@WebResult(name = "resultado")

```

Figura 31 Adición de tags

Para la ejecución del servidor se debe limpiar y construir previamente a la ejecución.

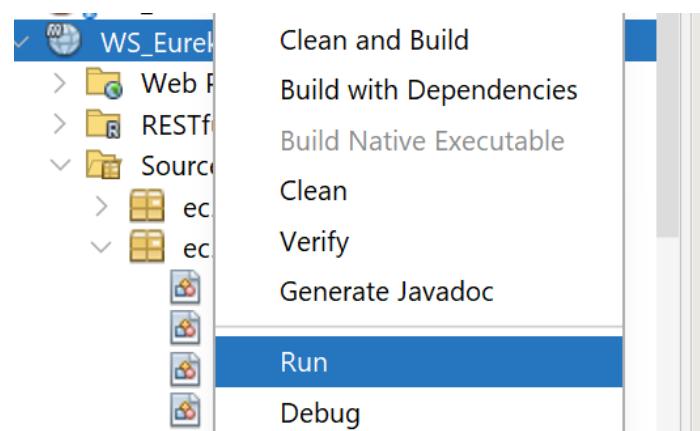
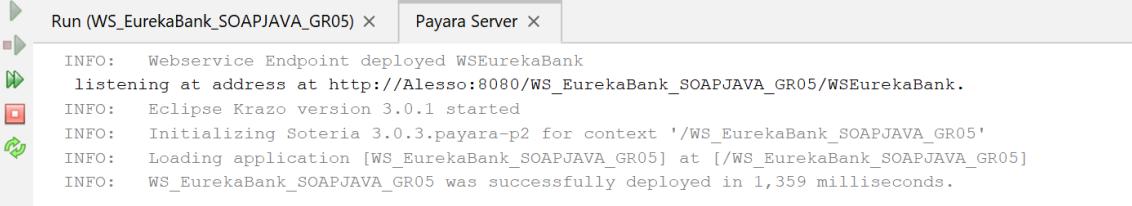


Figura 32 Despliegue de la aplicación

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



```
Run (WS_EurekaBank_SOAPJAVA_GR05) × Payara Server ×
INFO: Webservice Endpoint deployed WS_EurekaBank
      listening at address at http://Alesso:8080/WS_EurekaBank_SOAPJAVA_GR05/WS_EurekaBank.
INFO: Eclipse Krazo version 3.0.1 started
INFO: Initializing Soteria 3.0.3.payara-p2 for context '/WS_EurekaBank_SOAPJAVA_GR05'
INFO: Loading application [WS_EurekaBank_SOAPJAVA_GR05] at [/WS_EurekaBank_SOAPJAVA_GR05]
INFO: WS_EurekaBank_SOAPJAVA_GR05 was successfully deployed in 1,359 milliseconds.
```

Figura 33 Inicio del servidor

Se verifica si se levantó correctamente el servidor.



Web Services

Endpoint	Information
Service Name: {http://ws.monster.edu.ec/}WSEurekaBank Port Name: {http://ws.monster.edu.ec/}WSEurekaBankPort	Address: http://alesso:8080/WS_EurekaBank_SOAPJAVA_GR05/WS_EurekaBank WSDL: http://alesso:8080/WS_EurekaBank_SOAPJAVA_GR05/WS_EurekaBank?wsdl Implementation class: ec.edu.monster.ws.WSEurekaBank

Figura 34 Verificación de los servicios web creados

4.3 CREACIÓN DEL PROYECTO CONSOLA

Se crea una aplicación Java Ant.

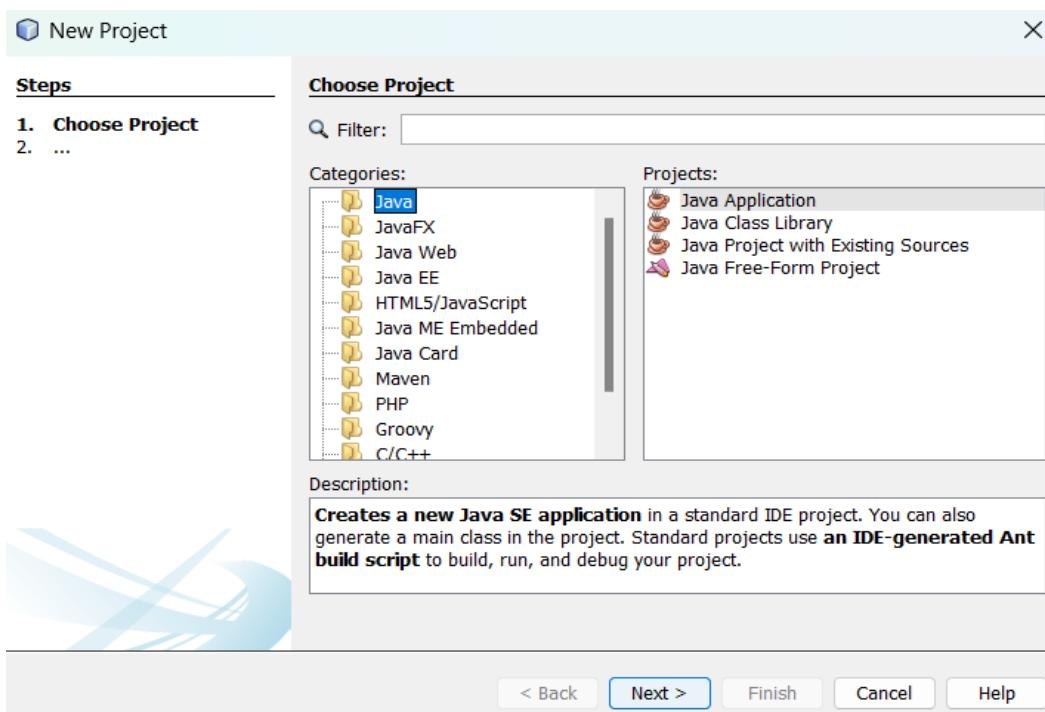


Figura 35 Creación Java Application Consola

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

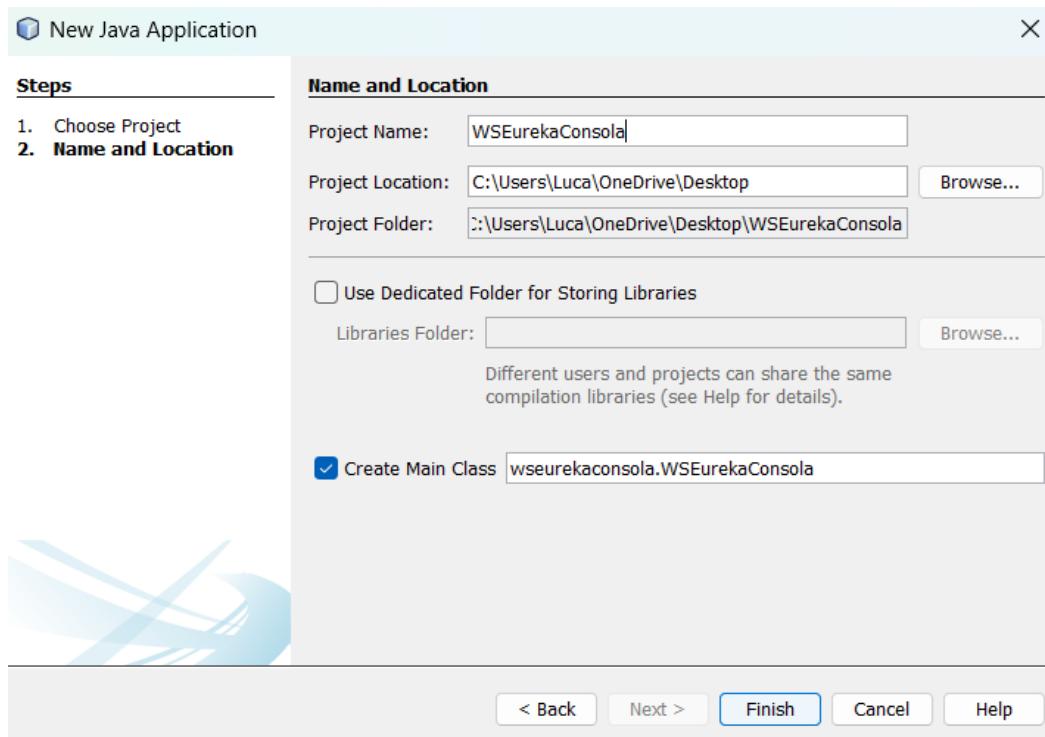


Figura 36 Creación proyecto estructura general

Se crea un cliente de servicio web.

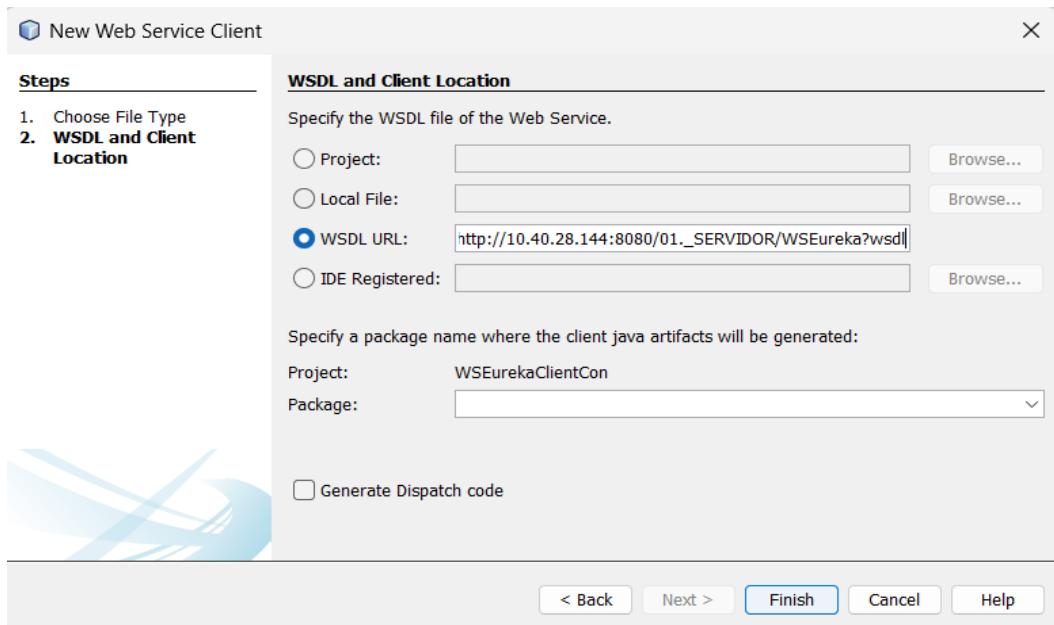


Figura 37 Creación Web Service Client

Si todo va bien se muestra la siguiente estructura.

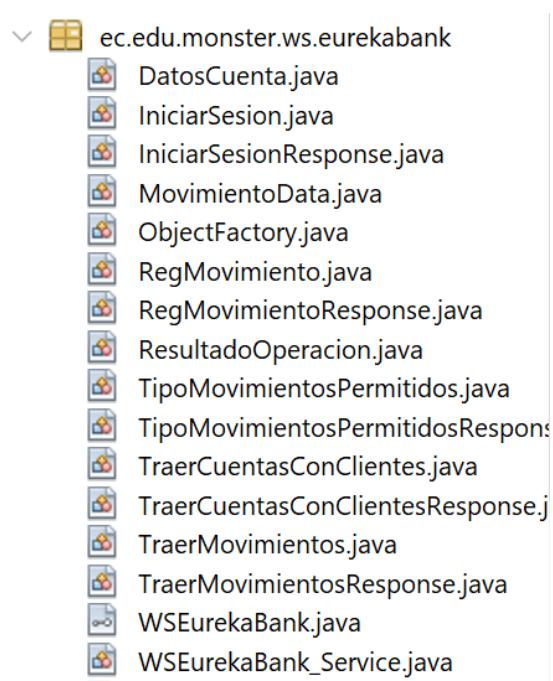


Figura 38 Servicios WSDL generados en el cliente

4.3.1 ESTRUCTURA MVC

CREACIÓN DE PAQUETE CONTROLADOR

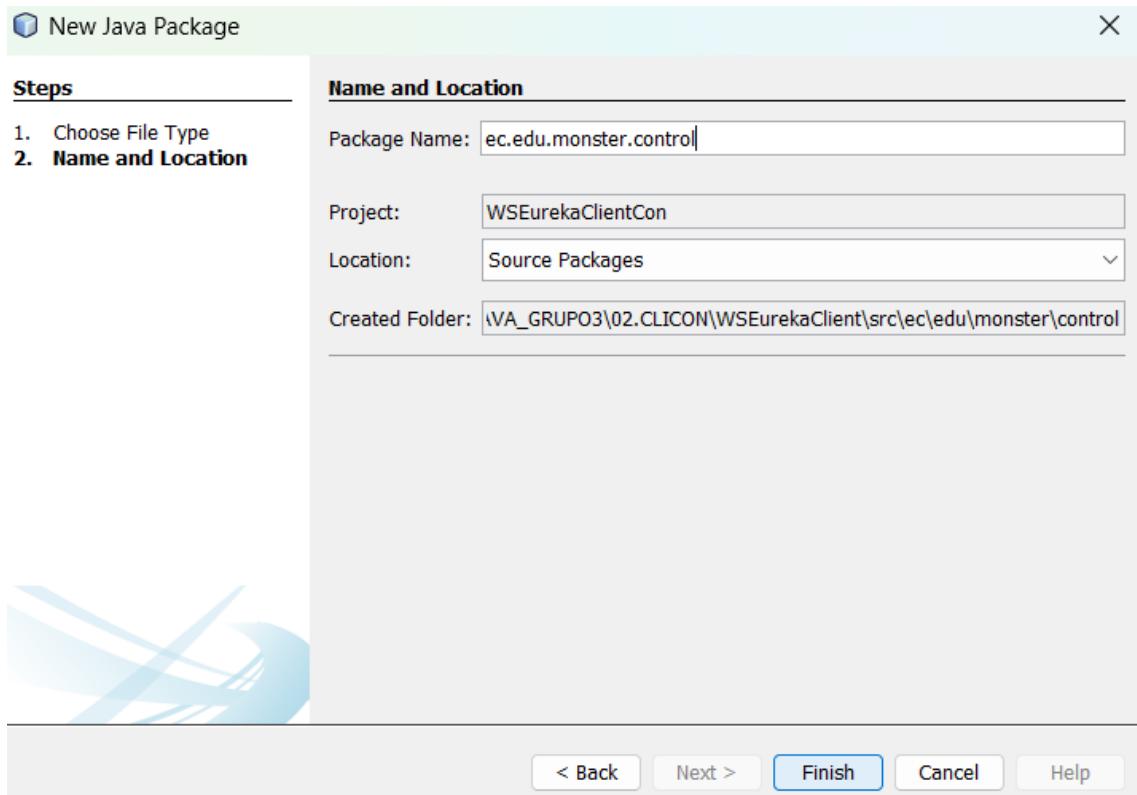


Figura 39 Creación paquete `ec.edu.monster.controlador`

CREACIÓN DE PACKAGE SERVICIO

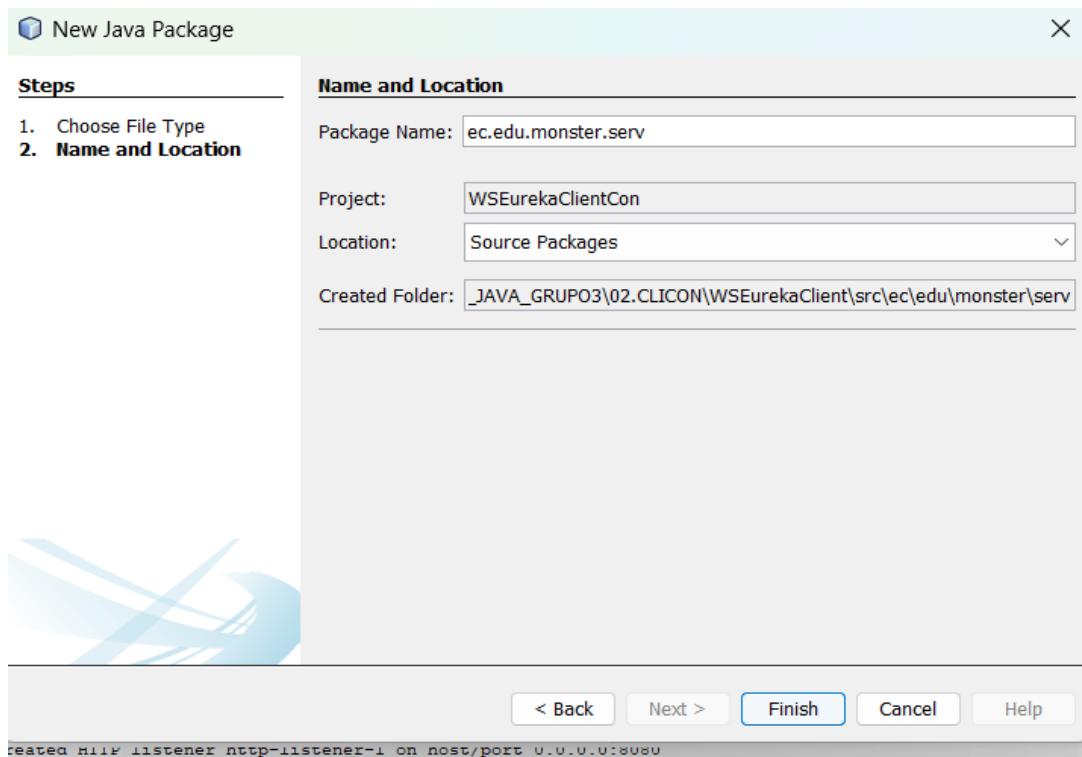


Figura 40 Creación paquete ec.edu.monster.servicio

CREACIÓN DE PACKAGE DB

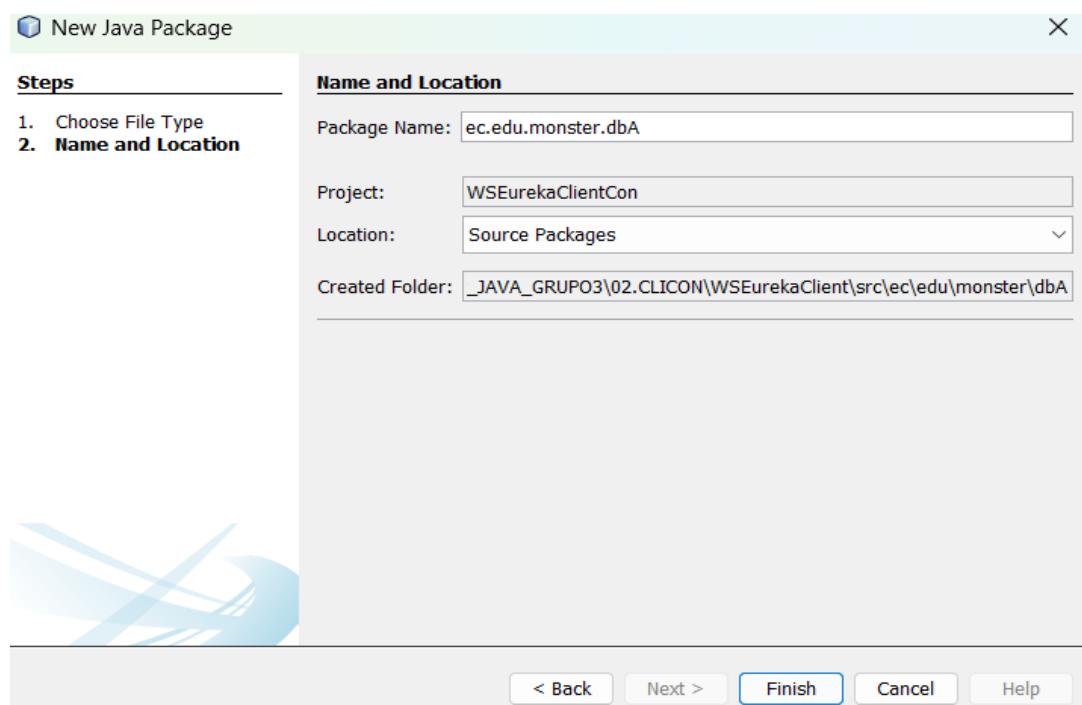


Figura 41 Creación paquete ec.edu.monster.db

CREACIÓN DE PACKAGE MODELO

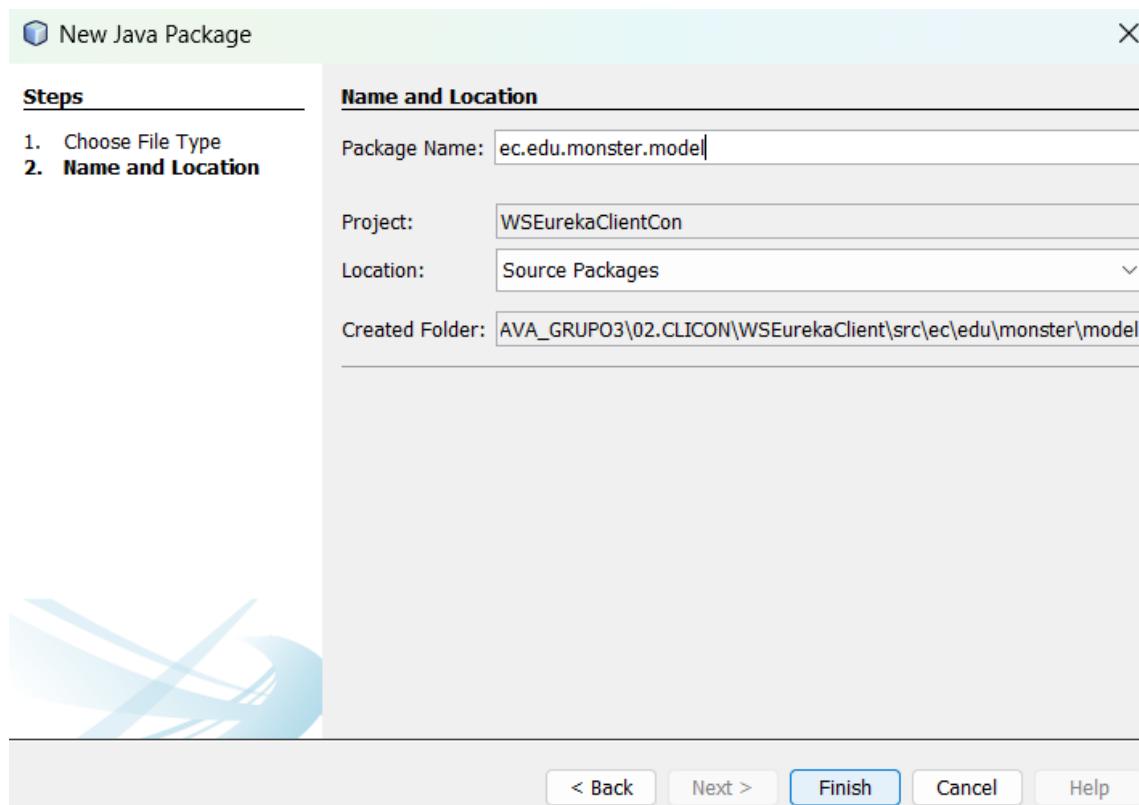


Figura 42 Creación paquete `ec.edu.monster.modelo`

PROYECTO ESTRUCTURADO

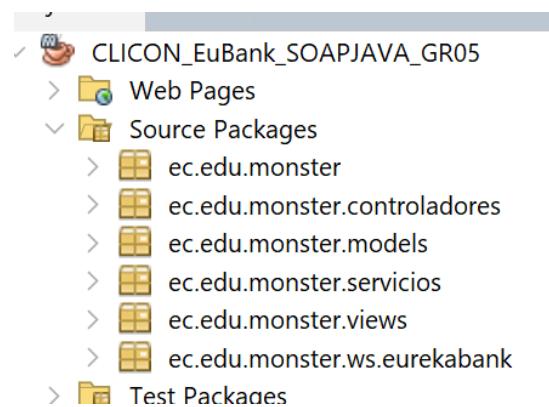


Figura 43 Estructura proyecto cliente

4.3.2 CODIFICACIÓN DE CLASES

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

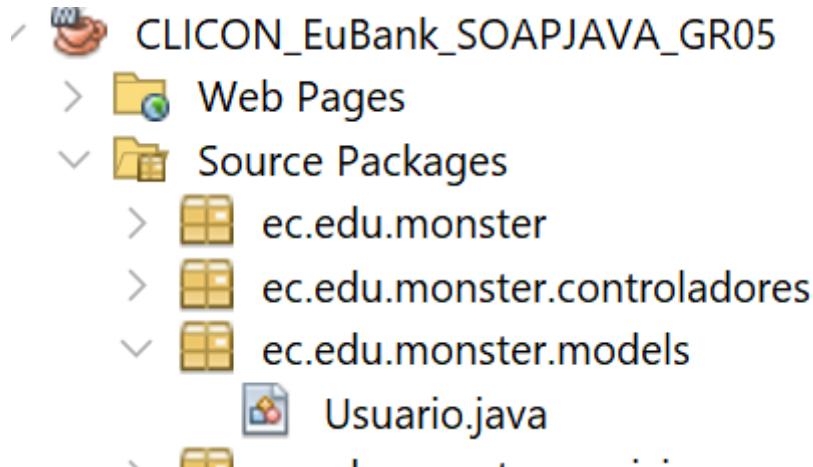


Figura 44 Creación clase Movimiento

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

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package ec.edu.monster.models;

/**
 *
 * @author joela
 */
public class Usuario {
    private String username;
    private String password;
    private boolean autenticado;

    public Usuario(String username, String password) {
        this.username = username;
        this.password = password;
        this.autenticado = false;
    }
}
```

Figura 45 Codificación clase Movimiento

Se procede a implementar los servicios.

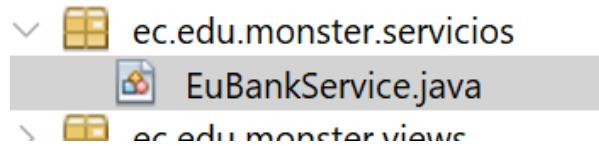


Figura 46 Creación clase EurekaService

```

16 import ec.edu.monster.ws.eurekabank.WSEurekaBank_Service;
17 import org.springframework.stereotype.Service;
18
19 import java.util.List;
20
21 @Service
22 public class EuBankService {
23     private final WSEurekaBank port;
24
25     public EuBankService() {
26         // Inicializa el cliente SOAP
27         WSEurekaBank_Service service = new WSEurekaBank_Service();
28         this.port = service.getWSEurekaBankPort();
29     }
30
31     // 1. Login (Asumiendo que el login está en el mismo WSDL)
32     public boolean iniciarSesion(String usuario, String contrasena) {
33         // El nombre "iniciarSesion" debe coincidir con el generado
34         return port.iniciarSesion(usuario, contrasena);
35     }

```

Figura 47 Codificación clase EurekaService

4.3.3 CREACIÓN DEL CONTROLADOR

Para crear el controlador, es necesario crear un paquete y crear una nueva clase llamada EurekaController donde se manejará todas las operaciones

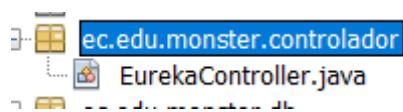


Figura 48 Creación clase EurekaController

Después se codifica la clase de la siguiente manera.

```

3 import ec.edu.monster.models.Usuario;
4 import ec.edu.monster.views.DashboardVista;
5 import ec.edu.monster.views.LoginVista;
6 import org.springframework.boot.CommandLineRunner;
7 import org.springframework.stereotype.Component;
8
9 @Component
10 public class ConsoleRunner implements CommandLineRunner {
11
12     private final LoginVista loginVista;
13     private final DashboardVista dashboardVista;
14
15     public ConsoleRunner(LoginVista loginVista, DashboardVista dashboardVista) {
16         this.loginVista = loginVista;
17         this.dashboardVista = dashboardVista;
18     }
19
20     @Override
21     public void run(String... args) throws Exception {
22         System.setOut(new java.io.PrintStream(System.out, true, java.nio.charset.StandardCharsets.UTF_8));
23
24         System.out.println("\n" + "=".repeat(20));
25         System.out.println("|| Cliente SOAP - EuBank (Consola) ||"); // <-- ¡CAMBIO!
26         System.out.println("||".repeat(20));
27
28         try {
29             // Esta lógica es IDÉNTICA a la de tu antiguo main!
30             Usuario_usuario = loginVista.mostrarLogin();

```

Figura 49 Codificación clase EurekaController

TABLA 5 Codificación controlador

```
package ec.edu.monster.controladores;

import ec.edu.monster.models.Usuario;
import ec.edu.monster.views.DashboardVista;
import ec.edu.monster.views.LoginVista;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

@Component
public class ConsoleRunner implements CommandLineRunner {

    private final LoginVista loginVista;
    private final DashboardVista dashboardVista;
```

```
public ConsoleRunner(LoginVista loginVista, DashboardVista dashboardVista) {  
  
    this.loginVista = loginVista;  
  
    this.dashboardVista = dashboardVista;  
  
}  
  
@Override  
  
public void run(String... args) throws Exception {  
  
    System.setOut(new java.io.PrintStream(System.out, true, java.nio.charset.StandardCharsets.UTF_8));  
  
    System.out.println("\n████████████████████████████████████████████████████████████████");  
    System.out.println(" || Cliente SOAP - EuBank (Consola) || "); // <-- ¡CAMBIO!  
    System.out.println("████████████████████████████████████████████████████████████████");  
  
    try {  
        // ¡Esta lógica es IDÉNTICA a la de tu antiguo main!  
    }  
}
```

```
Usuario usuario = loginVista.mostrarLogin();

if (usuario != null && usuario.isAutenticado()) {
    dashboardVista.mostrarMenuPrincipal() // <-- ¡CAMBIO!
} else {
    System.out.println("\nNo se pudo autenticar. Saliendo del sistema...");
}

} catch (Exception e) {
    mostrarError("Error crítico: " + e.getMessage());
}

System.out.println("\nPresione cualquier tecla para salir...");
System.in.read();
System.exit(0); // Cierra la aplicación
}
```

```
// (Puedes mover el 'mostrarError' aquí si quieres)

private void mostrarError(String mensaje) {

    System.out.print("\u001B[31m");

    System.out.println("\nX " + mensaje);

    System.out.print("\u001B[0m");

}

}
```

CREACIÓN DE LA VISTA

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

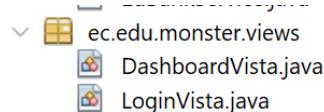


Figura 50 Creación clase WSEurekaClient

A continuación se codifica lo siguiente

```
Source History | 
```

```
1 package ec.edu.monster.views;
2
3 import ec.edu.monster.ws.eurekabank.*;
4 import ec.edu.monster.servicios.EuBankService;
5 import java.text.DecimalFormat;
6 import java.text.DecimalFormatSymbols;
7 import java.util.ArrayList;
8 import java.util.List;
9 import java.util.Locale;
10 import java.util.Scanner;
11 import org.springframework.stereotype.Component;
12
13 @Component
14 public class DashboardVista {
15
16     private final EuBankService euBankService;
17     private final Scanner scanner;
18     private final DecimalFormat decimalFormat;
19
20     public DashboardVista(EuBankService euBankService) {
21         this.euBankService = euBankService;
22         this.scanner = new Scanner(System.in, "UTF-8");
23         DecimalFormatSymbols symbols = new DecimalFormatSymbols(Locale.US);
24         this.decimalFormat = new DecimalFormat("#,##0.00", symbols);
25     }
26
27     public void mostrarMenuPrincipal() {
28         while (true) {
29             limpiarPantalla();
```

Figura 51 Codificación clase WSEurekaClient

TABLA 6 Codificación Clase WSEurekaClient

```
package ec.edu.monster.views;

import ec.edu.monster.ws.eurekabank.*;
import ec.edu.monster.servicios.EuBankService;
import java.text.DecimalFormat;
import java.text.DecimalFormatSymbols;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;
import java.util.Scanner;
import org.springframework.stereotype.Component;

@Component
public class DashboardVista {

    private final EuBankService euBankService;
    private final Scanner scanner;
    private final DecimalFormat decimalFormat;

    public DashboardVista(EuBankService euBankService) {
        this.euBankService = euBankService;
        this.scanner = new Scanner(System.in, "UTF-8");
        DecimalFormatSymbols symbols = new DecimalFormatSymbols(Locale.US);
        this.decimalFormat = new DecimalFormat("#,##0.00", symbols);
    }

    public void mostrarMenuPrincipal() {
        while (true) {
            limpiarPantalla();

            System.out.println("\n _____
=====");
        }
    }
}
```

```
System.out.println(" || Dashboard - EuBank (SOAP) || ");

System.out.println(" || ");
System.out.println("\n1. Ver Cuentas de Clientes");
System.out.println("2. Ver Movimientos (por cuenta o todos)");
System.out.println("3. Registrar Depósito");
System.out.println("4. Registrar Retiro");
System.out.println("5. Registrar Transferencia");
System.out.println("6. Salir");
System.out.print("\nSeleccione una opción: ");

int opcion;
try {
    opcion = Integer.parseInt(scanner.nextLine());
} catch (Exception e) {
    mostrarError("Opción no válida.");
    esperarTecla();
    continue;
}

switch (opcion) {
    case 1:
        mostrarCuentasClientes();
        break;
    case 2:
        mostrarMovimientos();
        break;
    case 3:
        registrarMovimiento("DEPOSITO");
        break;
    case 4:
        registrarMovimiento("RETIRO");
        break;
    case 5:
```

```

        registrarTransferencia();
        break;
    case 6:
        System.out.println("\n¡Gracias por usar EuBank! ¡Adiós!");
        return;
    default:
        mostrarError("Opción no válida.");
        esperarTecla();
        break;
    }
}
}

private void mostrarCuentasClientes() {
    limpiarPantalla();
    System.out.println("--- Cuentas de Clientes ---");
    try {
        System.out.print("Cargando cuentas...");
        List<DatosCuenta> cuentas = euBankService.traerCuentasConClientes();
        System.out.println(" ✓");
        if (cuentas.isEmpty()) {
            System.out.println("No se encontraron cuentas.");
        } else {
            String header = String.format("%-10s | %-30.30s | %-10s | %-10s | %12s",
                "CUENTA", "CLIENTE", "MONEDA", "ESTADO", "SALDO");
            System.out.println(header);
            System.out.println("-".repeat(header.length()));
            for (DatosCuenta c : cuentas) {
                System.out.printf("%-10s | %-30.30s | %-10s | %-10s | %12s%n",
                    c.getCodigo(),
                    c.getNombreCliente(),
                    c.getMoneda(),
                    c.getEstado(),
                    decimalFormat.format(c.getSaldo())));
            }
        }
    } catch (Exception e) {
        System.out.println("Ocurrió un error al cargar las cuentas: " + e.getMessage());
    }
}

```

```

        }

    }

} catch (Exception e) {
    mostrarError("Error al cargar cuentas: " + e.getMessage());
}

esperarTecla();

}

private void mostrarMovimientos() {
    limpiarPantalla();
    System.out.println("--- Movimientos por Cuenta ---");
    System.out.print("Ingrese el número de cuenta (ej. 00100001) o presione Enter para ver todos: ");
    String cuenta = scanner.nextLine().trim();

    if (cuenta.isEmpty()) {
        mostrarTodosLosMovimientos();
    } else {
        mostrarMovimientosDeUnaCuenta(cuenta);
    }
    esperarTecla();
}

private void mostrarTodosLosMovimientos() {
    System.out.print("Cargando todas las cuentas... ");
    try {
        List<DatosCuenta> cuentas = euBankService.traerCuentasConClientes();
        System.out.println(" ✓");
        List<MovimientoData> todosMovimientos = new ArrayList<>();

        System.out.print("Cargando todos los movimientos... ");
        for (DatosCuenta cuenta : cuentas) {
            try {
                todosMovimientos.addAll(euBankService.traerMovimientos(cuenta.getCodigo()));
            }
        }
    }
}

```

```

    } catch (Exception ex) {
    }
}

System.out.println(" ✓");

if (todosMovimientos.isEmpty()) {
    System.out.println("No se encontraron movimientos en ninguna cuenta.");
    return;
}

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());
});

String header = String.format("%-10s | %-4s | %-12s | %-25.25s | %12s | %12s",
    "CUENTA", "NRO", "FECHA", "TIPO", "IMPORTE", "SALDO");
System.out.println(header);
System.out.println("-".repeat(header.length()));

for (MovimientoData m : todosMovimientos) {
    String tipo = m.getTipo() != null ? m.getTipo() : "N/A";
    System.out.printf("%-10s | %-4d | %-12s | %-25.25s | %12s | %12s%n",
        m.getCodigoCuenta(),
        m.getNumero(),
        m.getFecha().toString().substring(0, 10),
        tipo,
        decimalFormat.format(m.getImporte()),
        decimalFormat.format(m.getSaldoActual()));
}

```

```

} catch (Exception e) {
    System.out.println(" X");
    mostrarError("Error al cargar todos los movimientos: " + e.getMessage());
}
}

private void mostrarMovimientosDeUnaCuenta(String cuenta) {
try {
    System.out.print("Cargando movimientos de la cuenta " + cuenta + "...");
    List<MovimientoData> movimientos = euBankService.traerMovimientos(cuenta);
    System.out.println(" ✓");

    if (movimientos.isEmpty()) {
        System.out.println("No se encontraron movimientos para la cuenta " + cuenta);
    } else {

        String header = String.format("%-4s | %-12s | %-25.25s | %12s | %12s",
            "NRO", "FECHA", "TIPO", "IMPORTE", "SALDO");
        System.out.println(header);
        System.out.println("-".repeat(header.length()));

        movimientos.sort((m1, m2) -> Integer.compare(m2.getNumero(), m1.getNumero()));

        for (MovimientoData m : movimientos) {
            String tipo = m.getTipo() != null ? m.getTipo() : "N/A";
            System.out.printf("%-4d | %-12s | %-25.25s | %12s | %12s%n",
                m.getNumero(),
                m.getFecha().toString().substring(0, 10),
                tipo,
                decimalFormat.format(m.getImporte()),
                decimalFormat.format(m.getSaldoActual()));
        }
    }
} catch (Exception e) {
}
}

```

```

        System.out.println(" X");
        mostrarError("Error al cargar movimientos: " + e.getMessage());
    }

}

private void registrarMovimiento(String tipo) {
    limpiarPantalla();
    System.out.println("--- Registrar " + tipo + " ---");

    System.out.print("Número de Cuenta: ");
    String cuenta = scanner.nextLine().trim();
    System.out.print("Importe: ");
    double importe;
    try {
        importe = Double.parseDouble(scanner.nextLine().trim().replace(',', '.'));
        if (importe <= 0) {
            mostrarError("El importe debe ser positivo.");
            esperarTecla();
            return;
        }
    } catch (Exception e) {
        mostrarError("Importe no válido.");
        esperarTecla();
        return;
    }

    try {
        System.out.print("Procesando " + tipo + "... ");
        ResultadoOperacion res = euBankService.regMovimiento(tipo, cuenta, "", importe);
        System.out.println(" ✓");

        if (res.getCodigo() == 1) {
            System.out.println("\n¡Operación exitosa!");
        } else {
    
```

```

        mostrarError(res.getMensaje());
    }

} catch (Exception e) {
    System.out.println(" X");
    mostrarError("Error al registrar movimiento: " + e.getMessage());
}

esperarTecla();

}

private void registrarTransferencia() {
    limpiarPantalla();
    System.out.println("--- Registrar Transferencia ---");

    System.out.print("Número de Cuenta Origen: ");
    String cuentaOrigen = scanner.nextLine().trim();
    System.out.print("Número de Cuenta Destino: ");
    String cuentaDestino = scanner.nextLine().trim();
    System.out.print("Importe: ");
    double importe;
    try {
        importe = Double.parseDouble(scanner.nextLine().trim().replace(',', '.'));
        if (importe <= 0) {
            mostrarError("El importe debe ser positivo.");
            esperarTecla();
            return;
        }
    } catch (Exception e) {
        mostrarError("Importe no válido.");
        esperarTecla();
        return;
    }

    if (cuentaOrigen.equals(cuentaDestino)) {
        mostrarError("Las cuentas de origen y destino no pueden ser la misma.");
    }
}

```

```

        esperarTecla();

        return;
    }

    try {
        System.out.print("Procesando Transferencia...");
        ResultadoOperacion res = euBankService.regMovimiento("TRANSFERENCIA",
cuentaOrigen, cuentaDestino, importe);
        System.out.println(" ✓");

        if (res.getCodigo() == 1) {
            System.out.println("\n¡Operación exitosa!");
        } else {
            mostrarError(res.getMensaje());
        }
    } catch (Exception e) {
        System.out.println(" X");
        mostrarError("Error al registrar movimiento: " + e.getMessage());
    }
    esperarTecla();
}

private void mostrarError(String mensaje) {
    System.out.print("\u001B[31m");
    System.out.println("\nX " + mensaje);
    System.out.print("\u001B[0m");
}

private void esperarTecla() {
    System.out.println("\nPresione Enter para continuar...");
    try {
        scanner.nextLine();
    } catch (Exception e) {
    }
}

```

```

}

private void limpiarPantalla() {
    try {
        if (System.getProperty("os.name").contains("Windows")) {
            new ProcessBuilder("cmd", "/c", "cls").inheritIO().start().waitFor();
        } else {
            System.out.print("\033[H\033[J");
            System.out.flush();
        }
    } catch (Exception e) {
        System.out.println("\n".repeat(50));
    }
}

```

4.3.4 EJECUCIÓN

Para la ejecución el servidor debe estar corriendo.

```

Payara Server × Run (CLICON_EuBank_SOAPJAVA_GR05) ×

Dashboard - EuBank (SOAP)

1. Ver Cuentas de Clientes
2. Ver Movimientos (por cuenta o todos)
3. Registrar Depósito
4. Registrar Retiro
5. Registrar Transferencia
6. Salir

Seleccione una opción: 1
↑--- Cuentas de Clientes ---
Cargando cuentas... ✓
CUENTA | CLIENTE | MONEDA | ESTADO | SALDO
-----
00100001 | ARANDA LUNA, ALAN ALBERTO | 01 | ACTIVO | 6,962.00
00100002 | ARANDA LUNA, ALAN ALBERTO | 02 | ACTIVO | 4,500.00
00200001 | FLORES CHAFLOQUE, ROSA LIZET | 01 | ACTIVO | 6,994.00
00200002 | CORONEL CASTILLO, ERIC GUSTAVO | 01 | ACTIVO | 6,832.00
00200003 | CHAVEZ CANALES, EDGAR RAFAEL | 02 | ACTIVO | 6,144.00

Presione Enter para continuar...

```

Figura 52 Ejecución clase WSEurekaClient

```

Output X
Payara Server X Run (CLICON_EuBank_SOAPJAVA_GR05) X
6. Salir

Seleccione una opción: 2
▲--- Movimientos por Cuenta ---
Ingresé el número de cuenta (ej. 00100001) o presione Enter para ver todos:
Cargando todas las cuentas... ✓
Cargando todos los movimientos...
✓

CUENTA | NRO | FECHA | TIPO | IMPORTE | SALDO
-----
00200003 | 30 | 2025-11-12 | Deposito Entrada | 30.00 | 6,144.00
00200003 | 29 | 2025-11-12 | Transferencia Salida | 50.00 | 6,114.00
00200003 | 28 | 2025-11-12 | Retiro Salida | 50.00 | 6,164.00
00200003 | 27 | 2025-11-12 | Deposito Entrada | 22.00 | 6,214.00
00200003 | 26 | 2025-11-12 | Deposito Entrada | 11.00 | 6,192.00
00200003 | 25 | 2025-11-12 | Deposito Entrada | 1.00 | 6,181.00
00200001 | 17 | 2025-11-12 | Transferencia Salida | 60.00 | 6,994.00
00200001 | 16 | 2025-11-12 | Transferencia Entrada | 54.00 | 7,054.00
00200002 | 11 | 2025-11-12 | Retiro Salida | 20.00 | 6,832.00
00100001 | 10 | 2025-11-12 | Transferencia Entrada | 60.00 | 6,962.00
00200002 | 10 | 2025-11-12 | Transferencia Entrada | 50.00 | 6,852.00
00100001 | 9 | 2025-11-12 | Deposito Entrada | 56.00 | 6,902.00
00100001 | 8 | 2025-11-12 | Transferencia Salida | 54.00 | 6,846.00
00200003 | 24 | 2025-11-11 | Deposito Entrada | 19.00 | 6,180.00
00200003 | 23 | 2025-11-11 | Deposito Entrada | 2.00 | 6,161.00
00200003 | 22 | 2025-11-11 | Deposito Entrada | 1.00 | 6,159.00

```

Figura 53 Ejecución clase WSEurekaClient

4.4 CREACIÓN DEL PROYECTO CLIENTE ESCRITORIO

Se crea con la misma lógica que la de consola

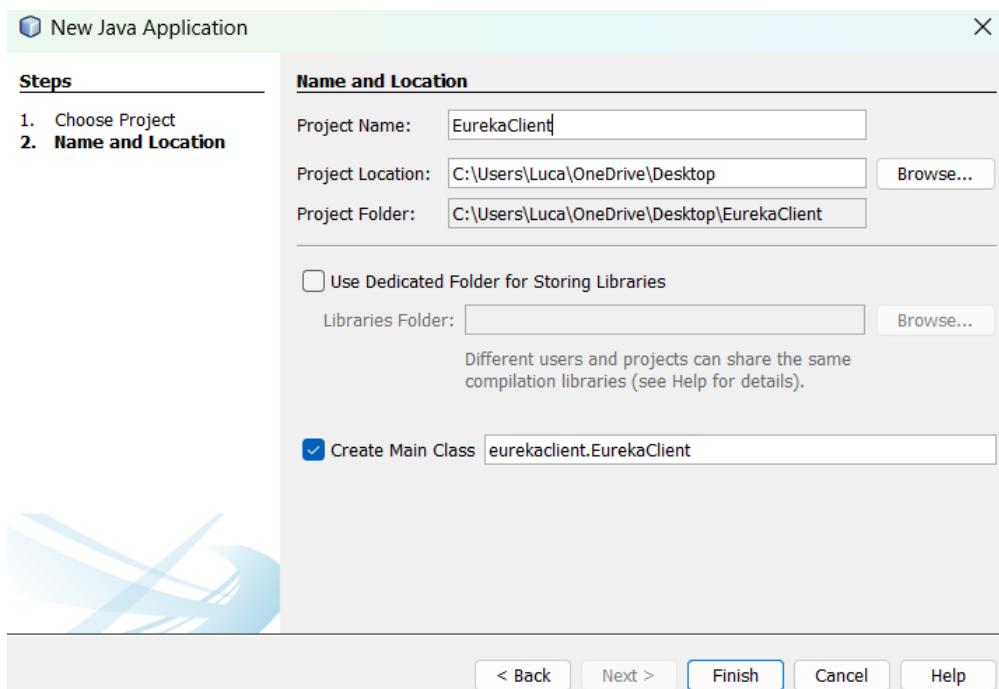


Figura 54 Creación del proyecto cliente

Se crea un cliente de servicio web.

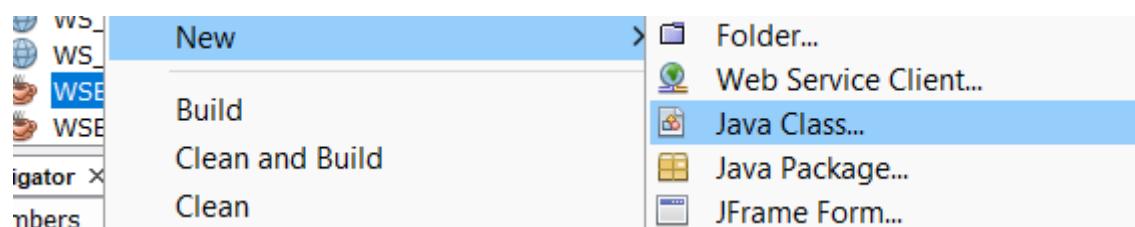


Figura 55 Creación del cliente del servicio web

Tomar en cuenta que el proyecto servidor debe estar desplegado.

Si todo va bien se muestra la siguiente estructura.

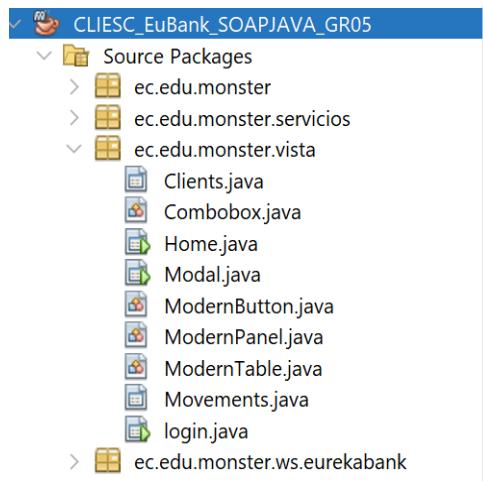


Figura 56 Estructura inicial del proyecto

4.4.1 CREACIÓN DE LA CAPA DE SERVICIO

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

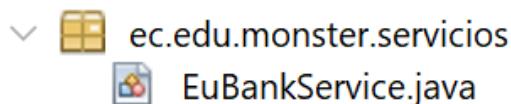


Figura 57 Creación de la clase EurekaService

Una vez creada la clase se procede a agregar las operaciones del servicio web, para lo cual se selecciona la opción Insert Code y luego Call Web Services Operation.

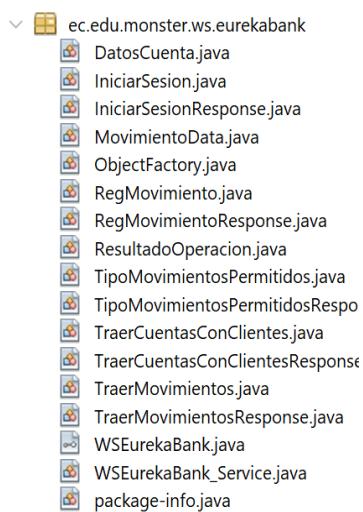


Figura 58 Llamada a los métodos del servicio web

TABLA 7 Clase llamada a servicios web

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package ec.edu.monster.servicio;

import ec.edu.monster.db.AccesoDB;
import ec.edu.monster.ws.WSEureka_Service;
import ec.edu.monster.ws.Movimiento;
import javax.xml.namespace.QName;
import javax.xml.transform.Source;
import javax.xml.ws.Dispatch;
import javax.xml.transform.stream.StreamSource;
import javax.xml.ws.Service;
import java.io.StringReader;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
```

```
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author leito
 */
public class EurekaService {

    public List<Movimiento> traerMovimientos(String cuenta) {
        WSEureka_Service service = new WSEureka_Service();
        QName portQName = new QName("http://ws.monster.edu.ec/", "WSEurekaPort");
        String req = "<traerMovimientos xmlns=\"http://ws.monster.edu.ec/\"><cuenta>" + cuenta + "</cuenta></traerMovimientos>";
        try {
            // Call Web Service Operation
            Dispatch<Source> sourceDispatch = service.createDispatch(portQName, Source.class, Service.Mode.PAYLOAD);
            Source result = sourceDispatch.invoke(new StreamSource(new StringReader(req)));
            // Process the result to extract List<Movimiento>
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
// Dummy return for example purposes
return new ArrayList<>(); // Replace with actual parsing logic
} catch (Exception ex) {
    ex.printStackTrace();
    return new ArrayList<>();
}
}

public List<ec.edu.monster.modelo.Movimiento> leerMovimientos(String cuenta){
    Connection cn = null;
    List<ec.edu.monster.modelo.Movimiento> lista = new ArrayList<ec.edu.monster.modelo.Movimiento>();
    String sql = "SELECT \n"
        + " m.chr_cuencodigo cuenta, \n"
        + " m.int_movingnumero nromov, \n"
        + " m.dtt_movifecha fecha, \n"
        + " t.vch_tipodescripcion tipo, \n"
        + " t.vch_tipoaccion accion, \n"
        + " m.dec_moviimporte importe \n"
        + "FROM tipomovimiento t INNER JOIN movimiento m \n"
        + "ON t.chr_tipocodigo = m.chr_tipocodigo \n"
        + "WHERE m.chr_cuencodigo = ?";
```

```
try{
    cn = AccesoDB.getConnection();
    PreparedStatement pstm = cn.prepareStatement(sql);
    pstm.setString(1, cuenta);
    ResultSet rs = pstm.executeQuery();

    while(rs.next()){
        ec.edu.monster.modelo.Movimiento rec = new ec.edu.monster.modelo.Movimiento();
        rec.setCuenta(rs.getString("cuenta"));
        rec.setNromov(rs.getInt("nromov"));
        rec.setFecha(rs.getDate("fecha"));
        rec.setTipo(rs.getString("tipo"));
        rec.setAccion(rs.getString("accion"));
        rec.setImporte(rs.getDouble("importe"));

        lista.add(rec);
    }
    rs.close();
}
```

```
        }catch(SQLException e){
            throw new RuntimeException(e.getMessage());
        }finally{
            try{
                cn.close();
            } catch(Exception e){
            }
        }
        return lista;
    }

    public void registrarDeposito(String cuenta, double importe, String codEmp){
        Connection cn = null;
        try{
            //obtener la conexion
            cn= AccesoDB.getConnection();
            //habilitar la transaccion
            cn.setAutoCommit(false);
            //paso 1: leer datos de la cuenta
            String sql = "select dec_cuensaldo, int_cuencontmov "
                        + "from cuenta "
```

```
+ "where chr_cuencodigo = ? and vch_cuenestado = 'ACTIVO'"  
+ "for update ";  
  
PreparedStatement pstm = cn.prepareStatement(sql);  
pstm.setString(1, cuenta);  
  
ResultSet rs =pstm.executeQuery();  
  
if(!rs.next()){  
    throw new SQLException("ERROR, cuenta no existe, o no esta activa");  
}  
  
double saldo = rs.getDouble("dec_cuensaldo");  
int cont = rs.getInt("int_cuencontmov");  
  
rs.close();  
pstm.close();  
  
//paso 2: actualizar la cuenta  
saldo += importe;  
cont++;  
  
sql = "update cuenta "  
+ "set dec_cuensaldo = ?, "  
+ "int_cuencontmov = ? "  
+ "where chr_cuencodigo = ? and vch_cuenestado = 'ACTIVO"';  
  
pstm = cn.prepareStatement(sql);  
pstm.setDouble(1, saldo);
```

```
pstm.setInt(2, cont);
pstm.setString(3, cuenta);
pstm.executeUpdate();
pstm.close();
//paso 3: Registrar movimientos
sql = "insert into movimiento(chr_cuencodigo,"
    + "int_movinumero,dtt_movifecha,chr_emplcodigo,chr_tipocodigo,"
    + "dec_moviimporte) values(?,?,SYSDATE(),?,'003',?)";
pstm = cn.prepareStatement(sql);
pstm.setString(1, cuenta);
pstm.setInt(2, cont);
pstm.setString(3, codEmp);
pstm.setDouble(4, importe);
pstm.executeUpdate();
//Confirmar transaccion
cn.commit();
}catch(SQLException e){
try {
    cn.rollback();
} catch (Exception el) {
}
}
```

```
        throw new RuntimeException(e.getMessage());
    }catch(Exception e){
        try {
            cn.rollback();
        } catch (Exception el) {
        }
        throw new RuntimeException("ERROR, en el proceso registrar deposito, intentelo mas tarde.");
    }finally{
        try {
            cn.close();
        } catch (Exception e) {
        }
    }
}

public static int regDepósito(java.lang.String cuenta, double importe) {
    ec.edu.monster.ws.WSEureka_Service service = new ec.edu.monster.ws.WSEureka_Service();
    ec.edu.monster.ws.WSEureka port = service.getWSEurekaPort();
    return port.regDepósito(cuenta, importe);
}
```

4.4.2 CREACIÓN DE LA CAPA DE PRUEBAS

Para la creación de la capa de pruebas es necesario crear un nuevo paquete.

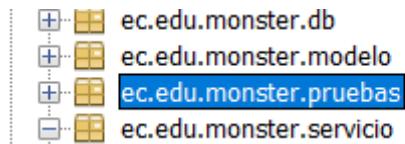


Figura 59 Generación de la capa de pruebas

Posteriormente se crean las clases de prueba.

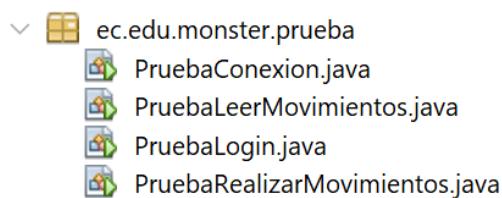


Figura 60 Clases de pruebas

TABLA 8 Clase llamada a servicios web

```
package ec.edu.monster.prueba;

import ec.edu.monster.db.AccesoDB;

import java.sql.Connection;

/**
 *
 * @author leona
 */
public class PruebaConexion {
```

```
public static void main(String[] args) {  
  
    try (Connection cn = AccesoDB.getConnection()) {  
  
        System.out.println("☑ Conexión exitosa a la base de datos!");  
  
    } catch (Exception e) {  
  
        e.printStackTrace();  
  
    }  
  
}
```

4.4.3 CREACIÓN DEL CONTROLADOR

Para crear el controlador, es necesario crear un paquete y crear una nueva clase.

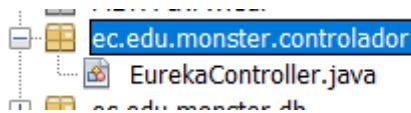


Figura 61 Creación del controlador

Después se codifica la clase de la siguiente manera.

TABLA 9 Codificación controlador

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package ec.edu.monster.modelo;

import java.util.Date;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name = "movimiento")
public class Movimiento {

    private String cuenta;
    private int nromov;
    private Date fecha;
    private String tipo;
    private String accion;
```

```
private double importe;

public Movimiento() {
}

public Movimiento(String cuenta, int nromov, Date fecha, String tipo, String accion, double importe) {
    this.cuenta = cuenta;
    this.nromov = nromov;
    this.fecha = fecha;
    this.tipo = tipo;
    this.accion = accion;
    this.importe = importe;
}

public String getCuenta() {
    return cuenta;
}

public void setCuenta(String cuenta) {
    this.cuenta = cuenta;
}
```

```
public int getNromov() {
    return nromov;
}

public void setNromov(int nromov) {
    this.nromov = nromov;
}

public Date getFecha() {
    return fecha;
}

public void setFecha(Date fecha) {
    this.fecha = fecha;
}

public String getTipo() {
    return tipo;
}
```

```
public void setTipo(String tipo) {
    this.tipo = tipo;
}

public String getAccion() {
    return accion;
}

public void setAccion(String accion) {
    this.accion = accion;
}

public double getImporte() {
    return importe;
}

public void setImporte(double importe) {
    this.importe = importe;
}
```

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

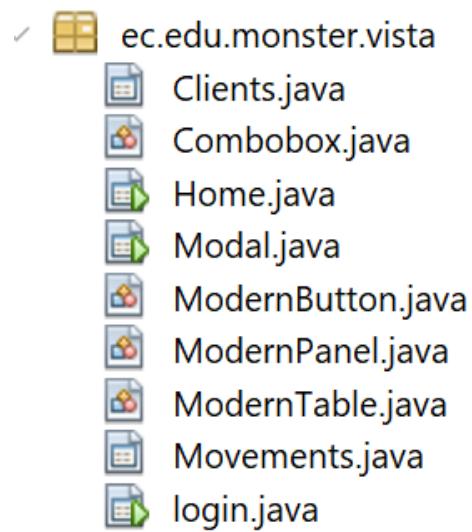


Figura 62 Creación de las vistas

Se diseña primero el login



Figura 63 Prueba a la interfaz gráfica principal

The screenshot shows the 'Gestión de Cuentas' (Account Management) section of the Eureka Bank application. At the top right is a close button ('X'). The title 'Gestión de Cuentas' is displayed above a sub-instruction 'Administra y consulta las cuentas bancarias'. To the left of the main content area is a sidebar with the Eureka Bank logo and two menu items: 'Cuentas de Clientes' and 'Movimientos'. Below these is a 'Cerrar Sesión' (Logout) button. The main content area contains a table showing five bank accounts:

Cuenta	Cliente	Moneda	Estado	Saldo
00100001	ARANDA LUNA, ...	01	ACTIVO	6962.00
00100002	ARANDA LUNA, ...	02	ACTIVO	4500.00
00200001	FLORES CHAF...	01	ACTIVO	6994.00
00200002	CORONEL CAS...	01	ACTIVO	6832.00
00200003	CHAVEZ CANAL...	02	ACTIVO	6144.00

Figura 64 Prueba a la interfaz gráfica de gestión de cuentas

The screenshot shows the 'Movimientos Bancarios' (Banking Movements) section of the Eureka Bank application. At the top right is a close button ('X'). The title 'Movimientos Bancarios' is displayed above a sub-instruction 'Revisa los movimientos realizados entre las cuentas bancarias'. To the left of the main content area is a sidebar with the Eureka Bank logo and two menu items: 'Cuentas de Clientes' and 'Movimientos'. Below these is a 'Cerrar Sesión' (Logout) button. The main content area contains a table showing recent banking movements:

Cuenta	Nro	Fecha	Tipo	Importe	Saldo
00200003	30	2025-11-12	Deposito En...	30.00	6,144.00
00200003	29	2025-11-12	Transferenc...	50.00	6,114.00
00200003	28	2025-11-12	Retiro Salida	50.00	6,164.00
00200003	27	2025-11-12	Deposito En...	22.00	6,214.00
00200003	26	2025-11-12	Deposito En...	11.00	6,192.00
00200003	25	2025-11-12	Deposito En...	1.00	6,181.00
00200001	17	2025-11-12	Transferenc...	60.00	6,994.00
00200001	16	2025-11-12	Transferenc...	54.00	7,054.00
00200000	14	2025-11-10	Deposito En...	20.00	6,000.00

Figura 65 Prueba a la interfaz gráfica de consulta de movimientos bancarios

TABLA 10 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))
        .addContainerGap(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(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 11 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().addContainerGap())
);
```

```
.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 12 DepositoView

```
/*
```

```
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
* Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JPanel.java to edit this template
*/
package ec.edu.monster.vista;

import ec.edu.monster.servicios.EuBankService;
import ec.edu.monster.ws.eurekabank.DatosCuenta;
import java.util.List;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author Dome
 */
public class Clients extends javax.swing.JPanel {
```

```
private EuBankService euBankService = new EuBankService();

/**
 * Creates new form Clients
 */

public Clients() {

    initComponents();
    cargarDatos();
}

public void setEuBankService(EuBankService euBankService) {

    this.euBankService = euBankService;

    // AHORA que ya tenemos el servicio, cargamos los datos.
}
```

```
private void cargarDatos() {  
    // Verificación por si algo falla  
    if (this.euBankService == null) {  
        System.err.println("Error: EuBankService no fue inyectado en el panel Clients.");  
        return;  
    }  
  
    // 1. Definir las columnas  
    String[] columnNames = {"Cuenta", "Cliente", "Moneda", "Estado", "Saldo"};  
  
    // 2. Crear un modelo de tabla vacío  
    DefaultTableModel model = new DefaultTableModel(columnNames, 0) {  
        // Hacemos que las celdas no sean editables  
        @Override  
        public boolean isCellEditable(int row, int column) {  
            return false;  
        }  
    };  
}
```

```
}

};

try {
    // 3. ¡Llamar al servicio SOAP!
    List<DatosCuenta> cuentas = this.euBankService.traerCuentasConClientes();

    // 4. Llenar el modelo con los datos
    for (DatosCuenta cuenta : cuentas) {
        Object[] row = new Object[] {
            cuenta.getCodigo(),
            cuenta.getNombreCliente(),
            cuenta.getMoneda(),
            cuenta.getEstado(),
            cuenta.getSaldo() // El JTable sabe cómo mostrar BigDecimal
        };
    }
}
```

```
model.addRow(row);
}

// 5. Asignar el modelo a la tabla
tableClients.getTable().setModel(model);

} catch (Exception ex) {
    // Mostrar un error si el SOAP falla
    JOptionPane.showMessageDialog(this, "Error al cargar datos de clientes: " + ex.getMessage(), "Error de Conexión", JOptionPane.ERROR_MESSAGE);
}

}

/** 
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
```

```
* regenerated by the Form Editor.

*/
@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();
    tableClients = 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("Gestión de Cuentas");  
  
lblTitle1.setFont(new java.awt.Font("Roboto", 0, 14)); // NOI18N  
lblTitle1.setForeground(new java.awt.Color(102, 102, 102));
```

```
lblTitle1.setText("Administra y consulta las cuentas bancarias");

javax.swing.GroupLayout modernPanel2Layout = new javax.swing.GroupLayout(modernPanel2);
modernPanel2.setLayout(modernPanel2Layout);
modernPanel2Layout.setHorizontalGroup(
    modernPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(modernPanel2Layout.createSequentialGroup()
        .addGap(14, 14, 14)
        .addComponent(lblimagesulli)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(modernPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(lblTitle)
            .addComponent(lblTitle1))
        .addContainerGap(203, Short.MAX_VALUE)))
);
modernPanel2Layout.setVerticalGroup(
```

```
modernPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(lblimagesulli)
    .addGroup(modernPanel2Layout.createSequentialGroup()
        .addGap(23, 23, 23)
        .addComponent(lblTitle)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(lblTitle1))
);

tableClients.setBackground(new java.awt.Color(255, 255, 255));
tableClients.setBorder(null);
tableClients.setBorderRadius(0);
tableClients.setContainerBackground(new java.awt.Color(255, 255, 255));
tableClients.setEvenRowColor(new java.awt.Color(242, 250, 253));
tableClients.setHeaderBackground(new java.awt.Color(153, 214, 234));
tableClients.setHoverRowColor(new java.awt.Color(204, 255, 255));
```

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

tableClients.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(28, 28, 28)
        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
            .addComponent(tableClients, 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))
        .addGapContainerGap(42, Short.MAX_VALUE)))
);
jPanel1Layout.setVerticalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel1Layout.createSequentialGroup()
        .addGap(28, 28, 28)
        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
            .addComponent(tableClients, 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))
        .addGapContainerGap(42, Short.MAX_VALUE)))
);
```

```
.addGroup(jPanel1Layout.createSequentialGroup()
    .addGap(21, 21, 21)
    .addComponent(modernPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE)
    .addGap(18, 18, 18)
    .addComponent(tableClients, javax.swing.GroupLayout.PREFERRED_SIZE, 390, javax.swing.GroupLayout.PREFERRED_SIZE)
    .addContainerGap(37, Short.MAX_VALUE))
);

add(jPanel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 680, 570));
}// </editor-fold>

// Variables declaration - do not modify
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 tableClients;  
  
// End of variables declaration  
  
}
```

4.5 CREACIÓN DEL PROYECTO WEB

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

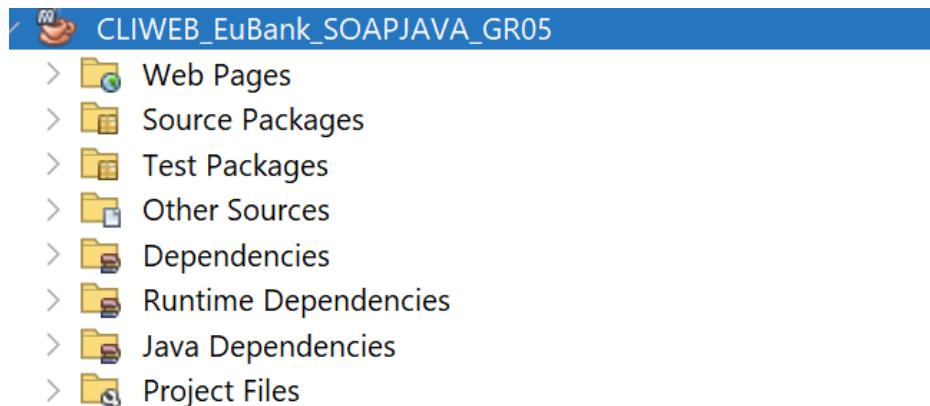


Figura 66 Creación directorio 02.CLIENTE WEB

4.5.1 CREACIÓN ENTORNO

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

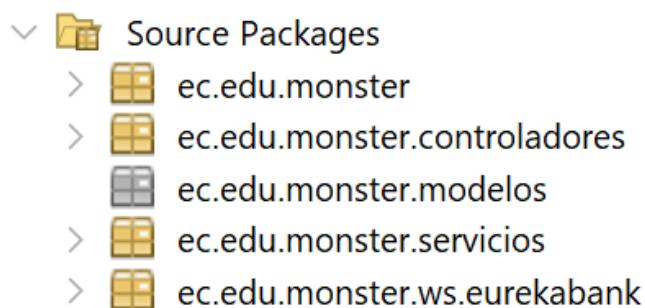


Figura 67 Creación Estructura Web

4.5.2 CREACIÓN CAPA SERVICIO

TABLA 13 Codificación App.py

```
package ec.edu.monster.servicios;

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change
this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */

/**
 *
 * @author joela
 */
import ec.edu.monster.ws.eurekabank.DatosCuenta;
import ec.edu.monster.ws.eurekabank.MovimientoData;
import ec.edu.monster.ws.eurekabank.ResultadoOperacion;
import ec.edu.monster.ws.eurekabank.WSEurekaBank;
import ec.edu.monster.ws.eurekabank.WSEurekaBank_Service;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class EuBankService {
    private final WSEurekaBank port;

    public EuBankService() {
        // Inicializa el cliente SOAP
        WSEurekaBank_Service service = new WSEurekaBank_Service();
        this.port = service.getWSEurekaBankPort();
    }
}
```

```

// 1. Login (Asumiendo que el login está en el mismo WSDL)
public boolean iniciarSesion(String usuario, String contrasena) {
    // El nombre "iniciarSesion" debe coincidir con el generado
    return port.iniciarSesion(usuario, contrasena);
}

// 2. Traer Cuentas
public List<DatosCuenta> traerCuentasConClientes() {
    return port.traerCuentasConClientes();
}

// 3. Traer Movimientos
public List<MovimientoData> traerMovimientos(String cuenta) {
    return port.traerMovimientos(cuenta);
}

// 4. Traer Tipos de Movimiento
public List<String> obtenerTipoMovimientosPermitidos() {
    return port.tipoMovimientosPermitidos();
}

// 5. Registrar Movimiento
public ResultadoOperacion regMovimiento(String tipo, String cuentaOrigen, String cuentaDestino, double importe) {
    // El "regMovimiento" debe coincidir con el generado
    return port.regMovimiento(cuentaOrigen, cuentaDestino, tipo, importe);
}

```

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 14 Codificación Vista Principal

```
<!DOCTYPE html>
<html lang="es" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Login - EuBank</title>
    <script src="https://cdn.tailwindcss.com"></script>
    <script th:inline="javascript">
        // Configuración de Tailwind con los colores de EuBank
        tailwind.config = {
            theme: {
                extend: {
                    colors: {
                        'eubank-primary': '#3B82F6',
                        'eubank-secondary': '#2563EB',
                    }
                }
            }
        }
    </script>
    <style>
        /* Estilos para los iconos en los inputs */
        .form-icon {
            position: absolute;
            left: 12px;
            top: 50%;
            transform: translateY(-50%);
            color: #6b7280;
        }
    </style>

```

```

        z-index: 10;
    }

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

    .input-with-icon input {
        padding-left: 40px !important; /* !important para sobreescribir tailwind */
    }

```

</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 SOAP Java</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>

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

<form method="post" th: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 - SOAP.JAVA Group 05</small>
    </div>
</div>

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

</body>
</html>

```

TABLA 15 Codificación realizarDepósitos

```

<!DOCTYPE html>
<html lang="es" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title th:text="${pageTitle ?: 'Dashboard'} + ' - EuBank'"></title>

    <script src="https://cdn.tailwindcss.com"></script>
    <script th:inline="javascript">
        tailwind.config = {
            theme: {
                extend: {
                    colors: {
                        'eubank-primary': '#3B82F6',
                        'eubank-secondary': '#2563EB',
                    }
                }
            }
        }
    </script>

```

```

        }

</script>

<script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>

<style>
.loading-spinner {
    border: 4px solid #f3f3f3;
    border-top: 4px solid #8B5CF6; /* Color eubank */
    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>
</head>
<body class="bg-gray-50 min-h-screen">

<header class="bg-white shadow-sm border-b border-gray-200">
<nav class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
<div class="flex justify-between h-16">
<div class="flex">
<div class="flex-shrink-0 flex items-center">

</div>
<div class="hidden sm:ml-6 sm:flex sm:space-x-8">
<a th:if="${session.Usuario != null}" th:href="@{/dashboard}"
class="border-transparent text-gray-500 hover:border-gray-300 hover:text-gray-700 inline-flex items-center px-1 pt-1 border-b-2 text-sm font-medium">
    Panel de Control
</a>
</div>

```

```

        </div>
        <div class="hidden sm:ml-6 sm:flex sm:items-center" th:if="${session.Usuario != null}">
            <div class="flex items-center space-x-4">
                <span class="text-sm text-gray-700">
                    Usuario: <strong class="text-eubank-primary" th:text="${session.Usuario}"></strong>
                </span>
                <a th:href="@{/logout}" class="text-red-600 hover:text-red-800 px-3 py-2 rounded-md text-sm font-medium transition duration-200">
                    Cerrar Sesión
                </a>
            </div>
        </div>
        </div>
    </nav>
</header>

<div class="max-w-7xl mx-auto py-6 sm:px-6 lg:px-8">
    <main role="main">

        <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>
    </main>
</div>

```

```

</div>

<div class="tab-content">
  <div id="cuentas" class="tab-pane active">
    <div class="bg-white rounded-lg 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">
          <div class="mb-6 bg-gradient-to-r from-blue-50 to-blue-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 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 0-2v1h16V6a2 2 0 0 0-2H4z"></path>
                <path fill-rule="evenodd" d="M18 9H2v5a2 2 0 0 2h12a2 2 0 0 2-2V9zM4 13a1 1 0 0 1-1h1a1 1 0 110 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="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 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>

<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">
                <div class="mb-6 bg-gradient-to-r from-blue-50 to-blue-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 Bancarios</h6>
    <p class="text-sm text-gray-600">Revisa los movimientos realizados entre las cuentas bancarias.</p>
</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 0-2v1h16V6a2 2 0 0 0-2H4z"></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 10 2H5a1 1 0 0 1-1zm5-1a1 1 0 1 100 2h1a1 1 0 1 100-2H9z" 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>

<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>
                <tr>
                    <td>Cuenta A&#x2022;</td>
                    <td>12345678901234567890</td>
                    <td>2023-10-01</td>
                    <td>Deposito</td>
                    <td>1000.00</td>
                    <td>1000.00</td>
                    <td>Saldo Actual</td>
                </tr>
                <tr>
                    <td>Cuenta B</td>
                    <td>98765432109876543210</td>
                    <td>2023-10-02</td>
                    <td>Retiro</td>
                    <td>-500.00</td>
                    <td>500.00</td>
                    <td>Saldo Actual</td>
                </tr>
            </tbody>
        </table>
    </div>
</div>

```

```

        </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>
</div>
</div>

<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">
            <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>
        <div class="px-6 py-4">
            <div class="flex gap-6">
                <div class="flex-1">
                    <form id="formCrearMovimiento">
                        <div class="space-y-4">
                            <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>
    <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 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>
    <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 id="modalError" class="mt-4 bg-red-100 border border-red-400
    text-red-700 px-4 py-3 rounded-lg hidden"></div>
    <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"
            class="px-6 py-2 bg-gradient-to-r from-eubank-primary to-
            eubank-secondary text-white rounded-lg hover:shadow-lg transform hover:-translate-y-0.5
            transition duration-200">
            Crear Movimiento
        </button>
    </div>
</form>
</div>
<div id="ImagenTipoMovimiento" class="hidden flex-shrink-0 w-86 flex
items-center justify-center">
    <div class="text-center">
        <img id="imgTipoMovimiento" src="" alt="Tipo de Movimiento"
        class="w-[50vh] object-contain">
        <p id="textoTipoMovimiento" class="mt-3 text-sm font-semibold text-
        eubank-primary"></p>
    </div>
    </div>
    </div>
    </div>
    </div>
    </div>
    </div>
    </div>
</main>
</div>

<th:block th:fragment="footer-scripts">
<script th:inline="javascript">
/*<![CDATA[*/
// --- ¡TRADUCCIÓN DE URLs! ---

```

```

const URL_GET CUENTAS = /*[@{/dashboard/get-cuentas}]]*/ 'error';
const URL_GET MOVIMIENTOS = /*[@{/dashboard/get-movimientos}]]*/ 'error';
const URL_GET TODOS MOVIMIENTOS = /*[@{/dashboard/get-todos-movimientos}]]*/ 'error';
const URL_GET TIPOS MOVIMIENTO = /*[@{/dashboard/get-tipos-movimiento}]]*/ 'error';
const URL_TEST SERVICE = /*[@{/dashboard/get-tipos-movimiento}]]*/ 'error'; // Re-usamos get-tipos

// --- ¡TRADUCCIÓN DE IMÁGENES! ---
const imagenesMovimiento = {
  'DEPOSITO': { img: /*[@{/images/Deposito.png}]]*/ "", texto: 'Depósito' },
  'RETIRO': { img: /*[@{/images/Retiro.png}]]*/ "", texto: 'Retiro' },
  'TRANSFERENCIA': { img: /*[@{/images/Transferencia.png}]]*/ "", texto: 'Transferencia' },
};

let cuentasData = [];
let tiposMovimiento = [];

// --- ¡TODO TU CÓDIGO JS DE .NET PEGADO AQUÍ! ---

$(document).ready(function() {
  console.log('Página cargada (Java/Thymeleaf)');

  $.get(URL_TEST_SERVICE, () => console.log('Servicio OK'))
    .fail(() => console.error('Error conectando al servicio SOAP'));

  cargarTiposMovimiento();
  cargarCuentas();

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

```

```

$('.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');

$('.tab-pane').addClass('hidden');

$('#' + tabName).removeClass('hidden');

if (tabName === 'movimientos' && !$('#selectCuenta').val()) {
    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();

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

```

```

        $('#divCuentaDestino').addClass('hidden');
        $('#cuentaDestino').removeAttr('required').val("");
    }

    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())
    };

    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;
    }

    crearMovimiento(formData);
}

```

```

});

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

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

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

}); // Fin de $(document).ready

function cargarTiposMovimiento() {
    console.log('Cargando tipos de movimiento...');
    $.ajax({
        url: URL_GET_TIPOS_MOVIMIENTO,
        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...');
    $('#cuentasContent').addClass('hidden');
    $('#cuentasError').addClass('hidden');
    $('#cuentasLoading').removeClass('hidden');
    $.ajax({
        url: URL_GET CUENTAS,
        type: 'GET',
        success: function(data) {
            console.log('Respuesta del servidor (cuentas):', 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 || 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">${cuenta.codigo}</td>
                <td class="px-6 py-4 whitespace nowrap text-sm font-medium text-gray-900">${cuenta.nombreCliente}</td>
                <td class="px-6 py-4 whitespace nowrap text-sm font-medium text-gray-900">${cuenta.emailCliente}</td>
                <td class="px-6 py-4 whitespace nowrap text-sm font-medium text-gray-900">${cuenta.telefonoCliente}</td>
                <td class="px-6 py-4 whitespace nowrap text-sm font-medium text-gray-900">${cuenta.moneda}</td>
            
        `;
        tbody.append(row);
    });
}

```

```

        <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 = $('#selectCuenta');
    select.empty();
    select.append('<option value="">-- Todas las cuentas --</option>');
    cuentas.forEach(function(cuenta) {
        select.append('<option value="${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);
}

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

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

    $.ajax({
        url: URL_GET_MOVIMIENTOS,
        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_GET_TODOS_MOVIMIENTOS,
        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');

```

```

const titulo = todosLosMovimientos ? 'Todos los Movimientos' : `Movimientos de la
Cuenta ${codigoCuenta}`;

$('#movimientosTitle').text(titulo);

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

if (!movimientos || movimientos.length === 0) {
    const mensaje = todosLosMovimientos ? 'No hay movimientos en ninguna cuenta'
    : `No hay movimientos para la cuenta ${codigoCuenta}`;
    $('#movimientosContent').addClass('hidden');
    $('#movimientosEmpty').text(mensaje).removeClass('hidden');
    return;
}

$('#movimientosEmpty').addClass('hidden');

movimientos.forEach(function(mov) {
    const tipoBadge = obtenerBadgeTipo(mov.tipo);
    // Los objetos de fecha de SOAP pueden ser complejos, los parseamos
    const fecha = new Date(mov.fecha).toLocaleDateString('es-ES', {
        year: 'numeric', month: '2-digit', day: '2-digit',
        hour: '2-digit', minute: '2-digit'
    });

    let cuentaInfo = '';
    if (todosLosMovimientos) {
        const cuenta = cuentasData.find(c => c.codigo === mov.codigoCuenta);
        cuentaInfo = cuenta ? `${cuenta.codigo} - ${cuenta.nombreCliente}` :
        mov.codigoCuenta;
    } else {
        const cuenta = cuentasData.find(c => c.codigo === codigoCuenta);
        cuentaInfo = cuenta ? `${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) {
    // Normaliza el tipo (ej. "DEPOSITO" -> "Deposito")
    let tipoNormalizado = tipo.charAt(0).toUpperCase() + tipo.slice(1).toLowerCase();

    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[tipoNormalizado] || { 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 = '$'; // default USD
    if (moneda === '01') simbolo = 'S/'; // Soles
    else if (moneda === '02') simbolo = '$'; // USD

    let numCantidad = Number(cantidad);
    if (isNaN(numCantidad)) {
        numCantidad = 0;
    }

    return simbolo + ' ' + numCantidad.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_CREAR_MOVIMIENTO,
    type: 'POST',
    contentType: 'application/json',
    data: JSON.stringify(datos),
    success: function(response) {
        console.log('Respuesta del servidor:', response);
        if (response.success) {
            cerrarModal();
            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);
    }
});

}

function refrescarMovimientosActuales() {
    const cuentaSeleccionada = $('#selectCuenta').val();
    if (cuentaSeleccionada) {
        cargarMovimientosCuentaEspecifica(cuentaSeleccionada);
    } else {
        cargarTodosMovimientos();
    }
}

/*]]>*/

</script>
</th:block>

```

```
</body>  
</html>
```

4.5.4 EJECUCION

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

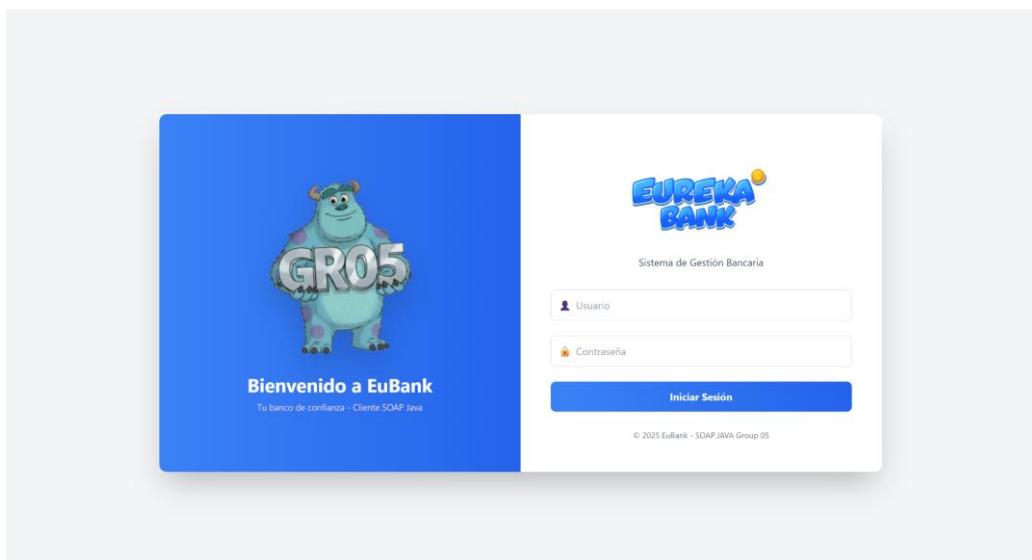


Figura 68 Ejecución Vista Inicio de Sesión

A screenshot of the EuBank main account management page. At the top, there is a navigation bar with links for "Cuentas de Clientes" (selected) and "Movimientos". On the right, it shows the user information "Usuario: MONSTER" and a "Cerrar Sesión" button. The main content area is titled "Lista de Cuentas" and features a sub-header "Gestión de Cuentas" with the sub-instruction "Administra y consulta las cuentas bancarias". Below this are three filter dropdowns: "Filtrar por Estado" (Todos), "Filtrar por Moneda" (Todas), and "Buscar por Cliente" (Nombre del cliente). A table lists six bank accounts with columns for Código, Cliente, Email, Teléfono, Moneda, Saldo, and Estado. The accounts are: ARANDA LUNA, ALAN ALBERTO (Saldo \$/ 6962.00, Estado Activo); ARANDA LUNA, ALAN ALBERTO (Saldo \$/ 4500.00, Estado Activo); FLORES CHAFLOQUE, ROSA LIZET (Saldo \$/ 6994.00, Estado Activo); CORONEL CASTILLO, ERIC GUSTAVO (Saldo \$/ 6832.00, Estado Activo); and CHAVEZ CANALES, EDGAR RAFAEL (Saldo \$/ 6144.00, Estado Activo).

Figura 69 Ejecución Vista Principal

Panel de Control

Usuario: MONSTER Cerrar Sesión

Cuentas de Clientes Movimientos

Movimientos Bancarios



Movimientos Bancarios
Revisa los movimientos realizados entre las cuentas bancarias.

Filtrar por Cuenta (opcional):
-- Todas las cuentas -- + Crear

Todos los Movimientos

CUENTA	NÚMERO	FECHA	TIPO	REFERENCIA	IMPORTE	SALDO ACTUAL
00200003 - CHAVEZ CANALES, EDGAR RAFAEL	30	12/11/2025, 00:00	Depósito Entrada	-	\$ 30.00	\$ 6144.00
00200003 - CHAVEZ CANALES, EDGAR RAFAEL	29	12/11/2025, 00:00	Transferencia Salida	00200002	\$ 50.00	\$ 6144.00
00200003 - CHAVEZ CANALES, EDGAR RAFAEL	28	12/11/2025, 00:00	Retiro Salida	-	\$ 50.00	\$ 6164.00
00200003 - CHAVEZ CANALES, EDGAR RAFAEL	27	12/11/2025, 00:00	Depósito Entrada	-	\$ 22.00	\$ 6214.00
00200003 - CHAVEZ CANALES, EDGAR RAFAEL	26	12/11/2025, 00:00	Depósito Entrada	-	\$ 11.00	\$ 6192.00
00200003 - CHAVEZ CANALES, EDGAR RAFAEL	25	12/11/2025, 00:00	Depósito Entrada	-	\$ 1.00	\$ 6181.00

Figura 70 Ejecución Vista Consultar Movimientosv

Panel de Control

Usuario: MONSTER Cerrar Sesión

Cuentas de Clientes Movimientos

Filtrar por Cuenta (opcional):
-- Todas las cuentas --

Crear Nuevo Movimiento

Tipo de Movimiento:
DEPÓSITO

Cuenta Origen:
Ej: 00100001

Monto:
0.00

Cancelar Crear Movimiento Depósito

Todos los Movimientos

CUENTA	NÚMERO	FECHA	TIPO	REFERENCIA	IMPORTE	SALDO ACTUAL
00200003 - CHAVEZ CANALES, EDGAR RAFAEL	30	12/11/2025, 00:00	Depósito Entrada	-	\$ 30.00	\$ 6144.00
00200003 - CHAVEZ CANALES, EDGAR RAFAEL	29	12/11/2025, 00:00	Transferencia Salida	00200002	\$ 50.00	\$ 6144.00
00200003 - CHAVEZ CANALES, EDGAR RAFAEL	28	12/11/2025, 00:00	Retiro Salida	-	\$ 50.00	\$ 6164.00
00200003 - CHAVEZ CANALES, EDGAR RAFAEL	27	12/11/2025, 00:00	Depósito Entrada	-	\$ 22.00	\$ 6214.00
00200003 - CHAVEZ CANALES, EDGAR RAFAEL	26	12/11/2025, 00:00	Depósito Entrada	-	\$ 11.00	\$ 6192.00
00200003 - CHAVEZ CANALES, EDGAR RAFAEL	25	12/11/2025, 00:00	Depósito Entrada	-	\$ 1.00	\$ 6181.00

Figura 71 Ejecución Vista Realizar Depósitos

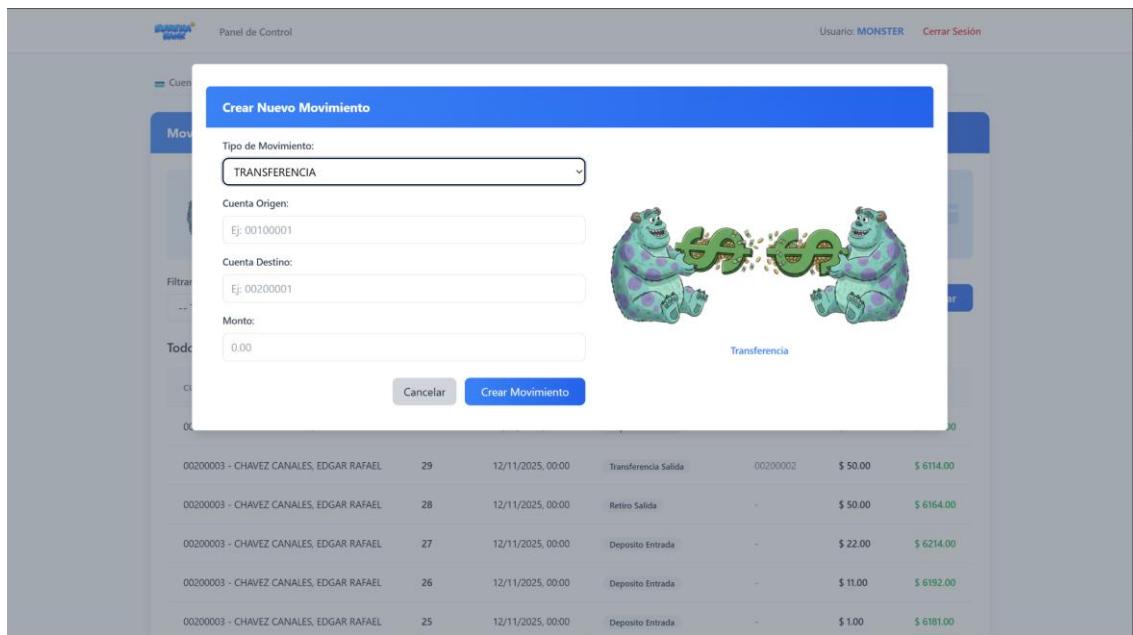


Figura 72 Ejecución Vista Realizar Transferencia

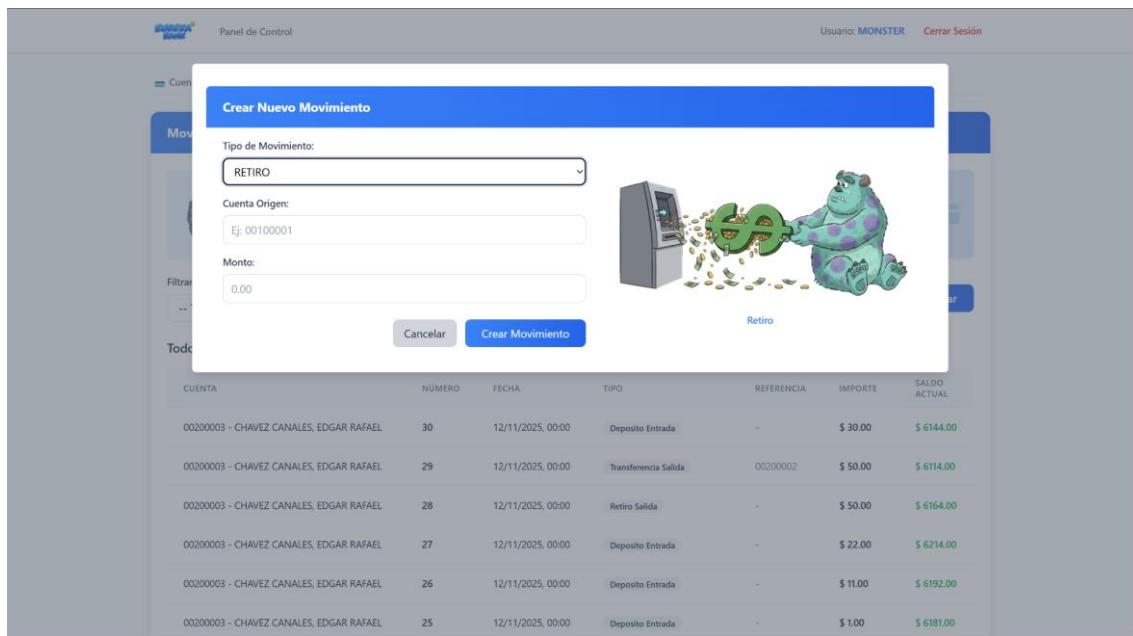


Figura 73 Ejecución Vista Retiro

4.6 CREACIÓN DEL PROYECTO MÓVIL

4.6.1 CREACIÓN PROYECTO MÓVIL

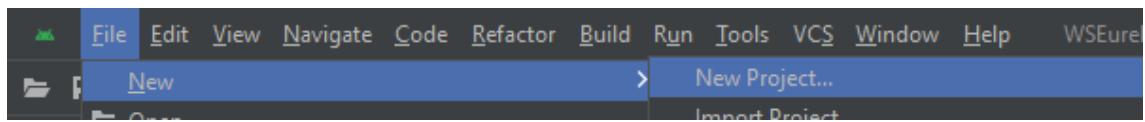


Figura 74 Creación Nuevo Proyecto

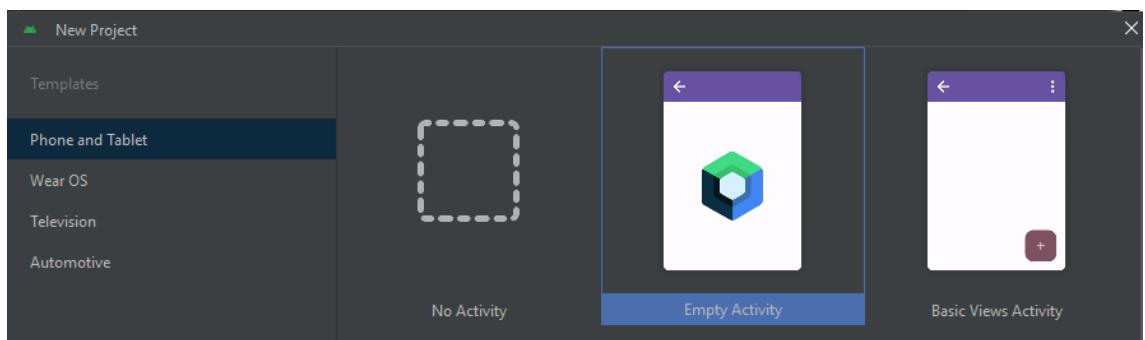


Figura 75 Creación Empty Activity

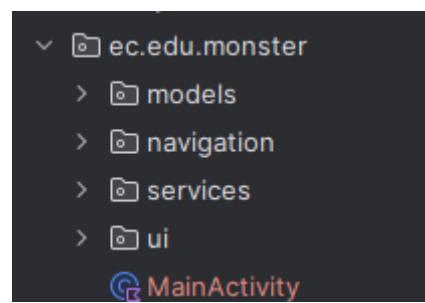


Figura 76 Configuración Nuevo Proyecto

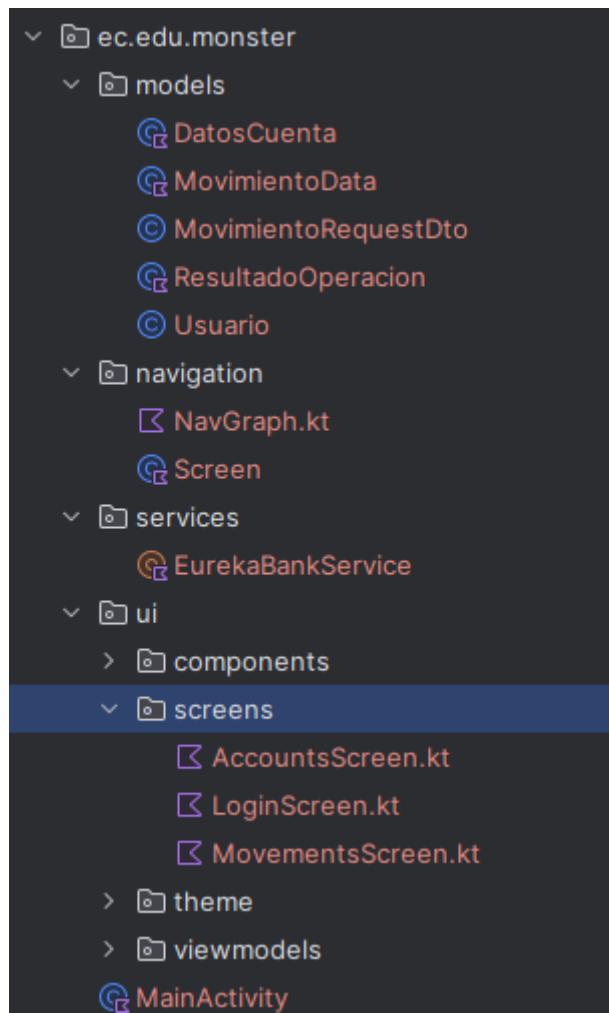


Figura 77 Estructura proyecto

4.6.2 CREACIÓN CAPA MODELO

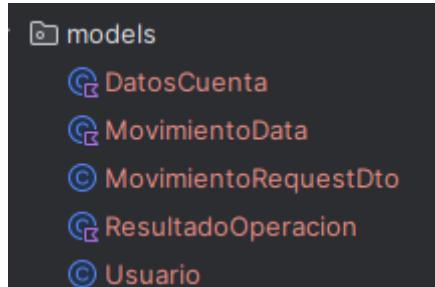


Figura 78 Creación nueva clase en paquete modelo

```
1 package ec.edu.monster.models
2
3 > import ...
5
11 Usages
6 data class MovimientoData(
7     var codigoCuenta: String = "",
8     var numero: Int = 0,
9     var fecha: Date? = null,
10    var tipo: String = "",
11    var importe: BigDecimal = BigDecimal.ZERO,
12    var referencia: String? = null,
13    var saldoActual: BigDecimal = BigDecimal.ZERO,
14    var nombreCliente: String = ""
15 )
```

Figura 79 Codificación clase MovimientoData

TABLA 16 Codificación clase MovimientoData

```
package ec.edu.monster.models

import java.math.BigDecimal
import java.util.*
```

```
data class MovimientoData(  
    var codigoCuenta: String = "",  
    var numero: Int = 0,  
    var fecha: Date? = null,  
    var tipo: String = "",  
    var importe: BigDecimal = BigDecimal.ZERO,  
    var referencia: String? = null,  
    var saldoActual: BigDecimal = BigDecimal.ZERO,  
    var nombreCliente: String = ""  
)
```

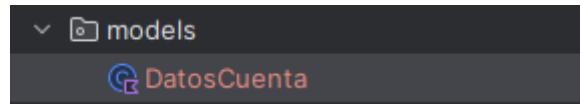


Figura 80 Creación clase DatosCuenta

```
1 package ec.edu.monster.models
2
3 > import ...
4
5
6
7
8
9 < data class DatosCuenta(
10
11     var codigo: String = "",
12     var nombreCliente: String = "",
13     var saldo: BigDecimal = BigDecimal.ZERO,
14     var estado: String = "",
15     var moneda: String = "",
16     var emailCliente: String = ""
17 ) : KvmSerializable {
18
19     override fun getProperty(index: Int): Any = when (index) {
20         0 -> codigo
21         1 -> nombreCliente
22         2 -> saldo
23         3 -> estado
24         4 -> moneda
25         5 -> emailCliente
26         6 -> telefonoCliente
27         else -> throw IndexOutOfBoundsException()
28     }
29
30     override fun getPropertyCount(): Int = 7
31
32     override fun setProperty(index: Int, value: Any) {
33         when (index) {
34             0 -> codigo = value.toString()
35         }
36     }
37 }
```

Figura 81 Codificación clase DatosCuenta

TABLA 17 Codificación clase DatosCuenta

```
package ec.edu.monster.models

import org.ksoap2.serialization.KvmSerializable
import org.ksoap2.serialization.PropertyInfo
import java.math.BigDecimal
import java.util.Hashtable

data class DatosCuenta(
    var codigo: String = "",
    var nombreCliente: String = "",
    var saldo: BigDecimal = BigDecimal.ZERO,
    var estado: String = "",
    var moneda: String = "",
    var emailCliente: String = "",
    var telefonoCliente: String = ""
) : KvmSerializable {

    override fun getProperty(index: Int): Any = when (index) {
        0 -> codigo
        1 -> nombreCliente
        2 -> saldo
        3 -> estado
        4 -> moneda
        5 -> emailCliente
        6 -> telefonoCliente
        else -> throw IndexOutOfBoundsException()
    }

    override fun getPropertyCount(): Int = 7

    override fun setProperty(index: Int, value: Any) {
        when (index) {
```

```

0 -> codigo = value.toString()
1 -> nombreCliente = value.toString()
2 -> saldo = if (value is BigDecimal) value else BigDecimal(value.toString())
3 -> estado = value.toString()
4 -> moneda = value.toString()
5 -> emailCliente = value.toString()
6 -> telefonoCliente = value.toString()
else -> throw IndexOutOfBoundsException()

}

}

override fun getPropertyInfo(index: Int, properties: Hashtable<*, *>?, info: PropertyInfo) {
when (index) {
0 -> { info.name = "Codigo"; info.type = PropertyInfo.STRING_CLASS }
1 -> { info.name = "NombreCliente"; info.type = PropertyInfo.STRING_CLASS }
2 -> { info.name = "Saldo"; info.type = BigDecimal::class.java }
3 -> { info.name = "Estado"; info.type = PropertyInfo.STRING_CLASS }
4 -> { info.name = "Moneda"; info.type = PropertyInfo.STRING_CLASS }
5 -> { info.name = "EmailCliente"; info.type = PropertyInfo.STRING_CLASS }
6 -> { info.name = "TelefonoCliente"; info.type = PropertyInfo.STRING_CLASS }
}
}
}

```

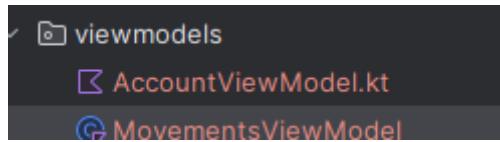


Figura 82 Creación clases de MovementsViewModel

Se crea la los modelos de carga de datos que se usaran posteriormente en las vistas.

```
ekaBankService.kt      MovementsViewModel.kt    AccountViewModel.kt
```

```
package ec.edu.monster.ui.viewmodels

import ...

2 Usages
class MovementsViewModel : ViewModel() {
    2 Usages
    private val _movimientos = MutableStateFlow<List<MovimientoData>>( value =
        1 Usage
        val movimientos: StateFlow<List<MovimientoData>> = _movimientos.asStateFlow()

    3 Usages
    private val _isLoading = MutableStateFlow( value = false)
    1 Usage
    val isLoading: StateFlow<Boolean> = _isLoading.asStateFlow()

    3 Usages
    private val _error = MutableStateFlow<String?>( value = null)
    val error: StateFlow<String?> = _error.asStateFlow()

    2 Usages
    fun loadMovimientos(cuenta: String) {
        _isLoading.value = true
        _error.value = null

        viewModelScope.launch {
            EurekaBankService.traerMovimientos(cuenta) { lista, err ->
                _movimientos.value = lista
                _error.value = err
                _isLoading.value = false
            }
        }
    }
}
```

Figura 83 Modelos de Carga de Datos

TABLA 18 Codificación clase MovementsViewModel

```
package ec.edu.monster.ui.viewmodels

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import ec.edu.monster.models.MovimientoData
import ec.edu.monster.services.EurekaBankService
import kotlinx.coroutines.*
import kotlinx.coroutines.launch

class MovementsViewModel : ViewModel() {

    private val _movimientos = MutableStateFlow<List<MovimientoData>>(emptyList())
    val movimientos: StateFlow<List<MovimientoData>> = _movimientos.asStateFlow()

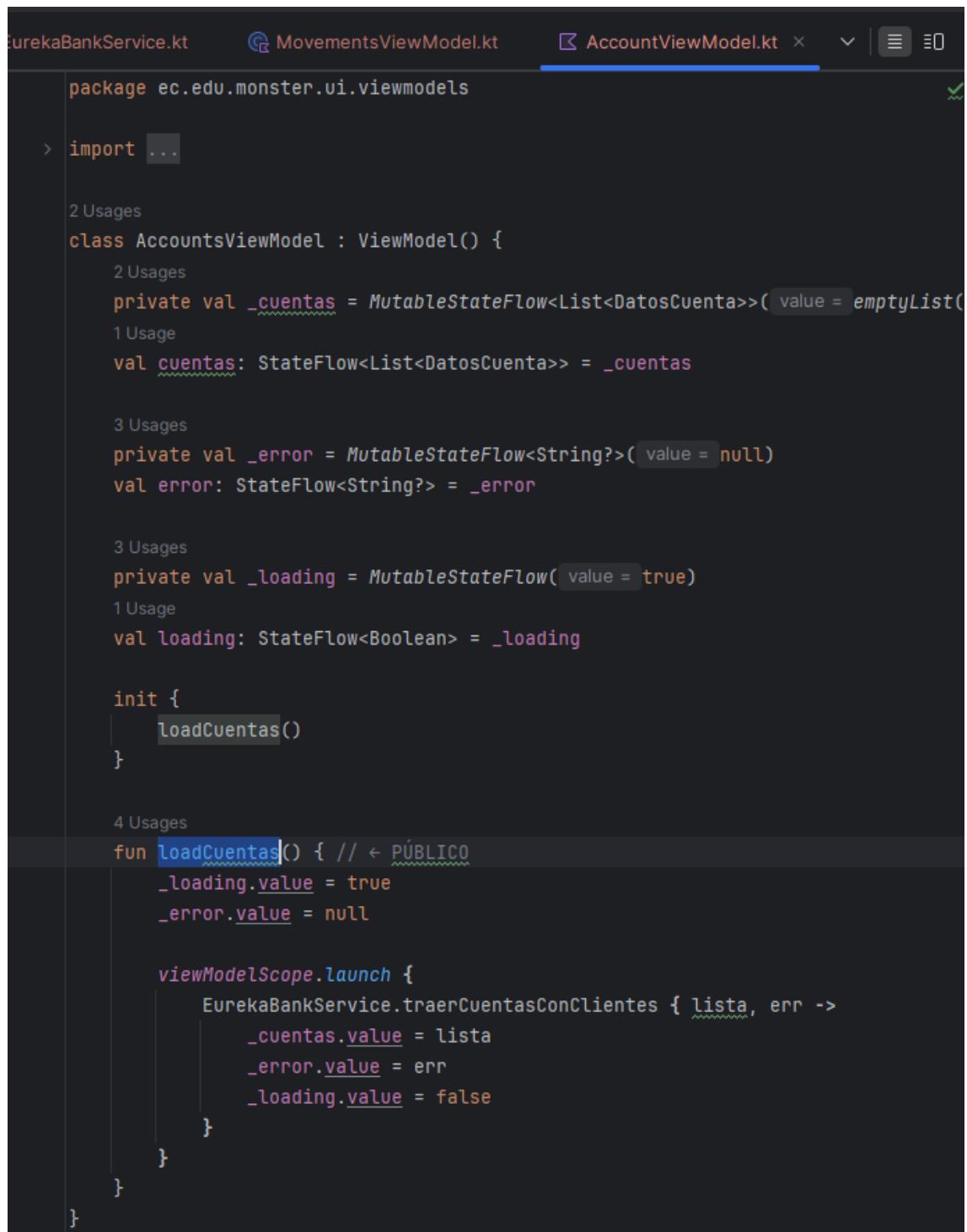
    private val _isLoading = MutableStateFlow(false)
    val isLoading: StateFlow<Boolean> = _isLoading.asStateFlow()

    private val _error = MutableStateFlow<String?>(null)
    val error: StateFlow<String?> = _error.asStateFlow()

    fun loadMovimientos(cuenta: String) {
        _isLoading.value = true
        _error.value = null

        viewModelScope.launch {
            EurekaBankService.traerMovimientos(cuenta) { lista, err ->
                _movimientos.value = lista
                _error.value = err
                _isLoading.value = false
            }
        }
    }
}
```

Se crea el modelo de la vista que maneja la carga de las cuentas



```
package ec.edu.monster.ui.viewmodels

> import ...

2 Usages
class AccountsViewModel : ViewModel() {
    2 Usages
    private val _cuentas = MutableStateFlow<List<DatosCuenta>>( value = emptyList()
        1 Usage
        val cuentas: StateFlow<List<DatosCuenta>> = _cuentas

    3 Usages
    private val _error = MutableStateFlow<String?>( value = null)
    val error: StateFlow<String?> = _error

    3 Usages
    private val _loading = MutableStateFlow( value = true)
    1 Usage
    val loading: StateFlow<Boolean> = _loading

    init {
        loadCuentas()
    }

    4 Usages
    fun loadCuentas(): // ← PÚBLICO
        _loading.value = true
        _error.value = null

        viewModelScope.launch {
            EurekaBankService.traerCuentasConClientes { lista, err ->
                _cuentas.value = lista
                _error.value = err
                _loading.value = false
            }
        }
    }
}
```

Figura 84 Modelo de Vista Carga Cuentas

TABLA 19 Codificación clase AccountViewModel

```
package ec.edu.monster.ui.viewmodels

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import ec.edu.monster.models.DatosCuenta
import ec.edu.monster.services.EurekaBankService
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.launch

class AccountsViewModel : ViewModel() {

    private val _cuentas = MutableStateFlow<List<DatosCuenta>>(emptyList())
    val cuentas: StateFlow<List<DatosCuenta>> = _cuentas

    private val _error = MutableStateFlow<String?>(null)
    val error: StateFlow<String?> = _error

    private val _loading = MutableStateFlow(true)
    val loading: StateFlow<Boolean> = _loading

    init {
        loadCuentas()
    }

    fun loadCuentas() { // ← PÚBLICO
        _loading.value = true
        _error.value = null

        viewModelScope.launch {
            EurekaBankService.traerCuentasConClientes { lista, err ->
                _cuentas.value = lista
                _error.value = err
                _loading.value = false
            }
        }
    }
}
```

```
    }  
}  
}  
}
```

4.6.3. CREACIÓN CAPA SERVICIO

En esta clase se crea la conexión entre el Cliente Móvil y el servidor utilizando la librería KSoap2 para el intercambio de datos.

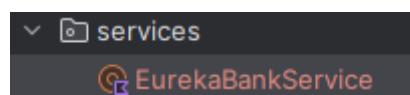


Figura 85 Creación clase EurekaBankService

```
1 package ec.edu.monster.services
2
3 > import ...
4
5
6 11 Usages
7 object EurekaBankService {
8     15 Usages
9     private const val TAG = "EurekaBankService"
10    10 Usages
11    private const val NAMESPACE = "http://ws.monster.edu.ec/"
12    5 Usages
13    private const val URL = "http://192.168.137.1:8080/WS_EurekaBank_SOAPJAVA_GR05/WSEU"
14    2 Usages
15    private const val METHOD_LOGIN = "iniciarSesion"
16    2 Usages
17    private const val METHOD_CUENTAS = "traerCuentasConClientes"
18    2 Usages
19    private const val METHOD_MOVIMIENTOS = "traerMovimientos"
20    2 Usages
21    private const val METHOD_REG_MOVIMIENTO = "regMovimiento"
22    // Hilo principal
23    14 Usages
24    private val mainHandler = Handler(Looper.getMainLooper())
25
26    1 Usage
27    fun iniciarSesion(
28        usuario: String,
29        contrasena: String,
30        callback: (success: Boolean, message: String?) -> Unit
31    ) {
32        Thread {
33            try {
34                Log.d(TAG, msg = "Iniciando sesión para usuario: $usuario")
35                val request = SoapObject(NAMESPACE, name = METHOD_LOGIN).apply {
36
37
38
39
```

Figura 86 Instancias de clase EurekaBankService

TABLA 20 Codificación clase EurekaBankService

```
package ec.edu.monster.services

import android.util.Log
import org.ksoap2.SoapEnvelope
import org.ksoap2.serialization.SoapObject
```

```

import org.ksoap2.serialization.SoapSerializationEnvelope
import org.ksoap2.transport.HttpTransportSE
import android.os.Handler
import android.os.Looper
import ec.edu.monster.models.DatosCuenta
import ec.edu.monster.models.MovimientoData
import ec.edu.monster.models.ResultadoOperacion
import org.ksoap2.serialization.SoapPrimitive
import java.math.BigDecimal
import java.text.SimpleDateFormat
import java.util.Locale

object EurekaBankService {
    private const val TAG = "EurekaBankService"
    private const val NAMESPACE = "http://ws.monster.edu.ec/"
    private const val URL = "http://192.168.137.1:8080/WS_EurekaBank_SOAPJAVA_GR05/WSEurekaBank?wsdl"
    private const val METHOD_LOGIN = "iniciarSesion"
    private const val METHOD_CUENTAS = "traerCuentasConClientes"
    private const val METHOD_MOVIMIENTOS = "traerMovimientos"
    private const val METHOD_REG_MOVIMIENTO = "regMovimiento"
    // Hilo principal
    private val mainHandler = Handler(Looper.getMainLooper())

    fun iniciarSesion(
        usuario: String,
        contrasena: String,
        callback: (success: Boolean, message: String?) -> Unit
    ) {
        Thread {
            try {
                Log.d(TAG, "Iniciando sesión para usuario: $usuario")

                val request = SoapObject(NAMESPACE, METHOD_LOGIN).apply {
                    addProperty("usuario", usuario)

```

```

        addProperty("contrasena", contrasena)
    }

    val envelope = SoapSerializationEnvelope(SoapEnvelope.VER11).apply {
        dotNet = false
        setOutputSoapObject(request)
    }

    val transport = HttpTransportSE(URL).apply {
        debug = true
    }

    val soapAction = "$NAMESPACE$METHOD_LOGIN"
    transport.call(soapAction, envelope)

    Log.d(TAG, "==== SOAP REQUEST DUMP ====")
    Log.d(TAG, transport.requestDump ?: "No request dump")
    Log.d(TAG, "==== SOAP RESPONSE DUMP ====")
    Log.d(TAG, transport.responseDump ?: "No response dump")

    val responseBody = envelope.bodyIn as? SoapObject
    val resultadoStr = try {
        responseBody?.getProperty("resultado")?.toString()
    } catch (e: Exception) {
        "null"
    }

    Log.d(TAG, "Valor de <resultado>: '$resultadoStr'")

    val result = resultadoStr?.trim()?.equals("true", ignoreCase = true) == true

    // LLAMAR CALLBACK EN HILO PRINCIPAL
    mainHandler.post {
        callback(result, if (result) "Login exitoso" else "Credenciales incorrectas")
    }
}

```

```
    }

} catch (e: Exception) {
    Log.e(TAG, "Error en SOAP", e)
    mainHandler.post {
        callback(false, "Error: ${e.message}")
    }
}

}.start()

}

fun traerCuentasConClientes(
    callback: (List<DatosCuenta>, String?) -> Unit
) {
    Thread {
        try {
            val request = SoapObject(NAMESPACE, METHOD_CUENTAS)

            val envelope = SoapSerializationEnvelope(SoapEnvelope.VER11).apply {
                dotNet = false
                setOutputSoapObject(request)
            }

            val transport = HttpTransportSE(URL).apply { debug = true }
            transport.call("$NAMESPACE$METHOD_CUENTAS", envelope)

            Log.d(TAG, "Cuentas Response: ${transport.responseDump}")

            val body = envelope.bodyIn as? SoapObject
            if (body == null) {
                mainHandler.post { callback(emptyList(), "Sin respuesta") }
                return@Thread
            }
        }
    }
}
```

```

val cuentas = mutableListOf<DatosCuenta>()

// ITERAR LAS 5 CUENTAS DIRECTAMENTE
for (i in 0 until body.propertyCount) {
    val cuentaObj = body.getProperty(i) as? SoapObject
    if (cuentaObj != null) {
        val cuenta = DatosCuenta().apply {
            codigo = cuentaObj.getPropertySafelyAsString("Codigo") ?: ""
            emailCliente = cuentaObj.getPropertySafelyAsString("EmailCliente") ?: ""
            estado = cuentaObj.getPropertySafelyAsString("Estado") ?: ""
            moneda = cuentaObj.getPropertySafelyAsString("Moneda") ?: ""
            nombreCliente = cuentaObj.getPropertySafelyAsString("NombreCliente") ?: ""

            saldo = cuentaObj.getPropertySafelyAsString("Saldo")?.let { BigDecimal(it) } ?: BigDecimal.ZERO
        }
        cuentas.add(cuenta)
    }
}

Log.d(TAG, "Cuentas parseadas: ${cuentas.size}")

mainHandler.post {
    callback(cuentas, null)
}

} catch (e: Exception) {
    Log.e(TAG, "Error al traer cuentas", e)
    mainHandler.post {
        callback(emptyList(), "Error: ${e.message}")
    }
}
}.start()

```

```

}

fun traerMovimientos(
    cuenta: String,
    callback: (List<MovimientoData>, String?) -> Unit
) {
    Thread {
        try {
            val request = SoapObject(NAMESPACE, METHOD_MOVIMIENTOS).apply {
                addProperty("cuenta", cuenta)
            }

            val envelope = SoapSerializationEnvelope(SoapEnvelope.VER11).apply {
                dotNet = false
                setOutputSoapObject(request)
            }

            val transport = HttpTransportSE(URL).apply { debug = true }
            transport.call("$NAMESPACE$METHOD_MOVIMIENTOS", envelope)

            Log.d(TAG, "Movimientos Response: ${transport.responseDump}")

            val body = envelope.bodyIn as? SoapObject
            if (body == null) {
                mainHandler.post { callback(emptyList(), "Sin respuesta") }
                return@Thread
            }

            val movimientos = mutableListOf<MovimientoData>()

            // MISMA LÓGICA QUE CUENTAS: body.propertyCount = número de movimientos
            for (i in 0 until body.propertyCount) {
                val movObj = body.getProperty(i) as? SoapObject
                if (movObj != null) {

```

```

        val mov = MovimientoData().apply {
            codigoCuenta = movObj.getPropertySafelyAsString("CodigoCuenta") ?: ""
            numero = movObj.getPropertySafelyAsString("Numero")?.toIntOrNull() ?: 0
            fecha = try {
                val dateStr = movObj.getPropertySafelyAsString("Fecha")
                SimpleDateFormat("yyyy-MM-dd", Locale.getDefault()).parse(dateStr)
            } catch (e: Exception) { null }
            tipo = movObj.getPropertySafelyAsString("Tipo") ?: ""
            importe = movObj.getPropertySafelyAsString("Importe")?.let { BigDecimal(it) } ?: BigDecimal.ZERO
            referencia = movObj.getPropertySafelyAsString("Referencia")
            saldoActual = movObj.getPropertySafelyAsString("SaldoActual")?.let { BigDecimal(it) } ?: BigDecimal.ZERO
            nombreCliente = movObj.getPropertySafelyAsString("NombreCliente") ?: ""
        }
        movimientos.add(mov)
    }
}

Log.d(TAG, "Movimientos parseados: ${movimientos.size}")
mainHandler.post { callback(movimientos, null) }

} catch (e: Exception) {
    Log.e(TAG, "Error al traer movimientos", e)
    mainHandler.post { callback(emptyList(), "Error: ${e.message}") }
}
}.start()
}

fun regMovimiento(
    cuentaOrigen: String,
    cuentaDestino: String?,
    tipo: String,
    importe: Double,
    callback: (ResultadoOperacion?, String?) -> Unit
)

```

```

) {
    Thread {
        try {
            val request = SoapObject(NAMESPACE, METHOD_REG_MOVIMIENTO).apply {
                addProperty("cuentaOrigen", cuentaOrigen)
                if (cuentaDestino != null) addProperty("cuentaDestino", cuentaDestino)
                addProperty("tipo", tipo)
                // CORREGIDO: String en vez de Double
                addProperty("importe", importe.toString())
            }

            val envelope = SoapSerializationEnvelope(SoapEnvelope.VER11).apply {
                dotNet = false
                setOutputSoapObject(request)
            }

            val transport = HttpTransportSE(URL).apply { debug = true }
            transport.call("$NAMESPACE$METHOD_REG_MOVIMIENTO", envelope)

            Log.d(TAG, "RegMovimiento Response: ${transport.responseDump}")

            val body = envelope.bodyIn as? SoapObject
            if (body == null) {
                mainHandler.post { callback(null, "Sin respuesta") }
                return@Thread
            }

            val resultadoObj = body.getPropertySafely("resultado") as? SoapObject
            val resultado = if (resultadoObj != null) {
                ResultadoOperacion(
                    codigo = resultadoObj.getPropertySafelyAsString("codigo")?.toIntOrNull() ?: 0,
                    mensaje = resultadoObj.getPropertySafelyAsString("mensaje") ?: "Sin mensaje"
                )
            } else {

```

```

        ResultadoOperacion(-1, "Formato inválido")
    }

    mainHandler.post { callback(resultado, null) }

} catch (e: Exception) {
    Log.e(TAG, "Error al registrar movimiento", e)
    mainHandler.post { callback(null, e.message) }
}

}.start()
}

private const val METHOD_TIPOS = "tipoMovimientosPermitidos"

fun obtenerTiposMovimiento(
    callback: (List<String>, String?) -> Unit
) {
    Thread {
        try {
            val request = SoapObject(NAMESPACE, METHOD_TIPOS)
            val envelope = SoapSerializationEnvelope(SoapEnvelope.VER11).apply {
                dotNet = false
                setOutputSoapObject(request)
            }

            val transport = HttpTransportSE(URL).apply { debug = true }
            transport.call("$NAMESPACE$METHOD_TIPOS", envelope)

            val body = envelope.bodyIn as? SoapObject
            if (body == null) {
                mainHandler.post { callback(emptyList(), "Sin respuesta") }
                return@Thread
            }

            val tipos = mutableListOf<String>()

```

```

        for (i in 0 until body.propertyCount) {
            val item = body.getProperty(i)
            if (item is SoapPrimitive) {
                tipos.add(item.toString())
            }
        }

        mainHandler.post { callback(tipos, null) }

    } catch (e: Exception) {
        mainHandler.post { callback(emptyList(), e.message) }
    }
}.start()
}

private fun Any.getPropertySafely(name: String): Any? {
    return try {
        (this as? SoapObject)?.getProperty(name)
    } catch (e: Exception) {
        null
    }
}

private fun Any.getPropertySafelyAsString(name: String): String? {
    return getPropertySafely(name)?.toString()
}

}

```

MainActivity

Figura 87 Modificación clase MainActivity

TABLA 21 Codificación clase MainActivity

```
package ec.edu.monster

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import ec.edu.monster.navigation.AppNavGraph
import androidx.compose.material3.Surface
import androidx.compose.material3.MaterialTheme
import androidx.navigation.compose.rememberNavController
import ec.edu.monster.ui.theme.CLIMOV_EurekaBank_SOAPJAVA_GR05Theme
import dagger.hilt.android.AndroidEntryPoint // IMPORT
@AndroidEntryPoint
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            CLIMOV_EurekaBank_SOAPJAVA_GR05Theme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                )
            }
            val navController = rememberNavController()
```

```
        AppNavGraph(navController = navController)
    }
}
}
}
```

4.6.4 Creación Capa Vista

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

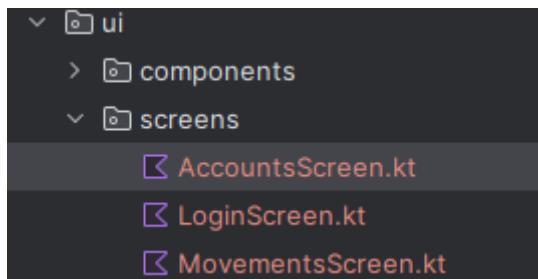


Figura 88 Creación archivos en la carpeta screens

TABLA 22 Codificación archivo AccountsScreen

```
package ec.edu.monster.ui.screens

import androidx.compose.foundation.Image
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.KeyboardArrowRight
import androidx.compose.material.icons.filled.Refresh
import androidx.compose.material3.*
```

```
import androidx.compose.runtime.Composable
import androidx.compose.runtime.LaunchedEffect
import androidx.compose.runtime.collectAsState
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.lifecycle.viewmodel.compose.viewModel
import ec.edu.monster.ui.theme.EurekaBankSOAPJava
import ec.edu.monster.ui.viewmodels.AccountsViewModel
import ec.edu.monster.R // ← OBLIGATORIO
import androidx.compose.ui.unit.dp

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun AccountsScreen(
    onAccountClick: (String) -> Unit,
    viewModel: AccountsViewModel = viewModel()
) {
    val cuentas = viewModel.cuentas.collectAsState().value
    val loading = viewModel.loading.collectAsState().value
    val error = viewModel.error.collectAsState().value

    LaunchedEffect(Unit) {
        viewModel.loadCuentas()
    }

    Scaffold(
        topBar = {
            CenterAlignedTopAppBar(
                title = {
                    Row(

```

```

        verticalAlignment = Alignment.CenterVertically,
        horizontalArrangement = Arrangement.Center
    ) {
    Image(
        painter = painterResource(id = R.drawable.sulli_general),
        contentDescription = "Logo",
        modifier = Modifier.size(32.dp).padding(end = 8.dp)
    )
    Text("Cuentas", fontWeight = FontWeight.Bold)
}
},
colors = TopAppBarDefaults.centerAlignedTopAppBarColors(
    containerColor = EurekaBankSOAPJava.primary,
    titleContentColor = EurekaBankSOAPJava.onPrimary
),
actions = {
// BOTÓN DE RECARGAR
    IconButton(onClick = { viewModel.loadCuentas() }) {
        Icon(
            imageVector = Icons.Default.Refresh,
            contentDescription = "Recargar",
            tint = EurekaBankSOAPJava.onPrimary
        )
    }
}
)
}
) { padding ->
Box(modifier = Modifier.padding(padding)) {
when {
loading -> {
CircularProgressIndicator(modifier = Modifier.align(Alignment.Center))
}
error != null -> {

```

```

Column(
    horizontalAlignment = Alignment.CenterHorizontally,
    modifier = Modifier.align(Alignment.Center)
) {
    Text("Error: $error", color = MaterialTheme.colorScheme.error)
    Spacer(Modifier.height(8.dp))
    Button(onClick = { viewModel.loadCuentas() }) {
        Text("Reintentar")
    }
}
}

cuentas.isEmpty() -> {
    Text("No hay cuentas disponibles", modifier = Modifier.align(Alignment.Center))
}
else -> {
    LazyColumn {
        items(cuentas) { cuenta ->
            Card(
                modifier = Modifier
                    .fillMaxWidth()
                    .padding(EurekaBankSOAPJava.spacingMedium)
                    .clickable { onAccountClick(cuenta.codigo) },
                colors      =      CardDefaults.cardColors(containerColor      =
EurekaBankSOAPJava.surface)
            ) {
                ListItem(
                    headlineContent = { Text(cuenta.codigo, fontWeight = FontWeight.Bold)
                },
                supportingContent = {
                    Column {
                        Text("${cuenta.nombreCliente} • ${cuenta.moneda}")
                        Text("Saldo:      S/      ${cuenta.saldo}",      color      =
EurekaBankSOAPJava.primary, fontWeight = FontWeight.Medium)
                        Text("${cuenta.estado} • ${cuenta.emailCliente}")
                    }
                }
            }
        }
    }
}

```

TABLA 23 Codificación archivo LoginScreen

```
package ec.edu.monster.ui.screens

import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.unit.dp
import androidx.compose.ui.tooling.preview.Preview
```

```

import ec.edu.monster.services.EurekaBankService
import ec.edu.monster.ui.theme.EurekaBankSOAPJava
import ec.edu.monster.R

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun LoginScreen(
    onLoginSuccess: () -> Unit,
    onLoginError: (String) -> Unit
) {
    var usuario by remember { mutableStateOf("") }
    var contrasena by remember { mutableStateOf("") }
    var loading by remember { mutableStateOf(false) }
    var mensaje by remember { mutableStateOf<String?>(null) }

    Scaffold(
        containerColor = EurekaBankSOAPJava.background
    ) { padding ->
        Column(
            modifier = Modifier
                .fillMaxSize()
                .padding(padding)
                .padding(EurekaBankSOAPJava.spacingLarge),
            horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.Center
        ) {

            // Logo Principal
            Image(
                painter = painterResource(id = R.drawable.sulli_logo),
                contentDescription = "Logo Principal",
                modifier = Modifier
                    .size(120.dp)
                    .padding(bottom = 12.dp) // ← Reducido, elegante
            )
        }
    }
}

```

```

    )

// Logo Secundario + Título (juntos)
Column(
    horizontalAlignment = Alignment.CenterHorizontally,
    verticalArrangement = Arrangement.spacedBy(4.dp) // ← Muy cerca
) {
    Image(
        painter = painterResource(id = R.drawable.soap_java),
        contentDescription = "Logo SOAP Java",
        modifier = Modifier
            .size(180.dp) // ← Reducido para mejor balance
    )

    Text(
        text = "Sistema de Gestión Bancaria",
        style = MaterialTheme.typography.titleLarge,
        color = EurekaBankSOAPJava.primary,
        modifier = Modifier.padding(horizontal = 16.dp) // ← Centrado con el logo
    )
}

// Campo Usuario
OutlinedTextField(
    value = usuario,
    onValueChange = { usuario = it },
    label = { Text("Usuario") },
    enabled = !loading,
    keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Text),
    modifier = Modifier.fillMaxWidth(),
    colors = OutlinedTextFieldDefaults.colors(
        focusedBorderColor = EurekaBankSOAPJava.primary,
        unfocusedBorderColor = EurekaBankSOAPJava.border
    )
)

```

```
)  
  
Spacer(modifier = Modifier.height(EurekaBankSOAPJava.spacingMedium))  
  
// Campo Contraseña  
OutlinedTextField(  
    value = contrasena,  
    onValueChange = { contrasena = it },  
    label = { Text("Contraseña") },  
    enabled = !loading,  
    visualTransformation = PasswordVisualTransformation(),  
    keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Password),  
    modifier = Modifier.fillMaxWidth(),  
    colors = OutlinedTextFieldDefaults.colors(  
        focusedBorderColor = EurekaBankSOAPJava.primary,  
        unfocusedBorderColor = EurekaBankSOAPJava.border  
    )  
)  
  
Spacer(modifier = Modifier.height(EurekaBankSOAPJava.spacingLarge))  
  
// Botón Login  
Button(  
    onClick = {  
        if (usuario.isBlank() || contrasena.isBlank()) {  
            onLoginError("Complete todos los campos")  
            return@Button  
        }  
        loading = true  
        mensaje = null  
  
        EurekaBankService.iniciarSesion(usuario, contrasena) { success, msg ->  
            loading = false  
            mensaje = msg  
        }  
    }  
)
```

```

        if (success) {
            onLoginSuccess()
        } else {
            onLoginError(msg ?: "Error desconocido")
        }
    },
    enabled = !loading,
    modifier = Modifier
        .fillMaxWidth()
        .height(56.dp),
    shape = RoundedCornerShape(12.dp),
    colors = ButtonDefaults.buttonColors(
        containerColor = EurekaBankSOAPJava.primary,
        disabledContainerColor = EurekaBankSOAPJava.border
    )
)
)

if (loading) {
    CircularProgressIndicator(
        color = EurekaBankSOAPJava.onPrimary,
        modifier = Modifier.size(20.dp),
        strokeWidth = 2.dp
    )
} else {
    Text("Iniciar Sesión", color = EurekaBankSOAPJava.onPrimary)
}
}

// Mensaje de error o éxito
mensaje?.let {
    Spacer(modifier = Modifier.height(EurekaBankSOAPJava.spacingMedium))
    Text(
        text = it,
        color = if (it.contains("exitoso")) EurekaBankSOAPJava.success else
EurekaBankSOAPJava.error,

```

```
        style = MaterialTheme.typography.bodyMedium
    )
}
}
}

@Preview(showBackground = true)
@Composable
fun LoginPreview() {
    LoginScreen(
        onLoginSuccess = {},
        onLoginError = {}
    )
}
```

TABLA 24 Codificación archivo MovementScreen

```
package ec.edu.monster.ui.screens

import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material.icons(Icons)
import androidx.compose.material.icons.filled.Add
import androidx.compose.material.icons.filled.ArrowBack
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
```

```
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import ec.edu.monster.ui.theme.EurekaBankSOAPJava
import ec.edu.monster.ui.viewmodels.MovementsViewModel
import androidx.lifecycle.viewmodel.compose.viewModel
import ec.edu.monster.R
import java.math.BigDecimal
import java.text.SimpleDateFormat
import java.util.*
import ec.edu.monster.ui.components.NuevoMovimientoModal

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun MovementsScreen(
    cuenta: String,
    onBack: () -> Unit,
    viewModel: MovementsViewModel = viewModel()
) {
    val movimientos by viewModel.movimientos.collectAsState()
    val isLoading by viewModel.isLoading.collectAsState()
    val error by viewModel.error.collectAsState()

    var showModal by remember { mutableStateOf(false) }
    LaunchedEffect(cuenta) {
        viewModel.loadMovimientos(cuenta)
    }

    Scaffold(
        topBar = {
            CenterAlignedTopAppBar(
                title = { Row(
                    verticalAlignment = Alignment.CenterVertically,
                    horizontalArrangement = Arrangement.Center
                ) {

```

```

) {
    // ÍCONO: sulli_general.png
    Image(
        painter = painterResource(id = R.drawable.sulli_general),
        contentDescription = "Logo",
        modifier = Modifier
            .size(32.dp)
            .padding(end = 8.dp)
    )
    Text("Movimientos - $cuenta", fontWeight = FontWeight.Bold)
} },
navigationIcon = {
    IconButton(onClick = onBack) {
        Icon(Icons.Filled.ArrowBack, "Volver")
    }
},
colors = TopAppBarDefaults.centerAlignedTopAppBarColors(
    containerColor = EurekaBankSOAPJava.primary,
    titleContentColor = EurekaBankSOAPJava.onPrimary
)
),
floatingActionButton = {
    FloatingActionButton(
        onClick = { showModal = true },
        containerColor = EurekaBankSOAPJava.primary
    ){
        Icon(Icons.Filled.Add, "Nuevo")
    }
}
) { padding ->
    if (showModal) {
        NuevoMovimientoModal(
            cuenta = cuenta,

```

```

        onDismiss = { showModal = false },
        onSuccess = {
            // Recargar movimientos
            viewModel.loadMovimientos(cuenta)
        }
    )
}

Box(modifier = Modifier.padding(padding)) {
    when {
        isLoading -> {
            CircularProgressIndicator(modifier = Modifier.align(Alignment.Center))
        }
        error != null -> {
            Text("Error: $error", color = MaterialTheme.colorScheme.error)
        }
        movimientos.isEmpty() -> {
            Text("No hay movimientos", modifier = Modifier.align(Alignment.Center))
        }
        else -> {
            LazyColumn {
                // CORREGIDO: items(movimientos) no items(size)
                items(movimientos) { mov ->
                    MovementItem(mov)
                    HorizontalDivider()
                }
            }
        }
    }
}

```

@Composable

```

fun MovementItem(mov: ec.edu.monster.models.MovimientoData) {
    ListItem(
        headlineContent = {
            Column {
                // DETECTAR ENTRADA / SALIDA POR EL TIPO
                val esEntrada = mov.tipo.contains("Entrada", ignoreCase = true)
                val esSalida = mov.tipo.contains("Salida", ignoreCase = true)

                val prefijo = if (esEntrada) "+" else if (esSalida) "-" else ""
                val colorImporte = if (esEntrada) {
                    EurekaBankSOAPJava.success // VERDE
                } else if (esSalida) {
                    EurekaBankSOAPJava.error // ROJO
                } else {
                    MaterialTheme.colorScheme.onSurface // gris por defecto
                }

                Text(
                    text = "${mov.tipo} $prefijo${mov.importe}",
                    fontWeight = FontWeight.Bold,
                    color = colorImporte
                )
                Text(
                    text = SimpleDateFormat("dd/MM/yyyy HH:mm", Locale.getDefault())
                        .format(mov.fecha ?: Date()),
                    style = MaterialTheme.typography.bodySmall
                )
            }
        },
        supportingContent = {
            Column {
                if (!mov.referencia.isNullOrEmpty()) {
                    Text("Ref: ${mov.referencia}")
                }
            }
        }
    )
}

```

```
        Text("Saldo: S/ ${mov.saldoActual}")  
    }  
}  
}  
}
```

TABLA 25 Codificación archivo NuevoMovimientoModal

```
package ec.edu.monster.ui.components  
  
import androidx.compose.foundation.Image  
import androidx.compose.foundation.layout.*  
import androidx.compose.foundation.text.KeyboardOptions  
import androidx.compose.material3.*  
import androidx.compose.runtime.*  
import androidx.compose.ui.Alignment  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.res.painterResource  
import androidx.compose.ui.text.font.FontWeight  
import androidx.compose.ui.text.input.KeyboardType  
import androidx.compose.ui.unit.dp  
import ec.edu.monster.services.EurekaBankService  
import ec.edu.monster.models.ResultadoOperacion  
import kotlinx.coroutines.launch  
import ec.edu.monster.R  
  
@OptIn(ExperimentalMaterial3Api::class)  
@Composable  
fun NuevoMovimientoModal(  
    cuenta: String,  
    onDismiss: () -> Unit,  
    
```

```

        onSuccess: () -> Unit
    ) {

        var tipos by remember { mutableStateOf<List<String>?>(null) }
        var tipo by remember { mutableStateOf("") }
        var importe by remember { mutableStateOf("") }
        var cuentaDestino by remember { mutableStateOf("") }
        var isLoading by remember { mutableStateOf(false) }
        var mensaje by remember { mutableStateOf<String?>(null) }
        val coroutineScope = rememberCoroutineScope()

        LaunchedEffect(Unit) {
            EurekaBankService.obtenerTiposMovimiento { lista, err ->
                tipos = lista
                tipo = lista.firstOrNull() ?: ""
            }
        }

        AlertDialog(
            onDismissRequest = onDismiss,
            title = { /* Título vacío, lo ponemos en el content */ },
            text = {
                Column(
                    verticalArrangement = Arrangement.spacedBy(12.dp),
                    horizontalAlignment = Alignment.CenterHorizontally
                ) {
                    // ÍCONO GRANDE ARRIBA
                    val iconRes = when (tipo.uppercase()) {
                        "DEPOSITO" -> R.drawable.deposito
                        "RETIRO" -> R.drawable.retiro
                        "TRANSFERENCIA" -> R.drawable.transferencia
                        else -> R.drawable.deposito // fallback
                    }
                    Image(
                        painter = painterResource(iconRes),

```

```

        contentDescription = tipo,
        modifier = Modifier
            .size(128.dp)
            .padding(bottom = 8.dp)
    )

// TÍTULO
Text(
    text = "Nuevo Movimiento - $cuenta",
    style = MaterialTheme.typography.titleLarge,
    fontWeight = FontWeight.Bold
)

HorizontalDivider()

// TIPOS
if (tipos == null) {
    CircularProgressIndicator()
} else {
    var expanded by remember { mutableStateOf(false) }
    ExposedDropdownMenuBox(
        expanded = expanded,
        onExpandedChange = { expanded = !expanded }
    ) {
        OutlinedTextField(
            value = tipo,
            onValueChange = {},
            readOnly = true,
            label = { Text("Tipo de movimiento") },
            trailingIcon = { ExposedDropdownMenuDefaults.TrailingIcon(expanded = expanded) },
            modifier = Modifier
                .fillMaxWidth()
                .menuAnchor()
        )
    }
}

```

```

ExposedDropdownMenu(
    expanded = expanded,
    onDismissRequest = { expanded = false }
) {
    tipos!!.forEach { option ->
        DropdownMenuItem(
            text = { Text(option) },
            onClick = {
                tipo = option
                expanded = false
            }
        )
    }
}
}

// IMPORTE
OutlinedTextField(
    value = importe,
    onValueChange = { if (it.matches(Regex("[0-9]*\\.?[0-9]*"))) importe = it },
    label = { Text("Importe") },
    keyboardOptions = KeyboardOptions(keyboardType = KeyboardType.Decimal),
    enabled = !isLoading,
    modifier = Modifier.fillMaxWidth()
)

// CUENTA DESTINO
if (tipo.contains("TRANSFERENCIA", ignoreCase = true)) {
    OutlinedTextField(
        value = cuentaDestino,
        onValueChange = { cuentaDestino = it },
        label = { Text("Cuenta Destino") },
        enabled = !isLoading,

```

```

        modifier = Modifier.fillMaxWidth()
    )
}

// MENSAJE
mensaje?.let {
    Text(
        text = it,
        color = if (it.contains("exitos")) MaterialTheme.colorScheme.primary else
MaterialTheme.colorScheme.error,
        style = MaterialTheme.typography.bodyMedium
    )
}
}

},
confirmButton = {
    TextButton(
        onClick = {
            val imp = importe.toDoubleOrNull() ?: run {
                mensaje = "Importe inválido"; return@TextButton
            }
            if (imp <= 0) {
                mensaje = "Importe debe ser mayor a 0"; return@TextButton
            }

            isLoading = true
            mensaje = null
        }
    )
}

coroutineScope.launch {
    EurekaBankService.regMovimiento(
        cuentaOrigen = cuenta,
        cuentaDestino = if (tipo.contains("TRANSFERENCIA")) &&
cuentaDestino.isNotBlank() cuentaDestino else null,
        tipo = tipo,
        importe = imp
    )
}

```

```
        ) { resultado, error ->
            isLoading = false
            if (resultado?.esExito() == true) {
                onSuccess()
                onDismiss()
            } else {
                mensaje = resultado?.mensaje ?: error ?: "Error"
            }
        }
    },
    enabled = !isLoading && tipos != null && importe.isNotBlank()
)
{
    if (isLoading) {
        CircularProgressIndicator(modifier = Modifier.size(16.dp), strokeWidth = 2.dp)
        Spacer(Modifier.width(8.dp))
    }
    Text(if (isLoading) "Enviando..." else "Registrar")
}
),
dismissButton = {
    TextButton(onClick = onDismiss, enabled = !isLoading) {
        Text("Cancelar")
    }
}
)
```

4.6.4 EJECUCIÓN

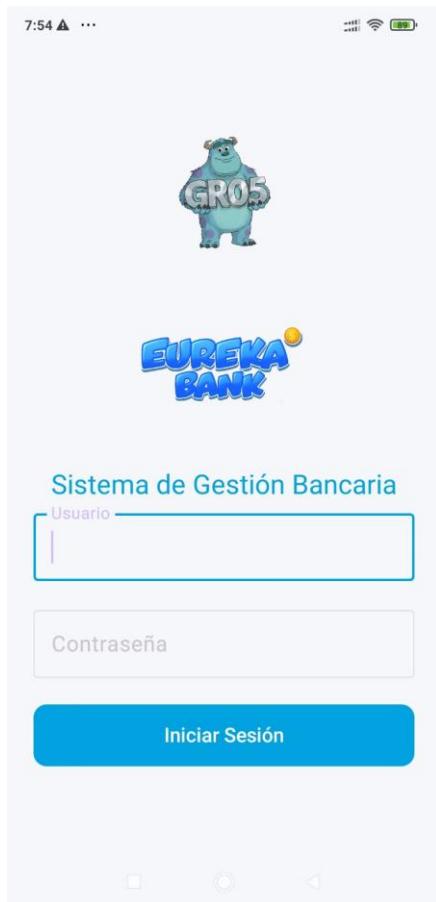


Figura 89 Ejecución Pantalla Inicio de Sesión Móvil

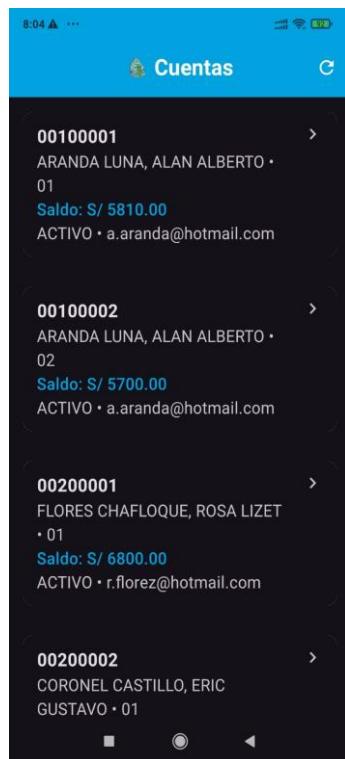


Figura 90 Ejecución Pantalla Principal Móvil



Figura 91 Ejecución Pantalla Depósitos



Figura 92 Ejecución Pantalla Consultas de una cuenta en específico



Figura 93 Ejecución Pantalla Retiro

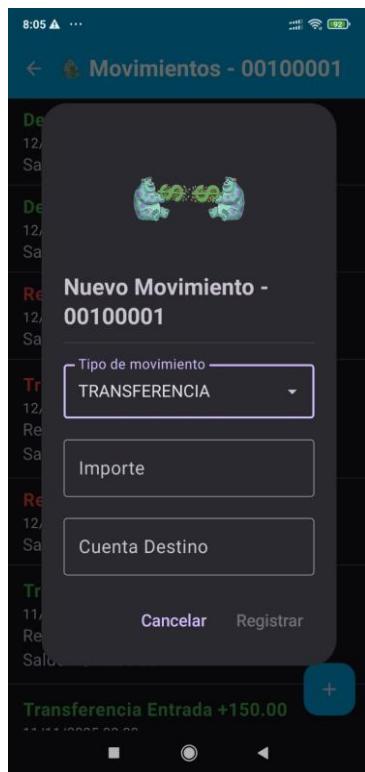


Figura 94 Ejecución Pantalla Transferencia

5. CONCLUSIONES

Los Servicios Web implementados bajo el protocolo SOAP (Simple Object Access Protocol) en Java y desplegados en servidores de aplicaciones robustos como Payara, constituyen una solución de elevada confiabilidad y solidez para contextos empresariales. Su valor radica en la capacidad de garantizar la interoperabilidad, seguridad y precisión en las transacciones de datos. Al adherirse a estándares definidos como WSDL y el formato de intercambio XML, estos servicios facilitan una integración efectiva a través de diversas plataformas y lenguajes, manteniendo la compatibilidad en sistemas inherentemente heterogéneos.

La utilización de *frameworks* especializados como JAX-WS optimiza notablemente el proceso de desarrollo y despliegue, permitiendo que los servicios satisfagan las rigurosas demandas de aplicaciones críticas en sectores como el financiero, sanitario y logístico. Esto consolida a SOAP como una elección arquitectónica estratégica para sistemas que requieren altos niveles de cumplimiento normativo y estabilidad operativa.

Por otra parte, la adopción del patrón de diseño MVC (Modelo-Vista-Controlador), en conjunto con las capacidades de los servicios SOAP y la plataforma Payara, 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 del servidor de aplicaciones Payara en su última versión estable, aprovechando su soporte nativo para JAX-WS y Jakarta EE. Es esencial configurar el entorno para maximizar la seguridad y el rendimiento, asegurando que la plataforma esté alineada con las mejores prácticas de *hosting* de servicios SOAP empresariales.
- Es fundamental adoptar y explotar eficientemente las capacidades del *framework* JAX-WS para la definición e implementación del servicio SOAP. Esto incluye la correcta generación de WSDL y la integración optimizada de datos con la base de datos Google Cloud MySQL a través de JDBC.
- Proveer formación técnica continua a los desarrolladores en conceptos arquitectónicos clave, como el estándar WSDL, la aplicación de seguridad de servicios web (WS-Security) y la implementación rigurosa del patrón MVC (Modelo-Vista-Controlador), garantizando así una base de código consistente, segura y escalable.