*ESCUELAS SALESIANAS MARÍA AUXILIADORA*

CICLO FORMATIVO DE GRADO SUPERIOR

DESARROLLO DE APLICACIONES MULTIPLATAFORMA

|  |
| --- |
| **PROYECTO 4: Banco para la Universidad de Sevilla** |

**ÍNDICE**

1. Estudio del problema y análisis del sistema.

1.1. Introducción.

1.2. Funciones y rendimientos deseados.

1.3. Objetivos.

1.4. Modelado de la solución.

1.4.1. Recursos humanos.

1.4.2. Recursos hardware.

1.4.3. Recursos software.

2. Ejecución de la práctica.

2.1.Documentación técnica e implementación de la aplicación.

3. Documentación del sistema.

3.1. Manual de instalación y configuración de la aplicación.

3.2. Manual de usuario.

4. Conclusiones finales.

4.1.Grado de cumplimiento de los objetivos fijados

4.2. Propuesta de modificaciones o ampliaciones futuras del sistema implementado.

5. Bibliografía.

***1. ESTUDIO DEL PROBLEMA Y ANÁLISIS DEL SISTEMA.***

**1.1 INTRODUCCIÓN**

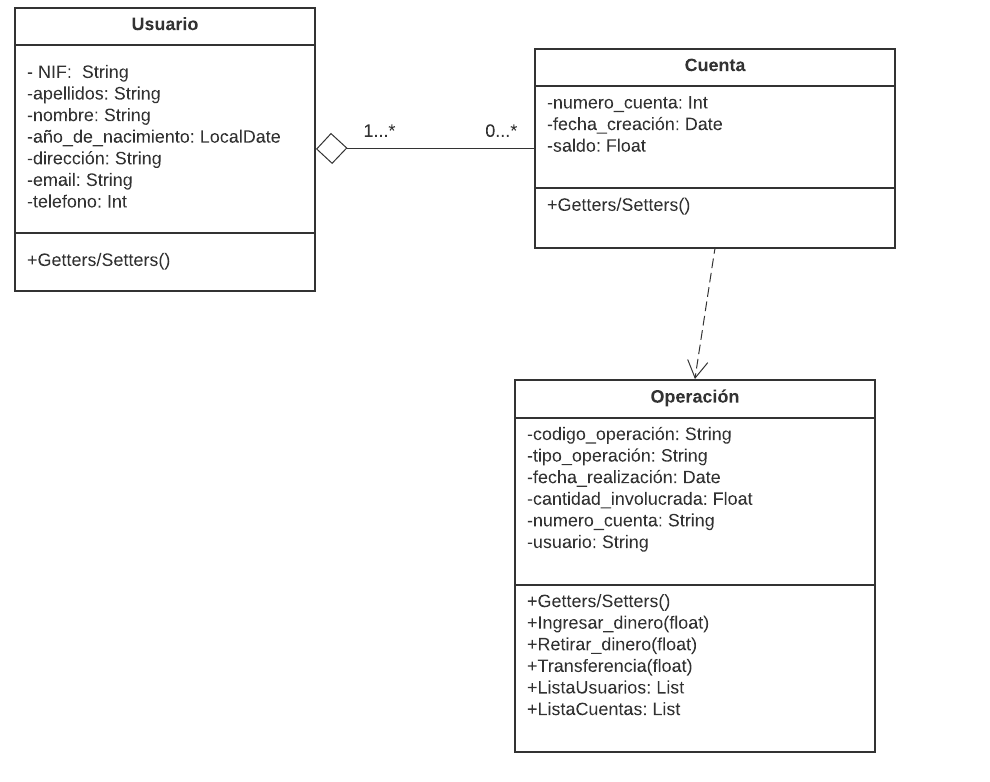
El proyecto está orientado a un entorno formativo. Se trata de una aplicación que ofrece una plataforma para la gestión de una empresa por parte de los administradores.

El enunciado que define las características del proyecto es el siguiente:

Se pretende desarrollar una aplicación que gestione los usuarios y las cuentas de la entidad bancaria asociada al US, US-Caja. Cada miembro de la comunidad universitaria será considerado un potencial usuario de dicha caja, por tanto sus datos se almacenarán en los sistemas de información de la entidad US-Caja. Los datos almacenados para cada usuario potencial son: NIF, apellidos, nombre, año de nacimiento, dirección, e-mail y teléfono de contacto.

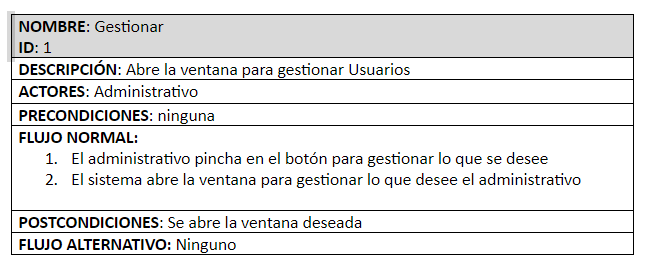
La información almacenada para cada cuenta bancaria es la siguiente: número de cuenta, fecha de creación y saldo que contiene.

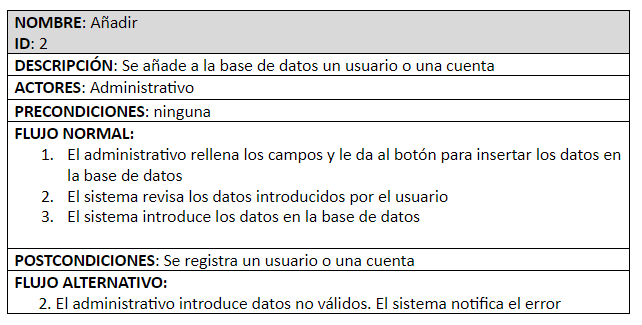
Para cada una de las operaciones bancarias se almacena la siguiente información: código de operación, tipo de operación, fecha de realización, cantidad involucrada en la operación, números de cuenta involucrados en la operación y usuario que realiza la operación.

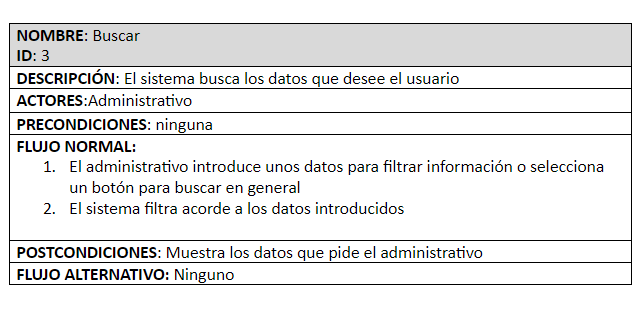


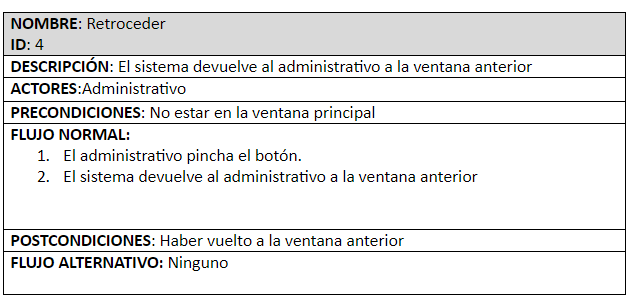
**1.2. FUNCIONES Y RENDIMIENTOS DESEADOS**

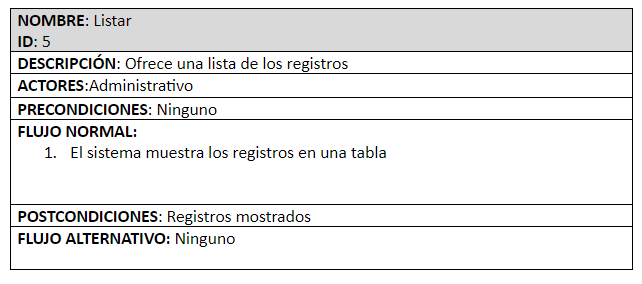
Se desea que la aplicación sea capaz, mediante un control previo, de manipular los datos de los usuarios y las cuentas. Además de la consulta y visualización de la información referida para listar sus datos, así como poder añadir a la base de datos cada uno de estos apartados.

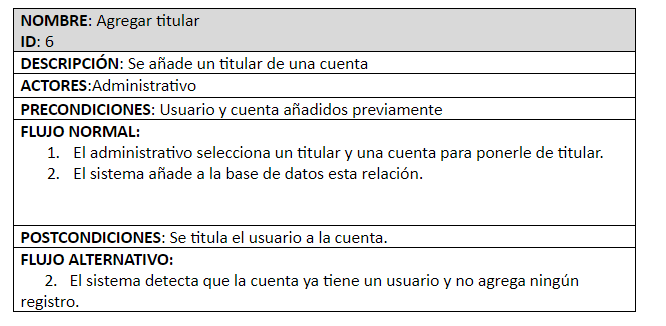


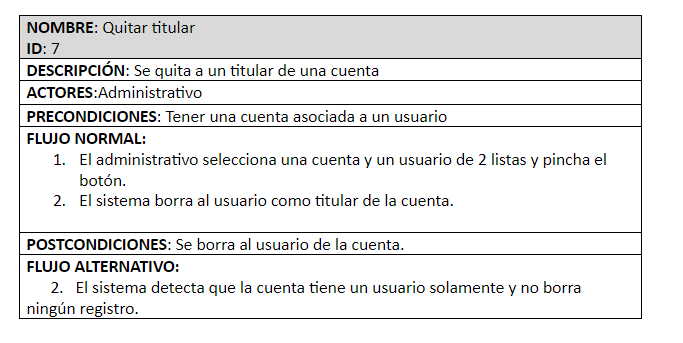


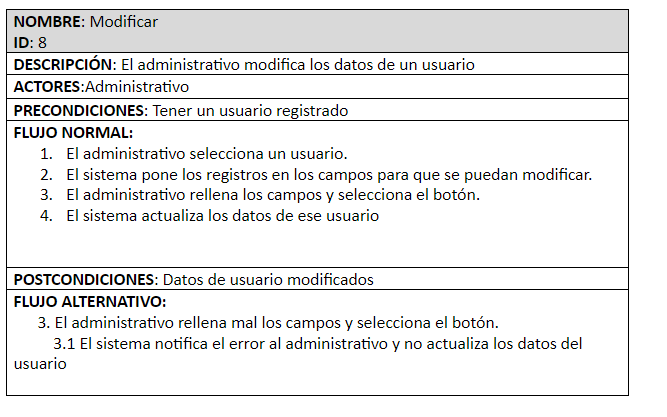


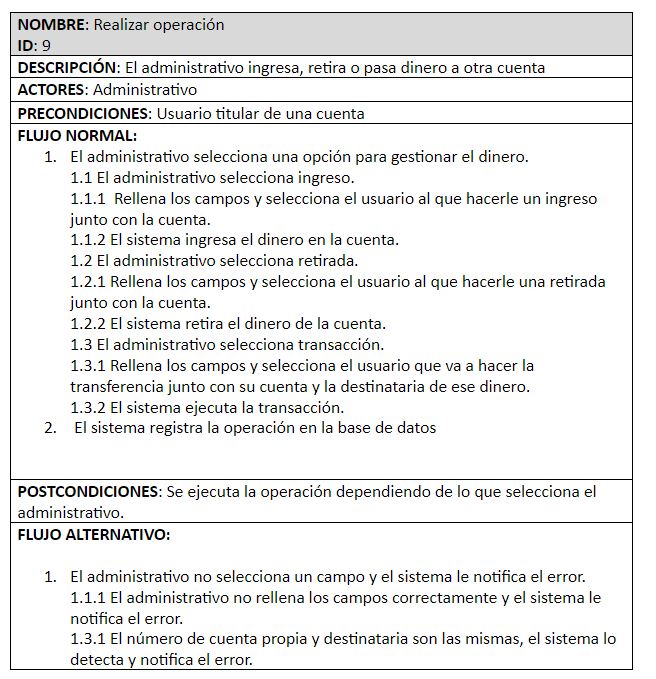












**1.3. OBJETIVOS**

La implementación de una plataforma que solvente las necesidades de gestión de los datos de la universidad, cubriendo las funcionalidades básicas de la capa de persistencia del software CRUD, a través de un administrador que cuente con todos los privilegios para que pueda consultar la información que les resulta relevante.

**1.4 MODELADO DE LA SOLUCIÓN**

1.4.1 RECURSOS HUMANOS

La plataforma requerirá mínimo de una persona, el administrador, que será además el encargado del mantenimiento de la base de datos.

1.4.2 RECURSOS HARDWARE

El único recurso de hardware necesario es el propio ordenador donde se ejecuta la aplicación y un servidor en el que se almacene la base de datos, en su defecto puede hacerse uso de una en local.

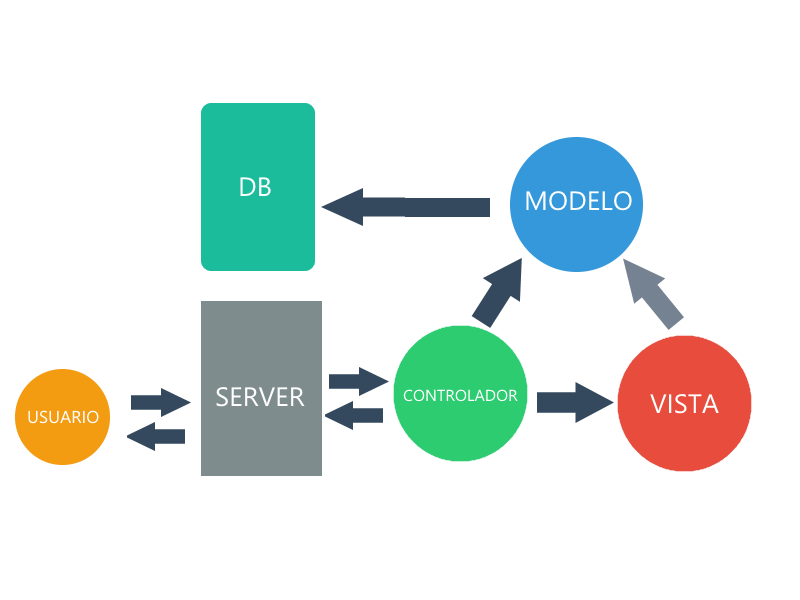
1.4.3 RECURSOS SOFTWARE

Para poder ejecutar la aplicación es necesario tener instalado en el ordenador un JRE, ya que la propia aplicación es un ejecutable de Java. Para la base de datos se requiere que de un sistema gestor basado en SQL, preferiblemente MYSQL.

***2. EJECUCIÓN DE LA PRÁCTICA***

2.1 DOCUMENTACIÓN TÉCNICA E IMPLEMENTACIÓN DE LA APLICACIÓN

Para la implementación de la aplicación hay que distinguir entre el patrón de diseño empleado para la propia aplicación en Java y el diseño normalizado de la base de datos



**Base de Datos:**

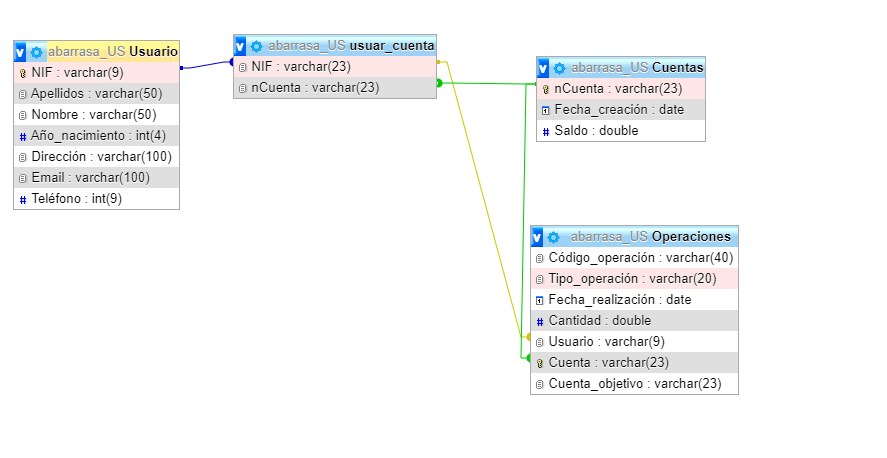
El gestor empleado es MYSQL de Oracle Corporation. Se ha optado por este sistema debido al requisito de emplear una base de datos relacional SQL. La base de datos ha sido implementada en un servidor externo.

Tras un estudio y análisis del problema se concluye en este esquema que supone la base del posterior modelo relacional que se traducirá a las tablas en SQL.

El núcleo del modelo está compuesto por las entidades Usuario, Cuentas y Operaciones relacionados entre sí, con la tabla Usuario\_cuentas para enlazar las cuentas y los usuarios

Este diagrama muestra el esquema que sigue la base de datos implementada en MYSQL en el servidor externo. Las relaciones [N:M] se han transformado en tablas propias.

La presente estructuración de las tablas junto a las claves primarias definidas en cada una responde al objetivo que cumpliera con los requerimientos de la tercera forma normal, que se ha considerado la más adecuada para lograr minimizar la redundancia de datos sin comprometer la coherencia del mismo con el problema real.



**Script SQL:**

**Definición de las tablas:**

CREATE TABLE Usuario (NIF VARCHAR(9) NOT NULL , Apellidos VARCHAR(50) NOT NULL , Nombre VARCHAR(50) NOT NULL , Año\_nacimiento INT(4) NOT NULL , Dirección VARCHAR(100) NOT NULL , Email VARCHAR(100) NOT NULL , Teléfono INT(9)NOT NULL, primary key(NIF)) ENGINE = InnoDB;

CREATE TABLE Cuenta ( nCuenta INT(10) NOT NULL , Fecha\_creación DATE NOT NULL , Saldo INT NOT NULL, primary key(nCuenta)) ENGINE = InnoDB;

CREATE TABLE Operacion (Codigo\_operacion VARCHAR(4) NOT NULL, Tipo\_operacion VARCHAR(20) NOT NULL, nCuenta INT(10) NOT NULL, Fecha\_realización DATE NOT NULL , Cantidad INT NOT NULL, Usuario VARCHAR(9) NOT NULL, primary key(Codigo\_operacion)) ENGINE = InnoDB;

**Control de la base de datos:**

Para velar por la integridad y la coherencia de los datos se han incluido una serie de rutinas, clasificados principalmente entre funciones y disparadores, que además se encargan de que no pueda haber registros que no respeten las condiciones del problema.

**Procedimientos:**

DELIMITER $$

CREATE DEFINER=`AdminUs`@`%` PROCEDURE `Agregar\_cuenta`(IN `vnCuenta` VARCHAR(23), IN `vFecha\_creacion` DATE, IN `vSaldo` DOUBLE)

NO SQL

BEGIN

INSERT INTO Cuentas VALUES (vnCuenta, vFecha\_creacion, vSaldo);

END$$

DELIMITER ;

DELIMITER $$

CREATE DEFINER=`AdminUs`@`%` PROCEDURE `Agregar\_titular`(IN `tNIF` VARCHAR(9), IN `tnCuenta` VARCHAR(23))

NO SQL

BEGIN

INSERT INTO usuar\_cuenta VALUES (tNIF, tnCuenta);

END$$

DELIMITER ;

DELIMITER $$

CREATE DEFINER=`AdminUs`@`%` PROCEDURE `Agregar\_usuario`(IN `vNIF` VARCHAR(9) CHARSET utf8, IN `vApellidos` VARCHAR(50) CHARSET utf8, IN `vNombre` VARCHAR(50) CHARSET utf8, IN `vAno\_nacimiento` INT(4), IN `vDireccion` VARCHAR(100) CHARSET utf8, IN `vEmail` VARCHAR(100) CHARSET utf8, IN `vTelefono` INT(9))

NO SQL

BEGIN

INSERT INTO Usuario VALUES (vNIF,vApellidos,vNombre,vAno\_nacimiento,vDireccion,vEmail,vTelefono);

END$$

DELIMITER ;

DELIMITER $$

CREATE DEFINER=`AdminUs`@`%` PROCEDURE `Ingreso`(IN `cuenta` VARCHAR(23), IN `suma` DOUBLE)

NO SQL

BEGIN

UPDATE Cuentas SET Saldo=Saldo+suma WHERE nCuenta LIKE cuenta;

END$$

DELIMITER ;

DELIMITER $$

CREATE DEFINER=`AdminUs`@`%` PROCEDURE `Modifica\_usuario`(IN `buscaNIF` VARCHAR(9), IN `mApellidos` VARCHAR(50), IN `mNombre` VARCHAR(50), IN `mAño` INT(4), IN `mDireccion` VARCHAR(100), IN `mEmail` VARCHAR(100), IN `mTelefono` INT(9))

NO SQL

BEGIN

UPDATE Usuario SET Apellidos = mApellidos, Nombre = mNombre, Año\_nacimiento = mAño, Dirección = mDireccion, Teléfono = mTelefono WHERE NIF = buscaNIF;

END$$

DELIMITER ;

DELIMITER $$

CREATE DEFINER=`AdminUs`@`%` PROCEDURE `Nueva\_operacion`(IN `cod` VARCHAR(40), IN `tipo` VARCHAR(20), IN `fecha` DATE, IN `dinero` FLOAT, IN `nif` VARCHAR(9), IN `cuenta` VARCHAR(23), IN `cuenta\_destino` VARCHAR(23))

NO SQL

BEGIN

INSERT INTO Operaciones VALUES(cod, tipo, fecha, dinero, nif, cuenta, cuenta\_destino);

END$$

DELIMITER ;

DELIMITER $$

CREATE DEFINER=`AdminUs`@`%` PROCEDURE `Quitar\_titular`(IN `qNIF` VARCHAR(9), IN `qnCuenta` VARCHAR(23))

NO SQL

BEGIN

DELETE FROM usuar\_cuenta where NIF=qNIF and nCuenta=qnCuenta;

END$$

DELIMITER ;

DELIMITER $$

CREATE DEFINER=`AdminUs`@`%` PROCEDURE `Retirada`(IN `cuenta` VARCHAR(23), IN `resta` FLOAT)

NO SQL

BEGIN

UPDATE Cuentas SET Saldo=Saldo-resta WHERE nCuenta=cuenta;

END$$

DELIMITER ;

DELIMITER $$

CREATE DEFINER=`AdminUs`@`%` PROCEDURE `Transaccion`(IN `propia` VARCHAR(23), IN `destino` VARCHAR(23), IN `cantidad` FLOAT)

NO SQL

BEGIN

UPDATE Cuentas SET Saldo=Saldo+cantidad WHERE nCuenta=destino;

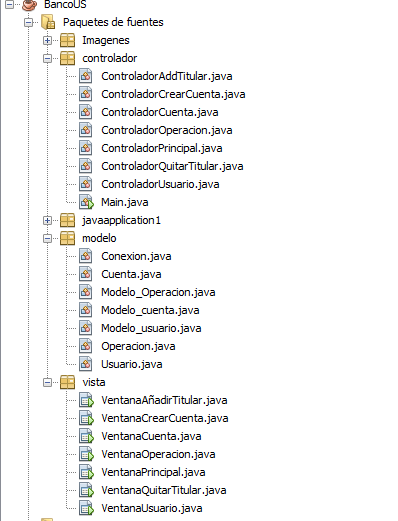
UPDATE Cuentas SET Saldo=Saldo-cantidad WHERE nCuenta=propia;

END$$

DELIMITER ;

**Aplicación:**

El proyecto se ha desarrollado siguiendo el patrón de diseño MVC o Modelo-Vista-Controlador



Este modelo divide la aplicación en tres partes: La vista: aquí se define la interfaz gráfica de la aplicación, todo el entorno visual con el que interactúan los usuarios

El modelo: esta parte se encarga de la conexión con la base de datos.

El controlador: Unión entre la vista y el modelo, en él se encuentran implementados todas las funcionalidades,

**3. DOCUMENTACIÓN DEL SISTEMA**

**3.1. MANUAL DE INSTALACIÓN**

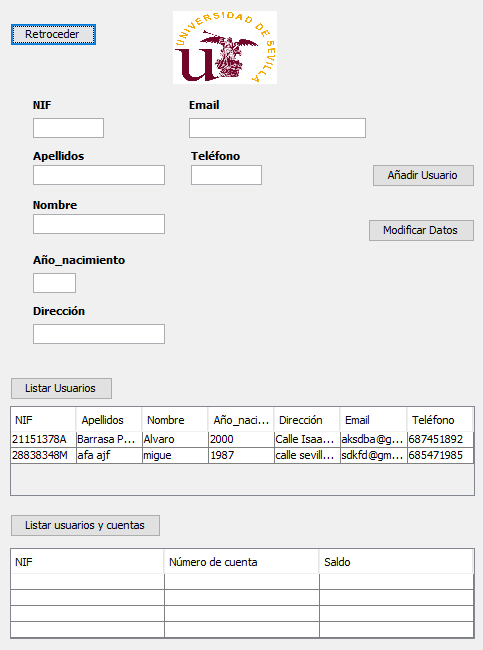
Para poder ejecutar la aplicación es necesario tener un equipo con un JRE instalado. Estos se pueden descargar a través de la página oficial de Oracle de forma gratuita: <http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html>

**3.2 MANUAL DE USUARIO**

ADMINISTRADOR:

Al iniciar la aplicación se abrirá la interfaz primaria donde mostrará tres botones que nos mandarán a sus respectivas interfaces.



En caso de seleccionar Gestionar Usuarios nos mostrará la siguiente interfaz: 

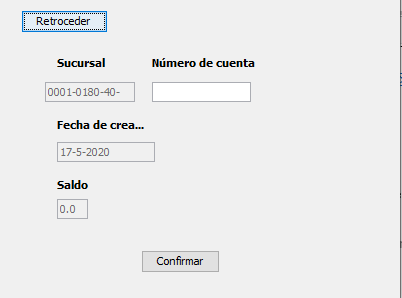
Donde podremos completar los campos del usuario para crearlo, estos se guardarán en la primera tabla para después poder modificarlos. Una vez creado un usuario podremos ver las cuentas que hay en total en la base de datos.

En caso de seleccionar Gestionar Cuentas nos mostrará la siguiente interfaz:

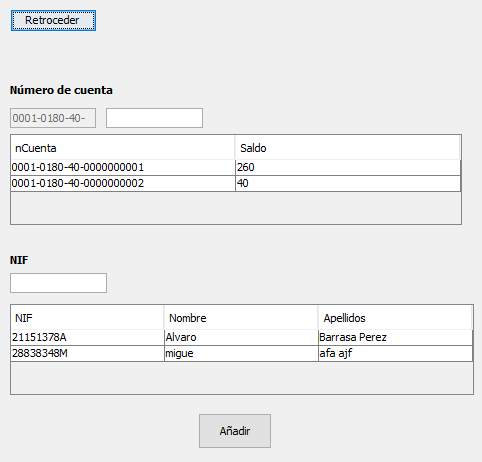


Donde podremos elegir entre crear una cuenta, añadir un titular a una cuenta o quitar un titular de una cuenta.

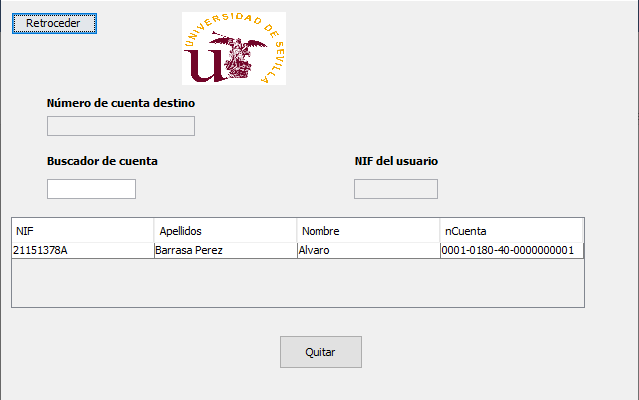
En caso de seleccionar Crear Cuenta nos redigirá a la siguiente interfaz:



Podremos crear una cuenta con un número de cuenta personalizado y único, y cuando le demos a confirmar nos enviará a la siguiente ventana:

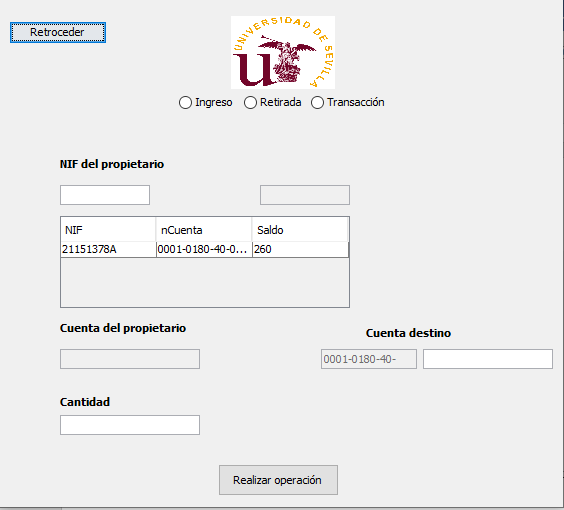


Se podrá seleccionar y filtrar para coger el DNI y el número de cuenta, siempre que esa relación no exista.

Si se elige quitar un titular aparecerá la siguiente interfaz:

Se podrá filtrar por el número de cuenta para borrar un titular de la cuenta siempre que esa cuenta tenga más de un titular.

En caso de seleccionar Realizar operación nos llevará a la siguiente ventana:



Se podrá seleccionar para hacer un ingreso, retirada o transacción (siempre que sea entre cuentas de la universidad). Se filtra por el DNI del usuario para coger su cuenta y realizar la acción deseada, y si se eligiese ingreso o retirada la casilla para rellenar la cuenta destino se deshabilitaría.

***4. CONCLUSIONES FINALES***

4.1. GRADO DE CUMPLIMIENTO DE LOS OBJETIVOS FIJADOS

El proyecto presentado es una aplicación que reúne los requisitos mínimos definidos por el problema y para ser funcional, pero puede mejorarse bastante en mi opinión.

La aplicación cubre los requerimientos básicos de cualquier CRUD, lee y muestra la información, permite insertar datos nuevos y manipular los ya existentes a través de una interfaz gráfica, se conecta a una base de datos externa MYSQL y ofrece además la posibilidad de generar ficheros con información relevante de los cursos y los empleados.

4.2 PROPUESTA DE MODIFICACIONES O AMPLIACIONES FUTURAS DEL SISTEMA IMPLEMENTADO

Como propuestas de mejora se la mejora de la visibilidad de la aplicación y la creación de tickets cada vez que se realice una operación en formato PDF.

***5. BIBLIOGRAFÍA***

• Apuntes y documentación de la asignatura de Programación

• Apuntes y documentación de la asignatura de Bases de Datos

• Apuntes y documentos de Entorno de Desarrollo

• Documentación oficial de MYSQL https://dev.mysql.com/doc/

• Manuales técnicos