Introducción

En este Proyecto se utilizó modelo MVC (Modelo, Vista, Controlador), es una forma de organización de código muy ordenada que es de fácil entendimiento. Este proyecto está orientado a una aplicación de Bancos que tiene diferentes funciones que se explicaran más afondo en el manual. Además es muy interactivo y de fácil entendimiento.

Requerimientos del Sistema

Lenguaje de Programación

Java (JDK 8 o superior)

Patrón de Diseño

Modelo-Vista-Controlador (MVC)

Librerías Utilizadas

javax.swing: Para la interfaz gráfica.

com.itextpdf: Para la generación de reportes en PDF.

Requisitos de Hardware

Procesador: 1 GHz o superior.

Memoria RAM: 2 GB o más.

Almacenamiento: 100 MB de espacio libre.

Requisitos de Software

Sistema Operativo: Windows, Linux o macOS.

Java Runtime Environment (JRE) 8 o superior.

Estructura del Proyecto

El sistema se organiza en los siguientes paquetes:

Controller (Controladores)

Contiene las clases que gestionan la lógica del negocio y la comunicación entre la vista y el modelo.

BController: Controlador principal que gestiona las vistas y acciones del menú.

BuscarController: Controlador para buscar cuentas asociadas a un cliente.

CrearCuentaController: Controlador para crear cuentas.

DepositosController: Controlador para realizar depósitos.

HistorialController: Controlador para mostrar el historial de transacciones.

LoginController: Controlador para el inicio de sesión.

RUsuarios: Controlador para registrar usuarios.

ReportesController: Controlador para generar reportes.

RetirosController: Controlador para realizar retiros.

Model (Modelos de Datos)

Contiene las clases que representan los datos del sistema.

Bitacora: Clase que registra las acciones en el sistema.

Clientes: Clase que representa a los clientes del banco.

Cuentas: Clase que representa las cuentas bancarias.

Reporte: Clase que representa un reporte.

Transaccion: Clase que representa una transacción (depósito o retiro).

View (Interfaz Gráfica)

Contiene las interfaces gráficas del sistema.

Bienvenida: Vista principal del sistema.

BitacoraView: Vista de la bitácora (no implementada completamente).

BuscarCuentas: Vista para buscar cuentas asociadas a un cliente.

CrearCuenta: Vista para crear cuentas.

Datos: Vista que muestra los datos del estudiante.

Depositos: Vista para realizar depósitos.

Historial: Vista para mostrar el historial de transacciones.

LoginView: Vista de inicio de sesión.

Reportes: Vista para generar reportes.

Retiros: Vista para realizar retiros.

Usuario: Vista para registrar usuarios.

Util (Utilidades)

Contiene herramientas auxiliares.

ReportePDF: Clase que genera reportes en formato PDF.

Funciones y Métodos Principales

Controladores

BController:

mostrarRegistroUsuario(): Muestra la vista de registro de usuarios.

mostrarCrearCuenta(): Muestra la vista de creación de cuentas.

mostrarRetiros(): Muestra la vista de retiros.

mostrarDepositos(): Muestra la vista de depósitos.

mostrarBuscarCuentas(): Muestra la vista de búsqueda de cuentas.

mostrarHistorialTransacciones(): Muestra la vista del historial de transacciones.

mostrarGenerarReportes(): Muestra la vista de generación de reportes.

mostrarDatosEstudiante(): Muestra los datos del estudiante.

BuscarController:

buscarCuentasAsociadas(String cuiCliente): Busca las cuentas asociadas a un cliente por su CUI.

CrearCuentaController:

crearCuenta(String cuiCliente, String idCuenta): Crea una nueva cuenta para un cliente.

DepositosController:

realizarDeposito(String idCuenta, double monto): Realiza un depósito en una cuenta.

HistorialController:

obtenerTransacciones(String idCuenta): Obtiene las transacciones de una cuenta.

LoginController:

iniciarSesion(String usuario, String password): Valida las credenciales de inicio de sesión.

RUsuarios

registrarUsuario(String cui, String nombre, String apellido): Registra un nuevo usuario.

ReportesController

generarReporteDepositos(String cuiCliente): Genera un reporte de depósitos para un cliente.

generarReporteRetiros(String cuiCliente): Genera un reporte de retiros para un cliente.

RetirosController

realizarRetiro(String idCuenta, double monto): Realiza un retiro de una cuenta.

Modelos

Bitacora:

registrar(String usuario, String accion, String resultado, String detalles): Registra una acción en la bitácora.

Clientes:

agregarCuenta(Cuentas cuenta): Agrega una cuenta a un cliente.

Cuentas:

depositar(double monto): Realiza un depósito en la cuenta.

retirar(double monto): Realiza un retiro de la cuenta.

ReportePDF:

generarReporte(Reporte reporte): Genera un reporte en formato PDF.

Conclusión

El sistema bancario implementado sigue el patrón MVC, lo que permite una separación clara entre la lógica de negocio, la interfaz gráfica y los datos. La implementación de una bitácora y la generación de reportes en PDF aseguran un seguimiento detallado de las acciones realizadas en el sistema, mejorando la trazabilidad y la eficiencia en la gestión bancaria. Y use todos los conocimientos en el laboratorio y de la clase magistral e investigación individual para la realización satisfactoria de este proyecto.