

**Diplomado en Tecnologías de Información**

**Curso:** Programación IV TI-151

Sistema de Tarjetas

Alcance 1

Integrantes:

Nestor Leiva Villalobos

Rodrigo Rene Elías Ramírez

Profesora:

Milena Mata Sojo

Cartago, 11 febrero 2026

Tabla de contenido

[Introducción 2](#_Toc1251869078)

[Problema por Resolver 2](#_Toc1494350937)

[Diagrama de Base de Datos 3](#_Toc916707468)

[Diagrama de Casos de Uso 4](#_Toc1842283627)

[Diagrama de Clases 5](#_Toc2046381173)

[Capitulo 2: Implementación de Base de Datos 7](#_Toc437062751)

[Script MySQL 7](#_Toc1124141118)

[Script SQLServer 8](#_Toc63470626)

[Capitulo 3: Autorizador de Transacciones 11](#_Toc447514461)

[AUT 1 & 2 11](#_Toc1701198534)

[AUT 3 14](#_Toc56592001)

[AUT 4 16](#_Toc398593764)

[Capitulo 4: Componentes del Core Bancario 18](#_Toc1464465763)

[Capitulo 5: Simulador de Transacciones 22](#_Toc930847067)

[Conclusiones 26](#_Toc94142667)

[LOGROS ALCANZADOS 27](#_Toc1796875134)

[Recomendaciones 28](#_Toc2128305317)

[A. Arquitectura y Escalabilidad 29](#_Toc343985635)

[B. Seguridad y Cumplimiento 29](#_Toc939403456)

[C. Monitoreo y Observabilidad 29](#_Toc1611299200)

[D. Calidad del Software 29](#_Toc2087187731)

[E. Base de Datos 29](#_Toc546720517)

[COMPETENCIAS CONSOLIDADAS 30](#_Toc1534801223)

[Bibliografía 30](#_Toc1982335044)

# Introducción

## Problema por Resolver

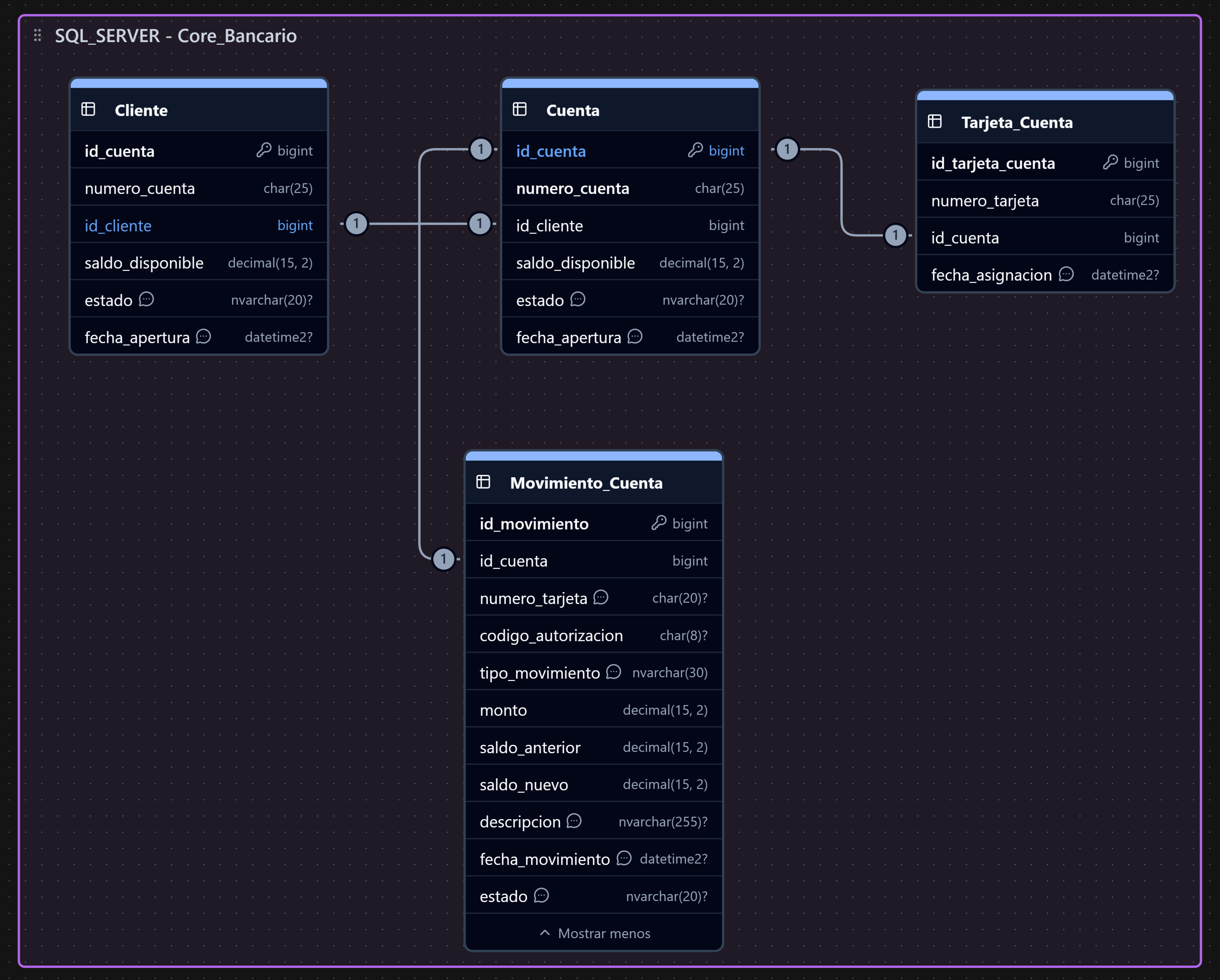
En la actualidad, el sector bancario enfrenta el desafío constante de ofrecer servicios más rápidos, seguros y eficientes para satisfacer las demandas de sus clientes. El Banco ABC ha identificado la necesidad de modernizar su sistema de procesamiento de transacciones con tarjetas de débito y crédito, debido a que su infraestructura actual presenta limitaciones en velocidad, integración y escalabilidad. Estas deficiencias dificultan la administración oportuna de los movimientos financieros y pueden afectar la experiencia del usuario final, especialmente ante el aumento del volumen de operaciones electrónicas.

El nuevo sistema busca optimizar la gestión de las transacciones, garantizando que cada operación —ya sea retiro, depósito, consulta o transferencia— se procese de forma segura y confiable. Además, el proyecto pretende mejorar la interconexión entre los distintos módulos del sistema, asegurando la integridad de los datos y una comunicación efectiva entre las plataformas internas del banco y las redes externas de pago. Con esta renovación tecnológica, el Banco ABC podrá ofrecer un servicio más competitivo, minimizando tiempos de espera y reduciendo los riesgos de errores operativos o fraudes.

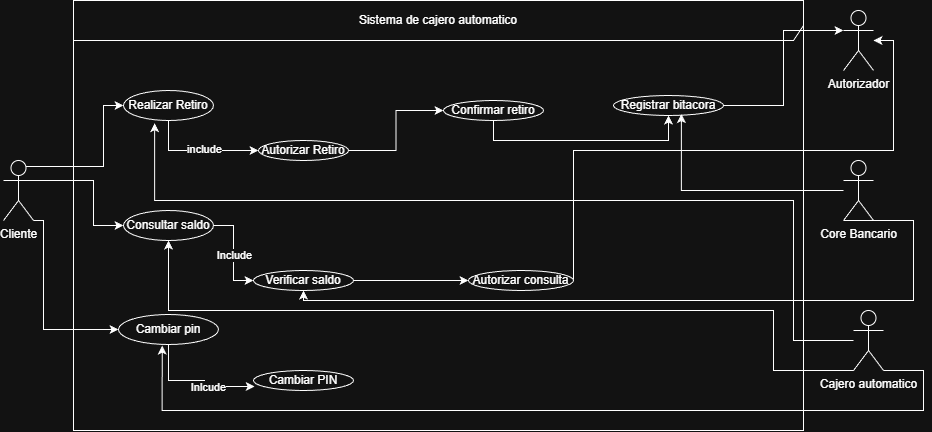
Este documento presenta la propuesta de diseño y desarrollo del nuevo sistema, detallando sus componentes, requerimientos y beneficios esperados. De esta manera, se busca sentar las bases para una infraestructura moderna que respalde el crecimiento futuro del banco y mejore la atención a sus clientes.

## Diagrama de Base de Datos

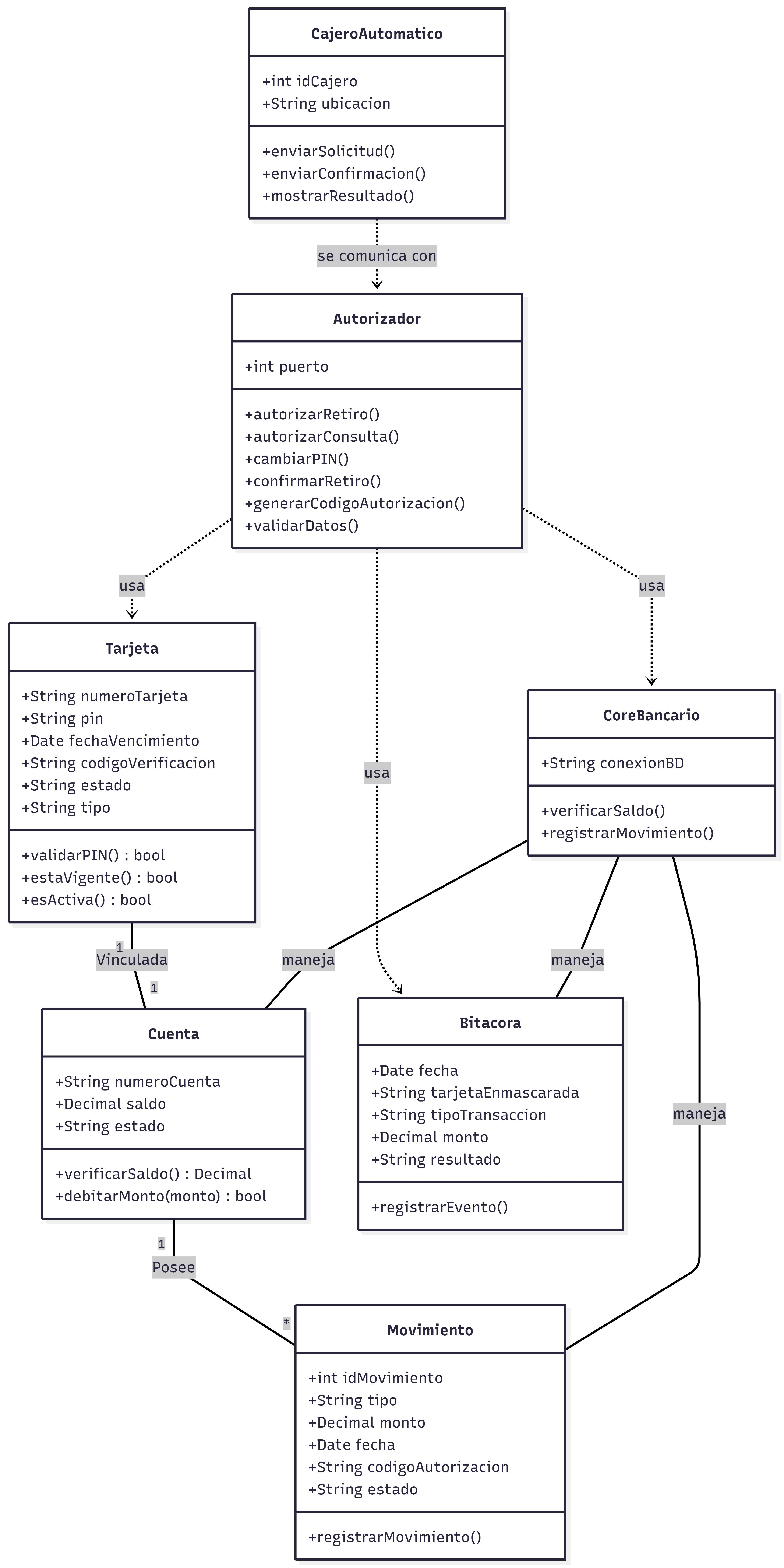




## Diagrama de Casos de Uso



## Diagrama de Clases



# Capitulo 2: Implementación de Base de Datos

## Script MySQL

-- ============================================================

-- Script De Sistema de Tarjetas de Credito / Debito

-- Proyecto Programacion IV - Colegio Universitario de Cartago

-- ============================================================

CREATE DATABASE IF NOT EXISTS sistema\_tarjetas CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci;

USE sistema\_tarjetas;

-- Tabla de cajeros automáticos

CREATE TABLE cajero ( id\_cajero BIGINT PRIMARY KEY AUTO\_INCREMENT, codigo\_cajero VARCHAR(20) UNIQUE NOT NULL, ubicacion VARCHAR(100), estado VARCHAR(20) DEFAULT 'ACTIVO' CHECK (estado IN ('ACTIVO', 'INACTIVO')) );

-- Tabla de códigos de motivo de rechazo (1-5)

CREATE TABLE codigo\_motivo ( id\_codigo\_motivo INT PRIMARY KEY AUTO\_INCREMENT, codigo INT UNIQUE NOT NULL CHECK (codigo BETWEEN 1 AND 5), descripcion VARCHAR(100) NOT NULL );

-- Tabla de tipos de transacción

CREATE TABLE tipo\_transaccion ( id\_tipo\_transaccion INT PRIMARY KEY AUTO\_INCREMENT, codigo\_tipo VARCHAR(30) UNIQUE NOT NULL, descripcion VARCHAR(100) NOT NULL );

-- Tabla principal de tarjetas

CREATE TABLE tarjeta ( id\_tarjeta BIGINT PRIMARY KEY AUTO\_INCREMENT, numero\_tarjeta VARCHAR(25) NOT NULL, -- Simplificación temporal pin VARCHAR(10) NOT NULL, -- Simplificación temporal fecha\_vencimiento DATE NOT NULL, cvv VARCHAR(5) NOT NULL, -- Simplificación temporal tipo\_tarjeta VARCHAR(10) NOT NULL CHECK (tipo\_tarjeta IN ('DEBITO', 'CREDITO')), estado VARCHAR(20) DEFAULT 'ACTIVA' CHECK (estado IN ('ACTIVA', 'INACTIVA', 'VENCIDA')), numero\_cuenta CHAR(25), identificacion\_cliente VARCHAR(20), limite\_adelanto\_efectivo DECIMAL(15,2) DEFAULT 0, disponible\_adelanto\_efectivo DECIMAL(15,2) DEFAULT 0, fecha\_actualizacion TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP );

-- Tabla de autorizaciones

CREATE TABLE autorizacion ( id\_autorizacion BIGINT PRIMARY KEY AUTO\_INCREMENT, codigo\_autorizacion CHAR(8) NOT NULL, id\_tarjeta BIGINT NOT NULL, id\_cajero BIGINT NOT NULL, id\_tipo\_transaccion INT NOT NULL, monto DECIMAL(15,2), estado VARCHAR(20) DEFAULT 'PENDIENTE' CHECK (estado IN ('PENDIENTE', 'APROBADA', 'RECHAZADA', 'CONFIRMADA')), fecha\_solicitud DATETIME DEFAULT CURRENT\_TIMESTAMP, fecha\_respuesta DATETIME NULL, respuesta VARCHAR(10), id\_codigo\_motivo INT NULL, CONSTRAINT fk\_autorizacion\_tarjeta FOREIGN KEY (id\_tarjeta) REFERENCES tarjeta(id\_tarjeta), CONSTRAINT fk\_autorizacion\_cajero FOREIGN KEY (id\_cajero) REFERENCES cajero(id\_cajero), CONSTRAINT fk\_autorizacion\_tipo FOREIGN KEY (id\_tipo\_transaccion) REFERENCES tipo\_transaccion(id\_tipo\_transaccion), CONSTRAINT fk\_autorizacion\_motivo FOREIGN KEY (id\_codigo\_motivo) REFERENCES codigo\_motivo(id\_codigo\_motivo) );

-- Tabla de movimientos de crédito

CREATE TABLE movimiento\_credito ( id\_movimiento BIGINT PRIMARY KEY AUTO\_INCREMENT, id\_tarjeta BIGINT NOT NULL, codigo\_autorizacion CHAR(8) NOT NULL, tipo\_movimiento VARCHAR(50) NOT NULL, monto DECIMAL(15,2) NOT NULL, estado VARCHAR(20) DEFAULT 'PENDIENTE' CHECK (estado IN ('PENDIENTE', 'APROBADO', 'RECHAZADO')), fecha\_movimiento DATETIME DEFAULT CURRENT\_TIMESTAMP, CONSTRAINT fk\_movimiento\_credito\_tarjeta FOREIGN KEY (id\_tarjeta) REFERENCES tarjeta(id\_tarjeta) );

-- Índices para mejorar el rendimiento

CREATE INDEX

idx\_tarjeta\_numero ON tarjeta(numero\_tarjeta); CREATE INDEX idx\_tarjeta\_pin ON tarjeta(pin); CREATE INDEX idx\_autorizacion\_codigo ON autorizacion(codigo\_autorizacion); CREATE INDEX idx\_autorizacion\_fecha ON autorizacion(fecha\_solicitud); CREATE INDEX idx\_cajero\_codigo ON cajero(codigo\_cajero);

## Script SQLServer

-- =====================================================

-- CORE BANCARIO - SQL SERVER

-- =====================================================

USE master; GO

-- creo la Base de Datos si No Existe

IF NOT EXISTS ( SELECT name FROM SYS.DATABASES WHERE NAME = 'core\_bancario') BEGIN CREATE DATABASE core\_bancario; PRINT '+ BASE DE DATOS CORE\_BANCARIO CREADA'; END ELSE PRINT ' ++ BASE DE DATOS YA EXISTE'; GO

USE core\_bancario; GO

-- =====================================================

-- TABLAS

-- =====================================================

-- TABLA CLIENTE

IF NOT EXISTS (SELECT \* FROM SYSOBJECTS WHERE name = 'cliente' AND xtype = 'U') BEGIN CREATE TABLE cliente ( id\_cliente BIGINT PRIMARY KEY IDENTITY(1,1), nombre NVARCHAR (100) NOT NULL, identificacion CHAR (20) UNIQUE NOT NULL, estado NVARCHAR(10) DEFAULT 'ACTIVO' CHECK (estado IN ('ACTIVO','INACTIVO')), fecha\_registro DATETIME2 DEFAULT SYSDATETIME() ); PRINT '+ TABLA CLIENTE CREADA'; END GO

-- TABLA CUENTA

IF NOT EXISTS (SELECT \* FROM SYSOBJECTS WHERE name = 'cuenta' AND xtype = 'U') BEGIN CREATE TABLE cuenta ( id\_cuenta BIGINT PRIMARY KEY IDENTITY (1,1), numero\_cuenta CHAR(25) UNIQUE NOT NULL, id\_cliente BIGINT NOT NULL, saldo\_disponible DECIMAL(15,2) DEFAULT 0 NOT NULL, estado NVARCHAR(10) DEFAULT 'ACTIVA' CHECK (estado IN ('ACTIVA', 'INACTIVA', 'BLOQUEADA')) NOT NULL, fecha\_apertura DATETIME2 DEFAULT SYSDATETIME() ); PRINT '+ TABLA CUENTA CREADA'; END GO

-- TABLA TARJETA\_CUENTA

IF NOT EXISTS (SELECT \* FROM SYSOBJECTS WHERE name = 'tarjeta\_cuenta' AND xtype = 'U') BEGIN CREATE TABLE tarjeta\_cuenta ( id\_tarjeta\_cuenta BIGINT PRIMARY KEY IDENTITY (1,1), numero\_tarjeta CHAR(25) NOT NULL, id\_cuenta BIGINT NOT NULL, fecha\_asignacion DATETIME2 DEFAULT SYSDATETIME(), estado NVARCHAR(10) DEFAULT 'ACTIVA' CHECK (estado IN ('ACTIVA', 'INACTIVA', 'VENCIDA')) ); PRINT '+ TABLA TARJETA\_CUENTA CREADA'; END GO

-- TABLA MOVIMIENTO\_CUENTA

IF NOT EXISTS (SELECT \* FROM SYSOBJECTS WHERE name = 'movimiento\_tarjeta' AND xtype = 'U') BEGIN CREATE TABLE movimiento\_tarjeta ( id\_movimiento BIGINT PRIMARY KEY IDENTITY(1,1), id\_cuenta BIGINT NOT NULL, numero\_tarjeta CHAR(20), codigo\_autorizacion CHAR(8), tipo\_movimiento NVARCHAR(30) NOT NULL, monto DECIMAL(15,2) NOT NULL, saldo\_anterior DECIMAL(15,2) NOT NULL, saldo\_nuevo DECIMAL(15,2) NOT NULL, descripcion NVARCHAR(255), fecha\_movimiento DATETIME2 DEFAULT SYSDATETIME(), estado NVARCHAR(20) DEFAULT 'PROCESADO' CHECK (estado IN ('PENDIENTE','PROCESADO','RECHAZADO')) ); PRINT '+ TABLA MOVIMIENTO\_TARJETA CREADA'; END GO

-- =====================================================

-- TABLAS

-- =====================================================

USE core\_bancario;

-- FK CUENTA -> CLIENTE IF NOT EXISTS (SELECT \* FROM sys.foreign\_keys WHERE name = 'FK\_cuenta\_cliente') BEGIN ALTER TABLE cuenta ADD CONSTRAINT FK\_cuenta\_cliente FOREIGN KEY (id\_cliente) REFERENCES cliente(id\_cliente); PRINT '+ FK CUENTA\_CLIENTE CREADA'; END GO

-- KF TARJETA -> CUENTA IF NOT EXISTS (SELECT \* FROM sys.foreign\_keys WHERE name = 'FK\_tarjeta\_cuenta') BEGIN ALTER TABLE tarjeta\_cuenta ADD CONSTRAINT FK\_tarjeta\_cuenta FOREIGN KEY (id\_cuenta) REFERENCES cuenta(id\_cuenta); PRINT '+ FK TARJETA\_CUENTA CREADA'; END GO

-- FK MOVIMIENTO\_TARJETA -> CUENTA IF NOT EXISTS (SELECT \* FROM sys.foreign\_keys WHERE name = 'FK\_movimiento\_cuenta') BEGIN ALTER TABLE movimiento\_tarjeta ADD CONSTRAINT FK\_movimiento\_cuenta FOREIGN KEY (id\_cuenta) REFERENCES cuenta(id\_cuenta); PRINT '+ FK MOVIMIENTO\_CUENTA CREADA'; END GO

-- =====================================================

-- INDICES

-- =====================================================

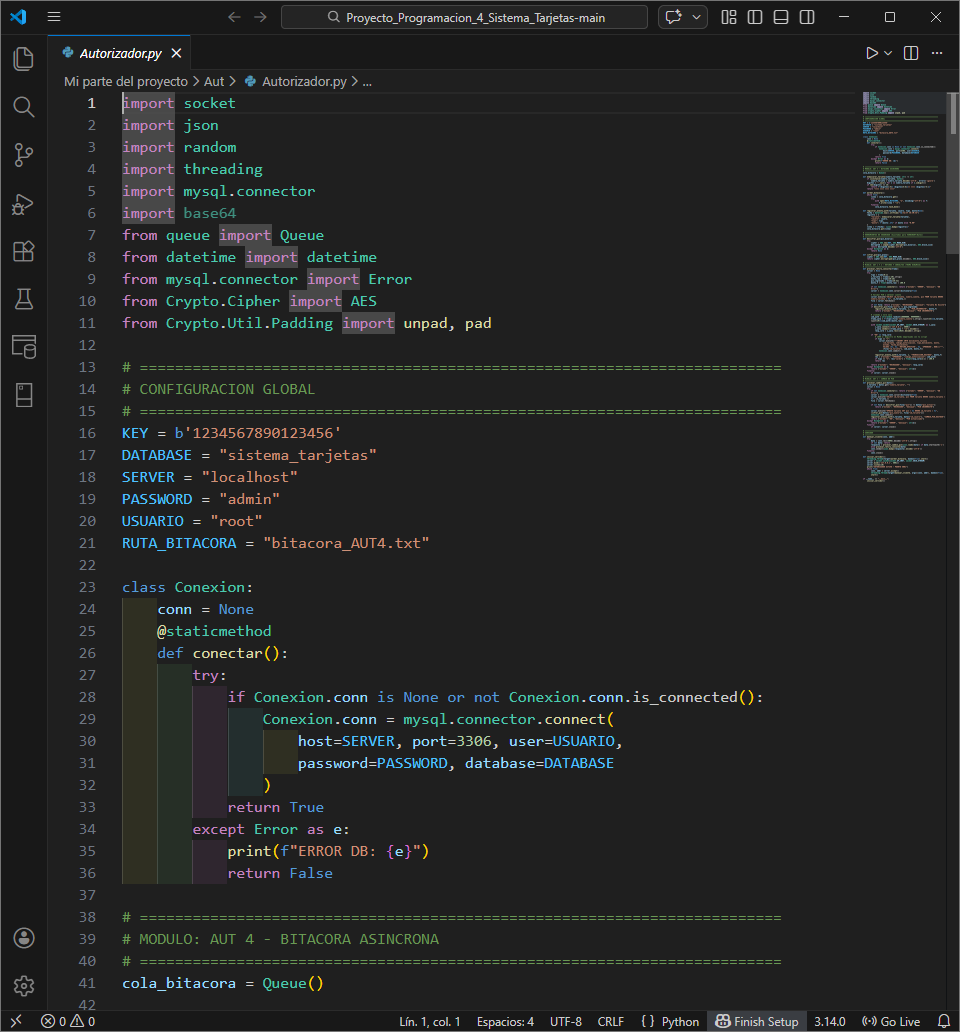
IF NOT EXISTS (SELECT \* FROM sys.indexes WHERE name = 'IX\_cuenta\_numero') CREATE INDEX IX\_cuenta\_numero ON cuenta(numero\_cuenta);

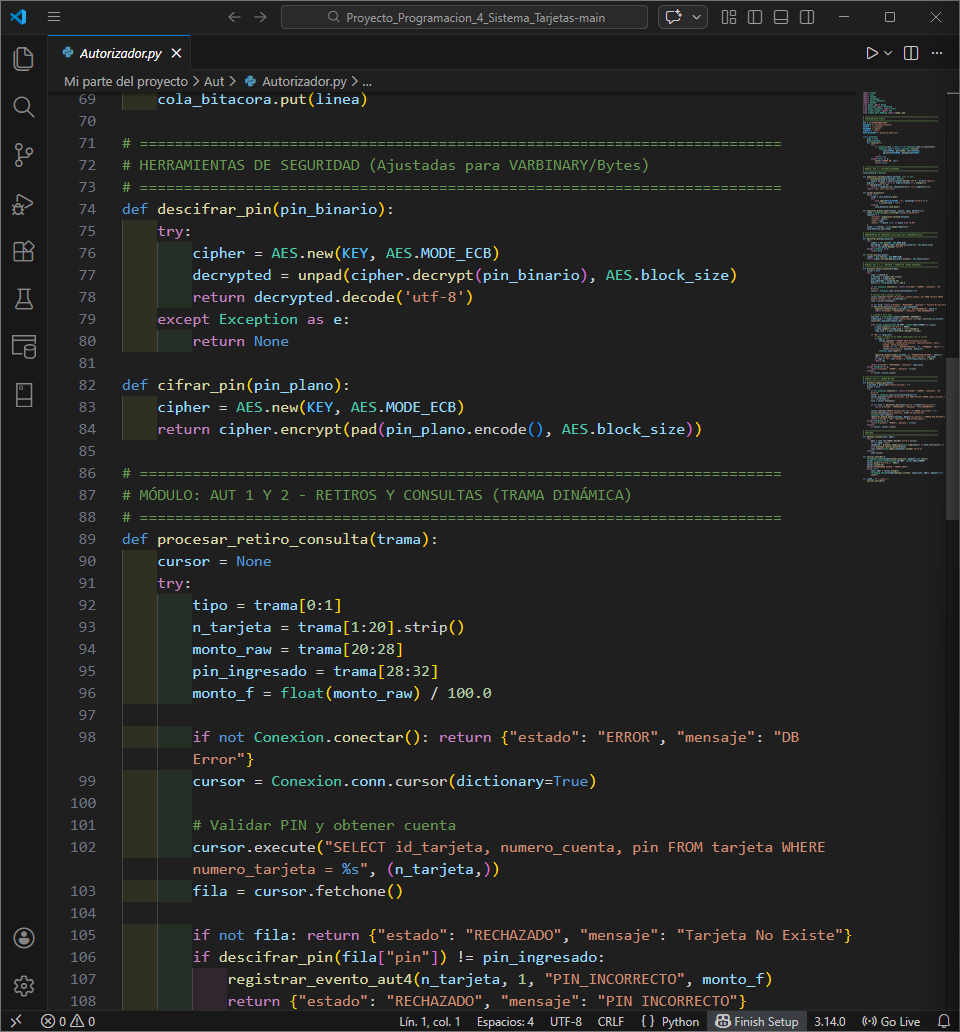
IF NOT EXISTS (SELECT \* FROM sys.indexes WHERE name = 'IX\_tarjeta\_numero') CREATE INDEX IX\_tarjeta\_numero ON tarjeta\_cuenta(numero\_tarjeta);

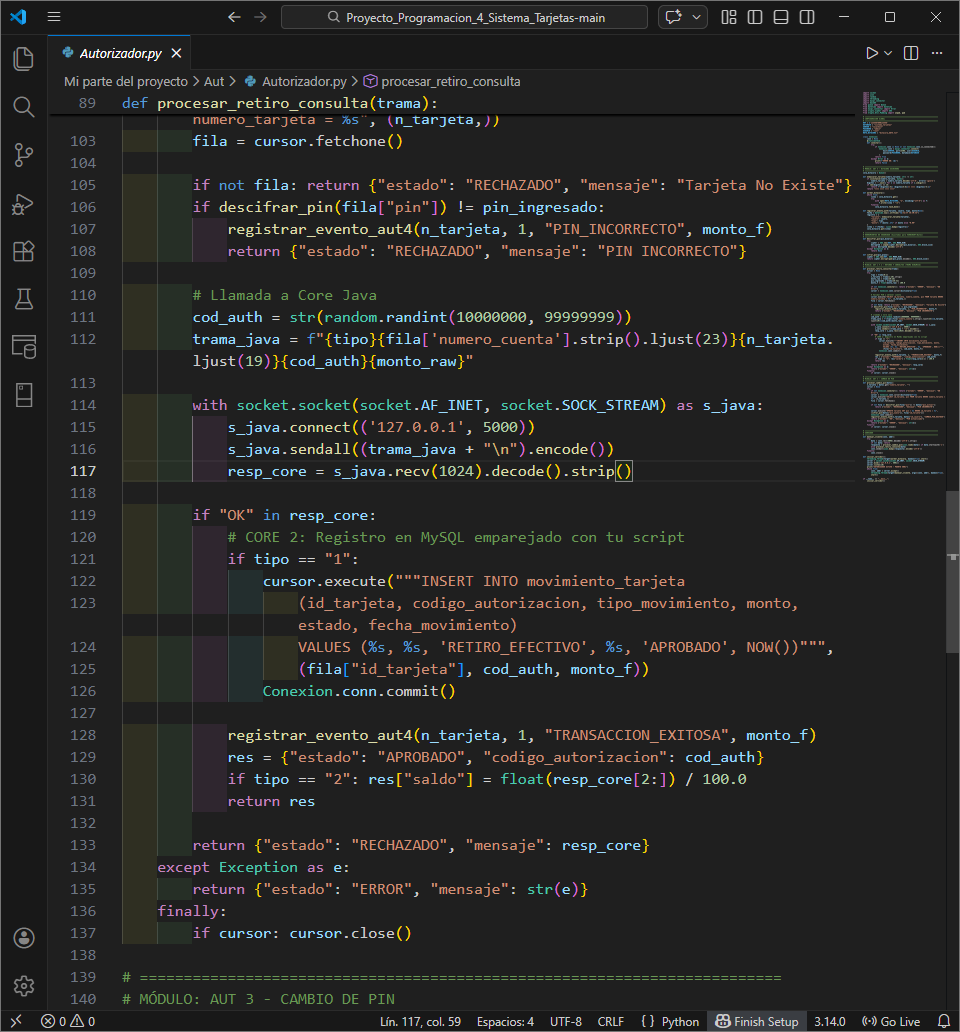
IF NOT EXISTS (SELECT \* FROM sys.indexes WHERE name = 'IX\_movimiento\_fecha') CREATE INDEX IX\_movimiento\_fecha ON movimiento\_tarjeta(fecha\_movimiento); GO

# Capitulo 3: Autorizador de Transacciones

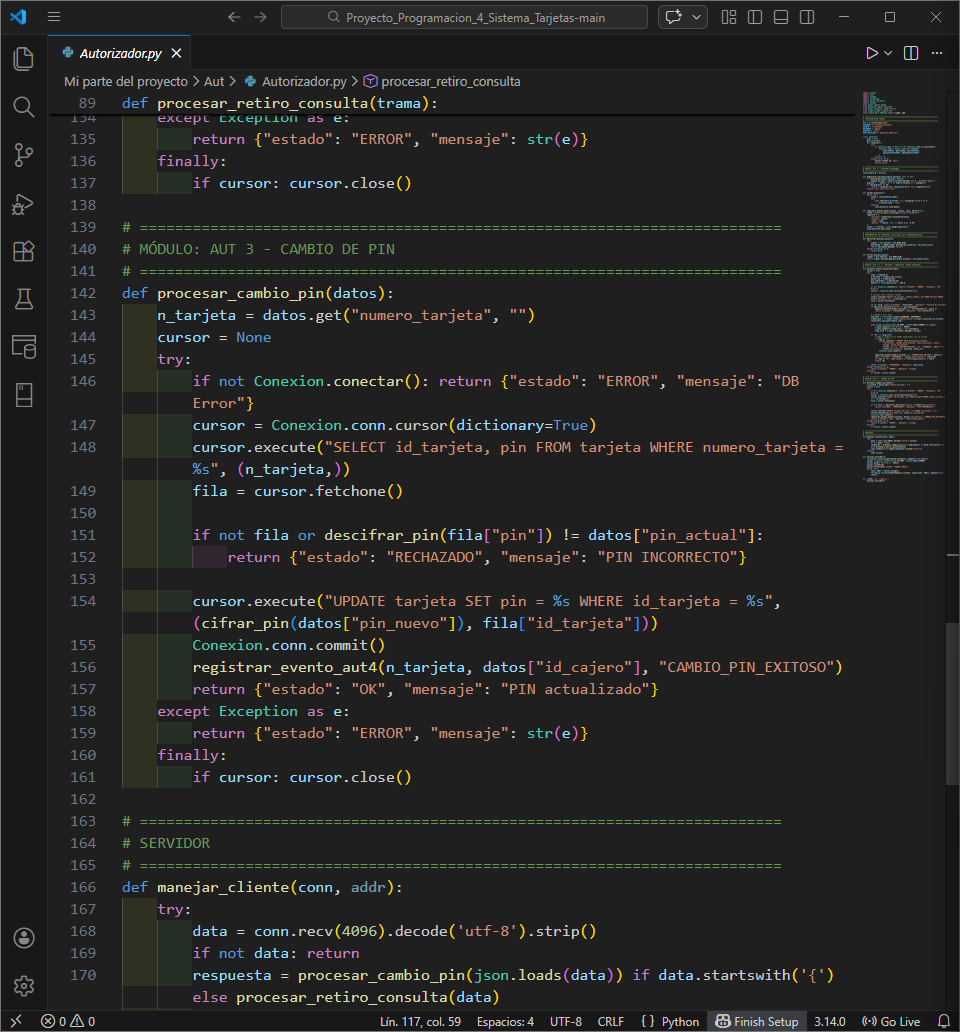
## AUT 1 & 2

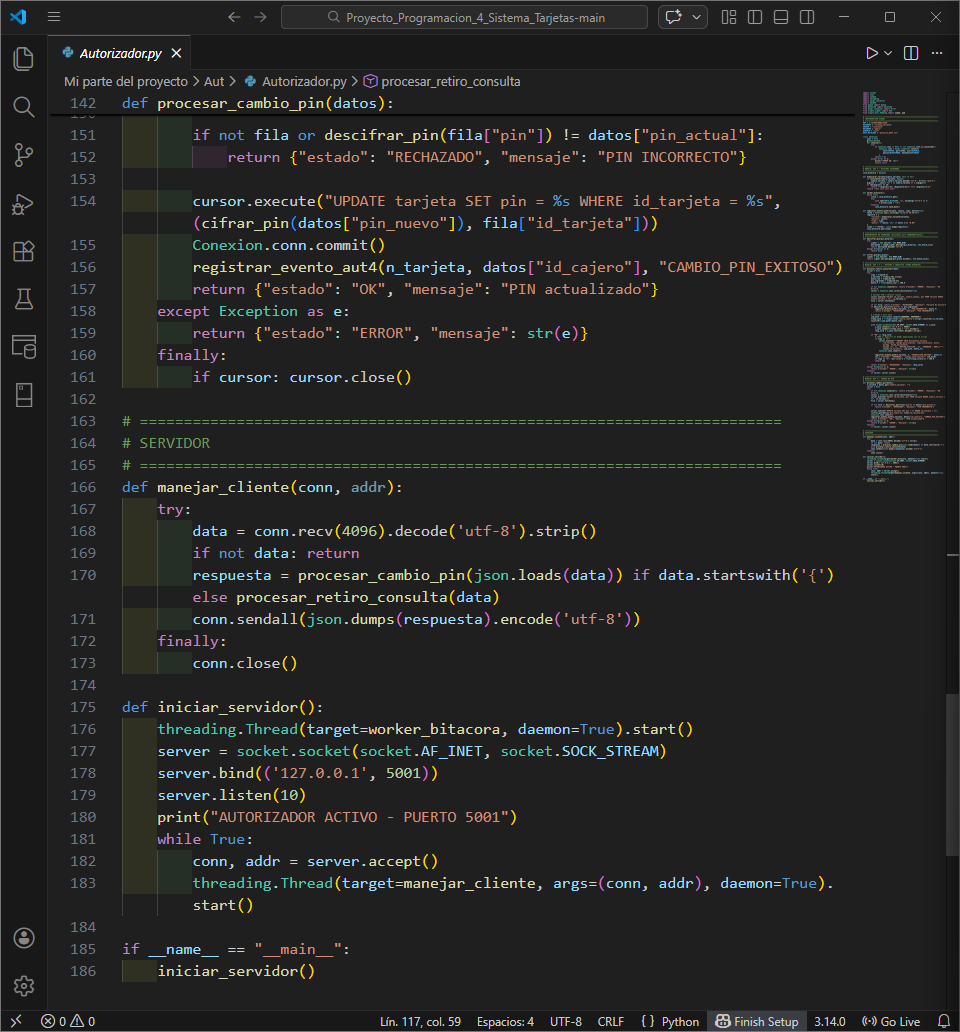




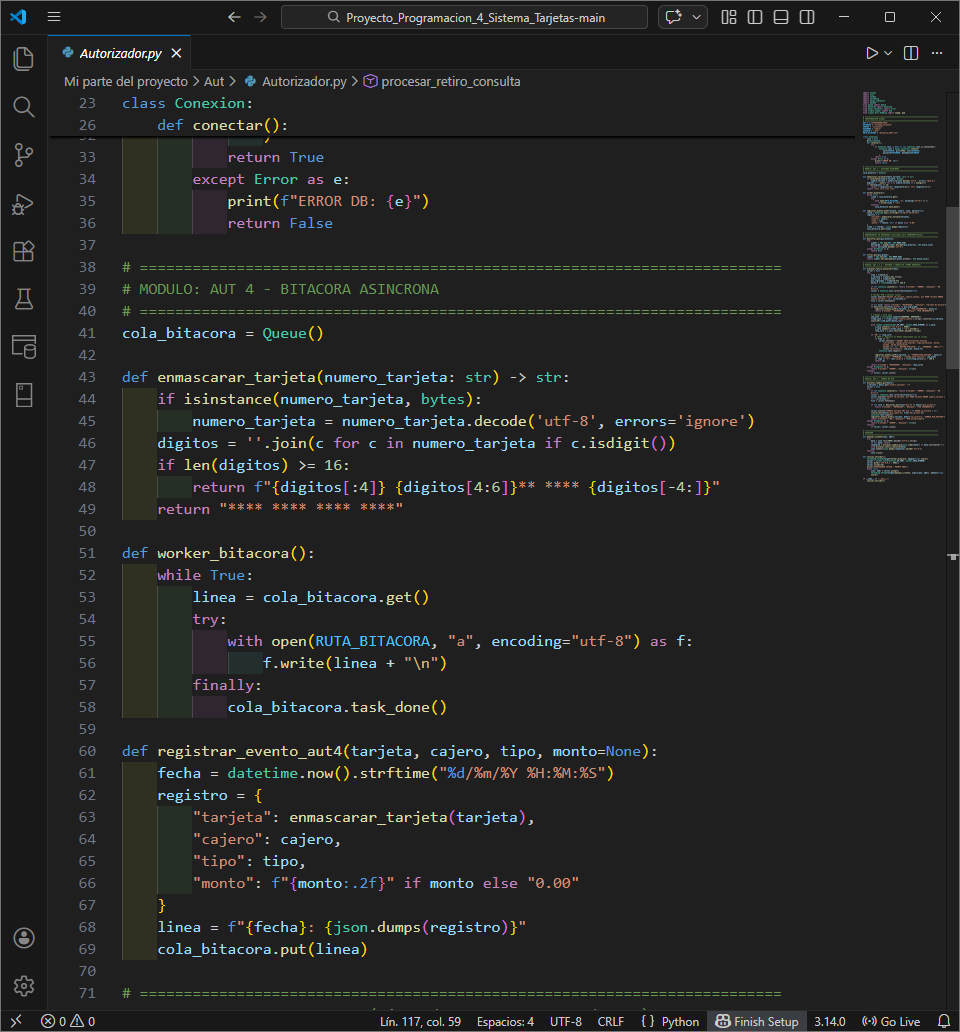


## AUT 3

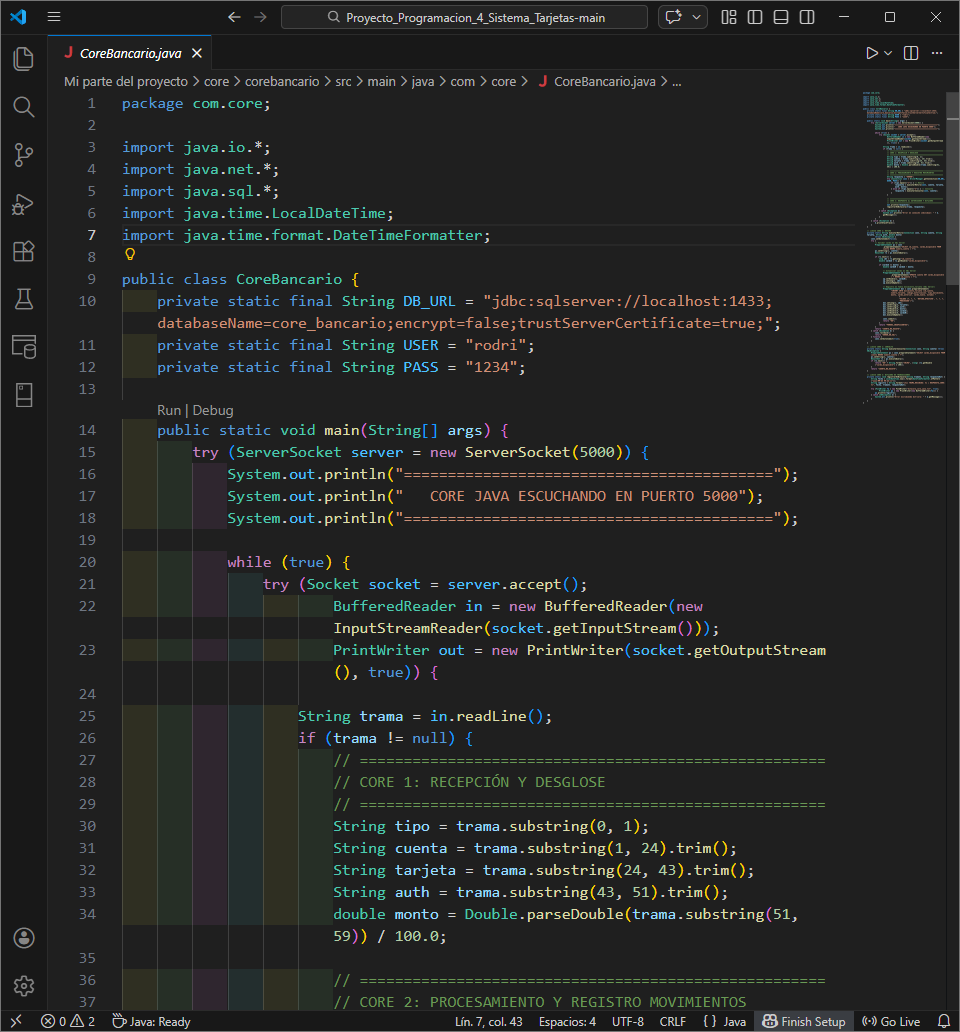


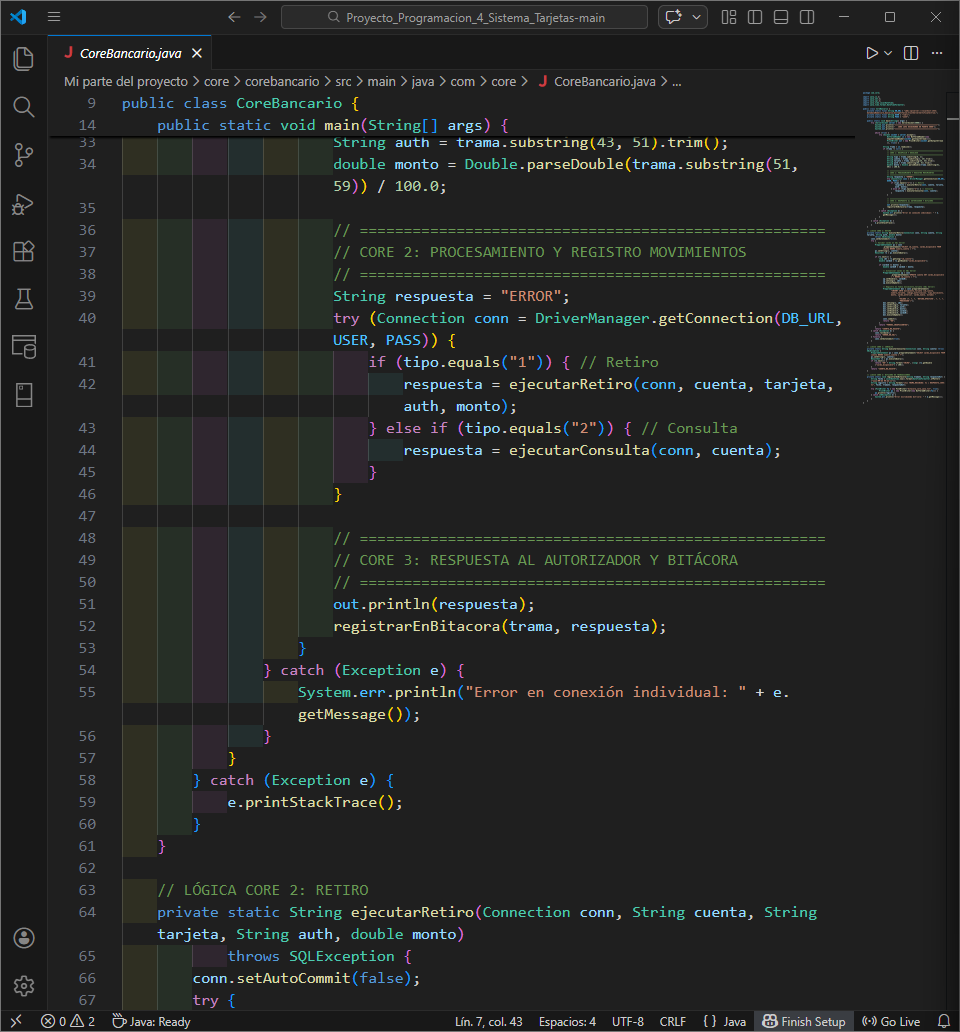


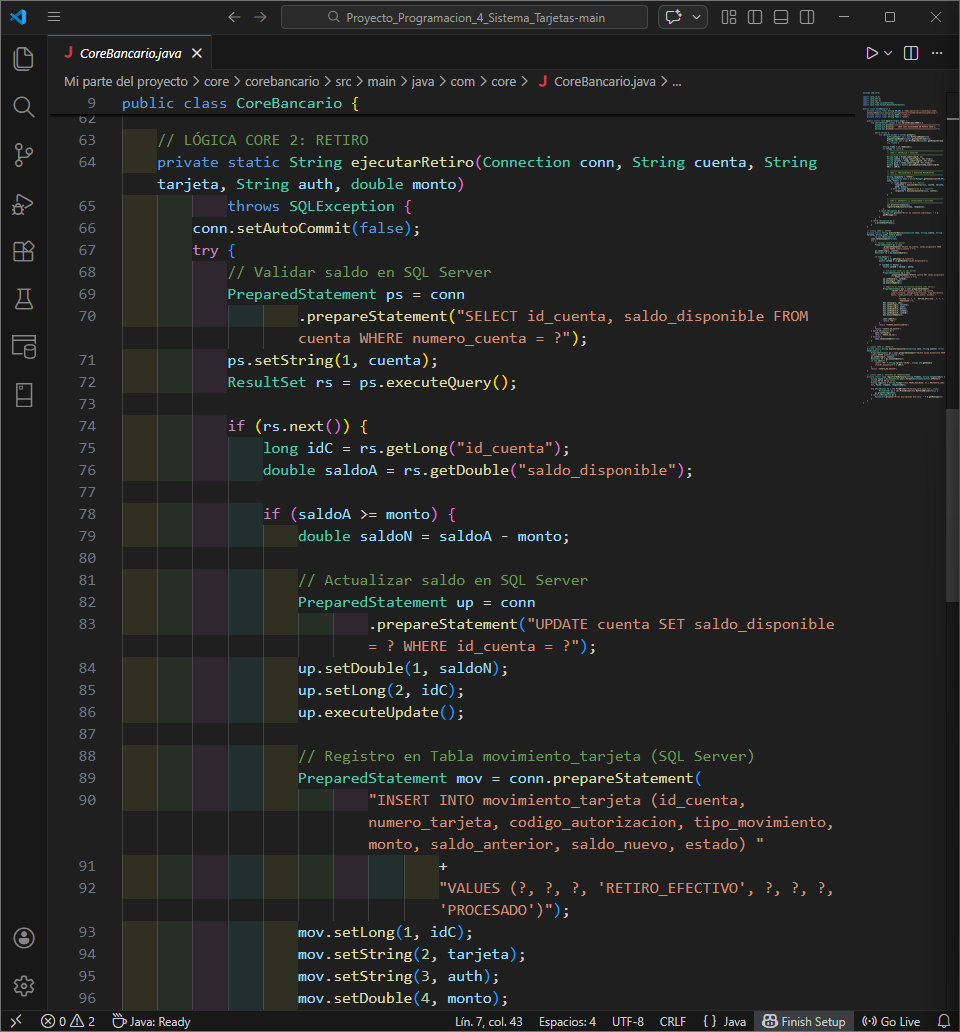
## AUT 4

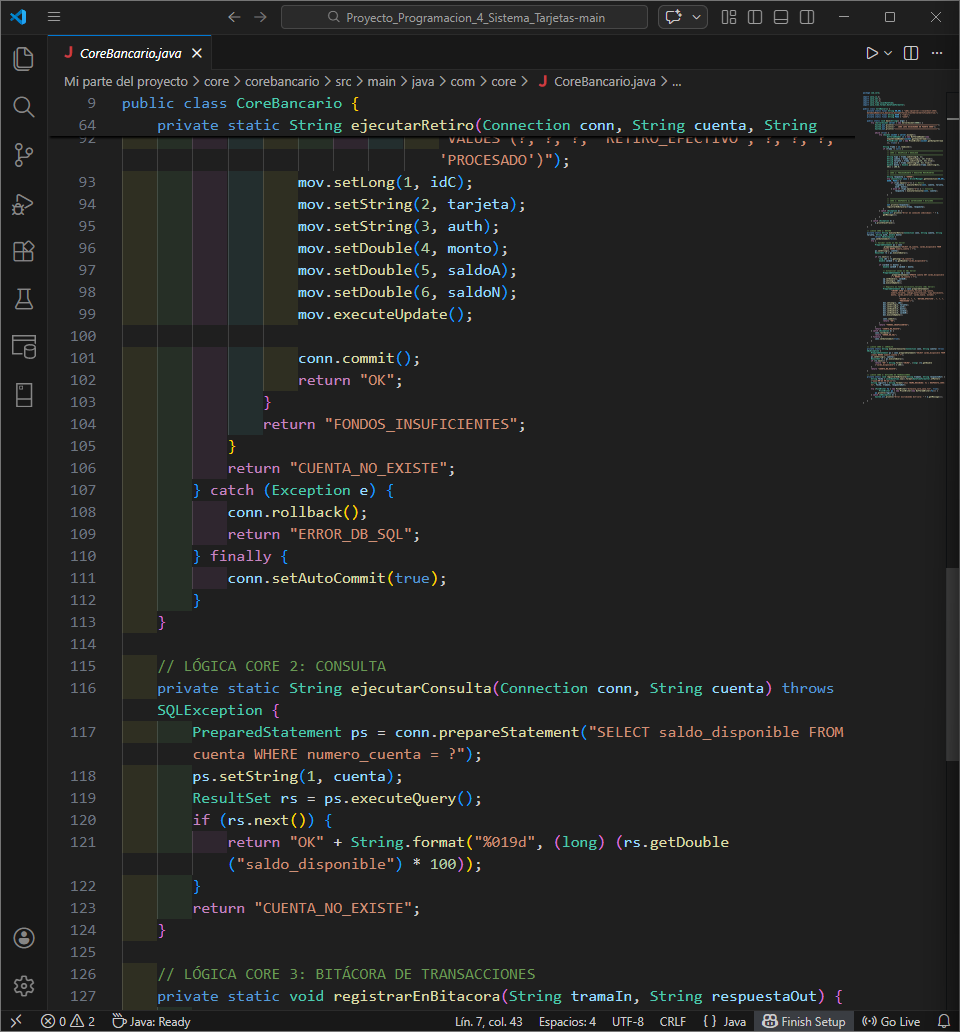


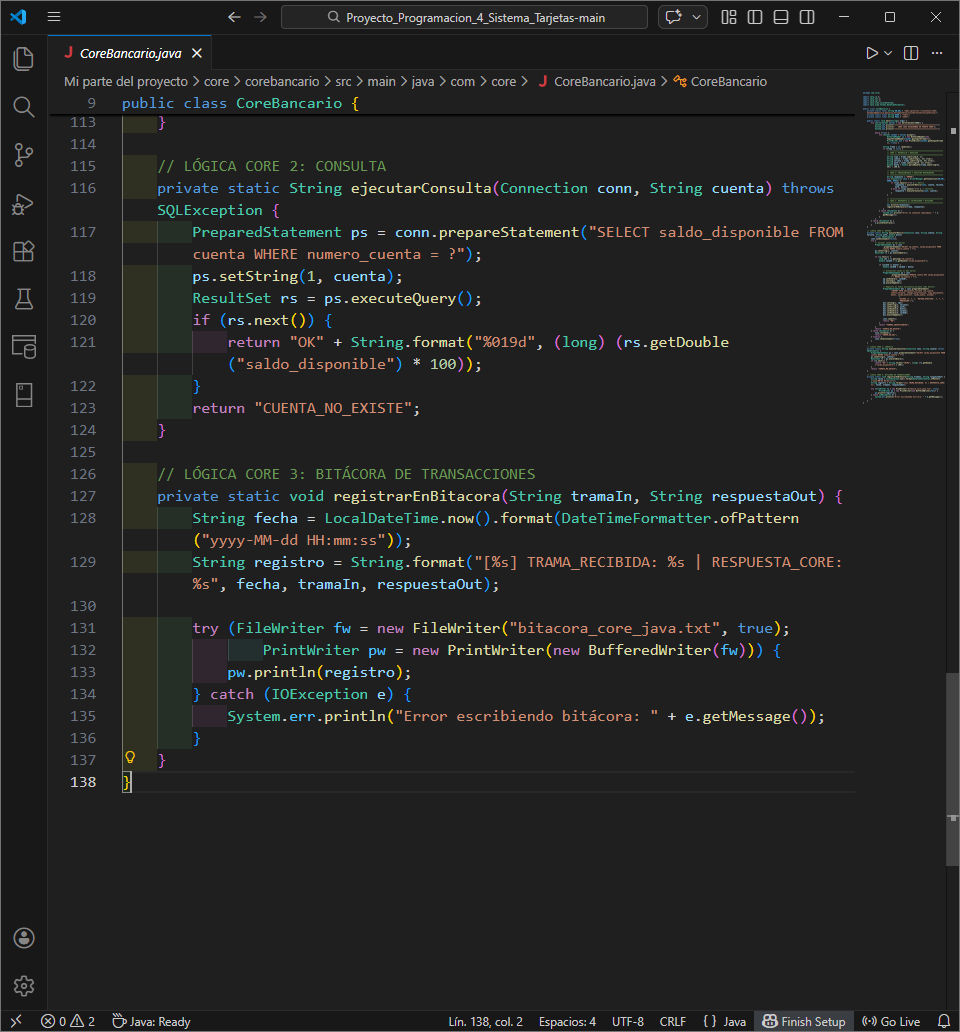
# Capitulo 4: Componentes del Core Bancario



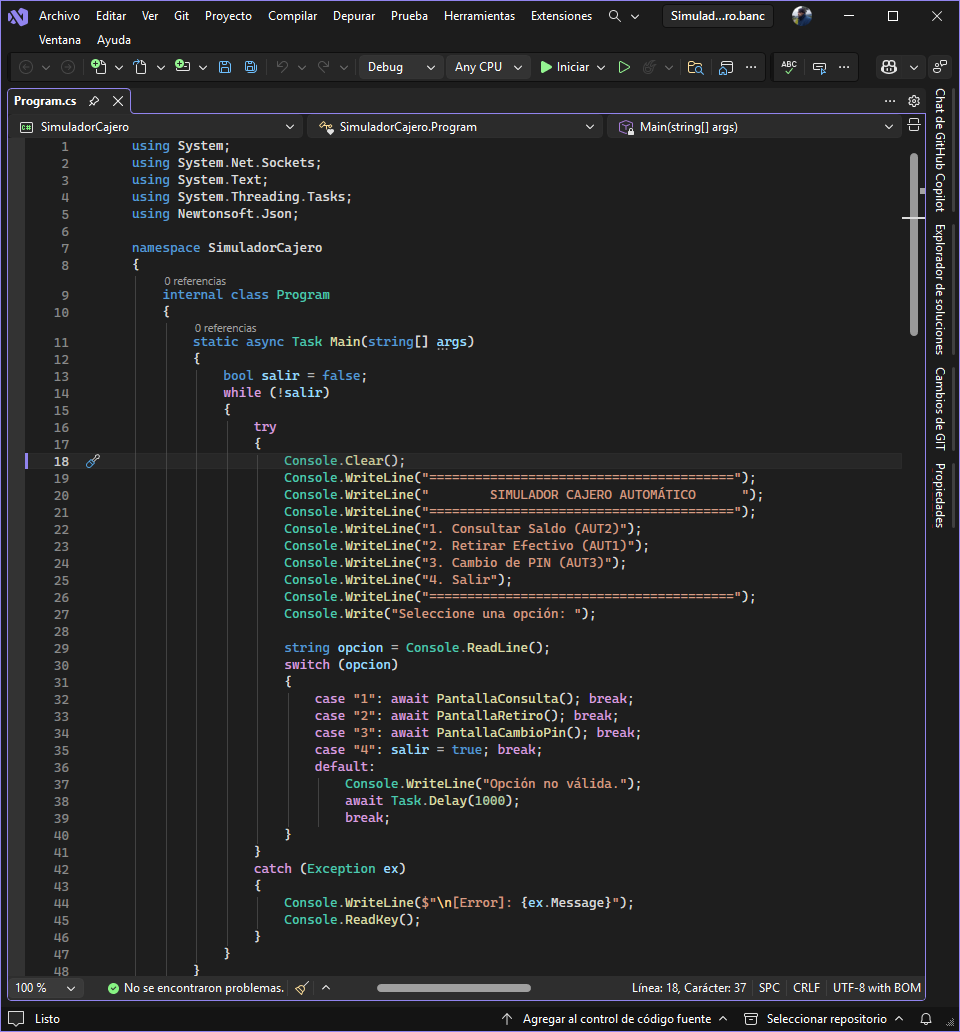


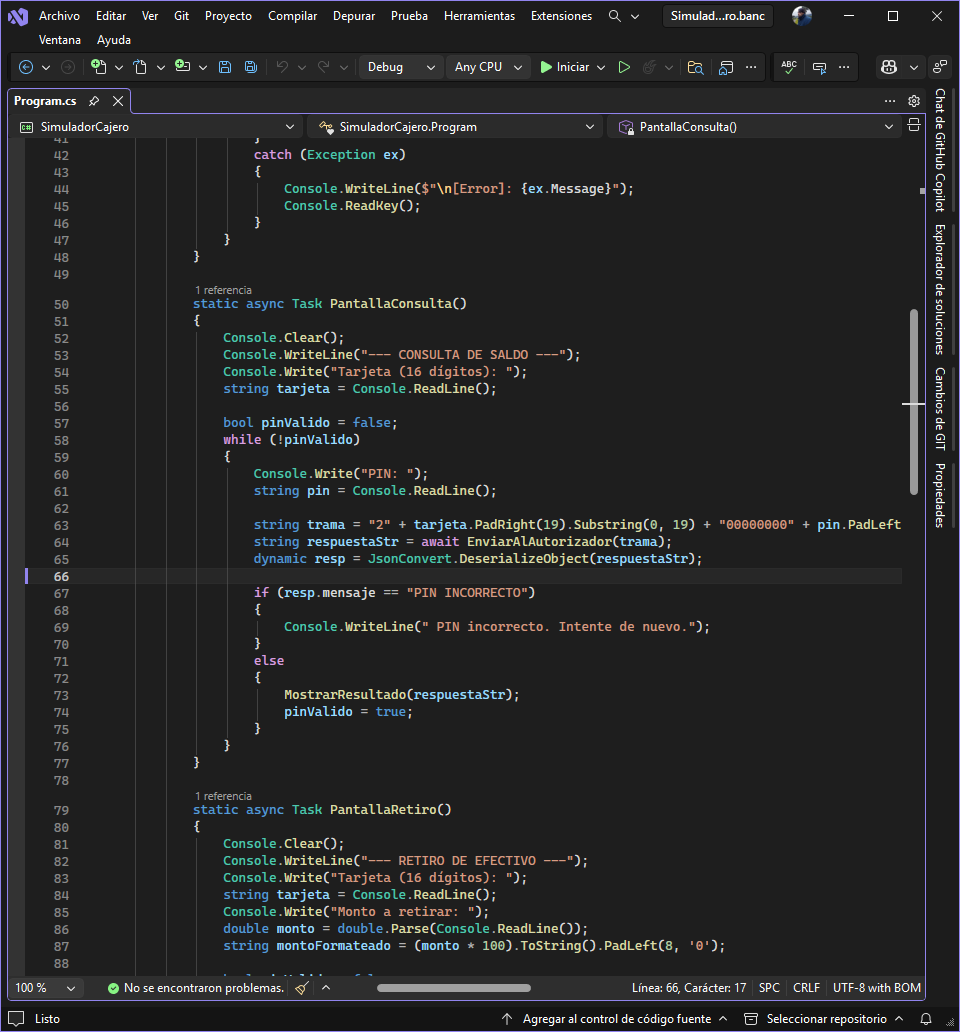




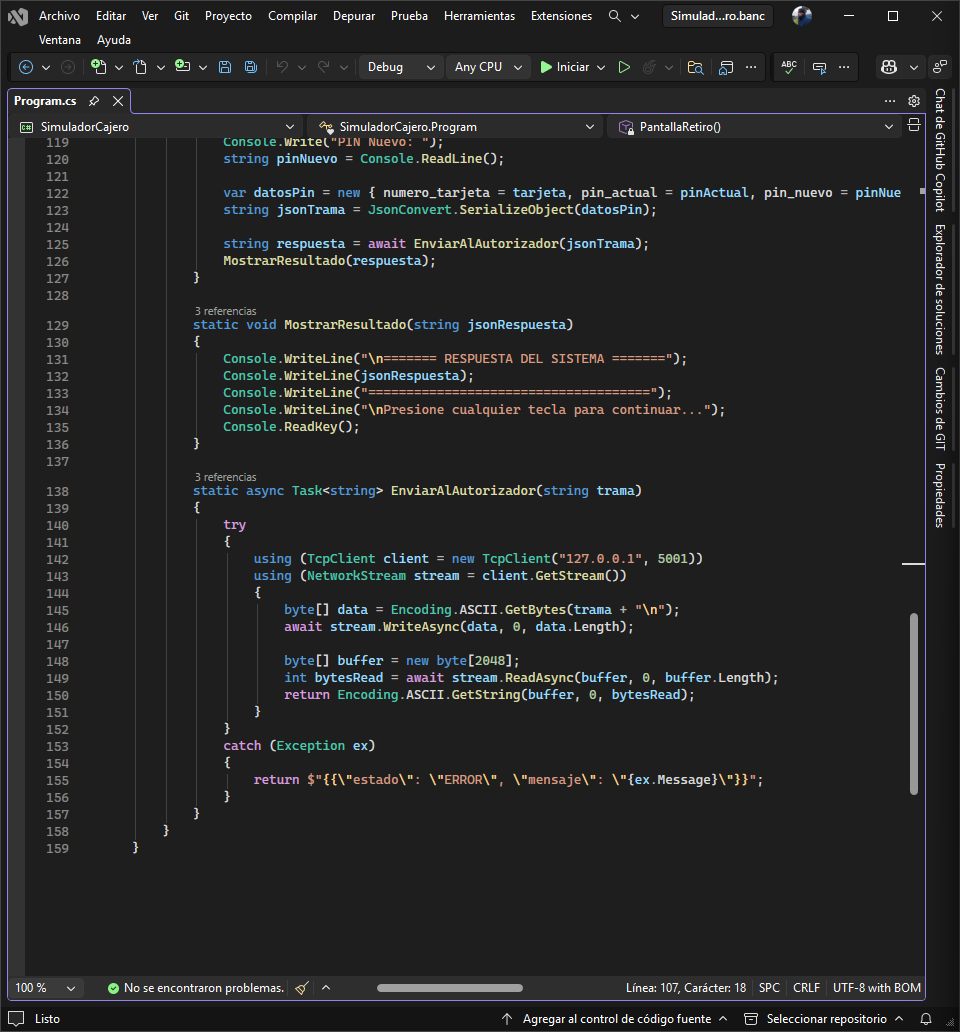


# Capitulo 5: Simulador de Transacciones









# Conclusiones

El desarrollo del Avance 1 del Sistema de Tarjetas representa un hito importante en la comprensión y aplicación práctica de los conceptos de programación distribuida, comunicación cliente-servidor, gestión de concurrencia y persistencia de datos en entornos reales. A lo largo de este primer avance, el grupo ha establecido las bases técnicas de un sistema bancario distribuido que integra múltiples componentes críticos de una arquitectura transaccional moderna.

La exitosa implementación del flujo autorizador muestra la habilidad del equipo para diseñar y ejecutar sistemas que manejan comunicación asíncrona, validación de seguridad y auditoría en tiempo real. La adopción de un modelo de bitácora en segundo plano mediante hilos independientes refleja un entendimiento profundo de los principios de no bloqueo y escalabilidad, que son clave en sistemas de alta disponibilidad como los que exige el sector financiero.

Además, el desarrollo del Core Bancario en Java con SQL Server muestra la competencia técnica del grupo en la integración de bases de datos relacionales empresariales y el manejo de transacciones. La correcta implementación del formato de trama fijo y el uso de PreparedStatements con commit/rollback aseguran la integridad referencial y la consistencia de datos, que son esenciales en cualquier sistema de gestión monetaria.

El proyecto ha permitido al equipo fortalecer sus conocimientos en arquitectura de microservicios, donde cada componente, opera de forma independiente pero coordinada, comunicándose mediante protocolos estandarizados. Esta separación de responsabilidades facilita el mantenimiento, la escalabilidad horizontal y la futura adición de nuevos servicios.

### LOGROS ALCANZADOS

Infraestructura distribuida robusta: Se estableció una arquitectura cliente-servidor multi-hilo capaz de manejar múltiples transacciones concurrentes sin pérdida de datos.

Sistema de auditoría completo: La implementación de bitácoras en tiempo real con encolamiento de mensajes asegura la trazabilidad total de operaciones y cumple con los estándares regulatorios bancarios.

Integración multiplataforma: La interoperabilidad entre Python, Java & SQL Server, demuestra habilidad en el manejo de heterogeneidad tecnológica.

Seguridad transaccional: Validaciones estrictas de PIN, tarjeta, fondos disponibles y códigos de autorización protegen la integridad del sistema.

Persistencia empresarial: El diseño de esquemas relacionales normalizados con índices optimizados asegura rendimiento y escalabilidad en SQL Server.

# Recomendaciones

### A. Arquitectura y Escalabilidad

Implementar balanceadores de carga para distribuir horizontalmente transacciones entre múltiples instancias del autorizador.

Usar colas de mensajes distribuidas en lugar de colas en memoria para asegurar la persistencia de eventos críticos.

Utilizar contenedores con Docker para despliegues consistentes en entornos de desarrollo, pruebas y producción.

### B. Seguridad y Cumplimiento

Usar algoritmos criptográficos robustos (AES-256) para cifrar datos sensibles en tránsito y reposo.

Implementar autenticación mutua entre componentes del sistema.

Gestionar la rotación automática de claves y auditorías de accesos a datos sensibles.

### C. Monitoreo y Observabilidad

Crear alertas en tiempo real para fallos críticos (conexiones a BD, saldos negativos).

### D. Calidad del Software

Mantener una cobertura de pruebas unitarias mayor al 85% para lógica crítica (validaciones, transacciones).

Realizar pruebas de integración automatizadas para validar la comunicación entre componentes.

Ejecutar pruebas de carga simulando más de 1000 transacciones concurrentes.

### E. Base de Datos

Implementar replicación de lectura en SQL Server para separar cargas.

Realizar backups automatizados con una estrategia 3-2-1 (tres copias, dos medios, una fuera del sitio).

F. Experiencia del Desarrollador

Estandarizar convenciones de nombres, formatos y documentación.

## COMPETENCIAS CONSOLIDADAS

El proyecto ha permitido al equipo dominar:

- Arquitectura de sistemas distribuidos

- Programación concurrente y paralela

- Integración empresarial multiplataforma

- Persistencia de datos ACID

- Gestión de transacciones financieras

- Auditoría y trazabilidad

- Optimización de rendimiento

PERSPECTIVAS

Este Avance 1 coloca al equipo en una buena posición para enfrentar desarrollos críticos en el sector financiero. La experiencia adquirida es aplicable a proyectos reales y sistemas de pagos.

Recomendación final: Mantener el rigor técnico demostrado, priorizar la calidad sobre la velocidad de desarrollo y documentar cada decisión arquitectónica para facilitar la incorporación de nuevos miembros y futuras auditorías.

# Bibliografía

*NestorLeiva. (s. f.). GitHub - NestorLeiva/Proyecto\_Programacion\_4\_Sistema\_Tarjetas: Avance 1. GitHub.* <https://github.com/NestorLeiva/Proyecto_Programacion_4_Sistema_Tarjetas>

*ChartDB – Database schema diagrams visualizer*. (2025, 11 agosto). <https://chartdb.io/> *Google Gemini*. (s. f.). Gemini. <https://gemini.google.com/app?hl=es>

*IA*. (s. f.). Perplexity. <https://www.perplexity.ai/>

*Sockets con Python. (s. f.). YouTube.* <https://www.youtube.com/playlist?list=PLoTnWByJggZUXdGv6aqRVkbR6qtVecZBG>