



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.....	3
Problema por Resolver.....	3
Diagrama de Base de Datos.....	4
Diagrama de Casos de Uso.....	6
Diagrama de Clases.....	7
Capítulo 2: Implementación de Base de Datos.....	8
Script MySQL.....	8
Script SQLServer.....	12
Capítulo 3: Autorizador de Transacciones.....	15
AUT 1 & 2.....	19
AUT 3.....	22
AUT 4.....	24
AUT 5.....	25
Capítulo 4: Componentes del Core Bancario.....	27
Capítulo 5: Simulador de Transacciones.....	32
Conclusiones.....	34
LOGROS ALCANZADOS.....	35
Recomendaciones.....	36
A. Arquitectura y Escalabilidad.....	36
B. Seguridad y Cumplimiento.....	36
C. Monitoreo y Observabilidad.....	36
D. Calidad del Software.....	36
E. Base de Datos.....	36
COMPETENCIAS CONSOLIDADAS.....	37
Bibliografía.....	38

Introducción

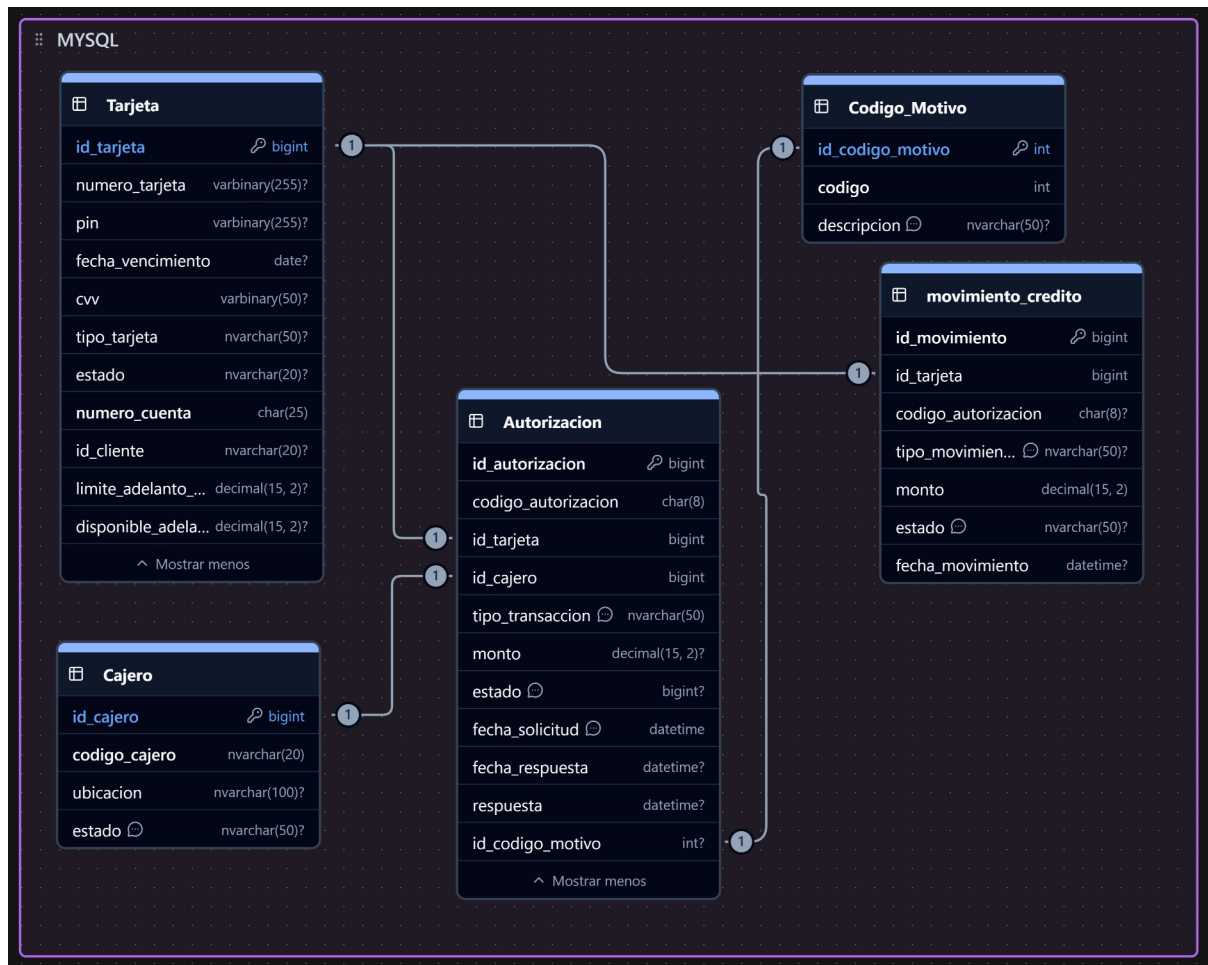
Problema por Resolver

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



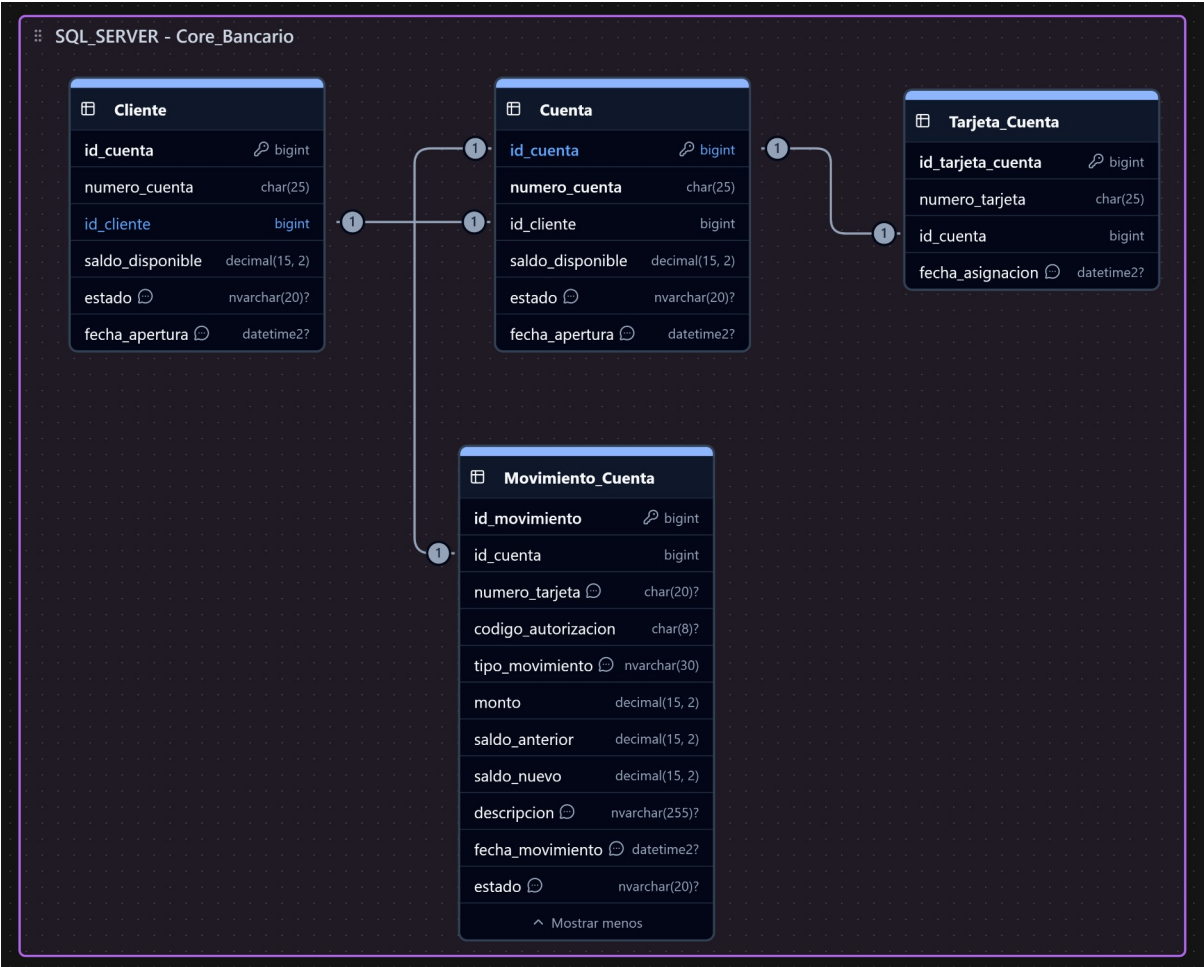


Diagrama de Casos de Uso

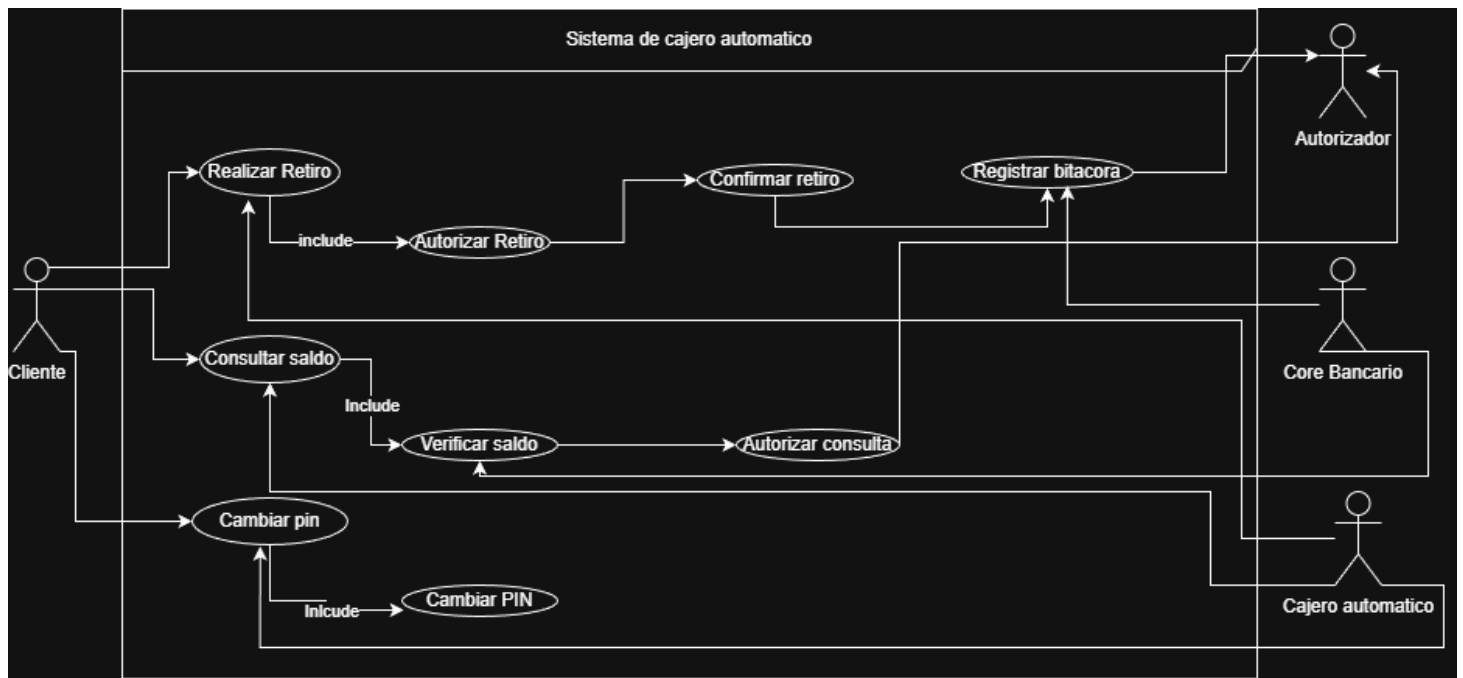
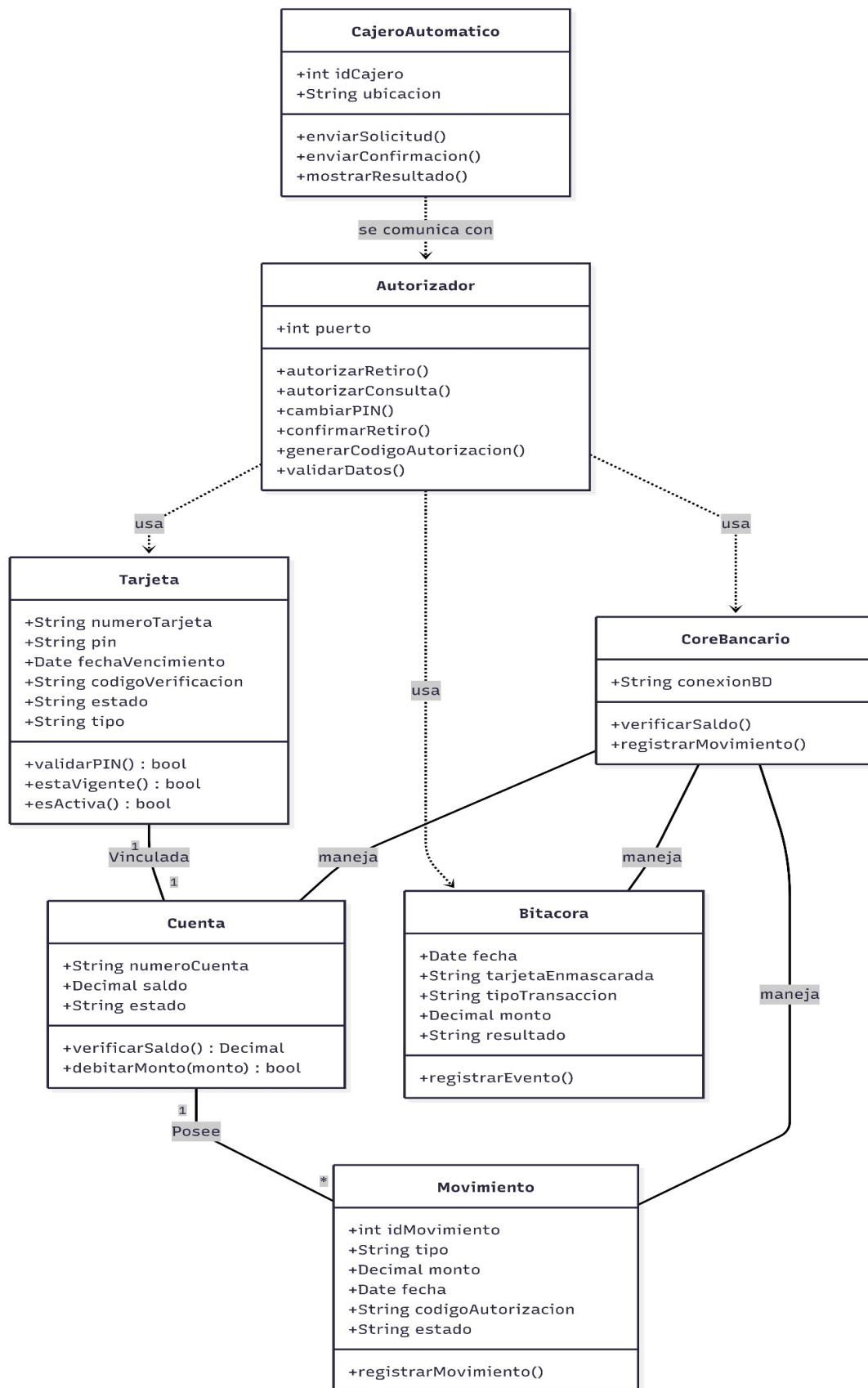


Diagrama de Clases



Capitulo 2: Implementación de Base de Datos

Script MySQL

```
--
=====
=====
--          SISTEMA DE TARJETAS DE CRÉDITO / DÉBITO CON AUDITORÍA
--          Proyecto Programación IV - Colegio Universitario de Cartago
--
=====
=====

DROP DATABASE IF EXISTS sistema_tarjeta;
CREATE DATABASE sistema_tarjeta CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;
USE sistema_tarjeta;

-- =====
-- TABLAS PRINCIPALES
-- =====

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'))
);

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
);

CREATE TABLE tipo_transaccion (
    id_tipo_transaccion INT PRIMARY KEY AUTO_INCREMENT,
    codigo_tipo VARCHAR(30) UNIQUE NOT NULL,
    descripcion VARCHAR(100) NOT NULL
);

CREATE TABLE cuenta (
    id_cuenta BIGINT PRIMARY KEY AUTO_INCREMENT,
    numero_cuenta VARCHAR(23) NOT NULL UNIQUE, -- Formato XXXX-XXXX-
XXXX-XX
    tipo_cuenta VARCHAR(10) NOT NULL CHECK (tipo_cuenta IN ('DEBITO',
'CREDITO')),
```



```

    saldo_disponible DECIMAL(15,2) NOT NULL DEFAULT 0 CHECK
(saldo_disponible >= 0),
    saldo_total DECIMAL(15,2) NOT NULL DEFAULT 0 CHECK (saldo_total >= 0),
    limite_credito DECIMAL(15,2) NOT NULL DEFAULT 0 CHECK (limite_credito >=
0),
    disponible_credito DECIMAL(15,2) NOT NULL DEFAULT 0 CHECK
(disponible_credito >= 0),
    estado VARCHAR(20) DEFAULT 'ACTIVA' CHECK (estado IN ('ACTIVA',
'INACTIVA', 'CERRADA'))
);

```

```

CREATE TABLE tarjeta (
    id_tarjeta BIGINT PRIMARY KEY AUTO_INCREMENT,
    numero_tarjeta VARCHAR(20) NOT NULL,
    pin VARCHAR(4) NOT NULL,
    fecha_vencimiento DATE NOT NULL,
    cvv VARCHAR(3) NOT NULL,
    tipo_tarjeta VARCHAR(10) NOT NULL CHECK (tipo_tarjeta IN ('DEBITO',
'CREDITO')),
    estado VARCHAR(20) DEFAULT 'ACTIVA' CHECK (estado IN ('ACTIVA',
'INACTIVA', 'VENCIDA')),
    id_cuenta BIGINT NOT NULL,
    identificacion_cliente VARCHAR(20),
    limite_adelanto_efectivo DECIMAL(15,2) DEFAULT 0 CHECK
(limite_adelanto_efectivo >= 0),
    disponible_adelanto_efectivo DECIMAL(15,2) DEFAULT 0 CHECK
(disponible_adelanto_efectivo >= 0),
    UNIQUE KEY uk_numero_tarjeta (numero_tarjeta),
    CONSTRAINT fk_tarjeta_cuenta FOREIGN KEY (id_cuenta) REFERENCES
cuenta(id_cuenta)
);

```

```

CREATE TABLE autorizacion (
    id_autorizacion BIGINT PRIMARY KEY AUTO_INCREMENT,
    codigo_autorizacion CHAR(8) NOT NULL UNIQUE,
    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)
);

CREATE TABLE movimiento (
    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_mov_credito_tarjeta FOREIGN KEY (id_tarjeta) REFERENCES
tarjeta(id_tarjeta),
    CONSTRAINT fk_mov_credito_aut FOREIGN KEY (codigo_autorizacion)
REFERENCES autorizacion(codigo_autorizacion)
);

-- TABLA AUDITORÍA CORREGIDA (sin DEFAULT USER())
CREATE TABLE auditoria (
    id_auditoria BIGINT PRIMARY KEY AUTO_INCREMENT,
    tabla_afectada VARCHAR(50) NOT NULL,
    accion VARCHAR(20) NOT NULL,
    id_registro_afectado BIGINT,
    numero_referencia VARCHAR(50),
    campo_modificado VARCHAR(50),
    valor_anterior TEXT,
    valor_nuevo TEXT,
    usuario VARCHAR(100) DEFAULT NULL,      -- CORREGIDO: NULL en lugar de
USER()
    ip_address VARCHAR(45) DEFAULT NULL,
    fecha_auditoria DATETIME DEFAULT CURRENT_TIMESTAMP,
    INDEX idx_fecha_accion (fecha_auditoria, accion),
    INDEX idx_numero_ref (numero_referencia),
    INDEX idx_tabla_fecha (tabla_afectada, fecha_auditoria)
);

-- TRIGGERS (capturan USER() dinámicamente)
DELIMITER $$
CREATE TRIGGER tr_tarjeta_update_auditoria
AFTER UPDATE ON tarjeta
FOR EACH ROW
BEGIN
    IF OLD.pin != NEW.pin THEN
        INSERT INTO auditoria (
            tabla_afectada, accion, numero_referencia,
            campo_modificado, valor_anterior, valor_nuevo,
            id_registro_afectado, usuario

```

```

        ) VALUES (
            'tarjeta', 'UPDATE', NEW.numero_tarjeta, 'pin',
            OLD.pin, NEW.pin, NEW.id_tarjeta, USER()
        );
    END IF;

    IF OLD.estado != NEW.estado THEN
        INSERT INTO auditoria (
            tabla_afectada, accion, numero_referencia,
            campo_modificado, valor_anterior, valor_nuevo, usuario
        ) VALUES (
            'tarjeta', 'UPDATE', NEW.numero_tarjeta, 'estado',
            OLD.estado, NEW.estado, USER()
        );
    END IF;
END$$

CREATE TRIGGER tr_cuenta_update_auditoria
AFTER UPDATE ON cuenta
FOR EACH ROW
BEGIN
    IF OLD.saldo_disponible != NEW.saldo_disponible OR
        OLD.saldo_total != NEW.saldo_total OR
        OLD.disponible_credito != NEW.disponible_credito THEN
        INSERT INTO auditoria (
            tabla_afectada, accion, numero_referencia, campo_modificado,
            valor_anterior, valor_nuevo, id_registro_afectado, usuario
        ) VALUES (
            'cuenta', 'UPDATE', NEW.numero_cuenta, 'saldos',
            CONCAT('disp:', OLD.saldo_disponible, '|total:', OLD.saldo_total, '|cred:',
OLD.disponible_credito),
            CONCAT('disp:', NEW.saldo_disponible, '|total:', NEW.saldo_total, '|cred:',
NEW.disponible_credito),
            NEW.id_cuenta, USER()
        );
    END IF;
END$$
DELIMITER ;

-- ÍNDICES
CREATE INDEX idx_cajero_codigo ON cajero(codigo_cajero);
CREATE INDEX idx_autorizacion_codigo ON autorizacion(codigo_autorizacion);
CREATE INDEX idx_autorizacion_fecha ON autorizacion(fecha_solicitud);
CREATE INDEX idx_cuenta_tipo ON cuenta(tipo_cuenta, estado);
CREATE INDEX idx_tarjeta_tipo ON tarjeta(tipo_tarjeta, estado);

--
=====
=====
--
ALTER TABLES

```

```
--
=====
=====

-- ALTER TABLE tarjeta
--      MODIFY numero_tarjeta VARBINARY (32), MODIFY pin VARBINARY (32),
MODIFY cvv VARBINARY (32);
```

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
```

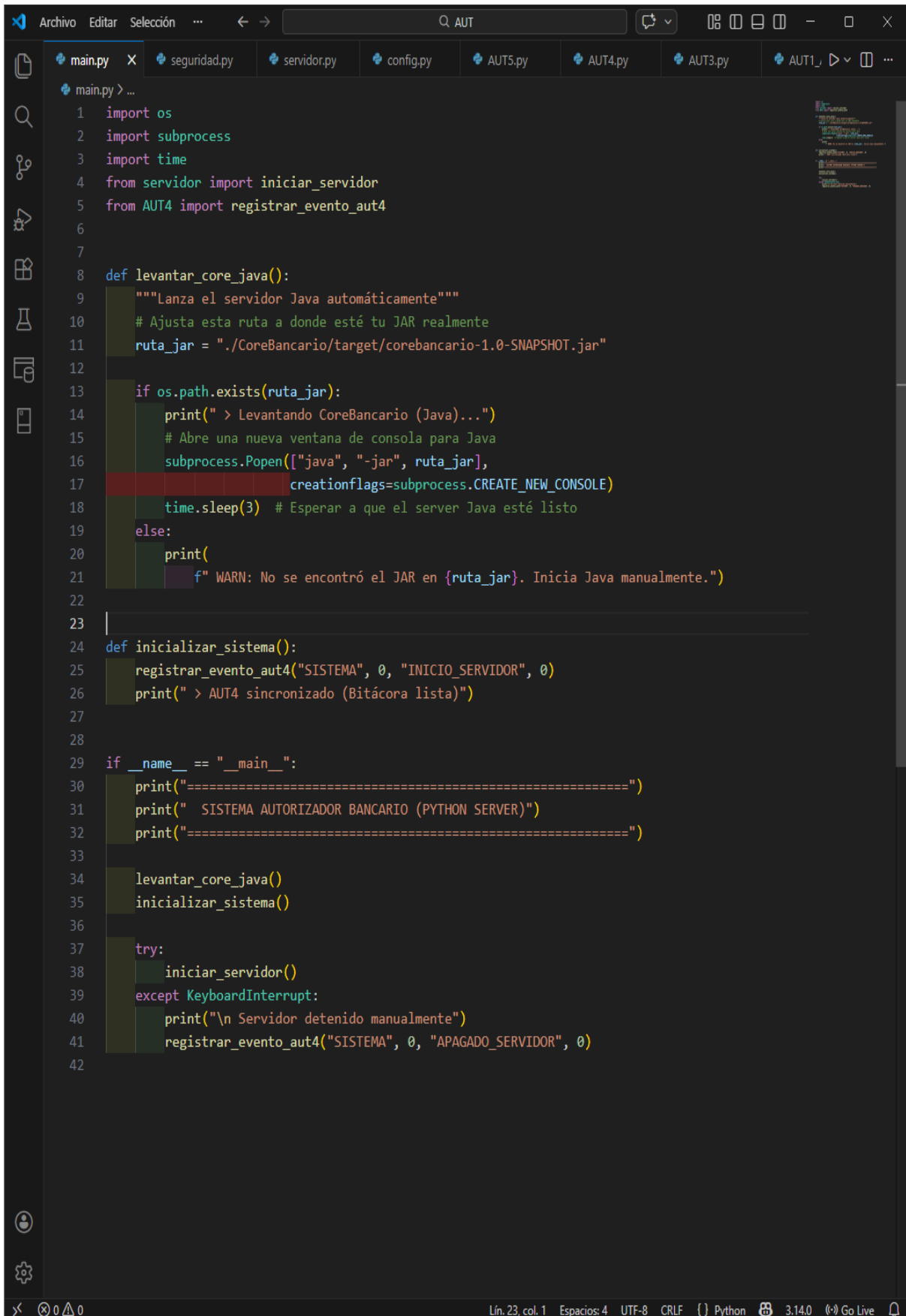
```
-- =====
```

```
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
```

Capítulo 3: Autorizador de Transacciones

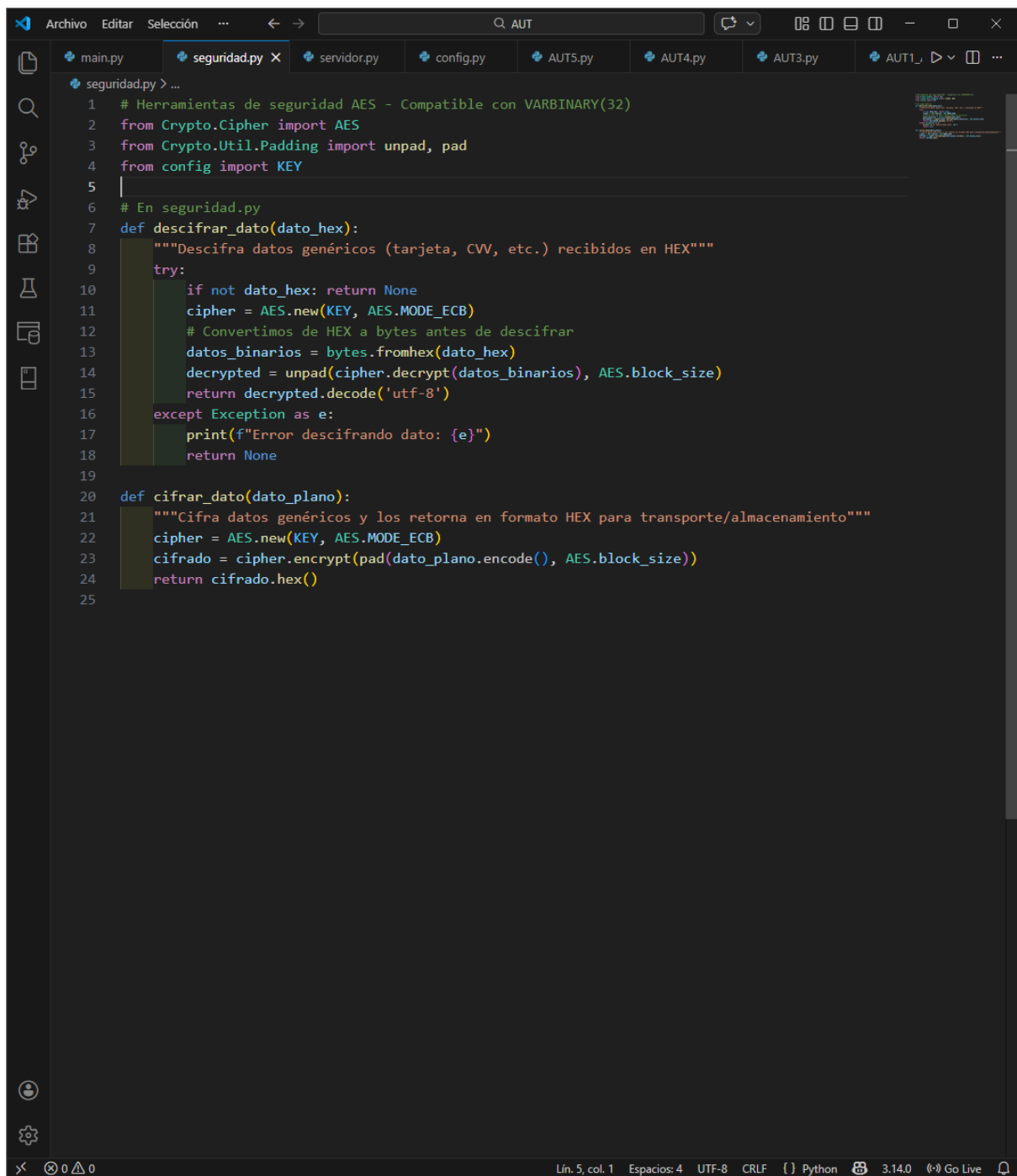


```
Archivo  Editar  Selección  ...  Q AUT  [Icons]  -  □  X

main.py X  seguridad.py  servidor.py  config.py  AUT5.py  AUT4.py  AUT3.py  AUT1_ [Dropdown] [Icons] ...

main.py > ...
1  import os
2  import subprocess
3  import time
4  from servidor import iniciar_servidor
5  from AUT4 import registrar_evento_aut4
6
7
8  def levantar_core_java():
9      """Lanza el servidor Java automáticamente"""
10     # Ajusta esta ruta a donde esté tu JAR realmente
11     ruta_jar = "./CoreBancario/target/corebancario-1.0-SNAPSHOT.jar"
12
13     if os.path.exists(ruta_jar):
14         print(" > Levantando CoreBancario (Java)...")
15         # Abre una nueva ventana de consola para Java
16         subprocess.Popen(["java", "-jar", ruta_jar],
17                          creationflags=subprocess.CREATE_NEW_CONSOLE)
18         time.sleep(3) # Esperar a que el server Java esté listo
19     else:
20         print(
21             f" WARN: No se encontró el JAR en {ruta_jar}. Inicia Java manualmente.")
22
23
24 def inicializar_sistema():
25     registrar_evento_aut4("SISTEMA", 0, "INICIO_SERVIDOR", 0)
26     print(" > AUT4 sincronizado (Bitácora lista)")
27
28
29 if __name__ == "__main__":
30     print("=====")
31     print("  SISTEMA AUTORIZADOR BANCARIO (PYTHON SERVER)")
32     print("=====")
33
34     levantar_core_java()
35     inicializar_sistema()
36
37     try:
38         iniciar_servidor()
39     except KeyboardInterrupt:
40         print("\n Servidor detenido manualmente")
41         registrar_evento_aut4("SISTEMA", 0, "APAGADO_SERVIDOR", 0)
42
```

Lin. 23, col. 1 Espacios: 4 UTF-8 CRLF [] Python 3,14.0 Go Live [Icon]

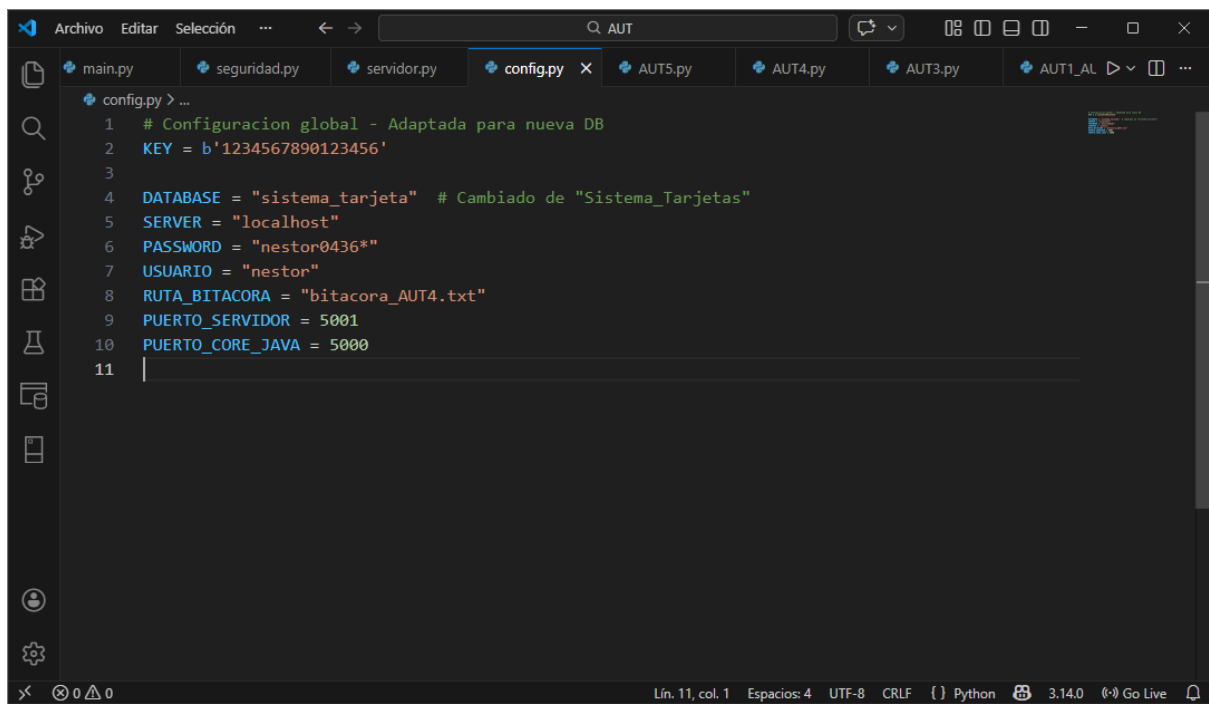


The image shows a code editor window with a dark theme. The top bar includes a menu (Archivo, Editar, Selección, ...), a search bar (Q AUT), and window controls. The tab bar shows several files: main.py, seguridad.py (active), servidor.py, config.py, AUT5.py, AUT4.py, AUT3.py, and AUT1.py. The editor displays the content of seguridad.py, which contains two functions for AES encryption and decryption using the Crypto module. The code is as follows:

```
1 # Herramientas de seguridad AES - Compatible con VARBINARY(32)
2 from Crypto.Cipher import AES
3 from Crypto.Util.Padding import unpad, pad
4 from config import KEY
5
6 # En seguridad.py
7 def descifrar_dato(dato_hex):
8     """Descifra datos genéricos (tarjeta, CVV, etc.) recibidos en HEX"""
9     try:
10         if not dato_hex: return None
11         cipher = AES.new(KEY, AES.MODE_ECB)
12         # Convertimos de HEX a bytes antes de descifrar
13         datos_binarios = bytes.fromhex(dato_hex)
14         decrypted = unpad(cipher.decrypt(datos_binarios), AES.block_size)
15         return decrypted.decode('utf-8')
16     except Exception as e:
17         print(f"Error descifrando dato: {e}")
18         return None
19
20 def cifrar_dato(dato_plano):
21     """Cifra datos genéricos y los retorna en formato HEX para transporte/almacenamiento"""
22     cipher = AES.new(KEY, AES.MODE_ECB)
23     cifrado = cipher.encrypt(pad(dato_plano.encode(), AES.block_size))
24     return cifrado.hex()
25
```

The status bar at the bottom shows: Lin. 5, col. 1, Espacios: 4, UTF-8, CRLF, Python, 3.14.0, and a Go Live button.


```
Archivo  Editar  Selección  ...  Q AUT  [Icons]  -  [Icons]  X
main.py  seguridad.py  servidor.py X  config.py  AUT5.py  AUT4.py  AUT3.py  AUT1_  [Icons]  ...
servidor.py > iniciar_servidor
1  import socket
2  import json
3  # Importamos la función centralizada desde tu archivo
4  from AUT1_AUT2 import procesar_retiro_consulta
5  from AUT3 import procesar_cambio_pin
6  from AUT5 import procesar_confirmacion_aut5
7
8
9  def iniciar_servidor():
10     puerto_escucha = 5001
11     server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
12     server.bind(('0.0.0.0', puerto_escucha))
13     server.listen(5)
14
15     print(f"=====")
16     print(f"  SERVIDOR AUTORIZADOR INTEGRADO ACTIVO")
17     print(f"  Puerto Escucha (C#): {puerto_escucha}")
18     print(f"=====")
19
20     while True:
21         client, addr = server.accept()
22         try:
23             # Recibimos la trama del simulador C#
24             data = client.recv(4096).decode('utf-8').strip()
25             if not data:
26                 continue
27
28             # 1. Si los datos vienen en formato JSON (AUT3 o AUT5)
29             if data.startswith('{'):
30                 datos_json = json.loads(data)
31                 tipo = datos_json.get('tipo', '')
32
33                 if tipo == 'cambio_pin':
34                     respuesta = procesar_cambio_pin(datos_json)
35                 elif 'cod_auth' in datos_json: # Flujo de confirmación AUT5
36                     respuesta = procesar_confirmacion_aut5(data)
37                 else:
38                     respuesta = {"estado": "ERROR",
39                                "mensaje": "Formato JSON no reconocido"}
40
41             # 2. Si es trama de texto fija (AUT1 o AUT2)
42             else:
43                 # Simplemente pasamos la trama completa a tu función en AUT1_AUT2.py
44                 # que ya se encarga de parsear [0:1], [1:17], etc.
45                 respuesta = procesar_retiro_consulta(data)
46
47             # Enviamos la respuesta de vuelta al simulador C#
48             client.send(json.dumps(respuesta).encode('utf-8'))
49
50         except Exception as e:
51             print(f"Error procesando cliente: {e}")
52             error_msg = {"estado": "ERROR", "mensaje": str(e)}
53             client.send(json.dumps(error_msg).encode('utf-8'))
54         finally:
55             client.close()
56
```



```
1 # Configuración global - Adaptada para nueva DB
2 KEY = b'1234567890123456'
3
4 DATABASE = "sistema_tarjeta" # Cambiado de "Sistema_Tarjetas"
5 SERVER = "localhost"
6 PASSWORD = "nestor0436*"
7 USUARIO = "nestor"
8 RUTA_BITACORA = "bitacora_AUT4.txt"
9 PUERTO_SERVIDOR = 5001
10 PUERTO_CORE_JAVA = 5000
11
```

AUT 1 & 2

```
Archivo  Editar  Selección  ...  Q AUT  [Icons]  -  x
AUT1_AUT2.py  seguridad.py  servidor.py  config.py  AUT5.py  AUT4.py  AUT3.py  AUT1_AUT2.py x [Icons] ...

AUT1_AUT2.py > ConexionDB > conectar
1  # AUT 1 y 2 - Retiros y Consultas - VERSION FINAL SINCRONIZADA
2  import socket
3  import random
4  import mysql.connector
5  from mysql.connector import Error
6  from config import DATABASE, SERVER, PASSWORD, USUARIO, PUERTO_CORE_JAVA
7  from AUT4 import registrar_evento_aut4
8
9  class ConexionDB:
10     """Conexion reutilizable para AUT 1/2 con patron Singleton básico"""
11     conn = None
12
13     @staticmethod
14     def conectar():
15         try:
16             if ConexionDB.conn is None or not ConexionDB.conn.is_connected():
17                 ConexionDB.conn = mysql.connector.connect(
18                     host=SERVER,
19                     port=3306,
20                     user=USUARIO,
21                     password=PASSWORD,
22                     database=DATABASE
23                 )
24             return True
25         except Error as e:
26             print(f"ERROR DB AUT1/2: {e}")
27             return False
28
29     def procesar_retiro_consulta(trama):
30         """
31         Procesa AUT1 (Retiro) y AUT2 (Consulta)
32         Trama esperada de C#: Tipo(1) + Tarjeta(16) + Monto(8) + PIN(4) + Cajero(4) = 33 caracteres
33         """
34         n_tarjeta_global = ""
35         try:
36             # 1. Parseo de la trama recibida del Simulador C#
37             tipo = trama[0:1] # '1' Retiro, '2' Consulta
38             n_tarjeta = trama[1:17].strip() # 16 dígitos
39             n_tarjeta_global = n_tarjeta
40             monto_raw = trama[17:25] # 8 dígitos (centavos)
41             pin_raw = trama[25:29] # 4 dígitos
42
43             # Extraer ID Cajero (posiciones 29 a 33)
44             try:
45                 id_cajero = int(trama[29:33])
46             except:
47                 id_cajero = 1 # Default si la trama viene incompleta
48
49             monto_f = float(monto_raw)
50
51             print(f"DEBUG - Procesando {'RETIRO' if tipo == '1' else 'CONSULTA'} en Cajero: {id_cajero}")
52             registrar_evento_aut4(n_tarjeta, id_cajero, f"SOLICITUD_{'RETIRO' if tipo == '1' else 'CONSULTA'}", monto_f)
53
54             # 2. Conexion a Base de Datos Local (MySQL)
55             if not ConexionDB.conectar():
56                 return {"estado": "ERROR", "mensaje": "Error de conexión con DB Autorizador"}
57
58             cursor = ConexionDB.conn.cursor(dictionary=True)
59
60             # 3. Validar existencia de Tarjeta y obtener datos de Cuenta
61             query = """
62             SELECT t.id_tarjeta, t.pin, t.estado as estado_tarjeta, t.id_cuenta,
63                    c.saldo_disponible, c.numero_cuenta, c.estado as estado_cuenta
64             FROM tarjeta t
65             JOIN cuenta c ON t.id_cuenta = c.id_cuenta
66             WHERE REPLACE(t.numero_tarjeta, '-', '') = REPLACE(%, '-', '')
67             """
68             cursor.execute(query, (n_tarjeta,))
69             tarjeta = cursor.fetchone()
70
71             # 4. Validaciones de Negocio
72             if not tarjeta:
73                 registrar_evento_aut4(n_tarjeta, id_cajero, "RECHAZADO_TARJETA_INEXISTENTE")
74                 return {"estado": "RECHAZADO", "mensaje": "Tarjeta no registrada"}
75
76             if tarjeta['estado_tarjeta'] != 'ACTIVA' or tarjeta['estado_cuenta'] != 'ACTIVA':
77                 registrar_evento_aut4(n_tarjeta, id_cajero, "RECHAZADO_ESTADO_INACTIVO")
78                 return {"estado": "RECHAZADO", "mensaje": "Tarjeta o cuenta bloqueada"}
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
250
```

```

Archivo  Editar  Selección  ...  <  >  Q AUT  [Icons]  -  [Icons]  X
seguridad.py  servidor.py  config.py  AUT5.py  AUT4.py  AUT3.py  AUT1_AUT2.py X

AUT1_AUT2.py > ConexionDB > conectar
29 def procesar_retiro_consulta(trama):
30
31     # 4. Validaciones de Negocio
32     if not tarjeta:
33         registrar_evento_aut4(n_tarjeta, id_cajero, "RECHAZADO_TARJETA_INEXISTENTE")
34         return {"estado": "RECHAZADO", "mensaje": "Tarjeta no registrada"}
35
36     if tarjeta['estado_tarjeta'] != 'ACTIVA' or tarjeta['estado_cuenta'] != 'ACTIVA':
37         registrar_evento_aut4(n_tarjeta, id_cajero, "RECHAZADO_ESTADO_INACTIVO")
38         return {"estado": "RECHAZADO", "mensaje": "Tarjeta o cuenta bloqueada"}
39
40     if tarjeta['pin'] != pin_raw:
41         registrar_evento_aut4(n_tarjeta, id_cajero, "RECHAZADO_PIN_ERRONEO")
42         return {"estado": "RECHAZADO", "mensaje": "PIN incorrecto"}
43
44     # Validar saldo localmente antes de ir al Core (solo para retiros)
45     if tipo == "1" and monto_f > float(tarjeta['saldo_disponible']):
46         registrar_evento_aut4(n_tarjeta, id_cajero, "RECHAZADO_SALDO_INSUF", monto_f)
47         return {"estado": "RECHAZADO", "mensaje": "Saldo insuficiente en autorizador"}
48
49     # 5. Comunicación con el CORE BANCARIO (Java)
50     # Sincronización: Tipo(1) + Cuenta(23) + Tarjeta(18) + CodAuth(8) + Monto(8) = 58
51     # caracteres
52     cod_auth = str(random.randint(10000000, 99999999))
53     n_tarjeta_limpia = n_tarjeta.replace('-', '')
54     n_cuenta_db = tarjeta['numero_cuenta']
55
56     trama_java = f"{tipo}{n_cuenta_db.ljust(23)}{n_tarjeta_limpia.ljust(18)}{cod_auth}{monto_f}"
57
58     print(f"INFO - Enviando trama al Core Java (Puerto {PUERTO_CORE_JAVA})...")
59
60     resp_core = "ERROR_COMUNICACION_CORE"
61     try:
62         with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s_java:
63             s_java.settimeout(5)
64             s_java.connect(('127.0.0.1', PUERTO_CORE_JAVA))
65             s_java.sendall((trama_java + "\n").encode())
66             resp_core = s_java.recv(1024).decode().strip()
67             print(f"INFO - Core Java respondió: {resp_core}")
68     except Exception as e:
69         print(f"CRITICAL - No se pudo contactar al Core Java: {e}")
70         return {"estado": "ERROR", "mensaje": "Core Bancario no disponible"}
71
72     # 6. Procesar respuesta del Core y afectar DB MySQL
73     cursor = ConexionDB.conn.cursor()
74     id_tipo_trans = 1 if tipo == '1' else 2 # 1: Retiro, 2: Consulta
75
76     if "OK" in resp_core.upper():
77         # A) Si es Retiro, rebajamos el saldo también en MySQL para mantener sincronía
78         if tipo == "1":
79             cursor.execute("""
80                 UPDATE cuenta
81                 SET saldo_disponible = saldo_disponible - %s
82                 WHERE id_cuenta = %s
83                 """, (monto_f, tarjeta['id_cuenta']))
84             print(f"SUCCESS - Saldo actualizado en MySQL para cuenta {n_cuenta_db}")
85
86             # B) Registrar la transacción como APROBADA
87             cursor.execute("""
88                 INSERT INTO autorizacion (codigo_autorizacion, id_tarjeta, id_cajero,
89                 id_tipo_transaccion, monto, estado, fecha_solicitud, respuesta)
90                 VALUES (%s, %s, %s, %s, %s, 'APROBADA', NOW(), %s)
91                 """, (cod_auth, tarjeta['id_tarjeta'], id_cajero, id_tipo_trans, monto_f,
92                 resp_core))
93
94             ConexionDB.conn.commit()
95             registrar_evento_aut4(n_tarjeta, id_cajero, "TRANSACCION_EXITOSA", monto_f)
96
97             # C) Preparar respuesta para C#
98             resultado = {"estado": "APROBADO", "codigo_autorizacion": cod_auth}
99             if tipo == "2":
100                 # Convertimos Decimal a float para que sea serializable en JSON
101                 resultado["saldo"] = float(tarjeta['saldo_disponible'])
102
103             return resultado
104
105     else:
106         # Si el Core rechazó (ej. "SALDO_INSUFICIENTE" o "ERROR_DB")
107         registrar_evento_aut4(n_tarjeta, id_cajero, f"RECHAZADO_POR_CORE_{resp_core}")

```

```
29 def procesar_retiro_consulta(trama):
133     registrar_evento_aut4(n_tarjeta, id_cajero, TRANSACCION_EXITOSA, monto_t)
134
135     # C) Preparar respuesta para C#
136     resultado = {"estado": "APROBADO", "codigo_autorizacion": cod_auth}
137     if tipo == "2":
138         # Convertimos Decimal a float para que sea serializable en JSON
139         resultado["saldo"] = float(tarjeta['saldo_disponible'])
140
141     return resultado
142
143 else:
144     # Si el Core rechazó (ej. "SALDO_INSUFICIENTE" o "ERROR_DB")
145     registrar_evento_aut4(n_tarjeta, id_cajero, f"RECHAZADO_POR_CORE_{resp_core}")
146     return {"estado": "RECHAZADO", "mensaje": f"Core indica: {resp_core}"}
147
148 except Exception as e:
149     print(f"FATAL ERROR en AUT1/2: {e}")
150     return {"estado": "ERROR", "mensaje": str(e)}
151 finally:
152     if 'cursor' in locals() and cursor:
153         cursor.close()
```

Lín. 130, col. 71 Espacios: 4 UTF-8 CRLF {} Python 3.14.0 Go Live

AUT 3

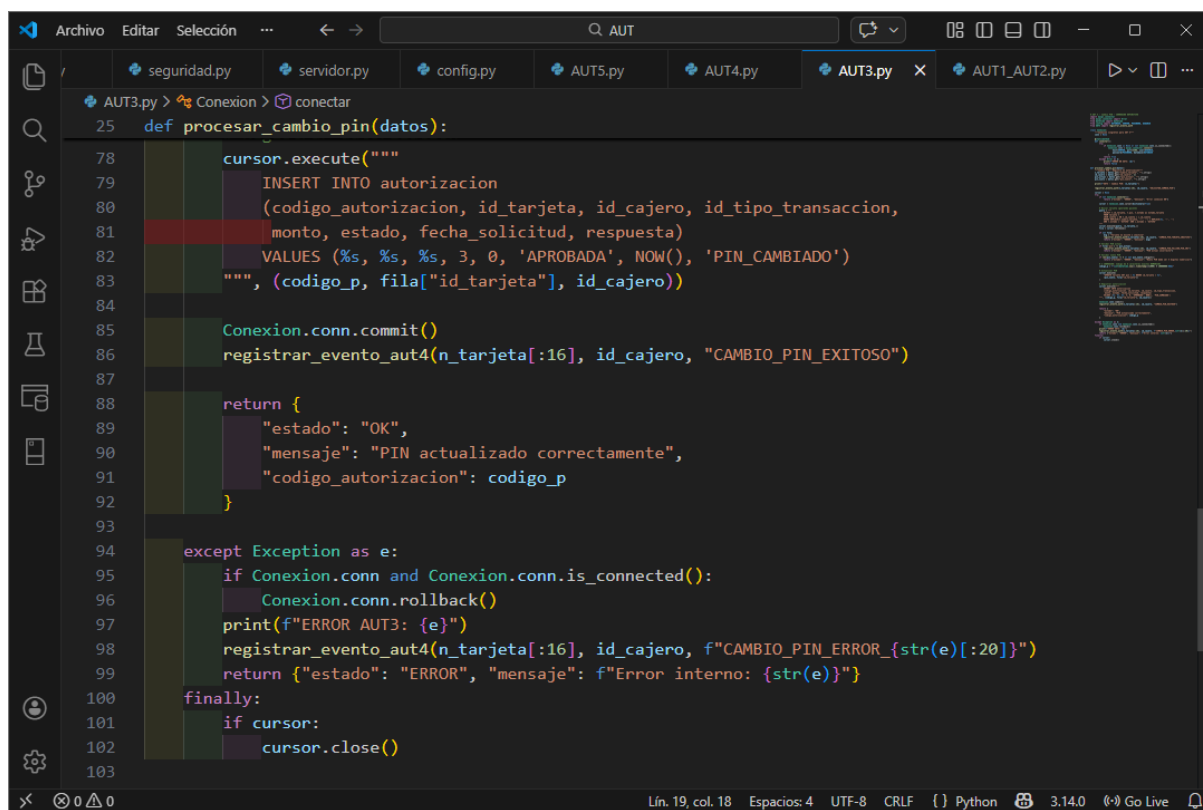
ArchivoEditarSelección...Q AUT

seguridad.pyserver.pyconfig.pyAUT5.pyAUT4.pyAUT3.pyxAUT1_AUT2.py

AUT3.py > Conexión > conectar

```
1 # AUT 3 - Cambio PIN - CORREGIDO DEFINITIVO
2 import mysql.connector
3 from mysql.connector import Error
4 from datetime import datetime
5 from config import DATABASE, SERVER, PASSWORD, USUARIO
6 from AUT4 import registrar_evento_aut4
7
8 class Conexion:
9     """Conexion singleton para AUT 3"""
10     conn = None
11
12     @staticmethod
13     def conectar():
14         try:
15             if Conexion.conn is None or not Conexion.conn.is_connected():
16                 Conexion.conn = mysql.connector.connect(
17                     host=SERVER, port=3306, user=USUARIO,
18                     password=PASSWORD, database=DATABASE
19                 )
20             return True
21         except Error as e:
22             print(f"ERROR DB AUT3: {e}")
23             return False
24
25     def procesar_cambio_pin(datos):
26         """Cambia PIN - Registra en autorizacion"""
27         n_tarjeta = datos.get("numero_tarjeta", "").strip()
28         id Cajero = datos.get("id_cajero", 1)
29         pin_actual = datos.get("pin_actual", "").strip()
30         pin_nuevo = datos.get("pin_nuevo", "").strip()
31
32         print(f"AUT3 - Cambio PIN: {n_tarjeta}")
33
34         registrar_evento_aut4(n_tarjeta[:16], id_cajero, "SOLICITUD_CAMBIO_PIN")
35
36         cursor = None
37         try:
38             if not Conexion.conectar():
39                 return {"estado": "ERROR", "mensaje": "Error conexion DB"}
40
41             cursor = Conexion.conn.cursor(dictionary=True)
42
43             # Buscar tarjeta ignorando guiones
44             query = """
45             SELECT t.id_tarjeta, t.pin, t.estado as estado_tarjeta
46             FROM tarjeta t
47             JOIN cuenta c ON t.id_cuenta = c.id_cuenta
48             WHERE REPLACE(t.numero_tarjeta, '-', '') = REPLACE(%s, '-', '')
49             AND t.estado = 'ACTIVA' AND c.estado = 'ACTIVA'
50             """
51             cursor.execute(query, (n_tarjeta,))
52             fila = cursor.fetchone()
53
54             if not fila:
55                 msg = "Tarjeta no existe o inactiva"
56                 registrar_evento_aut4(n_tarjeta[:16], id_cajero, "CAMBIO_PIN_TARJETA_INACTIVA")
57                 return {"estado": "ERROR", "mensaje": msg}
58
59             # Validar PIN actual
60             if fila['pin'] != pin_actual:
61                 registrar_evento_aut4(n_tarjeta[:16], id_cajero, "CAMBIO_PIN_FALLIDO_PIN_INC")
62                 return {"estado": "ERROR", "mensaje": "PIN actual incorrecto"}
63
64             # Validar nuevo PIN
65             if len(pin_nuevo) != 4 or not pin_nuevo.isdigit():
66                 return {"estado": "ERROR", "mensaje": "Nuevo PIN debe ser 4 digitos numericos"}
67
68             # CORREGIDO: Código de 8 caracteres exactos NUMERICOS
69             codigo_p = f"{int(datetime.now().timestamp()*1000) % 10000000:08d}"
70
71             # Actualizar PIN
72             cursor.execute(
73                 "UPDATE tarjeta SET pin = %s WHERE id_tarjeta = %s",
74                 (pin_nuevo, fila["id_tarjeta"])
75             )
76
77             # Registrar autorizacion
78             cursor.execute("""
79             INSERT INTO autorizacion
80             (codigo_autorizacion, id_tarjeta, id_cajero, id_tipo_transaccion,
```

Lín. 19, col. 18Espacios: 4UTF-8CRLF[]Python3.14.0Go Live



```
25 def procesar_cambio_pin(datos):
78     cursor.execute("""
79         INSERT INTO autorizacion
80         (codigo_autorizacion, id_tarjeta, id_cajero, id_tipo_transaccion,
81         monto, estado, fecha_solicitud, respuesta)
82         VALUES (%s, %s, %s, 3, 0, 'APROBADA', NOW(), 'PIN_CAMBIADO')
83     """, (codigo_p, fila["id_tarjeta"], id_cajero))
84
85     Conexion.conn.commit()
86     registrar_evento_aut4(n_tarjeta[:16], id_cajero, "CAMBIO_PIN_EXITOSO")
87
88     return {
89         "estado": "OK",
90         "mensaje": "PIN actualizado correctamente",
91         "codigo_autorizacion": codigo_p
92     }
93
94     except Exception as e:
95         if Conexion.conn and Conexion.conn.is_connected():
96             Conexion.conn.rollback()
97             print(f"ERROR AUT3: {e}")
98             registrar_evento_aut4(n_tarjeta[:16], id_cajero, f"CAMBIO_PIN_ERROR_{str(e)[:20]}")
99             return {"estado": "ERROR", "mensaje": f"Error interno: {str(e)}"}
100
101     finally:
102         if cursor:
103             cursor.close()
```

AUT 4

```
Archivo  Editar  Selección  ...  AUT
AUT4.py  seguridad.py  servidor.py  config.py  AUT5.py  AUT4.py  AUT3.py  AUT1_AUT2.py

1  import threading
2  import queue
3  import json
4  import os
5  from datetime import datetime
6
7  # Configuración de la ruta (ajusta si es necesario)
8  RUTA_BITACORA = "bitacora_4.txt"
9
10 # 1. La Cola para manejar las peticiones de forma ordenada
11 cola_bitacora = queue.Queue()
12
13 def enmascarar_tarjeta(tarjeta):
14     """Enmascara la tarjeta al formato: 1345 45** **** 2587"""
15     t = str(tarjeta).replace("-", "").replace(" ", "")
16     if len(t) >= 16:
17         # Formato específico según criterio 2.b
18         return f"{t[0:4]} {t[4:6]}** **** {t[12:16]}"
19     return t
20
21 def worker_bitacora():
22     """
23     Función que procesa la cola en segundo plano.
24     Este es el nombre que tu servidor.py está intentando importar.
25     """
26     while True:
27         # Obtiene el evento de la cola
28         evento = cola_bitacora.get()
29         if evento is None:
30             break
31
32         try:
33             fecha_str = datetime.now().strftime("%d/%m/%Y")
34             hora_str = datetime.now().strftime("%H:%M:%S")
35
36             # Estructura JSON según criterio 3
37             datos_json = {
38                 "tarjeta": enmascarar_tarjeta(evento.get('tarjeta')),
39                 "cajero": evento.get('cajero'),
40                 "cliente": evento.get('cliente'),
41                 "tipo": evento.get('tipo'),
42                 "monto": f"{float(evento.get('monto', 0)):.2f}" if evento.get('monto') is not
                        None else "0.00"
43             }
44
45             # Formato de línea: Fecha: {JSON}
46             linea = f"{fecha_str} {hora_str}: {json.dumps(datos_json, ensure_ascii=False)}\n"
47
48             with open(RUTA_BITACORA, 'a', encoding='utf-8') as f:
49                 f.write(linea)
50
51         except Exception as e:
52             print(f"Error en hilo de bitácora: {e}")
53         finally:
54             cola_bitacora.task_done()
55
56 # 2. Iniciar el hilo de forma automática al importar el módulo
57 hilo = threading.Thread(target=worker_bitacora, daemon=True)
58 hilo.start()
59
60 def registrar_evento_aut4(tarjeta, cajero, tipo, monto=None, cliente="112340456"):
61     """Función para encolar registros desde cualquier parte del sistema"""
62     evento = {
63         "tarjeta": tarjeta,
64         "cajero": cajero,
65         "cliente": cliente,
66         "tipo": tipo,
67         "monto": monto
68     }
69     cola_bitacora.put(evento)
```


AUT 5

```
Archivo Editar Selección ... AUT
seguridad.py servidor.py config.py AUT5.py X AUT4.py AUT3.py AUT1_AUT2.py
AUT5.py > procesar_confirmacion_aut5
1 import json
2 import socket
3 import mysql.connector
4 from config import DATABASE, SERVER, PASSWORD, USUARIO, PUERTO_CORE_JAVA
5 from seguridad import descifrar_dato, cifrar_dato
6 from AUT4 import registrar_evento_aut4
7
8 def procesar_confirmacion_aut5(trama_json):
9     conn = None
10    try:
11        datos = json.loads(trama_json)
12
13        # 1. Validar datos obligatorios
14        campos = ['cod_auth', 'tarjeta', 'vencimiento', 'cvv', 'id_cajero', 'monto']
15        if not all(k in datos for k in campos):
16            return {"estado": "ERROR", "mensaje": "Faltan datos obligatorios"}
17
18        # 2. Descifrar datos
19        n_tarjeta = descifrar_dato(datos['tarjeta'])
20        f_vencimiento = descifrar_dato(datos['vencimiento'])
21        cvv = descifrar_dato(datos['cvv'])
22
23        if not n_tarjeta or not f_vencimiento or not cvv:
24            return {"estado": "ERROR", "mensaje": "Fallo en descifrado de datos sensibles"}
25
26        # 3. Conexión a DB MySQL (Autorizador)
27        conn = mysql.connector.connect(
28            host=SERVER, user=USUARIO, password=PASSWORD, database=DATABASE
29        )
30        cursor = conn.cursor(dictionary=True)
31
32        # 4. Validaciones contra DB
33        query = """
34        SELECT t.id_tarjeta, t.id_cuenta, t.estado as t_estado, t.fecha_vencimiento, t.cvv,
35        c.tipo_cuenta, c.numero_cuenta, c.estado as c_estado
36        FROM tarjeta t
37        JOIN cuenta c ON t.id_cuenta = c.id_cuenta
38        WHERE REPLACE(t.numero_tarjeta, '-', '') = REPLACE(%, '-', '')
39        """
40        cursor.execute(query, (n_tarjeta,))
41        tarjeta_db = cursor.fetchone()
42
43        if not tarjeta_db or tarjeta_db['t_estado'] != 'ACTIVA':
44            return {"estado": "ERROR", "mensaje": "Tarjeta no existe o inactiva"}
45
46        # Comparar datos descifrados con DB
47        if str(tarjeta_db['fecha_vencimiento']) != f_vencimiento or tarjeta_db['cvv'] != cvv:
48            return {"estado": "ERROR", "mensaje": "Validación de seguridad fallida"}
49
50        # Validar Código de Autorización previo
51        cursor.execute("SELECT * FROM autorizacion WHERE codigo_autorizacion = %s AND
52        id_tarjeta = %s",
53            (datos['cod_auth'], tarjeta_db['id_tarjeta']))
54        if not cursor.fetchone():
55            return {"estado": "ERROR", "mensaje": "Código de autorización no corresponde"}
56
57        # 5. Lógica de Negocio (Débito vs Crédito)
58        monto = float(datos['monto'])
59
60        if tarjeta_db['tipo_cuenta'] == 'DEBITO':
61            # --- CONSTRUCCIÓN DE TRAMA SINCRONIZADA CON COREBANCARIO.JAVA ---
62            # Tipo(1) + Cuenta(23) + Tarjeta(18) + CodAuth(8) + Monto(8) = 58 caracteres
63            num_tarjeta_limpia = n_tarjeta.replace("-", "")[:18]
64
65            trama_core = (
66                "C" +
67                tarjeta_db['numero_cuenta'].ljust(23) +
68                num_tarjeta_limpia.ljust(18) +
69                datos['cod_auth'].ljust(8) +
70                f"{int(monto * 100):08d}"
71            )
72
73            try:
74                with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s_java:
75                    s_java.settimeout(5)
76                    s_java.connect(('127.0.0.1', PUERTO_CORE_JAVA))
77                    s_java.sendall((trama_core + "\n").encode())
78                    resp_core = s_java.recv(1024).decode().strip()
79
80            print(f"DEBUG: Trama enviada -> {trama_core}")
81
82    except Exception as e:
83        print(f"Error: {e}")
84    finally:
85        if conn:
86            conn.close()
```

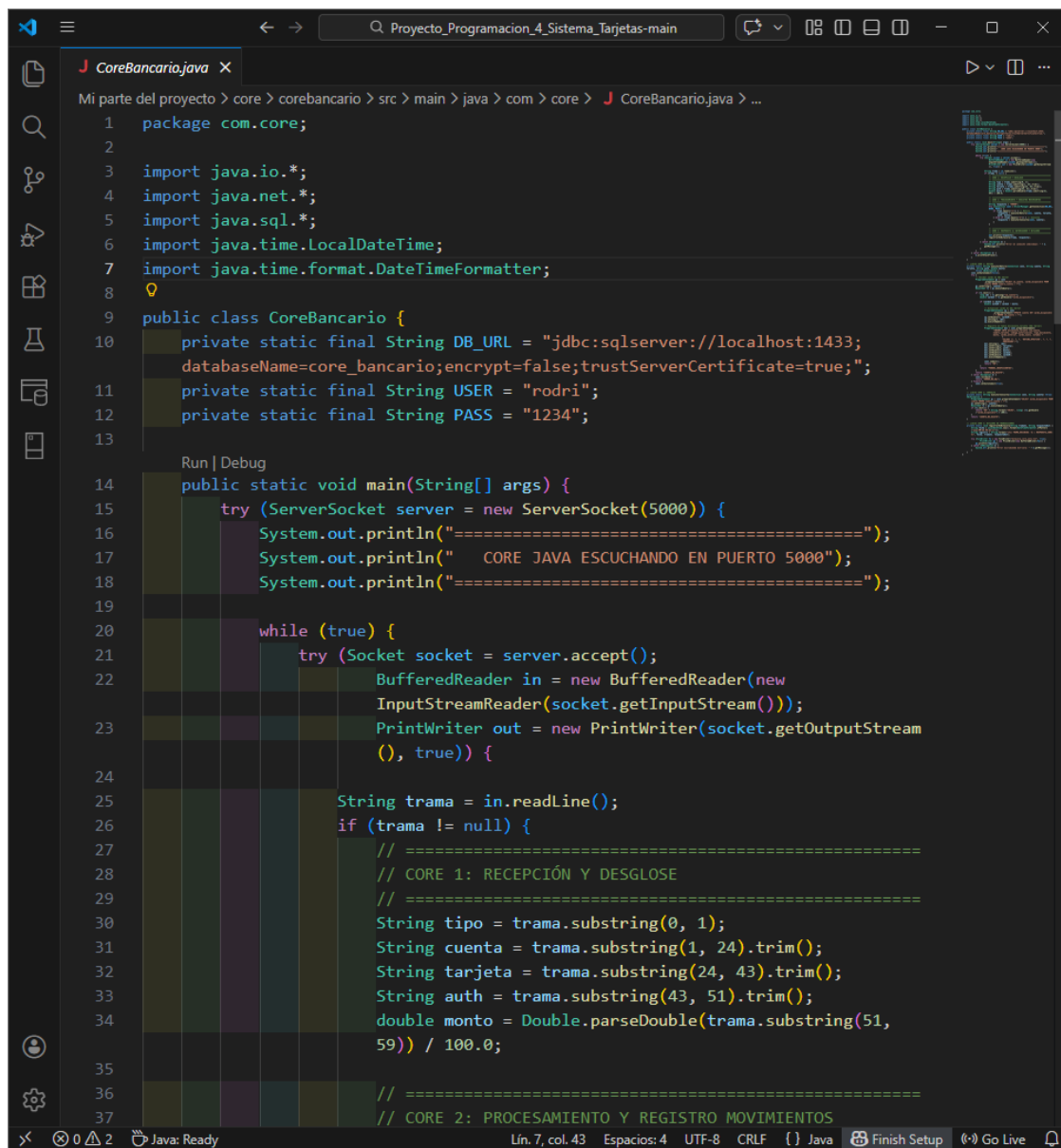
```
Archivo  Editar  Selección  ...  Q AUT  08  0  0  0  -  □  ×

seguridad.py  servidor.py  config.py  AUT5.py  AUT4.py  AUT3.py  AUT1_AUT2.py  ▶  ▢  ...

AUT5.py  >  procesar_confirmacion_aut5
8  def procesar_confirmacion_aut5(trama_json):
74      s_java.settimeout(5)
75      s_java.connect(('127.0.0.1', PUERTO_CORE_JAVA))
76      s_java.sendall((trama_core + "\n").encode())
77      resp_core = s_java.recv(1024).decode().strip()
78
79      print(f"DEBUG: Trama enviada -> {trama_core}")
80      print(f"DEBUG: El Core Java respondió -> '{resp_core}'")
81
82      if "OK" in resp_core.upper():
83          registrar_evento_aut4(n_tarjeta, datos['id_cajero'],
                                "CONFIRMACION_RETIRO", monto)
84          return {"estado": "OK", "codigo_autorizacion": datos['cod_auth']}
85      else:
86          return {"estado": "ERROR", "mensaje": f"Core respondió: {resp_core}"}
87
88      except Exception as e:
89          print(f"Error de conexión con Java: {e}")
90          return {"estado": "ERROR", "mensaje": "Core no disponible"}
91
92      else: # CRÉDITO
93          cursor.execute("""
94              INSERT INTO movimiento (id_tarjeta, codigo_autorizacion, tipo_movimiento,
95                                      monto, estado)
96              VALUES (%s, %s, 'RETIRO_CONFIRMADO', %s, 'PENDIENTE')
97              """, (tarjeta_db['id_tarjeta'], datos['cod_auth'], monto))
98          conn.commit()
99          registrar_evento_aut4(n_tarjeta, datos['id_cajero'],
                                "CONFIRMACION_CREDITO_PENDIENTE", monto)
100         return {"estado": "OK", "codigo_autorizacion": datos['cod_auth']}
101
102     except Exception as e:
103         return {"estado": "ERROR", "mensaje": str(e)}
104     finally:
105         if conn and conn.is_connected():
106             cursor.close()
107             conn.close()
```

0 0 0 Lín. 104, col. 13 Espacios: 4 UTF-8 CRLF {} Python 3.14.0 Go Live

Capítulo 4: Componentes del Core Bancario



```
1 package com.core;
2
3 import java.io.*;
4 import java.net.*;
5 import java.sql.*;
6 import java.time.LocalDateTime;
7 import java.time.format.DateTimeFormatter;
8
9 public class CoreBancario {
10     private static final String DB_URL = "jdbc:sqlserver://localhost:1433;
11         databaseName=core_bancario;encrypt=false;trustServerCertificate=true;";
12     private static final String USER = "rodri";
13     private static final String PASS = "1234";
14
15     public static void main(String[] args) {
16         try (ServerSocket server = new ServerSocket(5000)) {
17             System.out.println("=====");
18             System.out.println(" CORE JAVA ESCUCHANDO EN PUERTO 5000");
19             System.out.println("=====");
20
21             while (true) {
22                 try (Socket socket = server.accept();
23                     BufferedReader in = new BufferedReader(new
24                         InputStreamReader(socket.getInputStream()));
25                     PrintWriter out = new PrintWriter(socket.getOutputStream(), true)) {
26
27                     String trama = in.readLine();
28                     if (trama != null) {
29                         // =====
30                         // CORE 1: RECEPCIÓN Y DESGLOSE
31                         // =====
32                         String tipo = trama.substring(0, 1);
33                         String cuenta = trama.substring(1, 24).trim();
34                         String tarjeta = trama.substring(24, 43).trim();
35                         String auth = trama.substring(43, 51).trim();
36                         double monto = Double.parseDouble(trama.substring(51,
37                             59)) / 100.0;
38
39                         // =====
40                         // CORE 2: PROCESAMIENTO Y REGISTRO MOVIMIENTOS
41                         // =====
42                     }
43                 }
44             }
45         }
46     }
47 }
```

```
9 public class CoreBancario {
14     public static void main(String[] args) {
33         String auth = trama.substring(43, 51).trim();
34         double monto = Double.parseDouble(trama.substring(51,
35                                         59)) / 100.0;
36
37         // =====
38         // CORE 2: PROCESAMIENTO Y REGISTRO MOVIMIENTOS
39         // =====
40         String respuesta = "ERROR";
41         try (Connection conn = DriverManager.getConnection(DB_URL,
42                                                         USER, PASS)) {
43             if (tipo.equals("1")) { // Retiro
44                 respuesta = ejecutarRetiro(conn, cuenta, tarjeta,
45                                         auth, monto);
46             } else if (tipo.equals("2")) { // Consulta
47                 respuesta = ejecutarConsulta(conn, cuenta);
48             }
49         }
50         // =====
51         // CORE 3: RESPUESTA AL AUTORIZADOR Y BITÁCORA
52         // =====
53         out.println(respuesta);
54         registrarEnBitacora(trama, respuesta);
55     } catch (Exception e) {
56         System.err.println("Error en conexión individual: " + e.
57                             getMessage());
58     }
59 } catch (Exception e) {
60     e.printStackTrace();
61 }
62
63 // LÓGICA CORE 2: RETIRO
64 private static String ejecutarRetiro(Connection conn, String cuenta, String
65                                     tarjeta, String auth, double monto)
66                                     throws SQLException {
67     conn.setAutoCommit(false);
68     try {
```

```
Projecto_Programacion_4_Sistema_Tarjetas-main
CoreBancario.java
Mi parte del proyecto > core > corebancario > src > main > java > com > core > CoreBancario.java > ...

9 public class CoreBancario {
62
63 // LÓGICA CORE 2: RETIRO
64 private static String ejecutarRetiro(Connection conn, String cuenta, String
tarjeta, String auth, double monto)
65 throws SQLException {
66 conn.setAutoCommit(false);
67 try {
68 // Validar saldo en SQL Server
69 PreparedStatement ps = conn
70 .prepareStatement("SELECT id_cuenta, saldo_disponible FROM
cuenta WHERE numero_cuenta = ?");
71 ps.setString(1, cuenta);
72 ResultSet rs = ps.executeQuery();
73
74 if (rs.next()) {
75 long idC = rs.getLong("id_cuenta");
76 double saldoA = rs.getDouble("saldo_disponible");
77
78 if (saldoA >= monto) {
79 double saldoN = saldoA - monto;
80
81 // Actualizar saldo en SQL Server
82 PreparedStatement up = conn
83 .prepareStatement("UPDATE cuenta SET saldo_disponible
= ? WHERE id_cuenta = ?");
84 up.setDouble(1, saldoN);
85 up.setLong(2, idC);
86 up.executeUpdate();
87
88 // Registro en Tabla movimiento_tarjeta (SQL Server)
89 PreparedStatement mov = conn.prepareStatement(
90 "INSERT INTO movimiento_tarjeta (id_cuenta,
numero_tarjeta, codigo_autorizacion, tipo_movimiento,
monto, saldo_anterior, saldo_nuevo, estado) "
91 +
92 "VALUES (?, ?, ?, 'RETIRO_EFECTIVO', ?, ?, ?,
'PROCESADO')");
93 mov.setLong(1, idC);
94 mov.setString(2, tarjeta);
95 mov.setString(3, auth);
96 mov.setDouble(4, monto);

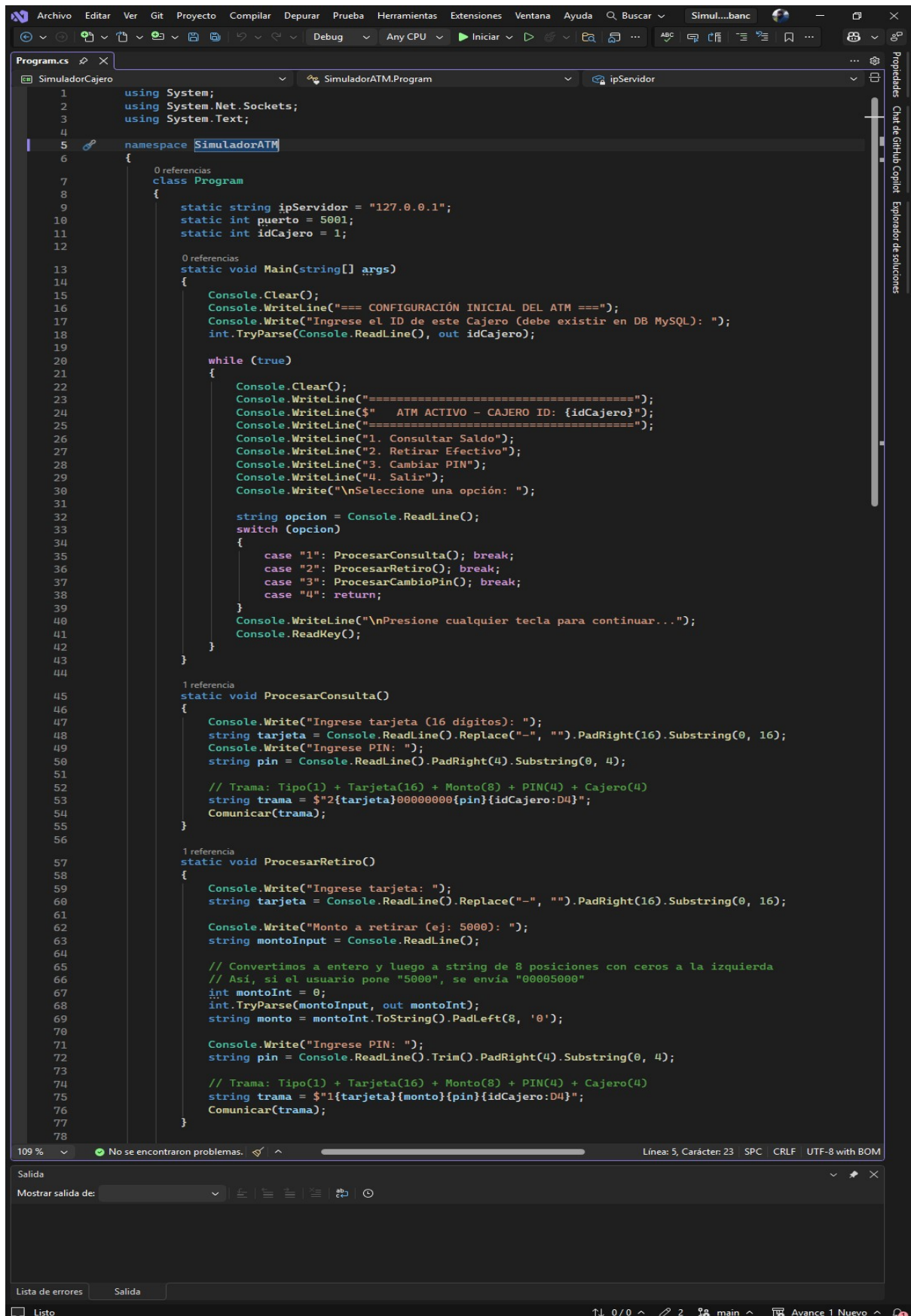
```

```
CoreBancario.java X
Mi parte del proyecto > core > corebancario > src > main > java > com > core > J CoreBancario.java > ...
9 public class CoreBancario {
64 private static String ejecutarRetiro(Connection conn, String cuenta, String
VALUES (?,?,?, ?, RETIRO_EFECTIVO , ?, ?, ?,
'PROCESADO'");
93 mov.setLong(1, idC);
94 mov.setString(2, tarjeta);
95 mov.setString(3, auth);
96 mov.setDouble(4, monto);
97 mov.setDouble(5, saldoA);
98 mov.setDouble(6, saldoN);
99 mov.executeUpdate();
100
101 conn.commit();
102 return "OK";
103 }
104 return "FONDOS_INSUFICIENTES";
105 }
106 return "CUENTA_NO_EXISTE";
107 } catch (Exception e) {
108 conn.rollback();
109 return "ERROR_DB_SQL";
110 } finally {
111 conn.setAutoCommit(true);
112 }
113 }
114
115 // LÓGICA CORE 2: CONSULTA
116 private static String ejecutarConsulta(Connection conn, String cuenta) throws
SQLException {
117 PreparedStatement ps = conn.prepareStatement("SELECT saldo_disponible FROM
cuenta WHERE numero_cuenta = ?");
118 ps.setString(1, cuenta);
119 ResultSet rs = ps.executeQuery();
120 if (rs.next()) {
121 return "OK" + String.format("%019d", (long) (rs.getDouble
("saldo_disponible") * 100));
122 }
123 return "CUENTA_NO_EXISTE";
124 }
125
126 // LÓGICA CORE 3: BITÁCORA DE TRANSACCIONES
127 private static void registrarEnBitacora(String tramaIn, String respuestaOut) {
```

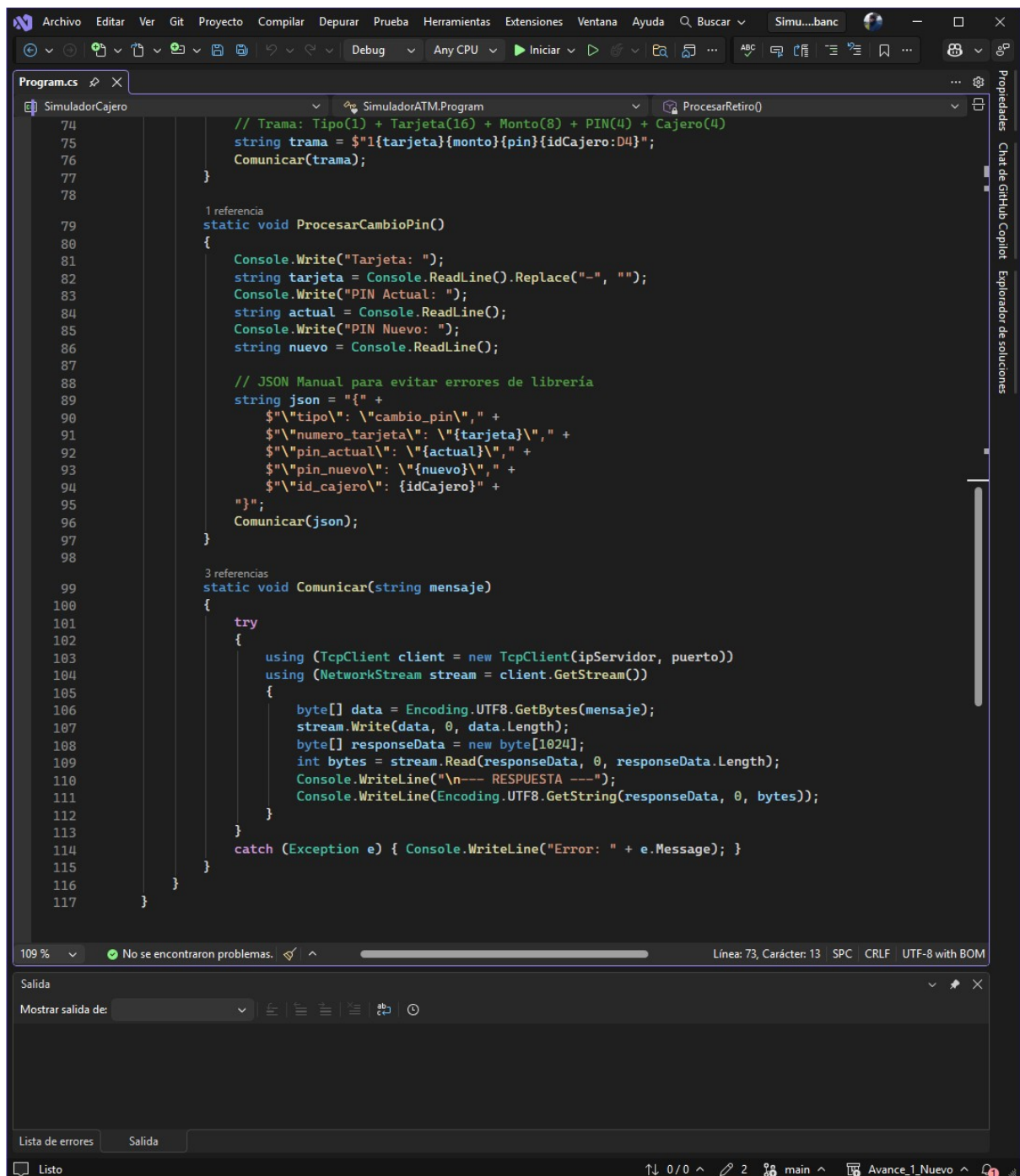
```
Projecto_Programacion_4_Sistema_Tarjetas-main
CoreBancario.java
Mi parte del proyecto > core > corebancario > src > main > java > com > core > CoreBancario.java > CoreBancario

9 public class CoreBancario {
113 }
114
115 // LÓGICA CORE 2: CONSULTA
116 private static String ejecutarConsulta(Connection conn, String cuenta) throws
    SQLException {
117     PreparedStatement ps = conn.prepareStatement("SELECT saldo_disponible FROM
        cuenta WHERE numero_cuenta = ?");
118     ps.setString(1, cuenta);
119     ResultSet rs = ps.executeQuery();
120     if (rs.next()) {
121         return "OK" + String.format("%019d", (long) (rs.getDouble
            ("saldo_disponible") * 100));
122     }
123     return "CUENTA_NO_EXISTE";
124 }
125
126 // LÓGICA CORE 3: BITÁCORA DE TRANSACCIONES
127 private static void registrarEnBitacora(String tramaIn, String respuestaOut) {
128     String fecha = LocalDateTime.now().format(DateTimeFormatter.ofPattern
        ("yyyy-MM-dd HH:mm:ss"));
129     String registro = String.format("[%s] TRAMA_RECIBIDA: %s | RESPUESTA_CORE:
        %s", fecha, tramaIn, respuestaOut);
130
131     try (FileWriter fw = new FileWriter("bitacora_core_java.txt", true);
132         PrintWriter pw = new PrintWriter(new BufferedWriter(fw))) {
133         pw.println(registro);
134     } catch (IOException e) {
135         System.err.println("Error escribiendo bitácora: " + e.getMessage());
136     }
137 }
138 }
```

Capítulo 5: Simulador de Transacciones



```
1 using System;
2 using System.Net.Sockets;
3 using System.Text;
4
5 namespace SimuladorATM
6 {
7     0 referencias
8     class Program
9     {
10         static string ipServidor = "127.0.0.1";
11         static int puerto = 5001;
12         static int idCajero = 1;
13
14         0 referencias
15         static void Main(string[] args)
16         {
17             Console.Clear();
18             Console.WriteLine("=== CONFIGURACIÓN INICIAL DEL ATM ===");
19             Console.WriteLine("Ingrese el ID de este Cajero (debe existir en DB MySQL): ");
20             int.TryParse(Console.ReadLine(), out idCajero);
21
22             while (true)
23             {
24                 Console.Clear();
25                 Console.WriteLine("=====");
26                 Console.WriteLine($"ATM ACTIVO - CAJERO ID: {idCajero}");
27                 Console.WriteLine("=====");
28                 Console.WriteLine("1. Consultar Saldo");
29                 Console.WriteLine("2. Retirar Efectivo");
30                 Console.WriteLine("3. Cambiar PIN");
31                 Console.WriteLine("4. Salir");
32                 Console.WriteLine("\nSeleccione una opción: ");
33
34                 string opcion = Console.ReadLine();
35                 switch (opcion)
36                 {
37                     case "1": ProcesarConsulta(); break;
38                     case "2": ProcesarRetiro(); break;
39                     case "3": ProcesarCambioPin(); break;
40                     case "4": return;
41                 }
42                 Console.WriteLine("\nPresione cualquier tecla para continuar...");
43                 Console.ReadKey();
44             }
45
46             1 referencia
47             static void ProcesarConsulta()
48             {
49                 Console.WriteLine("Ingrese tarjeta (16 dígitos): ");
50                 string tarjeta = Console.ReadLine().Replace("-", "").PadRight(16).Substring(0, 16);
51                 Console.WriteLine("Ingrese PIN: ");
52                 string pin = Console.ReadLine().PadRight(4).Substring(0, 4);
53
54                 // Trama: Tipo(1) + Tarjeta(16) + Monto(8) + PIN(4) + Cajero(4)
55                 string trama = $"2{tarjeta}00000000{pin}{idCajero:D4}";
56                 Comunicar(trama);
57             }
58
59             1 referencia
60             static void ProcesarRetiro()
61             {
62                 Console.WriteLine("Ingrese tarjeta: ");
63                 string tarjeta = Console.ReadLine().Replace("-", "").PadRight(16).Substring(0, 16);
64
65                 Console.WriteLine("Monto a retirar (ej: 5000): ");
66                 string montoInput = Console.ReadLine();
67
68                 // Convertimos a entero y luego a string de 8 posiciones con ceros a la izquierda
69                 // Así, si el usuario pone "5000", se envía "00005000"
70                 int montoInt = 0;
71                 int.TryParse(montoInput, out montoInt);
72                 string monto = montoInt.ToString().PadLeft(8, '0');
73
74                 Console.WriteLine("Ingrese PIN: ");
75                 string pin = Console.ReadLine().Trim().PadRight(4).Substring(0, 4);
76
77                 // Trama: Tipo(1) + Tarjeta(16) + Monto(8) + PIN(4) + Cajero(4)
78                 string trama = $"1{tarjeta}{monto}{pin}{idCajero:D4}";
79                 Comunicar(trama);
80             }
81         }
82     }
83 }
```

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?](https://www.youtube.com/playlist?list=PLoTnWByJggZUXdGv6aqRVkbR6qtVecZBG)

[list=PLoTnWByJggZUXdGv6aqRVkbR6qtVecZBG](https://www.youtube.com/playlist?list=PLoTnWByJggZUXdGv6aqRVkbR6qtVecZBG)