Introducción

El Proyecto 2 tiene como propósito extender el alcance del primer proyecto, llevando el diseño visual desarrollado anteriormente hacia una implementación funcional de backend mediante la creación de una API REST. Los estudiantes deberán construir una arquitectura básica que permita la comunicación entre el frontend y una base de datos relacional.

La actividad busca que los estudiantes comprendan los principios fundamentales del desarrollo de servicios web, incluyendo la estructuración de rutas, el manejo de peticiones y respuestas, la autenticación mediante API Key y JWT, y la integración con procedimientos almacenados (Stored Procedures) que aseguren la integridad de los datos y la correcta aplicación de reglas de negocio.

Objetivo general

Desarrollar una API REST funcional y segura que sirva como backend para el Proyecto 1, implementando autenticación, control de roles y conexión con una base de datos relacional mediante Stored Procedures, de forma que los componentes visuales creados anteriormente puedan interactuar con datos reales.

Objetivos específicos

- Diseñar e implementar una base de datos relacional que soporte las operaciones del sistema bancario, incluyendo usuarios, cuentas, tarjetas y movimientos.
- Construir endpoints REST que permitan realizar operaciones de creación, lectura, actualización y eliminación (CRUD) sobre las distintas entidades del sistema.
- Implementar mecanismos de autenticación y autorización utilizando API Key para accesos iniciales y JWT (JSON Web Token) para las operaciones protegidas.
- Validar la integridad de los datos en cada solicitud, asegurando que los clientes solo puedan acceder a la información que les pertenece, devolviendo los errores adecuados ante intentos no autorizados.
- Integrar los endpoints con Stored Procedures (SPs) para garantizar la correcta manipulación de los datos en la base de datos.
- Garantizar respuestas estructuradas y consistentes, utilizando códigos de estado HTTP y mensajes JSON uniformes.
- Desplegar la API en un entorno público o simulador en línea que permita su acceso remoto desde el frontend del Proyecto 1.

Parte 1 – Construcción de un de una BD

En esta primera parte del proyecto, los estudiantes deberán diseñar e implementar una base de datos relacional que sirva como soporte para las operaciones del sistema bancario definido en el Proyecto 1.

La base de datos será la encargada de almacenar toda la información relacionada con los usuarios, cuentas, tarjetas, movimientos, autenticación y seguridad del sistema, garantizando integridad, consistencia y trazabilidad de los datos.

La estructura deberá modelarse de acuerdo con los requerimientos funcionales descritos, aplicando los principios de normalización, uso adecuado de claves primarias y foráneas, y definiciones de tipos de datos coherentes con el contexto de cada entidad.

Además, todas las operaciones críticas sobre los datos deberán estar respaldadas por Stored Procedures (SPs) que manejen las validaciones, transacciones y reglas de negocio definidas. Estos procedimientos serán utilizados posteriormente por la API REST para realizar operaciones CRUD seguras y estructuradas.

En los anexos de este enunciado se presentan las tablas que conforman el modelo de base de datos y los procedimientos almacenados (SPs) requeridos para la implementación. Dichos anexos deberán ser utilizados como guía de referencia obligatoria para el desarrollo de esta parte del proyecto.

Requisitos generales:

- Incluir un diagrama entidad–relación (ER) o esquema visual que muestre la estructura general de la base de datos.
- Entregar los scripts DDL (creación de tablas) y los SPs en archivos separados, bien comentados y organizados.
- La base de datos deberá estar disponible en línea o accesible remotamente mediante una plataforma gratuita o pública

Parte 2 – Construcción de un API

En esta segunda parte del proyecto, los estudiantes deberán desarrollar una API REST funcional y segura que permita conectar el mockup visual del Proyecto 1 con la base de datos construida en la Parte 1 de este enunciado.

El objetivo es que los componentes de la interfaz puedan realizar operaciones reales sobre los datos mediante peticiones HTTP, aplicando los principios de diseño de APIs modernas y buenas prácticas de programación backend. La API deberá implementar las operaciones de creación, lectura, actualización y eliminación (CRUD) sobre las entidades principales del sistema bancario, incluyendo usuarios, cuentas, tarjetas y movimientos.

Además, será responsable de gestionar la autenticación y autorización de los usuarios utilizando API Keys y JSON Web Tokens (JWT), garantizando el acceso controlado a los recursos y la protección de los datos sensibles.

Todas las rutas deberán estar documentadas y respaldadas por los Stored Procedures (SPs) definidos en la base de datos, asegurando que la lógica de negocio se ejecute de manera centralizada y con integridad transaccional.

Los endpoints deberán estar estructurados bajo una arquitectura clara, manejando correctamente los códigos de estado HTTP y los mensajes de respuesta, tanto en casos exitosos como en errores.

Requisitos generales

- La API deberá construirse utilizando Node.js con Express.js o un framework equivalente que permita definir rutas REST. **No se permiten CMS**.
- Todas las rutas deben seguir una convención de versionado, utilizando el prefijo base /api/v1.
- Se debe mantener una organización modular del código (rutas, controladores, servicios, configuración, etc.).
- El servidor deberá conectarse a la base de datos creada en la Parte 1.
- Cada endpoint deberá ejecutar el Stored Procedure (SP) correspondiente para realizar las operaciones sobre los datos.

- Los endpoints iniciales (login, forgot-password, reset-password) deberán estar protegidos mediante API Key.
- Las demás operaciones deberán usar JWT (JSON Web Token) para autenticar al usuario.
- Se deben implementar middlewares de validación que verifiquen la validez del token y el rol del usuario (admin o client).
- Los endpoints que consulten datos de un cliente mediante un id (userId, accountId, cardId) deberán validar que el recurso solicitado pertenezca al usuario autenticado.
    - Si no pertenece, la API deberá responder con un error 403 (Forbidden).
- Los tokens deben tener expiración y ser firmados con una clave secreta almacenada de forma segura.
- El rol admin tendrá acceso completo a las operaciones de gestión (crear, editar, eliminar).
- El rol client solo podrá consultar información propia y realizar operaciones permitidas (lectura, solicitudes de OTP, vista temporal de PIN/CVV).
- Las respuestas deberán enviarse en formato JSON.
- Cada operación deberá retornar el código de estado HTTP correspondiente (200, 201, 400, 401, 403, 404, 500).
- En caso de error, se deberá devolver un objeto con el formato estándar
    - No deben incluirse mensajes de error genéricos del motor de base de datos.
- Deberán implementarse todos los endpoints definidos en los anexos de este enunciado, incluyendo aquellos protegidos con API Key y JWT.
- Cada ruta deberá ejecutar su SP asociado, realizando validaciones previas de entrada y manejo correcto de los resultados.
- La API deberá estar **desplegada en un entorno público o accesible remotamente**
- Debe entregarse una colección Postman o Insomnia que contenga las pruebas de todos los endpoints implementados.
- Incluir un archivo README.md con instrucciones para ejecutar el proyecto localmente, variables de entorno y dependencias utilizadas.

Instituto Tecnológico de Costa Rica
Escuela de Computación
IC8057 - Introducción al Desarrollo de Páginas Web
Proyecto 2

Lista de endpoints requeridos:

Categoría	Método	Ruta	Descripción	SP asociado
Autenticación y OTP	POST	/api/v1/auth/login	Autentica al usuario con su correo o nombre de usuario y contraseña, generando un JWT si las credenciales son válidas.	sp_auth_user_get_by_username_or_email, sp_api_key_is_active
Autenticación y OTP	POST	/api/v1/auth/forgot-password	Inicia el flujo de recuperación de contraseña generando un código OTP para el usuario.	sp_otp_create
Autenticación y OTP	POST	/api/v1/auth/verify-otp	Verifica y consume un OTP válido para el propósito indicado.	sp_otp_consume
Autenticación y OTP	POST	/api/v1/auth/reset-password	Resetea la contraseña del usuario tras verificar un OTP válido.	sp_otp_consume
Usuarios	POST	/api/v1/users	Crea un nuevo usuario (admin o cliente), validando unicidad de identificación, correo y nombre de usuario.	sp_users_create
Usuarios	GET	/api/v1/users/:identification	Consulta los datos de un usuario por su identificación (solo admin o cliente dueño).	sp_users_get_by_identification

Usuarios	PUT	/api/v1/users/:id	Actualiza la información de un usuario existente (solo admin).	sp_users_update
Usuarios	DELETE	/api/v1/users/:id	Elimina un usuario y sus registros asociados (solo admin).	sp_users_delete
Cuentas	POST	/api/v1/accounts	Crea una cuenta bancaria asociada a un usuario, indicando tipo, moneda y saldo inicial.	sp_accounts_create
Cuentas	GET	/api/v1/accounts	Lista las cuentas de un usuario específico (admin o cliente dueño).	sp_accounts_get
Cuentas	GET	/api/v1/accounts/:accountId	Consulta el detalle de una cuenta específica (admin o cliente dueño).	sp_accounts_get
Cuentas	POST	/api/v1/accounts/:accountId/status	Cambia el estado de una cuenta (activa, bloqueada, cerrada) validando reglas de saldo.	sp_accounts_set_status
Cuentas	GET	/api/v1/accounts/:accountId/movements	Lista los movimientos de una cuenta con filtros de fecha, tipo y búsqueda.	sp_account_movements_list
Transferencias	POST	/api/v1/transfers/internal	Realiza una transferencia entre cuentas del mismo banco, validando saldo,	sp_transfer_create_internal

			moneda y propiedad del cliente. Genera movimientos en cuenta origen y destino.	
Tarjetas	POST	/api/v1/cards	Crea una nueva tarjeta asociada a un usuario, almacenando PIN y CVV cifrados.	sp_cards_create
Tarjetas	GET	/api/v1/cards	Lista todas las tarjetas de un usuario (admin o cliente dueño).	sp_cards_get
Tarjetas	GET	/api/v1/cards/:cardId	Consulta el detalle de una tarjeta específica (admin o cliente dueño).	sp_cards_get
Tarjetas	GET	/api/v1/cards/:cardId/movements	Lista los movimientos de una tarjeta con filtros de fecha, tipo y búsqueda.	sp_card_movements_list
Tarjetas	POST	/api/v1/cards/:cardId/movements	Inserta un movimiento de tarjeta (compra o pago) validando el límite de crédito.	sp_card_movement_add
PIN / CVV (Flujos sensibles)	POST	/api/v1/cards/:cardId/otp	Genera un OTP para permitir una visualización temporal de PIN o CVV.	sp_otp_create
PIN / CVV	POST	/api/v1/cards/:cardId/view-details	Verifica OTP y permite visualizar el PIN temporalmente (sin registrar logs).	sp_otp_consume

Validación de cuentas (mismo banco)	POST	/api/v1/bank/validate-account	Verifica si una cuenta IBAN existe en el banco y devuelve información básica del titular.	sp_bank_validate_account

Rubrica de evaluación

La evaluación del laboratorio se realizará bajo los siguientes criterios:

Criterio de evaluación	Nivel 1 – Deficiente (1)	Nivel 3 – Satisfactorio (3)	Nivel 5 – Excelente (5)	%
Estructura y conexión de la base de datos	La base de datos no cumple con la estructura ni las relaciones establecidas; presenta errores graves en integridad o tipos de datos.	La base de datos está parcialmente implementada; cumple con las entidades principales, pero faltan algunas relaciones, validaciones o SPs.	La base de datos está completamente implementada, normalizada, con relaciones y SPs funcionando correctamente según los anexos.	15%
Implementación de endpoints REST	No se implementan los endpoints requeridos o no son funcionales.	Se implementa la mayoría de los endpoints definidos, aunque algunos presentan errores de conexión o validación.	Todos los endpoints están implementados y funcionales, ejecutan correctamente sus SPs y cumplen con la estructura REST solicitada.	20%
Autenticación y seguridad (API Key / JWT)	No se implementa autenticación o la API permite acceso libre sin control.	Se utiliza autenticación parcial (solo API Key o JWT) o con errores de validación en los roles.	Se implementa correctamente el uso de API Key y JWT con validación de roles y propiedad de recursos. Los accesos están protegidos y los tokens expiran adecuadamente.	15%
Validación de integridad y roles	No existe validación de propiedad; los usuarios pueden acceder a datos de otros.	Se valida parcialmente la propiedad, aunque algunos endpoints carecen de control de acceso.	Todos los endpoints validan la integridad y propiedad del recurso, devolviendo 403 Forbidden ante intentos no autorizados.	10%
Seguridad de la información	Contraseñas, PIN o CVV se almacenan o envían en texto plano; no hay manejo seguro de datos.	Se implementa cifrado parcial o solo en contraseñas, sin aplicar a todos los campos sensibles.	Toda la información sensible (contraseñas, PIN, CVV, OTP) se maneja cifrada/hasheada. No se exponen ni registran datos sensibles en logs o respuestas.	10%
Manejo de errores y respuestas JSON	Las respuestas son inconsistentes o los errores no se manejan adecuadamente.	Las respuestas JSON están estructuradas, pero algunos endpoints no devuelven códigos HTTP apropiados.	Todas las rutas devuelven respuestas JSON uniformes, con manejo coherente de errores y uso correcto de códigos HTTP (200, 201, 400, 401, 403, 404, 500).	10%
Documentación técnica y organización del código	No se entrega documentación ni instrucciones claras de ejecución.	Se entrega documentación básica o incompleta; el código es entendible pero poco estructurado.	Se entrega documentación completa (README, variables de entorno, dependencias, instrucciones). El código está modularizado, bien organizado y comentado.	10%
Pruebas y despliegue funcional	No se entrega evidencia de pruebas o la API no funciona.	Se entrega colección Postman básica o el despliegue tiene errores menores.	Se entrega colección Postman completa, la API está desplegada en línea y responde correctamente a todas las pruebas.	10%

Puntos extras (3%)

De forma opcional, los equipos podrán desarrollar un módulo de auditoría del sistema, cuyo objetivo es registrar y consultar las acciones relevantes realizadas por los usuarios dentro de la API.

Este módulo permitirá evidenciar eventos como la creación, modificación o eliminación de recursos, transferencias internas, cambios de estado de cuentas o la visualización de información sensible (por ejemplo, PIN o CVV).

El propósito de este componente es reforzar las buenas prácticas de trazabilidad, control y transparencia dentro del desarrollo backend, permitiendo a los administradores del sistema conocer el historial de acciones ejecutadas.

El módulo de auditoría se compone de una tabla y Stored Procedures (SPs) específicos, los cuales se describen en los anexos de este enunciado.
A partir de esta estructura, se deberá implementar al menos un endpoint de consulta que permita visualizar los registros de auditoría correspondientes a un usuario.

Categoría	Método	Ruta	Descripción	SP asociado
Auditoría	GET	/api/v1/audit/:userId	Permite consultar el historial de acciones registradas para un usuario específico. Retorna los eventos de auditoría asociados, ordenados por fecha.	sp_audit_list_by_user

Este endpoint deberá estar protegido mediante JWT, y únicamente podrá ser accedido por usuarios con rol admin o por el propietario de los registros.

Entrega

Cada grupo deberá entregar un archivo comprimido (.zip) con el proyecto completo, siguiendo estas instrucciones:

1. Una carpeta con el repositorio completo
2. Stored Procedures (SPs) funcionales para todas las operaciones de la API.
3. Colección **Postman o Insomnia** con todos los endpoints configurados y ejemplos de prueba funcional.
4. Un archivo denominado enlaces.txt que contenga
    a. URL del enpoint API (desplegada)
    b. URL y accesos a la BD (desplegada)
    c. URL de la documentación de Postman

Equipos de trabajo: 2 personas (mismo que el proyecto anterior)
Fecha límite: Domingo 2 de noviembre a las 11:55 p.m.
Forma de entrega: El archivo .zip debe subirse a la plataforma TEC Digital, en el espacio correspondiente al curso.
Portafolio del curso: Este proyecto forma parte del portafolio individual del curso.

Todo el trabajo presentado deberá ser **original y producto del esfuerzo del equipo**. Cualquier evidencia de **plagio total o parcial**, **uso indebido de código ajeno**, o la **utilización excesiva de herramientas de inteligencia artificial** sin un aporte real del estudiante será considerada una **falta grave a la integridad académica**.

En tales casos, el proyecto recibirá una **nota de 0 (cero)** y se procederá con la **inclusión del incidente en el expediente académico** del estudiante, conforme a la normativa institucional vigente del Tecnológico de Costa Rica.

El uso de tecnologías de apoyo (incluyendo IA) solo será aceptado si se emplea de manera responsable y con **criterio académico**, como herramienta de consulta o asistencia, y **no como sustituto del trabajo propio** en la redacción del código, la documentación o los entregables evaluados.

Anexos

A continuación se definen los campos requeridos por cada tabla:

usuario

Campo	Tipo	Descripción
id	UUID	Identificador único del usuario
tipo_identificacion	UUID (FK → tipoIdentificacion.id)	Tipo de identificación
identificacion	String	Número de identificación
nombre	String	Nombre del usuario
apellido	String	Apellido del usuario
correo	String	Correo electrónico único
teléfono	String	Teléfono del usuario
usuario	String	Nombre de usuario único
contrasena_hash	String	Contraseña cifrada (bcrypt u otro algoritmo seguro)
rol	UUID (FK → rol.id)	Define permisos y acceso
fecha_creacion	DateTime	Fecha de registro
fecha_actualizacion	DateTime	Última modificación de datos

rol

Campo	Tipo	Descripción
id	UUID	Identificador único del tipo rol
nombre	String	Admin, cliente
descripcion	String	Descripcion del tipo de rol

tipoIdentificacion

Campo	Tipo	Descripción
id	UUID	Identificador único del tipo de identificación
nombre	String	Nacional, DIMEX, Pasaporte
descripcion	String	Descripcion del tipo de identificación

cuenta

Campo	Tipo	Descripción
id	UUID	Identificador único de cuenta
usuario_id	UUID (FK → usuarios.id)	Propietario de la cuenta
iban	String	Código IBAN único del banco
alias	String	Nombre asignado por el usuario a la cuenta
tipoCuenta	UUID (FK → tipoCuenta.id)	Tipo de cuenta
moneda	UUID (FK → moneda.id)	Moneda de la cuenta
saldo	Decimal(18,2)	Saldo actual de la cuenta

estado	UUID (FK → estadoCuenta.id)	Estado actual
fecha_creacion	DateTime	Fecha de apertura
fecha_actualizacion	DateTime	Última modificación

tipoCuenta

Campo	Tipo	Descripción
id	UUID	Identificador único del tipo de cuenta
nombre	String	Ahorros, Corriente
descripcion	String	Descripción del tipo de cuenta

moneda

Campo	Tipo	Descripción
id	UUID	Identificador único del tipo de moneda
nombre	String	Colones, Dólares
iso	String	CRC, USD

estadoCuenta

Campo	Tipo	Descripción
id	UUID	Identificador único del estado de cuenta
nombre	String	Activa, Bloqueada, Cerrada
descripción	String	Descripción del tipo de cuenta

movimientoCuenta

Campo	Tipo	Descripción
id	UUID	Identificador único
cuenta_id	UUID (FK → cuentas.id)	Cuenta asociada
fecha	DateTime	Fecha del movimiento
tipo	UUID (FK → tipoMovimientoCuenta.id)	Tipo de operación
descripcion	String	Motivo o detalle del movimiento
moneda	UUID (FK → moneda.id)	Moneda del movimiento
monto	Decimal(18,2)	Monto de la transacción

tipoMovimientoCuenta

Campo	Tipo	Descripción
id	UUID	Identificador único del tipo de movimiento
nombre	String	Crédito, Debito
descripción	String	Descripción del tipo de movimiento

tarjeta

Campo	Tipo	Descripción
id	UUID	Identificador único de tarjeta
usuario_id	UUID (FK → usuarios.id)	Propietario
tipo	UUID (FK → tipoTarjeta.id)	Tipo de la tarjeta
numero_enmascarado	String	Ejemplo: 1234 **** **** 5678
fecha_expiracion	String (MM/YY)	Fecha de vencimiento
cvv_hash	String	Código CVV encriptado

pin_hash	String	PIN encriptado
moneda	UUID (FK → moneda.id)	Moneda de la tarjeta
limite_credito	Decimal(18,2)	Monto máximo de crédito
saldo_actual	Decimal(18,2)	Saldo utilizado o disponible
fecha_creacion	DateTime	Fecha de emisión
fecha_actualizacion	DateTime	Última modificación

tipoTarjeta

Campo	Tipo	Descripción
id	UUID	Identificador único del tipo de cuenta
nombre	String	Ahorros, Corriente
descripcion	String	Descripción del tipo de cuenta

movimientoTarjeta

Campo	Tipo	Descripción
id	UUID	Identificador único
cuenta_id	UUID (FK → tarjetas.id)	Tarjeta asociada
fecha	DateTime	Fecha del movimiento
tipo	UUID (FK → tipoMovimientoTarjeta.id)	Tipo de operación
descripcion	String	Motivo o detalle del movimiento
moneda	UUID (FK → moneda.id)	Moneda del movimiento
monto	Decimal(18,2)	Monto de la transacción

tipoMovimientoTarjeta

Campo	Tipo	Descripción
id	UUID	Identificador único del tipo de movimiento
nombre	String	Compra, Pago
descripción	String	Descripción del tipo de movimiento

Otps

Campo	Tipo	Descripción
id	UUID	Identificador único
usuario_id	UUID (FK → usuarios.id)	Usuario asociado
codigo_hash	String	Código OTP encriptado
proposito	Enum (password_reset, card_details)	Finalidad del OTP
fecha_expiracion	DateTime	Límite de validez
fecha_consumido	DateTime (nullable)	Cuándo se usó
fecha_creacion	DateTime	Cuándo se generó

apiKey

Campo	Tipo	Descripción
id	UUID	Identificador único

clave_hash	String	Valor de la API Key encriptado
etiqueta	String	Descripción o propósito
activa	Boolean	Estado (activa/inactiva)
fecha_creacion	DateTime	Fecha de generación

Auditoria (extras)

Campo	Tipo	Descripción
id	Integer	Identificador incremental
usuario_id	UUID (FK → usuarios.id)	Usuario que ejecutó la acción
accion	String	Tipo de acción (ej: LOGIN, CAMBIO_ESTADO_CUENTA)
detalles	JSON	Información adicional
fecha	DateTime	Fecha de ejecución

Resumen de relaciones

Relación	**Tipo**
usuarios → cuentas	1:N
usuarios → tarjetas	1:N
cuentas → movimientos_cuenta	1:N
tarjetas → movimientos_tarjeta	1:N
usuarios → otps	1:N

A continuación se definen procedimientos almacenados requeridos:

Nombre del SP	Entradas (IN)	Salidas (OUT)	Descripción / Propósito
sp_auth_user_get_by_username_or_email	username_or_email	user_id, contrasena_hash, rol	Obtiene la información de un usuario (ID, hash y rol) para el proceso de login.
sp_api_key_is_active	api_key_hash	is_active (boolean)	Verifica si una API Key existe y se encuentra activa.
sp_otp_create	user_id, proposito, expires_in_seconds, codigo_hash	otp_id	Crea un OTP cifrado con propósito específico (recuperación o validación temporal).
sp_otp_consume	user_id, proposito, codigo_hash	consumed (boolean)	Valida y marca como utilizado un OTP existente, asegurando que no se reutilice.
sp_users_create	tipo_identificacion, identificacion, nombre, apellido, correo, usuario, contrasena_hash, rol	user_id	Crea un nuevo usuario (admin o cliente) verificando unicidad de los campos principales.
sp_users_get_by_identification	identificacion	id, nombre, apellido, correo, usuario, rol	Retorna la información del usuario correspondiente a la identificación indicada.
sp_users_update	user_id, campos opcionales (nombre, apellido, correo, usuario, rol)	updated (boolean)	Actualiza los datos de un usuario existente (solo accesible por admin).
sp_users_delete	user_id	deleted (boolean)	Elimina un usuario y todos los registros asociados (cuentas, tarjetas, etc.).
sp_accounts_create	usuario_id, iban, alias, tipo, moneda, saldo_inicial, estado	account_id	Crea una nueva cuenta asociada a un usuario y establece su saldo inicial.
sp_accounts_get	owner_id?, account_id?	Lista de cuentas o una cuenta	Retorna las cuentas de un usuario o los detalles de una cuenta específica.
sp_accounts_set_status	account_id, nuevo_estado	updated (boolean)	Cambia el estado de una cuenta (activa, bloqueada, cerrada) validando saldo y reglas.

sp_account_movements_list	account_id, from_date?, to_date?, type?, q?, page, page_size	items[], total, page, page_size	Devuelve los movimientos de una cuenta con paginación y filtros.
sp_cards_create	usuario_id, tipo, numero_enmascarado, fecha_expiracion, cvv_encriptado, pin_encriptado, moneda, limite_credito, saldo_actual	card_id	Crea una tarjeta vinculada a un usuario con datos sensibles cifrados.
sp_cards_get	owner_id?, card_id?	Lista de tarjetas o una tarjeta	Obtiene las tarjetas asociadas a un usuario o una tarjeta específica.
sp_card_movements_list	card_id, from_date?, to_date?, type?, q?, page, page_size	items[], total, page, page_size	Devuelve los movimientos de una tarjeta con paginación y filtros.
sp_card_movement_add	card_id, fecha, tipo, descripcion, moneda, monto	movement_id, nuevo_saldo_tarjeta	Inserta un movimiento en una tarjeta (compra/pago) y actualiza su saldo.
sp_transfer_create_internal	from_account_id, to_account_id, amount, currency, description, user_id	transfer_id, receipt_number, status	Realiza una transferencia entre cuentas del mismo banco. Valida saldo, moneda y propiedad; genera movimientos en ambas cuentas.
sp_bank_validate_account	iban	exists (boolean), owner_name, owner_id	Verifica si una cuenta IBAN pertenece al banco y devuelve los datos básicos del titular.
sp_audit_log *(opcional / puntos extra)*	actor_user_id?, accion, entidad, entidad_id?, detalles_json	audit_id	Registra eventos críticos (cambio de estado, transferencias, vista de PIN/CVV, etc.).