Iteración 3 – Sistemas transaccionales

Alejandro Ahogado Prieto, Laura G. Quiroga Sánchez

Universidad de los Andes, Bogotá, Colombia

{a.ahogado, l.quiroga} @uniandes.edu.co

Fecha de presentación: noviembre 15 del 2021

Contenido

1. Análisis y modelo conceptual

Nombre	RF1. Registrar Usuario
Resumen	Un usuario administrador puede registrar usuarios de tipo empleado y un gerente de oficina puede registrar clientes en el sistema.
Entradas	
Tipo de documento	
Numero documento	
Departamento	
Código Postal	
Nacionalidad	
Ciudad	
Nombre	
Dirección	
Login	
Contraseña	
Correo	
Teléfono	
Resultados	
El usuario es registrado en el sistema con su información asociada y su tipo de usuario.	
RNF asociados	
Seguridad	
Persistencia	
Privacidad	

1.1 Requerimientos funcionales

Figura 1. Requerimiento funcional 1 – Registrar usuario

Este requerimiento es **transaccional** ya que maneja información sensible para el funcionamiento del banco, como lo es el registro de sus clientes y empleados. En caso de que el requerimiento no se realice de manera atómica los datos del usuario no quedarán registrados en su totalidad en el sistema y en caso de que no se realice de manera consistente, la información puede llegar a no ser correcta, causando fallas en el funcionamiento de otras operaciones. Por otro lado, el sistema debe permitir que varios usuarios sean registrados a la vez de manera aislada, tal que puedan realizar su registro simultáneamente y sin errores, de lo contrario el usuario no quedará registrada como parte del negocio. Por último, mientras el usuario sea parte del negocio debe poder consultar y modificar su información personal, por lo que su registro debe persistir en el sistema. De no cumplirse los criterios de transaccionalidad, habría repercusiones graves en el funcionamiento del negocio, al punto de perder un

Nombre	RF2. Registrar Oficina
Resumen	Un usuario administrador puede registrar una oficina en el sistema
Entradas	
Nombre	
Dirección	
Número de puestos de atención	
Gerente	
Resultados	
La oficina es registrada en el sistema con su información asociada y sus puntos de atención.	
RNF asociados	
Seguridad	
Persistencia	

cliente si no se le permite realizar las operaciones que necesite o de impedir que un empleado ejerza sus funciones.

Figura 2. Requerimiento funcional 2 – Registrar oficina

Este requerimiento **no es transaccional** ya que, si falla la operación de registrar una oficina,

Nombre	RF3. Registrar punto de atención
Resumen	Un usuario administrador puede registrar puntos de atención e ingresar la información correspondiente su tipo.
Entradas	
Tipo	
Localización	
Oficina (si es punto físico)	
Resultados	
El punto de atención es registrado en el sistema, con su tipo y su información asociada.	
RNF asociados	
Seguridad	
Persistencia	

se puede volver a ejecutar la operación sin tener grandes consecuencias como la perdida de información asociada a un cliente, teniendo en cuenta que al momento de registrarla solo se están ingresando datos pertenecientes a la oficina.

Figura 3. Requerimiento funcional 3 – Registrar punto de atención

Este requerimiento **no es transaccional** puesto que, en caso de no realizarse de forma atómica o consistente, el fallo no tendrá graves repercusiones para el negocio más allá de obligarlo a volver a realizar el registro del punto de atención o actualizar la información incorrecta. Sin embargo, la realización de esta operación debe realizarse de manera concurrente con las demás operaciones de BancAndes sin afectarlas y sin verse afectada por las demás. Además, el registro de la información del punto de atención debe persistir en la base de datos, tal que los empleados que trabajan allí puedan asociarse a éste y al igual que las operaciones bancarias que se realicen en él.

Nombre	RF4. Registrar cuenta
Resumen	Un gerente de oficina puede abrir la cuenta de un cliente en una oficina del banco.
Entradas	
Tipo	
Cliente	
Número de cuenta	
valor	
fecha	
Gerente Oficina	
Resultados	
La cuenta del cliente es registrada en el sistema con un estado de desactivada.	
RNF asociados	
Seguridad	
Persistencia	
Privacidad	

Figura 4. Requerimiento funcional 4 – Registrar cuenta

Este requerimiento funcional es **transaccional** ya que maneja información importante tanto para el banco como para el cliente al querer registrar una nueva cuenta. Si este requerimiento no se realiza de manera atómica, no se asegura que todos los datos del cliente relacionados con la cuenta queden guardados, así mismo pueden generarse errores en las operaciones relacionadas con la misma si los datos de la cuenta no son consistentes. Dado que el banco tiene diferentes sucursales, es importante que se puedan registrar cuentas al mismo tiempo cumpliendo el criterio de aislamiento. Finalmente, todos los registros de una cuenta, sin importar el estado en el que esta se encuentre, deben quedar de forma permanente en la base de datos. Si los criterios *ACID* (*Atomicity, Consistency, Isolation, Durability*) no se cumplen en este requerimiento, se puede estar perdiendo información importante del cliente y de sus operaciones.

Nombre	RF5. Cerrar cuenta
Resumen	Un gerente de oficina puede cerrar la cuenta de un cliente bajo su solicitud.
Entradas	
Cliente	
Número de cuenta	
fecha	
Gerente de oficina	
Resultados	
La cuenta del cliente es actualizada en el sistema con un estado de cerrada y un saldo de cero.	
RNF asociados	
Seguridad	
Persistencia	
Privacidad	

Figura 5. Requerimiento funcional 5 – Cerrar cuenta

El requerimiento de cerrar una cuenta es **transaccional** ya que se trata de una operación bancaria vital para el negocio y para el cliente. La operación consiste en una serie de pasos que debe realizarse todos o ninguno y debe garantizar la correctitud y completitud del registro de la operación en la base de datos, esto significa que debe ser tanto atómica como consistente. En caso de no cumplir con estos criterios de transaccionalidad es posible que el cliente pueda continuar haciendo operaciones sobre una cuenta que debería figurar en la base de datos como cerrada, esto puede ser grave para el negocio en el caso de que el banco le siga pagando los rendimientos al cliente por la cuenta que ha cerrado sin posibilidad de hacer ningún reclamo después. Además, dado el volumen de clientes, cuentas, puestos de atención y operaciones bancarias, es necesario que el requerimiento se realice de forma aislada, de manera que varias operaciones se ejecuten simultáneamente sin afectar a las demás. Más aún, una vez realizada la operación de cierre, la información relacionada a la operación y la actualización de la información de la cuenta debe persistir y sobrevivir aún en caso de haber fallos en el sistema.

Nombre	RF6. Registrar operación sobre cuenta
Resumen	Un cliente o un cajero pueden registrar operaciones sobre una cuenta del cliente.
Entradas	
Cliente (login)	
Número de cuenta	
valor	
fecha	
Tipo de operación	
Puesto de atención	
Empleado	
Resultados	
La operación es registrada en el sistema y la información de la cuenta asociada es actualizada.	
RNF asociados	
Concurrencia	
Persistencia	
Privacidad	
Distribución	
Seguridad	

Figura 6. Requerimiento funcional 6 – Registrar operación sobre cuenta

Registrar operación sobre una cuenta es un requerimiento **transaccional**, ya que se está modificando información importante de la cuenta asociada a un cliente. Esta operación debe ser atómica, ya que al tratarse de una transacción que actualiza el saldo de una cuenta, no debe ser posible que la transacción no se finalice completamente y por ejemplo se sume dinero en una cuenta sin restarse en otra. Así mismo, se debe cumplir con el criterio de consistencia y durabilidad ya que, al finalizar cada transacción u operación, se debe guardar la información actualizada, siempre y cuando siga respetando las reglas del negocio, como por ejemplo que el saldo de una cuenta no puede ser negativo. Por otro lado, teniendo en cuenta el tipo de negocio, es importante que estas operaciones puedan ejecutarse de forma simultánea y aislada sin afectar a los demás usuarios.

Nombre	RF7. Registrar préstamo
Resumen	Un gerente de oficina registra un préstamo aprobado para un cliente.
Entradas	
Cliente (login)	
Monto	
Interés	
Numero de cuotas	
Dia de pago	
Valor cuota mínima	
Saldo pendiente	
Fecha préstamo	
Gerente Oficina	
Resultados	
El préstamo se registra en el sistema con su información asociada.	
RNF asociados	
Persistencia	
Privacidad	
Distribución	
Seguridad	

Figura 7. Requerimiento funcional 7 – Registrar préstamo

El requerimiento funcional es **transaccional** porque se trata de registrar información muy importante para el cliente y para el negocio, operación que se debe realizar siguiendo los criterios *ACID*. Por un lado, la operación se debe realizar de manera atómica y consistente, de forma que no sea posible ejecutarla a medias, registrar información incompleta o que no cumpla las reglas del negocio. En caso de que no se cumpla esto, a pesar de que se realice un préstamo al cliente, puede ocurrir que no se registre la información asociada a las fechas de pago, cuotas mínimas o intereses en su totalidad o correctamente, esto puede resultar en problemas del banco como, por ejemplo, al momento de solicitar los pagos por parte del cliente. Por esta razón, también se debe cumplir que la operación persista en la base de datos, tal que la información asociada al producto bancario quede guardada de forma permanente en el sistema y pueda ser recuperado por los usuarios en el futuro. Es importante que el registro sea durable para que los clientes puedan consultar la información asociada al préstamo y sepan cuando deben realizar los pagos, así como para que el negocio sepa el dinero que ha prestado. Además, es una operación que debe poder ser realizada en cualquier momento, de manera simultánea con otras operaciones de otros clientes sin encontrar problemas de concurrencia.

Nombre	RF8. Registrar operación sobre préstamo
Resumen	Un cajero puede registrar la información asociada a una operación sobre un préstamo del cliente.
Entradas	
Cliente	
Cajero	
Punto de atención	
Id préstamo	
Monto por abonar	
Fecha	
Resultados	
La operación es registrada en el sistema y la información del préstamo es actualizada.	
RNF asociados	
Persistencia	
Privacidad	
Distribución	
Seguridad	

Figura 8. Requerimiento funcional 8 – Registrar operación sobre préstamo

El anterior requerimiento funcional es **transaccional**, ya que en esta operación se está actualizando información importante sobre un préstamo, como lo es su saldo pendiente luego de realizar un pago. Es importante que esta operación cumpla con los criterios *ACID*, ya que debe realizarse de manera atómica y consistente, siempre y cuando cumpla con las reglas del negocio. Si no se garantiza que esta operación se ejecute cumpliendo el criterio de atomicidad, podría existir el error que un cliente realice un pago, pero este no se acredite inmediatamente a su préstamo. De igual manera es importante que este registro cumpla con el criterio de durabilidad, porque si no se le estaría cobrando al cliente dinero que ya pagó. Por último, teniendo en cuenta el tipo de negocio es importante que esta operación se pueda realizar de forma simultánea, sin afectar a otros clientes.

Nombre	RF9. Cerrar préstamo
Resumen	Un gerente de oficina registra una operación para cerrar un préstamo de un cliente
Entradas	
Cliente (número de documento, tipo de documento)	
Id Préstamo	
Gerente Oficina	
fecha	
Puesto de atención	
Resultados	
La operación es registrada en el sistema y la información del préstamo del cliente es actualizada.	
RNF asociados	
Persistencia	
Privacidad	
Distribución	
Seguridad	

Figura 9. Requerimiento funcional 9 – Cerrar préstamo

Este requerimiento funcional es **transaccional** ya que el cierre de un préstamo es una operación muy importante para el negocio y para el cliente que debe cumplir con los criterios *ACID*. Si bien la operación se realiza cuando el cliente ya pagó la deuda en su totalidad y, por lo tanto, tiene el saldo pendiente del préstamo en ceros, en caso de que no se ejecute la operación de manera atómica, consistente y durable, puede haber repercusiones negativas para el negocio y para el cliente. A pesar de que el saldo pendiente del préstamo ya esté registrado con un valor de cero y por esto no aumenten los intereses de la deuda, si no se registra correctamente el cierre del préstamo o si no persiste la información en la base de datos, por no pagar varios meses consecutivos una cuota igual o superior a la cuota mínima, el banco agregaría al cliente al fichero de morosos, lo cual puede complicar la posibilidad del cliente a acceder a otro préstamo con el banco o con otra entidad financiera. Además, al igual que todas las operaciones bancarias, es vital que cumpla el criterio de aislamiento, de manera que se pueda realizar de manera concurrente con otras operaciones sin generar ningún tipo de error.

Nombre	RFC1. Consultar las cuentas en BancAndes
Resumen	Un cliente, gerente de oficina o gerente general puede realizar consultas en el sistema según un criterio deseado y debe poder agruparlas y ordenarlas según sus intereses.
Entradas	
Criterio para filtrar: tipo de cuenta, rango de saldos, fecha de creación, fecha de último movimiento, cliente	
Resultados	
Información solicitada.	
RNF asociados	
Concurrencia	
Privacidad	
Distribución	
Seguridad	

Figura 10. Requerimiento funcional de consulta 1 – Consultar las cuentas en BancAndes

El requerimiento funcional de consultar las cuentas en BancAndes **no es transaccional** porque la información solicitada es sensible para el cliente y para el negocio, sin embargo, no debe cumplir con todos los criterios *ACID*. La operación debe ser atómica para que al momento de obtener la información asociada a la consulta el usuario reciba toda la información o ninguna. Por ejemplo, en el caso de un cliente, debe obtener toda la información de todas sus cuentas o no recibir ninguna información. Asimismo, la información debe ser consistente, no deben presentar cambios que vayan en contra de la realidad o de las reglas del negocio. Lo anterior es importante porque se trata de una consulta que puede estar relacionada a grandes cantidades de dinero, en las que un error en los saldos o fechas de movimientos puede ser fatal. Además, teniendo en cuenta el volumen de clientes, cuentas, puestos de atención y operaciones bancarias, es necesario que el requerimiento se realice de forma aislada, de manera que varias operaciones de BancAndes se ejecuten simultáneamente sin afectar a las demás. Empero, dado que la operación de consulta no involucra ningún registro ni actualización de la información del sistema, y como no se guardará ningún historial de las consultas, no se necesita que el requerimiento cumpla con el criterio de durabilidad.

Nombre	RFC2 – Consultar Cliente
Resumen	Un cliente, gerente de oficina o gerente general puede realizar una consulta sobre la información de un cliente, debe poder filtrar la información, agruparla y ordenarla según sus intereses.
Entradas	
Tipo documento	
Número de documento	
Criterio para filtrar: Tipo cuenta, Cuenta, Saldo, Operaciones en rango de fechas y valor	
Resultados	
Información solicitada.	
RNF asociados	
Concurrencia	
Privacidad	
Distribución	
Seguridad	

Figura 11. Requerimiento funcional de consulta 2 – Consultar cliente

Consultar un cliente **no es un requerimiento funcional transaccional**, ya que se está filtrando diferente tipo de información relacionada con el cliente, pero esta información no está siendo actualizada o modificada en ningún momento, por lo tanto, no se debe comparar con un estado antes o después a la operación en la base de datos. De igual manera, como se está filtrando información ya existente de la base de datos, no se debe persistir el resultado de la búsqueda y si en algún momento llega a fallar la operación, no representa ninguna consecuencia grave como la perdida de la información, sino simplemente se podría volver a realizar la operación.

Nombre	RFC3 – Consultar los 10 tipos de operaciones de mayor movimiento en el sistema por oficina, en un rango de fechas
Resumen	Un gerente de oficina o gerente general puede consultar los tipos de operaciones de mayor movimiento en el sistema por oficina y rango de fechas.
Entradas	
Oficina	
Rango de fechas	
Resultados	
La información asociada a los 10 tipos de operaciones de mayor movimiento ordenadas por el número de veces realizada.	
RNF asociados	
Concurrencia	
Privacidad	
Distribución	
Seguridad	

Figura 12. Requerimiento funcional de consulta 3 – Consultar los 10 tipos de operaciones de mayor movimiento en el sistema por oficina, en un rango de fechas

El requerimiento de consultar los 10 tipos de operaciones de mayor movimiento en el sistema por oficina, en un rango de fechas**, no es transaccional**. Esto se debe a que solo se trata de un filtro sobre los tipos de operaciones que genera un ranking por el número de veces de ejecución, con posibilidades de agrupamiento y ordenamiento, sin embargo, en caso de fallar (no ser atómico, consistente o aislado) no tendrá repercusiones graves para el negocio. Asimismo, como no se hacen modificaciones a la información del sistema, ni se debe guardar un registro de estas consultas, la operación no deberá persistir.

Nombre	RFC4. Obtener los datos del usuario más activo
Resumen	Un gerente de oficina o gerente general pueden consultar cuál es el usuario más activo en el sistema según un criterio de búsqueda (tipo de operación o valor mínimo de operación)
Entradas	
Valor por filtrar	
Tipo de operación	
Resultados	
El usuario o los usuarios más activos del sistema según el criterio de búsqueda indicado.	
RNF asociados	
Concurrencia	
Privacidad	
Distribución	
Seguridad	

Figura 13. Requerimiento funcional de consulta 4 – Obtener los datos del usuario más activo

El requerimiento funcional de obtener los datos del usuario más activo **no es transaccional** ya que, a pesar de que se está solicitando información que debe ser correcta, si esta operación llega a fallar no tiene grandes consecuencias al no estar modificando ninguna parte de los datos. De igual manera,

como solo se está filtrando información, en ningún momento el resultado de esta búsqueda debe persistir en la base de datos y en cualquier error se puede volver a ejecutar la operación.

1.2 Modelo conceptual

Se indican en color rojo los cambios realizados en el modelo con respecto a la iteración 2.

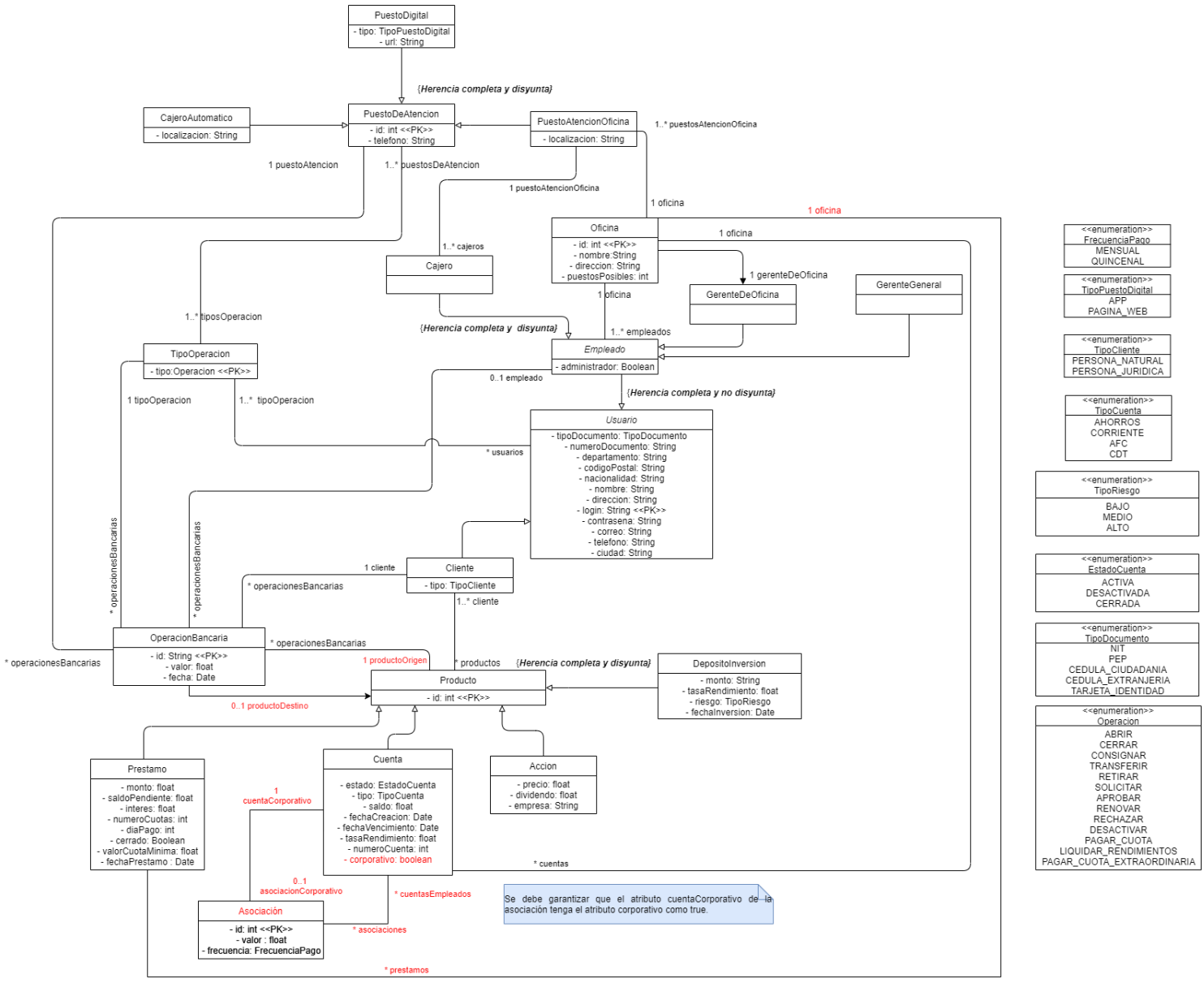

Figura 14. Modelo conceptual de BancAndes

Persistencia en el modelo conceptual:

De acuerdo a la manera en la que está caracterizada la información del caso de BancAndes, todos los conceptos (Usuario, Empleado, Oficina, GerenteDeOficina, GerenteGeneral, Cajero, PuestoAtencionOficina, PuestoAtencion, PuestoDigital, CajeroAutomatico, TipoOperacion, OperacionBancaria, Producto, Prestamo, Cuenta, Acción, DepositoInversion, Cliente, Asociación) deben ser persistentes, ya que solo se está guardando la información necesaria para el correcto funcionamiento del negocio y sus operaciones relacionadas. Todos estos conceptos deben guardarse en la base de datos y sobrevivir a las operaciones del sistema, permitiendo que se consulte el registro histórico asociado a cada uno.

2. Diseño de la base de datos

2.1 Modelo de datos relacional

A continuación, se muestran las tablas que representan las entidades y asociaciones del modelo conceptual planteado. Para las restricciones de los atributos de tipo check (CK) se utilizan expresiones regulares, donde w simboliza caracteres alfabéticos, s representa al carácter espacio blanco (‘ ’) y d representa a los dígitos numéricos. **Los cambios realizados con respecto a la iteración 2 se marcan en color rojo.**

TipoOperacion
tipo
PK, UA, CK(ABRIR,CERRAR,CONSIGNAR,TRANSFERIR, RETIRAR, SOLICITAR, APROBAR, RENOVAR, RECHAZAR, DESACTIVAR, PAGAR_CUOTA, LIQUIDAR_RENDIMIENTOS, PAGAR_CUOTA_EXTRAORDINARIA)

Figura 15. Representación de la entidad TipoOperacion en el modelo relacional

OperacionBancaria								
id	valor	fecha	cliente	productoOrigen	productoDestino	tipoOperacion	puestoAtencion	empleado
PK, SA	NN	NN	FK_Cliente.login, NN	FK_Producto.id, NN	FK_Producto.id	FK_TipoOperacion.tipo, NN	FK_PuestoDeAtencion.id, NN	FK_Empleado.login

Figura 16. Representación de la entidad OperacionBancaria en el modelo relacional

Usuario
login
PK,UA, CK(length(login)>5, charIndex(' ')=0

Figura 17. Representación de la entidad Usuario en el modelo relacional

Empleado
login
PK,UA, CK(length(login)>5, charIndex(' ')=0

Figura 18. Representación de la entidad Empleado en el modelo relacional

Cliente												
tipoDocumento	numeroDocumento	departamento	codigoPostal	nacionalidad	nombre	direccion	login	contrasena	correo	telefono	ciudad	tipo
NN, CK(NIT, PEP, CEDULA_CIUDADANIA, CEDULA_EXTRANJERIA, TARJETA_IDENTIDAD)	CK(d+), NN	NN, CK((w)+ (s w+)*)	NN, CK(d+)	NN, CK((w)+ (s w+)*)	NN, CK((w)+ (s w+)*)	NN	FK_Usuario. login, PK	NN, CK (length(contrasena) >8)	NN,CK((w|d)+ @ (w)+ (. w (w)+)+)	NN, ND, CK(d+, length(telefono) ¿'>6)	NN, CK((w)+ (s w+)*)	NN, CK(PERSONA_NATURAL, PERSONA_JURIDICA)

Figura 19. Representación de la entidad Cliente en el modelo relacional

Cajero														
tipoDocumento	numeroDocumento	departamento	codigoPostal	nacionalidad	nombre	direccion	login	contrasena	correo	telefono	ciudad	administrador	puestoAtencionOficina	oficina
CK(CEDULA_CIUDADANIA,CEDULA_EXTRANJERIA, PEP)	CK(length(numeroDocumento)>5, d+), NN	NN, CK((w)+ (s w+)*)	NN, CK(d+)	NN, CK((w)+ (s w+)*)	NN, CK((w) + (s w+)*)	NN	FK_Empleado.login, PK	NN, CK(length(contrasena) >8)	NN, CK((w|d)+ @ (w)+ (. w (w)+)+)	NN, ND, CK(d+, length(telefono) >6)	NN, CK((w) + (s w+)*)	NN, CK(TRUE, FALSE)	FK_PuestoAtencionOficina.id	FK_Oficina.id

Figura 20. Representación de la entidad Cajero en el modelo relacional

GerenteDeOficina												
tipoDocumento	numeroDocumento	departamento	codigoPostal	nacionalidad	nombre	direccion	login	contrasena	correo	telefono	ciudad	administrador
CK(CEDULA_CIUDADANIA, CEDULA_EXTRANJERIA, PEP)	CK(length(numeroDocumento)>5, d+), NN	NN, CK((w)+ (s w+)*)	NN, CK(d+)	NN, CK((w)+ (s w+)*)	NN, CK((w)+ (s w+)*)	NN	FK_Empleado.login, PK	NN, CK(length(contrasena)>8)	NN,CK((w|d)+ @ (w)+ (. w (w)+)+)	NN, ND, CK(d+, length(telefono)>6)	NN, CK((w)+ (s w+)*)	NN, CK(TRUE, FALSE)

Figura 21. Representación de la entidad GerenteDeOficina en el modelo relacional

GerenteGeneral													
tipoDocumento	numeroDocumento	departamento	codigoPostal	nacionalidad	nombre	direccion	login	contrasena	correo	telefono	ciudad	administrador	oficina
CK(CEDULA_CIUDADANIA,CEDULA_EXTRANJERIA, PEP)	CK(length(numeroDocumento)>5, d+), NN	NN, CK((w)+ (s w+)*)	NN, CK(d+)	NN, CK((w)+ (s w+)*)	NN, CK((w)+ (s w+)*)	NN	FK_Empleado.login, PK	NN, CK(length(contrasena)>8)	NN,CK((w|d)+ @ (w)+ (. w (w)+)+)	NN, ND, CK(d+, length(telefono)>6)	NN, CK((w)+ (s w+)*)	NN, CK(TRUE, FALSE)	FK_Oficina.id

Figura 22. Representación de la entidad GerenteGeneral en el modelo relacional

Oficina				
id	nombre	direccion	puestosPosibles	gerenteLogin
PK, SA	NN, CK((w)+ (s w+)*)	NN, ND	NN, CK(d+, length(puntosPosibles)>0)	FK_GerenteDeOficina.login, NN

Figura 23. Representación de la entidad Oficina en el modelo relacional

PuestoAtencionOficina			
id	telefono	localizacion	oficina
FK_PuestoDeAtencion.id, PK	NN, ND, CK(d+, length(telefono)>6)	NN	FK_Oficina.id

Figura 24. Representación de la entidad PuestoAtencionOficina en el modelo relacional

PuestoDigital			
id	telefono	tipo	url
FK_PuestoDeAtencion.id, PK	NN, ND, CK(d+, length(telefono)>6)	NN, CK(APP, PAGINA_WEB)	ND

Figura 25. Representación de la entidad PuestoDigital en el modelo relacional

CajeroAutomatico		
id	telefono	localizacion
FK_PuestoDeAtencion.id, PK	NN, ND, CK(d+, length(telefono)>6)	NN

Figura 26. Representación de la entidad CajeroAutomatico en el modelo relacional

PuestoDeAtencion
id
PK, SA

Figura 27. Representación de la entidad PuestoDeAtencion en el modelo relacional

Producto
id
PK, SA

Figura 28. Representación de la entidad Producto en el modelo relacional

Prestamo									
id	monto	saldoPendiente	interes	numeroCuotas	diaPago	valorCuotaMinima	fechaPrestamo	cerrado	oficina
FK_Producto.id, PK	NN, CK(>0)	NN, CK(>=0)	NN, CK(>=0)	NN, CK(>=0)	NN, CK[1..31]	NN, CK(>=0)	NN	NN, CK(TRUE,FALSE)	FK_Oficina.id

Figura 29. Representación de la entidad Préstamo en el modelo relacional

Cuenta									
id	numeroCuenta	estado	tipo	saldo	fechaCreacion	fechaVencimiento	tasaRendimiento	oficina	corporativo
FK_Producto.id, PK	NN, ND, CK(d+, length(numeroCuenta)>9)	NN, CK(ACTIVA, DESACTIVADA, CERRADA)	NN, CK(AHORROS, CORRIENTE, AFC, CDT)	NN, CK(d+, >=0)	NN	(fechaVencimiento>fechaCreacion)	NN, CK(>=0)	FK_Oficina.id	NN, CK(TRUE,FALSE)

Figura 30. Representación de la entidad Cuenta en el modelo relacional

Accion			
id	precio	dividendo	empresa
FK_Producto.id, PK	NN, CK(d+, >=0)	NN, CK(d+, >=0)	NN

Figura 31. Representación de la entidad Acción en el modelo relacional

DepositoInversion				
id	monto	tasaRendimiento	riesgo	fechaInversion
FK_Producto.id, PK	NN, CK(d+, >=0)	NN, CK(d+, >=0)	NN, CK(BAJO, MEDIO, ALTO)	NN

Figura 32. Representación de la entidad DepositoInversion en el modelo relacional

UsuarioTipoOperacion	
tipoOperacion	usuario
FK_TipoOperacion.tipo, PK	FK_Usuario.login, PK

Figura 33. Representación de la asociación entre Usuario y TipoOperacion en el modelo relacional

PuestoAtencionTipoOperacion	
tipoOperacion	puestoAtencion
FK_TipoOperacion.tipo, PK	FK_PuestoDeAtencion, PK

Figura 34. Representación de la asociación entre PuestoAtencion y TipoOperacion en el modelo relacional

ClientesProductos	
producto	cliente
FK_Producto.id, PK	FK_Cliente.login, PK

Figura 35. Representación de la asociación entre Cliente y Producto en el modelo relacional

Asociacion			
id	valor	frecuencia	cuentaCorporativo
PK	NN, CK(d+, >=0)	NN, CK(QUINCENAL, MENSUAL)	NN, ND, FK_Cuenta.id

Figura 36. Representación de la entidad Asociación en el modelo relacional

AsociacionCuentasEmpleados	
asociacion	cuentaEmpleado
FK_Asociacion.id, PK	FK_Cuenta.id, PK

Figura 37. Representación de la asociación entre Asociación y Cuenta de empleado en el modelo relacion

2.2 Nivel de normalización

A continuación, se presenta cada una de las entidades del modelo relacional, con sus atributos, llaves candidatas, llave principal, dependencias funcionales y su correspondiente forma normal.

Usuario(login)

Llaves candidatas: {login}

PK: login

Dependencias: login -> login

FNBC

Empleado (login)

Llaves candidatas: {login}

PK: login

Dependencias: login -> login

FNBC

TipoOperacion (tipo)

Llaves candidatas: {tipo}

PK: tipo

Dependencias: tipo -> tipo

FNBC

Producto (id)

Llaves candidatas: {id}

PK: {id}

Dependencias: id -> id

FNBC

PuestoDeAtencion (id)

Llaves candidatas: { id }

PK: id

Dependencias: id -> id

FNBC

Cliente (tipoDocumento, numeroDocumento, departamento, codigoPostal, nacionalidad, nombre, direccion, login, contrasena, correo, telefono, ciudad, tipo)

Llaves candidatas: {(tipoDocumento, numeroDocumento), login}

PK: login

Dependencias:

login -> tipoDocumento, numeroDocumento, departamento, codigoPostal, nacionalidad, nombre, direccion, login, contrasena, correo, telefono, ciudad, tipo

(tipoDocumento, numeroDocumento)-> tipoDocumento, numeroDocumento, departamento, codigoPostal, nacionalidad, nombre, direccion, login, contrasena, correo, telefono, ciudad, tipo

FNBC

Cajero (tipoDocumento, numeroDocumento, departamento, codigoPostal, nacionalidad, nombre, direccion, login, contrasena, correo, telefono, ciudad, administrador, puestoAtencionOficina, oficina)

Llaves candidatas: {(tipoDocumento, numeroDocumento), login}

PK: login

Dependencias:

login -> tipoDocumento, numeroDocumento, departamento, codigoPostal, nacionalidad, nombre, direccion, login, contrasena, correo, telefono, ciudad, administrador, puestoAtencionOficina, oficina

(tipoDocumento, numeroDocumento)-> tipoDocumento, numeroDocumento, departamento, codigoPostal, nacionalidad, nombre, direccion, login, contrasena, correo, telefono, ciudad, administrador, puestoAtencionOficina, oficina

FNBC

GerenteDeOficina (tipoDocumento, numeroDocumento, departamento, codigoPostal, nacionalidad, nombre, direccion, login, contrasena, correo, telefono, ciudad, administrador)

Llaves candidatas: {(tipoDocumento, numeroDocumento), login}

PK: login

Dependencias:

login -> tipoDocumento, numeroDocumento, departamento, codigoPostal, nacionalidad, nombre, direccion, login, contrasena, correo, telefono, ciudad, administrador

(tipoDocumento, numeroDocumento)-> tipoDocumento, numeroDocumento, departamento, codigoPostal, nacionalidad, nombre, direccion, login, contrasena, correo, telefono, ciudad, administrador

FNBC

GerenteGeneral (tipoDocumento, numeroDocumento, departamento, codigoPostal, nacionalidad, nombre, direccion, login, contrasena, correo, telefono, ciudad, administrador, oficina)

Llaves candidatas: {(tipoDocumento, numeroDocumento), login}

PK: login

Dependencias:

login -> tipoDocumento, numeroDocumento, departamento, codigoPostal, nacionalidad, nombre, direccion, login, contrasena, correo, telefono, ciudad, administrador, oficina

(tipoDocumento, numeroDocumento)-> tipoDocumento, numeroDocumento, departamento, codigoPostal, nacionalidad, nombre, direccion, login, contrasena, correo, telefono, ciudad, administrador, oficina

FNBC

Oficina(id)

Llaves candidatas: {id}

PK: {id}

Dependencias:

id -> nombre, nombre, dirección, puestosPosibles, gerenteLogin

FNBC

PuestoAtencionOficina(id, telefono, localización, oficina)

Llaves candidatas:{id}

PK: {id}

Dependencias:

id -> id, telefono, localización, oficina

FNBC

PuestoDigital(id, teléfono, tipo, url)

Llaves candidatas: {id}

PK:{id}

Dependencias:

id-> id, teléfono, tipo, url

FNBC

CajeroAutomatico(id, teléfono, localizacion)

Llaves candidatas: {id}

PK:{id}

Dependencias:

id-> id, teléfono, localización

FNBC

OperacionBancaria(id, valor, fecha, cliente, productoOrigen, productoDestino, tipoOperacion, puestoAtencion, empleado)

Llaves candidatas: {id}

PK:{id}

Dependencias:

id-> id, valor, fecha, cliente, productoOrigen, productoDestino, tipoOperacion, puestoAtencion, empleado

FNBC

Prestamo (id, monto, saldoPendiente, interes, numeroCuotas, diaPago, valorCuotaMinima, fechaPrestamo, cerrado, oficina)

Llaves candidatas: {id}

PK:{id}

Dependencias:

id-> i id, monto, saldoPendiente, interés, numeroCuotas, diaPago, valorCuotaMinima, fechaPrestamo, cerrado, oficina

FNBC

Cuenta(id, numeroCuenta, estado, tipo, saldo, fechaCreacion, fechaVencimiento, tasaRendimiento, oficina, corporativo)

Llaves candidatas: {id}

PK:{id}

Dependencias:

id-> id, numeroCuenta, estado, tipo, saldo, fechaCreacion, fechaVencimiento, tasaRendimiento, oficina, corporativo

FNBC

Accion (id, precio, dividendo, empresa)

Llaves candidatas: {id}

PK:{id}

Dependencias:

id-> id, precio, dividendo, empresa

FNBC

DepositoInversion(id, monto, tasaRendimiento, riesgo, fechaInversion)

Llaves candidatas: {id}

PK:{id}

Dependencias:

id-> id, monto, tasaRendimiento, riesgo, fechaInversion

FNBC

UsuarioTipoOperacion (tipoOperacion, usuario)

Llaves candidatas: {(tipoOperacion, usuario)}

PK:{(tipoOperacion, usuario)}

Dependencias:

(tipoOperacion, usuario)-> (tipoOperacion, usuario)

FNBC

PuestoAtencionTipoOperacion(tipoOperacion, puestoAtencion)

Llaves candidatas: { (tipoOperacion, puestoAtencion) }

PK:{ (tipoOperacion, puestoAtencion)}

Dependencias:

(tipoOperacion, puestoAtencion) -> (tipoOperacion, puestoAtencion)

FNBC

ClientesProductos (producto, cliente)

Llaves candidatas: {(producto, cliente)}

PK:{ (producto, cliente)}

Dependencias:

(producto, cliente)-> (producto, cliente)

FNBC

Asociación (id, valor, frecuencia, cuentaCorporativo)

Llaves candidatas: {id}

PK:{id}

Dependencias:

id-> id, valor, frecuencia, cuentaCorporativo

FNBC

AsociaciónCuentasEmpleados (asociación, cuentaEmpleado)

Llaves candidatas: {(asociación, cuentaEmpleado)}

PK:{ (asociación, cuentaEmpleado)}

Dependencias:

(asociación, cuentaEmpleado)-> (asociación, cuentaEmpleado)

FNBC

A partir del anterior análisis se puede determinar que el nivel de normalización en el que se encuentra el modelo es FNBC (Forma Normal Boyce-Codd).

3. Control de calidad del modelo

3.1 Diagramas de secuencia

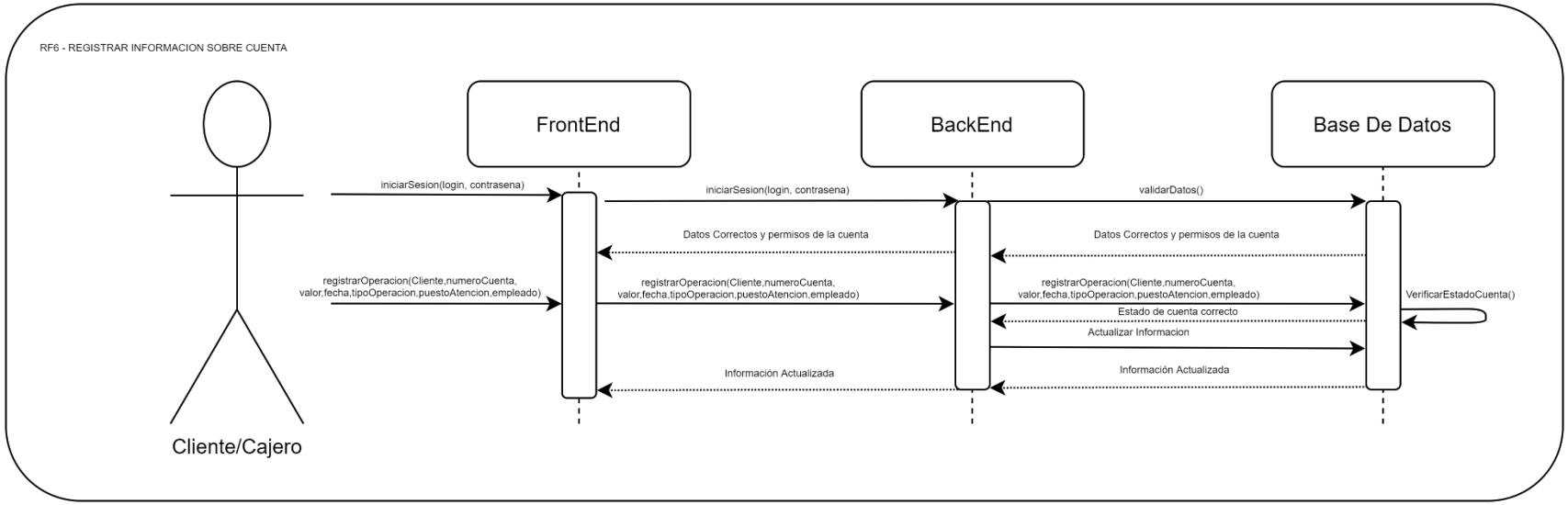

Figura 37. Diagrama de secuencia del RF6

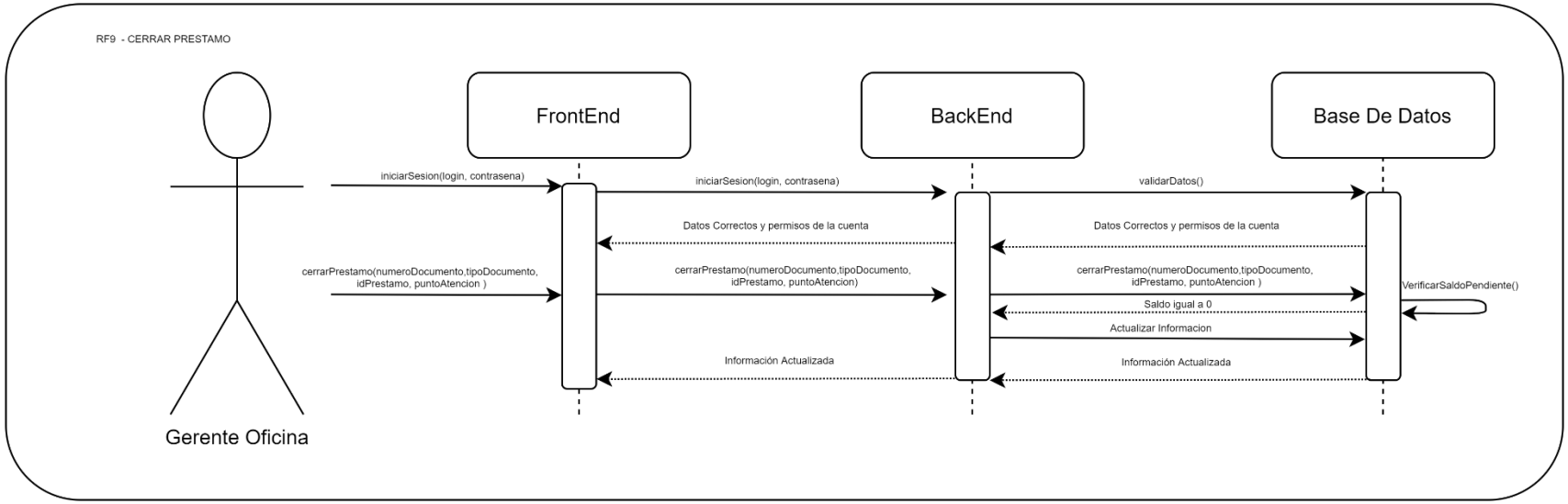

Figura 38. Diagrama de secuencia del RF9

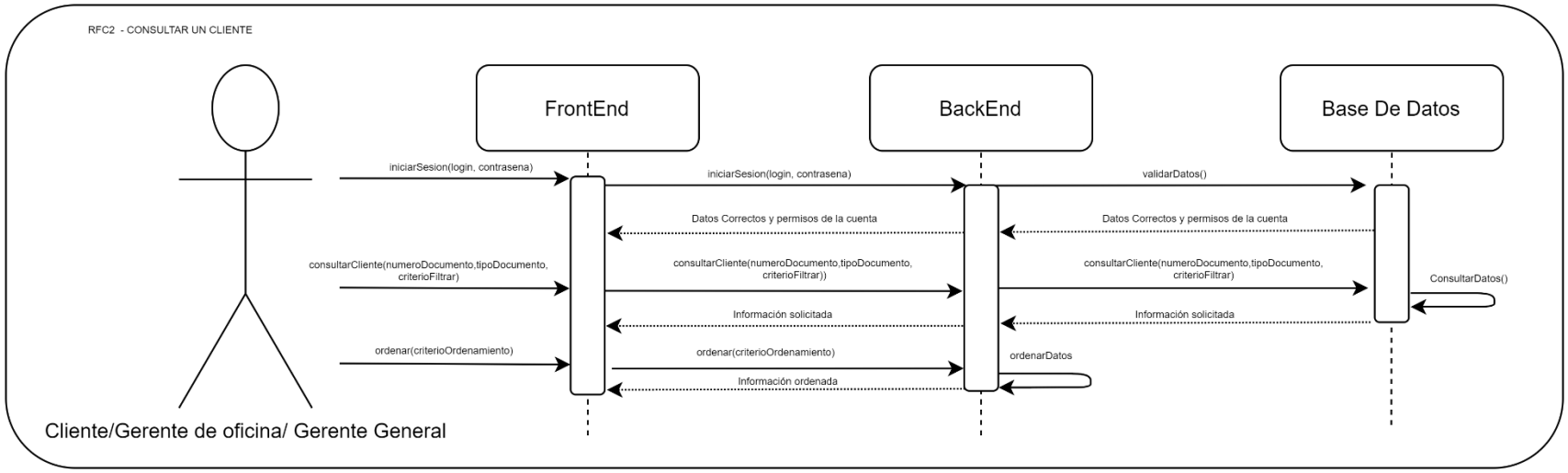

Figura 39. Diagrama de secuencia del RFC2

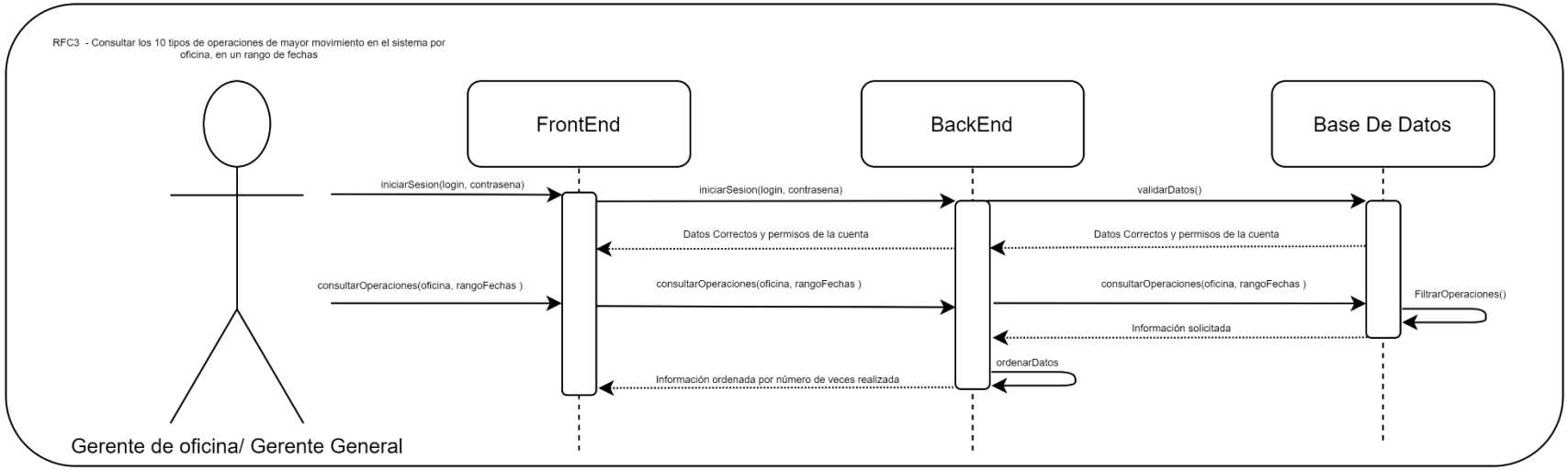

Figura 40. Diagrama de secuencia del RFC3

4. Reglas de negocio encontradas en el caso de estudio

A continuación, se muestran las reglas del negocio y suposiciones adicionales que se utilizaron en la creación del modelo conceptual:

- El login es único para cada usuario del banco.
- Puede haber diferentes tipos de clientes, se diferencian por su tipo de documento con el que se registren.
- Todo usuario del banco debe tener tipo de documento, numero de documento, departamento, código postal, nacionalidad, nombre, dirección, login, contraseña, correo, teléfono y ciudad.
- Un usuario no puede tener registrado más de un correo.
- Solo los usuarios de tipo empleado pueden tener permisos de administradores sobre el sistema.
- Los empleados también pueden ser clientes de BancAndes.
- Un mismo correo puede ser usado por diferentes usuarios.
- Un cliente puede tener varios prestamos, de cualquier tipo (para vivienda, estudio, automóvil, calamidad domestica o de libre inversión).
- Todos los puestos de atención tienen acceso a BancAndes, pero según su tipo restringen las operaciones que se pueden realizar en ellos.
- Si un cliente cierra o desactiva una cuenta, la información asociada se mantiene en la base de datos.
- Si un cliente termina de pagar su préstamo, este se mantiene en la base de datos por motivos estadísticos.
- Un préstamo solo puede ser cerrado si el monto alcanza un valor de cero.
- Debe haber un gerente de oficina por cada oficina registrada.
- Debe haber un cajero por cada punto de atención registrado.
- Varios clientes pueden tener un mismo producto en el caso de que se trate de préstamos o cuentas compartidas.
- Un cliente puede tener varios tipos de productos ofrecidos por el banco.
- El tipo de usuario restringe los requerimientos funcionales del sistema a los que tiene acceso.
- Las cuentas de los clientes pueden ser de tipo corriente, de ahorros, CDT o AFC.
- Solo las cuentas de tipo CDT tienen fecha de vencimiento.
- Un producto puede estar asociado a varios clientes (por ejemplo: una cuenta compartida).
- Un puesto de atención puede tener varios cajeros (al menos uno) pero un cajero solo tiene un puesto de atención de oficina.
- Solo las cuentas de tipo corriente pueden tener sobregiros.
- Las operaciones entre cuentas se pueden realizar de manera electrónica, por lo que el sistema soporta transacciones entre productos siempre y cuando la cuenta desde donde se realiza la transacción tiene suficientes recursos.
- Los clientes de tipo persona jurídica pueden asociar las cuentas de sus empleados a su cuenta corporativa para realizar el pago de nómina como una transferencia periódica según el valor y frecuencia especificado.

- Máximo puede haber una asociación por cada cuenta corporativa.
- Las cuentas de los empleados pueden estar asociadas a varias cuentas corporativas (dado que es posible que un cliente tenga varios empleos).
- Al momento de cerrar una cuenta se debe verificar si se trata de una cuenta corporativa con una asociación registrada, en caso de que sea así, se debe actualizar la asociación con otra cuenta corporativa que sea corporativa y esté activa antes de registrar el cierre.

5. Documentación de la lógica de los requerimientos nuevos -Iteración 3

RF10. Asociar cuenta de persona natural a cuenta de persona jurídica

Este requerimiento se soporta en la aplicación por medio de la adición de una tupla a la tabla de Asociaciones en la base de datos, en donde se especifica el id de la asociación, la cuenta del cliente corporativo que crea la asociación y el valor y la frecuencia con que se deben realizar los pagos de nómina. Una vez creada la asociación, se inserta una nueva pareja de Asociación-CuentaEmpleado por cada cuenta de empleado que se quiera asociar a la cuenta corporativa.

Para garantizar las propiedades de consistencia del requerimiento la aplicación solo permite crear una asociación con una cuenta corporativa existente en la base de datos, solo en caso de que se cree e inserte correctamente la tupla en la tabla de Asociaciones, con un valor mayor a cero de pago y una frecuencia mensual o quincenal, se podrá pasar a asociar cuentas de empleados. En cuanto a la atomicidad del requerimiento, en caso de que no se agregue ningún empleado válido se borrará la asociación. Por el contrario, si se inserta la asociación y se relaciona con al menos una cuenta de empleado, se hace *commit* y los resultados persistirán en la base de datos, esto es, serán durables. Se garantiza que el requerimiento se realice de forma aislada mediante el nivel de aislamiento de la transacción, el cual se configuró como *read-committed* de tal forma que no ocurran lecturas sucias, teniendo en cuenta que las lecturas no repetibles y *phantoms* no son un problema para las inserciones que realiza esta transacción.

RF11. Registrar operación sobre cuenta (V2)

Este requerimiento se soporta en la aplicación permitiendo que un usuario de tipo cajero o cliente registre una operación de tipo retiro, consignación o transacción en sus cuentas. Para esto, en caso de tratarse de una consignación o retiro se utiliza la misma implementación del requerimiento RF6. Para el caso de operaciones de transacción, se permite ingresar una cuenta de origen y una cuenta destino para la operación y se procede a utilizar como subtransacción la implementación del RF6 para retirar el valor especificado de la cuenta de origen y consignarlo en la cuenta de destino. En ambos casos, primero se registra la operación y, en caso de realizarse correctamente su inserción en la base de datos, se registra el cambio en el saldo de las cuentas involucradas en la operación.

Para garantizar que la transacción lleve la base de datos de un estado consistente a otro, solo se puede registrar una operación bancaria sobre cuentas existentes en la base de datos que tengan suficientes fondos para realizar la operación. De esta forma, solo se podrán realizar sobregiros si la cuenta desde

donde se hace un retiro o desde donde se origina la transacción es de tipo corriente. Además, para garantizar la propiedad de atomicidad, la operación bancaria solo se registra en la tabla de OperacionesBancarias si la operación se realiza correctamente. De manera similar, en caso de que no se pueda realizar la consignación en la cuenta destino durante una transacción, no se guardarán los cambios del retiro sobre la cuenta de origen ni del registro de la operación bancaria. Por el contrario, si se inserta la operación y se registran correctamente los cambios del saldo de las cuentas involucradas, se hace *commit* y los resultados persistirán en la base de datos, esto es, serán durables. Se garantiza que el requerimiento se realice de forma aislada mediante el nivel de aislamiento de la transacción, el cual se configuró como *serializable* de tal forma que no ocurran lecturas sucias ni lecturas inconsistentes al momento de registrar la operación y los cambios sobre las cuentas, teniendo en cuenta que los *phantoms* no son un problema porque solo se realizan las actualizaciones sobre máximo dos productos (origen y destino) pero que no se puede configurar como *repeatable-read* porque no se soporta ese nivel de aislamiento.

RF12. Registrar operación sobre préstamo (V2)

Este requerimiento se soporta en la aplicación permitiendo que un usuario de tipo cajero o cliente registre una operación de pago sobre un prástamo. Para esto, en caso de tratarse de un pago en efectivo se utiliza la misma implementación del requerimiento RF8. Para el caso de operaciones de transacción, es decir de abonar un pago al préstamo desde una cuenta, se permite ingresar una cuenta de origen y un préstamo de destino para la operación y se procede a utilizar como subtransacción la implementación del RF6 para retirar el valor especificado de la cuenta de origen y la implementación del RF8 abonar el pago al préstamo. En ambos casos, primero se registra la operación y, en caso de realizarse correctamente su inserción en la base de datos, se registra el cambio en el saldo de los productos involucrados en la operación.

Para garantizar que la transacción lleve la base de datos de un estado consistente a otro, solo se puede registrar una operación bancaria sobre productos existentes en la base de datos. En caso de tratarse de un pago electrónico solo podrá realizarse si como producto de origen se especifica una cuenta activa existente que tenga suficientes fondos para realizar la operación y un préstamo abierto como producto de destino. De esta forma, solo se podrán realizar sobregiros si la cuenta desde donde se origina la transacción es de tipo corriente. El tipo de operación se registrará como “pagar cuota” si el valor de la operación es igual a la cuota mínima del préstamo o como “pagar cuota extraordinaria” si es superior, no se podrá registrar la operación en caso de que el valor a abonar sea inferior a esta cifra. Además, para garantizar la propiedad de atomicidad, la operación bancaria solo se registra en la tabla de OperacionesBancarias si la operación se realiza correctamente. De manera similar, en caso de que no se pueda realizar el abono del pago durante un pago electrónico, no se guardarán los cambios del retiro sobre la cuenta de origen ni del registro de la operación bancaria. Por el contrario, si se inserta la operación y se registran correctamente los cambios del saldo de los productos involucrados, se hace *commit* y los resultados persistirán en la base de datos, esto es, serán durables. Se garantiza que el requerimiento se realice de forma aislada mediante el nivel de aislamiento de la transacción, el cual se configuró como *serializable* de tal forma que no ocurran lecturas sucias ni lecturas inconsistentes al momento de registrar la operación y los cambios sobre los productos, teniendo en cuenta que los

phantoms no son un problema porque solo se realizan las actualizaciones sobre máximo dos productos (origen y destino) pero que no se puede configurar como *repeatable-read* porque no se soporta ese nivel de aislamiento.

RF13. Pagar nómina de persona jurídica

Este requerimiento se soporta en la aplicación permitiendo que un usuario de tipo cajero o cliente registra una operación de pago de nómina. Para esto, basta con que el usuario especifique el login de la cuenta corporativa desde donde se realiza el pago, se busca en la tabla de Asociaciones si existe alguna tupla con la cuenta corporativa indicada y, en caso de que se encuentra, se procede a registrar la transacción por cada uno de los clientes asociados utilizando la implementación del RF11 como subtransacción con el valor especificado en la asociación.

Para garantizar que la transacción se lleve a cabo de manera consistente, solo se puede realizar la operación si la cuenta indicada existe en la base de datos, está activa, es de tipo corporativo y tiene registrada una asociación. Además, solo se realizará el pago de nómina para los empleados para los que alcancen los fondos de la cuenta corporativa. Para esto, se realiza un *savepoint* después de registrar la transacción a cada empleado (registro de la operación en la tabla de OperacionesBancarias y actualización del saldo de la cuenta corporativa y la cuenta del empleado) y en caso de falla o en caso de que en medio del pago de nómina se acaben los fondos de la cuenta corporativa, se hace *commit* de los pagos que se alcanzaron a hacer y se muestra al usuario una lista con todos los empleados que no recibieron el pago. En cualquier caso, una vez se informa al usuario que realiza la operación el resultado, se hace *commit* a los pagos que sí se alcanzaron a hacer, por lo que estos resultados serán durables en la base de datos. Se garantiza que el requerimiento se realice de forma aislada mediante el nivel de aislamiento de la transacción, el cual se configuró como *serializable* para que no sea posible que hayan lecturas sucias, lecturas inconsistentes ni *phantoms*, es decir, que no se puedan asociar más cuentas de empleados a la cuenta corporativa mientras se está pagando la nómina.

RF14. Cerrar cuenta (V2)

Este requerimiento se soporta en la aplicación permitiendo que un usuario de tipo gerente de oficina cierre una cuenta, en caso de que la cuenta no sea corporativa o de que sea corporativa pero que no tenga ninguna asociación con cuentas de empleados, se utilizará la misma implementación del RF5 y en caso contrario se tendrá que especificar una nueva cuenta corporativa para reemplazar en la asociación antes de dejar el saldo en 0 y cerrar la cuenta (esto es, utilizar la implementación del RF5 como subtransacción).

Dada la forma en que se modelaron las asociaciones en el modelo relacional, para hacer el cambio de la cuenta corporativa en la asociación bastará con registrar una vez el cambio en la tabla de Asociaciones; no habrá que hacer una actualización por cada cuenta de empleado asociada. Esto permite que se garanticen las propiedades de consistencia y atomicidad del requerimiento asegurando que solo se pueda cerrar la cuenta si se registró con éxito otra cuenta corporativa en la asociación y que, en caso de fallar el cierre de la cuenta, no se guarde la actualización en la base de datos. Por el contrario, si se registra la actualización de la cuenta corporativa en la asociación y se registra correctamente el cierre de la cuenta,

se hace *commit* y los resultados persistirán en la base de datos, esto es, serán durables. Se garantiza que el requerimiento se realice de forma aislada mediante el nivel de aislamiento de la transacción, el cual se configuró como *serializable* para que no pueda haber lecturas sucias ni inconsistentes, de manera que no pueda eliminarse o cerrarse la cuenta corporativa nueva a la que se actualiza la asociación mientras se realiza la transacción de cierre en su totalidad. Teniendo en cuenta que los *phantoms* no son un problema para este requerimiento ya que solo se realiza una actualización a la tupla de la asociación y a la tupla de la cuenta pero que no se puede configurar como *repeatable-read* porque no se soporta ese nivel de aislamiento.

RFC5. Consultar préstamos

Este requerimiento se soporta en la aplicación permitiendo que un usuario de tipo cliente, gerente de oficina o gerente general realice una consulta sobre los préstamos del sistema y, respetando la privacidad de los clientes, según el tipo de usuario podrá ver solo la información de sus préstamos, de los préstamos de su oficina o de todos los préstamos en BancAndes, respectivamente. Además, según los intereses del usuario se podrán filtrar u organizar las búsquedas a partir de los siguientes criterios: si el préstamo está cerrado, saldo, cuota mínima, monto, fecha de creación, cliente (en caso de que no sea un cliente), oficina (en caso de que no sea un cliente ni un gerente de oficina). Los resultados son consistentes y la consulta se realiza de forma aislada puesto que se realizan con un nivel de aislamiento de *read-committed* así que sólo podrán leer los datos a los que se les haya hecho *commit*, es decir que solo podrán ver estados consistentes de la base de datos. Sin embargo, no se garantiza la atomicidad ni la durabilidad de los resultados ya que el requerimiento no necesita comportamiento transaccional.

RFC6. Consultar operaciones

Este requerimiento se soporta en la aplicación permitiendo que un usuario de tipo cliente, gerente de oficina o gerente general realice una consulta sobre las operaciones en el sistema y, respetando la privacidad de los clientes, según el tipo de usuario podrá ver solo la información de sus operaciones, de las operaciones de su oficina o de todas las operaciones en BancAndes, respectivamente. Además, según los intereses del usuario se podrán filtrar u organizar las búsquedas a partir de los siguientes criterios: tipo de operación, producto, valor, fecha, cliente (en caso de que no sea un cliente), oficina (en caso de que no sea un cliente ni un gerente de oficina). Los resultados son consistentes y la consulta se realiza de forma aislada puesto que se realizan con un nivel de aislamiento de *read-committed* así que sólo podrán leer los datos a los que se les haya hecho *commit*, es decir que solo podrán ver estados consistentes de la base de datos. Sin embargo, no se garantiza la atomicidad ni la durabilidad de los resultados ya que el requerimiento no necesita comportamiento transaccional.

RNF5. Transaccionalidad

Se garantiza el comportamiento transaccional de los requerimientos RF10, RF11, RF13 y RF14 logrando que cada uno se realicen de manera atómica, consistente, durable y aislada. Es decir, toda la operación

se realiza o no se hace, a excepción del RF13, el cual implementa un esquema de *savepoints* para realizar el pago de nómina, de forma que si ocurre una falla en medio del procedimiento o se acaban los fondos de la cuenta corporativa desde donde se realizan los pagos, se guardan los pagos que se alcanzaron a hacer y se informa al usuario los pagos que quedaron pendientes. Además, garantizan que la base de datos pase de un estado consistente a otro, de manera que se cumplan las reglas de negocio y la información del sistema sea consistente con la realidad. Asimismo, los resultados de los requerimientos son durables en caso de que se ejecuten correctamente. También se garantiza que las operaciones relacionadas a estos requerimientos se puedan ejecutar de manera concurrente mediante los niveles de aislamiento escogidos para cada transacción, como se especificaron anteriormente.

6. Balance del plan de pruebas

En la carpeta de Tests se incluyeron archivos de pruebas unitarias para todos los requerimientos nuevos de la iteración 3. Allí se plantean escenarios de prueba que cumplen con las restricciones y suposiciones planteadas previamente en el modelo y que prueban la integridad de las tablas y de las restricciones insertadas. Además, se adjunta un documento de Excel con la documentación de las pruebas, allí se especifican los escenarios de prueba válidos y fallidos junto con los inputs y outputs de cada prueba.

7. Resultados logrados y no logrados

En esta iteración se lograron todos los resultados esperados. Para esto, primero se solucionaron los errores en los requerimientos de la anterior iteración. Luego, se realizaron los cambios pertinentes en el modelo conceptual para cumplir con los requerimientos nuevos. Estos cambios también se ajustaron en el modelo relacional, en las tablas creadas en la base de datos y en los datos ingresados allí para la prueba de los requerimientos. Después se realizaron los cambios en la aplicación, en los módulos de persistencia, de negocio y de interfaz. A partir de allí, se desarrollaron los nuevos requerimientos, esto es, los 5 requerimientos funcionales de modificación y los 2 requerimientos de consulta. Finalmente, se ajustaron los requerimientos para cumplir con las condiciones ACID y se diseñaron las pruebas para cada requerimiento.

Los requerimientos funcionales que sí se implementaron cumplen con las reglas del negocio encontradas en el enunciado y las suposiciones adicionales mencionadas anteriormente. Lo anterior significa que funcionan cuando deben funcionar, revisan los permisos del usuario para saber si les es permitido ejecutar o no una funcionalidad, respetan el requerimiento no funcional de privacidad, minimizan la posibilidad de ingreso de datos incorrectos por parte del usuario y son realizados utilizando la menor cantidad de sentencias SQL posibles.