ASIGNATURA: SISTEMAS TRANSACCIONALES
ESTUDIANTES: JUAN DAVID BRICEÑO, GABRIEL ARISTIZÁBAL, SERGIO FRANCO PINEDA
CÓDIGOS: 201812887, 202110699, 202116614
PROGRAMA DE ESTUDIOS: INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

PARTE A

CLASES DE JAVA CREADAS:

- CUENTA
- OPERACIONCUENTA
- CLIENTE
- EMPLEADO
- OFICINA
- PUNTOATENCION
- USUARIO
- PRESTAMOS
- OPERACIONPRESTAMOS

Todas las clases de java creadas en la lista superior cuentan con sus respectivos atributos definidos en la entrega anterior (tablas SQL), y sus respectivos getters y setters para los atributos.

Se creo en la carpeta repositorio sus respectivas clases

- CuentaRepository: posee las funciones darCuenas, darCuenta, insertarCuenta, actualizarCuenta.
- OperacionCuentaRepository: posee las funciones de darOperacionesPrestamos, registrarOperacionRetiro, retirarDinero, consignarOperacionCuenta, consignar, registrarOperacionTransferencia, restarSaldoOrigen y sumarSaldoOrigen.
- ClienteRepository: posee las funciones consultarCliente, darCliente e intertar cliente.
- EmpleadosRepository: posee la función de ingresarEmpleado.
- OficinaRepository: posee la función de ingresarOficina.
- PuntosAtencionRepository: posee las funciones de insertarPuntoAtencion y eliminarPuntoAtencion.
- UsuarioRepository: posee las funciones de darUsuarios, darUsuario e insertarUsuario.
- PrestamosRepository: posee las funciones de darPrestamos, darPrestamo e insertarPrestamo.
- OperacionPrestamosRepository: posee las funciones darOperacionesPrestamos, darOperacionPrestamo, darOperacionPrestamo, actualizarPrestamo.

Se creo en la carpeta de controladores sus respectivas clases

- ClienteController
- CuentaController
- OperacionCuentaController: Este controller posee las funciones que son llamadas y cumplen con las actividades que se pueden realizar desde la cuenta, como lo son consignarDinero, retirarDinero, trasferirDinero y ModificarCuenta.
- EmpleadoController
- OficinaControleer
- PuntoAtencion Controller
- UsuarioController

- PrestamosController
- OperacionPrestamosController: Este controller posee las funciones que son llamadas y cumplen con actividades de la clase, como lo son guardarOperaciones, llenarOperaciones y actualizarOperaciones sobre los préstamos.

Todas las clases tipo Controller, tienen el constructor de la clase. La excepción de esto son OepracionPrestamosController y OperacionCuentaController. Estos se profundizan arriba.

PARTE B

Niveles de aislamiento

SERIALIZABLE

SESIÓN 1:

Operación 1: Se coloca el nivel de aislamiento serializable.

Operación 2: Modificación del saldo de la CUENTA_1 sumándole 1 millón de pesos.

Operación 3: Registro en el log de la consignación de 1 millón de pesos en la CUENTA_1.

Operación 4: Modificación del saldo de la CUENTA_2 restando 50,000 pesos.

Operación 5: Registro en el log del retiro de 50,000 pesos en la CUENTA_2.

Operación 6: Cierre exitoso del primer Bloque de Transacción.

Consulta del saldo: El saldo actual de la CUENTA_1 es $5.970.000 y el saldo actual de la CUENTA_2 es $9.955.000.

SESIÓN 2:

Operación 1: Se coloca el nivel de aislamiento serializable.

Operación 2: Modificación del saldo de la CUENTA_1 restando 30,000 pesos por cuota de manejo.

Operación 3: Registro en el log del retiro de 30,000 pesos por cuota de manejo en la CUENTA_1.

Operación 4: Modificación del saldo de la CUENTA_2 sumándole 5,000 pesos de intereses.

Operación 5: Registro en el log de la consignación de 5,000 pesos de intereses en la CUENTA_2.

Operación 6: Cierre exitoso del primer Bloque de Transacción.

Consulta del saldo: El saldo actual de la CUENTA_1 es $5.970.000 y el saldo actual de la CUENTA_2 es $9.955.000.

Resultado Final:

Después de todas las transacciones y consultas de saldo, el estado final de las cuentas es el siguiente:

El saldo final de la CUENTA_1 es $5.970.000.

El saldo final de la CUENTA_2 es $9.955.000.

Estos saldos finales reflejan todas las operaciones realizadas en las cuentas durante las sesiones y están confirmadas por el COMMIT final en la segunda sesión.

READ COMMITED

SESIÓN 1:

Operación 1: Se coloca el nivel de aislamiento read commited.

Operación 2: Modificación del saldo de la CUENTA_1 sumándole 1 millón de pesos.

Operación 3: Registro en el log de la consignación de 1 millón de pesos en la CUENTA_1.

Operación 4: Modificación del saldo de la CUENTA_2 restando 50,000 pesos.

Operación 5: Registro en el log del retiro de 50,000 pesos en la CUENTA_2.

Operación 6: Cierre exitoso del primer Bloque de Transacción.

Consulta del saldo: El saldo actual de la CUENTA_1 es $5.970.000 y el saldo actual de la CUENTA_2 es $9.955.000.

SESIÓN 2:

Operación 1: Se coloca el nivel de aislamiento read commited.

Operación 2: Modificación del saldo de la CUENTA_1 restando 30,000 pesos por cuota de manejo.

Operación 3: Registro en el log del retiro de 30,000 pesos por cuota de manejo en la CUENTA_1.

Operación 4: Modificación del saldo de la CUENTA_2 sumándole 5,000 pesos de intereses.

Operación 5: Registro en el log de la consignación de 5,000 pesos de intereses en la CUENTA_2.

Operación 6: Cierre exitoso del primer Bloque de Transacción.

Consulta del saldo: El saldo actual de la CUENTA_1 es $8.970.000 y el saldo actual de la CUENTA_2 es $14.950.000.

Resultado Final:

Después de todas las transacciones y consultas de saldo, el estado final de las cuentas es el siguiente:

El saldo final de la CUENTA_1 es $8.970.000.

El saldo final de la CUENTA_2 es $14.950.000.

Estos saldos finales reflejan todas las operaciones realizadas en las cuentas durante las sesiones y están confirmadas por el COMMIT final en la segunda sesión.

RFC4 – CONSULTA DE OPERACIONES REALIZADAS SOBRE UNA CUENTA – SERIALIZABLE

RFC5 – CONSULTA DE OPERACIONES REALIZADAS SOBRE UNA CUENTA – READ COMMITED

Para realizar los nuevos RFC4 y RFC5, se tomo de base las requerimientos y funciones que ya se tenían en la clase de Java OperacionCuentaRepository y CuentaRepository. Se realizo un inner join entre estos de la siguiente forma:

"SELECT C.SALDO, C.FECHAULTIMATRACCION, C.ESTADO FROM CUENTA C INNERJOIN OPERACIONCUENTA OC ON C.NUMEROCUENTA = OC.NUMEROCUENTA WHERE C.NUMEROCUENTA = :NUMEROCUENTA"

Para poder obtener las operaciones que se realizaron sobre la cuenta a la cual se estaba pidiendo la información comparando sus respectivas variables ids.

En cuanto a los requerimientos funciones con niveles de aislamiento específicos, se utilizo el siguiente filtro de select:

"SELECT OC.ID, OC.TIPOPAGO FROM OPERACIONCUENTA OC WHERE OC.NUMEROCUENTA = :NUMEROCUENTA"

Sumado a esto se especificó el aislamiento utilizando el comando isolation, donde se llama la clase Isolation y luego llama la función ya diseñada donde se asigna el valor respectivo, se sea serializable o read commited.

RF6 - REGISTRAR OPERACIÓN SOBRE CUENTA

Operación de Retirar Dinero

Para la operación de retirar dinero, se implementó la funcionalidad correspondiente en el repositorio de operaciones de cuenta (OperacionCuentaRepository). Esta operación involucra dos pasos principales: registrar la operación de retiro en un log de operaciones sobre cuentas y actualizar el saldo de la cuenta. La consulta SQL utilizada para registrar la operación de retiro en el log es la siguiente:

INSERT INTO log_operaciones_cuentas (TIPOPAGO, NUMEROCUENTAAFECTADA, NUMEROCUENTA) VALUES ('Retiro', :cuentaOrigen, :cuentaOrigen)

Además, se ejecuta una consulta SQL para actualizar el saldo de la cuenta correspondiente:

UPDATE cuenta SET saldo = saldo - :monto WHERE numero = :cuentaOrigen

Para la operación de retirar dinero, se implementó un método en el controlador que recibe los parámetros necesarios, como el monto a retirar y el número de cuenta origen. Este método luego invoca las funciones correspondientes del repositorio de operaciones de cuenta para realizar el

retiro de dinero. Además, maneja posibles excepciones que puedan ocurrir durante el proceso y redirige al usuario a la página correspondiente con un mensaje de éxito o error

Operación de Consignar Dinero

Para la operación de consignar dinero, se implementó la funcionalidad correspondiente en el repositorio de operaciones de cuenta (OperacionCuentaRepository). Al igual que la operación de retiro, esta operación consta de dos pasos principales: registrar la operación de consignación en el log de operaciones sobre cuentas y actualizar el saldo de la cuenta. La consulta SQL utilizada para registrar la operación de consignación en el log es la siguiente:

```
INSERT INTO log_operaciones_cuentas (TIPOPAGO, NUMEROCUENTAAFECTADA, NUMEROCUENTA) VALUES (:TIPOPAGO, :NUMEROCUENTAAFECTADA, :NUMEROCUENTA)
```

Y se ejecuta una consulta SQL para actualizar el saldo de la cuenta correspondiente:

```
UPDATE cuenta SET saldo = saldo + :monto WHERE numero = :numeroCuenta
```

De manera similar, se implementó un método en el controlador para la operación de consignar dinero. Este método recibe los parámetros requeridos, como el monto a consignar y el número de cuenta destino, y luego invoca las funciones adecuadas del repositorio de operaciones de cuenta para realizar la consignación. Al igual que con la operación de retiro, se manejan excepciones y se redirige al usuario según corresponda

Operación de Transferir Dinero

La operación de transferir dinero involucra dos cuentas: la cuenta de origen y la cuenta de destino. Esta operación se realiza también en el repositorio de operaciones de cuenta (OperacionCuentaRepository). Para esta operación, se registró la operación de transferencia en el log de operaciones sobre cuentas y se actualizó el saldo de ambas cuentas. La consulta SQL utilizada para registrar la operación de transferencia en el log es la siguiente:

```
INSERT INTO log_operaciones_cuentas (TIPOPAGO, NUMEROCUENTAAFECTADA, NUMEROCUENTA) VALUES ('Transferencia', :cuentaDestino, :cuentaOrigen)
```

Además, se ejecutan consultas SQL para actualizar los saldos de ambas cuentas:

```
UPDATE cuenta SET saldo = saldo - :monto WHERE numero = :cuentaOrigen
```

```
UPDATE cuenta SET saldo = saldo + :monto WHERE numero = :cuentaDestino
```

En el controlador, se implementó un método para manejar esta operación. Este método recibe los parámetros necesarios, como el monto a transferir y los números de cuenta de origen y destino. Luego, invoca las funciones correspondientes del repositorio de operaciones de cuenta para realizar la transferencia de dinero entre las cuentas. Al igual que con las otras operaciones, se gestionan excepciones y se redirige al usuario según sea necesario.