ENTREGA 2

Sebastián Maldonado: 202311840

Camilo Sánchez: 202113020

Esteban Hernández:

Para realizar los RF y los RFC se creó una carpeta de services. Dentro hicimos uno para el RF 10 y otro para los RFC 6 Y 7. Luego de indicar las especificaciones como el nivel de aislamiento y el tiempo de espera editamos el controller para crear los paths para el Potsdam. Una vez en el postdam corrimos las pruebas que luego de múltiples intentos funcionaron en su totalidad.

RFC 6 Y 7 10:

```
You, 39 minutes ago | 1 author (You)
@Service
public class DocumentoIngresoPService {

    @Autowired
    private DocumentoRecepcionRepository documentoRecepcionRepository;

    @Transactional(rollbackFor = Exception.class, isolation = Isolation.SERIALIZABLE)
    public List<DocumentoRecepcion> consultarDocumentosSerializable(Integer bodegaId) throws InterruptedException {
        try {
            // Primera consulta a la base de datos
            List<DocumentoRecepcion> documentos = documentoRecepcionRepository.findByIdBodega(bodegaId);
            System.out.println("Cantidad de documentos en SERIALIZABLE: " + documentos.size());

            // Simulación de operación prolongada para mantener el bloqueo
            Thread.sleep(millis:30000);

            // Segunda consulta a la base de datos (para verificar que no ocurran cambios)
            documentos = documentoRecepcionRepository.findByIdBodega(bodegaId);
            return documentos;

        } catch (InterruptedException e) {
            System.err.println("Error en consultarDocumentosSerializable: " + e.getMessage());
            Thread.currentThread().interrupt(); // Restaurar el estado de interrupción
            return null;
        }
    }

    @Transactional(rollbackFor = Exception.class, isolation = Isolation.READ_COMMITTED)
    public List<DocumentoRecepcion> consultarDocumentosReadCommitted(Integer bodegaId) throws InterruptedException {

        try {
            // Primera consulta a la base de datos
            List<DocumentoRecepcion> documentos = documentoRecepcionRepository.findByIdBodega(bodegaId);
            System.out.println("Cantidad de documentos en READ_COMMITTED: " + documentos.size());

            // Simulación de operación prolongada para mantener el bloqueo
            Thread.sleep(millis:30000);
        You, 39 minutes ago • RFC 6 Y 7
            // Segunda consulta a la base de datos (para observar posibles cambios)
            documentos = documentoRecepcionRepository.findByIdBodega(bodegaId);
            return documentos;

        } catch (InterruptedException e) {
            System.err.println("Error en consultarDocumentosReadCommitted: " + e.getMessage());
            Thread.currentThread().interrupt(); // Restaurar el estado de interrupción
            return null;
        }
    }
}
```

RF 10:

```java
@Transactional
public void registrarIngresoProductos(Integer bodegaId, Integer ordenCompraId) {
    try {
        System.out.println(x:"Registrando ingreso de productos 1");
        DocumentoRecepcion documento = new DocumentoRecepcion();
        documento.setFechaEntregado(new Date());

        Bodega bodega = bodegaRepository.findById(bodegaId)
            .orElseThrow(() -> new EntityNotFoundException(message:"Bodega no encontrada"));
        OrdenCompra ordenCompray = ordenCompraRepository.findById(ordenCompraId)
            .orElseThrow(() -> new EntityNotFoundException(message:"Orden de Compra no encontrada"));
        documento.setIdBodega(bodega);
        documento.setOrdenCompra(ordenCompray);

        documentoRecepcionRepository.save(documento);
        System.out.println(x:"Documento de recepción guardado.");

        System.out.println(x:"Recuperando productos de la orden de compra...");
        System.out.println("ID de Orden de Compra: " + ordenCompraId);
        List<InfoExtraOrden> productosOrden = infoExtraOrdenRepository.findByPK_OrdenCompra(ordenCompray);
        System.out.println("Productos recuperados de la orden: " + productosOrden.size());

        for (InfoExtraOrden producto : productosOrden) {
            System.out.println("Procesando producto: " + producto.getPK().getProducto_id());
            System.out.println("Cantidad: " + producto.getCantidad());
            System.out.println("Costo Unitario: " + producto.getCostoUnitario());

            // Actualizar inventario en InfoExtraBodega
            InfoExtraBodegaPK pk = new InfoExtraBodegaPK(bodegaRepository.findById(bodegaId)
                .orElseThrow(() -> new EntityNotFoundException(message:"Bodega no encontrada")),
                producto.getPK().getProducto_id());

            InfoExtraBodega infoExtraBodega = infoExtraBodegaRepository
                .findByPK(pk)
                Long totalExistencias - uniandes.edu.co.proyecto.Services.IngresoProductoService.registrarIngresoProductos(Integer, Integer)
            Long totalExistencias = infoExtraBodega.getTotalExistencias();
            BigDecimal costoPromedio = infoExtraBodega.getCostoPromedio();
            BigDecimal costoUnitario = producto.getCostoUnitario();
            BigDecimal cantidad = BigDecimal.valueOf(producto.getCantidad());

            // Calculate the new average cost
            System.out.println(x:"Cálculo del nuevo costo promedio...");
            System.out.println("Costo Promedio Actual: " + costoPromedio);
            System.out.println("Total Existencias Antes: " + totalExistencias);
            System.out.println("Costo Unitario del Producto: " + costoUnitario);
            System.out.println("Cantidad del Producto: " + cantidad);

            BigDecimal nuevoCostoPromedio =
                    (costoPromedio.multiply(BigDecimal.valueOf(totalExistencias))
                    .add(costoUnitario.multiply(cantidad)))
                    .divide(BigDecimal.valueOf(totalExistencias + producto.getCantidad()), RoundingMode.HALF_UP);

            System.out.println("Nuevo Costo Promedio Calculado: " + nuevoCostoPromedio);
            System.out.println("Total Existencias Después: " + (totalExistencias + producto.getCantidad()));
```

Maquina 1	T0	Maquina 2
Ejecutar el RFC 6	T1	Ejecutar RF 10

En este proceso del RFC 6 se guardaron los datos sin embargo cuando se iniciaban ambos requerimientos al tiempo según el RF 10 se guardaban los datos, pero en la consola del RFC 6 no se retornaba nada, solo corchetes vacíos.

Maquina 1	T0	Maquina 2
Ejecutar el RFC 7	T1	Ejecutar RF 10

En este proceso del RFC 7 se presentaron inconsistencias, sin embargo, se retornaron los datos en su totalidad durante la acción.

```
[
    {
        "id": 59,
        "fechaEntregado": "2024-11-03T21:32:46.000+00:00",
        "idBodega": {
            "id": 4,
            "nombre": "Bodega Bogotá Norte 2",
            "tamanio": 130.0,
            "capacidad": 650,
            "sucursal": {
                "id": 2,
                "nombre": "Sucursal Bogotá Norte",
                "telefono": "987654321",
                "direccion": "Avenida 2 # 2-2",
                "ciudad": {
                    "id": 1,
                    "nombre": "Bogotá"
                },
                "tamaño": 200.0
            }
        },
        "ordenCompra": {
            "id": 2,
            "precio": 300.5,
            "fechaEntrega": "2024-10-07T05:00:00.000+00:00",
            "estado": "ENTREGADA",
            "proveedor": {
                "id": 13,
                "nit": "890654321-8",
                "nombre": "Hogar Limpio Ltda.",
                "direccion": "Carrera 45 #23-56, Medellín",
                "personaContacto": "Luisa Fernández",
                "telefono": "3106543210"
            },
            "sucursal": {
                "id": 2,
                "nombre": "Sucursal Bogotá Norte",
                "telefono": "987654321",
                "direccion": "Avenida 2 # 2-2",
```