Programción Concurrente

0341403

Práctica Spring Batch

Por

Jorge Alejandro Cadrecha Del Rey

139846

DECLARACIÓN

Yo, Jorge Alejandro Cadrecha Del Rey, confirmo que el trabajo presentado en este informe es mío. Si la información procede de otras fuentes, confirmo que así se indica en el informe.

_____

Jorge Alejandro

CONTENTS

LIST OF FIGURES

1. INTRODUCCIÓN

1.1. ENUNCIADO DEL EJERCICIO

Objetivo:

El objetivo de esta práctica es desarrollar una aplicación de procesamiento por batchs, utilizando la programación concurrente para optimizar el rendimiento y minimizar el tiempo de procesamiento.

El programa deberá leer un gran volumen de datos, procesar estos datos de manera eficiente utilizando programación concurrente y, finalmente, almacenar los datos procesados. El contexto de los datos será definido a continuación.

Contexto:

Imaginemos que somos parte de una gran organización que maneja una enorme cantidad de datos de transacciones bancarias. Estos datos son almacenados en archivos planos y se necesita un programa que pueda procesar estos datos por la noche, cuando la base de datos no está bloqueada por otras tareas.

Además, se necesita que el programa pueda reanudar el trabajo en caso de fallos y sea capaz de manejar transacciones. Cada transacción constará de un lote de procesamiento de datos, por ejemplo, una confirmación cada 5,000 inserciones.

El programa deberá estar construido de tal manera que pueda dividirse en pequeñas tareas que se ejecuten en secuencia, generando flujos de trabajo, de modo similar a los microservicios.

Detalles del Proyecto:

El programa deberá implementar las siguientes funciones:

Leer los datos del archivo de transacciones bancarias. Procesar los datos utilizando programación concurrente. Esta etapa deberá incluir la validación de datos y el procesamiento y cálculos sobre los datos. Almacenar los datos procesados en la base

de datos. Implementar un programador que encadene las tareas y administre las operaciones de flujo. Recursos:

Para realizar esta práctica, se sugiere utilizar Java con el framework Spring Batch debido a su eficacia para el procesamiento por batchs. No obstante, puedes elegir el lenguaje de programación y las herramientas que prefieras, siempre que permitan la programación concurrente y sean adecuados para el procesamiento por batchs.

Se proporcionará un conjunto de datos de transacciones bancarias ficticias para fines de prueba.

Entrega:

La entrega de la práctica deberá incluir el código fuente de la aplicación, junto con una documentación que explique el diseño del programa, cómo se implementó la programación concurrente, y cómo se gestiona el flujo de trabajo. Además, debes incluir un análisis del rendimiento de la aplicación y cómo la programación concurrente ha optimizado el tiempo de procesamiento.

1.2. RÚBRICA PARA LA PRÁCTICA DE PROGRAMACIÓN CONCURRENTE:

1. Funcionalidad del programa (40 puntos):

- El programa puede leer correctamente los datos del archivo de transacciones bancarias (10 puntos).

- El programa procesa los datos correctamente utilizando programación concurrente. Esto incluye la validación de los datos y los cálculos realizados sobre los datos (15 puntos).

- El programa almacena correctamente los datos procesados en la base de datos (10 puntos).

- Implementación correcta del programador que encadena las tareas y administra las operaciones de flujo (5 puntos).

2. Concurrencia y Optimización (30 puntos):

- El programa implementa correctamente la programación concurrente para optimizar el tiempo de procesamiento (15 puntos).

- El programa muestra una mejora significativa en el tiempo de procesamiento en comparación con una versión no concurrente del mismo (15 puntos).

3. Manejo de errores y robustez (10 puntos):

- El programa puede manejar errores y excepciones de manera adecuada, incluyendo la capacidad de reanudar trabajos erróneos (5 puntos).

- El programa es robusto y puede manejar diferentes casos de entrada, incluyendo datos inválidos o problemáticos (5 puntos).

4. Documentación (20 puntos):

- La documentación explica correctamente el diseño del programa, cómo se implementó la programación concurrente, y cómo se gestiona el flujo de trabajo (10 puntos).

- La documentación incluye un análisis del rendimiento de la aplicación, explicando cómo la programación concurrente ha optimizado el tiempo de procesamiento (10 puntos).

Los puntos se otorgarán de manera proporcional en cada categoría en función de la correcta implementación y cumplimiento de los requisitos. En caso de presentar dudas o dificultades durante la realización de la práctica, es recomendable consultar con el profesor para obtener la orientación necesaria.

2. CÓDIGO

2.1. ESTRUTURA

El código está estructurado en varios paquetes que contienen las clases correspondientes. Dentro del proyecto, se pueden identificar los paquetes "cliente","config","cuenta","model", "transaccion" y "util". Además, se encuentran las clases "HomeController","HtmxErrorController" y"SpringBatchProjectApplication", que es la que debe ejecutarse para iniciar el programa. La siguiente imagen ilustra esta estructura.

2.2. MODIFICACIÓNES

En el proyecto, fue necesario modificar dos archivos clave para establecer la conexión con la base de datos. El primero que ajusté para su correcto funcionamiento fue el archivo "application.yml", encargado de la configuración en aplicaciones Java con Spring Boot. Este archivo se utiliza para definir propiedades de la aplicación, como configuraciones de bases de datos y perfiles de entorno, simplificando la personalización y gestión de la aplicación sin necesidad de código adicional.

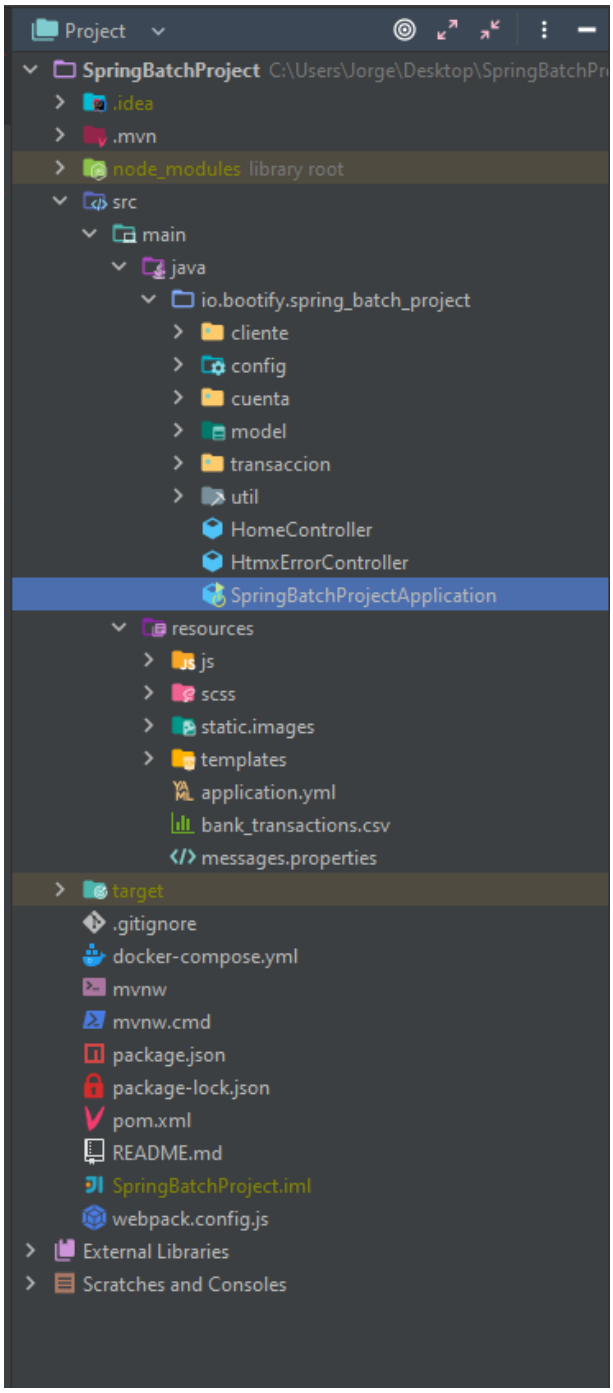

Figure 2.1: Organización del proyecto

3. INICIALIZACIÓN

Para iniciar el proyecto, lo primero que necesitamos es un entorno de desarrollo que admita la programación en Java. Posteriormente, debemos asegurarnos de tener instaladas las aplicaciones clave: PostgreSQL, encargada de la gestión de la base de datos; Docker, responsable del despliegue de la aplicación en contenedores de software. Finalmente, es necesario contar con la aplicación de Node.js para ejecutar los comandos de instalación de paquetes y para inicializar el servidor de desarrollo (DevServer).

3.1. COMANDOS

Han sido necesarios los comandos:

- `npm install`
- `npm run devserver`

3.2. IMPLEMENTACIONES

Para el funcionamiento del proyecto fueron necesarias ciertas dependencias en el `pom.xml`:

- `spring-boot-starter-web`
- `spring-boot-starter-validation`
- `spring-boot-starter-data-jpa`
- `postgresql`
- `spring-boot-starter-thymeleaf`
- `thymeleaf-layout-dialect`

- `springdoc-openapi-starter-webmvc-ui`
- `jakarta.persistence-api`
- `lombok`
- `spring-boot-devtools`
- `spring-boot-docker-compose`
- `spring-boot-starter-test`

3.3. DISEÑO DE LA BASE DE DATOS

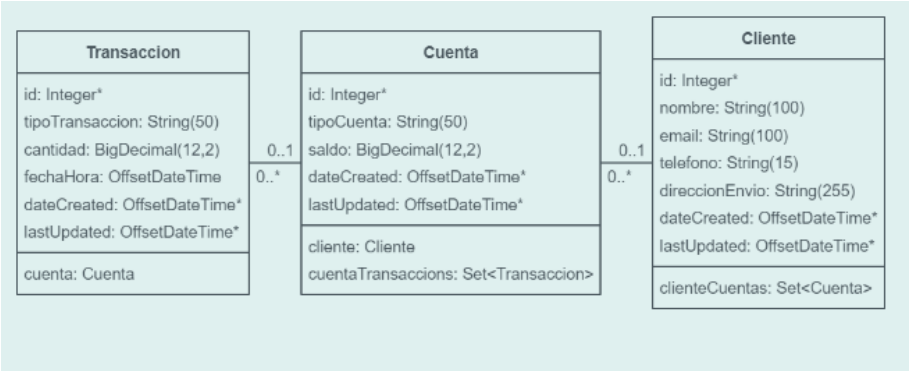

Figure 3.1: Descripción de la base de datos.

4. CONCLUSIÓN

4.1. CONCLUSION

En resumen, he adquirido conocimientos sobre la generación de código necesario para desarrollar una aplicación que utiliza Spring Batach. Además, he aprendido los pasos necesarios para la inicialización del proyecto, lo cual incluye la investigación necesaria para su desarrollo. Durante este proceso, también he adquirido habilidades en el uso de aplicaciones como PostgreSQL y Docker para garantizar una correcta puesta en marcha del sistema.