REPORTE TÉCNICO: PROYECTO SPRING BATCH - PROCESAMIENTO DE DATOS CSV

1. RESUMEN EJECUTIVO

El proyecto "batch-processing-demo" es una aplicación Spring Boot que implementa un sistema de procesamiento por lotes (batch processing) para importar datos de clientes desde un archivo CSV hacia una base de datos MySQL.

PROPÓSITO PRINCIPAL:

- Procesar archivos CSV con información de clientes
- Filtrar datos según criterios específicos (solo clientes de Estados Unidos)
- Almacenar información válida en base de datos MySQL
- Proporcionar endpoints REST para controlar la ejecución del batch

CARACTERÍSTICAS TÉCNICAS:

- Framework: Spring Boot 3.1.0 con Spring Batch 5.0.2
- Base de datos: MySQL 8.0.43
- Arquitectura: Microservicio con API REST
- Procesamiento: Asíncrono con chunks de 10 registros
- Puerto de ejecución: 9191

2. ARQUITECTURA DEL PROYECTO

El proyecto sigue la arquitectura estándar de Spring Batch con los siguientes componentes:

3. COMPONENTES PRINCIPALES

==========

3.1 CLASE PRINCIPAL (BatchProcessingDemoApplication.java)

- Anotaciones: @SpringBootApplication, @EnableJpaRepositories

- Responsabilidad: Punto de entrada de la aplicación

- Configuración: Habilita repositorios JPA automáticamente

3.2 CONFIGURACIÓN BATCH (SpringBatchConfig.java)

Componentes configurados:

- **FlatFileItemReader**: Lee archivo CSV línea por línea

- **CustomerProcessor**: Procesa y valida cada registro

- **RepositoryItemWriter**: Escribe datos válidos a la base de datos

- **Step**: Define chunk de procesamiento (10 items por transacción)

- **Job**: Orquesta la ejecución completa del batch

- **TaskExecutor**: Ejecutor asíncrono con límite de 10 hilos concurrentes

3.3 PROCESADOR DE DATOS (CustomerProcessor.java)

Lógica de negocio implementada:

```java

- Validación de campos obligatorios (firstName, lastName, email, country)

- Filtrado por país (solo "United States")

- Validación de valores nulos y vacíos

- Retorna null para registros no válidos (se descartan automáticamente)

```

3.4 CONTROLADOR REST (JobController.java)

Endpoint disponible:

- **POST /jobs/importCustomers**: Ejecuta el job de importación

- Manejo de errores específicos con ResponseEntity

- Parámetros únicos por ejecución (timestamp)

3.5 ENTIDAD DE DATOS (Customer.java)

Campos mapeados:

- ID (Primary Key)

- firstName, lastName, email (campos validados)

- gender, contactNo, country, dob

- Anotaciones JPA para mapeo a tabla CUSTOMERS_INFO

4. FLUJO DE PROCESAMIENTO

4.1 FLUJO COMPLETO DEL BATCH

1. **TRIGGER**: Llamada REST POST a /jobs/importCustomers

2. **INICIALIZACIÓN**: JobLauncher crea nueva instancia de Job

3. **LECTURA**: FlatFileItemReader lee customers.csv

4. **PROCESAMIENTO**: CustomerProcessor valida y filtra datos

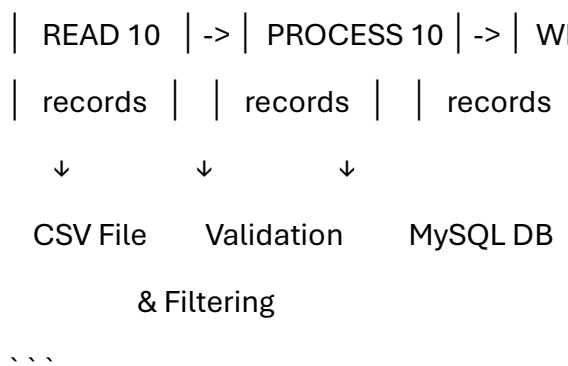
5. **ESCRITURA**: RepositoryItemWriter guarda en MySQL

6. **FINALIZACIÓN**: Job reporta estadísticas de ejecución

4.2 PROCESAMIENTO POR CHUNKS

```
```

Chunk Size: 10 registros




```

```

4.3 CRITERIOS DE FILTRADO

El sistema aplica los siguientes filtros:

- **País**: Solo registros con country = "United States"

- **Campos obligatorios**: firstName, lastName, email no pueden estar vacíos

- **Validación nulos**: Todos los campos principales deben tener valores

5. TECNOLOGÍAS UTILIZADAS

5.1 FRAMEWORK PRINCIPAL

- **Spring Boot 3.1.0**: Framework de aplicación

- **Spring Batch 5.0.2**: Motor de procesamiento por lotes

- **Spring Data JPA**: Acceso a datos simplificado

- **Spring Web**: Endpoints REST

5.2 BASE DE DATOS

- **MySQL 8.0.43**: Sistema de gestión de base de datos

- **HikariCP**: Pool de conexiones de alto rendimiento

- **Hibernate 6.2.2**: ORM para mapeo objeto-relacional

5.3 HERRAMIENTAS DE DESARROLLO

- **Java 17**: Lenguaje de programación
- **Maven**: Gestión de dependencias y construcción
- **Lombok**: Reducción de código boilerplate
- **Tomcat 10.1.8**: Servidor de aplicaciones embebido

6. CONFIGURACIÓN Y DEPENDENCIAS

6.1 CONFIGURACIÓN DE BASE DE DATOS (application.properties)

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/javatechie
spring.datasource.username=root
spring.datasource.password=admin123
spring.jpa.hibernate.ddl-auto=update
spring.batch.jdbc.initialize-schema=ALWAYS
spring.batch.job.enabled=false
spring.jpa.open-in-view=false
server.port=9191
```

6.2 DEPENDENCIAS MAVEN (pom.xml)

Dependencias principales:

- spring-boot-starter-batch

- spring-boot-starter-data-jpa

- spring-boot-starter-web

- mysql-connector-j

- lombok

6.3 ESQUEMA DE BASE DE DATOS

El proyecto incluye:

- **Tablas Spring Batch**: Para metadatos de ejecución (BATCH_JOB_*, BATCH_STEP_*)

- **Tabla de aplicación**: CUSTOMERS_INFO para almacenar datos procesados

- **Índices y restricciones**: Para integridad referencial

7. CORRECCIONES IMPLEMENTADAS

Durante la implementación se identificaron y corrigieron

8. CONCLUSIONES Y RECOMENDACIONES

8.2 RENDIMIENTO

- **Throughput**: Procesamiento en chunks de 10 registros

- **Concurrencia**: Hasta 10 hilos simultáneos configurados

- **Memoria**: Uso eficiente con procesamiento streaming

- **Base de datos**: Pool de conexiones optimizado con HikariCP

8 CASOS DE USO RECOMENDADOS

Este proyecto es ideal para:

- Migración de datos de sistemas legacy

- Procesamiento nocturno de archivos CSV

- Integración con sistemas externos vía archivos

- Procesos ETL (Extract, Transform, Load) simples

INFORMACIÓN DEL PROYECTO

Nombre: batch-processing-demo

Versión: 0.0.1-SNAPSHOT

MYSQL: JavaTechie

Framework: Spring Boot 3.1.0

Java Version: 17

Fecha de análisis: 01 de Agosto, 2025