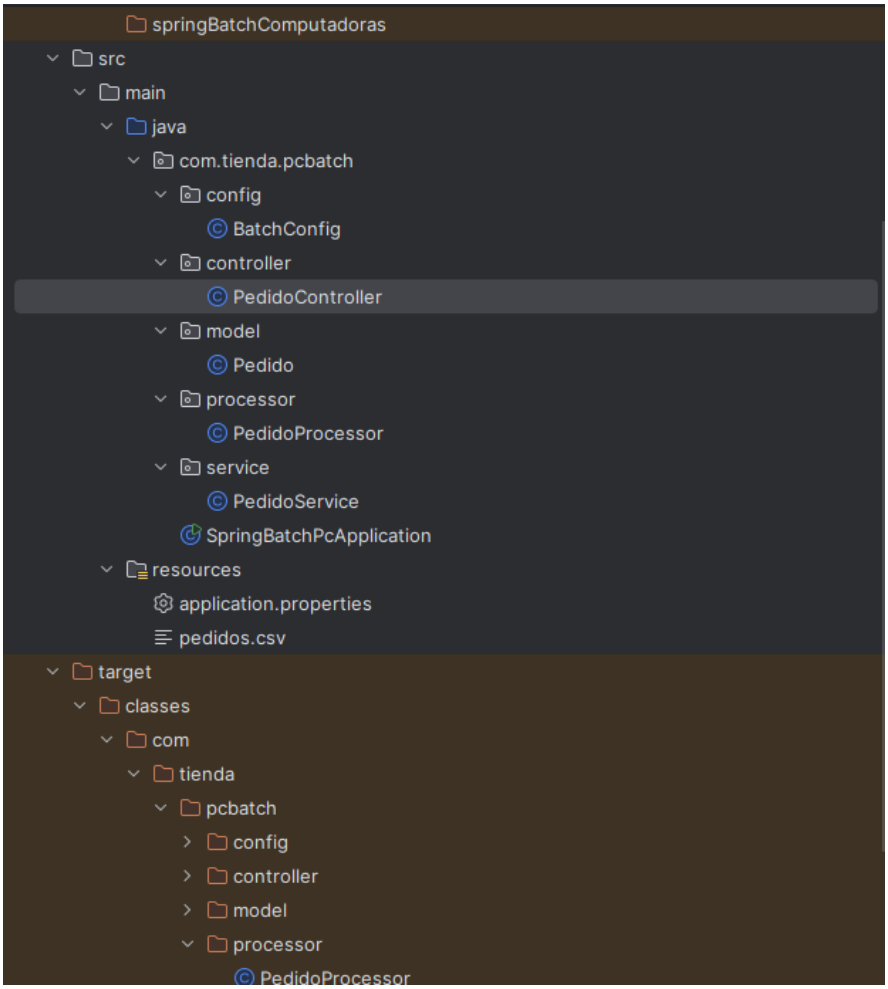
Proyecto SprintBatch con Sring Boot

Una visualización de la arquitectura para el procesamiento de archivos CSV.

Este proyecto implementa un proceso batch usando Spring Boot y Spring Batch, enfocado en la carga y procesamiento de pedidos desde un archivo CSV hacia una base de datos MySQL. Además, se incluye un servicio REST para la subida de archivos CSV mediante un endpoint /api/pedidos

Estructura del Proyecto



El flujo principal consiste en:

1. Lectura del archivo CSV.
2. Procesamiento de cada registro (por ejemplo, transformaciones o validaciones).
3. Escritura de los datos en la tabla pedidos de MySQL.
4. Manejo de errores y validación de integridad de dato

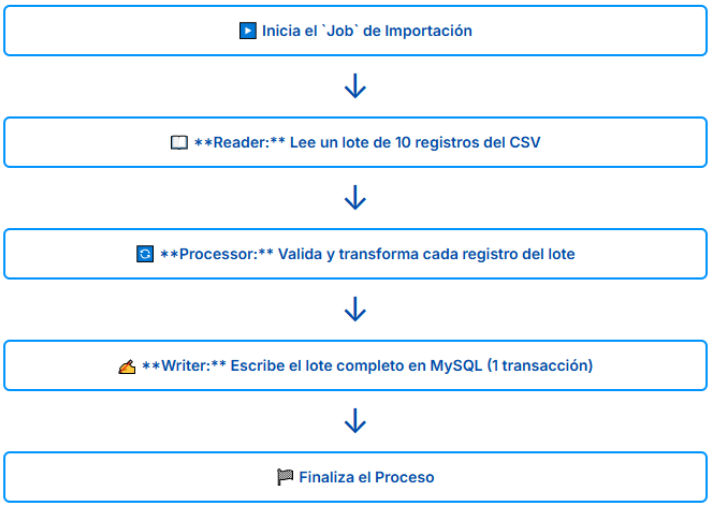
Arquitectura de Alto Nivel

El sistema está diseñado en capas, garantizando la separación de responsabilidades y la escalabilidad. El flujo de datos es claro y eficiente, desde la subida del archivo por el usuario hasta su persistencia final en la base de datos.

3. Configuración de Spring Batch (BatchConfig.java)

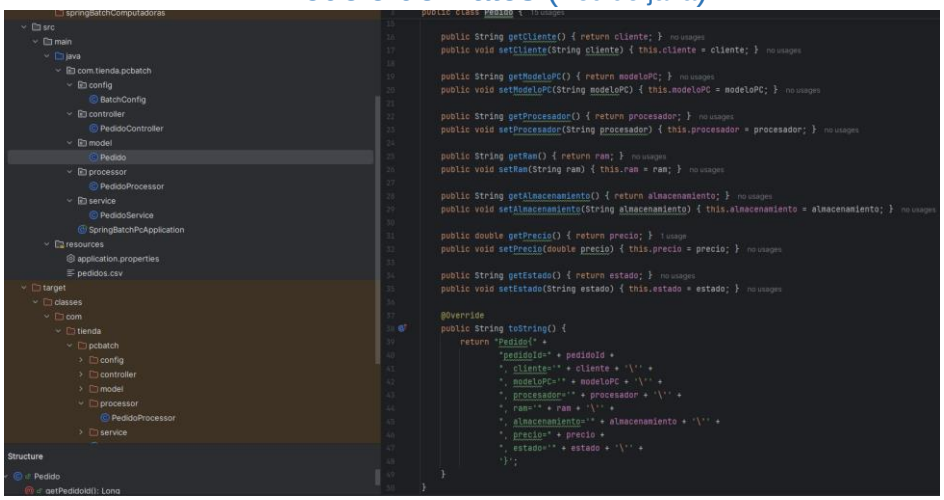
El método principal y recomendado. Procesa datos en lotes transaccionales (`chunks`), ofreciendo tolerancia a fallos, reinicio automático y un rendimiento óptimo para grandes volúmenes de datos.

3.1 Flujo 1: Proceso Robusto con Spring Batch



- ✓ FlatFileItemReader permite leer CSV automáticamente.
- ✓ chunk(10) procesa 10 registros por transacción.
- ✓ JdbcBatchItemWriter realiza la inserción en la base de datos usando JDBC.
- ✓ PedidoProcessor puede contener lógica de validación o transformación.

4. Modelo de Datos (Pedido.java)



Controlador REST (PedidoController.java)

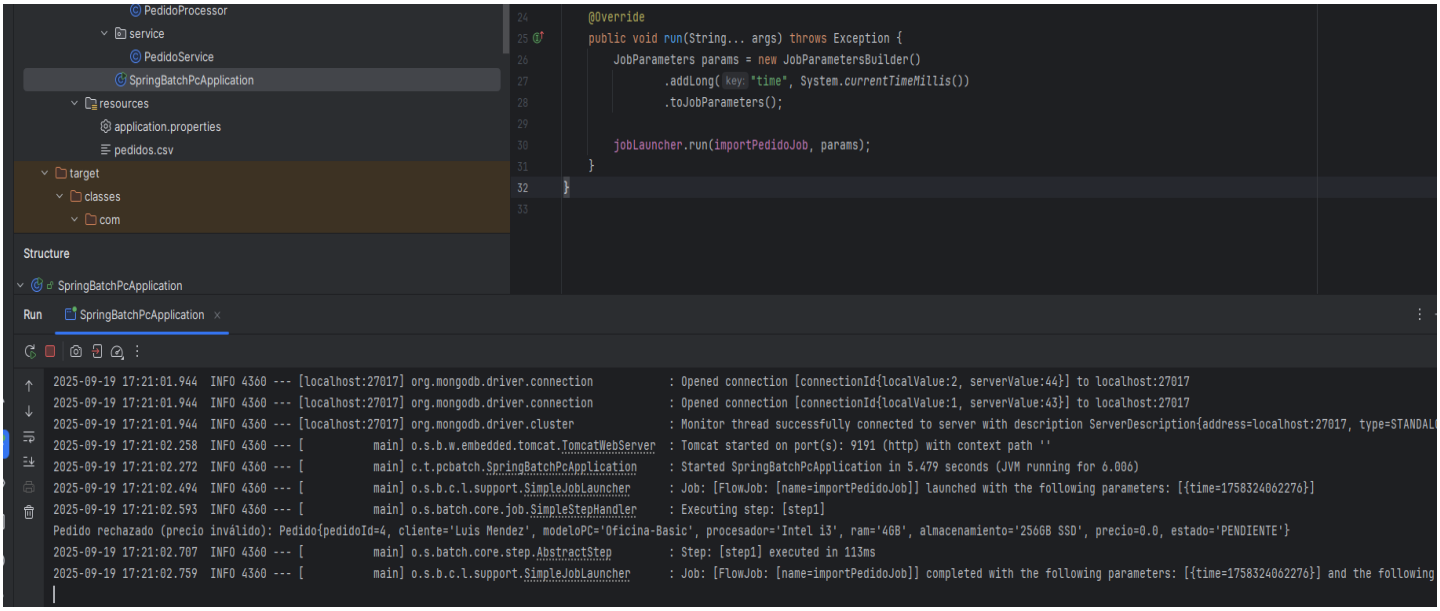
- ✓ El servicio procesa el archivo línea por línea.
- ✓ Endpoint POST /api/pedidos/upload recibe un archivo CSV.
- ✓ Llama al servicio PedidoService para procesar e insertar los datos.

```java
package com.tienda.pcbatch.controller;

import com.tienda.pcbatch.model.Pedido;
import com.tienda.pcbatch.service.PedidoService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

@RestController  no usages
@RequestMapping("/api/pedidos")
public class PedidoController {

    @Autowired  1 usage
    private PedidoService pedidoService;

    @PostMapping("/upload")  no usages
    public ResponseEntity<String> uploadCSV(@RequestParam("file") MultipartFile file) {
        try {
            pedidoService.processCSV(file);
            return ResponseEntity.ok( body: "Archivo procesado correctamente");
        } catch (Exception e) {
            return ResponseEntity.badRequest().body("Error al procesar CSV: " + e.getMessage());
        }
    }
}
```

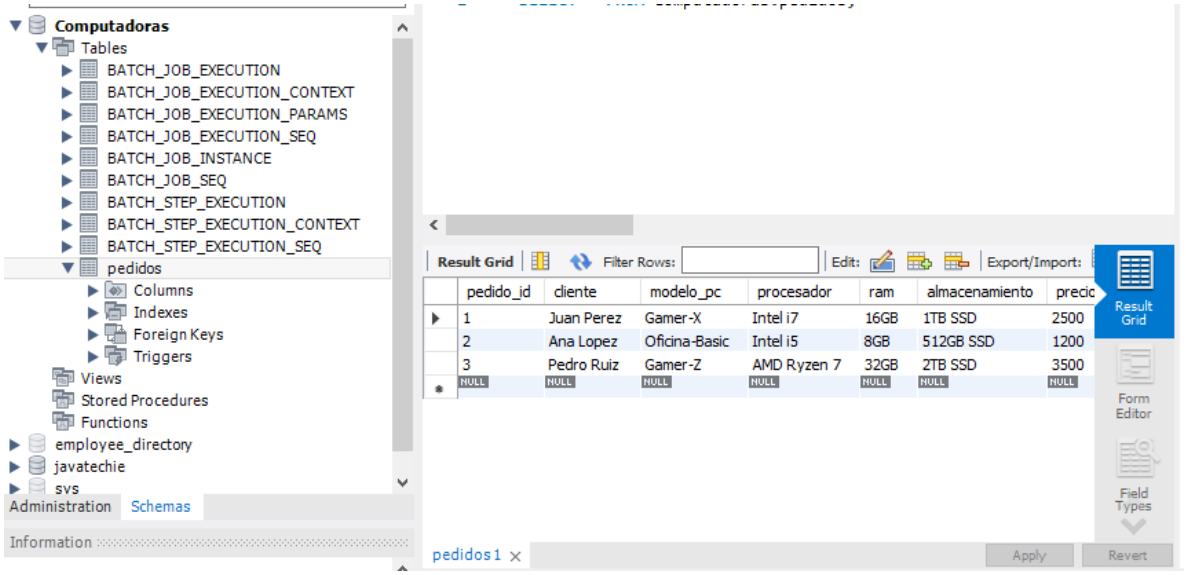
Flujo 2: Carga Manual vía Servicio REST

- ✓ Inserta cada registro en la base de datos usando JdbcTemplate.
- ✓ Se sube un archivo CSV en properties y se hace una consulta con el post para poder ver los registros hechos /api/pedidos
- ✓ PedidoService procesa cada registro:
- ✓ Valida número de columnas.
- ✓ Inserta datos en MySQL.

Pruebas y Resultados



Efectivamente está corriendo sin ningún problema



Configuración de Properties

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url = jdbc:mysql://localhost:3306/Computadoras
spring.datasource.username = root
spring.datasource.password = xideral234
spring.jpa.show-sql = true
spring.jpa.hibernate.ddl-auto = update
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
server.port=9191
spring.batch.initialize-schema=ALWAYS

#disabled job run at startup
spring.batch.job.enabled=false
```