Universidad De Las
Americas

Examen Práctico

Progreso 1 – P2

INTEGRACION DE SISTEMAS

Ing. en Software

Oscar Rodriguez

Contenido

TALLER 1

INTEGRACIÓN MEDIANTE TRANSFERENCIA DE ARCHIVOS CON APACHE CAMEL

Objetivo

El estudiante aprenderá a implementar un flujo de integración simple que lea un archivo CSV desde una carpeta de entrada, transforme su contenido y lo mueva a una carpeta de salida, aplicando el patrón clásico de integración por transferencia de archivos, usando Apache Camel.

Contexto

"El caso de TiendaSol"

La empresa TiendaSol, una cadena minorista, tiene dos sistemas antiguos:

- Un sistema de ventas (exporta las transacciones diarias en un archivo CSV).
- Un sistema de inventario, que debe recibir esa información para actualizar existencias.

Hasta ahora, los empleados copian manualmente el archivo ventas.csv de una carpeta a otra todos los días. El objetivo es automatizar ese proceso con una pequeña integración que lea transforme y mueva los archivos usando Apache Camel.

Requerimientos técnicos

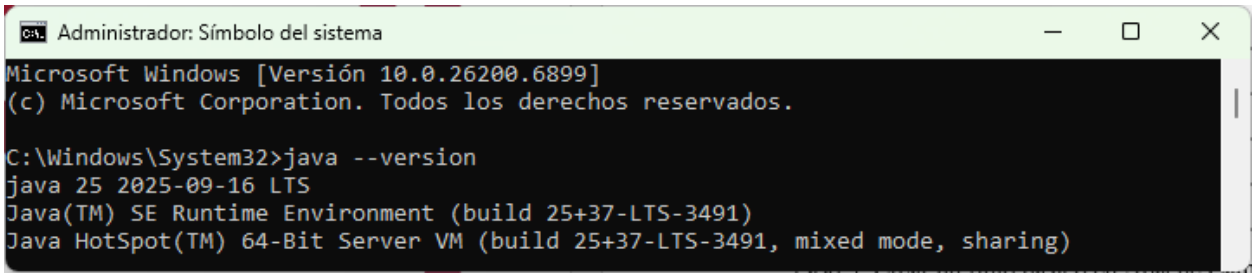
- Sistema operativo: Windows.
- Java 11 o superior instalado.
- Editor de texto o IDE (Visual Studio Code).

Paso a paso del taller

Paso 1: Instalación de herramientas

0.1.Instalar java

1. Verificar si tienes Java versión 11 o superior:



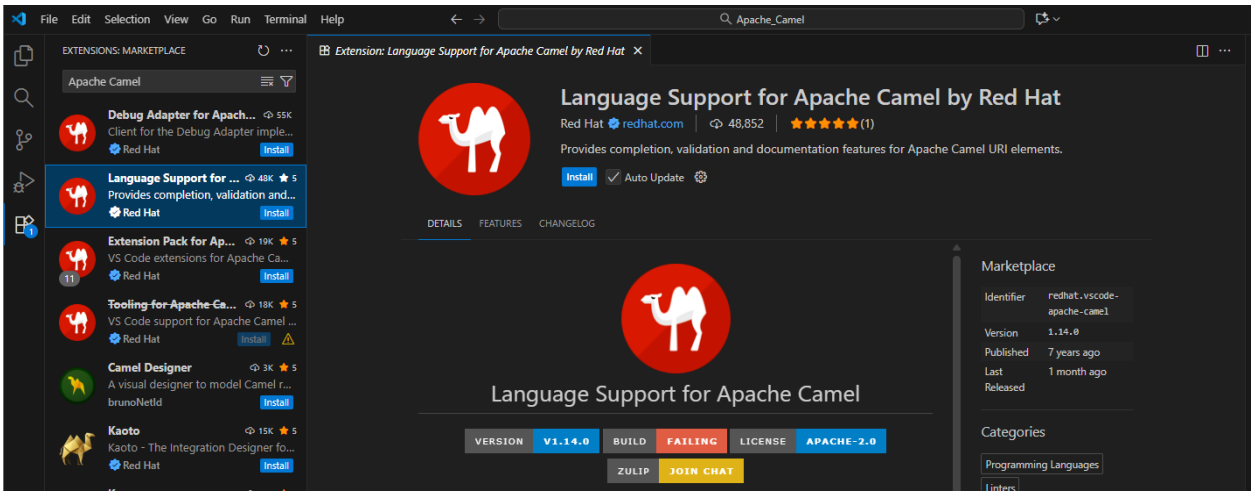
0.2.Descargar Apache Camel (Standalone)

1. Entra la página:
   https://camel.apache.org/download/
2. Descarga el archivo .zip o tar.gz de la versión estable (Camel 3.x o 4.x).
3. Descomprime el contenido en tu carpeta de trabajo:
   C:\Cursos\IntegracionSistemas\camel-lab\
4. Verifica la instalación:
   ```
   cd C:\Desarollo\IntegracionSistemas\camel-lab
   dir
   ```
   Deberías ver carpetas como bin, lib, etc.

0.3.Instalar un editor

En VS Code, instala la extensión:

"Apache Camel by Red Hat" → ayuda a visualizar rutas Camel.



Paso 2: Crear estructura de carpetas del flujo

Crea las siguientes carpetas dentro de tu proyecto:

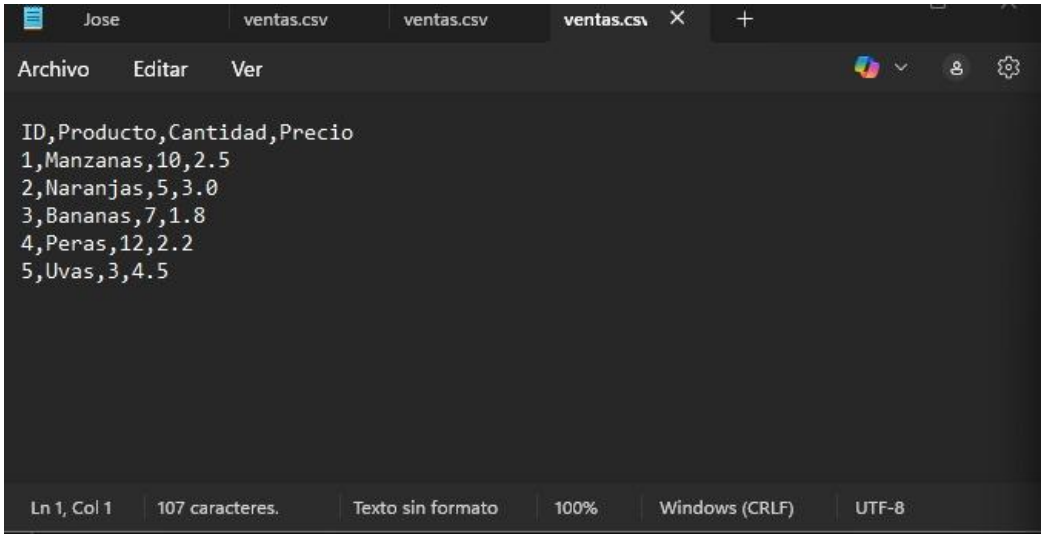
```
C:\Desarollo\IntegracionSistemas\camel-labs\lab01\
├── input\
├── output\
└── logs\
```

Estas carpetas simulan:

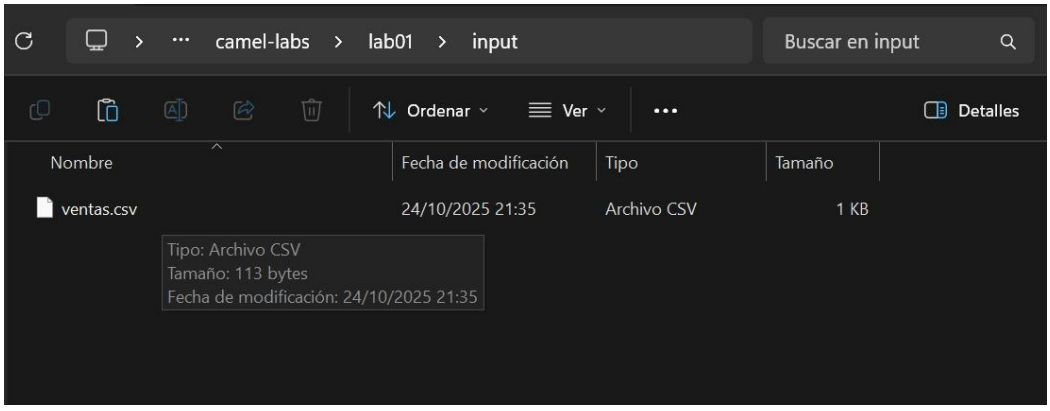
- input: carpeta donde el sistema de ventas deja los archivos.
- output: carpeta donde el sistema de inventario los recibe.
- logs: carpeta para registrar actividades.

Paso 3: Crear un archivo CSV de ejemplo

Dentro de la carpeta input, crea un archivo llamado ventas.csv con el siguiente contenido:



Este será el archivo que nuestro flujo integrará.



Paso 4: Crear un flujo básico en Apache Camel

Crea un archivo llamado FileTransferRoute.java con el siguiente contenido:

```java
package edu.udla.isw;

import org.apache.camel.builder.RouteBuilder;
import org.apache.camel.main.Main;

public class App extends RouteBuilder {

    public static void main(String[] args) throws Exception {
        Main main = new Main();
        main.configure().addRoutesBuilder(new App());
        main.run(args);
    }

    @Override
    public void configure() {
        from("file:/C:/Users/Jose Perez/Desarrollo/IntegracionSistemas/camel-labs/lab01/input?noop=true")
            .log("Procesando archivo: ${file:name}")
            .to("file:/C:/Users/Jose Perez/Desarrollo/IntegracionSistemas/camel-labs/lab01/output");
    }
}
```

Explicación: - `from("file:...input?noop=true")` → Lee archivos de la carpeta `input` sin borrarlos.
- `.log(...)` → Muestra el nombre del archivo que esta procesando.
- `.to("file:...output")` → Copia el archivo procesado a la carpeta output.

Paso 5: Ejecutar el flujo

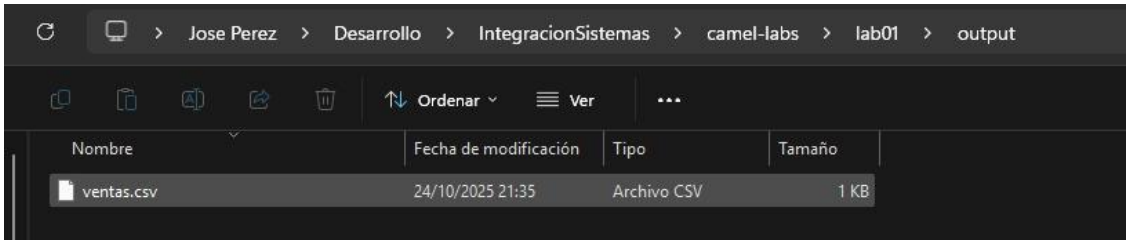
En tu terminal:

```
cd C:\Desarrollo\IntegracionSistemas\camel-lab\
javac -cp "lib/*" FileTransferRoute.java
java -cp ".;lib/*" FileTransferRoute
```

Deberías ver en la consola:

```
[main] INFO org.apache.camel.impl.engine.AbstractCamelContext -     Started route1 (file:///C:/Users/Jose%20Perez/Desarrollo/IntegracionSistemas/camel-labs/lab
01/input)
[main] INFO org.apache.camel.impl.engine.AbstractCamelContext - Apache Camel 4.7.0 (camel-1) started in 16ms (build:0ms init:0ms start:16ms)
[Camel (camel-1) thread #1 - file:///C:/Users/Jose%20Perez/Desarrollo/IntegracionSistemas/camel-labs/lab01/input] INFO route1 - Procesando archivo: ventas.csv
```

Y al revisar la carpeta output, encontrarás una copia del archivo ventas.csv.

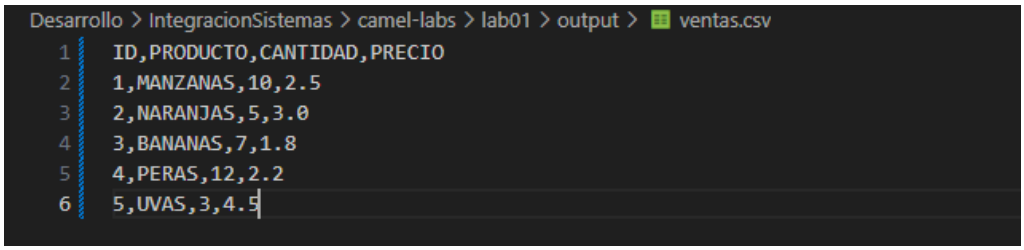


Paso 6: Agregar transformación

Queremos transformar el contenido antes de moverlo: convertir los nombres de productos a mayúsculas. Modifica el código:

```
13
14      @Override
15      public void configure() {
16          from("file:/C:/Users/Jose Perez/Desarrollo/IntegracionSistemas/camel-labs/lab01/input?noop=true")
17              .log("Procesando archivo: ${file:name}")
18              .transform().simple("${body.toUpperCase()}")
19              .to("file:/C:/Users/Jose Perez/Desarrollo/IntegracionSistemas/camel-labs/lab01/output");
20      }
21  }
22
```

Ahora, el archivo en la carpeta output mostrará los nombres en mayúsculas.



Paso 7: Ruta extra y filtros

- Agrega una segunda ruta que mueva los archivos procesados a una carpeta “archived”.

Desarrollo > IntegracionSistemas > camel-labs > lab01 > camel-lab > src > main > java > edu > udla > isw > App.java > ...

```java
package edu.udla.isw;

import org.apache.camel.builder.RouteBuilder;
import org.apache.camel.main.Main;

public class App extends RouteBuilder {

    public static void main(String[] args) throws Exception {
        Main main = new Main();
        main.configure().addRoutesBuilder(new App());
        main.run(args);
    }

    @Override
    public void configure() {
        from("file:/C:/Users/Jose Perez/Desarrollo/IntegracionSistemas/camel-labs/lab01/output?noop=true")
            .log("Procesando archivo: ${file:name}")
            .to("file:/C:/Users/Jose Perez/Desarrollo/IntegracionSistemas/camel-labs/lab01/archived");
    }
}

```

Desarrollo > IntegracionSistemas > camel-labs > lab01 > archived > ventas.csv

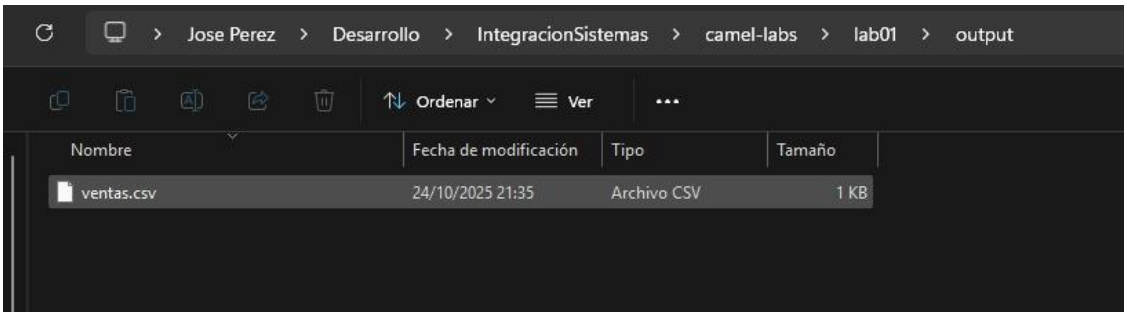
```
ID,Producto,Cantidad,Precio
1,Manzanas,10,2.5
2,Naranjas,5,3.0
3,Bananas,7,1.8
4,Peras,12,2.2
5,Uvas,3,4.5

```

- Registra un log adicional con fecha y hora del procesamiento.
- Experimenta con filtros, por ejemplo:

```
from("file:input?noop=true")
.filter(header("CamelFileName").endsWith(".csv"))
      .to("file:output");
```

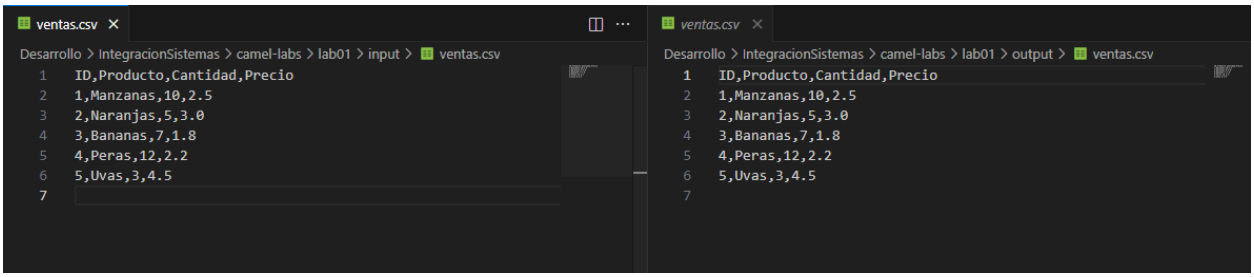
Paso 8: Validar y reflexionar

1. Verifica que el archivo ventas.csv se haya copiado correctamente a output.



2. Revisa la consola: Camel debe registrar el proceso.

3. Abre ambos archivos (input y output) y compara.



4. En el Informe correspondiente responde las siguientes preguntas:
   - ¿Qué representa el patrón “File Transfer”?

   El patrón File Transfer se utiliza para intercambiar información entre diferentes sistemas mediante el envío de archivos. Básicamente, permite transferir datos de un lugar a otro cuando los sistemas no están conectados directamente, usando los archivos como medio de comunicación entre ellos.

   - ¿Qué limitaciones tiene este enfoque?
     a. Necesita contar con un espacio físico o lógico donde se almacenen los archivos.
     b. No permite un control en tiempo real, ya que el proceso ocurre de forma diferida.
     c. Puede tener problemas de rendimiento si se manejan grandes cantidades de datos o muchos procesos simultáneos.
     d. Existe el riesgo de generar duplicados o errores si no se gestionan correctamente los indicadores de procesamiento.
   - ¿En qué escenarios reales sería útil?

a. En tareas que requieren procesar archivos por lotes, como reportes o facturación periódica.
b. En la conexión de sistemas antiguos que no cuentan con APIs o servicios modernos.
c. En la automatización de transferencias de datos entre plataformas que no están directamente integradas, como el intercambio de información entre un sistema ERP y un CRM.

Código fuente:

URL:
https://github.com/JosecoLoco/Apache_Camel/tree/main/Desarrollo/IntegracionSistemas/camel-labs

Conclusión

Este taller permitió comprender de forma práctica cómo funciona el patrón de integración por transferencia de archivos (File Transfer) y su implementación con Apache Camel. A través del ejercicio, se logró automatizar un proceso manual de lectura, transformación y envío de archivos CSV entre dos sistemas, aplicando conceptos de rutas, filtros y transformaciones de datos.

La experiencia demostró que, aunque este patrón es simple, sigue siendo útil en entornos donde existen sistemas antiguos o sin conexión directa mediante APIs. Sin embargo, también se evidencian limitaciones como la falta de inmediatez y control de errores en tiempo real.

En general, el taller permitió fortalecer conocimientos sobre integración de sistemas y middleware, mostrando cómo herramientas como Camel pueden mejorar la eficiencia y confiabilidad en el intercambio de información entre aplicaciones empresariales.

Recomendación

Creo que es importante que los estudiantes y profesionales se acostumbren a usar herramientas como Apache Camel y otras tecnologías modernas de integración, porque facilitan mucho la transferencia de archivos y la comunicación entre sistemas diferentes.

Más allá de solo mover archivos, es clave aprender a manejar la seguridad, los errores y la trazabilidad de la información. También es recomendable estar al tanto de nuevas tecnologías y métodos de integración que permitan conectar sistemas antiguos con aplicaciones más modernas. Practicar con estos escenarios ayuda a entender cómo diseñar soluciones más eficientes y confiables en entornos reales.