



# **FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS**

*Universidad de las Américas*

## **INTEGRACIÓN DE SOFTWARE**

### **PRUEBA PRÁCTICA PROGRESO 1**

Camila Cabrera

*29 de octubre de 2025*

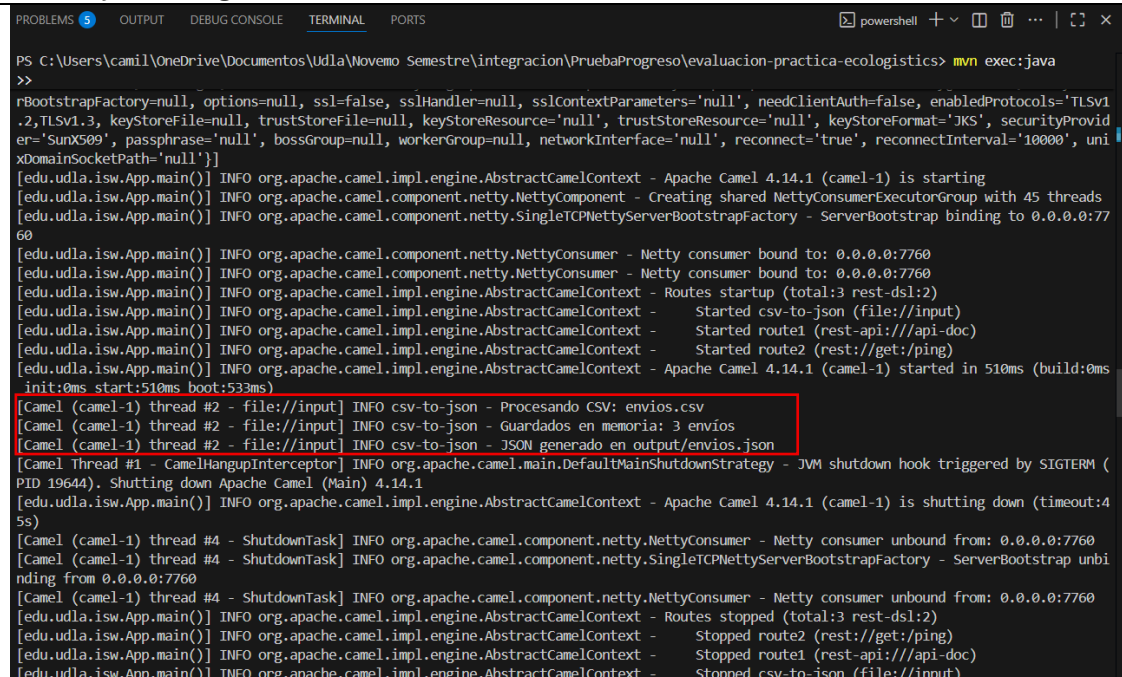
## Código fuente

Link repo github

[CamilaACT/evaluacion-practica-ecologistics](#)

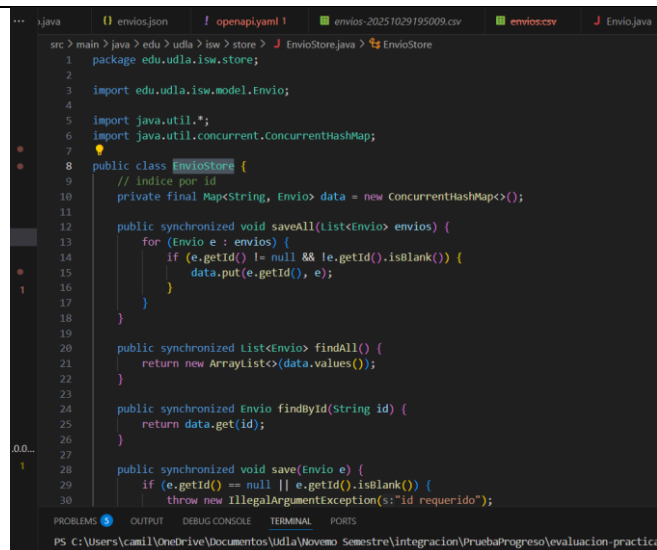
## Captura de pantalla

### Mensajes de log en consola



```
PS C:\Users\camil\OneDrive\Documentos\Udla\Novemo Semestre\integracion\PruebaProgreso\evaluacion-practica-ecologistics> mvn exec:java
>>
rBootstrapFactory=null, options=null, ssl=false, sslHandler=null, sslContextParameters='null', needClientAuth=false, enabledProtocols='TLSv1
.2,TLSv1.3, keyStoreFile=null, trustStoreFile=null, keyStoreResource='null', trustStoreResource='null', keyStoreFormat='JKS', securityProvid
er='SunX509', passphrase='null', bossGroup=null, workerGroup=null, networkInterface='null', reconnect='true', reconnectInterval='10000', uni
xDomainSocketPath='null'])
[edu.udla.isw.App.main()] INFO org.apache.camel.impl.engine.AbstractCamelContext - Apache Camel 4.14.1 (camel-1) is starting
[edu.udla.isw.App.main()] INFO org.apache.camel.component.netty.NettyComponent - Creating shared NettyConsumerExecutorGroup with 45 threads
[edu.udla.isw.App.main()] INFO org.apache.camel.component.netty.SingleTCPNettyServerBootstrapFactory - ServerBootstrap binding to 0.0.0.0:77
60
[edu.udla.isw.App.main()] INFO org.apache.camel.component.netty.NettyConsumer - Netty consumer bound to: 0.0.0.0:7760
[edu.udla.isw.App.main()] INFO org.apache.camel.component.netty.NettyConsumer - Netty consumer bound to: 0.0.0.0:7760
[edu.udla.isw.App.main()] INFO org.apache.camel.impl.engine.AbstractCamelContext - Routes startup (total:3 rest-dsl:2)
[edu.udla.isw.App.main()] INFO org.apache.camel.impl.engine.AbstractCamelContext - Started csv-to-json (file:///input)
[edu.udla.isw.App.main()] INFO org.apache.camel.impl.engine.AbstractCamelContext - Started route1 (rest-api:///api-doc)
[edu.udla.isw.App.main()] INFO org.apache.camel.impl.engine.AbstractCamelContext - Started route2 (rest://get:/ping)
[edu.udla.isw.App.main()] INFO org.apache.camel.impl.engine.AbstractCamelContext - Apache Camel 4.14.1 (camel-1) started in 510ms (build:0ms
init:0ms start:510ms boot:533ms)
[Camel (camel-1) thread #2 - file:///input] INFO csv-to-json - Procesando CSV: envios.csv
[Camel (camel-1) thread #2 - file:///input] INFO csv-to-json - Guardados en memoria: 3 envios
[Camel (camel-1) thread #2 - file:///input] INFO csv-to-json - JSON generado en output/envios.json
[Camel Thread #1 - CamelHangupInterceptor] INFO org.apache.camel.main.DefaultMainShutdownStrategy - JVM shutdown hook triggered by SIGTERM (
PID 19644). Shutting down Apache Camel (Main) 4.14.1
[edu.udla.isw.App.main()] INFO org.apache.camel.impl.engine.AbstractCamelContext - Apache Camel 4.14.1 (camel-1) is shutting down (timeout:4
5s)
[Camel (camel-1) thread #4 - ShutdownTask] INFO org.apache.camel.component.netty.NettyConsumer - Netty consumer unbound from: 0.0.0.0:7760
[Camel (camel-1) thread #4 - ShutdownTask] INFO org.apache.camel.component.netty.SingleTCPNettyServerBootstrapFactory - ServerBootstrap unbi
nding from 0.0.0.0:7760
[Camel (camel-1) thread #4 - ShutdownTask] INFO org.apache.camel.component.netty.NettyConsumer - Netty consumer unbound from: 0.0.0.0:7760
[edu.udla.isw.App.main()] INFO org.apache.camel.impl.engine.AbstractCamelContext - Routes stopped (total:3 rest-dsl:2)
[edu.udla.isw.App.main()] INFO org.apache.camel.impl.engine.AbstractCamelContext - Stopped route2 (rest://get:/ping)
[edu.udla.isw.App.main()] INFO org.apache.camel.impl.engine.AbstractCamelContext - Stopped route1 (rest-api:///api-doc)
[edu.udla.isw.App.main()] INFO org.apache.camel.impl.engine.AbstractCamelContext - Stopped csv-to-json (file:///input)
```

### Modelo EnvioStore



```
src > main > java > edu > udla > isw > store > EnvioStore.java > EnvioStore
1 package edu.udla.isw.store;
2
3 import edu.udla.isw.model.Envio;
4
5 import java.util.*;
6 import java.util.concurrent.ConcurrentHashMap;
7
8 public class EnvioStore {
9     // indice por id
10    private final Map<String, Envio> data = new ConcurrentHashMap<>();
11
12    public synchronized void saveAll(List<Envio> envios) {
13        for (Envio e : envios) {
14            if (e.getId() != null && !e.getId().isBlank()) {
15                data.put(e.getId(), e);
16            }
17        }
18    }
19
20    public synchronized List<Envio> findAll() {
21        return new ArrayList<>(data.values());
22    }
23
24    public synchronized Envio findById(String id) {
25        return data.get(id);
26    }
27
28    public synchronized void save(Envio e) {
29        if (e.getId() == null || e.getId().isBlank()) {
30            throw new IllegalArgumentException("id requerido");
31        }
32    }
33}
```

## Clase App.java

```
1 package edu.udla.isw;  
2  
3 import edu.udla.isw.model.Envio;  
4 import edu.udla.isw.store.EnvioStore;  
5 import org.apache.camel.Exchange;  
6 import org.apache.camel.builder.RouteBuilder;  
7 import org.apache.camel.main.Main;  
8  
9 import java.util.ArrayList;  
10 import java.util.List;  
11  
12 public class App extends RouteBuilder {  
13  
14     Run | Debug  
15     public static void main(String[] args) throws Exception {  
16         Main main = new Main();  
17         // Bind del store en memoria para usarlo como bean  
18         main.bind("envioStore", new EnvioStore());  
19         main.configure().addRoutesBuilder(new App());  
20         main.run(args);  
21     }  
22  
23     @Override  
24     public void configure() {  
25  
26         // REST base + OpenAPI  
27         restConfiguration()  
28             .component(componentId:"netty-http")  
29             .port(port:7760)  
30             .contextPath(contextPath:"/api")  
31     }  
32 }
```

## Archivos generados

```
▼ archived  
  envios-20251029195009.csv  
▼ input  
▼ output  
  {} envios.json
```

## Archivos modificado carpeta output

```
output > {} envios.json > ...  
1 : "Guayaquil", "estado": "Pendiente"}, {"id": "004", "cliente": "Ana Mora", "direccion": "Loja", "estado": "En tránsito"}]]
```

## Archivos copiados en carpeta archived

EXPLORER

▼ EVALUACION-PRACTICA-...  
 ▼ archived  
 envios-20251029195009.csv  
 ▼ input  
 ▼ output  
 {} envios.json  
 ▼ src  
 ▼ main  
 ▼ java \ edu \ udla \ isw

App.java | {} envios.json | ! openapi.yaml 1

archived > envios-20251029195009.csv

```
1 id_envio,cliente,direccion,estado  
2 001,Juan Pérez,Quito,En tránsito  
3 002,Marta Lema,Cuenca,Entregado  
4 003,Carlos Díaz,Guayaquil,Pendiente  
5
```

## Documentación Open Api

```
{
  "openapi": "3.0.0",
  "info": {
    "title": "API EcoLogistics",
    "version": "1.0.0"
  },
  "servers": [
    {
      "url": "http://0.0.0.0:7760/api"
    }
  ],
  "tags": [
    {
      "name": "/ping"
    }
  ],
  "paths": {
    "/envios": {
      "description": "Gestión de envíos"
    },
    "/envios/{id}"
  },
  "paths": {
    "/ping": {
      "get": {
        "tags": [
          "/ping"
        ],
        "operationId": "verbi",
        "responses": {
          "default": {
            "content": {
              "application/json": {
                "schema": {
                  "type": "object",
                  "properties": {
                    "status": {
                      "type": "string",
                      "example": "ok"
                    },
                    "service": {
                      "type": "string",
                      "example": "EcoLogistics"
                    },
                    "ts": {
                      "type": "string",
                      "example": "2025-10-29 19:45:00"
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "/envios": {
      "get": {
        "tags": [
          "/envios"
        ],
        "operationId": "verbi",
        "responses": {
          "default": {
            "content": {
              "application/json": {
                "schema": {
                  "type": "object",
                  "properties": {
                    "status": {
                      "type": "string",
                      "example": "ok"
                    },
                    "service": {
                      "type": "string",
                      "example": "EcoLogistics"
                    },
                    "ts": {
                      "type": "string",
                      "example": "2025-10-29 19:45:00"
                    }
                  }
                }
              }
            }
          }
        }
      },
      "post": {
        "tags": [
          "/envios"
        ],
        "operationId": "verbi",
        "responses": {
          "default": {
            "content": {
              "application/json": {
                "schema": {
                  "type": "object",
                  "properties": {
                    "status": {
                      "type": "string",
                      "example": "ok"
                    },
                    "service": {
                      "type": "string",
                      "example": "EcoLogistics"
                    },
                    "ts": {
                      "type": "string",
                      "example": "2025-10-29 19:45:00"
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    "/envios/{id}": {
      "get": {
        "tags": [
          "/envios/{id}"
        ],
        "operationId": "verbi",
        "parameters": [
          {
            "name": "id",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "default": {
            "content": {
              "application/json": {
                "schema": {
                  "type": "object",
                  "properties": {
                    "status": {
                      "type": "string",
                      "example": "ok"
                    },
                    "service": {
                      "type": "string",
                      "example": "EcoLogistics"
                    },
                    "ts": {
                      "type": "string",
                      "example": "2025-10-29 19:45:00"
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```


```
info:
  title: API EcoLogistics
  version: 1.0.0
  description: API para gestionar envíos (lectura desde CSV → JSON → memoria).
servers:
  - url: /api
paths:
  /ping:
    get:
      summary: Comprobación de estado
      tags: [Utils]
      responses:
        '200':
          description: Servicio en línea
          content:
            application/json:
              schema:
                type: object
                properties:
                  status: { type: string, example: ok }
                  service: { type: string, example: EcoLogistics }
                  ts: { type: string, example: "2025-10-29 19:45:00" }

  /envios:
    get:
```

### 1. ¿Qué patrón de integración aplicaste y como se refleja en tu solución?

En la solución desarrollada para el prototipo de la empresa EcoLogistics, se aplicó una combinación de patrones de integración “File Transfer” y “API RESTful”, los cuales permiten implementar un flujo completo de intercambio de información entre sistemas heterogéneos.

Por un lado, el patrón File Transfer se refleja en el proceso de carga y procesamiento de archivos CSV. El sistema recibe un archivo de datos de envíos dentro de la carpeta input/, el cual representa la salida de otro sistema o módulo operativo existente. Apache Camel detecta automáticamente la presencia de este archivo, lo lee y transforma su contenido a formato JSON. Este archivo



transformado se guarda en la carpeta output/, y el CSV original se mueve a archived/ para mantener un historial de las cargas procesadas.

Por otro lado, el patrón API RESTful se utiliza para exponer los datos procesados a través de una interfaz accesible y estandarizada. Los endpoints desarrollados (GET /api/envios, GET /api/envios/{id} y POST /api/envios) permiten a otros sistemas o aplicaciones consumir la información de manera inmediata, sin necesidad de intercambio manual de correos o archivos. Esto habilita la integración directa con aplicaciones web o móviles que necesiten acceder a los registros de envíos en tiempo real.

## **2. ¿Qué ventajas identificas al pasar de File Transfer a APIs REST?**

Al migrar del intercambio de archivos mediante File Transfer hacia un modelo basado en APIs REST, la empresa obtiene múltiples ventajas tanto en el plano técnico como en la gestión de procesos.

La primera ventaja es la integridad y consistencia de la información. Mientras que los archivos CSV son susceptibles a errores de formato, duplicidad de registros o manipulaciones no controladas, las APIs REST garantizan un flujo de datos estructurado, validado y protegido mediante estándares como JSON y HTTP. Esto reduce significativamente el riesgo de corrupción o pérdida de información durante la transferencia.


En segundo lugar, el uso de APIs permite una transferencia de datos en tiempo real, eliminando la necesidad de esperar a que se generen y procesen archivos. Los sistemas pueden consultar o enviar información de manera instantánea, lo que mejora la eficiencia operativa y acelera los tiempos de respuesta ante solicitudes o actualizaciones.

Otra ventaja clave es la facilidad de integración. Las APIs RESTful son independientes de la plataforma y lenguaje de programación, por lo que pueden conectarse fácilmente con aplicaciones móviles, sistemas ERP, plataformas web o módulos analíticos. Esto fomenta la interoperabilidad y el crecimiento modular del ecosistema tecnológico de la empresa.

Finalmente, esta transición impulsa la toma de decisiones basada en datos. Al disponer de información actualizada y accesible de forma continua, la organización puede realizar análisis más detallados, detectar patrones operativos y mejorar la planificación logística, generando así un valor tangible para el negocio.

## **3. ¿Qué riesgos o limitaciones encontraste en tu enfoque?**

Si bien la solución propuesta logra eliminar uno de los puntos críticos del proceso —el cliente que debía recibir y procesar manualmente los archivos CSV—, aún persisten ciertos riesgos y limitaciones inherentes al enfoque implementado.



En primer lugar, el sistema continúa dependiendo del usuario o proceso que genera el archivo CSV original. Si este archivo contiene errores, registros incompletos o datos inconsistentes, dichos problemas se propagarán a lo largo de toda la cadena de procesamiento. Este sigue siendo el punto de falla inicial del flujo.

Para mitigar este riesgo, se podría implementar una cabecera de control dentro del archivo CSV, que indique la cantidad de registros o un identificador de verificación. De esta manera, el sistema podría validar automáticamente si el archivo procesado coincide con los datos esperados antes de convertirlo a JSON o almacenarlo, reduciendo la posibilidad de inconsistencias.

Otra limitación observada es que la solución actual trabaja con archivos intermedios y no utiliza persistencia en base de datos, lo cual restringe la escalabilidad y la capacidad de auditoría del sistema. En un escenario real de producción, donde los envíos cambian constantemente y requieren trazabilidad, sería recomendable migrar hacia una arquitectura con almacenamiento persistente (por ejemplo, una base de datos relacional o documental) y servicios que operen de forma continua.

Finalmente, para un sistema de logística o seguimiento de envíos, la actualización en tiempo real resulta fundamental. Aunque la arquitectura basada en Apache Camel es sólida, sería conveniente evolucionar hacia integraciones orientadas a eventos o mensajería en tiempo real (por ejemplo, Kafka o WebSockets), lo que permitiría ofrecer información inmediata y mejorar la eficiencia de los procesos operativos.


#### **4. ¿Como escalarías esta integración si EcoLogistics tuviera 50 sistemas distintos?**

La estrategia de escalamiento dependería directamente de la naturaleza y propósito de los sistemas involucrados, ya que no existe una única plantilla universal de integración. Antes de definir una arquitectura técnica, es indispensable realizar un análisis funcional y de negocio que determine qué información se necesita compartir, con qué frecuencia y bajo qué estándares de seguridad y consistencia.

En el contexto de este prototipo, donde se prevé integrar alrededor de 50 sistemas se supone relacionado con procesos por ejemplo de envíos, logística, flota vehicular, facturación o atención al cliente, el primer paso sería clasificar los sistemas según su criticidad, tipo de tecnología y nivel de interoperabilidad (por ejemplo, sistemas legados, web in-house, servicios en la nube, etc.).

Con base en esa categorización, se podrían aplicar diferentes enfoques de integración:

- APIs RESTful para los sistemas modernos que requieren comunicación en tiempo real.
- Mensajería asincrónica (event-driven

- 
- Conectores batch o ETL para sistemas legados o de bajo nivel de actualización.
  - Gateways o ESB (Enterprise Service Bus) si se requiere un punto central de orquestación y seguridad entre múltiples servicios.

De esta manera, la arquitectura evolucionaría hacia un ecosistema híbrido y escalable, en el que cada sistema se integra usando el método más adecuado para su contexto, manteniendo la coherencia del flujo global mediante Apache Camel o una plataforma de integración empresarial (EIP).

En resumen, el objetivo no sería integrar todos los sistemas de manera indiscriminada, sino alinear la integración con los objetivos de negocio, asegurando trazabilidad, confiabilidad y crecimiento sostenible del ecosistema tecnológico de EcoLogistics.