



FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

Universidad de las Américas

INTEGRACIÓN DE SISTEMAS

Evaluacion-practica-ecologistics

Kristiany Cerón

29 de octubre de 2025

EVIDENCIA:

Link github: <https://github.com/krissceron/evaluacion-practica-ecologistics.git>

GETS Y POST:

The screenshot displays the Postman API client interface. The top navigation bar includes 'Home', 'Workspaces', and 'API Network'. The left sidebar shows the 'Personal Workspace' with a 'New Collection' button and a list of requests under 'P1Integración'. The main panel shows a 'New Request' for a GET method to the URL 'http://localhost:8081/envios'. The 'Params' tab is active, showing a table for Query Params. The 'Body' tab is also active, showing the JSON response of the request.

Query Params

Key	Value	Description
Key	Value	Description

Body

200 OK • 174 ms • 760 B

```
{
  "id": "001",
  "cliente": "Juan Pérez",
  "direccion": "Calle 12 #45",
  "estado": ""
},
{
  "id": "002",
  "cliente": "Maria Gómez",
  "direccion": "Avenida 10 #33",
  "estado": ""
},
{
  "id": "003",
  "cliente": "Luis Mora",
  "direccion": "Carrera 8 #22",
  "estado": ""
}
```

Home workspaces API NETWORK Search Postman

Personal Workspace New Import

Collections + Search collections

New Collection

- GET New Request
- GET New Request
- GET New Request
- PIIntegración
 - GET getAll
 - GET getById
 - POST createUpdate
- Portafolio Orión

Environments

Flows

History

New Collection / New Request

GET http://localhost:8081/envios/002 Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (12) Test Results 200 OK 8 ms 615 B Save Response

JSON Preview Visualize

```
1 {
2   "id": "902",
3   "cliente": "María Gómez",
4   "direccion": "Avenida 10 #33",
5   "estado": ""
6 }
```

Home workspaces API NETWORK Search Postman

Personal Workspace New Import

Collections + Search collections

New Collection

- GET New Request
- GET New Request
- GET New Request
- PIIntegración
 - GET getAll
 - GET getById
 - POST createUpdate
- Portafolio Orión

Environments

Flows

History

New Collection / New Request

POST http://localhost:8081/envios Send

Params Authorization Headers (9) Body Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON Schema Beautify

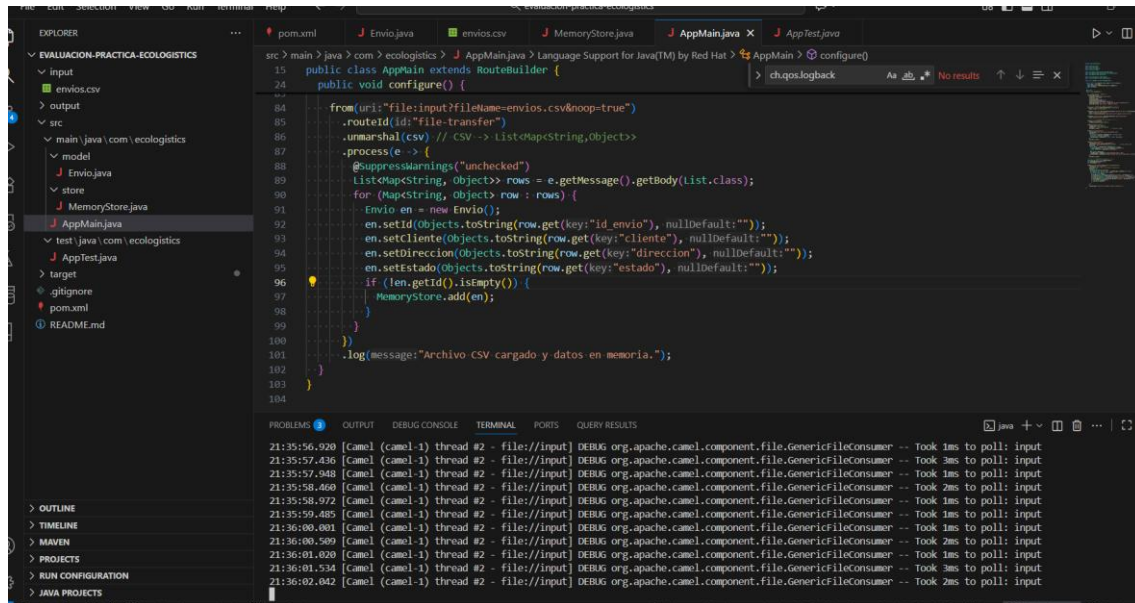
1 {
2 "cliente": "Kriss Ceiron",
3 "direccion": "Av. Central 55",
4 "estado": "En tránsito"
5 }

Body Cookies Headers (11) Test Results 201 Created 102 ms 577 B Save Response

JSON Preview Visualize

```
1 {
2   "mensaje": "Envío registrado correctamente"
3 }
```

Ejecución:



```
src > main > java > com > ecologistics > J AppMain.java > Language Support for Java(TM) by Red Hat > AppMain > configure()
15 public class AppMain extends RouteBuilder {
24 public void configure() {
84 from(uri:"file:input?fileName=envios.csv&noop=true")
85 .routeId(id:"file-transfer")
86 .unmarshal(csv) // CSV -> List<Map<String, Object>>
87 .process(e -> {
88 @SuppressWarnings("unchecked")
89 List<Map<String, Object>> rows = e.getMessage().getBody(List.class);
90 for (Map<String, Object> row : rows) {
91 Envio en = new Envio();
92 en.setId(Objects.toString(row.get("id_envio"), nullDefault:""));
93 en.setCliente(Objects.toString(row.get("cliente"), nullDefault:""));
94 en.setDireccion(Objects.toString(row.get("direccion"), nullDefault:""));
95 en.setEstado(Objects.toString(row.get("estado"), nullDefault:""));
96 if (!en.getId().isEmpty()) {
97 MemoryStore.add(en);
98 }
99 }
100 })
101 .log(message:"Archivo CSV cargado y datos en memoria.");
102 }
103 }
104 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS

```
21:35:56.920 [camel (camel-1) thread #2 - file://input] DEBUG org.apache.camel.component.file.GenericFileConsumer -- Took 1ms to poll: input
21:35:57.436 [camel (camel-1) thread #2 - file://input] DEBUG org.apache.camel.component.file.GenericFileConsumer -- Took 3ms to poll: input
21:35:57.948 [camel (camel-1) thread #2 - file://input] DEBUG org.apache.camel.component.file.GenericFileConsumer -- Took 1ms to poll: input
21:35:58.460 [camel (camel-1) thread #2 - file://input] DEBUG org.apache.camel.component.file.GenericFileConsumer -- Took 2ms to poll: input
21:35:58.972 [camel (camel-1) thread #2 - file://input] DEBUG org.apache.camel.component.file.GenericFileConsumer -- Took 1ms to poll: input
21:35:59.485 [camel (camel-1) thread #2 - file://input] DEBUG org.apache.camel.component.file.GenericFileConsumer -- Took 1ms to poll: input
21:36:00.001 [camel (camel-1) thread #2 - file://input] DEBUG org.apache.camel.component.file.GenericFileConsumer -- Took 1ms to poll: input
21:36:00.509 [camel (camel-1) thread #2 - file://input] DEBUG org.apache.camel.component.file.GenericFileConsumer -- Took 2ms to poll: input
21:36:01.020 [camel (camel-1) thread #2 - file://input] DEBUG org.apache.camel.component.file.GenericFileConsumer -- Took 1ms to poll: input
21:36:01.534 [camel (camel-1) thread #2 - file://input] DEBUG org.apache.camel.component.file.GenericFileConsumer -- Took 1ms to poll: input
21:36:02.042 [camel (camel-1) thread #2 - file://input] DEBUG org.apache.camel.component.file.GenericFileConsumer -- Took 2ms to poll: input
```

¿Qué patrón de integración aplicaste y cómo se refleja en tu solución?

En este proyecto apliqué el **patrón de integración File Transfer** combinado con una **exposición mediante API REST**.

Inicialmente, el sistema lee archivos CSV con información de envíos (patrón File Transfer), los transforma internamente en objetos Java y finalmente los expone en formato JSON a través de un servicio REST (patrón Request-Reply).

Esta integración se refleja en el flujo de Apache Camel, donde se define una ruta que consume archivos desde la carpeta input/, procesa los datos y los pone a disposición en memoria para ser consultados por los endpoints HTTP /envios y /envios/{id}.

En otras palabras, se implementa una **solución híbrida** que combina la integración por archivos con la exposición de servicios interoperables.

¿Qué ventajas identificas al pasar de File Transfer a APIs REST?

Las principales ventajas que identifiqué son la **interoperabilidad, inmediatez y escalabilidad**.


Mientras que el File Transfer depende de archivos y lotes, las APIs REST permiten acceder a los datos en tiempo real, sin necesidad de esperar a que se procese un archivo completo.

Además, REST facilita la integración entre sistemas heterogéneos mediante el uso de HTTP y JSON, eliminando dependencias de formato o plataforma.

También mejora la trazabilidad y control, ya que cada solicitud puede ser registrada, autenticada y monitoreada con mayor precisión.

¿Qué riesgos o limitaciones encontraste en tu enfoque?

El principal riesgo es que el sistema **mantiene los datos solo en memoria**, por lo que se pierden al reiniciar la aplicación.



Además, la lectura desde archivos puede generar inconsistencias si varios procesos intentan modificarlos simultáneamente.

Otra limitación es la **falta de persistencia y seguridad**: al ser una integración académica, no se implementó autenticación, cifrado ni validaciones de datos extensivas. Por último, el enfoque actual es monolítico, lo cual puede dificultar la escalabilidad si el volumen de datos o las peticiones crecen significativamente.

¿Cómo escalarías esta integración si EcoLogistics tuviera 50 sistemas distintos?

Para escalar la integración a un entorno con 50 sistemas, aplicaría una **arquitectura basada en microservicios y mensajería asíncrona**.

En lugar de depender de archivos locales, utilizaría una cola de mensajes (como Kafka o RabbitMQ) que permita procesar los datos de forma paralela y desacoplada.

Cada sistema tendría su propio microservicio responsable de publicar y consumir eventos, y se gestionaría la comunicación mediante un **API Gateway** y un **bus de integración empresarial (EIP)**.

Esto permitiría soportar grandes volúmenes de transacciones, asegurar la consistencia de los datos y facilitar el mantenimiento y la evolución del sistema sin afectar a los demás.