

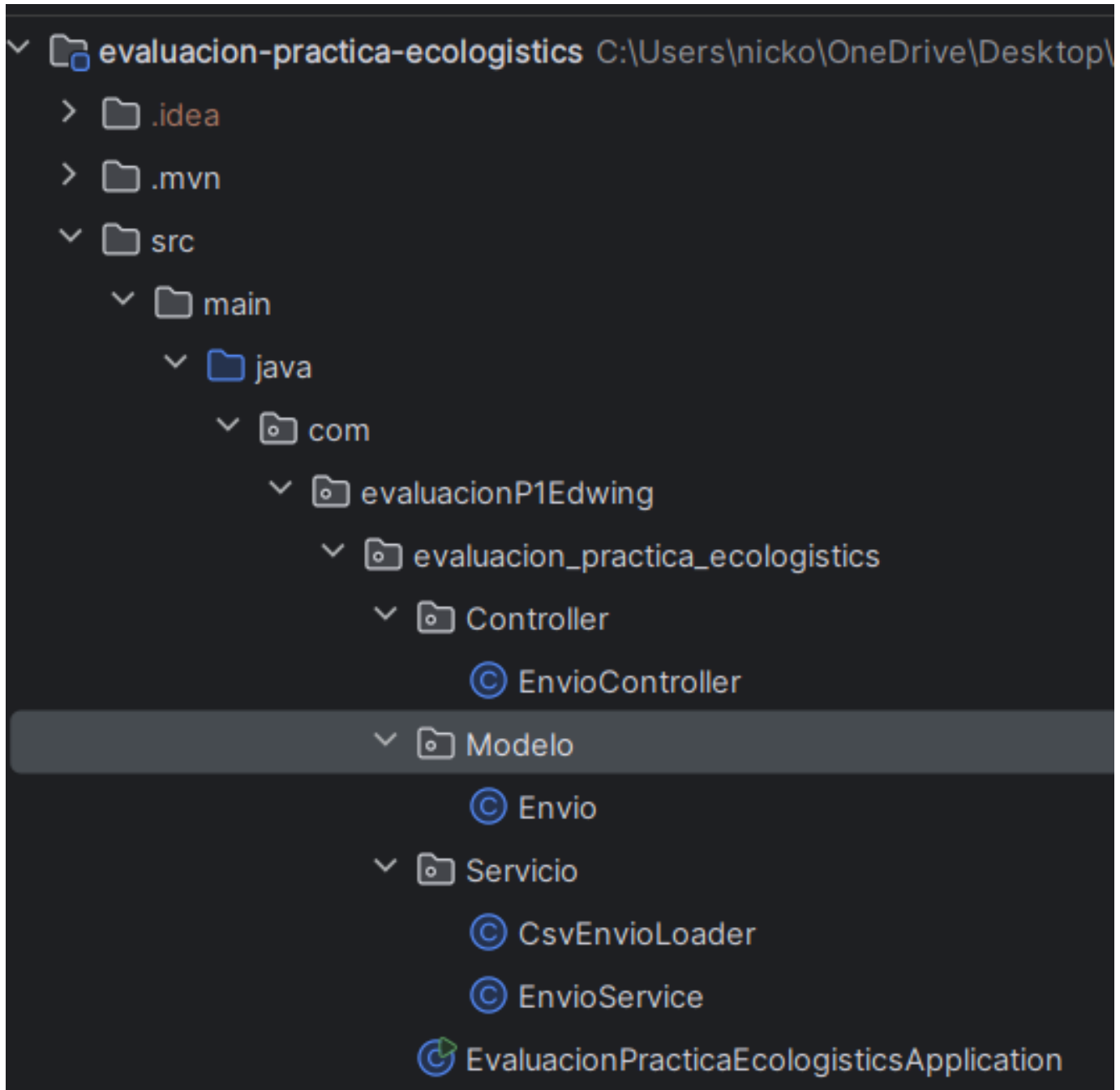
Edwing Garcia

Integración De Sistemas

---

Github: <https://github.com/EdwingGarcia/Eamen-P1-Edwing-Integraci-n>

Estructura Proyecto



## Modelo Envio

```
import com.fasterxml.jackson.annotation.JsonProperty;

public class Envio { 17 usages  Edwing Garcia
    @JsonProperty("id") 3 usages
    private String id;
    private String cliente; 3 usages
    private String direccion; 3 usages
    private String estado; 3 usages

    public Envio() {} no usages  Edwing Garcia

    public Envio(String id, String cliente, String direccion, String estado) { 1 usage  Edwing Garcia
        this.id = id;
        this.cliente = cliente;
        this.direccion = direccion;
        this.estado = estado;
    }

    public String getId() { return id; } 5 usages  Edwing Garcia
    public void setId(String id) { this.id = id; } 1 usage  Edwing Garcia

    public String getCliente() { return cliente; } no usages  Edwing Garcia
    public void setCliente(String cliente) { this.cliente = cliente; } no usages  Edwing Garcia

    public String getDireccion() { return direccion; } no usages  Edwing Garcia
    public void setDireccion(String direccion) { this.direccion = direccion; } no usages  Edwing Garcia

    public String getEstado() { return estado; } no usages  Edwing Garcia
    public void setEstado(String estado) { this.estado = estado; } no usages  Edwing Garcia
}
```

Externally added files can be added

## EnvioService

```
1 package com.evaluacionP1Edwing.evaluacion_practica_ecologistics.Servicio;
2 import com.evaluacionP1Edwing.evaluacion_practica_ecologistics.Modelo.Envio;
3 import org.springframework.stereotype.Service;
4 import java.util.*;
5 import java.util.concurrent.ConcurrentHashMap;
6
7 @Service 5 usages Edwing Garcia
8 public class EnvioService {
9     private final Map<String, Envio> store = new ConcurrentHashMap<>(); 4 usages
10
11     public List<Envio> findAll() { return new ArrayList<>(store.values()); } 1 usage Edwing Garcia
12     public Optional<Envio> findById(String id) { return Optional.ofNullable(store.get(id)); } 1 usage Edwing Garcia
13
14 @ public Envio save(Envio e) { 2 usages Edwing Garcia
15     if (e.getId() == null || e.getId().isBlank()) {
16         e.setId(UUID.randomUUID().toString());
17     }
18     store.put(e.getId(), e);
19     return e;
20 }
21
22 @ public void saveAll(Collection<Envio> envios) { 1 usage Edwing Garcia
23     for (Envio e : envios) save(e);
24 }
25
26 public int size() { return store.size(); } 1 usage Edwing Garcia
27 }
28
```

## EnvioLoader

```

package com.evaluacionP1Edwing.evaluacion_practica_ecologistics.Servicio;

import com.evaluacionP1Edwing.evaluacion_practica_ecologistics.Modelo.Envio;
import org.apache.commons.csv.CSVFormat;
import org.apache.commons.csv.CSVParser;
import org.apache.commons.csv.CSVRecord;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.ApplicationArguments;
import org.springframework.boot.ApplicationRunner;
import org.springframework.core.io.Resource;
import org.springframework.stereotype.Component;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.List;

```

@Component 1 usage Edwing Garcia

```
public class CsvEnvioLoader implements ApplicationRunner {
```

```
    private static final Logger log = LoggerFactory.getLogger(CsvEnvioLoader.class); 3 usages
```

```
    private final EnvioService servicio; 2 usages
```


```
    @Value("classpath:envios.csv") 2 usages
```

```
    private Resource csvResource;
```

```
    public CsvEnvioLoader(EnvioService servicio) { 2 usages Edwing Garcia
```

```
        this.servicio = servicio;
```

```
    }
```

 Externally added files

[View Files](#) [Always Ad](#)

@Override Edwing García

```
public void run(ApplicationArguments args) throws Exception {  
    if (!csvResource.exists()) {  
        log.warn("No se encontró envios.csv en resources. Se iniciará sin datos.");  
        return;  
    }  
  
    List<Envio> cargados = new ArrayList<>();  
    try (BufferedReader reader = new BufferedReader(  
        new InputStreamReader(csvResource.getInputStream(), StandardCharsets.UTF_8))) {  
  
        CSVParser parser = CSVFormat.DEFAULT.builder() Builder  
            .setHeader("id_envio", "cliente", "direccion", "estado")  
            .setSkipHeaderRecord(true)  
            .build() CSVFormat  
            .parse(reader);  
  
        for (CSVRecord r : parser) {  
            String id = r.get("id_envio").trim();  
            String cliente = r.get("cliente").trim();  
            String direccion = r.get("direccion").trim();  
            String estado = r.get("estado").trim();  
            cargados.add(new Envio(id, cliente, direccion, estado));  
        }  
    }  
    servicio.saveAll(cargados);  
    log.info("Archivo cargado con {} registros.", cargados.size());  
    log.info("Datos transformados a formato JSON (expuestos vía API).");  
}
```

Externally added files can  
[View Files](#) [Always Add](#)

## EnviosController

```
@RestController 1 usage  👤 Edwing Garcia
@RequestMapping("/envios")
@Tag(name = "Envios", description = "API para gestionar envíos")
public class EnvioController {


    private static final Logger log = LoggerFactory.getLogger(EnvioController.class); 2 u
    private final EnvioService servicio; 5 usages

    public EnvioController(EnvioService servicio) { this.servicio = servicio; } 3 usages

    @GetMapping  no usages  👤 Edwing Garcia
    @Operation(summary = "Listar envíos")
    public List<Envio> getAll() {
        log.info("GET /envios - total actuales: {}", servicio.size());
        return servicio.findAll();
    }

    @GetMapping("/{id}")  no usages  👤 Edwing Garcia
    @Operation(summary = "Obtener envío por ID")
    public ResponseEntity<Envio> getById(@PathVariable String id) {
        return servicio.findById(id).map(ResponseEntity::ok)
            .orElse(ResponseEntity.notFound().build());
    }

    @PostMapping  no usages  👤 Edwing Garcia
    @Operation(summary = "Crear/registrar un nuevo envío")
    public ResponseEntity<Envio> create(@RequestBody Envio nuevo) {
        Envio guardado = servicio.save(nuevo);
        log.info("POST /envios - creado {}", guardado.getId());
        return ResponseEntity.created(URI.create("/envios/" + guardado.getId()))
            .body(guardado);
    }
}
```

 External  
View F

## Swagger

Servers

http://localhost:8081 - Generated server url

Envios

API para gestionar envíos

GET

/envios

Listar envíos

POST

/envios

Crear/regarstrar un nuevo envío

GET

/envios/{id}

Obtener envío por ID

Schemas

GET

/envios

Listar envíos

Parameters

No parameters

Execute

Clear

Cancel

Responses

Curl

curl -X 'GET' \n'http://localhost:8081/envios' \n-H 'accept: \*/\*'

Request URL

http://localhost:8081/envios

Server response

Code

Details

200

Response body

[ \n {\n "cliente": "Juan Pérez",\n "direccion": "Calle 12 #45",\n "estado": "Entregado",\n "id": "001"\n },\n {\n "cliente": "María Gómez",\n "direccion": "Avenida 10 #33",\n "estado": "En tránsito",\n "id": "002"\n },\n {\n "cliente": "Luis Mora",\n "direccion": "Carrera 8 #22",\n "estado": "Pendiente",\n "id": "003"\n }\n]

Download

**POST** /envios Crear/regarstrar un nuevo envío

Parameters Cancel Reset

No parameters

Request body required application/json

```
{
  "cliente": "tralalero",
  "direccion": "tralala",
  "estado": "Entregado",
  "id": "006"
}
```

Execute Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8081/envios' \
  -H 'accept: */*' \
  -H 'content-type: application/json' \
  -d '{
    "cliente": "tralalero",
    "direccion": "tralala",
    "estado": "Entregado",
    "id": "006"
  }'
```

Request URL

`http://localhost:8081/envios`

**GET** /envios/{id} Obtener envío por ID

Parameters Cancel

Name	Description
id <small>required</small>	
string	006
(path)	

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8081/envios/006' \
  -H 'accept: */*'
```

Request URL

`http://localhost:8081/envios/006`

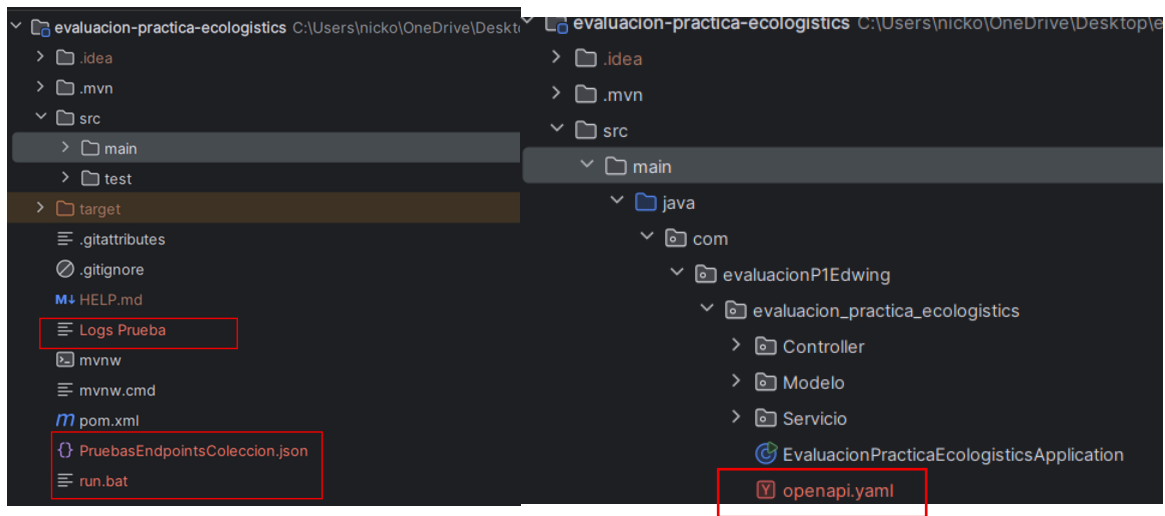
Server response

Code	Details
200	<p>Response body</p> <pre>{   "cliente": "tralalero",   "direccion": "tralala",   "estado": "Entregado",   "id": "006" }</pre> <p>Response headers</p> <pre>connection: keep-alive content-type: application/json date: Thu, 30 Oct 2025 01:50:35 GMT keep-alive: timeout=60 transfer-encoding: chunked</pre>

Download

Proyecto Principal con Tramas Logs, Colección Postman, Ejecutable Run, openApi.yaml





Pruebas Postman

Search collections

> New Collection

> PruebasEndpoints

GET ObtenerTodosEnvios

POST SubirEnvio

GET ObtenerEspecifico

PruebasEndpoints / ObtenerTodosEnvios

GET http://localhost:8081/envios

ParamsAuthorizationHeaders (8)BodyScriptsSettings

Query Params

	Key	Value	Description
	Key	Value	Description

BodyCookies (1)Headers (5)Test Results200 OK • 197 ms

{ } JSON

PreviewVisualize

```
1 [
2   {
3     "cliente": "Juan Pérez",
4     "direccion": "Calle 12 #45",
5     "estado": "Entregado",
6     "id": "001"
7   },
8   {
9     "cliente": "María Gómez",
10    "direccion": "Avenida 10 #33",
11    "estado": "En tránsito",
12    "id": "002"
13  },
14  {
15    "cliente": "Luis Mora",
16    "direccion": "Carrera 8 #22",
17    "estado": "Pendiente",
18    "id": "003"
```

Search collections

> New Collection

> PruebasEndpoints

GET ObtenerTodosEnvios

POST SubirEnvio

GET ObtenerEspecifico

PruebasEndpoints / SubirEnvio

POST http://localhost:8081/envios

ParamsAuthorizationHeaders (10)Body • ScriptsSettings

noneform-datax-www-form-urlencodedrawbinaryGraphQLJSON

SchemaBeautify

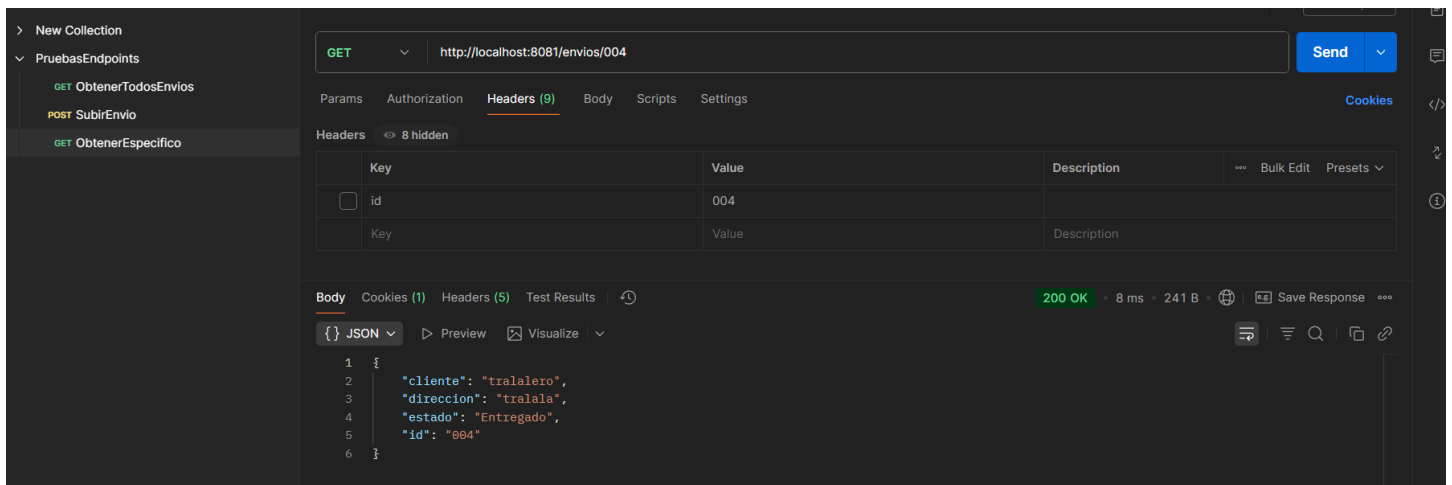
```
1 {
2   "cliente": "tralalero",
3   "direccion": "tralala",
4   "estado": "Entregado",
5   "id": "004"
6 }
```

BodyCookies (1)Headers (6)Test Results201 Created • 70 ms • 269 B • Save Response

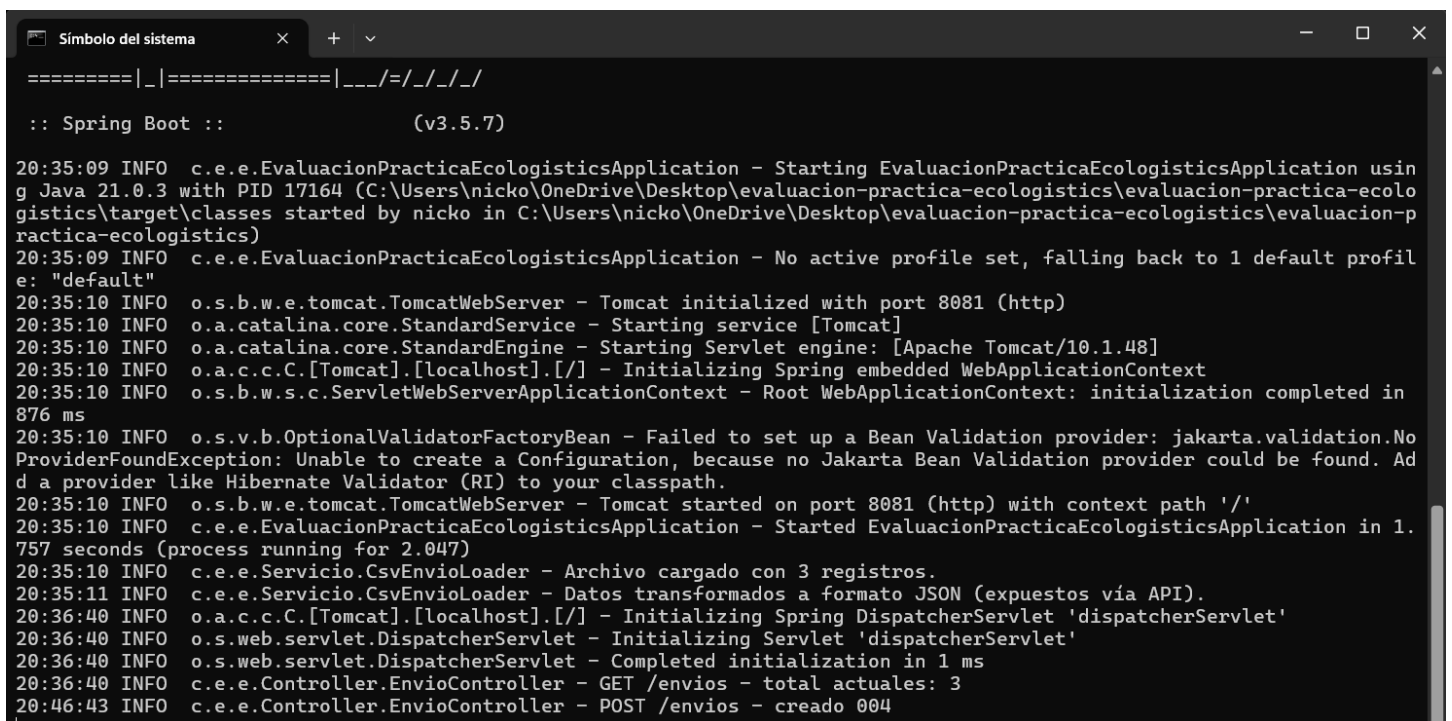
{ } JSON

PreviewVisualize

```
1 {
2   "cliente": "tralalero",
3   "direccion": "tralala",
4   "estado": "Entregado",
5   "id": "004"
6 }
```



## Logs



## Run.bat

```
C:\WINDOWS\system32\cmd. x + v
=====
Iniciando EcoLogistics Project
=====
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.evaluacionP1Edwing:evaluacion-practica-ecologistics >-----
[INFO] Building evaluacion-practica-ecologistics 0.0.1-SNAPSHOT
[INFO] from pom.xml
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- clean:3.4.1:clean (default-clean) @ evaluacion-practica-ecologistics ---
[INFO] Deleting C:\Users\nicko\OneDrive\Desktop\evaluacion-practica-ecologistics\evaluacion-practica-ecologistics\t
[INFO]
[INFO] >>> spring-boot:3.5.7:run (default-cli) > test-compile @ evaluacion-practica-ecologistics >>>
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ evaluacion-practica-ecologistics ---
[INFO] Copying 1 resource from src\main\resources to target\classes
[INFO] Copying 1 resource from src\main\resources to target\classes
[INFO]
[INFO] --- compiler:3.11.0:compile (default-compile) @ evaluacion-practica-ecologistics ---
[INFO] Changes detected - recompiling the module! :source
[INFO] Compiling 5 source files with javac [debug release 17] to target\classes
|
```

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

En mi solución apliqué principalmente el patrón de File Transfer, ya que los datos de los envíos se cargan desde un archivo CSV (envios.csv) al inicio de la aplicación. Posteriormente, se utilizó el patrón de Transform, convirtiendo las filas de dicho archivo a objetos y exponiéndolos en formato JSON. Finalmente, integré el patrón de API REST, implementando endpoints estándar para listar, obtener por ID y crear envíos, todo documentado con Swagger. De esta forma, se evidencia la transición desde un enfoque clásico basado en archivos hacia un modelo moderno basado en servicios RESTful.

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

La principal ventaja es el acceso en tiempo real a la información, eliminando la necesidad de esperar la transferencia periódica de archivos. Además, REST promueve la interoperabilidad al basarse en estándares ampliamente adoptados como HTTP y JSON, lo cual facilita la integración con diferentes plataformas y tecnologías. También permite un menor acoplamiento, ya que los consumidores interactúan únicamente con un contrato de servicio bien definido. Otra ventaja importante es la capacidad de manejar validaciones y errores de manera consistente, ofreciendo respuestas claras a cada solicitud y mejorando la confiabilidad del sistema.

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

Entre las limitaciones más relevantes se encuentra el hecho de que los datos actualmente se almacenan solo en memoria, lo cual implica que se pierden cada vez que se reinicia la aplicación. Además, trabajar con archivos CSV puede introducir problemas de calidad de datos, como errores de formato o inconsistencias. El uso de REST, al ser síncrono, también puede generar problemas de rendimiento si hay una gran cantidad de solicitudes simultáneas. Finalmente, cualquier cambio en el contrato de la API debe gestionarse cuidadosamente mediante versionado, ya que podría afectar a todos los consumidores que dependan de ella.

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

Sería recomendable utilizar un modelo de datos, que permita estandarizar la información compartida. Para garantizar un mejor desempeño, se podría complementar el uso de REST con eventos asíncronos usando RabbitMQ, permitiendo notificaciones y procesamiento paralelo sin sobrecargar la API. Además, sería fundamental contar con una base de datos transaccional junto con mecanismos de caché para optimizar las consultas.