Examen Práctico – Progreso 1

Caso empresarial: EcoLogistics S.A.

EcoLogistics S.A. es una empresa de transporte y distribución que busca modernizar su ecosistema tecnológico.

Actualmente, maneja la información de los **envíos** y **vehículos** de forma manual y descentralizada.

Cada día, los supervisores envían archivos CSV por correo electrónico con el listado de envíos realizados y los vehículos disponibles.

La Gerencia de TI ha solicitado construir un **prototipo funcional de integración**, que muestre cómo una solución moderna basada en **APIs RESTful** puede reemplazar el mecanismo tradicional de **envio de archivos por correo**, aplicando **patrones clásicos de integración**.

El equipo de arquitectura (ustedes) ha sido asignado para **prototipar la primera versión** de esta integración.

Objetivo general

Diseñar e implementar una **solución de integración funcional** para EcoLogistics que:

1. **Lea datos de un archivo local (patrón File Transfer)**.
2. **Transforme** los datos a formato JSON estandarizado.
3. **Exponga** una **API RESTful documentada con OpenAPI**, para consulta y registro de envíos.
4. **Demuestre** la transición desde un enfoque clásico hacia un modelo moderno de integración.

Competencias evaluadas

- Comprender la evolución histórica de la integración empresarial.
- Aplicar patrones clásicos (File Transfer, Transform, API Integration).

- Diseñar y exponer APIs RESTful interoperables.
- Documentar contratos de servicio usando OpenAPI.
- Analizar críticamente la diferencia entre modelos tradicionales y modernos.

Requerimientos técnicos

Opción A – Línea Base Oficial

- **Lenguaje:** Java 17
- **Herramienta de construcción:** Maven
- **Framework de integración:** Apache Camel
- **IDE recomendado:** Visual Studio Code
- **Testing:** Postman / Newman

Opción B – Stack Libre (alternativo)

Para la construcción de API podrán utilizar otro stack (Node.js, Python o .NET) **solo si cumplen las condiciones técnicas mínimas del contrato de entrega (ver sección siguiente)**.

Contrato de entrega uniforme

Con el fin de garantizar **evaluación justa, reproducibilidad y equivalencia funcional**, todos los proyectos (sin importar el lenguaje) deberán cumplir **el siguiente contrato técnico de entrega:**

Elemento	Requerimiento	Validación
1. Archivo fuente de datos	Archivo `envios.csv` con al menos 3 registros.	Estructura correcta: `id_envio,cliente,direccion,estado`
2. Transformación	Conversión de CSV → JSON estandarizado.	Verificación de estructura JSON `{id, cliente, direccion, estado}`
3. API REST	Debe exponer los endpoints: • `GET /envois` • `GET /envios/{id}` • `POST /envios`	Validación por Postman / Newman
4. Documentación OpenAPI	Contrato `openapi.yaml` o Swagger UI funcional.	Validación visual o por lint
5. Ejecución reproducible	Uno de los siguientes: • `Dockerfile` / `docker-compose.yml` • Script (`mvn clean package`, `npm start`, `uvicorn main:app`, etc.)	Revisión por consola o README
6. Pruebas	Colección Postman o Newman (`postman_collection.json`).	Validación automática o manual
7. Logs / Trazas	Mostrar logs en consola: carga, transformación y publicación.	Validación manual
8. Documentación técnica	Archivo `README.md` con instrucciones de instalación, ejecución y pruebas.	Revisión manual
9. Reflexión individual	Documento PDF con respuestas a las preguntas de reflexión.	Validación de comprensión conceptual
10. Estructura del proyecto	Entrega en carpeta o repositorio con esta estructura:`src/`, `envios.csv`, `openapi.yaml`, `postman_collection.json`, `README.md`, `reflexion.pdf`	Verificación directa

Especificaciones funcionales

1. Entrada: archivo CSV

```
id_envio,cliente,direccion,estado
001,Juan Pérez,Calle 12 #45,Entregado
002,María Gómez,Avenida 10 #33,En tránsito
003,Luis Mora,Carrera 8 #22,Pendiente
```

2. Proceso

- Leer el archivo local `envios.csv`.
- Transformar su contenido a formato JSON.
- Mantener los datos en memoria o estructura temporal (no requiere base de datos).

3. Salida: API REST

- `GET /envios`: lista todos los envíos.
- `GET /envios/{id}`: obtiene un envío específico.
- `POST /envios`: registra un nuevo envío.

4. Logs

Cada paso debe generar mensajes en consola (ejemplo):

```
[INFO] Archivo cargado con 3 registros.
[INFO] Datos transformados a formato JSON.
[INFO] API iniciada en puerto 8080.
```

Validaciones mínimas

1. El archivo CSV se procesa correctamente.
2. La API responde a los tres endpoints definidos.
3. Las respuestas son JSON válidos.
4. La documentación OpenAPI es visible o válida.
5. Se incluye una colección de pruebas funcionales.

6. Se evidencia el uso de patrones File Transfer y API REST.

Entregables

Link repositorio Git (público o con acceso al docente).

Formato de entrega:

```
/evaluacion-practica-ecologistics/
├── src/
├── envios.csv
├── openapi.yaml
├── postman_collection.json
├── README.md
└── reflexion.pdf (Documento con todas las evidencias de funcionamiento
del proyecto y respuestas a las preguntas de reflexión)
```

Condiciones para uso de stack alternativo

Los estudiantes que decidan usar un stack distinto al oficial (Java + Maven + Camel) deberán cumplir con **todas las condiciones del contrato técnico anterior** y además:

1. **Entregar un `Dockerfile` o instrucciones reproducibles exactas.**
   (Ejemplo: `npm install && npm start` o `uvicorn main:app`).
2. **Asegurar equivalencia funcional:**
   - Los endpoints deben tener los mismos nombres y comportamiento.
   - La estructura JSON debe ser idéntica a la del modelo de referencia.
3. **Incluir la documentación OpenAPI** en formato válido (`openapi.yaml` o generado por Swagger/FastAPI).
4. **Incluir una colección Postman o Newman funcional**, con resultados exitosos.
5. **Declarar el stack usado** en el `README.md` con versión del lenguaje, framework y comandos.

! Proyectos que no cumplan con estas condiciones serán penalizados.

Reflexión individual

Documento en PDF (máximo 1 página) que incluya todas las imágenes que el estudiante considere como evidencia a la implementación, pruebas y funcionamiento y además responda:

1. ¿Qué patrón de integración aplicaste y cómo se refleja en tu solución?
2. ¿Qué ventajas identificas al pasar de File Transfer a APIs REST?
3. ¿Qué riesgos o limitaciones encontraste en tu enfoque?
4. ¿Cómo escalarías esta integración si EcoLogistics tuviera 50 sistemas distintos?

Rúbrica de evaluación

Criterio	Descripción	Peso
Funcionamiento técnico	Cumple los requerimientos funcionales y ejecuta correctamente.	30%
Aplicación de patrones	Uso correcto de File Transfer, Transform y API REST.	25%
Documentación técnica (OpenAPI)	Contrato válido y bien descrito.	15%
Evidencias y pruebas (Postman/logs)	Incluye pruebas y trazas funcionales.	15%
Reflexión individual	Claridad conceptual y análisis crítico.	15%

Herramientas sugeridas

- Java + Maven + Apache Camel (línea base)
- Node.js + Express + Swagger UI
- Python + FastAPI
- .NET Minimal APIs + Swashbuckle

- Postman / Newman
- VS Code o IDE preferido