

Universidad de las Américas

Integración de sistemas

Examen Práctico Progreso 1

Docente: Darío Villamarín

Nombre: Doménica Arcos

Enlace Repositorio: <https://github.com/DoAle34411/Examen-P1-Integracion>

Índice

Reflexiones.....	3
Evidencias	4
Integración Camel + FastAPI.....	4
Código Camel.....	4
Código Python	5
Ejecución Integrada	5
Documentación y Swagger.....	7
Readme	7
SwaggerUI	8
Redoc	8

Reflexiones

En la presente práctica, se utilizó Apache Camel para la transferencia de archivos CSV por medio de un patrón File Transfer con una integración a un API realizado en Python con FastAPI. Este tipo de actividades, permiten reflexionar sobre la importancia de integrar sistemas, ya sea que se encuentren en estado deprecado o no. Durante la práctica, se encontró que la mejor manera de integrar estas dos tecnologías diferentes fue realizando una integración desde el código en Python, que observaría la carpeta de outputs que cuenta el programa de Java.

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

El patrón aplicado dentro de la integración fue File Transfer, evidenciado en el código de Java con Apache Maven; así como API Integration y Transform, ambos presentes en el código de Python.

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

En caso de que se migrara a un API REST completo, se evidenciarían mejores accesos a los datos en tiempo real, la eliminación de las dependencias de archivos planos y una mejor documentación.

Si se mantuviera el enfoque híbrido, permite que la consulta de estos datos no sea tan manual y pueda llevarse en otras implementaciones a ciertas bases de datos o entornos que permiten su análisis, no en tiempo real.

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

El enfoque tiene como riesgos que, al depender de dos tecnologías completamente diferentes, se debe tener mucho cuidado con los parches de seguridad y compatibilidad entre ambos sistemas; dependencia de la correcta escritura del CSV en el formato

establecido, con riesgos de bloqueos dentro de Apache Camel; actualmente solo se guardan los datos en memoria y una escalabilidad limitada al mantenerse en una sola carpeta.

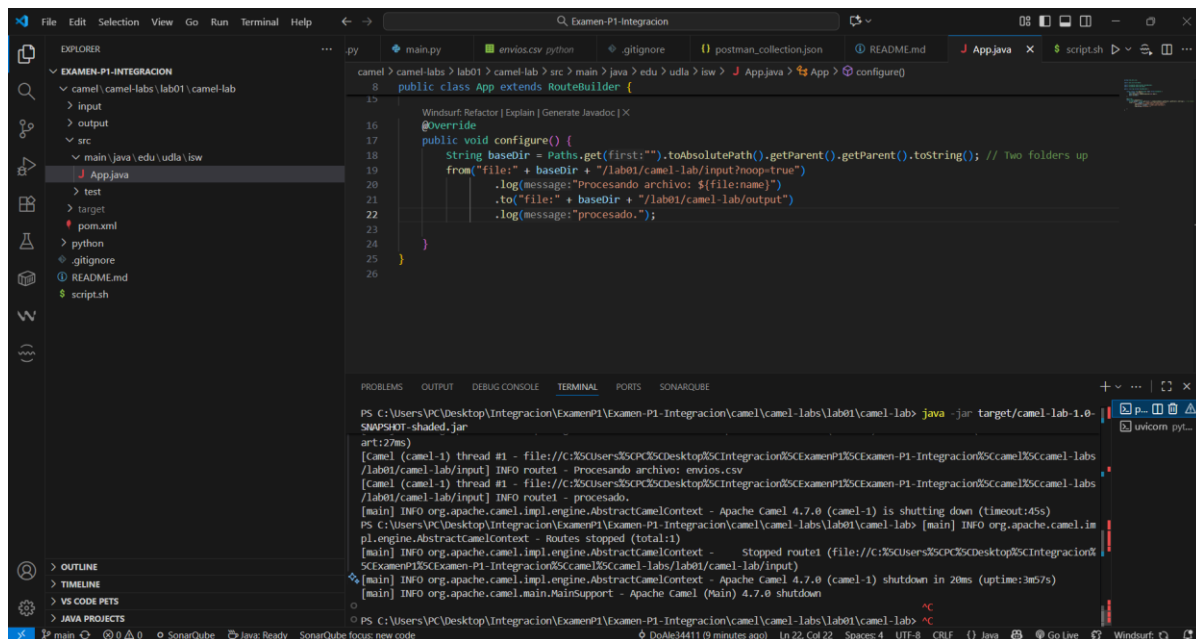
4. ¿Cómo escalarías esta integración si EcoLogistics tuviera 50 sistemas distintos?

Reemplazar la observación de la única carpeta con una arquitectura orientada a eventos, utilizar APIs validadas y mantener los datos en una base de datos.

Evidencias

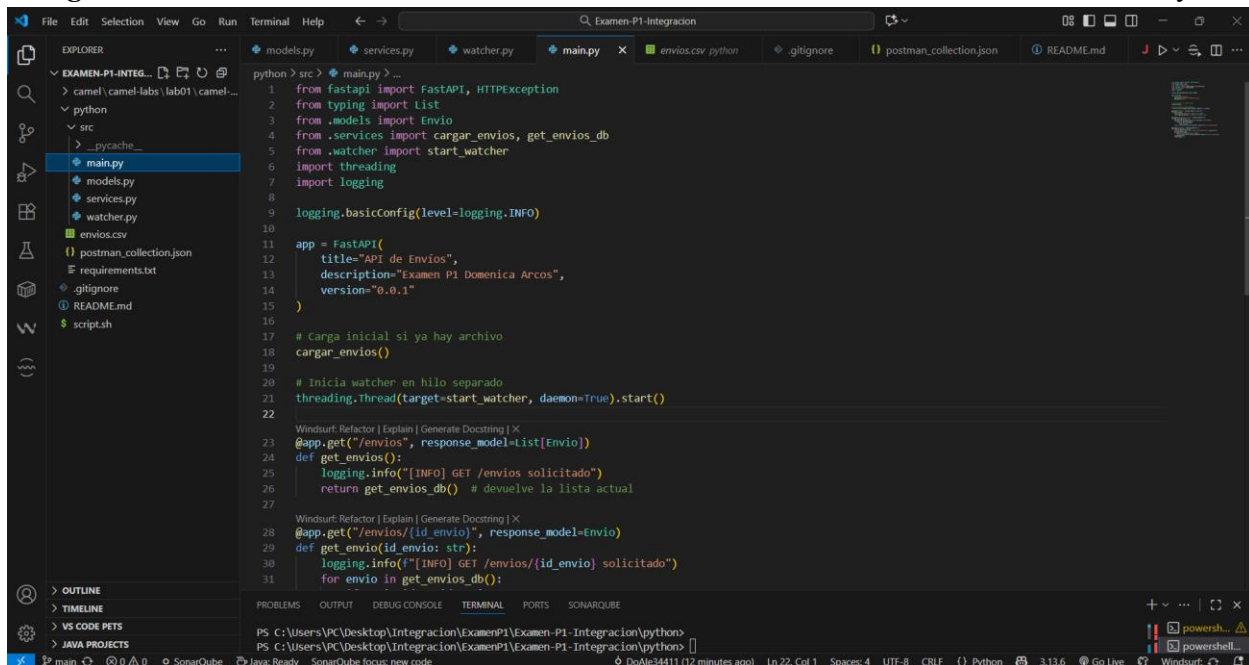
Integración Camel + FastAPI

Código Camel



The screenshot displays an IDE with the following components:

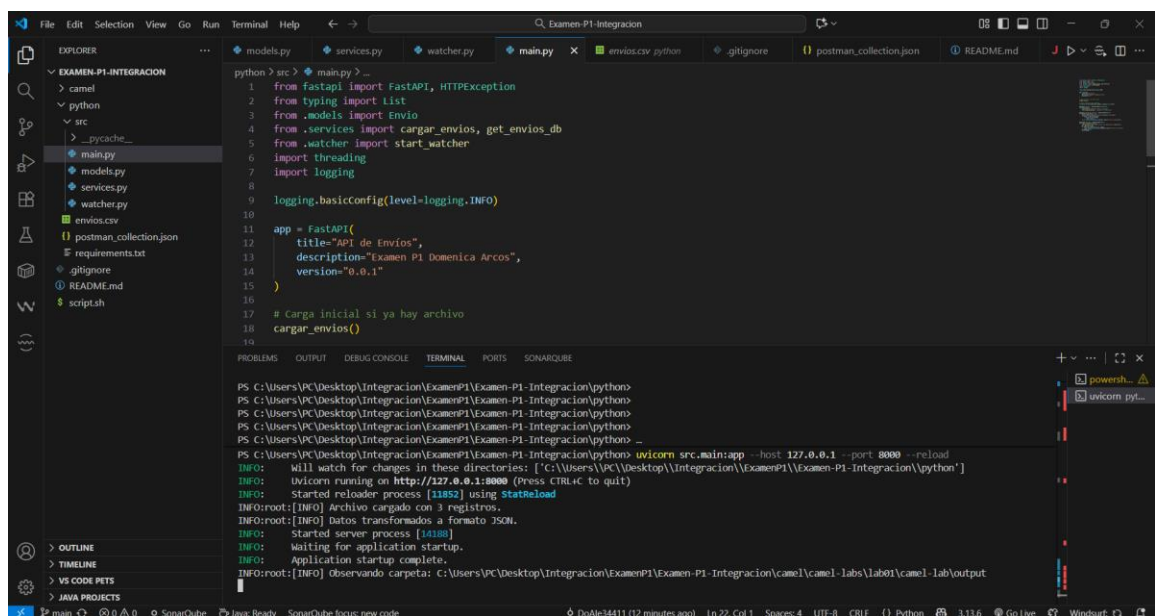
- EXPLORER:** Shows the project structure for 'EXAMEN-P1-INTEGRACION', including folders like 'camel-lab', 'input', 'output', 'src', and 'test', along with files like 'pom.xml', 'python', '.gitignore', 'README.md', and 'script.sh'.
- Code Editor:** Displays the 'App.java' file, which extends 'RouteBuilder'. The code defines a 'configure()' method that sets up a route to process CSV files from a specific input directory and log the results to an output directory.
- TERMINAL:** Shows the command 'java -jar target/camel-lab-1.0-SNAPSHOT-shaded.jar' being executed. The output logs the startup of Apache Camel 4.7.0, the configuration of the route, and the successful processing of the 'envios.csv' file, followed by the shutdown of the Camel context.



```
python > src > main.py > ...
1 from fastapi import FastAPI, HTTPException
2 from typing import List
3 from .models import Envio
4 from .services import cargar_envios, get_envios_db
5 from .watcher import start_watcher
6 import threading
7 import logging
8
9 logging.basicConfig(level=logging.INFO)
10
11 app = FastAPI(
12     title="API de Envios",
13     description="Examen P1 Domenica Arcos",
14     version="0.0.1"
15 )
16
17 # Carga inicial si ya hay archivo
18 cargar_envios()
19
20 # Inicia watcher en hilo separado
21 threading.Thread(target=start_watcher, daemon=True).start()
22
23 @app.get("/envios", response_model=List[Envio])
24 def get_envios():
25     logging.info("[INFO] GET /envios solicitado")
26     return get_envios_db() # devuelve la lista actual
27
28 @app.get("/envios/{id_envio}", response_model=Envio)
29 def get_envio(id_envio: str):
30     logging.info("[INFO] GET /envios/{id_envio} solicitado")
31     for envio in get_envios_db():
32         if envio.id == id_envio:
33             return envio
34     raise HTTPException(status_code=404, detail="Envio no encontrado")
35
36 if __name__ == "__main__":
37     uvicorn.run(app, host="0.0.0.0", port=8000)
```

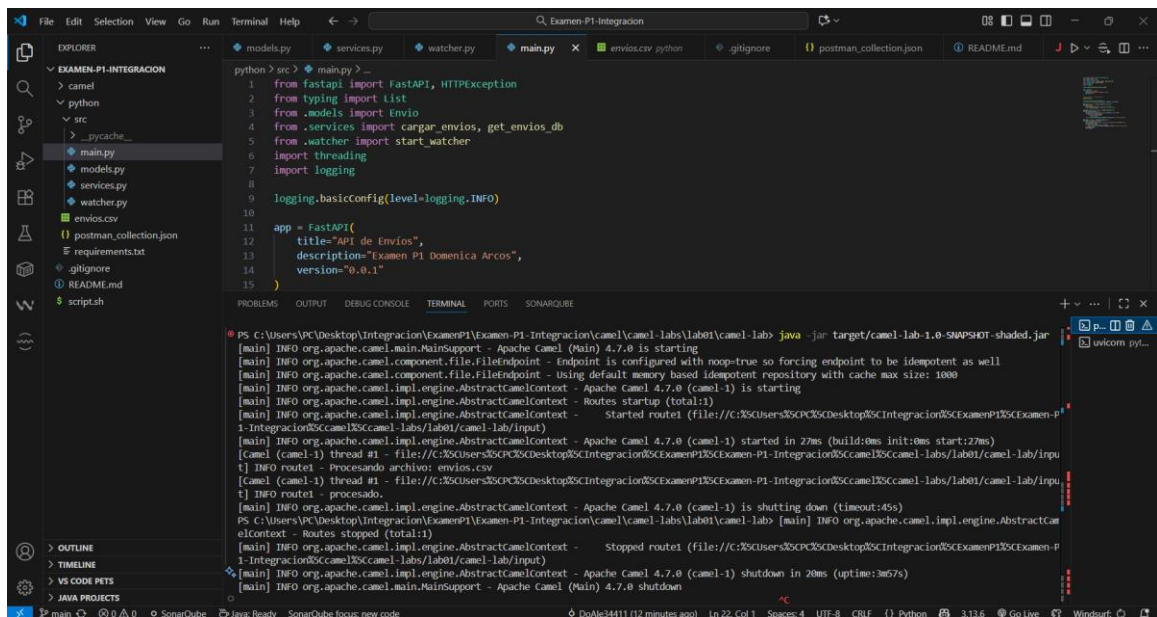
Ejecución Integrada

1. Ejecución Python (Obligatoria Primera)



```
PS C:\Users\PC\Desktop\Integracion\ExamenP1\Examen-P1-Integracion\python> python main.py
INFO: Will watch for changes in these directories: ['C:\Users\PC\Desktop\Integracion\ExamenP1\Examen-P1-Integracion\python']
INFO: Started reload process [11852] using StatReload
INFO:root:[INFO] Archivo cargado con 3 registros.
INFO:root:[INFO] Datos transformados a formato JSON.
INFO:root:[INFO] Started server process [14188]
INFO:root:[INFO] Waiting for application startup.
INFO:root:[INFO] Application startup complete.
INFO:root:[INFO] Observando carpeta: C:\Users\PC\Desktop\Integracion\ExamenP1\Examen-P1-Integracion\camel\camel-labs\lab01\camel-lab\output
```

2. Ejecución FTP en Apache Camel

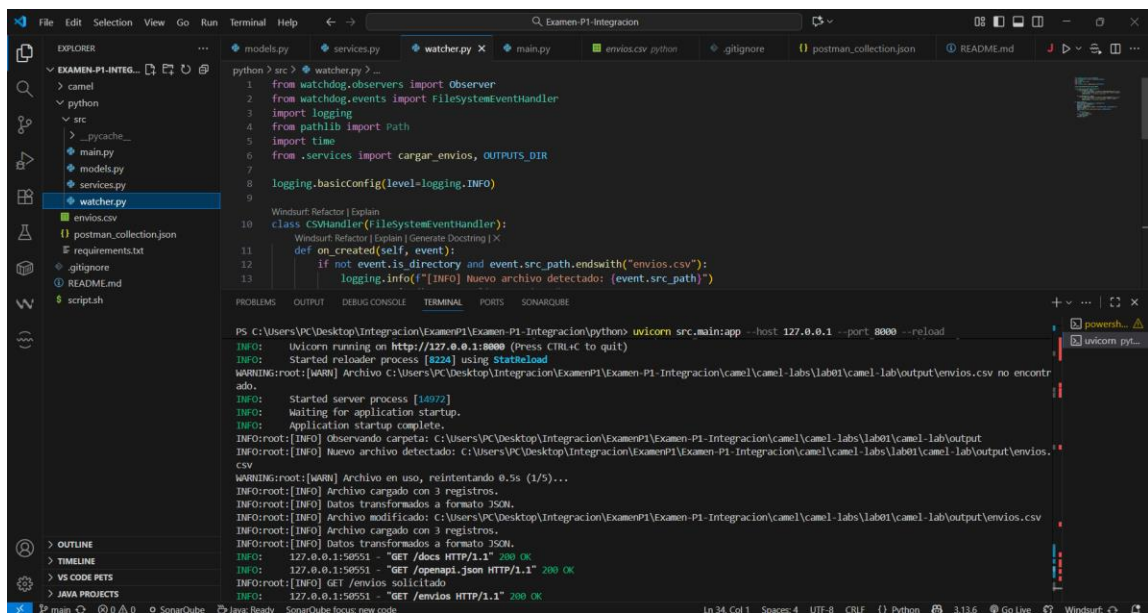


The screenshot shows the VS Code editor with the file explorer on the left displaying the project structure. The main.py file is open in the editor, showing a FastAPI application with a single endpoint. The terminal at the bottom shows the execution of the application using the command `java -jar target/camel-lab-1.0-SNAPSHOT-shaded.jar`. The output shows the application starting successfully and processing a file named `envios.csv`.

```
python > src > main.py > ...
1 from fastapi import FastAPI, HTTPException
2 from typing import List
3 from .models import Envio
4 from .services import cargar_envios, get_envios_db
5 from .watcher import start_watcher
6 import threading
7 import logging
8
9 logging.basicConfig(level=logging.INFO)
10
11 app = FastAPI(
12     title="API de Envios",
13     description="Examen P1 Domenica Arcos",
14     version="0.0.1"
15 )
```

```
[main] INFO org.apache.camel.main.MainSupport - Apache Camel (Main) 4.7.0 is starting
[main] INFO org.apache.camel.component.file.FileEndpoint - Endpoint is configured with noop=true so forcing endpoint to be idempotent as well
[main] INFO org.apache.camel.impl.engine.AbstractCamelContext - Apache Camel 4.7.0 (camel-1) is starting
[main] INFO org.apache.camel.impl.engine.AbstractCamelContext - Routes startup (total:1)
[main] INFO org.apache.camel.impl.engine.AbstractCamelContext - Started route1 (file:///C:/Users/PC/Desktop/Integracion/ExamenP1/Integracion/camel-lab/camel-lab/output/envios.csv)
[main] INFO org.apache.camel.impl.engine.AbstractCamelContext - Apache Camel 4.7.0 (camel-1) started in 27ms (build:0ms init:0ms start:27ms)
[main] INFO org.apache.camel.impl.engine.AbstractCamelContext - Stopped route1 (file:///C:/Users/PC/Desktop/Integracion/ExamenP1/Integracion/camel-lab/camel-lab/output/envios.csv)
[main] INFO org.apache.camel.impl.engine.AbstractCamelContext - Stopped route1 (file:///C:/Users/PC/Desktop/Integracion/ExamenP1/Integracion/camel-lab/camel-lab/output/envios.csv)
[main] INFO org.apache.camel.impl.engine.AbstractCamelContext - Apache Camel 4.7.0 (camel-1) is shutting down (timeout:45s)
[main] INFO org.apache.camel.impl.engine.AbstractCamelContext - Routes stopped (total:1)
[main] INFO org.apache.camel.impl.engine.AbstractCamelContext - Stopped route1 (file:///C:/Users/PC/Desktop/Integracion/ExamenP1/Integracion/camel-lab/camel-lab/output/envios.csv)
[main] INFO org.apache.camel.impl.engine.AbstractCamelContext - Apache Camel 4.7.0 (camel-1) shutdown in 20ms (uptime:3m67s)
[main] INFO org.apache.camel.main.MainSupport - Apache Camel (Main) 4.7.0 shutdown
```

3. Lectura dentro de Python del archivo subido gracias a watcher.py

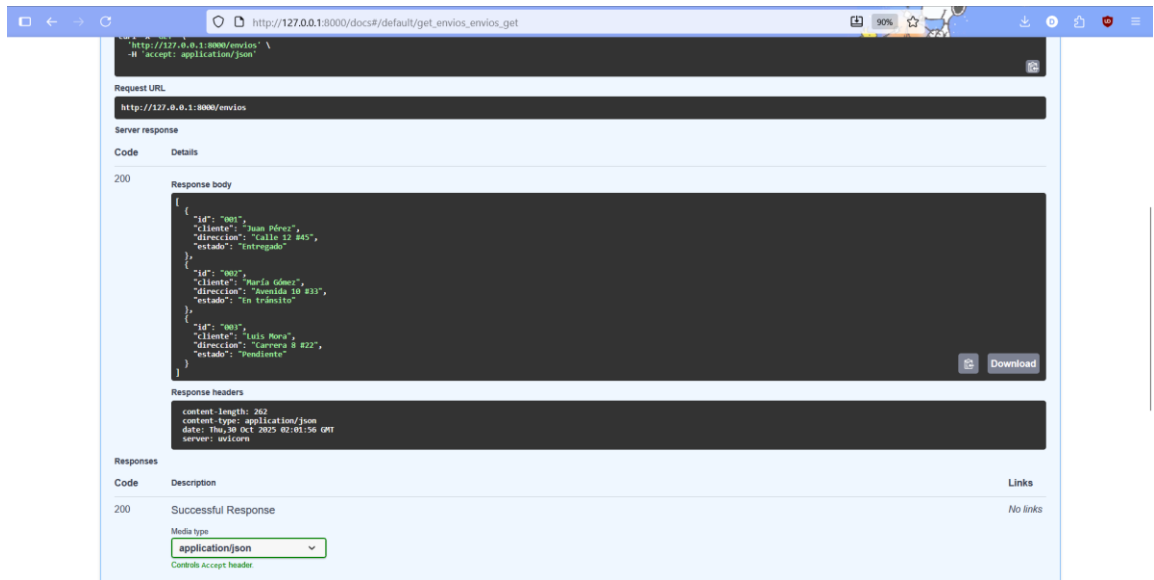


The screenshot shows the VS Code editor with the file explorer on the left displaying the project structure. The watcher.py file is open in the editor, showing a watchdog-based file watcher. The terminal at the bottom shows the execution of the application using the command `uvicorn src.main:app --host 127.0.0.1 --port 8000 --reload`. The output shows the application starting successfully and processing a file named `envios.csv`.

```
python > src > watcher.py > ...
1 from watchdog.observers import Observer
2 from watchdog.events import FileSystemEventHandler
3 import logging
4 from pathlib import Path
5 import time
6 from .services import cargar_envios, OUTPUTS_DIR
7
8 logging.basicConfig(level=logging.INFO)
9
10 class CSVHandler(FileSystemEventHandler):
11     def on_created(self, event):
12         if not event.is_directory and event.src_path.endswith("envios.csv"):
13             logging.info(f"[INFO] nuevo archivo detectado: {event.src_path}")
```

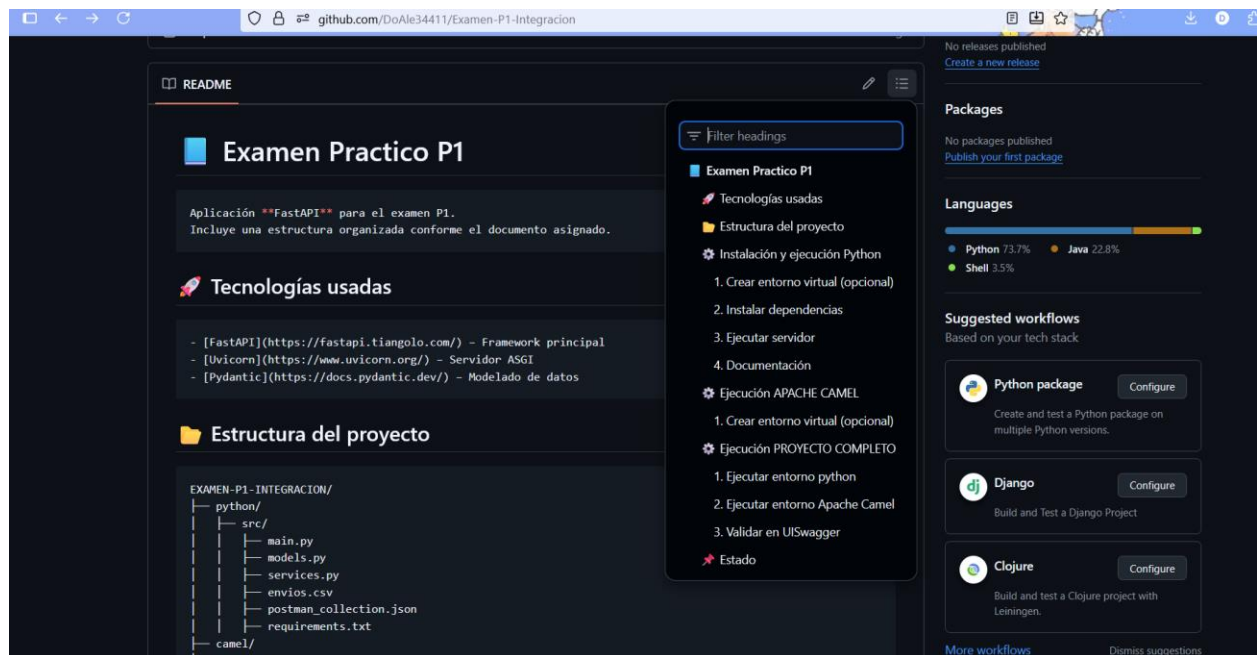
```
PS C:\Users\PC\Desktop\Integracion\ExamenP1\Integracion\python> uvicorn src.main:app --host 127.0.0.1 --port 8000 --reload
INFO: uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reload process [8224] using StarReload
WARNING:root:[WARN] Archivo C:\Users\PC\Desktop\Integracion\ExamenP1\Integracion\camel\camel-lab\lab\output\envios.csv no encontrado.
INFO:root:[INFO] Observando carpeta: C:\Users\PC\Desktop\Integracion\ExamenP1\Integracion\camel\camel-lab\lab\output
INFO:root:[INFO] Nuevo archivo detectado: C:\Users\PC\Desktop\Integracion\ExamenP1\Integracion\camel\camel-lab\lab\output\envios.csv
WARNING:root:[WARN] Archivo en uso, reintentando 0.5s (1/5)...
INFO:root:[INFO] Archivo cargado con 3 registros.
INFO:root:[INFO] Datos transformados a formato JSON.
INFO:root:[INFO] Archivo modificado: C:\Users\PC\Desktop\Integracion\ExamenP1\Integracion\camel\camel-lab\lab\output\envios.csv
INFO:root:[INFO] Archivo cargado con 3 registros.
INFO:root:[INFO] Datos transformados a formato JSON.
INFO: 127.0.0.1:50551 - "GET /docs HTTP/1.1" 200 OK
INFO: 127.0.0.1:50551 - "GET /openapi.json HTTP/1.1" 200 OK
INFO: 127.0.0.1:50551 - "GET /envios HTTP/1.1" 200 OK
```

4. Respuesta exitosa



Documentación y Swagger

Readme



SwaggerUI

API de Envíos 0.0.1 OAS 3.1

/openapi.json

Examen P1 Domenica Arcos

default

GET /envios Get Envios

Parameters

No parameters

Execute Clear

Responses

Curl

Request URL

Server response

Code Details

200

Response body

```
curl -X 'GET' \
  'http://127.0.0.1:8000/envios' \
  -H 'accept: application/json'
```

```
http://127.0.0.1:8000/envios
```

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

Redoc

API de Envíos (0.0.1)

Download OpenAPI specification Download

Examen P1 Domenica Arcos

Get Envios

Responses

> 200 Successful Response

Crear Envio

REQUEST BODY SCHEMA: application/json

required

id required string (id)

cliente required string (Cliente)

direccion required string (Direccion)

estado required string (Estado)

GET /envios

Response samples

200

Content type

application/json

Copy Expand all Collapse all

{
 "id": "string",
 "cliente": "string",
 "direccion": "string",
 "estado": "string"
}

POST /envios

Request samples

Payload

Content type

application/json

Copy

{
 "id": "string",
 "cliente": "string",
 "direccion": "string"
}