

Práctica API

Javier Rojo Llorens

App sobre local



App sobre el contenedor de Docker

	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
	autorizaciones-salud-ica	845d97cad4e4	autorizaciones-salud-ica	8000:8000	0.17%	38 seconds ago	

Modelos creados: (copiar archivo)

Servicio creado: (copiar archivo)

Router – init:

```
# TODO 1: Crear el router
router = APIRouter() # <-- Completa aquí

# TODO 2: Inicializar el logger
logger = LoggingService("FileInfoController")# <-- Completa aquí

# TODO 3: Inicializar el servicio FileInfoService pasándole el logger
file_service = FileInfoService(logger)# <-- Completa aquí
```

Router -decor:

```
# TODO 4: Crear el endpoint POST en la ruta "/get-info"
# Pistas:
# - Usa el decorador @router.post()
# - Define response_model=FileInfoResponse
# - Añade summary y description
@router.post(
    # <-- Completa el decorador
    "/get-info",
    response_model=FileInfoResponse,
    summary="Obtener información de archivo",
    description="Obtiene información básica de un archivo dado en Base64"
)
```

- Para el extra se añade el GET de formatos:

```
@router.get("/supported-formats")
async def supported_formats():
    formats = ["PDF", "JPEG", "PNG"]
    return {
        "formats": formats,
        "total": len(formats)
    }
```

Router – lógica:

```
async def get_file_info(request: FileInfoRequest):  
    """  
    Obtiene información básica de un archivo  
  
    - **file_base64**: Archivo codificado en Base64  
    - **filename**: Nombre del archivo (opcional)  
  
    Returns información básica del archivo  
    """  
    try:  
        # TODO 5: Llamar al método get_file_info del servicio  
        # Pista: file_service.get_file_info(...)  
        result = file_service.get_file_info(request.file_base64, request.filename)# <--  
  
        # TODO 6: Retornar FileInfoResponse con los datos del result  
        # Pista: return FileInfoResponse(**result)  
        return FileInfoResponse(**result)# <-- Completa aquí  
  
    except Exception as e:  
        # TODO 7: Lanzar HTTPException con status_code 400 y el mensaje de error  
        raise HTTPException(status_code=400, detail=str(e))# <-- Completa aquí
```

Integración en app.py:

```
# TODO 8: Importar el router file_info  
from app.routers import file_info  
  
# TODO 9: Registrar el router en la aplicación y añade el prefix="/v1/files"  
app.include_router(file_info.router, prefix="/v1/files")
```

Funciona correctamente:

The screenshot shows a REST client interface with the following details:

- Request body:** A JSON object with "file_base64" and "filename" fields.
- Execute button:** A blue button to run the request.
- Responses section:** Contains a curl command and the server's response.
- Request URL:** "http://localhost:8001/v1/files/get-info".
- Server response:** A 200 status code with a JSON response body.

```
curl -X 'POST' \  
  'http://localhost:8001/v1/files/get-info' \  
  -H 'accept: application/json' \  
  -H 'Content-Type: application/json' \  
  -d '{  
    "file_base64": "JVBERi0xLjQKJeljcz9W0yhuIG9iago8PC9ueXB1",  
    "filename": "test.pdf"  
  }'
```

Response body:

```
{  
  "message": "Archivo recibido correctamente",  
  "filename": "test.pdf",  
  "file_type": "pdf",  
  "file_size_kb": 0.03  
}
```

Opcional: **Opción A: Endpoint GET para formatos soportados**

GET

/v1/files/supported-formats

Supported Formats

Parameters

No parameters

Execute

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8001/v1/files/supported-formats' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:8001/v1/files/supported-formats
```

Server response

Code	Details
200	<div>Response body</div> <div><pre>{ "formats": ["PDF", "JPEG", "PNG"], "total": 3 }</pre></div> <div>Response headers</div> <div><pre>content-length: 42 content-type: application/json date: Sat,22 Nov 2025 12:31:48 GMT server: uvicorn</pre></div>