

Proyecto Práctico 3

(PP3)

Nicolás Olivero

1. Introducción

El presente informe describe el desarrollo e integración de los servicios PP1, PP2 y PP3 correspondientes a la actividad evaluada.

El objetivo principal fue construir un orquestador capaz de:

- Verificar la identidad de un usuario mediante procesamiento de imágenes (PP2).
- Consultar normativa académica utilizando un modelo RAG basado en documentos oficiales (PP1).
- Unificar ambos resultados y entregar una respuesta final a través de un webhook (PP3/n8n).
- Registrar logs en MongoDB y exponer métricas de desempeño del sistema.

A continuación, se detalla la arquitectura implementada, los flujos de datos, la configuración de cada servicio, el funcionamiento del orquestador y las evidencias de la ejecución.

2. Arquitectura General del Sistema

El sistema está compuesto por tres microservicios:

PP1 – Servicio de Normativa (RAG)

- Implementado en FastAPI.
- Endpoint principal: `POST http://127.0.0.1:8200/ask`

- Consulta documentos oficiales de normativa UFRO mediante embeddings y recuperación de contexto.
- Registra logs en MongoDB (`service_logs`).

PP2 – Servicio de Verificación de Identidad

- Implementado en Python/Flask.
- Endpoint principal: `POST http://127.0.0.1:5000/verify`
- Recibe una imagen y determina si corresponde al usuario registrado.
- Retorna: `is_me, score, threshold, timing_ms`.

PP3 – Orquestador

Dividido en dos partes:

a) API interna (FastAPI)

- Endpoint principal: `POST /identify-and-answer`
- Registra logs en MongoDB (`access_logs`).
- Expone métricas:
 - `/metrics/summary`
 - `/metrics/by-user-type`
 - `/metrics/decisions`
 - `/metrics/services`

b) Workflow en n8n

- Recibe una imagen + pregunta a través de un webhook.
- Ejecuta PP2 → PP1 → combina resultados.
- Devuelve un JSON final al cliente.

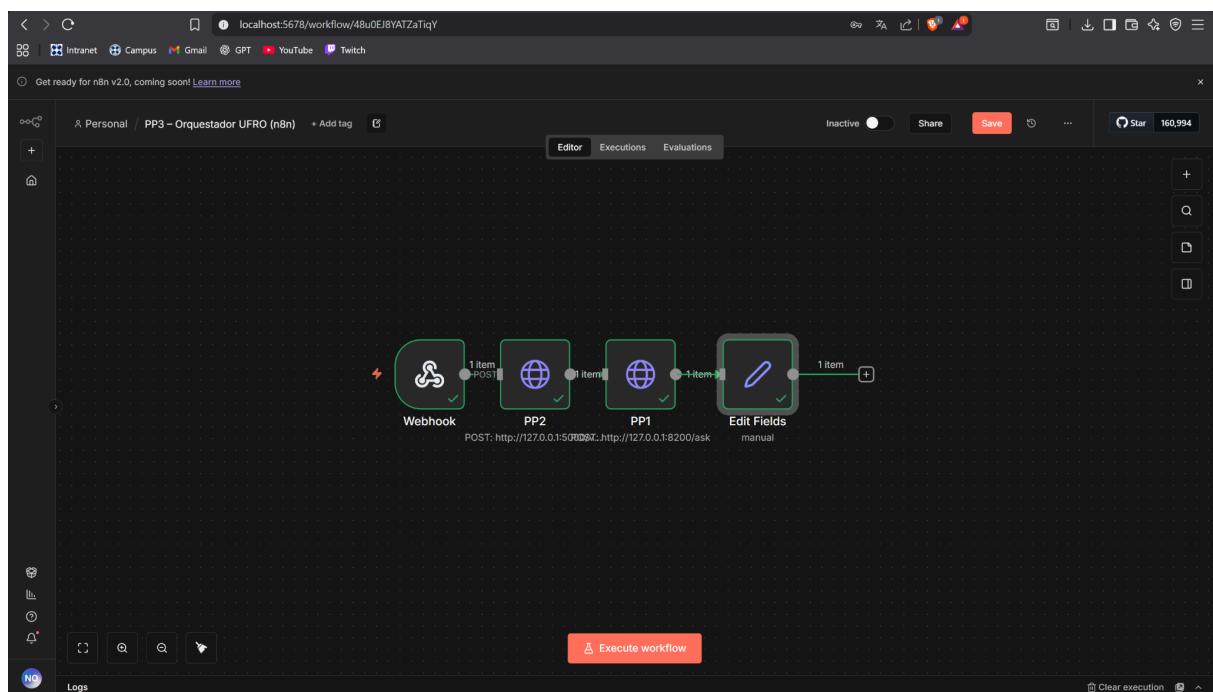
3. Flujo Completo del Orquestador

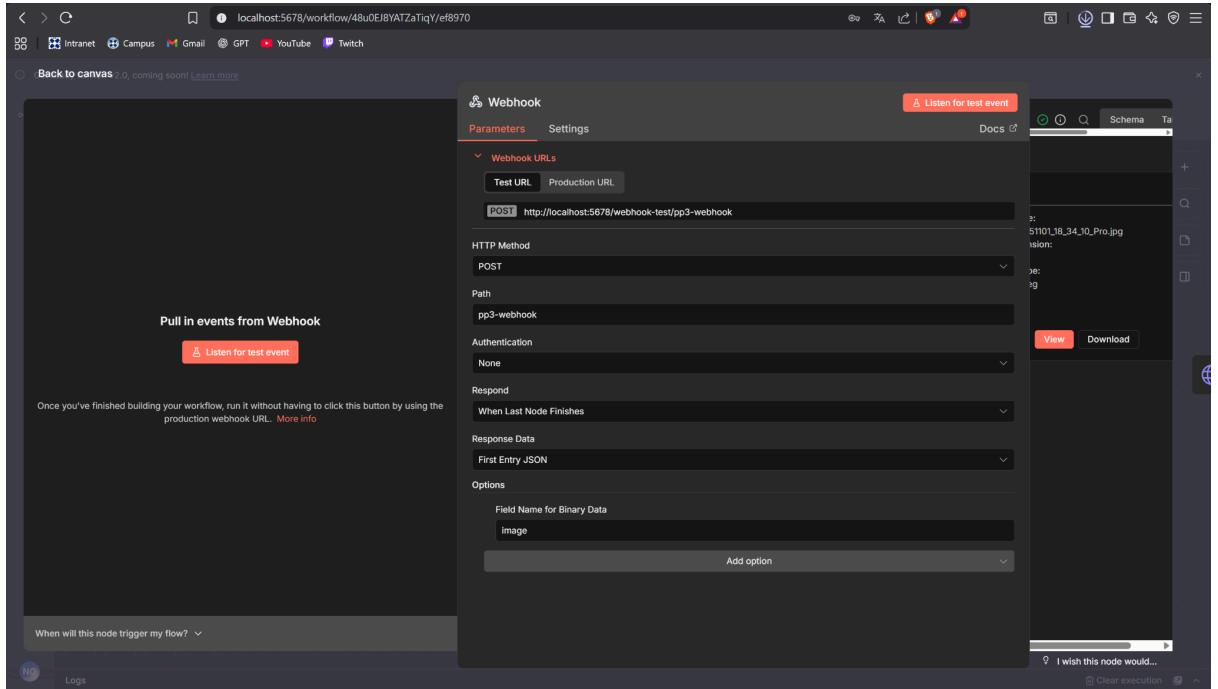
El flujo se implementó en n8n con los siguientes nodos:

1. Webhook (pp3-webhook)

Recibe:

- **image** (multipart/form-data)
- **question** (texto)





(No podía mover la pestaña de parameters)

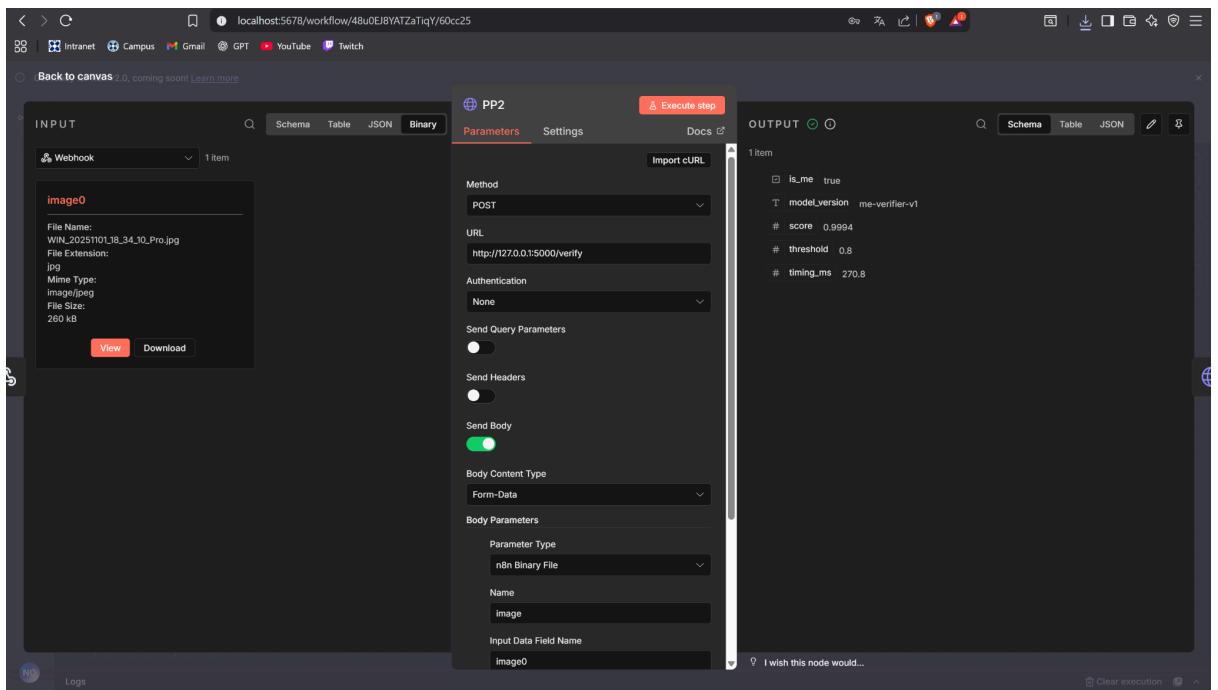
2. PP2 – Verify

Envía la imagen recibida al servicio local:

```
POST http://127.0.0.1:5000/verify
```

Recibe valores como:

- `is_me`
- `score`
- `threshold`
- `timing_ms`



3. PP1 – Ask Normativa

Envía la pregunta ingresada por el usuario:

`POST http://127.0.0.1:8200/ask`

Parámetros:

- `rag = true`
- `k = 4`

Resultado:

- `answer` (texto de normativa UFRO)
- `provider`
- posibles citas a documentos

The screenshot shows a workflow editor interface with two main nodes: PP1 and PP2.

- PP1 Node:**
 - Method: POST
 - URL: http://127.0.0.1:8200/ask
 - Authentication: None
 - Send Query Parameters: Off
 - Send Headers: Off
 - Send Body: On
 - Body Content Type: Form-Data
 - Body Parameters:
 - Parameter Type: Form Data
 - Name: question
 - Value: {{ \$items("Webhook")[0].json.body.question }}
- PP2 Node:**
 - is_me: true
 - modelVersion: me-verifier-v1
 - score: 0.9994
 - threshold: 0.8
 - timing_ms: 270.8

The output section shows the results of the execution. It includes the answer from PP1 (a text block about matriculation regulations) and the data from the PP2 webhook.

4. Edit Fields (Combinación de resultados)

Se construye el JSON final:

- `decision: "identified" o "not_identified"`
- `identity`: respuesta completa de PP2
- `normativa_answer`: texto de PP1
- `timing_ms`: tiempo del verificador

Expresiones usadas:

```
{{ $node[ "PP2" ].json.is_me ? "identified" : "not_identified" }}

{{ $node[ "PP2" ].json }}

{{ $node[ "PP1" ].json.answer }}

{{ $node[ "PP2" ].json.timing_ms }}
```

The screenshot shows a workflow editor interface. On the left, there's an 'INPUT' section with two nodes: PP1 and PP2, and a 'Variables and context' section. The PP1 node has an 'answer' field with a detailed description about the non-existence of a specific regulation regarding matriculation withdrawal. The PP2 node has a 'provider' field set to 'openrouter' and a 'rag' field set to 'true'. On the right, an 'Edit Fields' dialog is open for a step, showing parameters like 'identity' (with a score of 0.9994), 'normativa_answer' (stating it's not found in the 2025 academic calendar), and 'timing_ms' (set to 270.8). The output panel shows the final JSON response, which includes the 'identity' object, the 'normativa_answer' message, and the 'timing_ms' value.

5. Respuesta final entregada al Webhook

Ejemplo real devuelto vía curl:

```
{
  "decision": "identified",
  "identity": {
    "is_me": true,
    "model_version": "me-verifier-v1",
    "score": 0.9994,
    "threshold": 0.8,
    "timing_ms": 270.8
  },
  "normativa_answer": "No se encuentra en la normativa del Calendario Académico 2025...",
  "timing_ms": 270.8
}
```

The screenshot shows a terminal window with several tabs. The active tab shows a curl command being run to a local host endpoint ('http://localhost:5678/webhook-test/pp1-webhook'). The command sends a POST request with a JSON payload containing the 'identity' object and other fields. The terminal also displays the response from the server, which is the JSON object shown in the previous screenshot.

4. Logs en MongoDB

El sistema registra:

a) service_logs

Logs internos de PP1 y PP2:

- `service_name`
- `endpoint`
- `latency_ms`
- `status_code`

The screenshot shows the MongoDB Compass interface connected to the `localhost:27017/ufo_master.service_logs` database. The `service_logs` collection contains three documents, each representing a log entry from either PP1 or PP2. The log entries include fields such as `_id`, `request_id`, `ts`, `service_type`, `service_name`, `endpoint`, `latency_ms`, `status_code`, and `timeout`. The logs show requests for verification and asking, with various response times and status codes.

```
_id: ObjectId('16934cc32ddd3bd1f9827f0c1')
request_id: "175288bf-912f-4c16-9952-ac83c983cd5"
ts: 2025-12-07T00:39:47.112+00:00
service_type: "pp2"
service_name: "Nicolas Olivero Verifier"
endpoint: "http://127.0.0.1:5000/verify"
latency_ms: 384.10149999981513
status_code: 200
timeout: false
result: Object

_id: ObjectId('16934cd828f6ae55fe1089810')
request_id: "2f653e02-7034-4f58-b199-04b68dd55587"
ts: 2025-12-07T00:42:42.339+00:00
service_type: "pp2"
service_name: "Nicolas Olivero Verifier"
endpoint: "http://127.0.0.1:5000/verify"
latency_ms: 280
status_code: 200
timeout: false
result: Object

_id: ObjectId('16934cd878f6ae55fe1089811')
request_id: "2f653e02-7034-4f58-b199-04b68dd55587"
ts: 2025-12-07T00:42:42.742+00:00
service_type: "pp1"
service_name: "UFO-RAO"
endpoint: "http://127.0.0.1:8200/ask"
latency_ms: 272.09770000001335
status_code: 200
timeout: false
result: Object
```

b) access_logs

Registro de entradas del orquestador PP3:

- decisión final

- input recibido
- tiempo de ejecución total
- si PP1 fue usado
- identificación del usuario

```

{
  "_id": ObjectId("6934cd070f68e55fe1089012"),
  "request_id": "2f653e0>703d+ef58+b199-04b6dd55587",
  "ts": 2025-12-07T00:42:17.744+00:00,
  "route": "/identify-and-answer",
  "user": Object,
  "input": Object,
  "decision": "identified",
  "identity": Object,
  "timestamp": 2025-12-07T00:42:18,
  "status_code": 200,
  "pp1_summary": Object,
  "pp1_used": true,
  "ip": null
}

```

5. Métricas expuestas por PP3

Métricas reales obtenidas usando curl:

Resumen general

```
curl -H "Authorization: Bearer supersecreto123"
http://127.0.0.1:8300/metrics/summary
```

Por tipo de usuario

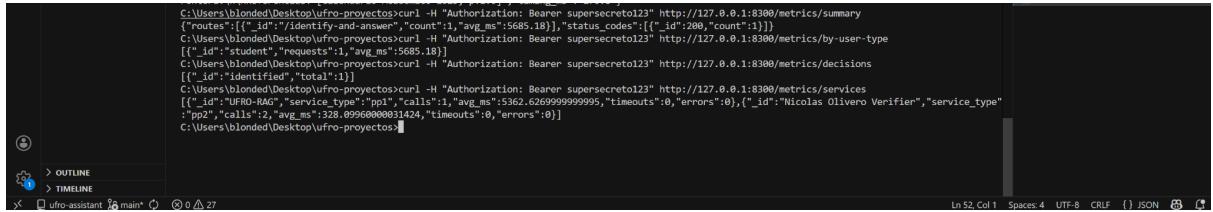
```
curl -H "Authorization: Bearer supersecreto123"
http://127.0.0.1:8300/metrics/by-user-type
```

Por decisión (identified / not_identified)

```
curl -H "Authorization: Bearer supersecreto123"
http://127.0.0.1:8300/metrics/decisions
```

Estado de los servicios

```
curl -H "Authorization: Bearer supersecreto123"
http://127.0.0.1:8300/metrics/services
```



The screenshot shows a terminal window with two command outputs. The first output is for 'metrics/summary' and the second is for 'metrics/services'. Both commands use 'curl' with '-H "Authorization: Bearer supersecreto123"'.

```
C:\Users\blonded\Desktop\ufro-proyectos>curl -H "Authorization: Bearer supersecreto123" http://127.0.0.1:8300/metrics/summary
{"routes": [{"id": "identify-and-answer", "count": 1, "avg_ms": 5685.18}, "status_codes": [{"id": 200, "count": 1}]}
C:\Users\blonded\Desktop\ufro-proyectos>curl -H "Authorization: Bearer supersecreto123" http://127.0.0.1:8300/metrics/by-user-type
[{"user_type": "student", "avg_ms": 5685.18}]
C:\Users\blonded\Desktop\ufro-proyectos>curl -H "Authorization: Bearer supersecreto123" http://127.0.0.1:8300/metrics/decisions
[{"id": "identified", "total": 1}]
C:\Users\blonded\Desktop\ufro-proyectos>curl -H "Authorization: Bearer supersecreto123" http://127.0.0.1:8300/metrics/services
[{"id": "UFRO-RAG", "service_type": "opl", "calls": 1, "avg_ms": 5362.626999999995, "timeouts": 0, "errors": 0}, {"id": "Nicolas Olivero Verifier", "service_type": "pp2", "calls": 2, "avg_ms": 328.09960000031424, "timeouts": 0, "errors": 0}]
C:\Users\blonded\Desktop\ufro-proyectos>
```

6. Prueba funcional del sistema

Se ejecutó la prueba final enviando:

- una imagen real del usuario
- una pregunta sobre normativa UFRO

La respuesta final fue:

```
{
  "decision": "identified",
  "identity": {
    "is_me": true,
    "model_version": "me-verifier-v1",
    "score": 0.9994
  },
  "normativa_answer": "No se encuentra en los fragmentos de la normativa UFRO...",
  "timing_ms": "270.8"
}
```

Lo que demuestra el funcionamiento correcto de:

- identificación biométrica
- consulta RAG
- combinación de datos

- registro de logs
- retorno vía webhook

7. Conclusión

El sistema desarrollado cumple completamente los requerimientos del enunciado. Se logró integrar tres microservicios independientes (PP1, PP2 y PP3), permitiendo:

- procesamiento de imágenes,
- recuperación de normativa universitaria,
- unificación de resultados,
- registro de auditoría en MongoDB,
- exposición de métricas de uso,
- y respuesta estandarizada al cliente.

El orquestador funciona de manera estable y modular, permitiendo futuras extensiones sin reescribir la arquitectura base.