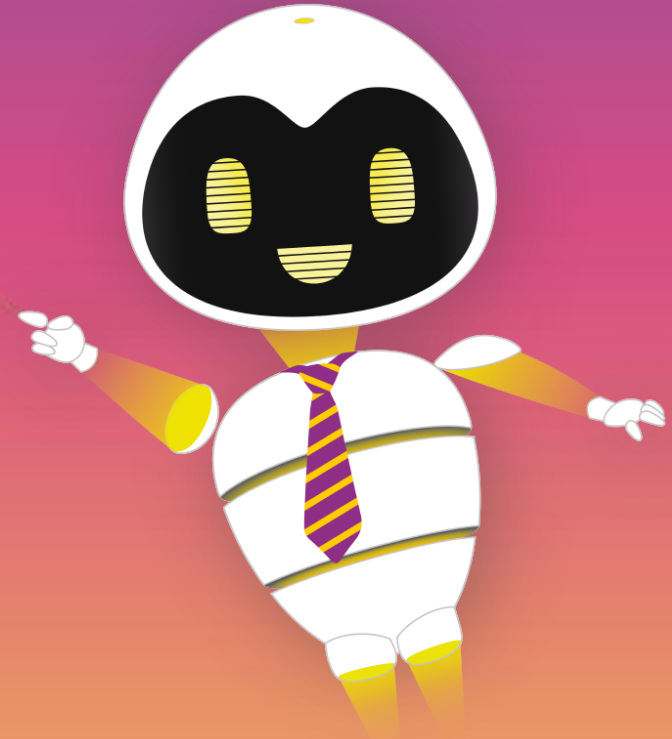




PI DATA STRATEGY & CONSULTING

get Talent

AI EDITION

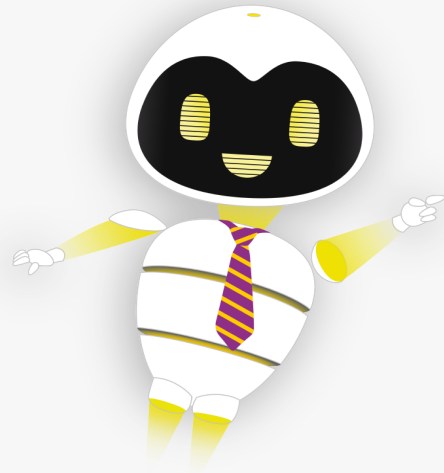


Arquitectura RAG (repaso)

APIs



¿Qué es una API?



PI DATA STRATEGY & CONSULTING



Application Programming Interface (API)

- Interfaz para que distintos sistemas se comuniquen.
- "Contrato" entre proveedor y consumidor del servicio

Beneficios principales

- Abstracción: Oculta complejidad interna
- Reutilización y escalabilidad



Paradigmas y estilos arquitectónicos

Representational State Transfer (REST)

- Recursos identificados por URIs
- Métodos HTTP estandarizados
- Comunicación stateless
- Cacheabilidad
- Uso (muy) amplio y sencillo

Ventajas:

- Simplicidad
- Ampla adopción
- Caché integrado con HTTP

Desventajas

- Over-Fetching (o Under-Fetching)
- Múltiples llamadas
- Evolución compleja

Métodos HTTP comunes en REST

- **GET:** Obtiene una representación de un recurso, sin modificarlo.
- **PATCH:** Actualiza parcialmente un recurso existente (aplica cambios puntuales).
- **PUT:** Actualiza un recurso existente de forma completa (reemplaza el recurso actual por el nuevo contenido).
- **POST:** Crea un recurso nuevo o envía datos para ser procesados por el
- **DELETE:** Elimina un recurso especificado.

Paradigmas y estilos arquitectónicos

GraphQL

- Permite definir forma y cantidad de datos exactos requeridos
- Esquema y lenguaje de consulta propio
- Uso amplio en múltiples tipos de clientes (móvil, web, IoT).

Ventajas:

- Consulta específica de datos
- Un solo endpoint
- Evolución sin rupturas
- Tipado fuerte

Desventajas

- Curva de aprendizaje
- Complejidad en caché HTTP
- Coste servidor
- Ecosistema joven

APIs en la nube e IA

- La nube permite exponer servicios complejos a través de APIs.
- Proveedores como Azure Cognitive Services ofrecen APIs para realizar traducciones, análisis de sentimientos, reconocimiento de entidades en texto, generación de embeddings, etc.
- Permite generar arquitecturas como RAG aprovechando de tales servicios, utilizando de embeddings y vector stores para recuperar información relevante (antes de generar una respuesta con un LLM)
 - Ejemplo:
 - API de embeddings (Azure OpenAI Service)
 - API de vector store (servicio que indexa y recupera información según similitud semántica - Azure AI Search)
 - API de LLM (Azure OpenAI Service)

Buenas prácticas

- No solo debemos considerar el diseño (consistencia, versionado, etc), sino también en cómo documentar su funcionamiento y en cómo mantenerlas arriba.

Estructuración y Diseño:

- Una buena estructura en una API facilita su uso, mantenimiento y escalabilidad, asegurando claridad, consistencia y facilidad para agregar nuevas funciones sin errores.

Documentación:

- Es posible utilizar OpenAPI/Swagger para una comunicación clara.

Observabilidad y monitoreo:

- Métricas y registros (logs) ayudan a entender el comportamiento de la API en producción, detectar bottlenecks y resolver problemas.

Cumplimiento normativo (y ético):

- La transferencia de datos a través de APIs deberían cumplir normativas de privacidad y seguridad. En el contexto de IA, se debería velar por no inducir sesgos y buscar proveer lineamientos éticos de la organización.

Documentación (OpenAPI)

- Se puede generar documentación manualmente o a través de algunos frameworks.
- Una forma de editar documentación es a través de ciertas webapps: <https://editor.swagger.io/>

The screenshot displays the Swagger Editor interface. On the left, a code editor shows the OpenAPI 3.0 definition for the Swagger Petstore. The definition includes a title, description, contact information, license, and a list of endpoints. The endpoints are categorized by HTTP method: PUT for updating a pet, POST for adding a new pet, and GET for finding pets by status or tags. The right panel shows the rendered documentation, including a title, description, terms of service, and a list of endpoints with their respective HTTP methods and descriptions.

```
1 openapi: 3.0.3
2 info:
3   title: Swagger Petstore - OpenAPI 3.0
4   description: |-
5     This is a sample Pet Store Server based on the OpenAPI 3.0
6     specification. You can find out more about
7     Swagger at [https://swagger.io](https://swagger.io). In the
8     third iteration of the pet store, we've switched to the
9     design first approach!
10    You can now help us improve the API whether it's by making
11    changes to the definition itself or to the code.
12    That way, with time, we can improve the API in general, and
13    expose some of the new features in OAS3.
14
15    _If you're looking for the Swagger 2.0/OAS 2.0 version of
16    petstore, then click [here](https://editor.swagger.io?url=
17    https://petstore.swagger.io/v2/swagger.yaml). Alternatively
18    , you can load via the 'Edit > Load Petstore OAS 2.0' menu
19    option!
20
21    Some useful links:
22    - [The Pet Store repository](https://github.com/swagger-api
23    /swagger-petstore)
24    - [The source API definition for the Pet Store](https://github
25    .com/swagger-api/swagger-petstore/blob/master/src/main
26    /resources/openapi.yaml)
27
28    termsOfService: http://swagger.io/terms/
29    contact:
30      email: apiteam@swagger.io
31    license:
32      name: Apache 2.0
33      url: http://www.apache.org/licenses/LICENSE-2.0.html
34    version: 1.0.11
35
36  externalDocs:
37    description: Find out more about Swagger
38    url: http://swagger.io
39
40  servers:
41    - url: https://petstore3.swagger.io/api/v3
42
43  tags:
44    - name: pet
45      description: Everything about your Pets
46      externalDocs:
47        description: Find out more
48        url: http://swagger.io
49    - name: store
50      description: Access to Petstore orders
51      externalDocs:
52        description: Find out more about our store
53        url: http://swagger.io
54    - name: user
55      description: Operations about user
56
57  paths:
58    /pet:
59      put:
60        tags:
61          - pet
62        summary: Update an existing pet
63        description: Update an existing pet by Id
64        operationId: updatePet
65        requestBody:
```

Swagger Petstore - OpenAPI 3.0 ^{1.0.11} ^{OAS 3.0}

This is a sample Pet Store Server based on the OpenAPI 3.0 specification. You can find out more about Swagger at <https://swagger.io>. In the third iteration of the pet store, we've switched to the design first approach! You can now help us improve the API whether it's by making changes to the definition itself or to the code. That way, with time, we can improve the API in general, and expose some of the new features in OAS3.

If you're looking for the Swagger 2.0/OAS 2.0 version of Petstore, then click [here](#). Alternatively, you can load via the [Edit > Load Petstore OAS 2.0](#) menu option!

Some useful links:

- [The Pet Store repository](#)
- [The source API definition for the Pet Store](#)

Terms of service
Contact the developer
Apache 2.0
Find out more about Swagger

Servers
https://petstore3.swagger.io/api/v3 Authorize

pet Everything about your Pets Find out more

PUT	/pet	Update an existing pet	Find out more
POST	/pet	Add a new pet to the store	Find out more
GET	/pet/findByStatus	Finds Pets by status	Find out more
GET	/pet/findByTags	Finds Pets by tags	Find out more
GET	/pet/{petId}	Find pet by ID	Find out more
POST	/pet/{petId}	Updates a pet in the store with form data	Find out more

Conclusión acerca de APIs

- Manifestación pragmática del principio de abstracción en ingeniería de software.
- Han evolucionado desde simples interfaces hacia mecanismos complejos que habilitan arquitectura de microservicios, integraciones en la nube y acceso a sistemas (potentes) de IA.
- Diseño guiado por principio y patrones que garantizan coherencia, confiabilidad y facilidad de consumo.
- Son la base principal para la comunicación entre diferentes entornos y plataformas.

Frameworks

- Conjunto de herramientas y bibliotecas que simplifican y estructuran el desarrollo de aplicaciones web y servicios de API RESTful.
- Proporcionan funcionalidades preconstruidas para manejar rutas, solicitudes HTTP, autenticación, validación de datos y mucho más, permitiendo a los desarrolladores centrarse en la lógica de negocio sin preocuparse demasiado por los detalles técnicos subyacentes.



Flask

Ejemplos de APIs (gratuitas)

- <https://pokeapi.co/>
- <https://v2.jokeapi.dev/>
- <https://public-apis.io/> (repositorio de APIs públicas)
- <https://opentdb.com/>

Ejemplos

Request

```
curl -k "https://v2.jokeapi.dev/joke/Any?type=single"
```

Response

```
{
  "error": false,
  "category": "Programming",
  "type": "single",
  "joke": "A guy walks into a bar and asks for 1.4 root beers.\nThe bartender says \"I'll have to charge you extra, that's a root beer float\".\nThe guy says \"In that case, better make it a double.\"\"",
  "flags": {
    "nsfw": false,
    "religious": false,
    "political": false,
    "racist": false,
    "sexist": false,
    "explicit": false
  },
  "id": 2,
  "safe": true,
  "lang": "en"
}
```



Ejemplos

Request (python)

```
import requests

# API URL
url = "https://v2.jokeapi.dev/joke/Any?type=single"

try:
    # Make the GET request
    response = requests.get(url)

    # Check if the response is successful
    if response.status_code == 200:
        joke = response.json()
        print("Joke:", joke.get("joke"))
    else:
        print(f"Error: {response.status_code} - {response.text}")

except requests.exceptions.RequestException as e:
    print(f"An error occurred: {e}")
```



Ejemplo FastAPI

- Crear entorno virtual
- Instalar FastAPI junto con su servidor:

```
pip install fastapi uvicorn
```

- Crear un archivo main.py que contendrá el código de la aplicación:

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def hello_world():
    return {"message": "Hola, Mundo"}
```

- Ejecutar el servidor:

```
uvicorn main:app --reload
```



Bibliografia

APIs:

- <https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design>
- <https://aws.amazon.com/compare/the-difference-between-graphql-and-rest/>
- <https://learn.microsoft.com/en-us/azure/api-management/>
- <https://learn.microsoft.com/en-us/azure/ai-services/>
- <https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design>
- <https://fastapi.tiangolo.com/>