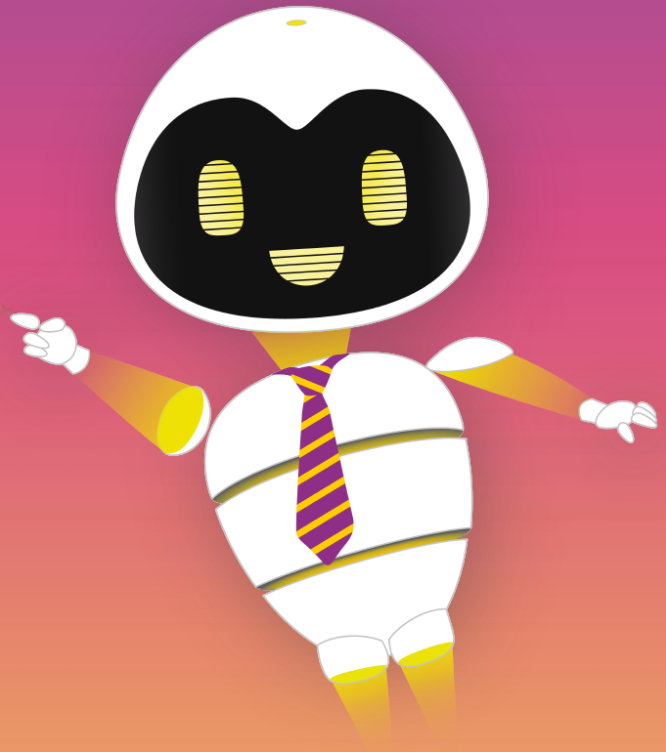
PI DATA STRATEGY & CONSULTING

get Talent

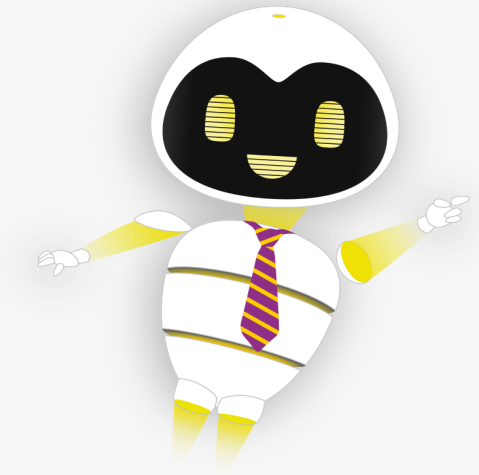
AI EDITION



Arquitectura RAG (repaso)

APIs

¿Qué es una API?



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
- Amplia adopción
- Caché integrado con HTTP

Desventajas

- Over-Fetching (o Under-Fetching)
- Multiples 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 jóven

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.

Estructuracion 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:

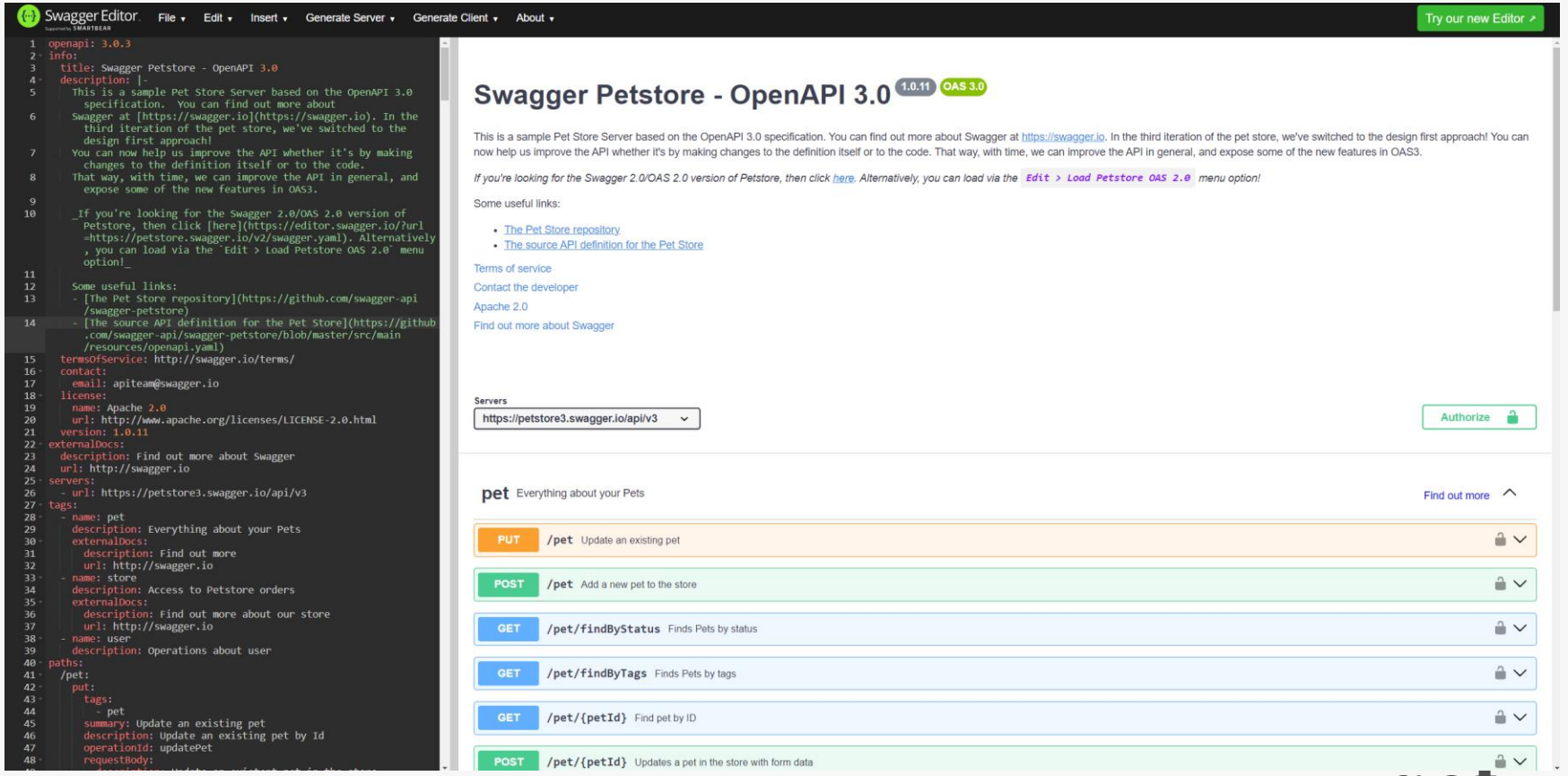
- Métricas y registros (logs) ayudan a entender el comportamiento de la API en producción, detectar bottlenecks y resolver problemas.

Cumplimiento normativo (y ético):

- Las 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/



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.

FastAPI

spring®

dj + django REST framework

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)

```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:

```python
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/