Introducción a microservicios con Flask

Sebastián Contreras Zambrano

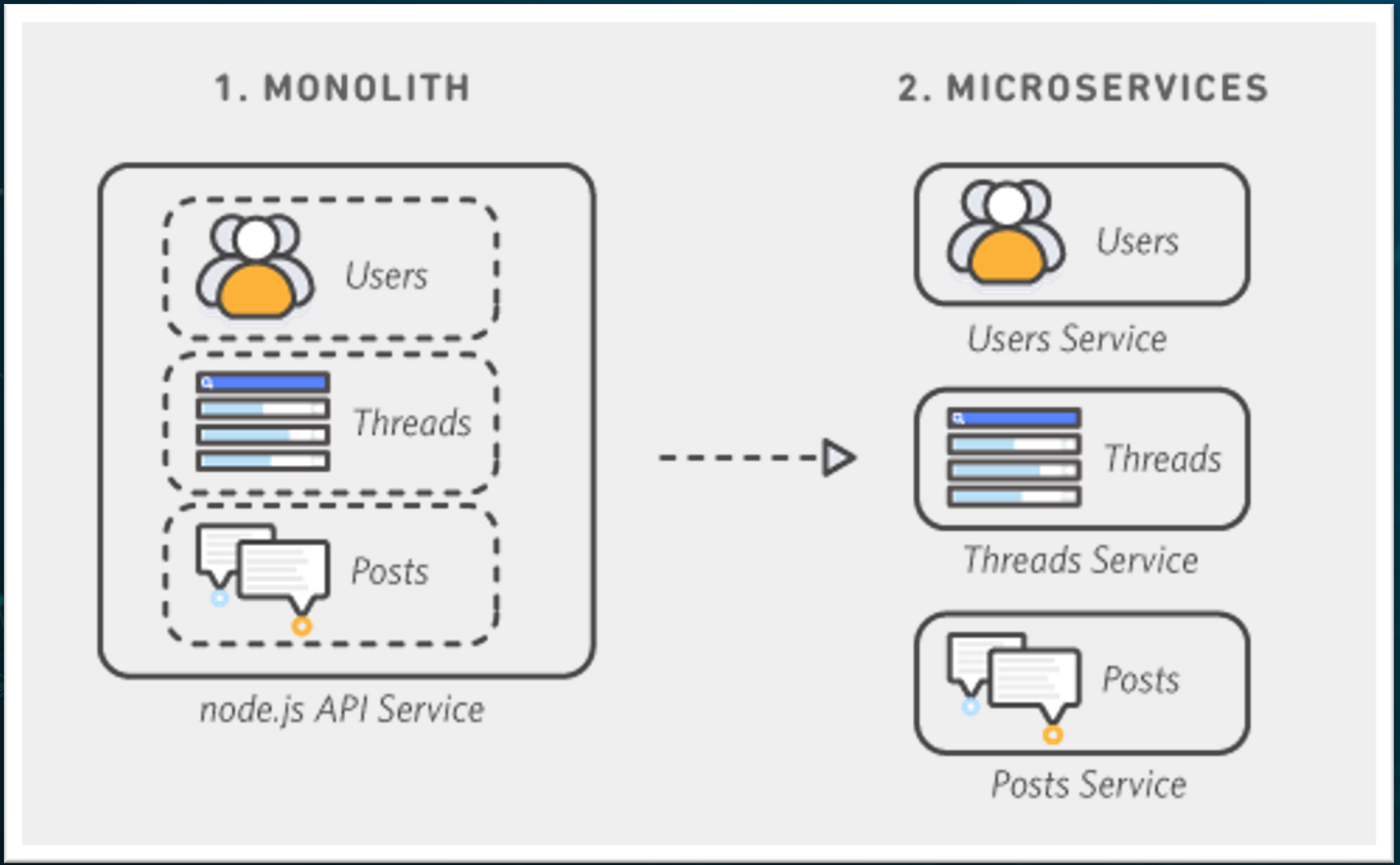
07 de julio, 2021

¿Qué son los microservicios?

- Los microservicios son un enfoque arquitectónico y organizativo para el desarrollo de **software donde el software está compuesto por pequeños servicios independientes que se comunican a través de API bien definidas**. Los propietarios de estos servicios son equipos pequeños independientes.
- Las arquitecturas de microservicios **hacen que las aplicaciones sean más fáciles de escalar y más rápidas de desarrollar**. Esto permite la innovación y acelera el tiempo de comercialización de las nuevas características.

Ref: ¿Qué son los microservicios? | AWS (amazon.com)

Ejemplo de monolítico vs microservicios

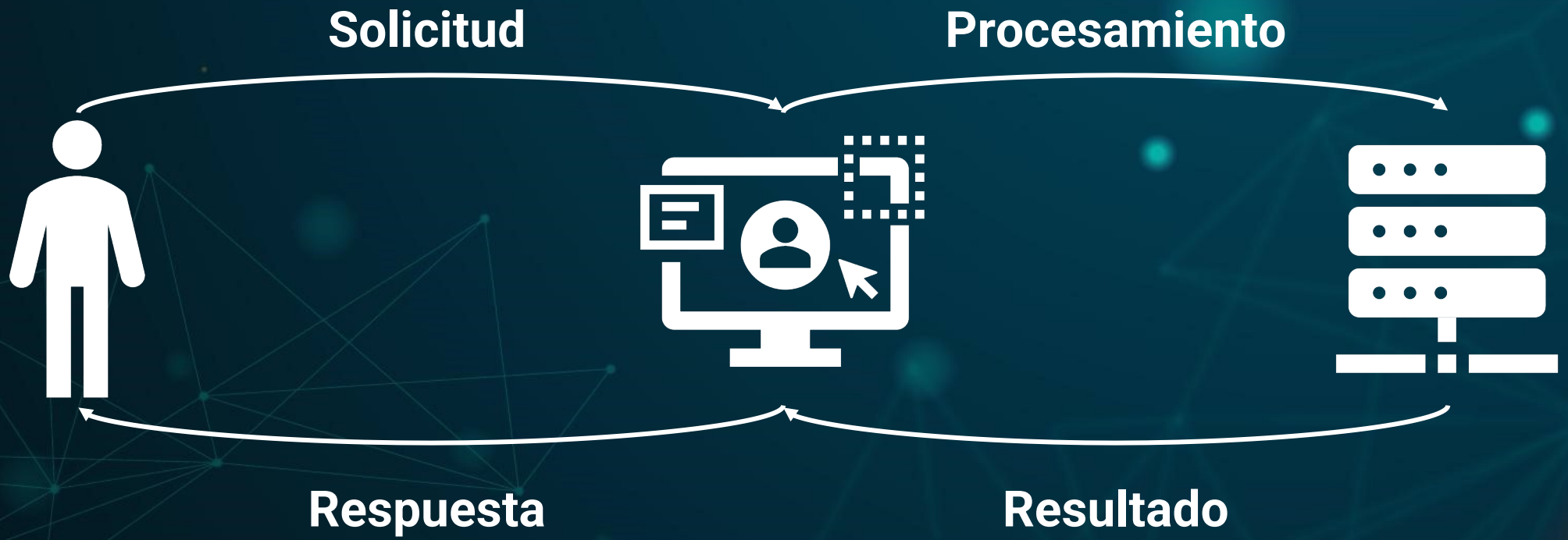


¿Qué es una API?

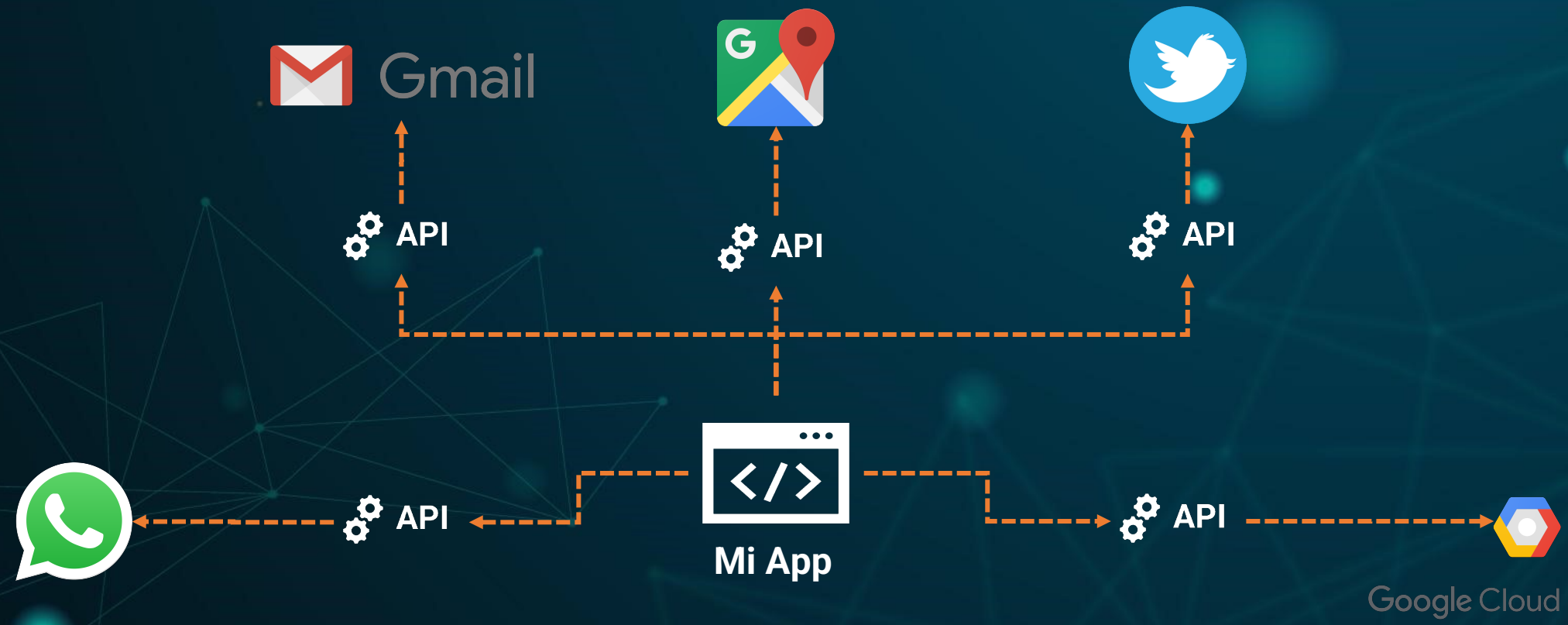
- API o **A**pplication **P**rogramming **I**nterfaces es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones.
- Las API permiten que sus productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados.
- Simplifican el desarrollo de las aplicaciones y permite ahorrar tiempo y dinero.

Ref: ¿Qué es una API? (redhat.com)

Esquema general de una API



Ejemplo de integración de APIs



Arquitecturas de las APIs

- Existen diversos tipos de arquitecturas de APIs, variando en formatos, protocolos e implementación de uso.

Las arquitecturas más utilizadas son:

- **SOAP (Simple Object Access Protocol):** Protocolo para el intercambio de información en XML entre cliente y un servicio.
- **REST (Representational State Transfer):** Arquitectura de desarrollo web basada en el estándar HTTP, haciendo uso de sus métodos (GET, POST, PUT, DELETE, PATCH, etc). Puede devolver mensajes en distintos formatos, por ejemplo, HTML, XML, raw text, JSON, etc.
- **GraphQL:** Lenguaje de consulta de datos para las APIs, desarrollado por Facebook el año 2012 y liberado como open source el año 2015, el cual en términos generales otorga las características de SQL al intercambio web, por lo cual, el cliente le consulta al servidor datos con ciertos parámetros, y el servidor en un entorno de ejecución los procesa y entrega la información al cliente.

Ejemplo de SOAP vs REST

SOAP

```
POST /obtener-precios HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml


<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:m="http://www.example.org">
    <soap:Header>
  </soap:Header>
    <soap:Body>
        <m:GetStockPriceRequest>
            <m:StockName>AAPL</m:StockName>
        </m:GetStockPriceRequest>
    </soap:Body>
</soap:Envelope>
```

REST

```
GET http://example.org/obtener-precios/AAPL
```

Ejemplo de REST vs GraphQL

REST

HTTP requests

```
GET http://example.org/obtener-precios/AAPL
```

HTTP response

```
{
    "name": "Apple Inc",
    "price": "139.96",
    "change": "2.69",
    "changePercent": "0.196",
    "changeType": "upside",
    "previousClose": "137.27",
    "open": "137.90",
    "volume": "78945572",
    "avgVolume": "82259870",
    "marketCap": "2.34T",
}
```

GraphQL

HTTP requests

```
POST http://example.org/graphql
query {
    stock (id: "APPL") {
        nombre: name,
        precio: price
    }
}
```

HTTP response

```
{
    “nombre": "Apple Inc",
    "precio": "139.96"
}
```

Frameworks para desarrollo API REST en Python

Existen muchos frameworks para el desarrollo de API REST en Python, alguno de los más polulares son:

- Flask
- Django
- Pyramid
- Bottle
- Falcon
- FastAPI
- Etc

¿Qué es Flask?

- Flask es un framework minimalista escrito en Python que permite crear aplicaciones web rápidamente y con un mínimo número de líneas de código. Está basado en la especificación WSGI de Werkzeug y el motor de templates Jinja2 y tiene una licencia BSD.

Ref: Flask - Wikipedia, la enciclopedia libre

Elementos claves en Flask

Desde la librería de `flask`, importamos la clase `Flask`

Instanciamos la clase `Flask`, a través de la variable `app`

Usamos el decorador `route` para crear un endpoint para nuestra `app`

Especifícamos los *metodos* HTTP que tendrá acceso el endpoint

Definimos el nombre de nuestro endpoint

Indicamos cuál será la respuesta que enviaremos al cliente cuando consulte el endpoint `/hola-mundo`

Ejecutamos nuestra `app` a través del método `run`

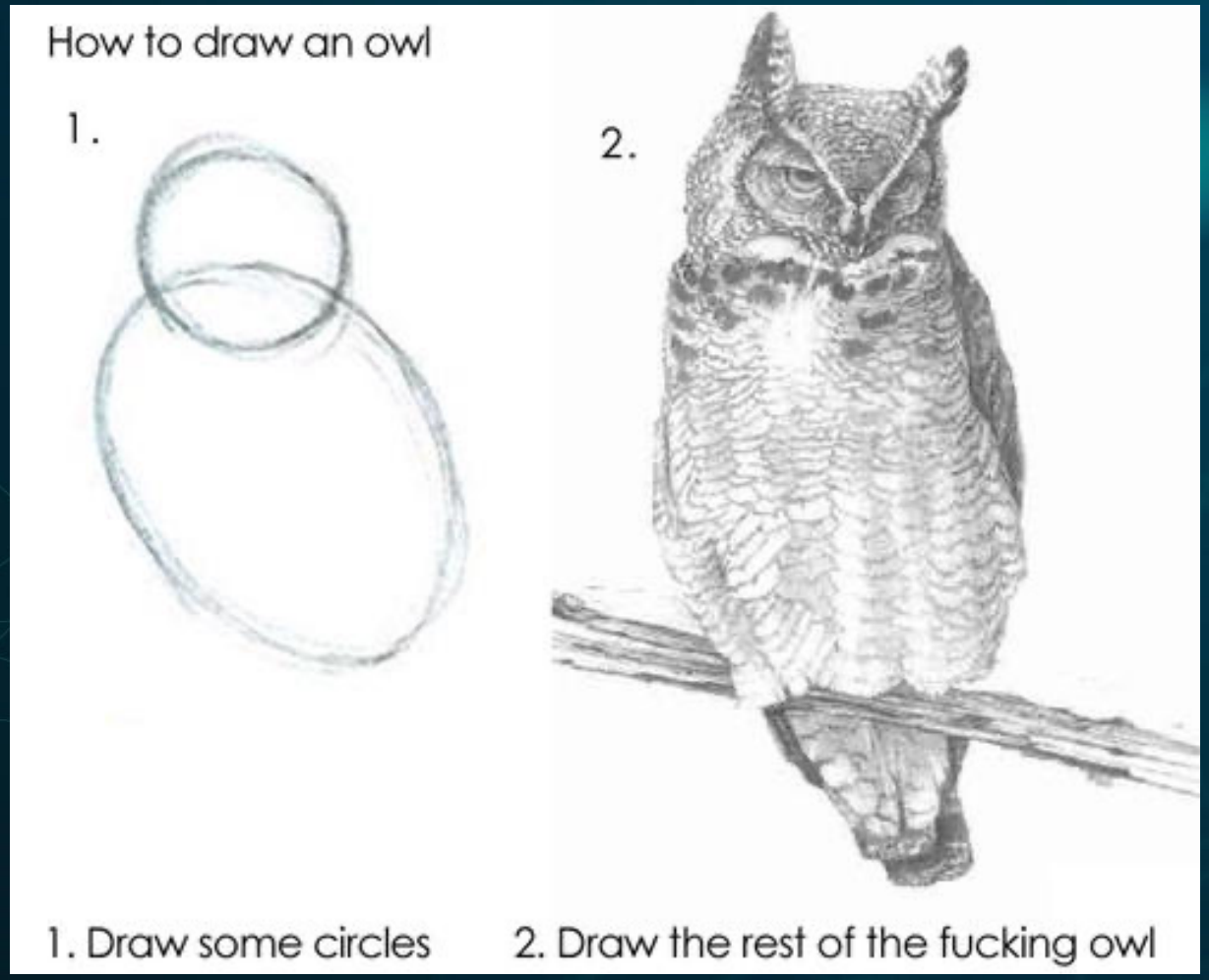
```python
from flask import Flask

app = Flask(__name__)

@app.route("/hola-mundo", methods=['GET', 'POST'])
def hello_world():
    return "<b>Hola Mundo!</b>"


if __name__ == '__main__':
    app.run()
```

Algunos métodos HTTP

- **GET:** El método GET solicita una representación de un recurso específico. Las peticiones que usan el método GET sólo deben recuperar datos.
- **POST:** El método POST se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor.
- **DELETE:** El método DELETE borra un recurso en específico.
- **PATCH:** El método PATCH es utilizado para aplicar modificaciones parciales a un recurso.

¡Vamos al código!

Espero no haya sido así...



Recursos

- Welcome to Flask — Flask Documentation (2.0.x) (palletsprojects.com)
- Métodos de petición HTTP - HTTP | MDN (mozilla.org)
- Códigos de estado de respuesta HTTP - HTTP | MDN (mozilla.org)
- Welcome to PyMySQL’s documentation! — PyMySQL 0.7.2 documentation
- Preventing SQL Injection Attacks With Python – Real Python
- Welcome to Flask-HTTPAuth’s documentation! — Flask-HTTPAuth documentation
- Basic Usage — flask-jwt-extended 4.2.3 documentation
- Quickstart — Requests 2.25.1 documentation (python-requests.org)
- python-dotenv · PyPI
- Postman | The Collaboration Platform for API Development
- Visual Studio Code - Code Editing. Redefined

¡Muchas gracias por su atención!

sebastiancz@live.cl
LinkedIn