Microservicios + APIs

•••

Fundamentals

Acerca del instructor...

Claudio César Sánchez Tejeda | DevOps Leader @ **Santander Tecnología**

demonccc@gmail.com / https://www.linkedin.com/in/demonccc / https://github.com/demonccc

Experiencia como:

- *Web Developer*
- *Sysadmin*
- *Cloud Architect*
- *DevOps / DevSecOps / BizDevOps*

¿Qué es XML?

XML es el acrónimo de **Extensible Markup Language**, es decir, es un lenguaje de marcado que define un conjunto de reglas para la codificación de documentos. .El lenguaje **XML** proporciona una plataforma para definir elementos para crear un formato y generar un lenguaje personalizado.

¿Qué es XML?

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <MUSICAS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  - <MUSICA>
      <NOME>A Fórmula Do Amor</NOME>
      <CANTOR>Kid Abelha</CANTOR>
      <LETRA>Eu tenho gestos aptos</LETRA>
    </MUSICA>
  - <MUSICA>
      <NOME>A Viagem</NOME>
      <CANTOR>Roupa Nova</CANTOR>
      <LETRA>Há tanto tempo que eu deixei você</LETRA>
    </MUSICA>
  - <MUSICA>
      <NOME>Águas De Março</NOME>
      <CANTOR>Elis Regina</CANTOR>
      <LETRA>É pau é pedra</LETRA>
    </MUSICA>
  </MUSICAS>
```

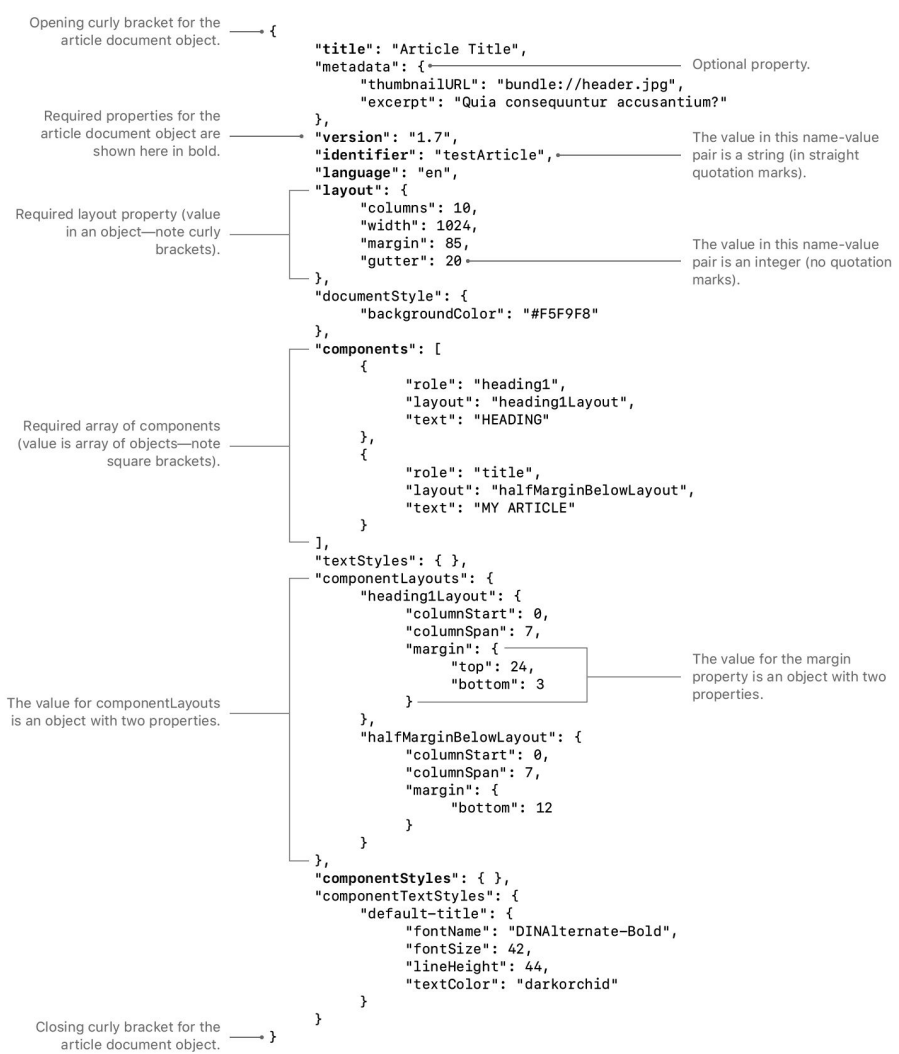
¿Qué es JSON?

JSON, cuyo nombre corresponde a las siglas **JavaScript Object Notation** o **Notación de Objetos de JavaScript**, se trata de un formato para guardar e intercambiar información que cualquier persona pueda leer. Los archivos **JSON** contienen solo texto y usan la extensión **.json**. Es básicamente una alternativa más simple y liviana al XML que cuenta con funciones similares.

JSON está constituído por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o sequencias.

¿Qué es JSON?




```
{
    "title": "Article Title",
    "metadata": {
        "thumbnailURL": "bundle://header.jpg",
        "excerpt": "Quia consequuntur accusantium?"
    },
    "version": "1.7",
    "identifier": "testArticle",
    "language": "en",
    "layout": {
        "columns": 10,
        "width": 1024,
        "margin": 85,
        "gutter": 20
    },
    "documentStyle": {
        "backgroundColor": "#F5F9F8"
    },
    "components": [
        {
            "role": "heading1",
            "layout": "heading1Layout",
            "text": "HEADING"
        },
        {
            "role": "title",
            "layout": "halfMarginBelowLayout",
            "text": "MY ARTICLE"
        }
    ],
    "textStyles": { },
    "componentLayouts": {
        "heading1Layout": {
            "columnStart": 0,
            "columnSpan": 7,
            "margin": {
                "top": 24,
                "bottom": 3
            }
        },
        "halfMarginBelowLayout": {
            "columnStart": 0,
            "columnSpan": 7,
            "margin": {
                "bottom": 12
            }
        }
    },
    "componentStyles": { },
    "componentTextStyles": {
        "default-title": {
            "fontName": "DINAlternate-Bold",
            "fontSize": 42,
            "lineHeight": 44,
            "textColor": "darkorchid"
        }
    }
}
```

¿Qué significa REST?

REST es una interfaz para conectar varios sistemas basados en el protocolo **HTTP** y sirve para obtener y generar datos y operaciones, devolviendo esos datos en formatos muy específicos, como **XML** y **JSON**.

El formato más usado en la actualidad es el formato **JSON**, ya que es más ligero y legible en comparación al formato **XML**. Elegir uno será cuestión de la lógica y necesidades de cada proyecto.

REST se apoya en **HTTP**, los verbos que utiliza son exactamente los mismos, con ellos se puede hacer **GET**, **POST**, **PUT** y **DELETE**.

Caracteristicas de REST

Cliente-servidor: El servidor se encarga de controlar los datos mientras que el cliente se encarga de manejar las interacciones del usuario. Esta restricción mantiene al cliente y al servidor débilmente acoplados (el cliente no necesita conocer los detalles de implementación del servidor y el servidor se “despreocupa” de cómo son usados los datos que envía al cliente).

Sin estado: aquí decimos que cada petición que recibe el servidor debería ser independiente y contener todo lo necesario para ser procesada.

Cacheable: debe admitir un sistema de almacenamiento en caché. Este almacenamiento evitará repetir varias conexiones entre el servidor y el cliente para recuperar un mismo recurso.

Interfaz uniforme: define una interfaz genérica para administrar cada interacción que se produzca entre el cliente y el servidor de manera uniforme, lo cual simplifica y separa la arquitectura. Esta restricción indica que cada recurso del servicio REST debe tener una única dirección o “URI”.

Sistema de capas: el servidor puede disponer de varias capas para su implementación. Esto ayuda a mejorar la escalabilidad, el rendimiento y la seguridad.

Qué es API REST

La palabra clave es **Interface** (interfaz), que es una capa de abstracción para que dos sistemas se comuniquen. En el ámbito web, podríamos decir que una **API** es un servicio backend que se utiliza para conectar dos aplicaciones.

Se definen una serie de métodos **HTTP** que pueden hacer lo que queramos: cambios en base de datos, autenticar usuarios, llamadas a otros procesos, etc. Tal y como he mencionado anteriormente, los verbos que se utilizan son: **GET**, **POST**, **PUT** y **DELETE**. Además, cualquier dispositivo que sepa cómo utilizar **HTTP** será capaz de consumir una **API REST**.

Una **API REST** es un backend capaz de contestar a las llamadas a una serie de URLs en formato **JSON** y que también es capaz de recibir **JSON** para gestionar la información que le enviemos.

Qué es API Gateway

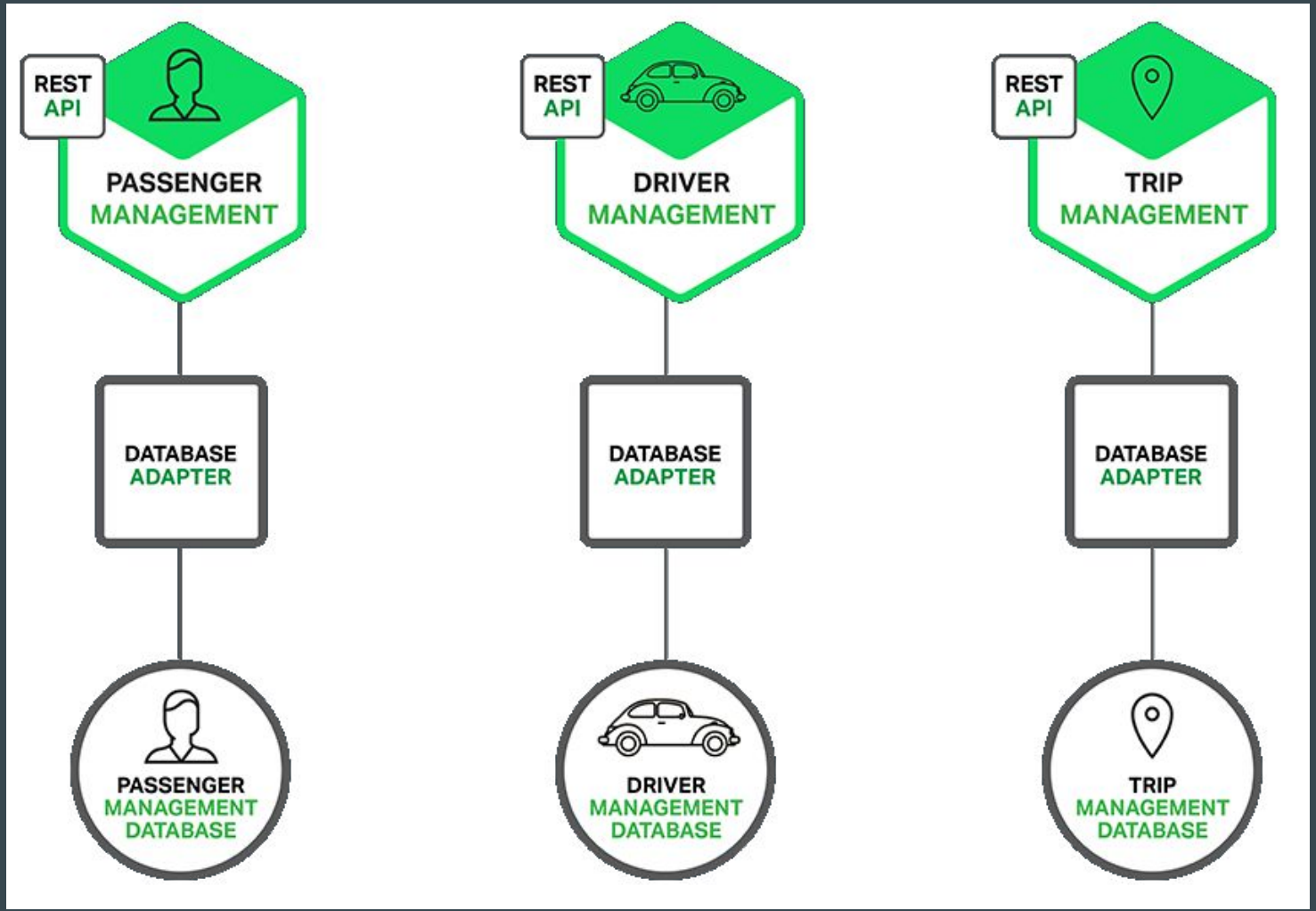
Un **API Gateway** es la pieza encargada de unificar la publicación de APIs para que sean consumidas por otras aplicaciones o por los desarrolladores. ... Intercambiador de APIs: Componente cuya principal función es la de habilitar la conexión entre los servicios y los clientes.

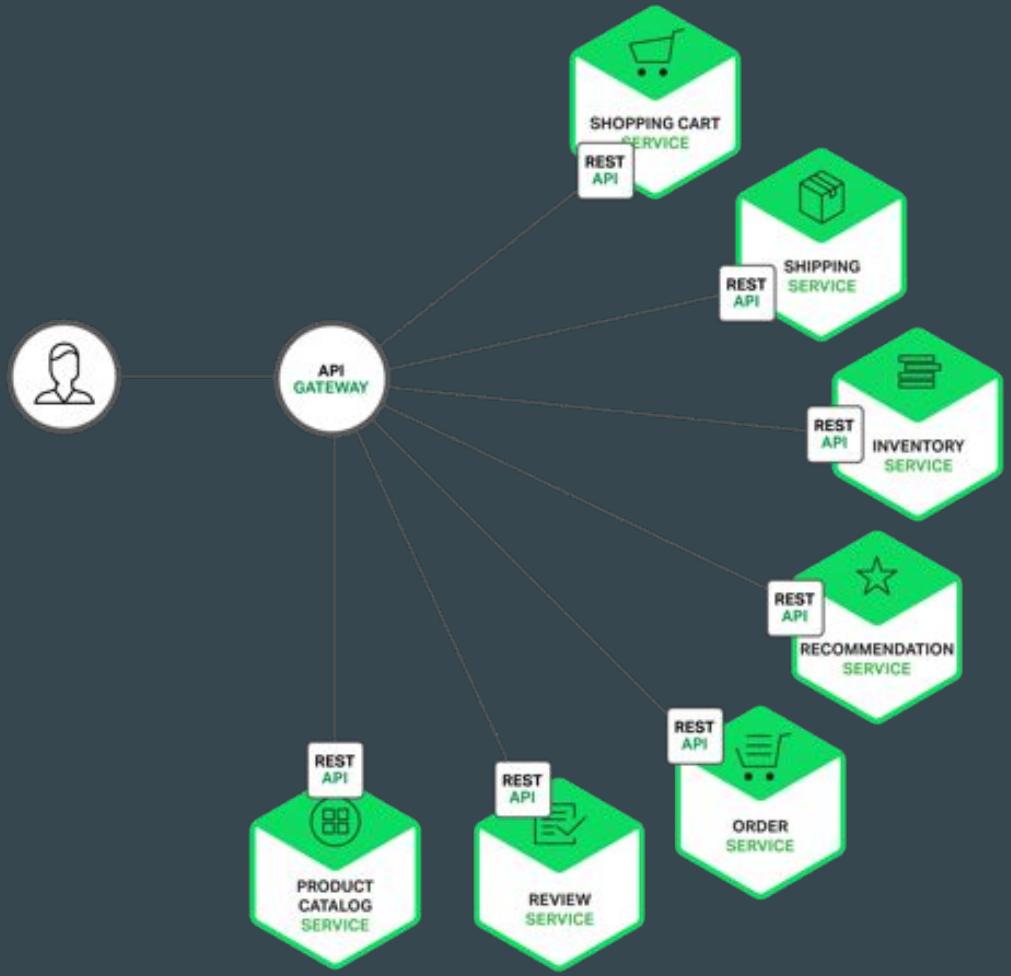
Microservicios

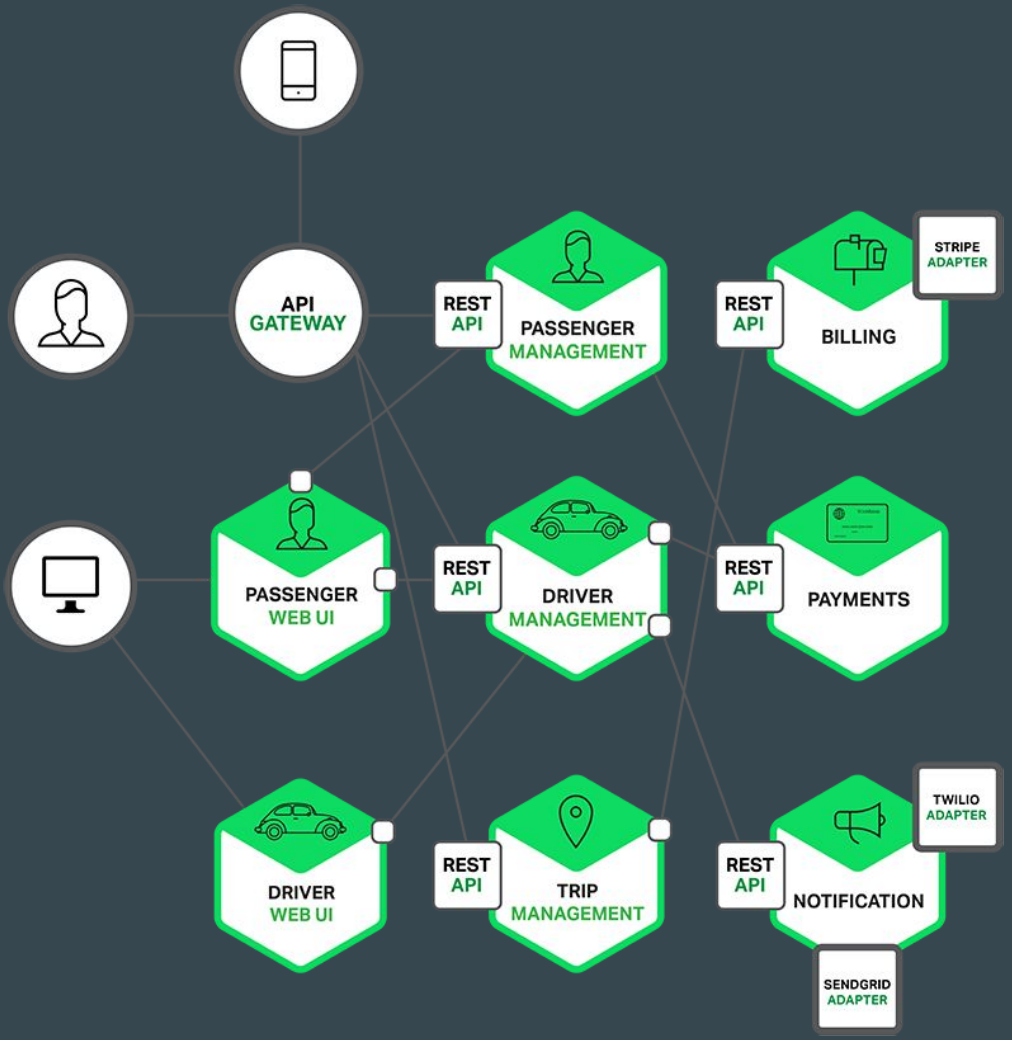
Una arquitectura de microservicios es un enfoque para desarrollar una aplicación software como una serie de pequeños servicios, cada uno ejecutándose de forma autónoma y comunicándose entre sí, por ejemplo, a través de peticiones HTTP a sus API.

Normalmente hay un número mínimo de servicios que gestionan cosas comunes para los demás (como el acceso a base de datos), pero cada microservicio es pequeño y corresponde a un área de negocio de la aplicación.


MYSQL
Monolithic Architecture
MYSQL ADAPTER
PASSENGER
DRIVER
REST API
TWILIO ADAPTER
PASSENGER MANAGEMENT
BILLING
NOTIFICATION
PAYMENTS
TRIP MANAGEMENT
DRIVER MANAGEMENT
WEB UI
SENDGRID ADAPTER
STRIPE ADAPTER


REST API
PASSENGER MANAGEMENT
DATABASE ADAPTER
PASSENGER MANAGEMENT DATABASE
REST API
DRIVER MANAGEMENT
DATABASE ADAPTER
DRIVER MANAGEMENT DATABASE
REST API
TRIP MANAGEMENT
DATABASE ADAPTER
TRIP MANAGEMENT DATABASE


SHOPPING CART
SERVICE
REST
API
SHIPPING
SERVICE
REST
API
API
GATEWAY
INVENTORY
SERVICE
REST
API
RECOMMENDATION
SERVICE
REST
API
ORDER
SERVICE
REST
API
REVIEW
SERVICE
REST
API
PRODUCT
CATALOG
SERVICE
REST
API


API
GATEWAY
REST
API
PASSENGER
MANAGEMENT
REST
API
BILLING
STRIPE
ADAPTER
PASSENGER
WEB UI
REST
API
DRIVER
MANAGEMENT
REST
API
PAYMENTS
DRIVER
WEB UI
REST
API
TRIP
MANAGEMENT
REST
API
NOTIFICATION
TWILIO
ADAPTER
SENDGRID
ADAPTER

Dudas y preguntas

Gracias!