UNIVERSIDAD TECNOLÓGICA DE PARR

REPORTE DE INVESTIGACIÓN

Tipos de Apis

INGENIERÍA EN REDES INTELIGENTES Y CIBERSEGURIDAD

P R E S E N T A:

T.S.U. Sandra Viviana Estrada Morales

D O C E N T E:

Judith Anahi Felix Felix

F E C H A:

06|07|2024

INDICE GENERAL

INDICE DE ILUSTRACIONES

INTRODUCCIÓN

En el ecosistema tecnológico actual, las APIs (Application Programming Interfaces) desempeñan un rol fundamental al facilitar la interacción entre diferentes aplicaciones y servicios.

El diseño y la arquitectura de una API no solo determinan su funcionalidad y eficiencia, sino que también impactan significativamente en la experiencia del desarrollador y del usuario final.

Este reporte investiga los diversos tipos de estilos de diseño y arquitecturas de APIs, proporcionando ejemplos concretos para ilustrar cada enfoque.

Desde los principios REST y RPC hasta las arquitecturas de microservicios y orientadas a eventos, cada modelo tiene sus características únicas que influyen en cómo se construyen, implementan y consumen las APIs en aplicaciones modernas.

Comprender estas diferencias es crucial para tomar decisiones informadas durante el desarrollo de software, optimizando la comunicación y la interoperabilidad entre sistemas distribuidos en un entorno cada vez más interconectado y dinámico.

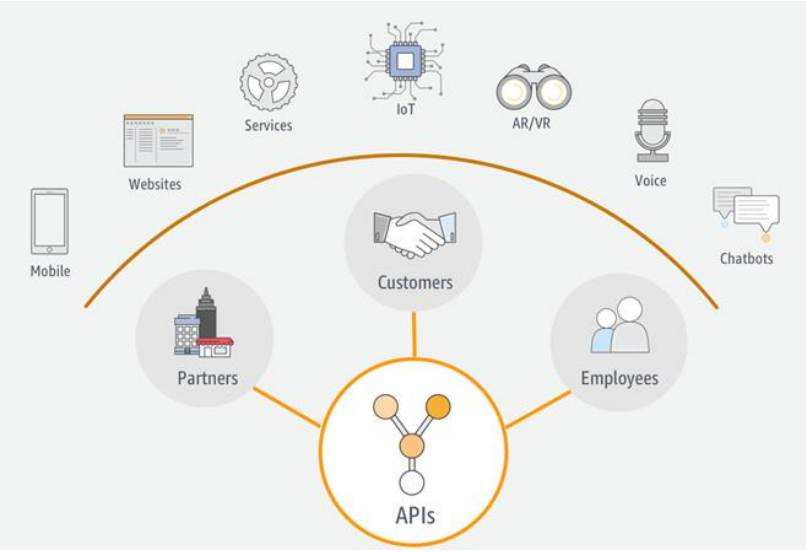

Ilustración 1 Como se implementan las Apis

DESARROLLO

2.1 Tipos de Estilos de Diseño de APIs

REST es un estilo arquitectónico que define un conjunto de restricciones y principios para el diseño de servicios web que se utilizan en aplicaciones web y en servicios web.

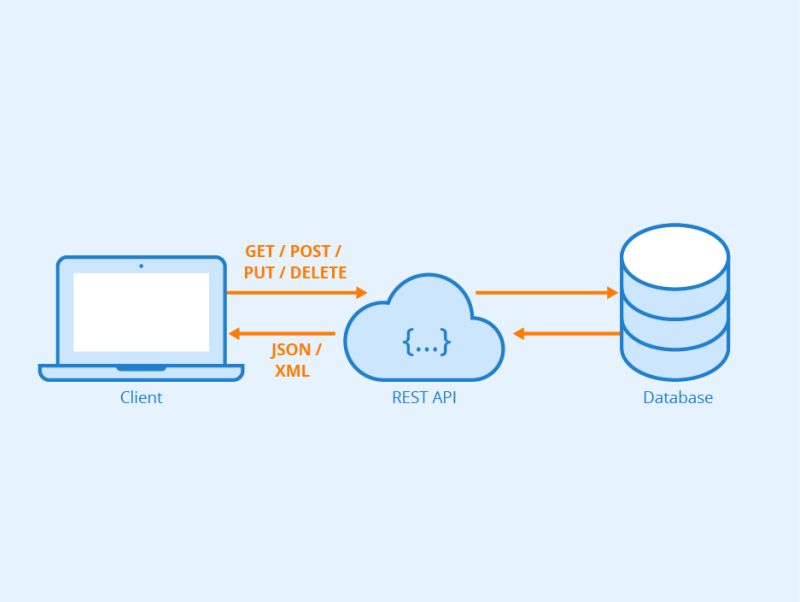

Ilustración 2 Rest Api

RPC es un estilo de diseño de API que permite a un programa solicitar un servicio de un programa situado en otra máquina en la red sin tener que entender los detalles de la red.

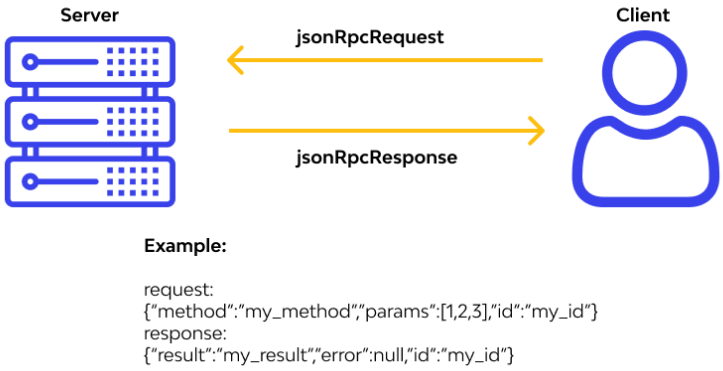

Ilustración 3 RPC Api

2.2 Ejemplo de API REST: GitHub API

GitHub proporciona una API pública que sigue los principios REST para permitir a los desarrolladores interactuar con los repositorios, usuarios, organizaciones y otros recursos de GitHub. Algunas características clave de la GitHub API son:

- **Recursos**: Cada entidad como repositorios, issues, pull requests, y usuarios se representa como un recurso identificado por una URI única.
- **Métodos HTTP**: Utiliza métodos estándar de HTTP como GET, POST, PUT y DELETE para realizar operaciones sobre estos recursos.
- **Formato de datos**: La API de GitHub generalmente responde con datos en formato JSON, facilitando su consumo y manipulación por parte de aplicaciones clientes.
- **Autenticación**: Utiliza OAuth para la autenticación de usuarios y tokens de acceso para la autorización de las solicitudes.

Ilustración 4 Api de GitHub

2.3 Ejemplo de API RPC: JSON-RPC

JSON-RPC es un protocolo ligero que permite a los clientes enviar solicitudes y recibir respuestas utilizando el formato JSON sobre una conexión HTTP o HTTPS. A diferencia de REST, JSON-RPC se centra en la invocación de métodos o funciones específicas del servidor sin necesidad de exponer recursos individuales.

Algunas características de JSON-RPC son:

- **Estructura de solicitud**: Las solicitudes y respuestas se envían como objetos JSON, con un campo "method" que indica la función a ejecutar y un campo "params" que contiene los parámetros necesarios para la función.
- **Simplicidad**: JSON-RPC se centra en la simplicidad y la eficiencia en la transmisión de datos, ideal para aplicaciones que necesitan ejecutar acciones específicas en un servidor remoto.
- **Ejemplo de solicitud JSON-RPC**: Una solicitud JSON-RPC para llamar a un método "sum" que suma dos números en un servidor remoto podría verse así:

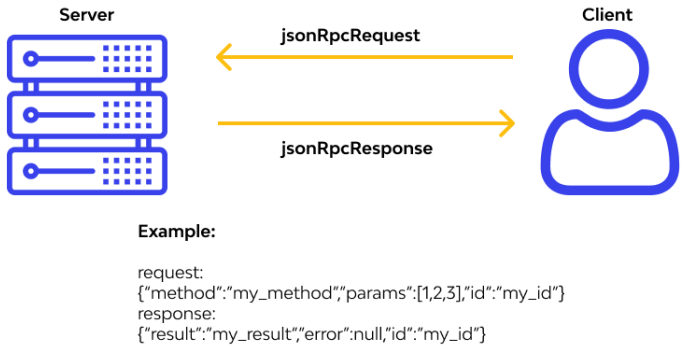

Ilustración 5 Ejemplo RPC

Tipos de Arquitecturas de APIs

2.4 Arquitectura basada en microservicios

La arquitectura de microservicios es un enfoque arquitectónico para desarrollar una sola aplicación como un conjunto de pequeños servicios, cada uno ejecutándose de manera independiente y comunicándose entre sí a través de protocolos ligeros como HTTP.

- **Desacoplamiento**: Cada microservicio es autónomo y se puede desarrollar, desplegar y escalar de forma independiente.
- **Escalabilidad**: Los microservicios permiten escalar partes específicas de una aplicación según la demanda, en lugar de escalar toda la aplicación.
- **Facilidad de mantenimiento**: Los equipos pueden trabajar de manera más eficiente en servicios más pequeños y específicos, facilitando las actualizaciones y correcciones de errores.

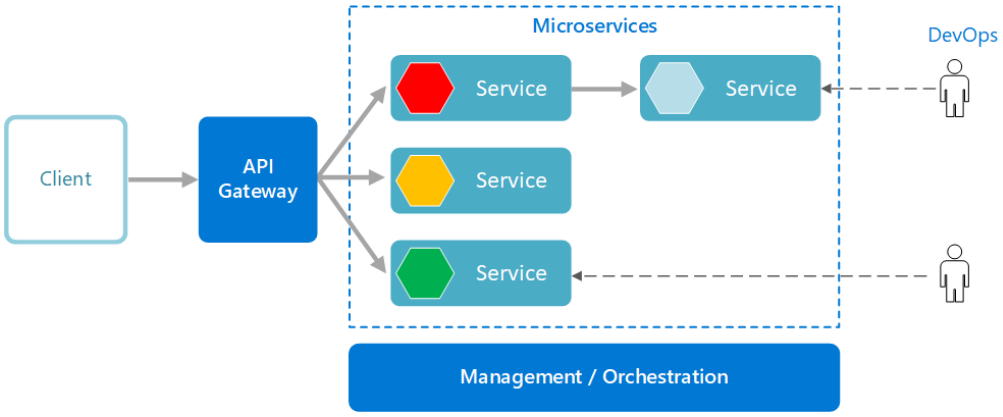

Ilustración 6 Arquitectura en microservicios

2.5 Arquitectura orientada a eventos

La arquitectura orientada a eventos se centra en la generación, detección, consumo y reacción a eventos dentro de un sistema informático. Los eventos pueden ser cualquier cambio de estado significativo que ocurra dentro del sistema o en el entorno externo. Algunas características de esta arquitectura son:

- **Desacoplamiento y escalabilidad**: Los componentes del sistema se comunican a través de eventos, lo que permite un acoplamiento más débil entre ellos y facilita la escalabilidad.
- **Procesamiento asincrónico**: Los eventos son procesados de manera asincrónica, lo que mejora el rendimiento y la capacidad de respuesta del sistema.
- **Resiliencia**: La arquitectura orientada a eventos facilita la tolerancia a fallos al manejar eventos perdidos o fallidos de manera efectiva.

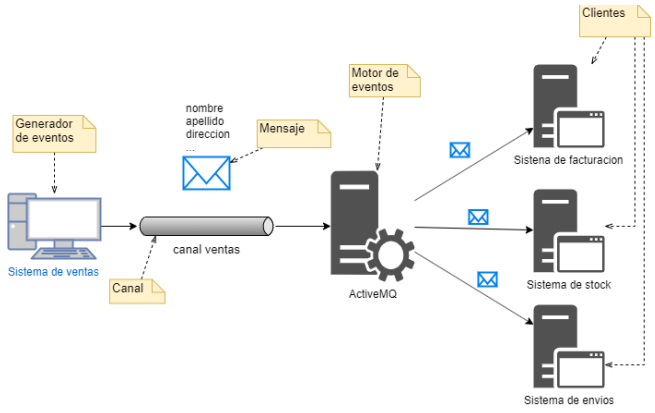

Ilustración 7Arquitectura orientada a eventos

CONCLUSIONES

El estilo REST, basado en principios como recursos identificables y operaciones estándar de HTTP, ofrece una estructura flexible y ampliamente adoptada que favorece la interoperabilidad y la simplicidad en la comunicación entre sistemas.

En cuanto a las arquitecturas, la adopción de microservicios permite a las organizaciones escalar y mantener aplicaciones de manera más eficiente al dividirlas en componentes independientes y gestionables. Esta arquitectura ha sido fundamental para plataformas como Netflix, que gestionan grandes volúmenes de datos y demandas de usuarios.

Referencias

Richardson, L., & Amundsen, M. (2013). RESTful Web APIs: Services for a Changing World. O'Reilly Media.

Fowler, M. (2014). Microservices: a definition of this new architectural term. Recuperado de https://martinfowler.com/articles/microservices.html

Kafka Documentation. Apache Kafka. Recuperado de https://kafka.apache.org/
GitHub Developer Documentation. GitHub API. Recuperado de https://docs.github.com/en/rest

Tilkov, S. (2015). REST und HTTP: Entwicklung und Integration nach dem Architekturstil des Web. dpunkt.verlag.