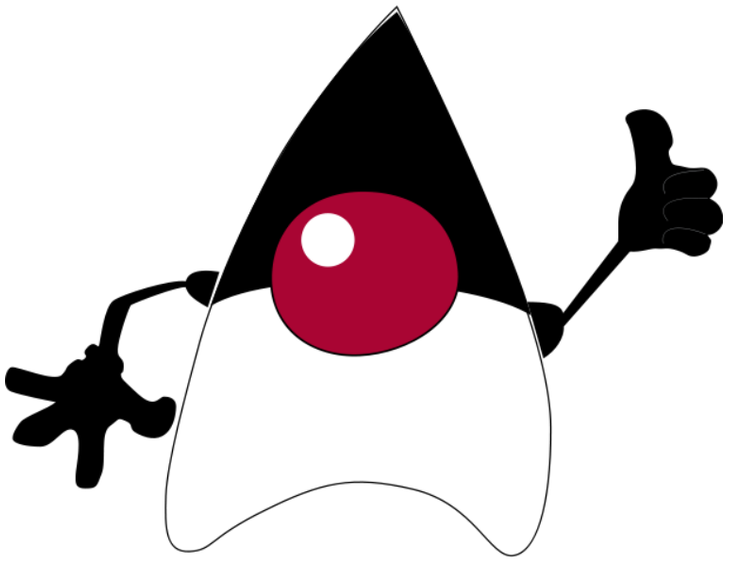
MAINT

Java

{ REST:API }

VERT.X



eclipse

Apache maven

Istio

spring BOOT {Microservicios}

mongoDB

docker

Manuel Vega Ulloa

- Master en “Desarrollo Estratégico y Gestión de la Innovación”
- Ingeniero de Sistemas
- Certificado en:
  - IBM App Connect Enterprise
  - IBM Integration Bus
  - IBM Message Broker
  - IBM DB2
  - VisualAge for Java
  - VisualAge for Smalltalk

API y Microservicios

Java

PSD2

PSD2 es una regulación europea en los servicios de pagos digitales.

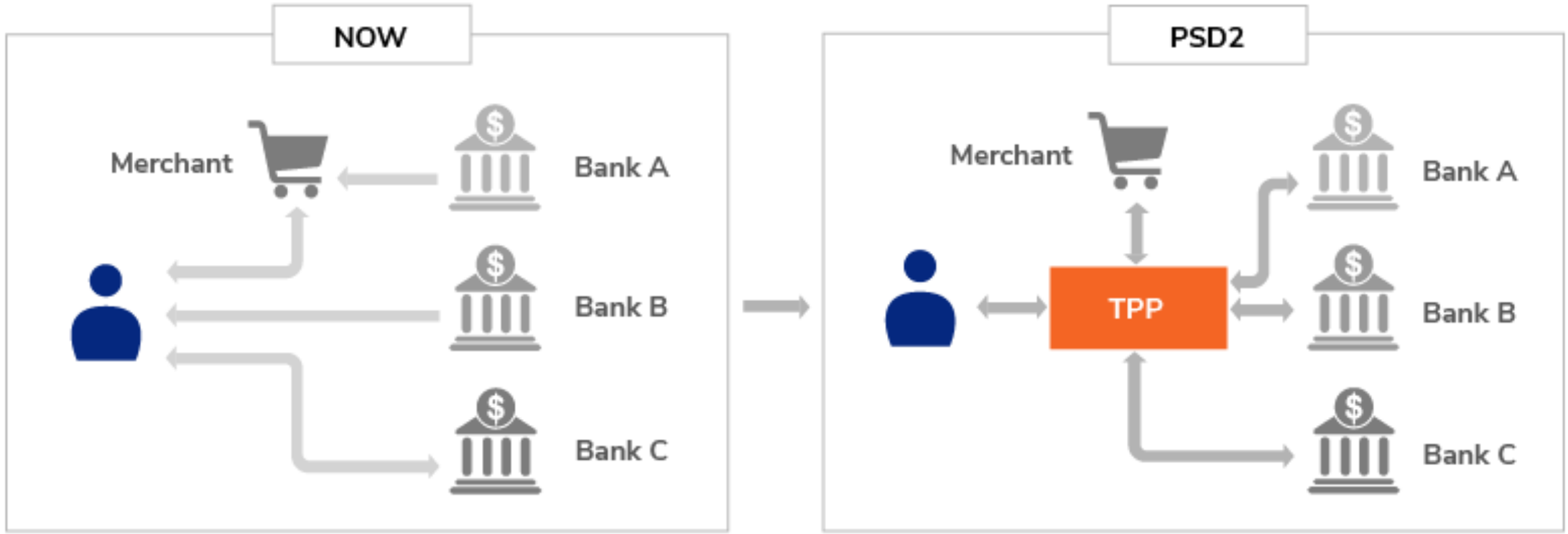
Este sistema facilita los pagos en toda Europa con mayor seguridad además de ofrecer un servicio bancario adaptado a las nuevas tecnologías.

Con ella se pone de manifiesto una vez más la importancia que está adquiriendo el mundo de las ‘APIS’ o ‘Application Program Interface’ en distintos sectores financieros.

PSD2

La verdadera novedad de PSD2 es la apertura de los servicios de pago de los bancos a terceras partes, los conocidos como Third Party Payment Service Providers (TPPs).

Esto significa que otras empresas tendrán acceso a las cuentas bancarias de los clientes, siempre que éstos aporten su consentimiento expreso, además del inicio de pagos en su nombre.


NOW
Merchant
Bank A
Bank B
Bank C
PSD2
Merchant
TPP
Bank A
Bank B
Bank C


Bank
APIs
TPP
Applications

APIs: clave en PSD2

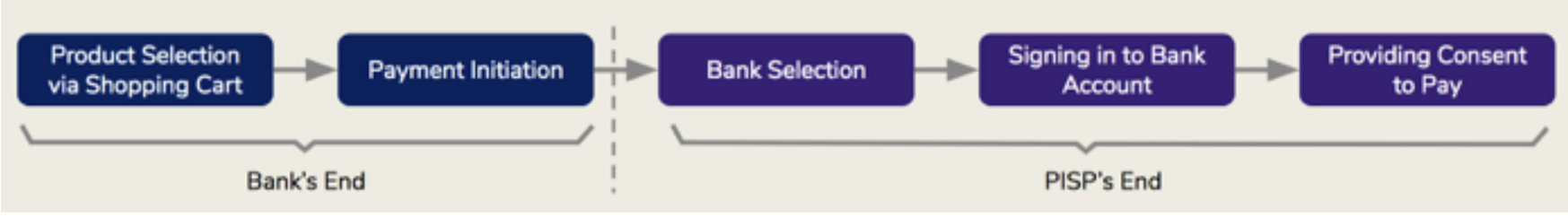
Una vez integradas las APIs, éstas permitirán compartir información entre los distintos proveedores:

- los PSIP (Proveedores de Servicios de Iniciación de Pagos), son aquellos que permiten que un cliente pueda realizar un pago desde su cuenta de forma directa a un establecimiento;

- y los PSIC (Proveedores de Servicios de Información sobre Cuentas), permiten al usuario informarse de forma conjunta de la situación de las distintas cuentas que tiene en los diferentes bancos.

MAINT.

APIs: clave en PSD2



APIs: clave en PSD2

Puede ser que tantas siglas te resulte lioso, lo importante es recordar que tanto los PSIP como los PSIC, son proveedores de servicio de pago a terceros (TTPs).

“Los bancos tienen que controlar quién accede a esa información y cómo la está utilizando.”

–Álvaro Martín, representante de BBVA

APIs: clave en PSD2

Gracias al PSD2, bueno concretamente a las APIs, las entidades financieras conocerán mejor a sus clientes, lo que permite ofrecerles una oferta de productos personalizada, optimizando el riesgo.

ETAPA 1:

Cumplimiento

Adherirse a las NORMAS PSD2

Una plataforma de administración de API para proporcionar datos a terceros.

Gestión de acceso de identidad y seguridad de API, con una potente autentificación del cliente (Strong Customer Authentication)

Monitoreo en tiempo real de uso de API.

La API permite la capacidad de monetización.

ETAPA 2:

Recuperar la lealtad

Convertirse en un PSP y AISP

Integración con las APIs de otros bancos para construir un portal central, para aquellos clientes que pertenezcan a múltiples entidades financieras.

Autenticación de usuario federado con los bancos de confianza.

Detección de fraude para alertar sobre anomalías en las transacciones.

Tableros de datos específicos y de API, para proporcionar diferentes puntos de vista sobre datos financieros.

ETAPA 3:

Transformación digital

Ofrecer servicios más allá de los bancarios

Conjunto de aplicaciones web y móviles que habilitan servicios digitales centrados en los clientes.

Adoptar una visión perspicaz sobre las ventas, para transformar el depósito de datos del cliente en información comercial.

Análisis exhaustivos para recopilar datos de varias APIs, con el fin de crear productos y servicios enfocados a conseguir más ventas.

Capacidades de la plataforma para construir tecnologías completas que se integran a la perfección con los sistemas existentes y nuevos.

Laboratorio: Instalación de ambiente

Demo

https://www.openbanking.org.uk

https://psd2-api.openbankproject.com/index

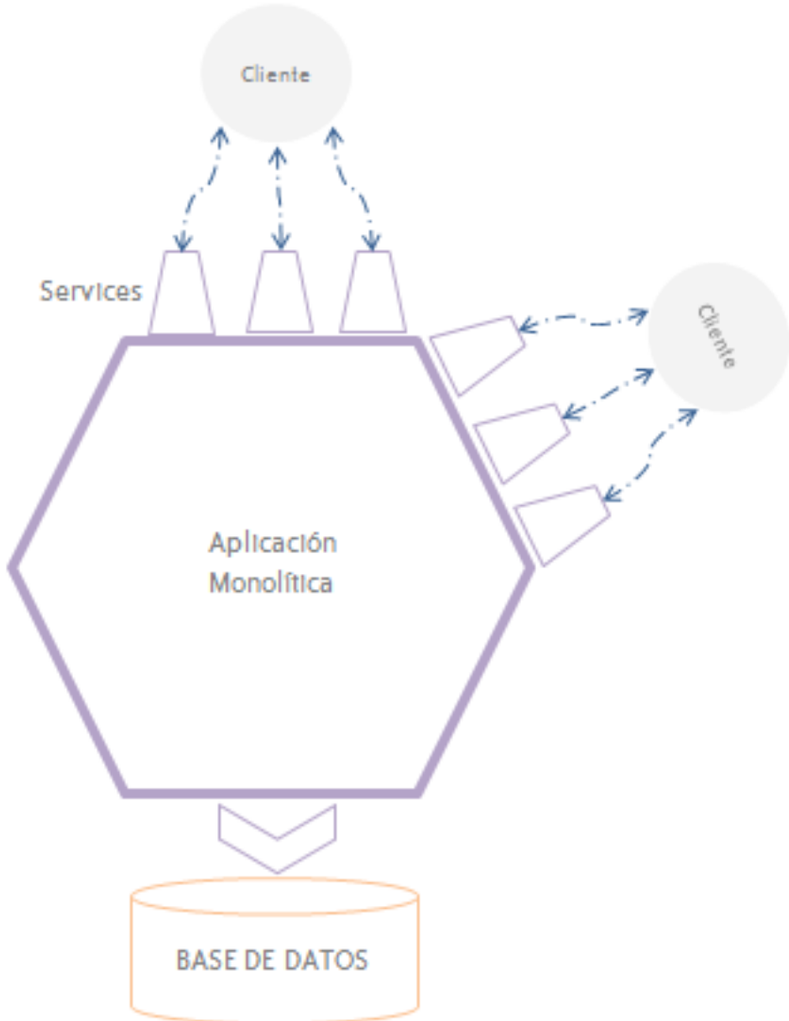
https://www.bbvaapimarket.com/products/customers

http://editor.swagger.io

Microservicios

Microservices, Modelos de Implementacion

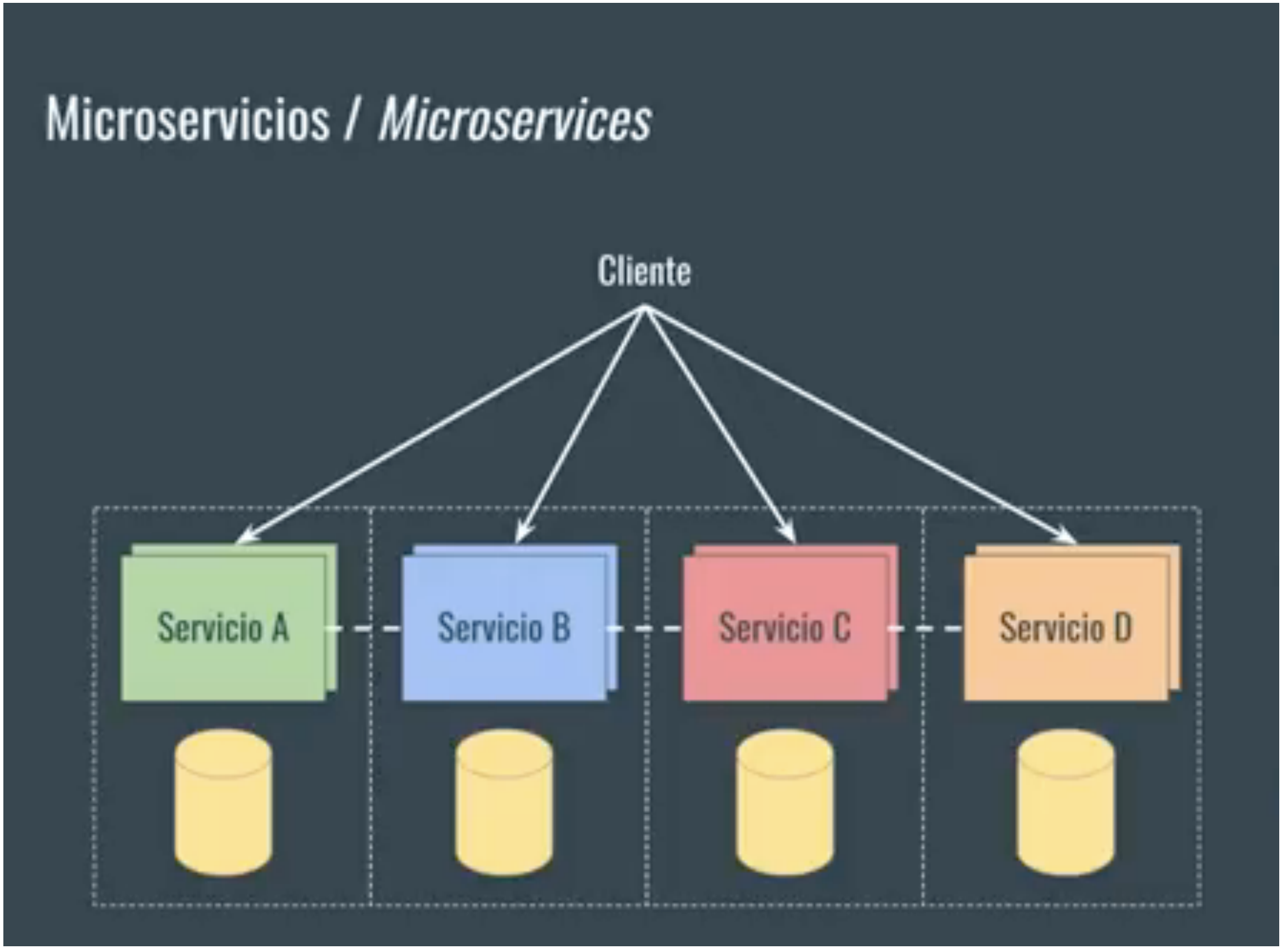
Historia


Cliente
Services
Cliente
Aplicación
Monolítica
BASE DE DATOS

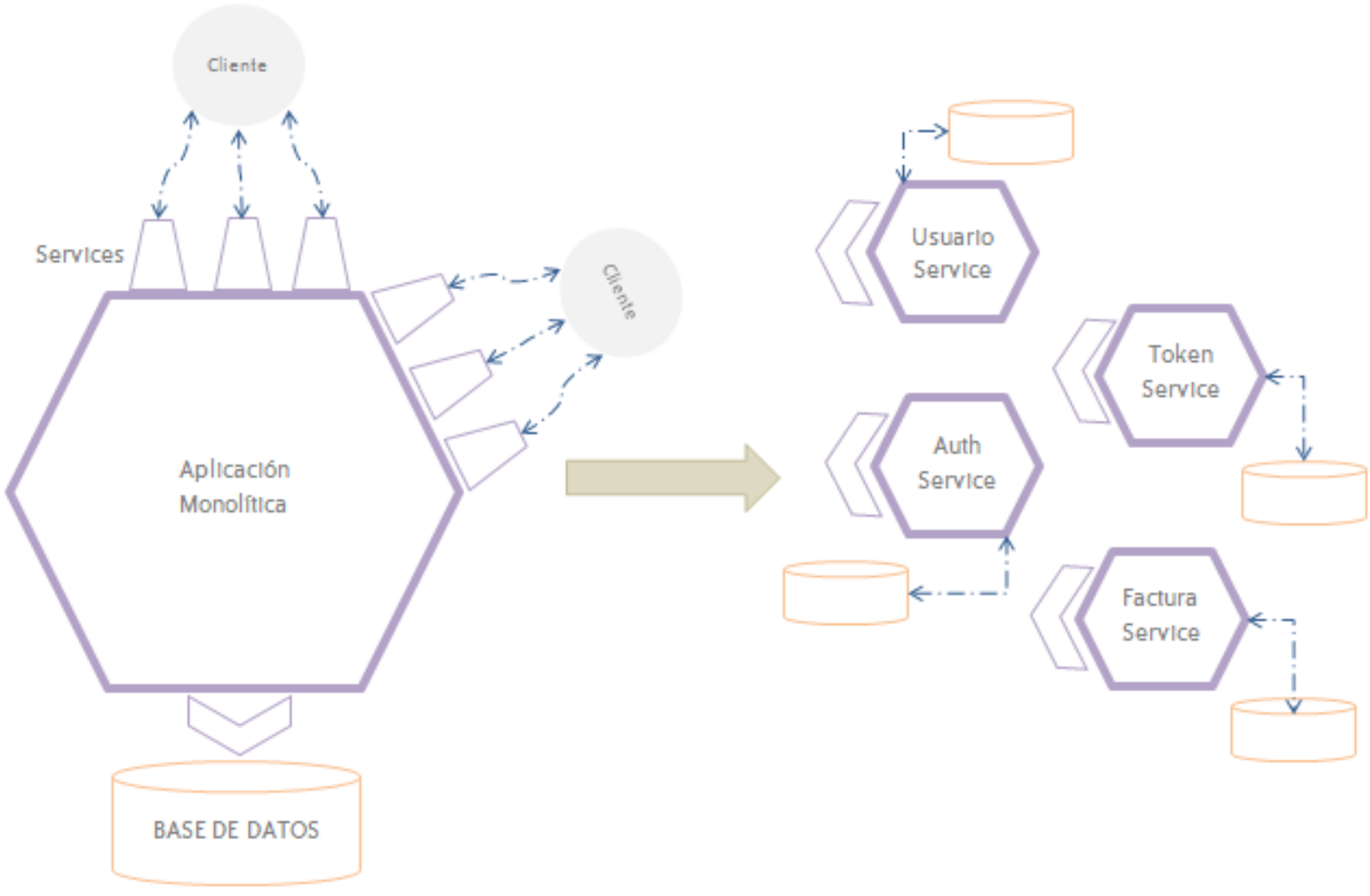
Microservicios, clave para el desarrollo de aplicaciones y servicios.

La base de la arquitectura de microservicios o MSA, se basa en el desarrollo de una única aplicación como un conjunto de grano fino y servicios independientes, encargándose cada microservicio de su propio proceso, desarrollo, y despliegue de forma independiente.

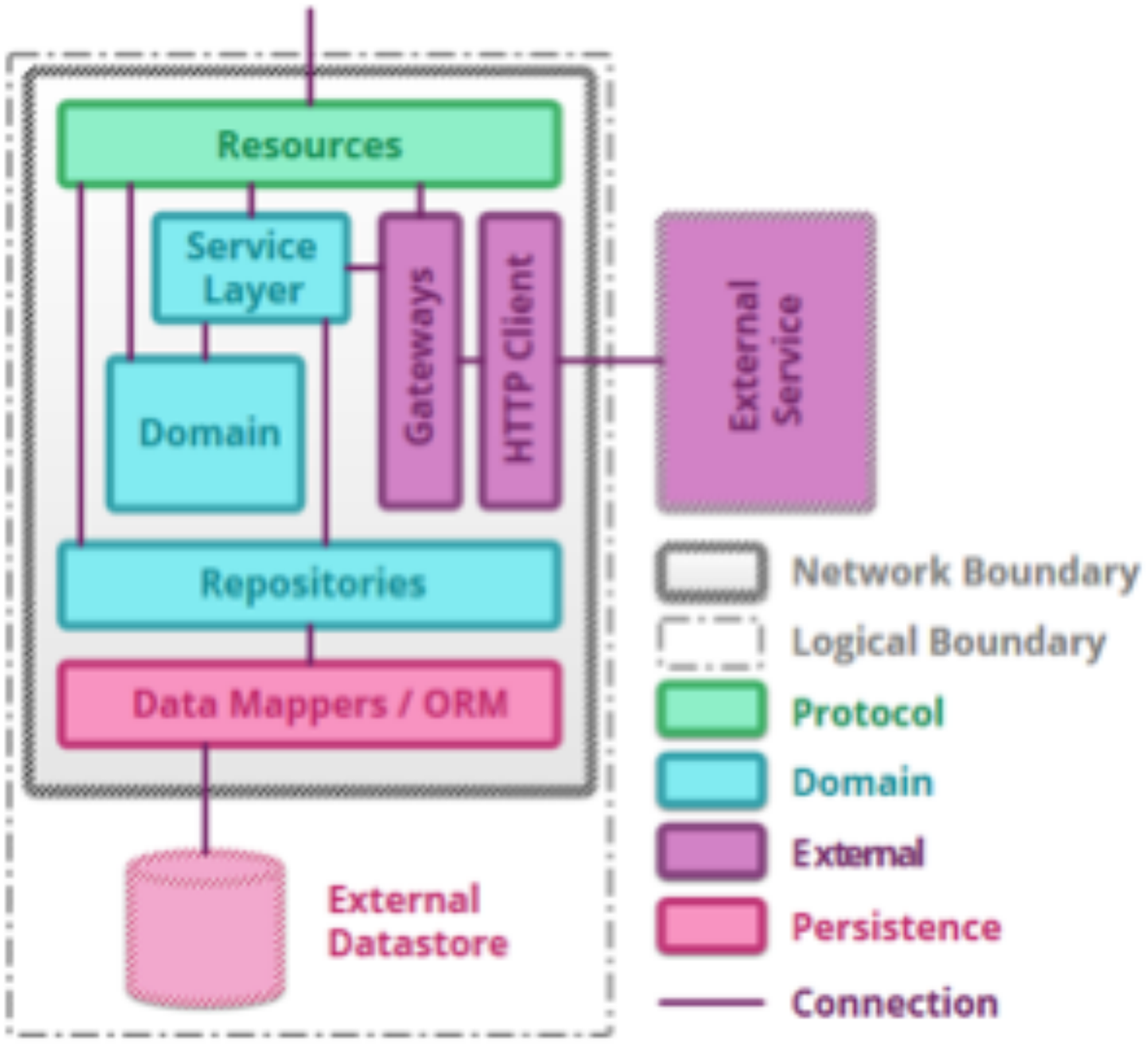
Los microservicios han ido evolucionando con el tiempo, y debido a ello, en la actualidad la lógica del negocio y la comunicación de red está dispersa en servicios independientes, lo que supone la eliminación del ESB central.


Microservicios / *Microservices*
Cliente
Servicio A
Servicio B
Servicio C
Servicio D

Qué es un microservicio

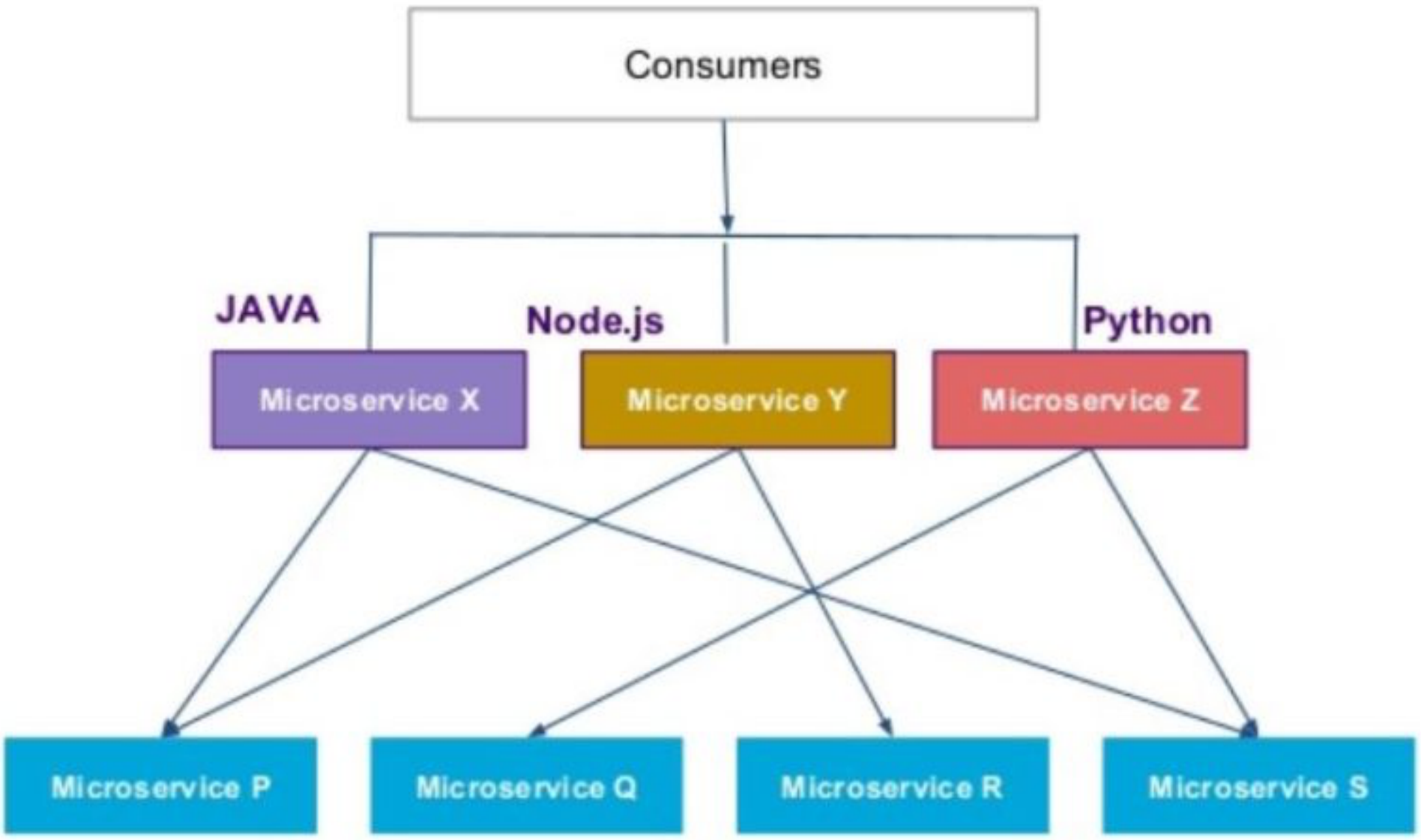


Qué es un microservicio



Microservicios, clave para el desarrollo de aplicaciones y servicios.

Los microservicios dependen en gran medida de la comunicación entre los mismos, a lo que añadía que existen varios patrones de comunicación, como el envío de mensajes sincronizados basados en protocolos muy conocidos y ampliamente adoptados (HTTP). Además, es importante destacar que la mensajería asincrónica basada en protocolos abiertos como MQTT o AMQP, es clave para fomentar la autonomía de los microservicios.


Consumers
JAVA
Node.js
Python
Microservice X
Microservice Y
Microservice Z
Microservice P
Microservice Q
Microservice R
Microservice S

“Con MSA, la lógica del negocio y la lógica de la comunicación de red, están dispersas a través de servicios independientes”.

Microservicios, clave para el desarrollo de aplicaciones y servicios.

Cuantos más microservicios se desplieguen y operen, más se deberá tener en cuenta el espacio de recursos individuales de cada microservicio.

Aquellos aspectos que deben tenerse en consideración en el momento de implantar una MSA son:

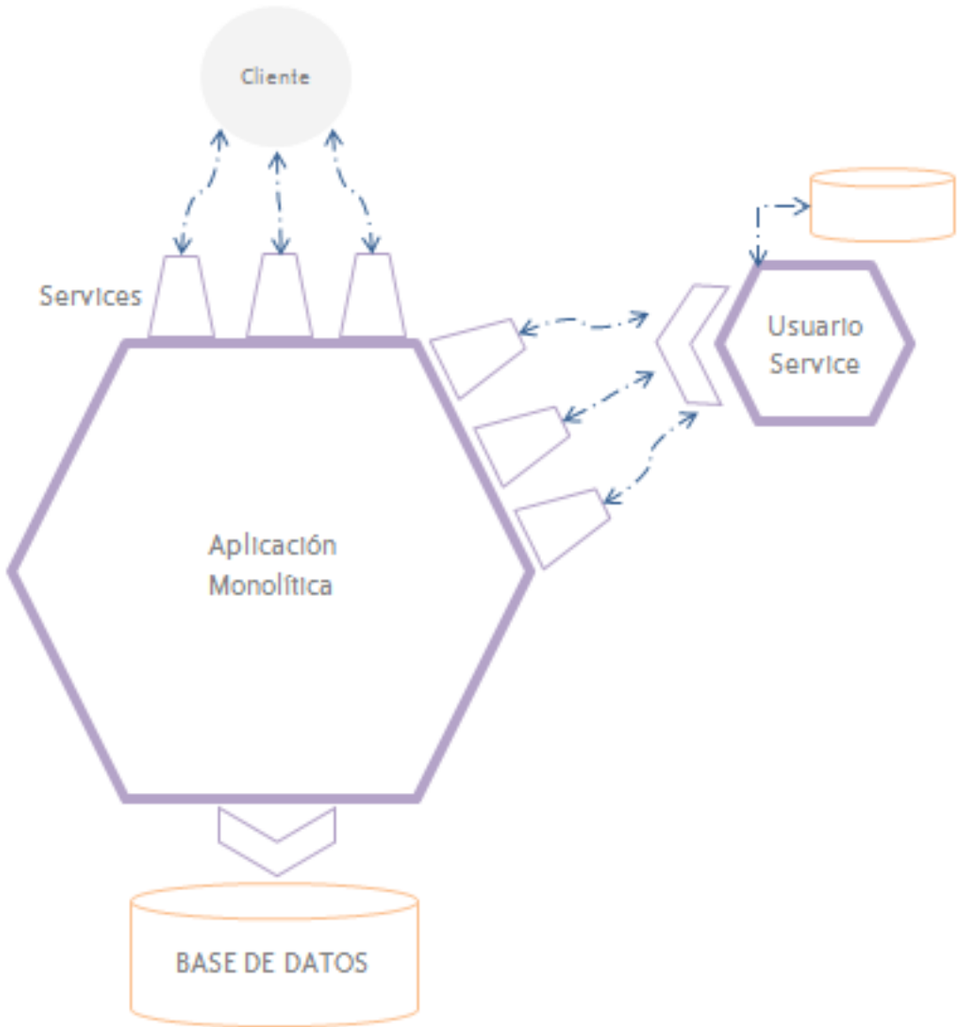
- Estrategias de implementación de microservicios.
- El papel del diseño dirigido por dominio en la arquitectura de microservicios.
- Asegurar microservicios (Oauth2, OpenIDI Connect y JWT).
- Granularidad de microservicios y su organización.
- Varios registros de servicios, ETCD.

Microservicios, clave para el desarrollo de aplicaciones y servicios.

También existe la posibilidad de crear microservicios compuestos mediante la integración de API web / SaaS, sistemas heredados y microservicios. El servicio API o Edge Service también es un microservicio de integración con algunas capacidades de puerta de enlace API.

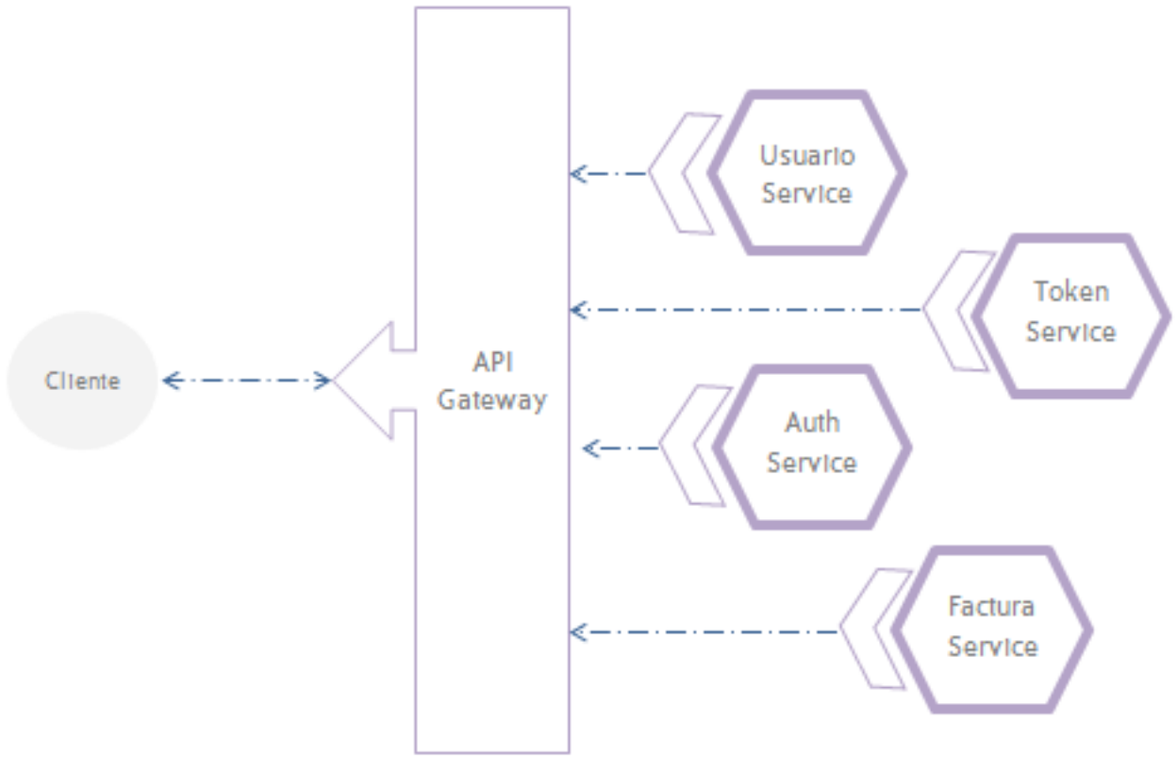
Tres aspectos importantes: la posibilidad de eliminar ESB centralizado con la arquitectura de microservicios, la importancia de crear servicios compuestos utilizando tecnología dedicada y la integración de microservicios con Ballerina.

Aplicaciones monolíticas

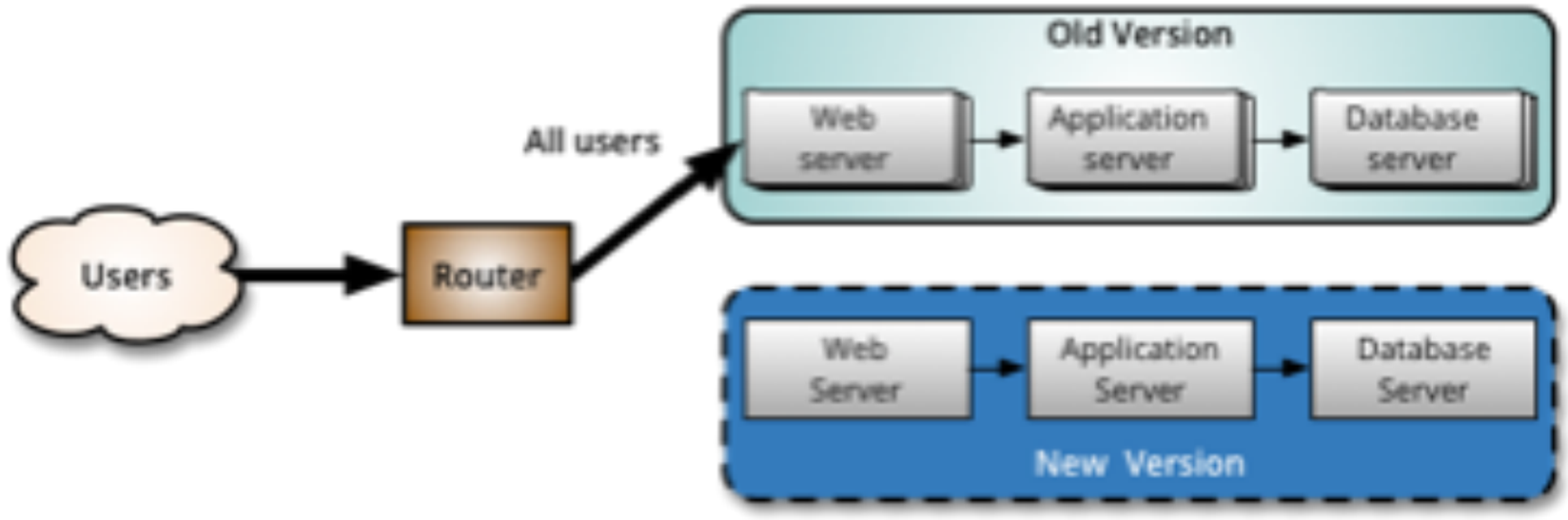


MAINT

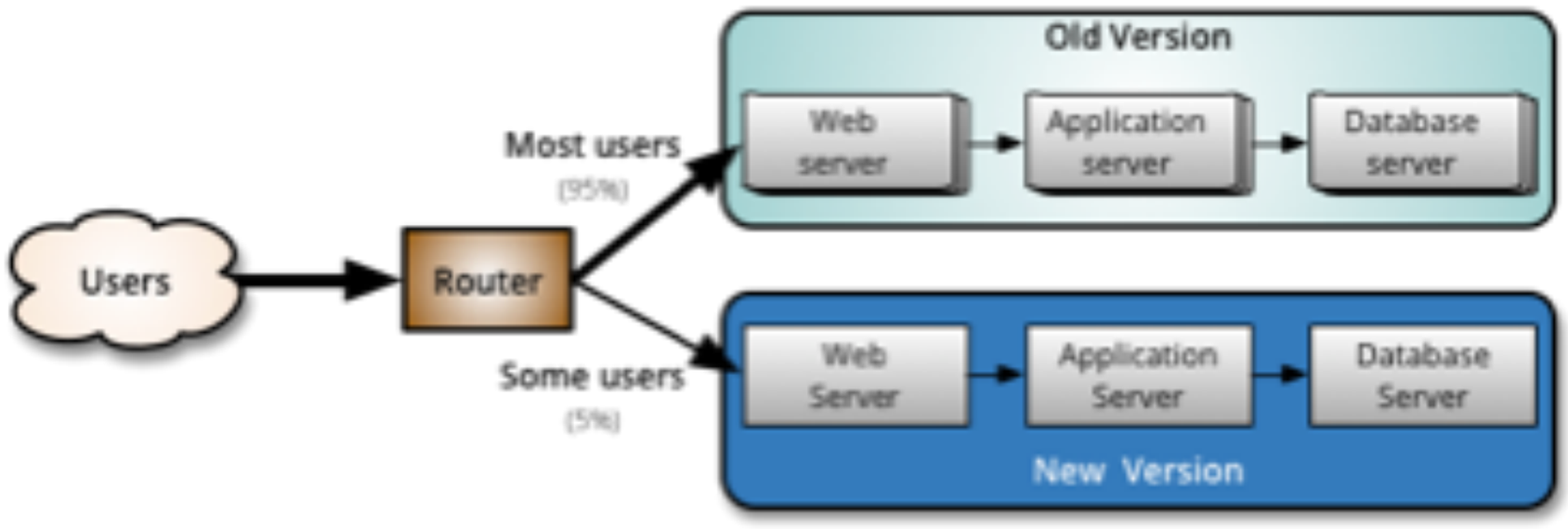
Microservices y un API Gateway



Arquitecturas Canary: Microservicios y ESB trabajando juntos

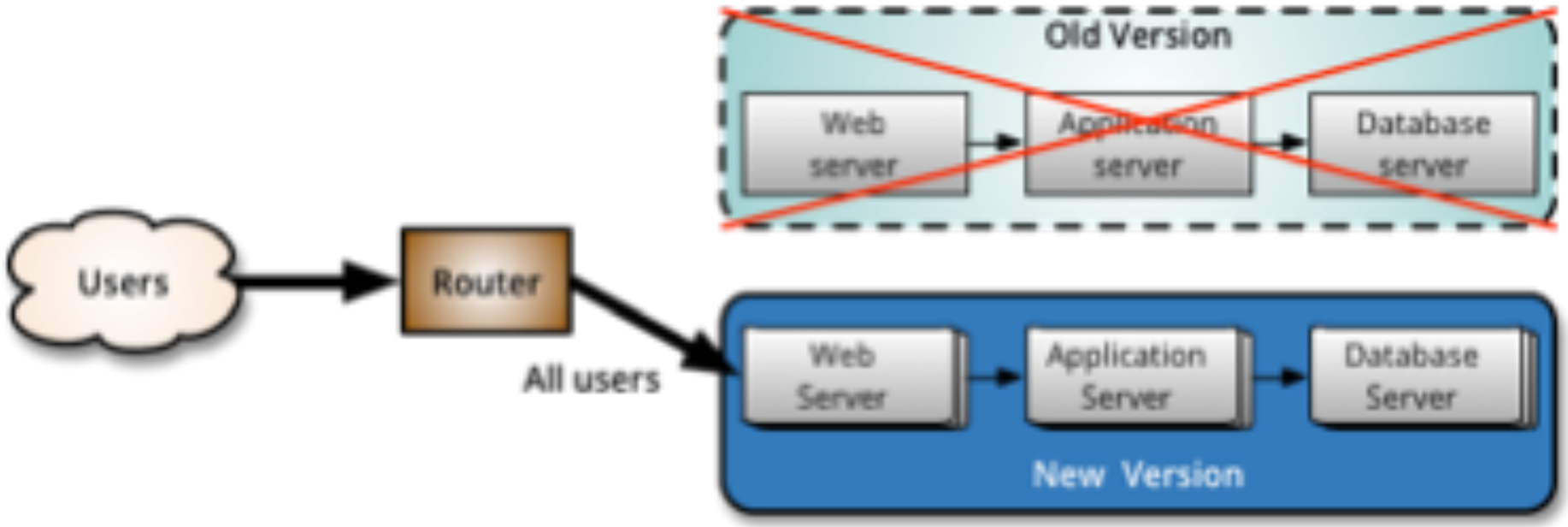


Arquitecturas Canary: Microservicios y ESB trabajando juntos



MAINT

Arquitecturas Canary: Microservicios y ESB trabajando juntos



Arquitecturas de microservicios: Accounting y Transaccionalidad

MAINT®

API Rest

REST Web Services

REST (Representational State Transfer), es un estilo de arquitectura de software dirigido a sistemas hipermedias distribuidos como lo es la web.

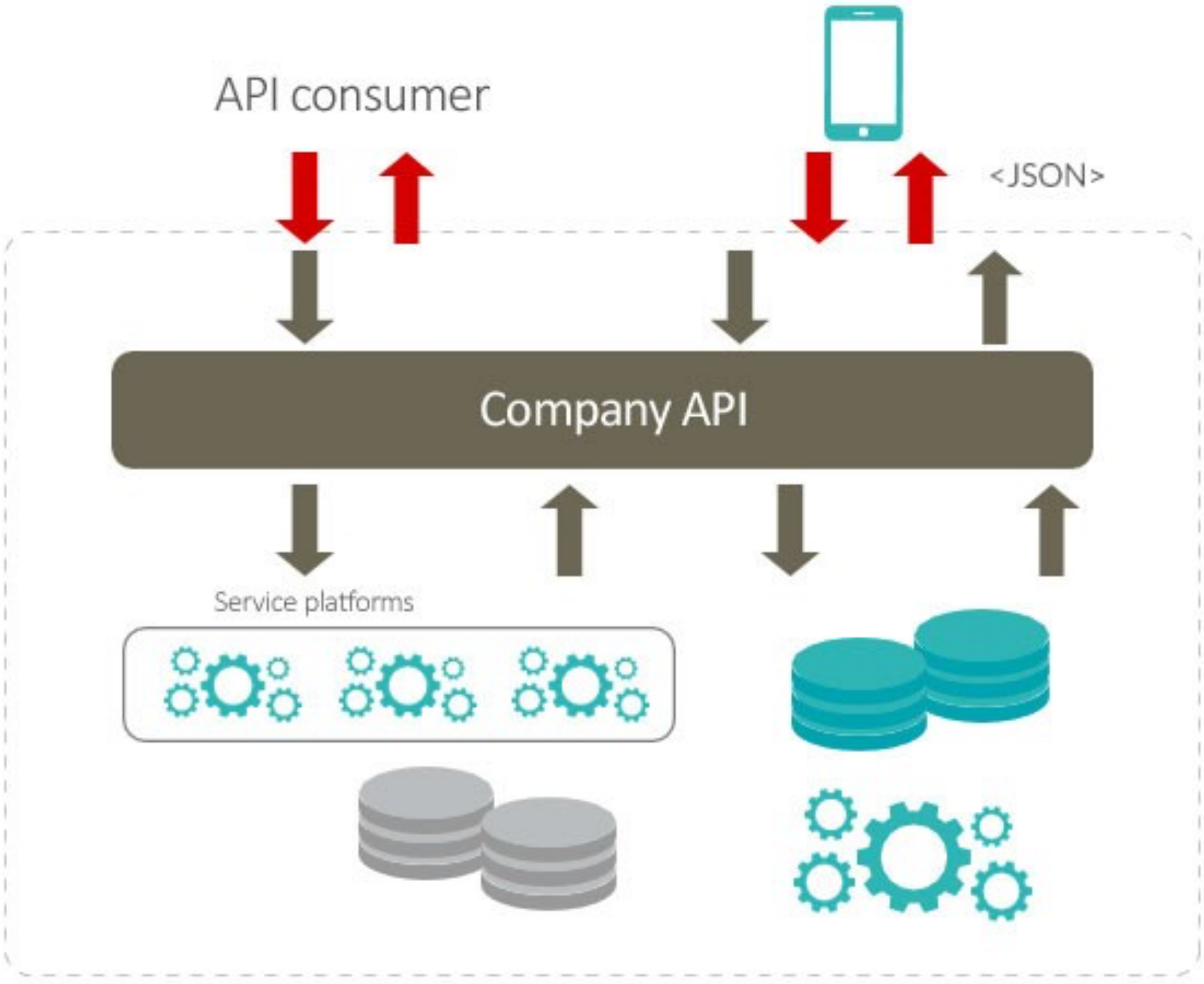
Este término se refiere específicamente a una colección de principios para el diseño de arquitecturas en red.

Existen varios proyectos que pueden verse beneficiados de una arquitectura REST. Concretamente aquellos en los que la idea principal está en la manera en la que se hacen las peticiones al servidor desde el cliente, basados en el recurso de interés.

REST Web Services

Es importante que la arquitectura REST cumpla con 6 principios:

- Cliente – Servidor
- Interface Uniforme
- Capaces de almacenar en caché
- No manejan estado
- Sistemas en capas
- Código baja demanda


API consumer
<JSON>
Company API
Service platforms

Laboratorios API Rest

Swagger

¿Qué pasa cuando dos personas no hablan un mismo idioma?

Las APIs no comparten un idioma común en demasiadas ocasiones y Swagger y Swagger UI trabajan para solucionarlo.

Las APIs nos permiten conectar y compartir datos, algo esencial para las empresas más vanguardistas.

MAINT.

Swagger

Pero surge un gran problema, las APIs no cuentan con un estándar de diseño común y ni si quiera existen unos parámetros comunes para documentarlas.

Es decir, no solo no hablan el mismo idioma, sino que no hay un diccionario que nos ayude a descifrarlas.

¿Tiene sentido en una plataforma cuyo objetivo es que colaboren diversas personas para conseguir los mejores desarrollos?

MAINT

“Una API pierde su sentido sino es accesible y si no tenemos una documentación que nos ayude a entenderla.”

Swagger

Una API definitivamente pierde su sentido sino es accesible y si no tenemos una documentación que nos ayude a entenderla.

¿Cómo van a ser accesibles?

Uno de los mayores problemas de las APIs es que en muchos casos, la documentación que les acompaña es inútil.

Swagger nace con la intención de solucionar este problema. Su objetivo es estandarizar el vocabulario que utilizan las APIs. Es el diccionario API.

Swagger

Cuando hablamos de Swagger nos referimos a una serie de reglas, especificaciones y herramientas que nos ayudan a documentar nuestras APIs.

De esta manera, podemos realizar documentación que sea realmente útil para las personas que la necesitan. Swagger nos ayuda a crear documentación que todo el mundo entienda.

MAINT

“Swagger es una serie de reglas, especificaciones y herramientas que nos ayudan a documentar nuestras APIs.”

Swagger

Aunque existen otras plataformas que ofrecen frameworks diferentes, Swagger es la más extendida.

¿Por qué? Porque ofrece otros beneficios.

El principal de ellos es que todo el mundo entiende Swagger, o casi todos, seas desarrollador o tengas pocos conocimientos sobre desarrollo podrás entenderla gracias al Swagger UI. Incluso las máquinas pueden leerlo, convirtiéndose en otra ventaja muy atractiva.

Swagger

Con Swagger la documentación puede utilizarse directamente para automatizar procesos dependientes de APIs.

Otra de las ventajas, es que lo podemos encontrar integrado en herramientas populares y potentes.

Por último, tampoco nos podemos olvidar de las herramientas que ofrece.

Swagger UI – La interfaz de usuario de Swagger

Swagger UI es una de las herramientas atractivas de la plataforma.

Para que una documentación sea útil necesitaremos que sea navegable y que esté perfectamente organizada para un fácil acceso. Por esta razón, realizar una buena documentación puede ser realmente tedioso y consumir mucho tiempo a los desarrolladores.

MAINT.

“Utilizando el Swagger UI para exponer la documentación de nuestra API, podemos ahorrarnos mucho tiempo.”

Swagger UI – La interfaz de usuario de Swagger

Con el Swagger UI podremos organizar nuestros métodos e incluso poner los ejemplos que necesitemos.

Swagger UI utiliza un documento JSON o YAML existente y lo hace interactivo.

Crea una plataforma que ordena cada uno de nuestros métodos (get, put, post, delete) y categoriza nuestras operaciones. Cada uno de los métodos es expandible, y en ellos podemos encontrar un listado completo de los parámetros con sus respectivos ejemplos. Incluso podemos probar valores de llamada.

MAINT

Swagger Petstore

This is a sample server Petstore server. You can find out more about Swagger at http://swagger.io or on irc.freenode.net, #swagger. For this sample, you can use the api key `special-key` to test the authorization filters.

Find out more about Swagger

http://swagger.io
Contact the developer
Apache 2.0

pet : Everything about your Pets

Show/Hide | List Operations | Expand Operations

POST /pet — Add a new pet to the store

Parameters

Parameter	Value	Description	Parameter Type	Data Type
body	(required)	**Pet object that needs to be added to the store**	body	Model | Example Value

Parameter content type: application/json

```
{
  "id": 0,
  "category": {
    "id": 0,
    "name": "string"
  },
  "name": "doggie",
  "photoUrls": [
    "string"
  ],
  "tags": [
```

Response Messages

HTTP Status Code	Reason	Response Model	Headers
405	Invalid input		

Try it out!

PUT /pet — Update an existing pet

GET /pet/findByStatus — Finds Pets by status

GET ~~/pet/findByTags~~ — Finds Pets by tags

DELETE /pet/{petId} — Deletes a pet

GET /pet/{petId} — Find pet by ID

POST /pet/{petId} — Updates a pet in the store with form data

POST /pet/{petId}/uploadImage — uploads an image

store : Access to Petstore orders

Show/Hide | List Operations | Expand Operations

user : Operations about user

Show/Hide | List Operations | Expand Operations

[BASE URL: /v2 , API VERSION: 1.0.0]

VALID {...}

Swagger UI – La interfaz de usuario de Swagger

Esta no es la única herramienta útil con la que cuenta Swagger, “Editor” es otra herramienta muy interesantes que nos ayudará a identificar los errores de nuestras documentación en YAML o JSON.

Simplemente subiendo nuestra documentación a la plataforma la comparará con las especificaciones.

Swagger Editor, no solo identificará los errores que hemos cometido, sino que nos planteará sugerencias y nos dará alternativas para que nuestra documentación sea perfecta.

MAINT.

“Una API bien documentada significa que es accesible por otros desarrolladores, y haciendo nuestras APIs más accesibles podremos mejorarlas.”

Swagger UI – La interfaz de usuario de Swagger

Sin duda, Swagger, Swagger UI y todas sus herramientas, son capaces de hacer el trabajo de nuestros desarrolladores mucho más fácil a la hora de documentar nuestras APIs.

Swagger, es un framework tan establecido que hasta está integrado en herramientas tan populares para la gestión de APIs como CA API Manager.

Gracias a su popularidad y sus resultados, Swagger hace posible que cada API tenga el mejor diccionario para entenderla.

Laboratorio: Protegiendo tu aplicación de fallos externos con el patrón Circuit Breaker

Laboratorio: Introducción a la gestión de Servicios Web con Spring Cloud y Netflix OSS

Laboratorio: SPRING BOOT REST

MAINT®

Laboratorio: Creación de la API REST de Java utilizando Spring Boot y MongoDB