



# 101: Introducción al mundo de las APIs

01 **INFOGRAFÍA**  
¿Qué es una API?

02 Características que  
debe tener una API

03 API REST: ventajas en  
desarrollo de proyectos

04 Herramientas para  
desarrollar APIs

05 Claves en la evolución  
de las APIs



# 01

INFOGRAFÍA

## Qué es una **API**

Es un conjunto de funciones o procedimientos utilizados por los programas informáticos para acceder a los servicios del sistema operativo, bibliotecas de software, u otros sistemas.

[View on website](#)



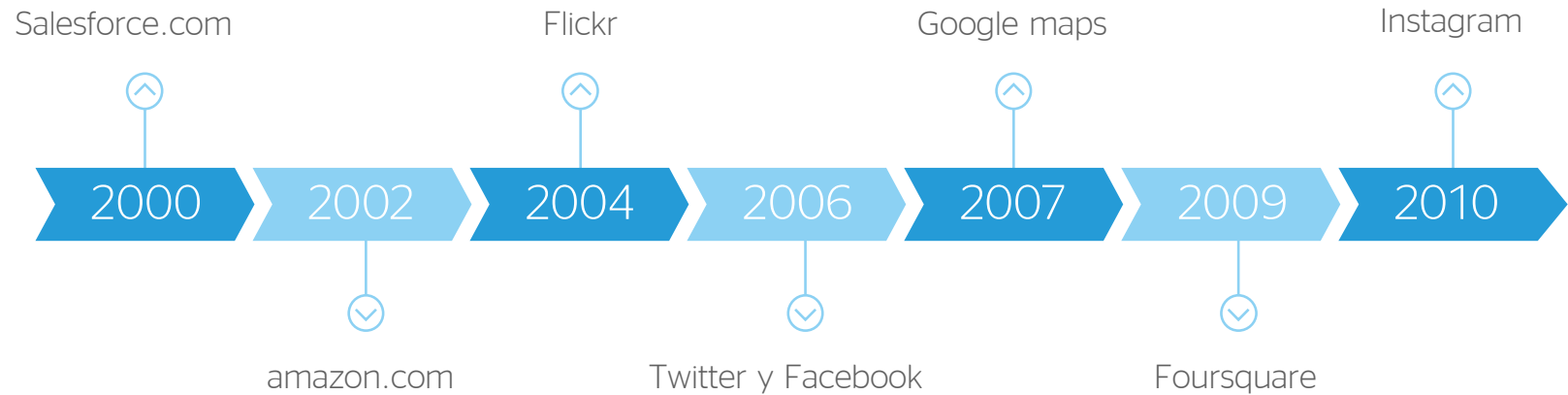
Hoy en día casi todo jefe de proyecto, diseñador o desarrollador habla de la 'economía API' como el nuevo mundo. Las interfaces de desarrollo de aplicaciones no son algo novedoso, pero sí es cierto que su universalización se está produciendo en estos días. 2016 y 2017 serán los años en los que la utilización de APIs se generalice entre la mayoría de empresas que busca aumentar y diversificar sus canales de creación y de ingresos. No solo grandes corporaciones, también pymes y *startups*.

Igual que una interfaz de usuario permite la interacción y comunicación entre un *software* y una persona, una API (acrónimo de *Application Programming Interface*) facilita la relación entre dos aplicaciones para el **intercambio de mensajes o datos**. Un conjunto de funciones y

procedimientos que ofrece una biblioteca para que otro *software* la utilice como capa de abstracción, un espacio de acceso e intercambio de información adicional en la parte superior. Así una aplicación se sirve de la información de la otra sin dejar de ser independientes.

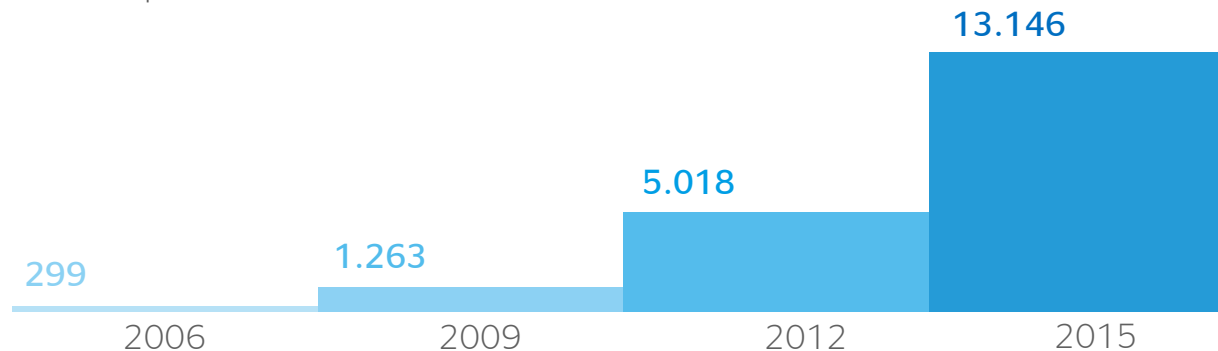
Cada API está diseñada en un lenguaje de programación concreto y con unas especificaciones distintas que la definen. Las APIs pueden incluir especificaciones para **estructuras de datos y rutinas, clases de objetos o variables**, a partir de las cuales se basa el uso de esa interfaz. Además, suele ser habitual que cada una de ellas disponga de documentación completa y eficaz (un conjunto de tutoriales, manuales y reglas de buenas prácticas para esa interfaz de programación).

## Cronología de la historia de las APIs

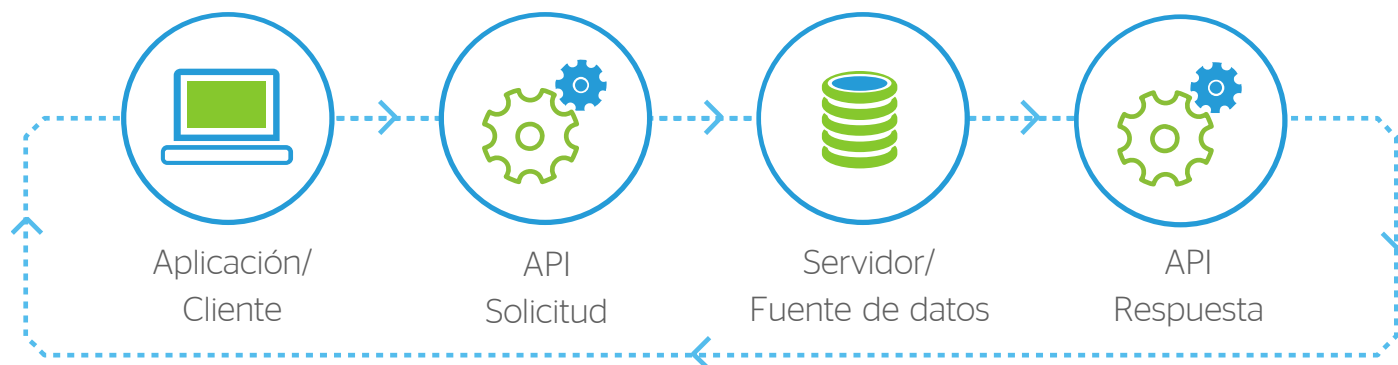


## Crecimiento

(Número de APIs publicadas)



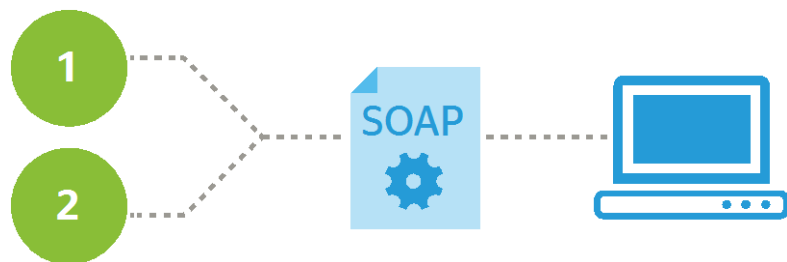
## Cómo funciona



## Usos



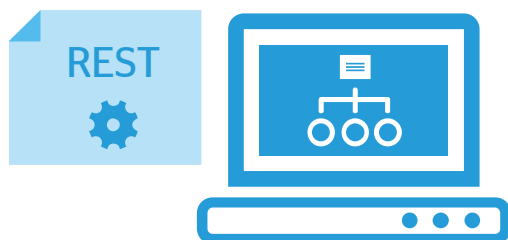
## Tipos de APIs



### SOAP

(Simple Object Access Protocol)

Es un protocolo estándar que define cómo dos objetos, en diferentes procesos, pueden comunicarse por medio de intercambio de datos XML.



### REST

(Transferencia de Estado Representacional)

Es una forma sencilla de enviar y recibir datos entre el cliente y el servidor y que no dispone de muchos estándares. Puede enviar y recibir datos como JSON, XML o incluso texto sin formato.

## Industria

### Social

Twitter, Facebook,  
Instagram, Flickr

### Finanzas

PayPal, BBVA,  
Yahoo Finance

### Empresas

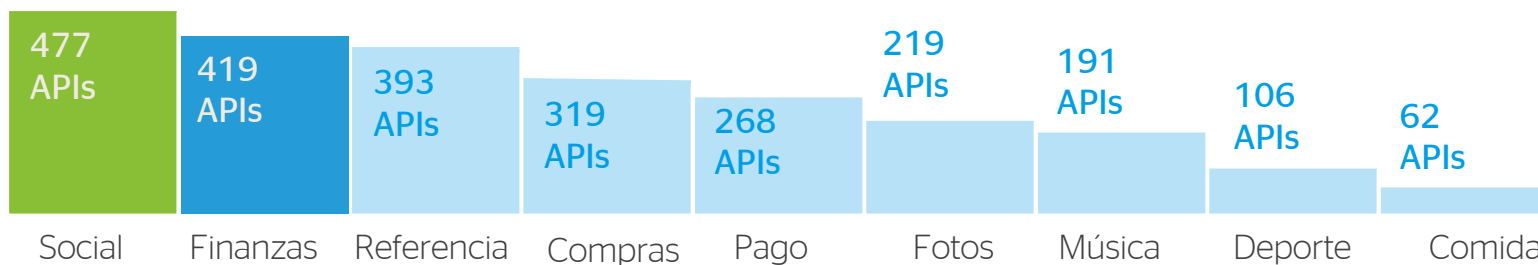
### Localización

Google Maps,  
Foursquare

### E-commerce

Amazon, eBay

### Administración Pública





# 02 Características que debe tener una API

La comunidad de desarrolladores siempre tiene presente cuatro características fundamentales para puntuar la calidad de una API: debe ser útil y fácil de aprender; estable frente a las mejoras; segura, y ofrecer una buena documentación.

[View on website](#)







No es ninguna novedad destacar la importancia que [tienen las APIs en el desarrollo de negocio de las empresas.](#)

Tanto es así que se creó una denominación *ad hoc* para ello: **economía API**. De ahí que impulsar una cultura de buenas prácticas en el diseño de interfaces de programación de aplicaciones sea una obligación para compañías y desarrolladores. Es recomendable crear productos útiles, reutilizables y abiertos.

Como casi todos los productos de desarrollo, el código y el uso correcto del lenguaje de programación es un tema especialmente complejo. En muchas ocasiones son necesarios parches para mejorar el funcionamiento y eso provoca que, a medida que la API soluciona problemas en primer término, también se convierte en **un producto más sucio, más complejo de usar, menos práctico.**

## Una API fácil de usar y de aprender

Si una API no es fácil de usar y su adopción por un desarrollador no es intuitiva, la API no cumplirá con su cometido: captar clientes y expandir la influencia de una empresa más allá de las cuatro paredes de su oficina. No tiene sentido desarrollar una API que no es un tentáculo, una extensión de los valores, del talento para dar servicio y generar ingresos. En ese camino, la primera meta es que **el desarrollador llegue lo antes posible a una implementación básica de la API.**

- La API es un instrumento, no una meta en sí misma: es [preferible utilizar formatos conocidos y habituales como JSON](#) que inventar la rueda. O seguir los pasos más racionales a la hora de [desarrollar una API REST o SOAP](#).
- Dar soporte para la corrección de errores: los desarrolladores pueden informar de fallos y posibles mejoras en una API si se genera un espacio para *feedback*. Así **se puede perfeccionar la API y, al mismo tiempo, generar comunidad.**



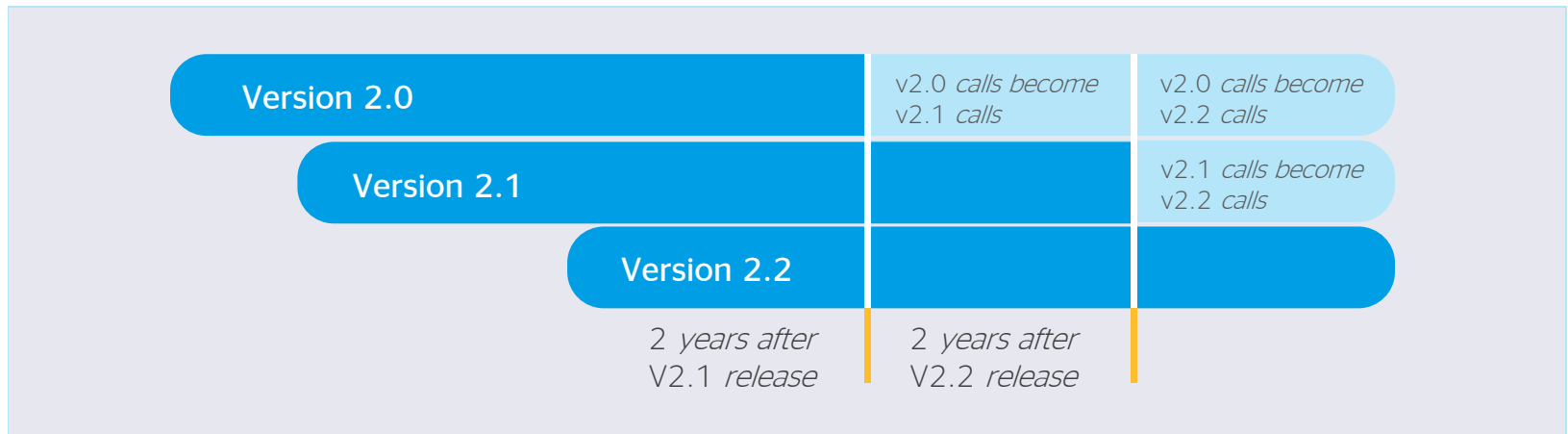
## Si no es estable, es un problema

Una de las peores pesadillas para un desarrollador es una API que cambia continuamente. Las mejoras están bien, pero a veces se corre el riesgo de que vayan en contra de la propia estabilidad del servicio. Sobre todo si no existe la posibilidad de mantener sin cambios las versiones previas de la propia API. Eso provocó que en 2011, la interfaz de desarrollo de aplicaciones de Facebook fuera calificada [como la peor API del mercado para la comunidad de desarrolladores](#). Una de las razones: los **cambios continuos en la herramienta sin previo aviso**.

Para evitar esos problemas, puede considerarse como una buena práctica en la mayoría de ocasiones el uso de un control de versiones en la API. Normalmente mediante la creación de distintas URLs por cada proceso de incorporación de

nuevas características y su impacto en las aplicaciones de terceros. Eso genera también un listado de versiones con fecha y novedades en cada caso. Sucede, por ejemplo, con servicios como [Microsoft Azure](#) y [Amazon Web Services](#).

En la misma línea ha trabajado Facebook desde 2011, introduciendo un sistema de control de versiones más razonable para la comunidad de desarrolladores. En ese procedimiento, cada desarrollador sabe de antemano que habrá cambios en la API cada dos años, con fechas ya publicadas: la versión 2.3 se publicó el 25 de marzo de 2015 y está prevista su expiración en agosto de 2017. El cuadro siguiente es un ejemplo de cómo es [el proceso de versiones de las APIs y los SDKs dentro de Facebook](#):



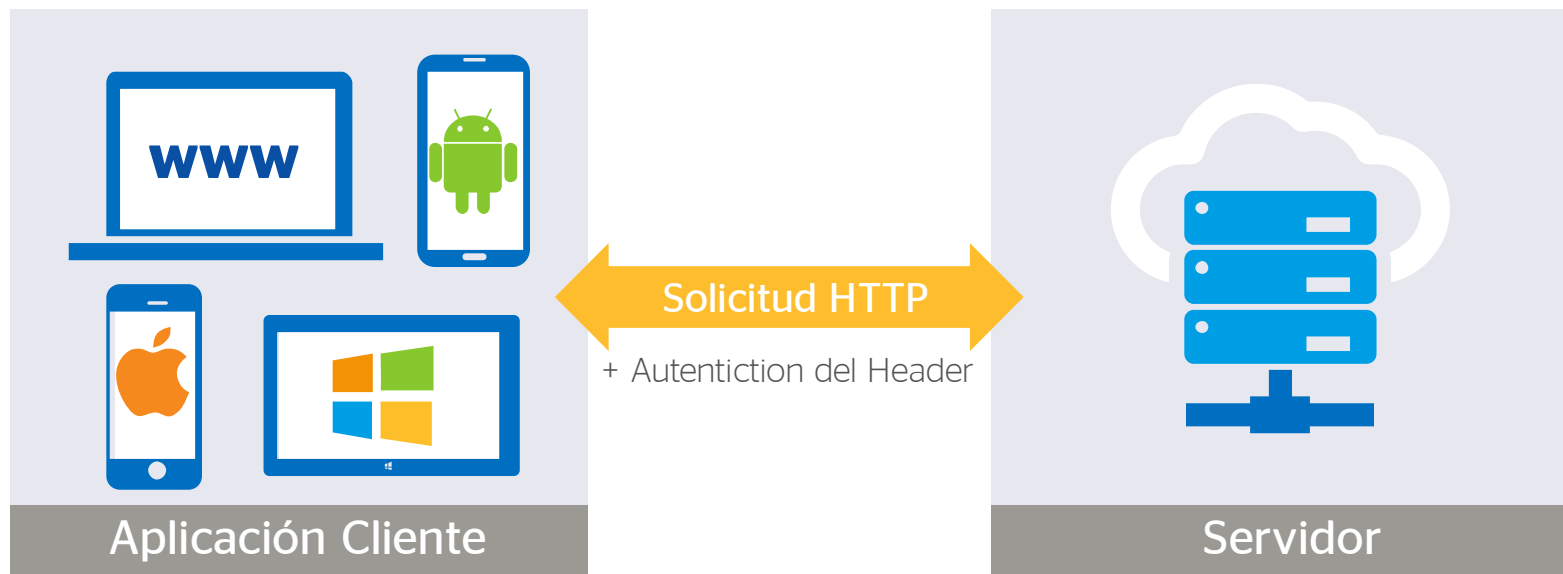
## No hay nada como algo seguro

El tema de la seguridad siempre es complejo. Hay que analizar [el papel que juegan los protocolos OAuth y OAuth2](#) en la necesidad de desarrollar APIs seguras en su uso por terceros. Los procesos de autenticación deben ofrecer las máximas garantías sin dificultar en exceso el acceso al servicio por parte de otras compañías.

OAuth 2 es un protocolo relativamente sencillo de implementar con [bibliotecas en numerosos lenguajes de programación](#): PHP, Java, Python, Scala, Objective C, Swift, Ruby, JavaScript, Node.js o .NET. Es cuestión de entrar y escoger la que se desee. Lo que permite el protocolo OAuth 2 basado en la asignación de *tokens* de acceso seguro para la identificación de cada usuario es evitar ataques CSRF ([Cross-Site Request Forgery](#) -

Falsificación de petición en sitios cruzados), un tipo de vulneración cada vez más frecuente basado en el uso de *cookies* para la identificación de usuarios. Añadir temporalidad a los *tokens* de acceso añade aún más solidez.

Las APIs que basan su seguridad en un protocolo de *token* de acceso lo que hacen es identificar a cada cliente que hace una petición HTTP mediante ese *token* que, previamente, ha sido almacenado en el lado del cliente (navegador) con JavaScript. [En este artículo de Carlos Azaustre se explica a la perfección el proceso de token.](#)



## La documentación de la API es clave

Cuando se desarrolla una aplicación, el público objetivo es el usuario medio. El uso del producto debe ser claro, sencillo, intuitivo...

Cuando se diseña una API, el público objetivo es un programador, un profesional con conocimientos técnicos. Aunque eso es así, la calidad de la interfaz final no solo depende del producto, también de la documentación que explica cómo hacer un mejor uso de él.

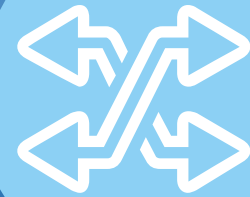
Aunque ese usuario final tenga conocimientos técnicos, una buena documentación es clave.

Una de las APIs mejor valoradas hoy en día es la de [Stripe](#), una pasarela de pagos online que compite en el mercado con PayPal o Braintree,

por poner dos ejemplos. La razón fundamental por la que Stripe tiene una gran acogida entre la comunidad de desarrolladores es su interfaz de desarrollo de aplicaciones. Y uno de sus puntos fuertes es la documentación, que se basa en dos pilares esenciales:

- Cada método de la API se documenta en varios idiomas y se utiliza un lenguaje sencillo, claro, sin excesiva jerga técnica. Es totalmente accesible.
- No solo contiene información sobre especificaciones de la API, sino también tutoriales prácticos bien documentados sobre cómo afrontar cada trabajo.





# 03 API REST

## qué es y cuáles son sus ventajas en el desarrollo de proyectos

El lanzamiento del nuevo sistema REST como protocolo de intercambio y manipulación de datos en los servicios de internet cambió por completo el desarrollo de *software* a partir de 2000. Ya casi toda empresa o aplicación dispone de una API REST para la creación de negocio.

[View on website](#)



REST cambió por completo la ingeniería de software a partir del 2000. Este nuevo enfoque de desarrollo de proyectos y servicios web fue definido por [Roy Fielding](#), el padre de la especificación HTTP y uno de los referentes internacionales en todo lo relacionado con la Arquitectura de Redes, en su disertación '[Architectural Styles and the Design of Network-based Software Architectures](#)'. En el campo de las APIs, REST (**Representational State Transfer-Transferencia de Estado Representacional**) es, a día de hoy, el alfa y omega del desarrollo de servicios de aplicaciones.

En la actualidad no existe proyecto o aplicación que no disponga de una API REST para la creación de servicios profesionales a partir de ese software. Twitter, YouTube, los sistemas de identificación con Facebook... hay cientos de empresas que

generan negocio gracias a REST y las APIs REST. Sin ellas, todo el crecimiento en horizontal sería prácticamente imposible. Esto es así porque REST es el estándar más lógico, eficiente y habitual en la creación de APIs para servicios de Internet.

Buscando una definición sencilla, REST es cualquier interfaz entre sistemas que use HTTPs para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON. Es una alternativa en auge a otros protocolos estándar de intercambio de datos como SOAP (*Simple Object Access Protocol*), que disponen de una gran capacidad pero también mucha complejidad. A veces es preferible **una solución más sencilla de manipulación de datos como REST**.



## Características de REST

- **Protocolo cliente/servidor sin estado:** cada petición HTTP contiene toda la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo para satisfacerla. Aunque esto es así, algunas aplicaciones HTTP incorporan memoria caché. Se configura lo que se conoce como **protocolo cliente-caché-servidor sin estado**: existe la posibilidad de definir algunas respuestas a peticiones HTTP concretas como *cacheables*, con el objetivo de que el cliente pueda ejecutar en un futuro **la misma respuesta para peticiones idénticas**. De todas formas, que exista la posibilidad no significa que sea lo más recomendable.
- **Los objetos en REST siempre se manipulan a partir de la URI.** Es la URI y ningún otro elemento el identificador único de cada recurso de ese sistema REST. La URI nos facilita acceder a la información para su modificación o borrado, o, por ejemplo, para compartir su ubicación exacta con terceros.






- **Interfaz uniforme:** para la transferencia de datos en un sistema REST, este aplica acciones concretas (POST, GET, PUT y DELETE) sobre los recursos, siempre y cuando estén identificados con una URI. Esto facilita la existencia de una interfaz uniforme que sistematiza el proceso con la información.
- **Sistema de capas:** arquitectura jerárquica entre los componentes. Cada una de estas capas lleva a cabo una funcionalidad dentro del sistema REST.
- **Uso de hipermedios:** hipermedia es un término acuñado por [Ted Nelson](#) en 1965 y que es una extensión del concepto de hipertexto. Ese concepto llevado al desarrollo de páginas web es lo que permite que el usuario pueda navegar por el conjunto de objetos a través de enlaces HTML. En el caso de una API REST, el concepto de hipermedia explica la capacidad de una interfaz de desarrollo de aplicaciones de proporcionar al cliente y al usuario los enlaces adecuados para ejecutar acciones concretas sobre los datos.

Para cualquier API REST es obligatorio disponer del principio HATEOAS (*Hypermedia As The Engine Of Application State* - Hipermedia Como Motor del Estado de la Aplicación) para ser una verdadera API REST. Este principio es el que define que cada vez que se hace una petición al servidor y éste devuelve una respuesta, parte de la información que contendrá serán los hipervínculos de navegación asociada a otros recursos del cliente.



[Devolución de una petición a una API REST según el principio HATEOAS](#) (enlaza a un tutorial explicativo del concepto de hipermedia en API REST con un ejemplo práctico de una petición a una base de datos de automóviles):

```
{
  "id": 78
  "nombre": "Juan",
  "apellido": "García",
  "coches": [
    {
      "coche":
      "http://miservidor/concesionario/api/v1/
clients/78/coche/1033"
    },
    {
      "coche":
      "http://miservidor/concesionario/api/v1/
clients/78/coche/3889"
    }
  ]
}
```

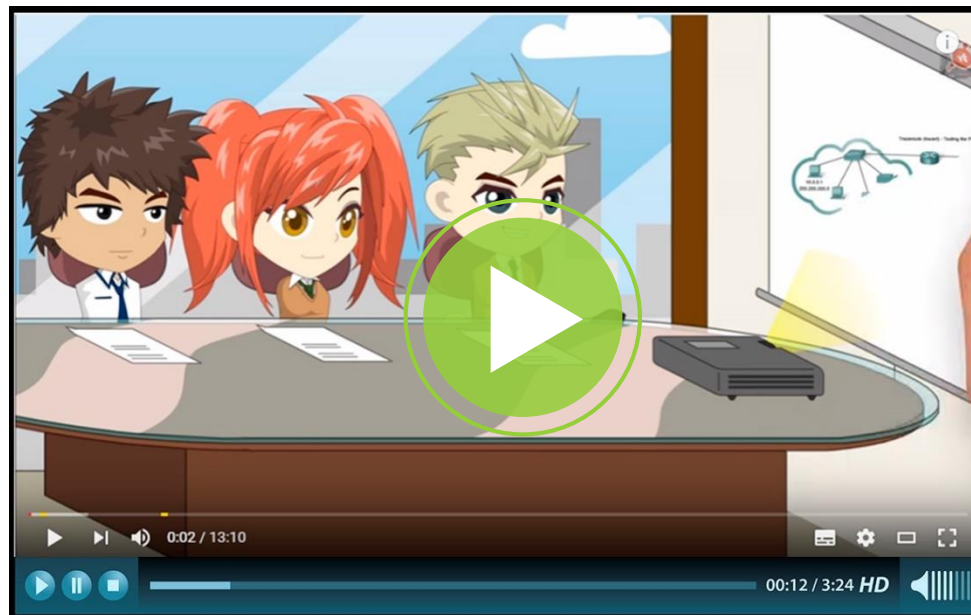


## Ventajas que ofrece REST para el desarrollo

1. **Separación entre el cliente y el servidor.** El protocolo REST separa totalmente la interfaz de usuario del servidor y el almacenamiento de datos. Eso tiene algunas ventajas cuando se hacen desarrollos. Por ejemplo, mejora la portabilidad de la interfaz a otro tipo de plataformas, aumenta la escalabilidad de los proyectos y permite que los distintos componentes de los desarrollos se puedan evolucionar de forma independiente.
2. **Visibilidad, fiabilidad y escalabilidad.** La separación entre cliente y servidor tiene una ventaja evidente y es que cualquier equipo de desarrollo puede escalar el producto sin excesivos problemas. Se puede migrar a otros servidores o realizar todo tipo de cambios en la base de datos, siempre y cuando los datos de cada una de las peticiones se envíen de forma correcta. Esta separación facilita tener en servidores distintos el *front* y el *back* y eso convierte a las aplicaciones en productos más flexibles a la hora de trabajar.



3. **La API REST siempre es independiente del tipo de plataformas o lenguajes.** Este tipo de API se adapta al tipo de sintaxis o plataformas con las que se estén trabajando, lo que ofrece una gran libertad a la hora de cambiar o probar nuevos entornos dentro del desarrollo. Con una API REST se pueden tener servidores PHP, Java, Python o Node.js. Lo único que es indispensable es que las respuestas a las peticiones se hagan siempre en el lenguaje de intercambio de información usado, normalmente XML o JSON.







# 04 Herramientas para desarrollar APIs

Restlet Studio, Swagger, API Blueprint, RAML o Apiary son algunas de las plataformas y herramientas con las que los equipos de desarrolladores diseñan, desarrollan, prueban mediante simulaciones automatizadas y documentan APIs en todo el mundo.

[View on website](#)



El mercado del desarrollo en los próximos años va estar protagonizado por una llave mágica que abre casi todas las puertas: las APIs. Para programar una API desde cero existen plataformas, herramientas y también lenguajes que permiten a los equipos de desarrolladores diseñar, desarrollar, probar y documentar sus propias APIs para facilitar la programación de productos a terceros y generar ingresos.

A día de hoy existen varios referentes importantes: [Restlet Studio](#), [Swagger](#), [API Blueprint](#), [RAML](#), [Mockable.io](#), [Loader.io](#), [BlazeMeter](#), [Apiary](#) e [InstantAPI](#). Existen más herramientas, pero estas son las más conocidas entre las comunidades de desarrolladores. A continuación se realiza un análisis de las características de algunas de ellas.



## Restlet: una Plataforma como Servicio para APIs

Restlet es un Entorno de Desarrollo Integrado (IDE) donde los programadores Java pueden diseñar sus APIs basadas en arquitectura REST (REST APIs). Con él se pueden desarrollar tanto aplicaciones por el lado del servidor como del cliente y es compatible con HTTP, HTTPS, XML o JSON. Este marco de desarrollo es de código abierto, [de descarga gratuita](#) y bajo licencia Apache. La herramienta dispone de diferentes planes de uso escalables: [uno gratuito y varios de pago](#). El primero permite el desarrollo de una API y el más caro no tiene limitaciones.

Restlet Studio está disponible para todas las plataformas ([Java SE/EE](#), [Google App Engine](#), [Google Web Toolkit](#), [OSGi](#) o [Android](#)). Todas las APIs desarrolladas con Restlet Studio permiten su integración con APISpark, la Plataforma como Servicio (PaaS) de Restlet para alojar y gestionar

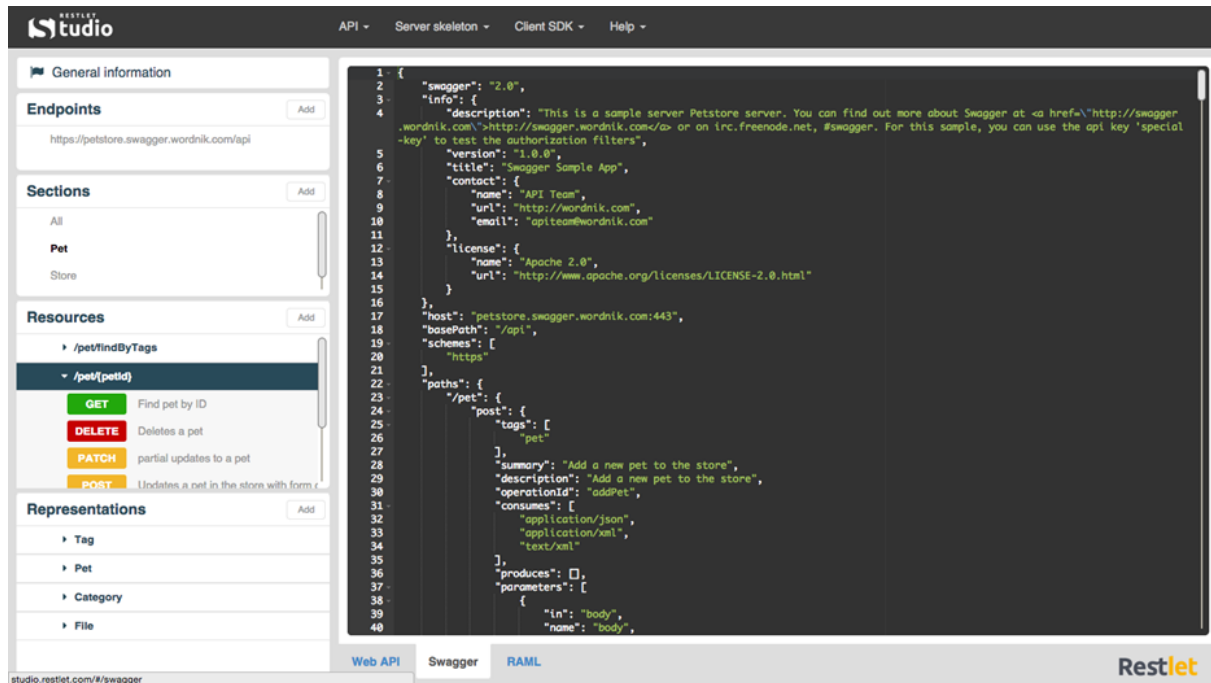
APIs por todo tipo de desarrolladores, en todo tipo de lenguajes como Java, PHP, Node.js o HTML, y marcos como AngularJS, [Jerome Louvel](#), CTO y fundador de Restlet, explicó en esta entrevista de noviembre de 2014 en InfoQ, [cuáles son las características fundamentales de APISpark](#).





Dos aspectos destacados en el desarrollo de APIs con Restlet Studio:

- El marco de desarrollo de Restlet **dispone de un conjunto de clases e interfaces a partir de las cuales se pueden diseñar APIs propias.**
- **Escalabilidad asegurada** independientemente del número de solicitudes.





## Swagger: el *framework* de APIs más popular

A día de hoy, [Swagger](#) va ya por la versión 2.0, es de código abierto y da servicio a plataformas o clientes tan importantes como Apigee, Getty Images, Microsoft o Paypal. Gracias a Swagger,

ellos han podido desarrollar sus propias APIs RESTful. **¿Qué es Swagger?** Se trata de un conjunto de herramientas para la programación de interfaces de desarrollo de aplicaciones en casi todos los lenguajes y entornos de desarrollo.

Entre las herramientas de Swagger destacan:

- [Swagger Editor](#): permite editar las especificaciones de una API en YAML (acrónimo de [YAML](#), *Ain't Another Markup Language*). YAML es un lenguaje de marcado ligero, un formato de datos inspirado en lenguajes como XML o Python que pone un mayor énfasis en los datos y no tanto en el marcado de los documentos. Este es el comando para ejecutarlo localmente en una máquina con Node.js:
- [Swagger UI](#): colección de activos HTML, JavaScript y CSS para generar de forma dinámica tanto documentación como una *sandbox* para cualquier API compatible con Swagger. Al no tener ninguna dependencia específica, la interfaz de usuario se puede alojar en cualquier servidor o en local.
- [Swagger Core](#): implementación en Java de Swagger. Un conjunto de librerías Java, de código abierto y disponibles en GitHub. [Aquí hay una gran cantidad de documentación específica para desarrolladores.](#)

```
git clone https://github.com/swagger-api/swagger-editor.git
cd swagger-editor
npm install
npm start
```



## API Blueprint: documentación para APIs

API Blueprint es un lenguaje basado en Markdown (un lenguaje de marcado ligero), que sirve fundamentalmente para documentar cualquier API de una forma sencilla. Lo realmente interesante de API Blueprint para los desarrolladores de APIs son las herramientas que funcionan de satélites de este lenguaje de marcado.

La más interesante es [Dredd](#) (en alusión al juez Dredd, personaje cinematográfico), que permite testear un servicio de *backend* a partir de la documentación de la API. Así es posible solucionar problemas de actualización de esa documentación. Soporta todo tipo de lenguajes como PHP, Python, Ruby, Perl, Node.js o Go.

Otra herramienta interesante integrada en Blueprint es [Drakov](#), que permite poner en marcha servicios de *mocks* para hacer *tests* con peticiones y respuestas a medida de la documentación de la API. Lo que viene siendo un banco de pruebas. [Este es un buen tutorial para empezar a trabajar con API Blueprint](#) (en inglés).



## RAML: gestión completa de APIs

RAML es el acrónimo de *RESTful API Modeling Language* (Lenguaje de Modelado de API RESTful). Su objetivo es facilitar la gestión del ciclo de vida de una API, desde el diseño y desarrollo hasta su utilización por terceros (testeo y documentación), poniendo el énfasis en el uso de un lenguaje fácil de interpretar por desarrolladores, y no solo por máquinas. La última versión de este lenguaje es la 1.0.

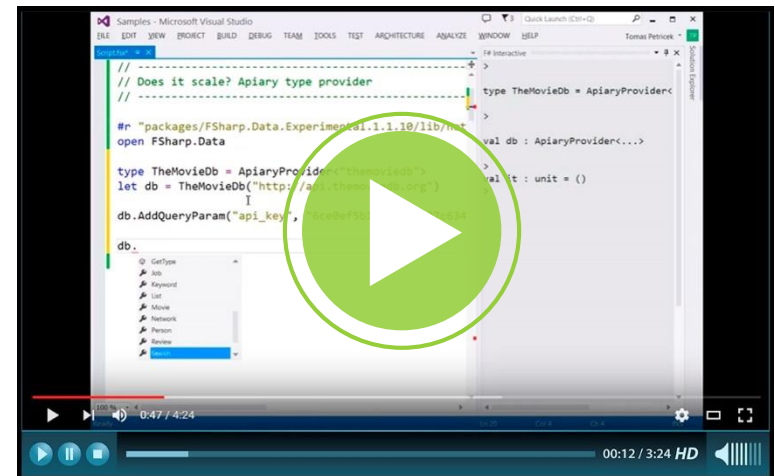


RAML permite el desarrollo de APIs en varias sintaxis: Node.js (JavaScript), Java, .NET y Python. Además, dispone de una gran variedad de herramientas para testear las interfaces de desarrollo de aplicaciones con RAML: [Abao](#), una herramienta de línea de comandos en Node.js para [probar la documentación de una API que esté escrita en RAML](#); Vigia; o [Postman](#), una extensión de Google Chrome muy utilizada entre los desarrolladores por su enorme sencillez y que sirve para testear una API a través de peticiones, ya sean GET, POST, PUT, PATCH o DELETE.

## Apiary: una API propia en 30 minutos

Suena algo arriesgado, pero la plataforma [Apiary](#) promete a los equipos de desarrollo todas las herramientas necesarias para tener una API propia en 30 minutos. Se ocupa de todo el ciclo de vida de una interfaz de desarrollo de aplicaciones: diseño y desarrollo, simulaciones automatizadas, validaciones, *proxies*, documentación... Apiary proporciona todo lo necesario para tener una API.

Apiary ofrece a los equipos *DevOps* los servidores de *mocks* para hacer pruebas y las simulaciones antes de empezar a codificar una API, algo similar a los *wireframes* en una interfaz de usuario. Un paso previo al del diseño es el de planificar para conocer las necesidades reales antes de destinar más recursos a desarrollo.







# 05 Claves

## en la evolución de las APIs

Las APIs se han convertido en la nueva herramienta indispensable para la mayoría de equipos de desarrollo y negocio (*DevOps*). A corto plazo se espera que las APIs se universalicen y ayuden a los bancos en sus procesos de transformación y en el impulso de sus microservicios.

[View on website](#)



Cuando se habla de 'economía API' no se exagera. Como casi toda irrupción en el mercado de la tecnología, las interfaces de desarrollo de aplicaciones han irrumpido con fuerza el panorama durante 2015, y este 2016 no se espera menos. No es algo nuevo, desde 2012 las APIs han ido creciendo como la espuma entre los equipos de creación de *software* y gestión de datos y la infraestructura de negocio. Sin ellas es complicado competir ahora y en el futuro.

Las APIs son la verdadera clave para crear productos y servicios y para generar ingresos y compromiso con la marca. A día de hoy nadie dentro del mercado duda de la importancia de las APIs dentro del negocio B2B ([\*business to business\*](#)), ya sea mediante interfaces públicas o mediante el desarrollo de APIs internas. Según el estudio '[Global API Market Forecast to 2017](#)', el negocio de las APIs generaba 113.000 millones de dólares en 2012, y se espera que crezca un 8% anual hasta 2017.

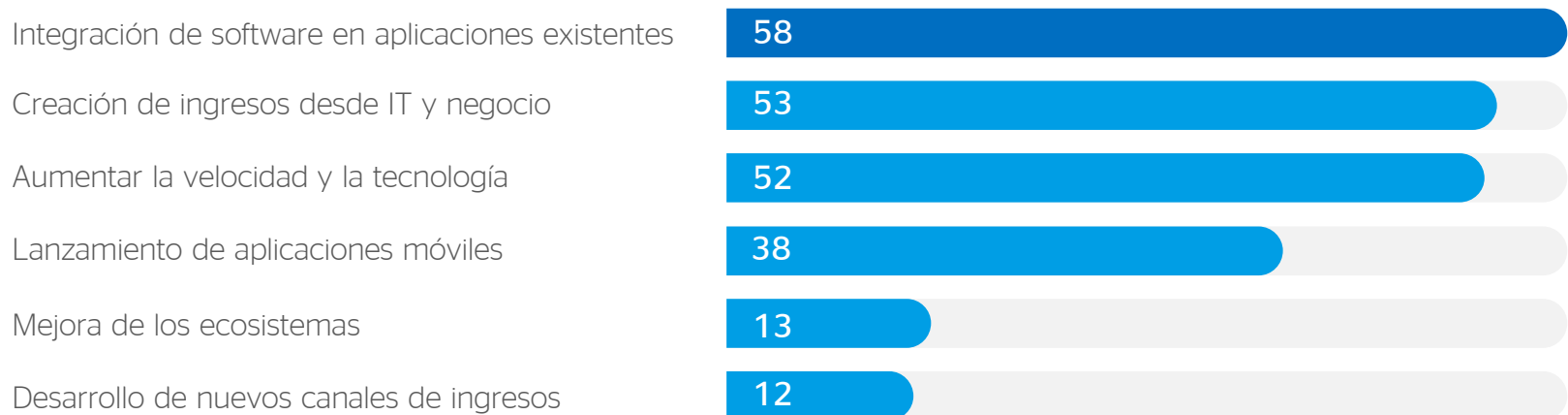




En el informe '2016 [Connectivity Benchmark Report](#)', la compañía Mulesoft aporta algunos datos sobre la 'economía API'. Una de las conclusiones más impactantes es que **el 91% de las empresas tiene una estrategia en torno a estas interfaces de desarrollo de aplicaciones**. En la mayoría de los casos, esta estrategia está centrada en tres campos de negocio: la integración de *software* con plataformas o aplicaciones ya existentes; la creación de ingresos desde IT y negocio; y el aumento de la velocidad para dotar al negocio de recursos de tecnología. En este gráfico se puede ver perfectamente cuál es la distribución de esfuerzos:

### Campos de negocio donde las empresas usan APIs

Reparto de uso (%) de interfaces de desarrollo de aplicaciones por parte de las compañías para generar ingresos.



**Fuente:** Mulesoft 2016 Connectivity Benchmark Report, Get the data

Otro de los datos que se desprende de este informe es lo que esperan ganar las empresas con el desarrollo de este tipo de estrategias relacionadas con APIs: el 33% de las organizaciones de más de 100.000 empleados **estima que ganará más de 10 millones de dólares usando estas interfaces de desarrollo de apps**. Además, el 26% de todos los encuestados tiene previsto generar de uno a cinco millones.

En el siguiente gráfico puedes consultar los resultados del informe elaborado por Mulesoft.

### Ingresos esperados por las empresas con el uso de APIs

Predicción de obtención de ingresos esperados por las compañías a través del uso de interfaces de desarrollo de aplicaciones o actividades relacionadas directamente con ellas.



Fuente: Mulesoft 2016 Connectivity Benchmark Report, Get the data

## Las APIs: un recurso para todos

En los próximos años ninguna compañía del mundo se podrá plantear que su futuro no dependa de una o varias APIs, propias o de terceros. La economía de productos y servicios digitales se ha convertido, sin ninguna duda, en la **economía de las APIs**, y esa es una realidad de la que nadie puede escapar. De hecho, acelerará el proceso de integración de muchos de los departamentos de las compañías más allá de los equipos técnicos. La universalización de las APIs permitirá cuatro aspectos:

- **Acelerar la digitalización de los procesos.**
- **Adoptar aplicaciones y plataformas de terceros.**
- **Analizar el negocio.**
- **Monetizar los productos y servicios de esos canales digitales.**

Todos estos cambios se deben al proceso de digitalización del consumo de información y servicios a través de numerosos dispositivos: no solo teléfonos inteligentes, también [wearables como relojes inteligentes o pulseras deportivas](#). Eso sin contar con [las perspectivas de negocio futuro del Internet de las cosas](#). Para aprovechar estas oportunidades será indispensable **un proceso de democratización del uso de las APIs**, facilitando su sencillez y accesibilidad.



## Banca abierta basada en APIs

Las APIs pueden convertirse en el gran elemento diferenciador de **la banca en el futuro**. Aquellas entidades que comprendan que las interfaces de desarrollo de aplicaciones son el instrumento perfecto para ganar dinero con terceros y pasar de la banca tradicional a ser Plataformas como Servicio (PaaS).

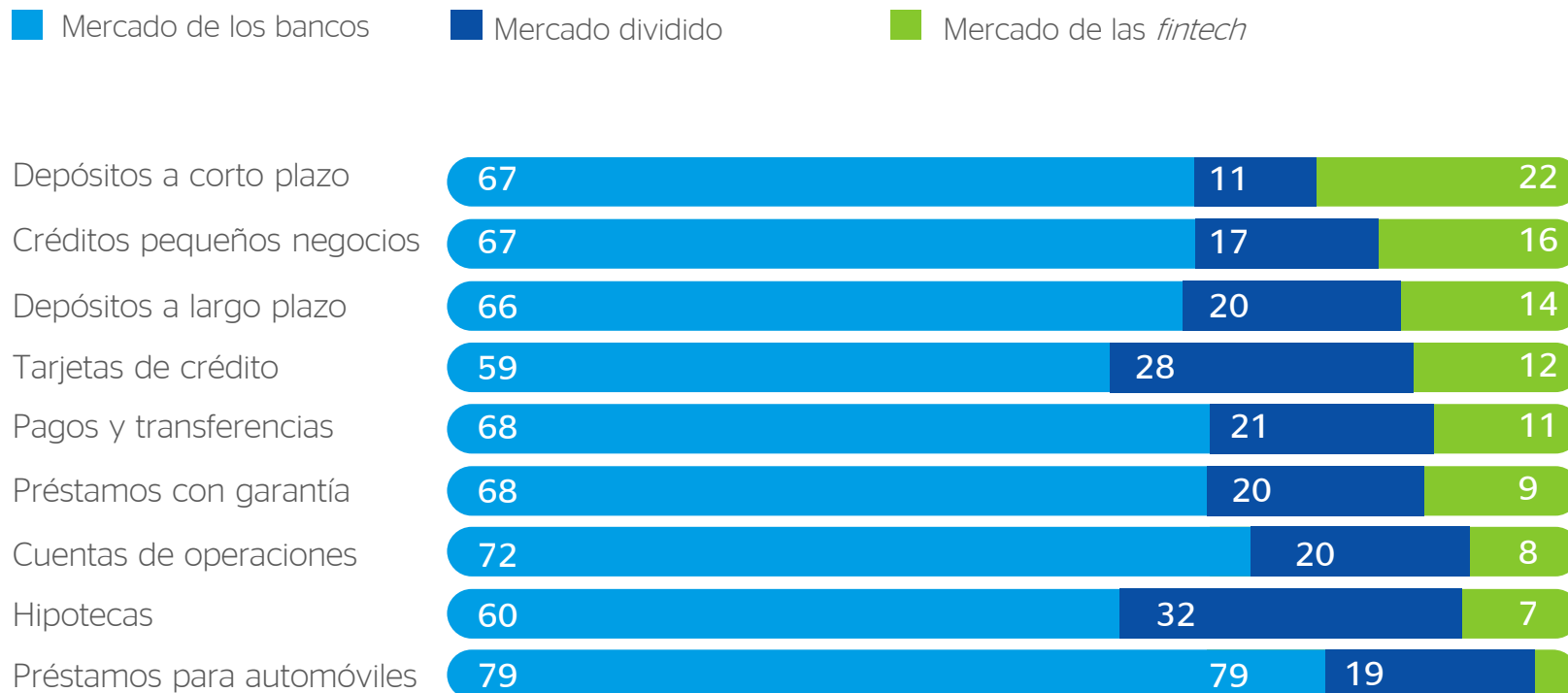
Para regular todo este contexto, la UE actualizó en octubre de 2015 [la Directiva sobre Servicios de Pago \(DSP2\)](#), que da una oportunidad a los bancos para hacer negocio como plataformas abiertas de servicios mediante el uso de APIs. La legislación obliga a los bancos a dar acceso a terceros, con permiso de sus clientes, a la información referente a cuentas y también a pagos. Entidades como BBVA o JP Morgan Chase han comprendido el

nuevo escenario y están en disposición de **convertir una obligación en una oportunidad: aprovechar ese activo de ingresos**. Esto es lo que se conoce, en inglés, como *“the ‘platformification’ of banking”*.

Sobre este proceso de transformación de la banca en plataformas de servicios, existen dos informes interesantes: [‘2016 Retail Banking Trends and Predictions’](#), un buen resumen [del cambio profundo en las entidades bancarias](#), con análisis de gran parte de los agentes que los están protagonizando; y un segundo informe, [‘The disruption of banking’](#), de la revista The Economist de finales de 2015. En este segundo informe se dan muchas claves sobre el nuevo escenario en el que los productos bancarios competirán con sus nuevos y poderosos rivales *fintech*.

### El balance futuro del mercado entre bancos y *fintech*’

Estimación de cómo será el equilibrio entre bancos y *fintech* por tipo de producto financiero dentro de cinco años.



Fuente: “The disruption of banking” (The Economist), Get the data

## APIs para arquitectura de aplicaciones basada en microservicios

La arquitectura de microservicios es el nuevo paradigma de los equipos *DevOps* y uno de los puntos que más va a evolucionar a partir de ahora. ¿En qué consiste? En el desarrollo de apps como un conjunto de microservicios que se ejecutan de forma independiente y se comunican a través de peticiones HTTP a las distintas APIs.

Esto tiene varias ventajas:

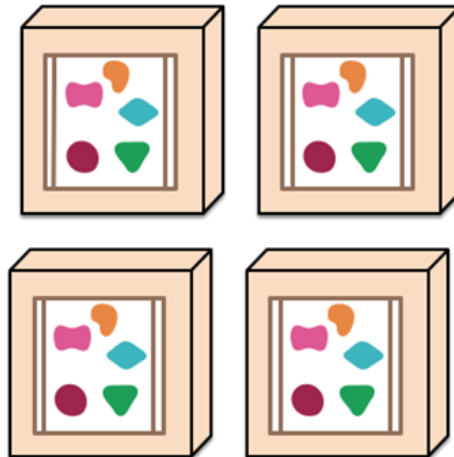
- Si algunos de los microservicios de una aplicación, ya sea la interfaz de usuario (*front-end*) o los servicios de *back-end*, tienen algún error, los equipos de desarrollo pueden solucionar los *bugs* por separado **sin tener que enviar toda la aplicación a producción con las mejoras en el desarrollo.**
- Algo similar sucede **con los procesos de escalabilidad, ya sean hacia arriba (crecimiento) o hacia abajo (decrecimiento).** En una aplicación basada en desarrollo de microservicios, el aumento de la capacidad es más sencilla y racional porque cada uno de ellos es un servicio independiente.
- **La lógica de negocio está separada y cada parte es independiente.**
- **Se rechazan los mecanismos de integración como ESB** ([\*Enterprise Service Bus\*](#) - Bus de Servicio Empresarial, componente de la Arquitectura Orientada a Servicios o SOA) y se apuesta decididamente por los mecanismos *lightweight* como las aplicaciones basadas en los llamados sistemas de colas.

Este artículo de **Martin Fowler** y **James Lewis** se cita con frecuencia para explicar [qué son y para qué sirven los microservicios](#) en el desarrollo de aplicaciones. En él explican cómo las APIs son clave en la flexibilidad y agilidad de los microservicios.

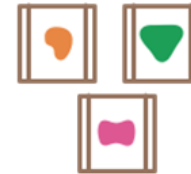
*A monolithic application puts all its functionality into a single process...*



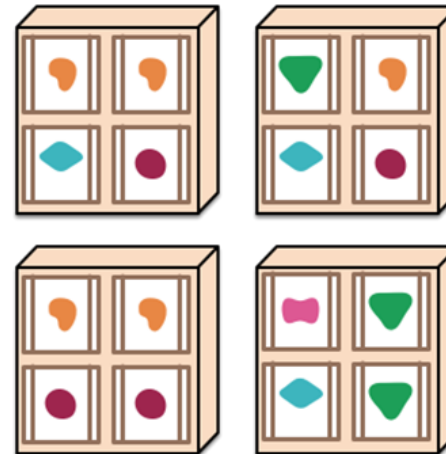
*... and scales by replicating the monolith on multiple servers*



*A microservices architecture puts each element of functionality into a separate service...*



*... and scales by distributing these services across servers, replicating as needed.*



Esta charla de Fowler es interesante también para entender los microservicios:



## Uso de las APIs: medir, medir, medir

Con las herramientas de monitorización de APIs, sin las cuales es complicado saber cuál es el rendimiento real de estas interfaces, los equipos de desarrollo pueden analizar el rendimiento y corregir a tiempo *bugs* o errores que interfieren en su desempeño. [Mashape Analytics](#), [Akana Envision](#) y [CA App Synthetic Monitor](#) son tres ejemplos.





## La seguridad en las APIs es una prioridad

El crecimiento de las APIs como instrumentos de desarrollo e integración de aplicaciones, sobre todo en el campo de desarrollo de dispositivos móviles, atrae a quienes se dedican a buscar debilidades. Hoy en día la mayoría de APIs basa su seguridad en un marco de creación de protocolos tan conocido como [OAuth2](#).

Grandes tecnológicas como Google, Facebook o Twitter, con uso capital de interfaces de desarrollo de aplicaciones para su negocio, utilizan este sistema de autorización y seguridad para terceros. [El sistema se basa en la creación de \*tokens\* de acceso](#) para que terceros desarrolladores puedan hacer uso de la API sin comprometer en exceso la herramienta. Un sistema de credenciales basado en usuario y contraseña es más delicado porque una quebrantación irregular de ese protocolo **provocaría cambiar el acceso para todos los clientes de esa API**.

BBVAOpen4U  
[www.bbvaopen4u.com](http://www.bbvaopen4u.com)



**SUSCRÍBETE**

para recibir la newsletter de  
BBVA Open4U: noticias,  
novedades, consejos, artículos...  
y los eventos más innovadores

Otros ebooks en BBVA Open4U



[Ebook: La revolución del fintech](#)



[Ebook: APIs e Internet de las cosas](#)



[Ebook: Open Source](#)

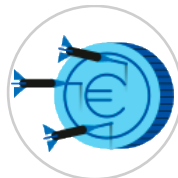
compartir



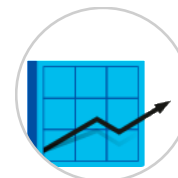
Prueba las APIs de BBVA en [www.bbvaapimarket.com](http://www.bbvaapimarket.com)



Identity



Accounts



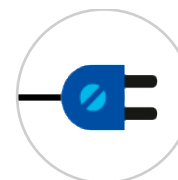
PayStats



Money Transfers



Cards



BBVA Connect

“Una empresa sin APIs es como un ordenador sin internet”

BRIAN KOLE