



101: Introduction to the world of APIs

01 **INFOGRAPHIC**
What is an API?

02 Features to have
an API

03 API REST: What is it, and
what are its advantages

04 Tools to develop APIs

05 Key Developments APIs



01 What is an API

INFOGRAPHIC

It is a set of functions or procedures used by computer programs to access operating system services, software libraries, or other systems.

[View on website](#)



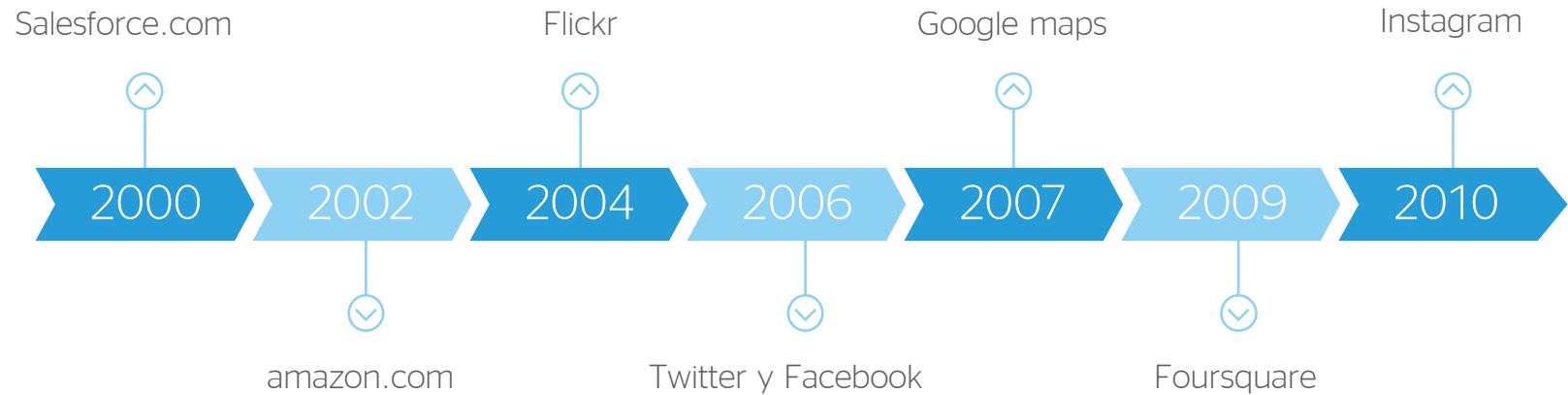
Nowadays, almost all project managers, designers or programmers are talking about the 'API economy' as though it were the new world. Application programming interfaces are nothing new, what is new is the universal use that is currently taking place. The years 2015, 2016 and 2017 will be when APIs will be put to use by most companies that intend to increase and diversify the channels they use for creativity and to generate revenue; not only major corporations, but also SMEs.

Just like a user interface allows interaction and communication between software and an individual, an API (acronym for Application Programming Interface) facilitates communication between two applications to **exchange messages**

or data. A set of functions and procedures that offers a library so other software can use it as an abstraction layer, an area to access and exchange additional information at the top. Thus, they both use each other's information without compromising their independence.

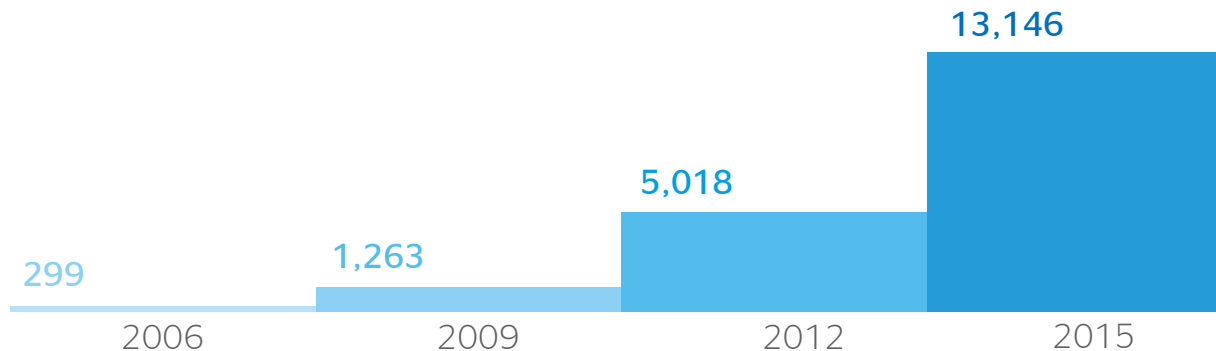
Each API is designed in a specific programming language and has several specifications that define it (APIs can include specifications for **data and routine structures, classes of objects or variables**, that are the cornerstones for the use of the interface). Moreover, complete and efficient documentation is usually available for each API (a set of guides, manuals or good practice rules).

Chronology of the history of APIs

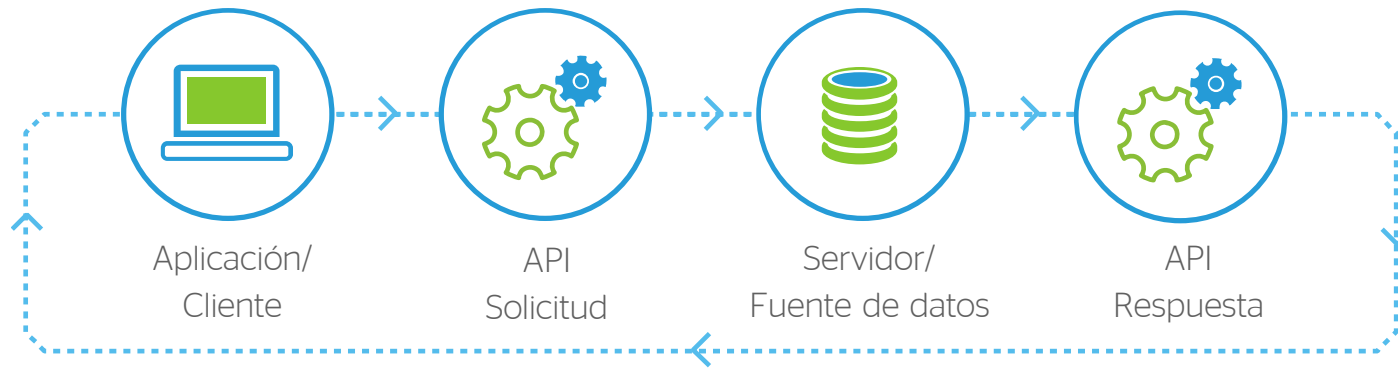


Growth

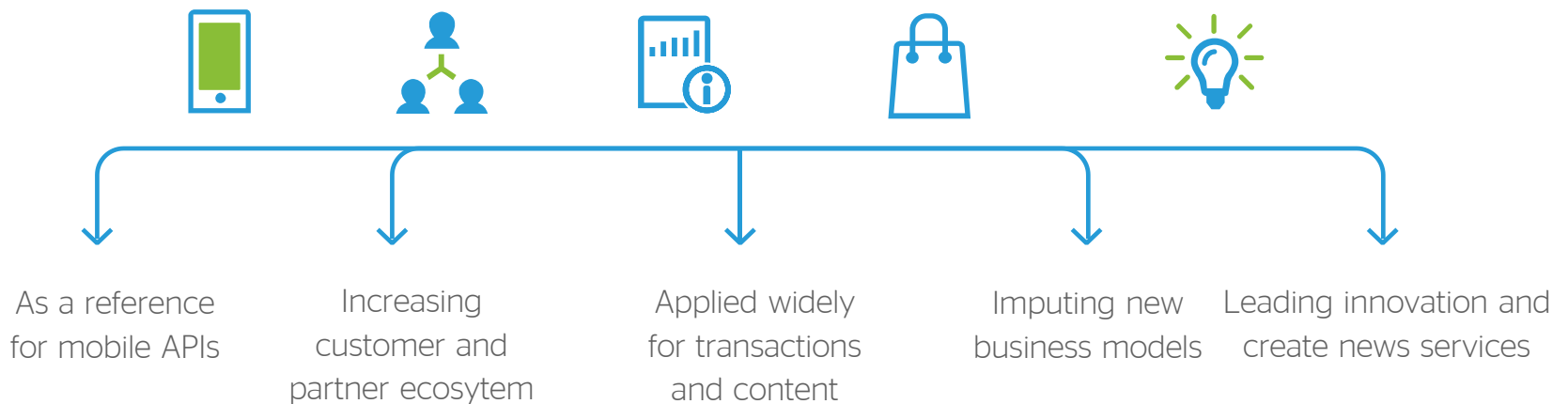
(Number of Public APIs)



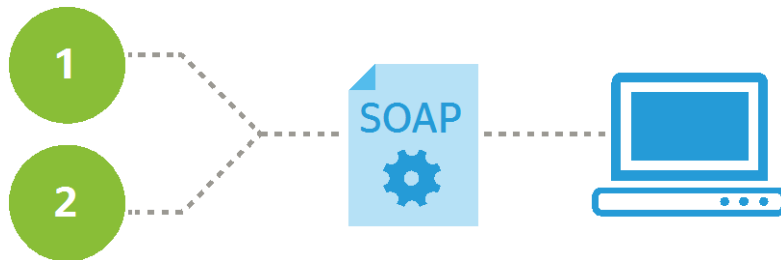
How it Works



Use



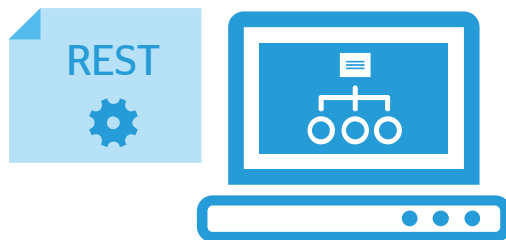
API types



SOAP

(Simple Object Access Protocol)

This is a standard protocol that defines how two objects in different processes can communicate through XML data exchange.



REST

(Representational State Transfer)

This is a simple way to send and receive data between the client and server and does not have many standards. It can send and received JSON, XML or plain text.

Sector

Social

Twitter, Facebook,
Instagram, Flickr

Financial

PayPal, BBVA,
Yahoo Finance

Companies

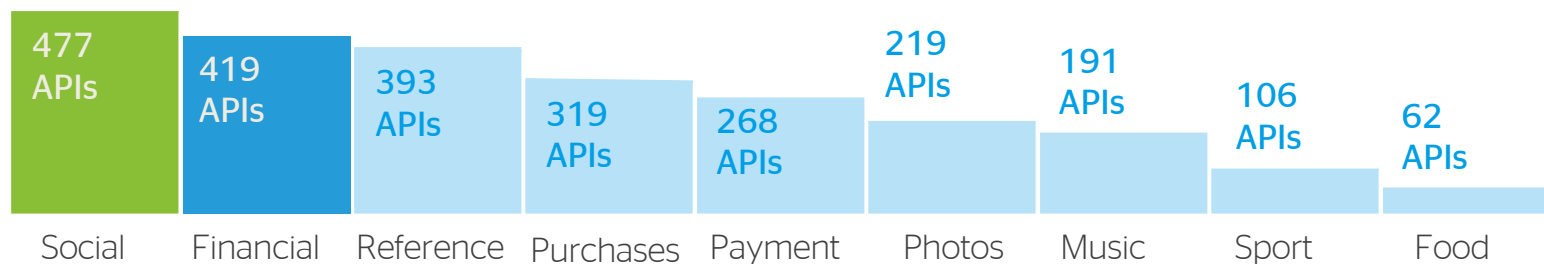
Location

Google Maps,
Foursquare

E-commerce

Amazon, eBay

Public Administration





02 Features to have an API

The developer community always looks at four essential features to rate the quality of an API: it must be useful and easy to understand, stable when making improvements, and it has to be secure and provide good documentation.

[View on website](#)





It is nothing new the enormous importance of [APIs for the development of a company's business](#). So much so that an ad hoc name has been created for it: **the API economy**. So promoting a culture of good practices in app design and programming interfaces is a must for companies and developers. The advice is to create useful, reusable and open products.

Like almost all development products, the code and the correct use of the programming language is a particularly complex issue. Patches are often needed to improve the way they work, and this means that while the API resolves problems up front, it also becomes a dirtier product that's more complex and less practical to use.

An API that's easy to use and learn

If an API isn't easy to use and if it can't be intuitively adopted by a developer, it won't be fulfilling its purpose -namely capturing customers and expanding the influence of a company beyond the four walls of the office. There's no sense in developing an API that doesn't act as a tentacle, an extension of the values and the talent, in order to provide service and generate income. Under this approach, the first goal is for the developer to be able to deploy a basic implementation of the API as soon as possible.

- The API is an instrument, it's not a goal in itself: it's preferable [to use familiar and habitual formats like JSON](#) than to reinvent the wheel, and to follow more logical steps when [developing an API REST or SOAP](#).
- Provide support to correct errors: developers can report faults and possible improvements in an API if there's a space available for feedback. This enables the API to be improved, and at the same time, generates community.



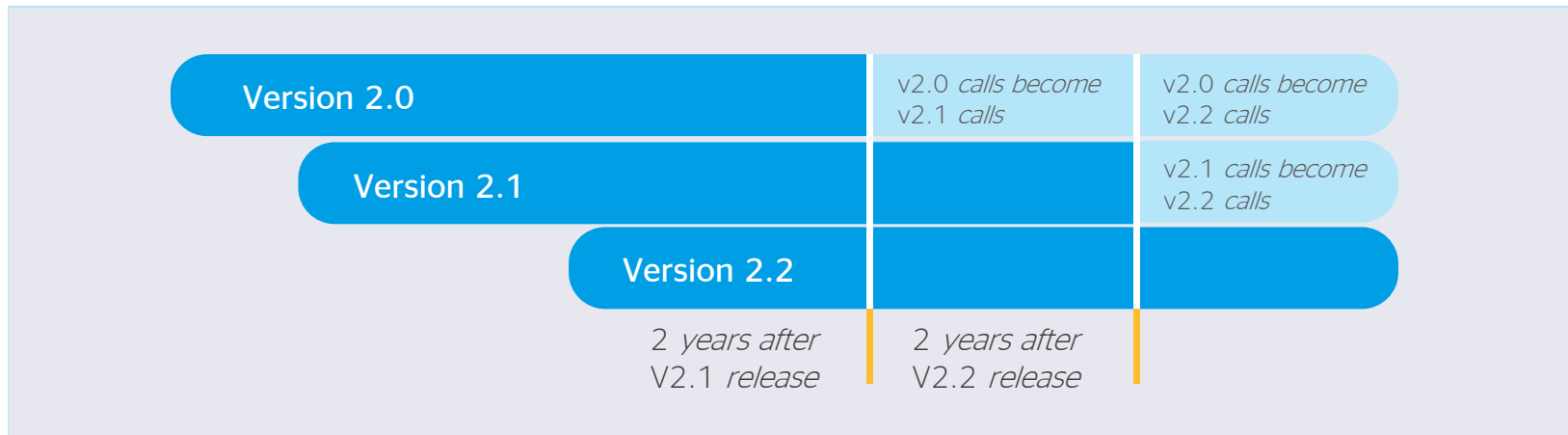
If it's not stable, we've got a problem

One of the worst nightmares for a developer is an API that's constantly changing. Improvements are good, but they sometimes run the risk of interfering with the actual stability of the service, and particularly if there is no possibility of maintaining the previous versions of the API unchanged. This was why in 2011 Facebook's app programming interface was rated [the worst on the market by the developer community](#). One of the reasons was the **continuous unannounced changes in the tool**.

To avoid these problems, it can be considered good practice to use a version control in the API on most occasions -usually by creating different URLs in each process of incorporating new features and their impact on third-party apps. This also generates a list of versions with the date and the

new developments in each case, something that occurs, for example such as [Microsoft Azure](#) and [Amazon Web Services](#).

Facebook has been working along the same lines since 2011, introducing a more reasonable version control system for the developer community. With this procedure, each developer knows beforehand that there will be changes in the API every two years, and the dates are already announced: version 2.3 was published on 25 March 2015 and is due to expire in August 2017. The following table is an example of [the version process for Facebook's APIs and SDKs](#):



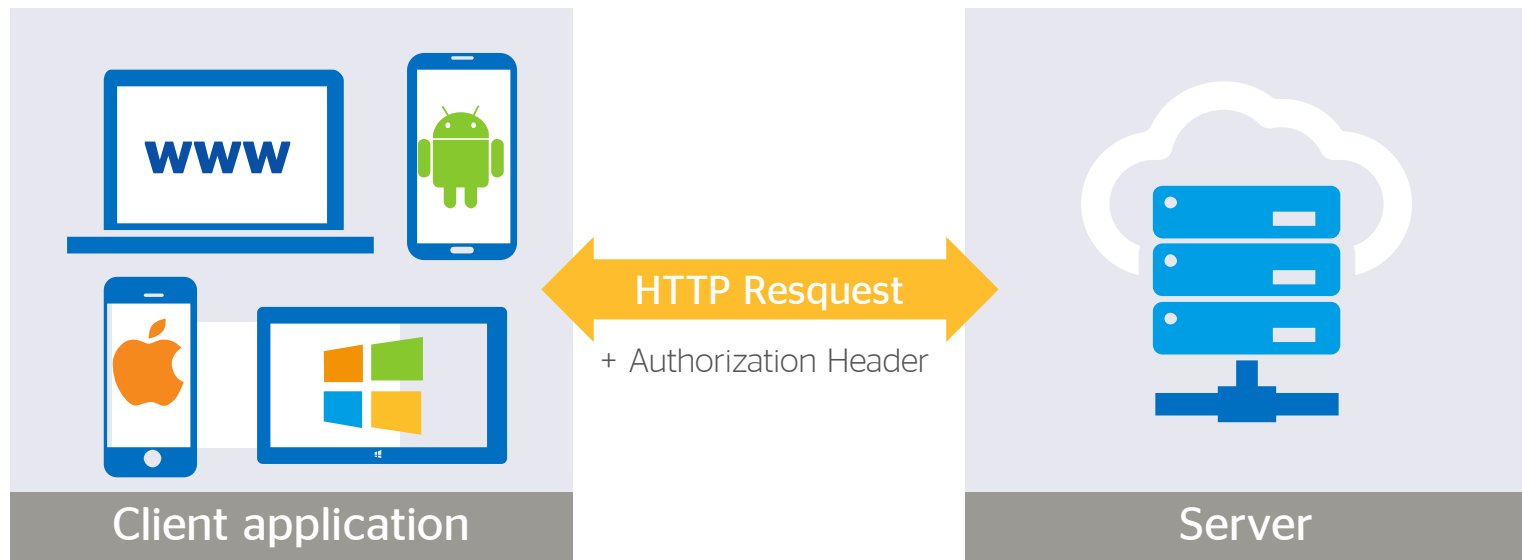
There's nothing like security

The issue of security is always complex. So much so that here at BBVAOpen4U we've already highlighted its importance and [the role of the OAuth and OAuth2 protocols](#) in the need to develop APIs that are secure when used by third parties. The authentication processes must offer the maximum guarantees without overly hindering access to the service by other companies.

OAuth 2 is a relatively simple protocol to implement with [libraries in numerous programming languages](#): PHP, Java, Python, Scala, Objective C, Swift, Ruby, JavaScript, Node.js and .NET. It's a matter of entering and choosing what you like. The OAuth 2 protocol works by assigning secure access tokens to identify each user, in order to avoid CSRF attacks ([Cross-Site Request Forgery](#)—falsification of the request in crossed sites), a type of violation that's

becoming ever more frequent based on the use of cookies for user identification. Temporarily adding the access tokens makes it even more solid.

APIs whose security is based on an access token protocol to identify each customer who makes a HTTP request through that token, which has previously been stored on the client side (navigator) with JavaScript. [This article by Carlos Azaustre explains the token process perfectly.](#)



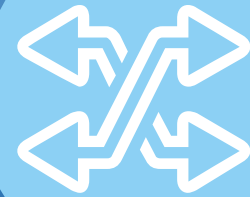
The API documentation is key

When someone develops an app, the target public is the average user. The product must be clear, simple and intuitive to use. When someone designs an API, however, the target public is a programmer, a professional with technical knowledge. Even so, the quality of the final interface not only depends on the product, but also on the documentation that explains how best to use it. Although the end user may have technical knowledge, it's still crucial to have good documentation.

One of the most highly rated APIs around today is [Stripe](#), an online payment gateway that's competing in the market with [PayPal](#) and [Braintree](#), to give just a couple of examples. The main reason Stripe is so popular with developers is its app development interface.

And one of its strong point is its documentation, which is based on two key pillars:

- Each API method is documented in several languages and uses plain and simple instructions without too much technical jargon, to make it totally accessible.
- It not only contains information on API specifications, but also well-documented practical tutorials on how to tackle each job.



03 REST API

What is it, and what are its advantages in project development?

The launch of the new REST system as a protocol for data exchange and management in Internet services completely revolutionized software development after 2000. Now almost every company or application has a REST API for business creation.

[View on website](#)



REST completely changed software engineering after 2000. This new approach to developing web projects and services was defined by [Roy Fielding](#), father of the HTTP specification and one of the leading international authorities on everything to do with Network Architecture, in his dissertation entitled "[Architectural Styles and the Design of Network-based Software Architectures](#)". In the field of APIs, REST (Representational State Transfer) is today the "be all and end all" in service app development.

Today there are no projects or applications that don't have a REST API for the creation of professional services based on this software. Twitter, YouTube, Facebook identification systems... hundreds of companies generate business thanks

to REST and REST APIs. Without them any horizontal growth would be practically impossible. This is because REST is the most logical, efficient and widespread standard in the creation of APIs for Internet services.

To give a simple definition, REST is any interface between systems using HTTP to obtain data and generate operations on those data in all possible formats, such as XML and JSON. This is an increasingly popular alternative to other standard data exchange protocols such as SOAP (Simple Object Access Protocol), which have a high capacity but are also very complex. Sometimes it's preferable to **use a simpler data-processing solution such as REST.**

REST's features

- Stateless client/server protocol: each HTTP contains all the necessary information to run it, which means that neither the client nor the server need to remember any previous state to satisfy it. Be that as it may, some HTTP applications incorporate a cache memory. This configures what is known as the stateless client-cache-server protocol: it is possible to define some of the responses to specific HTTP requests as cachable, so the client can run the same response for identical requests in the future. However, the fact that the option exists doesn't mean it is the most recommended.
- Objects in REST are always manipulated from the URL. It is the URL and no other element that is the sole identifier of each resource in this REST system. The URL allows us to access the information in order to change or delete it, or for example to share its exact location with third parties.
- There are four very important data transactions in any REST system and HTTP specification: POST (create), GET (read and consult), PUT (edit) and DELETE.).





- **Uniform interface:** to transfer data, the REST system applies specific actions (POST, GET, PUT and DELETE) on the resources, provided they are identified with a URI. This makes it easier to obtain a uniform interface that systematizes the process with the information.
- **Layer system:** hierarchical architecture between the components. Each layer has a functionality within the REST system.
- **Use of hypermedia:** hypermedia is a term coined by [Ted Nelson](#) in 1965 and is an extension of the concept of hypertext. This concept, taken to web page development, is what allows the user to browse the set of objects through HTML links. In the case of a REST API, the concept of hypermedia explains the capacity of an app development interface to provide the client and the user with the adequate links to run specific actions on the data.


All REST APIs must have the [HATEOAS](#) (Hypermedia As The Engine Of Application State) principle to be genuine. This principle is what ensures that each time a request is made to the server and it returns a response, part of the information it contains will be the browsing hyperlinks associated to other client resources.

[Return of a request to a REST API according to the HATEOAS](#) principle (links to an explanatory tutorial

on the hypermedia concepts in REST API with a practical example of a request to an automobile database):



```
{
  "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"
    }
  ]
}
```

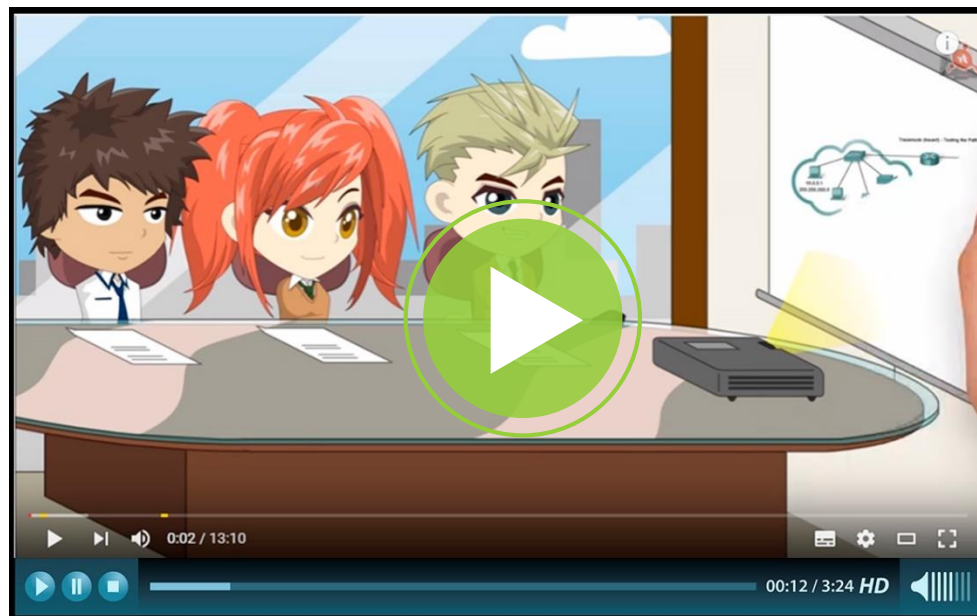


The advantages of REST for development

1. **Separation between the client and the server.** The REST protocol totally separates the user interface from the server and the data storage. This has some advantages when making developments. For example, it improves the portability of the interface to other types of platforms, it increases the scalability of the projects, and allows the different components of the developments to be evolved independently.
2. **Visibility, reliability and scalability.** The separation between client and server has one evident advantage, and that is that each development team can scale the product without too much problem. They can migrate to other servers or make all kinds of changes in the database, provided the data from each request is sent correctly. The separation makes it easier to have the front and the back on different servers, and this makes the apps more flexible to work with.



3. The REST API is always independent of the type of platform or languages. The REST API always adapts to the type of syntax or platforms being used, which gives considerable freedom when changing or testing new environments within the development. With a REST API you can have PHP, Java, Python or Node.js servers. The only thing is that it is indispensable that the responses to the requests should always take place in the language used for the information exchange, normally XML or JSON.





04 Tools to develop APIs

Restlet Studio, Swagger, API Blueprint, RAML and Apiary are some of the platforms and tools used by development teams all over the world to design, develop, test through automated mock-ups, and document APIs.

[View on website](#)



The market development in the coming years will star a magical key to almost all doors: APIs. BBVAOpen4U has explained [how to measure the performance of application programming interfaces](#); [described their advantages](#); explained [how developers can organize a large depository](#); and also [how they influence the Internet of Things](#) and [wearables](#). But we haven't explained from to program an API from scratch.

To do this, developers can use platforms, tools and languages to design, develop, test and document their own APIs and, as a result, facilitate product programming for third parties and generate revenue. There are currently several leading options: [Restlet Studio](#), [Swagger](#), [API Blueprint](#), [RAML](#), [Mockable.io](#), [Loader.io](#), [BlazeMeter](#), [Apiary](#) and [InstantAPI](#). There are other tools but these are the most well known among the community.



Restlet: Platform as a Service for APIs

Restlet is an Integrated Development Environment (IDE) where Java programmers can design their own web APIs based on REST architecture (REST APIs). They can develop server- and client-side applications and Restlet is compatible with HTTP, HTTPS, XML and JSON. This development framework is open source, free to download and under Apache license.

There are several pricing plans: one is for free and the others are paid. With the first, you can develop one API; the most expensive plan has no restrictions.

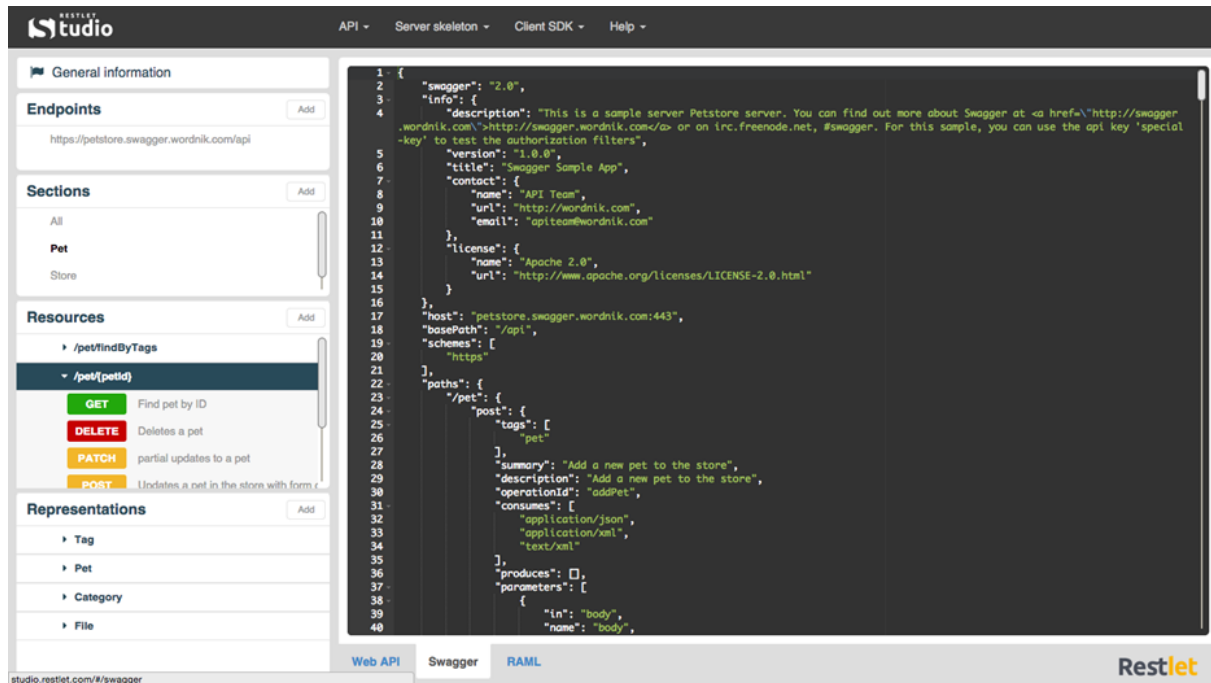
Restlet Studio is available for all platforms (Java SE/EE, Google App Engine, Google Web Toolkit, OSGI and Android). All APIs that are developed using Restlet Studio can be integrated with APISpark, Restlet's Platform as a Service (PaaS) for hosting and managing APIs by any developer in

any type of language, e.g. Java, PHP, Node.js or HTML, and in any framework, e.g. as AngularJS, etc. [Jerome Louvel](#), CTO and founder of Restlet, [explained the basic characteristics of APISpark](#) in an interview with InfoQ in November 2014.



Two main points with regard to developing APIs using Restlet Studio:

- Restlet's development framework offers a series of classes and interfaces which you can use to design your own APIs.
- Guaranteed scalability irrespective of the number of requests.





Swagger: the most popular API framework

[Swagger](#) is currently on version 2.0; it is an open-source framework; and it is used by very important platforms and customers such as Apigee, Getty Images, Microsoft and PayPal.

Swagger has allowed them to develop their own RESTful APIs. What is Swagger? A series of tools for programming application development interfaces in virtually every development language and environment.

Swagger's tools include:

- [Swagger Editor](#): edit API specifications in YAML (YAML - Ain't Another Markup Language).
YAML is a light markup language. It is a data format inspired in languages such as XML and Python that focuses more in data and less in document markup. To execute it locally in a workstation with Node.js use this command:
- [Swagger UI](#): collection of HTML, JavaScript and CSS assets that allow you to dynamically generate documentation and a sandbox for any API compatible with Swagger. Since it has no specific dependency, the user interface can be hosted in any server as well as locally.
- [Swagger Core](#): implementation of Swagger in Java. Series of open-source Java libraries that can be found in GitHub. Here you find [a lot of specific documentation for developers](#).

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



API Blueprint: documentation for APIs

API Blueprint is a language based on Markdown (light markup language) which is mostly used to document any API in a simple way. For API developers, what's really interesting about API Blueprint is the tools that work as its satellites.

The most intriguing of these tools is [Dredd](#) (a reference to the movie character Judge Dredd) which allows you to test a backend service from the API's documentation. As a result, you can solve document update issues. It supports all types of languages, e.g. PHP, Python, Ruby, Perl, Node.js and Go.

Another interesting tool is [Drakov](#), which allows you to launch mock-up services to run tests with requests and responses tailored to the API documentation. In other words, this is a test bank. [To starting working with API Blueprint you can read this tutorial](#).



RAML: full API management

RAML stands for RESTful API Modeling Language. Its aim is to facilitate API life cycle management, from design and development to use by third parties (testing and documentation). It stresses the use of a language that is easy to interpret by both developers and machines. The latest version is 1.0.

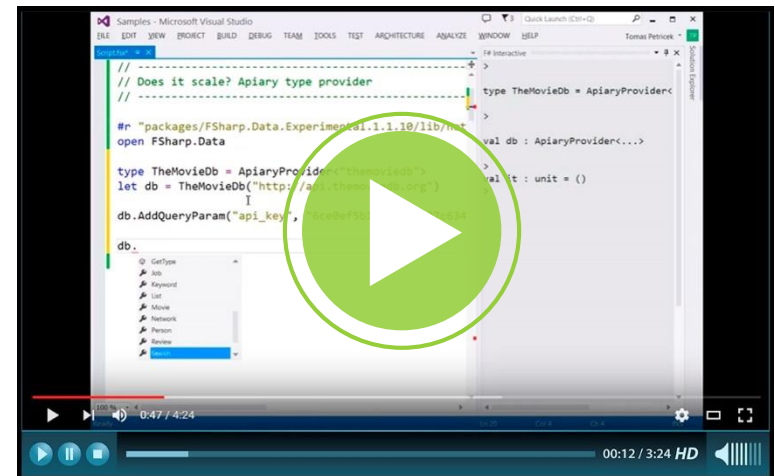
RAML allows you to develop APIs in several types of syntax: Node.js (JavaScript), Java, .NET and Python. It also offers a wide range of tools for testing application development interfaces with RAML: [Abao](#), a command line tool in Node.js to [test API documentation written in RAML](#); Vigia; and [Postman](#), Google Chrome extension that is very popular among developers due to its simplicity. It tests APIs through requests, i.e. GET, POST, PUT, PATCH and DELETE.



Apiary: your own API in 30 minutes

It sounds a bit risky but [Apiary](#) promises to offer developers all the tools they need to create their own API in 30 minutes. It takes over the entire life cycle of an application development interface: design and development, automated mock-ups, validation, proxies, documentation, etc. Apiary offers all you need to have an API.

Apiary provides mock servers to DevOps teams so that they can run tests and mock-ups before encoding an API, something like wireframes in user interfaces. Before you start designing, it's a good idea to plan so that you determine your actual needs and then assign the right number of resources to the project.





05 Key Developments APIs

APIs have become the new essential tool for most development and business equipment (DevOps). In the short-term future, it is expected that APIs become universal, turn banks into platforms and give microservices an extra boost.

[View on website](#)



The hype surrounding the API economy is not exaggerated. Like almost all eruptions on the IT market, application development interfaces caused an uproar in 2015, and no less is expected for 2016. It's not a new thing; since 2012, APIs have been growing as the real glue between software creation equipment, data management and the business infrastructure. Without them, it's difficult to compete now and in the future.

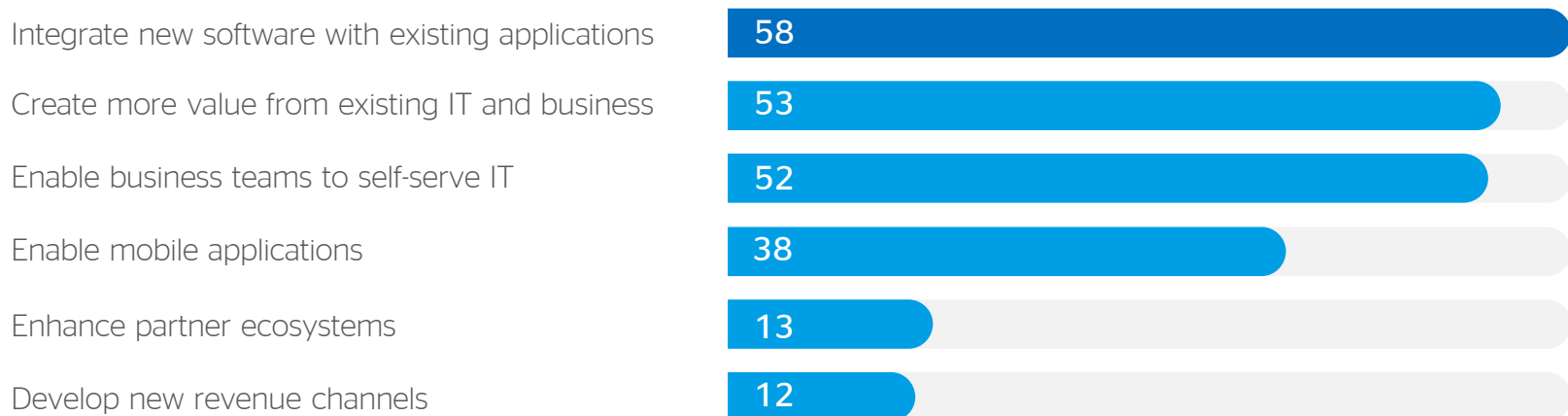
APIs are the real key for creating products and services and for generating revenue and brand commitment. Nowadays, no one in the market doubts the importance of APIs in the B2B business ([business to business](#)), either through public interfaces or the development of internal APIs. According to the study '[Global API Market Forecast to 2017](#)', the API business generated 113 billion dollars in 2012, and it is expected to grow by 8% per year until 2017.



In another recently published report, '[2016 Connectivity Benchmark Report](#)', the company Mulesoft provides some very interesting information about this API economy. One of the most impacting conclusions is that 91% of companies have a strategy that is currently being implemented based around these application development interfaces, or they are thinking of starting it up this year. In most cases, it is focused on three fields of business: software integration with existing platforms or applications; the creation of revenue from IT and business, and increasing speed and providing business with IT resources.

Business types in a API strategies

Cast of use (%) of API by companies to generate income



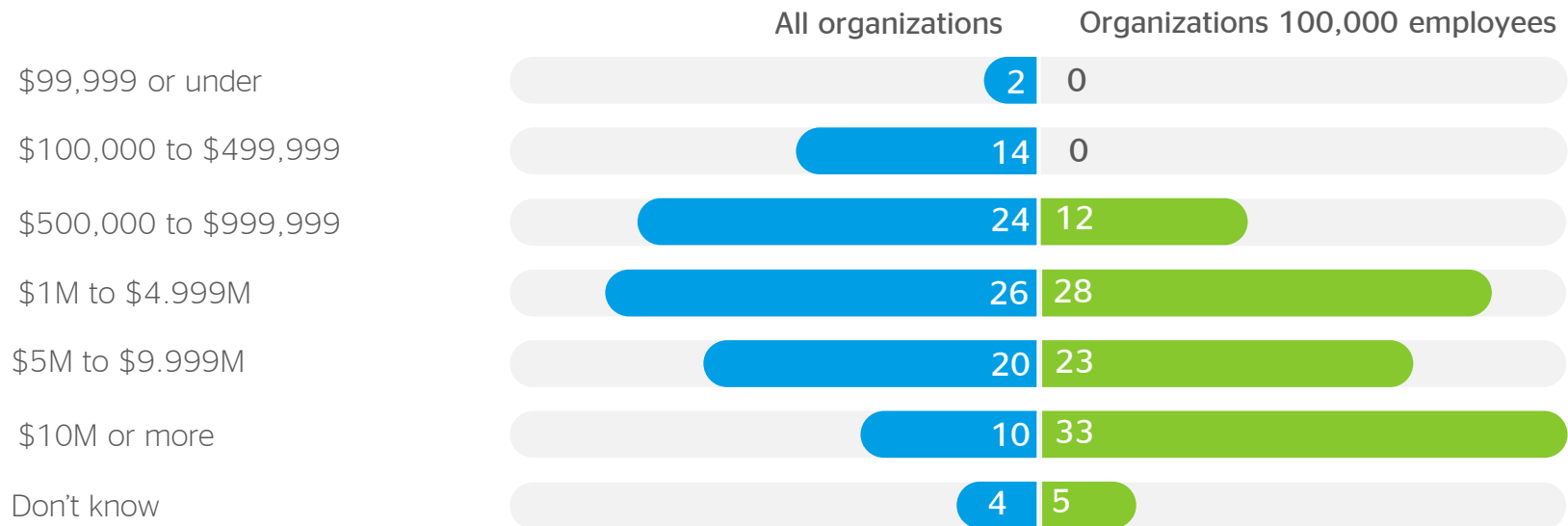
Source: Mulesoft 2016 Connectivity Benchmark Report, Get the data

Another piece of really interesting information from the report is what companies expect to earn with the development of these types of API related strategies: 33% of organizations with more than 100,000 employees in the study expect to earn more than 10 million dollars using these app development interfaces. Furthermore, 26% of those surveyed plan on generating between one and five million dollars.

Here is a chart with all the results from the report drawn up by Mulesoft.

Ingresos esperados por las empresas con el uso de APIs

Prediction earning expected by companies through the use of APIs or related.



Source: Mulesoft 2016 Connectivity Benchmark Report, Get the data

APIs: a resource for all

In the coming years, no company may raise your future does not depend on one or more APIs, own or third. The digital products and services economy has become the APIs economy. And that's a reality there's no escaping from. This will speed up the integration process of many company's departments beyond technical equipment. The universalization of APIs will enable four aspects

- Accelerating the digitalization of processes..
- The adoption of applications and platforms by third parties.
- Business analysis.
- The monetization of these digital channels' products and services.

Most of the blame comes down to the huge digitalization process of the consumption of information and services via numerous devices: not just smartphones, but also [wearables such as smart watches or sports wristbands](#). That's without taking into account [the perspectives of the future business of the Internet of Things](#). To take advantage of these opportunities, a democratization process is essential for the use of APIs, making them simpler and more accessible.



Open banking based on APIs

APIs could become the huge differentiating element of **banking in the future**. Banks who understand that application development interfaces are the perfect instrument to make money with third parties and that changing from traditional banking to becoming Platforms as a Service (PaaS) will win in the future with their relationships with account information services (AIS) and the new Payment Initiation Services (PIS), which are taking business away from financial institutions.

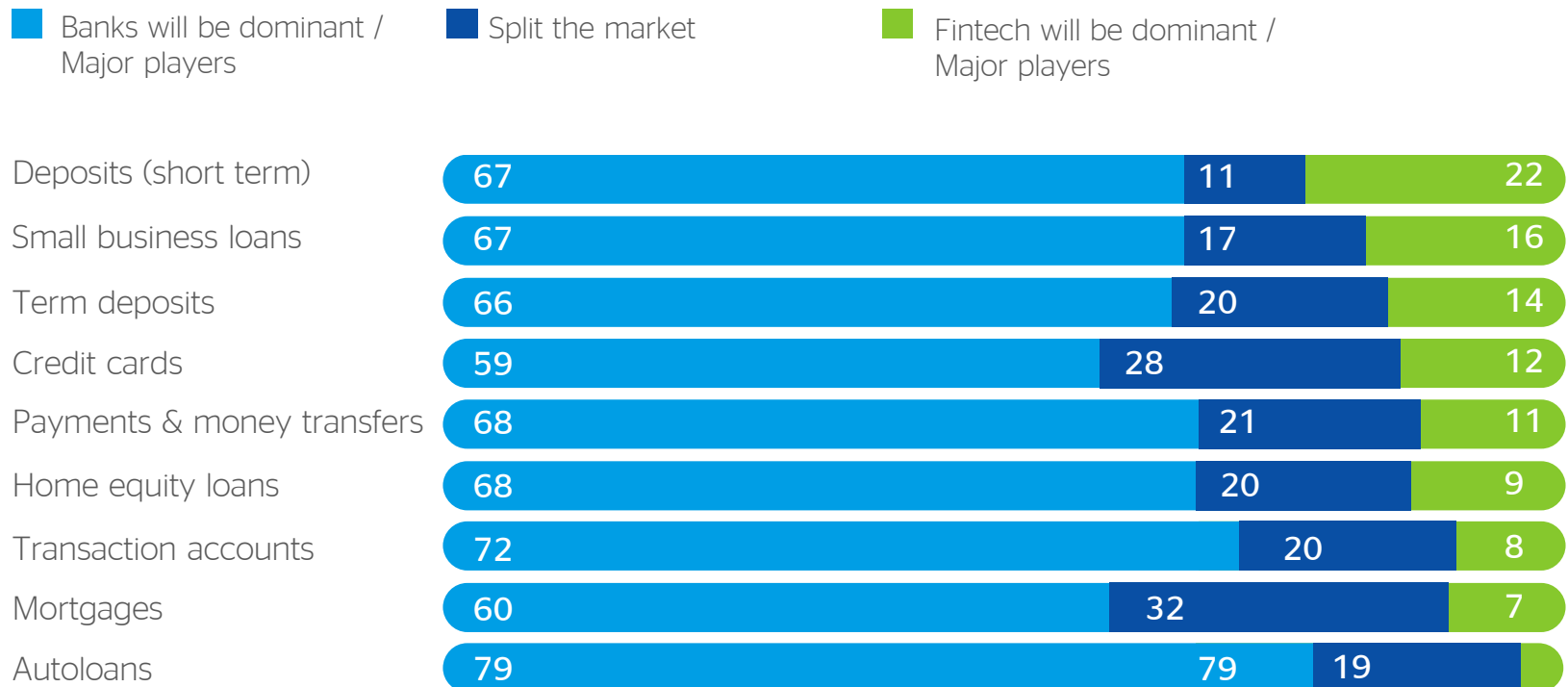
To regulate this context, in October 2015, the EU updated [the Payment Services Directive \(PSD2\)](#), which gives banks the opportunity to do business with open service platforms by using APIs. The legislation forces banks to give third parties access, with permission from their customers, to account and payment related information.

Banks such as BBVA and JP Morgan Chase have understood this new scenario and are willing to **convert an obligation into an opportunity: take advantage of this revenue asset**. This is what's known as the 'platformification' of banking.

There are two interesting reports about this transformation of banking into service platforms: ['2016 Retail Banking Trends and Predictions'](#) is a good [summary of the radical change in banking institutions](#), analyzing a large number of the agents who have implemented this change; and a second report, ['The disruption of banking'](#), which is an analysis by the magazine The Economist from the end of 2015. The second report gives many keys about the new scenario in which banking products will compete with their new and powerful rivals, fintech.

The future landscape - balance of banking and Fintech by product

For each banking product, what is the most likely competitive balance between banking and Fintech in five years?



Source: "The disruption of banking" (The Economist), Get the data

APIs for the architecture of applications based on microservices

The architecture of microservices is the new paradigm of DevOps equipment. What does it involve? In the development of applications as a group of microservices that are executed independently, and which normally communicate with the different APIs via HTTP requests.

This has several advantages:

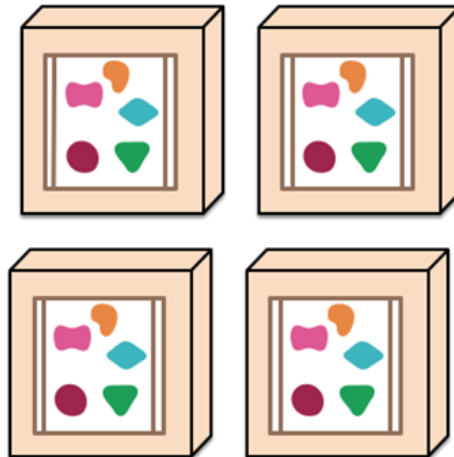
- If some of the microservices in an application, whether it be the user interface (front-end) or back-end services, have an error, the development equipment can resolve bugs separately **without having to send the entire application to production with the development improvements.**
- Something similar occurs with **scalability processes, either up (growth) or down (decrease)**. In an application based on microservices development, capacity growth is simpler and more rational because each one is an independent service.
- The business logic is separated and each part is independent.
- **Integration mechanisms such as ESB** (Enterprise Service Bus, a component of the Service Oriented Architecture or SOA) have been rejected in favor of lightweight mechanisms such as applications based on the queuing systems..

This article by [Martin Fowler](#) and [James Lewis](#) is often brought up to explain [what microservices are for and what they are used for](#) in the development of applications. The report explains how APIs are key to the flexibility and agility of microservices.

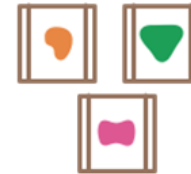
A monolithic application puts all its functionality into a single process...



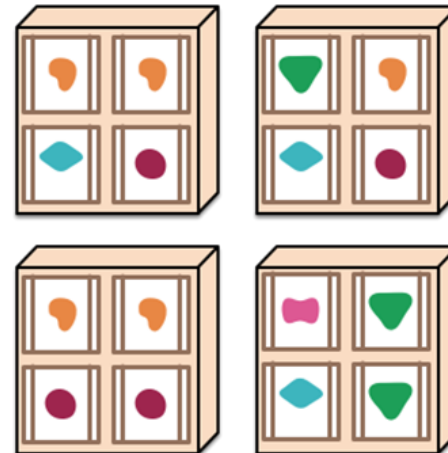
... and scales by replicating the monolith on multiple servers



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



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



Fowler's speech is also interesting for understanding microservices:



Use of APIs: measure, measure, measure

With the API monitoring tools, without which it is difficult to know what the actual performance of these interfaces is, development equipment can analyze performance and correct bugs or errors that interfere in its performance in a timely manner. [Mashape Analytics](#), [Akana Envision](#) and [CA App Synthetic Monitor](#) are three examples.



API security is a priority

The growth of APIs as application development and integration instruments, especially in the field of mobile device development, attracts those who work on finding weaknesses. Nowadays most APIs base their security on a protocol framework creation, such as [OAuth2](#).

By using application development interface capital for their business, large IT companies such as Google, Facebook, and Twitter use this authorization and security system for third parties. [The system is based on the creation of access tokens](#) so that third party developers can use the API without over committing the tool. A credentials system based on a username and password is more delicate, because an irregular breakdown of this protocol would result in having to change the access for all that API's customers.

BBVAOpen4U

www.bbvaopen4u.com



SIGN UP

to the BBVA Open4U newsletter and receive tips, tools and the most innovative events directly in your inbox.

Other ebooks in BBVA Open4U



[Ebook: Fintech revolutions](#)



[Ebook: APIs & Internet of Things](#)



[The top Fintech U stories](#)

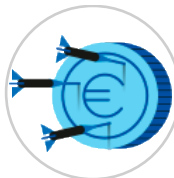
Share



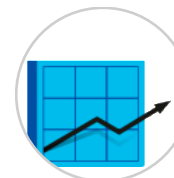
Try BBVA's APIs at www.bbvaapimarket.com



Identity



Accounts



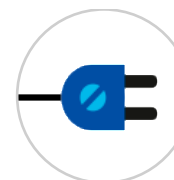
PayStats



Money Transfers



Cards



BBVA Connect

“A Company Without APIs Is Like A Computer Without Internet”

BRIAN KOLE

