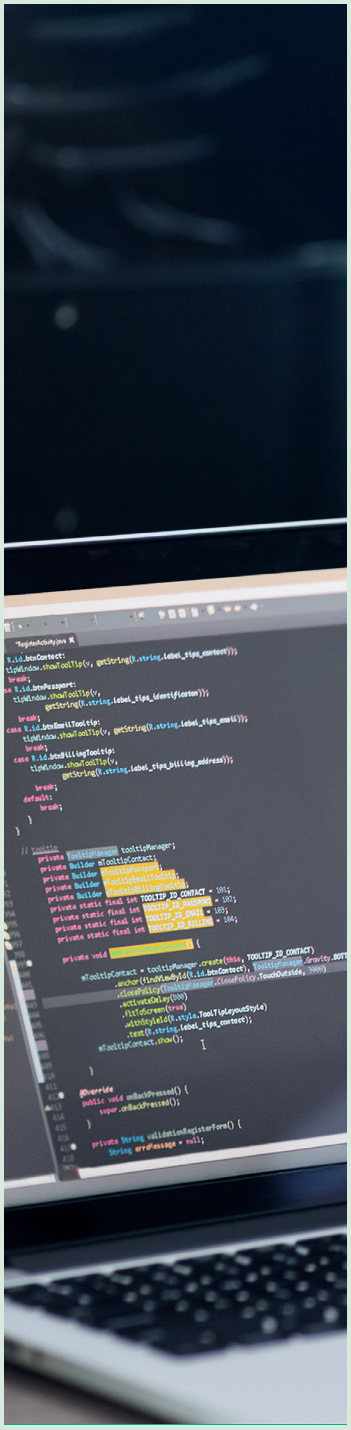
WHITE PAPER



NEXT GEN INTEGRATION AND API ECONOMY

Infosys®
Navigate your next

Executive Summary

Industry leaders say that Integration is the key to Digital Economy. APIs are the foundation of Next Gen Integration, be it Cloud integration, Application integration, B2B integration or Enterprise Integration.

In current context, APIs are simple to understand interfaces focused on business' recognizable assets, that facilitate integration with peer applications or systems in an Agile manner.

A new Developer Ecosystem is emerging where developers are designing innovative applications, reaching new customers and exploring new markets, all around APIs.

In an Enterprise Integration ecosystem, APIs help in exposing an enterprise's backend-as-a-service so that new applications can quickly be built on top of that. Why this works is because a significant useful data is almost locked inside big and complex legacy enterprise systems, and data warehouses, and APIs unlock that data, that precious data.

All such ecosystems emerged slowly in last few years, and monetized transactions on the APIs, which created new digital revenue streams, which in turn, led to API economy.

Businesses, both large and small, both B2C and B2B, all participate in the API economy.

This paper brings connectivity and integration into spotlight, showcasing the entire journey that has taken place and the pace it's moving ahead in.

Introduction

We live in an economy powered by digital computing technologies, cushioned by the Internet and the World Wide Web. It is called the digital economy, and it is worth three trillion dollars today! Cloud, cognitive computing, IoT and mobile apps are the drivers of this economy, and every one of us is part of it with online shopping, digital banking, social media, etc.

Behind each of the apps that we are associated with, there is at least one enterprise system from which data is fetched by one or more APIs.

Which API am I using?

Look at the following three scenarios:

1. You Uber to the railway station, and before and throughout the ride, you see your location on the map
2. On your way, if you like a news article on BBC, you can share it on your Twitter feed on click of share button
3. On the train, you can make payment for food and beverages using PayPal

In all the three scenarios, you have experienced integration via APIs:

1. Uber integrates with Google Maps to provide location service
2. BBC integrates with Twitter to share an article
3. PayPal integrates with Bank(s) to make payments

What is integration?

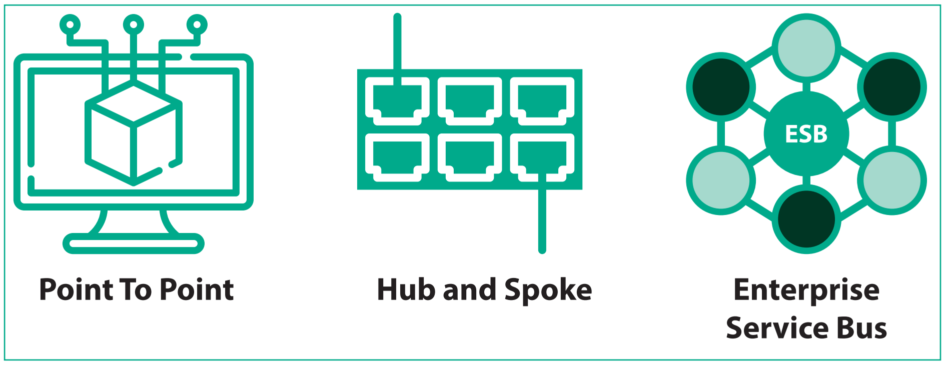
Integration is connecting two software systems together such that they can communicate with each other. The two most fundamental aspects of integration are and have always been – the protocol and the data format. Both systems need to talk on a common protocol and understand each other's data format.

Integration of systems dates back to the 1980s, and the entire integration journey can be divided into two – The Traditional and The Next Gen. APIs obviously falls under the latter.

Traditional integration

Traditional integration methods are:

1. Point to Point (P2P)
2. Hub and Spoke (H&S)
3. Enterprise Service Bus (ESB)



Point to Point integration

The first model was the point-to-point integration where a customized code was written at each application's end for integrating with any other application. The code took care of converting the protocol and the data format. Soon, third party integration tools emerged which dictated the transport protocol to connect the varied applications though the applications still had to take care of the data conversion aspect. The popular products are IBM MQ series from IBM, Active MQ, RabbitMQ, Apache Kafka.

Hub and Spoke

The second model was hub and spoke where the entire integration code was brought at one place called the hub. The applications, referred to as spokes, connected to the hub in turn. The hub took care of converting the data formats of messages, routing them to the right destination(s), filtering, aggregating data from several sources and delivering in an asynchronous way. This model did well when compared to P2P, but the delivery of the messages wasn't in real-time, and there was complete dependency on the runtime; if the runtime was down, there was nothing that could be done, since, both code and deployment was at one place.

Hub and Spoke is still used in B2B solutions, like Adeptia Connect, and as part of other architecture like in hybrid networks in Microsoft Azure's architecture, where the Hub isolates workloads between the virtual networks connected as spokes.

Enterprise Service Bus

The third model ESB (Enterprise Service Bus) is the long standing integration model, that came around 2000s and is still being used extensively today. SOA or Service Oriented Architecture was the driving force that made ESB happen.

Applications expose SOA based web services which integrate through ESB. The communication protocol that SOA uses for web services is SOAP i.e. Simple Object Access Protocol. SOAP works on top of HTTP, though it can ride a variety of network protocols. And the language SOA uses to describe the functionality offered by a web service, is WSDL i.e. Web Services Description Language.

Since both the open standards, SOA makes the communication between applications independent of vendors, products and technologies.

Enterprise Service Bus was built in SOA's image, where all applications could connect with ESB and send data, mostly in XML format, in real-time. ESB extended support to non-standard protocols and data formats too by allowing customization!

ESB has worked well from the 1990s to 2000s. The most popular products for ESB are IBM Integration Bus, Oracle ESB, TIBCO Business Works, Mule ESB, SAP NetWeaver, Microsoft BizTalk, JBoss Fuse.

Why not SOAP?

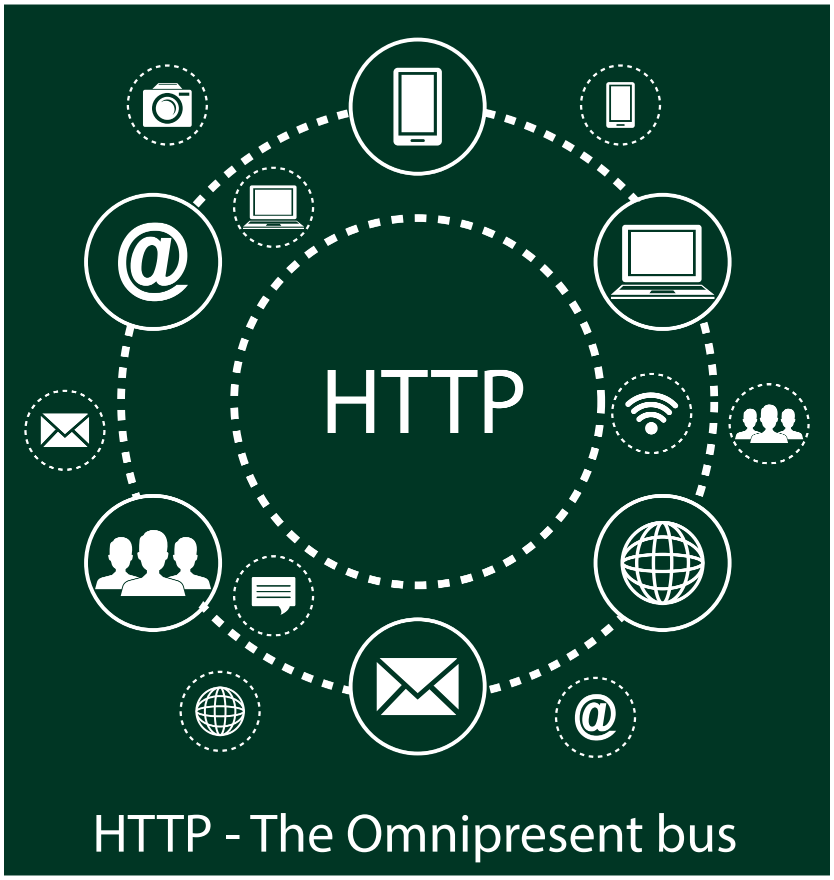
During the last decade, Cloud became quite a popular way to store data, and later use it as an infrastructure for enterprise use, as either host an application or the entire IT architecture. Soon IT found SOAP too heavy to work with in this context, especially for integrating apps on cloud. We needed a way to integrate in a more Agile manner and on Cloud.

REST

In the year 2000, Roy Fielding coined a term called REST (Representational State Transfer) as part of this Ph.D. dissertation. REST is an architectural style for creating web services. RESTful web services formed the second generation of web services, and it came as the much needed solution for interacting on Cloud. One of the key factors of REST is that it uses HTTP, the transport protocol of the internet, for interaction, and JSON, another self-defining format for data, which is not as complicated as XML. Both these factors removed the bulkiness of SOAP based web services.

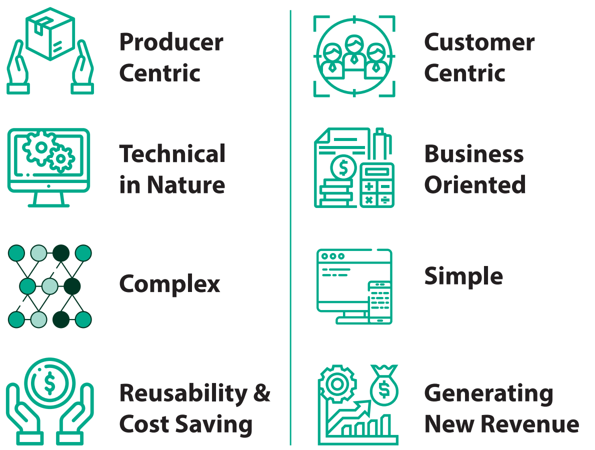
Further, SOAP based web service focused on what the application could give as service, whereas Restful web-services focused on what the customer wanted, and the interface was created to achieve that integration. Customer centricity brought a significant change in approach towards integration. The purpose of integration was no more just cost saving and reusability, but generation of new business and revenue. The term web service didn't do justice to the new approach, hence came the term API. It then started gaining the name Business API, Web API, or simply API. API stands for Application Programming Interface, and it is customer-centric.

So, on context of Agility and Customer Centricity, REST seemed a better fit over SOAP.



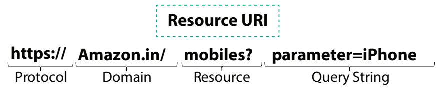
Customer centric context when added to a RESTful web service, makes it a RESTful API

SOAP vs REST



How to work with an API?

API needs to be always looked at like a product, with its own marketing and sales strategy. APIs are always created with customers in mind, where customers are developers who are involved in creating applications for either another developer or the end users. APIs identify the entities of the web application as resources, and HTTP methods are used to read, write, modify and delete them: GET, POST, PUT & PATCH, and DELETE, respectively.



```
Request: POST/products
Description: Creates a new product
Body: {
            "prodName  ": "laptop",
            "price":  43000,
            "yearMfg  ": 2019,
            "status": "active"
}

Response: 201 OK
Description: Returns a confirmation of added product with
hypermedia links
Body:
{
          "id": 105,
          "prodName": "Laptop",
          "price": 43000,
          "yearMfg": 2019,
          "status": "active"
}
```

API Lifecycle Management

The creation of an API is seen in three stages:

1. Design
2. Develop
3. Deploy

Design

An API takes a design-first approach where you identify and create a contract of what that API does that can be shared with stakeholders for feedback. API description languages like RAML (RESTful API Modelling Language) and Swagger, now called OAS (Open API Specification) are available to enable the contract-first design approach. Dedicated platforms are available to design API. The popular ones are MuleSoft Anypoint Platform's Design Center, AWS API Gateway, Google Apigee, IBM API Connect, Kong.

Develop

APIs are developed with backend implementation in the next phase. Backend could be developed in any stack like Springboot, .Net stack or Node.js, and it could be in monolith or microservices architecture as discussed in section How APIs connect to Microservices. And APIs can be developed in platforms like webMethods, MuleSoft Anypoint Studio, IBM Cloud.

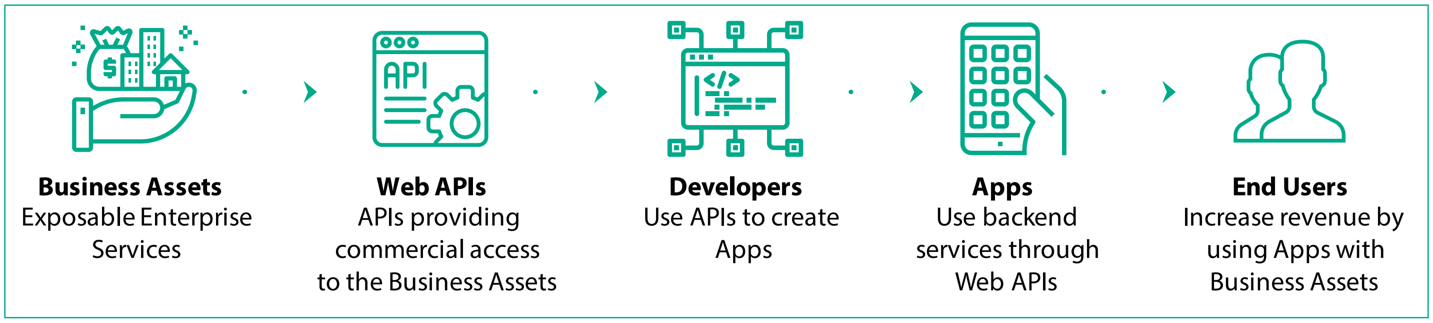
Deploy

Developers host their APIs on a platform where potential clients find them. This is called the API Management platform, an architectural layer that brokers the businesses core capabilities, data, and services with the digital application ecosystem. It takes care of developer engagement, documentation, API security, performance, traffic management, version management and API monetization. The popular ones are Google Apigee, Anypoint Exchange, Kong, WSO2, 3scale API Management, Layer 7, Axway Platform.

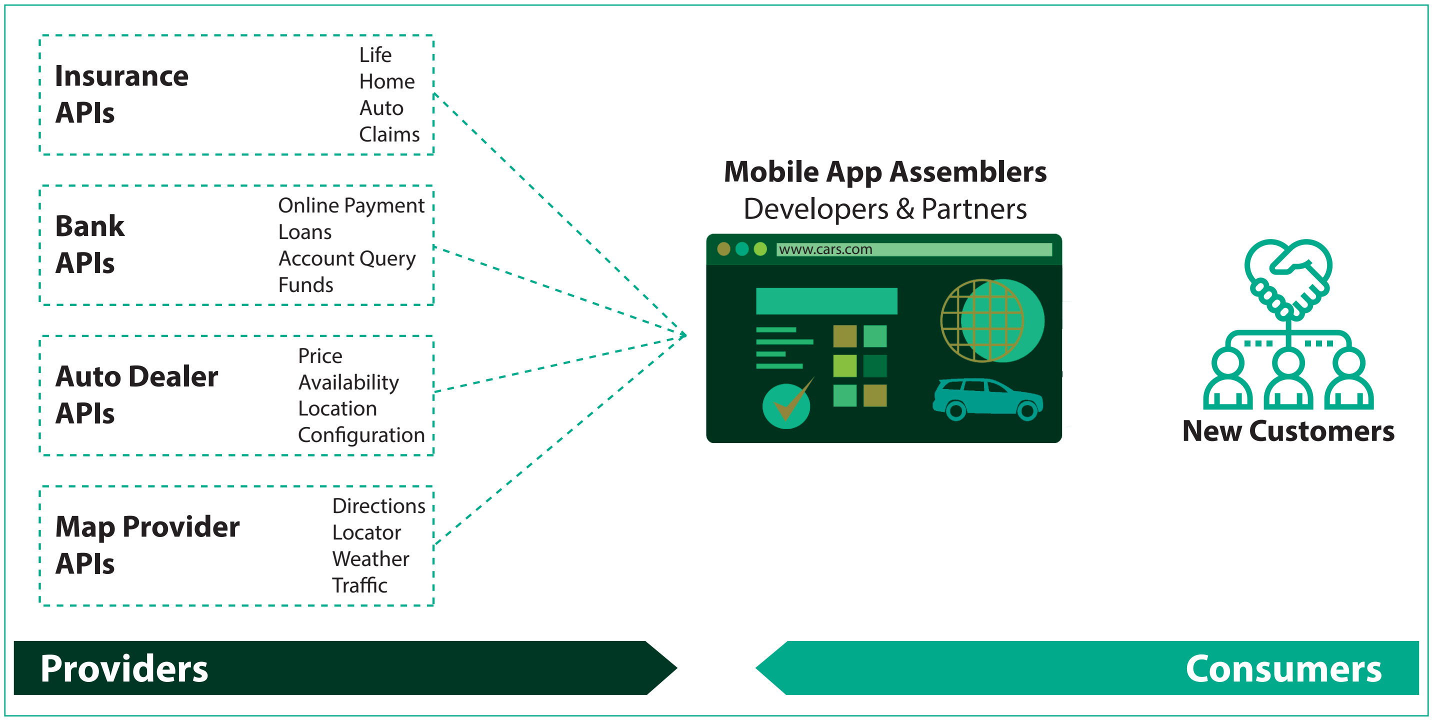
API Economy

We, as end customers, pay Uber for its services, and Uber pays Google in turn for using its GoogleMaps API. Both the vendors are earning and customer is also happy. This indicates API Economy. There is no strict definition, but we can loosely define it as the way APIs positively impact an organization's profitability. Businesses, both large and small, both B2C and B2B, all participate in the API economy.

The map for API Economy is:

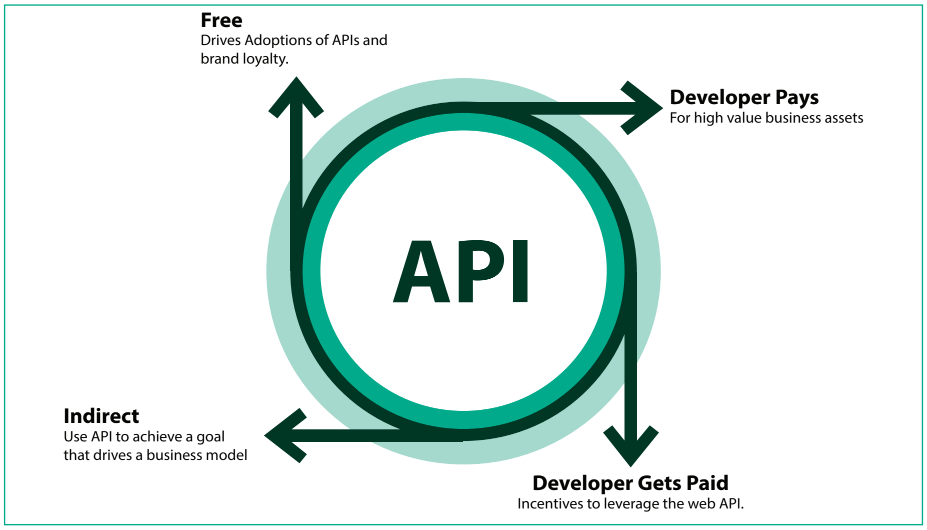

Business Assets
Exposable Enterprise Services
API
Web APIs
APIs providing commercial access to the Business Assets
Developers
Use APIs to create Apps
Apps
Use backend services through Web APIs
End Users
Increase revenue by using Apps with Business Assets

Developers are creating a significant portion of their app by using APIs, so it's no more restricted to integration, but APIs are becoming a way of creating or assembling new applications. For example, an app like cars.com have plethora of API vendors to choose from, for providing insurance functionality, banking functionality, car location using maps etc.

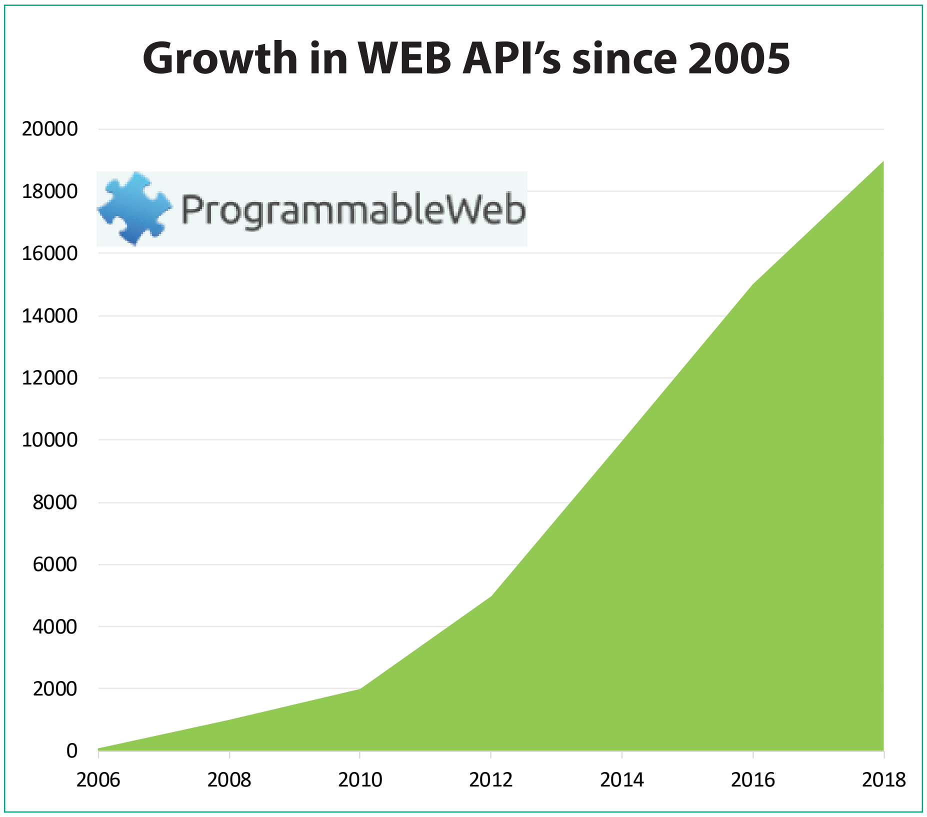


In banking and financial sectors, APIs are created and provided by banks to help with programmable access to functionalities and better connectivity. Stark Bank, based out of Brazil, is an open banking tool that helps companies simplify the process of making transfers and issuing charges. It helps create opening banking technology for companies to make transfers and issue charges in a transparent, simple and secure way.

When you create an API for specific audiences, chart out a plan for revenue generation:


Free
Drives Adoptions of APIs and brand loyalty.
Developer Pays
For high value business assets
API
Indirect
Use API to achieve a goal that drives a business model
Developer Gets Paid
Incentives to leverage the web API.

The growth of the web APIs since 2005 is phenomenal:
Source programmableweb.com



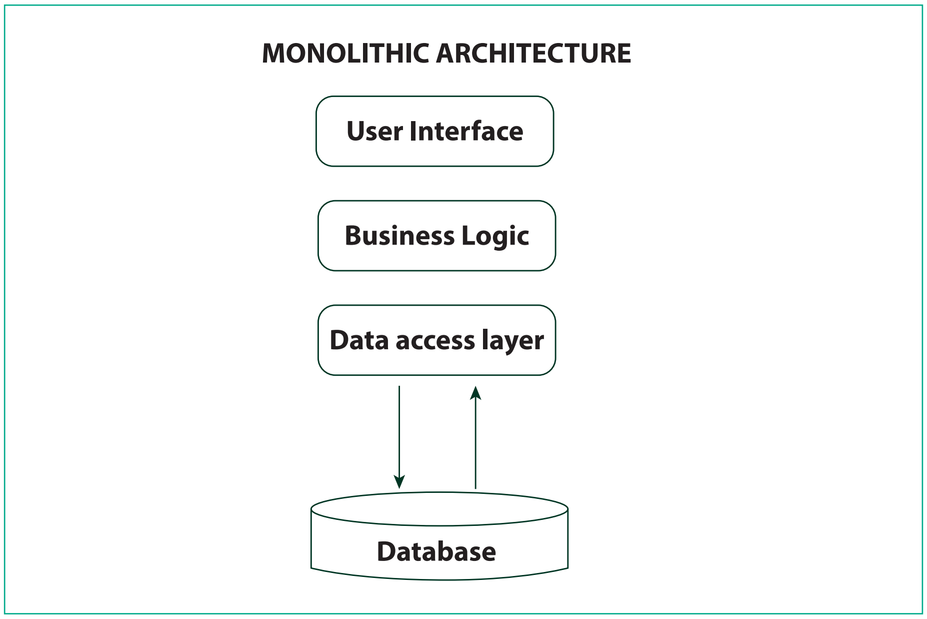
Products and application across industry domains have API developer platforms, few examples below:

Product / UI / User Login	Platform / API / Developer Login
https://www.uber.com/	https://developer.uber.com/
https://www.hsbc.com/	https://api.hsbc.com/
https://www.mercedes-benz.com/en/	https://developer.mercedes-benz.com/
https://www.coca-cola.com/	http://developer.cokecce.com/
https://content.finacle.com/	https://infosys-finacle.openbankproject.com/

API Categories

An API could be internal to a company, called private API. It could be for specific partners, called partner API. And the third category is external, which is for the whole public, called the public APIs.

It is recommended for enterprises to work inside out, create internal APIs for internal businesses. For example, a UK newspaper, called The Guardian have internal APIs that handle about 6 to 7 times the traffic when compared to external APIs.



API Drivers

There are the primary factors that drive the market towards the API economy:

1. **Agility:** Try new things in lesser time. When enterprise functionalities are encapsulated as APIs, developers can implement newer functionalities quacking using these already present APIs without bothering the existing systems
2. **Reach:** Reaching new customers, seeking those out who won't come to you. For example, comparison apps like Expedia, MakeMyTrip, Airbnb helps reach flights and hotels that wouldn't be found directly
3. **Innovation:** Innovating newer ways of doing business, and promoting to reach more customers. For example, Google Pay leveraged UPI to achieve money transfers and payments

How APIs connect to Microservices?

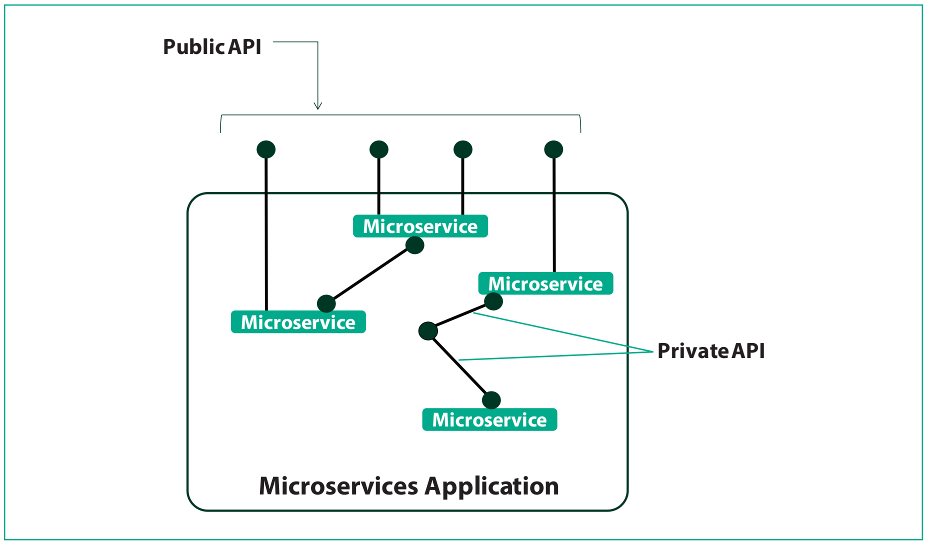
Here, we discuss about the API implementation.

Most businesses are based on Monolithic architecture, which is the entire application in one piece. It is self-contained, and the functionalities of the application are interconnected and interdependent rather than loosely coupled.

Consider a shopping application that is expecting huge traffic for Black Friday sale, and anticipate that the registrations to be surge high. If they wish to only scale the registration service, it wouldn't be possible without scaling up, or running multiple instances of the whole application.

In contrast, Microservices is an architectural style has been adopted since a decade ago that recognizes each application's functionality as an independent service i.e. independently deployable and hence independently scalable and maintainable. Each microservice is atomic in nature, centered around a business functionality

They become separately running processes in the network and communicate with each other using APIs, the private type of APIs. APIs are about the interfaces, while Microservices is how the application is implemented inside.



LinkedIn, Uber, Ebay, Amazon, Nike are some of the early adopters of Microservices.

Summary

APIs and API Economy are now past the hype stage. People are implementing real projects, new ecosystems are emerging, and in some cases even industry standards are being developed around APIs. The rate at which APIs are growing, and their usage across all Industry domains is such that that almost no business or application is untouched by their presence.

APIs are created as products which create the revenue generation stream, which promotes reuse, standardization and improves throughput. And with APIs, the way enterprises build their IT architecture is revolutionized.

However, though REST seems like a shiny new toy, it isn't, and definitely not the only one. It is one of the options available to the API developer among gRPC, a modern open source high performance RPC framework that can run in any environment, GraphQL, another open source data query and manipulation language for APIs, and Webhooks, a method to augment and alter the behavior of web applications using customer callbacks.

The choice of the method depends on the API developer and business requirement.

References

1. https://play.google.com/store/books/details?id=JXgBDQAAQBAJ
2. https://www.programmableweb.com/ss
3. https://starkbank.com/
4. https://www.mulesoft.com/resources/api/what-is-api-management
5. https://www.sciencedaily.com/terms/digital_economy.htm
6. https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf
7. https://smartbear.com/blog/develop/the-age-of-the-api-economy-what-it-is-why-it-matte/
8. https://developer.ibm.com/apiconnect/2014/08/12/api-economy-drivers/

For more information, contact askus@infosys.com

Infosys®
Navigate your next