MFSA FinSights | Enabling Technologies

Application Programming Interface (API) and Microservices

The financial sector is continuously evolving through the rapid development and adoption of new technologies. The term 'FinTech' generally refers to financial innovation that seek to provide enhanced financial service offerings through the utilisation of enabling technologies. These generally include Distributed Ledger Technology & Smart Contracts; Artificial Intelligence, Machine Learning & Big Data, Cloud Computing, Web 3.0, Application Programme Interfaces and Micro-Services; Robotic Process Automation and the Internet of Things.

As part of the MFSA's initiatives to generate awareness, drive culture and deliver a cross-sectoral knowledge platform which can support the MFSA's functions in preparing for the financial services of tomorrow, these insights will delve into enabling technologies, enabling innovations and their sectoral applications.

1 What is an API?

The Application Programming Interface ('API'), which emerged in the 1940s, is programmed code, that governs access points[1] and allows two different applications, systems, or software to communicate with each other, using a set of common standards. In other words, it facilitates direct database-to-database transmission of data whilst ensuring granular, real-time reporting with automated validation.

API integration refers to a process where two or more applications communicate with each other and exchange data via their APIs in order to synchronise data, enhance productivity and/or drive revenue. It is also used for seamless information/data sharing in many sectors and organisations, without human interaction. This is especially important for businesses that use cloud-based products and apps and need connected systems for data exchange between various other software tools.

1.1 Web APIs

Web APIs allow users certain access to data from other applications, upon a submission of a request. The request is received by the API, whereby it then fetches the data from the back-end and delivers it to the user via the front-end web application user interface, as depicted in Figure 1. The exchange of data between client and server through **HTTP**[2], would create a Web API.

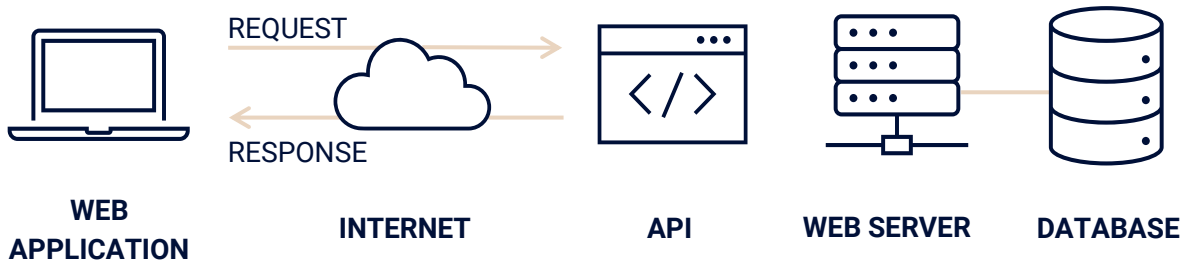

FIGURE 1: WEB APIs
Source: Authors' Own Sources

[1] Access points are a piece of computer equipment that allows connection between computers without wires to share information. Cambridge University Press (2022)
[2] Hypertext Transfer Protocol ('HTTP') is a set of instructions made by a computer program that allows your computer to connect to an internet document. Cambridge University Press (2022)

There are four types of Web APIs which indicate the intended use of the API:

1. **Open APIs –** also known as **public APIs**, are a type of Web APIs that are openly available and accessible on the internet, without any restrictions. Due to easy implementation and lack of restrictions, third parties may seamlessly leverage the data provided by the API. The use of public APIs has paved way for creation of open banking services that essentially allow external developers to create programs and services that facilitate access to certain data held by credit institutions.

2. **Partner APIs –** refer to Web API types, which are private and often require a prerequisite license or subscription that can be acquired through a partnership agreement, between a business and a service provider. Partner APIs often refer to Software as a Service ('SaaS') model whereby businesses gain access to software for a specific purpose and facilitate data sharing via cloud computing.

3. **Internal APIs –** also known as **private APIs** and can only be used within an organisation, company, department. Hence, this type of Web API is responsible for providing access to sensitive information of the company, such as salary or sales data, to the select team or individuals within the corporate structure.

4. **Composite APIs –** are a form of Web API that batches multiple API calls in a single request. This increases efficiency in the passing of data rather than having multiple separate responses. These are often used in eCommerce sites with final shopping cart API calls being executed all at once.

1.2 API Architectures

In order to develop an API-based application one must first decide on the **API architectural style** to be used. Such styles differ in their use cases and specifications, some of the common API architectures are as follows:

- **RPC (Remote Procedure Call) –** A RPC API is the simplest form of API interaction whereby the client executes code on another server. The most common RPC languages are JSON, and XML. Whereas the former is the most commonly used format due to its readability and ease of application, the latter is the long-standing API format that forms an integral part of every web application.

- **SOAP (Simple Object Access Protocol) –** SOAP API relies on XML[3] for the exchange of messages between the client and the server. The XML messages are always? made up of four blocks: envelope, header, body, fault, which serve together in SOAP APIs. This architectural style is more secure than RPC and handles a higher XML payload as compared to RPC.

- **REST (Representational State Transfer) –** REST is an API architectural type that relies on a few guiding principles such as uniform interfaces, stateless operations and a layered system of components that only interact hierarchically. One may thus deploy APIs on a server whilst having data stored on another server and rely on a third server for request authentication. Essentially, unlike SOAP, which is based on specific requirements such as XML messaging, REST is represented by a set of guidelines offering adaptable implementation.

- **GraphQL –** may be deemed as a consumer-centric API style having the design flexibility that allows it to be interoperable with any kind of database. GraphQL provides its clients with control over the data provided to the API consumer while also allowing them to describe their data requirements in JSON format. It also allows for faster client request execution by limiting the field to be queried.

[3] Extensible Markup Language (XML) is a text-based format that represents structured information such as documents, data, transactions, invoices, and more.

A summary of API Architectures is provided in the Table 1 below.

	RPC	SOAP	REST	GraphQL
Characteristics	Batches calls	Strict formatting	Uses less bandwidth, faster	Developer-friendly
Format	JSON, XML, and more	XML only	JSON, XML, HTML, plain text	JSON
Ease of use	Easy	Difficult	Easy	Medium
Use Cases	i. Command and action-oriented APIs ii. High performance communication in massive micro-services systems	i. Payment gateways ii. Identity management iii. CRM solutions iv. Financial and communication services v. Legacy system support	i. Public APIs ii. Simple resource-driven applications	i. Micro-services ii. Complex Systems iv. Mobile APIs

TABLE 2: API Architectures
Source: Comparing API Architectural Styles (2020)

2 Brief on application of API in Finance

APIs can be utilised in many ways: for example, in finance a centralised API may be utilised to identify and authenticate the users of privately and publicly administered open finance solutions. An in-depth analysis on Open Finance and APIs is presented in the BIS (2020) API scheme[4]. The report outlines that an open and standardised API scheme can facilitate interoperability for all entities participating in the open finance ecosystem. In particular, a central validator ('CV') is established and acts as an intermediary between financial institutions and third parties thereby eliminating any direct contact between them. In addition to ensuring that only necessary user information is received by the third parties that is sufficient for the transaction to occur, the CV also validates the requests received from the network by having all the involved parties certified.

In order for connections to occur in open finance it is vital that every user can be securely, efficiently, and remotely verified and authenticated. Such connections can be achieved through API providers like 'Plaid' that act as intermediaries between other financial apps and banks. Plaid provides information like bank balances securely to the app user, through their APIs via their seamless JSON requests. Additionally, APIs may not only be beneficial for data sharing but also for payment gateways, investment management and identity verification. Solutions like Addepar facilitate investment management for firms and advisors in the financial sector whilst others such as Trulioo ensure banks and financial institutions are KYC and AML compliant by providing API gateways for seamless identity verification.

[4] Further information on the analysed API scheme can be found in the BIS Paper "Enabling open finance through APIs", found here.

3 Microservices

API-based architecture allows for creation of an ecosystem of applications that are modular and reusable — which makes it ideal for microservices. Essentially, a microservice is a piece of software that exists within a larger application and performs a single, independent task, thus maintained separately and replaceable. Microservices break an application apart into packaged business capabilities (PBCs), connected via APIs. Microservices and APIs can be used complementary to each other whilst also operate independently from one another.

In practice, combining the described concepts together give rise to microservice-based application architecture. The development of a payment processing service encompasses many different functions/services that form part of the application. Such functions may include circulation of e-mails; contacting banks; executing transaction; reading, updating, and inserting information into a database. Although, all the listed services work independently, eventually they would all need an API to effectively communicate with each other. Also, other processes such as the creation and update of invoices, customer information and charges, may need to be independently called upon, without any human intervention.

Credit and other financial institutions may find it simpler to scale their software solutions by using a microservice architecture particularly because such *'applications are built as a suite of services, each running its own processes and communications'* (Deloitte, 2020). Financial institutions may use cloud platforms and other facilities to improve scalability whilst also leveraging the architecture of microservices to improve their own services internally; without having to restructure their entire software framework. The architecture of microservices may also benefit financial institutions in compliance, if such matters are encountered, they can be addressed separately without impacting the entire system.

4 Benefits and Risks

When evaluating new technology, benefits over the existing system are often revolutionary. However, it is vital to have a clear understanding and assessment of the risks before integrating innovation in business operations. Below is a non-exhaustive list of benefits and risk related to APIs and Microservices.

BENEFITS

Accessibility – Through APIs, applications and system components can communicate with each other on the Internet and in internal networks, making applications and services accessible for customers and partners.

Automation and Efficiency – APIs, as an emerging technology, can be used in a range of businesses, both internally and externally without human intervention, increasing productivity through their efficiency, seamless data sharing, and automation of existing systems.

Integration – A smooth and integrated user experience can be facilitated via APIs, which makes it possible for content to be interwoven and embedded in websites and applications, easily and comfortably accessible to the user.

Modularity and Cost Saving – Usage of APIs reduce costs, since developers can rely on third-party providers or internal APIs and focus their development efforts on other matters

Future-Ready – APIs can help support the adaptation process. Data Migration, data quality review and clean-up are being improved. API possess flexibility in providing services that are beneficial when an application has to be repurposed for an unanticipated use.

Flexibility – Microservice architecture[5] allows for changes to be made in a microservice without influencing the entire application, essentially replacing only what is necessary. Developers have consequently more flexibility to create lean, performant applications, in separate teams.

Threat Isolation – Since microservices are independent, threat isolation and bug fixes can be easily managed, without disrupting whole applications or services.

RISKS

Security – Data breaches present instances of API vulnerabilities which may lead to significant exposure of personal information, as was the case in the infamous Cambridge Analytica breach. Outdated APIs also present security risks of exploits over accounts and transactions.

Transparency – The build and implementation of the APIs is well-documented, which can give clues to criminal entities how to access internal databases, for example. Hackers can use the public information of the APIs, as points of entry to exploit applications.

Coding Bugs – Errors in code can have severe ramifications, with APIs providing sensitive information, such as passwords, personal information or other login data to malicious users.

Congestion – Constant API communication between microservices can cause a build-up of calls which may halt entire networks at times.

Compatibility – With constant updates to different microservices, users may have trouble with intra service compatibility.

To conclude, APIs and microservices have a wide variety of use cases, especially in functions related to business and finance. API technologies, although not risk-free, present a clear vision of efficient information sharing and continue to improve efficiency and automation within financial services.

Supplementary Reads…

Matt Hawkins (2020), The History and Rise of APIs. Forbes Technology Council. Available online.

Bank of International Settlements (BIS) (2020), Enabling open finance through APIs. *Report by the Consultative Group on Innovation and the Digital Economy (CGIDE).* Available online.

Microsoft Azure, Microservice architectural style. Available online.

Deloitte (2020), Open Banking through architecture re-engineering: A microservices based roadmap. Available online.

Altexsoft (2020), Comparing API Architectural Styles: SOAP vs REST vs GraphQL vs RPC. Available online.

Check our other **FinSights** and should you have any queries or wish to discuss your ideas within the context of our **MFSA Fintech Regulatory Sandbox**, contact us at **fintech@mfsa.mt**.

[5] Microservice architectural style is a microservice architecture consisting of small, autonomous services