THE OPEN PLATFORM CONCEPT

MARIUS MIHĂIȚĂ CORCIOVĂ[1], PATRICIA NEAGA[2]

Abstract: In computing, an open platform describes a software system which is based on open standards, such as published and fully documented external application programming interfaces (API) that allow using the software to function in other ways than the original programmer intended, without requiring modification of the source code. Using these interfaces, a third party could integrate with the platform to add functionality. An open platform implies that the vendor allows, and perhaps supports, the ability to do this. Using an open platform a developer could add features or functionality that the platform vendor had not completed or had not conceived of. An open platform allows the developer to change existing functionality, as the specifications are publicly available open standards. A service-oriented architecture allows applications, running as services, to be accessed in a distributed computing environment, such as between multiple systems or across the Internet. A major focus of Web services is to make functional building blocks accessible over standard Internet protocols that are independent from platforms and programming languages. [1]

Keywords: Open Platform, Application programming interfaces, Open standards

1. INTRODUCTION

An open platform adheres to the following principles in order to meet the need of all stakeholders:

1. Open Standards Based - The implementation should be based on agile open standards. Any willing party should be able to use these standards without charge to build an independent, compliant instance of the complete platform;

2. Shared Common Information Models - There should be a set of common information models in use by all instances of the open platform, independent of any given technical implementation;

3. Supporting Application Portability - Applications written to run on one platform implementation should be able to run with either trivial or no change on another platform that has been independently developed;

[1] *Ars Industrial SRL, Romania*
[2] *Innova Global Consulting SRL, Romania, office@innovacons.ro.*

4. Federatable - It should be possible to connect any implementation of the open platform to all others that were independently developed, in a federated structure, to allow the sharing of appropriate information and workflows between them; [2]

5. Vendor and Technology Neutral - The standards should not depend on particular technologies or require components from particular vendors. Anyone building an implementation of the open platform may elect to use any available technology and may choose to include or exclude proprietary components;

6. Supporting Open Data - Data should be exposed as needed (subject to good information governance practice) in an open, shareable, computable format in near to real-time. Implementors may choose to use this format natively in their persistence (storage) layer of the open platform itself or meet this requirement by using mappings and transformations from some other open or proprietary format;

7. Providing Open APIs - The full specification of the APIs (the means by which applications are connected to the platform) should be freely available;

8. Operability (as in DevOps) - The platform should support the principles of operability (this is all about the qualities of a system that enables applications to operate well throughout their full life cycle). Software systems which follow software operability good practice will tend to be simpler to operate and maintain, with a reduced cost of ownership, and almost certainly fewer operational problems. [2]

1. OPEN PLATFORM ARCHITECTURE

The specific architecture of an implementation of an open platform will vary from implementation to implementation. At a high conceptual level the architecture will take a form similar to that in the figure 1. [2]

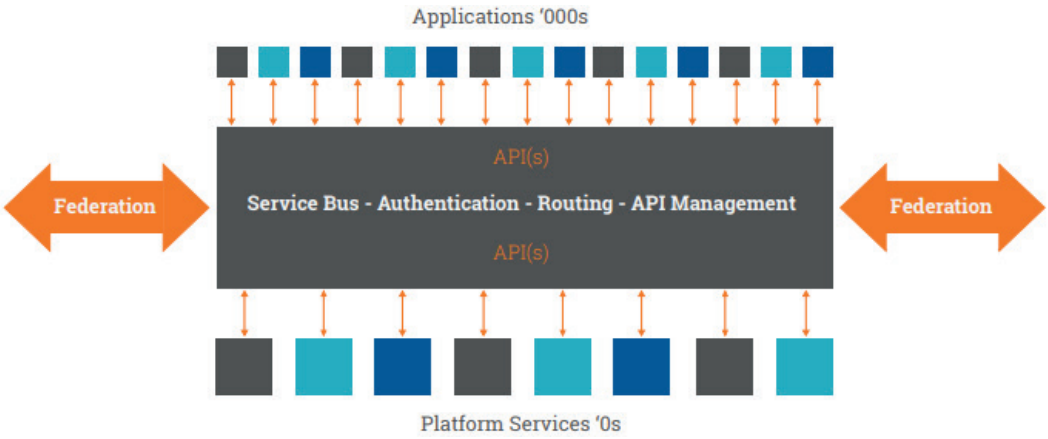

Fig.1. The block diagram of the Open Platform Architecture [2]

As a minimum the platform provider will provide a service bus that provides authentication, routing, and API management allowing applications to securely access

the APIs provided by platform service providers. Optionally, a platform provider may provide facilities to build or host applications and platform services; or they may provide some of the platform services themselves.

The platform provider may provide federation services to enable applications to access the data or services held on other compliant platforms. A typical platform will have the capability to support a large number of users running thousands of applications, and will offer a number of platform services (typically low double figures in number). [2]

1.1. Standards for an Open Platform

Open standards are critical to achieving the central aim of an open platform - that is, portability for data and substitutability for applications so that neither gets locked into a particular vendor's platform. This requires standards in two areas:

- Open Interface standards (APIs);
- Content standards.

Content standards can be applied to the way information is stored in an electronic record or to the payload of a message or API. Content standards typically describe how information describing a particular concept should be formatted, structured and coded to make it unambiguously computable.

An open platform appears to applications as nothing more than a set of open APIs which enable them to access platform services. The final goal for an open platform architecture is to have a complete set of platform microservices for all those things that can usefully be offloaded to the platform by the developers' app.

Each microservice will include a standard and stable open API expressed using the standards defined by the Open API Initiative. The number of platform microservices is potentially large (maybe some hundreds to a few thousand). However, the problem is a long-tailed one, which means an open platform with a small number of services (less than 10) can provide high utility to developers. [2]

1.2 Application programming interface

An application programming interface, or API, enables companies to open up their applications' data and functionality to external third-party developers, business partners, and internal departments within their companies. This allows services and products to communicate with each other and leverage each other's data and functionality through a documented interface.

Developers don't need to know how an API is implemented; they simply use the interface to communicate with other products and services. API use has surged over the past decade, to the degree that many of the most popular web applications today would not be possible without APIs.

An API is a set of defined rules that explain how computers or applications communicate with one another. APIs sit between an application and the web server, acting as an intermediary layer that processes data transfer between systems.

How an API works:

1. A client application initiates an API call to retrieve information—also known as a *request*. This request is processed from an application to the web server via the API's Uniform Resource Identifier (URI) and includes a request verb, headers, and sometimes, a request body.
2. After receiving a valid request, the API makes a call to the external program or web server.
3. The server sends a *response* to the API with the requested information.
4. The API transfers the data to the initial requesting application.

While the data transfer will differ depending on the web service being used, this process of requests and response all happens through an API. Whereas a user interface is designed for use by humans, APIs are designed for use by a computer or application.

APIs offer security by design because their position as middleman facilitates the abstraction of functionality between two systems—the API endpoint decouples the consuming application from the infrastructure providing the service. API calls usually include authorization credentials to reduce the risk of attacks on the server, and an API gateway can limit access to minimize security threats. Also, during the exchange, HTTP headers, cookies, or query string parameters provide additional security layers to the data. [3]

1.3 Types of APIs

Most application programming interfaces are web APIs that expose an application's data and functionality over the internet. The four main types of web API:

- Open APIs are open source application programming interfaces that can be accessed with the HTTP protocol. Also known as public APIs, they have defined API endpoints and request and response formats.
- Partner APIs are application programming interfaces exposed to or by strategic business partners. Typically, developers can access these APIs in self-service mode through a public API developer portal.
- Internal APIs are application programming interfaces that remain hidden from external users. These private APIs aren't available for users outside of the company and are instead intended to improve productivity and communication across different internal development teams.
- Composite APIs combine multiple data or service APIs. These services allow developers to access several endpoints in a single call. Composite APIs are useful in microservices architecture where performing a single task may require information from several sources. [3]

Traditionally, API referred to an interface connected to an application that may have been created with any of the low-level programming languages, such as Javascript. The modern API adheres to REST principles and the JSON format and is typically built for HTTP, resulting in developer-friendly interfaces that are easily

accessible and widely understood by applications written in Java, Ruby, Python, and many other languages. [3]

2. HYPERTEXT TRANSFER PROTOCOL

Hyper Text Transfer (HTTP) Protocol Request/Response includes Client and server exchange request/response messages, which uses the TCP protocol (figure 2). For Client and server exchange request/response, the typical port number is 80. [4]

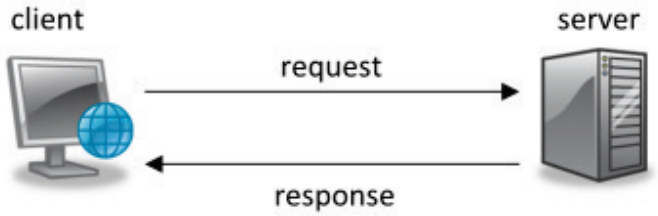

Fig.2. Basic block diagram of web application architecture which makes use of HTTP in it. [4]

HTTP Methods:

- GET - retrieves whatever information identified by the Request-URL;
- POST - sends data to the server for updates;
- PUT - requests that the enclosed entity stored under the supplied Request-URL;
- DELETE - requests the origin server to delete the resource identified by the Request-URL;
- HEAD - Similar to the GET method, except that the server must not return a message-body in the response;
- TRACE - Allows the client to see what received at the other end of the request chain and use that data for testing. [4]

CONCLUSIONS

- Creating an open platform is not about creating a single instance of a platform, but rather about creating an open platform ecosystem in which a number of platform vendors compete for business with each other. [2]
- Opening a platform can spur needs of user segments. At the same time, opening a platform typically reduces users' switching costs and increases competition among platform platform. [5]
- A platform is 'open' to the extent that: restrictions are not placed on participation in its development, commercialization or use, and any restrictions – for example, requirements to conform with technical standards or pay licensing fees – are reasonable and non-discriminatory, that is, they are applied uniformly to all potential platform participants.[5]

ACKNOWLEDGEMENTS

The project is founded by the EU under the Operational Program Competitivity (POC) from The European Regional Development Fund (FEDR), priority axis 2 – Information and Communication Technology (ICT) for a competitive digital economy, action 2.2.1. – Supporting the added value generated by the ICT sector and innovation in the field through cluster development.

REFERENCES

[1]. https://en.wikipedia.org/wiki/Open_platform
[2]. https://apperta.org/assets/Apperta_Defining_an_Open_Platform_April.pdf
[3]. https://www.ibm.com/cloud/learn/api
[4]. https://www.inspirisys.com/HTTP_Protocol_as_covered_in_RFCs-An_Overview.pdf
[5]. Eisenmann, Thomas R. and Parker, Geoffrey and Van Alstyne, Marshall W., *Opening Platforms: How, When and Why?* (August 31, 2008). This paper has been published under the same title as Chapter 6 in Platforms, Markets & Innovation (ed. Gawer, 2009) pp 131-162, Harvard Business School Entrepreneurial Management Working Paper No. 09-030, Available at SSRN: https://ssrn.com/abstract=1264012 or http://dx.doi.org/10.2139/ssrn.1264012