Building an API with a product mindset

Learn how you can use API as a product and its benefits

Table of Content

Executive Summary

APIs aren't just some back-office technical detail. They are much more than just that. They are interfaces that enable developers to repeatedly leverage data, functions, and applications to build new products and services. In other words, they're both products for the developers who build today's customer experiences and the mechanisms through which value is increasingly exchanged in modern economies.

This paper offers insights into:

- The main difference between API as project vs. API as product mindset
- Why API as a product will prove beneficial in the long term
- How a team can build API as a product mindset – a high-level view
- Key criteria in the process and qualities of an API product and the KPIs to track it

Getting back to basics

What is an API

Application Programming Interface (in short, API) is a mechanism that enables applications to talk among themselves, the way we humans use languages to communicate. If we know the language other humans speak, we can communicate with them. Similarly, API is the language in which applications communicate with each other.

While APIs have been around for a very long time, it was only in the early 2000s that the process to standardize them took shape. Most organizations began using the REST API standard, and its adoption helped develop the API ecosystem. Tools and frameworks were developed to help build and maintain these REST APIs.

That ecosystem is still evolving.

Today, the focus is on streamlining the API development and management process. However, APIs have been viewed solely from a project mindset in most cases. The key focus has always been to build and fulfill some front-end integration requirement or to support a new application that could use the existing infrastructure by building APIs around it.

The idea that APIs could be used as a product, both for internal and external usage, is still relatively new. It's time to shift from the "project mindset" to a "product mindset."

Before we delve into how this can be achieved, let's understand what a "product mindset" is all about.

Understanding API as a product mindset

Maximizing the value of API products involves technology and the operational lens through which the technology is viewed. API as a product mindset challenges the traditional view of API as a tool (which was used just for integration) to now look at API as a long-term value-generating medium. This mindset does not look at API just to meet a team's project requirements or for front-end integrations but requires that an API be managed like a product. Like any other product, it should be owned by a product manager and not by a project manager whose responsibility is to deliver on a list of requirements.

An API product mindset means designing and delivering APIs for long-term value at scale that will generate revenue, enable simple integration with the consumers, and evolve them over time to meet the dynamic customer requirements. APIs are built by keeping the product mindset at the core to ensure better reusability, exhibit simplicity, and possess the agility to adapt to market demands.

This is in contrast with approaching APIs as one-time projects or several discrete projects, where they deliver more limited value in terms of extensibility, longevity, and reach. A key part of an API product mindset also emphasizes that API teams should be structured to showcase the features of a modern agile product team.

Why choose the "API as a product" mindset /key consideration

Let's dig deeper to understand how we can leverage this mindset to our advantage and its key benefits. Creating API as a product will offer better longevity and agility to adapt to the ever-changing market demands.

Such a mindset emphasizes building and maturing the entire product over its life cycle instead of stressing success in individual phases. Adopting the product mindset for API will also launch the minimal viable product for developers. They can then start using it and provide early feedback, which can be incorporated in future iterations. This also ensures that we reach the market as early as possible to gain the first-mover advantage.

A product mindset also offers the added advantage of clarity. It forces you to think about what problem statement we are trying to solve. On the other hand, the project mindset will restrict that thought process and will ensure you build whatever has been requested. A clear problem statement in a product mindset will encourage the API team to devise multiple solutions and then select the one that can solve the problem best. This holds true even for internal APIs that are to be consumed within the organization. They should be considered technical products that multiple teams can use to solve the problem. They might not be monetized in terms of money, but a well-built API can still offer value in other aspects.

Imagine if there could be an authorization service to provide two-factor authentication. Such an API can be used by any team that needs to implement the two-factor authentication. But this will only be possible if the API is built with a product mindset. The developers would have considered all possible use cases that may be possible/needed. Although this will be consumed internally, it will ensure that no team will ever need to rebuild a two-factor authentication mechanism at their end. They can simply use the existing one. This kind of foresight ensures that if there are changes/security fixes to two-factor authentication, they can be managed at the API level, and other teams can continue using it.

Another scenario could be an OTP-based authorization service that can be utilized directly by retail and corporate banking applications to authorize any resource. While building such APIs may still not provide direct revenue in terms of money, it will expedite the pace of developing newer applications/features, thus reducing the time to market

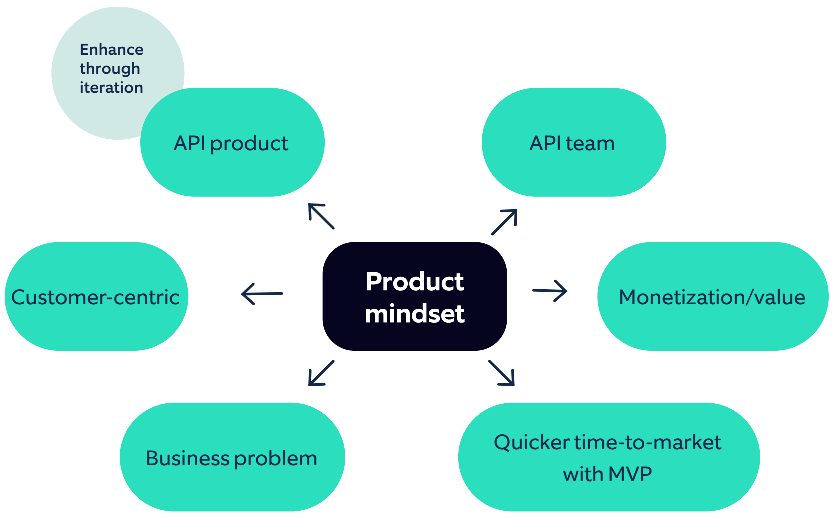
Banking on banks

Banks could also try to develop an API product based on their existing functions.

A good example is a banking KYC. All banks need a KYC solution and should invest independently in this functionality. They can also use this functionality to create an API product that could help other third-party systems fulfill authentication requirements. This is akin to features like Login with Google or Login with Facebook. Such a solution could greatly benefit other third-party providers like PayTM to use the existing KYC, thus opening a new revenue stream for banks.

A product mindset ensures that we think from the customer-first perspective. It enables us to design a simple solution, solves the problem in the best possible way, and is extensible to meet the constantly evolving demands. Another advantage of the product mindset is that it also ensures that we build a complete API package and bundle it to our customers. The API product should be self-sufficient to meet customer requirements.

Adopting the API product mindset also means that API development teams start looking at APIs as a product. This important shift in the mindset of the development teams will ensure simpler solutions and better products in the long term. The engineering team will start keeping the customer at the center of the solution, making the products more market-relevant.



How to build API with a product mindset/key consideration

The need for change

In most cases, APIs are designed to share data between two parties. They are looked at as a tool for back-end integration or simply to provide data to front-end applications. This introduces a bias in the thought and design process. Such a bias forces the design and development teams to focus on meeting the expectations of the front-end, and it completely ignores the capabilities of APIs. Consequently, while the front-end team will well understand such APIs, they will not have reusability as they will neglect the importance of using domain-specific language. Since the APIs are highly customized, they will require extensive documentation. They may also have poor versioning, which may not be considered part of the design and development process.

This type of outlook inhibits the product owners and developers from looking at APIs as a product rather than as standalone APIs. With a product mindset, the API design and development teams could align toward more homogeneous development, considering that the APIs could be exposed to multiple consumers in the future.

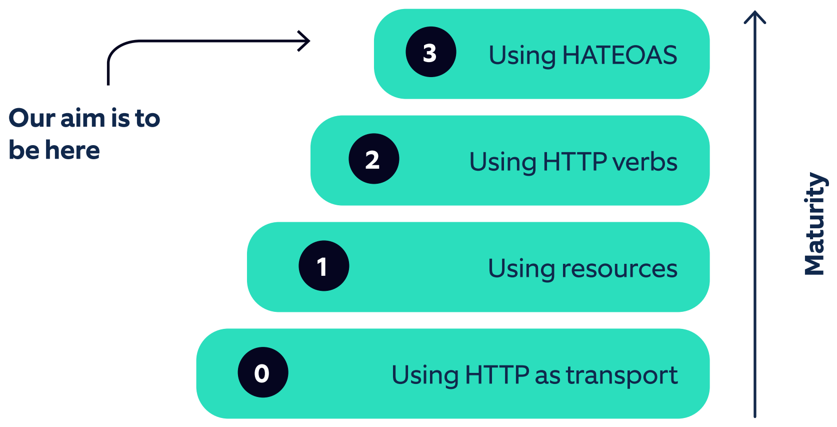
API maturity index

With the product mindset, API design and development teams must understand the API maturity levels to adapt to its homogeneous nature. Most of the APIs being built and consumed generally follow REST standards. Understanding the Richardson Maturity Model may be worthwhile to design and develop better APIs. It's a simple model to understand the maturity level of the API in terms of ease of use and consistency. A mature API will ensure better usability and quicker onboarding of new API consumers.

At the lowest level, the REST API will use HTTP simply as a transport layer and will not use any of the mechanisms of web. APIs at this level will be the hardest to understand, requiring extensive end-user documentation. At the other end of the spectrum, some APIs qualify for the highest level of maturity - these APIs will be easiest to understand because they will use HTTP verbs and different resources to perform various functions. The API at the highest maturity will also provide HATEOAS links to the next resources to be executed in the workflow. API at this level will require minimum documentation and can be self-explanatory to the end-users.

Another aspect of API maturity is how easy they are to understand and use. This might mean a lot of things to many consumers, but this means that the API should be self-explanatory at its core. The consumer should not need to know the APIs' internal working to understand their behavior.

One way to incorporate this mindset could be to look at APIs from the consumer's point of view and not from a developer's mindset. Considering factors like ease of use, understandability, consistency between various endpoints, and the use of domain-specific language as the centerpiece will help develop APIs from the consumer's perspective.



Documentation and versioning

Documentation and versioning are among the key features of a good product. Imagine buying a DIY cupboard from IKEA and not receiving the manual! You have all the pieces with you but still have no idea how to assemble them into a cupboard.

Another situation could be having the documentation for all the cupboard models but not knowing the model number to which the pieces belong. It will be a herculean task to assemble the cupboard in both cases. It's the same with APIs. Simply creating the APIs is not enough. They need to carry documentation that describes what they are supposed to do, the possible positive scenarios, and a list of possible error scenarios too.

A well-documented API will provide consumer delight – something that every product aims for. Documentation should not only be looked at as endpoint-specific documentation. It should also add a lot of value – for example, the API product team can document the most common use cases and the APIs required to perform these use cases. A lot of consumers will use these simple-to-follow tutorials as a starting point to implement some very common use cases before looking at more complex and innovative use cases as they get familiar with the product.

Versioning is as important as documentation

Imagine what would happen if API developers introduce a new mandatory field in payload and publishing the changes. The result of this change can be catastrophic. All consumer applications will start breaking. It's important to version any breaking changes so that the API consumers have enough time to migrate to the newer version of the API.

API documentation and versioning is usually the most neglected aspect in developing APIs for internal consumers. The accessibility or reachability of the API development team can play a big role in this. But this mindset needs to be changed when trying to look at APIs as a product rather than as a project. It is equally important to follow the best practices even when the APIs are being developed for internal consumers. This will be the key catalyst in shifting from a project to product mindset.

API-first approach

An API-first approach can help build a product mindset since the approach focuses on providing every functionality through APIs and treating APIs as the "first class citizen" for your product. In simple terms this approach suggests that any features created should be supported by APIs and nothing else.

APIs should cover the entire breadth of the business. The approach will ensure that developers and other stakeholders consider APIs as the primary product. Simply put, the API will be the first user interface of the application. This will also ensure that the API contracts are good enough to support the consumers of the API in meaningful and seamless integration.

This approach forces the developers and product owners to think about the uses cases they wish to support. This brings clarity to the feature and scope of the product. It will help developers to think ahead and design API contracts in a way that they are complete in every sense instead of waiting for feedback from UI/UX teams or other API consumers.

Tools that could help in the journey

A typical API journey involves multiple stages such as API design, implementation, testing, documentation, release/deployment, and versioning. Depending on the technology, many tools could be used during this journey. For instance, API implementation in Java is most popularly done by using Spring boot and Jersey framework, while the same is done through Express or NestJs in NodeJs.

API documentation is often independent of the technology being used. The most popular tools for documentation are OpenApi and Swagger. Both these tools work seamlessly across technologies. Other documentation tools include Spotlight and Readme. For API testing, one may also consider tools like Postman, RestAssured, and Karate.

API management tools are a good way to centralize your APIs' control, including analytics, access control, monetization, and developer workflows. The release and deployment teams have their own ecosystems of tools that can be leveraged to ensure smooth and robust DevOps capabilities for your APIs. Some popular APIM tools are Apigee, Microsoft Azure APIM, WSO2, and Red Hat 3scale.

KPIs that determine product mindset

API maturity

By combining the API maturity index, usage of domain-specific language, and documentation, we can create a KPI to understand the overall inclination towards a product mindset. The lower the API Maturity Index, the lower the API as a product mindset.

API monetization

APIs can enable new products, new business channels, and the digital transformation journey for an organization. This ultimately results in new revenue streams and other measurable values like meeting compliance and enabling rapid product development by reusing core APIs. An API built as a product will provide one or more ways to add value.

API strategy maturity

It can be another crucial parameter when judging the overall API as a product mindset. API strategy revolves around technology, business, and evolution. A product mindset will also mean that all 3 aspects are well-considered. Technology is the most common of all and is well understood across the developer community. The business focuses on promoting the API-first culture and ensures that the business revolves around APIs. Evolving APIs, their scalability, and adapting to technological changes are equally necessary to stay relevant. The higher the API Strategy Maturity score, the better the product mindset.

Usage metrics

For monetized APIs based on subscriptions, we can check the revenue being generated. For APIs that support internal products, the number of applications that consume them could indicate the usage metrics.

Conclusion

It is better to build APIs as a product rather than just as a project to be delivered. This mindset would ensure better value, agility, scalability, and improved quality of overall APIs. The product mindset approach will also inherit qualities like being customer-centric, a better approach towards complex business problems, higher monetization value, better API economy, and faster MVP to market.

This change will drive the shift in the organization’s mindset from looking at API as simple middleware to a product that can deliver value. The product mindset will drive a change in the organizational approach to adopt technologies that will ease the overall API development and management lifecycle. Such a shift in perspective will ensure a better API marketplace experience and a richer feature set for the end-users.

Additionally, API products can also allow monetization of data sources and help application developers design better products and use-cases for end users. Major banks like Nedbank and ICICI bank are coming up with their API Suites. It’s time to move away from a project mindset and start looking at APIs as a product that can add value and enhance customer experience. The key KPIs shared will help organizations track, correct, and maintain their product mindset to build a complete end-to-end API ecosystem that can serve all types of products, including future scalability.

About the authors

Ankur Sinha

Ankur has 9+ years of experience in the IT industry, working in multiple domains like BFSI and telecom. As a technical architect, he works on multiple technologies like Java and NodeJs. He has been actively designing API and API testing frameworks. He is primarily responsible for developing open banking APIs for one of the largest banks in Central and Eastern Europe.

Priyansu Das

Priyansu has 12+ years of experience in the IT industry, including 8+ years in BFSI. Currently, he leads the retail & SME division at one of the largest banks in Central and Eastern Europe. While he has worked as a delivery manager in multiple domains like BFSI, retail, e-commerce, and airlines, his primary experience is in open banking.

Editors
Deeksha Mamtani
Shailesh Dhaundiyal

Designer
Sri Harsha Dayanandachari