(https://nordicapis.com) NORDIC APIS

MENU SEARCH

Top Architectural Styles for APIs in 2023

Posted in Design(https://nordicapis.com/category/design/)

J Simpson(https://nordicapis.com/author/jsimpson/) February 9, 2023

(https://www.facebook.com/sharer.php?u=https%3A%2F%2Fnordicapis.com%2Ftop-architectural-styles-for-apis-in-2023%2F) (https://twitter.com/intent/tweet?url=https%3A%2F%2Fnordicapis.com%2Ftop-architectural-styles-for-apis-in-2023%2F&text=Top+Architectural+Styles+for+APIs+in+2023) (https://www.linkedin.com/shareArticle?url=https%3A%2F%2Fnordicapis.com%2Ftop-architectural-styles-for-apis-in-2023%2F&title=Top+Architectural+Styles+for+APIs+in+2023)

When you reassess your API strategy, you may find yourself rethinking your underlying API design style. Is REST still the best? Is GraphQL still valid, or was that just a fad? (spoiler alert: it's still a good choice!)

According to Postman's State of the API 2022 report (https://www.postman.com/state-of-api/api-technologies/#api-technologies), the most popular API architectural styles are:

- REST (89%)
- Webhooks (35%)
- SOAP (34%)
- GraphQL (28%)
- Websockets (26%)
- gRPC (11%)
- MQTT (9%)
- AMQP (8%)
- Server-Sent Events (6%)
- EDI (4%)
- EDA (3%)
- Other (1%)

With this in mind, let's take a look at the most popular API architecture styles for 2023. We'll outline the pros and cons of different API architecture styles, like REST vs. GraphQL, to help you decide which architectural style you should consider for your next API development project. Without further ado, here are the top API architecture styles of 2023.

The Top API Architecture Styles of 2023

REST

REST was one of the first standards for APIs, as laid out in Roy Fielding's dissertation (https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm), and it remains one of the best. REST is also the most popular, *by far*, making up a mammoth 89% of APIs.

REST stands for REpresentational State Transfer (https://nordicapis.com/all-you-need-to-know-about-rest-api-design/). It's a standard that lets users make API requests using simple HTTP commands (https://nordicapis.com/ultimate-guide-to-all-9-standard-http-methods/). REST's serverless architecture allows for easier decoupling between user and server, making it excellent for abstraction. It can also support a wide range of data formats, so it's quite flexible as well as scalable.

REST is not without its downsides, though. It returns a lot of metadata, for example, which can slow down response time and use up unnecessary bandwidth. This large format, along with issues regarding overfetching and underfetching, is one of the main reasons that Facebook developed GraphQL in 2012.

Webhooks

Sometimes we need real-time data interactions. Perhaps it's best for the application to automatically receive the latest updates instead of manually querying the API. Webhooks (https://nordicapis.com/your-guide-to-webhooks/) solve these situations for you.

A webhook can be used to send data whenever a new event occurs (https://nordicapis.com/why-your-api-needs-webhooks/). For example, imagine you've set up a new email list and want updates whenever you get a new subscription. A webhook can transmit that data, saving you the time, effort, and energy of making a query. This can also prevent wasting unnecessary resources making API calls when the data hasn't changed.

The main downside of webhooks is they're not as versatile or customizable as a fully-fledged API. It's simply a one-way data transmitter. There's also no indication if the receiver is working, so you won't necessarily know if the data being transmitted isn't getting where it's supposed to go. Webhooks have all manner of useful applications, though. You'll just want to have a monitoring solution in place to ensure you're notified if something goes wrong.

SOAP

SOAP is an XML-formatted structure standardized by W3C (https://www.w3.org/TR/soap/). It's the most widely used format for web services (https://nordicapis.com/whats-the-difference-between-microservices-and-web-services/) and useful for a vast number of different applications. SOAP is also excellent for privacy, allowing for encrypted messages and integrity inside each transaction.

However, SOAP is the biggest file format due to the verbose XML structure. SOAP is only appropriate when large bandwidths are available. It's also very rigid, though, so it's not very customizable, which significantly slows down its widespread adoption.

GraphQL

Facebook's GraphQL (https://nordicapis.com/rest-vs-graphql-a-side-by-side-comparison/) has a lot going for it. Its simple, JSON-like data structure is easy for non-technicians to understand and use. GraphQL is database-agnostic, meaning you can use it with virtually any database you can think of.

This makes your API eminently scalable and portable. These traits are key to an API's widespread adoption and ultimate success. GraphQL is also excellent for privacy, as it will expose certain functions while leaving the rest of the data secure. It also has detailed error logs, which are useful for large and unwieldy APIs.

GraphQL's not the best choice for complex queries, though, as too many nested queries can cause an overload and system shutdown.

WebSockets

If you want the advantages of webhooks but also want two-way communication, WebSockets (https://nordicapis.com/getting-started-with-websockets/) are the way to go. Like webhooks, WebSockets provide real-time data transmission. They're also able to receive data, making them the best of both worlds. WebSocket connections are much faster than HTTP (https://www.geeksforgeeks.org/what-is-web-socket-and-how-it-is-different-from-the-http/), as well, so they're a great pick if you're looking for the fastest connections possible.

However, WebSockets have problems, which explains why they're not more popular. They're especially rigid, for one thing, unlike other API architectural styles. If a WebSocket goes down, there's no load balancing or mechanism for reconnecting. They also don't support caching, unlike HTTP. Many proxy servers don't support WebSockets, either. This means you often have to have an HTTP streaming infrastructure in place in tandem with your WebSockets.

gRPC

For all of its many strengths, REST is not the fastest or most efficient API architecture out there. In fact, gRPC is **7 times faster** (https://blog.dreamfactory.com/grpc-vs-rest-how-does-grpc-compare-with-traditional-rest-apis] for receiving data and **10 times faster** for sending data. If you need lightning-fast data transmission, you might want to consider the [gRPC architectural style] (https://nordicapis.com/exploring-the-grpc-framework-for-building-microservices/). gRPC also has a native code compiler, removing the need for third-party tools. Based on HTTP 2, gRPC is also ideal for microservice environments (https://nordicapis.com/using-grpc-to-connect-a-microservices-ecosystem/) and useful for data streaming.

One thing to keep in mind, however, is that gRPC uses the Protobuf format (https://developers.google.com/protocol-buffers) instead of JSON. Protobuf is lightweight and efficient due to its data compression. It's not as understandable by humans, however. It's also not as widespread as JSON, either, so you might need to put a translator in place if you're integrating gRPC APIs into other API environments. gRPC is also more complicated and harder to implement than other API architectures. If you're looking for something that's fast and easy to set up, you'd do better to go with another API architectural style like REST.

MQTT

MQTT is a useful API architecture for IoT applications. It's popular for being lightweight and having low battery usage. It's also popular for its messaging accuracy (https://nordicapis.com/what-is-messaging-and-what-do-i-get-out-of-it/).

Unlike other API architectural styles, MQTT (https://nordicapis.com/7-protocols-good-for-documenting-with-asyncapi/) doesn't use HTTP for sending or receiving requests. Instead, it uses TCP/IP and a "publish-subscribe" model. This means the receiver needs an MQTT Broker to integrate with the device using MQTT, which acts as a sort of post office for MQTT messaging.

MQTT is limited in its functionality, though. It can only send and receive basic requests and file types. It's also not as interactive as REST, lacking the ability to POST, GET, DELETE, or PUT. With this in mind, MQTT is mainly appropriate for IoT applications ***specifically***. It's also suitable for situations with limited connectivity or low bandwidth.

AMQP

AMQP is an API architecture designed for microservices. AMQP stands for Advanced Messaging Queuing Protocol (https://www.amqp.org/) and is intended to be used whenever massive amounts of data or messaging are required.

AMQP (https://nordicapis.com/what-is-messaging-and-what-do-i-get-out-of-it/) is an amalgam of a few of the API architectural styles we've already discussed. It uses the publish/subscribe model of MQTT, for example. Its use of wire-level signals allows for continual data streams, making it similar to webhooks and WebSockets. It utilizes TCP like MQTT, too. AMQP is particularly secure as it uses the Quality of Service (QoS) protocol. It's also particularly adept at handling asynchronous messaging (https://nordicapis.com/why-asynchronous-apis-are-on-the-rise/), regardless of OS.

AMQP has its share of drawbacks, though. It's not backward compatible, for starters, so you'll only be able to integrate with the current version. It's also more complicated than HTTP. It also requires higher bandwidth than MQTT.

Server-Sent Events

Server-Sent Events (SSEs) are like a mix of webhooks and WebSockets but with many limitations removed. It offers the efficiency and connectivity of webhooks and WebSockets but also provides automatic reconnection should something go wrong.

SSEs use XHR to stream data over HTTP, making it a practical choice for projects requiring real-time data (https://nordicapis.com/why-enterprise-apps-need-real-time-data-streaming/), such as weather apps or stock services (https://nordicapis.com/10-real-time-stock-data-apis/). SSEs are fairly easy to set up and use and are accepted by most major browsers since they've been in use for years. But if it's not yet supported by the particular browser, it can be poly-filled using JavaScript.

The downside is SSEs are not that prevalent so there aren't that many libraries available, unfortunately. They support a limited range of data, too, and only transmit UTF-8. Additionally, browsers can only support a maximum of six SSEs simultaneously. Finally, SSEs are only uni-directional, so their usefulness is limited and specific, usually to real-time apps.

EDI/EDA

EDI stands for Electronic Data Interchange, and EDA is shorthand for event-driven architecture. Both offer powerful capabilities for data transfer in different ways.

EDI (https://nordicapis.com/what-is-the-difference-between-edi-and-api/) has been around since the late '70s. It uses a P2P network to transmit large quantities of standardized data. It's pretty straightforward to use and widely available, thanks to its longevity. Like AMQP, EDI isn't backward compatible, unfortunately. Three versions of EDI are available, and each only works with the same version.

EDI is also much more limited in its scope and usefulness, unfortunately. It's more like a tunnel than an exchange, as it can only connect two users. It's also relatively expensive and cumbersome to implement.

EDA (https://nordicapis.com/4-business-benefits-of-an-event-driven-architecture-eda/) has also been around since the '70s. It's a fairly simple concept, similar to webhooks in many ways. When something happens, something is triggered. EDA is also inherently scalable – events can be queued and retrieved at a later date. Subscriptions can be turned on and off, as well, making EDA useful for "pub/sub" or "publish/subscribe" applications.

You always need to be careful with real-time applications, though, as there are no built-in guardrails. If EDA is connected to a service that charges per transaction, this can quickly become a costly mistake. It can also cause performance issues, depending on the amount of traffic at a particular time.

Final Thoughts On API Architecture

As you can see, there's much more to APIs than just REST, although that remains the leader by a longshot. It's safe to say that as things like microservices, headless and serverless solutions become more prevalent, we will also see increased adoption of some of these alternate API architecture styles.

Even if you stick with REST, it's still good to know what's out there and what your options are for API development. Maybe you're developing an IoT portal or dashboard, for example. In that case, REST might not be the best for that particular application, and you might be better served using MQTT or AMQP instead.

When choosing your API architecture style, you'll want to think about what kind of libraries and third-party applications are available. You'll also want to consider how widely used it is, as you'll want others to be able to use your API, as well.

One thing is for certain: the API landscape is growing and evolving. It'll be fascinating to watch what the next 12 months have in store!

AMQP (https://nordicapis.com/tag/amqp/), API architecture (https://nordicapis.com/tag/api-architecture/), API design style (https://nordicapis.com/tag/api-design-style/), asynchronous messaging (https://nordicapis.com/tag/asynchronous-messaging/), bandwidths (https://nordicapis.com/tag/bandwidths/), caching (https://nordicapis.com/tag/caching/), data formats (https://nordicapis.com/tag/data-formats/), data streams (https://nordicapis.com/tag/data-streams/), EDA (https://nordicapis.com/tag/eda/), EDI (https://nordicapis.com/tag/edi/), GraphQL (https://nordicapis.com/tag/graphql/), gRPC (https://nordicapis.com/tag/grpc/), guardrails (https://nordicapis.com/tag/guardrails/), headless solutions (https://nordicapis.com/tag/headless-solutions/), HTTP 2 (https://nordicapis.com/tag/http-2/), HTTP commands (https://nordicapis.com/tag/http-commands/), IoT (https://nordicapis.com/tag/iot/), JSON (https://nordicapis.com/tag/json/), libraries (https://nordicapis.com/tag/libraries/), load balancing (https://nordicapis.com/tag/load-balancing/), messaging accuracy (https://nordicapis.com/tag/messaging-accuracy/), metadata (https://nordicapis.com/tag/metadata/), microservices (https://nordicapis.com/tag/microservices/), MQTT (https://nordicapis.com/tag/mqtt/), overfetching (https://nordicapis.com/tag/overfetching/), P2P network (https://nordicapis.com/tag/p2p-network/), performance issues (https://nordicapis.com/tag/performance-issues/), poly-filled (https://nordicapis.com/tag/poly-filled/), proxy servers (https://nordicapis.com/tag/proxy-servers/), Quality of Service (https://nordicapis.com/tag/quality-of-service/), Real-time data (https://nordicapis.com/tag/real-time-data/), rest (https://nordicapis.com/tag/rest/), scalability (https://nordicapis.com/tag/scalability/), Server-Sent Events (https://nordicapis.com/tag/server-sent-events/), serverless solutions (https://nordicapis.com/tag/serverless-solutions/), soap (https://nordicapis.com/tag/soap/), TCP/IP (https://nordicapis.com/tag/tcpip/), third-party applications (https://nordicapis.com/tag/third-party-applications/), underfetching (https://nordicapis.com/tag/underfetching/), UTF-8 (https://nordicapis.com/tag/utf-8/), WebHooks (https://nordicapis.com/tag/webhooks/), WebSockets (https://nordicapis.com/tag/websockets/)

(http[illegible]is.com/author/jsimpson/)

J Simpson

(https://nordicapis.com/author/jsimpson/)
J. Simpson lives at the crossroads of logic and creativity. He writes and researches tech-related topics extensively for a wide variety of publications, including Forbes Finds. He is also a graphic designer, journalist, and academic writer, writing on the ways that technology is shaping our society while using the most cutting-edge tools and techniques to aid his path. He lives in Portland, Or.

(https://masteringmodernity.wordpress.com) (https://www.twitter.com/for3stpunk)

‹ Why Traditional API...(https://nordicapis.com/why-traditional-api-monetization-needs-to-evolve/)

What Will PSD3 Mean for Open... › (https://nordicapis.com/what-will-psd3-mean-for-open-banking/)

0 Comments Piyali Das ▾

Start the discussion…

Share Best Newest Oldest

Be the first to comment.

Subscribe Privacy Do Not Sell My Data

Latest Posts

5 Pillars of Modern API Maturity

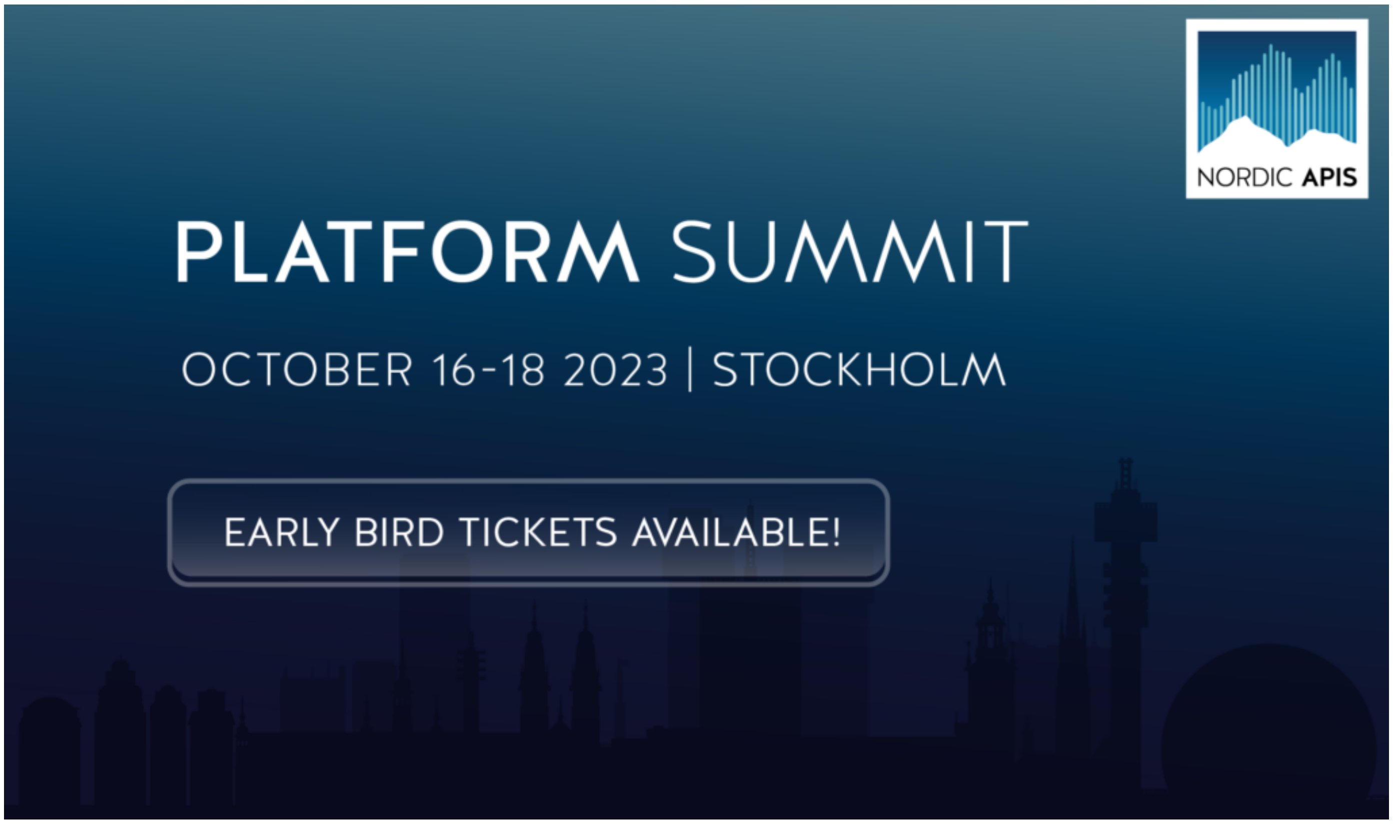
Gemma Sindall April 19, 2023
(https://nordicapis.com/5-pillars-of-modern-api-maturity/)

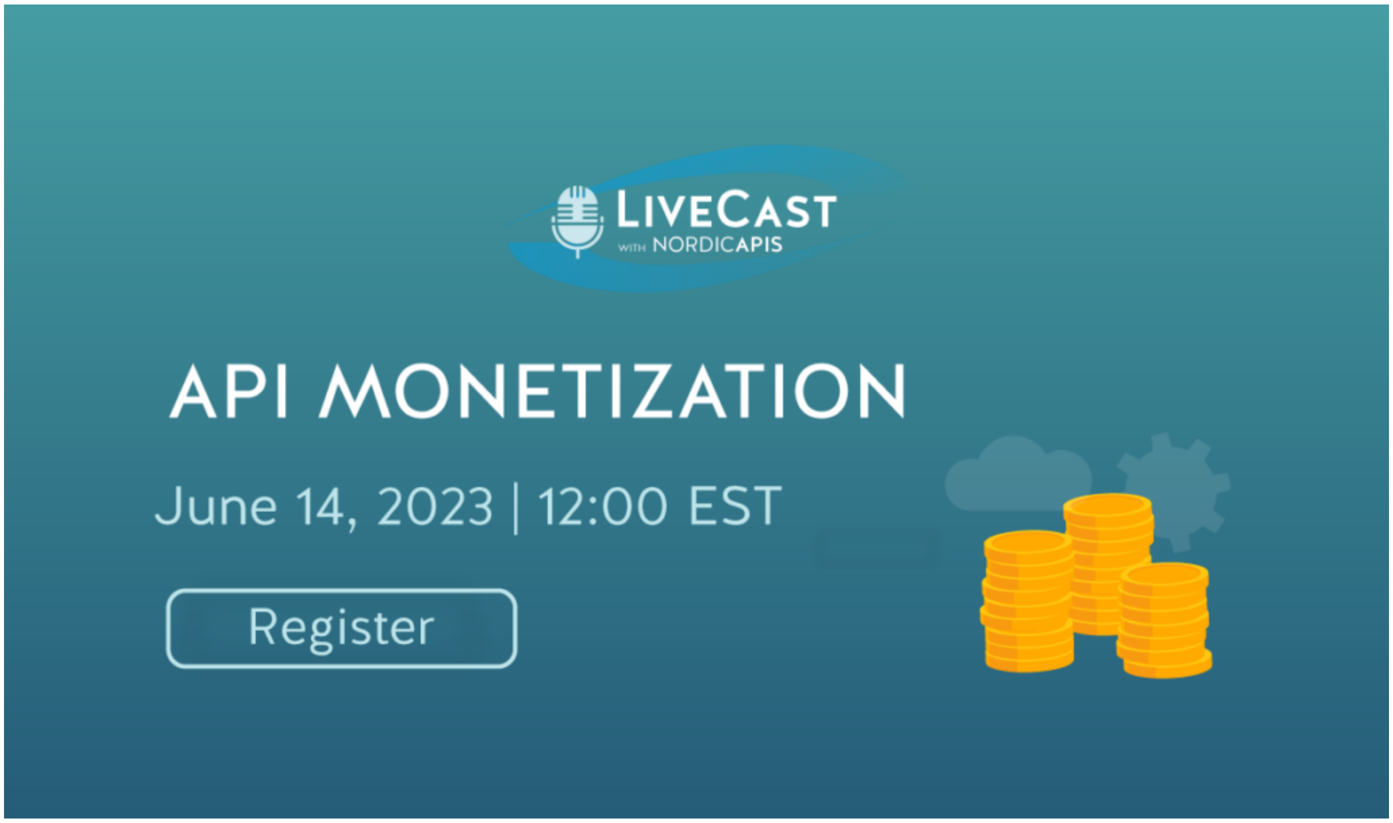
8 Tips For Monetizing Your API

J. Simpson April 18, 2023
(https://nordicapis.com/8-tips-for-monetizing-your-api/)

Using tRPC for TypeScript-Enabled APIs

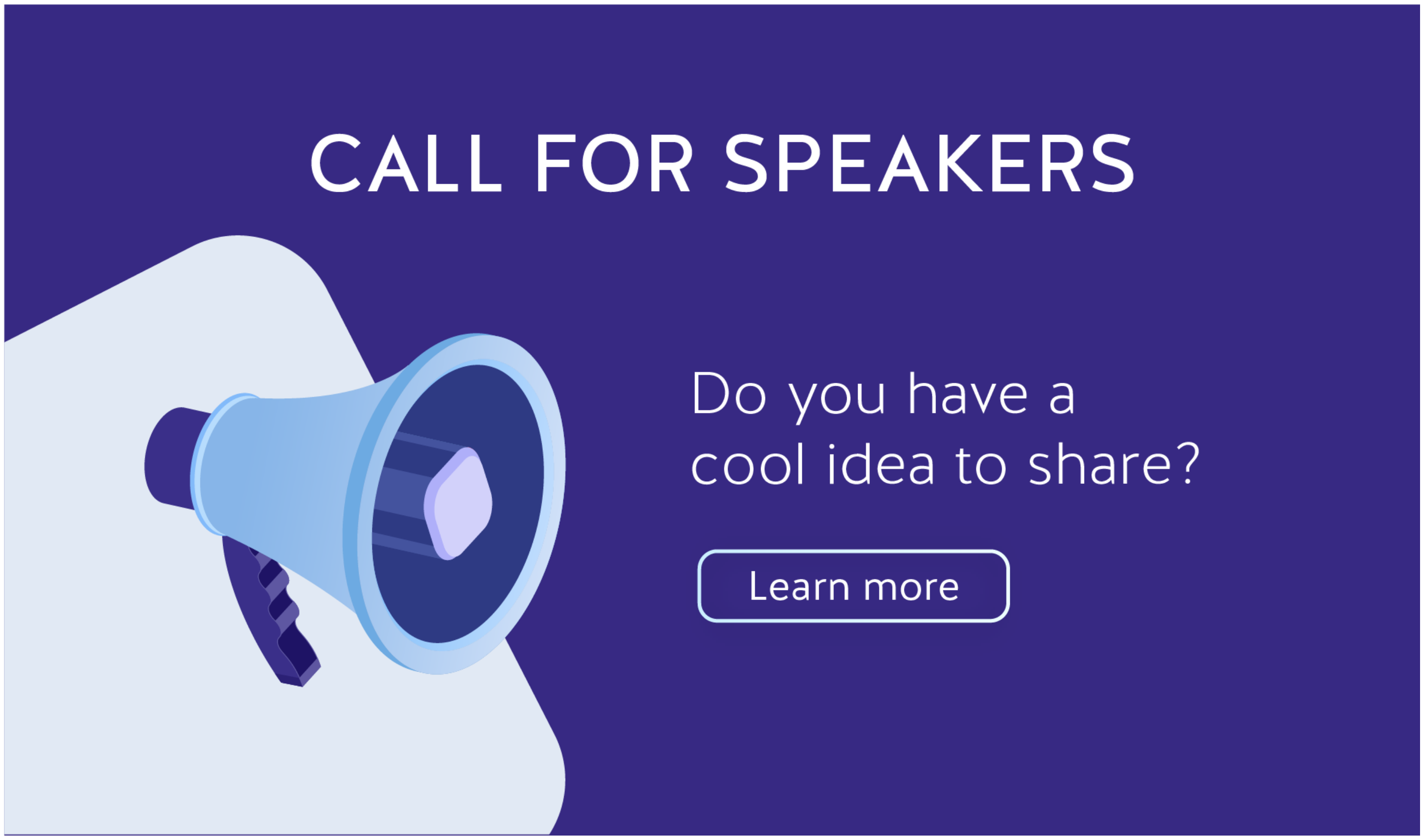
Kristopher Sandoval April 13, 2023
(https://nordicapis.com/using-trpc-for-typescript-enabled-apis/)



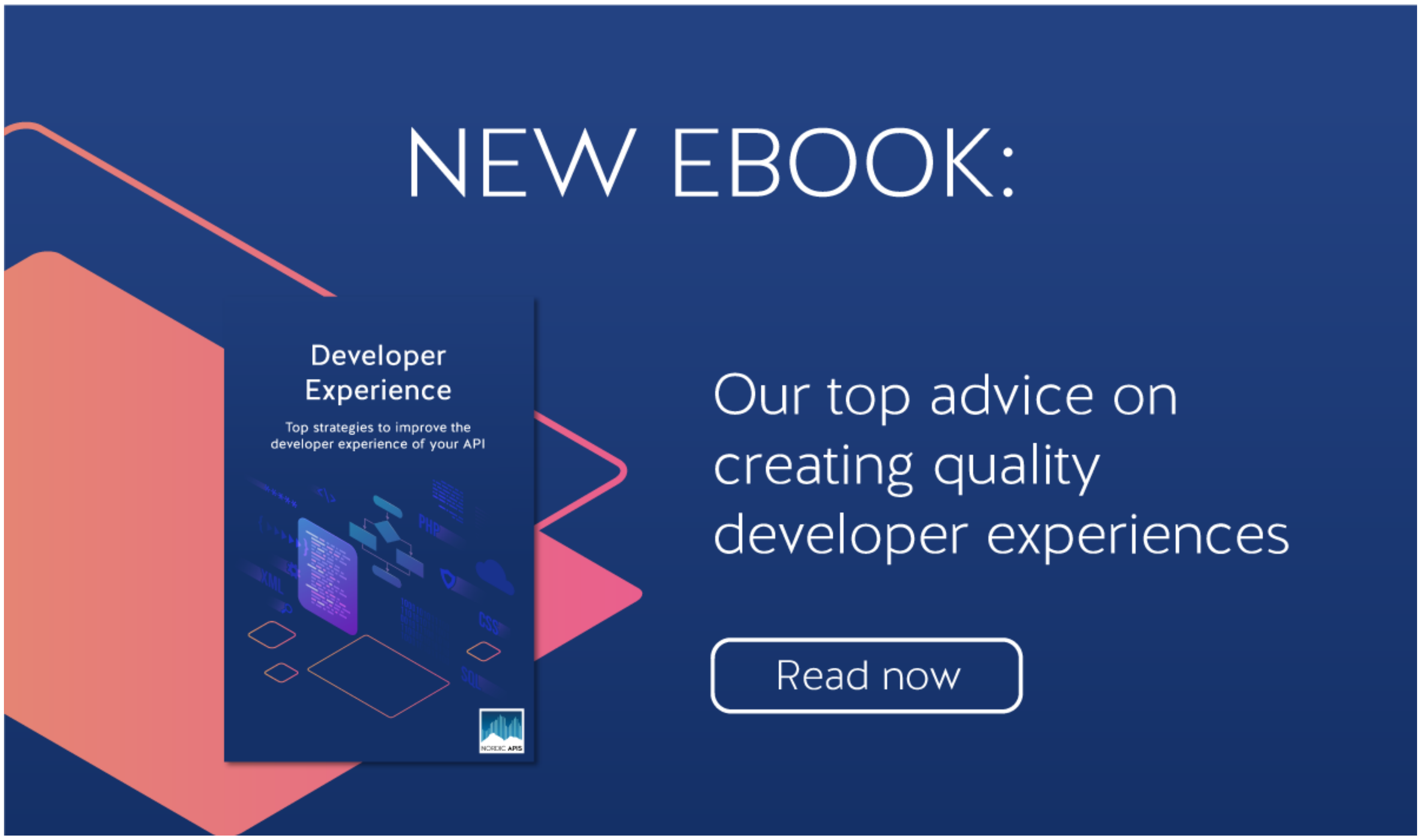


(https://nordicapis.com/events/api-monetization/)

(https://curity.io/gartner-digital-identity-hype-cycle/?utm_source=nordicapis&utm_medium=sidebar-widget&utm_campaign=nordicapis)



(https://nordicapis.com/call-speakers/)


NEW EBOOK:
Developer Experience
Top strategies to improve the developer experience of your API
Our top advice on creating quality developer experiences
Read now

(https://nordicapis.com/ebooks/developer-experience/)

Smarter Tech Decisions Using APIs

High impact blog posts and eBooks on API business models, and tech advice

Connect with market leading platform creators at our events

Join a helpful community of API practitioners

API Insights Straight to Your Inbox!

Can't make it to the event? Signup to the Nordic APIs newsletter for quality content. High impact blog posts on API business models and tech advice.

tim@apple.com

Subscribe

Join Our Thriving Community

Become a part of the world's largest community of API practitioners and enthusiasts. Share your insights on the blog, speak at an event or exhibit at our conferences and create new business relationships with decision makers and top influencers responsible for API solutions.

Write
(https://nordicapis.com/create-with-us/)

Speak
(https://nordicapis.com/call-speakers/)

Sponsor
(https://nordicapis.com/about/contact-us/)

Events

Events Calendar (https://nordicapis.com/api-event-calendar/)

Best Public API of 2021 (https://nordicapis.com/best-public-api-of-2021/)

Curity Webinars (https://curity.io/resources/webinars/)

Blog

Blog (/blog)

Business Models (https://nordicapis.com/category/business-models/)

Marketing (https://nordicapis.com/category/marketing/)

Platforms (https://nordicapis.com/category/platforms/)

Security (https://nordicapis.com/category/security/)

Strategy (https://nordicapis.com/category/strategy/)

Design (https://nordicapis.com/category/design/)

Open Banking (https://nordicapis.com/category/open-banking/)

Resources

eBooks (/api-ebooks/)

Blog Submission Guidelines (https://nordicapis.com/create-with-us/)

Call for Speakers (https://nordicapis.com/call-speakers/)

Code of Conduct (https://nordicapis.com/code-of-conduct/)

About

About (https://nordicapis.com/about/)

Nordic APIs for Women (https://nordicapis.com/nordic-apis-for-women/)

Volunteer (https://nordicapis.com/student-volunteer/)

Privacy Policy (https://nordicapis.com/nordic-apis-privacy-policy/)

Contact us (https://nordicapis.com/about/contact-us/)

Social

://twitter.com/nordicapis) (https://www.linkedin.com/company/nordic-apis) (https://www.facebook.com/NordicAPIs) (https://www.youtube.com/user/nordicapis)