

# How Developer Portals Help You **Thrive in the API Economy**



March 2023  
GUIDE



# Table of **contents**

**3** Introduction

**10** Consultants: The Missing  
Piece in Your API Puzzle

**4** Understanding Developer  
Portals: Key Concepts &  
Statistics

**11** A Proven, 7-Step Approach to  
Build Your Developer Portal

**5** API Adoption: Usability and  
the Birth of Developer Portals

**16** Developer Portal Workflows &  
Reference Architecture

**6** Why Developer Portals?

**19** Developer Portal Case Studies

**8** How I Learned to Stop  
Worrying and Build My Own  
Developer Portal

**21** About Our Author





# Introduction

API economy is not a new term. In fact, it has been a staple of the tech industry for more than a decade, as countless businesses have utilized APIs to exchange data and services with other applications to accelerate development cycles, improve the user experience, and increase revenue through monetization.

However, recently the industry has witnessed the transition of non-digital products building up their APIs to get closer to their customers. The API economy advantages have been envisaged, propagated, deconstructed, and evangelized across the IT spectrum. Yet, there must be a reason why for every success story beyond API transformation, there are many more stories of falling short.

Of course, there are some cases where the reasons are well known and simple to identify because the API was poorly designed or insecure. But what if you created the 'perfect' API and yet the monetization still isn't going exactly as planned? It's likely because the biggest challenge to the API product is the lack of API adoption.

Thankfully, one of the greatest gifts to overcome the adoption challenge has been present since the dawn of the API economy: Developer Portals. Keep reading for our expert point of view on how to design, develop, and maintain a world-class developer portal to give your API the best chance of success in the increasingly competitive API economy.



# Understanding Developer Portals: Key Concepts & Statistics

**Application Protocol Interface (API):** “An application programming interface (API) is a way for two or more computer programs to communicate with each other. It is a type of software interface, offering a service to other pieces of software.” ([Wikipedia](#))

**API Economy:** The API economy refers to the set of business models and practices designed around the use of APIs in today’s digital economy. It involves the exposure of an organization’s digital services and assets through APIs in a controlled way. ([TechTarget](#))

**Developer Portal:** A developer portal is a central information hub about API products, documentation, and microservices. These portals give API teams access to support in the form of information concerning the APIs. Developers can get different response shapes, look at the API documentation, and get support when needed. ([Cortex](#))

**Technical Writing:** “Writing or drafting technical communication used in technical and occupational fields, such as computer hardware and software, architecture, engineering, chemistry, aeronautics, robotics, finance, medical, consumer electronics, biotechnology, and forestry. A technical writer’s primary task is to communicate technical information to another person or party in the clearest and most effective manner possible.” ([Wikipedia](#))

**Documentation:** “Technical documentation encompasses articulated and straightforward

technical user guides. Technical documents explain the functionality of a product in simple terms. Writing technical documentation requires product understanding and user feedback. Many aspects involved in technical documents are exhaustive and involve intricate details.” ([Technical Writer HQ](#))

## 4 Main API Types

- Open APIs: Any developer can access.
- Partner APIs: Only authorized developers may access.
- Internal APIs: Only internal teams may access.
- Composite APIs: Combine multiple APIs. ([HubSpot](#))

## 3 Common API Architectures

- REST: A collection of guidelines for lightweight, scalable web APIs.
- SOAP: A stricter protocol for more secure APIs.
- RPC: A protocol for invoking processes that can be written with XML (XML-RPC) or JSON (JSON-RPC)

The API Management market size is expected to reach a value of around \$41.5B by 2030, growing at a compound annual growth rate (CAGR) of about 34.5% from 2022 to 2030. ([Custom Market Insights](#))

There are currently almost 3-million API repositories on GitHub! ([GitHub](#))

Nearly 90% of developers use APIs in some way. ([Nordic APIs](#))

# *API Adoption:* Usability and the Birth of **Developer Portals**

Defining the usability of an API product is hard. The API is like a universal tap, which pours out any beverage of your choice, from the available options. Seems extremely useful, right?

However, at what point does your customer know what options you have? Is your customer aware that your tap accepts requests with parameters and they can get a whole fat double shot of roasted Colombian coffee? Maybe they aren't aware that you also have a complementing universal grill that can give you the panini of your choice. And not only that, but customers can then use both the products to build a breakfast of champions.



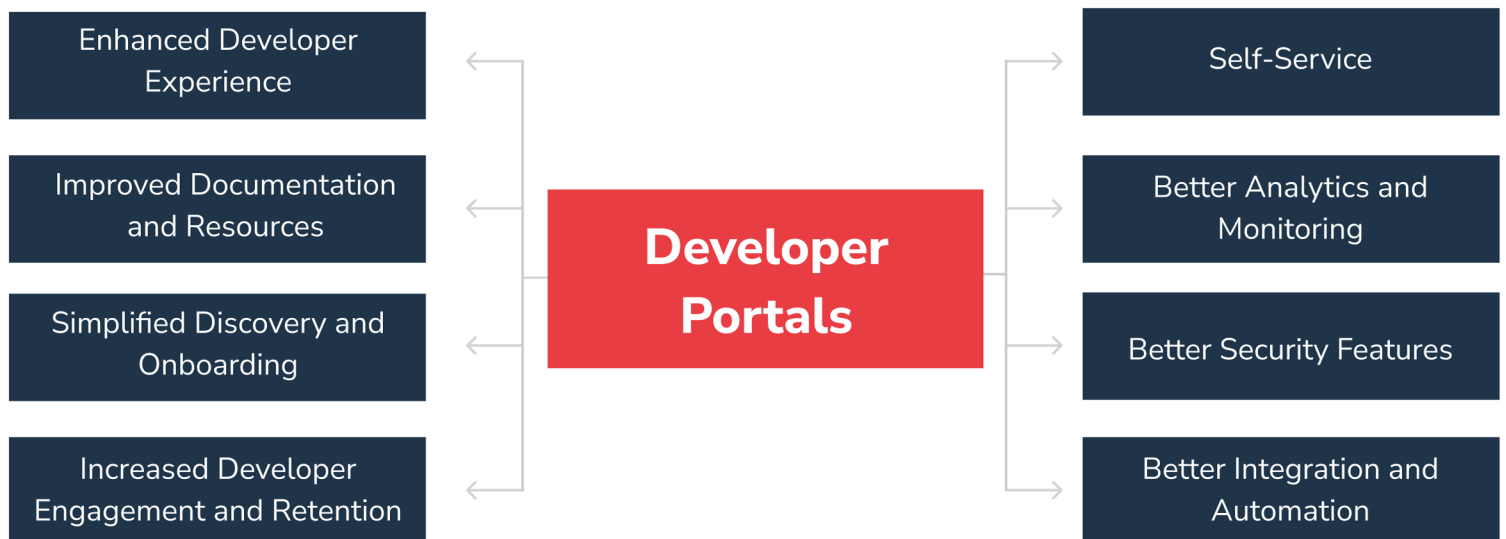
If you followed the previous example, you just witnessed that APIs, unlike UIs, cannot be intuitive. However, what your APIs can be are better documented. And not just in silos, but as a part of the whole product family. And that is how developer portals were born.

# Why Developer Portals?

Developer portals are a central location for developers to discover, learn about, and interact with APIs. Since developers are also your customers for your API products, this is where your customers decide how to do all the wonderful things you envisaged folks would do with your offering.

Developer portals include documentation, code samples, and interactive tools for testing and troubleshooting. Additionally, developer portals often include features such as forums and community support, making it easier for developers to connect with others and share knowledge. Overall, developer portals are helping to simplify the process of consuming APIs and making it more accessible for developers of all skill levels to build innovative new applications.

Some of the top reasons why developer portals work are:





**Simplified discovery and onboarding:** Developer portals provide a central location for customers to discover and learn about APIs, simplifying the process of finding the right API for their needs.

**Improved documentation and resources:** Developer portals include detailed documentation, code samples, and other resources to help customers understand and use APIs more effectively.

**Enhanced developer experience:** Developer portals often include interactive tools and features, such as forums and community support, to help customers connect with other developers and share knowledge.

**Self-service:** Developer portals enable customers to manage their own API keys and usage, providing more flexibility and control over their usage of the API.

**Better analytics and monitoring:** Developer portals often include analytics and monitoring features to help customers understand how their APIs are being used and identify any potential issues.

**Better security features:** Developer portals include features like OAuth2 and OpenID Connect for authentication and authorization, and other security features like rate limiting, to ensure the security and privacy of the data.

**Better integration and automation:** Developer portals include features like webhooks, Swagger/OpenAPI specifications, and SDKs, that allow for easy integration and automation of the API.

**Increased developer engagement and retention:** By providing a central location for developers to discover, learn about, and interact with APIs, Developer portals can help increase developer engagement and retention. This can lead to a more active and supportive developer community, which can ultimately help drive the success of the API and the overall API ecosystem.

# How I Learned to *Stop Worrying* and **Build My Own** Developer Portal

Some of the best known developer portals for the most widely used API products were built in-house. So, the next logical step seems to be building your own. Building your own developer portal can be a complex process, but it can also be broken down into the following key steps:

**1. Define your API strategy:** Before building your developer portal, it's important to have a clear understanding of your API strategy and the goals you want to achieve with your API. This includes understanding the types of developers who will be using your API, the types of applications they will be building, and the value your API will provide.

**2. Choose a platform:** You can build a platform from scratch, or use one of the several platforms available for building developer portals, including open-source options such as Docusaurus, as well as commercial options such as Readme. Choose a platform that best meets your needs and budget.

**3. Create your API documentation:** Once you have chosen a platform, you can begin creating your API documentation. This should include detailed information on how to use your API, including code samples and interactive tools for testing and troubleshooting.

**4. Build community features:** A developer portal should also include features that encourage community engagement and support, such as forums and social media integration.

**5. Create a developer account and API key management system:** A developer portal should also include a system for creating developer accounts, managing API keys and usage, and providing analytics and monitoring features.

**6. Test and launch:** Once your developer portal is complete, it's important to test it thoroughly before launching it to the public.

**7. Promote and maintain:** Once the portal is live, keep promoting the portal and make sure that the portal is up-to-date.

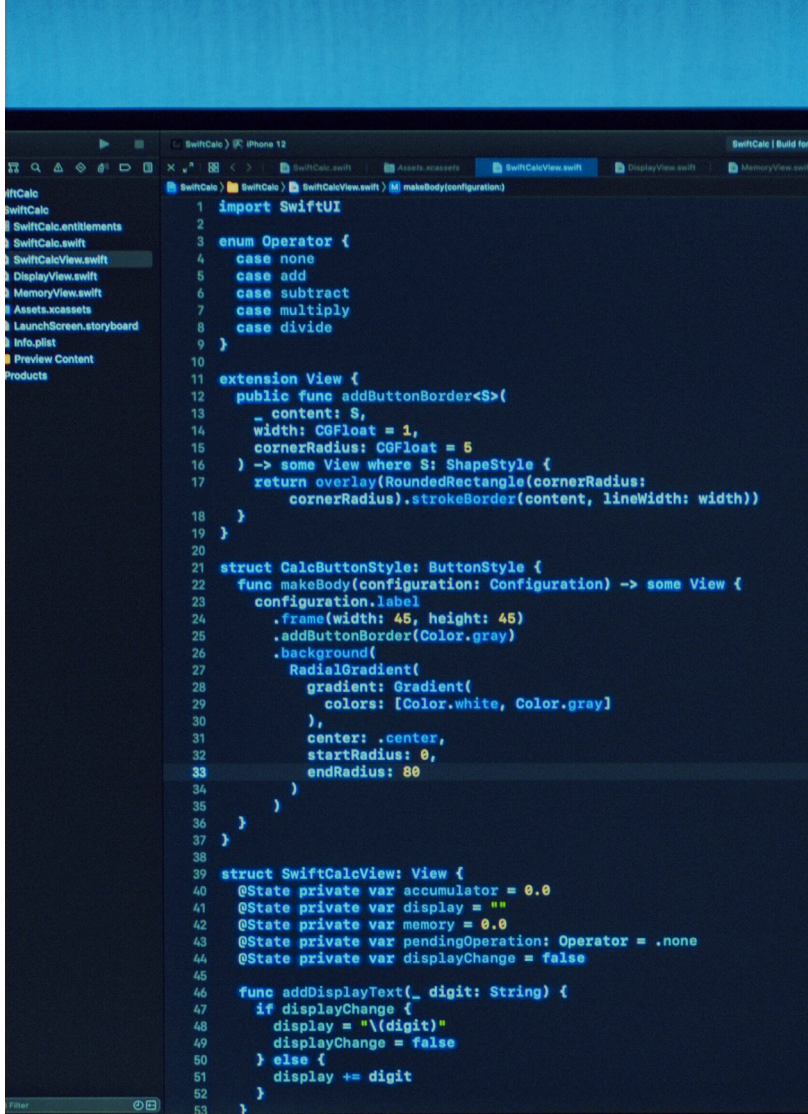
## **Pro Tip:**

Building a developer portal is an iterative process and you must monitor the usage and feedback to improve the portal and make it more developer friendly.

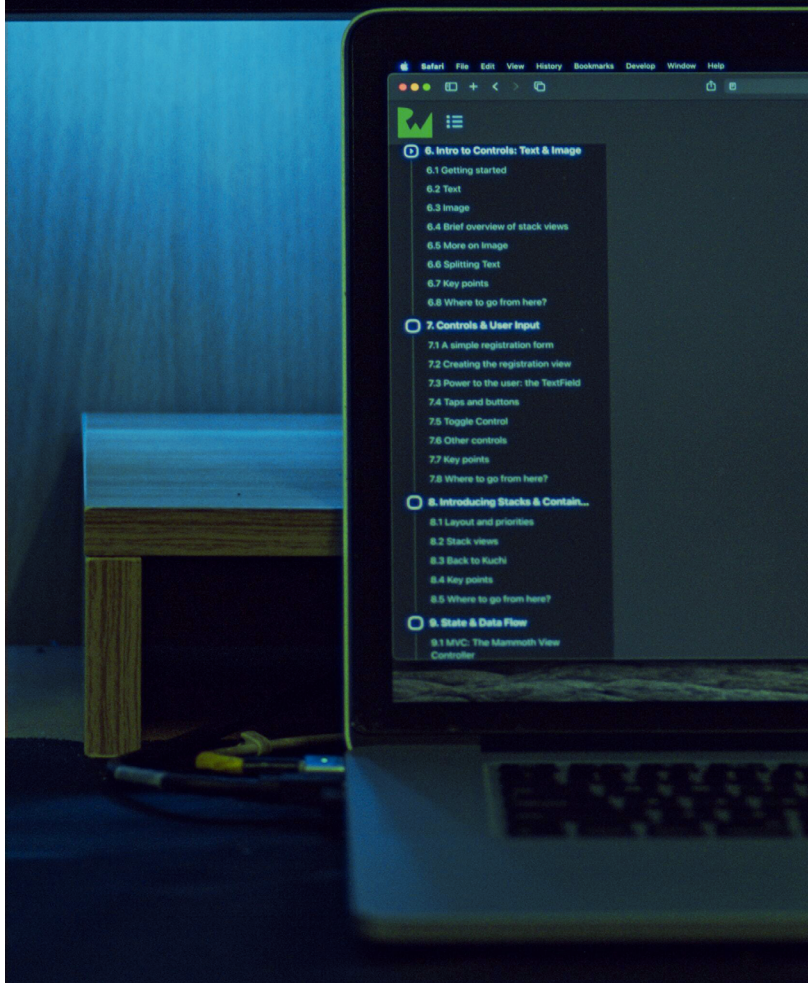


However, building your own developer portal can also open a can of worms. It's worth considering the following challenges and the resources you have available before deciding to build your own developer portal.

- **Unsustainable costs:** Building a developer portal from scratch can be a costly endeavor, especially if you choose to build it in-house. This can include costs for development, hosting, and maintenance.
- **Time-consuming:** Building a developer portal can be a time-consuming process, especially if you are starting from zero. It may require significant resources and effort to create the necessary documentation, interactive tools, and community features.
- **Limited scalability:** If you build your own developer portal, it may be more difficult to scale as your API usage grows. This can be a problem if you expect your API to be widely adopted.
- **Limited or unneeded features and functionality:** You may be limited in terms of the features and functionality that you can include. This can be a problem if you need advanced features such as analytics, monitoring, or security. Also, you may waste resources adding a feature that you never develop products for, or your customers never use.
- **Lack of expertise:** Building a developer portal requires a certain level of technical expertise. Without this expertise, the portal may be difficult to build, maintain, and improve.
- **Lack of maintenance:** Building a developer portal is one thing and maintaining it is another. It requires continuous maintenance in order to make sure that the portal is up-to-date and that the content is accurate.
- **Limited support:** Building your own developer portal may also limit the level of support you can provide to developers. This can be a problem if you expect your API to be widely adopted and need to provide a high level of support to developers.



```
1 import SwiftUI
2
3 enum Operator {
4     case none
5     case add
6     case subtract
7     case multiply
8     case divide
9 }
10
11 extension View {
12     public func addButtonBorder<S>(<
13         _ content: S,
14         width: CGFloat = 1,
15         cornerRadius: CGFloat = 5
16     ) -> some View where S: ShapeStyle {
17         return overlay(RoundedRectangle(cornerRadius:
18             cornerRadius).strokeBorder(content, lineWidth: width))
19     }
20 }
21
22 struct CalcButtonStyle: ButtonStyle {
23     func makeBody(configuration: Configuration) -> some View {
24         configuration.label
25         .frame(width: 46, height: 46)
26         .addButtonBorder(Color.gray)
27         .background{
28             RadialGradient{
29                 gradient: Gradient{
30                     colors: [Color.white, Color.gray]
31                 },
32                 center: .center,
33                 startRadius: 0,
34                 endRadius: 80
35             }
36         }
37     }
38 }
39
40 struct SwiftCalcView: View {
41     @State private var accumulator = 0.0
42     @State private var display = ""
43     @State private var memory = 0.0
44     @State private var pendingOperation: Operator = .none
45     @State private var displayChange = false
46
47     func addDisplayText(_ digit: String) {
48         if displayChange {
49             display = "\(digit)"
50             displayChange = false
51         } else {
52             display += digit
53         }
54     }
55 }
```



# Consultants: The Missing Piece in Your *API Puzzle*

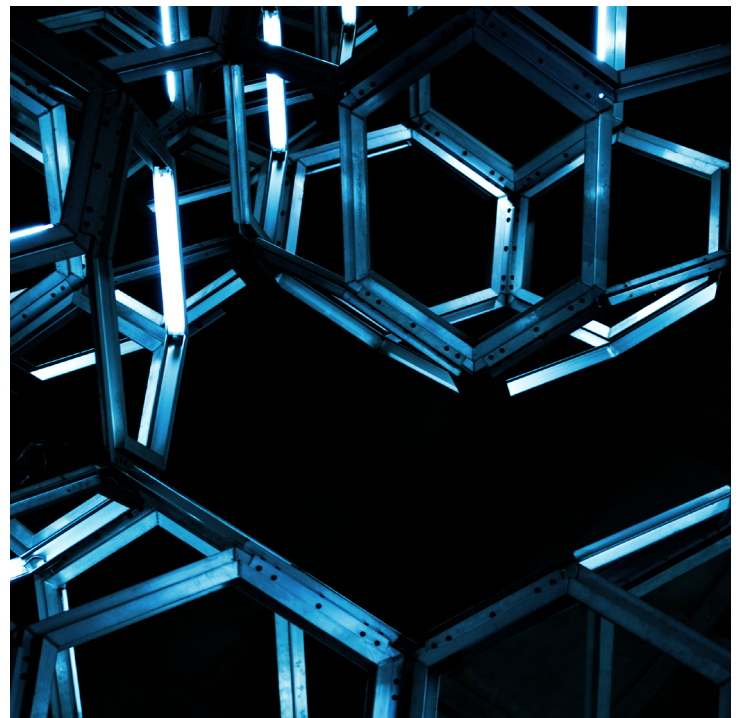
Based on the above information, you're not sure you want to assign considerable resources to build your own portal. Yes, this may become one of the key factors behind the success or failure of your API product. But you don't have the expertise or the industry experience to start on a basic portal, let alone make it into an expansive, usable platform that pushes your product to be better.

And you start looking. Maybe you can use one of the SaaS portals out there. They are clean, efficient, and easy to use. But, wait!! It seems you would need to create your own content. Well, developer portals are content-first, and without content, it may as well not exist. Well, you can always look for content consultants.

## Questions to Ask a Potential Developer Portal Consulting Partner

- Do they know how to create content for developer portals?
- Do they know how to work with the specific platform you chose or advise on how to pick the right one?
- Do they know how to assess where you are in your current journey and what the roadmap should be?
- Are they invested in the success of your portal?

- Can they leverage platform-agnostic expertise from a team that has created developer portals for products across multiple platforms and industries?
- Is their team supported by experts in other fields such as security, analytics, platform and product development, UX, and others?
- Do they have experience maintaining and improving the portal for the long run?



# A Proven, **7-Step** Approach to Build Your *Developer Portal*

The right consulting partner to build (and maintain) your developer partner will have a proven approach and track record of success. As a leader in consulting on developer portal best practices, we at Wizeline follow a platform-agnostic approach to building developer portals. We work with a variety of different technologies and platforms to build a portal that meets the specific needs of our customers.

At a high level, here's our 7-step approach to build the right developer portal for your needs:

1. Identify where you currently are and where you want to be
2. Identify your current and potential audience
3. Identify your ideal time to market
4. Identify the tools and platform(s) that will provide the maximum value
5. Ideate and create your platform
6. Strategize and develop your content
7. Innovate continuously



## Step 1: Identify Where You Currently Are and Where You Want to Be

Through our experience developing over 10 portals for clients, we have identified 4 phases of a developer portal. Our team collaborates with you to identify which phase you currently are in and where you want to and can be based on what has the most optimal impact for your product.

Where you  
**Currently are**

Where Wizeline  
**can take you**



### Knowledge Centralization

Migrate (locate, collate, and associate) information to a single source of truth.

### Business - Technical Portal

Fill the information gaps and transition the migrated content into a usable source of information.

### Developer Portal

Create a seamless developer experience in the form of a developer portal.

### Enhanced Developer Portal

The bells and Whistles that differentiates your portal from a collection of API documents.

### Continuous Improvement

Identification and implementation of aspects that improve your portal, and your overall product.

## Step 2: Identify Your Current and Potential Audience

Maybe you want to start small, but hope to eventually be a market leader for your API product. Maybe your API product is designed specifically for internal development teams. No matter the use case, Wizeline collaborates with you, and works to identify the portal that best fits your requirements.



### Step 3: Identify Your Ideal Time to Market

Wizeline will help you identify your ideal time to market. Our iterative product and content development process ensures that your time to market isn't limited by an incomplete product.

### Step 4: Identify the Tools and Platform(s) that Provide Maximum Value

Wizeline's service is platform-agnostic. It does not mean that we don't care about the platform. Instead, we work with multiple tools and platforms, and work with you to identify the tools and platform(s) that will provide the maximum value today, and in the future. Our expertise ranges from working with industry-leading SaaS tools and managed services to implementing standalone or combinations of open-source technologies.

For example, the following table is a small subset of our analysis of some of the tools available for developer portals in the market today:

Category	Value Proposition	MK Docs	Slate	Hugo Opt: Docsy	Gatsby	11ty
General	Hosted and managed by provider	✗	✗	✗	✗	✗
	Supported by provider	✗	✗	✗	✗	✗
	Support for all types of documents (conceptual and API reference)	✗	✗	✓	✗	✓
Customization	Highly Customizable Structure	✗	✗	✓	✓	✓
	Styling	✓	✓	✓	✓	✓
	Accessibility	✗	✗	✗	✓	✗
Features	CMS	✗	✗	✗	✗	✗
	Two way Git Integration (Repo to output to repo)	✗	✗	✗	✗	✗
	Rich Text Editing with two-way Git integration	✗	✗	✗	✗	✗
	Maintenance through Git Workflows	✓	✓	✓	✓	✓

We also look at your current and potential requirements and choose the tools and platform(s) that best suit your product.



## Step 5: Ideate and Create Your Platform

Based on your brand identity, we create your platform the way you always imagined it to be. We look at parameters such as UX personas of your audience and the technological aspects of your API product, and we work with you to build a platform that works as well for contributors on the inside as it does for your consumers on the outside.

## Step 6: Strategize and Develop Your Content

The technical documentation team in Wizeline are experts in their field. We work with you to define your content strategy and create your content so that it has maximum impact for your product's consumers.

Our developer portal content strategy is built upon the following 6 pillars of content, along with a healthy dose of usability, reliance, discoverability, and navigability across the spectrum:

- **Landing page(s)** that give your APIs a home they deserve
- **Concepts** that introduce and define your product. These can include sample architectures about pieces of your product that can lead to innovative solutions.
- **Instructions** on how to use your product. From 'Getting Started' guides to how to authenticate single endpoint use cases to multi-tier multi-API solutions.
- **Enriched references** for all that your product has to offer. Is that username merely a username for the account, or the username for the administrator account who deleted the record from the database?
- **Samples** such as code segments to get those creative juices flowing by letting your customers reduce their turnaround time for their solutions.
- **Interactions** so that your consumers can test your product without ever leaving the portal ecosystem.

We seamlessly integrate these and many more nuggets and tweaks into your content, so that using your product is easier than ever before.

## Step 7: Innovate Continuously

Wizeline development teams are never satisfied, and our developer portal service exemplifies this. Our teams learn and leverage new technologies and trends such as ChatGPT that are available in the industry to provide better and faster iterations.

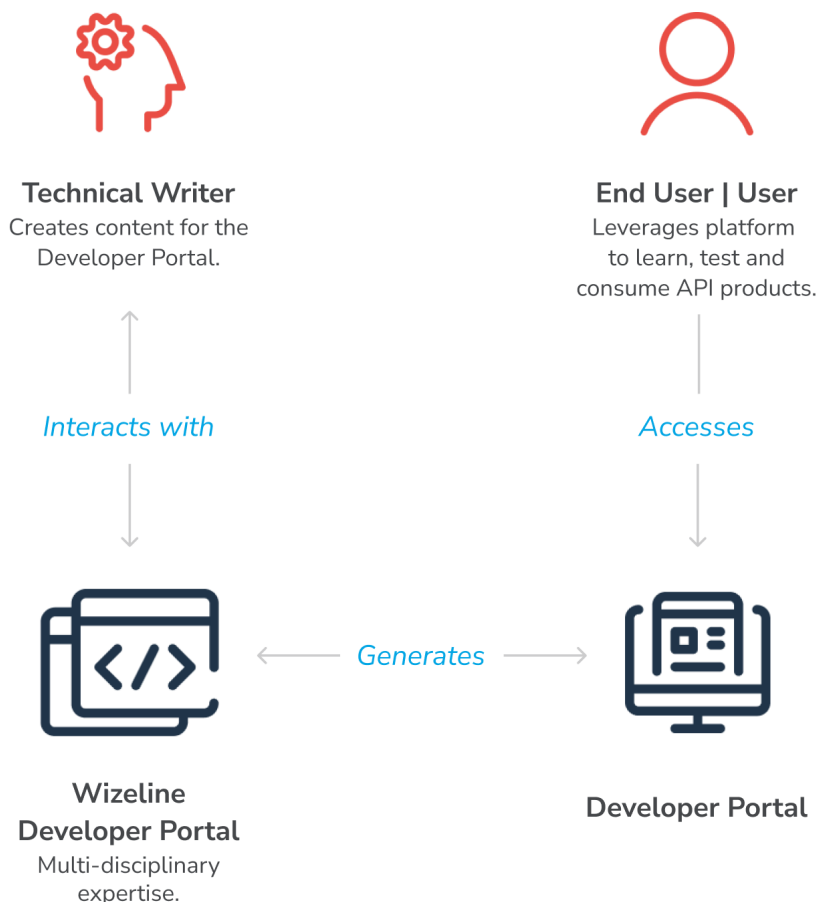
- **AI-assisted technical documentation:** With the rise of natural language processing (NLP) and machine learning, AI-assisted technical writing is becoming more prevalent. AI language models like GPT-X can analyze existing information, generate baseline content and additional value from prompts, and implement standardized content creation and review practices, which for you translates to a faster time to market and a seamless experience.
- **Interactive assets:** Interactive assets such as diagrams can enable you to improve the reader experience by letting them be as high-level or deep as they want into the information that you wish to convey.
- **Different interaction models:** Using cutting-edge models such as Voice User Interfaces like Google Assistant actions or Amazon Alexa skills can enable you to reach a wider audience, and improve the usability and adaptability of your product.
- **Leverage the industry:** Wizeline is always looking at unleashing innovation, whether it's developed internally or using third-party tech. We leverage the solutions and implement them, for you, so that when you want us to help build a car, you don't need to start with reinventing the wheel.



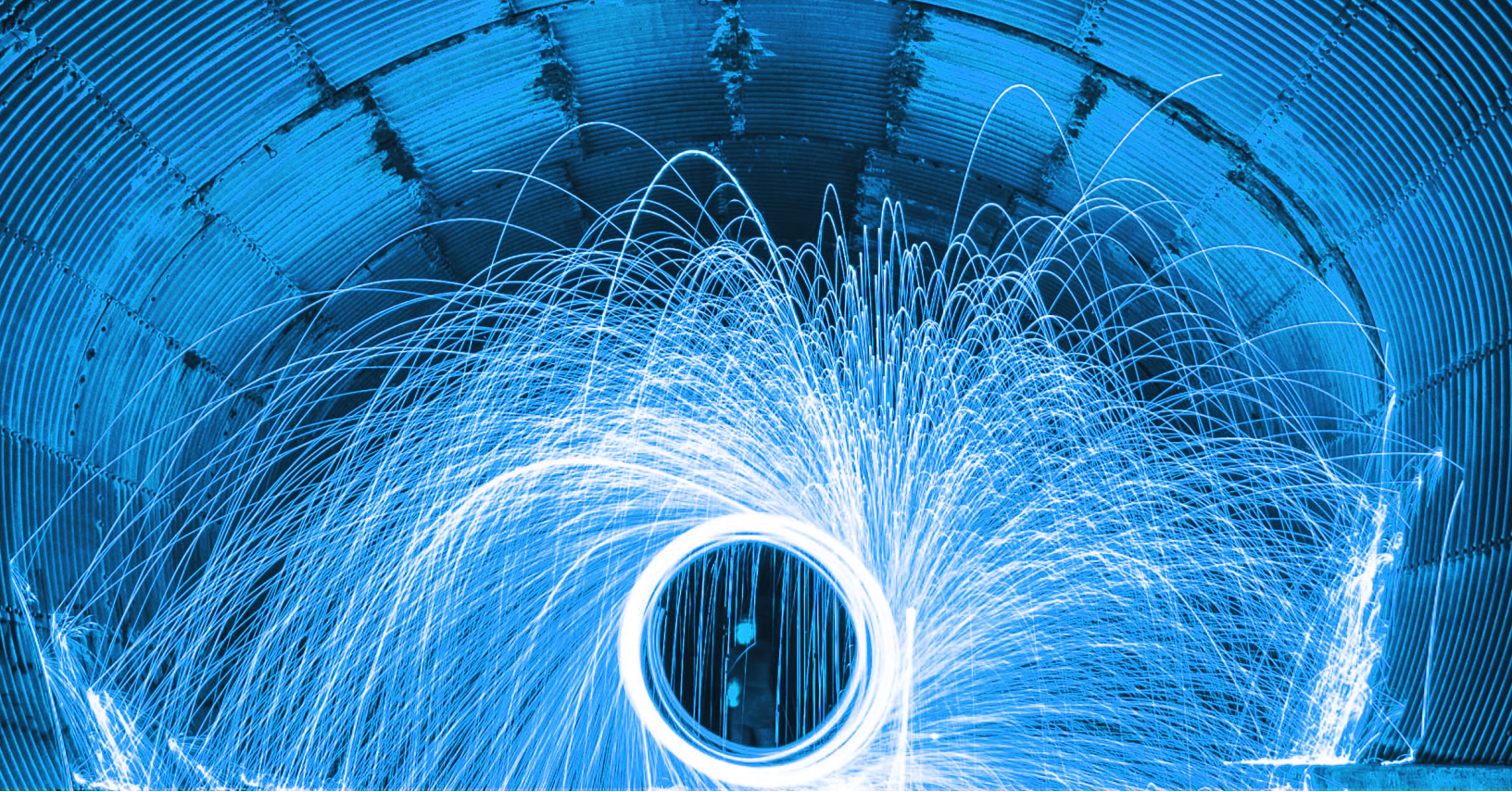
# Developer Portal Workflows & Reference Architecture

Wizeline develops developer portals as a content-first product. The technical documentation team leads the developer portal development, while collaborating with a multidisciplinary team and using modern practices such as reusable assets and AI-assisted content development.

## System Level Architecture for Developer Portals

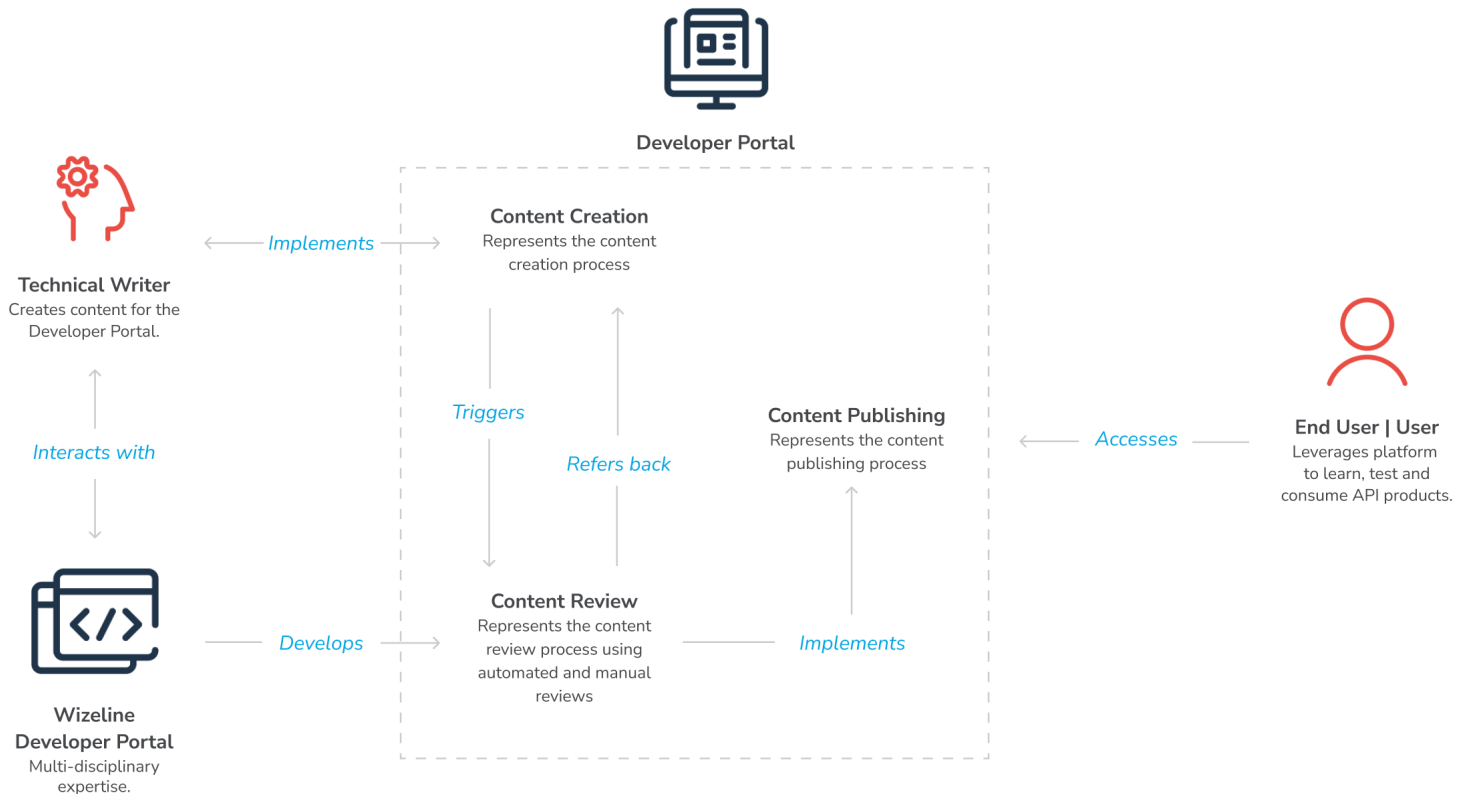






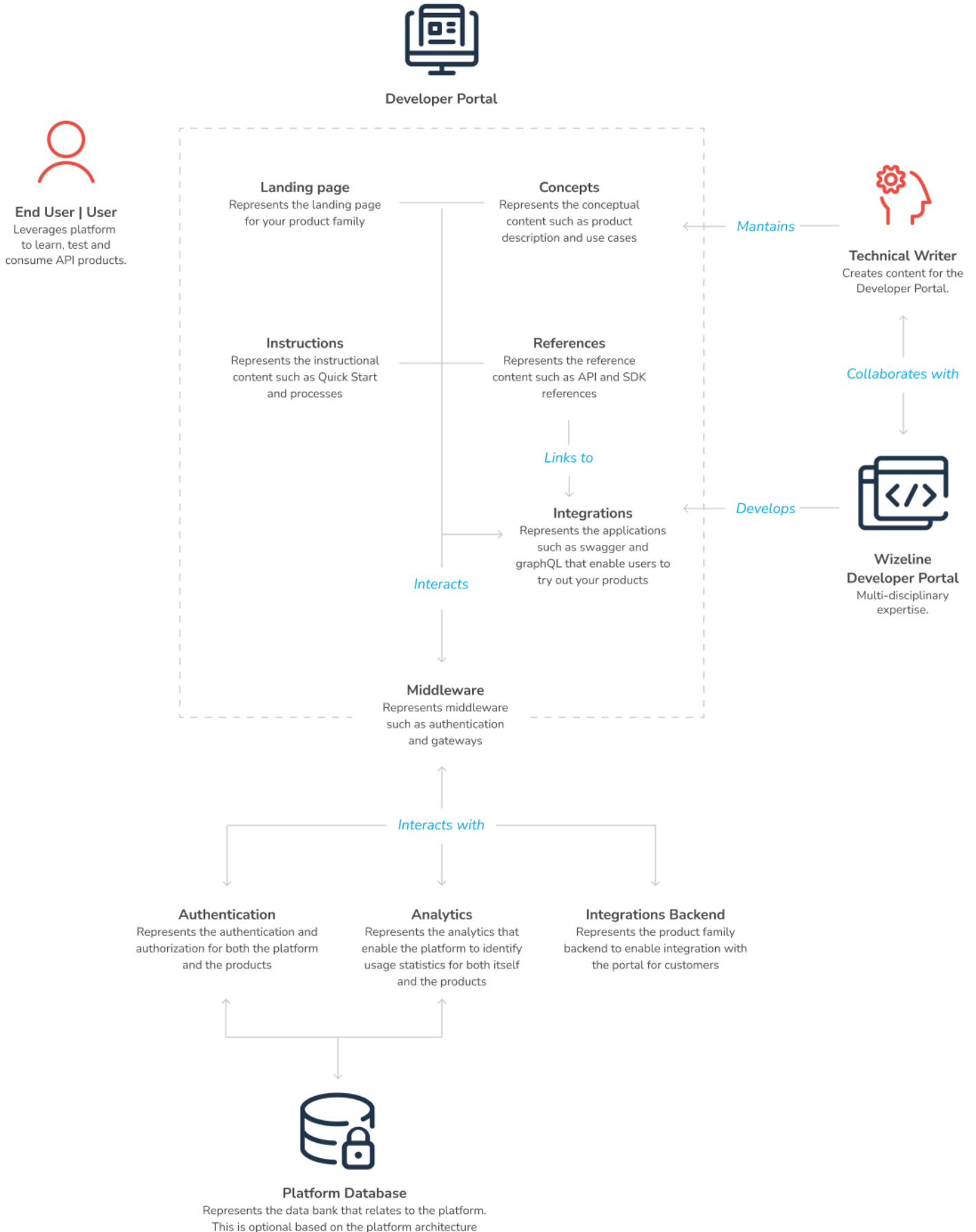
## Content Creation Flow of Wizeline Developer Portals

Wizeline implements a DevOps-inspired flow (DocOps) to develop and generate content for your developer portals, leading to a faster time to market.



# X-Ray of a Wizeline Developer Portal

Wizeline identifies what works best for you, and solves your opportunity area that works best for you.





# Developer Portal Case Studies

So there you have it – the complete guide to ideating, building, and maintaining developer portals. Now that you've seen our thinking on how to best imagine, build, and maintain a developer portal, you may be wondering about what kind of results we've generated for our customers. Here are a few examples:

- **Media company**

The client's developer portal is the primary documentation platform for their customers. The portal has evolved from a basic API documentation site to a complete solution with documentation integrated into a single source.

Wizeline was initially contracted to migrate existing documentation from multiple platforms to a single source. Using text automation scripts, Wizeline completed the migration process two months ahead of schedule. Wizeline further consulted for a more exhaustive and modern solution for the dev portal.

End-user interaction with the company's products, especially lesser-known and newer ones, has seen a marked rise through the new developer portal.

- **Delivery company**

The client required resources to improve the integration process with other companies. The client experienced slow and unsuccessful integrations with their partners, causing delays and spending resources solving these matters.

Wizeline worked to improve their API documentation for easier understanding. We presented a hybrid solution that combined two separate tools (MkDocs and Slate) for user and API documentation and a set of new documents that helped define standards and guide the users during the complete integration process.

The team's tools aim for shorter integration times and a more significant percentage of successful interactions. This tool also gives a better understanding of the complete process and its standards.

## • AI company

The client didn't have any updated documentation to teach end-users how to use their product. Initially, the documentation was published in ZenDesk. However, this tool was not agile enough, and it took a long time to update documentation. The company wanted to use open source tools for their documentation to avoid additional costs.

Wizeline launched the documentation in a static site to improve the user experience and enable content creators to update documentation faster. We employed a CI/CD process with a CI linter that checked grammar and ensured no broken links or images existed. In addition, the team incorporated Google Analytics to the static site.

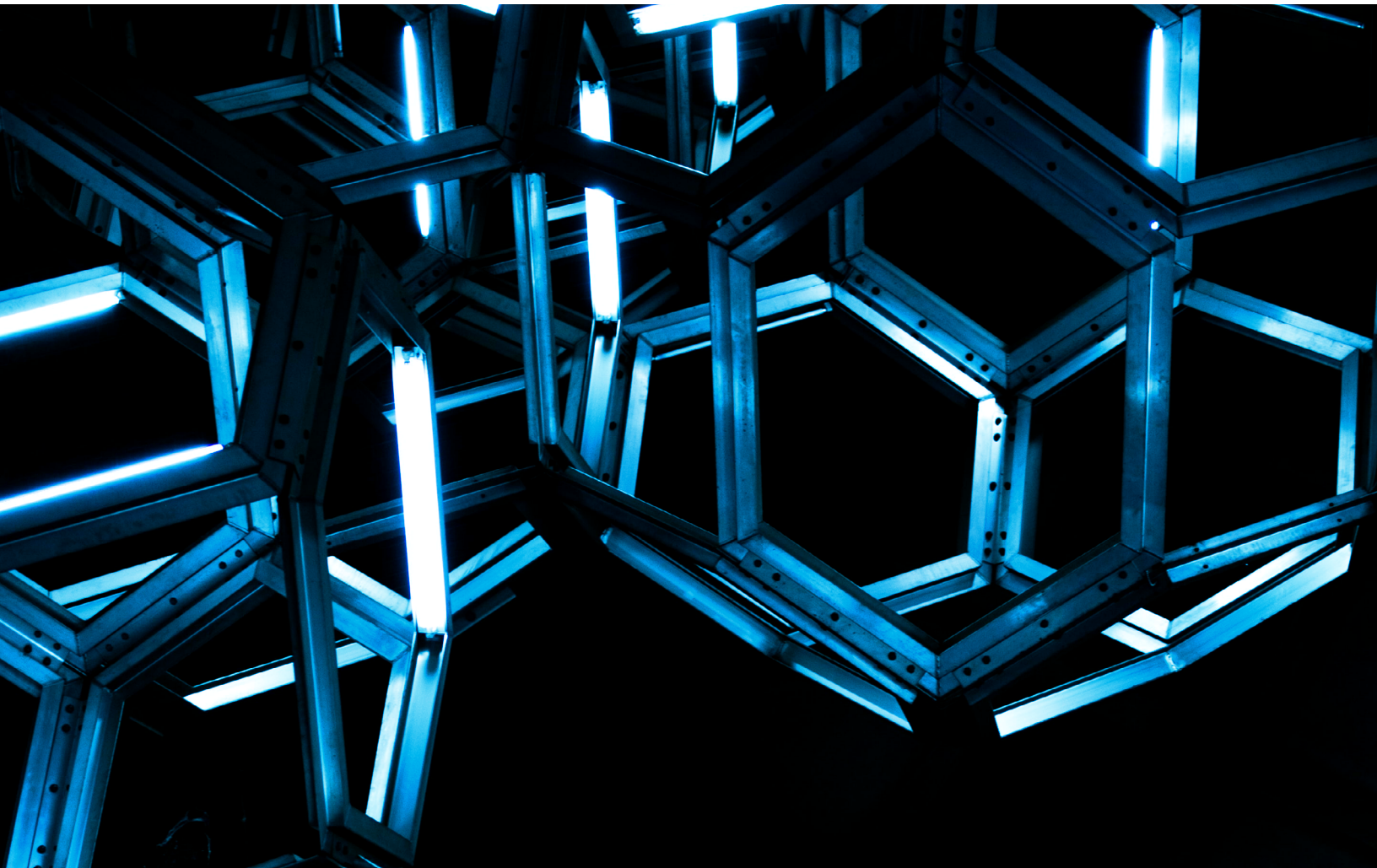
The team created awareness about the value of documentation in the company. The company can now follow best practices on how to create and publish documentation. They can use the style guide, testing tools, and templates that the team created.

## • Big Four firm

The client had insufficient and outdated documentation. Information was lost when team members left the company.

We performed a gap analysis on the documentation needs for different products. Based on the outcome, we created, updated, organized, and migrated content with templates, information architecture, and guidelines to maintain consistent documentation and to improve the developer experience.

Looking to achieve similar results with Wizeline? We would be happy to guide you, whether that means building your first developer portal from scratch, optimizing an existing offering, or simply maintaining your developer portal to take your API usage to the next level. We look forward to getting started!





# About *Our* **Author**

**Arunabh Nag** is a veteran of the Technical Documentation industry and has worked on domains such as VoIP, SIP, Intelligent Networks, Education, Aviation and Aviation Cargo, End-to-end Infrastructure monitoring, and Media, among others. Currently the Technology Director collaborating in Documentation, Developer Portals, Developer Productivity, and Hyper Automation in Wizeline, Arunabh leverages his technical education background, his industry experience, and his penchant for keeping up to date with what's new to evangelize and implement innovative solutions.



# WIZELINE

## Start the Conversation Today

To learn more about how Wizeline can help you implement or optimize your API strategy, contact our team or visit our website today!

[consulting@wizeline.com](mailto:consulting@wizeline.com)

[www.wizeline.com/partners](http://www.wizeline.com/partners)

<https://www.wizeline.com/developer-portals/>