MuleSoft

WHITEPAPER

Secrets of a great API

Core principles for delivering successful APIs

Table of contents

Secrets of a great API

APIs are not new. They've served as interfaces that enable applications to communicate with each other for decades. But the role of APIs has changed dramatically in the last few years.

Innovative companies have discovered that APIs can be joined together in an application network to gain momentum across the business. APIs allow companies to monetize digital assets, work more productively with partners, and connect to customers across channels and devices.

When you create an API, you are allowing others within or outside of your organization to make use of your service or product to create new applications, attract customers, or expand their business. Internal APIs enhance the productivity of development teams by maximizing reusability and enforcing consistency in new applications. Public APIs can add value to your business by allowing third-party developers to enhance your services or bring their customers to you.

As developers find new applications for your services and data, a network effect occurs, benefiting the bottom line. For example, Expedia opened up their travel booking services to partners through an API to launch the Expedia Affiliate Network, building a new revenue stream that now contributes $2 billion in annual revenue. Salesforce released APIs to enable partners to extend the capabilities of their platform and now generates half of their annual revenue through those APIs.

The pitfalls of a mediocre API

Organizations often decide to build an API without fully considering key success factors or without first engaging their stakeholders.

In either case, organizations risk creating an API that does not fit the needs of their users. And APIs that don't fit the needs of users have a high cost: limited adoption by developers and ultimately, a failure to meet business objectives. Once an API is designed and built, undoing these mistakes is difficult and time-consuming. In most cases, a developer must start over again, redesigning a new API, implementing it by connecting to back-end services, then rolling it out again to the developer community. Worst of all, you will have to transition all existing users to the new API. This will require additional work which your engineers may not have the time or willingness to do. At that point, you'll be faced with a tough choice: continue to support the original API and its users until they eventually (hopefully) migrate, or shut it off and potentially alienate and lose those users.

"What you need is an API that is simple to understand and easy to use. Developers should be able to assess the functionality of your API and start using it in just a few minutes."

Another common pitfall of API programs is allowing the design of your API to be dictated by the constraints of internal systems or processes. This is never a good idea, but is particularly perilous when the back-end functionality lives in legacy systems whose data schemas are overly complex or whose business logic has been extended over the years using hard-coded workarounds and convoluted logic. Exposing this kind of dirty laundry to your API consumers is a recipe for failure. APIs modeled after internal systems are difficult to understand and use and developers simply won't invest the time it takes to become productive with them.

What you need is an API that is simple to understand and easy to use. Developers should be able to assess the functionality of

your API and start using it in just a few minutes. The only way to deliver that kind of ease of use is to design for it upfront.

The value of a great API

A successful API is more than a feature. When you view your API as a product, it can be an enabler of your business strategy. Part of the magic of APIs is that creative developers find uses that the API designers never envisioned. If your API is well-designed and easy to use, this can be an enormous benefit and opportunity, turning your service into a platform that can grow in many ways.

"A great API can help you grow an ecosystem of employees, customers, and partners who use your API and help evolve it in ways that are mutually beneficial."

A great API encourages use. Developers share the API with others, creating a virtuous cycle where each additional successful implementation leads to more engagement and more contributions from other developers, adding value to your service. A great API can help you grow an ecosystem of employees, customers, and partners who use your API and help evolve it in ways that are mutually beneficial.

But the promise of APIs can only be realized when target consumers begin to use them. For internal developers, APIs introduce a new way of working that requires some buy-in. In addition, internal developers won't use your API if they don't believe it's the best, most efficient way to achieve their goals. Well-designed APIs that are easy to use will encourage adoption by internal developers, paving the way to a better-defined, more consistent, and maintainable approach to development. For public APIs, the situation is even more competitive. An ever-increasing pool of APIs is competing for

developers' attention, making the design and ease of use of your API critical to its adoption and, ultimately, its success.

Unfortunately, too many API providers build their APIs before thinking through the critical success factors, resulting in APIs that fail to meet business objectives. Delivering a great API isn't hard if you follow a few proven principles. In this paper we'll demystify API strategy by reviewing the four secrets of a great API.

Secret #1: design for great user experience

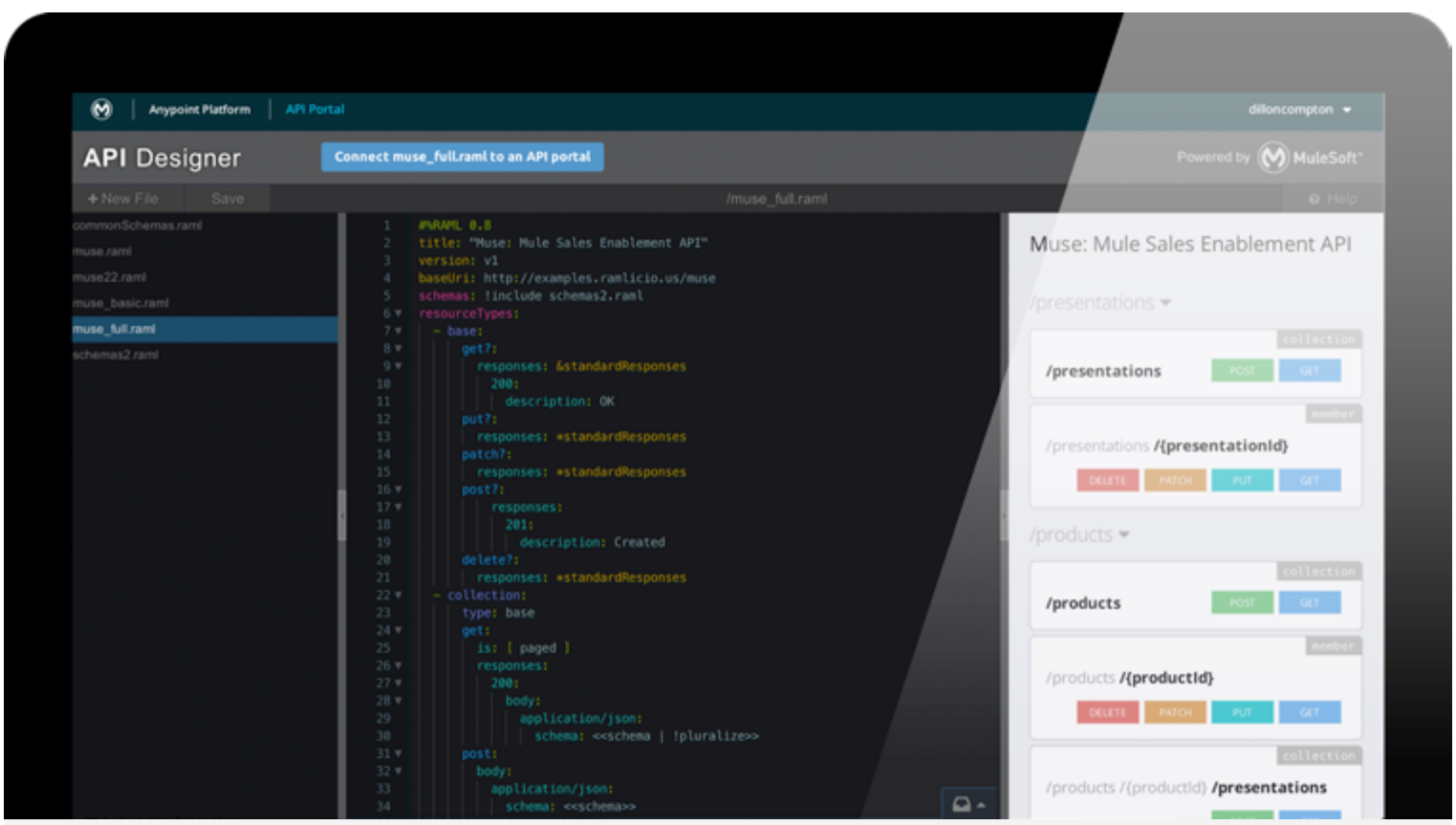


Figure 1: Anypoint Design Center's API designer provides an editor for drafting the API's structure while rendering in real time.

To deliver great APIs, design must be a first-order concern. Much like optimizing for UX (user experience) has become a primary concern in user interaction (UI) development, optimizing for API user experience (APX) should be a primary concern in API development. An optimal API design enables application developers to easily understand the purpose and functionality of the API so that they can quickly become productive using it. It allows organizations to focus on getting API design right and avoid expensive and time-consuming mistakes in back-end implementation.

The best way to design an API that developers want to use is to iteratively define the structure of the API in an expressive manner and get feedback from developers on its usability and functionality along the way. The API designer is an example of this concept in action. The API designer is an open-source design environment that leverages RAML®, the RESTful API Modeling Language. The API designer provides an editor for

drafting the structure of an API while rendering in real time an interactive console to enable interaction with it. As the API is designed, application developers can test its behavior, thanks to an integrated mocking service that returns the values a call to the live API would produce. Because APIs designed in RAML are concise and easy to understand, application developers can rapidly assess the API's functionality and usability and offer concrete feedback on ways to improve it.

Secret #2: optimize for the use case

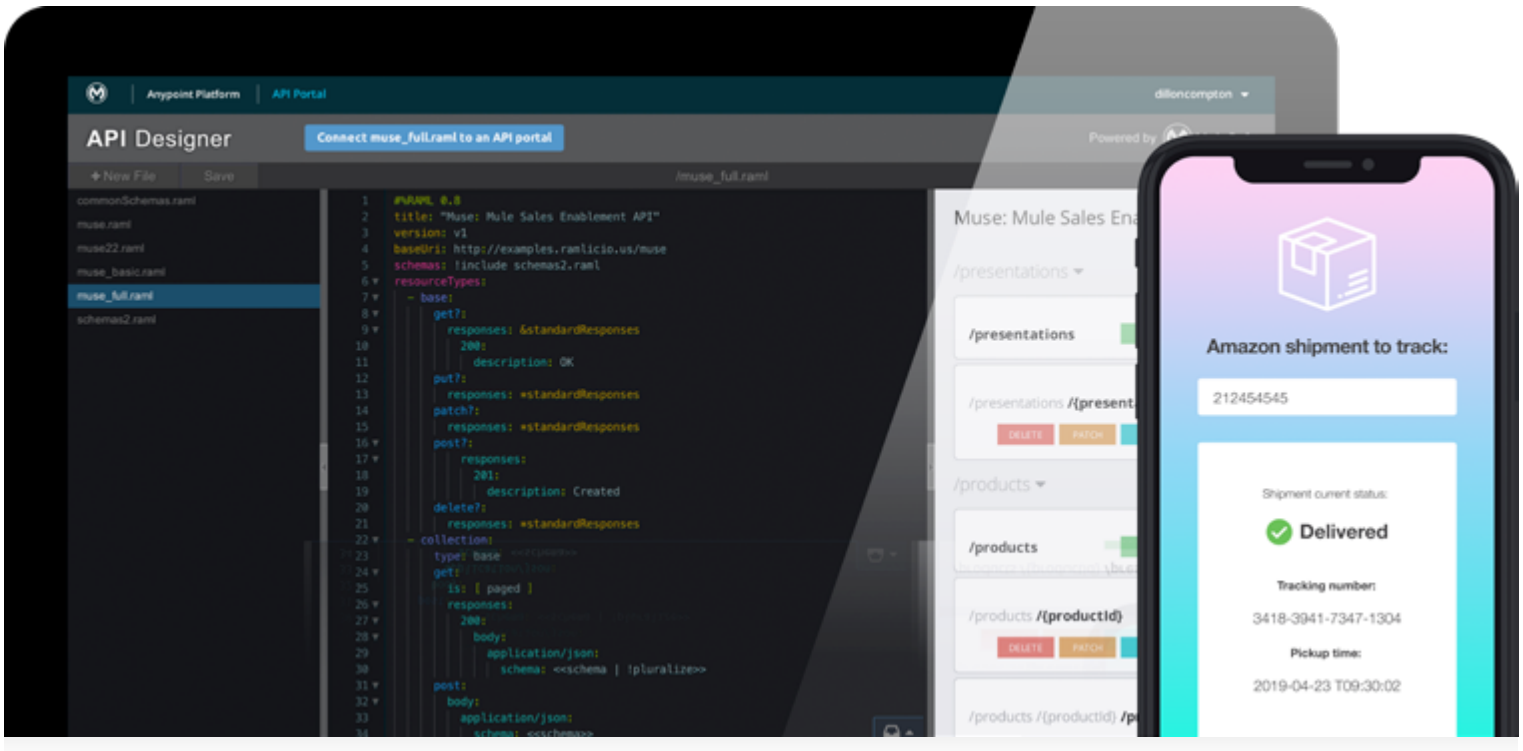


Figure 2: Developing a mobile application with API designer.

There is no such thing as a one-size-fits-all API. Even for the same underlying service or set of services, multiple APIs might be required to support different types of users and use cases. An API should be optimized to fulfill a specific business request in a specific context. Too often APIs are modeled after the design of the back-end services or applications they expose instead of the use case they fulfill. This results in poor performance of the client application, poor user experience, and, ultimately, poor adoption.

To optimize your API for a specific use case, think about how coarse or fine-grained it should be. For example, if you're designing an API to enable access to sales order status from a mobile device, you need to consider certain constraints. A mobile application has a higher sensitivity to the number of network trips, latency, and data size than a web application. Accordingly, this API should be designed to limit back-end calls and minimize the size of data returned. In addition, this use case is fairly granular. The API will look up an order based on an order number and return a status. Therefore, the API

should expose this specific fine-grained functionality so it can be invoked independently. If the underlying service it accesses is coarse-grained and you anticipate building additional APIs on that service to address additional use cases, consider a tiered approach. Expose fine-grained services that users can access directly, and add coarse-grained services on top of them to support broader use cases. Users can choose to call the fine-grained APIs directly or if they need the combined functionality of multiple fine-grained calls they can use the coarse-grained APIs. This API designed in API designer is an example of an API optimized for this case.

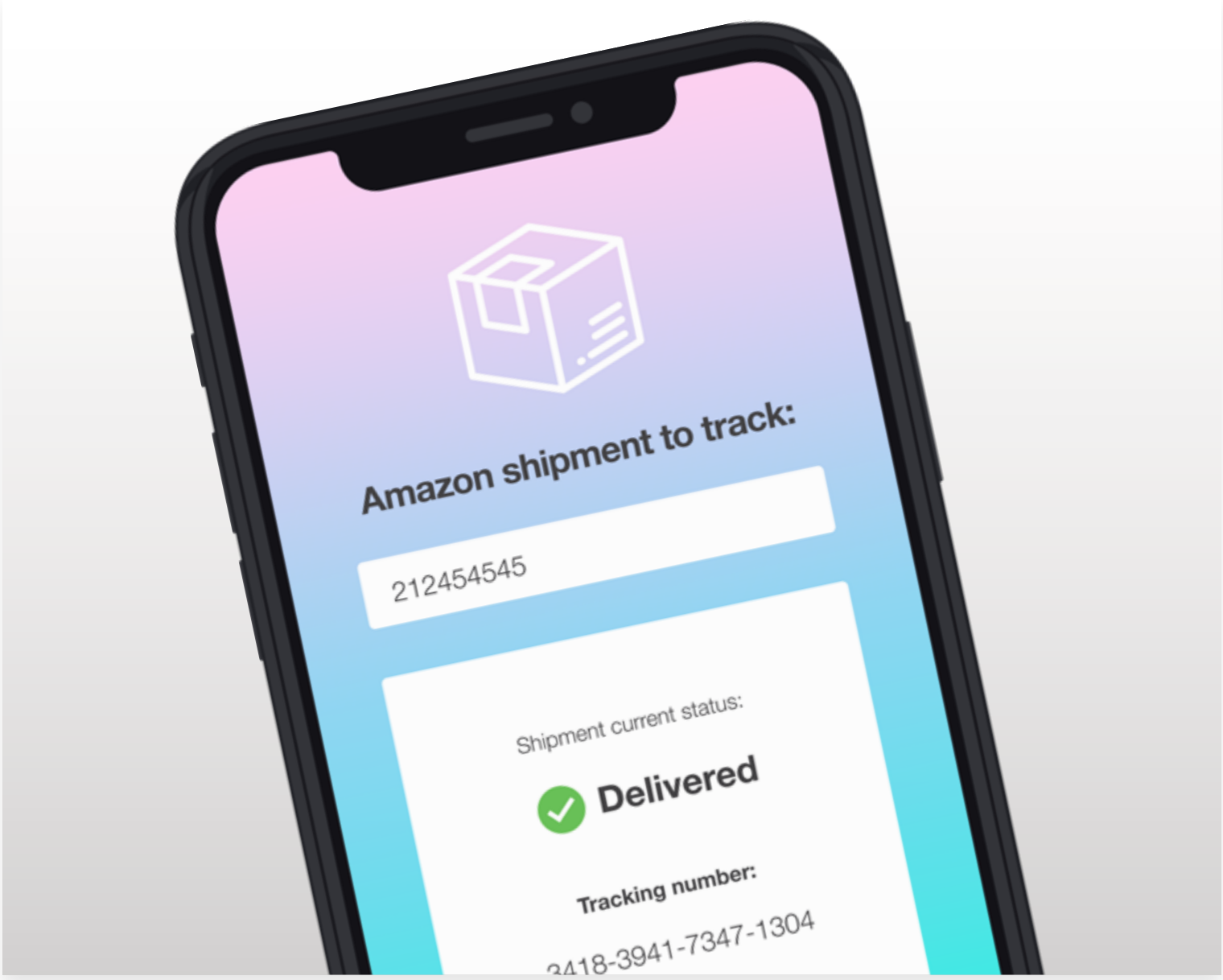


Figure 3: Use API designer to expose fine-grained services that users can access directly, and add coarse-grained services on top of them to support broader use cases such as this delivery tracking app.

Secret #3: provide easy access

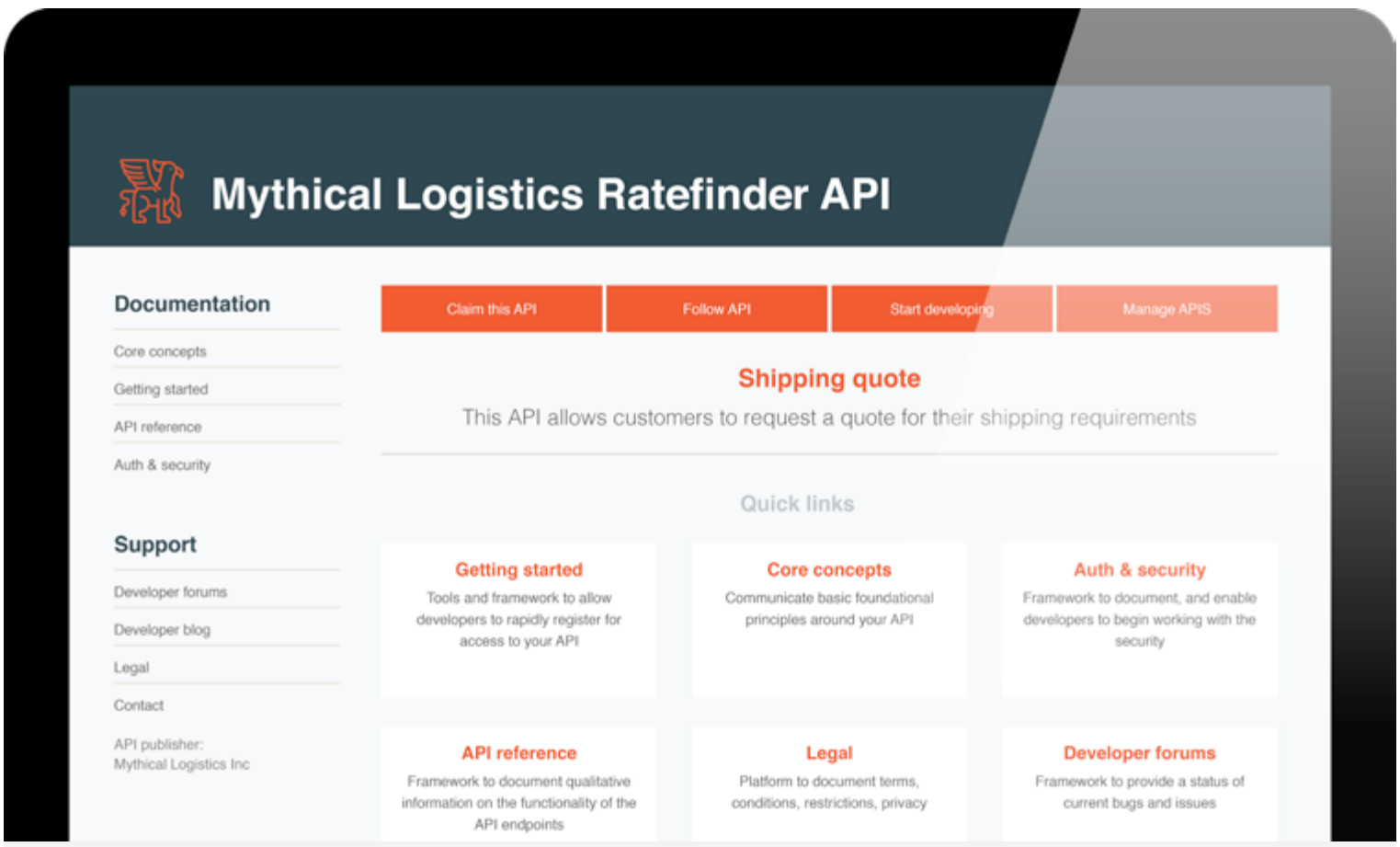


Figure 4: Your developer portal should include all of the tools developers need to learn about and begin using your API.

Finding an audience for your API begins with publishing it to a portal that allows developers to discover and evaluate it for their use case. The developer portal should include all of the tools developers need to learn about and begin using your API. Developers reviewing your API will only invest a few minutes before deciding whether or not to continue. Having information available in a clear and easy-to-consume format will encourage them to stick around rather than go elsewhere. Developers will quickly scan your documentation to get an overview of its functionality then zero in on what it will take for them to get up and running. From there, they'll quickly want to see some examples and, ideally, start interacting with the API. Avoid static documentation pages. Developers are far more likely to use an API if they can interact with it as they learn about it.

> "The API portal delivered in MuleSoft's Anypoint Platform™ is a good example of the value-added features that make it easy for application developers to engage with and start using an API."

API portal, delivered in MuleSoft's Anypoint Platform™, is a good example of the value-added features that make it easy for application developers to engage with and start using an API. The API portal includes interactive documentation that not only describes the endpoint but also the fields required to call that API and the data that is returned. In addition, you can add code samples to give developers a head start in building the code to access your API in the applications they build. Finally, the API Console includes "try it" functionality that allows developers to interact with and test the API. During the design phase before the API has been implemented, the mocking service allows developers to test the API's behavior and see the resulting body that a call to that API would produce. Once the API is implemented, developers can test the live API.

Secret #4: build a community

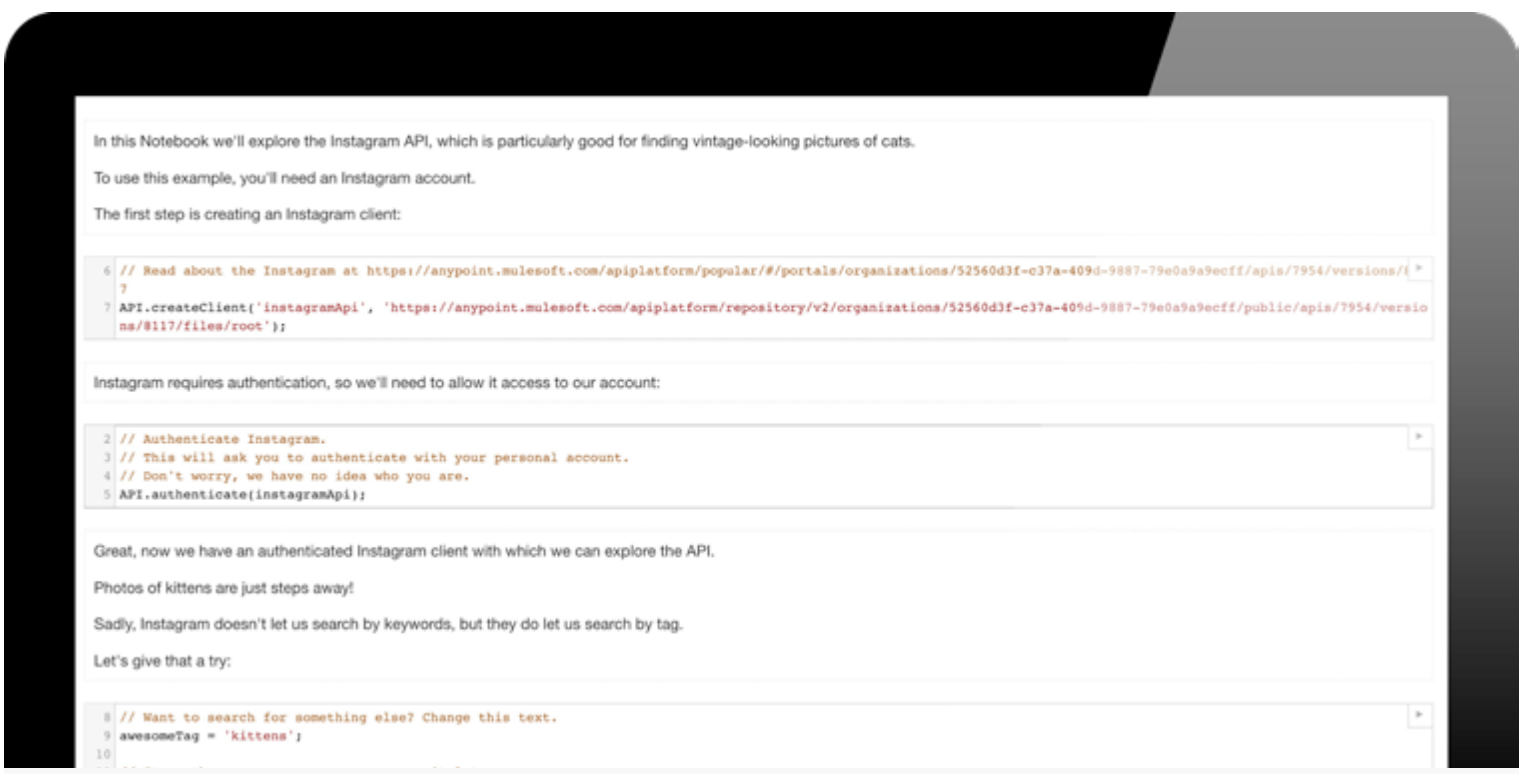


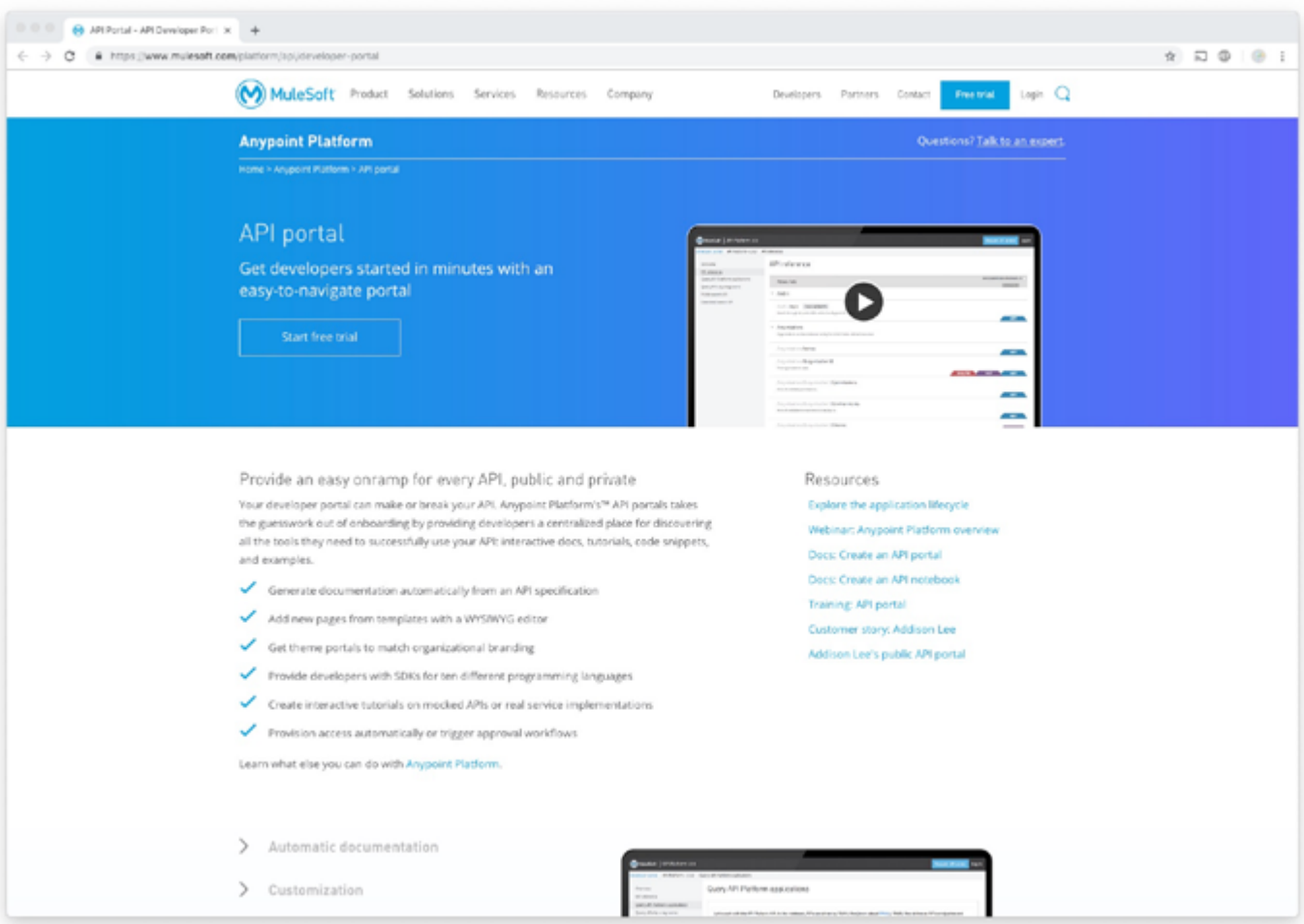
Figure 5: API Notebook, a feature of the API portal, allows developers to document new uses for, and to grow the addressable market for your API.

The application developers who consume your API are not just your customers; they are the ecosystem that will drive the success of your API. Treating them as valued members of your community can drive significant mutual benefit. An obvious benefit of a thriving developer community is a wider reach for your API. To support the organic growth of your API, your developer portal should include an easy way for developers to share knowledge with each other. The API Notebook of the API portal demonstrates this concept in action. It allows developers to discover and document new uses for your API and grow the addressable market for it. In addition, they can share tips and tricks in forums and even add code samples to make it easy for others to get started quickly with your API. Finally, a valuable benefit of community that is sometimes overlooked is that the greater the number of developers using your API, the faster bugs and issues will be identified and communicated so that you can continue to improve the value of your API.

> "To support the organic growth of your API, your developer portal should include an easy way for developers to share knowledge with each other."

In addition, there is great benefit in having an established communication channel with your developer community. Your API is not a static entity. As new use cases are identified and use of your API expands, enhancements and fixes are inevitable. When you release a new version of your API, you can easily communicate the enhancements in the new version through your developer portal. You can also quickly assess who is using each version of your API and communicate an upgrade path to them as you deprecate older versions. Finally, understanding your developer community and having accurate insight into use cases and patterns will provide invaluable knowledge that you can use to enhance your API over time.

Summary



APIs are becoming ubiquitous as the potential for API programs and application networks to transform business is becoming widely recognized. But delivering a successful API program that achieves defined business objectives requires a systematic approach to designing and managing APIs. Great APIs aren't difficult to develop if you design your API with an outside-in mindset. The key is to design for your users and the business processes the API will support. Before you know it, developers will be actively using your API and providing feedback to make it better. By treating your developer community as an extension of your business, you'll create not only great APIs but the flexible, resilient connections needed to speed up processes and make way for innovation at every level.

Get started designing APIs at **Anypoint Platform**

About MuleSoft

MuleSoft, a Salesforce company

MuleSoft's mission is to help organizations change and innovate faster by making it easy to connect the world's applications, data, and devices. With its API-led approach to connectivity, MuleSoft's market-leading Anypoint Platform™ empowers over 1,600 organizations in approximately 60 countries to build application networks. By unlocking data across the enterprise with application networks, organizations can easily deliver new revenue channels, increase operational efficiency, and create differentiated customer experiences.

For more information, visit **mulesoft.com**