API Commerce Trilogy #2

APIs and microservices: The surest path to reinvention

sales@moltin.com
moltin.com

moltin

Contents

Introduction

"Application programming interfaces open new business opportunities. However, too many CIOs make the mistake of seeing APIs as technology only instead of basing their company's business models, digital strategies and ecosystems on them." [1]

Your business has decided they need to take that leap into the future and take a closer look at the API approach. Great! If you've read the first book in our series, "Experience-driven commerce, "you already know how important it is to be technologically flexible in order to stay competitive and that APIs allow you to do just that. Implemented fully or as a microservice plugged into your existing system, they present a long-term solution to that never-ending challenge of staying ahead of the curve.

However, it is imperative to evaluate potential API candidates first, not just jump feet-first into any solution. A well-written API can provide some very important long-term benefits, but a hastily built one can harm the security of the entire framework and introduce major inconsistencies.

To help you make a well-informed decision, this book will walk you through some of the most important aspects of APIs. We're going to explain what APIs and microservices are and analyze why having them is so beneficial by showcasing some business scenarios.

[1] Source: Gartner, Top 10 Things CIOs Need to Know About APIs and the API Economy

moltin

Why focus on APIs?

The transformational power of an API is that you can plug it into your existing ecosystem without the need to restructure the entire architecture. It complements rather than substitutes.

It also lets you focus on the functionality and the end result rather than fret about the execution. It lets you implement the latest technology and be the first one there — wherever “there” currently is.

APIs explained

API (application programming interface) is an interface providing communication between other systems. It lies on top of an application or library and warrants continuous interaction between two or more structures.

In other words, an API allows one piece of software to “talk” to another. It’s a program that takes requests, tells a system what needs to be done and then returns a response.

API as a waiter in a restaurant

One of the best examples explaining the notion of an API and how it works is the restaurant analogy concocted by Kin Lane, an API evangelist.

Basically, for the restaurant to work specifically for you as a customer, you need a waiter to get your order from you at your table and take it to the kitchen where the food is made.

If we turn the waiter into the API, the waiter (API) takes your order (the request) to the kitchen and returns with your food (the response). The waiter is passing data along and returning it to give you a personalized experience based on what you've asked for. And this is exactly what an API gives you. It tells the back-end system what to do on the front-end and vice versa with you as a customer deciding how, when and what you need.

All APIs typically work in a similar way. When you use an interface (push a button, navigate through a menu, etc.), the functionality that lies underneath is triggered by your action, most often to go and retrieve some information. Now, instead of retrieving information from within the same system, web APIs call multiple remote services on the web.

Web services are web-based applications that provide resources in a format consumable by other machines. These are basic request-and-response interactions between clients and servers.

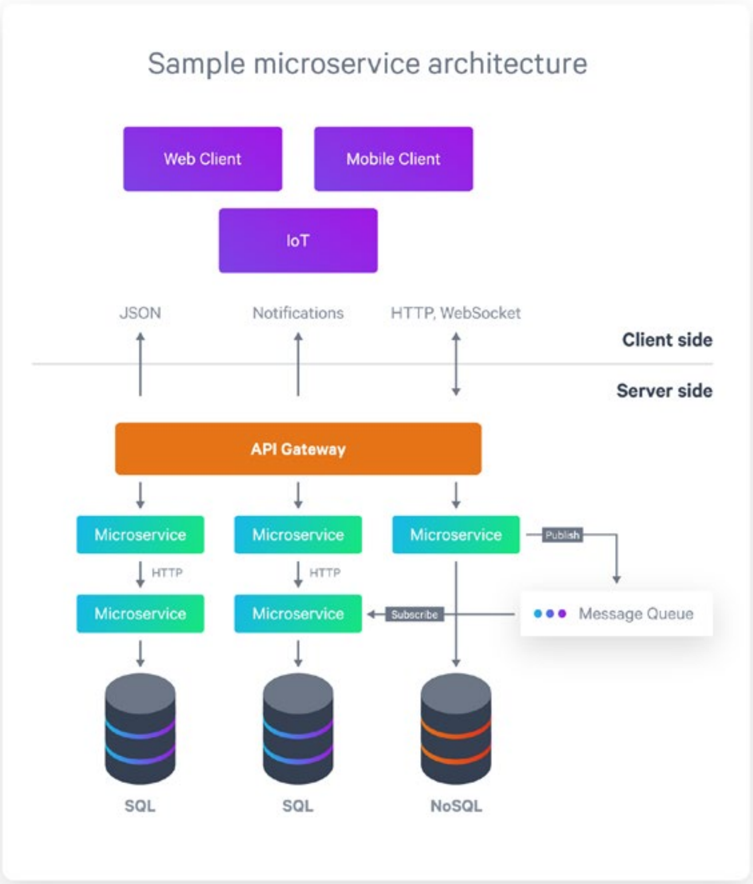
Web APIs are most commonly referred to as REST APIs and are currently considered the most flexible approach to any system architecture.

Microservices — great for retailers, loved by developers

A microservice is an architectural approach to creating applications in an iterative fashion. An application developed from a set of microservices will consist of multiple loosely coupled units that each draw from their own data set (typically stored in a database), each capable of functioning individually as well as coherently interacting with each other.

An entire suite of such microservices, each running their own process, will communicate with each other through a HTTP protocol. This is where the APIs come in handy, allowing each individually deployable unit to interact with each other.

Sample microservice architecture
Companies that choose a microservice-based architecture show that they value both their own employees and their customers. They understand that each person and each company is unique, has different needs and comes with different personal preferences. As such, each microservice uses a technology stack best suited for the task at hand and preferred by the team using it.



Problems of a traditional approach

"Many of these products have been around for well over a decade and are very established with what they do. However, speed of delivery, cost of platform and complexity of tooling are often at odds with more modern, lightweight approaches to IT project delivery [APIs]. Conversely, the open-source frameworks and platforms that have matured over the last several years are a much better fit for the new modes of delivery." [2]

The more traditional approach to designing complex software is based on a monogamous system lying on top of a single database with multiple layers of components that are tightly coupled and thus not independently consumable. Typically, the components have to be called in a specific order for the entire system to work as expected.

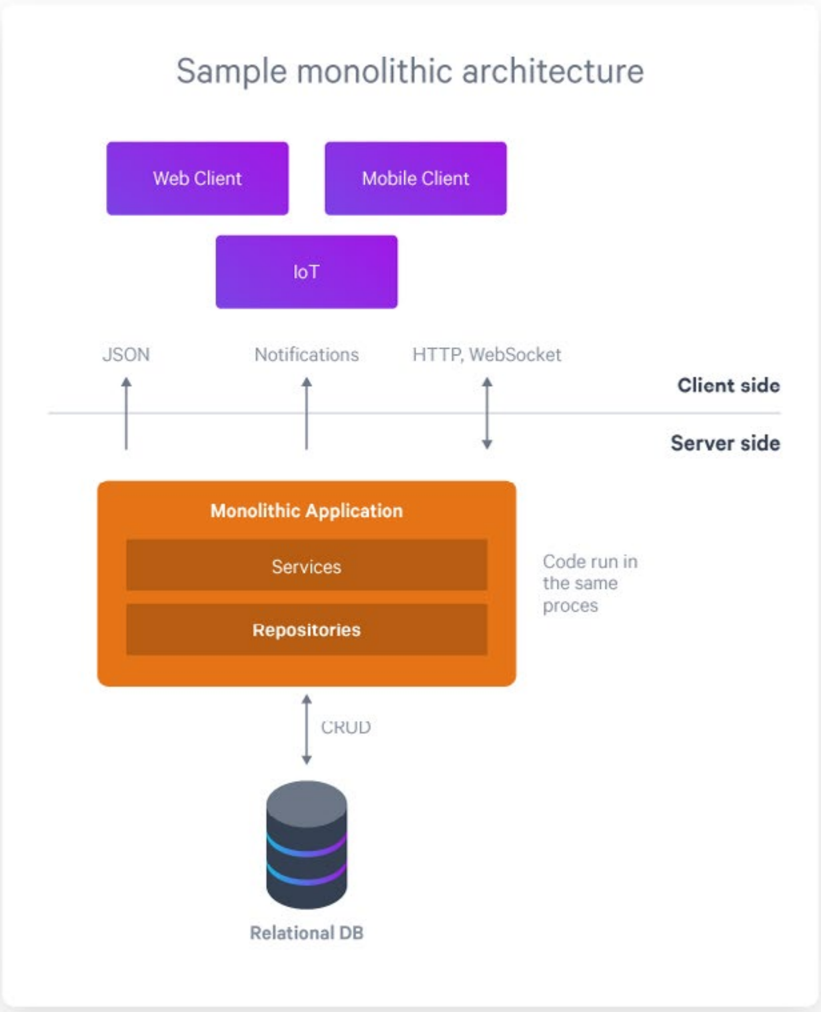
Such monolithic systems are based on huge libraries that create internal dependencies, allowing the components to interact with one another. Updating such systems without breaking existing integrations requires highly trained specialists and is time-consuming.

Moreover, monolithic software is often distributed as packaged applications that are deployed on-premise and customized to meet the individual business needs. Due to the heavy data handling, a full onboarding process generally takes several months and involves delegating a trained field engineer to work full-time on a single task.

[2] Source: Gartner, Top 10 Things CIOs Need to Know About APIs and the API Economy

While fine for brands that consider it enough to have a physical store and a website to complement the experience, this type of approach proves difficult to handle for companies that want to jump straight into the most innovative technologies and deploy their software on multiple devices like phone, tablet, wearables, smart TVs, etc.

It's not only about outputting the software into many channels but also about having a unified and consistent experience on all of them.



Benefits of APIs and microservices

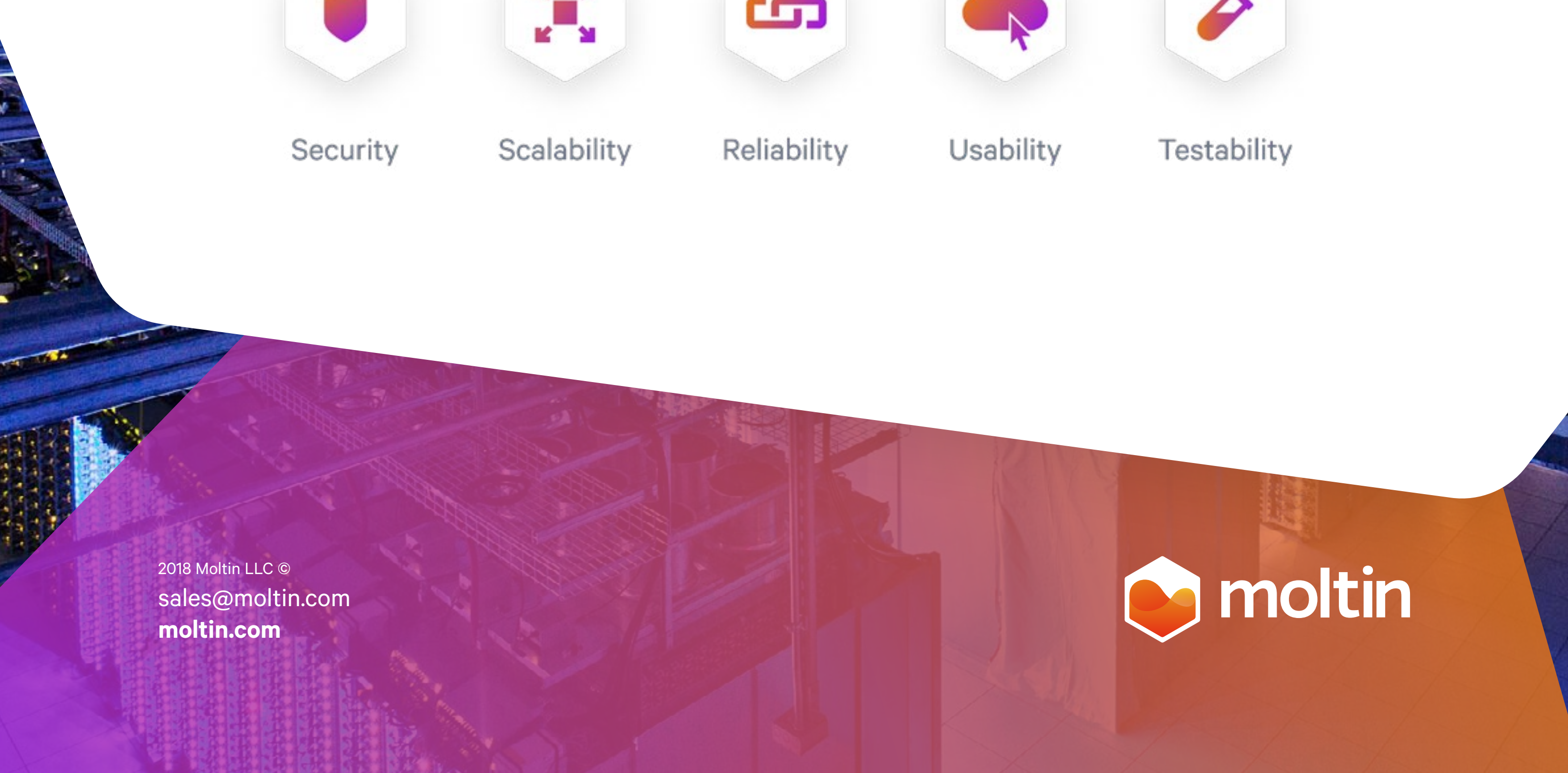
With the sudden rise in popularity, there's a lot of talk about APIs and how beneficial they are from the software development and the overall business model point of view.

There are indeed many advantages to using API architecture. The biggest is its flexibility and adaptiveness as well as fast development and deployment time that benefit both parties: the retailers that want to provide the most innovative experience to their customers and developers that need to “develop” these expectations into reality.

And since web APIs are built exactly around business capabilities, the retailers pay for exactly what they want — no more, no less.

This section will take a closer look at some of the most beneficial and key features of APIs, explaining what makes them so advantageous. Use this as your checklist when hunting for those perfect APIs to reinvent your commerce.

Key qualities of a well-designed
API & API architecture
Security
Scalability
Reliability
Usability
Testability



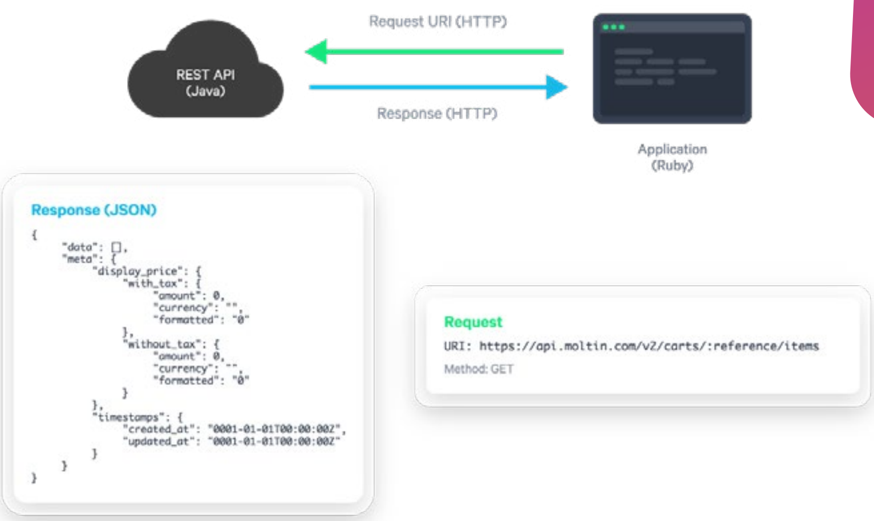
The best APIs are ...

Language agnostic

All APIs are interoperable across different platforms and systems that use various programming languages.

This interoperability works both ways, which means that a client and a server-side can both use different languages and still be able to communicate with each other.

REST API Model



```
{
    "data": [],
    "meta": {
        "display_price": {
            "with_tax": {
                "amount": 0,
                "currency": "",
                "formatted": "0"
            },
            "without_tax": {
                "amount": 0,
                "currency": "",
                "formatted": "0"
            }
        },
        "timestamps": {
            "created_at": "0001-01-01T00:00:00Z",
            "updated_at": "0001-01-01T00:00:00Z"
        }
    }
}
```

```
URI: https://api.moltin.com/v2/carts/:reference/items
Method: GET
```

Why is it good for me?
This particular feature is great for companies that employ many teams with different language preferences. It doesn't matter if a microservice is developed in Ruby, JavaScript, Python, PHP, or any other language or framework for that matter. As long as they communicate with each other through HTTP, they're going to work seamlessly with each other.

An additional advantage of being language-independent is that your in-house developers don't need to learn any new programming languages but instead utilize their existing expertise to focus solely on the feature, not on the tools.

Cacheable

All APIs are cacheable, which means that the resources are cached into the browser, and each time they're called, the browser can simply return it from its cache rather than getting the resource from the server again.

Why is it good for me?
This extremely vital feature of the API allows for a much faster, smoother and reliable performance.

Stateless

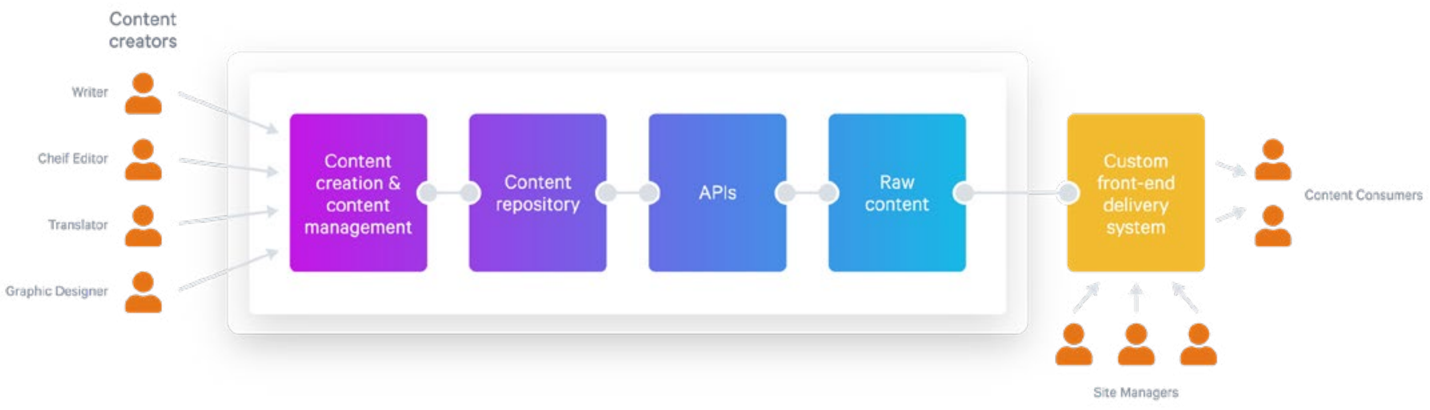
The statelessness of an API means that each time a resource is accessed through an endpoint, the API provides the same response: The same input will always produce the same output.

Why is it good for me?

Excluding the use of state in an application greatly improves the modularity of that app and makes it easier to scale out. Stateless apps are also easier to manage and can be worked on by individual developers independently.

Headless

Headless architecture unchains the front-end from the back-end by splitting the code cleanly between server-side code that defines data and business logic from the client-side code that defines how it can interact with the user.



Why is it good for me?

Headless offers a truly innovative way of managing the web content and user experience. Front-end developers can finally focus on providing highly innovative content without the need to worry about the back-end.

Due to a separate display logic, any headless application tends to work faster, is more responsive and easier to maintain or update. It is also better prepared for localization and translation to reach the global audience if need be.

Such a flexible approach is ideal for brands that aspire to evolve and grow in the future or all those who need a very specific front-end.

Lightweight and multidisciplinary

APIs are lightweight, which means they use up less memory of a device that they run on. They can also be deployed to multiple devices and are not constrained by any platform.

Why is it good for me?
The lightweight nature of an API allows for fast development and deployment time. Functionality contained within limited lines of code can be reused and injected into multiple devices, from traditional websites to social platforms, wearable devices, IoT, etc.

Additionally, a typical runtime is very different from big platforms, especially on load. An expected loading time can start from a few milliseconds with a load as little as a few kilobytes. Microservices, due to their reduced programmatic complexity and size, are easy to use, fast to load and straightforward to update.

Scalable

Studies show that it takes 2.8 seconds to distract a person. Slow-loading pages drive customers away. The more people use a page, making an increased number of requests, the more it becomes overloaded, becoming painfully slow.

Why is it good for me?
If an application is scalable, just as any API-driven architecture is, it can handle an exponential growth of online traffic and increased load.

Additionally, as a single service starts to experience more load, it can easily be scaled in isolation and without downtime. Requests will then be load-balanced across all available instances of this service.

To that point, if a single instance of a service experiences issues, it can be automatically removed from the pool to ensure minimal disruption for users. Services can be deployed to multiple geographic locations, meaning an issue in one availability zone does not have to affect the full application.

A great example of a scalable infrastructure is Skyscanner that currently has to cater to over 8.5 million visitors per month. Without a scalable architecture, the site would die as quickly as it rose in popularity without the needed speed and agility to provide best-suited flight suggestions.

Multiple integrations

Due to the modularity of the APIs, retailers can streamline their business by integrating as many APIs as they need from multiple providers to expand their ecosystems. There's no longer a requirement to stay loyal to only one provider constrained by one digital solution. It also removes the need to buy the entire technology stack.

Why is it good for me?
A retailer can cherry-pick the functionality they're interested in, and implement just the parts that are absolutely necessary for the healthy growth of their business, knowing that in time they can easily change or expand the approach taken.

The vast diversity of possible integration requirements and the demand for fast and seamless integrations cannot be met with a one-size-fits-all approach. Businesses that cannot adapt to the fast pace of technological progress are doomed to stay behind.

Independent and flexible

Decentralized and not constrained by a programming language, an API-based service is fairly easy to plug into any existing IT infrastructure, including a monolithic self-contained unit.

Why is it good for me?
This is great news for all those companies locked into big legacy platforms that want to expand their functionality.

The ever-expanding need to be innovative and different makes it more logical and cost-effective to build a software or a service around one's business rather than buy rigid, pre-packaged, third-party programs.

They're beneficial, but are APIs secure?

Since the main job of an API is to surface, expose and manipulate (update or delete) data, it brings up a logical question: Are APIs secure when allowing for such communication?

Typical to any web-based system, it all depends on the business-specific architecture. Jordan French, a CIO advisor[3], states that a secure API needs to effectively control six separate risks: data, operations, infrastructure, people, network and incident response. Insecure APIs are those that somehow failed to follow now-default API behavior.

In January 2014, insecure API infrastructure allowed for a data leak that affected almost five million users.[4] The hackers were able to match the usernames with their phone numbers, and while the API is not directly to be blamed, the way the ecosystem was designed and deployed made it insecure and easy to breach.

In June 2018, Ticketmaster announced their system was hacked in an attempt to steal credit card information from targeted third-party suppliers integrated with their website. The massive credit card hacking campaign affected more than 800 e-commerce sites. The attackers used malicious scripts that injected into e-commerce websites, allowing them to record the credit card data customers entered as part of the usual checkout process. Again, not an API fault, but the way it was implemented put thousands of customers at risk.

Below, we list most the important security defaults a typical secure API would have. Use it as a checklist when selecting an API provider. This is an absolute must-have when it comes to API security.

[3] Source: CIO From IDG
[4] Source: TechTarget|SearchSecurity

✓ Communication via HTTPS protocol

An API typically uses an encrypted HTTPS protocol for all its communication. The protocol ensures that all data passed between the web API and browser remains private and integral with no additional strain on system performance. Moreover, since most modern browsers use HTTPS as a preferred means of communication, following that trend makes the browsing experience faster and smoother.

✓ Authentication through an API Key

All trusted APIs will provide you with an API Key that increases security. An API Key is typically generated per user, and it's a unique value that authenticates a user. In other words, the Key proves that a user trying to get into the system is the same user as before. Without an API Key, it's relatively easy to exploit the system and steal user data, like usernames and passwords.

✓ Authorization through a Token

While providing a good security layer, an API key itself is not enough to make an API secure and breach-proof. A good API provider will also give a selection of possible authorization methods a developer can implement on their end. Authorization regulates a set of actions permitted per user type and complements authentication.

The most common authorization method is oauth, which includes elements of both authorization and authentication. It imposes a token on each API endpoint, which authenticates a user, and then, as part of the authorization process, expires with time and revokes privileges. A user needs to verify credentials so that a new token can be generated. Typically, the token enables client credentials an implicit type of authentication. These allow developers to impose further restrictions on the system, giving users the read-only permissions (implicit) or full access (client credentials), including updating and deleting the endpoints.

Authorization makes sure each endpoint in use is additionally protected even after a user has been authenticated through an API key.

Summary

It is clear that nowadays it is an agile API that drives the digital business and stands behind every major digital strategy. However, many enterprises fail to unlock the full potential of an API by implementing a half-baked, API-esque solution. By doing so, they put their customers at risk: leaking sensitive data or exposing vulnerable endpoints making them easy to hack and hijack.

Secure and stable APIs are the golden nugget of any industry, so make sure that the API you choose brings value to your company by monetizing any innovative experience you can think of for your customers.

The next book in the series will introduce you to Moltin's API solution, a secure and innovative way to engineer your way into the future.

moltin