🏷 pattern (/tags/pattern) 🏷 application api (/tags/application api) 🏷 inter-service communication (/tags/inter-service communication) 🏷 application architecture (/tags/application architecture)

Pattern: API Gateway / Backends for Frontends

Context

Let's imagine you are building an online store that uses the Microservice architecture pattern (microservices.html) and that you are implementing the product details page. You need to develop multiple versions of the product details user interface:

- HTML5/JavaScript-based UI for desktop and mobile browsers - HTML is generated by a server-side web application
- Native Android and iPhone clients - these clients interact with the server via REST APIs

In addition, the online store must expose product details via a REST API for use by 3rd party applications.

A product details UI can display a lot of information about a product. For example, the Amazon.com (http://Amazon.com) details page for POJOs in Action (http://www.amazon.com/POJOs-Action-Developing-Applications-Lightweight/dp/1932394583) displays:

- Basic information about the book such as title, author, price, etc.
- Your purchase history for the book
- Availability
- Buying options
- Other items that are frequently bought with this book
- Other items bought by customers who bought this book
- Customer reviews
- Sellers ranking
- …

Since the online store uses the Microservice architecture pattern the product details data is spread over multiple services. For example,

- Product Info Service - basic information about the product such as title, author
- Pricing Service - product price
- Order service - purchase history for product

- Inventory service - product availability
- Review service - customer reviews …

Consequently, the code that displays the product details needs to fetch information from all of these services.

Problem

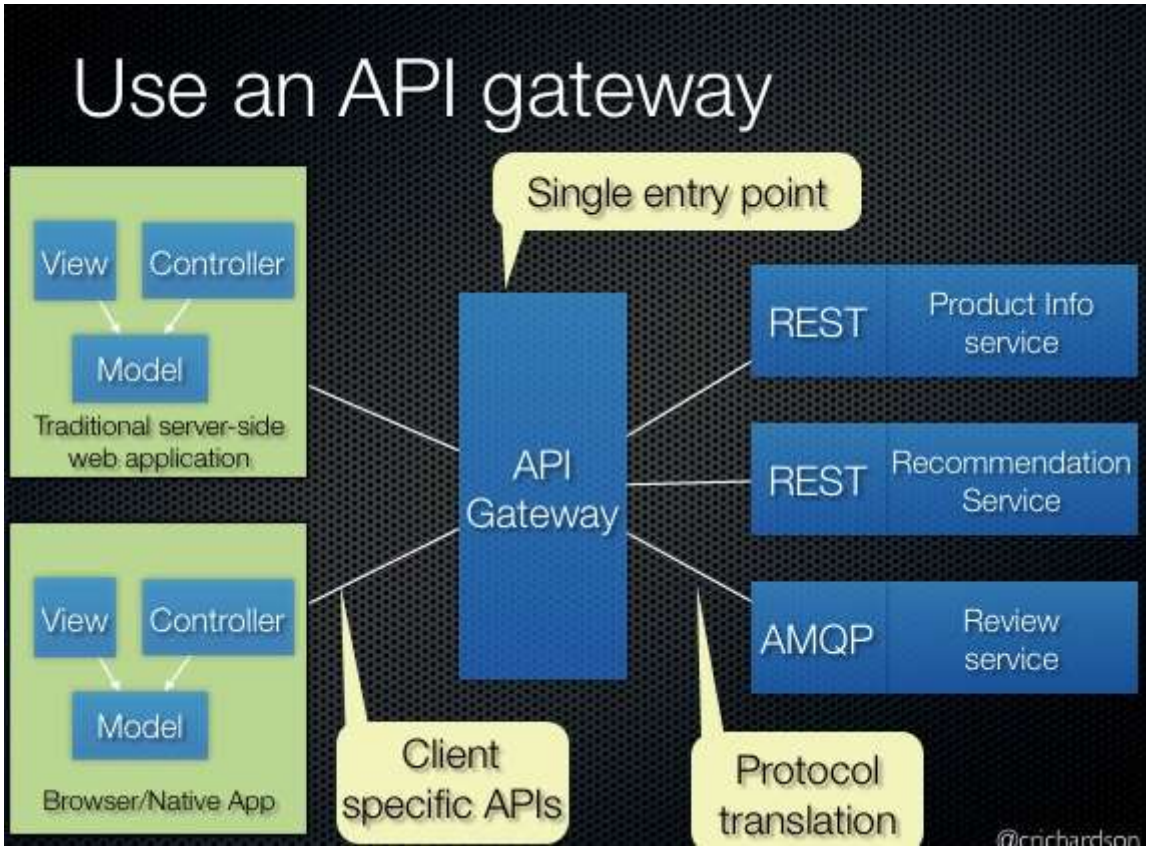
How do the clients of a Microservices-based application access the individual services?

Forces

- The granularity of APIs provided by microservices is often different than what a client needs. Microservices typically provide fine-grained APIs, which means that clients need to interact with multiple services. For example, as described above, a client needing the details for a product needs to fetch data from numerous services.
- Different clients need different data. For example, the desktop browser version of a product details page desktop is typically more elaborate then the mobile version.
- Network performance is different for different types of clients. For example, a mobile network is typically much slower and has much higher latency than a non-mobile network. And, of course, any WAN is much slower than a LAN. This means that a native mobile client uses a network that has very difference performance characteristics than a LAN used by a server-side web application. The server-side web application can make multiple requests to backend services without impacting the user experience where as a mobile client can only make a few.
- The number of service instances and their locations (host+port) changes dynamically
- Partitioning into services can change over time and should be hidden from clients
- Services might use a diverse set of protocols, some of which might not be web friendly

Solution

Implement an API gateway that is the single entry point for all clients. The API gateway handles requests in one of two ways. Some requests are simply proxied/routed to the appropriate service. It handles other requests by fanning out to multiple services.



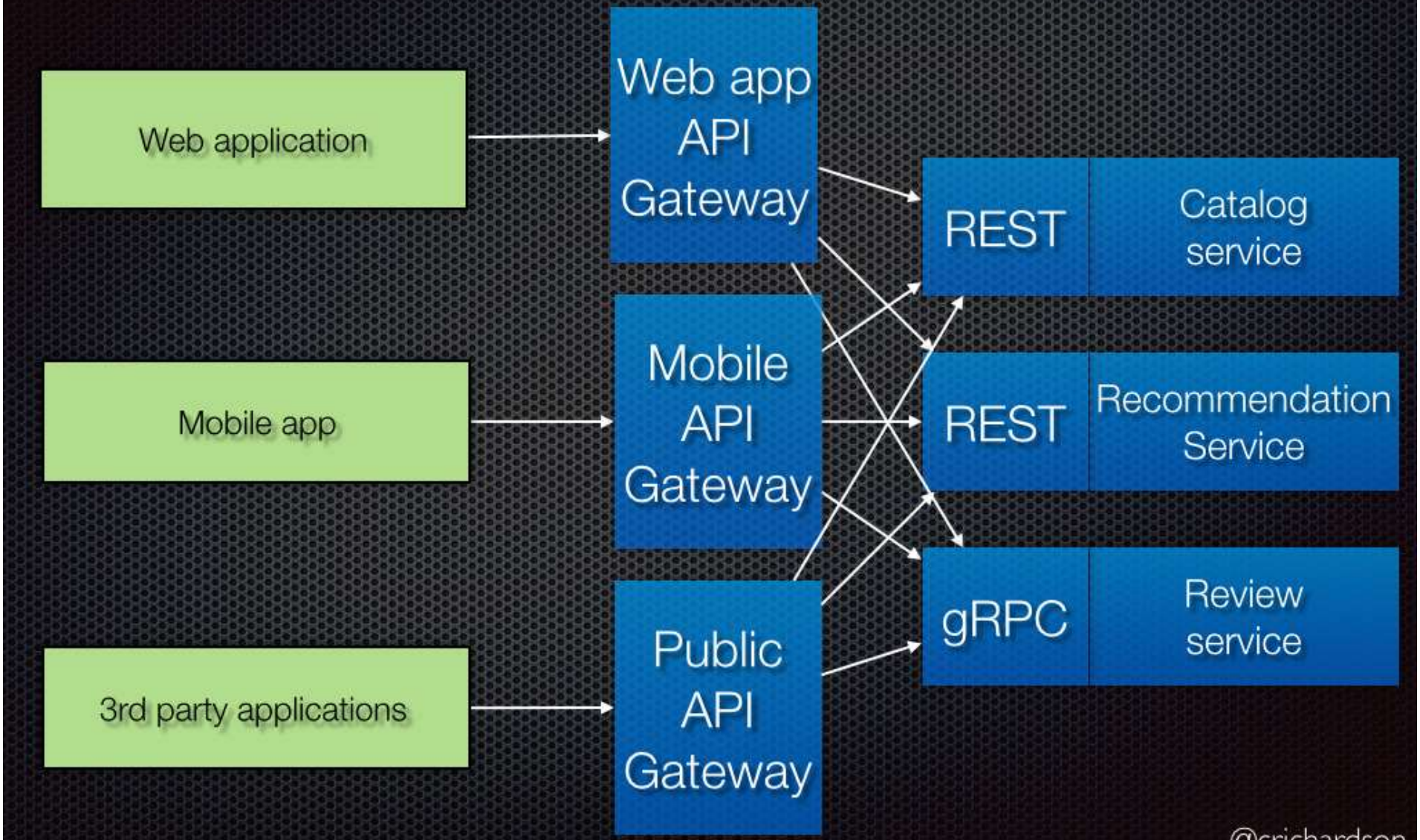
Rather than provide a one-size-fits-all style API, the API gateway can expose a different API for each client. For example, the Netflix API (http://techblog.netflix.com/2012/07/embracing-differences-inside-netflix.html) gateway runs client-specific adapter code that provides each client with an API that's best suited to its requirements.

The API gateway might also implement security, e.g. verify that the client is authorized to perform the request

Variation: Backends for frontends

A variation of this pattern is the Backends for frontends pattern. It defines a separate API gateway for each kind of client.

Variation: Backends for frontends



In this example, there are three kinds of clients: web application, mobile application, and external 3rd party application. There are three different API gateways. Each one is provides an API for its client.

Examples

- Netflix API gateway (http://techblog.netflix.com/2013/01/optimizing-netflix-api.html)
- A simple Java/Spring API gateway (https://github.com/cer/event-sourcing-examples/tree/master/java-spring/api-gateway-service) from the Money Transfer example application (https://github.com/cer/event-sourcing-examples).

Resulting context

Using an API gateway has the following benefits:

- Insulates the clients from how the application is partitioned into microservices
- Insulates the clients from the problem of determining the locations of service instances
- Provides the optimal API for each client
- Reduces the number of requests/roundtrips. For example, the API gateway enables clients to retrieve data from multiple services with a single round-trip. Fewer requests also means less overhead and improves the user experience. An API gateway is essential for mobile applications.
- Simplifies the client by moving logic for calling multiple services from the client to API gateway
- Translates from a “standard” public web-friendly API protocol to whatever protocols are used internally

The API gateway pattern has some drawbacks:

- Increased complexity - the API gateway is yet another moving part that must be developed, deployed and managed
- Increased response time due to the additional network hop through the API gateway - however, for most applications the cost of an extra roundtrip is insignificant.

Issues:

- How implement the API gateway? An event-driven/reactive approach is best if it must scale to scale to handle high loads. On the JVM, NIO-based libraries such as Netty, Spring Reactor, etc. make sense. NodeJS is another option.

Related patterns

- The Microservice architecture pattern (microservices.html) creates the need for this pattern.
- The API gateway must use either the Client-side Discovery pattern (client-side-discovery.html) or Server-side Discovery pattern (server-side-discovery.html) to route requests to available service instances.
- The API Gateway may authenticate the user and pass an Access Token (security/access-token.html) containing information about the user to the services

- An API Gateway will use a Circuit Breaker (reliability/circuit-breaker.html) to invoke services
- An API gateway often implements the API Composition pattern (/patterns/data/api-composition.html)

Known uses

- Netflix API gateway (http://techblog.netflix.com/2012/07/embracing-differences-inside-netflix.html)

Example application

See the API Gateway that part of my Microservices pattern's example application (https://github.com/microservice-patterns/ftgo-application). It's implemented using Spring Cloud Gateway.

🏷 pattern (/tags/pattern) 🏷 application api (/tags/application api) 🏷 inter-service communication (/tags/inter-service communication) 🏷 application architecture (/tags/application architecture)

ALSO ON **MICROSERVICES**

Transactional outbox

4 years ago • 9 comments

Pattern: Transactional outbox Also known as Application events …

Polling publisher

5 years ago • 2 comments

Pattern: Polling publisher Context You have applied the Transactional Outbox …

Strangler application

5 years ago • 2 comments

About Microservices.io Microservices.io is brought to you by Chris …

Refactoring a monolith to microservices

4 years ago • 1 comment

Refactoring a monolith to microservices Note: This page is work in progress. …

In August: des microservice .

10 months ago • 1

August 29th: pul designing a micr architecture wor

41 Comments

Login ▾

G

Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS

Name

♡ 25 Share

Best Newest Oldest

A

Asha Sebastian

5 years ago

Hi Chris,
Nice article. I've one doubt. If you have a micro service supporting write intensive data ingestion flows and another set of microservices which caters to read requests, is it better to keep them segregated under two different API gateways? People are suggesting to keep it separate but then we have to duplicate authentication logic etc across both gateways. I didn't understand the benefit we get by keeping it separate based on the call rate of APIs or based on read and write flows. Gateway is scalable based on our requirement, right? I would like to know your thoughts on it.

25 8 Reply • Share ›

Philip Patrick ↗ Asha Sebastian

3 years ago

I didn't see a need to separate gateways either. But the problem with duplicating authentication logic, can be solved by extracting all authentication into a separate microservice. You will also gain encapsulation here and one place to change if you will add more authentication providers in the future. Plus a flexibility to pick any gateway

you want - yours, or maybe one from Azure or whatever.

0 0 Reply • Share ›

S **Shravan Dhar** → Asha Sebastian

3 years ago

Separating them based on increased traffic doesn't make sense to me and you're right that you can increase the docker instances for your API Gateway. There is a reason it is called "single point of entry" and you should abide by that. In this way you'll reduce the number of requests to the Authentication service and the mess to syncronize between the two API Gateways

0 0 Reply • Share ›

Love Hasija

4 years ago

I am not really a great fan of this architectural pattern as a standardization of micro-service integration. It might be a good architecture at certain places where the need is to offer composite services to the end-user who might not be capable enough to query multiple data sources. As in the case of the client facing consumer services used by the mobile client and web client, it very well makes sense to setup an API Gateway rather than making 10 - 20 queries and handling the versioning changes. However, when it comes to internal service communication, it is definitely another piece of ever growing middleware component acting as a bottleneck, yet another ESB. I like the way Clemens Vesters has put it here https://www.youtube.com/wat...

4 0 Reply • Share ›

Chris Richardson Mod → Love Hasija

4 years ago

The API gateway pattern is not for internal inter-service communication. Please see the problem section.

11 1 Reply • Share ›

MK → Chris Richardson

4 years ago

Great article Chris. Few questions:
1. Aren't web and mobile apps in an enterprise internal consumers (apps) trying to get to the APIs? (So should it not fall under internal service communication - note not referring to inter-service communication)
2. If these Catalog Service, Recommendation Service etc are independent Product/Domain DevOps teams say running their own deployment clusters, wouldn't having another team manage API gateway create unnecessary dependency for them?
3. What are your thoughts on more decentralized API Gateway Pattern? Product/Domain driven API Gateways?

4 1 Reply • Share ›

Love Hasija → Chris Richardson
4 years ago

Yep, missed that. It absolutely makes sense. Great article.

2 0 Reply • Share ›

D **disqus_0Ou0Uh363I** → Love Hasija
4 years ago

lovehasija already

0 0 Reply • Share ›

Grumpys view
4 years ago

Starts to look like a monolith structure in disguise

11 5 Reply • Share ›

Apostolis Anastasiou → Grumpys view
2 years ago edited

How exactly? The difference is pretty clear. In a monolithic structure you

- Can't have multiple teams working on different features, you have one big monolithic team.
- There are no separate domains with separate teams, like b2b, b2c, without breaking the whole application (hem hem, like 2 microservices).
- You can't have different languages, and different solutions for each problem. You use the same technologies and language for all the services.
- All the team is working on one repository and complexities and conflicts occur all the time

I dont see how in any way this pattern looks anything like a monolith. It's clearly very freaking different in so many ways. Just because it has 1-2 similarities that you thought of, it doesn't mean it's anything even remotely close to a monolith. I really can't make any sense of these oversimplified arguments.

1 0 Reply • Share ›

Usman Bashir

5 years ago

Would not this add another bottleneck if all the required services are being channelled through the same API. Even though this is facade yet it may interfere the basic principle of micro-services pattern itself to distribute your application into smaller chunks so you can provide clear separation of concern. In this way, you may have modelled a great architecture hidden but ending up exposing through the same channel of single API which may fell down if other API's failed. The only way you can avoid is still exposing those API's separately by allowing gateway to manage the traffic and truly reflecting its gateway nature.

2 0 Reply • Share ›

G

Greg → Usman Bashir

5 years ago

The API Gateway should not be yet another single point of failure. In order to do that, you need to apply High Availability (e.g. by having two gateway instances in active/passive mode) and Circuit Breaker (e.g. fail fast for the back-end service that is down or takes too long to respond).

In my opinion, the advantages (e.g. multiple protocol support, basic orchestration or service wrapping, discovery, circuit breaking, throttling) of having an API Gateway in front of your microservices outnumber the disadvantages (mostly monitoring/alerting since development is pretty straight-forward and is more of a configuration kind of thing).

On a side note, the API Gateway should better be reactive so that it can sustain a huge load of requests and be easily scalable.

6 0 Reply • Share ›

Chris Richardson Mod → Greg

5 years ago

For high availability, you can simply run multiple API Gateways (e.g. a Kubernetes Deployment with multiple replicas) behind a load balancer (e.g an AWS ELB).

Using a reactive architecture might be a good idea. e.g. see https://www.infoq.com/news/...

One obvious point, an API gateway should be designed to meet its requirements, which may or may not be high scalability :-)

8 0 Reply • Share ›

G

Greg → Chris Richardson

5 years ago

AWS ELB, HAProxy or anything basically that could guarantee that there is at least one API Gateway instance running.
Regarding scalability, this may not be a hard requirement but it's always good to be pro-active (and re-active :-P)

1 0 Reply • Share ›

Manuel Silva → Chris Richardson

4 years ago

(y)

0 0 Reply • Share ›

U **usama sheikh** → Usman Bashir

4 years ago

Chuss he maari ap ny

0 0 Reply • Share ›

C **Chuck Zheng**

4 years ago

Thanks to Chris putting up this info hub of many microservices related patterns for sharing and active discussions. eBay has reasonably successfully extended this pattern and the Server-side page fragment composition pattern to support mobile native app, mobile web, and desktop web to both
- providing consistent customer experience and faster response (with less roundtrips across internet/cellular networks)
- and reducing duplicated engineering develop/test/deployment/maintenance cost on
- not only domain functional logic for building the user interface by orchestrating invocation to downstream domain entity & process microservices, data synthesization form those responses to form what's needed by the front-end
- but also complex cross cutting issues of L10N/i18n, User Behavior Tracking, Experimentation, etc.

There is a blog about it. And now we are moving onto building more fine-grain module provider microservices and solidifying eBay's business entity microservice pattern to work with newer more scalable data storage technology. We already leverage many patterns captured here and exploring more.

1 0 Reply • Share ›

 **Abhishek Upadhyay** → Chuck Zheng

4 years ago edited

It is like SOA service with REST interface. The presentation work is also not design specific now - With progressive web apps (framework such as Ionic), in my company we write once and run on all devices (Android, iOS, and Web). PWA (Onrefront end) with one backend (experience services) for all channels

0 0 Reply • Share ›

C **Chuck Zheng** → Abhishek Upadhyay

4 years ago

Our Mobile Native App teams evaluated HTML5 based solution and think it's still inferior to actual Mobile Native App, so they reject HTML5 based solution (like ionic).

0 0 Reply • Share ›

D **Diptnndu Dtta** → Chuck Zheng

4 years ago

Would HTML5 based client solution aid the use of Websockets for communication?

0 0 Reply • Share ›

Manuel Silva → Chuck Zheng

4 years ago

nice

0 0 Reply • Share ›

hadafnet

4 years ago

hi dear
I want to know which API gateway is best for dotnet core microservices
Ocelot or Kong
Masoud from Hadafnet

1 1 Reply • Share ›

A

Allan Chua

5 years ago

Hi,

This article helped me a lot in cracking the concept of API gateways.

In fact, I've written one series of articles that explains API gateways and how to build one using ASP.net core.

Link:
https://pogsdotnet.blogspot...

1 1 Reply • Share ›

W

Wojciech Pacynko

7 months ago

In big, global companies providing many products and services, building many customer facing systems how to work with api gateways? Can we have API gatwey behind API gateway? Like a Gate way for a domain och solution?

0 0 Reply • Share ›

N

NITIN AGRAWAL

2 years ago

Hi Chris,
Do u really think that API gateway is a good to have as it adds one extra hop. For an application involving more data trnsfer between services will be slower due to addition of a hop.

0 0 Reply • Share ›

K

Kevin Chou

2 years ago

Type report: Forces #2: "...more elaborate then the mobile version".

0 0 Reply • Share ›

Adrian

2 years ago

Take a look at: https://community.axway.com...

0 0 Reply • Share ›

Iqc

2 years ago

To anyone looking at the Netflix example from 2012, 8 years later they have moved to a next iteration of this approach - Federated GraphQL Gateway:

https://netflixtechblog.com...

Worth looking at.

0 0 Reply • Share ›

G

Gerald Witichis

3 years ago edited

Since JS in web browsers does not allow fetches from multiple domains, because of cross site scripting protection what you call a pattern here is a technical necessity and not a solution which is optional or has alternatives by default.

A javascript web app talks to it's server and nothing else - no pattern involved.

Anyway having a database/table per service results in all services aggregated again in the frontend service.
So no pattern here but a technical consequence with no alternative.

0 0 Reply • Share ›

Sanjo Ganguly

3 years ago

Hi Cris, Do we really need to 3 separate gateway for Mobile & Web and 3rd API

0 0 Reply • Share ›

Sanjo Ganguly

3 years ago

Hi Chris, I have question whether do we need multiple public gateway for Mobile Apps , Web Application and 3rd party gateway

0 0 Reply • Share ›

Number23 → Sanjo Ganguly

3 years ago

Sanjo, from my reading of the article. He presented two options.

Either one API gateway for all, or 3 gateways for the different functions.

So it's up to you. Depending on your use case.

0 0 Reply • Share ›

L

luigi

3 years ago

Hi , im new in the API's gateway topic , the way of contain many API's definitions for many microcervices is not a problem in a microservices philosophy where we have all components separed?

0 0 Reply • Share ›

I

Iman Borumand

3 years ago

hi
how check authenticate and authorize in every microservice?

authorize in gateway or every microservice?

0 0 Reply • Share ›

V

Victor Tseng

3 years ago

Just found a potential typo:

Just found a potential typo:

> An event-driven/reactive approach is best if it must scale to scale to handle high loads.

(duplicated "scale to")

0 0 Reply • Share ›

J

John Hammond

3 years ago edited

Great article! Api Gateway pattern is really well explained.

There is a .Net Core 3 Api Gateway package called **AspNetCore.ApiGateway**.

Helps you build an Api Gateway.

Source code on GitHub:
https://github.com/VeritasSoftware/AspNetCore.ApiGateway

0 0 Reply • Share ›

B

binnyg

3 years ago

Great article. I have a question about the deployment architecture. Where should we deploy the API Gateway? In DMZ or within the intranet?