Mathematical Explanation of API Gateway Aggregation and how it speeds up the whole process.

by

Khaqan Ashraf

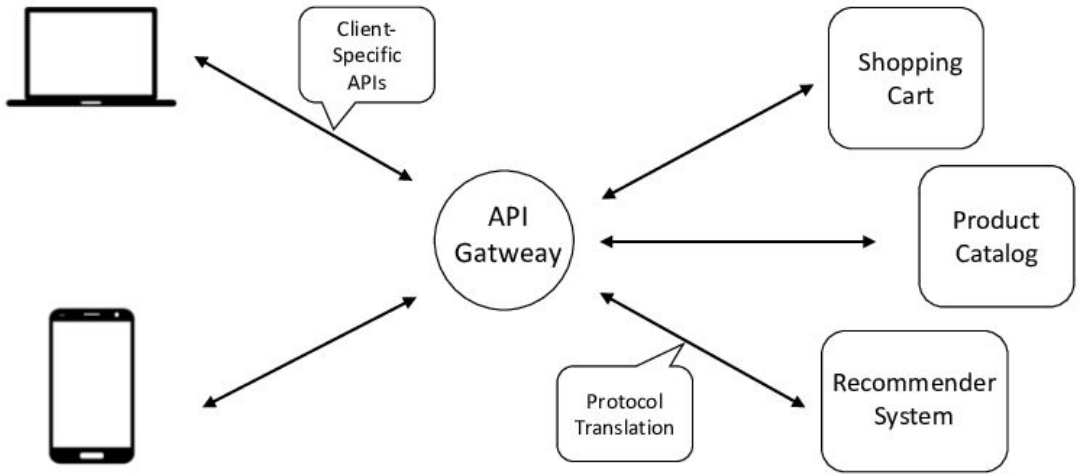
Software Engineer at **tossdown Inc.**

What will you learn

- What is API Gateway.
- What are Microservices.
- Difference between Monolithic and Microservices.
- Applications, Limitations and Solutions to the Limitations of Microservices Architecture.
- Aggregation.
- Mathematical representation of Aggregation and the analysis of speed.

What is an API Gateway?

An API gateway takes all API calls from clients, then routes them to the appropriate microservice with request routing, composition, and protocol translation and then return aggregated response.



Pic credit: https://www.nginx.com/learn/api-gateway/

Microservices

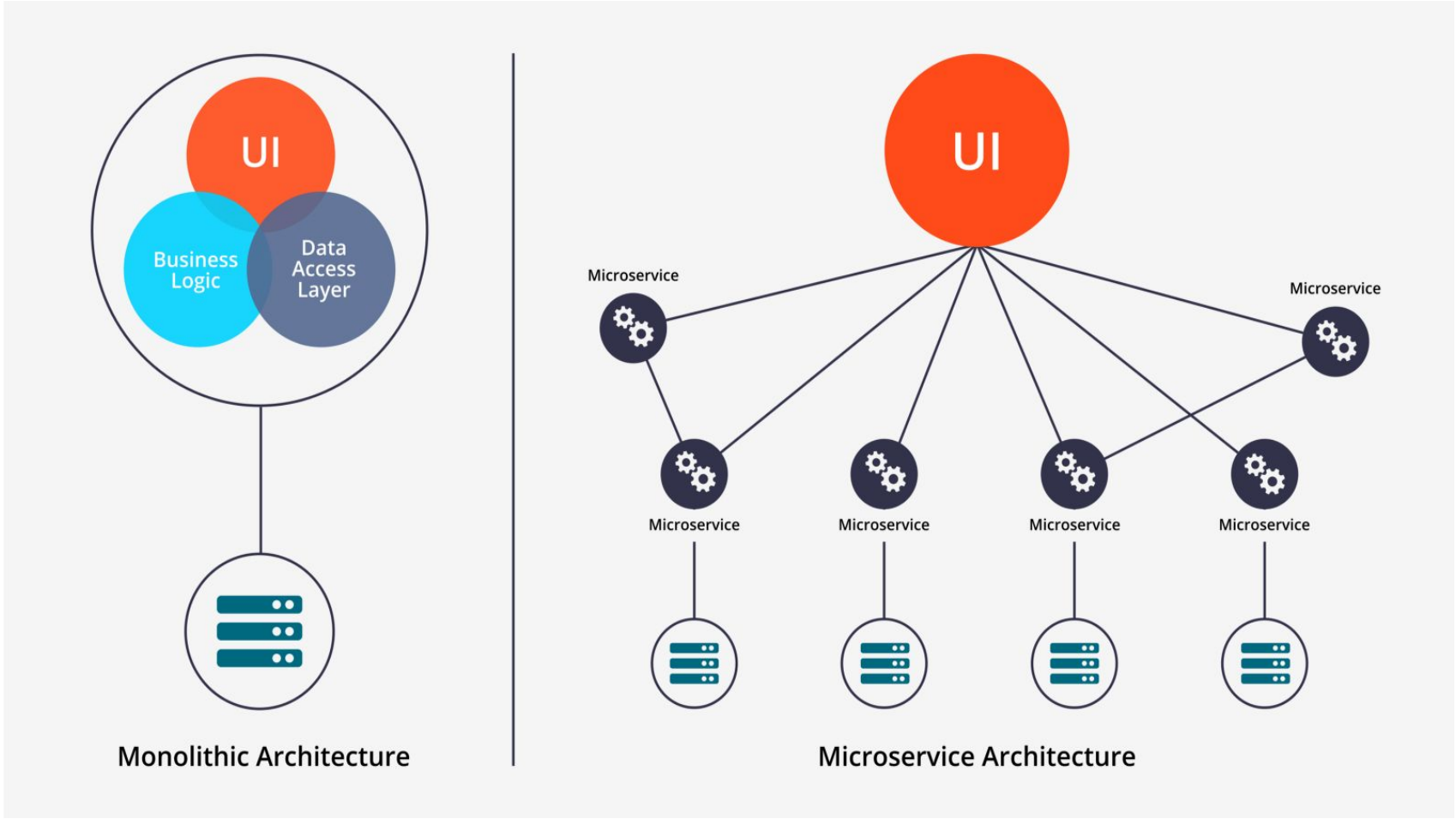
Microservices are the independent service modules that provide one or more functionality of the whole application. Microservices work independently from other services. As a whole, microservices coordinates with each other to provide system functionality. The underline evolving architecture is called as **Microservices Architecture.**

Pic credit: http://www.sevenhillsehealth.com/modules.html

Monolithic vs Microservices

In traditional Monolithic systems the whole application must be integrated, tested and then deployed. And for every other requirement after adding modules to the system the whole system must be tested and deployed again.

Reference: http://www.sevenhillsehealth.com/modules.html



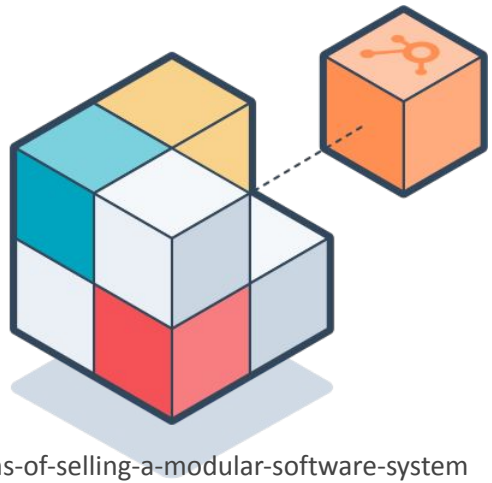
Pic credit: https://www.educative.io/blog/how-to-design-a-web-application-software-architecture-101

Benefits of Microservices

Microservices Architecture has following benefits over Monolithic Architecture:

1. Independent Modules

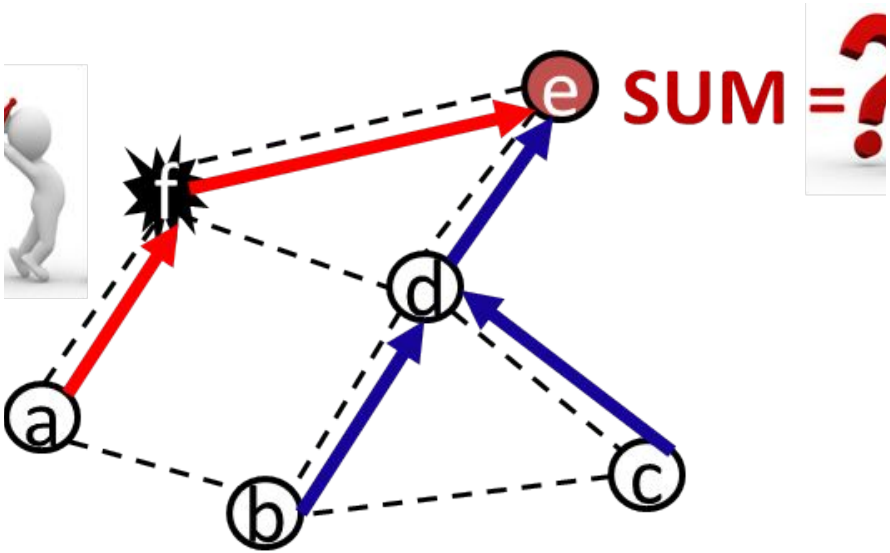
Microservices are independent service modules running independently on multiple servers or multiple server instances.



Reference: https://devan.codes/blog/2019/1/31/the-problems-of-selling-a-modular-software-system

2. Fault Tolerance

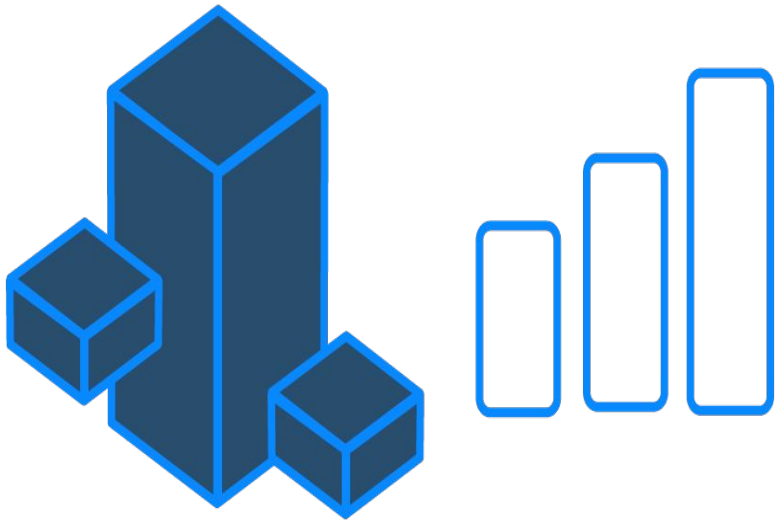
If there's a memory leak or something similar issue with a service then the issue would not be propagated to other services. Also, if a service shuts down due to any reason then it would not force the whole system to shut down.



Pic credit: https://publish.illinois.edu/integrative-security-assessment/fault-tolerant-communication/

3. Scalability

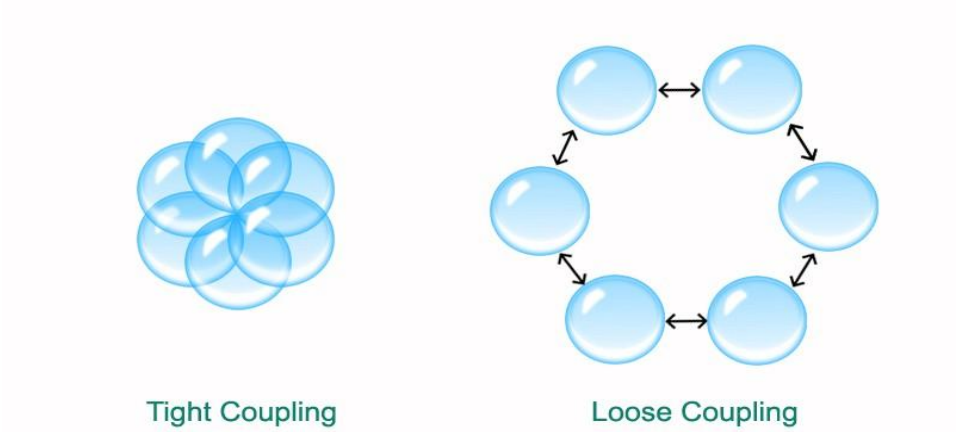
New functionality could be added to system easily and independently.



Pic credit: https://www.educative.io/path/scalability-system-design

4. Loosely Coupled

Microservices are loosely coupled as it is working independently of each other.



Pic credit: https://dzone.com/articles/all-you-need-to-know-about-dependency-injection

5. Maintenance

Microservices is not high maintenance. It could be easily taken care off and tested individually and separately.

Pic credit: https://dzone.com/articles/all-you-need-to-know-about-dependency-injection

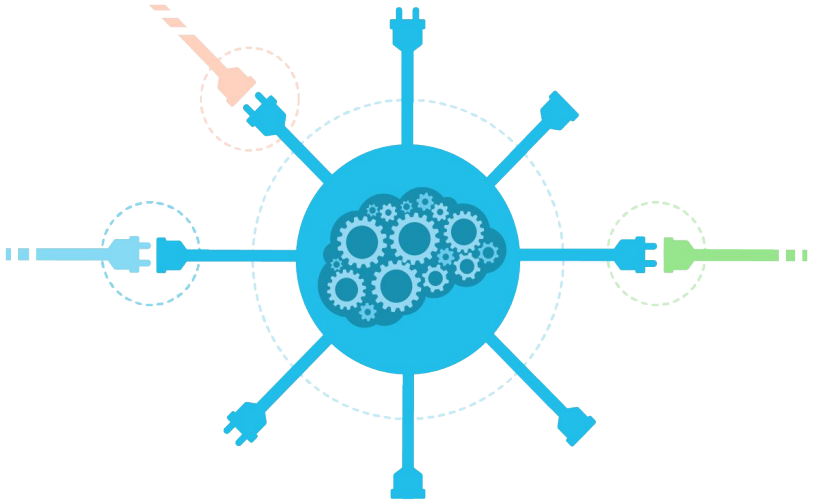
Example:

Let’s assume in a fully functional complex system we are required to add a module for image processing. Now we must use faster GPU and deep learning libraries like Kares or Tensorflow of Python while our rest of the application is built in PHP. It might be very difficult on Monolithic architecture but unfortunately Microservices architecture allow us to implement this service independently on faster system with Python libraries and provide a standard API so that other microservices of the system and client’s application could use this functionality.

APIs

In microservices architecture APIs are used to access and use functionality of a microservices. APIs provide standardized interface responsible for inter microservice communication as well.

For microservices we use RESTful APIs.

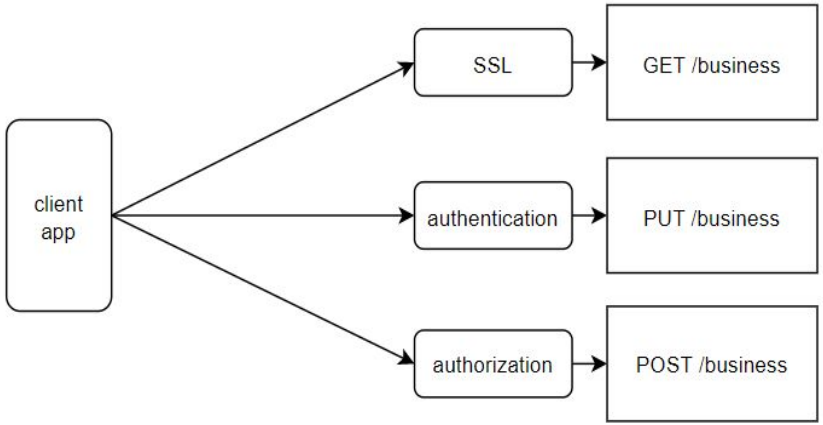


https://realtimeapi.io/what-is-a-realtime-api/

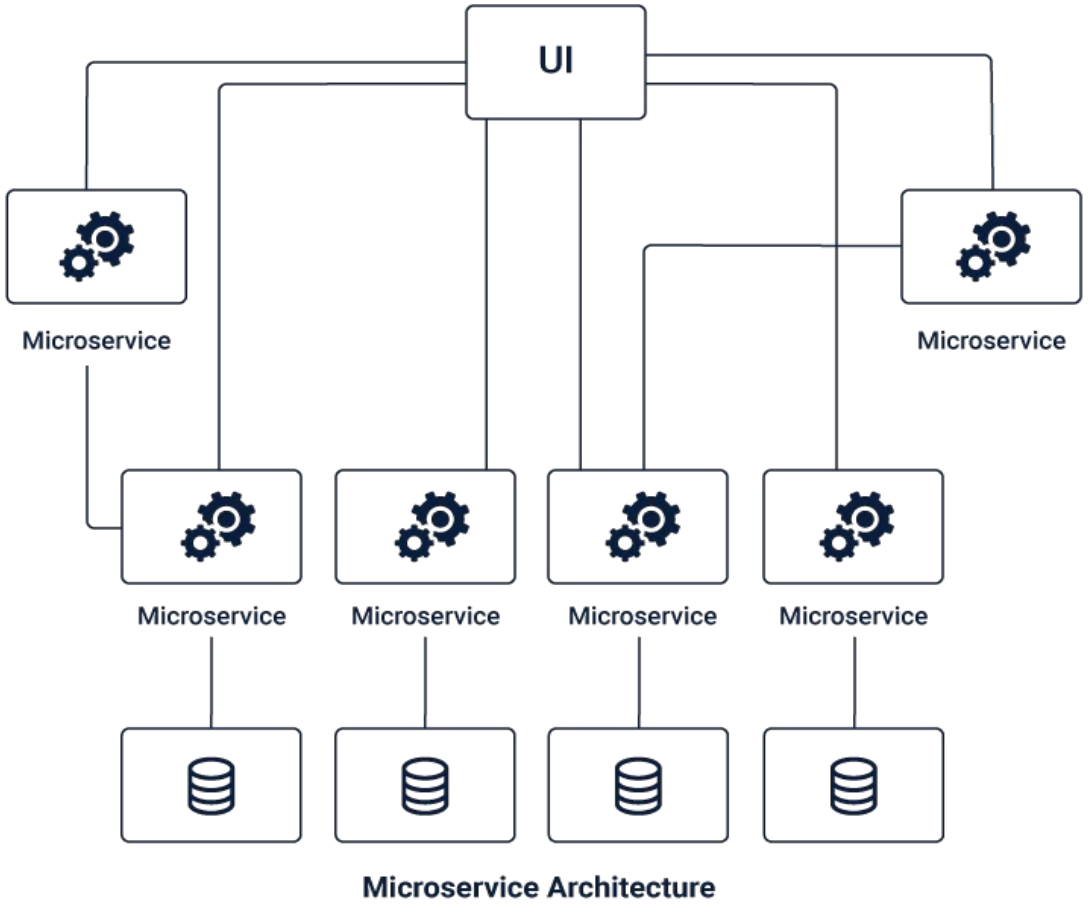
Limitations of the Architecture

Microservices architecture is so flexible such that it could be used in complex application and we can extend and evolve those applications over time. But with the mentioned features it has some points that must be covered before using Microservices as an architecture.

1. We have to perform some security check on following APIs separately
   a. SSL on GET APIs
   b. Authentication on PUT APIs
   c. Authorization on POST APIs



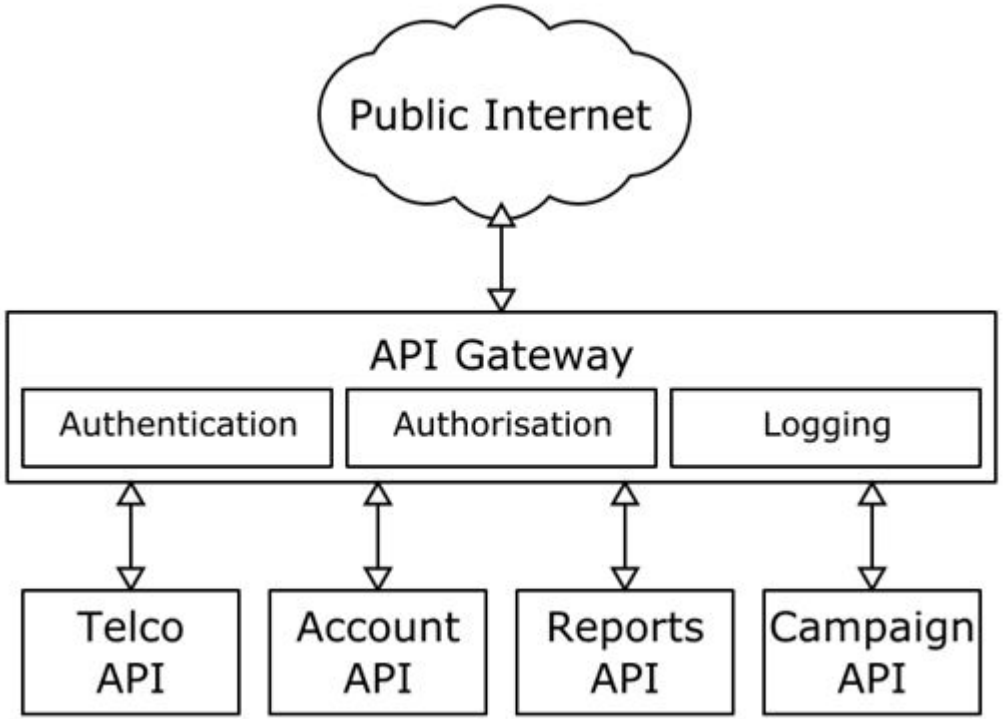
2. If the client application uses APIs directly then the client application is tightly coupled with the system.



PIc credit: https://www.cuelogic.com/blog/microservices-in-practice-from-architecture-to-deployment

3. We have to enable following services separately on each microservice.

a. Caching
b. Logging
c. Throttling (Rate limiting)

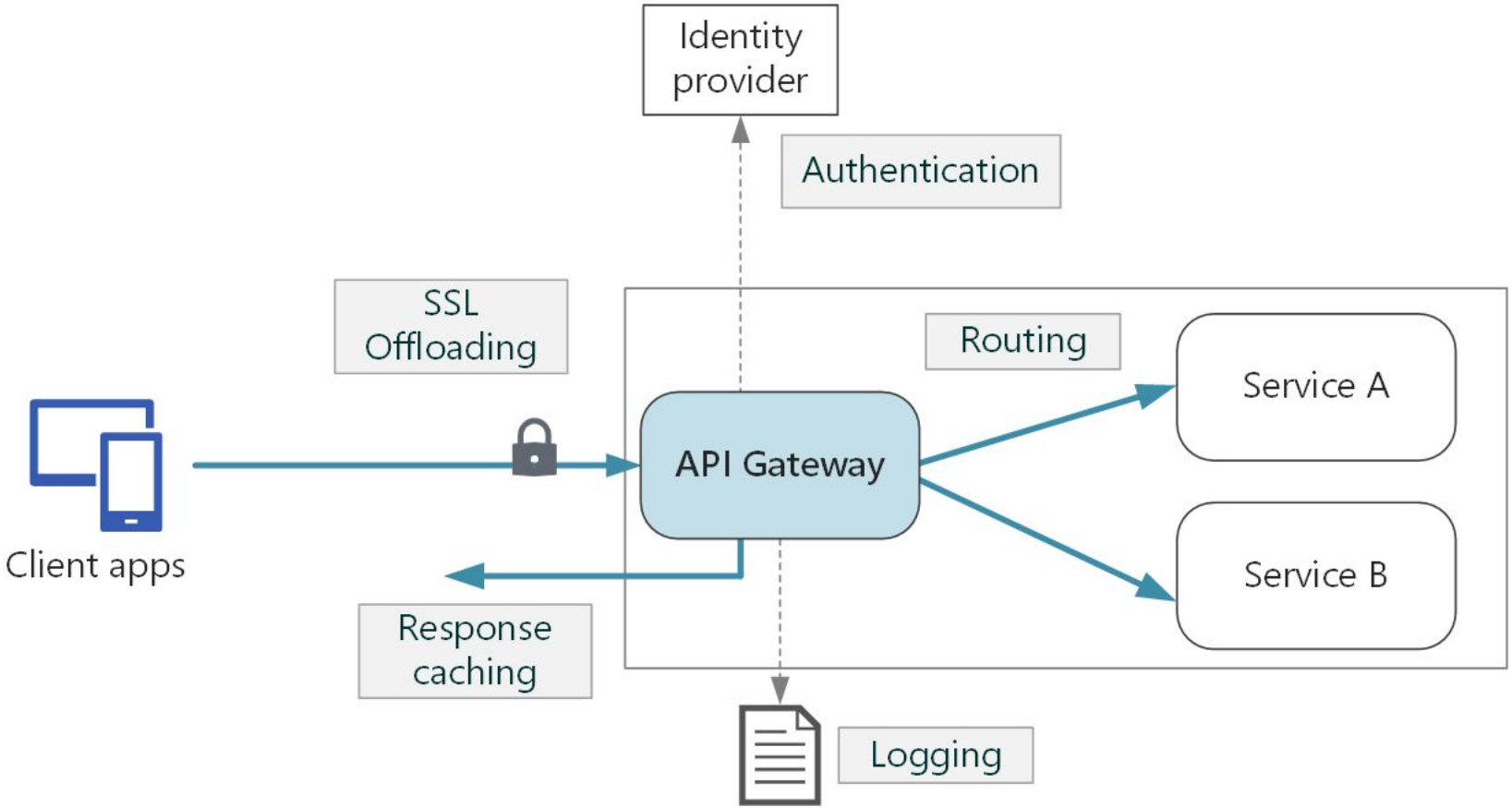


Pic credit: https://medium.com/@craig552uk/how-to-choose-an-api-gateway-ac64f04f3683

Solutions to combat the limitations

To address the mentioned issues, here comes the use of API Gateway:

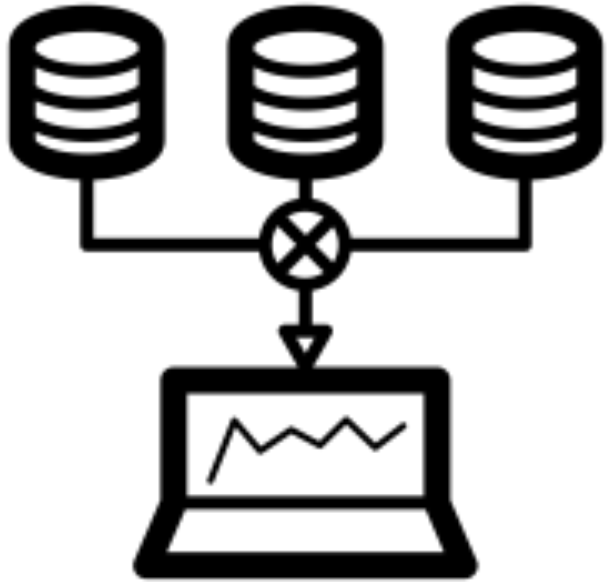
1. It authenticate requests for once
2. Caching and logging are handled on API Gateway as well
3. And client application is loosely coupled with the system through API Gateway



Pic credit: https://docs.microsoft.com/en-us/azure/architecture/microservices/design/gateway

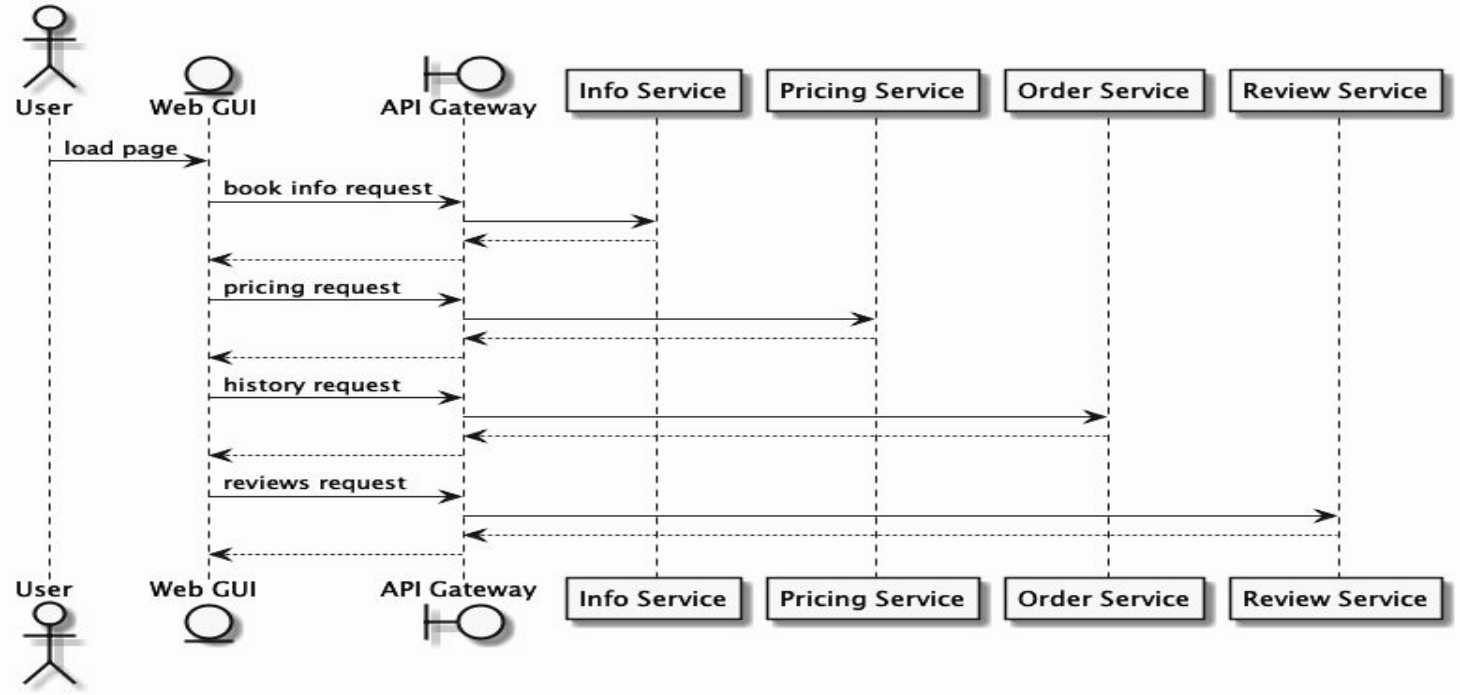
Aggregation

Another benefit of using API Gateway is aggregation. In response to one request API Gateway calls one or more apis, aggregate their response and return this aggregated response to client.



Pic credit: https://thenounproject.com/term/data-aggregation/1630955/

Without aggregation

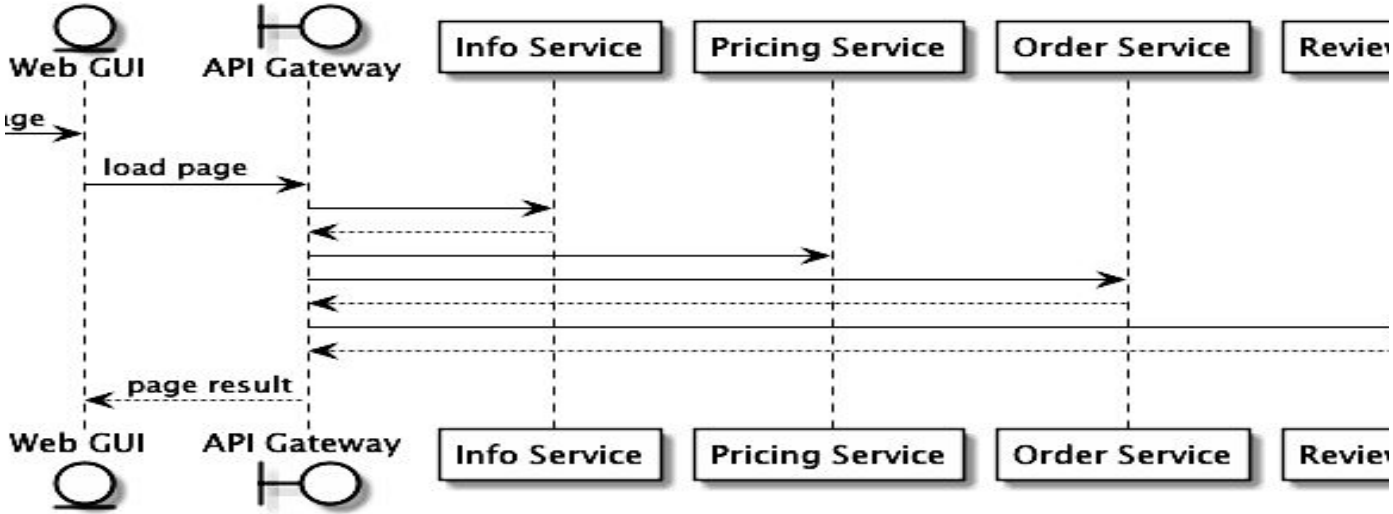


p = time to send packet from WebGUI to API-Gateway
q = time to send packet from API-Gateway to Service
n = number of request to different services
R1= total response time

$R1 = 2n(p+q)$

Pic credit: https://sookocheff.com/post/api/overambitious-api-gateways/

With aggregation



p = time to send packet from WebGUI to API-Gateway
q = time to send packet from API-Gateway to Service
n = number of request to different services
R2= total response time

$$R2 = 2(p+nq)$$

Pic credit: https://sookocheff.com/post/api/overambitious-api-gateways/

Mathematical Analysis

lets take,

$$\frac{R1}{R2} = \frac{2n(p+q)}{2(p+nq)}$$

$$\frac{R1}{R2} = \frac{n(p+q)}{p+nq}$$

substitute $p = 2q$,

$$\frac{R1}{R2} = \frac{n(2q+q)}{2q+nq}$$

$$\frac{R1}{R2} = \frac{3n}{2+n}$$

Result

$$\frac{R1}{R2} = \frac{3n}{2+n}$$

Sr.	Number of requests (n)	Without API Gateway aggregation (R1)	With API Gateway aggregation (R2)
1	**3**	**9**	**5**
2	**4**	**12**	**6**
3	**5**	**15**	**7**

Conclusion

API Gateway is a adequate service for microservices architecture. It enhances overall response time of complex applications and makes client application loosely coupled with system that too with all the essential security checks. Furthermore, we can use Load Balancer along with API Gateways to ensure application’s reliability.

Popular API Gateway services

1. NGINX
2. Amazon API Gateway
3. KrakenD
4. Kong
5. Apache APISIX