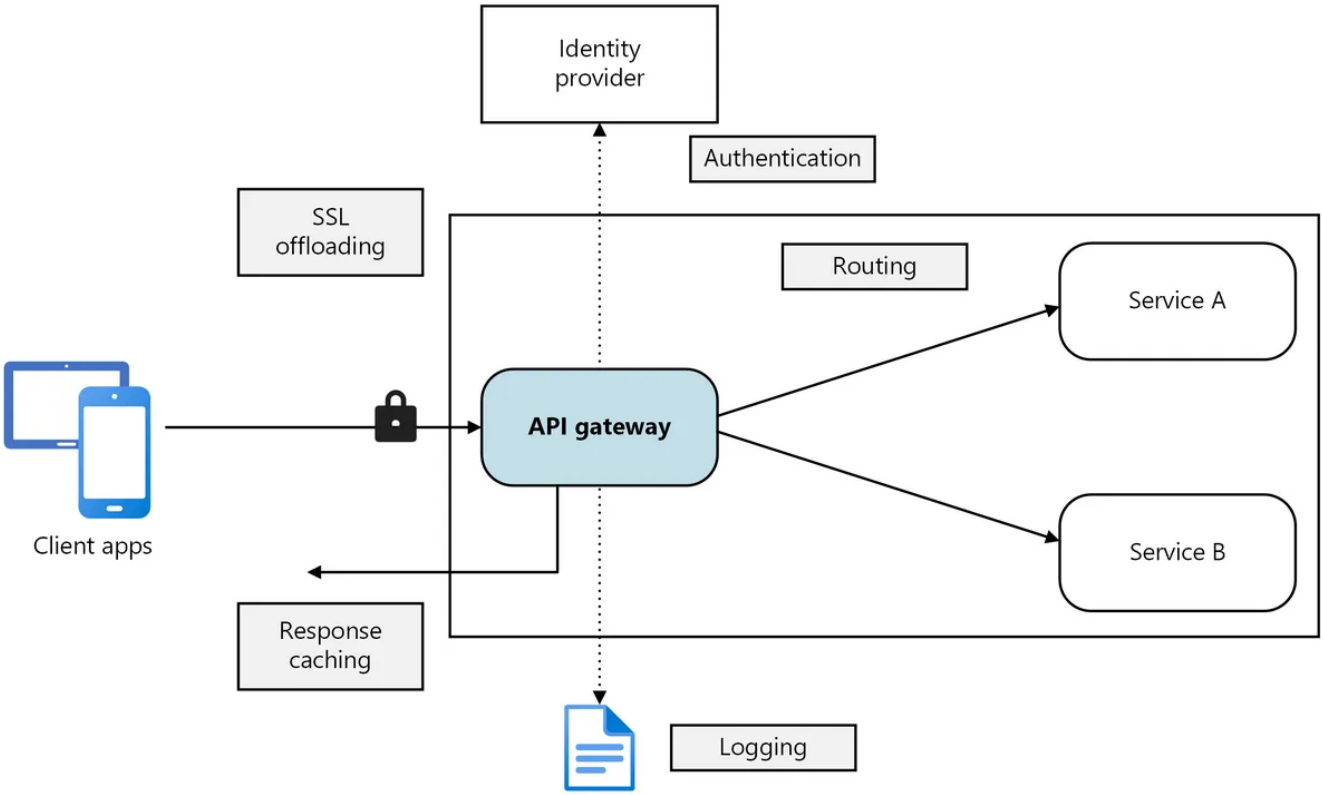
✨ Member-only story

What is API Gateway Pattern in Microservices Architecture? What Problem Does it Solve?

The API Gateway can help in managing authentication, request routing, load balancing, and caching in Microservices architecture.

Soma · Follow
Published in Javarevisited
7 min read · Mar 4

Listen Share ••• More



Hello friends, if you are learning Microservices architecture then you must have come across patterns like SAGA, CQRS, API Gateway etc. These are common Microservices patterns which are used to solve common problems which comes on Microservices architecture.

In the past, I have shared Microservices Interview Questions as well as essential Microservices design principles and patterns and in this tutorial we will deep dive into API Gateway pattern.

The **API Gateway Pattern** is a design pattern used in microservices architecture to **provide a single entry point for clients to access multiple microservices.** In this pattern, a gateway service acts as a facade for all incoming requests from clients and forwards them to the appropriate microservices.

The API Gateway is responsible for handling tasks such as authentication, request routing, load balancing, and caching. By providing a single entry point, the API Gateway simplifies the client's interaction with the microservices system, as clients don't need to know the location or details of individual microservices.

In addition, **the API Gateway Pattern can also help with implementing cross-cutting concerns such as logging, monitoring, and security** in a centralized manner, rather than having to implement these features in each individual microservice. This can help to reduce development and maintenance costs, as well as improve the overall performance and reliability of the microservices system.

What problem API Gateway Pattern Solve?

When you first start with Microservices architecture, everything is easy, but when Microservices grows from 10 to 100 then things started becoming more and more difficult.

For example, if your client needs to connect with 10 services, it needs to remember the host and port URL for them, it's not just a problem at the client side but also on server side you need to implement authentication, authorization and security on each of those Microservices, API Gateway can solve many of these problems

In short, The API Gateway Pattern solves several problems that arise in microservices architecture, including:

1. Simplifying the client's interaction with the microservices system by providing a single entry point.
2. Providing a centralized location for implementing cross-cutting concerns such as logging, monitoring, and security.

3. Reducing the complexity of the microservices system by hiding the implementation details of individual microservices.

Here is an **example to illustrate how the API Gateway Pattern can help solve these problems:**

Suppose we have a system with several microservices that perform different tasks, such as authentication, user management, and product catalog management.

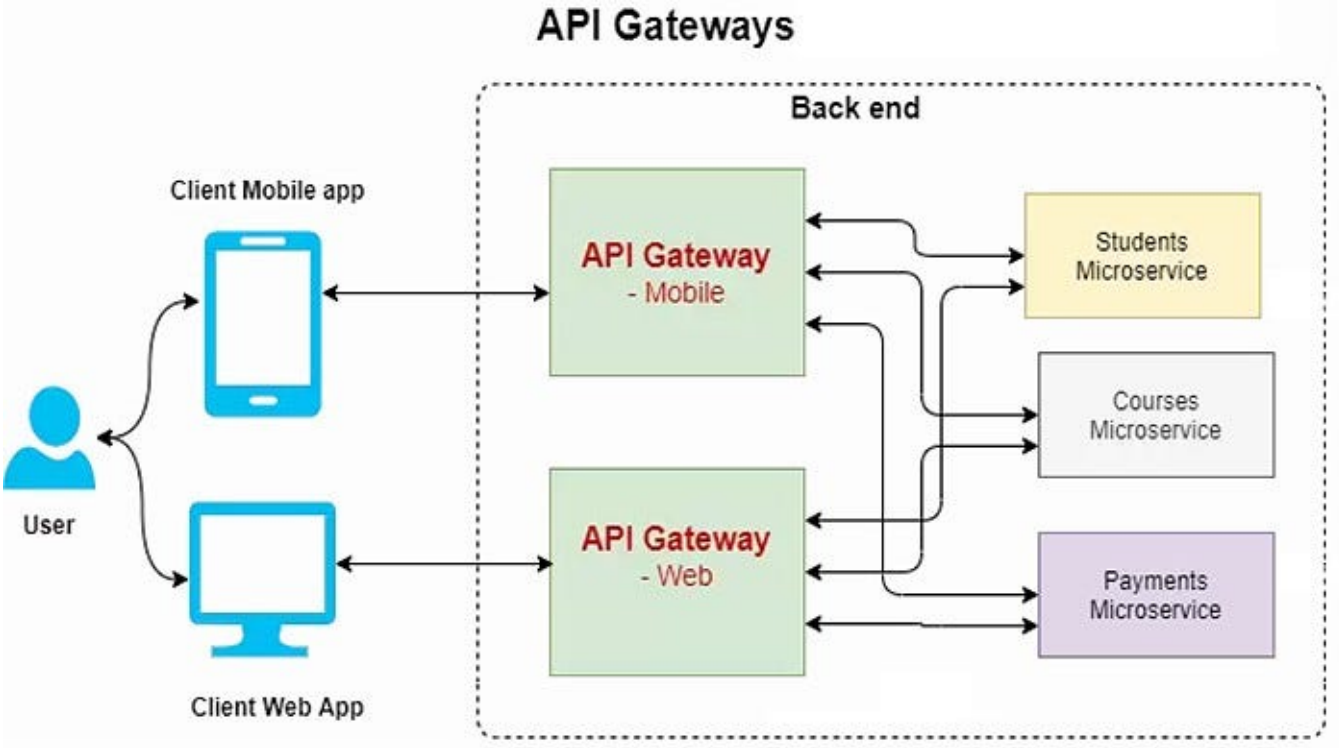
A **client application needs to access these microservices** to perform various tasks, such as logging in, viewing user profiles, and browsing products.

Without an API Gateway, the client would need to know the location and details of each individual microservice and make requests to each one separately. This can be complex and error-prone, especially as the number of microservices in the system grows.

By introducing an API Gateway, the client can make all requests to a single entry point, and the gateway is responsible for routing the requests to the appropriate microservices.

The gateway can also implement cross-cutting concerns such as authentication and caching in a centralized manner, simplifying the implementation of these features in individual microservices.

Overall, the API Gateway Pattern helps to simplify the interaction between clients and microservices, and reduce the complexity and cost of implementing cross-cutting concerns in a microservices system.



How to implement API Gateway Pattern in Java?

Yes, Spring Cloud provides support for implementing API Gateway using the Spring Cloud Gateway project. Spring Cloud Gateway is a lightweight, yet powerful API Gateway that is fully integrated with Spring Boot and Spring Cloud.

To implement API Gateway using Spring Cloud Gateway, you can follow these steps:

1. Add the Spring Cloud Gateway dependency to your project:

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-gateway</artifactId>
</dependency>
```

2. Define the routes for your microservices in a configuration file, such as `application.yml`:

```
spring:
  cloud:
```

```
    gateway:
      routes:
        - id: service1
          uri: lb://service1
          predicates:
            - Path=/service1/**
        - id: service2
          uri: lb://service2
          predicates:
            - Path=/service2/**
```

In this example, the API Gateway will route requests to `http://<gateway>/service1/**` to the `service1` microservice and requests to `http://<gateway>/service2/**` to the `service2` microservice.

1. Start the API Gateway:

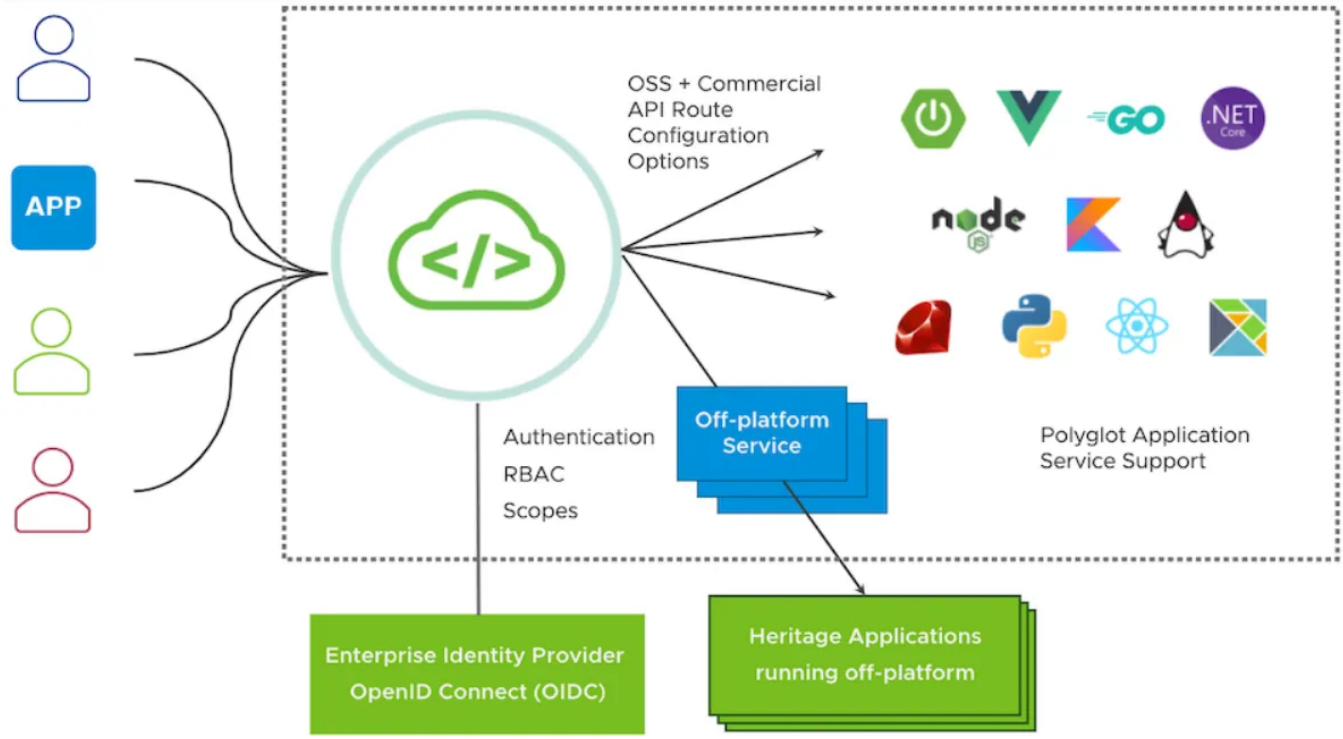
```
@SpringBootApplication
public class GatewayApplication {

    public static void main(String[] args) {
        SpringApplication.run(GatewayApplication.class, args);
    }

}
```

This will start the API Gateway and load the configuration file defined in step 2.

Spring Cloud Gateway provides many advanced features, such as circuit breaking, rate limiting, request/response logging, and more, which means you don't need to reinvent the wheel again. You can also customize the behavior of the API Gateway by writing custom filters and interceptors.



Pros and Cons of API Gateway in Microservices

There is no pattern in this world which only comes with pros and no cons and API Gateway is no exception. API Gateway can provide several benefits to micro services-based architecture, including:

Pros:

Security

An API Gateway can provide a centralized location for authentication and authorization, helping to secure your microservices.

Traffic Management

API Gateway can help manage traffic to your microservices by providing load balancing and routing capabilities.

Scalability

By routing traffic to the appropriate microservices, API Gateway can help ensure that your system is scalable.

Simplified Client Code

API Gateway can simplify the client-side code by aggregating data from multiple microservices and returning the result as a single response.

Analytics

An API Gateway can capture and aggregate analytics data from all of the microservices, providing valuable insights into usage patterns.

Cons:

Single point of failure

The biggest disadvantages of API Gateway is the single point failure. If the API Gateway fails, the entire system could fail.

Increased complexity

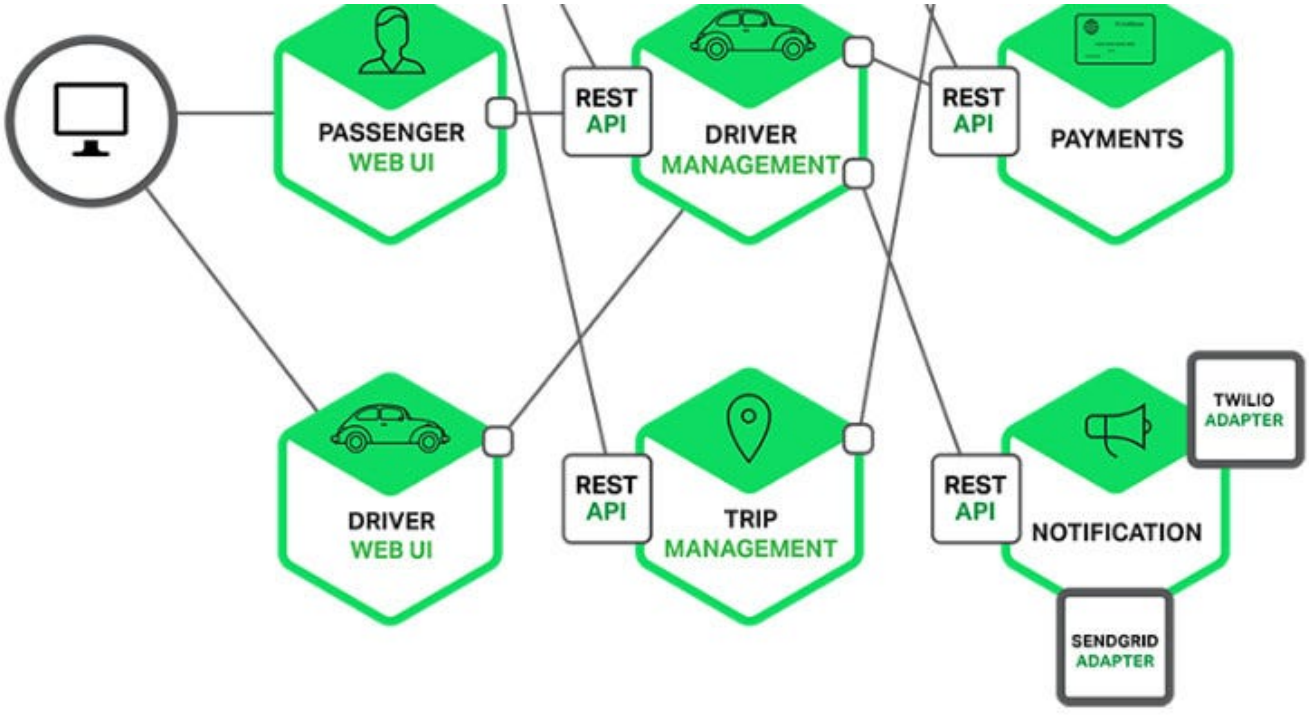
Another disadvantage is that An API Gateway can add complexity to your architecture, and if not implemented correctly, it can become a bottleneck.

Additional latency:

API Gateway can also introduce additional latency as requests are routed through the gateway.

Debugging Headache

API Gateway can also make Debugging difficult as requests are passed through the gateway.



Java and Spring Interview Preparation Material

Before any Java and Spring Developer interview, I always use to read the below resources

Grokking the Java Interview

Grokking the Java Interview: click here

I have personally bought these books to speed up my preparation.

You can get your sample copy here, check the content of it and go for it

Grokking the Java Interview [Free Sample Copy]: click here

> *If you want to prepare for the Spring Boot interview you follow this consolidated ebook, it also contains microservice questions from spring boot interviews.*

Grokking the Spring Boot Interview

You can get your copy here — **Grokking the Spring Boot Interview**

That's all about **API Gateway Pattern in Microservices Architecture.** It has a lot of benefit and I believe every developer should be familiar with these pattern if they are working in Microservices architecture.

For Java developers Spring Cloud provides several modules that make it easy to implement an API Gateway, including Spring Cloud Gateway and Netflix Zull.

You can use these modules provide features such as routing, load balancing, and security, making it easier to implement an API Gateway in your microservices based architecture.

And, if you are not a Medium member then I highly recommend you to join Medium so that you can not only read these articles but also from many others like Google Engineers and Tech experts from FAANG companies. You can **join Medium here**

Join Medium with my referral link - Soma

Read this and every story from thousands of great writers on Medium). Read more from great writers on Medium like...

medium.com

My **Other Articles** you may like to read

Difference between Horizontal Scalability vs Vertical Scaling in System Design and Microservices

Hello folks , one question which I have been asked repeatedly on many System design and Java Developer interviews are...

medium.com

20 Advanced Core Java Interview Questions for Experienced Developers of 5 to 10 Years

These are difficult and advanced core Java questions for 5 to 10 years Experienced Java developers

medium.com

15 Side Hustles Developers can do to Earn Passive/Additional Income in 2023

These are the 10 most effective ways for programmers to create additional income streams and passive income in 2023

medium.com

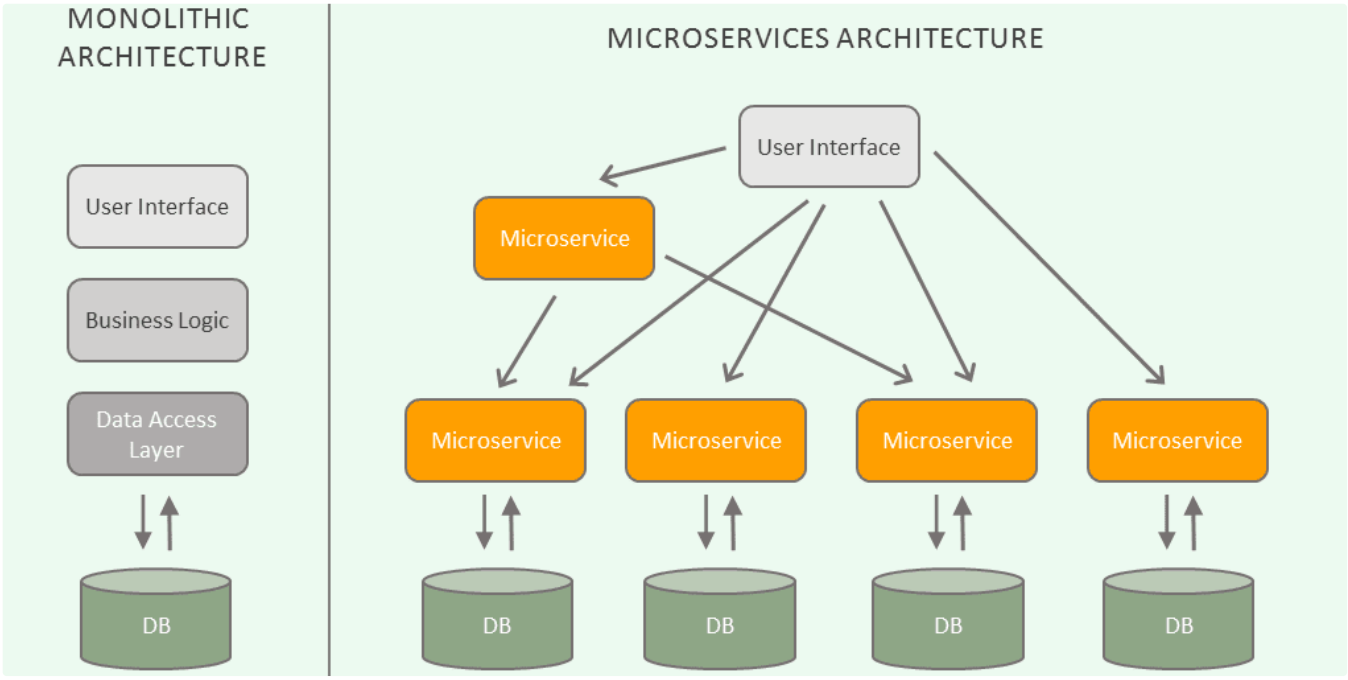
Follow

Written by Soma

4.2K Followers · Editor for Javarevisited

Java and React developer, Join Medium (my favorite Java subscription) using my link 👉 https://medium.com/@somasharma_81597/membership

More from Soma and Javarevisited



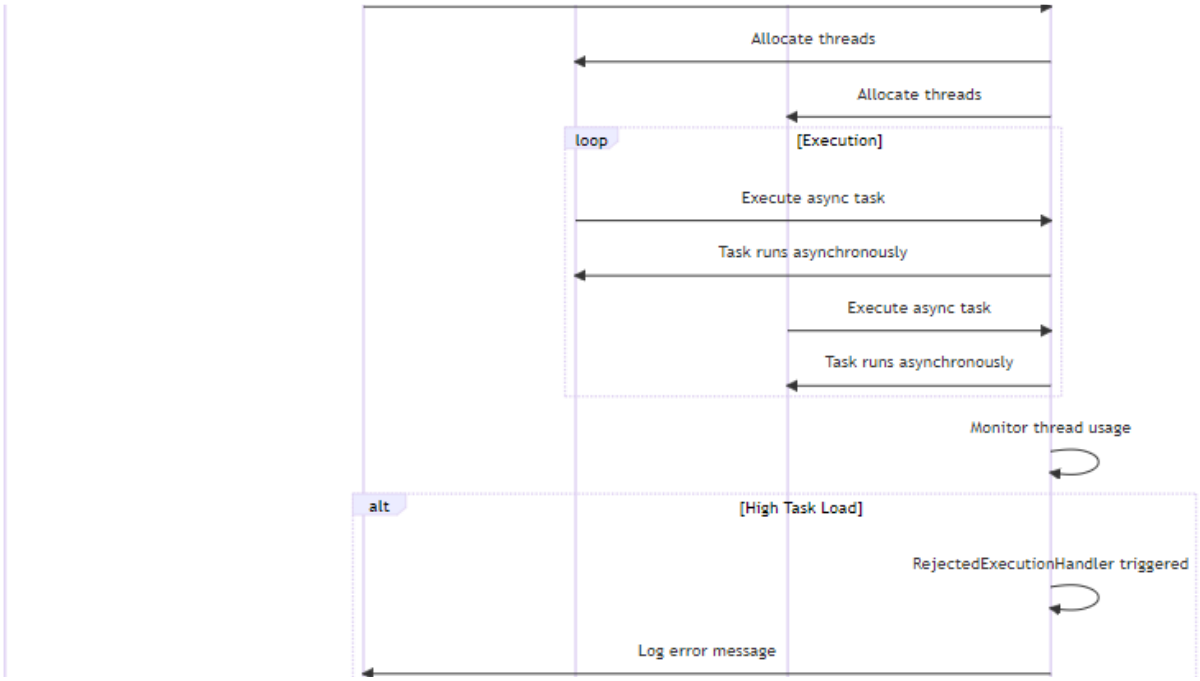
Soma in Javarevisited

18 Advanced Microservices Interview Questions for Experienced Developers: Challenges and Insights

From Microservice Architect to Scalability: 18 Interview Questions to Test the Microservices Skills of Developers with 5 to 10 Years of...

13 min read · 3 days ago

114



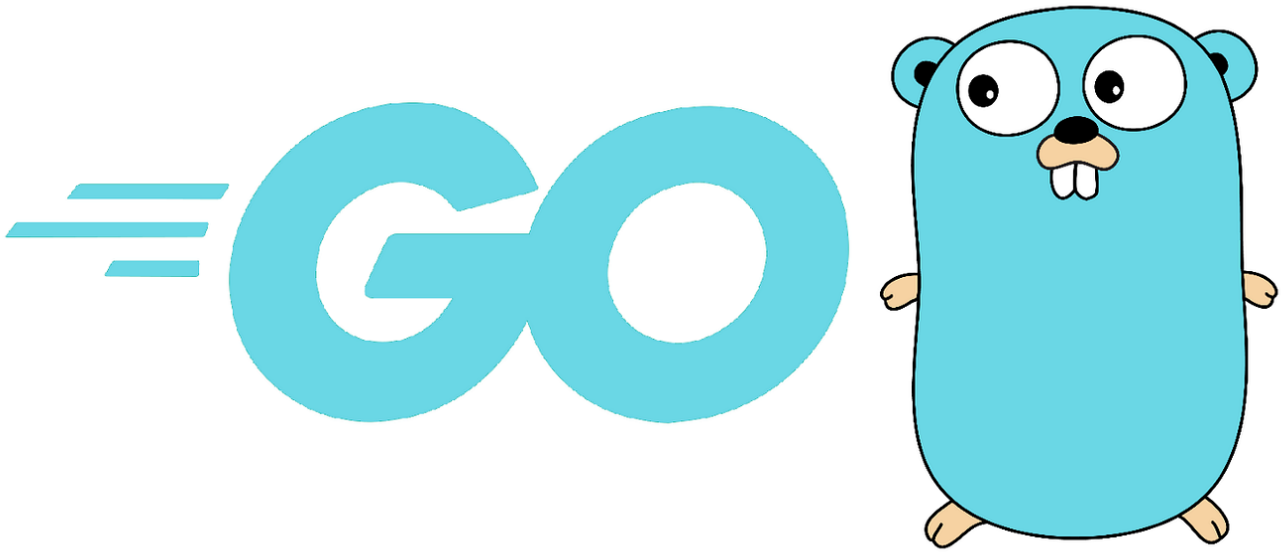
Srikanth Dannarapu in Javarevisited

Using Async Schedulers in Spring Boot

Problem Statement: Efficiently Managing Multiple Asynchronous Schedulers in a Spring Boot Application

5 min read · Aug 7

109 1



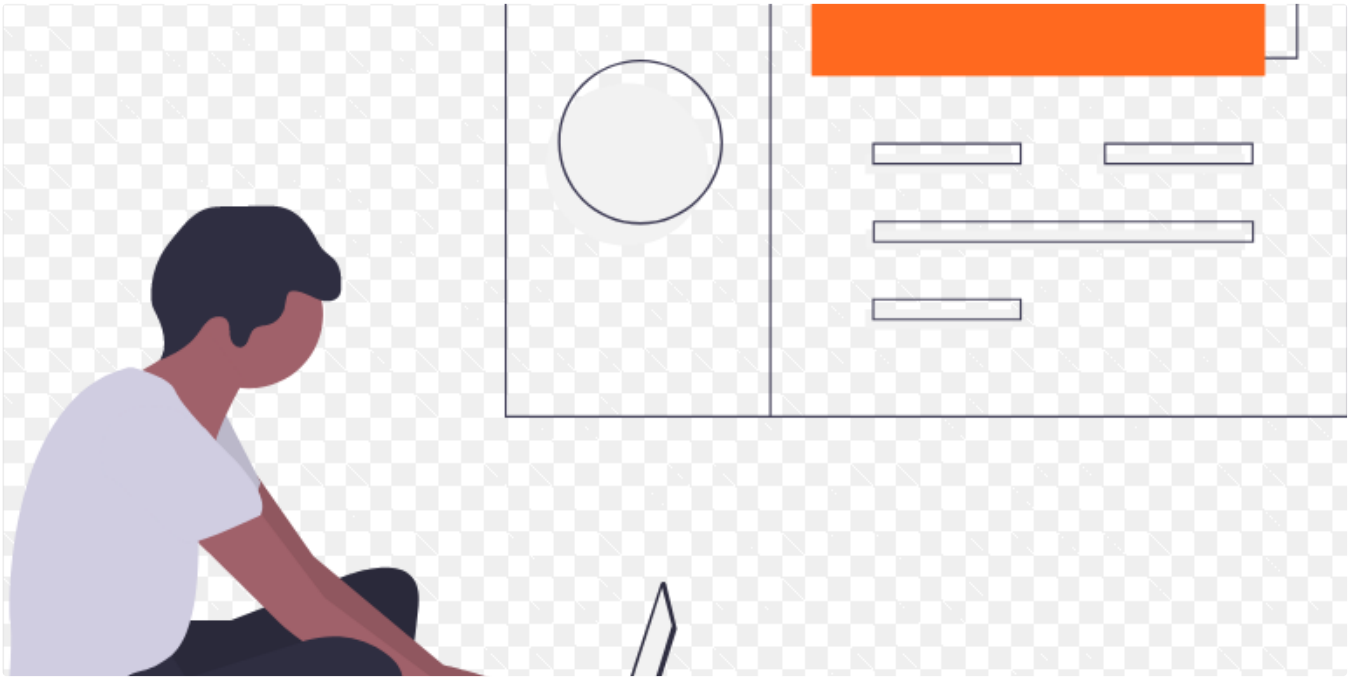
javinpaul in Javarevisited

10 Projects You Can Build to Learn Golang in 2023

My favorite Golang Projects with courses for beginners and experienced developers

11 min read · Aug 7

98 1



Soma in Javarevisited

Difference between @RequestMapping and @GetMapping Annotations in Spring MVC Web Framework

Hello friends , if you are preparing for Java Developer and Spring Developer interviews then you must prepare for questions like...

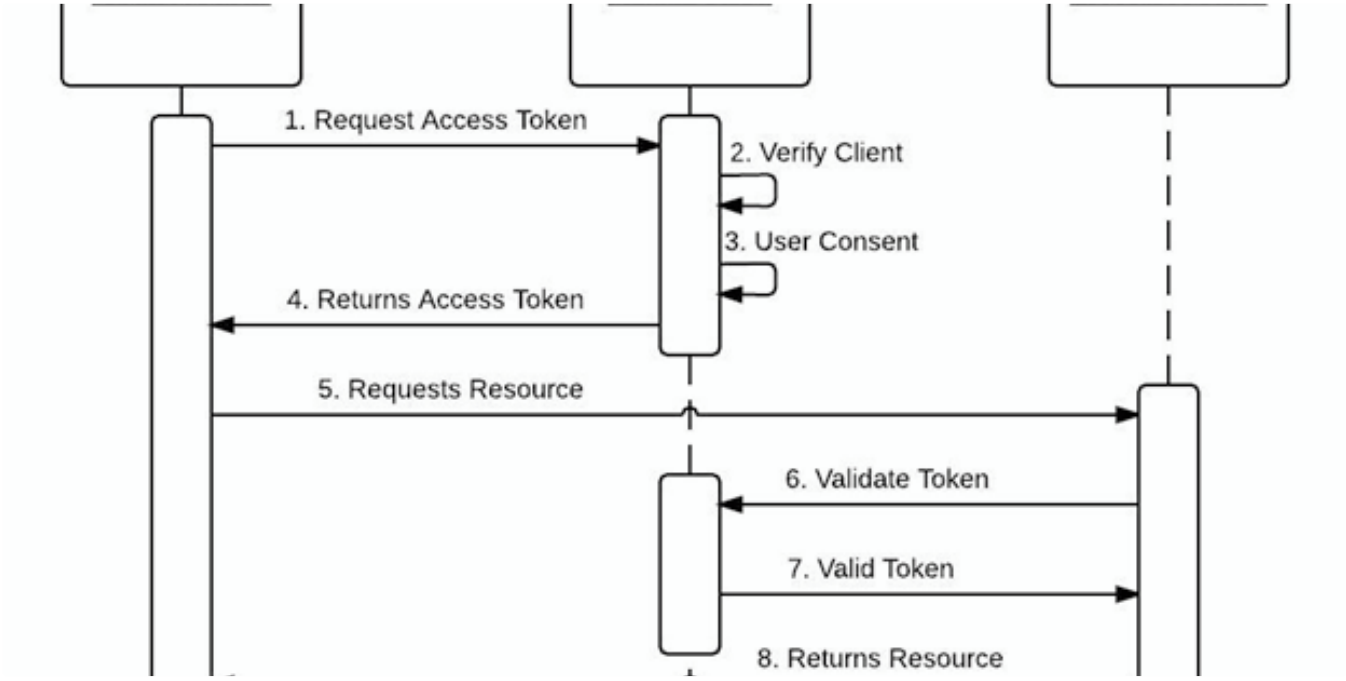
· 6 min read · Aug 19

49

See all from Soma

See all from Javarevisited

Recommended from Medium



Somal Chakraborty in Dev Genius

Securing a Microservice with OAuth 2.0 (Part 2 : How Token Based Security Works Internally)

Table of content

5 min read · May 1

37 1

Eray Araz

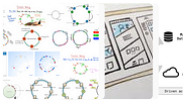
Spring Webhook

Introduction

3 min read · Apr 2

 108  1

Lists



General Coding Knowledge

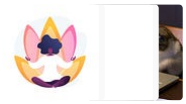
20 stories · 255 saves

It's never too late or early to start something

15 stories · 91 saves

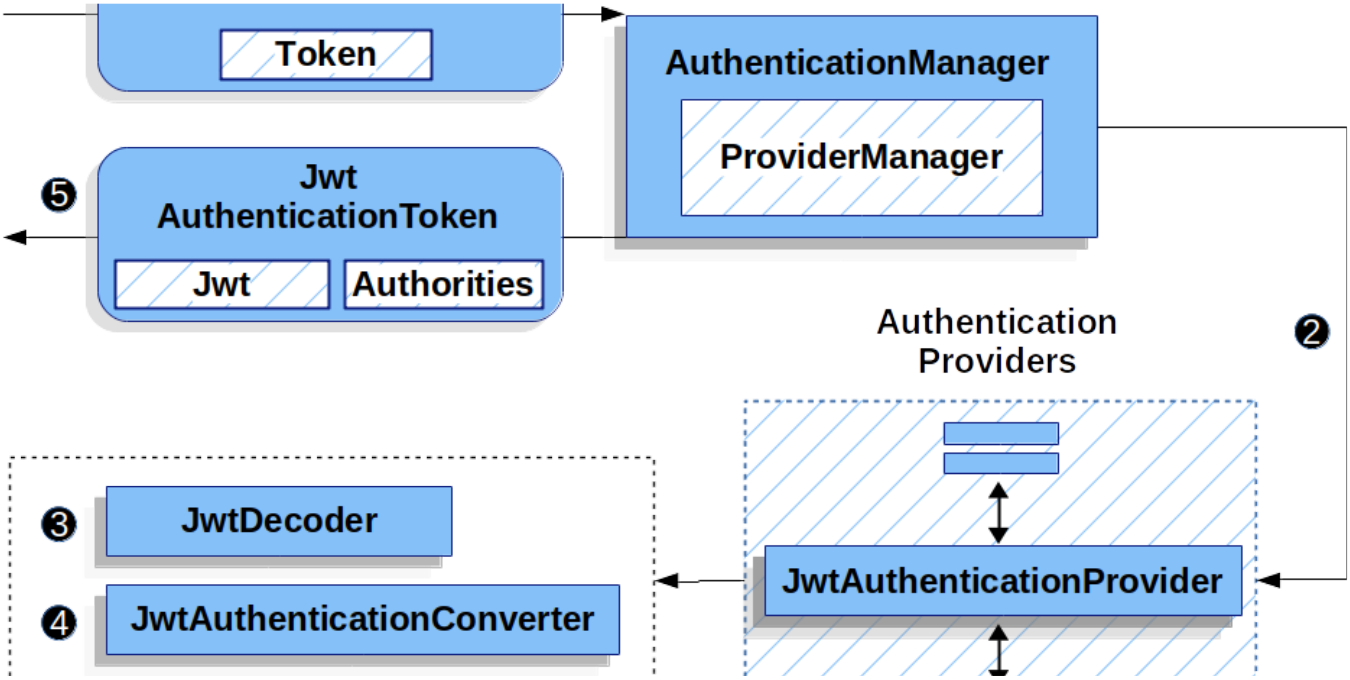
Coding & Development

11 stories · 129 saves



Stories to Help You Grow as a Software Developer

19 stories · 308 saves



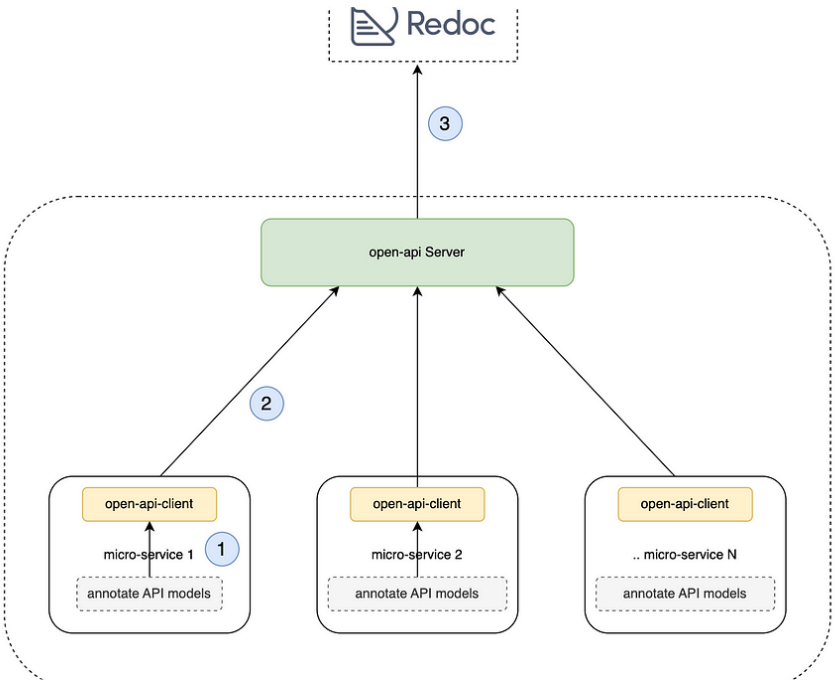
Alper Kurtoglu

Spring OAuth2 with OIDC: Moving from 'Scope' to 'Roles'

In this tutorial, I will discuss how to use authorities instead of scopes as roles in Spring OAuth2.

3 min read · Aug 17

71



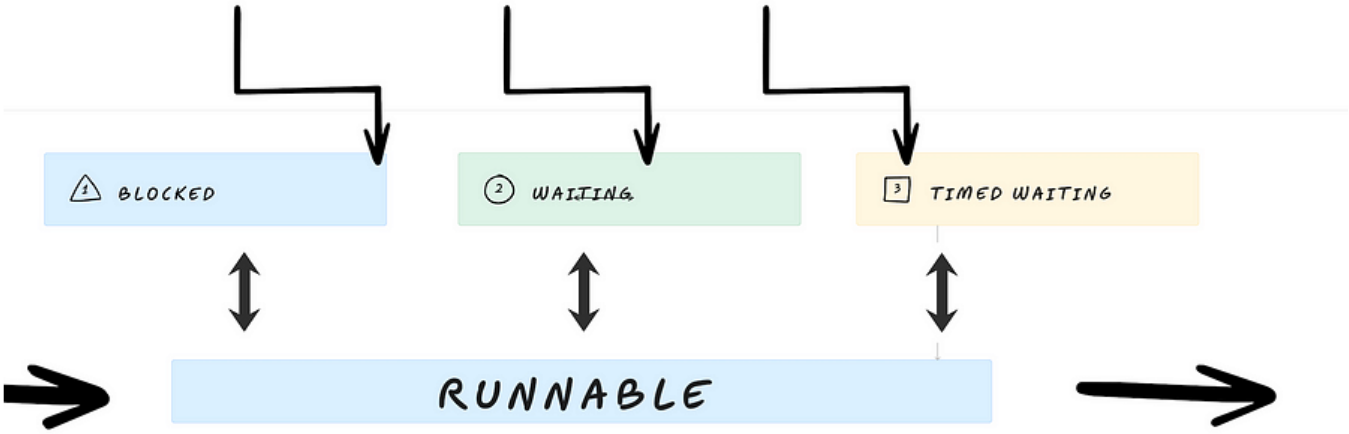
Rajesh Gupta

Centralised API Documentation

Problem Statement

4 min read · Jun 18

12



Vikas Taank

All about Java Threads

Thread: What is a Thread , thread is a light weight process that denotes and execution in operating system which has a life cycle and that...

6 min read · Jul 14

18