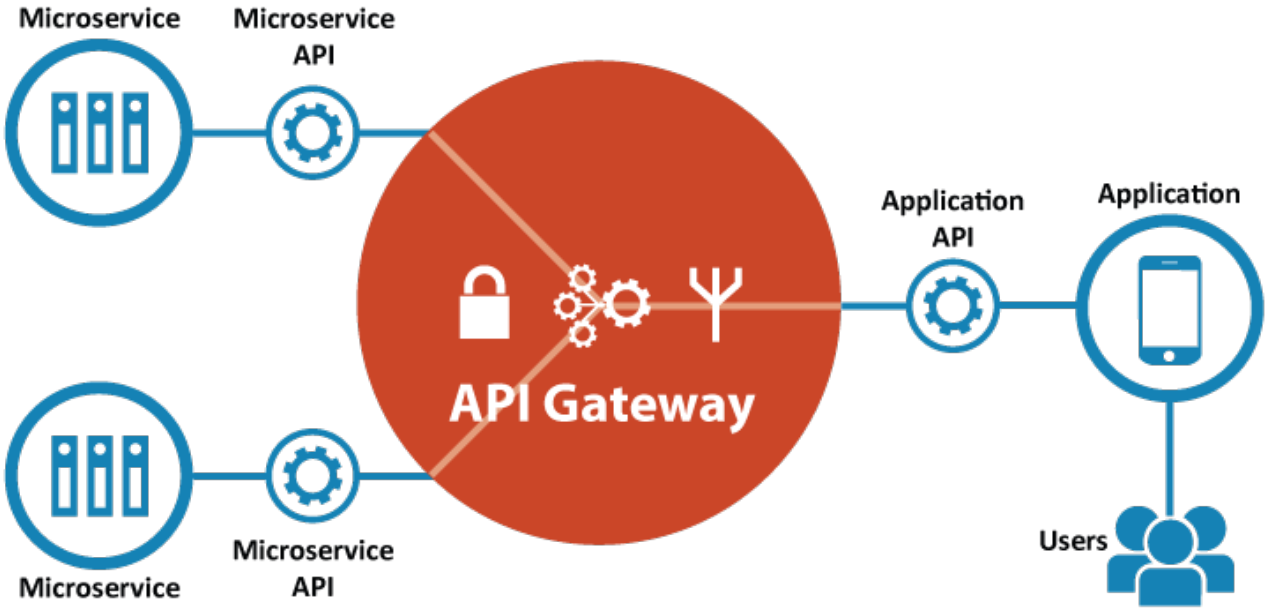
Api Gateway

Microservices Design Pattern - Api Gateway Yaklaşımı



Kullanıcı uygulamadan bir request attığında arkada neler dönüyor bunu bilmez. Api-gateway bu requeste cevap verebilmek için içerde belki de onlarca mikroservisi kullanabilir. İşte tam burada datanın nasıl fetch ve aggregate edileceği konusunda api-gateway pattern'i neye ihtiyacımız olduğunu belirleyecek unsurdur.

Api-Gateway temel davranışları;

- Router: Mikroservisler arasında haberleşmeyi sağlar. Bir servisten diğerine gelen istekleri iletir(Service Discovery).
- Data aggregator: Mikroservisler arasında bilgi toplayarak ve bunları zengin bir response halinde bağlı olduğu Api Consumer'a iletir. Bu durumda Backend For Frontend (BFF) gibi davranmış olur.
- Centralized error management: Bir servise ulaşılamadığı zaman veya servis aşırı yavaş cevap vermeye başladığı zaman api-gateway ölümcül hataların yayılmaması için cache den default response'lar sağlamaya başlar. Sistemi daha güvenilir ve esnek hale getirmek için erişelemeyen servis yeniden ayağa kalkana kadar kapatılır.

Spring boot
Örnek: https://github.com/spring-cloud-samples/spring-cloud-gateway-sample/blob/master/src/main/java/com/example/demogateway/DemogatewayApplication.java

```
@Bean
   public RouteLocator myRoutes(RouteLocatorBuilder builder) {
       return builder.routes()
                // Add a simple re-route from: /get to:
http://httpbin.org:80
                // A a simple "Hello:World" HTTP Header
                .route(p -> p
                  .path("/get") // intercept calls to the /get
path
                  .filters(f -> f.addRequestHeader("Hello",
"World")) // add header
                  .uri("http://httpbin.org:80")) // forward to
httpbin
                .build();
   }
```