Microservice Nedir?

Microservice mimarisinin avantajlarından bahsetmeden önce Service Oriented Architecture(SOA)'a ufak bir bakış yapıp, geleneksel SOA'da kullanılan monolithic mimarinin dezavantajlarına da değineceğiz.

Service Oriented Architecture (SOA)

Basit anlamda Türkçe'ye çevirirsek, Servis Odaklı Mimari kendini az çok anlatıyor zaten. Entegrasyon veya Omni-Channel ihtiyacı ile birlikte aynı veri katmanı ve/veya business'ı kullanmak isteyen birden fazla uygulama için üretilen servisler ve bu servisleri tüketen uygulamalar için kullandığımız mimariye verdiğimiz ad.

Servis odaklı mimari, veri paylaşması istenen uygulamalar, entegrasyonlar vs. gibi ihtiyaçlar ile çıktı kimimizin karşısına ama tahminimce en büyük ihtiyaç, mobil cihazların hayatımıza girmesiyle yaşandı. Mobil cihazlar, veri katmanına direkt erişimi engelleyerek aslında bizi zorunlu olarak servis odaklı mimariye geçirdi. Ardından Single Page Application(SPA) olarak adlandırdığımız Angular, Vue, React vs. gibi framework'lerin önlenemez yükselişi ile daha da önem kazandı ve günümüz dünyasında ise olmazsa olmazımız.

Monolithic Architecture Nedir?

İsmini daha önce duymadıysanız da muhtemelen kullanıyorsunuz çünkü geleneksel uygulama mimarilerimize Monolithic ismini veriyoruz. Monolithic, geliştirilen uygulamanın tüm katmanlarını ve modüllerini tek bir proje içerisinde barındıran mimari şeklidir. Monolithic mimari içerisinde, *authorization*, *presentation*, *business logic*, *data* gibi katmanlar yer alabilir.

Bir e-commerce uygulaması geliştirdiğinizi düşünün, en azından kullanıcı bilgilerini tuttuğunuz, login ettiğiniz, ürünleri listelettiğiniz ve ödeme aldığınız *business* ve *data* katmanınız vardır. Tüm bu modülleri, parçalamadan, aynı proje içerisinde sunan mimaridir.

Monolithic Architecture ile tasarlanmış olan SOA'larda zaman ilerledikçe çeşitli problemler ile karşılaşmaya başlıyoruz. Gereksinimlerin artması, entegrasyon ihtiyaçları veya farklı modüllerin implemente ihtiyacı ile servis uygulamamız büyümeye başlıyor. Bu büyüme ile birlikte, *tight coupling* yapıda olan servis projesine yeni geliştiriciler dahil etmek, projeyi de bizi de yavaş yavaş kaosa doğru itiyor.

Monolithic Architecture Dezavantajları

- Projeye dahil ettiğimiz yeni arkadaşa günlerce eğitim vermek zorundayız çünkü tüm kompleksitesiyle birlikte projeyi bu arkadaşa teslim ediyoruz.
- Tüm servisler tek bir proje içerisinde yer aldığı için minik bir bug olduğunda dahi tüm projenin tekrar derlenmesi gerekiyor. Yıllarca büyüttüğümüz

projemizde bu işlem çok uzun sürebiliyor. Uzun sürmesinin dışında, örneğin ürünlerin listelenme sırası ile ilgili yaptığımız ufak bir geliştirmenin canlıya alınması sırasında, ödeme sistemi de tekrar derleniyor. Bu aşamada, o anda ödeme yapmaya çalışan bir müşteride alakasız bir şekilde probleme sebep olabiliyor. Büyüyen projenin ayağa kalkma süresi de daha uzun olabiliyor.

- Tüm servislerin aynı programlama dili ile geliştirilmesi gerekiyor. *.NET Core* ile başlanmış bir servis içerisine, *image processing* içeren bir servis dahil etmek ne kadar mantıklı olur? Veya istatistiksel işlemler için *R* kullanmak isteyen bir ekip, monolithic yapıda *.NET Core* kullanmaya devam etmek zorunda kalacak. Bu da programlama dillerini amaçları dışında kullanmaya itecek.
- Monolithic yapıda geliştirilen servisi scale etmek istediğimizde arkaya bir load balancer koyup tüm servisin komple kopyasını almamız gerekiyor. Halbuki ihtiyacımız olan sadece *Token* servisini scale etmek olabilir.
- Herhangi bir modülde yaşanacak bir bug, tüm sistemin çökmesine sebep olabilir. Örneğin *memory cache* kullandığınız resim boyutlandırma servisiniz memory leak'e sebebiyet veriyor olabilir. Bu durumun ödeme almaya etki etmemesi gerekir. Resimleri bir süre göstermesiniz de satış yapmayı engellememeniz gerekiyor.

Tüm bu sebepler ve bu sayılanların dışında kalan endirekt sebeplerden dolayı microservices günden güne daha da popüler hale geliyor.

Microservice Architecture

Microservice'ler, yalnızca tek bir işi gerçekleştirmekten sorumlu servislerdir. Birden fazla microservice'in birleşmesi ile oluşan yapıya Microservice Mimari adını veriyoruz. Az önce anlattığım Monolithic SOA'larda yaşanan çeşitli sorunları çözmek adına geliştirilmişlerdir. Buradaki amaç, uygulamayı parçalarına ayırıp, birbirleriyle fiziksel bağı olmayan minik uygulamacıklar oluşturmaktır. Oluşturulan bu uygulamacıkların derleme, test, deployment adımları da, DevOps yöntemleri uygulanarak otomatize edilir

Unix felsefesi der ki;

Do one thing and do it well / Bir şey yap ve bunu iyi yap

Microservice'ler de aynı felsefe ile geliştirilir. En başından beri e-ticaret'ten gidiyoruz, yine devam edelim. "Ürün" dediğimiz nesneyi düşünen ekibin, diğer nesneler ile etkileşimini düşünmemesi gerekir. Basit anlamda düşünürsek;

- Yeni ürün ekleyebilmeli,
- Ürün güncelleyebilmeli,
- Ürün silebilmeli,
- Ürünleri listeleyebilmeli,
- Tek bir ürün getirebilmeli

> Ürün microservice'ini geliştiren ekibin, bu ürün sepete nasıl eklenir, satın alması nasıl yapılır gibi kavramları düşünmemesi gerekiyor. Daha da ileri gidelim. Ürün sepete eklendiğinde bunu MongoDB'de tutmaya karar vermiş bir ekip var ama Ürün servisini geliştiren ekip MSSQL kullanmak isteyebilir. Microservice Mimari sayesinde bunlar gerçekleştirilebilir.

Microservice Avantajları ve Dezavantajları

Microservice'lerin avantajları olduğu gibi dezavantajları da vardır. Öncelikle avantajlarını listeleyelim;

- Her bir microservice farklı dil kullanılarak geliştirilebilir. Bu sayede farklı farklı ekipler projeye dahil edilebilir.
- Her microservice'in veri katmanı farklı olabilir. Biri MSSQL+Redis kullanırken, diğeri MongoDB veya bir .json dosyası kullanabilir.
- Her microservice birbirinden bağımsız olarak scale edilebilir.
- Yukarıda bahsettiğim gibi, microservice'ler küçük ve basit yapılar olduğu için yönetimi ve versiyonlaması kolay olacaktır.
- Servisler farklı ortamlarda barınacakları ve birbirleri ile ilişkileri minimum düzeyde olacağı için herhangi birinde yaşanabilecek bug diğerlerini etkilemeyecek. Yani Monolithic Architecture bölümünde bahsettiğimiz *image resize* servisinde yaşanacak problem diğer kaynakları etkilemeyecek.
- Her bir microservice küçük bir modül olduğu için derleme süresi hızlanacak, testler küçük modüllerde daha etkili yapılabilecek ve CI/CD pipelineları ile *Time to Market* hızlanacak.

Avantajları olduğu gibi tabiyiki dezavantajları da var. En azından şu ana kadar bizim karşılaştığımız sıkıntılardan bir derleme yapacak olursak;

- Servislerin birbirleri ile iletişimini sağlamak her zaman kolay olmayabiliyor. Her bir iletişim kanalı için bir messaging queue açıp bunu consume etmek gerekebiliyor.

- Problem kök sebebini bulmak monolithic yapıda daha kolay. Karşılaşılan problemin hangi microservice ile alakalı olduğunu bulabilmek, bu microservice'in içeriğinde mi yoksa RabbitMQ vb. bir mesajlaşma aracında mı problem olduğunu çözebilmek kimi zaman can sıkabiliyor.
- Start-up hosting maliyetleri monolithic yapıya göre daha yüksek. Farklı microservice'lerde farklı veri katmanları kullanma özgürlüğü ve her microservice'ın farklı bir ortamda host edilebiliyor olması ister istemez lisans maliyetlerini arttırıyor.

Microservice tercih etmeli miyim?

Microservices'a geçmeden önce iyice düşünülmeli, günlerce üzerine yatılıp/kalkılmalı ve gerçekten ihtiyaç varsa aksiyon alınmalı. Aksi taktirde yaşanacak dezavantajlar katlanarak büyüyecektir.

Aşağıdaki maddelerin çoğu size uyuyorsa microservices kullanmayı düşünebilirsiniz;

- Çok fazla modülüm var, birbirleri ile *tightly-coupled* yapı kurmak işi kompleksleştirecek,
- Aktif kullanıcı sayımın fazla olmasını bekliyorum, gönül rahatlığıyla scale edebilmem gerekiyor,
- Geniş bir geliştirici kadrom var,
- Belirli bir dil ile kısıtlı kalmak istemiyorum,
- Çok hızlı deploy almam gerekiyor