# Monolitik Mimari ve Microservice Mimarisi Arasındaki Farklar

## Monolitik Mimari Yapısı

Monolitik bir uygulama, birden fazla modül içeren tek bir kod tabanına sahiptir. Modüller, fonksiyonel veya teknik özelliklerine göre ayrılmıştır. Tüm uygulamayı build eden tek bir derleme sistemine sahiptir. Ayrıca tek bir çalıştırılabilir veya deploy edilebilir dosyaya sahiptir.

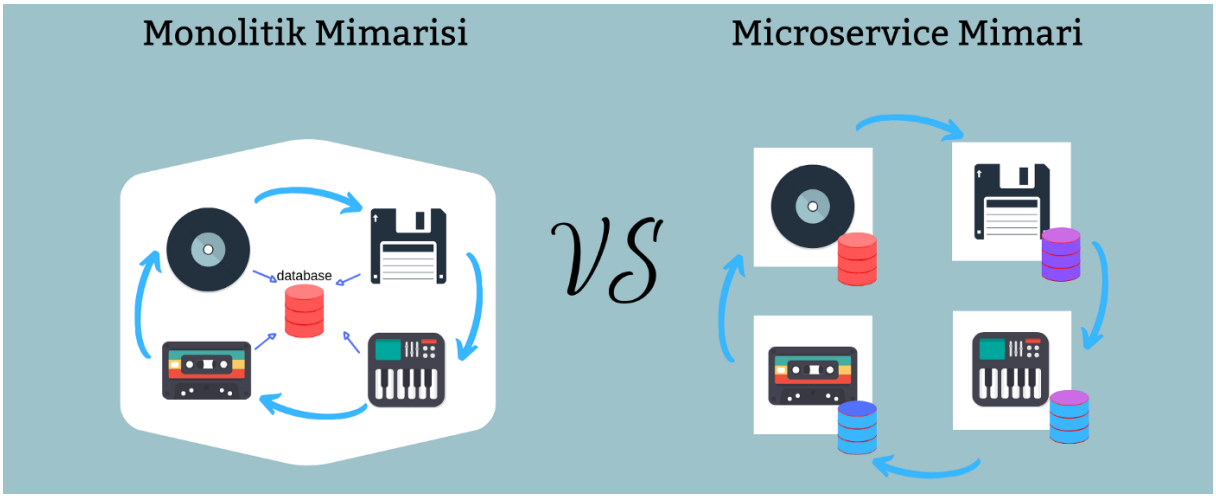
Genellikle, uygulamalar üç kısımdan oluşmakta; client kısmı yani kullanıcıya gösterilen ve kullanıcının işlemlerinin gerçekleştirildiği kısım web sayfası, mobil uygulama gibi, bir de sunucu uygulaması kısmı tüm isteklerin yönetildiği ve gerekli algoritmaların bulunduğu yapı olarak düşünülebilir ve son olarak verileri kaydettiğimiz ve depoladığımız database kısmından oluştuğu söylenebilir. Tek bir parçadan oluşan server kısmını düşündüğümüzde, sistemin tümüyle hareket ettiğini görebilirsiniz.

Başlatılacağı zaman ya da durdurulacağı zaman ve hatta çöktüğü zaman uygulamanın tamamı bu durumdan etkilenir. Yani uygulama, yaşamı boyunca tek bir parça halinde hareket eder. Herhangi bir güncelleme, değişiklik ya da versiyon yükseltmesi işlemi için uygulamanın tümüne müdahale etmemiz gerekir.

## Microservice Mimari Yapısı

Microservice, tek işe odaklı uygulamanın işlevsel küçük bir parçasını ifade eder. Microservice olarak tasarlanan bir servis, ihtiyaca bağlı olarak birden fazla projede(uygulamada) veya farklı projelerde tekrar kullanılabilir. Servisler arasındaki bağımlılıklar, loose-coupled ilkesine uygun olarak en aza indirgenir. Tek sorumluluk odaklı servisler bir standarda uygun olunca, diğer servisler üzerindeki değişikliklerden etkilenmezler.

Microservice yaklaşımı ölçeklenebilirlik, esneklik, çeviklik gibi önemli avantajlar barındırır. Bu yüzdendir ki birçok önemli ve büyük şirket Netflix, Google, Amazon, Spotify gibi teknoloji liderleri, monolitik mimarilerinden microservice mimarisine geçiş yapmayı başarıyla tamamlamışlardır.




**Monolotik Mimarinin Güçlü Yanları**

- Daha kolay hata ayıklama ve uçtan uca test etme: Microservice mimarisinin aksine, monolitik uygulamaların hata ayıklaması ve test edilmesi çok daha kolaydır. Monolitik bir uygulama tek bir bölünmez proje olduğundan, uçtan uca testleri çok daha hızlı şekilde gerçekleştirebilirsiniz.
- Kolay deployment: Monolitik uygulamaların tek bir parça oluşu ile bağlantılı bir avantajı ise kolay dağıtımdır. Tek bir parçayı deploy etmek onlarca servisi deploy etmekten çok daha kolaydır.
- Monolitik bir uygulamayı genel prensiplere uyarak modüler bir biçimde yazmak, her bir modülü servis olarak yazmaktan çok daha kolaydır.

**Monolitik Mimarinin Zayıf Yanları**

- Karmaşıklık: Bir monolitik uygulama büyüdüğünde, anlaşılamayacak kadar karmaşık hale gelir. Ayrıca, bir uygulama içindeki karmaşık kod yapısını yönetmek zordur, ne kadar prensiplere uyuyor olsanız da kodun büyümesi karmaşıklığı her zaman artıracaktır.
- Değişiklik yapma zorluğu: Bu kadar büyük ve karmaşık bir uygulamada değişiklik yapmak oldukça zahmetlidir. Herhangi bir kod değişikliği tüm sistemi etkiler, bu nedenle tamamen bütün parçaların kontrol edilmesi gerekir. Bu, genel geliştirme sürecini çok daha uzun hale getirebilir.
- Ölçeklenebilirliği(Scaleable) düşük: Bileşenleri bağımsız olarak scale edemezsiniz. Tüm uygulamayı ölçeklemeniz gerekir.
- Yeni teknoloji uygulayamamak: Monolitik bir uygulamada yeni bir teknolojinin uygulanması son derece problemlidir, uygulamayı yeni bir teknolojiye entegre etmek demek, tüm uygulama yeniden geliştirmek anlamına gelir. Yeniliklere adapte olamamak da birçok alandan kayba yol açabilir.

**Microservice Mimarisinin Güçlü Yanları**

- Bağımsız bileşenler:
  —Her bir servis bağımsız bir şekilde dağıtılabilir ve güncellenebilir, bu da daha fazla esneklik sağlar.
  — Bir microservice deki bir hatanın yalnızca belirli bir hizmet üzerinde etkisi vardır ve tüm uygulamayı etkilemez. Ayrıca, bir microservice uygulamasına yeni özellikler eklemek, monolitik olandan daha kolaydır.
- Anlaşılabilirlik. Daha küçük ve daha basit bileşenlere ayrılan bir microservice uygulamasının anlaşılması ve yönetilmesi daha kolaydır. Sadece sahip olduğu bir sorumluluk ile ilgili olan belirli bir hizmete odaklıdır.
- Daha iyi ölçeklenebilirlik: Her bir servis bağımsız olarak ölçeklendirilebilir. Bu nedenle tüm uygulamanın scale edilmesi gerekmez. Bu da zaman ve maliyet açısından büyük bir kazanç sağlar. Ek olarak, her monolitik uygulamanın ölçeklenebilirlik açısından sınırları vardır. Ancak microservice üzerinde trafik oluşan bir servisi çoklamak daha az zahmetli ve bütün yükü kaldırabilecek düzeydedir. Sırf bu yüzden büyük bir kullanıcı kitlesine hitap eden çoğu proje microservice mimarisini benimsemeye başlamıştır.
- Teknoloji Çeşitliliği: Ekipler servislerin geliştirileceği teknolojileri tamamen seçmek zorunda değildir, yani zamanla geliştirecekleri servislere uygun teknoloji seçebilirler. Örneğin Java üzerinde geliştirilen microservicelerin yanına “machine learning” özelliklerini kullanabilmek için python programlama dili ile bir servis geliştirilebilir. Her microservice için istenilen teknoloji ya da database kullanılabilir.
- Daha yüksek çeviklik seviyesi. Bir microservice uygulamasındaki herhangi bir hata, tüm uygulamayı değil yalnızca belirli bir servisi etkiler. Bu nedenle tüm değişiklikler ve testler daha düşük riskler ve daha az hata ile gerçekleştirilir.
- Yeniden kullanılabilirlik: Bağımsız olarak geliştirdiğiniz bir servisi ihtiyacınıza göre ufak değişiklikler ile başka bir projede de kullanılabilir. Örneğin “User Management” servisi neredeyse tüm uygulamalarda benzer sorumlulukları ve işlevi barındırır ve benzer görevleri gerçekleştirir. Bu servisi başka bir projenizde direkt olarak konumlandırabilir ve kullanılabilir. Böylece bir servis geliştirme zaman ve maliyetinden tasarruf edililebilir.

**Microservice Mimarisinin Zayıf Yanları**

- Küçük ölçekli ve küçük kitleye hitap eden uygulamalar için asla microservice mimarisi tercih edilmemelidir.
- Ekstra karmaşıklık. Bir microservice mimarisi dağıtık bir sistem olduğundan, tüm modüller ve veritabanları için ayrı ayrı yapılandırma yapmanız gerekir. Ayrıca, böyle bir uygulama bağımsız hizmetler içerdiği sürece hepsinin bağımsız olarak dağıtılması gerekir.
- Sistem dağıtımı Bir microservice mimarisi, birden fazla servis ve veri tabanından oluşan karmaşık bir sistemdir, bu nedenle tüm bağlantıların dikkatle ele alınması gerekir. Deployment her servis için ayrı bir süreci gerektirir.
- Yönetim ve izlenebilirliğin zorluğu: Bir microservice uygulaması oluştururken birden fazla durumla ile ilgilenmeniz gerekecektir. Harici yapılandırmayı, metrikleri takip edebilme ve microservicelerin yönetilebildiği ortamlar gerekir.

- Test etmenin zorluğu: Birbirinden bağımsız olarak konuşlandırılan çok sayıda servis test sürecini oldukça zorlaştırır.