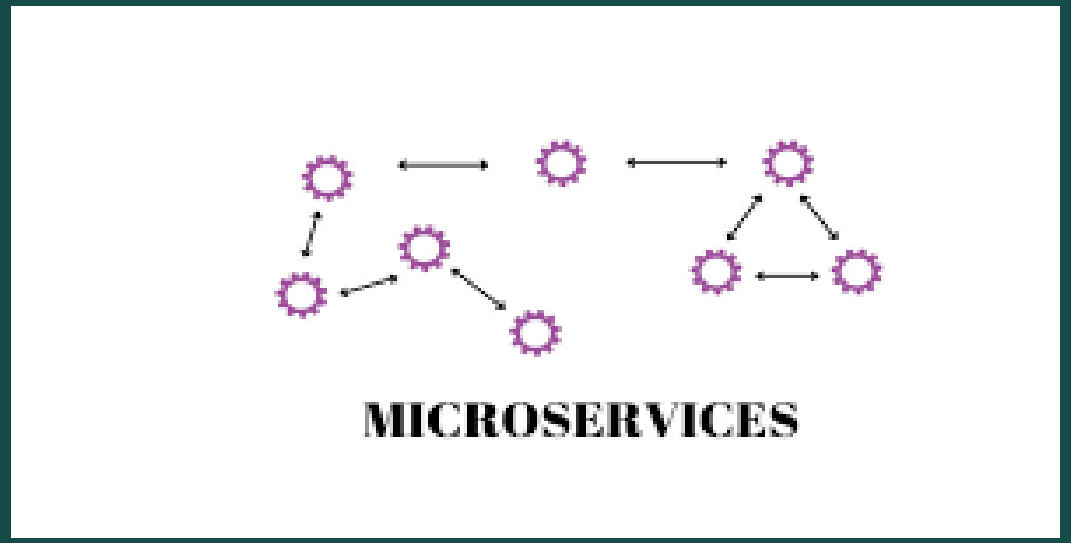
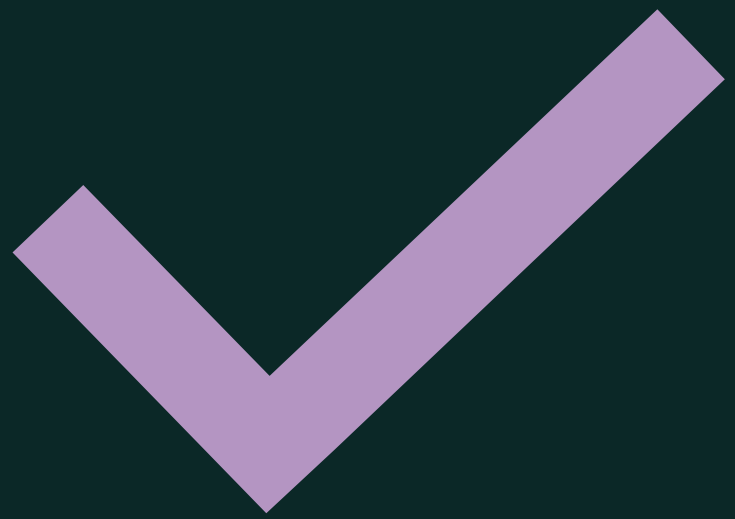


MICROSERVICES

Ahmet Cemalettin Kumru

İnteraktif ve Web Projeleri – Yazılım Geliştirme Stajyeri

İçindekiler



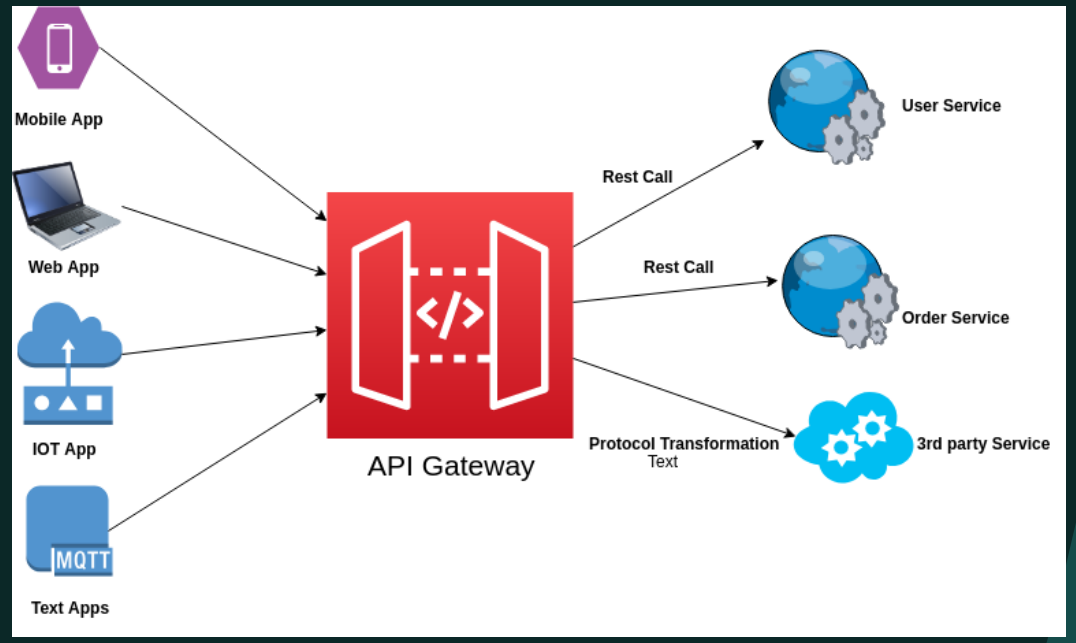
Mikroservis Nedir?

- Temelde bir yazılım uygulamasında belirli özellik yada fonksiyonu sağlayan, tek bir amaca hizmet eden, birbirinden bağımsız yazılım servisleridir. Bu hizmetler bağımsız olarak bakımı yapılabilir, izlenilebilir ve dağıtılabilir bir yapıya sahip olmalıdır.
- Mikroservis tek başına sorumluluğu olan ve tek iş yapan sadece o işe ait işleri yürüten modüler projelerdir.
- Mikroservis Service Oriented Architecture üzerine kurulmuş bir mimaridir. Service Oriented Architecture , uygulamaların birbirleriyle tek bir makine ve ya ağ üzerinden birden çok makineye dağıtıldığında , servislerin dağıtım sisteminde iletişim kurabilmesini sağlayan bir mimaridir.

- Mikroservis Mimarisi, her bir servisin kendi işini ve iletişimini yürütebilen, çok karmaşık olmayan ve başka servislere bağımlılığı az olan mekanizmalara sahip bir yaklaşımdır.
- Bu servisler kendilerinin sorumlu olduğu tek bir işe odaklı ve bağımsız çalışabilen, otomatize bir deployment mekanizmasına sahip bir yapıdadır.
- Farklı programlama dillerinde geliştirilebilir ve farklı veri tabanı teknolojileri kullanılabilir. Microservice mimarisine uygun geliştirilen herhangi bir servis üzerinde, geliştirici bağımsız olarak çalışabilir, böylece geliştirme aşaması da kolaylaştırılmış olur.

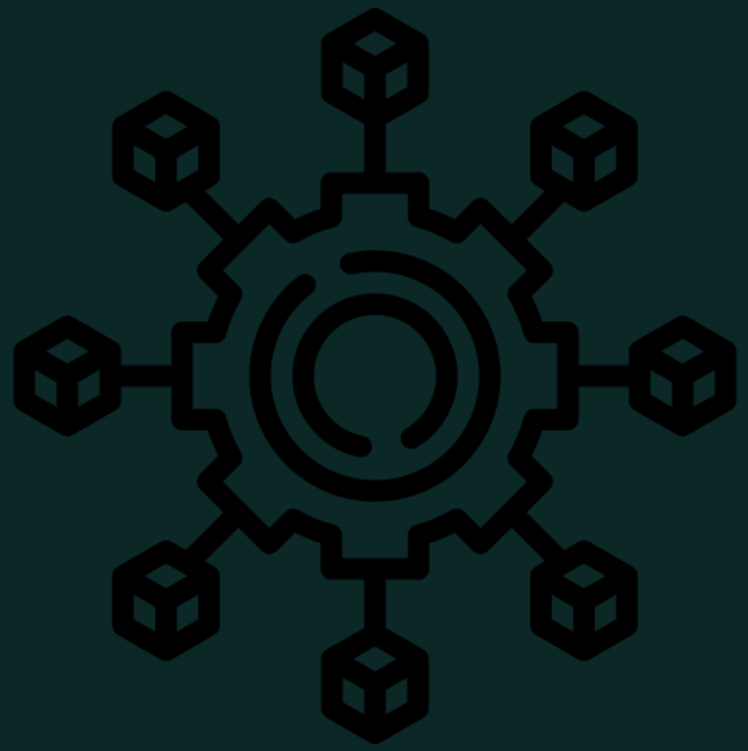
Mikroservis Mimarisi Nasıl Olmalıdır ve Avantajları Nelerdir?

- Microserviceler farklı makinelerde çalışabilir ve tüm servisler birbirleriyle kendi başlarına iletişim kurabilir olmalıdır. Yeni bir servis diğer servislerde bir değişikliğe neden olmadan geliştirilebilir ve deploy edilebilir olmalıdır.
- Microserviceler, otomatize bir biçimde gerekli aşamalardan geçerek (Unit Tests, Integration Tests, Sonarqube,Automation Tests) deploy edilmelidir. Bu mekanizma, projede kaliteyi artırır.
- Projenin herhangi bir servisinde oluşacak bir sorun, projenin diğer servislerini etkilemeyeceği için, sistem hala çalışmaya devam eder.
- Herhangi bir serviste oluşacak trafik tüm sistemin scale olmasını gerektirmeden, trafik altındaki servis çoklanarak sorun ortadan kaldırılabilir.
- Her bir bileşen kendi ihtiyacına göre ölçeklenebilir, tüm bileşenleri ölçeklendirmeye gerek yoktur.

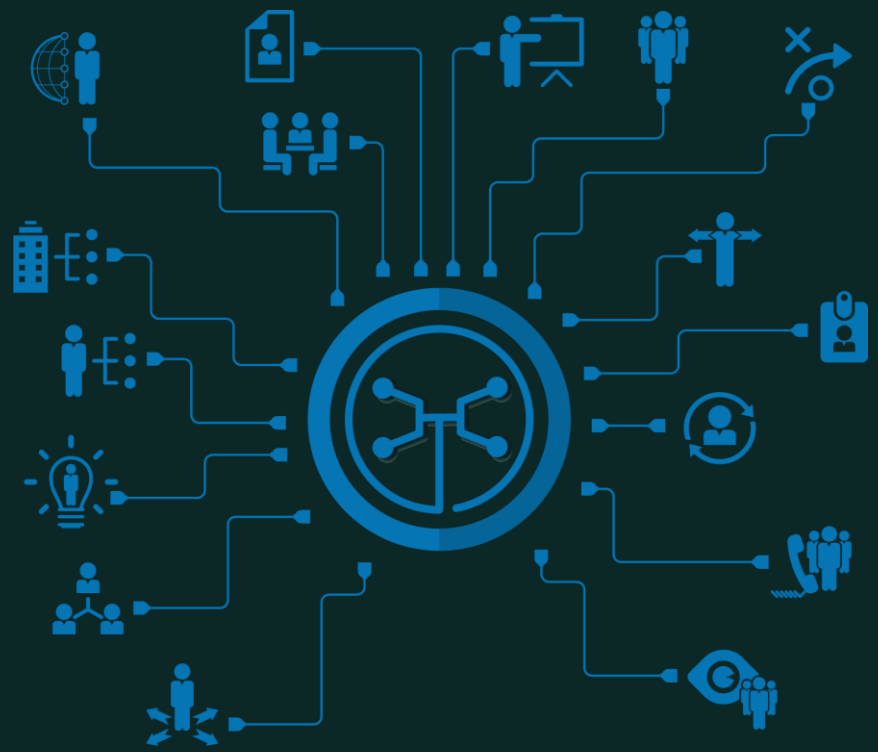


- Gereksinimleri karşılamak için en son yada en uygun teknoloji kullanılabilir. Her bir servis farklı dil veya farklı database kullanabilir. Her farklı özellik, diğer özelliklerden bağımsız olarak farklı bir ekip tarafından farklı bir teknoloji kullanılarak gerçekleştirilebilir.
- Tüm servisler, alanlarına, görevlerine ve özelliklerine göre farklı microservice ayrılmalıdır. Bu microserviceler, fonksiyonlarını yerine getirmek için kendi yük dengeleyici(load balancing) ve uygulama ortamlarına sahiptir ve aynı zamanda kendi veritabanlarında veri saklamalıdır.
- Tüm microserviceler birbirleriyle REST veya Message Bus üzerinden haberleşmelidir. İkisi birden de kullanılabilir.
- Microserviceler tarafından gerçekleştirilen tüm işlevler API Gateeway üzerinden istemcilere iletilebilir olmalıdır. Tüm iç end-pointler API Gateway'e bağlanır. Böylece, API Gateway'e bağlanan herhangi biri otomatik olarak tüm sisteme bağlanabilmelidir.

Mikroservis Temel Özellikleri Nelerdir?



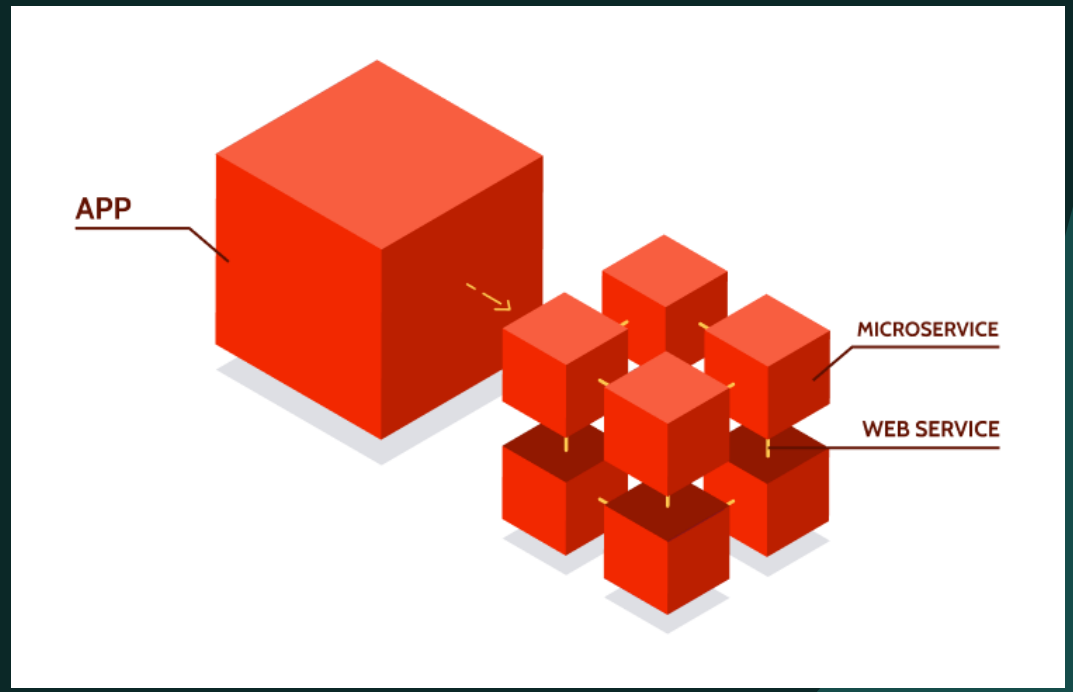
- **Decoupling:** Servisler büyük ölçüde birbirinden bağımsızdır.
- **Componentization:** Microserviceler, kolayca değiştirilebilen ve versiyonları artırılabilen bağımsız bileşenlerdir.
- **Business Capabilities:** Bir microservice basit bir yapıdadır ve tek bir göreve odaklanır.
- **Autonomy:** Geliştiriciler ve ekipler birbirlerinden bağımsız olarak çalışabilir, böylece hızlıca geliştirme ve test süreçleri yürütülebilir.
- **Continuous Delivery:** Yazılım geliştirme, test etme ve onaylama sistematik otomasyonu ile sık sık yazılım sürümlerini canlıya almaya otomatik bir biçimde izin verir. Servisler ayrı ayrı deploy edilebileceği için, deployment süresi kısalır ve maliyet zamanla azalır.



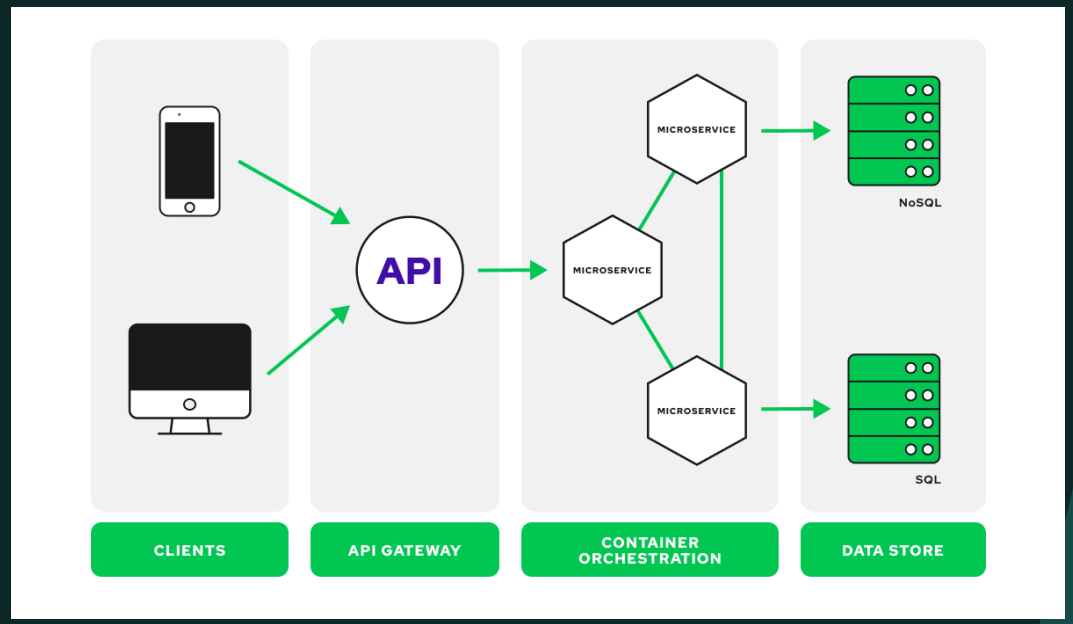
Decentralized Governance: Odak, doğru iş için doğru aracı kullanmaktır. Bu, standart bir teknoloji veya yapıyı zorunlu kılmadığı anlamına gelir. Geliştiriciler, sorunlarını çözmek için en iyi teknolojiyi seçme özgürlüğüne sahiptir.

Agility: Microserviceler çevikliği destekler. Herhangi bir yeni özellik hızla geliştirilip sisteme adapte edilebilir.

Mikroservis Eksiklikleri ve Dikkat Edilmesi Gereken Noktalar Nelerdir?

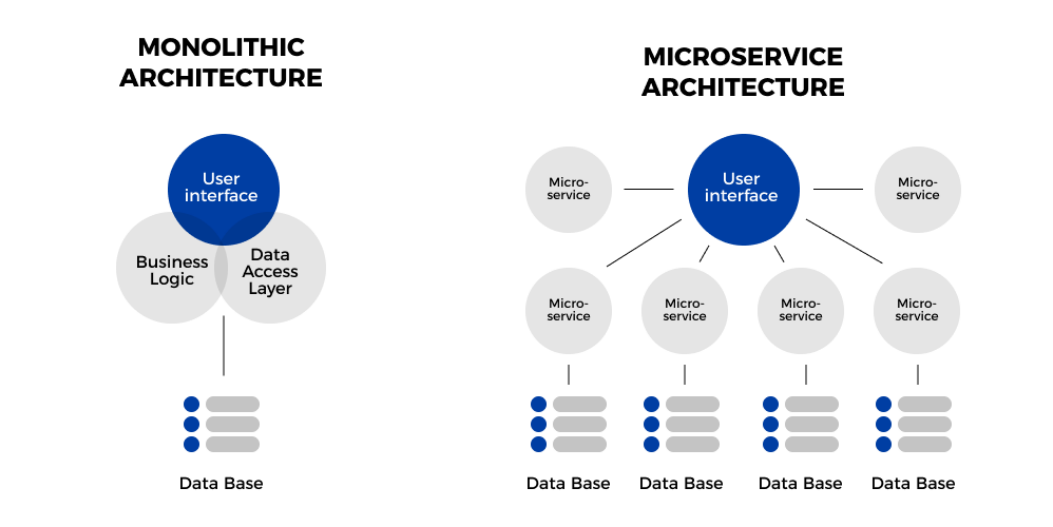


- Her bir microservice için ayrı bir derleme, dağıtım ve sürüm iş akışı gerekir. Bu nedenle, kurulum iş akışının otomatik hale getirilmesini sağlamak önemlidir, aksi takdirde operasyon ekipleri için artan iş yükü demektir.
- Her bir servisi ayrı ayrı izlenebilir ve bakımı yapılabilir olmalıdır. Prometheus bunun için iyi bir araçtır.
- Microserviceler, yapılandırma sistemindeki iş yükünü arttırır. Harici servislerin yapılandırılması genellikle servisler arasında paylaşılır ve servis sayısı arttıkça zaman alıcı bir iş haline gelebilir.
- Servisler arasındaki REST trafiği performans açısından olumsuz bir yön alabilir. Bu sorunu çözmek amacıyla önbelleğe alma ve eşzamanlılık genellikle performansı artırılabilir.
- Microserviceler üzerinde güvenlik, parallellik karmaşıklığı artıran etmenlerdir. Bu karmaşıklığı iyi yönetmek gerekir.

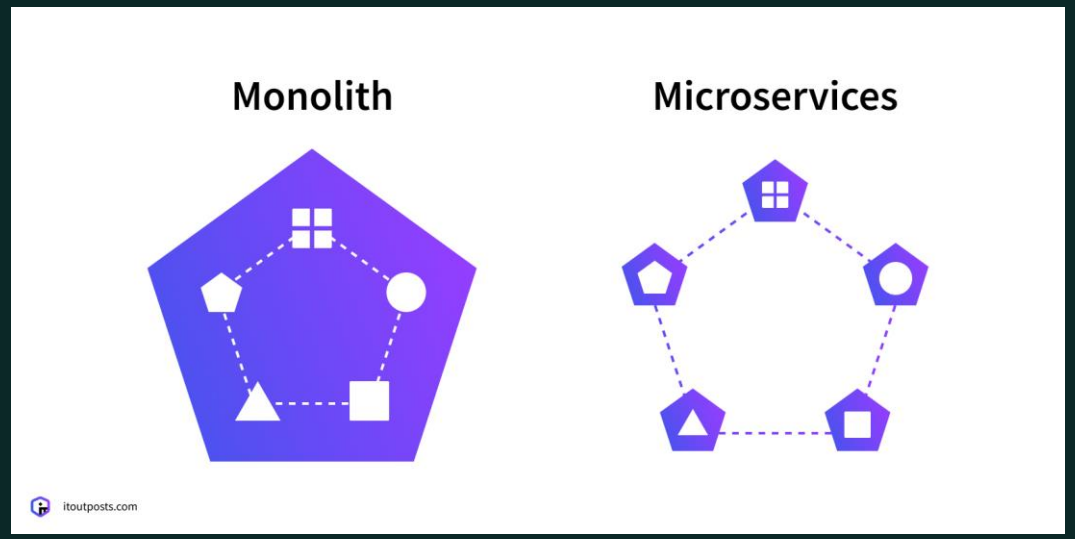


- Yeni özelliklerin mevcut hizmetlerin işlevselliği bozulmadığından emin olmak için uygulama düzeyinde testler önem vermek gerekir.
- Her hizmetin kendi sürümleri, yayın planı ve yayın döngüleri olduğu için dokümantasyon çalışması daha fazladır.
- Uygulamaların ve kod tabanlarının sayısı arttığında, onu düzgün bir şekilde korumak ve yönetmek için çaba harcanır.
- Bağımlılığı azaltmak için kod tekrarı yaşanabilir, ancak performans göz önüne alındığında bu kod tekrarları göz ardı edilebilir.

Mikroservis Mimarisi ve Monolitik Mimari



- **1.Teknoloji Çeşitliliği:** Mikroservislerin her birinin içinde farklı teknolojiler kullanabiliriz. Bu her iş için doğru ve etkin bir aracı seçmemize olanak sağlar.
- **2.Esneklik**: Monolitik bir uygulamada, servis başarısız olursa, her şey çalışmayı durdurur. Böyle bir sistemle, hata şansımızı azaltmak için birden fazla makinede çalışabiliriz, ancak mikro servislerle, hizmetlerin toplam başarısızlığını ele alan ve buna bağlı olarak işlevselliği bozan sistemler oluşturabiliriz.
- **3.Ölçekleme**: Büyük, Monolitik bir uygulamada, her şeyi birlikte ölçeklemek zorundayız. Bazen ölçekleme amacımız projenin küçük bir kısmı içindir ve buna rağmen genel sistemimizi ölçeklendirmeliyiz. Daha küçük hizmetlerle, ölçeklendirmeye ihtiyaç duyan hizmetleri ölçeklendirebiliriz. Farklı sistemler ve platformlarda çalışabilen farklı servisler olarak geliştirildikleri için, ihtiyaça göre genişletilebilirler. Mesela yoğun "memory" kullanan bir servis ile yoğun I/O işlemi yapan bir servisi farklı şekillerde "scale" edip, kaynak kullanımını optimize etmek oldukça kolay olacaktır.



4.Deployment Kolaylığı: Monolitik bir uygulamada belki bir satırlık değişikliğin deployu ve release işlemi için tüm uygulamanın deploymenta ihtiyacı vardır. Mikroservis mimarisi ile , yaptığımız her değişiklik kendi işi için özelleştirilmiş servislerin içerisinde olacağından , bu servisin deployu çok daha kolay ve sistemin geri kalanından bağımsız olacaktır. Bu da bizim için deployment sürecini hızlandıracaktır. Eğer bir problem meydana gelirse problemin meydana geldiği bağımsız serviste problem kolayca çözülebilecek ve hızlı bir şekilde rollback yapılma imkanı doğar.

5.Organizasyonların Düzenlenmesi: Mikroservisler, mimarimizi organizasyonumuza daha iyi uydurmamızı sağlar ve herhangi bir kod tabanında çalışan kişi sayısını minimize etmemize yardımcı olur. Ayrıca, ekiplerin hizmetler arasında sahipliğini tek bir hizmet alanında çalışan kişileri tutmaya çalışacak şekilde değiştirebiliriz. Ayrıca ekibe sonradan katılan bir kişinin bu kadar büyük bir yapıya, mimariyi öğrenmesi yerine, bu kişinin hangi dilde ve hangi DB ortamında işi yapmasını biliyor ise bu ortamda bu ufak iş mantığını geliştirme imkanı vermeyi sağlar.

	Monolithic	Microservices
Deployment	Simple and fast deployment of the entire system	Requires distinct resources, making orchestrating the deployment complicated
Scalability	It is hard to maintain and handle new changes; the whole system needs to be redeployed	Each element can be scaled independently without downtime
Agility	Not flexible and impossible to adopt new tech, languages, or frameworks	Integrate with new technologies to solve business purposes
Resiliency	One bug or issue can affect the whole system	A failure in one microservice does not affect other services
Testing	End-to-end testing	Independent components need to be tested individually
Security	Communication within a single unit makes data processing secure	Interprocess communication requires API getaways raising security issues
Development	Impossible to distribute the team's efforts due to the huge indivisible database	A team of developers can work independently on each component

- **6.Reuseability'i Optimize Etme:**Her projede kimsenin dokunmak istemediği , proje için hayati öneme sahip bölümler mevcuttur ve bu bölümler eski teknolojilerle hayata geçirilmiş ve yaşam ömrünü yıllar önce kaybetmiş makinelerde çalışıyor olabilir. Peki bunlar neden değiştirimez. Cevabı basit, bu değiştirme ve geçiş işlemi çok büyük ve riskli bir iş.
- Mikroservis mimarisi ile bağımsız servisler küçük boyutlarda olduğundan, bunların yerine daha iyi bir uygulama ile yer değiştirme, dönüştürme ya da bunları tamamen silme maliyetini yönetmek daha kolaydır. Mikroservis yaklaşımlarını kullanan ekipler, gerektiğinde hizmetlerin tamamen yeniden yazılabilmesinde ve artık ihtiyaç duyulmadığında bir servisi yok etmekte eskiye kıyasla sıkıntı yaşamazlar.

Kaynakça

- https://gokhana.medium.com/microservice-mimarisi-nedir-microservice-mimarisine-giri%C5%9F-948e30cf65b1
- https://gokhana.medium.com/monolitik-mimari-ve-microservice-mimarisi-aras%C4%B1ndaki-farklar-bd89ac5b094a