

# MikroServis Mimarisine Giriş



# Monolitik Mimari Nedir?

- Bütün iş mantığının ve bileşenlerin **tek bir uygulama yapısında** bir araya geldiği mimarisel çözümlerdir
- Örneğin, klasik bir kurumsal java web uygulaması **tek bir war dosyasından** oluşur
- Sistemin çapı belirli **bir büyüklüğe ulaşana kadar** geliştirmesi, deploy etmesi ve ölçeklenmesi oldukça **kolaydır**

# Monolitik Mimarinin Sorunları

- Uygulamanın codebase'i büyüdükçe developer'lar tarafından **anlaşılabilirliği** ve **yönetilebilirliği** zorlaşır
- Büyük bir codebase'e sahip uygulamayı IDE içerisinde **açmak** ve kod **geliştirmek** de daha zahmetli bir hal alır
- Uygulamanın **deployment süresi** oldukça uzar, özellikle **üretim ortamına** yapılacak kurulumlar sancılı olur

# Monolitik Mimarinin Sorunları

- Ölçeklenebilirliği ve modüllerin farklı kaynaklara olan **bağımlılıklarının** yönetimi gittikçe zorlaşır ve birbirleri ile çelişir
- Herhangi bir modüldeki **hata veya başarısızlık sistemin geneline** etki eder
- Geliştirme ekibinin organize olması, **iş bölümü yapması** ve birlikte çalışması zorlaşır
- Uygulamanın kullandığı teknolojileri değiştirmek, **upgrade** etmek daha zordur

## Sonuç olarak...

Günümüz iş ve IT dünyasındaki ihtiyaçlara daha kapsamlı ve uzun vadeli çözümler üretebilen, değişimlere ve gelişmelere daha hızlı adapte olabilen farklı bir mimarisel yaklaşıma (**microservice mimarisi**) ihtiyaç vardır

# Mikroservis Nedir?

- Web ortamında **belirli bir işi yapmaya odaklanmış** küçük uygulamalardır
- Bu uygulamalar **birbirleri ile işbirliği yaparak** çalışırlar
- **Otonom sistemlerdir**, birbirlerinden bağımsız biçimde çalıştırılırlar
- Dış dünya ile konuşmak ve hizmet sunmak için **belirli bir takım API'ler** kullanırlar ve sunarlar

# Mikroservis Mimarisinin Temel Faydaları

- Kompleks ve büyük bir sistem parçalarının ayrı ayrı ve birbirinden **bağımsız biçimde ele alınmasını ve değiştirilmesini** sağlar
- Büyük bir sistemin **daha alt ve basit bileşenlerin bir araya getirilmesi** ile oluşturulmasına (composability) imkan tanır
- Büyük ve karmaşık sistemlerin **daha kolay ve kademeli biçimde deploy** edilmesine imkan tanır

# Mikroservis Mimarisinin Temel Faydaları

- Kurumun organizasyonel yapısına daha uygun bir **mimari model** oluşturmayı sağlar
- Kurumların projelerinde **teknolojik çeşitliliğe** imkan tanır
- Hataları daha kolay, kapsamlı ve hızlı biçimde **düzeltmeye, toparlamaya** (resilient) imkan tanır
- Uygulamaların daha rahat ve verimli bir biçimde **ölçeklenebilir** (scaleable) olmasını sağlar



# Mikroservis Mimarisinin Zorlukları ve Problemleri

- Temel problem noktalarından birisi **servislerin büyüklüğü ve çapının** sağlıklı biçimde belirlenmesidir
- Mikroservisler dağıtık mimaride işlev sunmaktadırlar ve **dağıtık mimarinin** kendine has kalıtsal bir takım **problemleri** söz konusudur
- RPC iletişimi, network veya sistem **hataları, gecikmeler** vb bu mimaride etraflıca ele alınmalıdır

# Mikroservis Mimarisinin Zorlukları ve Problemleri

- **Veri bütünlüğünün ve tutarlılığının** uygulama genelinde sağlanması ayrı bir yaklaşım gerektirir
- Mikroservislerin **test edilmesi** de zorluk noktalarından birisidir
- Birbirine bağımlı birden fazla mikroservisin üzerinde **değişiklik yapılması** da belirli bir sıralama ve koordinasyon gerektirir
- Mikroservislerin **deploy edilmesi** de kompleks ve zor bir süreçtir

# Mikroservis Mimarisinin Temel Özellikleri

- **Loose Coupling**
  - Herhangi bir servise diğer servislerden **bağımsız** müdahale edilebilir
  - Bu servis diğer servisleri **etkilemeden** deploy edilebilir
  - Bu işlemler diğer servislerde herhangi bir **müdahaleye** gerek bırakmamalıdır
  - Sistemin geri kalanında **değişikliğe** gidilmemelidir

# Mikroservis Mimarisinin Temel Özellikleri

- **High Cohesion**

- Her bir servisin hizmet vereceği domain'in sınırları net biçimde çizilmiş ve belirlenmiş (**bounded context**) olmalıdır
- Servislerin birbirleri ile işbirliği yaparken kullanacakları ortak model dışında geri kalan modelleri kendi içlerinde tamamen gizlenmiş (**shared & hidden models**) olmalıdır

# Mikroservis Mimarisinin Temel Bileşenleri

- Mikroservislerin her birisi **farklı teknoloji ve programlama dilleri** ile geliştirilebilir
- Ancak mikroservis uygulamalarında bütün bu **teknoloji ve programlama dillerinden bağımsız** karşımıza çıkan bir takım **temel mimarisel bileşenler** mevcuttur
- Kurumsal olarak mikroservislerle çalışırken, hedef ve amaçlar doğrultusunda bu **örüntülere göre bileşenleri uygun biçimde bir araya getirmek** çok önemlidir

# Mikroservis Mimarisinin Temel Bileşenleri

- Service Discovery
- Configuration Service
- Edge Service & API/SSO Gateway
- Circuit Breaker
- Load Balancer
- Monitoring & Management
- Logging & Tracing

# Nereden Nasıl Başlanmalı?

- Öncelikle domain'e **tam ve net** bir biçimde hakim olunmalıdır
- Sistemi **güdük ve sınırları belirsiz** biçimde bileşenlerine ayıran bir domain modellemesinden kaçınılmalıdır
- İşe **mevcut bir codebase** üzerinden başlamak ve bunu ayrı bileşenlere ayırmak çok daha kolay olacaktır
- Sistemdeki **mevcut modüller**, potansiyel mikroservis adaylarıdır

# Neye Odaklanmalı?

- Domain model'de tespit edilen bounded context'lerin sunduğu **kabiliyetlere ve davranışlara odaklanılmalıdır**
- **Öncelik kabiliyetlere** olmalıdır
- Bu kabiliyetlerin sergilenmesi sırasında **veri paylaşımı** söz konusu olacaktır (shared model)
- Ancak paylaşılan **veri ikinci planda kalmalıdır**, aksi anemik CRUD domain modellerine yol açabilir



# Neye Odaklanmalı?

- Domain model ve bounded context'ler üzerinde çalışırken **sorulacak ilk sorular**:
  - Bu context ne yapar veya ne yapmalı?
  - Hangi kabiliyetleri sunar veya sunmalı?
- Davranış ve kabiliyetlerin tespiti ardından daha **sonra sorulacak sorular**:
  - Bu davranış ve kabiliyetleri sergilemek için hangi veriye ihtiyaç var?
  - Bu kabiliyetlere ait veri nasıl ifade edilebilir?

# Monolitik Mimariden Mikroservislere Geçiş

- Monolitik uygulamanın mikroservislere dönüştürülmesi bir tür **modernizasyon projesidir**
- Monolitik bir uygulamanın tek bir seferde mikroservis mimarisine uygun biçimde yeniden yazılması “**big bang**” olarak adlandırılır ve **oldukça risklidir**
- Bunun yerine “**strangler örüntüsü**” tercih edilmelidir

# Strangler Örüntüsü

- Yağmur ormanlarındaki **bir tür sarmaşıktan** esinlenilmiştir
- Sarmaşık güneş ışığına ulaşmak için **büyük gövdeli bir ağaca dolanarak** gelişmektedir
- Ağaç zaman içerisinde ölebilmekte ve sarmaşık **onun bedenine uyumlu** bir hal almaktadır



# İletişim

- **Harezmi** Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- <http://www.java-egitimleri.com>
- [info@java-egitimleri.com](mailto:info@java-egitimleri.com)

