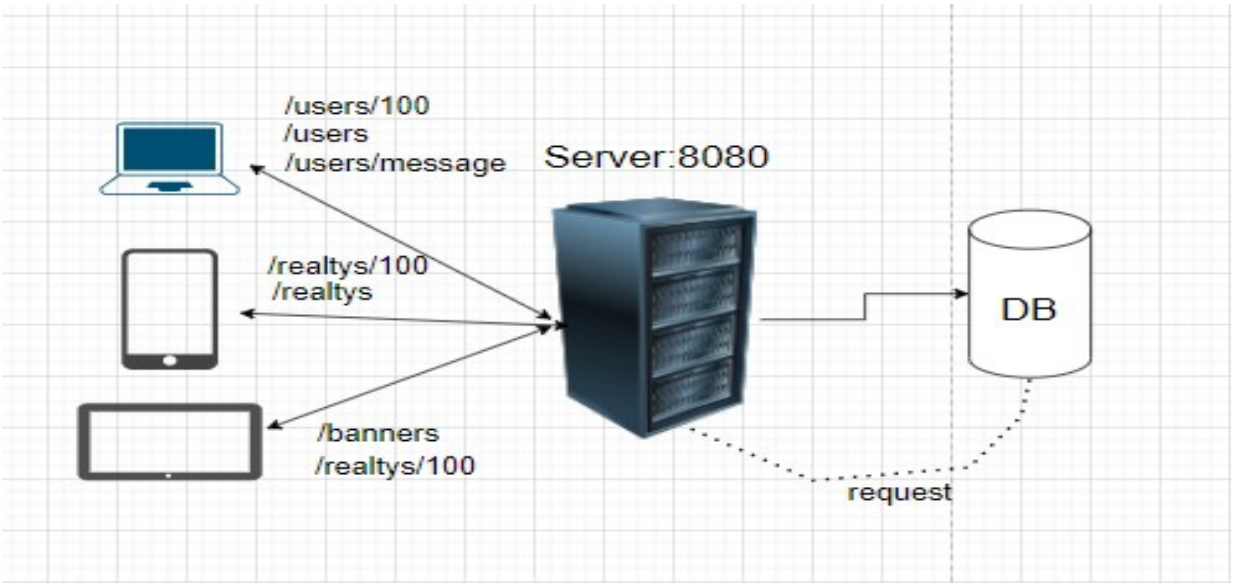
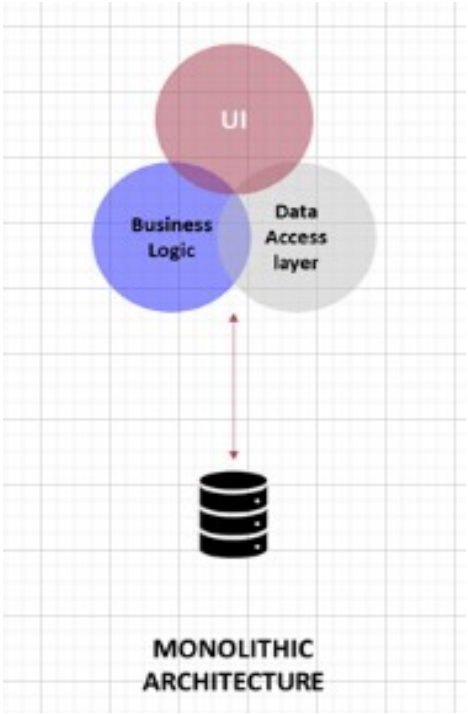
K147. Kodluyoruz Java Bootcamp

Hafta 3 - Ödev

4)
Günümüzde bir çok uygulama çok faza veriyi göndermek ve almak zorundadır. Yapmış oldugumuz emlacepte uygulamasında kullanıcı oluşturma, ilan yayınlama , mesaj gönderme , ilan arama için istekte bulunuruz veya post işlemi gerçekleştiririz. Uygulama sunucuya gelen duruma baglı olarak veri tabanından verileri ister veya gelen verileri veritabanına kaydeder.



Bu uygulamada kullandıgımız yapıya **Monolith mimari** denir.
Uygulamanın tüm parçalarının tek bir çatı altında geliştirilmesi ve sunulması olarak tanımlayabiliriz. Üç katmandan oluşur :



Presentation Layer : Son kullanıcının gördügü katman (UI)

Business Layer : Mantıksal işlerin yapıldıgı katman.

Data Access Layer: Veritabanıyla iletişim halindeki katman.

Sunucunun kullanıcı isteklerine cevap verebilmesi için tek sunucu yeterli gelmeyebilir bu durumda birden çok sunucu kullanılabilir.Bu durumda load balancer kullanılır hangi sunucu müsait ise kullanıcı ona yönlendirilir.

Avantajları

Geliştirmesi, test edilebilirliği kolaydır,

Yapının anlaşılması nispeten kolaydır,

Küçük ekipler için yönetilmesi ve geliştirmesi kolaydır ,

Deployment kolaydır

Dezavantajları

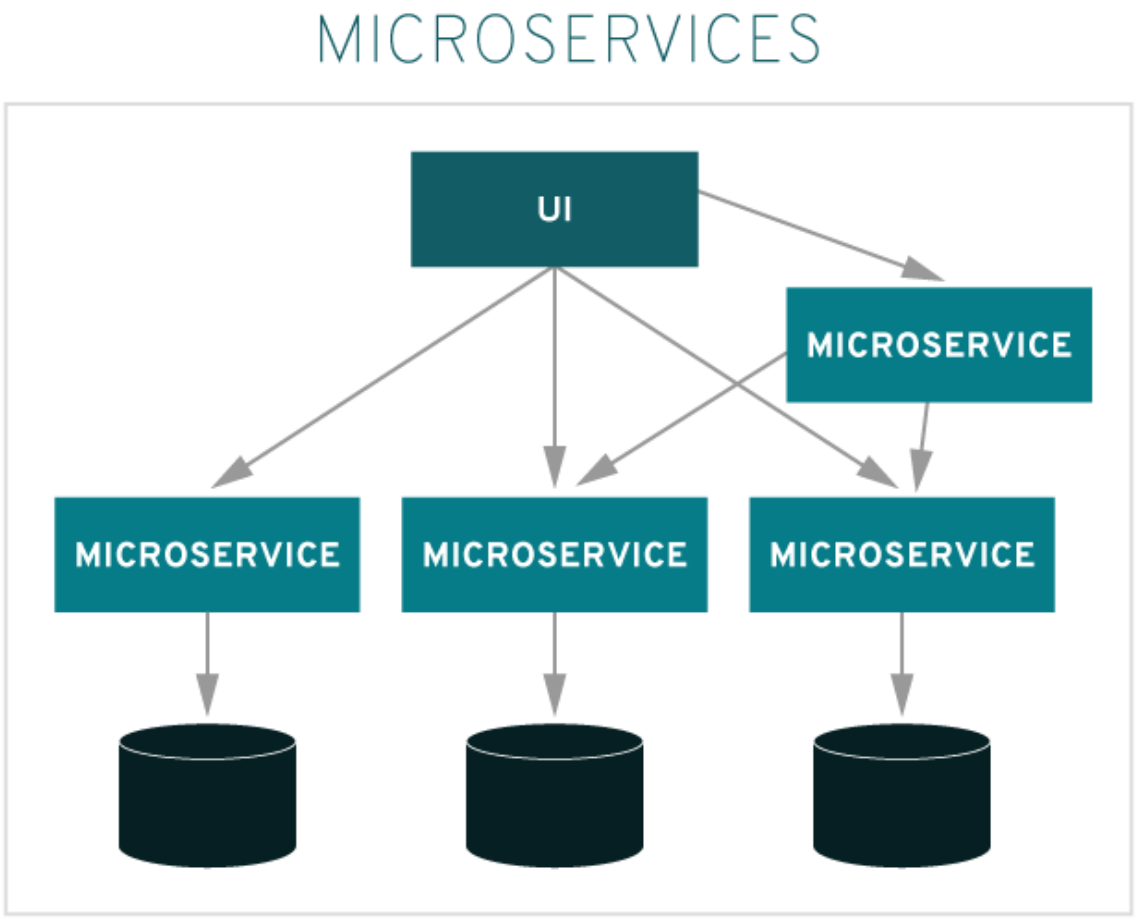
Uygulama büyüdükçe yeni özellik geliştirilmesi ve mevcut kodun bakımı zorlaşır.

Birbirlerine olan bağımlılıklarından dolayı, bir fonksiyonalitede yapılan değişiklik diğer yerleri etkileyebilir.

Uygulamada aynı programlama dili ve aynı frameworklerin kullanılması gerekir.

Uygulamada yapılan küçük bir değişiklikte bile bütün uygulamanın deploy olması gerekir.

Ekibe yeni bir developer katıldığı zaman uygulamanın bütün yapısını öğrenmek zorunda olması.



Microservice mimari birbirinden bağımsız olarak çalışan ve birbirleriyle haberleşen bireysel servislerdir. Her servis kendisine ait olan iş mantığını yürütür ve diğer servislerin iş mantığı ile ilgilenmez.
Servisler arasında ki iletişim HTTP, AMQP, TCP, UDP vb… gibi protokoller ile gerçekleşir.

Avantajları
Yeni özellik eklenmesi ve mevcut kodun bakımı kolaydır,

Her servis birbirinden bağımsız ve sadece kendi iş mantıklarını bulundurduğu için servisin code base'i oldukça sade olacaktır,

Ekibe yeni bir developer katldığı zaman uygulamanın bütün yapısını öğrenmek yerine katkı vermesi beklenen servisin yapısını öğrenmesi yeterlidir,

Bir servisde yapılan değişiklik, sadece ilgili servisin deploy yapılması yeterlidir.

Servisler farklı dillerde ve farklı frameworkler ile yazılabilir. Her servisin kendine ait farklı veritabanı olabilir.

Dezavantajları

Birden fazla servis ve birden fazla veritabanı olduğu için transaction yönetimi zorlaşacaktır.

Servisler arası iletişim ve uyumlu şekilde çalışması zorlayıcıdır.

Debug kolay değildir

Hata yönetimi kolay değildir.