**MİKROSERVİCE NEDİR ?**

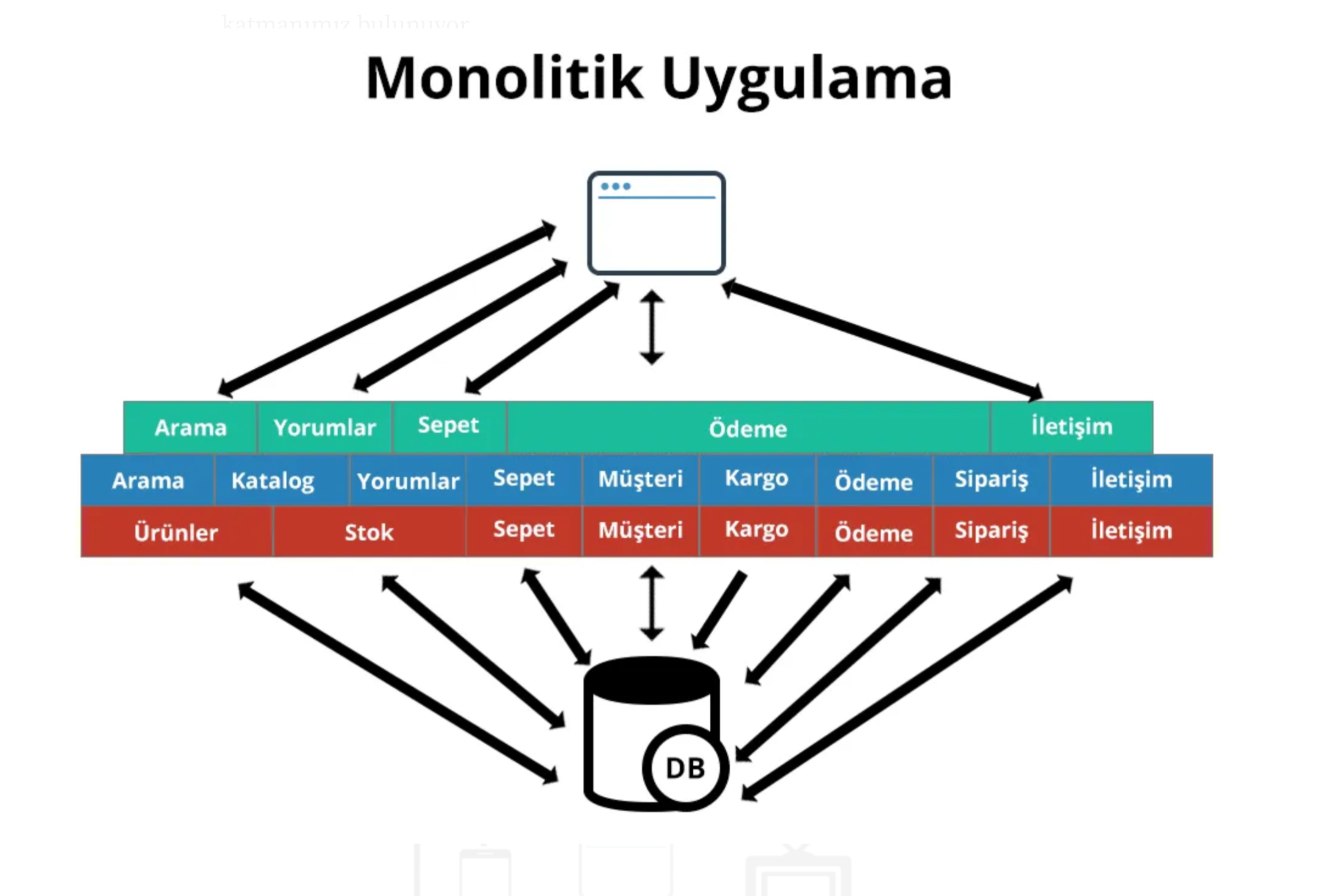
Mikroservis mimarisinin basitçe bir tanım yapacak olursak; Yekpare bir sistemin, her biri bağımsız olarak çalışan ve açık protokoller (örneğin http) vasıtasıyla birbiri ile iletişim kuran küçük servislere ayrılması diyebiliriz. Motolitik yapıya bir alternatiftir.

Yani uygulamanın küçük yapılara ayrılmasını diyebiliriz.

Aklımıza gelen uygulamalar ise

E ticaret -Bankacılık

Tek bir noktadan çıkıp tek bir noktaya gidiyor.



Bu katmanlara yakından bakacak olursak daha fazla detay görebiliriz.

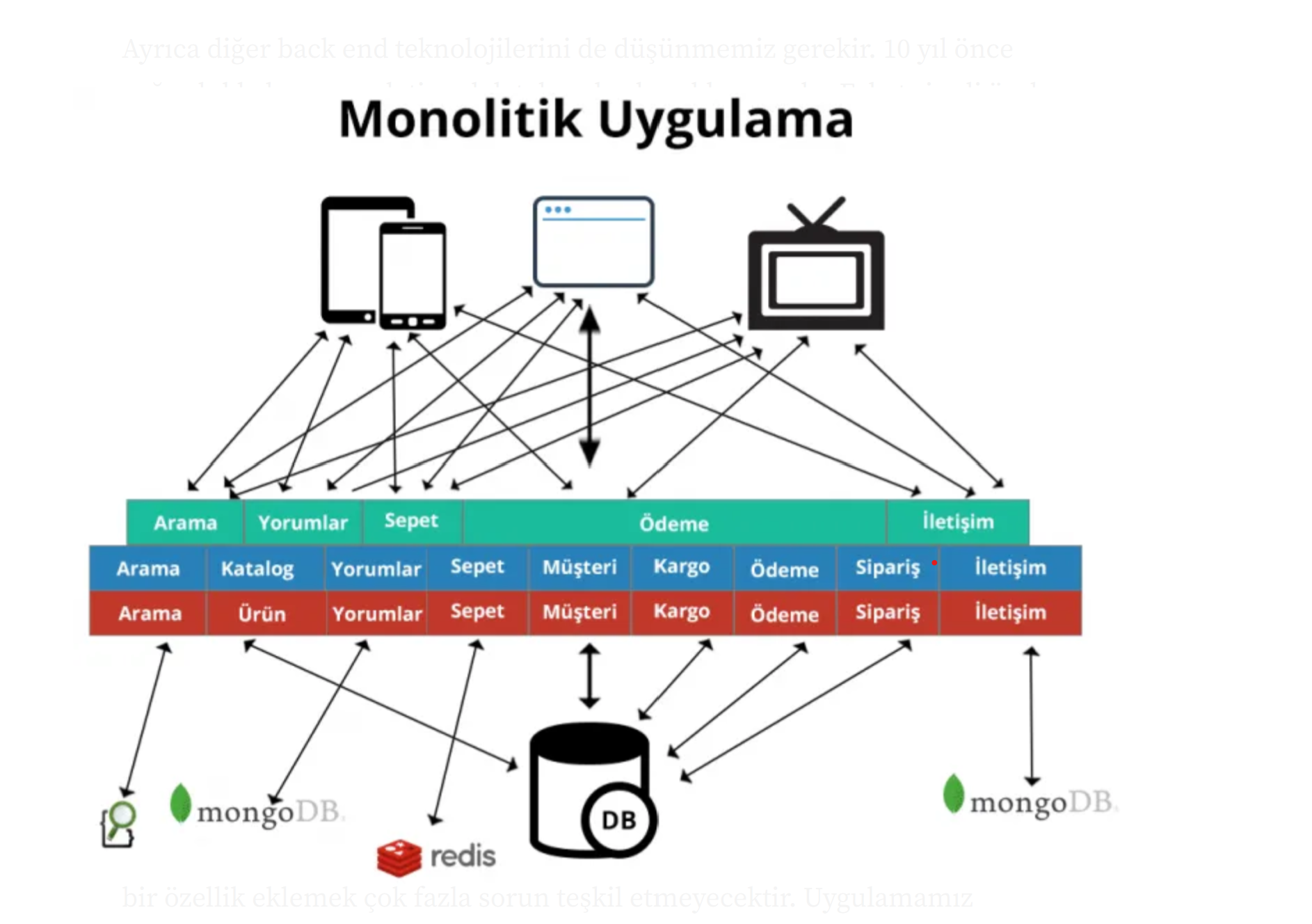
Uygulamamızda Arama, Yorumlar, Sepet gibi sayfalar için birçok Controller olduğunu görüyoruz.

Servis katmanı kullanım senaryolarımızı (use case) modellemeye çalışıyor.

Arama, Katalog, Yorumlar, Sepet gibi özellikleri kontrol etmek için ayrı ayrı servislerimiz var.  
Repository katmanında veriler veritabanı içerisinde ifade edildiği gibi veri modeline benzemektedir.

Yukardaki uygulama yapısı (structure) iyi organize edilmiş ve iyi şekilde anlaşılıyor. Peki neden farklı bir yapıya yönelim ya da ihtiyaç var?

Sektörde karşı karşıya olduğumuz zorlukları düşünmeye başlayalım. 10 yıl önce yalnızca web arayüzünü düşünmek zorundaydık. Şimdi ise akıllı telefonlar, tabletler, oyun konsolları ve Televizyonlar. Hatta akıllı saatler ve diğer giyilebilir teknolojiler. Controller web arayüzüleri için tasarlanmıştı, diğer kanallar için değil.

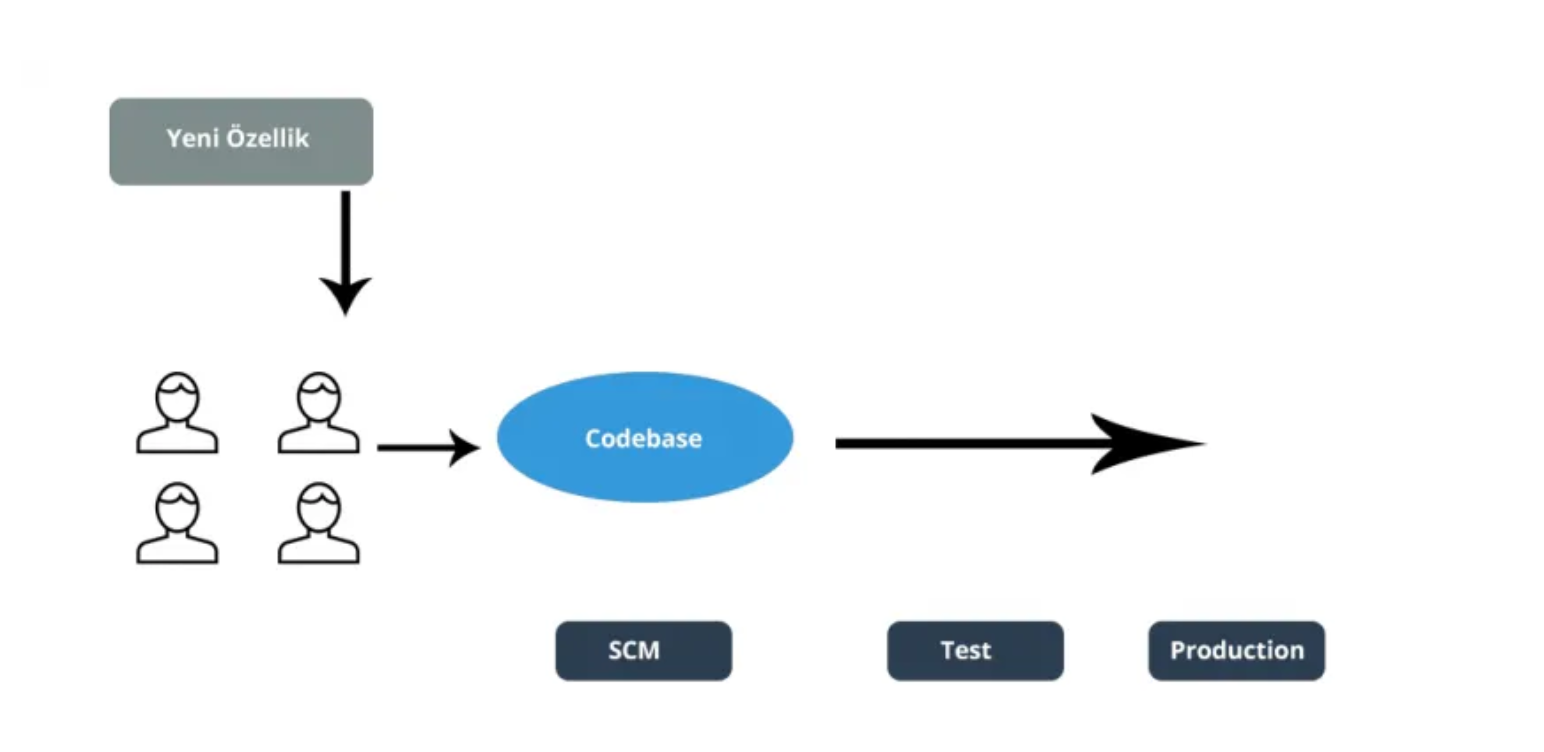


Ayrıca diğer back end teknolojilerini de düşünmemiz gerekir. 10 yıl önce çoğunlukla her şey relational databaselerde saklanıyordu. Fakat şimdi özel kullanım senaryoları (use case) için özelleşmiş, üstün performans sunan teknolojilere erişebiliyoruz. Örneğin uygulamamızdaki ürün arama kısmı. Ürün arama kısmı için relational database yerine **Elastic Search** gibi özelleşmiş bir teknoloji kullanırsak arama özelliğimiz çok daha hızlı olacaktır.

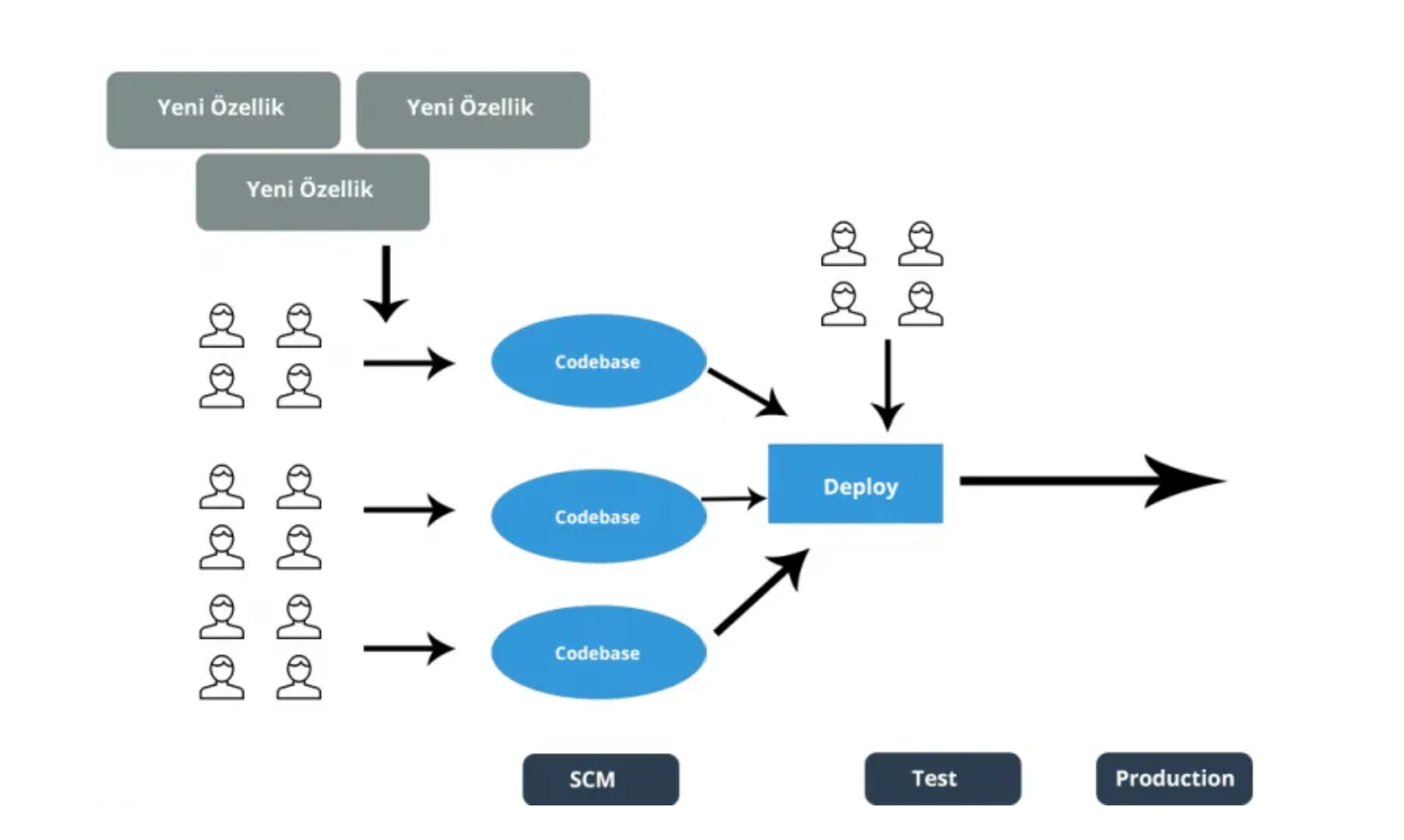
Ya da ürün yorumlarını **mongoDB** gibi document storage kullanan bir yapıda daha esnek bir şekilde saklayabiliriz. **Alışveriş sepeti Redis gibi key value şeklinde depolama yapan bir teknoloji ile çok daha hızlı ve basit olacaktır.**

Geleneksel uygulamalar(monolitik) tek bir dille yazıldığı için, tek bir dille yazılmış bu tek uygulamaya, yukarıda bahsettiğimiz back end teknolojilerini implemente etmek istediğimizde, bütün bağımlılıkları ve destekleyici kütüphaneleri projemize eklememiz gerekir. Tek bir executable uygulamada bu kadar çok bağımlılık ve destekleyici kütüphane olması projeyi biraz kasıntı bir duruma sokar.

Uygulama büyüdükçe kod yönetimi zorlaşacak. Aynı durum hata yönetimi için de geçerlidir .



Codebase büyüdükçe ekibi nispeten daha küçük takımlara ayırıp her bir takımın ayrı bir özelliği geliştirmesi düşünülebilir. Bu durumda ise aşağıdaki resimde de görebileceğiniz üzere ayrı takımların geliştirdiği özellikleri birleştirmek ve uygulamayı deploy etmek için ayrı ekip kurulması gerekebilir.

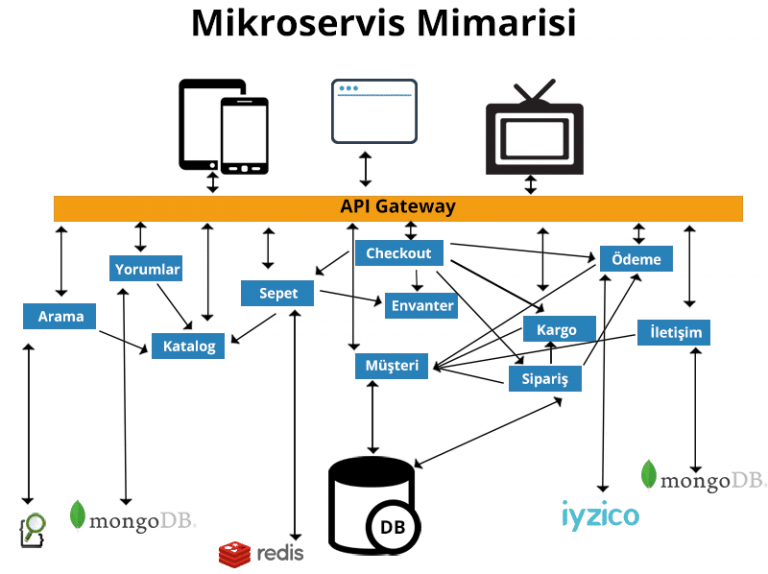


Buradaki en büyük problem ise bu hattın sonunda bir hata (bug) keşfedilirse ne olacağıdır. Doğal olarak hata düzeltilene kadar ya da bu hataya sebep olan özellik geri çekilene kadar uygulama deploy edilemeyecek. Bu durumda deploy ekibi her bir ekibin geliştirdiği modüllerin en iyi versiyonlarını bulup, bu versiyonlarla uygulamayı deploy etmek için sürekli çalışacak. Bütün bunlar deploy ekibi için büyük bir karışıklığa sebep olacak.

**Monolitik Mimarinin Dezavantajları**

• Yapının(structure) anlaşılması nispeten kolay, fakat hazmetmek zor.  
• Tek bir programlama dili ile geliştirilmesi. (PHP ile başlandıysa PHP ile geliştirilmeye devam edilmek zorunda)  
• Uygulamanın modülaritesinin geliştirilen programlama diline bağlı olması.  
• Uygulama büyüdükçe codebasein yönetilmesinin, bakımının ve deploy edilmesinin zorlaşması.  
• Ekibe yeni bir developer katıldığı zaman uygulamanın bütün yapısını(structure) öğrenmek zorunda olması ve projeye katkı vermeye (contribute) başlama süresinin artması.

**Peki Monolitik yapıya göre Mikroservislerin ne gibi farkları var?**



Mikroresvisler yukarıdaki resimde de görüldüğü üzere küçük, bağımsız, bütün sistemin fonksiyonel yapısı etrafında inşa edilmiş uygulamalardır. Birbirleriyle çalışmaya ihtiyaç duydukları zaman açık prorokoller üzerinden (http, udp, messaging) birbirleriyle serbestçe iletişim kurarlar. Genellikle bir ya da iki kalıcı teknolojiyle (mongoDB, redis vb) desteklendiklerinden bu teknolojiye en uygun programlama dili ile geliştirilebilirler.  
Resimde API Gateway isimli bir katman görüyorsunuz.Günümüzde birden fazla client teknolojisi (browser, mobil cihazlar, TVler) servislerimizi kullanmak istiyorlar. Bu clientların servislerimize direk erişmesi birçok probleme sebebiyet verebilir. Bu sebepten ötürü servislerimiz ve clientlar arasındaki bağlantıyı API Gateway ismini verdiğimiz bileşen üzerinden yapıyoruz.

# Mikroservislerin Avantajları

• Mikroservisler birbirinden bağımsız ve tek bir işe odaklanmış uygulamalar olduklarından, her bir servisi farklı bir programlama dili ile geliştirmek mümkün. Bu da uygulamanın bir programlama diline olan bağımlılığını ortadan kaldırıyor.  
• Büyük bir codebasein depoloy süresine oranla servislerin build ve deploy süreleri birbirinden bağımsız olacağından, developerlar açısından ciddi bir zaman kazancı sağlaması.  
• Ekibe yeni bir developer katldığı zaman uygulamanın bütün yapısını(structure) öğrenmek yerine katkı vermesi (contribute) beklenen servisin yapısını (structure) öğrenip katkı vermeye başlama süresinin oldukça çabuk olması.  
• Uygulamanın yatay eksende scale edilebilmesi.