BACKEND MİMARİLERİ

---

1 Monolitik Mimari

Tanım: Monolitik mimari, bir uygulamanın tüm bileşenlerinin tek bir kod tabanı üzerinde geliştirildiği ve yayınlandığı geleneksel backend mimarisidir.

Kullanım Alanları:

- Küçük ve orta çaplı projeler
- İlk kez geliştirilen MVP (Minimum Viable Product) uygulamaları

Temel Prensipler:

- Tüm bileşenler tek bir kod tabanında bulunur.
- Güncellemeler bütün uygulamayı etkiler.
- Veritabanı ve API katmanı genellikle aynı uygulamanın içinde yer alır.

Kullanım Senaryosu: E-ticaret siteleri, blog sistemleri

Dosya Yapısı:

- /controllers
- /models
- /views
- /routes
- /config

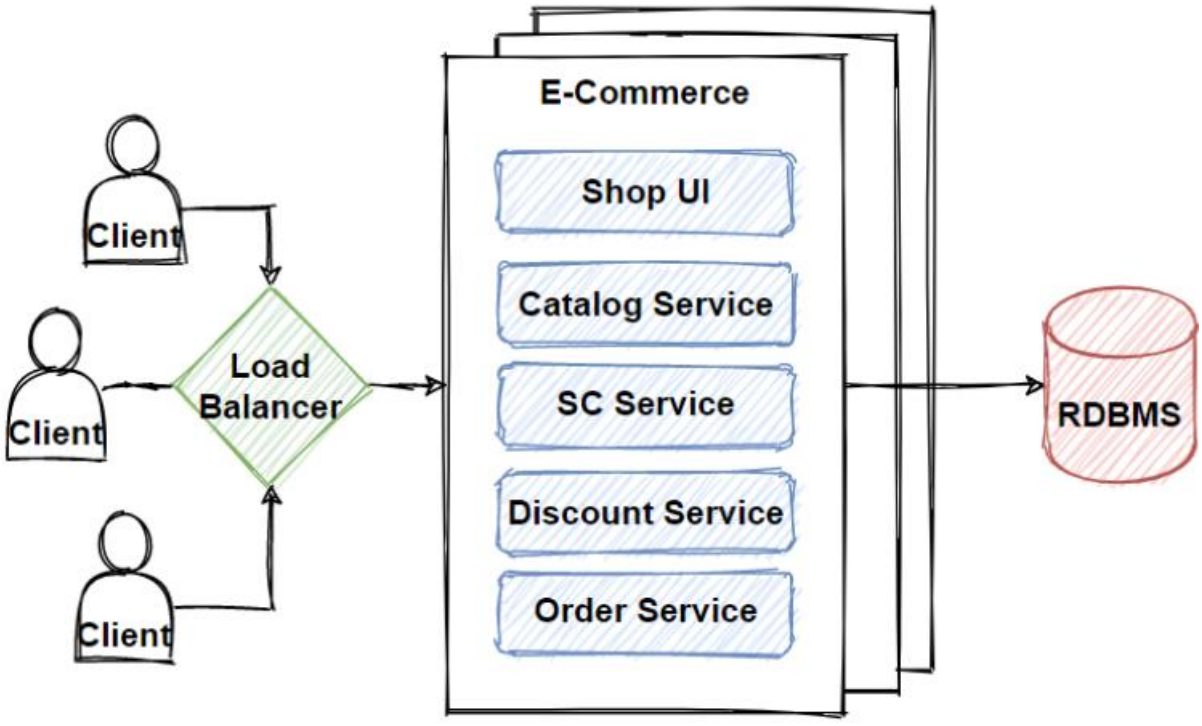
Avantajlar:

- Basit geliştirme süreci
- Düşük maliyet

Dezavantajlar:

- Büyük projelerde karmaşıklık artar
- Deployment süresi uzar

Monolithic Architecture



---

2 Mikroservis Mimari

Tanım: Mikroservis mimarisi, her bir uygulama bileşeninin bağımsız olarak geliştirildiği, çalıştırıldığı ve yönetildiği bir backend modelidir.

Kullanım Alanları:

- Büyük ölçekli projeler
- Sürekli güncellenmesi gereken sistemler

Temel Prensipler:

- Her servis bağımsız bir bileşendir.
- API üzerinden haberleşir.
- Otomatik ölçeklenebilirlik sağlar.

Kullanım Senaryosu: Amazon, Netflix

Dosya Yapısı:

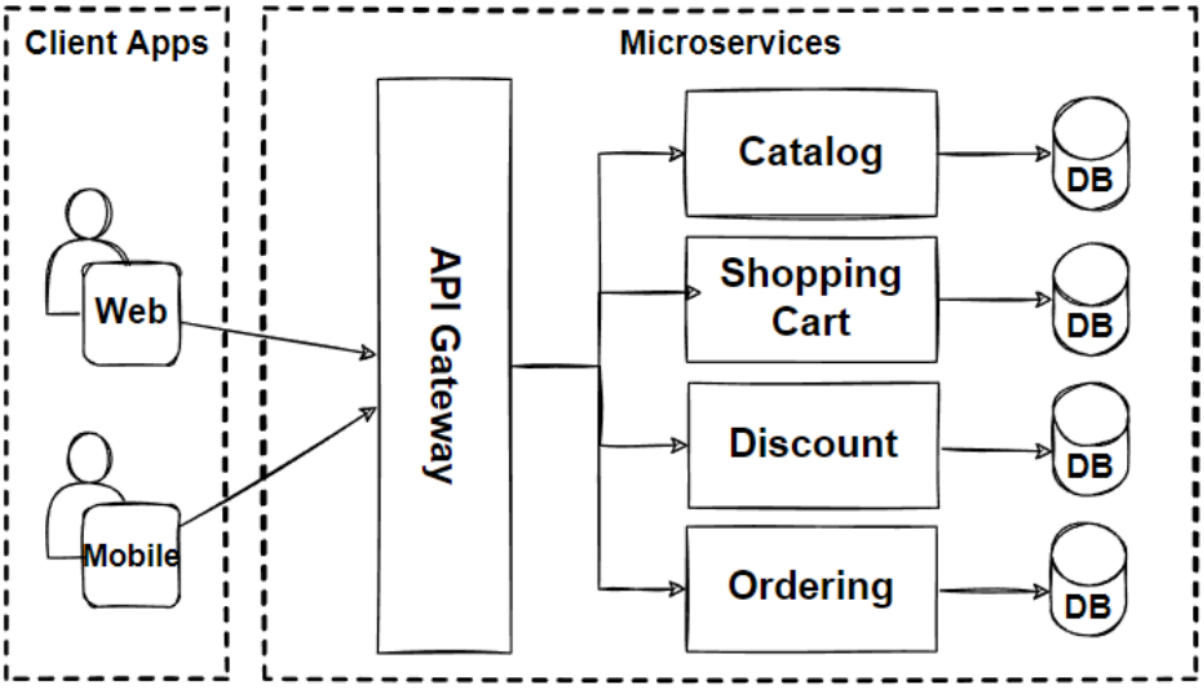
- /service1
- /service2
- /gateway

Avantajlar:

- Esneklik
- Bağımsız deployment

Dezavantajlar:

- Yüksek karmaşıklık
- Daha fazla kaynak tüketimi


Client Apps
Microservices
Web
Mobile
API Gateway
Catalog
Shopping Cart
Discount
Ordering
DB
DB
DB
DB

---

3 Serverless Mimari

Tanım: Serverless mimari, geliştiricilerin sunucu yönetimiyle uğraşmadan kodlarını çalıştırmalarına olanak tanıyan bir yapıdır.

Kullanım Alanları:

- Ani trafik artışı gösteren uygulamalar
- Gerçek zamanlı veri işleme sistemleri

Temel Prensipler:

- Otomatik ölçekleme
- Pay-per-use maliyetlendirme

Kullanım Senaryosu: AWS Lambda, Firebase Functions

Dosya Yapısı:

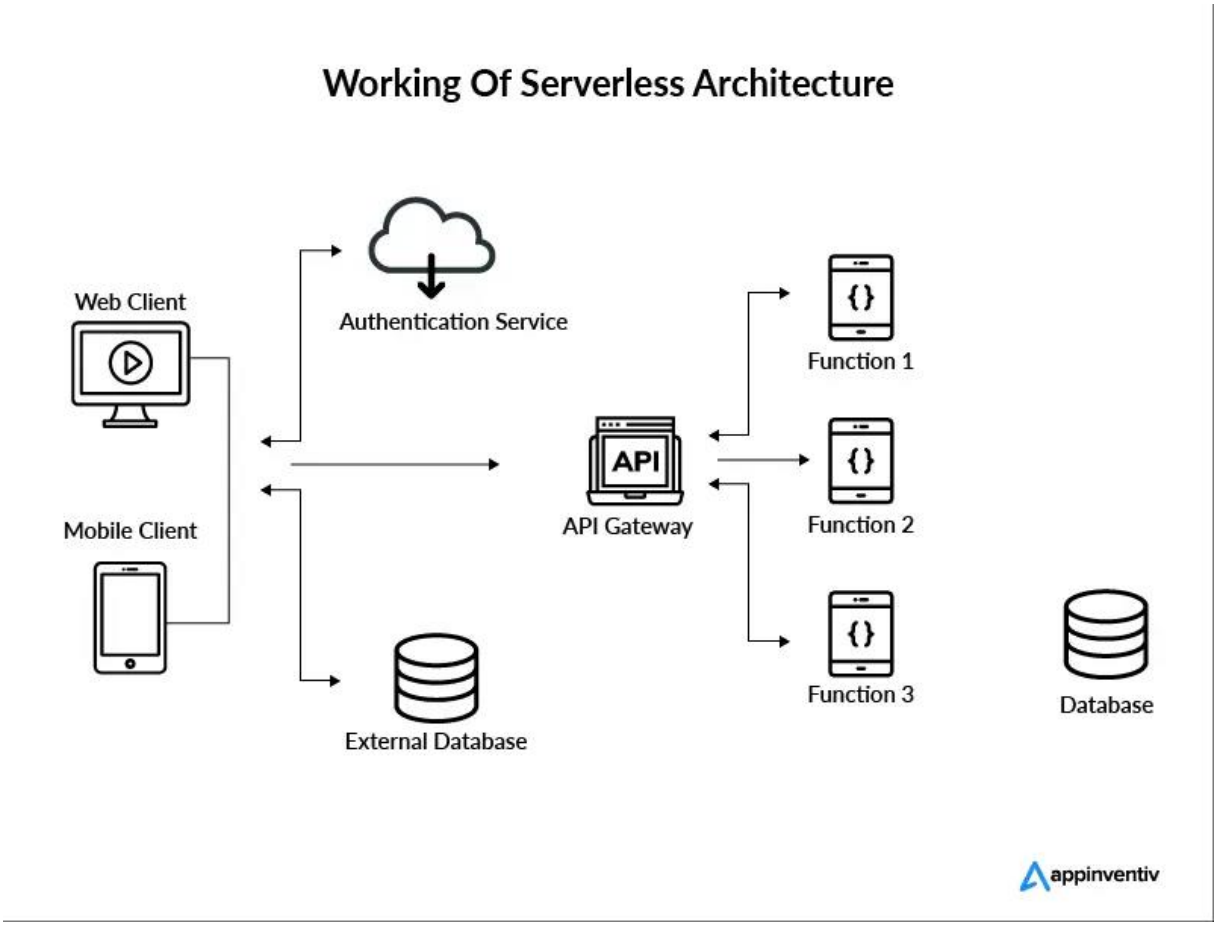
- /functions
- /triggers
- /logs

Avantajlar:

- Düşük maliyet
- Kolay ölçeklenebilirlik

Dezavantajlar:

- Soğuk başlatma süreleri
- Debugging zorluğu


Working Of Serverless Architecture
Web Client
Mobile Client
Authentication Service
API
API Gateway
Function 1
Function 2
Function 3
Database
External Database
appinventiv

---

4 Clean Architecture

Tanım: Clean Architecture, kodun okunabilirliğini, anlaşılabilirliğini ve sürdürülebilirliğini artırmayı amaçlayan bir yazılım geliştirme prensibidir.

Kullanım Alanları:

- Uzun vadeli projeler
- Ekip çalışması gerektiren sistemler

Temel Prensipler:

- Bağımsızlık
- Katmanlı mimari
- Single Responsibility Principle (Tek Sorumluluk Prensibi)

Kullanım Senaryosu: Bankacılık ve finans yazılımları

Dosya Yapısı:

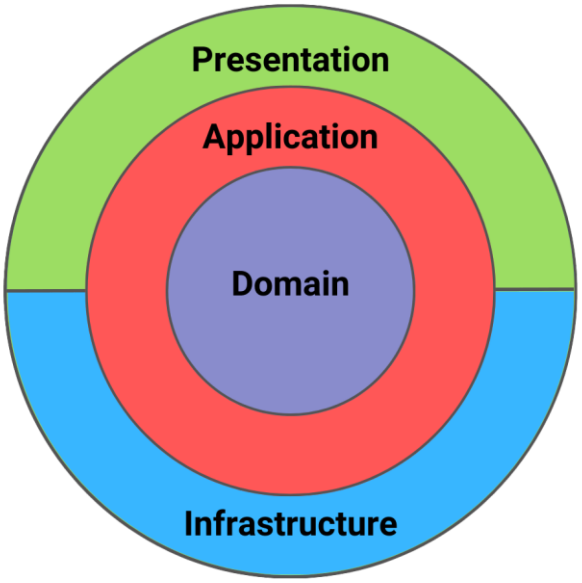
- /domain
- /usecases
- /infrastructure

Avantajlar:

- Uzun vadeli sürdürülebilirlik
- Modüler yapı

Dezavantajlar:

- İlk geliştirme maliyeti yüksek olabilir



Arif Sefa Bölükbaşı