Mikroservisler, 101

Serhat Kayıkçı

Workshop Github: Israrla isteyiniz!

serhatfreelancing@gmail.com

2008: Netflix Down for 3 Days

Source: https://archive.nytimes.com/bits.blogs.nytimes.com/2008/08/15/lessons-from-netflixs-fail-week/

- Single bug

- Entire site affected

- Millions in losses

- All because everything was connected

Today: Netflix handles 37% of internet traffic

- Can deploy 100+ times per day

- Single service failure doesn't crash site

- How? They learned to split services right

Neler konuşacağız?

1. Monolit nedir?
2. Efsaneler & Avantajları
3. Cohesion & Coupling
4. Workshop
5. Geçişe Başlama
6. Başka bir Monolit
7. Geçiş Öncesi Sorulacak Sorular
8. Sonraki Adımlar...

Ben kimim?

Teknik Uzmanlık

- Java, Spring Boot ve Mikroservisler
- Bulut Çözümleri (AWS ve Azure)
- Yazılım Mimarisi (ISAQB Sertifikalı)
- MLOps ve Bulut Mimarisi

Profesyonel Hizmetler

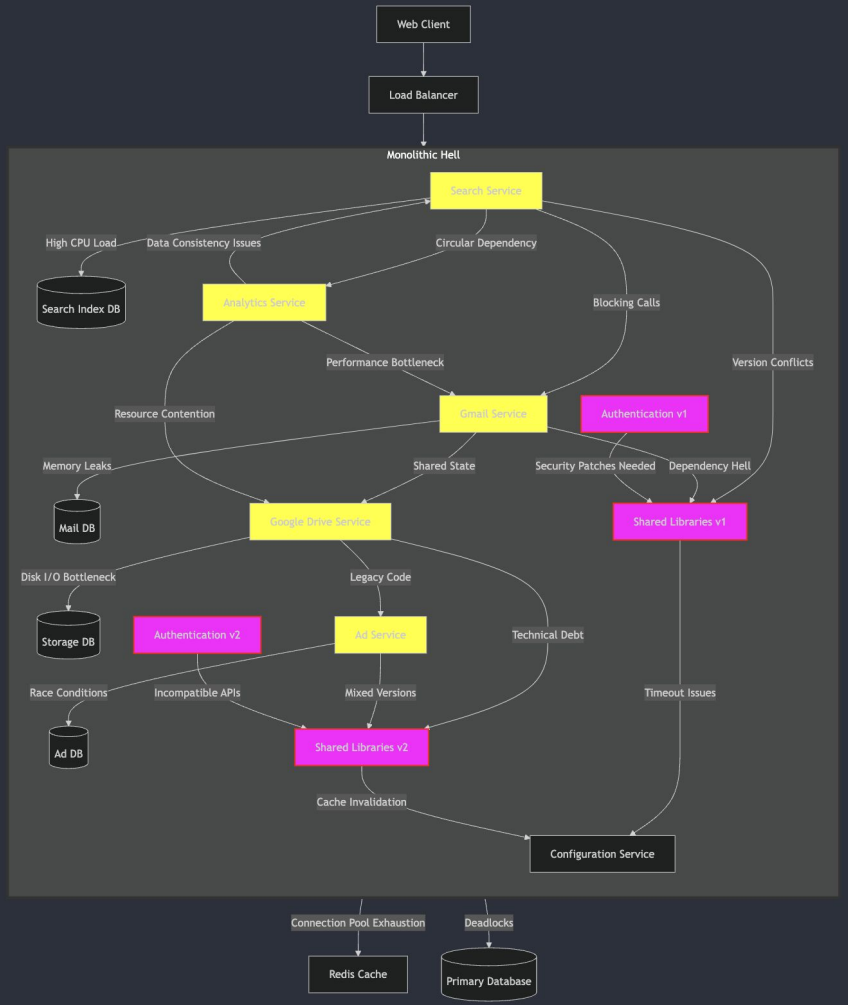
- Bulut Mimarisi Danışmanlığı
- Mikroservis Uygulamaları
- Teknik Mimari Tasarımı
- MLOps Çözümleri ve Danışmanlık
- Teknik Liderlik
- Teknoloji Girişimciliği

Almanya'da yerleşik | Serbest Çalışma Projelerine Açık

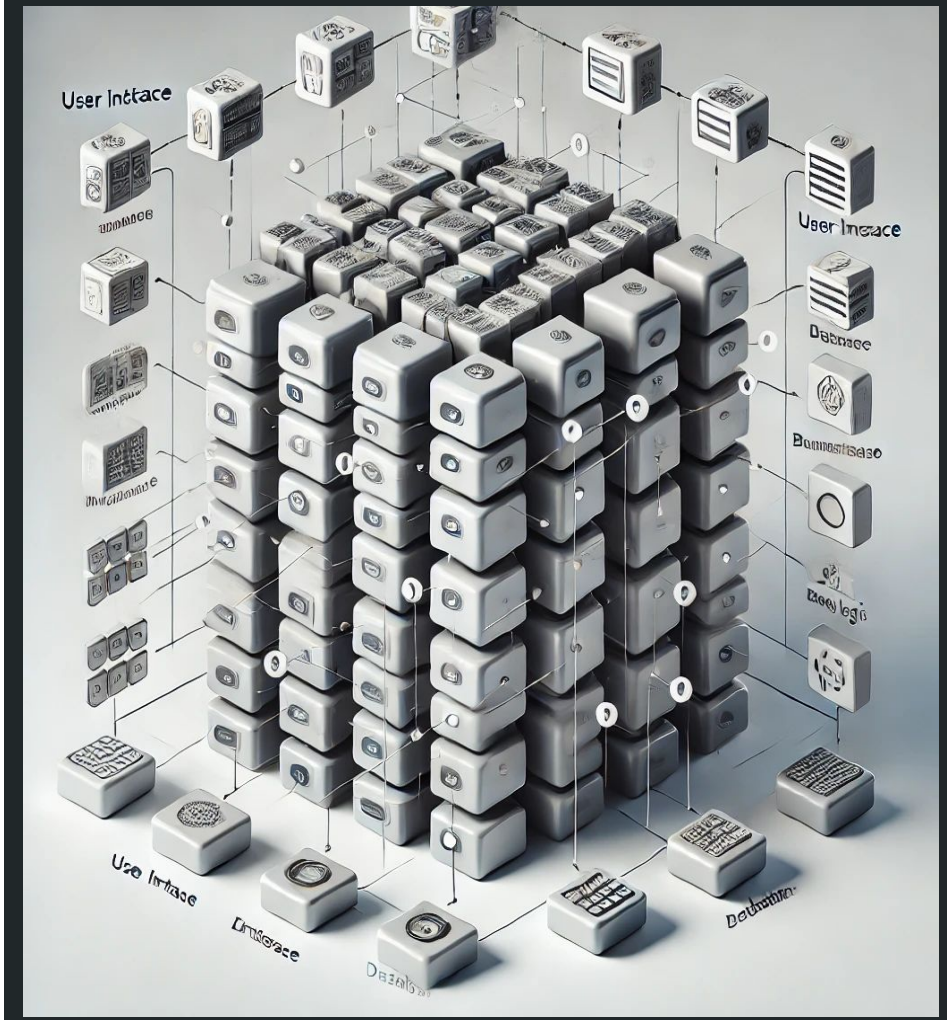

OMG IT'S
ICEBREAKER

Google
olsa

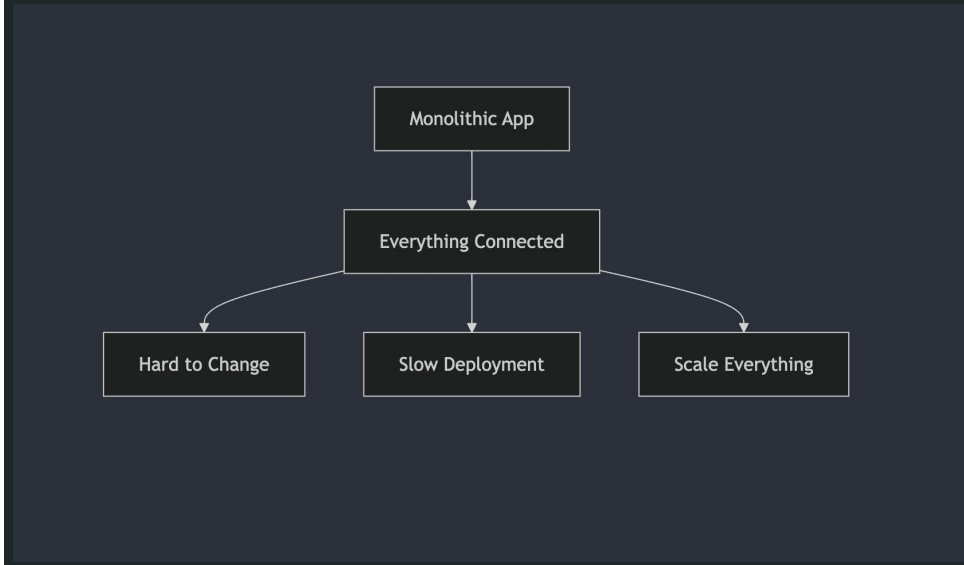
Neler yanl



Monolit nedir?



Efsaneler ve Avantajlar



Bağlanma & Yapışma (Cohesion & Coupling)

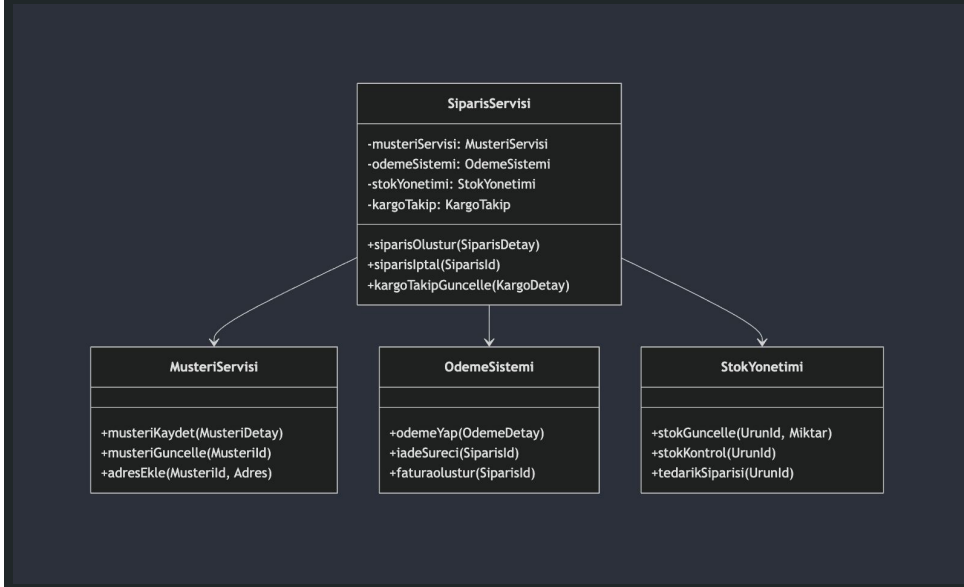
SCREEN-SHARING

WORKSHOP E-Ticaret Platformu

- Yoğun dönemlerde tüm sistem etkileniyor
- Black Friday'de sipariş modülü yavaşladığında ödeme sistemi de yavaşlıyor
- Stok kontrolü tüm sistemi bloklayabiliyor
- Her küçük değişiklik için tüm sistem deploy ediliyor
- Deploy süresi 45+ dakika
- Rollback karmaşık ve riskli
- Takımlar birbirlerine bağımlı ve teknik bağımlılıklar yüksek (aynı database, aynı codebase, vb)

Github Linki: Israrla isteyiniz!

WORKSHOP E-Ticaret Platformu



Good Signs for Splitting:

✓ Different scaling needs

✓ Separate business capabilities

✓ Clear data boundaries

✓ Independent teams

Warning Signs:

⚠️ Too many distributed transactions

⚠️ Chatty communication

⚠️ Shared mutable state

Geçişe Başlıyoruz

Phase 1: Assessment & Planning

1. Create Modularity
2. Domain Analysis
3. Identify Bounded Contexts
4. Map Dependencies

Phase 2: Foundation Setup

1. Setup Container Infrastructure
2. Implement API Gateway
3. Establish CI/CD Pipeline
4. Configure Monitoring

Te

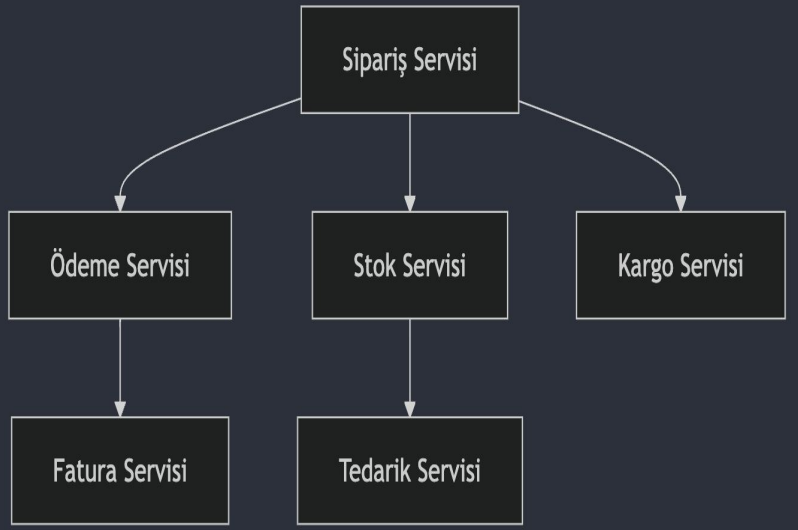
```java
// Before: Everything breaks together
@Service
public class OrderService {
    @Autowired
    private Everything everything; // 😱

    public void process() {
        // One bug crashes all
    }
}

// After: Independent services
@Service
public class OrderService {
    @Autowired
    private PaymentClient paymentClient; // 🤞

    public void process() {
        // Problems isolated
    }
}
```

1. Strangler Fig Pattern
2. Database per Service
3. Event-Driven Architecture
4. Service Mesh Implementation

Organizational Changes

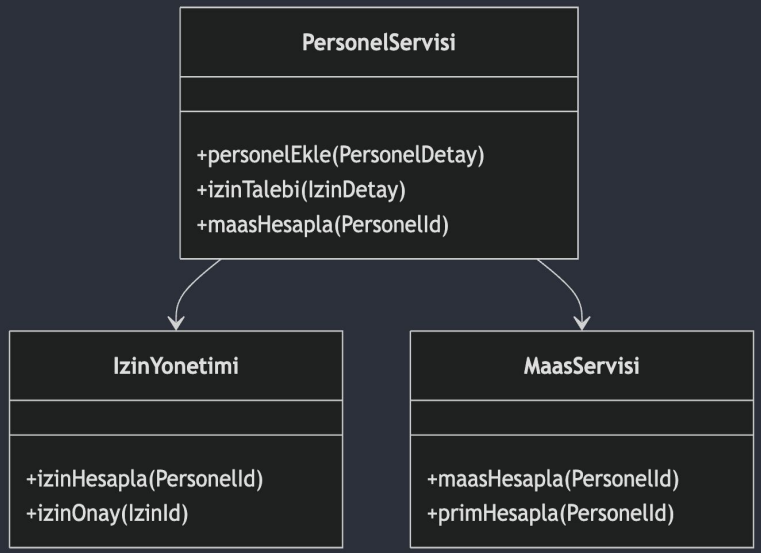
1. Cross-functional Teams
2. DevOps Culture
3. Documentation
4. Training & Skills

Örnek bir çözüm



Başka bir Monolit

Use case - İnsan Kaynakları Yönetim Sistemi



Neden ayrılmamalı?

- Takım sadece bu modül üzerinde çalışıyor
- Takım 3-5 kişi arasında
- Altyapı müsait değil
- Şirketin TTM gereklilikleri, oncelikleri
- Domain’ler ayrılamıyor
- Database yapısı karmaşık
- Sadece içeride kullanılıyor

Remember:

1. Start with monolith

2. Split when necessary

3. Clear boundaries

4. Handle failures

5. Test thoroughly

Soracağınız Sorular

- Know-how silo?
- Takım ve organizasyon hazır mı?
- Takımdaki Senior’lar rahatsız mı?
- Takımdaki yönetici bu değişikliğe hazır mı?
- DevOps takımınız var mı?
- Domain’leriniz belli mi?
- Zaman?
- Şans ;)

Sonraki adımlarda

- Service Templates
- Service Discovery
- Circuit Breakers
- Distributed Tracing
- DDD
- Service Mesh

- ???

- Microservice Patterns
- Event Driven Arch.
- Reactive Programming
- DB Consistency
- Load Balancing
- HATEOAS

TEŞEKKÜRLER!

Serhat Kayıkçı

serhatfreelancing@gmail.com

Feedback Form: [QR Code]