Yazılım Mimarisi ve Mikrohizmetler Nedir?

Çalışma Kağıdı

Yazılım mimarisi, büyük bir sistemin nasıl organize edileceğine dair bir plan iken, mikrohizmetler bu planın, bağımsız olarak dağıtılabilen küçük, odaklanmış hizmetler şeklinde uygulanmasıdır.

Sorular

1. Mikrohizmet mimarisinin temel amacı nedir?

A) Tüm işlevleri tek bir kod tabanında birleştirmek
B) Sistemi küçük, bağımsız ve yönetilebilir servislere ayırmak
C) Veritabanı karmaşıklığını artırmak
D) Geliştirme hızını yavaşlatmak

2. Aşağıdakilerden hangisi mikrohizmet mimarisinin bir avantajı DEĞİLDİR?

A) Teknolojik çeşitlilik
B) Hata izolasyonu
C) Artan operasyonel karmaşıklık
D) Bağımsız dağıtım

3. Monolitik bir uygulamada tek bir özelliğin güncellenmesi genellikle neyi gerektirir?

A) Sadece o özelliğin kodunu değiştirmeyi
B) Tüm uygulamanın yeniden dağıtımını
C) Diğer servislerin yeniden başlatılmasını
D) API Gateway'in güncellenmesini

4. Monolitik bir e-ticaret sitesi ile mikrohizmet tabanlı bir e-ticaret sitesi arasındaki temel farklar nelerdir?

5. Bir mikrohizmet mimarisinde servisler birbirleriyle nasıl iletişim kurar?

6. Mikrohizmet mimarisinin getirdiği temel zorluklar nelerdir?

7. Tanımla: Yazılım Mimarisi Nedir?

8. Tanımla: Mikrohizmet Nedir?

9. Tanımla: Monolitik Mimari Nedir?

Cevap Anahtarı

1. B) Sistemi küçük, bağımsız ve yönetilebilir servislere ayırmak — Mikrohizmet mimarisinin temel amacı, büyük ve karmaşık sistemleri daha küçük, bağımsız ve yönetilebilir servisler halinde bölerek geliştirme, dağıtım ve ölçeklendirme süreçlerini kolaylaştırmaktır.

2. C) Artan operasyonel karmaşıklık — Artan operasyonel karmaşıklık, mikrohizmet mimarisinin bir dezavantajıdır, avantajı değil. Servislerin bağımsız dağıtımı, teknolojik çeşitliliğe izin vermesi ve hataların diğer servisleri etkilememesi (hata izolasyonu) avantajlarıdır.

3. B) Tüm uygulamanın yeniden dağıtımını — Monolitik mimaride, tek bir özelliğin bile güncellenmesi genellikle tüm uygulamanın yeniden derlenmesini, test edilmesini ve yeniden dağıtılmasını gerektirir, bu da süreci yavaşlatır.

4. Monolitik yapıda tüm özellikler (ürün kataloğu, ödeme, kullanıcı yönetimi) tek bir büyük kod tabanında birleşiktir. Mikrohizmetlerde ise her özellik (örneğin, sadece ödeme işlemi) kendi bağımsız servisidir. Bu, bir servisteki hatanın tüm sistemi çökertmesini engeller ve farklı servislerin farklı teknolojilerle geliştirilmesine olanak tanır.

5. Servisler genellikle hafif protokoller (RESTful API'ler, gRPC) veya mesaj kuyrukları (Kafka, RabbitMQ) aracılığıyla senkron veya asenkron olarak iletişim kurar. Bu iletişim, servislerin birbirlerinin iç işleyişinden bağımsız kalmasını sağlar.

6. Dağıtık sistemlerin doğası gereği karmaşıklık artar. Servisler arası iletişim, veri tutarlılığı, dağıtık hata ayıklama ve operasyonel yönetim (izleme, dağıtım) gibi konularda ek çaba gerektirir.

7. Bir yazılım sisteminin bileşenleri, yapıları, davranışları ve aralarındaki ilişkileri tanımlayan soyutlama.

8. Bağımsız olarak geliştirilebilen, dağıtılabilen ve ölçeklenebilen, belirli bir işlevi yerine getiren küçük, otonom servis.

9. Tüm işlevlerin tek bir büyük, birleşik kod tabanında toplandığı geleneksel yazılım mimarisi.