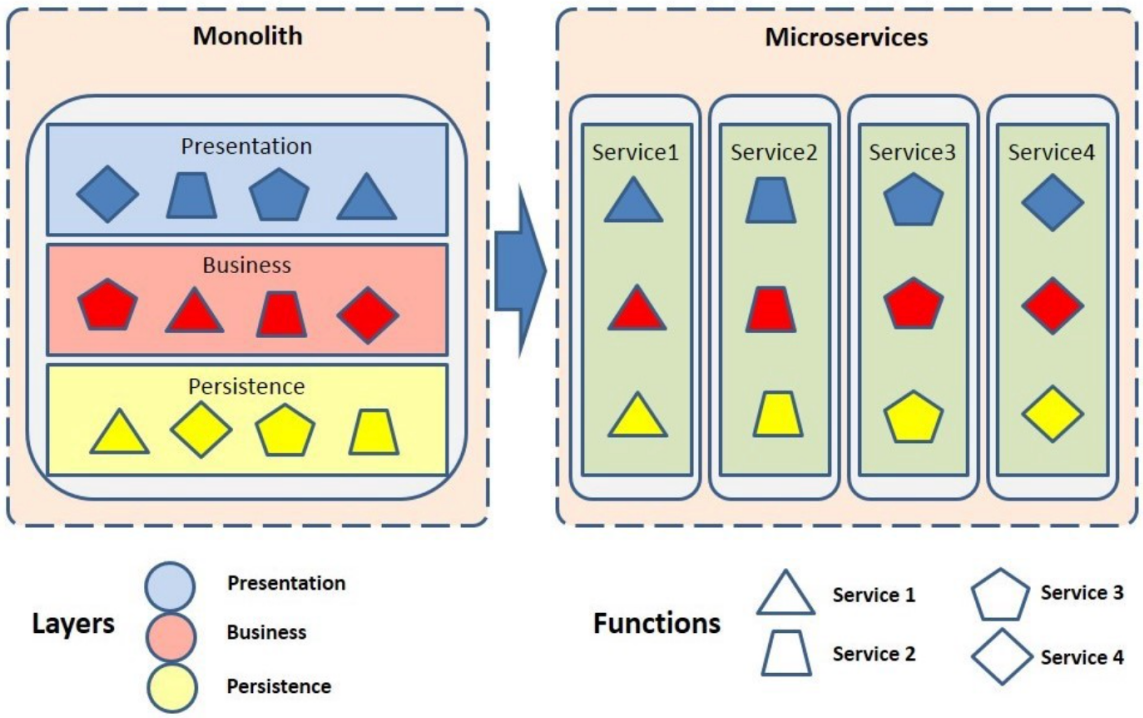

Monolithic ve Microservice Mimarileri

Microservice Nedir?

Microservice'ler, yalnızca tek bir işi gerçekleştirmekten sorumlu servislerdir. Birden fazla microservice'in birleşmesi ile oluşan yapıya Microservice Mimari adını veriyoruz. Az önce anlattığım Monolithic SOA'larda yaşanan çeşitli sorunları çözmek adına geliştirilmişlerdir. Buradaki amaç, uygulamayı parçalarına ayırıp, birbirleriyle fiziksel bağı olmayan minik uygulamacıklar oluşturmaktır. Oluşturulan bu uygulamacıkların derleme, test, deployment adımları da, DevOps yöntemleri uygulanarak otomatize edilir.

Monolithic Nedir?

Monolithic mimari kendi kendine yetebilecek bir uygulamada ki bütün fonksiyonalitelerin tek bir çatı altında geliştirilmesidir.
Günümüze kadar geliştirilen uygulamaların neredeyse hepsi monolithic mimari ile geliştirilmiştir.
Monolithic mimarisinin avantajları ve dezavantajları bulunmaktadır.

Avantajları

- Yönetilebilirlik ve monitoring kolaydır.
- Küçük çaplı projeler için geliştirilmesi ve bakımı kolaydır. Hızlı bir şekilde uygulama geliştirilebilir.
- Fonksiyonalitelerin birlikte çalışabilirliği tutarlıdır.
- Transaction yönetimi kolaydır.

Dezavantajları

- Uygulama büyüdükçe yeni özellik geliştirilmesi ve mevcut kodun bakımı zorlaşır.
- Projede çalışan ekip sayısının ve çalışan sayısının artması ile geliştirme ve bakım daha güç hale gelir.
- Birbirlerine olan bağımlılıklarından dolayı, bir fonksiyonalitede yapılan değişiklik diğer yerleri etkileyebilir.
- Spesifik bir fonksiyonaliteyi scale edebilme imkanı yoktur. (Örneğin geliştirdiğiniz uygulamada sürekli fatura oluşturuluyor ve burası uygulamanın dar boğazı. Siz bu fonksiyonaliteyi birden fazla instance da çalıştırmak isteseniz bile uygulamanız monolithic mimaride olduğu için sadece ilgili servis yerine bütün uygulamayı scale etmek zorunda kalırsınız.)
- Versiyon yönetimi zorlaşır.
- Uygulamada aynı programlama dili ve aynı frameworklerin kullanılması gerekir.
- Uygulamada yapılan küçük bir değişiklikte bile bütün uygulamanın deploy olması gerekir.

JUnit Platform: Bize sağladığı olanak, JUnit4 ve Junit5 ile birlikte çalışmayı sağlayan bir platformdur. JUnit Platform ile çalışmamızın sebebi projede private ve public methodları birlikte test etmeyi sağlar. Private methodlar için PowerMockito kullanıyoruz(JUnit4 te çalışıyor). Public methodlar için Mockito kullanıyoruz(JUnit5 te çalışıyor)
JUnit Vintage(JUnit4) : JUnit4 bize Private methodlarda Powermockito kullanarak database ile bağlantımızı kesip mocklamamızı sağlıyor.
JUnit Jupiter(JUnit5) : JUnit5 Public methodarda kullanacağımız nesneyi mocklayarak testimizi yazamaya olanak sağlar.
Assertion method
JUnit5 ile yapılan test işlemleri için Assertions sınıfında yer alan overload edilmiş statik assert*** metotları kullanılır.

Bazı assert metotları şunlardır;

- assertEquals
- assertNotEquals
- assertArrayEquals
- assertIterableEquals
- assertNull
- assertNotNull
- assertLinesMatch
- assertNotSame
- assertSame
- assertTimeout
- assertTimeoutPreemptively
- assertTrue

- assertFalse
- assertThrows