Bir web uygulaması geliştirilirken iki esas vardır

bunlardan biri veriyi üreten kısımdır.(backend)

bir diğeri ise backendin ürettiği veriyi dış dünyaya veren o veriyi kullanan kısımıdır.(client)

KATMANLI MİMARİ

dış dünyadan birtane katmana istek geliyor bu katman kendisini bağlayan diğer katmana veriyi iletiyor.oda ihtiyaç doğrultusunda diğerine iletiyor.

(her katman bir sonraki katmana iletiyor)

3 katman

işte en sonuncu katman veriyi üretiyor ortadaki katmana gönderiyor ortadaki katmanda ilk katmana iletiyor o katmanda veriyi isteyen adama iletiyor

İSİMLENDİRMESİ

PL: Dış dünyadan gelen istekleri biz pl presntation layer dediğimiz katmanda karşılıyoruz. ne katmanıdır bu sunum katmanı.

bir isteği almakta o istek neticesinde gerçekleşen şeyi sunmak hep bu sunum katmanının işi

mesela : son zamanlarda satılan en çok çorapların listesini istiyorsun isteği yaptın. PL bu isteği alıyor daha sonra geleneksel mimarimizde Bl dediğimiz buisness layer yani işlem yapan algoritmaların olduğu işlemlerin yapıldığı katmandır. servislerin olduğu algoritmaların operasyonların gerçekleştiği buisness iş yapacağın katman burasıdır.

1 ile 1 i mi toplayacaksın plde toplamazsın son katmanda da toplamazsın blde toplarsın.

istek geldi en çok satılan çoraplar bl ilgili operasyonları çalıştırdı. BLde nereye gidecek DAL dediğimiz DATA ACCESS Layere veri tabanına gidecek bağlancak ordan en çok satış yapılan çorapları getirecek. al sana en çok satış yapılan çorapları getirecek bunlar blye dönecek blde işlenecek, blden sunum edilebilir hale getirldikten sonra plye gönderilecek plde de sunum edilecek

hay hay bu yıllardır birçok projede kullanılan mimari

bu geliştirilebilir araya yok entity girebilir. test katmanı girebilir vs.

Temeldeki fıtratta olması gereken katmanlar bunlar.

şimdi biz neden bunu kullanmaycağız?

Bunun birçok sebebi var ama en önemli sebeplerinden biri değişikliğe o kadar çok dirençli ki herhangibir ihtiyaca binaen bir değişiklik yapmamız gerektiği zaman burda inanılmaz bir efor sarf etmemiz gerekiyor.

Bunu yaşayan bilir tecrübeyle sabittir.

O yüzden günümüzdeki birçok yazılımda geleneksel mimariyle geliştirlmiş proje bulmanız zordur.

Peki nedir bizden bunu soğutan diye tek tek maddeleri yazmaya başlayalım.

* Katmanlar arası sıkı bağımlılık mevcuttur.(katmanlar arası çok sıkı bağ var ne bi attraction var burda soyutlanma yapılanması mevcut bl direkt dala bağlanmış pl aynı şekilde bl üzerindeki servisleri kullanıyor. Yani uzun lafın kısası biz sıkı bağımlılık olan tüm yapılardan kaçıyoruz. Çünkü günümüzdeki ihtiyaçlar o kadar değişken ki burdaki bağımlıklar bizleri yapılması erekn işlerden koparıyor. Sadece buradaki bağımlıklarla uğraştırıyor. )
* Şimdi her katman hiyeraşik olaarak altındaki katmana bağlılık bağımlılık gösteriyor. Her katman tabikide birbirne bağlılık göstercek onion architecturda da bu böyle ama büyük ölçekli ve karmaşık uygulamalarda bir süre sonra yetersiz kalıyor.
* Bağımlık biryerden sonra hakikaten ayak bağı olmaya başlıyor