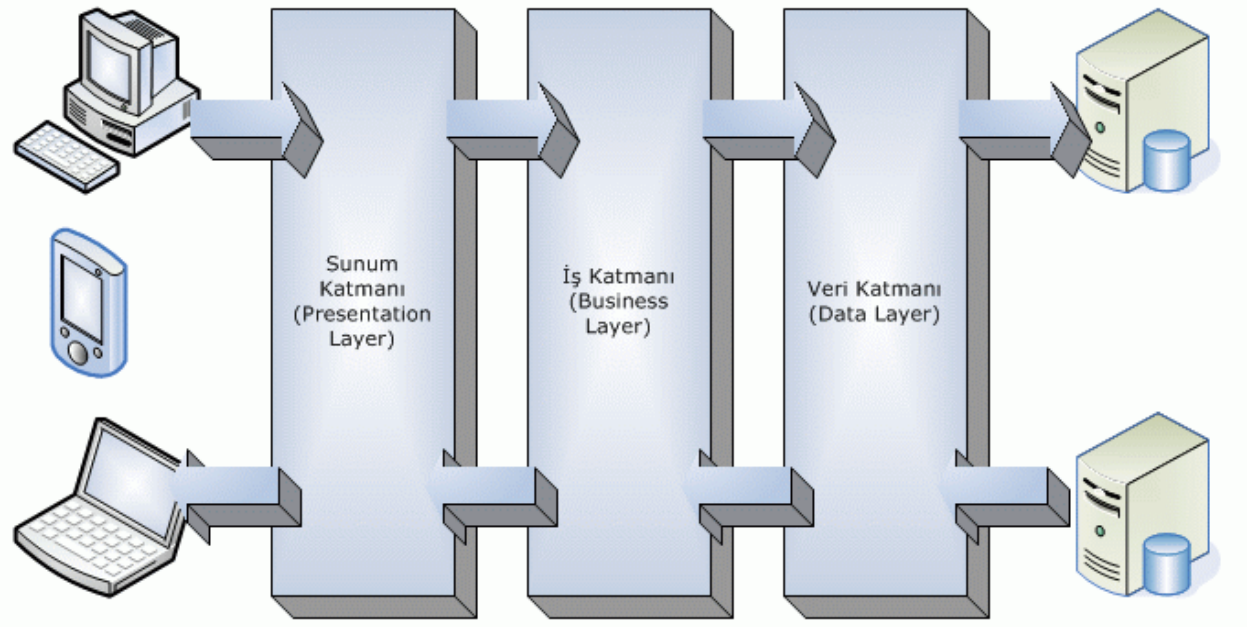
Katmanlı Mimari

Merhaba Arkadaşlar,

Günümüzde teknolojinin cok hızlı bir şekilde gelişmesi ile kullanıcılardan gelen isteklerin sayısı çok hızlı bir şekilde artmaktadir. Projeye yeni isteklerin implemente edilmesi yazılım projesinin daha karmaşık bir yapıya dönmesine sebep olmaktadır. Bu karmaşık yapıdan dolayı projenin okunabilirliği azalmaktadır. İşte bu karmaşayı yönetebilmek için katmanlıı mimari ortaya çıkmıştır. Katmanlı mimari projelerimizi belirli bir standart ve düzene göre geliştirmemizi sağlayan, kodun okunabilirliğini arttıran, projelerimizin daha derli toplu olmasını sağlayan ve hata yönetimini daha kolay hale getiren bir yapıdır.

Yazılımlarda veriye nasıl erişileceği, üzerinde nasıl işlemler yapılacağı ve bu işlemlerib kullanıcıya nasıl gösterileceği gibi işlemleri katmanlı mimari ile çok iyi bir şekilde yönetebiliyoruz. Katmanlı mimari sayesinde bu yapıyı parçalara ayırarak bu işlemlerin daha iyi yönetilebilmesini sağlıyoruz. Örneğin Windows form uygulaması geliştriyorsunuz ve veritabanındaki bilgileri ekrana yazdırmak istiyorsunuz bunu ilgili formun altına gerekli kodlara yazarak yapabilirsiniz. Ancak yarın birgün projede bir değişiklik yapılacağı zaman tüm kodlarınızı bu değişkiliğe göre değiştirmek durumunda kalabilirsiniz. Birden fazla formunuzun olması durumunda ve bu formların hepsinde benzer işlemleri yapıyorsanız gereksiz kod tekrarı yapıyorsunuz demektir. Bundan dolayı projemizi parçalara bölerek projemizi sürdürülebilir duruma getiriyoruz.

Peki nasıl bir yapıya sahip bu katmanlı mimari ? Katmanlı mimari temelde 3 katmandan oluşmaktadır. Bu 3 katman genelde her projenizde olması gereken katmanlardır. Siz bu 3 katmandan daha fazla bir katmanlı yapıda oluşturabilirsiniz bunada çok katmanlı mimari denilmektedir. Hadi gelin temeldeki 3 katmanı inceleyelim.



Katmanlı Mimari

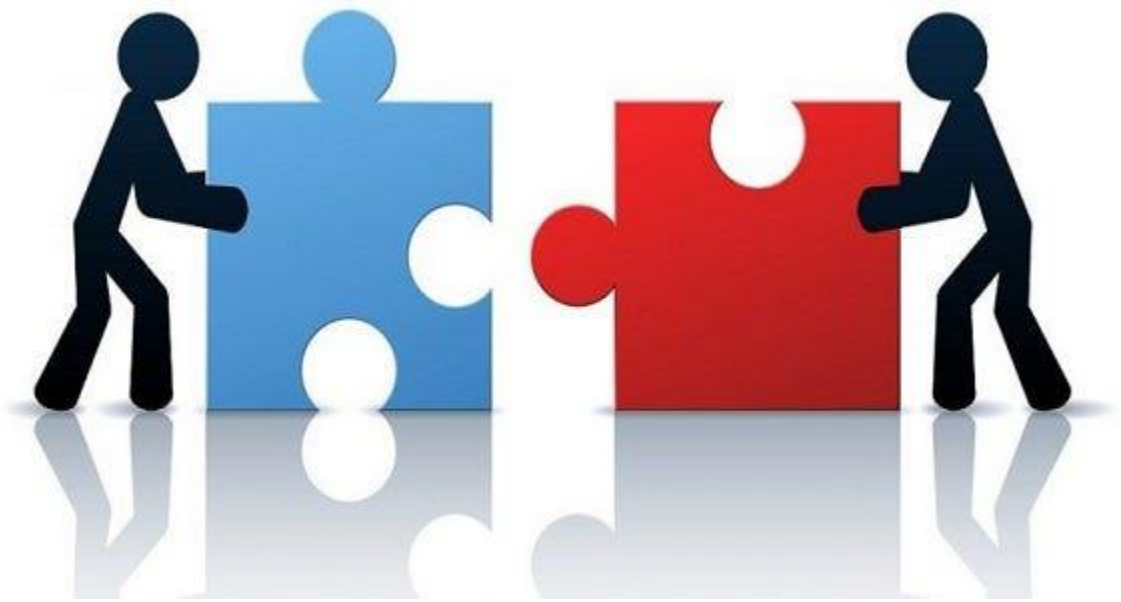
Data Access Layer : Bu katmanda sadece veritabanı işlemleri yapılmaktadır. Bu katmanın görevi veriyi ekleme, silme, güncelleme ve veritabanından çekme işlemidir. Bu katmanda bu işlemlerden başka herhangi bir işlem yapılmamaktadır.

Business Layer : Bu katmanda iş yüklerimizi yazıyoruz. Öncelikle şunu söylemeliyim bu katman Data Access tarafından projeye çekilmiş olan verileri alarak işleyecek olan katmandır. Biz uygulamalarımızda Data Access katmanını direk olarak kullanmayız. Araya Business katmanını koyarak bizim yerimize Business'ın yapmasını sağlarız. Kullanıcıdan gelen veriler öncelikle Business katmanına gider oradan işlenerek Data Access katmanına aktarılır. Bu katmanda ayrıca bu verilere kimlerin erişeceğini belirtiyoruz. Örneğin IT ve Muhasebe bölümü var. IT bölümünün veri tabanına ekleme işlemleri yapmasını istiyoruz ama Muhasebe bölümünün sadece verileri çekmesini istiyorsak bunu Business Katmanında gerçekleştiriyoruz.

Presentation Layer: Bu katman kullanıcı ile etkileşimin yapıldığı katmandır. Burası Windows form da olabilir, Web'te olabilir veya Bir Consol uygulamasida olabilir. Burada temel amac kullaniciya verileri göstermek ve kullanıcıdan gelen verileri Business Katmanı ile Data Access'e iletmektir.

Entities : Arkadaşlar temeldeki 3 katmanı inceledik. Entities Katmanında ise genelde domain olarak adlandırılan classlarımızı tanımlıyoruz. Bu Entities katmanının ismini domain olarakta değiştirebilirsiniz veya Common katmanıda yapabilirsiniz. Ben Entities ismini tercih ediyorum. Bu katmanda proje boyunca kullanacağımız ana classlarımızı

belirliyoruz yani gerçek nesnelerimizi belirlediğimiz yer burası. Daha anlaşılabilir bir şekilde anlatmak için birkaç örnek vereyim. Örneğin bir Stok veritabını sistemi yapmak istiyorsunuz. Bu sistemde Ürün bilgileriniz, Kategori bilgileriniz ve Satış bilgilerinizin olduğunu varsayalım. İşte bu bilgilerinizi burada tanımlıyorusunuz. Ürün classınız içerisinde Property olarak ÜrünAdı, ÜrünID si, ÜrünStokMiktari gibi, ÜrünFiyatı gibi propertyler olabilir. Bu katmanı hem Data Access hem Business hemde Presentation katmanı kullanmaktadır.



Yukarıda bahsettiğim yapıyı eğer projelerinizde uygulayabilirseniz sisteminizin sürdürülebilirliği artacaktır ve böylece herhangi bir değişiklik durumunda gerekli yeri tak çıkar mantığı ile istediğiniz şekilde değiştirebileceksiniz bu sayede diğer katmanlar bu değişimden etkilenmeyecektir. Bir sonraki yazımda ise önce Dependency Inversion prensibini anlatacağım ardından Katmanlı mimari hakkında bir örnek yaparak bu konuları daha anlaşılabilir bir hale getireceğiz. Bir sonraki yazımda görüşünceye kadar kendinize iyin Koda Bağımlı Kalmayın Arkadaşlar...