T.C.
İZMİR KATİP ÇELEBİ ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ
YAZILIM MÜHENDİSLİĞİ ANABİLİM DALI

Bilgisayar yazılımında web geliştirme ve web kütüphanelerin incelenmesi

Yüksek Lisans Projesi

ALTAN AKDAR

İZMİR-2022

T.C.
İZMİR KATİP ÇELEBİ ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ
YAZILIM MÜHENDİSLİĞİ ANABİLİM DALI

BİLGİSAYAR YAZILIMDA WEB GELİŞTİRME VE WEB KÜTÜPHANELERİN İNCELENMESİ

Yüksek Lisans Projesi

ALTAN AKDAR

DANIŞMAN: PROF. DR. FEMİN YALÇIN KÜÇÜKBAYRAK

İZMİR-2022

YEMİN METNİ

Yüksek Lisans Projesi olarak sunduğum "Bilgisayar yazılımında web geliştirme ve web kütüphanelerin incelenmesi" adlı çalışmanın, tarafımdan, akademik kurallara ve etik değerlere uygun olarak yazıldığını ve yararlandığım eserlerin kaynakçada gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve bunu onurumla doğrularım.

05/12/2022

Adı SOYADI

ALTAN AKDAR

DÖNEM PROJESİ ONAY FORMU

ÖZET

Yüksek Lisans Projesi

PROJENİN ADI

Bilgisayar Yazılımında Web Geliştirme Ve Web Kütüphanelerin İncelenmesi

Adı SOYADI

Altan Akdar

İzmir Katip Çelebi Üniversitesi

Fen Bilimler Enstitüsü

Yazılım Mühendisliği Anabilim Dalı

Bu çalışmada Dünya'da ve Türkiye'de web ortaya çıkışı yaygınlaşması ve web çalışma mantığı açıklanmıştır daha sora ise dünya üzerinde web uygulamaları yapılırken kullanılan web geliştirme dilleri ile beraber bu dillerin yazımını kolaylaştıran daha dinamik sayfalar oluşturmamızı sağlayan web kütüphanelerinin ne olduğu ile ilgili olarak bilgi verilmiştir.

Uygulama kısmımızda c# yazılım dili ile birlikte Entityframework kütüphanesi ile kurumsal mimari projesi uygulaması yapılmıştır. Bu mimari profesyonel web sitesi backend kısmını oluştururken kod okunurluğunu büyük ölçüde kolaylaştırmaktadır. Buda başka geliştiricilerin kodumuzu okumasını kolaylaştırır bir nevi düzenli kod yazma tekniği diyebiliriz.

İÇİNDEKİLER

ÜÇÜNCÜ BÖLÜM

C# ASP.NET 6.0 WEB BACKEND PROJESİ

3.0 WEB PROJE UYGULAMASINDA YAPILACAKLAR

BİRİNCİ BÖLÜM

DÜNYA'DA VE TÜRKİYE'DE İNTERNETİN GELİŞİMİ VE WEB SİTELERİNİN DOĞUŞU

1.1 DÜNYA'DA İNTERNET GELİŞİMİ

İnternet 1950 yıllarında bilgisayarların gelişmesiyle ortaya çıkmıştır. Ağ tasarımları ilk olarak ABD, İngiltere ve Fransa'daki çalışmalarla yavaş yavaş şekillenmiştir.

1960'lı yılların başında ise ABD savunma bakanlığının desteklediği ağ çalışmalarından birisi olan ve internet protokolünü(IP) ilk kullanan Arpa Net'tir. İlk mesajını ise Los Angeles Kaliforniya Üniversitesi laboratuvarından Stanford araştırma merkezindeki bilgisayara göndermişti.

1960'ların sonlarında ve 1970 başlarında birçok haberleşme protokolü kullanılarak ağlar geliştirildi. Bu ağla Arpa Net, NPL network, CYCLADES, Merit Network, Tym Net, Tele Net diyebiliriz. İngiltere ulusal fizik laboratuvarında dünyada kendi sınıfında bir ilk olan paket anahtarlamalı bir ağ kurup ağ teorisini pratiğe dönüştüren ilk kişi ise Donald Davies'tir.[1][2]

Farklı ağların daha büyük ağlara bağlanmasını da Arpa Net mümkün kılmıştır. İnternet gelişiminin böylece önünü açmış oldu. Arpa Net 1981 yılında Computer Science Network ve National Science Foundation tarafından fonlanarak erişimi daha fazla genişledi. 1982 yılında TCP/IP, Arpa Netin standart ağ protokolğü olarak tanıtıldı. 1980 yılları başında NSF tarafından çeşitli üniversitelerde bilgisayar merkezlerinin kurulması desteklendi ve 1986 yılında NSFNET projesi ile bu merkezler birbirine bağlanmış oldu. 1980 yılları sonlarına doğru ticaret internet sağlayıcıları doğdu.

Tim Berners Lee, World Wide Web çalışmaları sayesinde zenginleştirilmiş text dökümanları ile çalışan bir ağ sistemi entegre etme konusunda ilerleme kaydetti.[3] Modern internet bu şekilde ortaya çıkmaya başladı.

1990'da Arpa Net, 1995'te NSFNET miadını doldurdu internete veri taşıma amaçlı erişimin önündeki engelde kalkıyordu 1990 yıllarının ortalarından itibaren anlık mesajlaşmalar, e-posta, VoIP video görüşmeler sosyal ağ ve online alışveriş siteleriyle gelen World wide web kültürümüzde ve toplumumuzda bir devrim yaratmış oldu.[4]

1.2 TÜRKİYE'DE İNTERNET GELİŞİMİ

Türkiye'de internetin gelişimi 1987 yıllarına dayanmaktadır. İlk adımı Türkiye Üniversite ve Araştırma Kurumları ağı olan (TÜVAKA) atmıştır.[5]
Bankalarda 1980'li yıllarda şubeler arasında iletişim kurabilmek amacıyla bilgisayar ağları kurmuşlardır bu ağlar sadece bankaların kullanımına sunulmuştur.[6] Türkiye'deki ilk internet bağlantısı ise 1993 yılında Türkiye ve Washington arasında oluşturulan ağ ile sağlanmıştır.
Böylece Türkiye NSF-NET (National Science Foundation Network)'e katılmıştır.[7] Türkiye'de internet bağlantısının olduğu ilk üniversiteler ODTÜ, Ege, İTÜ, Bilkent ve Boğaziçi'dir.[8] 1993 yılında başlayan Türkiye İnternet Proje Grubu (TR-NET) ve daha sonra 1996 yılında başlayan Türkiye Ulusal İnternet altyapı ağı (TURNET) projeleriyle

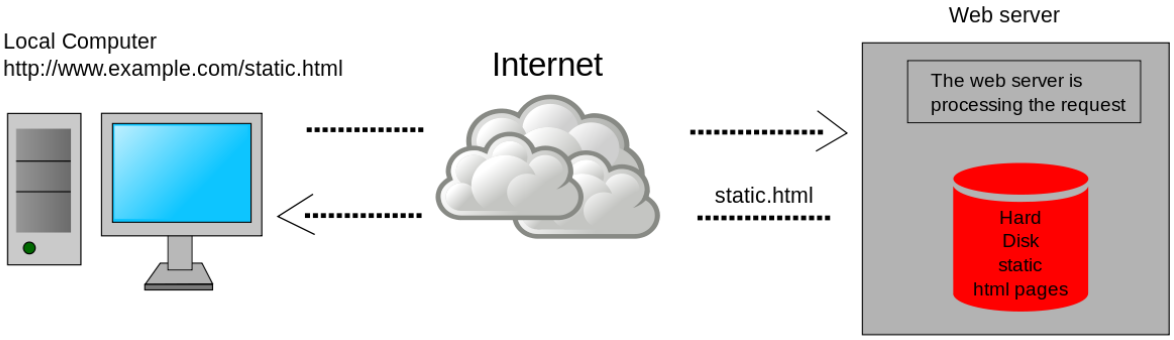
beraber Türkiye'de internet kullanımı çeşitli sektörlerdeki kurum ve kuruluşlarda değişik şehirlerde yaygınlaşması planlanmıştır.[9]

1.3 WEB SAYFASI NEDİR

Bir Web sayfası yaygın olarak html'de yazılmış olan, internet sağlayıcısını kullanan ve diğer ağlar tarafından da ulaşılabilen bir belge olarak adlandırabiliriz. Web sayfasına URL adresi girilerek ulaşabiliriz. Web sayfaları, diğer web sayfalarına ve dosyalarında metin, grafik ve köprüler içerebilir. Web sayfalarına ulaşabilmek için Chrome, Firefox, Edge ve Safari gibi tarayıcılar gerekebilir.

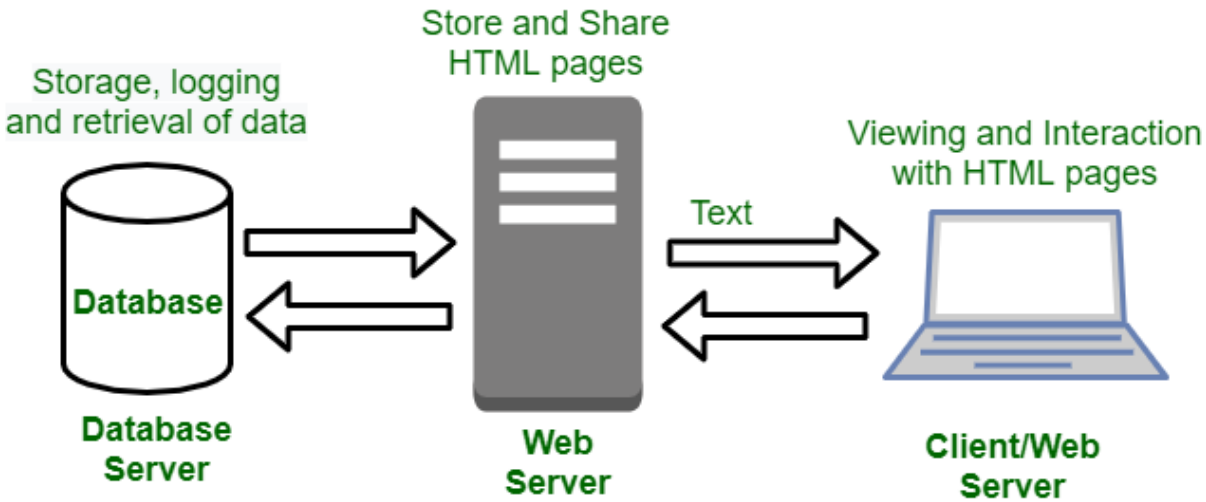
1.4 WEB SİTESİ VE WEB SAYFASI

Web siteleri bir domain yani alan adını paylaşan herkes tarafından erişilebilir ve web sayfalarını birbirine bağlayan koleksiyondur. Web siteleri çeşitli amaçlara hizmet etmek için bir kişi, grup, işletme ve kuruluşlar tarafından oluşturulur.
Web siteleri çok çeşitli olarak karşımıza çıkar. Eğitim siteleri, haber siteleri, bilim siteleri, forumlar, sosyal medya siteleri, e ticaret siteleri vb. yüzlerce internet siteleri mevcuttur.
Web sayfası, internet bağlantılı adreslerden, dijital içeriklerden, işaretleme ve programlama kod parçacıklarından oluşur. Sayfalarda kullanılan Html kodları bir dizi protokollerden meydana gelir. Bu protokollerle birlikte web sayfaları kullanıcılar tarafından, bilgisayar, tablet, cep telefonu gibi internete bağlanan donanımlar ile birlikte web tarayıcılarında (browser) yardımıyla görüntülenebilir. "Web sayfalarında görüntü ve yazının organizasyonu bilgiden önce kullanıcının dikkatini çekmekte ve web sayfalarıyla etkileşimi daha kolay hale getirmektedir. "[10] Web sayfaları statik ve dinamik olmak üzere iki boyutta ele alınabilir. Statik web sayfaları genel olarak standart işaretleme diliyle yazılmış bilgi verme amaçlı web sayfalarıdır. İnteraktif bir iletişimi yoktur.
Web sayfası içerisindeki dökümanları sürekli güncellemenin gerek olmadığı karşılıklı ziyaretçilerin etkileşimde olmadığı tanıtım ve bilgi amaçlı oluşturulmuş web siteleridir.
Statik içerikli bir web sayfası kısacası, ziyaretçini içeriği görüntüleyeceği sırayı seçmekten başka, sitenin içeriğiyle etkileşime girme yeteneği minimum düzeydedir. Statik bir siteye girmek basılı bir dergiyi okumak gibidir. Kullanıcı sayfalar arasında ileri geri çevirmeyi ve makaleleri farklı bir sırada okumayı seçebilir. Fakat sunum nispeten katılımsız olabilir. Ekranda okumak, kağıtta okumak için yazdırmak veya başka bir yerde kullanmak üzere içerik parçalarını kopyalamak dışında, statik bir sitenin içeriğiyle ilgili kullanıcı açısından pek bir şey yapılamaz[11]



Şekil 1. Statik web sayfa sunucu örneği. Static web page (t.y.) [Resim]. https://upload.wikimedia.org/wikipedia/commons/thumb/5/57/Scheme_static_page_en.svg /1200px-Scheme_static_page_en.svg.png sayfasından erişilmiştir

Dinamik web sayfaları ise php, asp.net, java, python gibi programlama dilleriyle yazılmış ve bir veri tabanı olan web siteleridir. Bu bilgiler veri tabanında saklanır ve istenildiği zaman bu bilgiler üzerinde değişiklik yapılabilir.[12] Sitenin kullanıcıları sitedeki içerikle veya sitenin diğer kullanıcıları ile birlikte doğrudan iletişime girebilir. Bu iletişim (üye olma, yorum yazma, beğenme) şeklinde olabilir. Kişileştirilmiş bilgiler sunabilirler. İçerikleri istenildiği zaman site yöneticileri veya editörler tarafından kodsuz bir şekilde değiştirilebilir. Kendine özgü yönetim panelleri bulunmaktadır. İçerik eklemek veya düzeltmek için kod bilgisine gerek yoktur. İşlevselliği ve güncellemesi ise gayet kolaydır.[13]

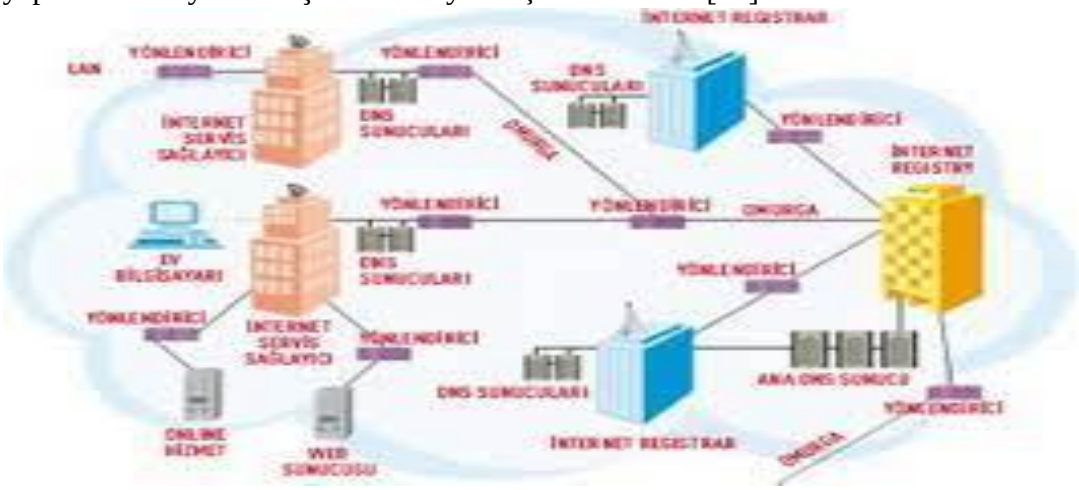


Şekil2: Dinamik site çalışma mantığı örneği Resim https://www.geeksforgeeks.org/dynamic-websites/

Web siteleri üst düzey alan adlarına göre kategorilere ayrılmışlardır.
Bazı örnekleri ise;
Devlet kurumları web siteleri = .gov
Eğitim kurumları web siteleri = .edu
Kar amacı gütmeyen kuruluş web siteleri = .org
Ticari web siteleri = .com
Bilgi siteleri = .info

1.5 Www (World Wide Web)

World Wide Web (Dünya çapında ağ), arka planda işlenen protokollere dayanan kod dizelerinden oluşmaktadır. Web' de internet yolu ile birçok bilgisayar birbirine bağlanmaktadır. Bu bağlantı ile milyarlarca web sitesi arasında iletişim sağlanır.[14]Bu bağlantı ile beraber örümcek ağına benzeyen çok karmaşık bir yapı ortaya çıkar.[15] Web sayfalarında resim, yazı, ses ve hareketli görüntüleri destekleyen birden fazla dosya türü bulundura bilinir. Bu dosya türlerini aynı anda veya birden fazla sayfada yayınlamak mümkündür.[16] Www sayesinde ücretsiz olarak gazete ve dergi rahatlıkla okuyabilmekte, çeşitli bilgi bankalarına ve kütüphanelere rahat bir şekilde erişilebilmektedir. Bu durum araştırma yapanlar için zaman ve para açısından büyük tasarruf sağlar. Akademik çalışma yapanlar bu sayede bir çok makaleye ulaşabilmektedir.[17]



Şekil 3: https://www.chip.com.tr/haber/internet-nasil-calisir_1809.html

İKİNCİ BÖLÜM

WEB GELİŞTİRME ARAÇLARI VE WEB KÜTÜPHANELERİN İNCELENMESİ

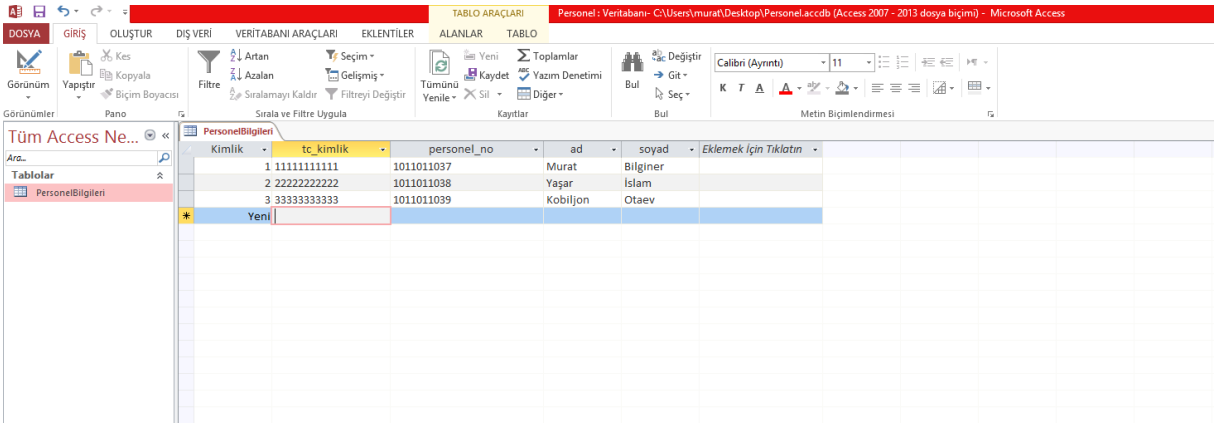
2. WEB GELİŞTİRME BACKEND ARAÇLARI

Web geliştirmesi yapılırken bir istemci ve sunucu tarafı vardır. Web geliştirmede backend tarafı sunucu tarafını geliştirmektir.
Sunucu tarafını geliştirirken yapmamız gerekenler ise;

1. Sunucu tarafı mantığı: Kullanıcının hesap bilgilerinin doğru olduğunu kontrol etmek, ve hangi bölümleri görmesine izin verileceğini kararlaştırmaktadır. Bir web sayfasındaki görevin işleme alındığından emin olmak ve bir hata çıkmamasını sağlamaktadır. Ayrıca web sayfasının her bir işleminin kesintisizce çalıştığından emin olmak ve aynı zamanda en hızlı şekilde işlediğini görmesi gerekmektedir.

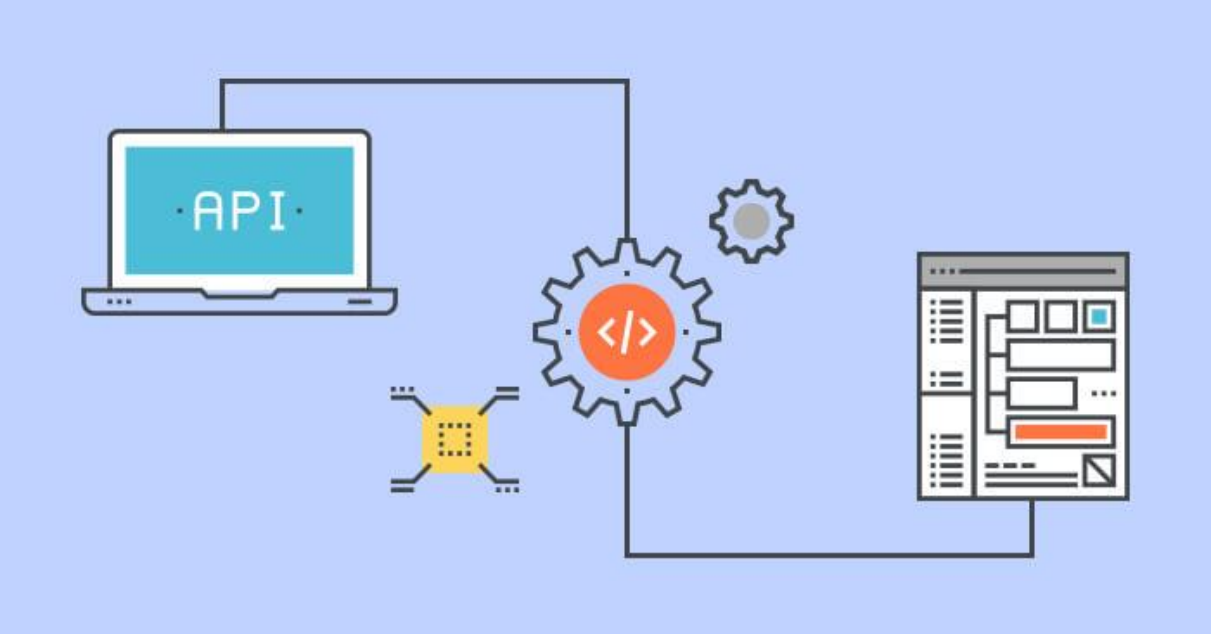
2. Otomatik bildirimler: Örnek vermemiz gerekirse olursak internet sitesindeki adres doğrulamaların otomatik olarak gerçekleştirilmesi gerekir diyebiliriz.

3. Veri tabanı erişimi: Bir web sitesi için tutulan çeşitli verilerin erişilmesi gerekmektedir. Veri tabanındaki bilgiler programlama dilindeki frameworkler yardımıyla yazılan kodlar ile çekilebilir. Veri tabanları yapılandırılmış bilgi veya verilerin depolandığı alanlardır. Bilgi artışı ile birlikte bilgisayarlarda bilgi depolama ve bilgiye erişim konularında yeni yöntemlere ihtiyaç duyulmuştur. Veri tabanları büyük miktardaki verileri depolamada geleneksel yöntem olan dosya işleme sistemine alternatif olarak geliştirilmiştir. Telefonlarda bulunan kişi rehberi veri tabanlarına küçük bir örnek olarak gösterebiliriz. İnternet sitelerinde bulunan üyelik sistemleri, akademik dergilerin ve üniversitelerin tez yönetim sistemleri de veri tabanı kullanımına örnek gösterebiliriz. [18]



Şekil 4: http://muratbilginerncfkr.blogspot.com/2014/04/51-c-gorsel-programlama-51-c-da-access.html

4. Api'ler: Api karmaşıklığı yazılım geliştiriciden gizleyen, kodları düzenleyen ve bileşenleri yeniden kullanılır hale getiren bir mekanizmadır. (Application Programming Interface) Başka bir değişle apileri bir yazılımın başka bir yazılımda tanımlanmış işlevlerini kullanabilmesi için oluşturulmuş günümüzde ise çoğunlukla ile web yazılımında kullanılan istemci ve sunucu arasındaki iletişimi sağlayan bir sözleşme diyebiliriz. İstemci tarafı spesifik bir biçimde veri talep eder ve sunucudan belirli formatta cevap alır. Bu durum Web Apı olarakta adlandırılır.[19]



Şekil 4: Argenova.com.tr/api-nedir

2.1 JAVA VE HİBERNİTE FRAMEWORK KÜTÜPHANESİ

Java Programlama dili günümüzde yaygın olarak kullanılır ve oldukça esnektir. Yüksek performanslı, çok işlevli, adım adım işleten bir dildir.[20] Java, sun firması tarafından geliştirmeye başlanmıştı. İlk sürümü 1995 yılında ortaya çıkmıştır.

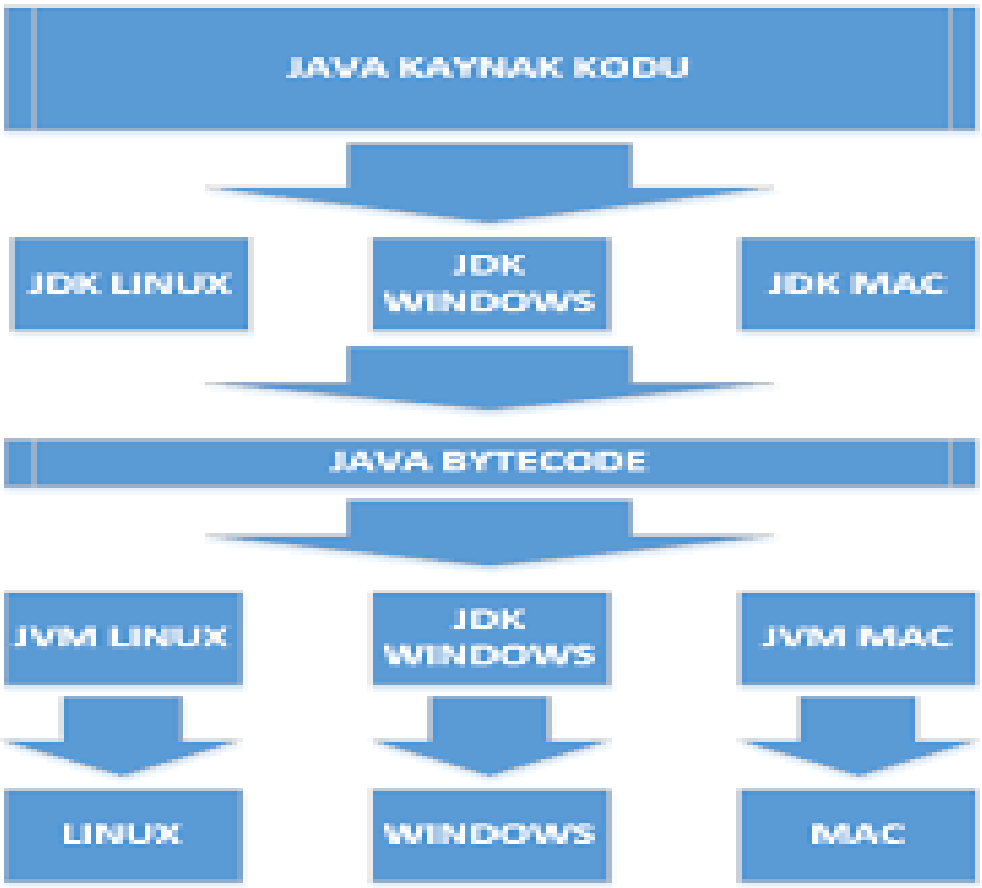
Platform bağımsız çalışması, açık kaynak kodlu olması ve nesne yönelimli olması sebebiyle kullanım alanları çok fazladır.[21]

Java dili bilgisayar ağaları üzerinde farklı bilgisayar platformlarında kullanılacak yazılımları geliştirmek amacıyla da geliştirilmiştir.[22]

Cep telefonlarındaki uygulamalarda, bilgisayarlarda, yazıcılarda, dijital televizyonlarda kullanılmaktadır. Yeni üretilen programlanabilir birçok cihazda yazılım dili olarak java tercih edilmektedir.[23]

Şekil 5: https://www.dijitalders.com/icerik/44/343/java_terimleri.html

Java programlama dilini öne çıkaran unsur ise "Bir kere Java kodu yaz her yerde kullan " felsefesidir. Java işletim sistemi gözetmeden herhangi framework kullanmadan notpad'de yazılarak dahi çalıştırılır. Java Programlama dili çalışma mantığı aşağıdaki şekildeki gibidir.

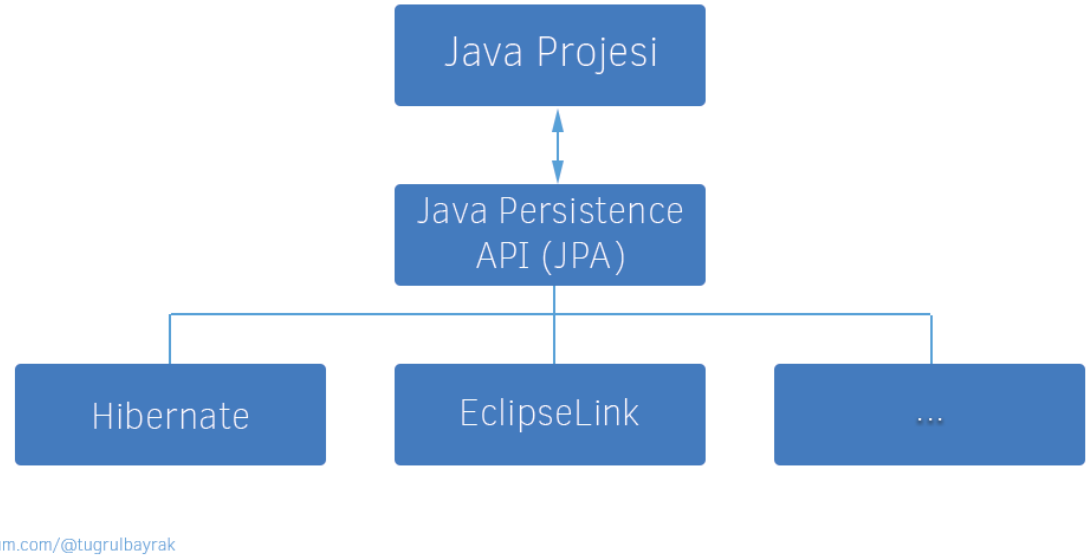


Şekil 6: https://acikbilim.yok.gov.tr/bitstream/handle/20.500.12812/709572/yokAcikBilim

Ayrıca Java dili birçok teknoloji ile birlikte uyum içinde çalışmaktadır. Yazılımcılar tarafından oluşturulan yazılım projelerinin esnek ve aynı zamanda dinamik olması yazılımcıların işini kolaylaştıran Java ile birlikte kullanılan bir takım araçlar vardır.
Bunların en önemlileri; Yazılan kodların kalitesini analiz eden (SonarQube, Fisheye), Java sanal makinesini ayarlayan (Eclipse Memory Analizer), izleme performans (Tuning & Monitoring) araçları, Java ideleri (Eclipse, NetBeans, IntelliJ), Java web programlama araçları ve dilleri (JSP, JSF) ve Java ver tabanı işleri için kullanılan (Jdbc, Hibernate, NoSql) araçlarıdır.

2.1.1 HİBERNATE

Hibernate bir ORM aracı olmasının yanında veri tabanı işlerinde de kullanılır. Veri sorgulama veri çekme işlemlerini sağladığı gibi Java kodunun üzerinden veri tabanında tablolar oluşturma, tablolar arasındaki ilişkiyi oluşturma İşlemlerini de yapabiliriz. Ayrıca Hibernate' nin bir sorgulama dili olan HQL ile birçok işlem imkanını yazılım geliştiricilere sunmaktadır.
Nesneye yönelik tasarım ve ilişkisel veri tabanı kulanımı günümüz yazılım geliştirmede oldukça yaygındır. Bu iki gözde modelin en önemli problemi ise en az onlar kadar yaygın olan ve dahası önü açık olan kuruluş uygulamaları (enterprise applications) ile birlikte kullanıldıklarında oldukça karışık, yorucu ve zaman alıcı olmalarıdır. Hibernate bir nesne ve ilişkisel eşleme yani Object Relational Mapping aracıdır. Burada nesne ilşkisel eşleme terimi ise nesne modelindeki veri tanımlarının ilişkisel veri modeline eşleme yani mapping tekniğini ifade etmektedir. Hibernate teknolojisi sadece Java sınıflarından veri tablolarını veya Java veri tiplarinde SQL vei tiplerine dönüşümü yapmaz. Hibernate veri sorgulama (data query) ve veri çekme (data retrival) işlemlerinide kullanıcı için sağlamaktadır. Tüm bu özelliklerle beraber Hibernate geliştirme kolaylığı ve zamandan kazanç sağlamaktadır. Hibernate teknolojisi olmadan SQL ve JDBC olanaklarından faydalanılarak manuel veri işleme çok zor olacaktır.[24]



Şekil 7: https://miro.medium.com/max/1400/1*NewjznGvudKRZBBftOV7cQ.png

2.2 PYTHON DJANGO VE FLASK KÜTÜPHANESİ

Python 1980'lerin sonlarında ABC programlama diline alternatif olarak tasarlanmıştı. Python 2.0 ilk kez 2000 yılında yayınlandı. 2008 yılında yayınlanan Python 3.0 önceki versiyonla çok uyumlu değildi ve Python 2.0'da yazılan kodların Python 3.0 da çalıştırılabilmesi için değiştirilmesi gerekmekteydi. Python 2.0 geliştirme süreci, dilin son sürümü olan Python 2.7.0 seri versiyonu ardından 1 ocak 2020 itibari ile sona erdi.[25]
Python ismi dilin yaratıcısı olan Guido Van Rossom'un programlama dilini geliştirirken keyif aldığı İngiliz komedi grubu olan Monty Python'dan gelmektedir. Monty Python'a ait olan birçok atıf kod yapısında sıklıkla görülebilir.[26] 2003 yılı itibari ile Python TIOBE programlama topluluğu endeksine göre en popüler 10 programlama dili arasında yer almaktadır. Ekim 2021 itibariyle ise Java ve C programlama dillerini geçerek en popüler programlama dili olmuştur.[27] Python nesne yönelimli, yorumlamalı, birimsel ve etkileşimli yüksek seviyeli bir programlama dilidir.[28] Girintilere dayalı basit söz dizimi, dilin öğrenilmesini ve akılda kalmasını oldukça kolaylaştırmaktadır.

```python
# python_functions.py
# ------------------------
import math

def f(x):
    return math.exp(-(x ** 2))

def integrate_f(a, b, N):
    s = 0
    dx = (b - a) / N
    for i in range(N):
        s += f(a + i * dx)
    return s * dx
```

```python
# cython_functions.pyx
# ------------------------
from libc cimport math

cdef double f(double x):
    return math.exp(-(x ** 2))

def integrate_f(double a, double b, int N):
    cdef double s = 0
    cdef double dx = (b - a) / N
    cdef int i
    for i in range(N):
        s += f(a + i * dx)
    return s * dx
```

Şekil 8: https://www.analiyz.com/wp-content/uploads/2019/05/Python-Fonksiyonlar.png

Python modüler yapısı, sınıf sistemini ve her türlü veri alanı girişini desteklemektedir. Hemen hemen her türlü platformda da çalışabilmektedir(Unix, Linux, Mac, Windows, Amiga, Symbian). Python programlama dili ile kullanıcı arabirimi programlama, ağ programlama, web programlama, uygulama, veri tabanı yazılımı, veri bilimi, yapay zeka gibi birçok alanda yazılım geliştirilir. [29]

2.2.1 DJANGO

Python programlama dili için hazırlanan ve BSB lisansı ile lisanslanmış yüksek seviyeli bir web çatısıdır. Basit kurulumu ve kullanımı, detaylı hata raporu sayfaları ve sunduğu ara yüz kodlama yöntemleri ile diğer sunucu yazılımı ve çatılardan kendini ayırmaktadır.
Django ile hızlı ve etkili web siteleri oluşturabiliriz.Yönetici arayüzü. Varsayılan veri tabanı SQLite gibi özellikleri kullanıcılara sağlar. Bir web sitesi oluştururken benzer özellikler kullanırız. Fakat authentication, yönetim paneli, dosya işleme yolu gibi özellikleri django kullanıcılara otomatik olarak sağlamaktadır.
Django'yu Mükemmel dökümantasyon ve yüksek ölçeklenebilir, Popüler şirketler kullandığı için örnek verecek olursak (Instagram, Spotify, Youtube gibi), öğrenmesi kolay olduğu için, kısa sürede tam anlamıyla web uygulamaları geliştirebilmek için, tercih edebiliriz.

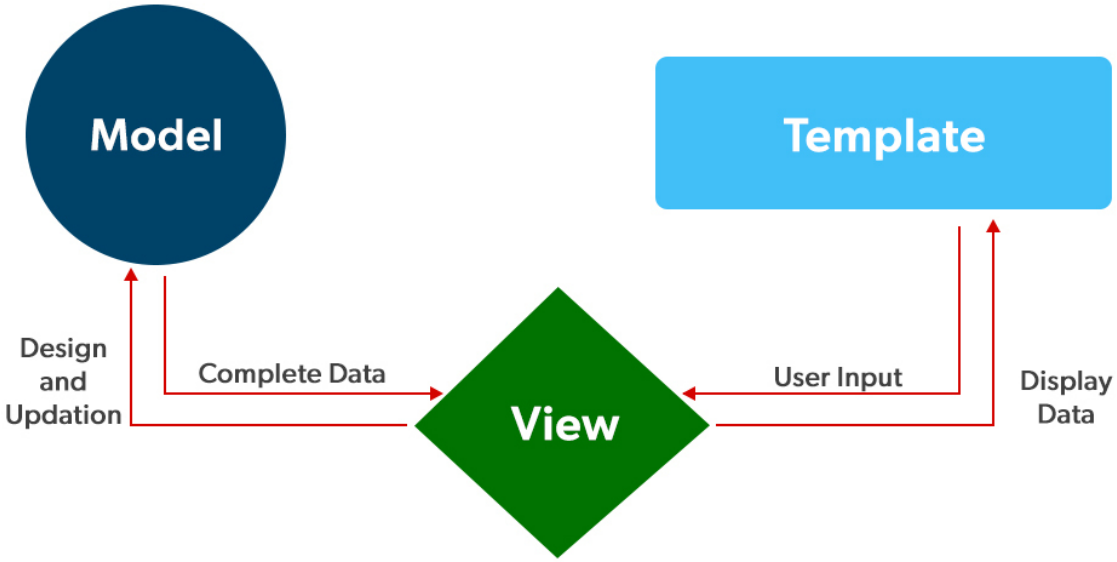
2.2.2 DJANGO MİMARİSİ

Django üç bölümden oluşan MODEL-VİEW-TEMPLATE mimarisine dayanmaktadır.

MODEL: Model, verilerin ara yüz görevi görür ve verilerin korunmasından sorumludur. Uygulamanın arkadaşında yatan mantık veri yapısı ve verilerin saklanılmasında yatmaktadır.

VİEW: View bir web sitesi oluşturduğunuzda tarayıcıda görünen arayüzdür. Html, css, Javascript ve Jinja dosyaları ile temsil edilir.

TEMPLATE: Template istenen html çıktısının statik bölümlerinden ve dinamik içeriğinin nasıl ekleneceğini açıklayan bazı söz dizelerinden oluşur.[30]



Şekil 9: Geeksforgeeks.org/python-web-development-django-tutorial

2.3 JAVASCRİPT VE WEB KÜTÜPHANESİ

Javascript Html ve Css ile birlikte World Wide Web'in temel teknolojilerinden biri olan programlama dilidir.[31] Web sitelerinin %97'sinden fazlası, web sayfaları istamci tarafında javascript kullanılmaktadır.[32] Kullanılan kodlar genellikle üçüncü taraf kitaplıkları içerir. Tüm büyük web tarayıcılarında, kaynak kodunu kullanıcıların cihazlarında yürütüldüğü özel javascript motoru bulunmaktadır.
Javascript Ecma script standartlarına tam olarak uyan, çoğu zaman eşzamanlı olarak derlenen üst dizey dil olarak karşımıza çıkmaktadır.[33] Dinamik yazma, prototip tabanlı obje yönelimi ve birinci sınıf işlevleri bulunmaktadır. Olay odaklı işlevsel ve zorunlu programlama stillerini destekleyen çoklu paradigmadır. Metin, tarihler, düzenli ifadeler, standart veri yapıları ve belge obje modeli (DOM) ile çalışmak için uygulama programlama ara yüzüne (APİ) sahiptir.

Ecmascript standardı, ağ oluşturma, depolama veya grafik olanakları gibi herhangi bir giriş çıkış (IO)içermez. Web tarayıcısı veya diğer çalıştırma ortamları, I/O için javascript API' leri sağlarlar. Javascript başlangıç olarak web tarayıcısı olarak kullanılıyordu fakat günümüzde farklı sunucu ve uygulamaların temel birleşenini oluşturmaktadır. En popüler çalışma zamanı sistemi Nodejs'dir.

```
<html>
  <head>
     <div>
            <div>

                <form method="post" action="#" id="formvalue" onkeyup="
                drawChart()" />

                </form>
            </div>
     </div>



     <script type="text/javascript" src="https://www.google.com/jsapi"></
       script>
     <script type="text/javascript">

     var bid = 43;
     var ask = 21;


       google.load("visualization", "1", {packages:["corechart"]});
       google.setOnLoadCallback(drawChart);
       function drawChart() {
         var data = google.visualization.arrayToDataTable([
           ['Price', 'Quantity'],
           ['Value #1', bid],
           ['Value #2', ask],
         ]);
```

Java script kaynak kodu ekran görüntüsü.

Şekil 10: https://tr.wikipedia.org/wiki/JavaScript#/media/Dosya:JavaScript_screenshot.png

2.3.1 JQUERY

JQuery, John Resign tarafından 2006 yılında geliştirilmiş ve geniş bir jQuery ekibi tarafından gelişimi sürdürülen açık kaynak JavaScript kütüphanesidir.
Bu kütüphane en popüler çarpra platform Java Script kütüphanesi diyebiliriz. Yoğun plarak animasyonlarda kullanılır. Flash'ın alternatifi olarak kullanılan bu bu teknolojiyle Flash galeri, sekme menü, sayfa geçişleri gibi birçok işlem yapılmaktadır. Boyutu küçük ve işlevi çok olduğundan epey yaygın olarak kullanılır.
JQuery kütüphanesi iki şekilde kullanılmaktadır. İlk yöntem kendi sitesiden indirim çalışma alanına eklememiz gerekmektedir.
Bunun için kod satırını head tagları arasında kullanmamız yeterlidir.

<script src="jquery.js"></script>

İkinci yöntem JQuery kütüphanesini indirmeye gerek kalmadan direk CDN ile projemize entegre etmektedir.

<script src=https://code.jquery.com/jquery-latest.min.js></script>

2.3.2 ANGULAR

Angular google' daki Angular ekibi, bireyler ve şirketler topluluğu tarafından yönetilen TypeScript tabanlı özgür ve açık kaynaklı bir web uygulaması çerçevesidir. AngularJs' yi oluşturan aynı ekip tarafından oluşturulmuştur.[34]
Kısacası dinamik web uygulamaları için yapısal bir çerçevedir. HTML bir şablon dili olarak kullanmanıza ve HTML etiketlerini uygulama bileşenlerini açıkça temsil edecek şekilde genişletilmesine olanak sağlamaktadır. Temel amaç olarak, tek bir site üzerinde dinamik olarak gelişme imkanı sağlamaktadır. Verileri değiştirmek görünümü değiştirir görünüm değişince de veriler değişmektedir. Peki neden Angular kullanmayı tercih etmeliyiz. Angular kullanımı sayesinde tek sayfa uygulaması. Javascript'in yeni bir HTML sayfası yüklemek yerine geçerli sayfanın değiştirilmiş olan sayfasının DOM öğesini değiştirerek yeni bir sayfanın içeriğini dinamik olarak oluşturduğu bir web sitesi tasarımı yaklaşımıdır.[35]
Angular'ı popüler yapan avantajlarına bakacak olursak,

İki Yönlü Veri Bağlama: AngularJs mimarisi JavaScript ile HTML' i bağladığı için her ikisi senkronizedir. Böylece çerçeve web geliştiricileri için oldukça zaman kazandırır.

Yönergeler: Çerçeve HTML dosyalarının işlevlerini yönergelerle genişletir. Wb tasarımcıları yönergeleri etkinleştirmek için HTML özelliklerine ng önekini eklerler. Örnek olarak aşağıda görebiliriz.

```
  <!DOCTYPE html>
  <html>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.j
s"></script>
  <body>

  <div ng-app="" ng-init="age='20'">

  <p>Input your age:</p>
  <p>Age: <input type="text" ng-model="age"></p>
  <p>You wrote: {{ age }}</p>

  </div>

  </body>
  </html>

  <!DOCTYPE html>
```

```
  <html>
  <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.j
s"></script>
  <body>

  <div ng-app="" ng-init="age='20'">

  <p>Input your age:</p>
  <p>Age: <input type="text" ng-model="age"></p>
  <p>You wrote: {{ age }}</p>

  </div>

  </body>
  </html>
```

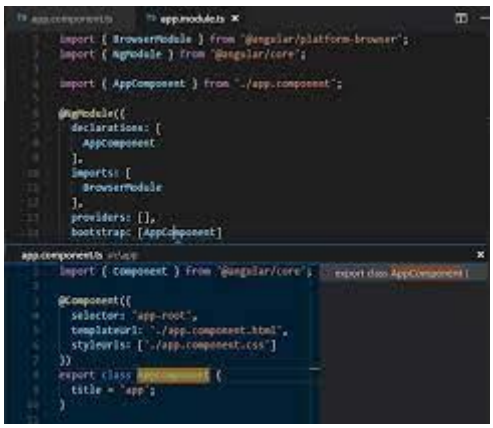
Kod Yapısı: Angular şablonlar sağlar temiz kodlu uygulamalar üretilmesine izin verir. Yalnızca zamandan tasarruf etmemize olanak sağlamaz aynı zamanda uygulamamızı değiştirmeyi ve düzenlemeyi sağlar.

Test Etme: Çerçeve, birim ve entegrasyon testini desteklemektedir.

Parlak Gelecek: Angular geleceği, işlevselliği ve popülerliği nedeniyle oldukça parlaktır. Kullanıcı tabanı büyümeye devam ediyor ve sürekli ve güncellenen çok sayıda belgeye sahiptir.

Mobil ve Masaüstü Uyumluluğu: Angular çoğu web tarayıcısında çalışabilir.[36]

Angular kod örneğini aşağıda görebiliriz



Şekil 11: https://www.bayramucuncu.com/category/angular/

2.3.3 REACT

React açık kaynaklı, kullanıcılara ara yüz yaratmak için oluşturulmuş olan JavaScript kütüphanesidir. Bu ara yüzlerin yaratılması için yeniden kullanılması gereken bileşenler, component isimli yapılar tarafından kullanılırlar.

React ortaya çıkışı 2011 döneminde Facebook üzerinden XHP isimli bir JavaScript portu yayınlanmıştır. Bu port PHP içine yerleştirilmiştir. XHP bileşeni kullanıcı tarafından gelen bilgileri temizleme sorumluluğunu PHP' den alan bir yapıya sahiptir. Bu gelişmelerle beraber Facebook inanılmaz derecede yükselişe geçmiştir. Bu gelişimle beraber daha kontrollü bir yapı ihtiyacı doğmuş ve facebook çalışanı olan Jordan Walke, XHP' den esinlenerek FaxJs ismi ile İlk React versiyonunu çıkarmıştır.

React' ın nasıl çalıştığına gelecek olursak HTML kodlarının JavaScript' de yazıldığı gibi sistemin mantığı aynıdır. Bir element fonksiyonu oluşturulması ile çalışmaya başlar. Örnek verecek olursak:

React.createElement("div",{className:"red"}, "Children Text");

React.createElement(MyCounter,{count:3+5});

HTML üzerindeki söz dizimi, XML bileşenleri ile çok benzer özellikler göstermektedir. Fakat genel olarak kullanılan DOM sınıfı yerine React kullanılması daha doğru olmaktadır.

React' ın öne çıkan özellikleri:

Öğrenilmesi Kolaydır, yeniden kullanılabilir bir sistem özelliği olması nedeni ile yaratılan sistemi diğer sistemlerle de kullanabiliriz. Virtual DOM ile daha hızlı sonuç alınır. Geniş destekçisi vardır ve devamlı olarak gelişmektedir.[37]

React kod örneğini aşağıda görebiliriz.

```
class Mouse extends React.Component {
  constructor(props) {
    super(props);
    this.handleMouseMove = this.handleMouseMove.bind(this);
    this.state = { x: 0, y: 0 };
  }

  handleMouseMove(event) {
    this.setState({
      x: event.clientX,
      y: event.clientY
    });
  }

  render() {
    return (
      <div style={{ height: '100vh' }} onMouseMove={this.handleMouseMove}>

        {/*
          Instead of providing a static representation of what <Mouse> renders,
          use the `render` prop to dynamically determine what to render.
        */}
        {this.props.render(this.state)}
      </div>
    );
  }
}
```

Şekil:12 https://medium.com/frontend-development-with-js/react-docs-3-code-sharing-947b4fdbba7e

2.4 C# ENTİTYFRAMEWORK

C# Microsoft tarafından .Net teknolojisi için geliştirilen modern bir programlama dilidir. Söz dizimi C programlama diline çok benzer. Microsoft tarafından geliştirilmiş olsa dahi ECMA ve ISO standartlarındadır.[39]

C programlama dilinde bir tam sayı değişkeni 1 arttırmak için değişkenden sonra "++" eki kullanılmaktadır. C++ dilinin adı ise C diliyle nesle yönelimli programlama yapabilme olanağı için eklentiler sağlar ve bundan dolayı C++ şeklindedir. Benzer şekilde C++ diline eklentiler yapılarak bir adım daha ileriye götürülmüş ve tamamen nesneye yönelik tasarlanmış olan C# programlama dili ortaya çıkmıştır. Bir çok alanda Java programlama dilini kendisine örnek alır. C# programlama dilinde .NET kütüphanelerini kullanmak amacıyla yazılan programların çalıştığı bilgisayarlarda uyumlu bir kütüphane ve yorumlayıcının bulunması gerekmektedir. Bu Microsoft kütüphanesi olacağı gibi ECMA standartlarına uygun farklı bir kütüphane ve yorumlayıcıda olabilir. Nesne yönelimli programlama kavramının gelişmesine katkıda bulunan aktif programlama dillerinden biridir.[40]

C# ve .NET orta seviyeli programlama dillerindendir. Hem makine hem de insan algısına yakın seviyededir. Orta ifadesinin anlamı dilin gücünü değil makine dili ve günlük konuşma diline olan mesafesini göstermektedir.

ECMA standartları tarafından C# tasarım hedefi şunlardır.[41]

- C#; basit, modern, genel amaçlı, nesneye yönelik programlama dili olarak tasarlanmıştır.
- Çünkü yazılımın sağlamlığı, güvenirliği ve programcıların üretkenliği önemlidir. C# yazılım dili güçlü tipleme kontrolü, dizin sınırlar kontrolü, tanımlanmamış değişkenlerin kullanım tespiti ve otomatik artık veri toplama gibi özelliklerine sahiptir.
- Programcı kullanım özelliğini C ve C++ dilleri ile tecrübesi olanlar için çok önemlidir.
- Enternasyonel hale koymak için verilen destek çok önemli olmaktadır.
- C# programlama dili sunucu ve gömülü sistemler için tasarlanmıştır. Bununla birlikte C# programlama dili en basit işlevsel fonksiyonlardan işletim sistemi kullanılan en teferruatlısına kadar kullanılmaktadır.
- C# uygulamaları hafıza ve işlemci gereksinimleri ile tutumlu olmak üzere tasarlanmıştır.

C# programlama dili Çalışma örneği

```
// Konsol uygulamaları yazılması için System isim uzayı eklenir.
// Bu sayede derleyici, System.dll'i kullanması gerektiğini bilir.
using System;

// Sınıf tanımlamasıdır.
class Program
{
    // .NET çalışma zamanında ön tanımlı olarak Main() fonksiyonunu
çalıştırır.
    static void Main()
    {
        // Console sınıfı içerisindeki Writeline() fonksiyonu
çalıştırılır
        Console.WriteLine("Merhaba Dünya!");
        // Kullanıcıdan herhangi bir tuşa basarak çıkması için bir tuş
okunur.
        Console.ReadKey();
    }
}
```

C# Basit bir hesap makinesi örneği

```
using System;

class Program
{
    double sayi1, sayi2, cevap;
    string islem;
    static void Main(string[] args)
    {
        Console.Write("Lütfen ilk tam sayıyı giriniz: ");
        sayi1 = Convert.ToDouble(Console.ReadLine());
        Console.Write("Lütfen yapacağınız işlemi giriniz (+, -, /, *):
");
        islem = Console.ReadLine();
        Console.Write("Lütfen ikinci tam sayıyı giriniz: ");
        sayi2 = Convert.ToDouble(Console.ReadLine());
        switch (islem)
        {
            case "-":
                cevap = sayi1 - sayi2;
                break;
            case "+":
                cevap = sayi1 + sayi2;
                break;
            case "/":
                cevap = sayi1 / sayi2;
                break;
            case "*":
                cevap = sayi1 * sayi2;
                break;
```

```
            default:
                cevap = 0;
                break;
        }
        Console.WriteLine(sayi1.ToString() + " " + islem + " " +
sayi2.ToString() + " = " + cevap.ToString());
        Console.ReadLine();
    }
}
```

2.4.1 ENTİTYFRAMEWORK

EntityFramework ORM (Object Relational Mapping) raçlarından biridir. ORM ilişkisel veri tabanı ile nesneye yönelik programlama arasında köprü vazifesi gören bir araçtır. Bu köprü ilişkisel veri tabanındaki bilgilerimizi yönetmek için nesne modellerimizi kullandığımız bir yapıdır. EntityFramework Ado.Net altyapısını kullanmaktadır. içerisinde ise UnitOfWork tasarım yapısını kullanır. UnitOfWork yapılan her değişikliği anlık olarak yansıtması yerine, bu işlemleri toplu halde tek bir kanalüzerinden yürütülmesinden sorumludur.

EntityFrameWork ile dört farklı yöntem kullanılarak proje geliştirilir.

1. Model First: Bu yöntem ide üzerinden boş bir model dosyası eklenerek veri tabanını bu model üzerinden tasarlanabilmesine olanak sağlıyor.
2. Database First: Bu yöntem önceden oluşturulmuş olan veri tabanını projeye model olarak bağlayarak gerekli sınıflarımız EntityFrameWork tarafından oluşturulur.
3. Code First(Kod yazarak): sınıflarımızı ide üzerinde oluşturmaya başlayarak gerçekleştirmiş olduğumuz bir yöntemdir. Veri tabanımız bu sınıflardan türetilir. Mapping tarafı yazılımcı tarafından sınıflar türetilirken Attributeler tarafından yapılabilmektedir. Attribute yanında farklı yöntemlerle de bu işlemleri gerçekleştirebiliriz. Fluent Api ve Fluent Validation gibi araçlar mapping işlemleri için popüler olarak kullanılmaktadır.
4. Code First(Var olan veri tabanını kullanma): Bu yöntemle sınıflarımız ve mapping kodları yazılımcı tarafından oluşturulur. Veri tabanımız sınıfların ve modellemenin durumuna göre güncellenir.

ÜÇÜNCÜ BÖLÜM

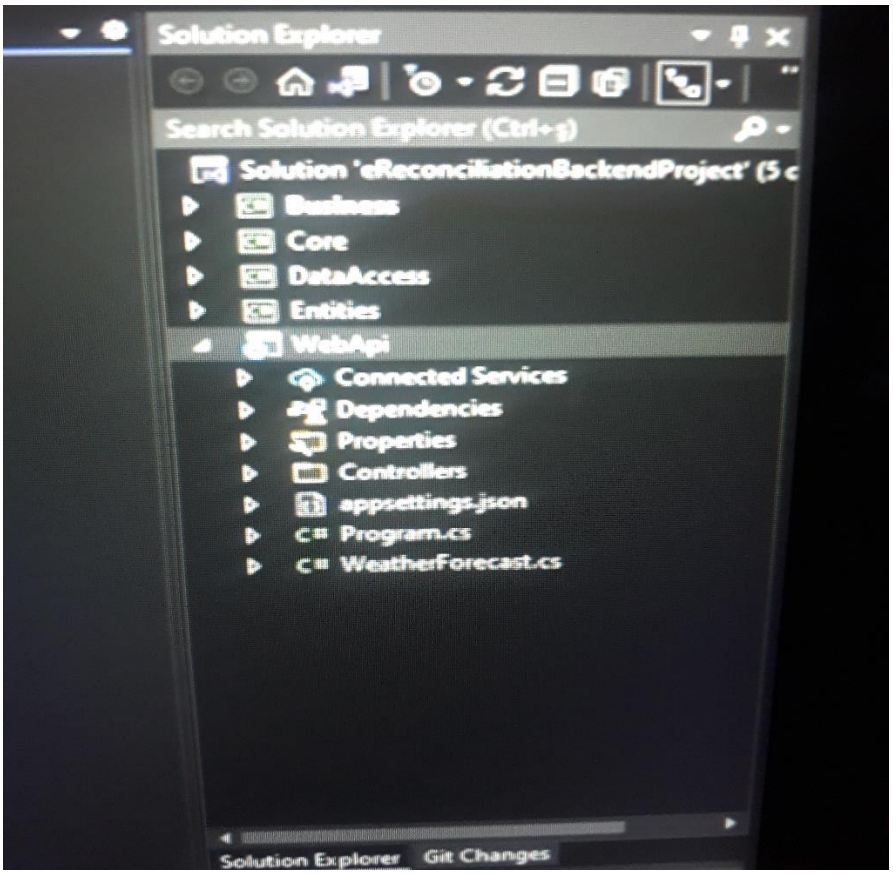
C# ASP.NET 6.0 WEB BACKEND WEB PROJESİ

3. WEB PROJE UYGULAMASINDA YAPILACAKLAR

Web backend projesi çalışması yapılacaktır. Çalışma yapılırken örnek kodlar üzerinden açıklamalı olarak kodlar eklenecektir.

3.1 PROJE KATMANLARININ OLUŞTURULMASI

Proje'nin Backend kısmı Visual Studio 2022 idesi ile oluşturulacaktır. Projeyi oluştururken ide üzerinden blank solution kısmı seçilir. İlk olarak class library seçilir ve ilk katman olarak business katmanı daha sonra core, Dataaccess, entities ve son katman olarak ise Asp Net Core Web Api katmanı seçilir.



Projelerde katmanlı mimari oluşturmanın amacı her bir katmanda kendi iş kodlarının yazılması yapılan projelerin sürdürülebilir olması açısından çok önemlidir.

3.1.2 ENTİTY KATMANININ KODLANMASI

Entities katmanı üzerinde 2 adet klasör oluşturulur bu klasörler concrete ve abstract klasörleridir. abstract soyut nesneleri tutar, concrete somut varlıklarımızı tutar. Fakat entity(varlık) katmanımızı oluştururken bu varlıkları referans tutması gereken imzasının olması gerekiyor bu imzayı ise core katmanımızda entity adında klasör oluşturarak bir interface şeklinde yazmamız gerekiyor. Oluşturmuş olduğumuz interface her bir entity öğemize referans olacaktır.

Artık somut sınıflarımızı sırasıyla yazmaya başlayabiliriz.

İlk sınıfımızın adı AccountReconciliation olacak

```
namespace Entities.Concrete
{
    public class AccountReconciliation : IEntity
    {
            public int Id {get; set;}
            public int CompanyId { get; set; }
            public int CurrencyAccountId { get; set; }
            public DateTime StartingDate { get; set; }
            public DateTime EndingDate { get; set; }
            public int CurrencyId { get; set; }
            public decimal CurrencyDebit { get; set; }
            public decimal CurrencyCredit { get; set; }
            public bool IsSendEmail { get; set; }
            public DateTime? SendEmailDate { get; set; }
            public bool? IsEmailRead { get; set; }
            public DateTime? EmailReadDate { get; set; }
            public bool? IsResultSucceed { get; set; }
            public DateTime? ResultDate { get; set; }
            public string? ResultNote { get; set; }
            public string? Guid { get; set; }
}
```

İkinci sınıfımızın ismi ise AccountReconcalitionDetail olacak.

```
namespace Entities Concrete
```

```
{
    public class AccountReconcalitionDetail : IEntity
{
    public int Id { get; set; }
    public int AccountReconciliationId { get; set; }
    public DateTime Date { get; set; }
    public string Description { get; set; }
    public int CurrencyId { get; set; }
    public decimal CurrencyDebit { get; set; }
    public decimal CurrencyCredit { get; set; }
}
}
```

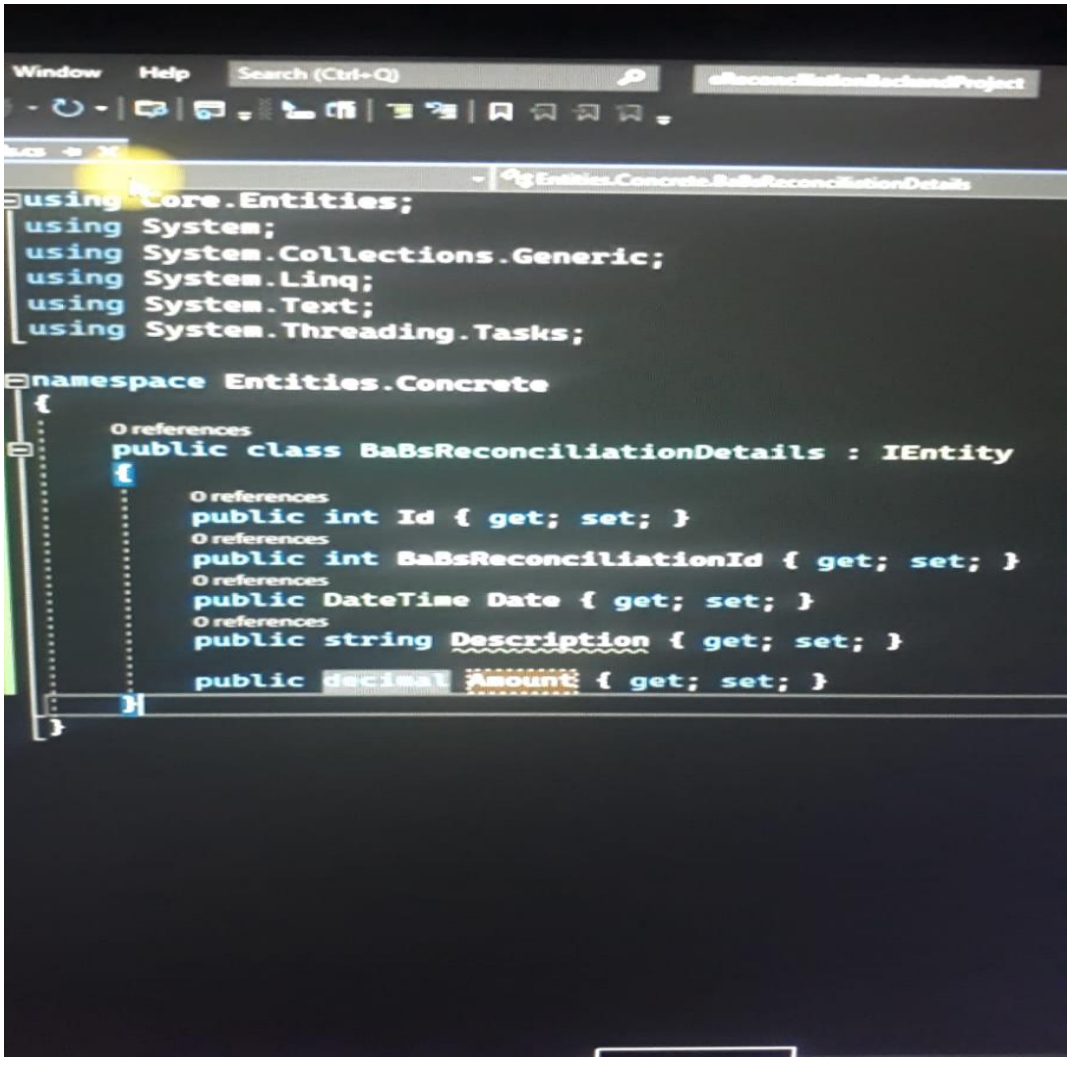
Üçüncü sınıfımızın ismi ise BaBsReconcalition olacak.

```
namespace Entities Concrete
{
public class BaBsReconcalition : IEntity
{
    public int Id { get; set; }
    public int CompanyId { get; set; }
    public int CurrencyAccountId { get; set; }
    public string Type { get; set; }
    public int Mounth { get; set; }
    public int Year { get; set; }
    public int Quantity { get; set; }
    public decimal Total { get; set; }
    public bool IsSendEmail { get; set; }
    public DateTime? SendEmailDate { get; set; }
    public bool? IsEmailRead { get; set; }
    public DateTime? EmailReadDate { get; set; }
```

```
        public bool? IsResultSucceed { get; set; }
        public DateTime? ResultDate { get; set; }
        public string? ResultNote { get; set; }
        public string? Guid { get; set; }
    }
    }
```

Diğer sınıfımızı örnek olması açısından görsel ile ele alacağız.

Dördüncü sınıfımız olarak BaBsReconcalitionDetail olacaktır.



Beşinci sınıfımız Company sınıfı.

public class Company : IEntity

```
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Address { get; set; }
    public string? TaxDepartment { get; set; }
    public string? TaxIdNumber { get; set; }
    public string? IdentityNumber { get; set; }
    public DateTime AddedAt { get; set; }
    public bool IsActive { get; set; }

 }
```

Altıncı sınıfımız Currency sınıfı.

```
public class Currency : IEntity

{
    public int Id { get; set; }
    public string Code { get; set; }
    public string Name { get; set; }
}
```

Yedinci sınıfımız ise CurrencyAccount.

```
public class CurrencyAcconut : IEntity

{

    public int Id { get; set; }
    public int CompanyId { get; set; }
    public string Code { get; set; }
    public string Name { get; set; }
    public string Address { get; set; }
    public string? TaxDepartment { get; set; }
    public string? TaxIdNumber { get; set; }
    public string? IdentityNumber { get; set; }
    public string? Email { get; set; }
    public string? Authorized { get; set; }
    public DateTime AddedAt { get; set; }
    public bool IsActive { get; set; }
}
```

Sekizinci sınıfımız ise ForgotPassword

```
public class ForgotPassword : IEntity
```

```
{
    public int Id { get; set; }
    public int UserId { get; set; }
    public string Value { get; set; }
    public DateTime SendDate { get; set; }
    public bool IsActive { get; set; }
}
```

Dokuzuncu sınıfımız ise Mailparameter.

```
public class MailParameter : IEntity

{
    public int Id { get; set; }
    public int CompanyId { get; set; }
    public string Email { get; set; }
    public string Password { get; set; }
    public string SMTP { get; set; }
    public int Port { get; set; }
    public bool SSL { get; set; }
}
```

Onuncu sınıfımız ise MailTemplate.

```
public class MailTemplate : IEntity

{

    public int Id { get; set; }
    public int CompanyId { get; set; }
    public string Type { get; set; }
    public string Value { get; set; }
}
```

Onbirinci sınıfımız ise TermsandCondition.

```
public  class TermsandCondition : IEntity

{

    public int Id { get; set; }
    public string Description { get; set; }
}
```

Onikinci sınıfımız ise UserRelationShip.

```
public class UserRelationShip : IEntity
```

```
{

    public int Id { get; set; }
    public int AdminUserId { get; set; }
    public int UserUserId { get; set; }

}
```

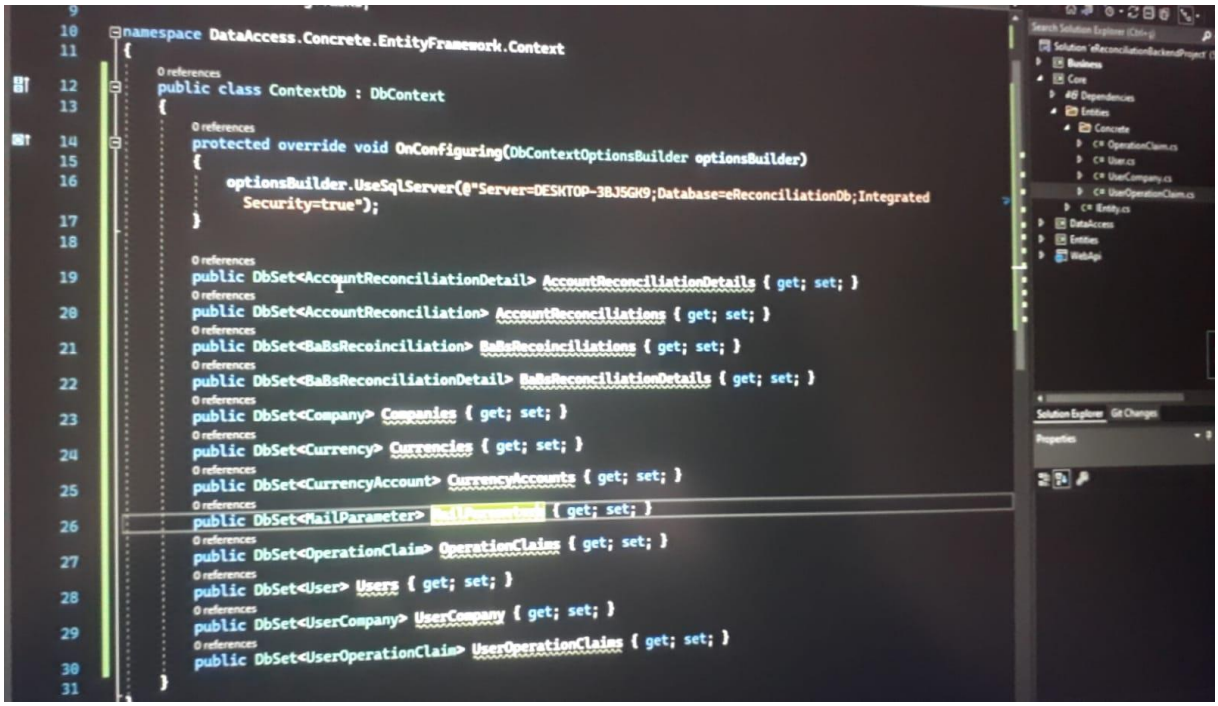
Onüçüncü ve son sınıfımız ise UserThemeOption

```
public class UserThemeOption : IEntity

{
    public int Id { get; set; }
    public int UserId { get; set; }
    public string SidenavColor { get; set; }
    public string SidenavType { get; set; }
    public string Mode { get; set; }
}
```

3.1.3 DATA ACSESS KATMANI

Data Access katmanında 2klasör oluşturarak başlayabiliriz bu klasörler abstract ve concrete klasörleridir. Abstract klasörümüzde imza tutucular dediğimiz interfaceleri, concrete klasörümüzde ise somut nesnelerimizi tutuyoruz.Bu katmanda nesnelerin birbirleri ile iletişim kurması amacı ile bir klasör daha oluşturuyoruz. Bu klasörümüzde Microsoft' un bize sağlamış olduğu kolaylık olan databese iletişimini sağlıyoruz. Klasörümüzün adı entityframework olacak. Nugetpackage aracılığı ile Microsoft.Entity.Framework.SqlServer paketini kuruyoruz. EntityFramework klasörün altına bir klasör daha oluşturup buna da yazılımda kullanılan isimlendirme adı olan Context olacak. Context klasörüne ContextDb isimli class oluşturuyoruz.

Oluşturmuş olduğumuz kod bloğu aşağıdaki şekildeki gibi oluşturuluyor. Böylece database bağlantımızı sağlamış oluyoruz. Bu kodlamadan sonra yapımıza CRUD işlemlerini yapıyor olmamız gerekiyor. CRUD işlemlerinin anlamı ise anlamı Create, Read, Update, Delete olmaktadır.



Sırada abstract klasörümüz var abstract klasörümüzde İnterface oluşturuyoruz. Fakat bunları generic yapılar içinde bir defa oluşturup diğer entity classlarımıza referans vermemiz gerekiyor. Generic yapımızı Core katmanında oluşturmamız gerekiyor core katmanında klasör oluşturup adını DataAccsess diyoruz. İnterface ismini ise IEntityRepository olarak veriyoruz. Oluşturulan kadlara örnek verecek olursak,

```
public interface IEntityRepository<T> where T : class, IEntity, new()
{
    void Add(T entity);

    void Update(T entity);

    void Delete(T entity);

    List<T> GetList(Expresion<Func<T,bool>>filter = null);

    T Get(int id);
}
```

Generic yapıda olduğu ve diğer classlara referans verileceği için kodumuz bu şekilde oluşturulması gerekiyor.
DataAccess katmanımıza bir klasör daha oluşturup İsmini EntityFramework veriyoruz. Nuget paket yükleyicisinden Microsoft.EntityFrameworkCore.SqlServer isimli paketi projemize dahil ediyoruz. Oluşturduğumuz olan klasöre EfEntityRepositoryBase isimli bir class oluşturuyoruz.
Örnek kodlamamız ise şu şekilde olmaktadır.

```
public class EfEntityRepositoryBase<TEntity,TContext> : IEntityRepository<TEntity>
where TEntity : class, IEntity, new()
where TContext : DbContext, new()
{
    public void Add(IEntity entity)
    {
      using(var context = new TContext())
       {
            var addedEntity = context.Entry(entity);
            addedEntity.State = EntityState.Added;
            context.SaveChanges();
       }
    }

    public void Delete(IEntity entity)
    {
      using(var context = new TContext())
       {
            var deletedEntity = context.Entry(entity);
            deletedEntity.State = EntityState.Deleted;
            context.SaveChanges();
       }
    }

    public void Update(IEntity entity)
    {
      using(var context = new TContext())
       {
            var updatedEntity = context.Entry(entity);
            updatedEntity.State = EntityState.Modified;
            context.SaveChanges();
       }
    }

    public List<TEntity> GetList(Expression<Func<TEntity, bool>> filter = null)
    {
            using(var context = new TContext())
            {
                return filter == null
                    ?context.Set<TEntity>().ToList()
                    : context.Set<TEntity>().Where(filter).ToList();
            }
```

```
    public TEntity Get(Expression<Func<TEntity, bool>> filter = null)
            using(var context = new TContext())
            {
                return context.Set<TEntity>().SingleOrDefault(filter)
            }

    }


}
```

Oluşturmuş olduğumuz generik yapıyı DataAccess katmanında interface oluşturup tüm class'lara implament edeceğiz örnek verecek olursak.

```
public interface ICompanyDal : IEntityRepository<Company>
{

}
```

Daha sonra DataAccsess katmanımız daki EntitiyFramework klasörümüze entityleri oluşturup gereken imzaları veriyoruz. Bu imzaların örnek kodu ise;

```
public class EfCompanyDal : EfEntityRepositoryBase<Company,ContextDb>,ICompanyDal
{

}
```

imzalar böylelikle otomatik olarak bu yapı içersinde oluşmuş oldu. Diğer entitilere bu katmanda aynı şekilde oluşturuyoruz.

Sırada EntityFramework classlarını oluşturmada DataAccess katmanında Entitiy klasörlerin altında class oluşturup her bir entity öğemizi örnekteki gibi kodluyoruz.

```
public class EfAccountReconciliationDal : EfEntityRepositoryBase<AccountReconciliation, ContextDb>IAccountReconciliationDal

{

}
```

DataAccess katmanımızın oluşturulması bitti sırada Business katmanımız var.

3.1.4 CORE KATMANI

Core katmanımızda sınıflarımızın temel operasyonlarını yazıp business katmanımızın bu katmandan referans almasını sağlıyoruz.
Core katmanımızda ise User, UserCompany, UserOperationClaim ve OperationClaim sınıflarımızı örnekteki gibi oluşturmamız gerekiyor.

İlk olarak User sınıfımızın nasıl oluşturulduğuna bakalım.

```
public class User : IEntity
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    public byte[] PasswordSalt { get; set; }
    public byte[] PasswordHash { get; set; }
    public DateTime AddedAt { get; set; }
    public bool IsActive { get; set; }
    public bool MailConfirm { get; set; }
    public string MailConfirmValue { get; set; }
    public DateTime MailConfirmDate { get; set; }

}
```

UserCompany Sınıfımızı oluşturuyoruz

```
public class UserCompany : IEntity
{
    public int Id { get; set; }
    public int UserId { get; set; }
    public int CompanyId { get; set; }
    public DateTime AddedAt { get; set; }
    public bool IsActive { get; set; }

}
```

UserOperationClaim Sınıfımızı oluşturuyoruz

```
public class UserOperationClaim : IEntity

{
    public int Id { get; set; }
    public int UserId { get; set; }
    public int OperationClaimId { get; set; }
    public int CompanyId { get; set; }
    public DateTime AddedAt { get; set; }
    public bool IsActive { get; set; }
}
```

Son olarak ise Bu katmanımızda bulunan OperationClaim sınıfımızı oluşturmamız gerekiyor.

```
public class OperationClaim : IEntity
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Description { get; set; }
    public DateTime AddedAt { get; set; }
    public bool IsActive { get; set; }
}
```

3.1.5 BUSİNESS KATMANI

Sunum katmanından tarafımıza gelen bilgileri gerekli koşullara göre işleyerek veya denetleyerek veri katmanının sağladığı metotları kullanarak veri tabanına gönderen aynı şekilde veri tabanından alan ve gerekli olan süreçlerden geçiren katmandır.

Business katmanımıza da kodlama açısından kolaylık sağlaması için abstract ve Concrete isimli iki klasör daha ekliyoruz.

Bu katmanımıza metot imzalarımızı oluşturacağız. Business katmanında yapmış olduğumuz işlemler web api katmanı tarafından çağrılmaktadır.

abstract klasörümüze interfaceimizi oluşturuyoruz. Servis metod imzalarımızı burada tutacağız. CRUD işlemlerindeki ihtiyacımız olanları burada çağıracağız. Ayrıca bu katmanımızda Kullanıcı yetkisi, transcaption, loglama ve validation işlemlerini yapacağız. Daha Sonra concrete klasöründe servis sınıflarımızı tutuyor olacağız.

ICompany Servisimiz için oluşturduğumuz kodlarımız:

```
public interface ICompanyService

{
   IResult Add (Company company);
   IResult Update(Company company)
   IDataResult<Company> GetById(int id);
   IResult AddCompanyAndUserCompany(CompanyDto compantDto);
   IDataResult<List<Company>> GetList();
   IDataResult<List<Company>> GetListByUserId(int userId);
   IDataResult<List<Company>> GetCompany(int userId);
```

```
    IResult CompanyExist(Company company);
    IResultUserCompanyAdd(int userId, int companyId);

}
```

Diğer sınıflarımızada aynı şekilde kullanmak istediğimiz CRUD operasyonlarını oluşturuyoruz. Daha sonra concrete klasörümüzde somut sınıflarımızı oluşturuyoruz.

```
public class CompanyManager : ICompanyService

{
    private readonly ICompanyDal _companyDal;
    private readonly IOperationClaimService _operationClaimService;
    private readonly IUserOperationClaimService _userOperationClaimService;
    private readonly IMailTemplateService _mailTemplateService;




public CompanyManager (ICompanyDal companyDal, IOperationClaimService operationClaimService, IUserOperationClaimService userOperationClaimService, IMailTemplateService mailTemplateService)

{
    _companyDal = companyDal;
    _operationClaimService = operationClaimService;
    _userOperationClaimService = userOperationClaimService;
    _mailTemplateService = mailTemplateService;

}
[CacheRemoveAspect("ICompanyService.Get")]
[ValidationAspect(typeof(CompanyValidator))]
[TransactionScopeAspect]

public IResult AddCompanyAndUserCompany(CompanyDto companyDto)

{
    Company company = new Company()

    {

    Id = companyDto.Id,
    Name = companyDto.Name,
    TaxDepartment = companyDto.TaxDepartment,
    TaxIdNumber = companyDto.TaxIdNumber,
```

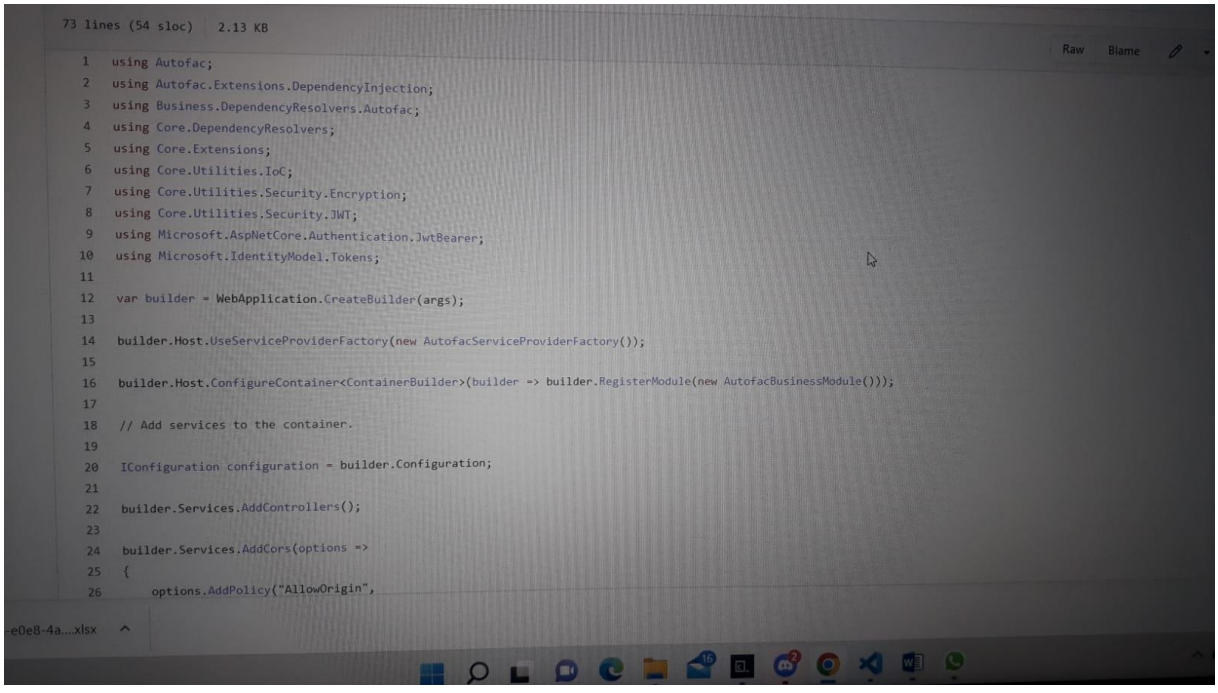
```
    IdentityNumber = companyDto.IdentityNumber,
    Address = companyDto.Address,
    AddedAt = companyDto.AddedAt,
    IsActive = companyDto.IsActive
    };

    _companyDal.Add(company);
    _companyDal.UserCompanyAdd(companyDto.UserId, company.Id);

    var operationClaims = _operationClaimService.GetList().Data;

    foreach (var operationClaim in operationClaims)
    {
    if (operationClaim.Name != "Admin")
    {
    UserOperationClaim userOperation = new UserOperationClaim()

    {
    CompanyId = company.Id,
    AddedAt = DateTime.Now,
    IsActive = true,
    OperationClaimId = operationClaim.Id,
    UserId = companyDto.UserId

   };
    _userOperationClaimService.Add(userOperation);


  }
    }

    var mailTemplate = _mailTemplateService.GetByCompanyId(4).Data;
    mailTemplate.Id = 0;
    mailTemplate.Type = "Mutabakat";
    mailTemplate.CompanyId = company.Id;
_mailTemplateService.Add(mailTemplate);
return new SuccessResult(Messages.AddedCompany);
}
public IResult CompanyExists(Company company)

{
var result = _companyDal.Get(c=> c.Name == company.Name && c.TaxDepartment ==
company.TaxDepartment && c.TaxIdNumber == company.TaxIdNumber &&
c.IdentityNumber == company.IdentityNumber);

if (result != null)
```

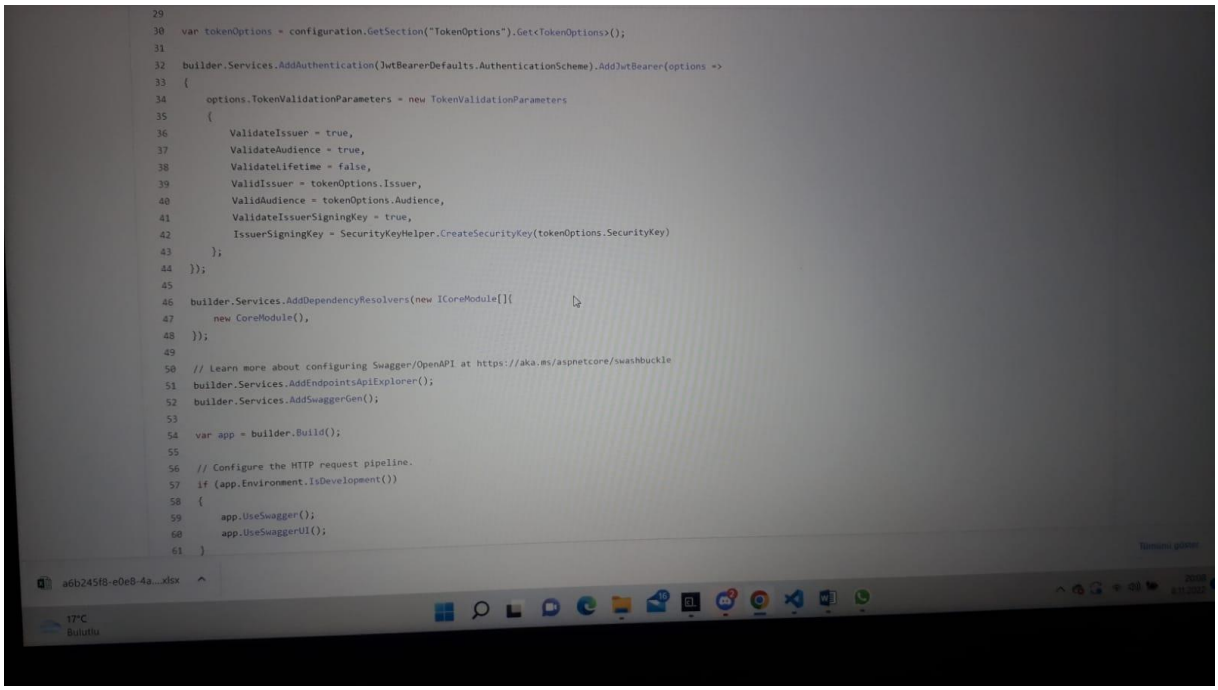
```
{
return new ErrorResult(Messages.CompanyAlreadyExists);
}
return new SuccessResult();
```

3.1.6 WEB APİ KATMANI

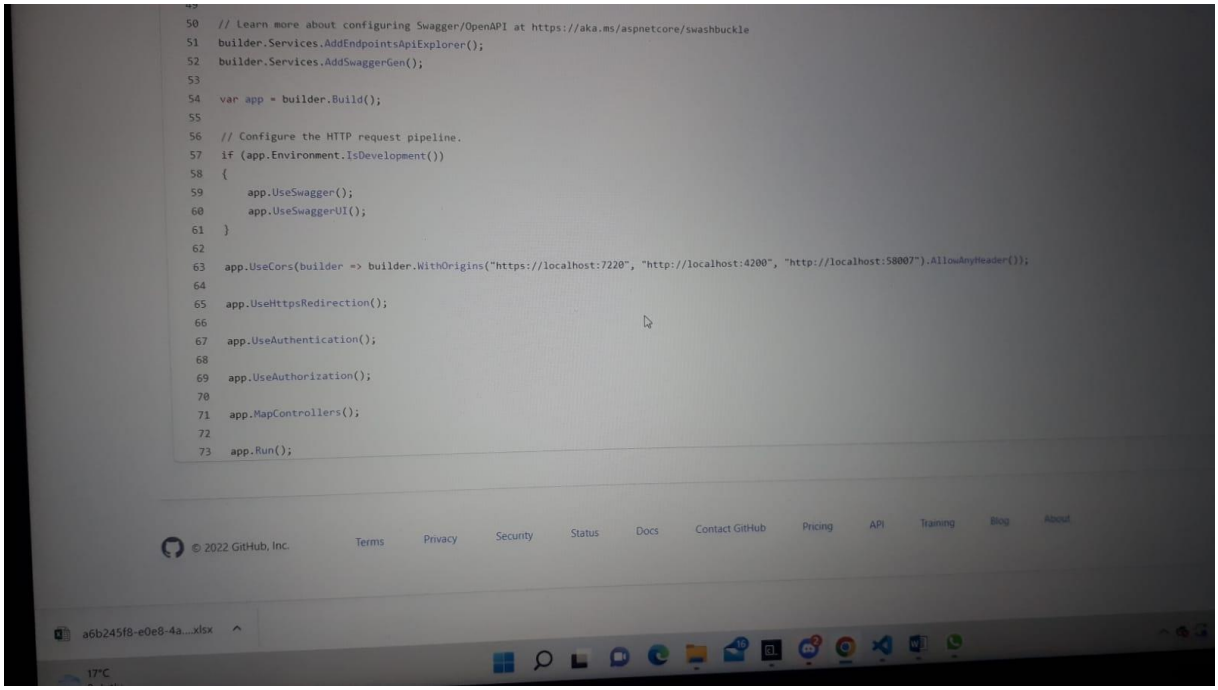
Genel olarak Core işlemlerinin içindeki iş yapan sınıflar Business katmanında bulunur. Bu katmanda Entitiy'deki tüm sınıflar örnekteki gibi yazılır. Bu katmandan sonra Ara yüzle iletişim kurabilmemiz için Web Api katmanına ihtiyacımız bulunmaktadır.
WebApi hakkında bilgi verecek olursak.
etkileşimlerin kuruluşlar ile uygulamalar arasında gerçekleştiği arayüzler diyebiliriz. Hypertext Transfer Protocol (HTTP) istek mesajları ile extensible markup language (XML) veya javascript object notation (JSON) formatında bulunan tepki mesajlarının tanımıdır.[38]

WebApi katmanımızda Content, Controllers, Properties dosyalarında kodlamalarımızı yaparız ve bu yaptığımız kodlar Program.cs Sınıfımızda örnekteki gibi çalışmaktadır.





```
var tokenOptions = configuration.GetSection("TokenOptions").Get<TokenOptions>();

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme).AddJwtBearer(options =>
{
    options.TokenValidationParameters = new TokenValidationParameters
    {
        ValidateIssuer = true,
        ValidateAudience = true,
        ValidateLifetime = false,
        ValidIssuer = tokenOptions.Issuer,
        ValidAudience = tokenOptions.Audience,
        ValidateIssuerSigningKey = true,
        IssuerSigningKey = SecurityKeyHelper.CreateSecurityKey(tokenOptions.SecurityKey)
    };
});

builder.Services.AddDependencyResolvers(new ICoreModule[]{
    new CoreModule(),
});

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}
```



```
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseCors(builder => builder.WithOrigins("https://localhost:7220", "http://localhost:4200", "http://localhost:58007").AllowAnyHeader());

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();
```

SONUÇ

Çok katmanlı yazılım mimari yapısı günümüzde popüler olan mimari kalıplarından biridir. Bu katman uygulamalarda oluşan karmaşıklığı azaltmaktadır. Daha çevik bir şekilde çalışmalarımızı da kolaylaştırmaktadır. Çok katmanlı mimari yapı her bir katman farklı hizmet ve entegrasyona karşılık gelen çeşitli katmanlardan oluşur. Bu katmanın bize sağlamış olduğu en önemli faydalar ise Kompleks ve gitgide karmaşıklaşan projelerde daha kolay hakimiyet sağlamamıza, hata ayıklamanın kolaylaşmasına, projede ekip olarak çalışmayı kolaylaştırmayı, değişim sürecini hızlandırmasını ve bize esnek bir yapı sağlaması olarak özetleyebiliriz.

KAYNAKÇA

[1] “Turing’s Legacy: A History of Computing at the National Physical Laboratory 1945-1995”

[2] “Data Communicatians at the National Physical Laboratory 1965-1975”

[3] Couldry, Nick (2012)

[4] Sofroniou, Andreas, Surfing the internet then, now, later p.241

[5]Beyzan ve Özbilen, 2011

[6]Özgit ve Çağıltay, 1996

[7]Beyzan ve Özbilen, 2011

[8]Kaya, 2005

[9]Özgit ve Çağıltay, 1996

[10]Onursoy, 2001 s.28

[11]Powell,2002 s.155

[12]Yücel, 2007, s. 20

[13]Nandini,2014, s1 11

[14]Yücel, 2007, s.18

[15]Arı, 2002, s.15

[16] Yücel, 2007, s.18

[17] Keş,2009 s.8

[18] Alp,S. Özdemir, S ve Kilitçi, A.(2011)

[19]Braunstain, Mark L.(26 Temmuz 2018)

[20]Abdallah, M.M & AL-Rifaee, M.M (2017)

[21]Aderhold, M., & Kochtchi, A. (2013)

[22]Ayewah, N., Pugh, W.,Hovemeyer(2008)

[23]Bajwa, M. S., Agarwal, A. P.&Gupta, N.(2016)

[24]opendart.com/makaledetay_hibernate-nedir-6-109.html

[25]www.python.org/doc/sunset-python-2

[26]”Whetting Your Appetite”. The Python Tutorial

[27]TIOBE – The software Quality Company.Archived from the original on 12 October2021.

[28]Docs.Python.org

[29]Geeksforgeeks.org

[30] Geeksforgeeks.org/python-web-development-django-tutorial

[31]Beijing Farnham: O'Reilly. 18Nisan 2011

[32]w3tech.com. 13 Ağustos 2021

[33]EcmaScript 2020 Language Specification

[34]www.sitepoint.com

[35]alastyr.com/blog/angular-nedir-ne-ise-yarar/

[36]hostinger.web.tr/rehberler/angular-nedir/

[37]natro.com/blog/react-nedir-nasil-çalışir/

[38]apikütüphanesi.com/web-api-nedir/

[39]Standart ECMA-334 2 Mayıs 2020

[40]Top Programming Languages To Follow in 2020

[41]ECMA Specifications for C# 2 mayıs 2020