İZMİR KÂTİP ÇELEBİ ÜNİVERSİTESİ
2010
FEN BİLİMLERİ ENSTİTÜSÜ

# Otomotiv Araçlarının Görüntüleneceği Web Uygulaması

Yazılım Mühendisliği Ana Bilim Dalı
Dönem Projesi

Mesut Solak

ORCID 0009-0001-0764-6387

Proje Danışmanı: Prof.Dr. Doğan Aydın

Ocak 2024

# Otomotiv Araçlarının Görüntüleneceği Web Uygulaması

# Öz

Otomotiv araçlarının markalara göre listelendiği bir web uygulamasıdır.Web uygulaması araç listeleme , detay ve filtreleme işlemlerinden oluşmaktadır. Kullanıcının en çok tercih ettiği araç modeli , en başta ve sarı bir çerçevede gelecek şekilde listelenmektedir.Detay kısmında araç modellerinin alt modelleri ve detaylı arama kısmı bulunmaktadır.Ayrıca ilgili markanın araç için belirlemiş olduğu kurallarda bulunmaktadır.Detaylı aramada ise model ve fiyata göre arama yapılabil mektedir.

**Anahtar Sözcükler:** otomotiv , marka , fiyat , model , filtreleme

# Web Application To Display Automotive Vehicles

## Abstract

It is a web application where automotive vehicles are listed by brands. The web application consists of vehicle listing, detail and filtering operations.The user's most preferred vehicle model is listed first and in a yellow frame. In the detail section, the vehicle models are listed first.There are sub-models and a detailed search section. There are also rules set by the relevant brand for the vehicle. In detail.You can search by model and price.

**Keywords:** automotive, brand, price, model, filtering

*Proje çalışmamı zor zamanımda yanımda olan abime ithaf ediyorum..*

# Teşekkür

Projenin başlangıcında beni yönlendiren eski danışman hocam Prof. Dr. Femin Yalçın Küçükbayrak hocama , projeyi geliştirmemde süreçlerimi takip eden ve tavsiyelelerde bulunan Prof. Dr. Doğan Aydın hocama çok teşekkür ederim.

# Şekiller Listesi

# İçindekiler

# Bölüm 1

## Giriş

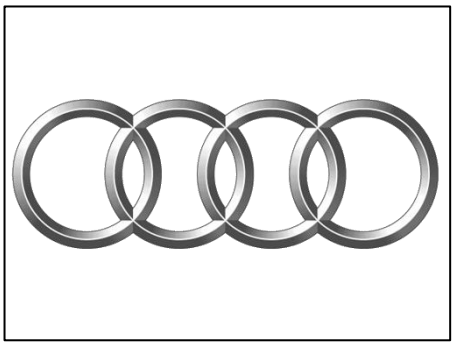
Otomotiv araçlarının markalara göre listelendiği ve detaylarını görebildiği bir web uygulamasıdır.Ayrıca kullanıcının en çok tercih ettiği model araç listeleme sayfasının başında gelmektedir. (Audi Türkiye, 2023)

Araç markaları : Audi , Seat , Skoda , Scania , Porsche  gibi markalardan oluşabilir.

Uygulamanın alt yapısı marka bazlı kırılıma oldukça uygundur.

Projenin asıl amacı kullanıcıların seçilen marka modeline göre araç modellerini , modelinin detayını ve alt modellerini görmesini sağlamaktır.

Bunun yanı sıra kullanıcılar fiyat bilgisini de görebilmektedir.Fiyat bilgileri ve araç detayları yıllara göre değişkenlik gösterebilmektedir.



Şekil 1.1: Audi logo

Örnek olması açısından projede audi markası kullanılmıştır.Audi, Alman menşeli bir otomobil şirketidir ve Volkswagen grubunun bir markasıdır. Şirketin merkezi Ingolstadt, Bavyera'da bulunmaktadır.

Şirketin geçmişi 1899 yılına ve August Horch'a dayanmaktadır. İlk Horch otomobili kendisi tarafından 1901 yılında tasarlanmıştı. 1910 yılında Horsche, şirket dışına atılmış ve kendi adını, eski ortaklarıyla olan anlaşmazlıklar nedeniyle yaptığı tasarımlarda kullanamayacak hale gelmişti. Eski Almancada anlamı "Dinle!" olan "Horch", Latincede aynı anlama gelen Audi'yi kendi markası olarak kullanmaya başladı.

1932 yılında Audi, Auto Union'ı oluşturmak üzere Horch, DKW ve Wanderer şirketleri ile birleşti.

II. Dünya Savaşı sonrasında şirket, DKW etrafında ürünlerini sunmaya çalıştı; ancak iki çekişli motoru o kadar ünlü olamadı. Eylül 1965'te Audi, dünyanın en modern motorlarından biriyle tekrar bir çıkış yaparak 72 beygirlik 4 kapılı sedanını piyasaya sundu. (Vikipedi Audi, 2016)

# Bölüm 2

## Materyal ve Yöntem

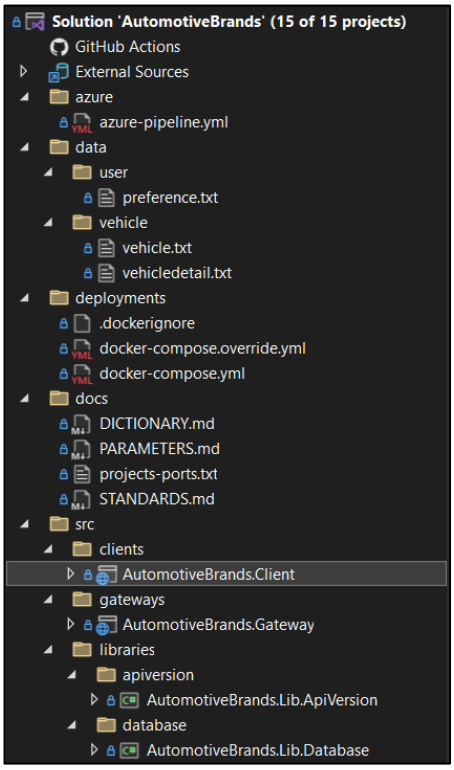
Proje geliştirilirken C# programlama dili , .NET 6 kütüphanesi ve veritabanı olarak PostreSQL kullanılmıştır.Uygulamalarda kullanılacak olan paketler , **libraries** klasörü altında toplanmaktadır. (Microsoft .NET, 2023)

Uygulama kullanılan bazı paketler:

- Fluent validation
- Ratelimit
- Npsql
- Entity framework
- Swagger
- Redoc
- Ocelot

Projeyi daha iyi anlamak için öncelikle klasör mimarisine bakmalıyız.Klasör mimarisini ne kadar iyi tutarsak geliştirmeleriz o kadar iyi olacaktır. Uygulamamız tek bir solution içerisinde birden fazla projeden oluşmaktadır.

Projenin büyük olması durumunda uygulamalarımızı klasörlere yani işi parçalara ayırarak yönetebilirliği ve hızlı geliştirme yapmamıza olanak sağlamaktadır.




Şekil 2.1: Klasör mimarisi - 1

**Azure** klasörü solution içerisinde var olan projelerin azure üzerinde pipeline kullanabilmesi için gerekli dosyaları içermektedir.

**Data** klasörü solution içerisinde var olan servis uygulamalarının örnek verilerinin tutulduğu klasördür.Data klasörü 2 servis olmasından dolayı user ve vehicle servislerini içermektedir.

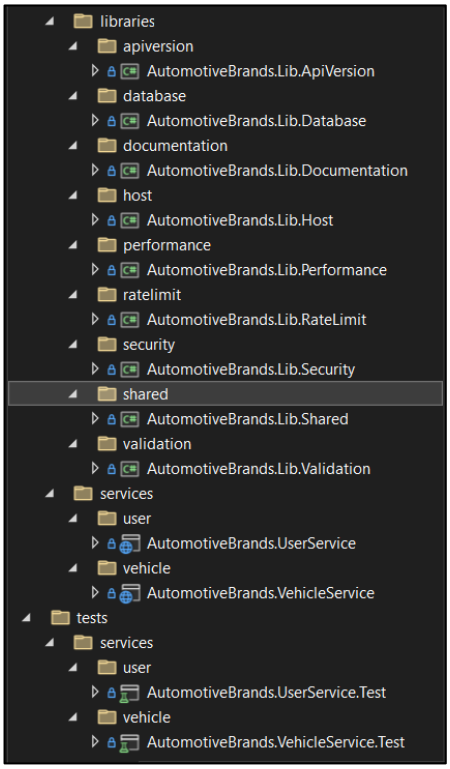
**Deployments** klasörü solution içerisinde docker container dosyalarının bulunacağı klasördür.

**Docs** klasörü solution içerisinde projeyi tanıtan ve kurallarını içeren dosyaların tutulacağı klasördür.

**Src** klasörü kendi içerisinde clients , gateways , libraries ve services olmak üzere 4 tane klasörden oluşmaktadır.Genellikle projenin ana dosyaları burda bulunmaktadır.

**Clients** klasörü uygulamada kullanılacak olan client uygulamaların dosyalarını içermektedir

**Gateways** klasörü mikroservislerin ihtiyacına göre api gateway projelerinin bulunduğu dosyaları içermektedir.




Şekil 2.2: Klasör mimarisi - 2

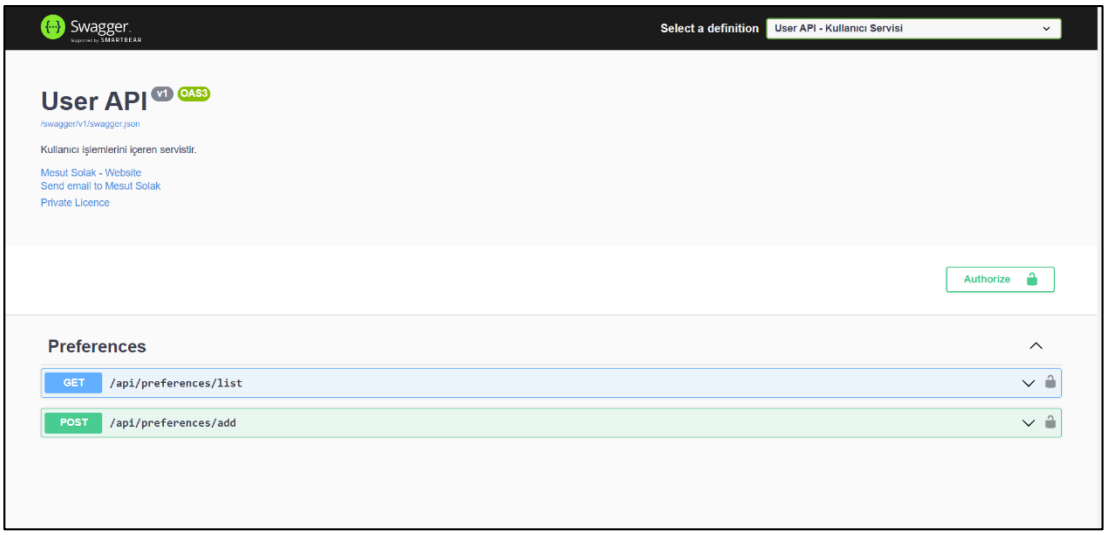
**Libraries** klasörü uygulamalarda ortak olarak kullanılacak olan dosyaları tek bir yerden ulaşmamızı sağlayan dosyaları içermektedir.Validasyon , loglama , performans , güvenlik ve api versiyonlama gibi örnekler verebiliriz.

**Services** klasörü mikroservislerin bulunduğu klasördür.

**Tests** klasöründe ise özellikle mikroservislerin ve ihtiyaca göre test yazılması gereken tüm uygulamaların unit test projeleri içermektedir.

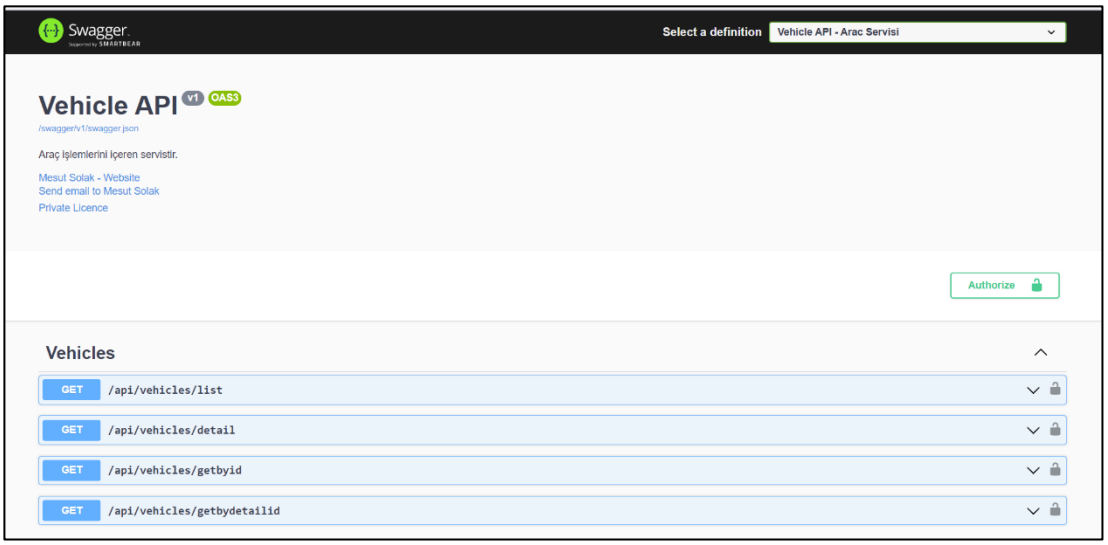
Mikroservis projelerinden bahsetmek gerekirse user ve vehicle olmak üzere 2 tane mikroservis projemiz bulunmaktadır.Kullandığımız api projelerini daha iyi anlamak için swagger dokümantasyonu kullanılmaktadır.Swagger kütüphanesi özellikle client uygulamaları yazan geliştiriciler ve ürün sahipleri için önemli bir yere sahiptir.

Swagger kullanıyor olmamız doküman yazmayacağımız anlamına gelmemektedir. Özellikle iş akışlarını gösterebilmek için dokümantasyon yazabiliriz. (Swagger, 2024)




Şekil 2.3: Kullanıcı api swagger

Kullanıcı işlemlerini içeren servistir.Tercihleri listelemek ve tercih eklemek için kullanılmaktadır.Bu servisi kullanarak kullanıcın en çok tercih ettiği araç modelini bulmaktayız.




Şekil 2.4: Araç api swagger

Araç işlemlerini içeren servistir.Araç listeleme , araç alt modellerini listeleme ve araç detayı getirme gibi işlemleri içermektedir.Api sadece get olarak çalışmaktadır.Veri ekleme ve çıkarma işlemlerini veritabanı üzerinden gerçekleştirebiliriz.

## 2.1 C#

C# programlama dili, Microsoft tarafından ECMA ve ISO standartlarında geliştirilmiş orta seviye bir programlama dilidir. ECMA ve ISO yazılım dünyasında diller konusunda standartları belirleyen en yetkin iki kuruluştur. Microsoft, her ne kadar ilk başlarda bu standartlar çerçevesinde geliştirmeye başlasa da C# 3.0 ile birlikte kendi standardını oluşturmuş ve dili tamamen .NET için geliştirmeye başlamıştır. Dolayısıyla herhangi bir kurum ya da kuruluşa bağlı kalmaksızın, yazılım geliştiricilerin ihtiyaçlarını karşılamak için oldukça kullanışlı, güçlü ve strong typing (strongly typed) programlama dili geliştirilmiştir.

Adının temeli C ve C++ dilinden gelir. C dilinde bir tamsayı değişkenin değerini artırmak için "++" operatörü kullanılır. Bu yüzden C dilinden sonra geliştirilen ve Nesne Yönelimli (Object Oriented) özellikleri taşıyan yeni dil için, C'nin bir fazlası anlamında C++ adı kullanılmıştır. C++'tan sonra geliştirilen C# dili ise adını; ((C++)++) ifadesinde artı işaretlerinin bir araya gelmesinden alır.

Bu dili tasarlayan ekibin başında Anders Hejlsberg bulunur ve C# dilinin geliştirilmesi ona atfedilir. Daha önce Pascal ve Delphi derleyicileri de tasarlamış olan Anders Hejlsberg; yeni geliştirdiği dilde, o dönemin en yaygın ve en beğenilen Nesne Yönelimli dili olan Java'nın söz dizimini temel almıştır.

Bir programlama dili için orta seviye ifadesinin kullanılması, o dilin gücünden kaynaklanmaz. Yazılım dünyasında diller, makine diline ya da günlük konuşma diline yakın olmalarına bağlı olarak düşük, orta ve yüksek seviye diller olarak ifade edilir. Yüksek seviye diller günlük konuşma kalıplarına benzer söz dizimlerine sahiptir. C# hem makine diline, hem de günlük konuşma diline eşit seviyede yer verir ve bundan ötürü orta seviye bir dil olarak nitelenir.

Şekil 2.5: C#

### 2.1.1 C# Programlama Dili ile Neler Yapılır?

C# programa diliyle neler yapılabileceği çokça merak edilen sorular arasında yer alıyor. Diğer programla dillerinde olduğu gibi C# ile de farklı uygulamalar ve programlar oluşturulabilir. Masaüstü, mobil, web ve oyun geliştirmede kullanılır. Ayrıca loT, bulut ve API'ler gibi her türlü uygulamayı C# kullanarak yapabiliriz.

- C# ile web geliştirme: Web geliştirmede C# oldukça popülerdir. .NET platformunda profesyonel ve dinamik web site geliştirmede kullanılır. C# ile oluşturulan web hizmetleri hızlı ve güvenilirdir. C# ile web sitelerinin alt yapılarını çok işlevli hale getirebilirsiniz.
- C# ile mobil uygulamalar geliştirme: Xamarin platformunu kullanarak C# programlama dili ile iOS, Android ve Windows için modern mobil uygulama geliştirebilirsiniz.
- C# ile oyun geliştirme: Oyun geliştirmek için kullanılan programla dillerinin başında C# gelmektedir. Bağımlılık yapan, yüzlerce kez indirilen oyunlardan oluşturmak istiyorsanız C# öğrenmek kesinlikle doğru bir seçenek olacaktır. 770 milyondan fazla kullanıcısıyla en popüler oyun motorlarından biri olan Unity ile de sorunsuz çalışmaktadır. Sadece Unity ile değil piyasadaki birçok oyun motoru tarafından desteklenmektedir.
- C# ile DLL yazma: Öncelikle DLL'in ne demek olduğunu kısaca açıklayalım. Dinamik Bağlantı kitaplığı (DLL), aynı anda birden fazla program tarafından kullanılabilen işlevler ve kodlar içeren bir kitaplıktır. DLL dosyalarını C# ile kodlayabilmek mümkündür.

### 2.1.2 C# Neden Tercih Edilir?

- C# öğrenmesi nispeten kolaydır. Debugger özelliği ile "syntax" hatalarının kolayca tespitini yapar. Sorunlar hakkında endişelenmenize gerek kalmaz. Ayrıca, size kod yazmayı daha hızlı ve kolay hale getiren kitaplıkları da mevcuttur.
- C# açık kaynak kodlu bir programlama dilidir. Bu da esnek ve herkes tarafından geliştirilebilen bir dil olduğunun göstergesidir. Ayrıca program geliştirme ve bakım işlemleri daha kolay yapılabilmektedir.
- C# hızlıdır. C# ile hızlı kod yazmak diğer dillere göre daha kolaydır. Yüksek performanslı uygulamaları hızlı bir şekilde C# ile yazabilirsiniz.
- C#, nesne yönelimli programlamanın temel prensiplerini destekler. Bu, kodun düzenlenmesini, bakımını ve tekrar kullanılabilirliği artırarak yazılım geliştirmeyi kolaylaştırır.
- C#, zaman içinde birçok dil özelliği ekleyerek gelişmiştir. Lambda ifadeleri, LINQ (Language Integrated Query), async/await gibi özelliklerle modern programlama paradigmalarına uygun bir dil olmuştur.

## 2.2 .NET

.NET, herhangi bir işletim sisteminde yerel olarak çalışabilen masaüstü, web ve mobil uygulamalar oluşturmaya yönelik açık kaynaklı bir platformdur. .NET sistemi, modern, ölçeklenebilir ve yüksek performanslı yazılım geliştirmeyi destekleyen araçlar, kütüphaneler ve diller içerir.

- Verimli yazılım geliştirme için yardımcı programlar sağlar. Örneğin, mevcut saati bulabilir veya ekranda metin yazdırabilir.
- Bilgisayarda metin, sayı ve tarih gibi bilgileri depolamak için bir dizi veri türü tanımlar.

### 2.2.1 .NET Framework

.NET Framework özgün .NET uygulamasıdır. Windows'ta çalışan web siteleri, hizmetleri, masaüstü uygulamaları ve daha fazlasını destekler. Microsoft 1990'ların başında .NET Framework'ü yayınlamıştır.

### 2.2.2 .NET Core

Microsoft, .NET geliştiricileri için platformlar arası destek sağlamak üzere 2014 yılının sonlarında .NET Core'u kullanıma sunmuştur. Şirket, Kasım 2020'de .NET Core'un en yeni sürümü olan .NET 5.0'ı yayınlamış ve adını .NET olarak değiştirmiştir. Bu makaledeki .NET terimi .NET 5.0'a atıfta bulunur. .NET Core GitHub'da açık kaynaklıdır.

### 2.2.3 .NET Standard

.NET Standard, farklı işlevlerin (API'ler olarak adlandırılır) kurallı belirtimidir. Farklı .NET uygulamaları aynı kodu ve kütüphaneleri yeniden kullanabilir. Her uygulama hem .NET Standard API'leri hem de üzerinde çalıştığı işletim sistemlerine özgü benzersiz API'leri kullanır.


Şekil 2.6: .NET

### 2.2.4 Neden .NET'i seçmelisiniz?

**Geliştirme kolaylığı**

Geliştiriciler, çalışmalarını kolaylaştıran birçok araç içerdiğinden .NET kullanmayı severler. Örneğin, geliştiriciler, Visual Studio paketini kullanarak kodu daha hızlı yazabilir, verimli bir şekilde iş birliği yapabilir ve kodlarını verimli bir şekilde test edip düzeltebilir.

**Yüksek performanslı uygulamalar**

.NET uygulamaları daha hızlı yanıt süreleri sağlar ve daha az bilgi işlem gücü gerektirir. Güçlü yerleşik güvenlik önlemlerine sahiptirler ve veri tabanı erişimi gibi sunucu tarafı görevleri verimli bir şekilde yerine getirirler.

**Topluluk desteği**

.NET açık kaynaktır, bu da herkesin özgürce kullanmak, okumak ve değiştirmek için erişebileceği anlamına gelir. Aktif bir geliştirici topluluğu, .NET yazılımını geliştirir ve bakımını yapar.

### 2.2.5 .NET mimarisinin bileşenleri nelerdir?

.NET modüler ve optimize edilmiş bir mimariye sahiptir. Kullanıcılar, yazılım geliştirme gereksinimlerini karşılamak için farklı bileşenler seçebilirler.

Bunlar üç ana .NET bileşenidir:

- .NET dilleri
- Uygulama modeli çerçeveleri
- .NET çalışma zamanı

Geliştiriciler, .NET uygulamalarını oluşturmak için .NET programlama dillerini ve uygulama modeli çerçevelerini kullanırlar. .NET çalışma zamanı daha sonra bunları yürütür ve çalıştırır.

### 2.2.6 .NET çalışma zamanı nedir?

**Tam zamanında derleme**

CLR, geliştirici kodu yazarken derler. Derleme sırasında CLR, kodu Ortak Ara Dil'e (CIL) çevirir. Örneğin, C# ile yazılmış kod İngilizce benzeri söz dizimi ve sözcüklere sahiptir. .NET bu kodu CIL'e derler veya çevirir.

**Yürütme**

.NET çalışma zamanı CIL kodunun yürütülmesini yönetir. CIL platformlar arası uyumludur ve herhangi bir işletim sistemi bunu işleyebilir. Platformlar arası uyumluluk, bir uygulamanın minimum modifikasyonla birden çok farklı işletim sisteminde çalışma yeteneğini ifade eder.

## 2.3 PostreSQL

PostgreSQL, hem ilişkisel (SQL) hem de ilişkisel olmayan (JSON) sorgulamayı destekleyen açık kaynaklı, kurumsal sınıf ve gelişmiş bir nesne-ilişkisel veri tabanı sistemidir. SQL dilini, karmaşık veri iş yüklerini yüksek verimlilikle işlemesini sağlayan ek özelliklerle kullanır ve genişletir.

### 2.3.1 Kısa Tarihi

- PostgreSQL'in yolculuğu 1977'den itibaren Ingres projesiyle başladı. Kaliforniya Üniversitesi, Berkeley'de geliştirildi.
- 1986'da Profesör Michael Stonebraker, POSTGRES projesini yönetti.
- 1987'de ilk demo sürümünü çıkardılar.
- 1994 yılında Postgres'e bir SQL Interpreter eklediler.
- 29 Ocak'ta geliştiriciler, ilk PostgreSQL olan ve sürüm 6.0 olarak bilinen 1997'yi yayınladı.
- 1997'den beri geliştiriciler, PostgreSQL Global Development Group'un izniyle PostgreSQL'i geliştirmeye ve sürdürmeye devam ediyor.

### 2.3.2 Özellikleri

PostgreSQL'in sunabileceği tonlarca özellik vardır. Geliştiricilerin uygulama oluşturmalarına ve herhangi bir veri hacmini yönetmelerine yardımcı olurken, aynı zamanda yöneticilerin veri bütünlüğünü sağlamasına izin verir.

**Geniş İşletim Sistemi Uyumluluğu:** Windows, macOS, Linux, UNIX vb. gibi önde gelen tüm işletim sistemleriyle uyumludur.

**Dil Desteği:** C#, C/C+, Java, Python, JavaScript (Node.js), Ruby vb. dahil tüm popüler programlama dillerini destekler.

**Geniş Veri Tipleri Uyumluluğu:** Primitives (String, Numeric, Integer, Boolean), Structured (Dizi, Date/Time, UUID, Range), Geometry (Poligon, Line, Point, Circle), Document gibi geniş veri tiplerini destekler ( XML, JSON/JSONB).

**Veri Bütünlüğü Desteği:** Yabancı anahtarlar, birincil anahtarlar, dışlama kısıtı, tavsiye kilitleri, açık kilitler, NOT NULL ve UNIQUE için destek sunarak veri bütünlüğünü sağlar.

**Güvenlik:** Sağlam bir erişim kontrol sistemi, kimlik doğrulama (LDAP, SSPI, GSSAPI, Sertifika vb.), çok faktörlü kimlik doğrulama ve sütun ve satır düzeyinde güvenlik sunarak güvenli bir veri ortamı sağlar.

PostgreSQL ile sağlanan diğer özelliklerden bazıları; belirli bir noktadan sonra kurtarma, eşzamansız çoğaltma, iç içe işlemler, tablo bölümleme ve daha fazlasını içerir.

### 2.3.3 Avantajları

Zengin özellik listesi, topluluk desteği ve birinci sınıf performansa dayanan PostgreSQL'in dünya çapında pek çok işletme tarafından favori seçenek olarak görülmesi şaşırtıcı değildir.

**Ücretsiz ve Açık Kaynak Lisansı**

PostgreSQL, BSD veya MIT lisanslarına benzer şekilde liberal bir açık kaynak lisansı olan PostgreSQL Lisansı altında yayınlanır. Bu, kaynak kodunun erişilebilir olduğu anlamına gelir ve kullanıcılara veri projelerinin taleplerine göre kullanma, değiştirme, paylaşma ve uygulama özgürlüğü verir. Herhangi bir lisans ücreti veya herhangi bir sözleşme sorunu yoktur.

**Güvenilir – Topluluk Odaklı**

25 yılı aşkın bir süredir özel bir PostgreSQL topluluğu, hataları düzenli olarak düzeltir ve genel performansı iyileştirir. Böylece veri tabanı sisteminin verimliliğini sürekli olarak iyileştirir.

**Genişletilebilirlik**

PostgreSQL kolayca ve geniş ölçüde genişletilebilirdir. Örneğin, kullanıcılara kendi veri türlerini tanımlama, özel işlevler tasarlama veya veri tabanlarını yeniden derlemeden diğer programlama dillerinden kod yazma yetkisi verilir.

**Koddaki Yorumlar**

Diğer birçok veri tabanı sistemiyle karşılaştırıldığında, PostgreSQL benzersiz bir şekilde kodda yorumlar sağlar. Bu, kullanıcıların, uygulamalarına dağıtmadan önce belirli bir kodun neler yapabileceğini hızlı bir şekilde anlamak için yorumları okuyabileceği anlamına gelir. Bu, topluluğun kalitesini ve kapasitesini genel olarak iyileştirmeye olanak tanır.

**Yeni Veri Tabanı Teknolojilerine Uyum**

PostgreSQL, işletmelerin bulut hizmetleri, makine öğrenimi, veri ambarı analitiği, IoT, buluta geçiş hizmetleri vb. gibi yeni veri tabanı teknolojilerini uygulamalarına olanak tanır. PostgreSQL'in artık bu kadar popüler olmasının ve işletmeler tarafından giderek daha fazla kullanılmasının nedenlerinden biri de budur.

**Yüksek Kullanılabilirlik ve Yük Dengeleme**

Sürekli planlama, yedek sunucu çalışması, birincil sunucuyu yedek sunucular için hazırlama, yedek sunucu kurma, akış çoğaltma, basamaklı çoğaltma, senkronize çoğaltma ve sürekli arşivleme yoluyla yüksek kullanılabilirlik ve yük dengeleme sağlar.

**Yedekleme ve Geri Yükleme**

PostgreSQL veri tabanları, değerli verilerin düzenli olarak yedeklenmesini sağlamak için yapılandırılabilir. Verileri yedeklemeye yönelik temelde farklı üç yaklaşım vardır: SQL dökümü, dosya sistemi düzeyinde yedekleme ve sürekli arşivleme.

## 2.4 Swagger

Swagger, özellikle API (Application Programming Interface) geliştirme süreçlerinde kullanılan açık kaynaklı bir API belgeleme aracıdır. Swagger, API'leri tasarlamak, geliştirmek ve belgelemek için kullanılan bir dizi araç ve standartları içeren bir proje ekosistemini ifade eder.

İlk olarak, Swagger şu unsurları içeren bir API belirleme standartını temsil eder:

- Swagger OpenAPI Belirtimi (Swagger OpenAPI Specification): API'nin nasıl çağrılacağını, yanıtlarını, hata durumlarını ve diğer detaylarını tanımlayan bir JSON veya YAML formatında bir belgedir. Bu belge, API'nin işlevselliğini ve kullanımını açıklar.
- Swagger Editor: Swagger OpenAPI Belirtimi'nin yazılmasını ve düzenlenmesini sağlayan bir web tabanlı araçtır. Geliştiriciler bu aracı kullanarak API'lerini tasarlayabilir ve belgeleyebilir.
- Swagger UI: Swagger belgesini web tabanlı bir arayüzde görsel olarak sunan bir araçtır. Bu sayede, API'nin kullanımını anlamak ve test etmek daha kolay hale gelir.
- Swagger Codegen: API belgesinden başlangıçta çalışan bir istemci veya sunucu uygulaması oluşturmak için kullanılan bir araçtır. Bu, geliştiricilere API'yi kullanmak için temel bir kod çerçevesi sağlar.

Swagger, API geliştiricilerine, hızlı ve tutarlı bir şekilde API belgeleri oluşturarak, API'lerini diğer geliştiricilere daha anlaşılır bir şekilde sunma imkanı tanır. Bu da geliştirme sürecini daha etkili hale getirir ve API kullanıcılarının daha kolay ve doğru bir şekilde entegre olmalarını sağlar.

### 2.4.1 Avantajları

**Anlaşılır API Belgesi:** Swagger, açık ve standart bir belirleme kullanarak API'leri anlaşılır bir şekilde belgeleme imkanı sağlar. Bu, geliştiricilerin API'nin nasıl kullanılacağını hızlıca anlamalarına yardımcı olur.

**İstemci ve Sunucu Kod Üretimi:** Swagger Codegen, API belgesinden başlayarak istemci ve sunucu kodlarını otomatik olarak üretebilir. Bu, geliştiricilere başlangıç noktası olarak kullanabilecekleri bir temel sağlar.

**Kod Tutarsızlığı Önleme:** Swagger, API'nin belirli bir standartta tasarlanmasını ve belgelenmesini sağlar. Bu da kod tutarsızlıklarını önler ve geliştiricilerin aynı standartlara göre çalışmalarına yardımcı olur.

**API Güvenilirliği:** Doğru belgeleme, API'nin güvenilir bir şekilde kullanılmasını sağlar. Geliştiriciler, belgeler aracılığıyla API'nin özelliklerini, giriş ve çıkışlarını daha doğru bir şekilde anlayabilirler.

**Hızlı Prototipleme:** Swagger, API belgesini kullanarak hızlı prototipler oluşturmayı kolaylaştırır.

## 2.5 Redoc

Redoc, OpenAPI belgelerini görsel ve etkileşimli bir şekilde sunmak için kullanılan açık kaynaklı bir API belgeleme aracıdır. Redoc'un temel odak noktası, API belgelerini daha anlaşılır, kullanıcı dostu ve interaktif bir şekilde sunarak geliştiricilerin ve diğer kullanıcıların API'yi daha iyi anlamalarını sağlamaktır.

### 2.5.1 Avantajları

**Görsel Olarak Zengin Belgeleme:** Redoc, API belgelerini görsel olarak zenginleştirir. Renkli ve etkileşimli bir arayüzle kullanıcı dostu bir belgeleme sunar.

**Etkileşimli API Gezintisi:** Redoc, kullanıcılara API belgesi içinde etkileşimli bir şekilde gezinme imkanı tanır. Bu, belgeleme içinde hızlı ve kolay bir şekilde bilgi bulmalarını sağlar.

**Canlı Örnek Çalıştırma:** Redoc, API belgesindeki örnek istekleri doğrudan belgeleme üzerinden çalıştırma olanağı sunar. Bu, geliştiricilerin API'yi test etmelerini ve sonuçları görmelerini sağlar.

**Hızlı Tepki Süresi:** Redoc, hızlı bir kullanıcı deneyimi sunar. Sayfalar arasında hızlı geçişler ve API belgesinin anında yüklenmesi, geliştiricilere zaman kazandırır.

**Yüksek Derecede Özelleştirilebilirlik:** Redoc, belgeleme temasını ve stilini geniş bir şekilde özelleştirme imkanı sağlar. Bu, şirket içi tasarım standartlarına uygun belgeler oluşturmayı kolaylaştırır.

**Tamamen Uyumlu OpenAPI Standardı:** Redoc, OpenAPI standartlarına tamamen uyumlu bir şekilde çalışır. Bu, OpenAPI belgeleri ile sorunsuz entegrasyon sağlar.

**Swagger ve OpenAPI Desteği:** Redoc, Swagger 2.0 ve OpenAPI 3.0 belgelerini destekler. Bu, mevcut belgelerin kolayca Redoc'a entegre edilebileceği anlamına gelir.

**Otomatik Yenileme:** Redoc, belgeleme sayfalarını otomatik olarak yenileyebilir. API belgesinde yapılan değişikliklerin hızlı bir şekilde kullanıcılara yansıtılmasını sağlar.

**SEO Dostu:** Redoc, SEO dostu bir yapıda API belgelerini oluşturur. Bu, internet arama motorları tarafından daha iyi indekslenmeyi ve bulunabilirliği sağlar.

**Topluluk Desteği ve Gelişmiş Ekosistem:** Redoc, geniş bir geliştirici topluluğu tarafından desteklenir ve sürekli olarak geliştirilmektedir. Bu da yeni özelliklere hızlı erişim ve sorunların çözümü için güçlü bir destek anlamına gelir.

**Entegrasyon Kolaylığı:** Redoc, API belgesini düzenlerken ve güncellerken kolay bir entegrasyon süreci sunar. OpenAPI belgelerini veya Swagger şemalarını doğrudan içe aktarabilir ve hızla etkileşimli belgeler oluşturabilir.

**Üçüncü Taraf Hizmetlerle Uyum:** Redoc, çeşitli üçüncü taraf hizmetlerle uyumlu bir şekilde çalışabilir. Örneğin, API belgelerini GitHub üzerinde barındırabilir ve Redoc'u entegre ederek, otomatik olarak güncellenen ve sürekli iyileşen belgeler elde edebilirsiniz.

## 2.6 Orm

Uygulamalarımızda, geliştirdiğimiz projelerde veritabanına bağlanma ihtiyacı duyduğumuzda eskiden genellikle ADO.Net denilen, veritabanına doğrudan uygulama içerisinden bağlandığımız, veritabanı işlemlerini SQL sorgularıyla yaptığımız yöntem kullanılırdı. İlerleyen yazılım teknolojisi, nesne tabanlı programlama tabanlı dillerin de gelişmesi ve yaygınlaşması ile ADO.net'in karmaşıklığını basite indiren ve daha okunaklı kod satırları yazabildiğimiz ORM (Object Relational Mapping) yapıları ortaya çıktı.

Ado.net kullanılan projelerde veritabanı ile ilgili yapılan işlemlerde yazılımcı doğrudan sql sorguları yazmak durumundadır. ORM ile kod içerisine yazılan sql satırları ortadan kalkmıştır. Veritabanımız içerisinde yer alan tablolar bir sınıf (class), kolondaki alanlarımızın her biri değişken (property) olarak tanımlanmakta, veritabanındaki kayıtlara da ait olduğu sınıfta bir obje olarak erişebilmekte ve kullanabilmekteyiz.

Programlama dillerinin de kendine ait farklı ORM framework'leri bulunmaktadır.

**Dillere göre sık kullanılan ORM örnekleri;**

**C#:** Entity Framework, Dapper, ECO, XPO, Norm

**Java:** Hibernate, Ebean, Torque, JPA,MyBattis

**Php:** CakePHP, Codelgniter, RedBean, Doctrine,Propel, PdoMap

**Python:** Django, South,Storm

**ORM kullanmanın avantajları**

- Nesne tabanlı programlama standartlarına uygun olarak kod yazma imkanı verir.
- Minimum SQL bilgisi ile veritabanı işlemleri yapmak imkanı tanır.
- Veritabanı platformu bağımlılığı yoktur. Oracle kullanıyorken MSSQL geçişini sorunsuzca gerçekleştirebiliriz.
- Ado.net'e karşı daha güvenlidir. Sql Injection gibi bilinen saldırılara karşı güvenlik önlemleri vardır.
- Kod yazma süresini kısaltır.
- Kod okunabilirliğini arttırır.

**ORM kullanmanın dezavantajları**

- Ado.net'e kıyasla performans olarak daha yavaştır.
- Veritabanı nesneler üzerinden modellendiğinden nesneler arasında bağ bulunmaktadır.
- İlk kez başlayacak olanlar için yazım kuralları (syntax) farklı gelebilir.

## 2.7 Monolithic Mimari

Monolithic; mono/tek

Monolithic yaklaşım, bir sistemin/nesnenin/olgunun tek bir parça olacak şekilde tasarlanmasıdır. Monolithic mimari ise bu tasarımın stratejik yapılanmasıdır. Monolithic yaklaşım, üretilecek sistemin/nesnenin/olgunun bileşenlerini(component) birbirlerine bağlı(interdependent) olarak ve kendi kendine yetecek(self-contained) şekilde tasarlanmasını sağlayan ve böylece tek bir bütünsel varlık olarak nihai sonuca varılmasını sağlayan mimaridir.

Monolithic yaklaşımı benimsemiş uygulamaların tüm fonksiyonaliteleri tek bir çatı altında geliştirilirler.

Uygulama dünyasında günümüze kadar eşlik eden monolithic yaklaşım getirdiği büyük avantajların yanında aşağıda ele aldığımız gibi kritik dezavantajlar barındırmaktadır.

### 2.7.1 Avantajları

- Yönetilebilirliği, geliştirilebilirliği, bakımı ve monitoring'i(izleme) oldukça kolaydır.
- Küçük ve orta ölçekli projeler için geliştirilmesi hızlı ve maliyetsizdir.
- Component ve fonksiyonlar çalışma açısından kendi aralarında tutarlı ilişki kurabilmektedirler.
- Transaction yönetimi oldukça rahat ve irade altındadır.

### 2.7.2 Dezavantajları

Tüm hizmetler tek bir uygulama üzerinden sunulmaktadır. Böylece herhangi bir noktada düzeltme yahut geliştirme yapılması gerektiği taktirde uygulama baştan sona tekrar derlenmesi gerekmekte ve böylece uygulamanın varsa yayın durumu kısmi kesintilere gireceği anlamına gelmektedir.

Tüm bileşenler bütünsel bir parça içerisinde tek bir bütün olarak değerlendirilmektedir. Bu durumda bir noktada yapılan çalışmanın alakasız başka bir noktayla olan teması yüzünden bloklanması ve süreçten etkilenmesi demektir.

Takım çalışmalarından birden fazla kişi tarafından geliştirilen uygulamalarda ister istemez birçok kod ve yapı karmaşası meydana gelmektedir. Misal; ameliyat masasında mideden ameliyat olan bir hastaya biryandan da diş doktoru tarafından dolgu yapılması ne kadar sıkıntılı bir süreçse projelerde de benzer sıkıntılı süreçler ve gerginlikler yaşanabilmektedir.

Uygulama tek çatı altında geliştirileceği için tüm component ve modüller kendi aralarında sıkı bağlılık göstereceklerdir.

## 2.8 Microservice Mimari

Microservice mimarilerine odaklanabilmek için öncelikle monolithic yaklaşımın temel bir prensibi çiğnediğinin farkında olunması gerekmektedir. Bu prensip sürdürülebilirliktir ilkesidir. Sürdürülebilirlik; bir yazılımın, üzerinde yapılan tüm değişiklik yahut onarma faaliyetleri esnasında bile verdiği hizmetin bütününde bir aksaklık olmaması ve sistemin kesintiye uğramaksızın her an çalışabilir vaziyette olması demektir.

Microservice mimari, birbirinden bağımsız olarak çalışan ve birbirleriyle haberleşerek bir bütün olarak hareket eden servis(ler) yapılanmasıdır. Her servisin bir diğerinden bağımsız olarak iş mantığını yürütmesi ve bir başka servis ile ilgilenmemesi, bir onarım yahut restorasyon durumunda uygulamanın bütününü etkilemeyeceğinden dolayı sürdürülebilirlik ilkesi desteklenmiş olacak ve böylece bodoslama olarak tabir edeceğimiz monolithic yaklaşımındaki karmaşıklığı ortadan kaldırmış ve yönetimi daha da kolaylaştırmış olacaktır.

Burada esas olan, her bir servisin bir diğerinden bağımsız olmasının geliştirme ve düzenleme operasyonlarında getirisidir. Örneğin; Bir e-ticaret uygulamasında ürün işlemlerinin, sepetin ve ödeme sisteminin ayrı servisler tarafından gerçekleştirildiğini düşünürsek, süreçte ödeme sistemindeki oluşan herhangi bir aksaklık yahut restorasyon sistemin bütününü değil sadece o servisi etkileyeceğinden dolayı haliyle sadece o servisle ilgilenilmesi yeterli olacaktır. Bu durumda sistem bütünsel olarak işlevselliğe devam edecek lakin kesintiye sadece ilgili servis uğramış olacaktır. Haliyle bizler sistemin kendisinden ziyade local olarak sadece tek bir servisi ile ilgilenerek gerekli onarımı sağlayabilir, hızlıca testlere tabii tutabilir ve monolithic yapılanmalarda olduğu gibi uygulamayı topyekün derleme ve yayınlama ihtiyacını duymaksızın kısa zamanda yeni sürümle hizmete devam edebiliriz.

### 2.8.1 Avantajları

Uygulama boyutundan bağımsız olmak üzere yeni bir özelliğin eklenmesi yahut mevcutiyetin bakımı sadece ilgili servislerle ilgilendirme gerektireceğinden dolayı oldukça kolaydır.

Ekip çalışmasına yatkındır. Özellikle ekibe yeni katılım gösteren arkadaşların devasa bir proje ve kod içerisinde kaybolmasının önüne geçmekte, sadece ilgileneceği servisin kaynağını çözümlemesi gerekmektedir.

Her bir service ihtiyaca binaen farklı dil ve platformda yazılabilmektedir.

Versiyon yönetimi oldukça kolaydır.

### 2.8.2 Dezavantajları

Birden fazla service ve birden fazla veritabanı söz konusu olacağı için transaction yönetimi zorlaşacaktır.Servislerin yönetilebilirliği ve monitoringi zorlaşacaktır.

## 2.9 Javascript

JavaScript, geliştiricilerin etkileşimli web sayfaları oluşturmak için kullandığı bir programlama dilidir. JavaScript işlevleri, sosyal medya akışlarını yenilemekten animasyonlar ve etkileşimli haritalar göstermeye kadar, bir web sitesi kullanıcısının deneyimini iyileştirebilir. İstemci tarafındaki bir betik dili olarak, World Wide Web'in temel teknolojilerinden biridir. Örneğin, internette gezinirken bir görsel döngüsü, görmek için tıkla açılır menüsü ya da bir web sayfasında dinamik olarak değişen öğe renkleri gördüğünüzde JavaScript efektlerini görmüş olursunuz.

### 2.9.1 Ne için kullanılır?

Tarihsel olarak web sayfaları bir kitaptaki sayfalara benzer şekilde statikti. Statik bir sayfa bilgileri çoğunlukla sabit bir düzende görüntülüyordu ve şimdi modern bir web sitesinden beklediğimiz her şeyi yapmıyordu. JavaScript, web uygulamalarını daha dinamik hale getirmek için tarayıcı bazlı bir teknoloji olarak ortaya çıktı. Tarayıcılar JavaScript kullanarak kullanıcı etkileşimine yanıt verebilir ve web sayfasındaki içerik düzenini değiştirebilir.

Dil olgunlaştıkça JavaScript geliştiricileri kitaplıklar, çerçeveler ve programlama uygulamaları oluşturdu ve bunları web tarayıcılarının dışında kullanmaya başladı. Bugün JavaScript'i hem istemci tarafı hem de sunucu tarafı geliştirme için kullanabilirsiniz.

Tarihsel olarak web sayfaları bir kitaptaki sayfalara benzer şekilde statikti. Statik bir sayfa bilgileri çoğunlukla sabit bir düzende görüntülüyordu ve şimdi modern bir web sitesinden beklediğimiz her şeyi yapmıyordu. JavaScript, web uygulamalarını daha dinamik hale getirmek için tarayıcı bazlı bir teknoloji olarak ortaya çıktı. Tarayıcılar JavaScript kullanarak kullanıcı etkileşimine yanıt verebilir ve web sayfasındaki içerik düzenini değiştirebilir.

### 2.9.2 Nasıl çalışılır ?

Tüm programlama dilleri, İngilizce benzeri söz dizimini işletim sisteminin çalıştıracağı makine koduna çevirerek çalışır. JavaScript, genel olarak bir betik dili veya yorumlanmış bir dil olarak sınıflandırılır. JavaScript kodu yorumlanır, yani bir JavaScript motoru tarafından doğrudan temeldeki makine dili koduna çevrilir. Diğer programlama dillerinde bir derleyici tüm kodu ayrı bir adımda makine kodunda derler. Bu nedenle, tüm betik dilleri programlama dilleridir ancak tüm programlama dilleri betik dili değildir.

JavaScript motoru, JavaScript kodunu çalıştıran bir bilgisayar programıdır. İlk JavaScript motorları yalnızca yorumlayıcılardı ancak tüm modern motorlar performansı artırmak için tam zamanında veya çalışma zamanı derlemesi kullanır.

### 2.9.3 İstemci tarafı Javascript

İstemci tarafı JavaScript, JavaScript'in tarayıcınızda çalışma şeklini ifade eder. Bu durumda JavaScript motoru tarayıcı kodunun içindedir. Tüm büyük web tarayıcıları kendi yerleşik JavaScript motorlarıyla birlikte gelir.

Web uygulaması geliştiricileri, fare tıklaması veya fareyle üzerine gelme gibi çeşitli olaylarla ilişkili farklı işlevlerle JavaScript kodu yazar. Bu işlevler HTML ve CSS'de değişiklikler yapar.

İstemci tarafı JavaScript'in nasıl çalıştığına dair bir genel bakış:

- Tarayıcı, ziyaret ettiğinizde bir web sayfası yükler.
- Yükleme sırasında tarayıcı, sayfayı ve düğmeler, etiketler ve açılır kutular gibi tüm öğelerini Belge Nesne Modeli (DOM) adı verilen bir veri yapısına dönüştürür.
- Tarayıcının JavaScript motoru, JavaScript kodunu bayt koduna dönüştürür. Bu kod, JavaScript söz dizimi ile makine arasında bir aracıdır.
- Bir düğmeye fare tıklaması gibi farklı olaylar, ilişkili JavaScript kod blogunun yürütülmesini tetikler. Motor daha sonra bayt kodunu yorumlar ve DOM'da değişiklikler yapar.

### 2.9.4 Sunucu tarafı Javascript

Sunucu tarafı JavaScript, arka uç sunucu mantığında kodlama dilinin kullanımını ifade eder. Bu durumda JavaScript motoru doğrudan sunucuya oturur. Sunucu tarafı JavaScript işlevi veri tabanına erişebilir, farklı mantıksal işlemler gerçekleştirebilir ve sunucunun işletim sistemi tarafından tetiklenen çeşitli olaylara yanıt verebilir. Sunucu tarafı komut dosyası oluşturmanın birincil avantajı, web sitesi yanıtını gereksinimlerinize, erişim haklarınıza ve web sitesinden gelen bilgi isteklerine göre büyük ölçüde özelleştirebilmenizdir.

### 2.9.5 İstemci tarafına karşı sunucu tarafı

Dinamik kelimesi hem istemci tarafı hem de sunucu tarafı JavaScript'i tanımlar. Dinamik davranış, gerektiğinde yeni içerik oluşturmak için web sayfası görüntüsünü güncelleme yeteneğidir. İstemci tarafı ve sunucu tarafı JavaScript arasındaki fark, yeni içerik oluşturma biçiminde yatmaktadır. Sunucu tarafı kodu uygulama mantığını kullanarak ve veri tabanındaki verileri değiştirerek dinamik olarak yeni içerik oluşturur. İstemci tarafı JavaScript ise, kullanıcı arayüzü mantığını kullanarak ve zaten istemcide bulunan web sayfası içeriğini değiştirerek tarayıcı içinde dinamik olarak yeni içerik oluşturur. İki bağlamda anlam biraz farklıdır ancak ilişkilidir ve her iki yaklaşım da kullanıcı deneyimini geliştirmek için birlikte çalışır.

Dinamik özelliklerdeki uygulama dışında, iki JavaScript kullanımı arasındaki diğer bir fark, kodun erişebileceği kaynaklardır.

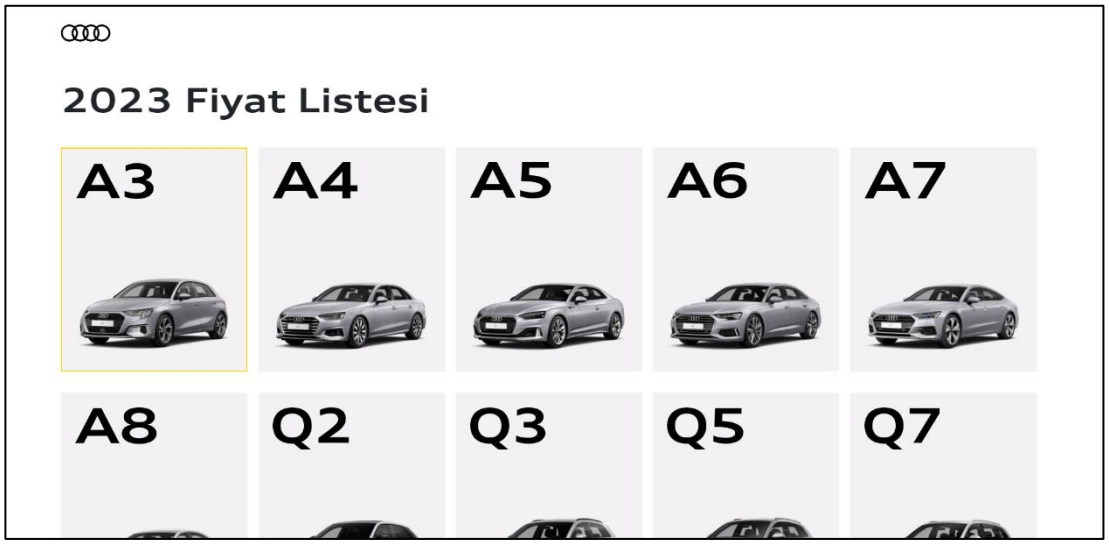
# Bölüm 3

## Bulgular

Uygulama araç listeleme ve araç detay olmak üzere 2 kısımdan oluşmaktadır.Araç detay sayfası araç listeleme sayfasına göre daha detaylı bilgi içermektedir.

### 3.1 Araç Listeleme

Uygulamanın default sayfası araç listeleme sayfasıdır.Araç listeleme sayfası parametre olarak marka değeri almaktadır.Marka bilgisi ise uygulamada default değeri audi olarak verilmektedir.Araç listeleme sayfasında güncel yıla göre markaya ait araç modelleri listelenmektedir.




Şekil 3.1: Araç liste sayfası

Bu sayfanın amaçlarından biride kullanıcının en çok tercih ettiği ve baktığı araç modelini en başta ve sarı bir renkte çerçeveleyerek göstermektedir.Yukarıdaki örnekte kullanıcının en çok incelediği araç modeli a3 olduğu için a3 en başta ve çerçeveli bir şekilde gösterilmektedir.

Tercih edilme işleminin akışı araç modelinin detayına girdiği zaman bildirilmektedir ve her ana sayfaya girildiğinde tekrar sorgulanmaktadır.Uygulamada bir kullanıcı girişi olmadığı için bilgiler ip adresi üzerinden tutulmaktadır.Sayfalarda ilgili anasayfasına yönlendirmek için linkler , araç logusu ve araçlarla ilgil bilgiler bulunmaktadır.Bu bilgiler ise aracın marka değerine göre değişmektedir.

Bir card tasarımında araç modeli ve araç modeline uygun bir şekilde resim bilgisi bulunmaktadır.Araç modelleri ve resimleri dinamik bir şekilde gelmekte ve tasarıma giydirilmektedir.

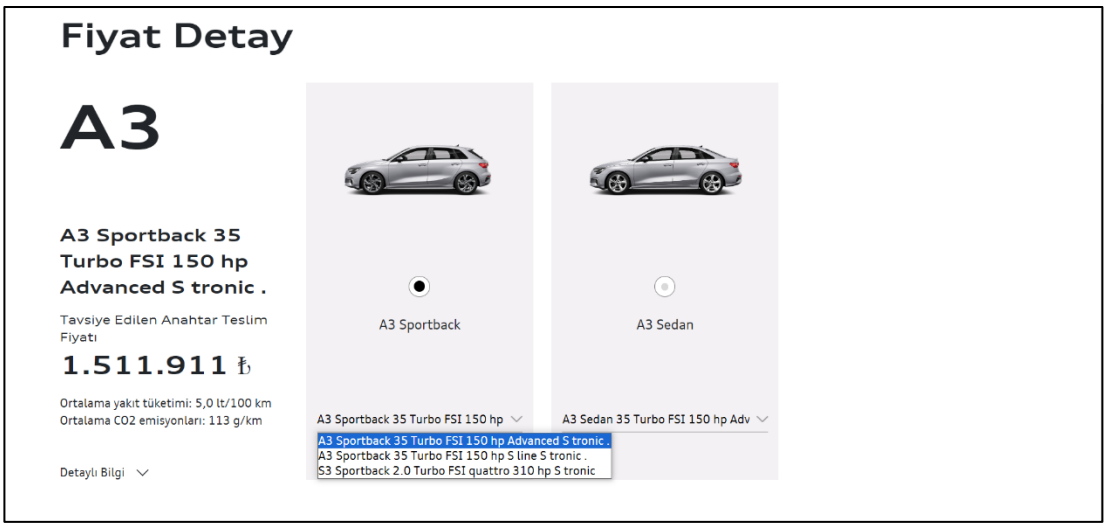
## 3.2 Araç Detayı

Araç logosu markanın ana sayfasına yönlendirirken modeller butonu ise araç modellerine yani ana sayfaya yönlendirmektedir.Araç detay sayfasının başında filtreleme alanı mevcuttur.

Bu alanda araç tipi , net fiyat , ötv ve kdv değerlerine göre arama yapabilmektedir.Araç tipi bilgisi detaya girilen modele göre dinamik bir şekilde gelmektedir.Erişmek istediğimiz modele hızlı bir şekilde ulaşabilmek için filtreleme alanını kullanabiliriz.

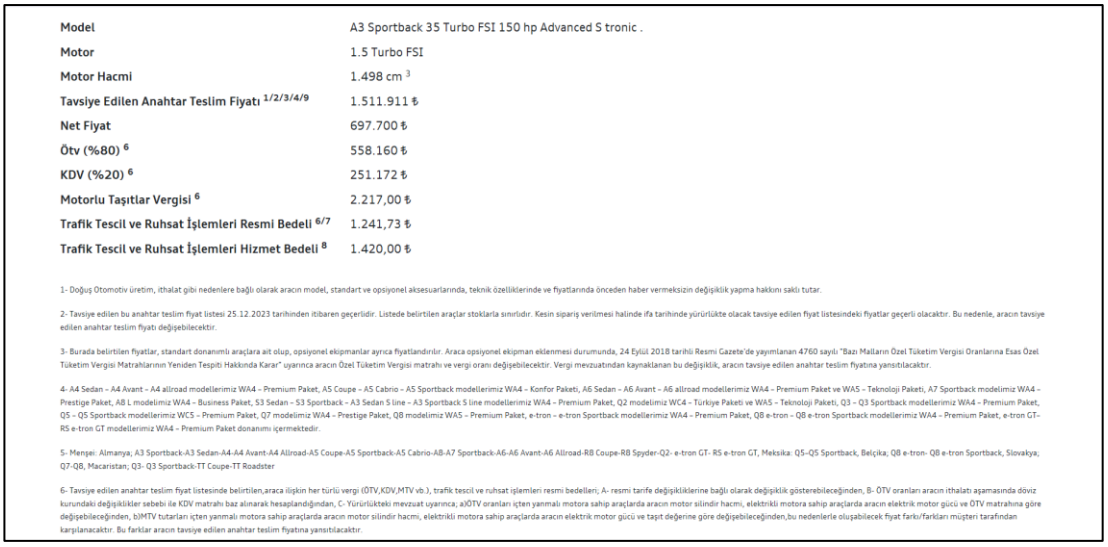

Şekil 3.2: Araç detay sayfası - 1

Araç detayında ilgili modelin tipleri ve tiplere ait alt modeller bulunmaktadır.Seçilen tipin alt modeline göre fiyat bilgisi , model tanımı , yakıt tüketimi ve emisyon verileri değişmektedir.Ayrıca detaylı bilgi kısmında ilgili markanın araç bilgileri için kuralları bulumaktadır.




Şekil 3.3: Araç detay sayfası - 2

Detaylı bilgi kısmında hem markanın bilgileri hem de araç için bulunan bilgiler bulunmaktadır.



| | |
|---|---|
| **Model** | A3 Sportback 35 Turbo FSI 150 hp Advanced S tronic . |
| **Motor** | 1.5 Turbo FSI |
| **Motor Hacmi** | 1.498 cm $^3$ |
| **Tavsiye Edilen Anahtar Teslim Fiyatı** [1/2/3/4/9] | 1.511.911 ₺ |
| **Net Fiyat** | 697.700 ₺ |
| **Ötv (%80)** [6] | 558.160 ₺ |
| **KDV (%20)** [6] | 251.172 ₺ |
| **Motorlu Taşıtlar Vergisi** [6] | 2.217,00 ₺ |
| **Trafik Tescil ve Ruhsat İşlemleri Resmi Bedeli** [6/7] | 1.241,73 ₺ |
| **Trafik Tescil ve Ruhsat İşlemleri Hizmet Bedeli** [8] | 1.420,00 ₺ |

1- Doğuş Otomotiv üretim, ithalat gibi nedenlere bağlı olarak aracın model, standart ve opsiyonel aksesuarlarında, teknik özelliklerinde ve fiyatlarında önceden haber vermeksizin değişiklik yapma hakkını saklı tutar.

2- Tavsiye edilen bu anahtar teslim fiyat listesi 25.12.2023 tarihinden itibaren geçerlidir. Listede belirtilen araçlar stoklarla sınırlıdır. Kesin sipariş verilmesi halinde ifa tarihinde yürürlükte olacak tavsiye edilen fiyat listesindeki fiyatlar geçerli olacaktır. Bu nedenle, aracın tavsiye edilen anahtar teslim fiyatı değişebilecektir.

3- Burada belirtilen fiyatlar, standart donanımlı araçlara ait olup, opsiyonel ekipmanlar ayrıca fiyatlandırılır. Araca opsiyonel ekipman eklenmesi durumunda, 24 Eylül 2018 tarihli Resmi Gazete'de yayımlanan 4760 sayılı "Bazı Malların Özel Tüketim Vergisi Oranlarına Esas Özel Tüketim Vergisi Matrahlarının Yeniden Tespiti Hakkında Karar" uyarınca aracın Özel Tüketim Vergisi matrahı ve vergi oranı değişebilecektir. Vergi mevzuatından kaynaklanan bu değişiklik, aracın tavsiye edilen anahtar teslim fiyatına yansıtılacaktır.

4- A4 Sedan – A4 Avant – A4 allroad modellerimiz WA4 – Premium Paket, A5 Coupe – A5 Cabrio – A5 Sportback modellerimiz WA4 – Konfor Paketi, A6 Sedan – A6 Avant – A6 allroad modellerimiz WA4 – Premium Paket ve WA5 – Teknoloji Paketi, A7 Sportback modelimiz WA4 – Prestige Paket, A8 L modelimiz WA4 – Business Paket, S3 Sedan – S3 Sportback – A3 Sedan S line – A3 Sportback S line modellerimiz WA4 – Premium Paket, Q2 modelimiz WC4 – Türkiye Paketi ve WA5 – Teknoloji Paketi, Q3 – Q3 Sportback modellerimiz WA4 – Premium Paket, Q5 – Q5 Sportback modellerimiz WC5 – Premium Paket, Q7 modelimiz WA4 – Prestige Paket, Q8 modelimiz WA5 – Premium Paket, e-tron – e-tron Sportback modellerimiz WA4 – Premium Paket, Q8 e-tron – Q8 e-tron Sportback modellerimiz WA4 – Premium Paket, e-tron GT- RS e-tron GT modellerimiz WA4 – Premium Paket donanımı içermektedir.

5- Menşei: Almanya; A3 Sportback-A3 Sedan-A4-A4 Avant-A4 Allroad-A5 Coupe-A5 Sportback-A5 Cabrio-A8-A7 Sportback-A6-A6 Avant-A6 Allroad-RB Coupe-RB Spyder-Q2- e-tron GT- RS e-tron GT, Meksika: Q5-Q5 Sportback, Belçika; Q8 e-tron- Q8 e-tron Sportback, Slovakya; Q7-Q8, Macaristan; Q3- Q3 Sportback-TT Coupe-TT Roadster

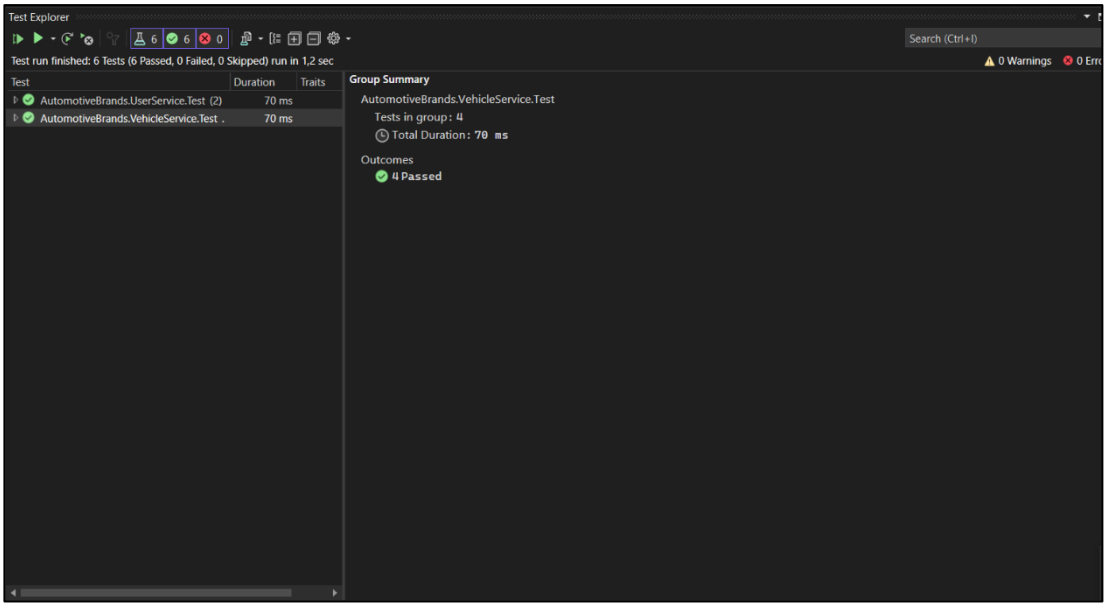
6- Tavsiye edilen anahtar teslim fiyat listesinde belirtilen,araca ilişkin her türlü vergi (ÖTV,KDV,MTV vb.), trafik tescil ve ruhsat işlemleri resmi bedelleri; A- resmi tarife değişikliklerine bağlı olarak değişiklik gösterebileceğinden, B- ÖTV oranları aracın ithalatı aşamasında döviz kurundaki değişiklikler sebebi ile KDV matrahı baz alınarak hesaplandığından, C- Yürürlükteki mevzuat uyarınca; a)ÖTV oranları içten yanmalı motora sahip araçlarda aracın motor silindir hacmi, elektrikli motora sahip araçlarda aracın elektrik motor gücü ve ÖTV matrahına göre değişebileceğinden, b)MTV tutarları içten yanmalı motora sahip araçlarda aracın motor silindir hacmi, elektrikli motora sahip araçlarda aracın elektrik motor gücü ve taşıt değerine göre değişebileceğinden,bu nedenlerle oluşabilecek fiyat farkı/farkları müşteri tarafından karşılanacaktır. Bu farklar aracın tavsiye edilen anahtar teslim fiyatına yansıtılacaktır.

Şekil 3.4: Araç detay sayfası - 3

## 3.3 Test

Projemizdeki uygulamaları test edebilmek için öncelikle proje kodlarının test yazılmaya uygun olması gerekmektedir.Yazılan kodların daha doğru ve hatasız çalışabilmesi için unit test yazmamız gerekmektedir.Unit test yazabilmek için projemizde mock kütüphanesini kurmamız gerekmektedir.

Unit test , projede mikroservisler için yazılmıştır.Toplamda 6 servise test yazılmış ve 70 ms sürede gerçekleştirilmiştir.




Şekil 3.5: Unit test projesi

# Bölüm 4

## Tartışma

Uygulamanın başarısı , kullanıcı geri bildirimlerine göre değerlendirilmiş ve benzer projelerle birlikte karşılaştırma işlemi yapılmıştır.Araç listeleme sayfası benzer projelere göre daha fazla göz alıyor ve tasarımı oldukça hoş.Tasarımsal olarak hareket edilmesi kullanıcı gözünde oldukça iyi bir izlenim bırakmıştır.

Uygulamayı geliştirirken teknik anlamda bazı zorluklar meydana gelmiş fakat onlar uygulamanın temelini etkilemediği için yapılmamıştır.Bunlardan birine örnek vermek gerekirse swagger entegrasyonu aslında ocelot için yapılması gerekirken tek tek mikroservisler için yapılmıştır.

Swagger api dokümantasyonu için önemli de olsa uygulamanın kullanımını etkilememektedir.Araştırmalarımda karşılaştığım sorunla alaklı bir çözüm bulunamamıştır.

Klasör mimarisi bir uygulama için oldukça önemlidir.Bunun nedeni projeyi anlamak , geliştirme yapmak ve temiz kodlamak için olmazsa olmazlarındandır.Projeye başlarken klasör mimarisi kurmak oldukça zaman almıştır.

Klasör mimarisi içerisinde bulunan **libraries** klasörü daha önce bahsettiğim gibi uygulamaların ortak ihtiyaçlarını gidermek için oluşturulmuştur.Örneğin validasyon sadece mikroservisler için değil **client** uygulamalar içinde kullanılmaktadır.

# Bölüm 5

## Sonuç

Proje genel olarak markaların araçları hakkında bilgilerini detaylı bir şekilde gösterebileceğimiz bir uygulamadır.Projemiz daha çok tasarım yönüyle ön plana çıktığı için oldukça kullanıcı dostudur.Araç fiyatları , yakıt tüketimi vb gibi bilgileri anlık olarak görebileceğimiz bir uygulamadır.

Proje sadece araç modelleriyle alakalı bilgi vermemekte markanın belirlemiş oldukları kurallarıda araç detay sayfasında görebiliriz.Aracı almadan önce kullanıcının bilmesi gereken bilgiler burda bulunmaktadır.

Projede yapılsa güzel olur dediğim geliştirmeler bulunmakta.Özellikle içerik girişleri için dinamik bir cms tabanlı bir uygulama geliştirilebilir.Böylelikle geliştirici olan kişinin veritabanına manuel olarak veri girmesi yerine kullanıcı dostu olan cms uygulaması tercih edilerek veri girme işlemlerini yapabiliriz.

Api gateway projesine ocelot kullanarak nasıl swagger entegre edebiliriz bunlara bakılabilir.Böylelikle her mikroservis projesinin swagger'ına gitmeden de gateway için api dokümantasyonunu görebiliriz.(Ocelot, 2024)

# Kaynaklar


Audi Türkiye (2023, Ocak 1). *Ana sayfa*

www.audi.com.tr/tr/web/tr.html

Vikipedi Audi (2016, Ocak 1). *Audi*

https://tr.wikipedia.org/wiki/Audi

Microsoft .NET (2023, Ocak 1). *.NET Belgeleri*

https://learn.microsoft.com/tr-tr/dotnet/

Swagger (2024, Ocak 1). *Ana sayfa*

https://swagger.io/

Ocelot (2024, Ocak 1). *Ana sayfa*

https://ocelot.readthedocs.io/en/latest/introduction/gettingstarted.html