ARAÇ ARIZALARI İÇİN DİĞİTAL KAYIT VE TAKİP SİSTEMİ

Yazılım Mühendisliği Ana Bilim Dalı

Dönem Projesi

Büşra Uçar Tekinay

ORCID 0000-0000-0000-0000

Proje Danışmanı: Prof. Dr. Doğan Aydın

Şubat 2024

ARAÇ ARIZALARI İÇİN DİĞİTAL KAYIT VE TAKİP SİSTEMİ

Öz

Bu proje, araç sahipleri ve servis uzmanları için araç arızalarını daha etkili bir şekilde kaydetmelerini ve yönetmelerini sağlamak amacıyla geliştirilen bir web tabanlı uygulamayı içermektedir. Sistem, ASP.NET MVC (Model-View-Controller) mimarisi kullanılarak tasarlanmıştır ve kullanıcı dostu bir web sitesi ile yönetici panelini bir araya getirerek araç bakım ve onarım süreçlerini optimize etmeyi hedeflemektedir. Bu model, Model-View-Controller mimarisini (MVC) temel alır ve güvenlik, form oluşturma ve doğrulama, veritabanı erişimi ve yönlendirme gibi başka birçok yararlı bileşene sahiptir. Admin paneli, yöneticilere özel kontroller ve raporlama araçları sunarak sistemi etkin bir şekilde yönetmelerini sağlar.

Anahtar Sözcükler: ASP.NET MVC, C#, SQL, Entity Framework, Araç Arıza Takip

Digital Tracking and Recording System for Vehicle Malfunctions

Abstract

This project includes a web-based application developed for vehicle owners and service professionals to enable them to record and manage vehicle faults more effectively. The system is designed using ASP.NET MVC (Model-View-Controller) architecture and aims to optimize vehicle maintenance and repair processes by combining a user-friendly website and admin panel. This model is based on the Model-View-Controller architecture (MVC) and has many other useful components such as security, form creation and validation, database access and routing. The admin panel provides administrators with special controls and reporting tools, allowing them to manage the system effectively.

Keywords: ASP.NET MVC, C#, SQL, Entity Framework, Vehicle Fault Tracking

İçindekiler

Şekiller Listesi

Kısaltmalar Listesi

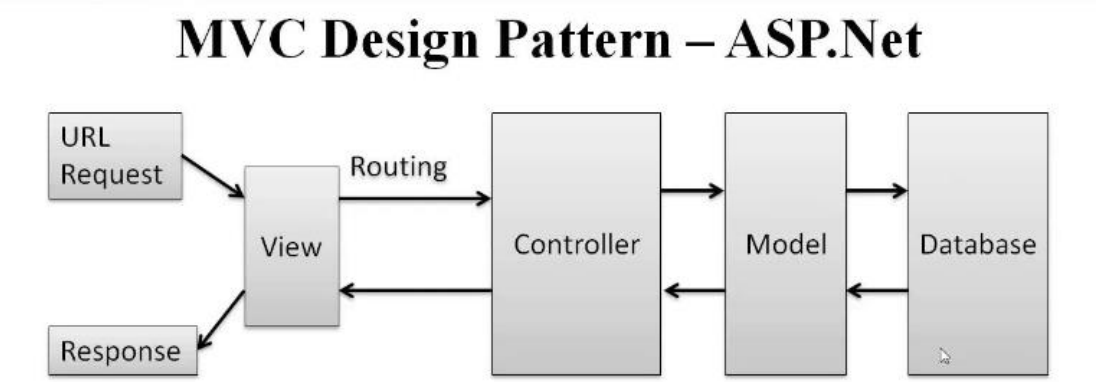
ASP	Active Server Pages
MVC	Model-Controller-View
VS	Visual Studio
SQL	Structured Query Language
OOP	Object Orianted Programming
EF	Entity Framework
ORM	Object Relational Mapping

Bölüm 1

Giriş

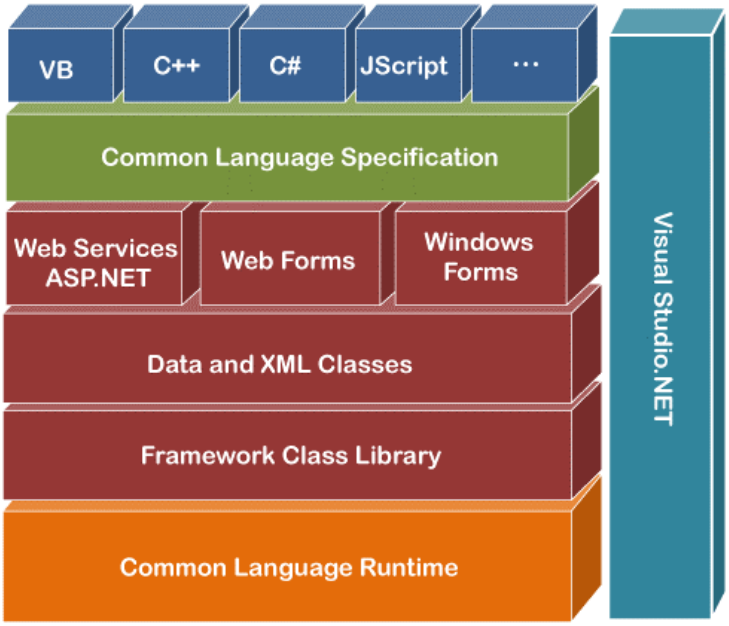
Araba arızaları, aracın çeşitli parçalarında ya da çalışma sisteminde meydana gelen birtakım problemlerdir. Hem trafik güvenliğini tehlikeye atmamak hem de sürücülerin can güvenliğini sağlamak adına araçta herhangi bir arıza fark edildiği takdirde bunların mutlaka kontrol altına alınması gerekir. Bazı araba arızalarının giderilmesini araç sahipleri kendileri yapabilir ancak yolda kalınabilecek araç arızalarında uzman bir ekipten yardım alınması önemlidir. Uzman ekibi bulmak için de internette web sitelerinden yararlanılmaktadır. Bu çalışmada da bu web sitesine bir örnek yapılmıştır. ASP.NET MVC(Model-View-Controller) mimarisi kullanılarak panel destekli web sitesi yapılmıştır. ASP.Net, Microsoft tarafından sağlanan bir web geliştirme platformudur, Web tabanlı uygulamalar oluşturmak için kullanılır. ASP.NET MVC Framework uygulamalarının sağladığı en önemli özellikler; daha sade html içeriğe ve URL Routing (URL yönlendirme) ile daha anlaşılabilir URL adreslerine sahip, test edilebilir, kolay erişilebilir uygulamalar tasarlanabilmesidir. ASP.Net, standart HTTP protokolüyle çalışacak şekilde tasarlanmıştır. ASP.NET mimarisi, güçlü bir olay odaklı programlama modeli sağlamak için .NET Framework ile iyi bir şekilde birleştirilmiştir. ASP.NET mimarisinde. NET Framework işletim sistemiyle birlikte çalışır. Bir web istemcisi, bir veritabanı, Web Hizmeti, COM bileşeni veya bileşen sınıfı dahil olmak üzere tüm ek kaynakları birleştiren Internet Information Server (IIS) aracılığıyla teslim edilen bir Web Formu (ASPX) kaynağı ister. Bunların tümü, IIS'nin Web kökü içindeki bin dizininde bulunan Web uygulamasından derlenmiş bir derleme (DLL) aracılığıyla teslim edilir(Mesbah et al., 2002). ASP.NET, çözümlerde oluşturulan farklı sayfa türleri için bazı yeni dosya uzantıları içerir. Yeni uzantılar, ASP.NET'in hiçbir dosya adı çakışması olmadan aynı sunucuda ASP 3.0'ın yanında yer almasına olanak tanır. Bu sayfa türlerinin her birinin program mantığının

saklandığı bir arka plan kodu sayfası vardır. ASP.NET'teki yeni özelliklerden biri, sunucudaki kökten kaynaklanan JavaScript (.is) dosyalarından sayfalara özel doğrulamanın yerleştirilmesi yeteneğidir; ASP.NET'in kullanabileceği bir dizi doğrulama kontrolü vardır.



Şekil 1.1: MVC Tasarım Deseni

.NET Framework, Microsoft tarafından geliştirilen, açık İnternet protokolleri ve standartları üzerine kurulmuş bir "uygulama" geliştirme platformudur. NET Framework, uygulamaların çalıştırılması için bir çalışma ortamı sağlayan birkaç katmandan oluşan genel bir yapıya sahiptir. Bu katmanlar, Common Language Runtime (CLR), Framework Class Library (FCL) ve web uygulamaları için ASP.NET'i içerir(Milroy et al., 2002). NET Framework, uygulamaların çalıştırılması için bir çalışma ortamı sağlayan birkaç katmandan oluşan genel bir yapıya sahiptir. Bu bileşenler, standart bir dizi kurallar ve protokoller kullanarak birlikte çalışır. CLR, herhangi bir .NET dilinde yazılmış kodu yürütmekten sorumludur, FCL ise tüm .NET dillerinde kullanılabilen ortak işlevselliği sağlar(Milroy et al., 2002). Geliştirme araçları, geliştiricilerin .NET uygulamalarını kolayca oluşturmasına, hata ayıklamasına ve dağıtmasına olanak tanıyan bir entegre geliştirme ortamı (IDE) sağlar. Ek olarak, ASP.NET bileşeni, web uygulamaları oluşturmak için kullanılır ve herhangi bir .NET dili ile kullanılabilir. Sonuç olarak, .NET Framework'ün çeşitli bileşenleri, geniş bir uygulama yelpazesi oluşturmak için güçlü ve esnek bir geliştirme platformu sağlamak için birlikte çalışır.



Şekil 1.2 – .NET Mimarisi

.NET Framework mimarisi, bir dizi temel bileşen içerir .NET Framework mimarisinin ana bileşenleri:

i. Common Language Runtime (CLR): .NET Framework'ün temel bileşenlerinden biridir. CLR, .NET uygulamalarının çalışma zamanını sağlar. Just-In-Time (JIT) derleme ile çalışır, yani kaynak kodlar çalışma anında makine diline çevrilir. Bellek yönetimi, güvenlik, hata işleme ve diğer temel özellikleri sağlar(Milroy et al., 2002).

ii. Framework Class Library (FCL): .NET Framework'ün temel sınıf ve yöntem koleksiyonunu içerir. Geniş bir sınıf seti sunar ve geliştiricilere genel görevleri daha hızlı bir şekilde gerçekleştirmeleri için araçlar sağlar. GUI uygulamalarından veri tabanı işlemlerine kadar birçok farklı alanda kullanılır(Milroy et al., 2002).

iii. Common Language Specification(CLS): .NET Framework içinde, dil bağımsızlığı ve uyumluluk sağlamak amacıyla kullanılır. .NET Platformunda ki farklı dilelrin birbirleriyle etkileşimde bulunmalarını sağlamak için belirli bir dil setini standardize eder(Milroy et al., 2002). Bu standartlar, .NET dilleri arasında bir uyumluluk seviyesi sağlar, böylece Farklı dillerde yazılmış kütüphaneler birbirleriyle uyumlu bir şekilde kullanılabilir.

iv. ASP.NET: ASP.NET, web uygulamaları geliştirmek için kullanılan bir bileşendir. Web Forms ve MVC (Model-View-Controller) gibi farklı yaklaşımları destekler(Freeman ve Sanderson, 2011). Sunucu tarafında çalışan, dinamik ve etkileşimli web siteleri oluşturmak için kullanılır.

v. Windows Forms: Windows Forms, masaüstü uygulamaları geliştirmek için kullanılan bir bileşendir. GUI uygulamalarının tasarlanması ve geliştirilmesi için kullanılır(Milroy et al., 2002).

vi. Windows Presentation Foundation (WPF): WPF, zengin ve görsel olarak çekici masaüstü uygulamaları oluşturmak için kullanılır. Grafiksel olarak zengin kullanıcı arayüzleri tasarlamak için XAML (eXtensible Application Markup Language) kullanır(http://msdn.microsoft.com/en-us/library/zw4w595w(v=VS.100).aspx).

vii. ADO.NET: ADO.NET, veritabanlarına erişim sağlamak için kullanılan bir bileşendir. Bağlantı (Connection), Komut (Command), Veri Okuma (DataReader), Veri Seti (DataSet) gibi sınıfları içerir(Milroy et al., 2002).

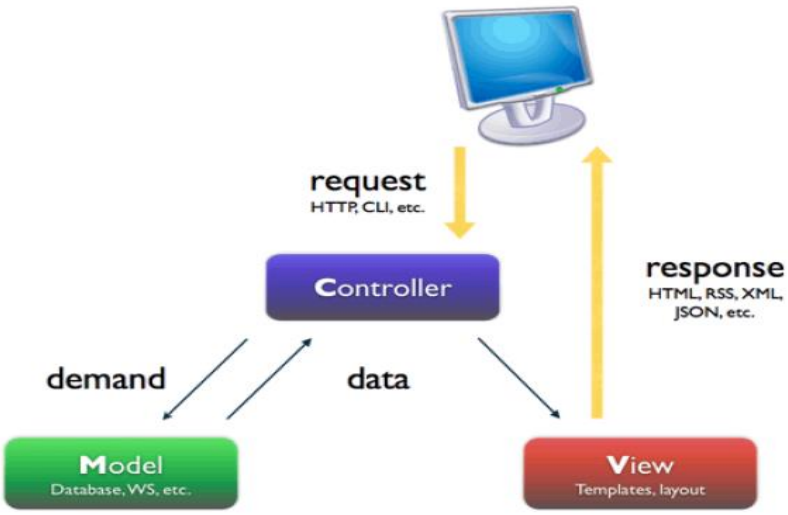
viii. Language Integrated Query (LINQ): LINQ, .NET platformuna entegre edilmiş bir sorgu dilidir. Veri kaynakları üzerinde sorgular yazmayı ve işlemeyi sağlar.

ix. Entity Framework: Entity Framework, veritabanı etkileşimini soyutlayan ve nesne yönelimli programlamayı destekleyen bir ORM (Object-Relational Mapping) framework'üdür(Krasner ve Pope, 1988).

Bu bileşenler, .NET Framework'ün temelini oluşturur ve geliştiricilere farklı uygulama türlerini geliştirmek için geniş bir araç seti sunar. Bu bileşenler, .NET uygulamalarının geliştirilmesini, dağıtılmasını ve yönetilmesini kolaylaştırır.

MVC deseni, 3 katmandan oluşmaktadır ve katmanları birbirinden bağımsız (birbirini etkilemeden) olarak çalışmaktadır. Bu sebeple çoğunlukla büyük çaplı projelerde projelerin yönetiminin ve kontrolünün daha rahat sağlanabilmesi için tercih edilmektedir. Model, MVC'de projenin iş mantığının (business logic) oluşturulduğu bölümdür. İş mantığıyla beraber doğrulama (validation) ve veri erişim (data access) işlemleri de bu bölümde gerçekleştirilmektedir. View, MVC'de projenin arayüzlerinin oluşturulduğu bölümdür(Galloway et al., 2011). Bu bölümde projenin kullanıcılara sunulacak olan HTML dosyaları yer almaktadır. View'ın bir görevi de, kullanıcılardan alınan istekleri controller'a iletmektir. Controller, MVC'de projenin iç süreçlerini

kontrol eden bölümdür. Bu bölümde View ile Model arasındaki bağlantı kurulur. Kullanıcılardan gelen istekler (request) Controller'larda değerlendirilir, isteğin detayına göre hangi işlemlerin yapılacağı ve kullanıcıya hangi View'ın döneceği (response) belirtilir.



Şekil 1.3 – MVC Deseni

MVC tasarım modeli ilk olarak 1970'lerde Trygve Reenskaug tarafından tasarlandı. Ona göre, "MVC'nin temel amacı, insan kullanıcının zihinsel modeli ile bilgisayardaki dijital model arasındaki boşluğu doldurmaktır(Galloway et al., 2011).

1.1 Model

Model, sistemin verilerle ilgili tüm görevleri yöneten parçasıdır: doğrulama, oturum durumu ve kontrolü, veri kaynağı yapısı (veritabanı). Model, geliştiricinin yazması gereken kodun karmaşıklığını büyük ölçüde azaltır(Quinton, 2017). Model katmanı, bir uygulamanın iş mantığından sorumludur. Verilere (veritabanları, dosyalar vb.) erişme yöntemlerini kapsayacak ve yeniden kullanılabilir bir sınıf kütüphanesini kullanıma sunacaktır. Genellikle bir Model, veri soyutlaması, doğrulama ve kimlik doğrulama göz önünde bulundurularak oluşturulur. Üstelik Model, ilgi alanını tanımlayan sınıflardan oluşur. Etki alanına ait olan bu nesneler çoğu zaman veritabanlarında depolanan verileri kapsar, ancak aynı zamanda bu verileri işlemek ve

iş kurallarını uygulamak için kullanılan kodu da içerir. Sonuç olarak, veri erişimi soyutlamayı ve doğrulamayı ele almaktadır. Model, farklı veri kaynaklarıyla etkileşime yönelik yöntemler içerir.

Model sisteminin MVC dünyasına getirdiği yenilik, yalnızca veri yapısını değil aynı zamanda verinin kendisini de takip eden, geçiş fikrine dayanan bir sürüm kontrolü sistemidir. Sistem, geçişler arasında verileri depolamak için xml dosyalarını kullanır ve bu da veri tabanları için sürüm oluşturma sürecini kolayca gerçekleştirilebilir bir hale getirir.

1.2 View

View katmanı normalde web tasarımı veya şablonlar olarak adlandırılabilecek katmandır. Verilerin görüntülenme şeklini ve kullanıcının verilerle nasıl etkileşimde bulunduğunu kontrol eder. Ayrıca kullanıcılardan veri toplamanın yollarını da sağlar. Görünümlerde temel olarak kullanılan teknolojiler HTML, CSS ve JavaScript'tir (Galloway et al., 2011). Genel bir kural olarak, tasarımcının onunla çalışmasını kolaylaştırmak için bir görünüm hiçbir zaman uygulama mantığına ait öğeler içermemelidir.

1.3 Controller

Uygulamaya gelen talepleri yöneten katmandır. Gelen bir http isteği routing tablosuna bağlı olarak doğrudan controller sınıflarına ve o sınıflardaki action metotlarına aktarılır. "Controller" nesneleri içerisinde bulunan action isimli metotlar, gelen talepleri (bir veriye erişmek, güncellemek vs.) uygun görünüm ve model nesneleri üzerinden değerlendirmekle görevlidirler(Quinton, 2017). Sayfaya ulaşan talepler URL bazlı olarak çözümlenerek ilgili eylem metoduna iletilir. Kontrol katmanında bulunan Eylem metotları uygulamanın görünüm ve model bileşenleriyle doğrudan iletişim kurabilir.

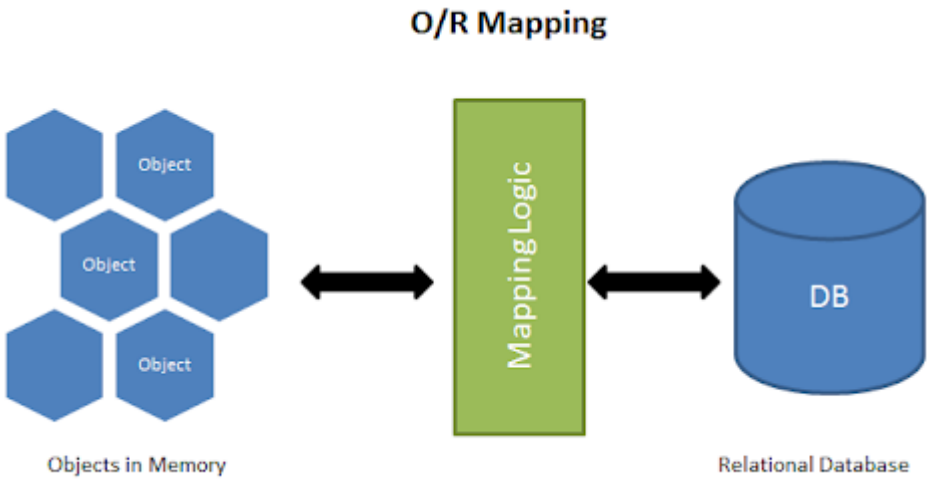
ASP.NET MVC ile hazırlanan web sayfalarına gelen istekler Routing ile alınarak Controller katmanına iletilir. Controller katmanı da isteğe göre Model katmanına veri ekler veya veri alır. İşlem sonucunu Controller dönüş türüne göre bir değer döndürür. Web sayfaları için bu genellikle View metodu yani ViewResult

nesnesidir(Badurowicz,2011). View metodu çalıştırıldığında Shared/Controller/Action dosyalarına bakar. Bu dosyalar View katmanı olarak tanımlanır ve içerisinde yer alan değerler kullanıcıya gönderilir.

Controller katmanında ki veriler View katmanına gönderilerek View Engine ile sayfaya dahil edilir.

1.4 Veri tabanı Soyutlaması

Nesneye yönelik programlamada uygulamalar nesneler kullanılarak oluşturulur. Bu nesneler gerçek hayattaki nesneleri yansıtır ve hem verileri hem de davranışları içerir. Uygulamalarda veri depolama için yaygın olarak kullanılan ilişkisel model, verileri depolamak için tabloları ve bu verilerle etkileşim kurmak için veri işleme dillerini kullanır. Bazı veri tabanı yönetim sistemleri nesne yönelimli özelliklere sahiptir ancak bunlar tam olarak uyumlu değildir. Bu iki mimarinin artık yaygın olarak kullanıldığı ve uzun süre de öyle kalacağı çok açık. Üstelik her ölçekte uygulama oluşturmak için çoğu zaman ilişkisel model ve nesne yönelimli programlama bir arada kullanılmaktadır(Krasner ve Pope, 1988).



Şekil 1.4 – Nesne İlişkisel Eşleme(OOP)

Nesne ilişkisel haritalama sistemi (ORM), nesne yönelimli sistemlerin verileri güvenli bir şekilde ve uzun bir süre boyunca bir veritabanında depolaması için bir metodoloji ve mekanizma sağlayan, bunlar üzerinde işlemsel kontrole sahip olan, ancak bunlar üzerinde işlem kontrolü sağlayan bir araç olarak tanımlanır(Krasner ve Pope, 1988).

Bir ORM sistemi, geliştiriciyi veritabanı yapısını veya şemasını bilme endişesinden kurtarır. Veri erişimi, geliştiricilerin iş nesnelerini yansıtan kavramsal bir model kullanılarak yapılır. ORM tekniği çoğu nesne yönelimli programlama dillerinde kullanılabilir. Bu teknik, nesne yönelimli programlamayı (OOP) kullanan bir programın, ilişkisel bir veritabanıyla etkileşimde bulunmasını kolaylaştırır. Projenin işleyişi ile alakalı tercih edilebilecek ORM araçları değişkenlik gösteribilmektedir. Bazı projelerde EF kullanırken bazı projelerde Dapper veya hibrit bir model kullanılabilmektedir.

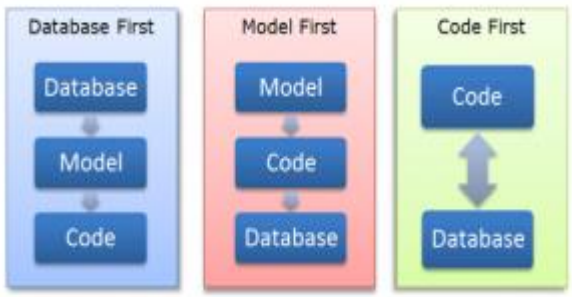
1.5 Entity Framework

Entity Framework (EF), Microsoft tarafından geliştirilen bir ORM (Object-Relational Mapping) framework'üdür(Lerman, 2010). ORM, veritabanındaki ilişkisel verileri nesne tabanlı programlama dilindeki nesnelerle eşleştirmek ve bu nesneler arasında geçiş yapmak için kullanılan bir tekniktir. Entity Framework, .NET platformu üzerinde çalışan uygulamalar için veritabanı etkileşimini yönetmek ve sadece veri tabanındaki tabloları değil, aynı zamanda bu tablolara karşılık gelen nesneleri de programlama dili üzerinde temsil etmek amacıyla kullanılır. EF, veritabanındaki tabloları, ilişkileri ve diğer nesneleri .NET programlama dilindeki sınıflarla temsil etmek için bir model oluşturur. Bu modellere Entity Data Model (EDM) denir. LINQ sorgularını kullanarak, EF veritabanından veri çekme ve sorgulama işlemlerini gerçekleştirmek için güçlü bir araç sunar(http://msdn.microsoft.com/en-us/library/zw4w595w(v=VS.100).aspx).

Entity Framework yaklaşımları 3'e ayrılır:

i. Code First Yaklaşım: Bu yaklaşım, önce uygulama kodlarını yazarak ardından Entity Framework'e veritabanı şemasını oluşturması için talimatlar verilmesini içerir(Lerman, 2010). Veritabanı şeması, kod tarafından oluşturulan sınıflara dayanır
ii. Model First Yaklaşım: Bu yaklaşım, önce uygulama kodlarını yazarak ardından Entity Framework'e veritabanı şemasını oluşturması için talimatlar verilmesini içerir. Veritabanı şeması, kod tarafından oluşturulan sınıflara dayanır(Lerman, 2010).

iii. Database First Yaklaşım: Bu yaklaşım, önce var olan bir veritabanı şemasının olduğu durumda, Entity Framework'ü bu veritabanı şemasına dayanarak uygulama modellerini oluşturmak için kullanır(Lerman, 2010).



Şekil 1.5: Entity Framework Yaklaşımları

EF, ilişkisel veritabanı yapıları arasında gezinirken, ilişkili verileri yüklemek için "Lazy Loading" ve "Eager Loading" gibi teknikleri destekler. Lazy Loading, veriye gerçekten ihtiyaç duyulduğunda ilişkili verileri yükler. Eager Loading ise ilişkili verileri anında yükler(Peretiatko ve Shirokopetleva, 2022). EF, ilişkisel veritabanları ile uyumlu çalışır ve birçok popüler veritabanı yönetim sistemini destekler (SQL Server, MySQL, PostgreSQL, vb.).

DbContext, Entity Framework'te temel veri erişim noktasıdır. Bu sınıf, veritabanı bağlantısını yönetir, sorguları yürütür ve değişiklikleri takip eder. Entity Framework'te, veritabanı tablolarını temsil eden .NET nesnelerine "entity" denir. Bu nesneler, genellikle DbContext sınıfında belirtilen veritabanındaki tablolara karşılık gelir. LINQ, .NET platformuna entegre edilmiş bir sorgu dilidir. Entity Framework, LINQ kullanarak veritabanı sorgularını .NET dilinde yazmanıza olanak tanır. Veritabanındaki tablo ve sütunlar ile .NET sınıfları ve özellikleri arasındaki ilişkiyi belirten bir "mapping" yapısı vardır. Bu, Entity Framework'in veritabanı ile nesne modeli arasında bağlantı kurmasını sağlar. EF, LINQ sorguları gibi nesne odaklı programlama özelliklerini kullanarak veritabanı işlemlerini kolaylaştırır. Bir proje geliştirildiğinde, veritabanı şemasında değişiklikler olabilir ve bu değişiklikleri veritabanına uygulamak için migration işlemleri kullanılır. Migration işlemleri, EF Code First yaklaşımında sıklıkla kullanılır. Bu yaklaşım, önce .NET sınıfları

tanımlanır, sonra bu sınıfların temelinde veritabanı şeması otomatik olarak oluşturulur veya güncellenir. Entity Framework, arka planda SQL sorgularını oluşturarak veritabanı işlemlerini gerçekleştirir. Migration işlemleri sırasında EF, uygun SQL komutlarını üreterek veritabanı şemasını günceller.

Bölüm 2

Yöntemler

Visual Studio, Microsoft tarafından geliştirilen bir entegre geliştirme ortamıdır (IDE). Visual Studio, Microsoft Windows, Windows Mobil, Windows CE, .NET Framework, .NET Compact Framework ve Microsoft Silverlight tarafından desteklenen tüm platformlar için yönetilen kod ile birlikte yerel kod ve web siteleri, web uygulamaları ve web hizmetleriyle ilgili kodlar yazmak için kullanışlı bir programdır. Bu çalışmada IDE olarak VS kullanılmıştır. Bu projede web sitesi ve admin panelli site oluşturularak bir web uygulaması geliştirilmiştir.

2.1 Website Oluşturulması

Günümüzde, şirketlerin çevrimiçi varlıkları, müşteri ilişkilerini güçlendirmek ve iş faaliyetlerini sürdürmek adına kritik bir öneme sahiptir. Bu bağlamda, projede yapılan web sitesi, şirketin dijital varlığını kurma ve sürdürme stratejilerinde kilit bir role sahiptir. Web sitesi, potansiyel müşterilere şirket hakkında bilgi sağlama, ürün ve hizmetlerini tanıtma ve en önemlisi, müşterilerle etkileşime geçme olanağı sunar. İnternet üzerinde erişilebilir bir platform olarak tasarlanan bu web sitesi, müşterilerin şirketle doğrudan etkileşime geçmelerine olanak tanır. Ziyaretçiler, web sitesi aracılığıyla şirketin faaliyet alanları, hizmetleri ve iletişim bilgilerine kolayca erişebilirler. Web sitesi aynı zamanda, kullanıcıların taleplerini iletmeleri ve şirketle iletişim kurmaları için çeşitli interaktif öğeler içerir, böylece müşteri memnuniyeti ve ilişkilerin güçlendirilmesi mümkün olur. Bu çalışmada, web sitesinin tasarımı ve işlevselliği, müşterilerle etkileşim süreçleri ve çevrimiçi varlığın şirket stratejilerine olan katkısı ayrıntılı bir şekilde ele alınmıştır.

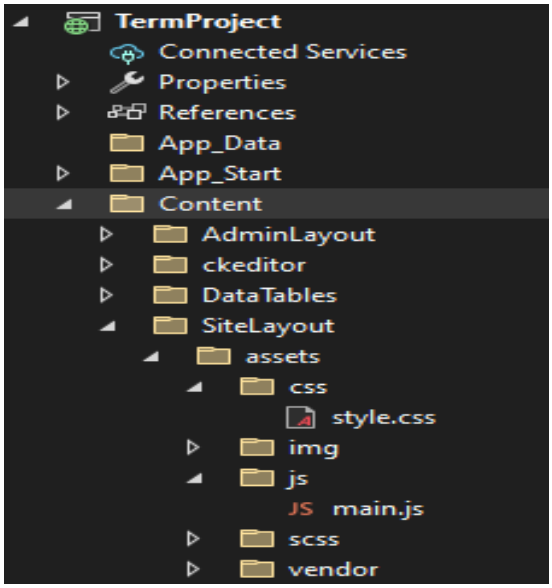
Web sitesi tasarımı, bir dizi temel tasarım ilkesine dayanmaktadır. Bu ilkelere uygun bir tasarım, kullanıcıların etkileşimini artırır, bilgiye erişimi kolaylaştırır ve genel olarak kullanıcı deneyimini olumlu yönde etkiler.

i. Web sitesi, kullanıcıların rahatlıkla gezinebilmelerini ve istedikleri bilgilere kolayca ulaşabilmelerini sağlamak amacıyla kullanıcı dostu bir arayüz tasarımına sahiptir. Menüler, düğmeler ve içerikler, kullanıcının beklentilerine uygun olarak yerleştirilmiş ve düzenlenmiştir.
ii. Tasarım sürecinde bütünlük ilkesine özel bir önem verilmiştir. Web sitesinde kullanılan renk paleti, tipografi, ve grafik öğeleri birbiriyle uyumlu bir görünüm sunar. Bu, kullanıcılara tutarlı bir marka deneyimi yaşatır.
iii. Web sitesi, erişilebilirlik ilkelerine uygun olarak tasarlanmıştır. Metinlerin okunabilirliği, renk kontrastları, ve klavye erişimine yönelik düzenlemeler, engelli kullanıcıların da rahatlıkla siteye erişimini sağlar.
iv. @media sorguları kullanılarak, web sitesinin farklı ekran boyutlarına uyum sağlaması ve mobil cihazlarda optimize bir görüntü sunması sağlanmıştır. Bu, kullanıcıların herhangi bir cihazda tutarlı bir deneyim yaşamasını sağlar.
v. Google Fonts üzerinden alınan Open Sans ve Roboto fontları, kullanıcı dostu ve okunabilir bir metin görseli oluşturmak için tercih edilmiştir. Renk paleti, okunabilirlik ve estetik açıdan dikkatle seçilmiştir.
vi. Bootstrap, sayfa düzeni ve bileşenlerin oluşturulmasında kullanılmıştır. Bu, hızlı ve tutarlı bir tasarım süreci sağlar. Ayrıca, diğer CSS framework'leri de projede kullanılarak belirli özelliklerin entegrasyonu sağlanmıştır.
vii. Sayfanın alt kısmında bulunan footer ve üst kısımdaki navigasyon çubuğu, kullanıcıların kolayca gezinmelerini sağlamak amacıyla tasarlanmıştır. Temiz ve düzenli bir tasarım, kullanıcıların istedikleri bilgilere hızlıca ulaşmalarını sağlar.

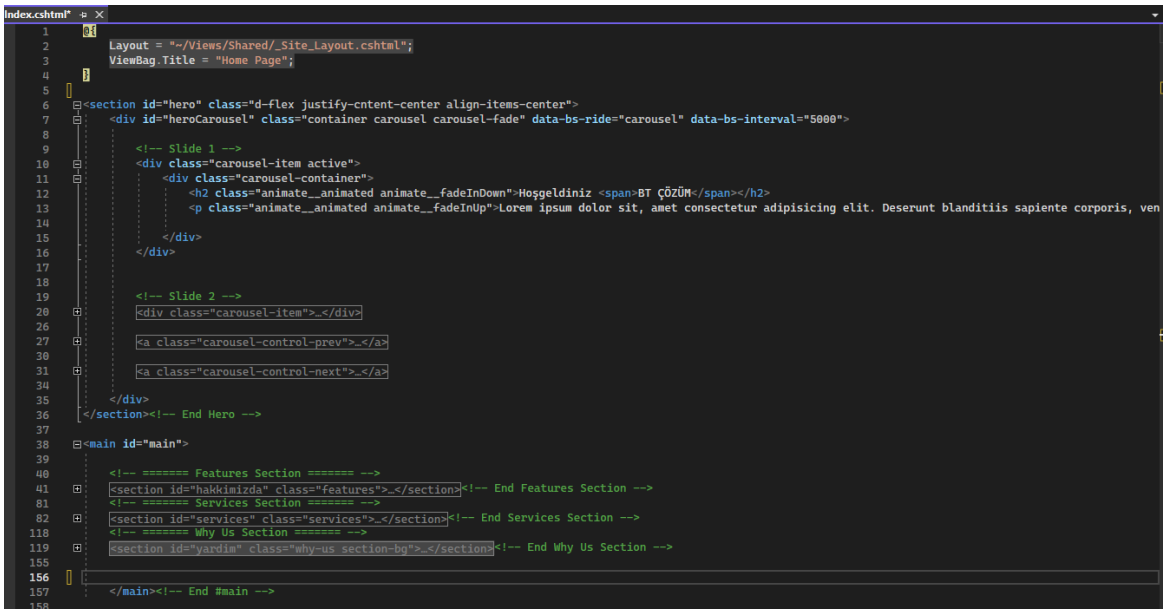
Bu tasarım ilkeleri, web sitesinin kullanıcılarla etkileşimini optimize etmek ve genel kullanılabilirliği artırmak adına dikkatle uygulanmıştır. Kullanıcılar, bu tasarım ilkeleri sayesinde web sitesini kullanırken kolaylıkla yönlendirilebilir ve olumlu bir deneyim yaşayabilirler.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>BT ÇÖZÜM</title>
    <meta content="" name="description">
    <meta content="" name="keywords">

    <!-- Favicons -->
    <link href="/Content/SiteLayout/assets/img/favicon.png" rel="icon">
    <link href="/Content/SiteLayout/assets/img/apple-touch-icon.png" rel="apple-touch-icon">

    <!-- Google Fonts -->
    <link href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,700,700i|Roboto:300,300i,400,400i,500,500i,700,700i&display=swap" rel=

    <!-- Vendor CSS Files -->
    <link href="/Content/SiteLayout/assets/vendor/animate.css/animate.min.css" rel="stylesheet">
    <link href="/Content/SiteLayout/assets/vendor/aos/aos.css" rel="stylesheet">
    <link href="/Content/SiteLayout/assets/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">
    <link href="/Content/SiteLayout/assets/vendor/bootstrap-icons/bootstrap-icons.css" rel="stylesheet">
    <link href="/Content/SiteLayout/assets/vendor/boxicons/css/boxicons.min.css" rel="stylesheet">
    <link href="/Content/SiteLayout/assets/vendor/glightbox/css/glightbox.min.css" rel="stylesheet">
    <link href="/Content/SiteLayout/assets/vendor/swiper/swiper-bundle.min.css" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.1/css/all.min.css" />

    <!-- Template Main CSS File -->
    <link href="/Content/SiteLayout/assets/css/style.css" rel="stylesheet">


</head>
<body>

    <!-- ======= Header ======= -->
    <header id="header" class="fixed-top d-flex align-items-center header-transparent">…</header><!-- End Header -->
    <!-- ======= Hero Section ======= -->
    @RenderBody();
    <!-- ======= Footer ======= -->
    <footer id="footer">…</footer><!-- End Footer -->

    <a href="#" class="back-to-top d-flex align-items-center justify-content-center"><i class="bi bi-arrow-up-short"></i></a>

    <!-- Vendor JS Files -->
    <script src="/Content/SiteLayout/assets/vendor/purecounter/purecounter_vanilla.js"></script>
    <script src="/Content/SiteLayout/assets/vendor/aos/aos.js"></script>
    <script src="/Content/SiteLayout/assets/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
    <script src="/Content/SiteLayout/assets/vendor/glightbox/js/glightbox.min.js"></script>
```

Şekil 2.1 : Website – Layout



Şekil 2.2 : Website Dosyaları



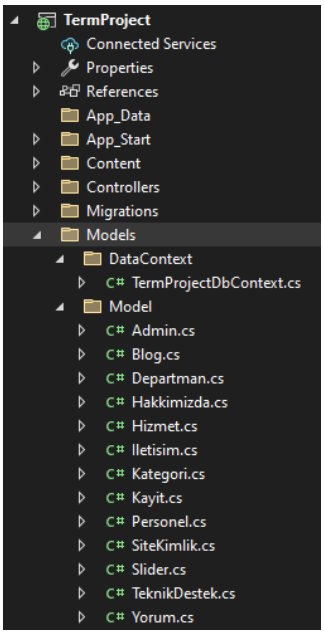
Şekil 2.3: /Home/Index

2.2 Admin Panelli : MVC Modeli ile Yapılandırma

Bu projede, web uygulamasının yönetici paneli, Model-View-Controller (MVC) tasarım deseni kullanılarak oluşturulmuştur. Bu tasarım deseni, uygulamayı mantıksal olarak parçalara böler ve her bir parçanın belirli bir sorumluluğu bulunur. Aşağıda, bu projedeki MVC yapılandırması ve yönetici paneli ile ilgili başlık ve altındaki bilgiler hakkında bir açıklamada bulunmaktadır.

2.2.1 Model Katmanı : Veri ve İş Mantığı Yönetimi

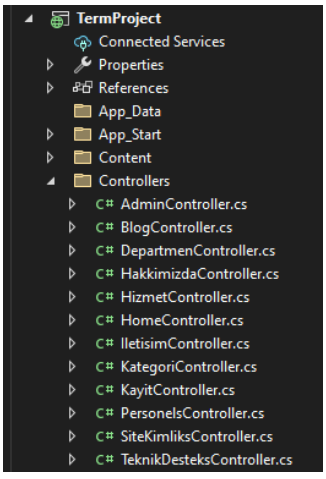
Model katmanı uygulamanın very tabanı etkileşimini ve iş mantığını yönetir. Yönetici paneli için oluşturulan modeler, veritabanındaki tabloları temsil eder ve very manipülasyonu işlemlerini gerçekleştirir. Kullanıcı bilgilerini, içerik yönetimini bu katmanda ele alınabilir. Bu projede de kullanılacak Admin, Departman, Personel, Arıza Kayıt, Teknik Destek gibi sınıflar bu katmanda oluşturulmuştur.



Şekil 2.4 : Model Katmanı

2.2.2 Controller Katmanı: İşlemlerin Yönlendirilmesi ve Kontrolü

Controller katmanı, kullanıcının etkileşimleri sonucu hangi işlemlerin gerçekleştirileceğini belirler. Yönetici panelindeki her sayfa ve her bileşen kendi kontrolörüne sahiptir. Kullanıcı yönetimi, içerik düzenleme, rapor oluşturma işlemleri bu katmanda tanımlanmıştır.



Şekil 2.5 : Controller Katmanı

2.2.2.1 Admin Controller

Bu bölümde, projenin yönetici paneli işlemlerini gerçekleştiren “AdminController” sınıfının temel fonksiyonları anlatılmaktadır. Bu sınıf projenin yönetici paneli için giriş, çıkış, şifre hatırlatma, admin listeleme ve admin düzenleme gibi işlevleri sağlar. Ayrıca yeni admin eklemek ve mevcut adminleri düzenlemek veya silmek gibi yönetici paneli işlemlerini yönetir. “Index” metodu, yönetici paneline giriş yapıldığında çalışır. Giriş işlemi, kullanıcının veritabanındaki kayıtlarla eşleştirilerek gerçekleşir. Başarılı giriş durumunda, ilgili bilgiler Session değişkenleri aracılığıyla saklanır ve yönetici ana sayfasına yönlendirilir. Ayrıca, Logout metodu ile de oturum kapatma işlemi gerçekleştirilir. Şifre hatırlatma işlemi, kullanıcının kayıtlı e-posta adresine yeni bir şifre göndermeyi amaçlar. Bu işlem, rastgele oluşturulan bir şifrenin, kullanıcının e-posta adresine gönderilmesi ve veritabanındaki kaydın güncellenmesi üzerinden gerçekleştirilir. Adminler metodu, veritabanındaki adminleri listeler ve Edit metodu ile düzenleme işlemi gerçekleştirilir. Düzenleme işlemi, seçilen adminin bilgilerinin güncellenmesini sağlar. Create metodu, yeni bir admin eklemek için kullanılır. Bu işlem, kullanıcı tarafından girilen bilgilerin geçerliliği kontrol edildikten sonra, veritabanına yeni bir admin eklenerek gerçekleştirilir. Delete metodu, seçilen bir adminin veritabanından silinmesini sağlar. Silme işlemi, adminin varlığı kontrol edildikten sonra gerçekleştirilir.

```
public class AdminController : Controller
{
    TermProjectDbContext db=new TermProjectDbContext();
    // GET: Admin

    0 references
    public ActionResult Index()...

    0 references
    public ActionResult Login() ...

    0 references
    public dynamic GetViewBag()...

    [HttpPost]

    0 references
    public ActionResult Login(Admin admin, dynamic viewBag )...

    0 references
    public ActionResult Logout()...

    0 references
    public ActionResult Rememberme()...
    [HttpPost]
    0 references
    public ActionResult RememberMe(string eposta)...

    0 references
    public ActionResult Adminler()...

    0 references
    public ActionResult Create() ...
    [HttpPost]
    0 references
    public ActionResult Create(Admin admin, string sifre, string eposta)...

    0 references
    public ActionResult Edit(int id)...
    [HttpPost]
    0 references
    public ActionResult Edit(int id, Admin admin, string sifre, string eposta )...
    0 references
    public ActionResult Delete(int id)...
}
```

Şekil 2.6 : AdminController

2.2.2.2 DepartmanController

Bu bölümde, projenin DepartmanController sınıfının temel işlevleri anlatılmıştır. Bu Controller sınıfı, Departman işlemlerini gerçekleştiren metodları içerir. Departman işlemleri, departmanların oluşturulması, düzenlenmesi, silinmesi ve listelenmesi gibi temel CRUD (Create, Read, Update, Delete) işlemlerini içerir. Index metodu, tüm departmanları listeleyerek kullanıcıya sunar. Bu metot, Departman ve Personel ilişkisini içerir ve ilgili departmanın bağlı olduğu personel sayısını gösterir. Veritabanından tüm departmanları çeker ve bu bilgileri View'e ileterek kullanıcıya sunar. Bu Controller sınıfındaki metodlar, projenin departman yönetimi üzerinde CRUD işlemlerini gerçekleştirir. Bu sayede kullanıcılar, departmanları oluşturabilir, düzenleyebilir, silebilir ve listeyebilirler. Controller sınıfındaki [HttpPost] nitelikleri, formdan gelen verilerin işlenmesi ve veritabanına kaydedilmesi amacıyla kullanılır.

```
public class DepartmenController : Controller
{
    private TermProjectDbContext db = new TermProjectDbContext();

    // GET: Departmen
    0 references
    public ActionResult Index()...

    // GET: Departmen/Details/5
    0 references
    public ActionResult Details(int? id)...

    // GET: Departmen/Create
    0 references
    public ActionResult Create()...


    [HttpPost]
    [ValidateAntiForgeryToken]
    0 references
    public ActionResult Create([Bind(Include = "Id,Ad")] Departman departman)...

    // GET: Departmen/Edit/5
    0 references
    public ActionResult Edit(int? id)...

    [HttpPost]
    [ValidateAntiForgeryToken]
    0 references
    public ActionResult Edit([Bind(Include = "Id,Ad")] Departman departman)...

    // GET: Departmen/Delete/5
    0 references
    public ActionResult Delete(int? id)...


    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    0 references
    public ActionResult DeleteConfirmed(int id)...

    0 references
    protected override void Dispose(bool disposing)...
}
```

Şekil 2.7 : DepartmanController

2.2.2.3 PersonelController

Bu bölümde, projenin **PersonelsController** sınıfının temel işlevleri anlatılmıştır. Bu Controller sınıfı, Personel işlemlerini gerçekleştiren metodları içerir. Personel işlemleri, personellerin oluşturulması, düzenlenmesi, silinmesi ve listelenmesi gibi temel CRUD (Create, Read, Update, Delete) işlemlerini içerir. Index metodu, tüm personelleri listeleyerek kullanıcıya sunar. Bu metot, Personel ve Departman ilişkisini içerir ve ilgili personelin bağlı olduğu departman bilgisini gösterir. Veritabanından tüm personelleri çeker ve bu bilgileri View'e ileterek kullanıcıya sunar. Bu Controller sınıfındaki metodlar, projenin personel yönetimi üzerinde CRUD işlemlerini gerçekleştirir. Bu sayede kullanıcılar, personelleri oluşturabilir, düzenleyebilir, silebilir ve listeyebilirler. Controller sınıfındaki **[HttpPost]** nitelikleri, formdan gelen verilerin işlenmesi ve veritabanına kaydedilmesi amacıyla kullanılır.

```
public class PersonelsController : Controller
{
    private TermProjectDbContext db = new TermProjectDbContext();

    // GET: Personels
    0 references
    public async Task<ActionResult> Index()
    {
        return View(await db.Personels.Include(p => p.departman).ToListAsync());

    }

    // GET: Personels/Details/5
    0 references
    public async Task<ActionResult> Details(int? id)...

    // GET: Personels/Create
    0 references
    public ActionResult Create()
    {
        return View();
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    0 references
    public async Task<ActionResult> Create([Bind(Include = "Id,Ad,Soyad,Telefon,DepartmanId")] Personel personel)...

    // GET: Personels/Edit/5
    0 references
    public async Task<ActionResult> Edit(int? id)...


    [HttpPost]
    [ValidateAntiForgeryToken]
    0 references
    public async Task<ActionResult> Edit([Bind(Include = "Id,Ad,Soyad,Telefon,DepartmanId")] Personel personel)...

    // GET: Personels/Delete/5
    0 references
    public async Task<ActionResult> Delete(int? id)...

    // POST: Personels/Delete/5
    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    0 references
    public async Task<ActionResult> DeleteConfirmed(int id)...
```

Şekil 2.8 : PersonelController

2.2.2.4 TeknikDestekController

Bu bölümde, projenin TeknikDesteksController sınıfının temel işlevleri anlatılmıştır. Bu Controller sınıfı, Teknik Destek işlemlerini gerçekleştiren metodları içerir. Teknik Destek işlemleri, teknik destek kayıtlarının oluşturulması, düzenlenmesi, silinmesi ve listelenmesi gibi temel CRUD (Create, Read, Update, Delete) işlemlerini içerir. Index metodu, tüm teknik destek kayıtlarını listeleyerek kullanıcıya sunar. Bu metot, Personel ve TeknikDestek ilişkisini içerir ve ilgili teknik destek kaydının hangi personele ait olduğunu gösterir. Veritabanından tüm teknik destek kayıtlarını çeker ve bu bilgileri View'e ileterek kullanıcıya sunar.

Bu Controller sınıfındaki metodlar, projenin teknik destek yönetimi üzerinde CRUD işlemlerini gerçekleştirir. Bu sayede kullanıcılar, teknik destek kayıtlarını oluşturabilir, düzenleyebilir, silebilir ve listeyebilirler. Controller sınıfındaki [HttpPost] nitelikleri, formdan gelen verilerin işlenmesi ve veritabanına kaydedilmesi amacıyla kullanılır.

```
public class TeknikDesteksController : Controller
{
    private TermProjectDbContext db = new TermProjectDbContext();

    // GET: TeknikDesteks
    0 references
    public async Task<ActionResult> Index()
    {
        var teknikDesteks = db.TeknikDesteks.Include(t => t.Personel);
        return View(await teknikDesteks.ToListAsync());
    }

    // GET: TeknikDesteks/Details/5
    0 references
    public async Task<ActionResult> Details(int? id)...

    // GET: TeknikDesteks/Create
    0 references
    public ActionResult Create()
    {
        ViewBag.PersonelList = new SelectList(db.Personels, "Id", "Ad");
        return View();
    }


    [HttpPost]
    [ValidateAntiForgeryToken]
    0 references
    public async Task<ActionResult> Create([Bind(Include = "Id,Telefon,MusteriEposta,MusteriTelefon,ArızaCozum,ArızaBaslangicTarihi,ArizaBaslangicSaati,ArizaBitisTarih

    // GET: TeknikDesteks/Edit/5
    0 references
    public async Task<ActionResult> Edit(int? id)...


    [HttpPost]
    [ValidateAntiForgeryToken]
    0 references
    public async Task<ActionResult> Edit([Bind(Include = "Id,Telefon,MusteriEposta,MusteriTelefon,ArızaCozum,ArızaBaslangicTarihi,ArizaBaslangicSaati,ArizaBitisTarihi,

    // GET: TeknikDesteks/Delete/5
    0 references
    public async Task<ActionResult> Delete(int? id)...

    // POST: TeknikDesteks/Delete/5
    [HttpPost, ActionName("Delete")]
```

Şekil 2.9 : TeknikDestekController

2.2.2.5 KayıtController

Bu bölümde, projenin KayitController sınıfının temel işlevleri anlatılmıştır. Bu Controller sınıfı, müşteri kayıtlarıyla ilgili işlemleri gerçekleştiren metodları içerir. Müşteri kayıt işlemleri, müşteri bilgilerinin kaydedilmesi, düzenlenmesi, silinmesi ve listelenmesi gibi temel CRUD (Create, Read, Update, Delete) işlemlerini içerir.

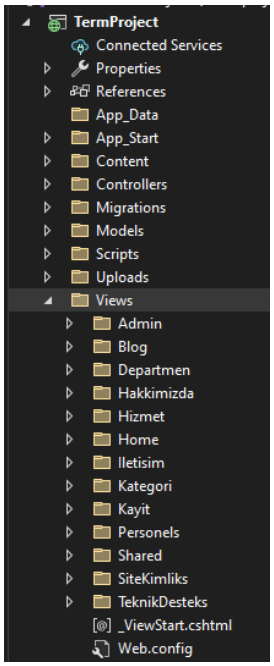
Index metodu, tüm müşteri kayıtlarını listeleyerek kullanıcıya sunar. Veritabanından tüm müşteri kayıtlarını çeker ve bu bilgileri View'e ileterek kullanıcıya gösterir. Bu Controller sınıfındaki metodlar, projenin müşteri kayıt yönetimi üzerinde CRUD işlemlerini gerçekleştirir. Bu sayede kullanıcılar, müşteri kayıtlarını oluşturabilir, düzenleyebilir, silebilir ve listeyebilirler. Controller sınıfındaki [HttpPost] nitelikleri, formdan gelen verilerin işlenmesi ve veritabanına kaydedilmesi amacıyla kullanılır.

```
public class KayitController : Controller
{
    private TermProjectDbContext db = new TermProjectDbContext();

    // GET: Kayit
    0 references
    public async Task<ActionResult> Index()
    {
        return View(await db.Kayits.ToListAsync());
    }

    // GET: Kayit/Details/5
    0 references
    public async Task<ActionResult> Details(int? id)...

    // GET: Kayit/Create
    0 references
    public ActionResult Create()
    {
        return View();
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    0 references
    public async Task<ActionResult> Create([Bind(Include = "KayitId,MüşteriAdı,MüşteriSoyadı,Email,Telefon,Adres,Tarih,AracMarka,AracModel,YakıtTipi,ArızaBilgisi,ResimU

    // GET: Kayit/Edit/5
    0 references
    public async Task<ActionResult> Edit(int? id)...


    [HttpPost]
    [ValidateAntiForgeryToken]
    0 references
    public async Task<ActionResult> Edit([Bind(Include = "KayitId,MüşteriAdı,MüşteriSoyadı,Email,Telefon,Adres,Tarih,AracMarka,AracModel,YakıtTipi,ArızaBilgisi,ResimUrl

    // GET: Kayit/Delete/5
    0 references
    public async Task<ActionResult> Delete(int? id)...

    // POST: Kayit/Delete/5
    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    0 references
    public async Task<ActionResult> DeleteConfirmed(int id)...
```

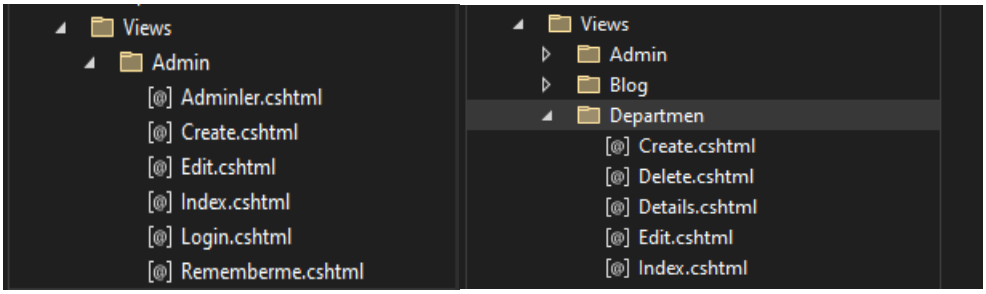
Şekil 2.10 : KayıtController

2.2.3 View Katmanı: Kullanıcı Ara yüz ve Temsil

View katmanı, kullanıcı arayüzünü temsil eder. Her bir sayfa veya bileşen kendi görünüm dosyasına sahiptir. Bu görünüm dosyaları, kullanıcıya gösterilen içeriği ve uygulamanın genel estetiğini belirler.



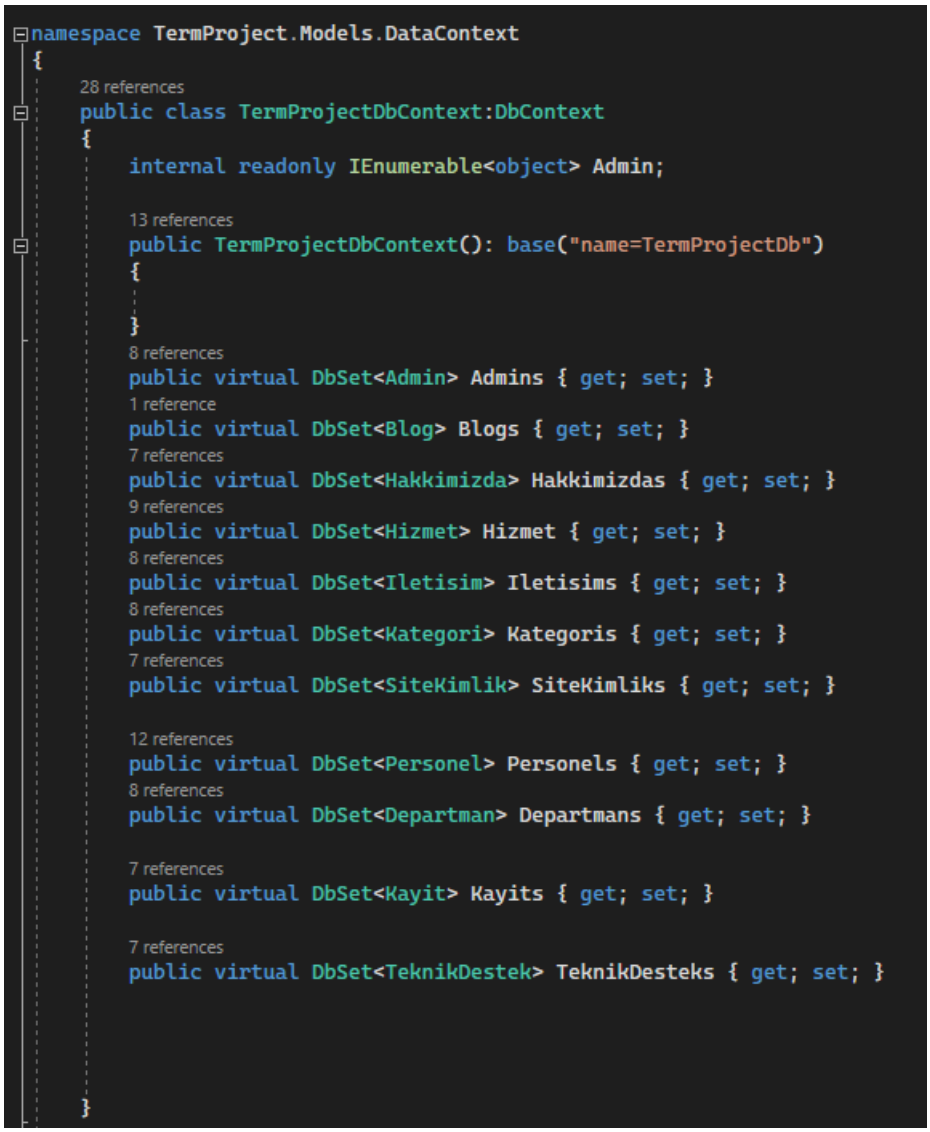
Şekil 2.11 : View Katmanı



Şekil 2.12: View Model Katmanları

2.3 VeriTabanı Oluşturulması

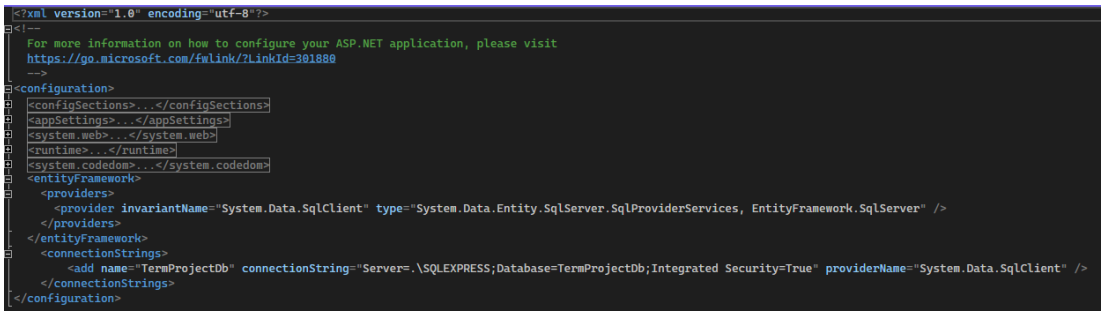
Veritabanı oluşturulurken, Entity Framework Code First yaklaşımı kullanılmıştır. Bu yaklaşım, veritabanının model sınıfları üzerinden otomatik olarak oluşturulmasına olanak tanır. İlk olarak, veri modelini temsil eden sınıflar oluşturulmuştur. Bu sınıflar, uygulamanın temel veri yapısını belirler. DbContext sınıfı oluşturuldu. Bu sınıf, uygulamanın veri modeli ve veritabanı bağlantısını yönetir. DbSet özellikleri, veritabanındaki tabloları temsil eder. Daha sonra, Entity Framework migration'ları kullanılarak veritabanına bu modellerin yansıtılması sağlanmıştır. Migration işlemleri, veri modelinde yapılan değişiklikleri algılar ve bu değişiklikleri veritabanına uygular.



```
namespace TermProject.Models.DataContext
{
    28 references
    public class TermProjectDbContext:DbContext
    {
        internal readonly IEnumerable<object> Admin;

        13 references
        public TermProjectDbContext(): base("name=TermProjectDb")
        {

        }
        8 references
        public virtual DbSet<Admin> Admins { get; set; }
        1 reference
        public virtual DbSet<Blog> Blogs { get; set; }
        7 references
        public virtual DbSet<Hakkimizda> Hakkimizdas { get; set; }
        9 references
        public virtual DbSet<Hizmet> Hizmet { get; set; }
        8 references
        public virtual DbSet<Iletisim> Iletisims { get; set; }
        8 references
        public virtual DbSet<Kategori> Kategoris { get; set; }
        7 references
        public virtual DbSet<SiteKimlik> SiteKimliks { get; set; }

        12 references
        public virtual DbSet<Personel> Personels { get; set; }
        8 references
        public virtual DbSet<Departman> Departmans { get; set; }

        7 references
        public virtual DbSet<Kayit> Kayits { get; set; }

        7 references
        public virtual DbSet<TeknikDestek> TeknikDesteks { get; set; }



    }
```

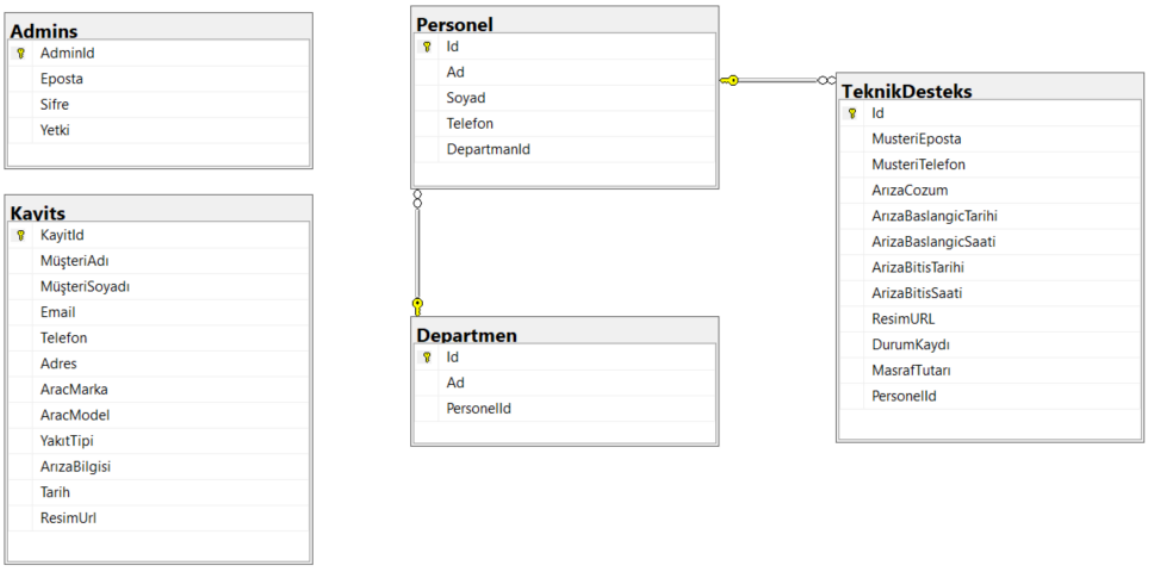
Şekil 2.13 : DbContext Sınıfı

Entity Framework Code First yaklaşımında, veri tabanının türü ve bağlantı dizesi web.config dosyası içinde belirtilmiştir. Bu ayarda, uygulamanın veri tabanına nasıl bağlanacağı ve hangi türde bir veri tabanı kullanılacağı tanımlanmıştır.



```
<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  https://go.microsoft.com/fwlink/?LinkId=301880
  -->
<configuration>
  <configSections>...</configSections>
  <appSettings>...</appSettings>
  <system.web>...</system.web>
  <runtime>...</runtime>
  <system.codedom>...</system.codedom>
  <entityFramework>
    <providers>
      <provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
    </providers>
  </entityFramework>
    <connectionStrings>
        <add name="TermProjectDb" connectionString="Server=.\SQLEXPRESS;Database=TermProjectDb;Integrated Security=True" providerName="System.Data.SqlClient" />
    </connectionStrings>
</configuration>
```

Şekil 2.14 : Web.Config dosyası



Şekil 2.15: Database Diyagram

Bölüm 3

Bulgular

Bu proje kapsamında, ASP.NET MVC mimarisi kullanılarak geliştirilen web sitesinde elde edilen bulgular, kullanıcı deneyimi, veritabanı etkileşimleri üzerine odaklanmaktadır.

3.1 WebSitesi

Bu proje kapsamında geliştirilen web sitesi, müşteriler ve ziyaretçiler için kullanıcı dostu bir deneyim sunmayı amaçlamaktadır. Web sitesine giriş yapan ziyaretçiler, aşağıda belirtilen alanlar üzerinde etkileşimde bulunabilirler.

i. Anasayfa : Web sitesine ilk giriş yapan ziyaretçiler, anasayfa üzerinde temel bilgilere hızlı bir erişim sağlarlar.
ii. Hakkımızda : Müşterilere ve ziyaretçilere şirket hakkında detaylı bilgiler sunar.

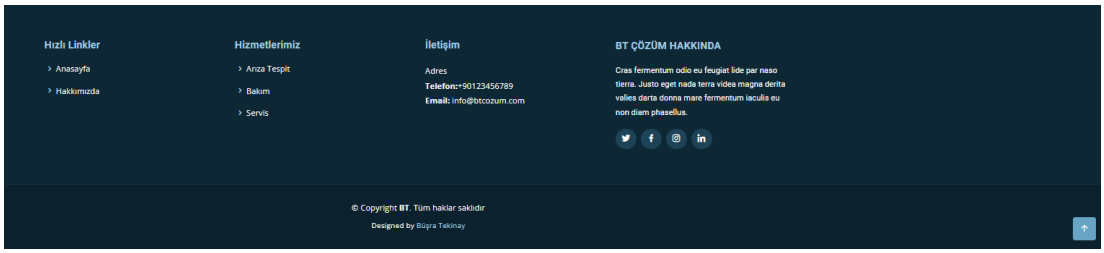
iii. Hizmetler : Müşterilere sunulan hizmetleri detaylı bir şekilde açıklar. Arıza tespiti, bakım ve servis gibi hizmetler, ziyaretçilere net bir şekilde sunularak, ihtiyaçlarına uygun hizmetlere kolayca ulaşmaları sağlanır.

iv. Yardım : Müşterilerin firma ile telefonla ya da mesaj gönderilerek iletişime geçmesi için oluşturulur.

v. İletişim : Müşterilerin ve ziyaretçilerin şirketle iletişime geçmelerini sağlar. Adres bilgileri, telefon numaraları ve iletişim formu aracılığıyla kolayca ulaşılabilir.

3.1.1 Header ve Footer

Şekil 3.1: Site – Header Kısmı



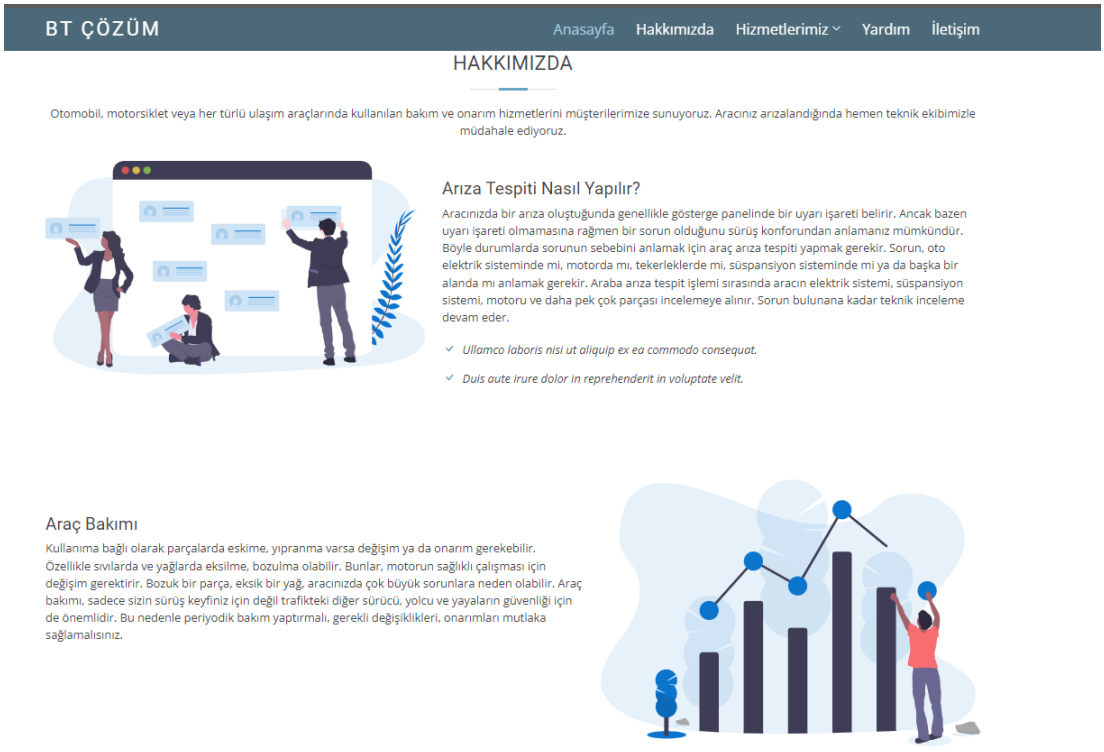
Şekil 3.2: Site – Footer Kısmı

Web sitesinin tasarımında, kullanıcıların site içinde gezinirken genel erişim ve deneyimlerini kolaylaştırmak için header ve footer kısımları stratejik bir şekilde tasarlanmıştır.

Web sitesinin üst kısmında bulunan header, kullanıcılara anında önemli bilgilere ulaşma imkanı sunar. Web sitesinin alt kısmında yer alan footer, kullanıcılara genel bilgiler ve ek bağlantılar sunar.

3.1.2 Hakkımızda

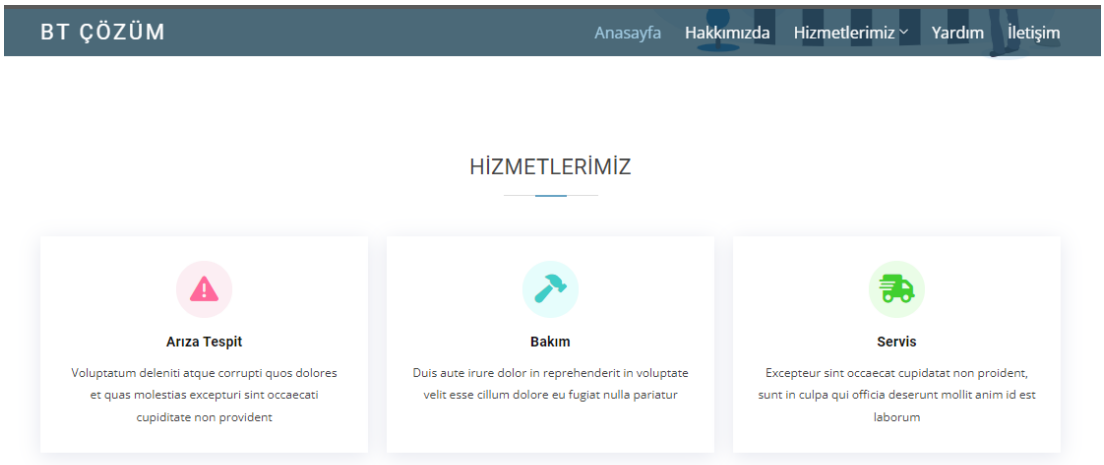
Bu kısımda şirketin ilgilendiği işlerle ilgili genel bilgiler yer almaktadır.



Şekil 3.3: Site-Hakkımızda Kısmı

3.1.3 Hizmetler

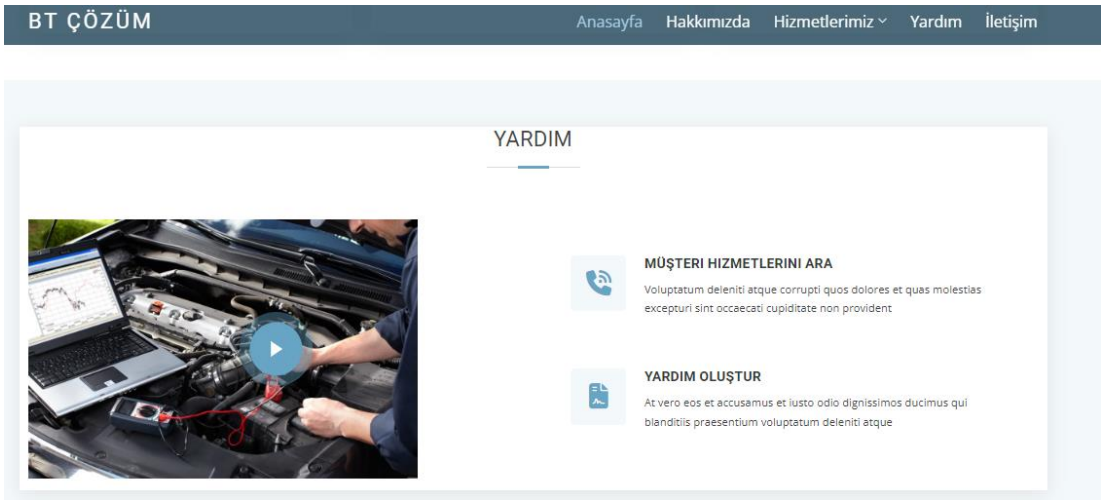
Bu kısımda firmanın müşterilerine sunduğu hizmetlerle ilgili bilgiler yer almaktadır.



Şekil 3.4 : Site – Hizmetlerimiz Kısmı

3.1.4 Yardım

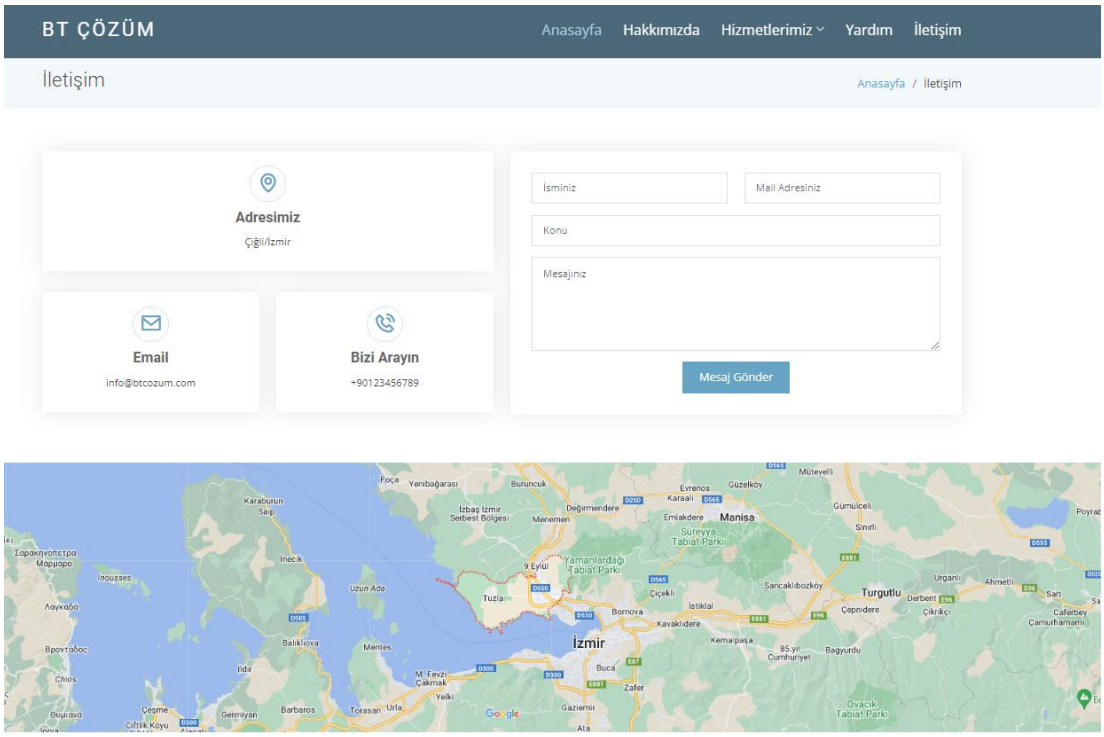
Bu kısımda müşterinin sorunu ile ilgili irtibata geçeceği yerlerle ilgili bilgiler yer almaktadır. Telefonla ya da mail yoluyla firma ile iletişime geçilebilmektedir.



Şekil 3.5 : Site –Yardım Kısmı

3.1.5 İletişim

Bu kısımda firmanın irtibat bilgileri, harita üzerinde konumu yer almaktadır.



Şekil 3.6 : Site –İletişim Kısmı

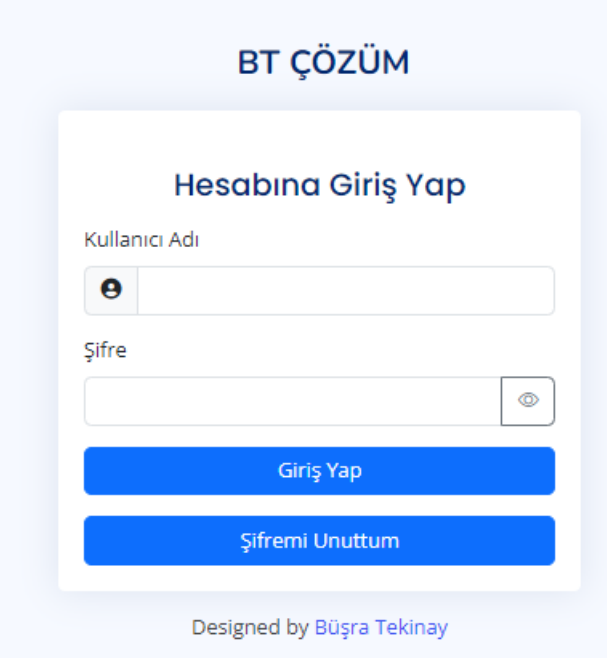
3.2 Admin Paneli

Proje kapsamında oluşturulan web sitesinde, yönetici (admin) paneli, sistemi yönetmek, içeriği düzenlemek ve kullanıcıların etkileşimini takip etmek için tasarlanmış bir arayüzü içermektedir. Admin paneli, özel bir giriş yetkisi olan kullanıcılara, yönetici, moderatör veya içerik yöneticisi gibi rollerde bulunan kişilere sunulur. Bu projemizde de admin panelinde müşterilerden gelen ilgili sorunlara göre personel desteği sağlanıp, hangi personel ne işleri yaptıysa gösterilip performans değerlendirmesi yapılabilmektedir.

Panel kısmında ilgili yetkilendirme alanında düzenleme yapılmıştır. Yetkisi Admin olan kişiler panelle ilgili tüm yetkilendirmelere sahiptir. Admin olmayan kişiler panelde sadece ilgili oldukları alanı görürler.

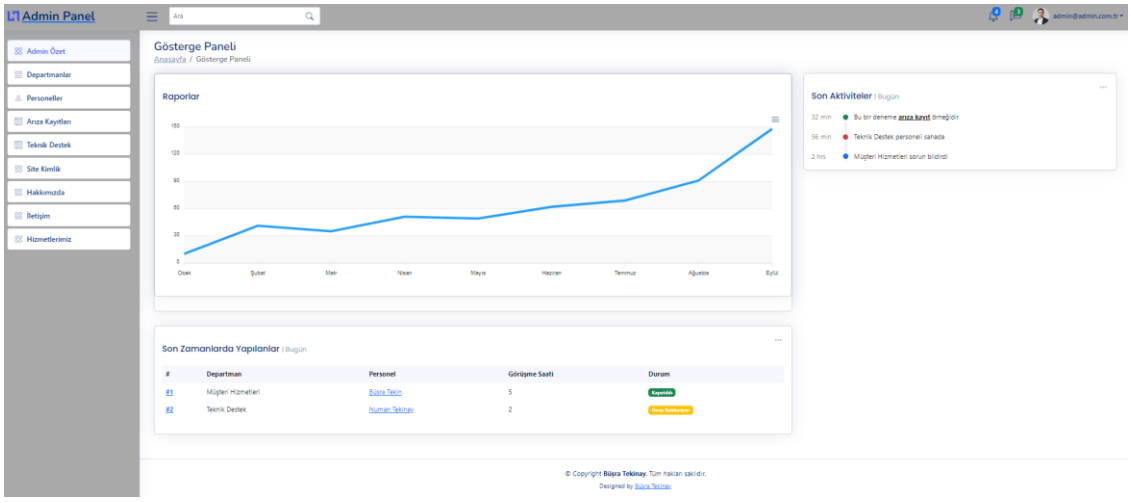
3.2.1 Admin Giriş

Admin paneline belirli kullanıcı girişleri ve yetkileri olanlar eposta ve şifre ile giriş yapabilmektedir.



Şekil 3.7 : Admin Paneli - Giriş

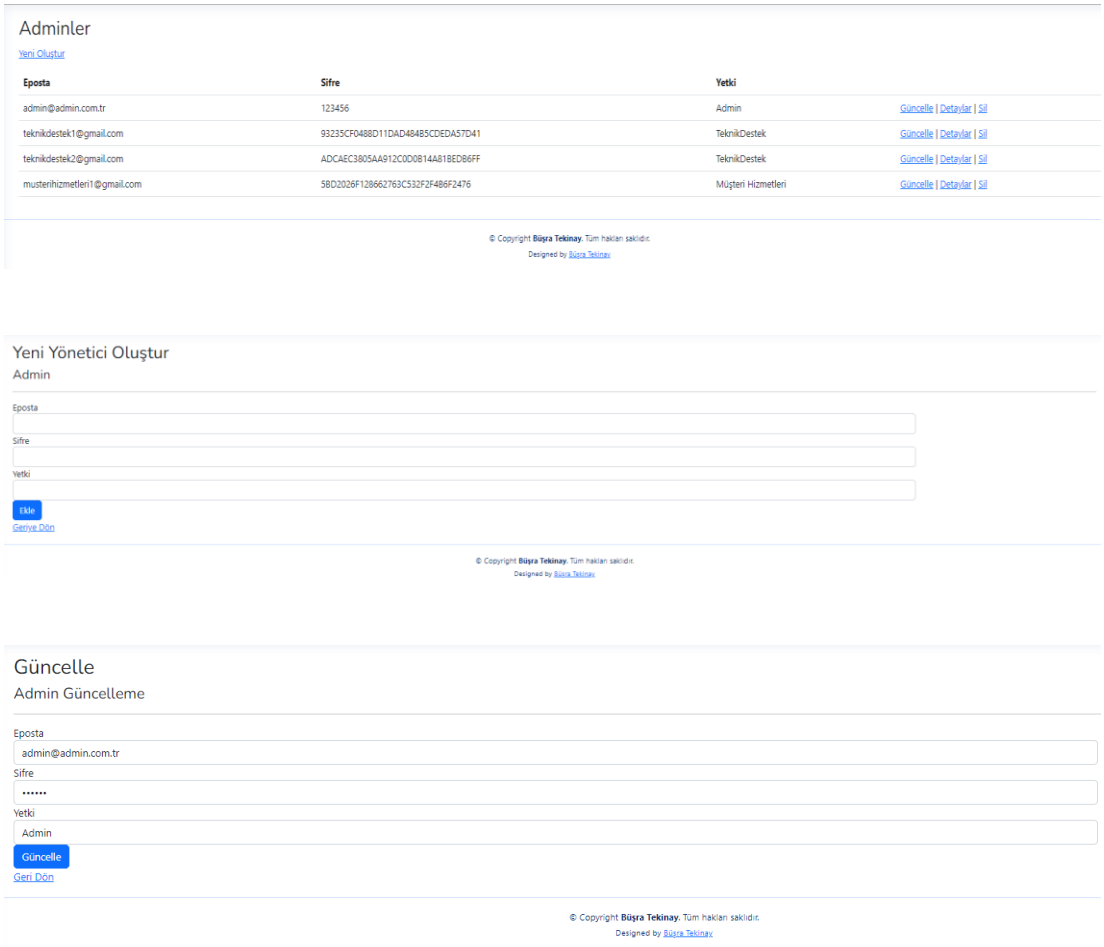
3.2.1.1 Admin Gösterge Paneli



Şekil 3.8 : Admin Paneli Gösterge

3.2.1.2 Admin Bilgileri

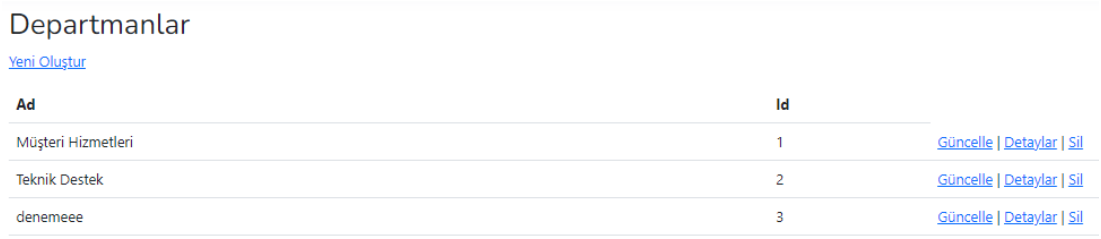
Bu kısımda admin bilgileri yer almaktadır. Yeni her departman için kullanıcı eklenebilmektedir. Admin, Teknik Destek ve Müşteri Hizmetleri personellerinin panel yetkileri bu kısımda sağlanılmaktadır.



Şekil 3.9 : Admin Bilgileri

3.2.1.3 Departmanlar

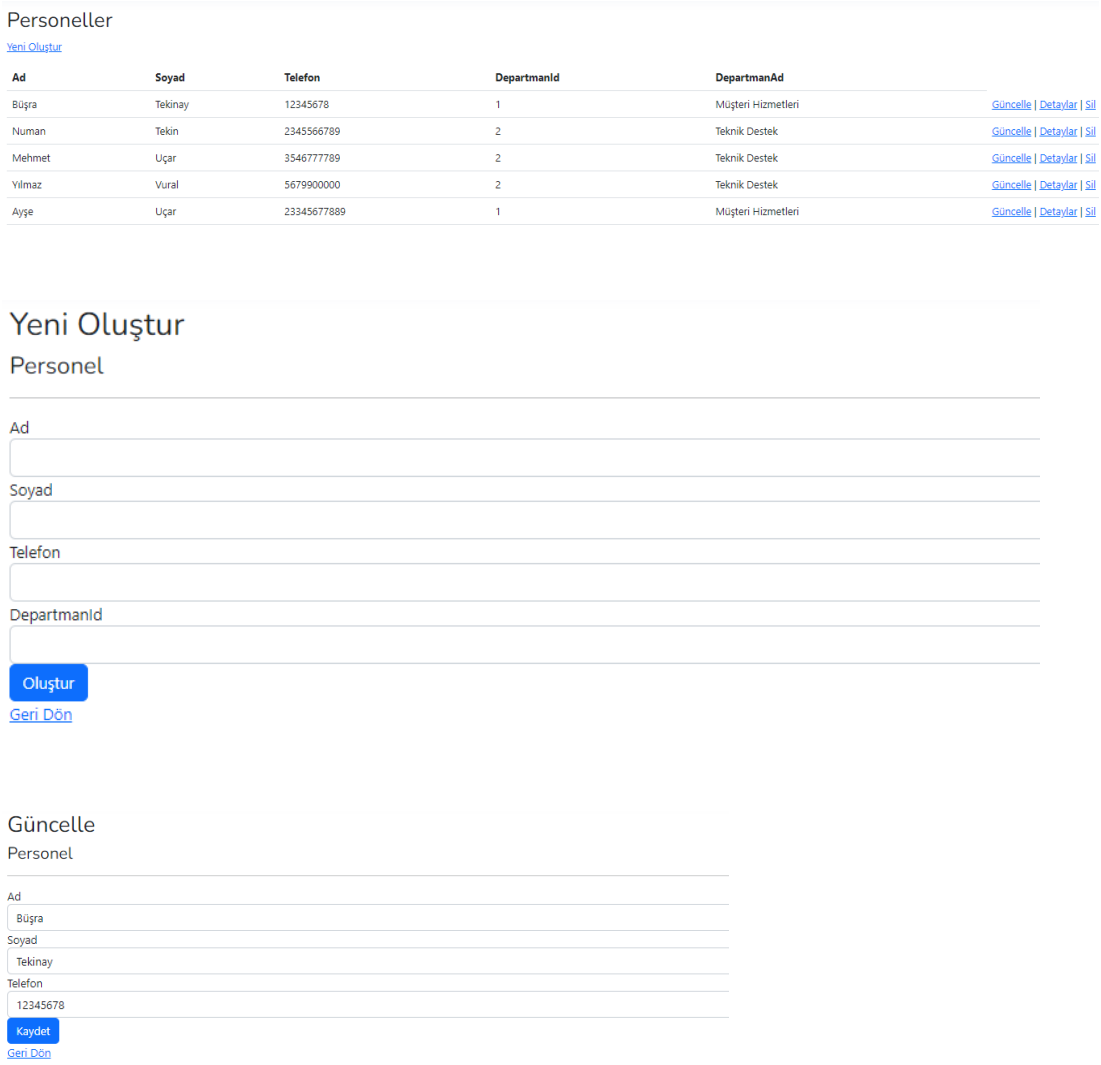
Admin yetkilisi departmanlarla ilgili bilgileri bu alanda doldurmaktadır.



Şekil 3.10 : Admin Paneli – Departman Bilgileri

3.2.1.4 Personeller

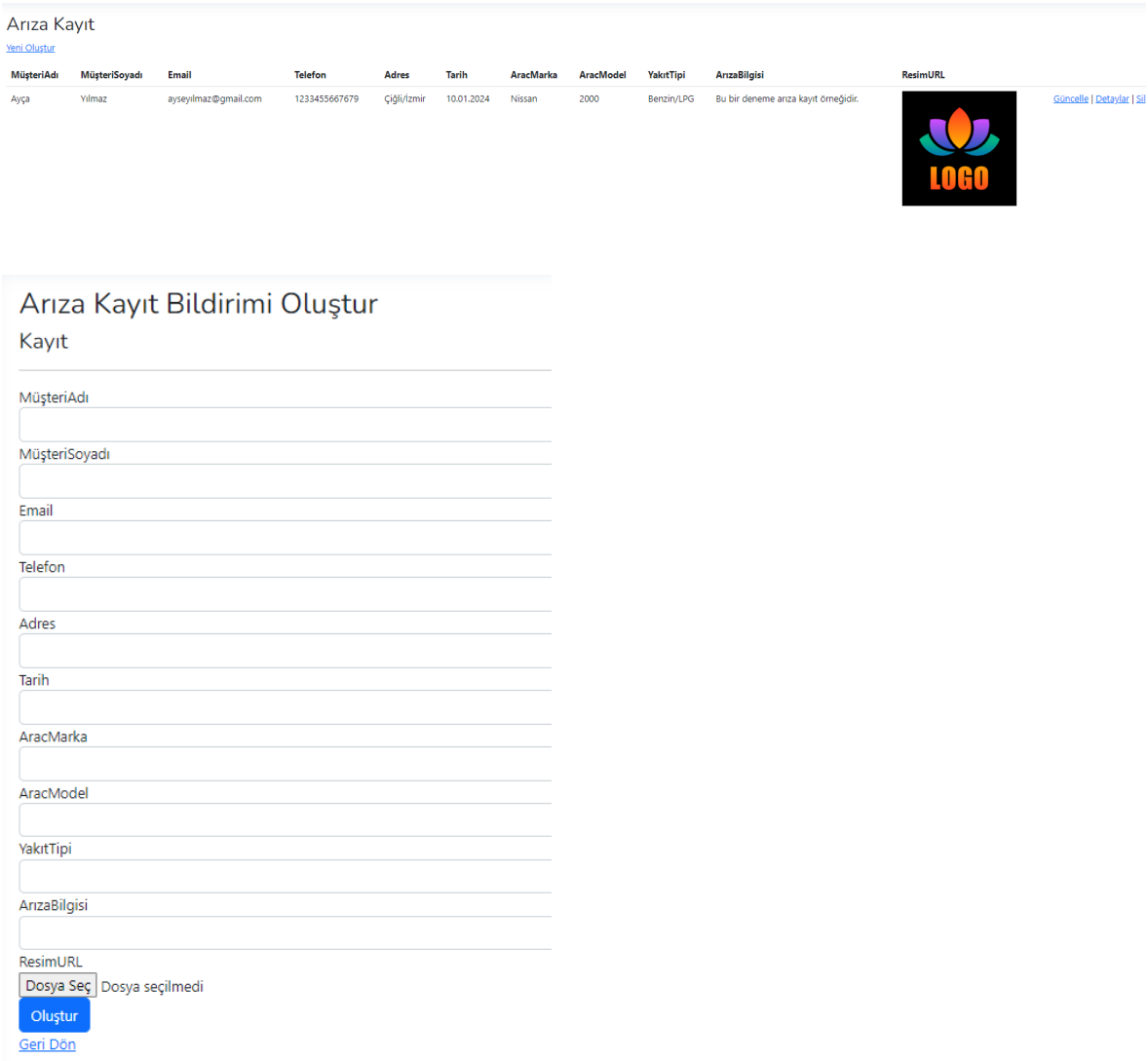
Admin yetkilisi personellerle ilgili bilgileri bu alanda doldurmaktadır.



Şekil 3.11 : Admin Paneli –Personel Bilgileri

3.3 Arıza Kayıt

Müşteri hizmetleri birimi, araç arızası ile ilgili gelen bildirim doğrultusunda müşterilerle ilgili bilgileri kayıt altına alır. Bu kayıt ilgili teknik destek ekibinin ilgilenmesi ile ilgilidir.

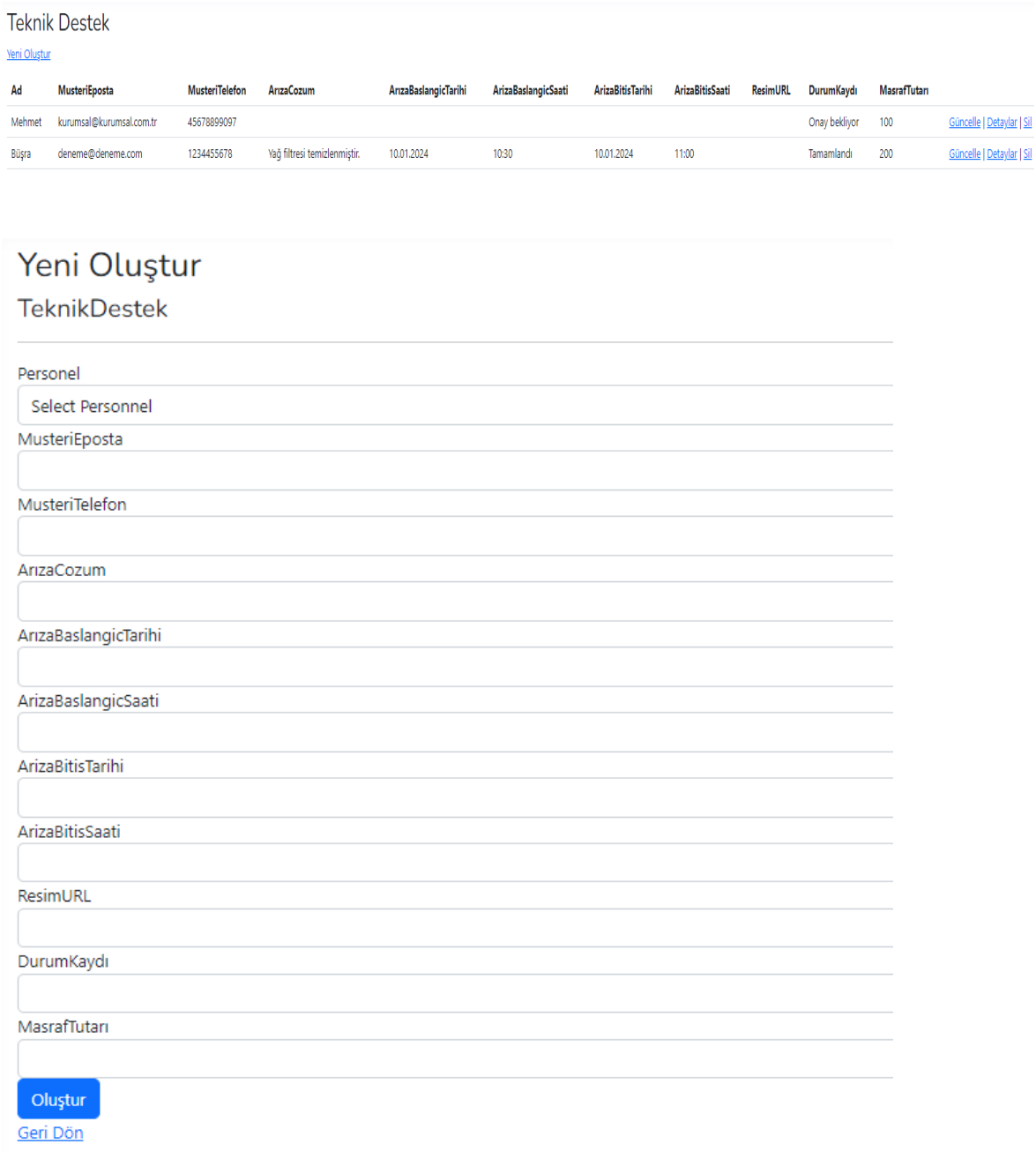


Şekil 3.12 : Admin Paneli – Arıza Kayıt Bilgileri

3.4 Teknik Destek

Arıza Kayıt kısmında arıza bildirimi ile ilgili sorunun belirtilmesinden sonra teknik destek birimi gelen mail bildiriminin ardından ilgili soruna destek olur. Sorunla

ilgilenen personel panel kısmında kendi kullanıcı bilgileri ile birlikte arıza ile ilgili bilgileri kayıt altına alır.

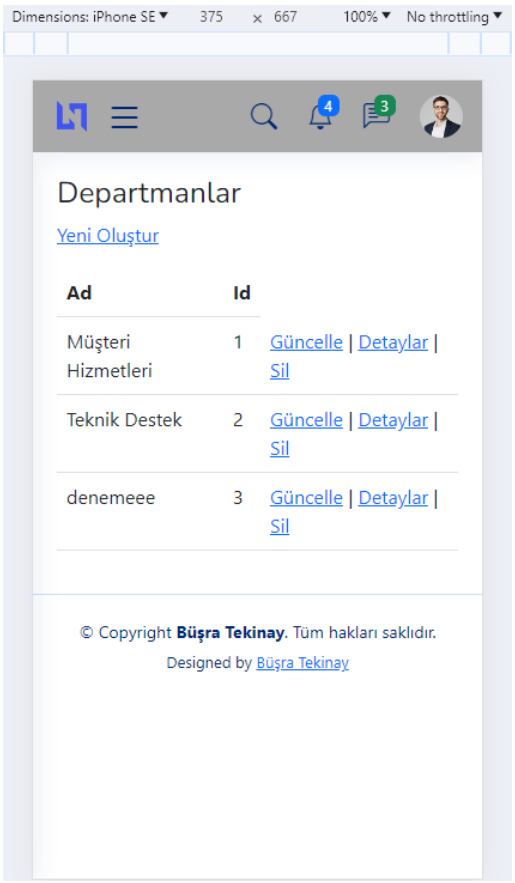


Şekil 3.13 : Admin Paneli – Teknik Destek Bilgileri

3.5 Responsive Tasarım

Responsive tasarım, web sitesinin farklı ekran boyutlarına ve cihazlara otomatik olarak uyum sağlamasını ifade eder. Bu, kullanıcıların masaüstü bilgisayarlar, tabletler ve akıllı telefonlar gibi çeşitli cihazlarda web sitesine erişebilmelerini sağlar.

Web sitesi tasarımında kullanılan responsive (duyarlı) tasarım, ziyaretçilerin farklı cihazlarda, özellikle mobil cihazlarda, en iyi kullanıcı deneyimini yaşamalarını sağlamak amacıyla geliştirilmiş bir yaklaşımdır. Projede, web sitesinin responsive tasarımı gerçekleştirilmiştir. Bu tasarım yaklaşımı, ziyaretçilerin web sitesini herhangi bir cihazda kolayca kullanmalarını sağlar, bu da genel kullanıcı memnuniyetini artırır ve sitenin erişilebilirliğini maksimum düzeyde tutar.



Şekil 3.14 : Responsive Tasarım

Şekil 3.15 : Responsive Tasarım

Bölüm 4

Sonuçlar

Bu proje, arıza araçlarına yönelik bir web tabanlı platformun geliştirilmesini ve bu platform üzerinden müşterilerin ilgili firma ile iletişime geçebilmesini amaçlamaktadır. Yönetim paneli(Admin Paneli) entegrasyonu sağlanmıştır. Ayrıca,

müşteri iletişimlerini, araç arızalarını, personel ve departman bilgilerini kaydedebilmek amacıyla, MVC mimarisini kullanarak Entity Framework Code First yaklaşımı ile bir veritabanı oluşturulmuştur. MVC mimarisi, kullanıcı arayüzü, iş mantığı ve veritabanı işlemlerini ayrı modüllerde organize ederek projenin düzenli bir şekilde geliştirilmesini sağlar. Bu mimari, arıza araçlarıyla ilgili bilgileri müşterilerin erişimine sunmak üzere tasarlanan web sitesinin temel yapısını oluşturmuştur.

Proje, Entity Framework Code First yaklaşımıyla bir veritabanı oluşturmayı içermiştir. Bu süreçte, C# programlama dilinde sınıflar kullanılarak veritabanı tablolarının modellemesi yapılmış, DbContext sınıfları ile bu sınıfların veritabanına eşlenmesi sağlanmıştır. Ardından, Code First Migration aracılığıyla bu sınıfların temel alındığı tabloların ve ilişkilerin veritabanına uygulanması gerçekleştirilmiştir. MVC mimarisine uygun olarak, Controller sınıfları oluşturularak iş mantığı bu sınıflara entegre edilmiştir. Controller sınıfları, kullanıcının web sitesi üzerindeki etkileşimlerini yönetirken, View sınıfları ise kullanıcı arayüzünü temsil etmiştir. Bu aşamada, CRUD (Create, Read, Update, Delete) operasyonlarını destekleyen Controller metotları oluşturularak, kullanıcıların veritabanındaki bilgilerle etkileşimde bulunması sağlanmıştır.

Entity Framework Code First yaklaşımı, veritabanı şemalarının kod tarafından tanımlanmasını ve daha sonra bu tanımlamaların kullanılarak veritabanının oluşturulmasını sağlar. Bu sayede, müşteri iletişim bilgileri, araç arızaları, departmanlar ve personel bilgileri gibi verilerin uygun bir şekilde tutulduğu bir veritabanı oluşturulmuştur. Oluşturulan web sitesi, müşterilerin arıza araçlarıyla ilgili bilgilere erişebilmelerine olanak tanımakta ve firma ile iletişime geçebilmelerini sağlamaktadır. Ayrıca, firmanın müşteri iletişimlerini ve araç arızalarını kaydedebilmesi için kullanılan veritabanı, Entity Framework Code First yaklaşımı ile başarılı bir şekilde oluşturulmuştur. Bu proje, araç arızalarıyla ilgili bilgilerin etkili bir şekilde yönetilmesini ve müşteri-firma iletişimini kolaylaştırmayı amaçlamaktadır. Ayrıca, MVC mimarisi ve Entity Framework Code First yaklaşımının bir araya getirilmesi, proje geliştirme sürecinde düzenlilik ve verimlilik sağlamıştır.

Bölüm 5

Tartışmalar

Bu proje, araç sahipleri ve servis uzmanları için araç arızalarını kaydetme ve yönetme süreçlerini optimize etmeyi amaçlayan bir web tabanlı uygulamayı içermektedir. Geliştirilen sistem, ASP.NET MVC (Model-View-Controller) mimarisi kullanılarak tasarlanmıştır ve kullanıcı dostu bir web sitesi ile birlikte yönetici panelini içermektedir. Bu sistem, araç bakım ve onarım süreçlerini daha etkili bir hale getirmeyi hedeflemektedir. Bu hedefler doğrultusunda sistem üzerinde yapılandırmalar yapılarak proje daha da iyileştirilebilir.

i. Admin paneli üzerinde daha kapsamlı raporlama araçları eklenerek, personel performansı ve departman verimliliği hakkında daha detaylı analizler sunulabilir.

ii. Sistem yöneticilerinin kullanım kolaylığını artırmak adına admin panelinin kullanıcı arayüzü üzerinde ergonomik düzenlemeler yapılabilir.

iii. Personel ve departman yönetimi için daha fazla seçenek ve yetki kontrolü eklenerek, yöneticilerin daha etkin bir şekilde sistem üzerinde kontrol sağlaması desteklenebilir.

iv. Her arıza kaydı için bir yorum alanı eklenerek kullanıcılar arasında bilgi paylaşımı teşvik edilebilir.

v. Kullanıcıların yaptığı yorumlar, admin paneli üzerinden yönetilebilir ve moderasyon süreçleri entegre edilerek olası istenmeyen içeriklerin kontrolü sağlanabilir.

vi. Web sitesinde araç bakımı, onarım teknikleri ve güncel sektör haberleri gibi konuları içeren blog yazıları eklenerek, kullanıcılara bilgi sunulabilir.

vii. Blog yazıları, SEO stratejileri ile entegre edilerek web sitesinin görünürlüğünü artırabilir ve kullanıcıların web sitesini düzenli olarak ziyaret etmeleri teşvik edilebilir.

viii. Personellerin teknik arızaya müdahale sırasında daha etkili olmalarını sağlamak amacıyla, personellerin konumlarını gösteren bir harita desteği eklenerek, en yakın personelin atanması ve arıza yerine daha hızlı ulaşım sağlanabilir.

ix. Harita üzerinden personellerin güzergah planlaması yapmalarını ve trafik durumunu takip etmelerini sağlayarak, sahada geçirilen süreyi optimize etmek mümkün olabilir.

x. Teknik arıza bildirimi alındığında, ilgili personel için otomatik bir bildirim sistemi eklenerek, hızlı bir müdahalenin gerçekleşmesi sağlanabilir.

xi. Personellerin teknik arızalara daha hızlı ve etkili bir şekilde müdahale edebilmeleri için mobil uygulama entegrasyonu düşünülebilir. Böylece, sahada iken bile arızaların yönetimi kolaylaştırılabilir.

Bu iyileştirme önerileri, projenin gelecekteki versiyonlarına yön verebilecek ve sistemin kullanıcı deneyimini, yönetilebilirliğini ve etkinliğini artırabilecek potansiyeli içermektedir. Bu geliştirmeler, projenin daha geniş bir kullanıcı kitlesi tarafından benimsenmesine ve endüstri standartlarını aşmasına katkı sağlayabilir.

Kaynaklar

Ahmed, M., Garrett, C., Faircloth, J., Payne, C., Lee, W. M., Ortiz, J. (2002). ASP.Net Web Developer's Guide. DotThatCom.com.

Milroy, S., Cox, K., Safford, D., Barker, L., Kalani, A., & Lee, W. M. (2002). Introduction to the Microsoft .NET Framework. In .NET Mobile Web Developer's Guide (pp. 59-97). .NET Developers Series.

A Freeman, S Sanderson, Pro ASP.NET MVC 3 Framework, Apress, 2011.

Glenn E. Krasner, Stephen T. Pope, "A cookbook for using the model-view controller user interface paradigm in Smalltalk-80", Journal of Object-Oriented Programming, vol. 1, no. 3, 1988, pp. 26-49.

J. Galloway, P. Haack, B. Wilson și K. S. Allen, Professional ASP.NET MVC 3, John Wiley & Sons, Inc., 2011.

Quinton, É. (2017). Using the MVC Model to Structure the Application. In Safety of Web Applications: Risks, Encryption and Handling Vulnerabilities with PHP (pp. 177-188).

Badurowicz, M, "Mvc Architectural Pattern In Mobile Web-Applications", Actual Problems Of Economics, 2011, pp.305-309.

J. Lerman, Programming Entity Framework, O'Reilly, 2010.

".NET Framework Conceptual Overview", Internet: http://msdn.microsoft.com/en-us/library/zw4w595w(v=VS.100).aspx

Peretiatko, M., Shirokopetleva, M., Lesna, N. (2022) "Research of methods to support data migration between relational and document data storage models", Innovative technologies and scientific solutions for industries, No. 2 (20), P. 64–74.