**DÜZCE ÜNİVERSİTESİ**

**TEKNOLOJİ FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ**

**BİTİRME TEZİ ÇALIŞMASI**

**JetForm Anket Uygulaması**

**172120013 – Ömer ESER**

**Mayıs 2021**

# **İÇİNDEKİLER DİZİNİ**

[**İÇİNDEKİLER DİZİNİ** 2](#_Toc72877747)

[**ŞEKİLLER DİZİNİ** 4](#_Toc72877748)

[**BÖLÜM 1** 5](#_Toc72877749)

[**GİRİŞ** 5](#_Toc72877750)

[**BÖLÜM 2** 6](#_Toc72877751)

[**JetForm Anket Uygulaması** 6](#_Toc72877752)

[**2.1 .NET 5** 6](#_Toc72877753)

[**2.2 Entity Framework Core** 6](#_Toc72877754)

[**2.3 Kontrolün Tersine Çevrilmesi (Inversion of Control-IoC)** 6](#_Toc72877755)

[**2.4 Görünüşe Yönelik Programlama (Aspect Oriented Programming- AOP)** 7](#_Toc72877756)

[**2.5 Yakalayıcılar (Interceptors)** 7](#_Toc72877757)

[**2.6 Doğrulama (Fluent Validation)** 7](#_Toc72877758)

[**2.7 Bağımlılık Enjeksiyonu (Dependency Injection)** 8](#_Toc72877759)

[**2.8 AutoFac** 8](#_Toc72877760)

[**2.9 Dil ile Tümleşik Sorgu (Language-Integrated Query-Linq)** 8](#_Toc72877761)

[**2.10 ASP .NET Web Api** 9](#_Toc72877762)

[**2.11 Blazor** 9](#_Toc72877763)

[**2.12 Microsoft Visual Studio** 9](#_Toc72877764)

[**2.13 MS SQL Server** 10](#_Toc72877765)

[**2.14 Postman** 10](#_Toc72877766)

[**2.15 Swager** 10](#_Toc72877767)

[**2.13 Çalışmanın Gerçekleştirilmesi** 11](#_Toc72877768)

[**2.13.1 SOLID Prensibi** 11](#_Toc72877769)

[**2.13.2 DRY (Kendini Tekrar Etme-Don’t Repeat Yourself) Prensibi** 11](#_Toc72877770)

[**2.13.3 Temiz Kod Yazma Teknikleri (Clean Code)** 11](#_Toc72877771)

[**2.13.4 Çekirdek Katmanı** 12](#_Toc72877772)

[**2.13.5 Güvenlik** 13](#_Toc72877773)

[**2.13.6 JetForm Anket Uygulaması** 14](#_Toc72877774)

[**2.13.6 Amaç** 14](#_Toc72877775)

[**2.13.7 Uygulamaya Erişim** 14](#_Toc72877776)

[**2.13.8 Formlar** 14](#_Toc72877777)

[**2.13.7 Sisteme Kayıt** 15](#_Toc72877778)

[**BÖLÜM 3** 16](#_Toc72877779)

[**SONUÇ** 16](#_Toc72877780)

[**KAYNAKLAR** 17](#_Toc72877781)

# **ŞEKİLLER DİZİNİ**

[Şekil 1 Swager Dökümantasyon Örneği. 10](#_Toc72766826)

[Şekil 2 Geliştirilen Sistemin Blok Şeması. 12](#_Toc72766827)

[Şekil 3 Kullanıcı Girişi Akış Diyagramı. 13](#_Toc72766828)

# **BÖLÜM 1**

## **GİRİŞ**

Bu çalışmada Blazor platformunda, Asp .Net Web Api aracılığı ile iletişim kuran, Azure platformunda yürütülmesi gerçekleştirilmiştir. Çalışma gerçekleştirilirken olabildiğince en güncel olan teknolojiler, standartlar araştırılarak uygulanması hedeflenmiştir. Ağırlıklı olarak kodlama teknikleri, mimari oluşturmalar işlenmiştir. Güncel teknolojiler ve kod yazma kuralları birlikte işlendiğinde ortaya çıkan çalışmanın, spagetti olarak adlandırılan kod yapılarından uzak olup, olabildiğince anlaşılır ve uyarlanabilir olduğu gözlemlenmiştir.

# **BÖLÜM 2**

## **JetForm Anket Uygulaması**

## **2.1 .NET 5**

.NET 5, masaüstü, web, mobil, bulut gibi farklı platformlar üzerinde çalışan, kısacası farklı çerçeveleri birleştiren, kod karmaşıklığını azaltan ve platformlar arası erişimi önemli ölçüde kolaylaştıran bir platform güncellemesidir. Bu güncellemenin gelmesiyle birlikte eskiye yönelik ASP.NET Web Forms ve Windows Workflow (WF) desteklenmemektedir. Bunlara alternatif olarak ASP.NET Core Blazor, Razor Pages ve CoreWF kullanılabilir [1].

## **2.2 Entity Framework Core**

Entity Framework (EF) Core, Entity Framework ver erişim teknolojisinin, sade, daha geniş çalışma alanı sunan açık kaynaklı ve farklı platformlarda kullanılabilen bir teknolojidir.

EF Core ile birlikte, mevcut bir veri tabanından model oluşturma (database first), veri tabanı eşleştirmeleri (model first) veya proje esnasında önce kod (code first) yaklaşımları kullanılabilir [2].

EF Core teknolojisinin bize sunduğu bir başka imkân ise dil tümleşik sorguları (language integrated query-LINQ)dır. Linq veri tabanı ile C# arasında entegre olarak çalışan sorgular yazmamıza olanak sağlar.

## **2.3 Kontrolün Tersine Çevrilmesi (Inversion of Control-IoC)**

Tam adı ile kontrolün tersine çevrilmesi, bir tasarım prensibidir. Projeler derinleştikçe ve temiz kod yazma kurallarına bağlı kalmaya çalıştıkça, eklenen obje durumlarının (instance) yönetimi zorlaşmaktadır. Bu prensip ile geliştirilen çerçevelerin avantajları ise şunlardır

* Farklı implementasyonlar arasında kolayca geçiş yapmamıza olanak sağlar,
* Projelerin modülerliğini arttırır,
* Bağımlılıkları minimum seviyeye indirdiği için test etmeyi/yazmayı kolaylaştırır [3]

## **2.4 Görünüşe Yönelik Programlama (Aspect Oriented Programming- AOP)**

Projeler derinleştikçe katman bağımsız, projeyi dikine kesen fonksiyonlarımız (securty, caching, exeption, logging vs.) oluşmaktadır. Aop prensibinin çıkış noktası, projeyi dikine kesen bu fonksiyonlardır. Geliştirilen kodun modülerliği, okunabilirliği, bakımı gibi konularda bizlere kolaylık sağlamaktadır. Bu prensibin temel amacı kod tekrarını engelleyerek sistemden soyutlayarak birden fazla yerde kullanmamızı sağlamaktır.

Aop prensibi ile kayıt tutma, doğrulama işlemleri, istisna işleme gibi işlemleri modüller haline getirerek fonksiyonlara özellik (attribute) ekleme şeklinde tek satır işlemlerle birden fazla işlevi yerine getirebiliyoruz.[4].

## **2.5 Yakalayıcılar (Interceptors)**

Yakalayıcılar, uygulamanın çalışma esnasında, çalıştığımız herhangi bir nesne hakkında bilgi toplayabileceğimiz, bu toplanılan bilgiye göre istediğimiz zaman bir metodu çalıştırabiliriz. Yakalayıcılara sıklıkla ihtiyaç duyduğumuz durumlardan bir tanesi de çalışma anında hangi nesne ile çalışacağımızı bilmiyor oluşumuzdur. Kullanıcının girdiği değerlere göre, projenin yaşam döngüsü bitmeden ne yapmamız gerektiğine karar verebiliyoruz.

Başka bir kullanım alanı ise, nesnelerin özellikleri, varsa özellikleri (attribute) ve bunların parametreleri gibi bilgileri alabilmemizdir.Yakalayıcıların bu özellikleri bize aop programlama prensibini uygulamamıza olanak sağlamıştır [5][6].

## **2.6 Doğrulama (Fluent Validation)**

Fluent Validation, .NET kütüphanesi içerisinde bulunan doğrulama sınıflarından en çok tercih edilen bir kütüphanedir.

Geliştirilen uygulamalarda, kullanıcıdan alınan verilerin doğruluğu ve zararsız olmasının kontrolü (sql injection) gibi konularda kullanılan bir teknolojidir. Bu teknolojinin önemli bir özelliği ise server-side doğrulama yapabilmemizdir.

Fluent Validation özellikler (attribute) ile kullanılır. ASP.NET CORE içerisinde bulunan metotlar haricinde, kullanıcı kendi geliştirdiği özellikleri (attribute) de ekleyebilir, bu özellikleri(attribute) özelleştirebilir [7].

## **2.7 Bağımlılık Enjeksiyonu (Dependency Injection)**

Bağımlılık enjeksiyonu, bir nesnenin bağlı olduğu alt nesneleri almasını sağlayan bir tekniktir. Bu alt nesnelere bağımlılıklar denir. Bağımlılık enjeksiyonu altındaki amaç, ilişkili nesne kullanımında bağımlılıkları yönetmektir.

Bağımlılık enjeksiyonu deseninin faydalarından biri ise, bağlılık oluşturulacak nesneleri doğrudan oluşturmak yerine, bu nesnelerin dışardan verilmesi ile sistem içerisindeki bağımlılığı minimize edebiliyor olmamızdır. Bunun sonucunda ise bağımlılıkları bulunan sınıfların/nesnelerin üzerindeki değişikliklerden korunuruz [8].

## **2.8 AutoFac**

AutoFac, .NET için yazılmış bir IoC kapsayıcısıdır. Çok az altyapı ve entegrasyon kodu, daha basit kullanımı ve çeşitli uygulama tasarımını desteklemektedir. Bağımlılık enjeksiyonu gibi teknikleri kullanmamız için altyapıyı bizim adımıza ayarlayıp daha sade bir kullanım sunmaktadır [9].

## **2.9 Dil ile Tümleşik Sorgu (Language-Integrated Query-Linq)**

Program ve veri arasındaki ilişkiyi başka bir boyuta taşıyan devrimsel bir programlama metodolojisidir. LINQ, bir programlama ara yüzü sunar. C# diline getirdiği eklentilerle SQL benzeri tek bir söz dizimi ile farklı tiplerdeki verilerin sorgulanmasını sağlar [10].

## **2.10 ASP .NET Web Api**

ASP .NET Web Api, farklı türde istemciler (tarayıcılar, mobil telefonlar, tabletler, bilgisayarlar vs.) tarafından özelleştirilebilen http protokolü üzerinden iletişim sağlayan servisler oluşturmak için kullanılabilen bir çerçevedir(framework). Asp .NET MVC ile birçok ortak özelliğe sahip olduklarından birtakım benzerlikler göstermektedir fakat MVC çerçevenin (framework) bir parçası değildir.

ASP .NET Web Api Core, ASP .NET’ in bir parçasıdır ve MVC veya diğer web aplikasyonları ile birlikte kullanılabilir. Açık kaynaklı olması nedeni ile sürekli olarak geliştirilip, yeni özellikler eklenmektedir [11][12].

## **2.11 Blazor**

Blazor, açık kaynaklı bir web çerçevesidir. C# ve HTML kullanarak web uygulamaları geliştirmemize olanak sağlamaktadır.

Blazor ile tek sayfa aplikasyonları (single page application) oluştururz. Günümüzde alternatif olarak kullanılan teknolojilere angular veye react örnek olarak verilebilir. 2018 yılında duyrulan Blazor, .NET ile interaktif bir şekilde kullanılabilen istemci tabanlı bir web kullanıcı arayüzü oluşturmamızı sağlayan bir çerçevedir.

Blazorun bize sağladığı olanaklardan örnek vermek gerekirse, istemcinin kullanması için yazdığımız C# kodunu bir eklentiye ihtiyaç duymadan ya da dönüştürme işlemleri gerçekleştirmeden tüm tarayıcılarda çalışabilir şekilde olmasıdır. Kısaca özetlemek gerekirse, arka plan için yazılmış kodlar ile geliştirme yapabiliyor olmamıza imkân sağlamasıdır [13].

## **2.12 Microsoft Visual Studio**

Microsoft Visual Studio, Microsoft tarafından geliştirilen bir geliştirme ortamıdır (IDE). Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework ve Microsoft Silverlight tarafından desteklenen tüm platformlar için yönetilen kod ile birlikte yerel kod ve Windows Forms uygulamaları, web siteleri, web uygulamaları ve web servisleri ile birlikte konsol ve grafiksel kullanıcı arayüzü uygulamaları geliştirmek için kullanılır

## **2.13 MS SQL Server**

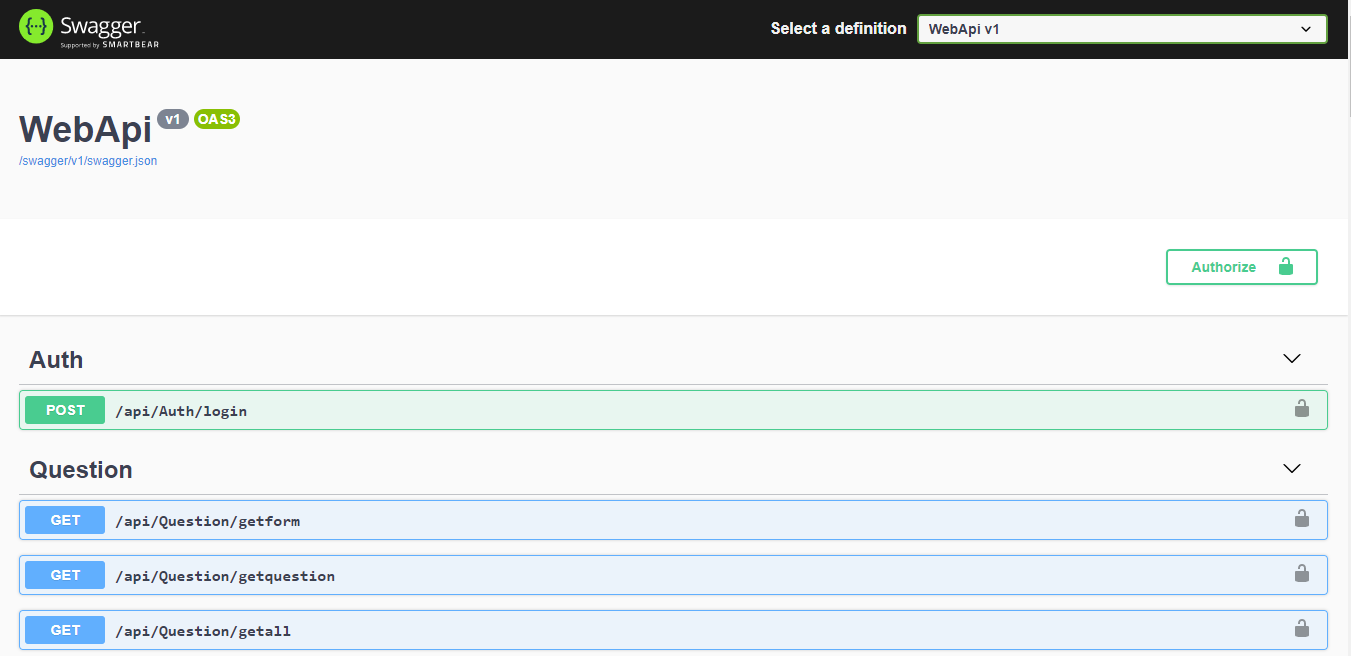
MS SQL Server, kurumsal çaplı bir ilişkisel veri tabanı yönetim sistemidir. Birbiri ile ilişkili verileri, sistematik bir şekilde depolamamıza ve yönetmemize olanak sağlamaktadır.

## **2.14 Postman**

Postman, oluşturduğumuz apileri test etmemiz için tasarlanmış kullanışlı bir istemcidir. Oldukça kullanışlı olan arayüzü ile geliştirdiğimiz apileri hızlıca test etmemize olanak sağlamaktadır. Aynı zamanda istekler arasında veri aktarımı yapıp, bir apinin çıktısını diğerinin girdisi olarak ayarlayıp geliştirmeler yapmamızı sağlayan bir uygulamadır [14].

## **2.15 Swager**

Swager, postman benzeri bir istemcidir. Postmandan farklı olarak apimizde yazdığımız metotları bir doküman haline getirerek, nasıl kullanıldığını, hangi formatları içerdiğini, ne sonuç döndürdüğünü bize göstermektedir. Bu oluşturulan dökümantasyon geliştiriciler için çok fazla fayda sağlamaktadır. Bahsedilen örnek doküman şekil 1 de gösterilmiştir [15].



Şekil 1 Swager Dökümantasyon Örneği.

## **2.13 Çalışmanın Gerçekleştirilmesi**

Proje geliştirirken uymamız gereken birtakım kurallar vardır. Bu kurallar, çalışma esnasında uygulamayı kıracak şekilde hataya yol açmıyor olsa da uygulamanın derinleşmesi, kod okunurluğuna, spagetti kod oluşumuna, farklı yerlerde birbirinin aynısı metotların yazılmasına sebebiyet verebilir.

Bu kurallar çerçevesinde oluşturulan proje mimarilerinde standart CRUD (okuma, güncelleme ve silme, veri depolama) operasyonları çok zahmetsiz şekilde gerçekleştirilebilir. Bunun yanı sıra şirketler için kurumsal hafıza oluşturup, bu kurallar ile hazırlanmış mimariler, yeni eleman alımlarında o kişinin şirket projelerine dahil olma süresini kısaltacaktır.

### **2.13.1 SOLID Prensibi**

Solid, yazılım geliştirme kurallarını kapsayan bir kısaltmadır. Bu kurallar;

* S: Tek Sorumluluk Prensibi (Single Responsibility Principle)
* O: Açık-Kapalı Prensibi (Open-Closed Principle)
* L: Liskov’un Yerine Geçme Prensibi(Liskov Substitution Principle )
* I: Arayüz Ayırma Prensibi (Interface segregation principle)
* D: Bağımlılıkları Tersine Çevirme Prensibi (Dependency Inversion Principle)

### **2.13.2 DRY (Kendini Tekrar Etme-Don’t Repeat Yourself) Prensibi**

Uygulama geliştirirken bir kod parçacığını birden fazla yerde kullanma durumunda, bu kod parçacığının genel bir metot içerisine alınmasını öneren bir prensiptir. Bu prensibin uygulanmaması, projemizi, spagetti kod dediğimiz anlaşılması güç, okunurluğu zor olan bir hale sokacaktır.

### **2.13.3 Temiz Kod Yazma Teknikleri (Clean Code)**

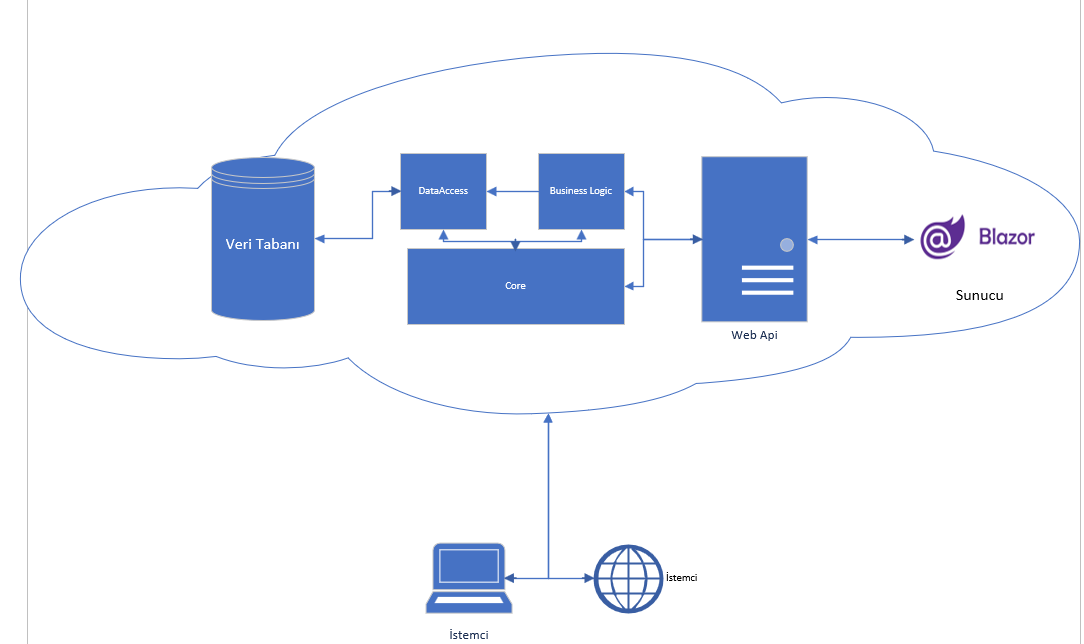
Temiz kod yazılan kodun anlaşılması ve üzerinde yeni geliştirmeler yapılması kolay olan, okuyucu odaklı bir geliştirme biçimidir. Bir projeye başlarken geliştirme hızının genellikle yüksek olduğu hepimizin malumudur. Projeler büyüdükçe hantallaşır ve bu bağlamda hareket kabiliyetini kaybetmeye başlar. Yeniliklere adapte olarak yeni özellikler sunabilmek, çıkabilecek muhtemel sorunları giderebilmek gibi birçok ihtiyaç doğar. Bunun yanı sıra ekiplerin değişmesi ve büyümesiyle birlikte kodların okunaklı ve temiz yazılmış olması birim zaman maliyetinde çok ciddi bir tasarrufa neden olacaktır. Böylece geliştirilebilirlik ve devamlılık özelliği sağlanmış olur [16].

Yukarıda bahsedilen prensipler doğrultusunda hazırlanan çalışma sonucunda, her ne kadar karmaşık ve çok fazla konu başlığı mevcut olsa da projenin iskeletini oluşturduktan sonra geliştirmeler yapmak çok daha basit hale geliyor.

### **2.13.4 Çekirdek Katmanı**

Bu prensipler/teknolojiler doğrultusunda geliştirilmiş olan çekirdek (Core) katmanı içerisinde temel CRUD operasyonları, görünüş (aspect) işlemleri, önbellek (cache) yönetimi gibi birçok metot/sınıf bulunmaktadır. Bu çekirdek katmanının en önemli özelliği ise, bir defa kurulduktan sonra istenilen tüm projeler için kullanılabilir olmasıdır.

Gerekli mimari kurulduktan sonra projeyi bulut sistemlerine aktararak, bulut üzerinden sorunsuzca çalışır hale getirebiliriz. Şekil 2 de verilen blok şemada sistemin haberleşmesi gösterilmiştir.

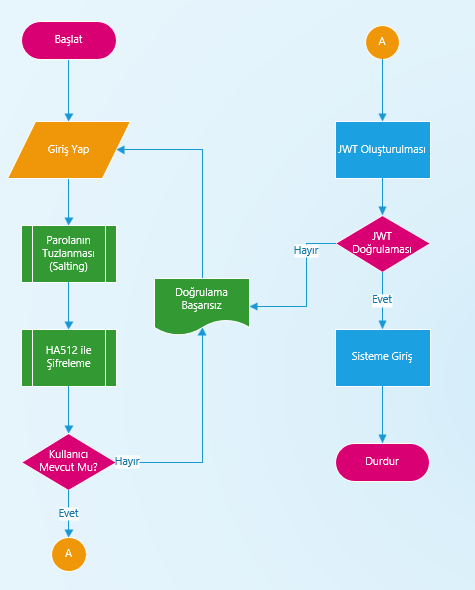


Şekil 2 Geliştirilen Sistemin Blok Şeması.

### **2.13.5 Güvenlik**

Projenin güvenlik aşaması JWT ile sağlanmıştır. JWT, IETF tarafından tasarlanmış, haberleşen iki veya daha fazla sistem arasında doğrulama işlemlerini yapan bir jeton biçimidir. Base64 biçiminde kodlanmış, 3 ayrı JSON parçasından oluşur.

Kullanıcı parolaları ise veri tabanında şifreli olarak tutulmaktadır. Geliştirici tarafından oluşturulan bir anahtar ile tuzlanan (salting) parolalar, HA512 şifreleme algoritması ile şifrelenip veri tabanına kaydedilmektedir. Kullanıcı girişi için gerçekleşen akış diyagramı Şekil 3 de verilmiştir. Sisteme giriş yapılmadığı taktirde, başka sayfalara geçiş yapılması engellenmiştir.



Şekil 3 Kullanıcı Girişi Akış Diyagramı.

### **2.13.6 JetForm Anket Uygulaması**

JetForm, sistem yöneticisinin formlar oluşturarak sorular hazırlayabileceği, bu soruları çoktan seçmeli, açılır menü ve onay kutusu şeklinde düzenleyebileceği, hazırlanan formların istenilen kullanıcıya sunulması, formları cevaplayan kullanıcıların sonradan cevaplarını, doğru yanlış durumlarını görmesi ve sistem yöneticisinin bu bilgileri inceleyebileceği bir sistemdir.

FS, sistem yöneticisinin son kullanıcıya formlar hazırlayabileceği, kullanıcıların doldurdukları formları inceleyebileceği/değerlendirebileceği bir sistemdir.

### **2.13.6 Amaç**

Kişi veya kuruluşların, ilgili kişilere sunacağı formlar ile, ilgili kişilerin görüşlerini toplaması, yeterlilik testleri yapabilmesi, anketler oluşturarak yapılacak olan işlerin planlamasını yapabilmeyi sağlamaktır.

### **2.13.7 Uygulamaya Erişim**

Uygulama herhangi bir kurulum gerektirmemekle birlikte bulut tabanlı olduğundan dolayı, internet erişimi olan herhangi bir cihaz ile uygulamaya erişilebilir.

Aynı zamanda kaynak kodlarına GitHub üzerinden github.com/EserOmer/Bitirme-Tezi adresine giderek ulaşılabiliriz.

### **2.13.8 Formlar**

Yöneticiler formları istedikleri gibi tasarlamakta özgürdürler. Oluşturulan sorularda çoktan seçmeli, onay kutuları, kısa metin, paragraf gibi farklı türlerde yanıtlar oluşturabileceklerdir. Soruların sırasını değiştirebilecek, istedikleri kadar seçenek ekleyebileceklerdir.

Önceden hazırlanıp paylaşılmış olan formların yanıtlarını ise isterlerse tablo halinde görüntüleyebileceklerdir.

### **2.13.7 Sisteme Kayıt**

Herkes şahıs olarak ya da şirket adına başvurabilir. Uygulamada üyelik kısıtlaması bulunmamaktadır. Her şahıs ya da şirket (bundan sonra yönetici olarak bahsedilecektir.) kendi üyeliklerinde istedikleri formları oluşturmada özgürdür. Oluşturulan formlar yöneticiler ve yöneticilerin davet gönderdiği kişiler tarafından tekrar düzenletilebilmektedir.

### **2.13.8 Sisteme Giriş**

<https://omereser.info.tr/login> adresine girdikten sonra açılan giriş sayfasına, test amaçlı oluşturulmuş olan [test@test.test](mailto:test@test.test) kullanıcı adı ve Test.12 parolası ile giriş yapabilir ya da yeni bir üyelik oluşturarak, oluşturmuş olduğunuz kullanıcı adı ve şifre ile sisteme erişebilirsiniz.

# **BÖLÜM 3**

## **SONUÇ**

Bu çalışmada Asp .NET Web Api ve Blazor kullanılarak bir form uygulaması yapılmıştır. Çalışmada öncelik olarak proje mimarileri kullanılmıştır. Uygulanan kurallar sonucu kurulmuş olan mimarinin, PnP (Tak Çalıştır) olması ve geliştirmeye açık olduğu gibi Core katmanının da bütün projelere entegre edilerek hem büyük oranda zaman kazancı hem de kurumsal hafıza/kod standardı oluşturma konusunda büyük avantajlar sağladığı gözlemlenmiştir.

Bu çalışmaya ek olarak Web Api sayesinde bütün plaformlarda aynı proje yürütülebilir, platform bağımsız çalışılabilir. Çalışmanın temellerinde yer alan teknikler birçok kurumsal ölçekli firmalarda kullanıldığı gibi uzun doğacak çerçeve (framework) güncellemeleri/değişimleri karşısında esnek yapısı sayesinde sıkıntısız bir yeniden yapılandırma sürecine girilebilir. Veri tabanı bağlantısının esnekliği sayesinde farklı türdeki veri tabanlarına geçiş konusunda hızlı ilerleme sağlanılabilir.

# **KAYNAKLAR**

[1] <https://docs.microsoft.com/tr-tr/dotnet/core/dotnet-five>

[2] <https://docs.microsoft.com/tr-tr/ef/core>

[3] <https://gokhana.medium.com/inversion-of-control-ioc-nedir-ve-avantajları-nelerdir-cf05e42c16e4>

[4] <https://devnot.com/2020/aspect-oriented-programming/>

[5] https://www.btkakademi.gov.tr/portal/course/player/deliver/c--7008

[6] <https://docs.microsoft.com/en-us/ef/core/logging-events-diagnostics/interceptors>

[7] <https://docs.fluentvalidation.net/en/latest/>

[8] <https://gokhana.medium.com/dependency-injection-nedir-nasıl-uygulanır-kod-örneğiyle-44f4b0d576e4>

[9] <https://www.codeproject.com/Articles/25380/Dependency-Injection-with-Autofac>

[10] <https://docs.microsoft.com/tr-tr/dotnet/csharp/programming-guide/concepts/linq/>

[11] <https://docs.microsoft.com/tr-tr/aspnet/core/web-api/?view=aspnetcore-5.0>

[12] <https://www.tutorialsteacher.com/webapi/what-is-web-api>

[13] <https://medium.com/kodcular/asp-net-blazor-teknolojisi-c11ede7cddd9>

[14] <https://medium.com/kodcular/postman-nedir-ce39bef14db7>

[15] <https://medium.com/android-türkiye/swagger-nedir-ne-i̇şe-yarar-e8c12a9e9e7f>

[16] Martin, Robert C. CleanCode, Boston: Pearson Education, Inc, 2009