**T.C.**

**FIRAT ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**ENGELLI VATANDAŞLAR İÇIN QR KOD SISTEMI**

Muhammet İsmail YANIK

**Tez Danışmanı**

Dr. Ögr. Üyesi Erdal ÖZBAY

**BİTİRME TEZİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**ELAZIĞ - 2024**

**T.C.**

**FIRAT ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**ENGELLI VATANDAŞLAR İÇIN QR KOD SISTEMI**

**Muhammet İsmail YANIK**

**BİTİRME TEZİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

Bu bitirme tezi ….../……/2024 tarihinde, aşağıda belirtilen jüri tarafından oybirliği/oyçokluğu ile başarılı/başarısız olarak değerlendirilmiştir.

(imza) (imza) (imza)

Dr. Ögr. Üyesi Erdal ÖZBAY Üye Üye

ÖZGÜNLÜK BİLDİRİMİ

Bu çalışmada, başka kaynaklardan yapılan tüm alıntıların, ilgili kaynaklar referans gösterilerek açıkça belirtildiğini, alıntılar dışındaki bölümlerin, özellikle projenin ana konusunu oluşturan teorik çalışmaların ve yazılım/donanımın benim tarafımdan yapıldığını bildiririm.

Fırat Üniversitesi 12.01.2024

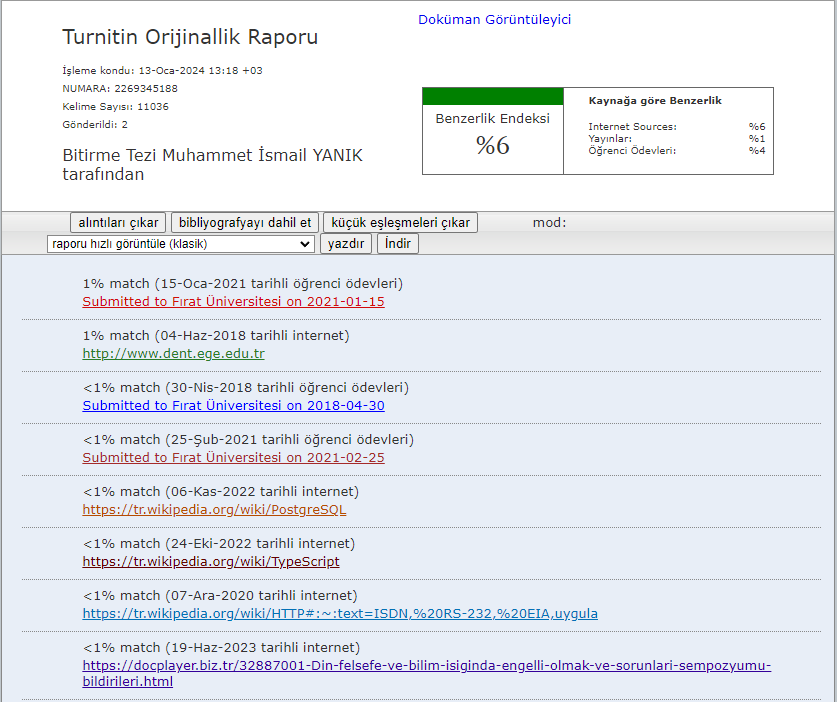
Bilgisayar Mühendisliği

23119 Elazığ

Öğrencilerin Adı Soyadı

Muhammet İsmail YANIK

BENZERLİK BİLDİRİMİ

****

TEŞEKKÜR

Çalışma sürem boyunca benden yardımlarını esirgemeyen çok değerli hocam Dr.Öğr.Üyesi ERDAL ÖZBAY’a ve bu çalışma ile birlikte bütün bir lisans hayatım boyunca yol göstererek yardımcı olan Prof. Dr.Mehmet KARAKÖSE’ye teşekkürlerimi sunarım.

**Muhammet İsmail YANIK**

**ELAZIĞ – 2024**

İÇİNDEKİLER

**Sayfa No**

[ÖZGÜNLÜK BİLDİRİMİ III](#_Toc155797251)

[BENZERLİK BİLDİRİMİ IV](#_Toc155797252)

[TEŞEKKÜR V](#_Toc155797253)

[İÇİNDEKİLER I](#_Toc155797254)

[ŞEKİLLER LİSTESİ IV](#_Toc155797255)

KISALTMALAR LİSTESİ……………………………………………………………………..VI

[ÖZET VI](#_Toc155797256)I

[ABSTRACT VII](#_Toc155797257)I

[1. GİRİŞ 1](#_Toc155797258)

[2. ENGELLİLİK 3](#_Toc155797259)

[1.1. Engellilik Nedir? 3](#_Toc155797260)

[1.1.1. Fizyolojik Engel: 3](#_Toc155797261)

[1.1.2. Zihinsel- Nörolojik Kontrol Problemi: 3](#_Toc155797262)

[1.1.3. Fiziksel (Bedensel) Engel: 4](#_Toc155797263)

[1.2. Engellinin Toplumdaki Yeri̇ 4](#_Toc155797264)

[2. KULLANILAN TEKNOLOJILER: 5](#_Toc155797265)

[2.1. QR Kod: 5](#_Toc155797266)

[2.2. HTML: 5](#_Toc155797267)

[2.2.1. HTML Ne İşe Yarar: 6](#_Toc155797268)

[2.2.2. HTML Kullanım Alanları: 6](#_Toc155797269)

[2.3. CSS: 6](#_Toc155797270)

[2.3.1. Dahili ekleme: 6](#_Toc155797271)

[2.3.2. Harici Ekleme: 7](#_Toc155797272)

[2.3.3. Satır İçi Stil Ekleme: 8](#_Toc155797273)

[2.3.4. CSS Ne İşe Yarar: 8](#_Toc155797274)

[2.3.5. CSS Kullanım Alanları: 8](#_Toc155797275)

[2.4. JavaScript: 8](#_Toc155797276)

[2.4.1. Java Script Ne İşe Yarar: 9](#_Toc155797277)

[2.4.2. Java Script Kullanım Alanları: 9](#_Toc155797278)

[2.5. Java: 9](#_Toc155797279)

[2.5.1. Java Ne işe yarar: 9](#_Toc155797280)

[2.5.2. Java Kullanım Alanları: 9](#_Toc155797281)

[2.5.3. Java ve JavaScript farkları: 10](#_Toc155797282)

[2.6. TypeScript: 10](#_Toc155797283)

[2.6.1. TypeScript Ne İşe Yarar: 11](#_Toc155797284)

[2.6.2. TypeScript Kullanım Alanları: 11](#_Toc155797285)

[2.7. Angular: 11](#_Toc155797286)

[2.7.1 Angular Ne İşe Yarar: 11](#_Toc155797287)

[2.7.2. Angular Kullanım Alanları: 11](#_Toc155797288)

[2.8. Spring Boot : 11](#_Toc155797289)

[2.8.1. Spring Boot Ne İşe Yarar: 12](#_Toc155797290)

[2.8.2. Spring Boot Nerelerde Kullanılır: 12](#_Toc155797291)

[2.9. SQL: 13](#_Toc155797292)

[2.9.1. SQL Ne İşe Yarar: 13](#_Toc155797293)

[2.9.2. SQL Kullanım alanları: 14](#_Toc155797294)

[2.10. Primeng: 14](#_Toc155797295)

[2.10.1. Primeng Ne İşe Yarar: 14](#_Toc155797296)

[2.10.2. Primeng Kullanım Alanları: 14](#_Toc155797297)

[2.11. Bootstrap: 15](#_Toc155797298)

[2.11.1. Bootstrap Ne İşe Yarar: 15](#_Toc155797299)

[2.12.Monolitik Mimari: 15](#_Toc155797300)

[2.12.1. Monolitik Mimari Ne İşe Yarar: 15](#_Toc155797301)

[2.12.2. Monolitik Mimarinin Kullanım Alanları: 16](#_Toc155797302)

[2.13. Mikroservis: 16](#_Toc155797303)

[2.13.1. Mikroservis Mimarisi Ne İşe Yarar: 17](#_Toc155797304)

[2.14. API: 18](#_Toc155797305)

[2.14.1. Mimari Yapılarına Göre API’ler 19](#_Toc155797306)

[2.14.1.1.REST API: 20](#_Toc155797307)

[2.14.1.2. SOAP: 20](#_Toc155797308)

[2.15. PostgreSQL: 20](#_Toc155797309)

[2.16. Http Protokolü: 21](#_Toc155797310)

[2.16.1. POST: 22](#_Toc155797311)

[2.16.2. PUT: 22](#_Toc155797312)

[2.16.2.HEAD: 22](#_Toc155797313)

[2.16.3.DELETE: 22](#_Toc155797314)

[2.16.4. CONNECT: 22](#_Toc155797315)

[2.16.5. OPTİONS: 23](#_Toc155797316)

[2.16.6. TRACE: 23](#_Toc155797317)

[3. YÖNTEM 24](#_Toc155797318)

[3.1. Veri Tabanının Tasarlanması ve geliştirilmesi: 26](#_Toc155797319)

[3.2. Ön Yüz Tarafının Tasarlanması ve Geliştirilmesi: 31](#_Toc155797320)

[3.3. Projenin Backend Tarafının Tasarlanması 42](#_Toc155797321)

[3.3.1. Api Geliştirilmesi 45](#_Toc155797322)

[4. SONUÇ 51](#_Toc155797323)

[5. KAYNAKLAR 52](#_Toc155797324)

[6. ÖZGEÇMİŞ 54](#_Toc155797325)

ŞEKİLLER LİSTESİ

**Sayfa No**

[**Şekil 2.1.** Dahili CSS Ekleme Örneği 7](#_Toc155797204)

[**Şekil 2.2.** Harici CSS Ekleme Örneği 7](#_Toc155797205)

[**Şekil 2.3.** CSS Dosya Örneği 8](#_Toc155797206)

[**Şekil 2.4.** Satır içi CSS ekleme örneği 8](#_Toc155797207)

[**Şekil 2.5.** Mikroservis mimarisi 16](#_Toc155797208)

[**Şekil 2.6.** Mikroservis mimarisinin farklı teknolojileri desteklemesi 17](#_Toc155797209)

[**Şekil 2.7.** Mikroservis mimarisi çalışma anlatımı 18](#_Toc155797210)

[**Şekil 2.8.** HTTP Protokolü 21](#_Toc155797211)

[**Şekil 3.1.** Kullanıcı erişim yetki şeması 25](#_Toc155797212)

[**Şekil 3.2.** Hasta kayıt örneği 27](#_Toc155797213)

[**Şekil 3.3.** Hastalık kayıt örneği 27](#_Toc155797214)

[**Şekil 3.4.** Hastaların hastalıklarının kayıt örneği 27](#_Toc155797215)

[**Şekil 3.5.** Kullanıcı kayıt örneği 29](#_Toc155797216)

[**Şekil 3.6.** Kullanıcı yetkileri kayıt örneği 29](#_Toc155797217)

[**Şekil 3.7.** Yetki kayıt örneği 29](#_Toc155797218)

[**Şekil 3.8.** Kullanıcı yetkisi kayıt anlatımı 30](#_Toc155797219)

[**Şekil 3.9.** Veri̇ tabanı ili̇şki̇ şeması 31](#_Toc155797220)

[**Şekil 3.10.** DTO örneği 33](#_Toc155797221)

[**Şekil 3.11.** component.ts içeri aktarımları 34](#_Toc155797222)

[**Şekil 3.12.** İstek yapılacak adresin tutulması 34](#_Toc155797223)

[**Şekil 3.13.** Component constructer’ı 34](#_Toc155797224)

[**Şekil 3.14.** Servis örneği 35](#_Toc155797225)

[**Şekil 3.15.** Hata yakalama 35](#_Toc155797226)

[**Şekil 3.16.** Interceptor çalışma anlatımı 36](#_Toc155797227)

[**Şekil 3.17.** Interceptor için içeri aktarımlar 36](#_Toc155797228)

[**Şekil 3.18.** Interceptor Kullanımı 37](#_Toc155797229)

[**Şekil 3.19.** Kullanıcı paneli form öğesi 38](#_Toc155797230)

[**Şekil 3.20.** Form öğesi HTML kodları 38](#_Toc155797231)

[**Şekil 3.21.** Kullanıcı bilgileri güncelleme 39](#_Toc155797232)

[**Şekil 3.22.** QR kod oluşturma aracı 40](#_Toc155797233)

[**Şekil 3.23.** QR kod oluşturma aracı HTML kodları 40](#_Toc155797234)

[**Şekil 3.24.** Tek yönlü data binding örneği 41](#_Toc155797235)

[**Şekil 3.25.** Yönlendirme için gerekli içeri aktarımlar 42](#_Toc155797236)

[**Şekil 3.26.** Yönlendirme örneği 42](#_Toc155797237)

[**Şekil 3.27.** Veri tabanı tablosunun java ortamında entity olarak tanımlanması 43](#_Toc155797238)

[**Şekil 3.28.** Sorgu örneği 44](#_Toc155797239)

[**Şekil 3.29.** Arkayüz için DTO nesnesi örneği 44](#_Toc155797240)

[**Şekil 3.30.** DTO dan entity nesnesine dönüşüm örneği 45](#_Toc155797241)

[**Şekil 3.31.** Entity den DTO nesnesine dönüşüm örneği 45](#_Toc155797242)

[**Şekil 3.32.** API için gerekli anotasyonların tanımlanması 46](#_Toc155797243)

[**Şekil 3.33.** Resource örneği 46](#_Toc155797244)

[**Şekil 3.34.** Servis örneği 47](#_Toc155797245)

[**Şekil 3.35.** Base servis 47](#_Toc155797246)

[**Şekil 3.36.** Servis metot örneği 48](#_Toc155797247)

[**Şekil 3.37.** Base servis metot örneği 48](#_Toc155797248)

[**Şekil 3.38.** JPA Repository örneği 49](#_Toc155797249)

[**Şekil 3.39.** Geliştirilen arkayüz’ün çalışma sisteminin anlatımı 49](#_Toc155797250)

**KISALTMALAR LİSTESİ**

**HTML:** Hyper Text LAnguage

**HTTP:** Hyper Texy Transfer Protocol

**HTTPS:** Hyper Text Transfer Protocol Secure

**CSS:** Cascading Style Sheet

**ID:** Identification

**QR:** Quick Response

**CERN:** Conseil Européen pour la Recherche Nucléaire

**MVC:** Model View Control

**SPA:** Simple Page Application

**XML:** Extensible Markup Language

**DTO:** Data Transfer Object

**SQL:** Structed Query Language

**GRPC:** Google Remote Procedure Call

**API:** Application Programmin Interface

**JSON:** avaScript Object Notation

**REST**: Representational State Transfer

**SOAP:** Simple Object Access Protocol

**URL:** Uniform Resource Loader

**JPA:** Java Persistence API

ÖZET

Bu tez hasta ve engelli vatandaşların sahip oldukları olumsuz durumlar sebebi ile topluma kazandırılmasını sağlamak amacı ile geliştirilmiş bir projeyi ele almaktadır.

Teknolojinin hızla ilerlediği bu zamanlarda her geçen gün teknolojinin farklı yönleri ve yöntemleri günümüz problemlerini çözmek için kullanılmaktadır. Hayatımızın hemen her yerinde bulunan teknolojik araçlar ve yöntemler yaşantımızı kolaylaştırmakta ve hayat kalitemizi arttırmaktadır. Geliştirilen ve kullanılan birçok teknolojik araçlara ve yöntemlere rağmen hala hayatımızda aktif olarak kullanabileceğimiz birçok teknoloji bulunmakta.

QR kod barkod sistemlerinin en gelişmiş hallerinden birisi olmakla beraber giderek hayatımızın her alanında kullanılmaya başlıyor. Büyük veriler barındırabilmesi, hızı ve okunması için özel bir araç gerektirmemesi sebebi ile günümüzde kullanılan en yaygın barkodlardan birisidir. Hemen her bireyin kullandığı cep telefonu, tablet gibi araçlar ile anında okutulabilir.

Engellilik ise insanlığın varlığından bu yana insanlık ile gelmiş olup teknoloji sayesinde bu durum en aza indirmeye ve hayat standartları en üst düzeye çıkarılmaya çalışılıyor. Bu noktada farklı teknolojiler ile bu durumun her seferinde farklı yönlerine çareler bulunuyor. Hasta veya engelli vatandaşlarımız olumsuz durumlarından dolayı tek başlarına kalmaktan çekinmekte ve yanlarında refakatçileri olmadan tek başlarına sosyalleşmekte sorun yaşamaktadırlar. Projenin amacı ise bu tür hastalık veya engellerin getirmiş olduğu olumsuzlukların önüne geçerek kullanıcılarımıza daha sosyal ve korkusuz bir hayat sunmaktır. Bu projedeki amaç ise hasta ve engelli bireyler için acil durumlarda veya oluşması istenmeyen durumlarda gerek diğer insanlara gerek ise ilk yardım ekiplerine bilgi sağlamak amacı ile kişisel ve hastalık veya engellilik bilgilerini kayıt ederek internete bağlanabilen ve kameraya sahip olan hemen her cihazdan ulaşabilecekleri bir sistem oluşturmak.

Geliştirilen proje web tabanlı mikroservis mimarisi ile geliştirilip, API ile iletişim kurulacaktır. Halka açık bilgiler QR kod teknolojisi sayesinde hızlı erişim sağlanabilecek şekilde şifrelenerek baskı ile bileklik veya benzeri aksesuarlara takılabilecek ve böylece internete bağlanabilen hemen her cihazdan bu sisteme erişim sağlanabilecektir.

**Anahtar Kelimeler :** QR kod, WEB, Engellilik, API, Angular.

ABSTRACT

This thesis deals with a project developed to ensure that sick and disabled citizens are integrated into the society due to their negative conditions.

In these times when technology is advancing rapidly, different aspects and methods of technology are used to solve today's problems. Technological tools and methods are available almost everywhere in our lives, making our lives easier and improving our quality of life. Despite the many technological tools and methods developed and used, there are still many technologies that we can actively use in our lives.

QR code is one of the most advanced forms of barcode systems and is increasingly being used in all areas of our lives. It is one of the most common barcodes used today because it can contain large data, its speed and the fact that it does not require a special tool to read it. It can be read easily with tools such as cell phones and tablets used by almost every person.

Disability has been with humanity since the existence of humanity, and with the help of technology, this situation is tried to be minimized and living standards are tried to be maximized. At this point, remedies are found for different aspects of this situation each time with different technologies. Our sick or disabled citizens are afraid to stay alone due to their negative conditions and have problems socializing alone without their companions. The goal of the project is to offer our users a more social and fearless life by preventing the negativities brought by such diseases or disabilities. The goal of this project is to create a system that can be accessed from almost any device that can be connected to the internet and has a camera by recording personal and disease or disability information in order to provide information to other people and first aid teams in emergency situations or unwanted situations for sick and disabled persons.

The project will be developed with a web-based microservice architecture and will be communicated via API. Public information will be encrypted in a way that can be accessed quickly thanks to QR code technology and can be printed and attached to wristbands or similar accessories, so that access to this system can be provided from almost any device that can connect to the internet.

**Key Words:** QR Code, WEB, Disability, API, Angular.

1. GİRİŞ

Engellilik olgusu fiziksel, psikolojik, sosyal, ekonomik vb. bakımlardan çok boyutlu bir olgudur. Engellilik, temelde bireysel bir durum gibi görülse de aslında toplumsal bir durumdur. Engelli bireylerin bağımsız bir şekilde toplumsal rol alabilmesini sağlamak ve yaşam kalitelerini artırmak hepimizin sorumluluğudur.

Teknolojik gelişmeleri de bu doğrultuda değerlendirmek, teknolojinin en yararlı kullanımıdır. QR kod sistemi son 10 yılda hızla dünyaya yayılmış ve günlük hayatımızda çok sık kullandığımız bir teknoloji hâline gelmiştir. Engelli bireylere özel geliştirilen QR Kod sistemi ile bedensel veya zihinsel engeli bulunan bireylerin yardıma ihtiyaçları olduğu kaybolma, beklenmedik sağlık sorunları gibi durumlarda, yapılması gerekenlere hızlı bir şekilde ulaşılıp, doğru yaklaşımın sağlanması amaçlanmaktadır.

Engelli veya özel bir hastalığı olan bireylerimizin ne zaman ne tür sorunlar yaşayacağını bilemeyebiliriz. Her an için yanlarında bir bakıcı, vasi veya yakın bir bireyi olmayabilir. Yalnızken yaşayacakları tıbbi bir sorunda gerek çevredeki vatandaşların yanlış müdahalesi gerek ilk yardım ekiplerinin hasta hakkındaki yetersiz bilgisi sonucu hastaya verilecek zararın önüne geçmeyi amaçlamaktayız.

Hasta vatandaşlarımızı, sahip oldukları hastalıkların getirdiği korku ile kendilerini soyutlamış oldukları topluma, korkularını yıkarak tekrar kazandırıp yaşadıkları psikolojik tahribatı düzeltmek için çalıştık. Hasta bireyler yalnızken hastalığın bir semptomunun nüks etmesi ve bunun sonucunda toplum içerisinde yanında yardımcı olmadan aciz bir durumda kalmaktan korktukları için sosyalleşmekten geri kalarak farkında olmadan psikolojilerini tahrip ediyorlar. Amacımız hastalarımız cesaretlendirerek geliştirdiğimiz sistem yardımı ile hastalığın yalnızken bile nüks etmesi hâlinde onları çaresiz bırakmamak ve çaresiz olmadıklarına inandırmaktır.

“Vatandaşlarımızı bu tür sorunlardan korurken bilgilerini kötü amaçlı yazılım, saldırı veya gereksiz bilgi ifşasından da kaçınmayı planlıyoruz. Diğer projelerden farklı olarak hasta bir bireyin bütün bir zamanında yaşayabileceği bir olumsuzluğu gidermeye çalışarak kendilerinin her an tam bir güven duygusu ile psikolojisinde devamlı bir iyileşmeyi amaçlıyoruz. Sadece araştırma ile değil hasta bireylerin ve ailelerinin kullanabileceği çalışır ve ücretsiz bir sistem ile kendilerine yardımcı olacağız. Projemize yön verebilecek hasta gruplarımızı ve onların yaş ortalamalarını belirledik.”1

“Günlük hayatımızın alışveriş, bankacılık, eğitim, ulaşım ve turizm gibi çeşitli alanlarında yaygın biçimde kullanılan QR kod uygulamaları, engelli bireylerin hayatını kolaylaştırmak ve yaşam kalitelerini artırmak için de geliştirilmektedir. Bu uygulamaların ülkemizde ve dünyada bazı örnekleri vardır: tekerlekli sandalye kullanıcıları için QR kodlu yer işaretlerine dayalı navigasyon sistemi”2,

“Görme engelliler için alışverişte ürünlere QR kod yerleştirme, görme engelli bireyler için nesnelerin QR kod okuma ile tanımlanması.” 3

“Biz de QR kod teknolojisinden yararlanarak engelli bireylerin engellilik durumları sebebi ile bir sorun oluştuğunda, tek başlarına karşılayamayacakları ihtiyaçları vb. durumlar gerçekleştiğinde nasıl yardımcı olunabileceği, aile/vasi iletişim bilgileri, acil çağrı servisi iletişim bilgileri gibi verilerin, her engelli vatandaşa özel şablonlar sayesinde, web sitesinde toplanıp, ihtiyaç hâlinde, vatandaşın kullandığı bileklik, bandana vb. aksesuarlara iliştirilen QR kod sayesinde hızlı şekilde ulaşılabilmesini amaçlıyoruz. Araştırmalarımıza konu hakkındaki eksiklik fark edilmiş ve konu üzerine araştırma yapılmıştır.” 4

Yapılan araştırma incelendiğinde iki proje arasında fark olarak, üzerinde çalıştığımız projenin bulut tabanlı olması ve hemen her cep telefonu ile uygulama indirmeksizin web tarayıcısı ile görüntülenebilen bir sistem olduğunu söyleyebiliriz. Böylece çok daha kolay ve çözüme yönelik bir proje geliştirmeyi amaçlamaktayız.

2. ENGELLİLİK

1.1. Engellilik Nedir?

“Engellilik kelimesi ilk olarak hareket yeteneği kısıtlanmış bireyleri çağrıştırırken temelde doğuştan veya sonradan herhangi bir nedenle bedensel, zihinsel, ruhsal, duyusal ve/ veya sosyal yeteneklerini çeşitli derecelerde kaybetmesi nedeni ile toplumsal yaşama uyum sağlama ve günlük gereksinimlerini karşılama güçlükleri olan korunma, bakım, rehabilitasyon, danışmanlık ve destek hizmetlerine ihtiyaç duyan bireylerin durumunu karşılamaktadır.” 8

“Dünya nüfusunun yaklaşık %10’u, başka bir deyişle 650 milyon kişi engellidir. Türkiye’de ise nüfusun % 12,28’sini engelli bireyler oluşturmaktadır ki bu da 8,5 milyonu engelli birey anlamına gelmektedir.” 9

“Engellilik, bireyler arasında büyük çeşitlilik gösterir. Kimi insanlar doğuştan gelen bir engel ile yaşarken, kimileri sonradan edindikleri bir engel durumuyla karşılaşabilirler. Engellilik, geçici veya sürekli olabilir ve zaman içinde değişebilir. Ayrıca, engelli bireylerin yetenekleri de büyük ölçüde değişkenlik gösterir; bazıları belirli alanlarda kısıtlı olabilirken, diğer yeteneklerde üstün olabilirler. Genel hatlarıyla ele alınırsa üç ana türde engellilik durumu söz konusudur. Vücudun bütünselliği anlamındaki bir engelin dolaylı olarak birçok farklı ve değişik şekillerde bireye engeller oluşturduğu görünmektedir.”10

Bu üç ana türdeki engellilik şöyledir;

1.1.1. Fizyolojik Engel:

“Kişi, doğuştan veya sonradan geçirdiği hastalık veya kaza nedeniyle vücut içerisinde yer alan hayati organların fonksiyon yetersizliği nedeniyle oluşan durumunda kişi fizyolojik olarak engelli olmaktadır. Mobilite bozuklukları, solunum bozuklukları, organ yetmezlikleri bunun örnekleridir.”11

1.1.2. Zihinsel- Nörolojik Kontrol Problemi:

“Doğum öncesi, doğum sırası ve doğum sonrasında çeşitli nedenlere bağlı olarak gelişimsel dönemde ortaya çıkan, uyumlu davranışlarda görülen yetersizliğe ilaveten dikkat, algılama, bellek ve muhakeme gibi genel zekâ fonksiyonları açısından normalin altında olma durumudur.” 12

1.1.3. Fiziksel (Bedensel) Engel:

“Doğum öncesi veya sonrasında çeşitli faktörlere bağlı olarak gerçekleşen ve bireyin kemik, sinir ve kas sistemindeki bozukluk durumlarından dolayı bedensel yeteneklerini kaybetme durumlarına bedensel engel adı verilmektedir.”13

“Dünya Sağlık Örgütüne göre; engellilik durumu herkesin hayatının belirli bir döneminde yaşayabileceği, insan işlevleri üzerinde kendini gösteren geçici ya da kalıcı zorluklar olarak tanımlanabilir. (Dünya Sağlık Örgütü, Dünya Engellilik Raporu)”

1.2. Engellinin Toplumdaki Yeri̇

İnsanlık tarihine baktığımızda engelli bireylerin toplumda sürekli olarak baskıya maruz kalmış, sosyal, ekonomik ve politik olarak soyutlanmış bireyler olarak görmekteyiz.

“Çağdaş dünyanın geldiği noktada; engelli vatandaşlarımızın, ulaşım, eğitim, barınma, tıbbi ve sosyal bakım ve rehabilitasyonlarının sağlanması, ailelerinin desteklenmesi, engellilerimizin üreten bireyler olarak kendi kendilerine yeterli duruma gelmeleri gerekmektedir.”14

Engelli bireylerin bağımsız bir şekilde toplumsal rol alabilmesini sağlamak ve yaşam kalitelerini artırmak hepimizin sorumluluğudur. Toplumlar, engellilikle ilgili olarak eşitlik, adalet ve insan hakları konularında çeşitli politika ve programları uygularlar. Engelli bireylerin toplumsal hayata tam ve etkili bir şekilde katılımını sağlamak, erişilebilirlik ve engelli hakları gibi konular önemli hâle gelir. Engellilik, sadece bireyin bireysel bir durumu değil, aynı zamanda toplumsal bir mesele olarak da ele alınır, bu nedenle toplumun tüm kesimlerini kapsayan çözümler ve anlayışlar gereklidir. Engellilikle ilgili farkındalığı artırmak, toplumsal destek sistemlerini güçlendirmek ve engelli bireylerin yaşamlarını daha anlamlı hâle getirmek için teknolojik gelişmelerden yararlanmak bizlere oldukça kolaylık sağlayacaktır. Örneğin QR kod sistemi, son 10 yılda dünyaya hızla yayılmış ve günlük hayatımızda sıklıkla kullandığımız teknolojik gelişmeler arasında yerini almıştır.

2. KULLANILAN TEKNOLOJILER:

2.1. QR Kod:

“QR kod ingilizcesi Quick Response olan Türkçeye karekod olarak geçmiş, asıl çevrisi hızlı yanıt olan kelimelerin kısaltılmış hâlidir.”5

“QR kod Japon kökenli bir mühendis olan Masahiro Hara Tarafından 1994 yılında geliştirilmiş olup popülerliğini Denso Wave Adlı bir otomotiv şirketi sayesinde kazanmıştır.”6

“QR kod beyaz bir zemin üzerine yapılan matris hâlinde işlenmiş işaretlerin aralıklarındaki boşlukların okunması ile çalışır.”7

QR kod hızlı okunabilirliği sayesinde popülerliği ve kullanım alanı giderek artmıştır. Günümüzde ulaşım, e ticaret ödemeleri, pos ödemeleri, yayıncılık, tekstil, otomotiv gibi daha birçok alanda yaygın bir şekilde kullanılmakta.

Karekod üretimi günümüzde hemen her teknolojinin desteklediği açık kaynaklı bir yazılım sayesinde kolayca oluşturulabilir hâle gelmiştir. Karekod okuma uygulamaları birçok cihaz tarafından desteklenmekte ve istenilen hemen her cihaza entegre edilebilmektedir. Buda kullanımını kolaylaştırdığı için karekodun kullanımını arttırmış ve günümüz dünyasında birçok bilginin bir kare matris içerisinde saklanarak paylaşılmasına olanak tanımıştır. Kullanımındaki kolaylık sebebi ve ücretsiz kullanıma sahip olması üretim ve hizmet sektörleri gibi irili ufaklı birçok alanda tercih edilmesini sağlamıştır.

Karekod içerisinde birçok farklı türde bilgiyi saklayabilir. Kendisi bir platform olmasa da genellikler bir platformun adresini veya bilginin kendisinin karekod hâlini barındırır. Buda bilginin herkes tarafından değil yalnızca karekoda fiziken erişimi olabilenlerin kullanmasını sağlar. Böylece fiziki bir çaba ile de olsa bilgi güvenliği sağlanmış olur.

2.2. HTML:

HTML genel olarak bilinenin aksine bir programlama dili değildir. HTML açılımı hiper metin işaretleme dili (Hyper Text Markup Language) olan bir metin işaretleme dilidir. Bu dil ile oluşturulan metinler Chrome, Explorer gibi tarayıcılar tarafından okunarak kullanıcıya gösterilir. Bir web sayfasının barındırdıklarını ve genel hatlarını barındırır. 1980 yılında CERN’de çalışanların birbirileri ile dosya ve veri paylaşmak amacı ile ilk taslağı geliştirilmiş, 1993 yılında oluşturulmuştur ve günümüze kadar çeşitli geliştirmeler ile gelerek hâlâ yaygın bir şekilde kullanılmaktadır.

2.2.1. HTML Ne İşe Yarar:

Sahip olduğu etiketler yardımı ile web sayfasındaki metinlerin nasıl okunacağını belirler. Böylece düz metinlerimiz kendisine verilen etiketin özelliklerini alarak istediğimiz şekilde görünmüş olur.

2.2.2. HTML Kullanım Alanları:

* Web projeleri
* Dokümantasyon

2.3. CSS:

İnternet kullanımın giderek artması ile HTML’in arayüzü yeterli görülmemiş bu nedenle arayüz konusunda daha geniş bir yelpazeye sahip olan CSS geliştirilmiştir. CSS, basamaklanmış stil katmanları anlamına gelen Cascading Style Sheets’in kısaltmasıdır. CSS W3C tarafından 17 Aralık 1996’da CSS1 sürümü ile yayınlanmıştır. Her yapılan geliştirme CSS1’in üzerine inşa edilerek zamanla 1998 yılında CSS2 ve son olarak CSS’in son sürümü 1998 yılının haziran ayında CSS3 çıkmıştır. CSS3 farklı şekilde HTML sayfalarına dahil edilebilir. Bunlar;

* Satır içine ekleme
* Dahili ekleme
* Harici ekleme

2.3.1. Dahili Ekleme:

CSS kodlarını dahili şekilde eklediğimiz zaman bu eklenen kodlar yalnızca bu sayfa için geçerli olur. Tekrar kullanılamadığı için geliştiriciye gereksiz iş yüküne sebep olabilir. Ancak özel olarak tasarımı ayrı olan sayfalar için dahili ekleme yöntemi kullanılabilir.

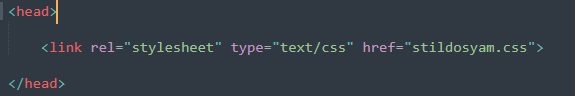
HTML etiketlerinden olan head etiketlerinin arasına style etiketi açılarak CSS kodları yazılır ve yine style kapatma etiketi ile kapatılarak tanımlanmış olur.



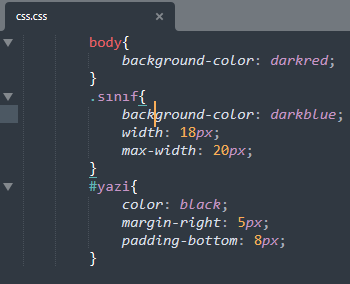
**Şekil 2.1.** Dahili CSS Ekleme Örneği

2.3.2. Harici Ekleme:

Bu yöntemde CSS kodları uzantısı “.css” olan bir dosyaya yazılarak dosya kaydedilir. Daha sonra oluşturulan HTML sayfasına eklenmek istenen harici CSS dosyası head etiketleri arasında link etiketi ile rel özelliği stylesheet olarak tanımlanarak href özelliği ile dosya yolu verilerek dosyaya dahil edilmiş olur. Böylece yazılanlar tekrar kullanılabilir ve bütün bir sayfanın yapısına entegre edilmiş olur.



**Şekil 2.2.** Harici CSS Ekleme Örneği



**Şekil 2.3.** CSS Dosya Örneği

2.3.3. Satır İçi Stil Ekleme:

Satır içi stil eklemede stil yalnızca bir öğe için geçerli olur. Bu sebep ile her öge için istenilen stilin tekrar yazılması gerekir. Satır içi stil ekleme, stilin ekleneceği HTML öğesinin style özelliğinin içerisine CSS kodlarının yazılması ile yapılır.



**Şekil 2.4.** Satır içi CSS ekleme örneği

2.3.4. CSS Ne İşe Yarar:

CSS kodları HTML öğelerinin yazılan kodlara göre görünüşlerinin değiştirilmesini sağlar. Böylece kullanışlı ve göze hitap eden sayfalar oluşturulur.

2.3.5. CSS Kullanım Alanları:

* Web sayfaları

2.4. JavaScript:

JavaScript Brendan eich tarafından 4 Aralık 1995 yılında çıkarılmıştır. Yazılım geliştiriciler tarafından uygulama, web sayfaları ve sunucu geliştirirken kullanılan bir programlama dilidir. Hafif ve dinamik olması JavaScript'in önemli özelliklerindendir. Günümüzde popüler olan birçok tarayıcı tarafından desteklenmektedir. Bir çok framework ve JavaScript kütüphanesine sahip olan bu dil tek başına programlama yapmaktan ziyade küçük scriptler yazılarak web sayfalarının etkileşimini sağlar.

2.4.1. Java Script Ne İşe Yarar:

JavaScript dinamik ve etkileşimli web sayfaları oluşturulmasını sağlar. HTML ve CSS ile etkileşime girerek canlı hâlde türlü değişiklikler yapılmasına olanak sağlar.

2.4.2. Java Script Kullanım Alanları:

* Web sayfaları
* Sunucu uygulamaları
* Web uygulamaları
* Mobil uygulamalar
* Oyun geliştirme

2.5. Java:

Java, James Goslin tarafından tasarlanan ve Oracle tarafından geliştirilen bir programlama dilidir. İlk defa 23 Mayıs 1995 de yayımlanmış olup hâlâ günümüzde popülerliğini korumaktadır. Açık kaynak kodlu nesneye yönelik ve bağımsız platform bir dildir. Yüksek seviye bir dil olup anlaşılması ve okunması kolaydır.

2.5.1. Java Ne işe yarar:

Büyük çaplı projelerden küçük çaplı projelere her projede verimli olan bu dil, işlemci tabanlı birçok uygulamanın geliştirilmesinde görev alır. Nesne tabanlı olan bu dil gerçek dünya problemlerinin bilgisayar ortamında çözülmesini sağlar.

2.5.2. Java Kullanım Alanları:

* Oyun geliştirme
* Bilgi bulut işlem
* Büyük veri
* Yapay zekâ
* Nesnelerin interneti
* Web uygulamaları
* Mobil programlama
* Veri analitiği
* Gömülü sistemler
* Bilgisayar uygulamaları
* Eğitim

2.5.3. Java ve JavaScript Farkları:

* Java Daha büyük ve karmaşık projelerde kullanılırken JavaScript daha küçük scriptler yazmada kullanılır.
* Java nesne tabanlı bir programlama diliyken JavaScript daha çok betik bir dildir.
* JavaScript web sayfalarını yalnızca dinamik ve etkileşimli hâle getirmede kullanılırken Java aynı zamanda sunucu tarafı uygulamaları ve bağımsız programlarda kullanılır.
* Java kodların yeniden kullanımını sağlar, sınıflar ve nesneler ile daha derli bir düzen sunarken JavaScript'in buna desteği yoktur.
* Java kalıtım, veri kapsülleme ve polimorfizm gibi birçok özelliği destekleyebilir.
* Java birçok platforma çalışabilirken JavaScript yalnızca tarayıcılarda çalışır.
* Java derlenirken JavaScript yorumlanır.

2.6. TypeScript:

TypeScript Microsoft tarafından hazırlanan ve ilk defa 1 Ekim 2012’ de yayımlanan bir programlama dilidir. Bünyesindeki derleyici ile yazılan TypeScript kodlarını JavaScript'e çevirir ve çıktı olarak Alınan JavaScript kodları kullanılır. TypeScript’i ister sunucu istersek de istemci tarafında kullanabiliriz. JavaScript'in sahip olduğu bütün özellikleri barındıran TypeScript aynı zamanda yeni eklenen özellikler ile daha kullanışlı bir programlama dili olmayı başarmıştır. Büyük ve daha karmaşık projelerde JavaScript’e göre daha kullanışlı, okunabilir, geliştirmeye müsait ve verimli bir sonuç çıkartır. TypeScript'i JavaScript'e derlemek için ya varsayılan TypeScript Checker kullanılabilir ya da Babel derleyicisi çağrılabilir. 22

2.6.1. TypeScript Ne İşe Yarar:

JavaScript'in yaptığı hemen hemen her işi yapan bu teknoloji web sayfalarının dinamik bir hâle getirilmesi için kullanılabilir.

2.6.2. TypeScript Kullanım Alanları:

* Web sunucusu
* Web sayfaları
* Sunucu uygulamaları
* Web uygulamaları
* Mobil uygulamalar
* Oyun geliştirme

2.7. Angular:

Angular Google tarafından geliştirilerek ilk defa 14 Eylül 2016 tarihinde yayımlanmıştır. Hâlâ geliştirilme süreci devam eden bu teknoloji sürekli güncelleme almaktadır. Çapraz platform olan Angular bu özelliği sayesinde birçok platformda çalışabilmektedir. Angular SPA yapısını desteklemektedir. İngilizcesi “Simple Project Application” olan SPA’nın Türkçe karşılığı “Tekil Projedir”. Yapısında MVC yani Model View Controller mimarisini barındırır. Bu sayede daha kullanışlı, okuması ve yazması daha kolay ve geliştirmeye açık bir yapı sunmuş olur.

2.7.1 Angular Ne İşe Yarar:

Yeni bir sayfa yüklemek yerine yeni sayfanın ögelerini dinamik olarak yazarak hazır hâle getirilmesini sağlar. TypeScript ile yazılan bu teknoloji, web sayfaları geliştirmede kullanılır.

2.7.2. Angular Kullanım Alanları:

* Web sayfaları
* Web uygulamaları

2.8. Spring Boot :

Spring Boot Java programlama diline ait Spring framework ile geliştirilmiştir. Kendisi bir framework değildir yalnızca Spring framework’ünün XML yapıları ile uğraşmamamızı sağlayan bir yapıdır. Java programlama dilinde mikro servis yazılmasını sağlar. Spring framework ile geliştirilen uygulamalar birçok özel ayara ihtiyaç duyar. Bunun sebebi Spring framework’ün çok geniş bir geliştirme yelpazesinin olmasıdır. İçerisinde birçok kütüphane barındıran bu framework bazen her uygulama için standart ayarlamaları gerektirirken çok karmaşık ve yorucu olabilmektedir. Spring Boot ise sizin için bu ön ayarlamaları varsayılan olarak yaparak projenizi ayağa kaldırır. Ancak sizin yapacağınız değişiklikler var ise müdahale ederek yeri projenize göre ayarlayabilirsiniz. Böylece geliştiriciyi birçok iş yükünden kurtarırken kullanışlı projeler ortaya koymamıza olanak tanır. İçerisinde barındırdığı anotasyonlar ile geliştiricileri büyük bir yükten kurtarır. Anotasyonlar uygulamanın davranışını belirler. Spring Boot geliştiricinin en stabil programı hızlı ve uygun yöntemler ile geliştirmesini sağlamaktır. Bu bağlamda sahip olduğu anotasyonlar bir projede birçok yerde kullanmamız gerekecek bazı kod öbeklerinin tek bir çağırma işlemi ile uygulama davranışını en uygun şekilde belirler. MVC mimarisini kullanan Spring Boot bu sayede kullanışlı ve geliştirilmeye açık projeler ortaya çıkmasını sağlamaktadır. Spring Boot Spring framework’ün sahip olduğu Dependecy injection özelliğini barındırır bu sayede bir nesnenin ihtiyacı olan ve bir deyime bağlı olan nesneleri tekrar oluşturmak yerine dışarıdan enjekte etmektedir. Böylece bizi yine zamandan tasarruf etmeye, oluşabilecek hataların önüne geçmeye ve kod tekrarından yani iş yükünden kurtaracaktır.

2.8.1. Spring Boot Ne İşe Yarar:

Spring Boot ile mikroservisler yazılabilir, APIler oluşturulabilir. Yani web tabanlı uygulamaların backend tarafını geliştirebiliriz. Geliştirilen mikro servis ile kötü amaçlı yazılımlara karşı güvenlik önlemleri alarak sistemimizi daha güvenli hâle getirebiliriz. Front end tarafından yapılan istekleri yakalayarak bunlara gerekli şekillerde cevap verebiliriz. Veri tabanı ile bağlantı kurarak verileri çekebilir ve Java nesnesine dönüştürerek hata payını en aza indirip modeller yardımı ile kullanabiliriz. Sahip olduğu DTO (Data Transfer Object) yapısı ile veri fazlalığından kurtularak sistemdeki yükü hafifletebiliriz. Geliştirilen mikro servisler sayesinde sistemimiz daha hızlı ve güvenli olacaktır. Yani kısaca Spring Boot ile geliştirilen uygulamalar sayesinde veri tabanı ile bağlantı kurabilir veri tabanından elde ettiğimiz verileri modeller yardımı ile Java nesnesine çevirerek view katmanında gösterilmeye hazır hâle getirebiliriz.

2.8.2. Spring Boot Nerelerde Kullanılır:

* Yapay zekâ
* Bulut sistemler
* Veri depolama
* Veri analizi
* Nesnelerin interneti
* Güvenlik ve idare
* API GATEWAY oluşturma
* Mikro hizmetler
* Web tabanlı projeler

2.9. SQL:

SQL, İngilizcesi “Structed Query Language” olan yapılandırılmış sorgu dili’nin kısaltılmış hâlidir. SQL verilerin farklı boyutlarda, farklı yapılarda saklayabilen ve yapılan sorgulara cevap verebilen bir veri tabanı uygulamasıdır. SQL sayesinde bu verilerin yönetilmesi, silinmesi, aktif edilmesi, güncellenmesi sağlanır. Çıktığı ilk günden bu yana popülerliğini koruyan hâlâ yapısı ile birçok projede yeterlilik gösteren SQL günümüzde en popüler veri tabanlarından ve sorgu dillerinden bir tanesidir. SQL genel bilinenin aksine bir programlama dili olmamasına karşın veri tabanında kullanılan bir alt dildir. Bazı SQL dilinin kullanıldığı veri tabanları aşağıda listelenmiştir:

* MySQL
* MsSQL
* Microsoft SQL Server
* Oracle
* IBM Database 2 (IBM DB2)
* PostgreSQL
* Sybase
* IBM Informix
* Progress
* Firebird
* Access

2.9.1. SQL Ne İşe Yarar:

SQL sayesinde verilerin yönetilmesi, silinmesi, aktif edilmesi ve güncellenmesi sağlanır. Barındırdığı sorgu yöntemleri ile veri tabanında tablo oluşturma, tablo bağlama, sütün satır seçme ve birleştirme gibi birçok veri tabanı yönetim yöntemlerini kullanmamızı sağlamaktadır. SQL dili sayesinde birden fazla tablolardaki satır ve sütunları hücresinde barındırdığı veriye göre filtreleyebilir ve bu filtre akabinde seçime, silme, güncelleme veya kaydetme işlemi yapabilmekteyiz.

2.9.2. SQL Kullanım Alanları:

* Web tabanlı uygulamalar
* Veri Depolama
* Veri analizi
* Windows form uygulamaları
* Oyun geliştirme
* Yapay zekâ
* Ve veri yönetmeninin olduğu her alanda SQL kullanılabilir.

2.10. Primeng:

Türk mühendisler tarafından kurulan ve geliştirilen prime teknolojilerinin 2009 yılında ortaya çıkardığı PrimeFaces kütüphanesinin Angular için oluşturulmuş hâlidir. Primeng Front end tarafında kullanılan HTML, CSS ve JavaScript teknolojilerinin kalıplaşmış kodlarının tekrar tekrar yazılmasından bizi kurtararak zamandan ve iş yükünden tasarruf etmemizi sağlar. Anlaşılacağı üzere bir Angular kütüphanesidir.

2.10.1. Primeng Ne İşe Yarar:

Primeng etiketlerini kullandığımız zaman HTML için öncesinde yazılmış özellikleri dahil bir şekilde kullanabiliriz. Aynısı CSS ve JavaScript içinde geçerlidir. Böylece her projede yazdığımız artık standartlaşmış bazı kalıpların iş yükünden bizi kurtarırken daha düzenli, kullanışlı ve göze hitap eden tasarımlar ortaya çıkarmamızda yardımcı olmaktadır.

2.10.2. Primeng Kullanım Alanları:

* Web tabanlı projeler
* Angular
* VUE
* React.js

2.11. Bootstrap:

Bootstrap, HTML, CSS ve JavaScript kullanılarak oluşturulmuş bir Framework’ tür. Geliştiricilerin ve tasarımcıların Zamandan ve iş yükünden tasarruf ederek etkileşimli web siteleri yazmalarına olanak tanımaktadır. Bootstrap framework’ ü Twitter'ın eski çalışanlarından Mark Otto ve Jacob Thornton tarafından geliştirilmiştir. İlk defa 19 Ağustos 2011 tarihinde yayımlanan Bootstrap web programlama alanında kullanılan en yaygın frameworklerden biridir.

2.11.1. Bootstrap Ne İşe Yarar:

Bootstrap, CSS ve JavaScript teknolojilerinden yararlanarak oluşturulmuş özellikleri tek bir çatı altında bize sunmaktadır. Böylece elimizde tekrar kullanılabilir bir kod paketi olur. Bize zamandan ve iş yükünden tasarruf etmemize olanak sağlamaktadır. Boostrap geliştirme aracında tanımlanan özellikler HTML öğelerinin “class” özelliğine tanımlanarak framework’ten çağırılabilir ve tanımlı özellikler kullanılabilir. Eğer Kullanılan paketin içeriğinde olmasını istemediğimiz bir özellik varsa override edilerek düzeltilebilir. Baskın gelmesi istenilen özellik HTML ögesinin style özelliğinin içerisine CSS kodları ile override edilebilir.

2.12. Monolitik Mimari:

Monolitik mimari, bir uygulamanın tüm parçalarının tek bir çatı altında sunulması olarak tanımlanabilir. Geliştirilecek olan projenin az kaynak tüketmesi, zamandan ve insan gücünden tasarrufu için küçük çaplı projelerde Monolitik Mimariyi kullanmak faydalı olacaktır. Böylece geliştirme kısmı hızlıca tamamlanarak proje kaynaklarından tasarruf edilerek ürün en hızlı şekilde ortaya çıkarılmış olacaktır. Ancak geliştirme ve kullanım safhasında geniş kitlelerle ve yoğun isteklerle karşı karlıya kalındığında Monolitik Mimari yetersiz kalabilir. Bu nedenle küçük çaplı ve az karmaşık projelerde tercih edilmektedir. Küçük çaplı ekiplerin yönetilmesi ve uygulama geliştirmesi kolaydır. Birden fazla ekibe ihtiyaç duymaz ve birden fazla modülle çalışmamıza olanak sağlamaktadır.

Yazılım tek bir kod tabanında olduğundan dolayı güvenlik endişesi düşüktür. Ağ gecikmesi çok yapılı mimarilere göre daha azdır ve yapı tek çatı altında olduğundan dolayı hataları bulmak ve veri akışını izlemek daha kolaydır.

2.12.1. Monolitik Mimari Ne İşe Yarar:

Monolitik Mimari kaynaklardan tasarruf ile elimizde olan az miktardaki kaynakları kullanarak projeler geliştirmeye olanak sağlar. Bir projenin gereksinim duyduğu her şeyi bir çatı altında toplayarak amacını gerçekleştirmeye çalışır.

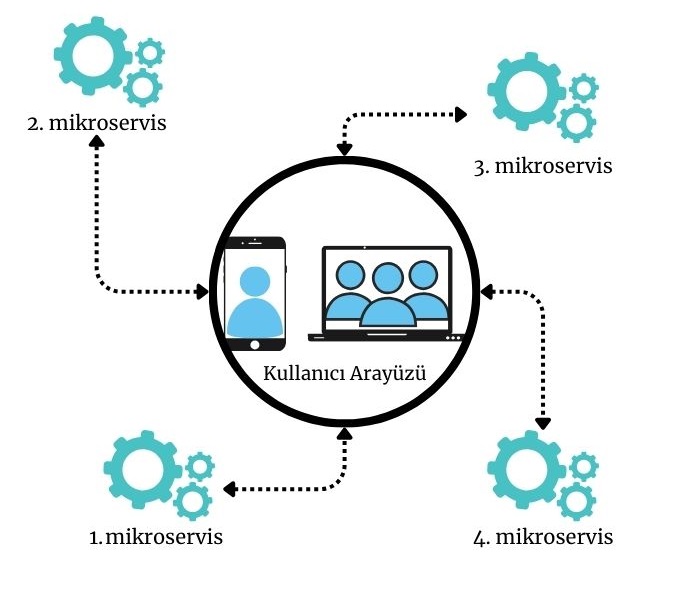
2.12.2. Monolitik Mimarinin Kullanım Alanları:

* Web tabanlı projeler
* Küçük çaplı projeler
* Az karmaşık projeler

2.13. Mikroservis:

Mikroservis Mimarisi monolitik yapının yeterli gelmediği yerlerde geliştiricileri büyük çaplı projeleri küçük servislere bölme ihtiyacıyla gelişmiştir. Öncesinde kullanılan monolitik mimari barındırdığı dezavantajlardan dolayı yetersiz gelmeye başlamış ve geliştiricileri yeni arayışlara yöneltmiştir. Bu sayede büyük bir ekiple büyük ve karmaşık bir projeyi tek çatı altında geliştirmektense bu projeyi küçük mikroservislere bölerek ve yönetilmesi zor olan büyük ekipleri de parçalayarak her ekibe faklı bir görevler verilmiştir. Böylece mikroservis parçaları yönetilmesi kolay ekipler tarafından geliştirilerek belirlenen bazı standartlar doğrultusunda senkron veya asenkron olarak çalışan bir hizmet çıkarmıştır.

Birden fazla teknolojiyi bir arada kullanmamıza olanak sağlayan Mikroservis Mimarisi bu özelliği ile geliştirilen projelerin daha yenilikçi, değiştirilebilir ve geliştirilebilir olmasını sağlamıştır.

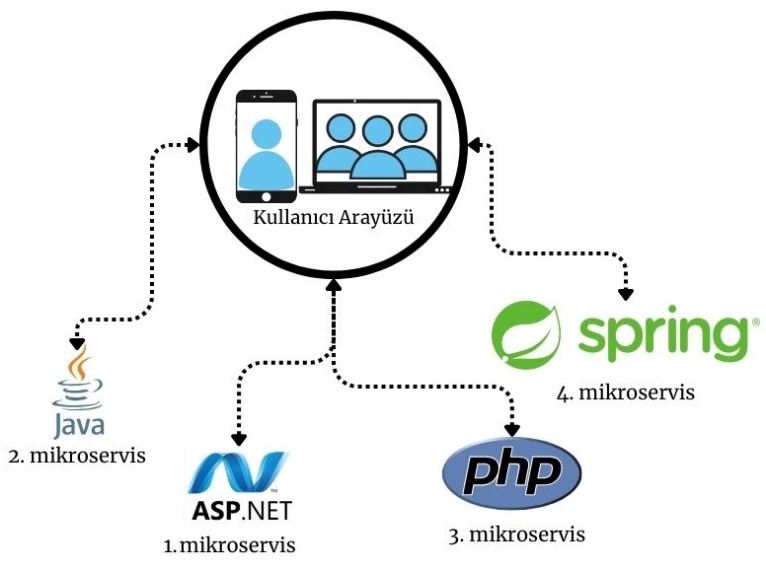


**Şekil 2.5.** Mikroservis mimarisi

Farklı yapıların bir araya gelmesiyle oluşan bu sistem, ortaya çıkan bir arızanın bütün sistemi etkilemesinin önüne geçmiştir. Böylece arıza sadece bağlı olduğu servisleri etkileyerek büyük çaplı hatalardan geliştiricileri kurtarmıştır. Geliştirilmesi ve değiştirilmesi kolay olan bu sistemde arızalar küçük çaplı olduğu için arıza çözümleri de kolay ve hızlıdır. Yine farklı yapıların bir araya gelmesi ile oluşturulduğu için sisteme yeni bir sistem entegre etmek veya entegre bir sistemi çıkartmak monolitik yapıya göre oldukça kolaydır. Sistemin tamamı ile değil sadece düzenleme yapılacak olan hizmeti ile ilgilenilir. Böylece kolayca ekleme kaldırma ve değişiklik yapılabilir. Düzenleme yapılacak hizmet sistemden çıkarılsa dahi sistemin geriye kalan kısmı çalışmaya ve hizmet vermeye devam eder.

2.13.1. Mikroservis Mimarisi Ne İşe Yarar:

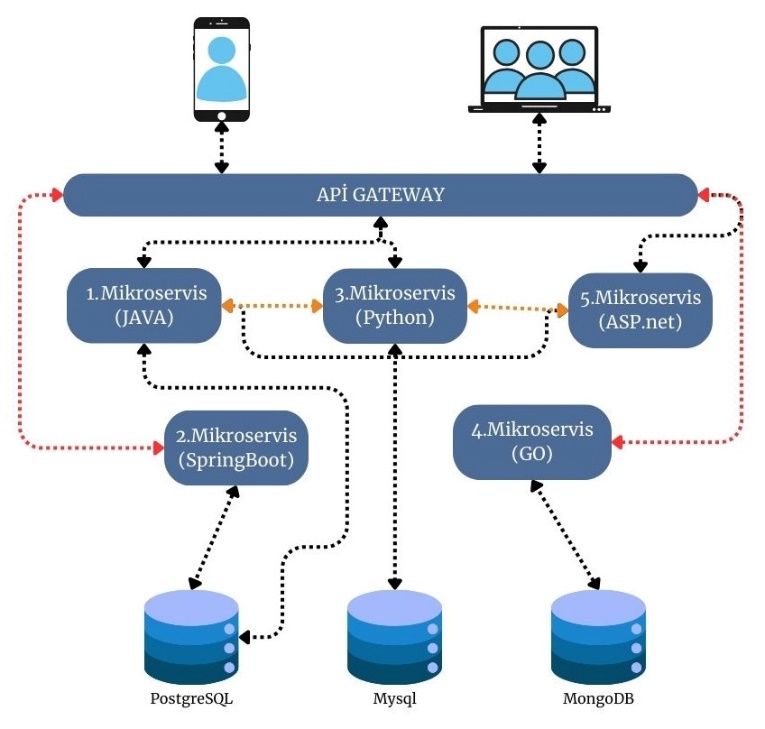
Mikroservis Mimarisi yapılacak projeyi birden fazla mikro hizmetlere bölerek kullanıcının client kısmında yaptığı her isteğe bir başka mikroservis’in cevap vermesini sağlar. Yine kullanıcının her isteği uygulama tarafından karşılanır, Sadece her istek farklı bir hizmet tarafından farklı bir veri tabanından çekilerek karşılanabilir. Böylece büyük projemiz küçük mikro hizmetlere bölünmüş olur. Bunu sağlarken her mikro hizmeti o işi en iyi şekilde yapacak teknolojiyi kullanarak yazabiliriz.



**Şekil 2.6.** Mikroservis mimarisinin farklı teknolojileri desteklemesi

Mikroservis Mimarisinde servisler arası haberleşmeyi ve servislerin client ile haberleşmesini yani talep göndermek ve gönderilen talepleri karşılamak için http veya GRPC gibi bazı protokoller kullanılır. Kullanıcı uygulamada bulunan geliştiricilerin öncesinde belirlediği bir şekilde etkileşime girerek sunucu client tarafından http ile bir talep gönderir. Sunucu bu talebi yine http protokolünü kullanarak alır ve gerekli olan işlemleri yaparak bilgiyi hazır hâle getirir. Hazır hâle gelen bilgi yine http protokolü ile sunucu tarafından client tarafına gönderilir. Client kendisine gelen bilgiyi alarak gösterilmesini istediğimiz şekilde kullanıcıya sunar.

Mikro hizmetlerin client ile haberleşmesini sağlarken en sık kullanılan yöntem bir API geliştirmektir. Yazdığımız mikroservis’e bir API GATEWAY geliştirerek mikroservis’in hangi istekleri nasıl karşılayacağını belirleyebiliriz. Böylece bu API’ye uygun gelen her istek hangi client’dan gelirse gelsin buna uygun bir şekilde karşılayabilir ve verileri yine aynı protokol ile sunabiliriz. Burada yaptığımız şey aslında bir standart belirlemektir. Mikro hizmetimize giriş ve çıkış standardı oluşturuyoruz. Bu standardın ne olduğunu bildikten sonra uygulamamız, her platformda uygulama geliştirilebilir hâle gelmiş oluyor.



**Şekil 2.7.** Mikroservis mimarisi çalışma anlatımı

2.14. API:

API, İngilizcesi “Application Programming Interface” olan uygulama programlama arayüzü kelimelerinin kısaltılmış hâlidir. API terimi, ilk defa 1974 yılında Christopher J. Date tarafından yayımlanan "The Relational and Network Approaches: Comparison of the Application Programming Interface" adlı makalede geçmiştir.15 Yazılım uygulamalarının birbirileri ile iletişim kurmasını sağlayan bir arayüzdür. API’ler bir programın bazı işlevselliklerini diğer programlar tarafından erişilmesi ve kullanılması sağlar. Programlar arası iletişimi kurarak aralarında bir veri yolu oluşturur. Böylece farklı teknolojilerde de olsalar birçok yazılım programlarının birileriyle entegre edilmesini ve veri alışverişini sağlayarak birlikte bir iş birliği içerisinde olmasına imkân tanır.

API’ler birçok farklı türde ve teknolojide geliştirilebilir. Örneğin web API’ler web tabanlı uygulamalar arası veri iletişimini kolaylaştırır. Web sitelerinin veri tabanlarına ulaşmasını sağlar. Farklı şekillerde üretilmişlerde olsa farklı teknolojilerdeki web sitelerini “Web API Keyler” sayesinde birbirilerine kolayca entegre olabilirler. Böylece Bir web sitesi bir sunucudan hava durumu bilgilerini alırken başka bir sunucudan altın, döviz, borsa gibi bilgileri alarak başka bir sunucu ile ödeme işlemleri yapabilir.

API’ler HMTL, XML, JSON gibi protokolleri kullanırlar, bu sayede farklı sistemlerin birilerine entegrasyonunda veri alışverişi için bilginin geldiği formatı bilmek yeterlidir. Böylece birbirlerinin hangi teknolojilerde nasıl geliştirildiklerine kadar detaylı bilgilere ihtiyaç duymadan sadece API keyler’i kullanarak kolayca entegre edebilir ve veri akışını sağlayabiliriz. Verdiğimiz örnekler web API üzerinde olsa da API’ler işletim sistemleri, veri tabanları, gömülü sistemler gibi birçok alanda kullanılabilirler.

“API’ler sistemlerin birbirilerine entegre olmalarında kolaylık sağlarlar. Böylece iki farklı sistemde herhangi bir değişikliğe gerek duymadan sistemler birbirilerine entegre olabilirler. Bu da geliştiricilere kaynakları daha verimli kullanarak zamandan, ücretten, iş yükünden tasarruf etmeyi sağlar.”16

API’lerin amacı iletişim sağlamak olduğu için iletişim kurmak isteyen sistemlere karşı güvenlik önlemleri almak gerekmektedir. Zamanla API için birçok güvenlik yöntemi geliştirilmiştir. Bunlardan ilki yetkilendirme kimlik doğrulamadır. Veri güvenliğini sağlamak için öncelikle sistemimize giren kişinin kimliğini öğrenmeli ve giriş iznin olup olmadığını kontrol etmeliyiz. Eğer giriş izni varsa sırada verilerimizden hangilerine erişebiliyor bunun kontrolünün sağlanası gerekir. Bu noktada yetkilendirme yapmak ve yetkisi dahilindeki verileri sunmalıyız.

Veri alışverişi sırasında verilerimizin güvenliği için veri şifreleme yöntemleri kullanılır. Taleplerimizde kullandığımız http metodunu, hassas bilgileri saklamak için https yani http Secure protokolünü kullanabiliriz. Böylece verilerimizi iletişim esnasında güvence altına almış oluruz.

İstek sınırlama; sistemimize çok sayıda istek göndererek kaynaklarımızı doldurmayı amaçlayan kötü niyetli saldırıların hizmet reddiyle önlenmesi kaynaklarımızın doğru kullanımını veri güvenliğini ve kullanıcı memnuniyetini arttıracaktır.

Bunlar gibi günümüze kadar karşılaşılan her soruna üretilmiş birçok hata önleme ve güvenlik önlemleri bulunmaktadır.

2.14.1. Mimari Yapılarına Göre API’ler

APIler mimari yapılarına göre sınıflandırılabilir. Bunlar;

2.14.1.1.REST API:

REST API, İngilizcesi “Representational State Transfer” olan Temsili durum aktarımı kelimelerinin kısaltılmış hâlidir. Http protokolünü kullanan bir API servisidir. Http protokolünün delete, put, post, get işlevlerini kullanarak çalışmaktadır. Veri alışverişinde XML veya diğer formatlar yerine JavaScript Object Notation (JSON) veri formatını kullanmaktadır.

REST API, yeni nesil web tabanlı projelerin geliştirilmesi için sıklıkla kullanılan bir mimari hâline gelmiştir. İstenilen hemen her ihtiyacı karşılaması, bakım kolaylığı, diğer sistemlere entegre edilmesinin kolay bir mimari olması ve hızlı, hafif bir yapıya sahip olması günümüzde web tabanlı uygulamalarda en çok tercih edilen mimarilerden biri olmasını sağlıyor.

2.14.1.2. SOAP:

SOAP, İngilizcesi Simple Object Access Protocol olan basit nesne erişim protokolü anlamına gelen kelimelerin baş harfleridir. API’ler içerisinde sıkı bir güvenlik yapısına sahip olması ile bilinir. SOAP veri transferinde XML formatını kullanarak iletişim kurar. REST API’ye göre yapılandırması zor olsa da sağladığı güçlü güvenlik yapısı kendisin büyük ve güvenliğin ilk planda bulunduğu projelerde sıklıkla başvurulan bir mimari hâline getirmiştir.

2.15. PostgreSQL:

“İsmi konusunda, başlarda Kaliforniya Üniversitesi'nde geliştirilen Ingres veri tabanının devamı niteliğinde olduğu için bu isme atıfta bulunularak POSTGRES adını aldı.”17

“1996 yılında SQL desteğini vurgulamak maksatlı PostgreSQL şeklinde değiştirildi. Son olarak 2007 yılında isim değişikliği konusu topluluğun tartışmasına açıldı ve geliştirme ekibi PostgreSQL isminin devam etmesine, Postgres isminin de takma isim olarak kalmasına karar verdi.”18

PostgreQL hem ilişkisel hem de ilişkisel olmayan veri yapısını destekleyen bir sistemdir. Açık kaynaklı kullanımı ücretsiz olan bu veri tabanı kurumsal sınıfı desteklediği gibi gelişmiş bir nesne ilişkisel bir veri tabanıdır. SQL desteği sunarak karmaşık verileri ve akabinde büyük iş yüklerini yüksek performans sergileyerek işlemektedir.

Windows, macOS, linux, UNIX gibi günümüzde popülerliğini koruyan işletim sistemleriyle uyumludur. JavaScript, Python, Java, C#, C, C++ gibi güncel dillere destek vermektedir. Primivites, Structured, geometry, document gibi birçok veri tipini bünyesinde saklayabilir ve işleyebilir. Yabancı anahtarlar, birincil anahtarlar, dışlama kısıtı, tavsiye kilitleri, açık kilitler, NOT NULL ve UNIQUE gibi özelikleri bünyesinde barındırır ve bu sayede veri bütünlüğünü korur. LDAP, SSPI, GSSAPI gibi birçok kimlik doğrulama protokollerini kullanır. Çok faktörlü kimlik doğrulama gibi birçok güvenlik önlemi alarak barındırdığı veriler için güvenli bir ortam sağlar.

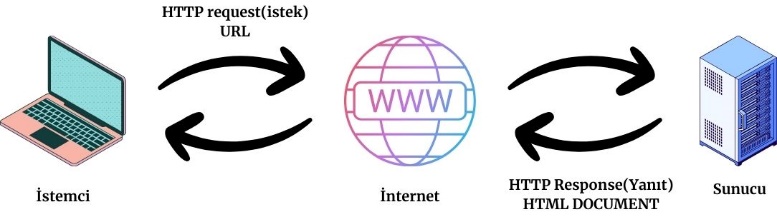
2.16. Http Protokolü:

“Http, İngilizcesi “Hyper Text Transfer Protocol” olan hiper-metin aktarım protokolü kelimelerinin baş harflerinden oluşmakta. Http bir kaynaktan dağıtılan ve ortak kullanıma açık olan hiperortam bilgi sistemleri için uygulama seviyesinde bir iletişim protokolüdür”.19

“İnternetteki web sayfalarına erişim sağlamak kullanılan en yaygın protokoldür. 1989 yılında CERN de Tim Benrners Lee adlı kişi tarafından geliştirilmeye başlandı”. 23

Http’yi geliştirmeye başlatan kişi aynı zamanda WWW kısaltmasıyla da bildiğimiz World Wibe Web ve HTML’inde geliştiricisidir.

İnternete bağlı cihazlar arasındaki iletişim ve bilgi aktarımı için geliştirilen http World Wibe Web’in bir uygulama katmanı protokolüdür. Bir alt katman olan ulaşım katmanında TCP üzerine kuruludur. Bundan dolayı http protokolünün kullanıldığı haberleşme anında meydana gelen veri kaybı veya ulaşılamayan verinin tekrar yollanması gibi işler ile TCP ilgilenir.



**Şekil 2.8.** HTTP Protokolü

İstemci tarafından herhangi bir web adresine bağlanılmak istenildiğinde bu bağlantıyı sağlamak için gönderdiğimiz adresin başına http veya https eklenmelidir. Böylece gönderdiğimiz adres http protokolü tarafından gönderilmiş olur. Http bu adresi kendi protokolüne uygun bir hâle getirerek adresin bağlı olduğu sunucuya istek yapar. Sunucu da gerekli cevabı aynı şekilde verir. Http bütün bilgileri açık bir şekilde gönderir böylece istemci ile adresin bağlı olduğu internet sunucusu arasında bir bilgi akışı oluşturulur ve bilgiler açık metin olarak birbirine aktarılır. Bu durum ise güvenlik açıklarına sebep olabilmektedir. Bunun için https gibi daha güvenlikli protokoller geliştirilmiştir.

Http isteklerinde internet sunucularının cevap verebilmek için ihtiyacı olan bilgileri barındırma şeklidir. İnternet üzerinden yapılan her http isteği her bilgi türü için düzenlenmiş bir dizi kodlanmış veri taşır. Http isteklerinin genel yapısında aşağıdakiler bulunur:

* Protokol versiyonu
* URL
* Http body
* Http request, headers
* Http method 20

Http metotları http isteğinin sorgulanan internet sunucusundan beklediği geri dönüt eylemini göstermektedir. Başlıca http metotları aşağıda açıklamaları ile belirtilirmiştir.

2.16.1. POST:

POST metodu ile adresin bağlandığı sunucuya veri yazdırılabilir. Post metodu istek parametrelerini URL ve gövde içerisinde gönderebilir. Bilgileri URL içerisinde göndermek güvenlik açığına sebep olacağından dolayı sadece mesaj gövdesi içerisinde göndermek güvenliği arttıracaktır. Tarayıcıda bir önceki adrese geri dönmek isteğimizde POST isteğinin gövde içerisinde bulunan parametreleri tekrar göndermek isteyip istemediğimizi sormaktadır. Bunun nedeni ise bir işlevi birden fazla yapmayı engellemektir. Bu nedenle güvenlikli bir iletişim sağlanmak istendiğinde POST metoduna başvurulması gerekmektedir.

2.16.2. PUT:

PUT metodu ile internet servisi üzerindeki bir sağlayıcının sahip olduğu bir kaynağı güncelleyebilir. PUT metodu kullanılırken güncellemek istenilen kaynağın eşsiz kimlik bilgisi gönderilmelidir.

2.16.2.HEAD:

Metotlardan GET metodu ile benzer özellikler gösteren HEAD metodunun farkı geri gönderilen yanıtın içerisinde mesaj gövdesinin yer almamasıdır. HEAD metodu genellikle bir kaynağın varlığının kontrolünde kullanılmaktadır.

2.16.3.DELETE:

DELETE metodu ile internet sunucusundaki istenilen veriler silinebilmektedir.

2.16.4. CONNECT:

Bir Proxy sunucusu aracılığı ile bir başka sunucuya bağlanabilmek için kullanılmaktadır.

2.16.5. OPTİONS:

OPTİONS metodu ile belirli bir kaynak için, sunucudan, http metotlarınızdan hangilerinin kullanılabileceğini sorgulama işlevini yapmaktadır.

2.16.6. TRACE:

TRACE metodu teşhis amaçlı kullanılır. İstemci ve sunucu arasında kullanılan başka bir sunucu varsa bu sunucuyu ve yapılan değişiklikleri teşhis etmek mümkündür.

Yapılan isteklerde geri dönen http yanıtı yapılan isteğin türüne göre bilgiler barındırır. Genel olarak geri dönüt yanıtları aşağıdakileri içerir:

* Http status code
* Http response headers
* Optionel http body

Http durum kodları geri dönen http yanıtının durumunu belirtir. Aşağıda bazı durum kod kümeleri açıklamaları ile verilmiştir:

**1xx Durum Kodu Kümesi**:

İstemci tarafından gönderilen isteğin sunucuya ulaştığını ve işlemlerin başlandığını gösteren durum kodu kümesidir.

**2xx Durum Kodu Kümesi**:

İstemci tarafından gönderilen isteğin sunucuya ulaştığını ve başarılı olduğunu belirten durum kodu kümesidir.

**3xx Durum Kodu Kümesi**:

Ulaşılmak istenen kaynağın başka bir kaynağa taşındığını ve erişim için bir yönlendirme olduğunu belirten durum kodu kümesidir.

**4xx Durum Kodu Kümesi**:

İstemci tarafından yapılan isteğin yerine getirilemediğini ilgili adrese ulaşılamadığını belirten durum kodu kümesidir.

**5xx Durum Kodu Kümesi**:

İstemci tarafından gönderilen isteğin başarılı bir şekilde adresin bağlı olduğu sunucuya ulaştığını ancak sunucu kaynaklı problemlerden dolayı isteğin amacına ulaşamadığını belirten durum kodu kümesidir.

3. YÖNTEM

Geliştirecek proje; kullanımı kolay, sade ve anlaşılır olmalıdır. Bilişimden ve teknolojiden uzak olan ailelerin faydalanabilmesi için kullanıcının arayüze ilk erişim sağladığı andan itibaren kolaylıkla anlayabileceği ve kullanabileceği şekilde sade bir tasarıma sahip olmalıdır. Bu sadelik içerisinde işlevsel bir tasarım oluşturulması gerekmektedir. Bunun için bahsedilen teknolojilerden ön yüz geliştirme kısmında Angular teknolojisi kullanılmıştır. Angular içerisinde ön yüz yapısı için HTML, güzel ve kullanışlı bir tasarım oluşturmak için CSS ve PrimeNG kullanılmıştır. Angular ise sahip olduğu yapı sayesinde proje ön yüzünü işlevsel ve etkileşimli hale getirmiştir.

Ön yüze veri sağlayacak olan arka yüz ise Java Spring Boot ile geliştirilmiştir. Projeye sağlayacağı güvenlik imkanları ve veri iletişim hızı sayesinde acil vakalarda açık vermeden verilerin iletimi gerçekleştirilebilir. Yine Spring Boot ile geliştirilen APİ sayesinde tam entegre bir sistem haline gelen proje geliştirmeye ve değiştirilmeye açıktır.

Verilerin depolandığı ve veriler üzerinde işlem yapılan veri tabanı ise ücretsiz olarak projenin bütün ihtiyaçlarını karşılayan PostgreSQL olmuştur. Barındırdığı SQL dili ile gerçekleştirilmesi gereken işlevleri hızlı ve güvenli bir şekilde yerine getirecektir.

Geliştirilen proje APİ sayesinde iletişim kuracak olup belirtilen adresler dışında gelen her isteği reddedecektir. Belirtilen adresler ile iletişim sağlamaya çalışan kullanıcıların ise yetkileri kontrol edilerek yetkisiz yapılan istekler yine reddedilecektir. Geliştirilen projede 3 adet kullanıcı olacaktır. Bunlar:

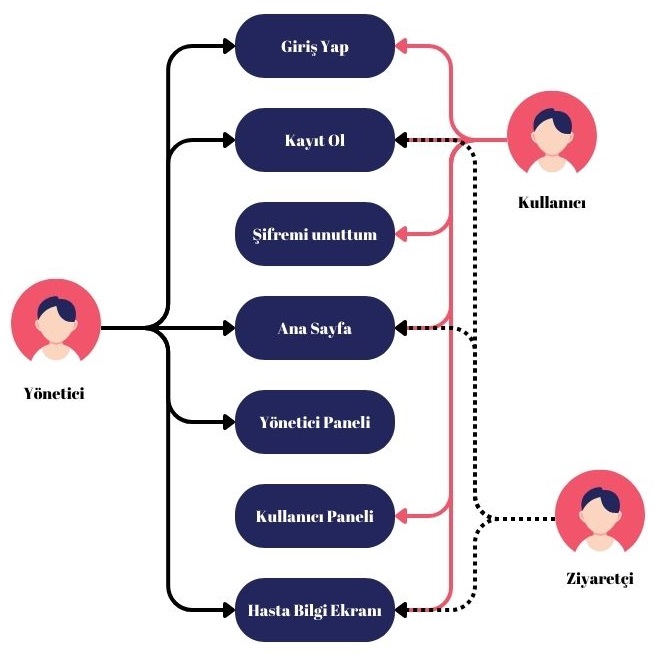
* Yönetici
* Engelli/Hasta
* Ziyaretçidir.

Her kullanıcı daha öncesinde yetkilendirilmiş bilgilere ulaşabilecektir. Kişisel verilerin korunması amacı ile kullanıcının giriş yapmadığı veriler umumi ortama açılmamış ve açılan bilgiler ise yalnızca salt okunur yani değiştirilemezdir. Her kullanıcının rolü ve bu rol yetkisi sayesinde erişim sağlayabileceği servisler belirlidir.

**Yönetici:** Yöneticinin yetkileri; veri tabanına yeni hastalık/engel eklemek, veri tabanına eklenmiş olan engelleri/hastalıkları güncellemek veya silmektir. Hasta/engelli vatandaşlarımızın eklediği hastalıkları/ engelleri kaldıramaz veya değiştiremez.

**Engelli/Hasta Yetkisi**: Engelli/hasta vatandaşlar kişisel bilgilerini ekleyebilir ve veri tabanındaki yönetici tarafından eklenmiş olan engellerden/hastalıklardan sahip olduklarını kendilerine tanımlayabilirler. Diğer engellilerin/hastaların bilgilerini görüntüleyebilir ancak değiştiremez veya silemez. Veri tabanına yeni hastalık ekleyemez, veri tabanında bulunan hastalıkları/engelleri değiştiremez ya da silemez.

**Ziyaretçi Yetkisi**: Ziyaretçiler QR kodu okutarak veya başka bir yöntem ile gönderdikleri URL yardımıyla gönderilen hasta/engelli eşsiz kimliğine bağlı olan hastaların/engellilerin verilerine ulaşabilir ve görüntüleyebilir. Engellilerin/hastaların bilgilerini değiştiremez veya silemez. Veri tabanına yeni hastalık ekleyemez veya veri tabanında bulunan hastalıkları/engelleri değiştiremez ya da silemez.



**Şekil 3.1.** Kullanıcı erişim yetki şeması

Bütün kullanıcılar, uygulamanın giriş yapma ve kayıt olma sayfasına erişebilir. Ziyaretçiler kayıt ol sayfasından T.C. kimlik numarası, isim, soy isim, telefon numarası ve şifre oluşturarak uygulamaya kayıt olabilir. Girilen bilgiler kayıt edilmeden önce veri tabanından sorgulatılarak kontrol edilir. Eğer aynı T.C. kimlik numarasına sahip bir kayıt var ise kayıt işlemi iptal edilerek kullanıcıya hata mesajı gösterilir. Kayıt olan kullanıcılar Engelli/Hasta yetkisi kazanır.

Giriş yap sayfasındaki gerekli bilgiler doldurularak uygulamanın kullanıcı panellerine giriş yapabilir. Girilen her bilgi, kontrol edilerek sahip olduğu yetki paneline aktarılır.

**3.1. Proje Geliştirme Aşamaları:**

Proje 5 aşamada gerçekleştirilmiştir. Bunlar sıralı olmayıp birbirileri ile etkileşimde olmuştur. Aşamalar şu şekildedir:

* Veri tabanının tasarlanması ve geliştirilmesi
* Arka yüz tarafının tasarlanması ve geliştirilmesi
* Ön yüz tarafının tasarlanması ve geliştirilmesi
* Test aşaması
* Hata giderme aşaması

3.1. Veri Tabanının Tasarlanması ve geliştirilmesi:

Veri tabanı PostgreSQL tarafından sağlanmış olup verilerimiz SQL dili ile işlenecektir. Veri tabanında tablolar oluşturularak ilişkileri kurulmuştur.

**Hasta Tablosu:** Bütün hasta veya engelli kullanıcılarının bilgilerinin bulunduğu hasta adında bir tablo oluşturulmuştur. Siteye kayıt olurken ve kayıt olduktan sonra hasta veya engelli vatandaşların girdiği veriler bu tabloda depolanmıştır. Kayıt olan her hasta veya engelli vatandaşa bir adet eşsiz kimlik verilmiştir. Veri çekme, kayıt etme, silme ve güncelleme işlemleri bu eşsiz kimlik ile yapılmıştır. Tabloda bulunan veriler, tipleri ile aşağıda belirtilmiştir.

* id (int) : Boş bırakılamaz, birincil anahtar, eşsiz kimlik, otomatik yükseltme
* adi (varchar) : boş bırakılamaz
* tckimlik (varchar) : boş bırakılabilir
* tel (varchar) : boş bırakılabilir
* yakin\_tel1 (varchar) : boş bırakılabilir
* yakin\_tel2 (varchar) : boş bırakılabilir
* dogum\_tarihi (date) : boş bırakılabilir
* cinsiyet (varchar) : boş bırakılabilir

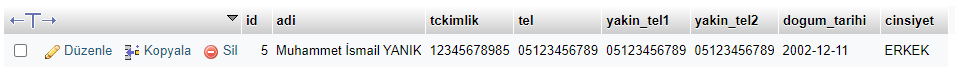
**Hastalık Tablosu:** Yalnızca yönetici tarafından işlenebilen diğer kullanıcılara salt okunur olan hastalık veya engellerin bulunduğu tablodur. Yönetici hastalık ekleme, silme, güncelleme gibi işlemlerini bu tablo üzerinde gerçekleştirir. Eklenen her hastalık veya engele bir eşsiz kimlik verilir ve bütün işlemler bu eşsiz kimlik ile yapılır. Hastalık tablosunun barındırdığı sütunlar veri tipleri ile birlikte aşağıda verilmiştir.

* id (int) : Boş bırakılamaz, birincil anahtar, eşsiz kimlik, otomatik yükseltme
* adi (varchar) : Boş bırakılamaz
* tanim (varchar) : Boş bırakılamaz
* belirtiler (varchar) : Boş bırakılamaz
* yapilmasi\_gerekenler (varchar) : Boş bırakılamaz
* yapilmamasi\_gerekenler (varchar) : Boş bırakılamaz

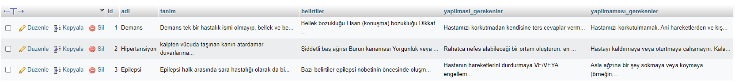
**Hasta-hastalık tablosu:** Hasta-hastalık tablosu hasta veya engelli kullanıcıların sahip olduğu hastalıkları veya engelleri kendilerine tanımladığı tablodur. Bu tablo, 4 adet sütun barındırır. Bunlar aşağıda verilmiştir:

* id (int) : Boş Bırakılamaz, birincil anahtar, eşsiz kimlik, otomatik yükseltme
* hasta\_id (int) : Boş bırakılamaz, yabancı anahtar
* hastalik\_id (int) : Boş Bırakılamaz, yabancı anahtar
* teşhis\_tarihi (date) : Boş bırakılabilir

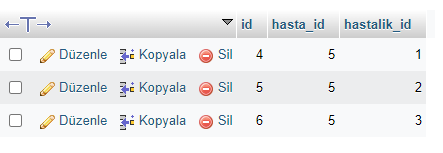
Hasta-hastalık tablosu, hasta tablosu ve hastalık tablosunun idleri ile yabancı anahtar ilişkisi kurar. Her hasta veya engelli birden fazla hastalık veya engele sahip olabilir. Bu nedenle hastalıkları hastaya tanımlamak için hastaların hastalıklarını tutan bir tablo gereksinimi ortaya çıkmıştır. Burada bu gereksinimi hasta-hastalık tablosu karşılar. Aşağıdaki resimler ile bu durum örneklendirilmiştir.



**Şekil 3.2.** Hasta kayıt örneği



**Şekil 3.3.** Hastalık kayıt örneği



**Şekil 3.4.** Hastaların hastalıklarının kayıt örneği

Resimlerde görüldüğü gibi 4,5 ve 6 numaralı id’ye sahip olan satırların “hasta\_id” veri kısmında 5 rakamı yazmaktadır. Bu 5 rakamı, 5 numaralı id’ye sahip olan hastayı belirtmektedir. 4,5 ve 6 rakamlı id’ye sahip verilerin “hastalık\_id” kısımları ise farklı hastalık id’lerine sahiptir. Anlaşılacağı üzere 5 numaralı id’ye sahip olan hasta, hastalık kısmında yazan id’li hastalık veya engellerin tümüne sahiptir.

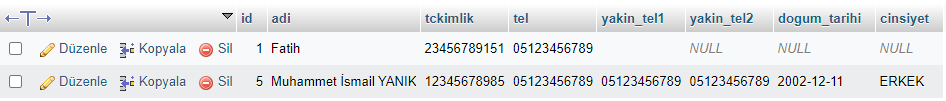
**Kullanıcı Tablosu:** Kullanıcı tablosu, uygulama kullanıcılarının bilgilerinin barındırıldığı tablodur. Yönetici ve hasta/engelli kullanıcıların ortak olan bilgilerinin bulunduğu tablodur. Bu tablo genellikle şifre işlemleri ve elektronik posta işlemlerinde kullanılır. Güvenlik amaçlı olarak veriler tek tabloda tutulmayıp, kullanıcı bilgileri iki tablo halinde ayrılmıştır. Böylece hasta veya engelli kullanıcıların hastalık bilgileri sorgulanırken şifre ve mail bilgilerine erişilecek olan yollar ayrılmıştır. Bu yöntem ile siber saldırılara önlem alınmıştır. Kullanıcı tablosu sütun isimleri, veri tipleri ve özellikleri aşağıda verilmiştir.

* id (int) : Boş Bırakılamaz, birincil anahtar, eşsiz kimlik, otomatik yükseltme
* kullanici\_kodu (varchar) : Boş bırakılamaz
* tckimlik (varchar) : Boş bırakılabilir
* kullanici\_adi (varchar) : Boş bırakılamaz
* sifre (varchar) : Boş bırakılamaz
* mail (varchar) : Boş bırakılamaz
* active (boolean) : Boş bırakılamaz

Kullanıcı tablosunun “kullanici\_id” sütunu ile “kullaniciRole” tablosunun “kullanici\_id” sütunu birbirine yabancı anahtar ile bağlanmıştır. “KullaniciRole” tablosunda kullanıcılarımızın yetkilendirmeleri bulunmaktadır. Böylece bu tablo ile giriş esnasında yetki ve kimlik kontrolü yapılabilmektedir.

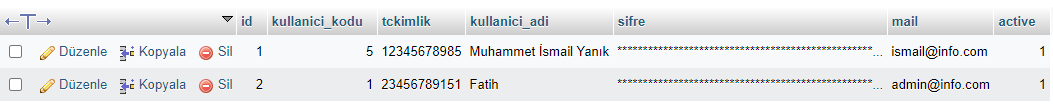
**KullaniciRole Tablosu:** “KullaniciRole” tablosu, kullanıcıların yetkilerini barındıran tablodur. Bu tabloda iki adet veri bulunmaktadır. Bunlardan biri kullanıcı eşsiz kimliği bir diğeri ise yetki tipidir. Böylece kullanıcılara yetkilendirme yapılmış olur. Giriş esnasında yetkilendirme kontrolü bu veri tablosundan çekilen veri ile sağlanır. Tabloda bulunan sütun adları, veri tipleri ve özellikleri aşağıda verilmiştir.

* id (int) : Boş Bırakılamaz, birincil anahtar, eşsiz kimlik, otomatik yükseltme
* kullanici\_id (int) : Boş Bırakılamaz
* role (varchar) : Boş bırakılamaz



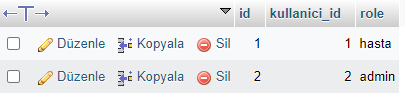
**Şekil 3.5.** Kullanıcı kayıt örneği

Resimde görüldüğü üzere “Fatih” adlı kullanıcı hasta tablosunda 1 numaralı eşsiz kimlik ile kayıtlıdır.



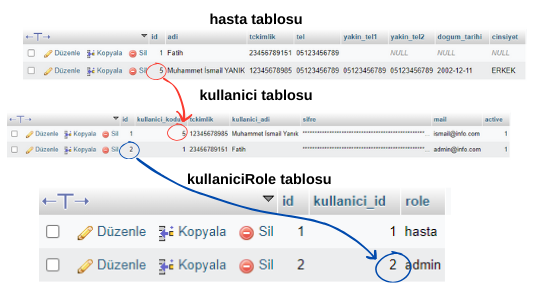
**Şekil 3.6.** Kullanıcı yetkileri kayıt örneği

Resimde görüldüğü üzere kullanıcı tablosunda 2 numaralı eşsiz kimliğe sahip olan satırın “kullanici\_kodu” sütunu, “1” verisine sahiptir. Bu sütun “hasta” adlı tablonun id’si ile yabancı anahtar kullanarak ilişkilendirilmiştir. Bu bağıntıdan “kullanici” adlı tablonun 2 numaralı eşsiz kimlik verisinin hasta adlı tablodaki 1 numaralı eşsiz kimliğe sahip kişi olduğu anlaşılmaktadır.



**Şekil 3.7.** Yetki kayıt örneği

Yukarıdaki resimde görüldüğü üzere “kullaniciRole” tablosunun “id” sütunu “2” olan satırın, “kullanici\_id” sütununda “2” verisi bulunmaktadır. Aynı satırın “role” sütununda admin verisi bulunmaktadır. Bu sütunda “kullanici\_id” sütunundaki kullanıcıların yetkileri belirtilir.



**Şekil 3.8.** Kullanıcı yetkisi kayıt anlatımı

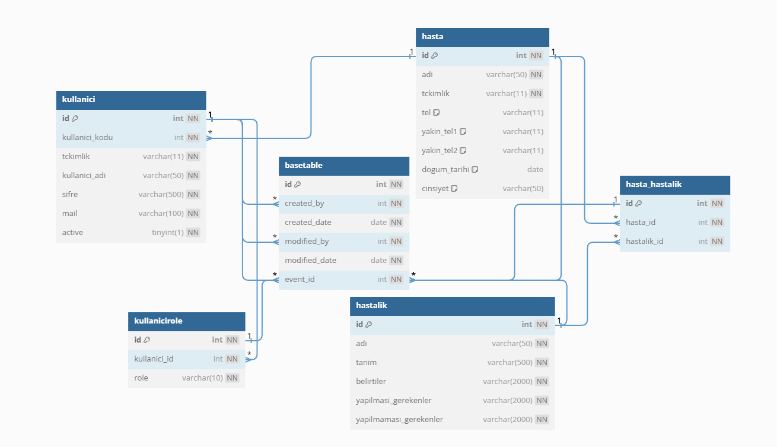
Yukarıdaki resimde, anlatılanlar şekil ile gösterilmiştir.

**BaseTable Tablosu:** “BaseTable” tablosu, veri tabanında yapılan değişiklikleri tutar. Bu tablo, veri tabanının “log tablosu” olarak tanımlanabilir. Yapılan işlerin kimin tarafından ve ne zaman yapıldığını, değiştirilen verilerin kimin tarafından ve ne zaman değiştirildiğinin bilgisini tutar. Bu tablonun sütun adları, veri tipleri ve sütun özellikleri aşağıda verilmiştir.

* id (int) : Boş Bırakılamaz, birincil anahtar, eşsiz kimlik, otomatik yükseltme
* created\_by (varchar) : Boş bırakılamaz, değiştirilemez
* created\_date (date) : Boş bırakılamaz, değiştirilemez
* last\_modified\_by (varchar) : Boş bırakılamaz
* last\_modified\_date (date) : Boş bırakılamaz
* event\_id (varchar) : Boş Bırakılamaz

Yapılan işlerin kontrolü ve her işin buraya kaydedilmesi veya güncellenmesi projenin arka yüz kısmı tarafından kontrol edilecektir.

Veri tabanının oluşturulmuş tabloları ve ilişkilendirmeleri aşağıdaki şekilde gösterilmiştir.



**Şekil 3.9.** Veri̇ tabanı ili̇şki̇ şeması

3.2. Ön Yüz Tarafının Tasarlanması ve Geliştirilmesi:

Projenin ön yüz tarafı bahsedilen teknolojilerden olan Angular ile geliştirilmiştir.

Projede 7 adet sayfa bulunmaktadır. Bu sayfalar aşağıda maddeler halinde verilmiştir.

* Ana sayfa
* Giriş yap sayfası
* Kayıt ol sayfası
* Şifremi unuttum sayfası
* Yönetici paneli
* Kullanıcı paneli
* Hasta/Engelli bilgi sayfası

Bahsedilen sayfalar açıklamaları ile birlikte aşağıda verilmiştir.

**Ana sayfa:** Ana sayfa sitenin tanımını yapan, siteye ilk erişilen yerdir. Bu sayfada projenin tanıtımı yapılmaktadır. Hakkımızda kısmında projenin amacı, faydaları ve özellikleri bulunmaktadır. “Hizmetimiz” kısmında ise kullanıcılara sunulan hizmet şablonlar ile açıklanmaktadır. “Sık Sorulan Sorular” kısmında ise kullanıcıların merak ettiği ve sıkça sorulan soruların cevaplarıyla birlikte yer almaktadır. İletişim kısmında kullanıcıların, geliştiricilere elektronik posta göndererek ulaşabileceği bir alan bulunmaktadır. Bunun dışında sayfanın “footer” kısmında iletişim bilgileri ve sayfa haritası bulunmaktadır.

**Kaydol sayfası:** “Kaydol” sayfası ad, soy ad, elektronik posta ve şifre bilgileri girilerek yeni bir hesap oluşturulan sayfadır. Buradan kaydolarak hasta/engelli profili oluşturulur.

**Giriş yap sayfası:** “Giriş yap” sayfası kayıtlı kullanıcıların elektronik posta ve şifreleriyle kullanıcı panellerine giriş yapacakları sayfadır. Bu sayfa, kullanıcı bilgilerini kontrol ettikten sonra kullanıcıyı yetkilendirerek kullanıcı paneline yönlendirir.

**Şifre yenileme sayfası:** “Şifre yenileme” sayfası elektronik posta adresi ve telefon numarası bilgilerini alarak sistemden doğruluklarını kontrol eder. Eğer bilgiler doğrulanır ise sistem, kayıtlı elektronik posta adresine güvenlikli yeni bir şifre gönderir.

**Kullanıcı paneli:** Kullanıcı paneli; kullanıcıların bilgilerinin görüntülendiği, değiştirilebilindiği ve silinebildiği sayfadır. Bu sayfa sayesinde kullanıcı bilgilerini ekleyebilir, değiştirebilir ve silebilir. QR kod oluşturucusu ile QR kodu indirilebilir. “Şifre değiştir” kısmından eski şifre doğrulanarak şifre yenisi ile değiştirilebilir. Kullanıcı, “Tanımlı Hastalıklar” kısmında tanımlı hastalıklarını görüntüleyebilir ve silebilir. “Hastalıklar” tablosundan ise hastalık veya engel tanımlanabilir. “Hastalık” tablosundan “Tanımlı Hastalıklar” tablosuna tanımlanan hastalıklar, artık “Hastalıklar” tablosunda görüntülenmeyeceğinden bir hastalık birden fazla kez tanımlanamaz.

**Engelli/ hasta bilgi sayfası:** “Engelli/Hasta Bilgi Sayfası” ile ID’si URL adresinde gönderilen hastanın kayıtlı bilgilerine ve üzerindeki tanımlı hastalıklara ulaşılır. Tek bir tıklama ile “112 Acil Çağrı” butonuna tıklanarak acil çağrı merkezi aranabilir.

**Yönetici paneli:** “Yönetici Paneli” yöneticinin hastalık ekleyip silebileceği ve güncelleme yapabileceği paneldir.

Angular projelerinde APİ servislerinden veri çekme işleminde proje parçalara ayrılarak geliştirmek gerekir. Bu parçalar genel olarak model, view ve service olarak adlandırılır. Her bir parça farklı bir amaca hizmet etmektedir. Böylece tekrar kullanılabilmektedir. Bu sayede geliştirmeye açık bir proje ve düzenli kod yazımı elde edilebilmektedir.

**Model:** Angular’da modeller veri aktarımında kullanılan nesnelerdir. Api’den veri çekme işlemi yapılırken gönderdiğimiz istek sonucu dönen veriler angulara “json” veri formatında gelirken kullanım kolaylığı sebebi ile angular, bu verileri gelecek veri demeti için oluşturulmuş modellere dönüştürür. Bu sayede cevap olarak gelen veriyi işlerken “json” formatıyla değil java nesneleri ile işlenebilmektedir.

Projemizde veri tabanından, yönetici paneli için hastalık tablosunda bulunan gerekli veriler, kullanıcı panelinde görüntülenecek olan hasta hastalık tablosundaki hastaya ait hastalık bilgileri, hasta tablosundaki hasta bilgileri, hastalık tablosundaki hastalık bilgileri ve şifre değiştirmek için ise kullanıcı tablosundaki şifre bilgilerine ihtiyaç vardır. Gerektiği yerlerde bu bilgiler sunucudan istenerek işlenecektir. Bu nedenle yanıt olarak gelen bu bilgileri alabilmek için yazılan her servisin yanıtında gelecek olan bilgileri modele çevirmemiz gerekmektedir.

Bu modeller, “DTO” olarak adlandırılır. “Data Transfer Object” kelimelerinin kısaltılmış hâlidir. Veri alışverişi sırasında gizli kalması istenen, tekrar edilmesi istenmeyen veya gereksiz bilgileri taşımayarak daha hızlı bir sistem geliştirmek ve kaynak tasarrufu sağlamak için geliştirilmiştir. Yalnızca kullanılacak verilerin bulunduğu bir model oluşturularak fazlalık bilgiler dahil edilmez.



**Şekil 3.10.** DTO örneği

Yukarıdaki resimde görüldüğü üzere veri tabanındaki tablodan gelen verilerin java nesnesine çevrilebilmesi için isimlerinin aynı olması ve veri tiplerinin uyumlu olması gerekmektedir.

Uygulama sayfalarında gösterilmek üzere hastalık bilgileri için istek yollandığında dönecek olan hastalık verileri için “HastalikDTO” adında bir model tanımlanır.

Servisler: Servisler “APİ” iletişiminde sunuculara istek gönderilen katmandır. Hangi sunucuya ne istek gönderileceği bu katmanda belirlenir. Gönderilen isteklerin sonuçları yine bu katmanda yakalanır ve işlenir.

Servis tanımlarken öncelikle gerekli importlar yapılarak başlanır. Angular çekirdeğini içeri aktardıktan sonra sırası ile http protokolünü kullanmak için httpClient ve HttpErrorResponse modülleri içeri aktarılır.

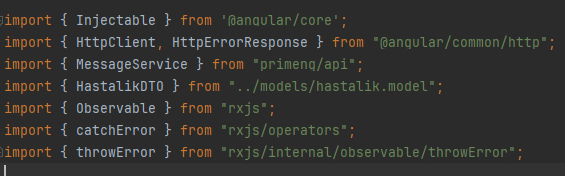
**MessageService:** Kullanıcıya bilgi mesajlarını göstermek için primeNG framewörkünün sahip olduğu MessageService modülü içeri aktarılır.

**HastalikDTO:** apiden gelen yanıttaki json formatındaki verileri java nesnesine dönüştürmek için HastalikDTO modeli yüklenir.

**Observable:** observable modülü, sunucu tarafına yapılan istekleri dinlemek ve değişiklikleri yakalayabilmek için içeri aktarılır.

**catchError :** istek sonucu gelen hataları yakalamak için kullanılır.

**throwError :** bazı hataları görmezden gelmek için kullanılır.



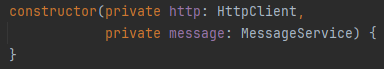
**Şekil 3.11.** component.ts içeri aktarımları

“baseUrl” adında “string” tipinde bir değişken oluşturarak içerisine isteğin yapılacağı sunucu adresi yazılır.



**Şekil 3.12.** İstek yapılacak adresin tutulması

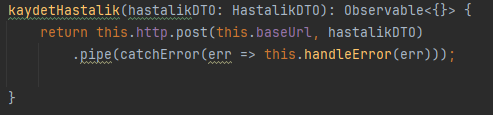
Servislerin bulunduğu sınıfın başlatıcısını http protokolünü kullanmak için “HttpClient” nesnesini parametre olarak gönderilir ve başlatıcı içerisinde “MessageService” modülü daha sonra kullanabilmek için nesneleştirilir.



**Şekil 3.13.** Component constructer’ı

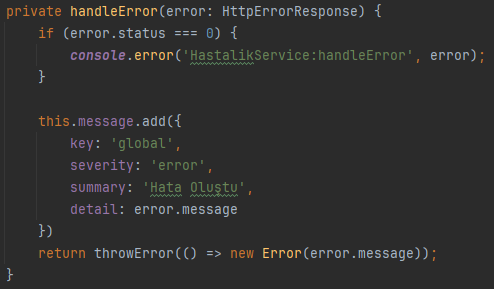
Örneğin hastalık kaydetmede kullanılan metot şu aşamalar ile oluşturulur.

Metot oluşturulurken metodun adını yazarak parametre olarak istekte bulunurken gönderilecek bilgilerin bulunduğu java nesnesi eklenir. “Observable” ile metot dinamik dinleyici haline getirilir. Return ile işlemin sonucu yakalanır ve çağrılan yere gönderilir. Daha önce belirtildiği üzere this.http.post kodu ile bunun bir kayıt işlemi olduğu ve http protokolü ile gönderileceği bilinmektedir. Post metodu birçok parametre içerir, bu servis için iki parametre yeterli olacaktır. Öncelikle isteğin yapılacağı adres belirtilir ve gönderilecek bilgilerin bulunduğu java nesnesi eklenir. Geri dönüş verisi, pipe ile dönüştürülerek hata yakalama nesnesi için yazılmış handleError metoduna gönderilir. Gelen veri ilk başta yazılan return ile metodun çağrıldığı yere geri dönecektir.



**Şekil 3.14.** Servis örneği

“handleError” metodu kendisine gelen hata mesajını kontrol ederek başlatıcıda oluşturulan “message” ögesi ile kullanıcı arayüzüne sunmaktadır.

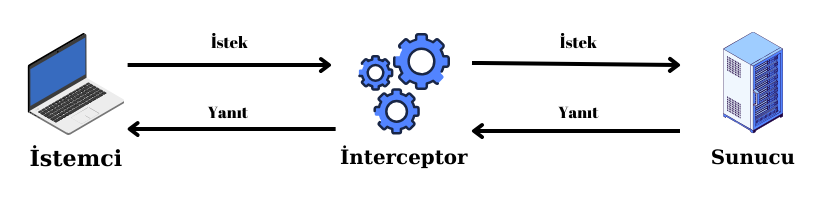


**Şekil 3.15.** Hata yakalama

Bir APİ ile iletişime geçmek istenildiği zaman servis ile istek gönderilir. Gelen cevap hazırlanmış olan modele dönüştürülür ve view katmanına gönderilerek kullanıcıya sunulması sağlanır.

Sunucu tarafından gelen isteğin kimlik doğrulaması yapılarak yetkisi kontrol edilir. Eğer iki aşamadan da geçer ise istek işleme alınır. İstek yapılırken bu aşamaları doğrulamak için http isteğinin header kısmında token gönderilir. Gönderdiğimiz token kimlik ve yetki bilgilerimizi içerir. Token gönderme işlemi her istekte bulunmalıdır. Gönderilen her isteğin içerisine token ekleme işlemi fazladan iş gücü olacağından anguların interceptor özelliği kullanılır.

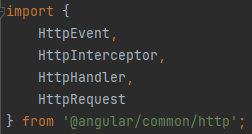
İnterceptor istemci ile sunucu arasında oluşan iletişimi kontrol eden bir yapıdır. İstemci ile sunucu iletişimin tam ortasında yer alır ve gelip giden bütün istek ve yanıtlar interceptor kontrolünde sağlanır. İnterceptor ile istemci ve sunucu arasındaki iletişimde araya girerek gerekli kayıt yönlendirme, değer ekleme gibi işlemleri tek bir yapı ile sağlanabilir.



**Şekil 3.16.** Interceptor çalışma anlatımı

Yapılan isteklerde kimlik doğrulaması ve yetki kontrolü için gönderilecek header öğesi ınterceptor yapısına tanımlanır. Böylece her seferinde gönderilmesi gereken header ögeleri, tek seferlik bir tanımlama ile her istekle birlikte gönderilebilir.

HttpEvent, HttpInterceptor, HttpHandler ve HttpRequest modülleri http olaylarını kontrol etmek, Interceptorü kullanabilmek, http istek ve cevaplarını yakalayabilmek ve istekleri yönetebilmek için içeri aktarılır.



**Şekil 3.17.** Interceptor için içeri aktarımlar

Interceptor sınıfı oluşturmak için HttpInterceptor sınıfı implement edilir. Böylece HttpInterceptor sınıfından yararlanabilir özellikleri kullanılabilir. Sınıfımızın başlatıcısı olan constructor metodunun içerisine interceptor metodu tanımlanır. Interceptor metodu, http isteklerini yakalayacağı için parametre olarak http isteği tanımlanır. Http olaylarını takip etmesi için observable özelliği eklenir. Header a token eklemek için sayfaya erişim sağlayan kullanıcının sahip olduğu tokena ihtiyaç vardır. Bunun için siteye erişim sağlayan kullanıcıların json formatında localstoreg’a depolanan bilgilerin arasından tokenını almak gerekir.

Currentuser adında bir değişken oluşturup içerisine tokenı kayıt edilir. Token, localstorage.getItem metodu kullanılarak elde edilir ve JSON.parse metodu ile elde edilen json formatındaki veri parçalanarak java nesnesi haline getirilir. Eğer current user dolu ise istek, clone metodu ile klonlanarak setHeaders ile header öğesi değiştirilir. İsteğin headers öğesindeki Authorization tarafına token eklenir.

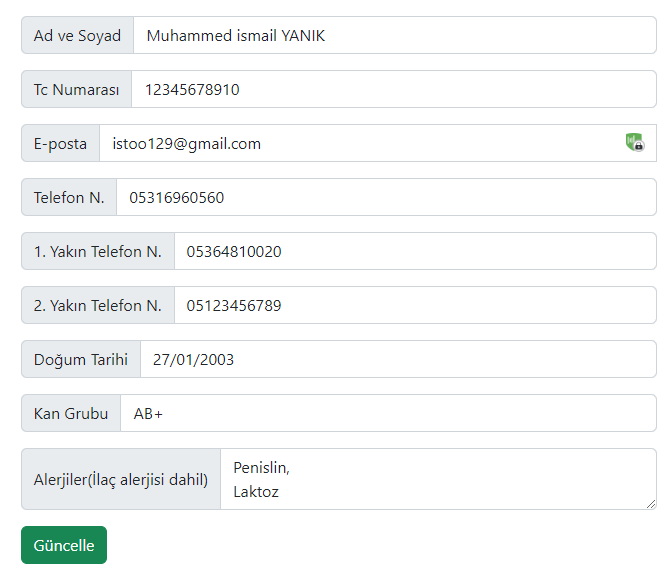


**Şekil 3.18.** Interceptor Kullanımı

**View:** View katmanı, elde edilen verilerin görüntüleme işleminin yapıldığı katmandır. View katmanında görüntülemek istenilen veriler için html css gibi teknolojiler kullanılarak sayfalar oluşturulur ve model ile gelen veriler, anguların data binding özelliği ile gösterilmesi istenen kısımlarda kullanıcıya sunulabilir. Html ile oluşturulmuş sayfalar angular yardımı ile etkileşimli hale gelir. Sunucuya yapılan istekler ve dönen yanıtlar yine angular tarafından yönetilir.

Projemizin view kısmını oluştururken html kodları ile sayfanın elementleri oluşturulur. Css ve primeng yardımı ile sayfanın görselliği geliştirerek daha kullanışlı hale getirilebilir.

Kullanıcı paneli sayfasında kullanıcı bilgilerini görüntülemek, güncellemek ve kayıt edebilmek için sayfaya bir adet form ögesi ve buton eklenir.



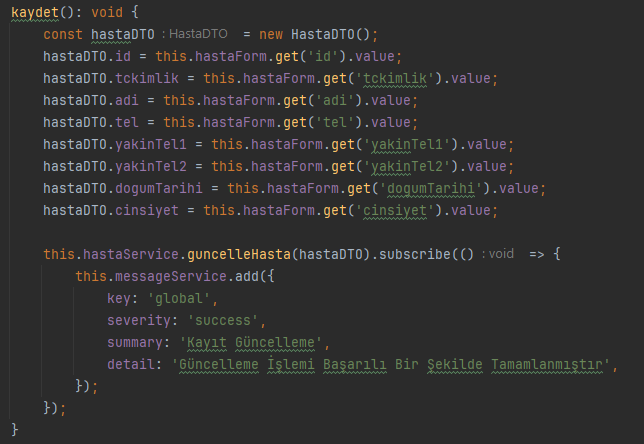
**Şekil 3.19.** Kullanıcı paneli form öğesi

Form ögesi, primeNG ve html elementleri ile oluşturulur. Anguların farkı ise nesne tabanlı programlamaya destek verdiği için isteği form ögesi değil angular kendisi yapacaktır. Bunun için güncelle butonunun tipini submit yapılır ve kaydet metoduna bağlanır.



**Şekil 3.20.** Form öğesi HTML kodları

Kaydet metoduna yeni bir hastaDTO nesnesi oluşturulur. Angular data bindin ile formdaki bilgileri alarak oluşturulan hastaDTO nesnesine aktarılır. Sunucuya istek yapmak için oluşturulan observable özelliği olan metot, burada çağırılarak tetiklenir. Bunun için hastaservis adı ile nesneleştirilen servis sınıfının içerisindeki guncellehasta metoduna formdan bilgiler çekerek oluşturulan hastaDTO nesnesi parametre olarak verilip observable özelliğini aktif etmek için subscribe metodu ile primeNG frameworkünün sağladığı bildirim mesajı özelliği kullanılarak çağırılır.



**Şekil 3.21.** Kullanıcı bilgileri güncelleme

Kullanıcının QR kodunu oluşturması ve bilgi sayfasını görebilmesi için QR kod oluşturma aracı gereklidir. Bunun için de angular açık kaynak kütüphanelerinden olan angularx-QRcode kütüphanesi npm ile yüklenerek QRcode tagları ile kullanılabilir.



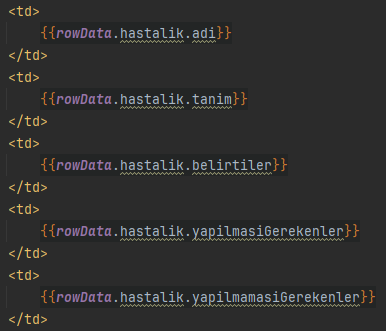
**Şekil 3.22.** QR kod oluşturma aracı

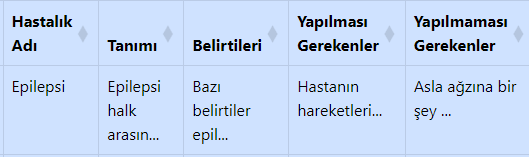


**Şekil 3.23.** QR kod oluşturma aracı HTML kodları

Angular teknolojisinde apiden çekilen verileri view katmanında istemcide görüntülemek için veya istemciden alınmak istenen bir girdiyi typescript modülünde kullanmak için anguların data binding yani veri bağlama özelliği kullanılır. Anguların bu özelliği sayesinde tek yöblü veri bağlama yapılır ise typescript modülünde oluşturulan bir veriyi html dosyalarında görüntülemek mümkündür. Eğer iki yönlü veri bağlama yapılırsa typescript değişkenleri html dosyalarına taşındığı gibi html inputları ile alınan veriler de anlık olarak typescript değişkenine atanabilir.

Tek yönlü veri bağlama işlemi html kodları arasına iki defa açılmış süslü parantez arasına yazılan değişken ile sağlanır. Böylece süslü parantezlerin açıldığı yere typescript değişkeninin değeri yazılmış olur.

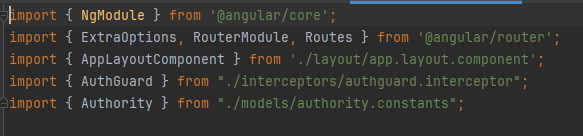




**Şekil 3.24.** Tek yönlü data binding örneği

**Angular Yönlendirmelerinin oluşturulması:**

Projede her sayfa için farklı modüller kullanılmış ve bu modüller html ve css sayfalarına bağlanmıştır. Şimdi ise bu modüllerin adreslere yönlendirilmesi gereklidir. Angular ile yönlendirme, projede app-routing.ts adlı dosyada gerçekleştirilecektir. Böylece sitedeki trafiği kontrol edilerek gerekli adreslerin yakalandığı anda doğru yönlendirmeler ile modüller yüklenecektir. Yönlendirme yapmak ve trafiği kontrol etmek için routing modülleri içeri aktarılmalıdır. Bunun için RouterModule, Routes modülleri içeri aktarılmalıdır. Yetki kontrolü için ise AuthGuard ve Authority modülleri içeri aktarılmalıdır.



**Şekil 3.25.** Yönlendirme için gerekli içeri aktarımlar

Const tipinde routes adında bir değişken oluşturularak gerekli yönlendirmeler içerisine tanımlanır. Kullanılacak metot, her yönlendirmeyi modül paketine yapıp modül paketinde yeni yönlendirmeye tabi tutmak olacaktır. AuthGuard modülü aktif hale getirilerek yönetici ve kullanıcı panellerinin modülleri yetkiye tabi tutulacaktır. Kullanıcı giriş yaptığında giriş bilgileri kontrol edilerek gerekli yetkilendirme sunucu tarafından verilecek, ardından verilen yetkilendirme kontrol edilerek yetkisinin giriş yapabileceği modüller yüklenecektir. Eğer kullanıcı sunucudan ADMIN yetkisi almış ise views klasörünün içerisinde bulunan tanımlar klasöründe bulunan “tanimlar.module” modülü yüklenecektir. Eğer kullanıcı USER yetkisi almış ise views kalsöründe bulunan işlemler klasöründeki “işlemler.module” modülü yüklenecektir.

Bu iki modül dışındaki modüller her kullanıcıya açık olduğu için diğer yönlendirmeler yetki kontrolüne tabi tutulmadan yapılacaktır.



**Şekil 3.26.** Yönlendirme örneği

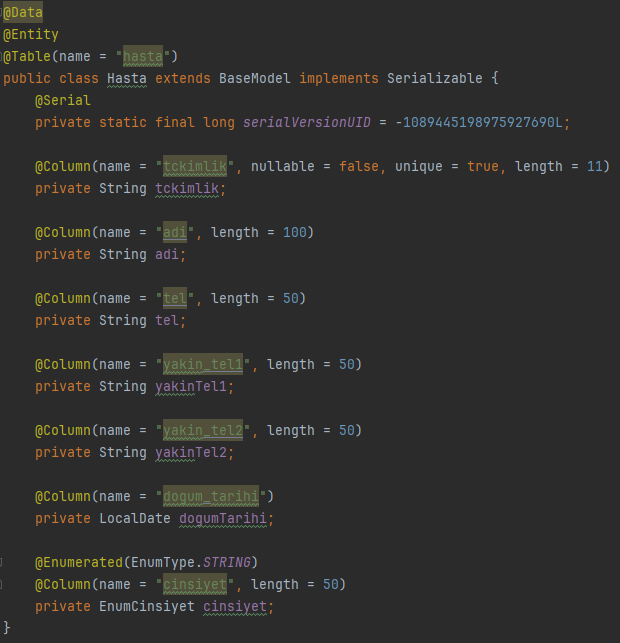
3.3. Projenin Backend Tarafının Tasarlanması

Projede backend kısmı bilgilerin tutulduğu, istemci tarafından yapılan istekleri yakalayarak bu istek doğrultusunda yanıt hazırlayıp hazırlanan yanıtı doğru formatta tekrar istemciye gönderecek olan kısımdır. Amaç, mikroservis mimarisi ile entegre olabilecek bir API oluşturup gelen istekleri yanıtlayabilmektir.

API servisi oluşturmak için hazırlanan veri tabanını en etkili şekilde kullanmak için java dilinde modellemek gereklidir. Böylece veri tabanı javanın işleyebileceği şekilde nesneleştirilerek daha etkin kullanılabilir.

Veri tabanındaki tabloları java nesnesi olarak tanımlama işleminde yapılması gereken tabloları sınıf olarak, sütunları ise @Column anotasyonu ile değişken olarak tanımlamaktır. @Column anotasyonundan sonra parantez içerisinde sütun ismi ve sütunun diğer kuralları tanımlanmalıdır. Bu kuralların birebir aynı olması karşılaşılacak hataların önüne geçer. Projede her tablo bahsedilen şekilde java nesnesi olarak tanımlanmıştır.

Eklenen @Data anotasyonu, java nesnelerinde güvenli veri alışverişi için kullanılan get ve set metotları, hashcoding gibi kod fazlalığı oluşturacak metotları eklemeden projede varmış gibi derleme zamanında eklenerek iş yükü azaltılır. Bu sayede @Data anotasyonu ile bu metotları kullanılabilir.

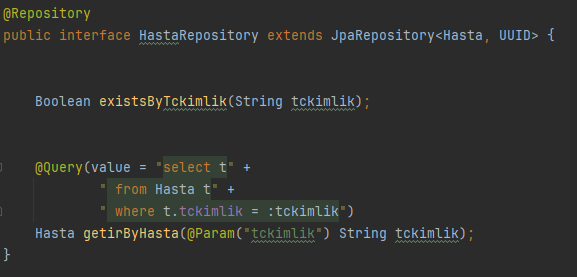


**Şekil 3.27.** Veri tabanı tablosunun java ortamında entity olarak tanımlanması

Nesneleştirilen tablo yardımı ile yapılacak sorgular java arayüzü olarak yazılır ve isteğin doğrultusunda servis bu arayüze ulaşarak istenilen sorguyu gerçekleştirebilir.

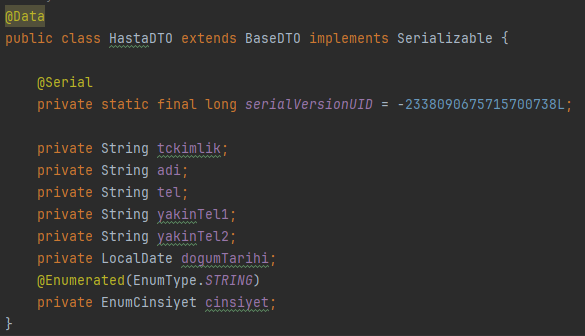
“Sorguların yazılacağı arayüze @Repository anotasyonu verilir. @Repository Annotation; sınıfın nesneler üzerinde depolama, alma, güncelleme, silme ve arama işlemi için mekanizma sağladığını belirtmek için kullanılır.”21

Sorgular, @Query anotasyonu ile HQL dilinde yazılabilir.



**Şekil 3.28.** Sorgu örneği

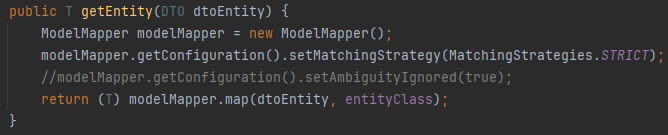
Springboot ile geliştirilen projelerde de angularda tercih edildiği gibi DTO yapıları çokça tercih edilmektedir. Gerek bilgi güvenliği gerek kaynak tasarrufu açısından birçok faydası bulunur. Bazı sorgularda, tablodaki bütün bilgilere ihtiyaç olmayan yerlerde, entity ile değil DTO ile veri akışı sağlanır. Bazı durumlarda ise özel dto’lar yazılması gerekir. Bu durumlar tabloların tek başına isteği karşılayamadığı durumlarda veya özel sorgularda meydana gelir. HastaDTO nesnesi Hasta Entity nesnesi ile değişken isimleri ve veri tipleri aynı olmak üzere tasarlanmıştır. Bunun sebebi, varsayılan model mapper kurallarına uyarak varsayılan model mappırı projede kullanmaktır. Böylece model mapperda herhangi bir değişiklik yapmadan DTO’lar, Entitylere veya Entityleri DTO’lara dönüştürülebilir.



**Şekil 3.29.** Arkayüz için DTO nesnesi örneği

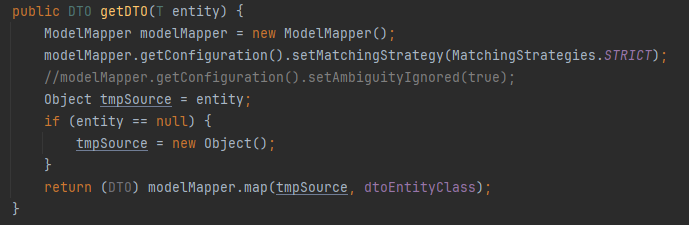
Springboot un sahip olduğu ModelMApper kütüphanesi ile Entityler ve dto’lar arası dönüşüm sağlanır. Geliştiriciler model mapper özelliğini amaçlarına uygun şekilde geliştirerek sahip oldukları sınıfları birbirlerine dönüştürebilirler.

getEntity adında geri dönüş tipi Entity olan bir metot oluşturup parametre olarak bir dto nesnesi gönderilir. Bu metot içerisine ModelMapper sınıfı nesneleştirilerek bu nesne ile dto nesnesi entit’ye dönüştürülebilir. Elde edilen entity nesnesi geri gönderilir.



**Şekil 3.30.** DTO dan entity nesnesine dönüşüm örneği

Entity nesnelerini DTO nesnelerine dönüştürmek için getDTO adında DTO geri dönüş tipine sahip ve parametre olarak Entity nesnesi alan bir metot oluşturulur. Yine ModelMApper sınıfı nesneleştirilir. Bu nesne ile entity’yi dto’ya çevirme işlemi yapılır.

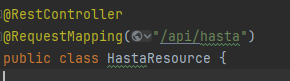


**Şekil 3.31.** Entity den DTO nesnesine dönüşüm örneği

İstekte bulunan istemcinin isteğindeki yetkileri ve hasta bilgilerinin taşınmasındaki ve işlenmesindeki kolaylığı sağlamak için cinsiyet ve rol tanımlamalarını sabit değerlere bağlamak için enum kullanılmıştır. EnumRoles sınıfı admin ve user olan iki adetenum içerir. Enum Cinsiyet ise erkek ve kadın olmak üzere iki adet enunm içerir.

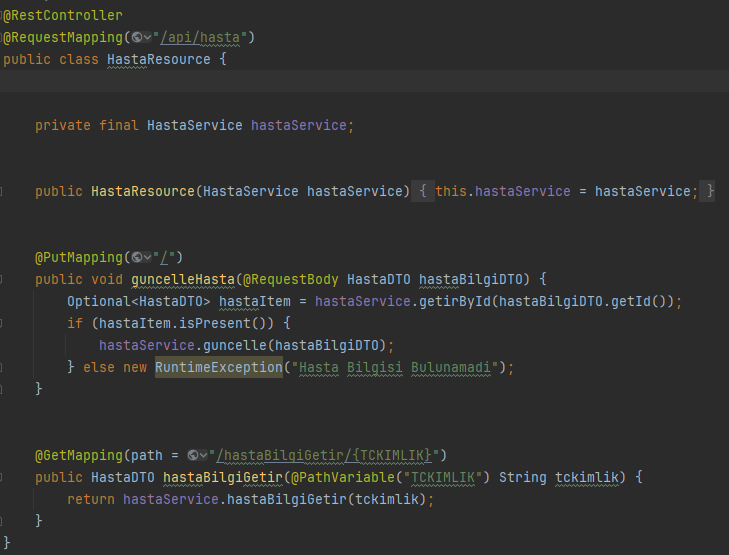
3.3.1. Api Geliştirilmesi

Geliştirilen apinin ilk yapması gereken istemciden apinin adresine gelen istekleri yakalayarak değerlendirmesidir. Bu istekleri yakalayabilmek için Resource adında yapılar oluşturulmuştur. Oluşturulan sınıflara Sprinboot’un @RestController anotasyonu verilerek bunun APİolduğu belirtilmiştir. @RequestMApping anotasyonu ile bu apilerin adresleri belirlenmiştir.



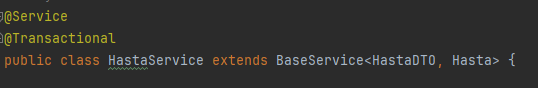
**Şekil 3.32.** API için gerekli anotasyonların tanımlanması

Böylece tanımlanan adrese gelen istekler bu sınıf tarafından karşılanacak ve işlenecektir. Doğru adres ile doğru Mapping kullanılarak yapılan isteklerin metotları tetikleyebilmesi için metotlara @PutMapping anotasyonu verilerek kök url’den sonra gelenlerin hangi metotları tetikleyeceği belirlenir. Bu metotlar, gelen istekleri gerekli servisler sayesinde çalıştıracağı için gerekli olan servisler burada nesneleştirilir. Gelen istek doğrultusunda tetiklenen metot çalışarak gerekli olan servis özelliğini çağırır.

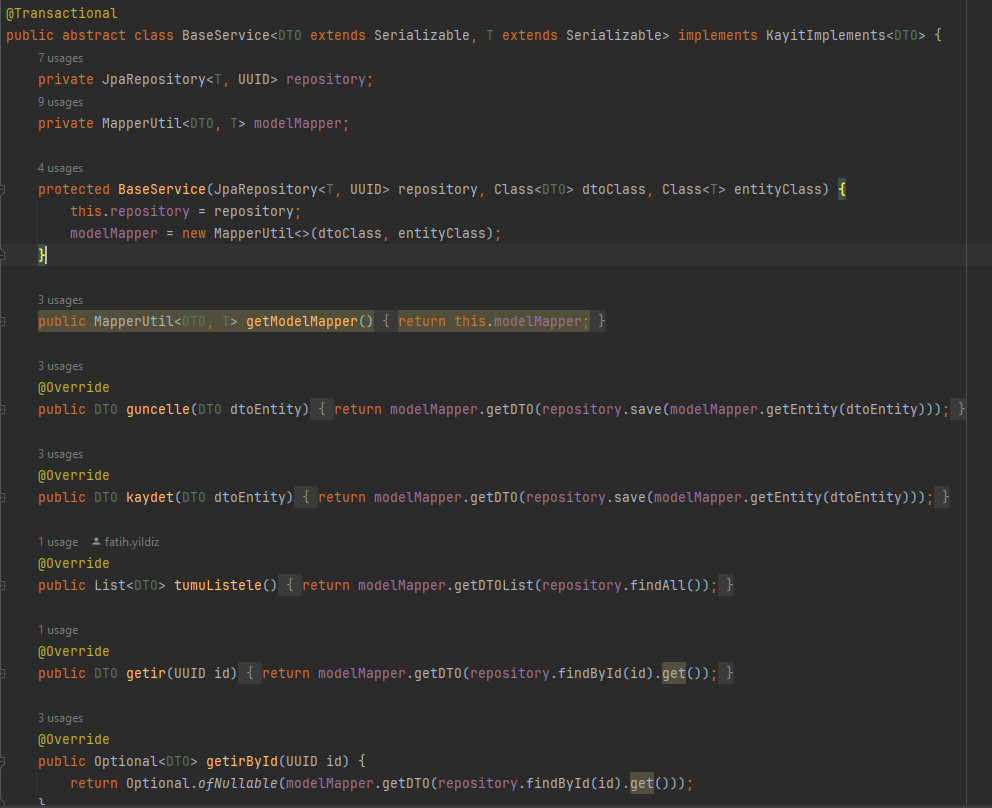


**Şekil 3.33.** Resource örneği

Servis sınıfı @Service anotasyonu verilerek bir servis olduğu tanımlanır. Servis sınıfı resourceler tarafından kendilerine gelen işlemin tanımını yaptıktan sonra isteğin cevabının verilmesi için oluşturulur. Yani ne işlem yapılacağını, isteğin ne olduğunu tanımlayan resource iken isteğin yanıtını hazırlayan ve gerekli yerlere veri isteği yapan service kısmıdır. BaseServis’den miras alarak soyut sınıf olan base servis servisinde tanımlanmış işleri yapar.

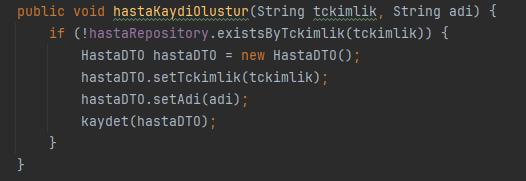


**Şekil 3.34.** Servis örneği



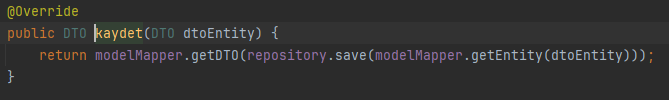
**Şekil 3.35.** Base servis

Örnek olarak “hastaKaydiOlustur” adındaki iki adet String parametre alan bu servis metodu, hastaların kaydını oluşturmak için geliştirilmiştir. Bu metot çalıştırıldığında veri tabanına kaydı yapılacak olan veriyi hazırlayarak kayıt kısmına gönderir. Öncelikle her tckimlik üzerine sadece bir hesap açılacağı için tckimlik ile daha önce kayıtlı kullanıcı olup olmadığına bakılır. Eğer kayıt bulunamaz ise hasta veri taşıma nesnesi olan “hastaDTO” sınıfı nesneleştirilerek bir hastaDTO elde edilir. Setter metotlarını kullanarak bu dto’nun bilgileri set edilir. Base serviste bulunan kaydet metoduna parametre olarak gönderilir.



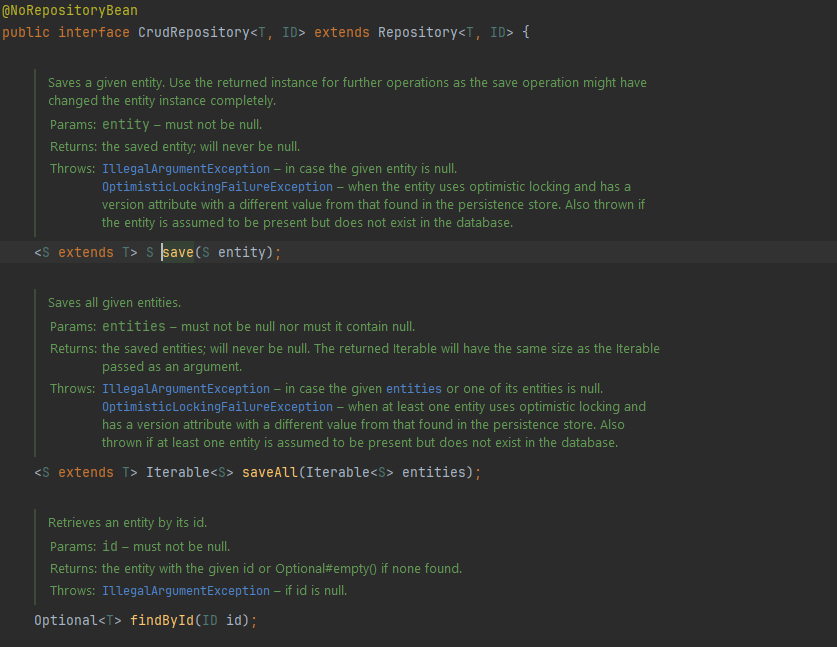
**Şekil 3.36.** Servis metot örneği

@Overrşde anotasyonu ile kaydet metodu, miras alınan metotlar da dahil olmak üzere ilk çalıştırılacak metot olarak ayarlanır. Parametre olarak alınan DTO nesnesi, kayıt için ModelMApper ile Entity’ye çevirilerek reprository’deki save metoduna gönderilir. Geri dönen nesne, tekrar ModelMApper ile dto’ya çevirilerek servise gönderilir.

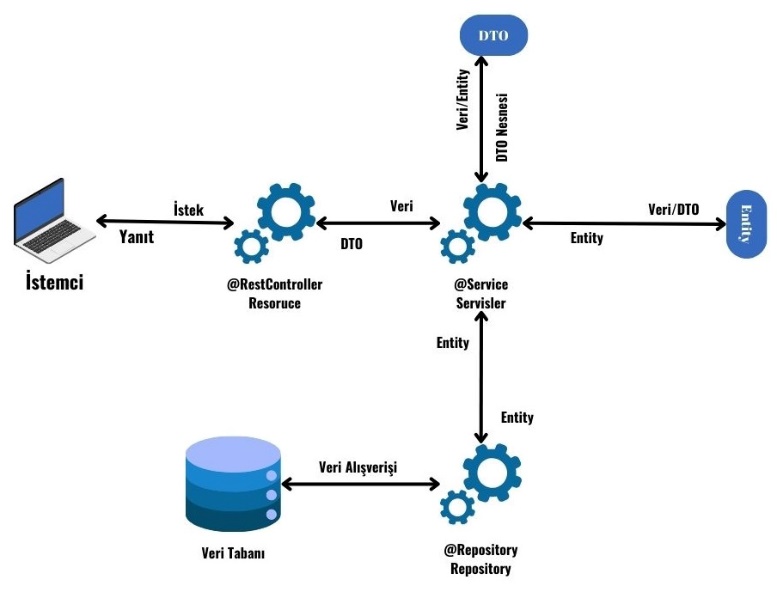


**Şekil 3.37.** Base servis metot örneği

Bu reprository Springboot içerisinde bulunan JPA reprository’dir. Hazırlanan diğer reprository’ler JPA reprository’den miras alır. Save, updateById, findAll, findById gibi crud işlemlerini barındıran bu reprository entityler ile hazırlanmış veri tabanı modelleri ile etkileşime geçerek bu verileri veri tabanına kaydedilebilir. Böylece nonSQL bir kod yazmak mümkündür. Gerekli veri tabanı yaralaması java sınıfları olarak tanımlanırsa bu yöntem ile JPA reprository kullanılarak SQL yazmadan crud işlemlerini gerçekleştirilebilir.



**Şekil 3.38.** JPA Repository örneği



**Şekil 3.39.** Geliştirilen arkayüz’ün çalışma sisteminin anlatımı

Java Springboot ile geliştirilen mikroservis yapısının bir isteği yanıtlaması farklı yöntemler ile sağlanabilir. Bu projede uygulanan yöntem, maddeler halinde aşağıda sunulmuştur.

* Resource: İstemci tarafından adrese gönderilen isteği yakalar. Kendisine tanımlanan adres yolları ile karşılaştırarak istek adresini ve metodunu kontrol eder. Eğer kontroller başarılı ise tanımlı sınıf metodunu çalıştırır. Bu metot, isteğe göre yazılmış servisi çağırır ve verileri servise gönderir. Servisten gelen verileri ise tekrar istemciye gönderir.
* Servis: Resource tarafından çağrılan servis, kendisine gelen parametreler ile ne işlem yapılacağını belirler. Yapılan işleme göre DTO ve Entity dönüşümlerini sağlar. İsteğin amacına göre gerekli repository modülüne verileri gönderir. Repositoryden gelen yanıtı gerekli veri yapısına dönüştürerek resource’ye gönderir.
* Repository: SQL sorgularının gerçekleştirildiği yerdir. Veri tabanı ile bağlantıyı kurar ve servisten gelen veriyi kullanarak veri tabanında gerekli işlemleri gerçekleştirir. Sorgu sonucunu servise gönderir.

4. SONUÇ

Proje tasarım aşamasında iken, proje sonuçlandırıldıktan sonra oluşabilecek sorunlar tespit edilerek bu sorunlara karşı tedbirler alınmıştır. Veri güvenliğinin sağlanması için JWT Baerer Token kullanılmış olup sisteme girişlerde yetki ve kimlik kontrolü sağlanmıştır. Yetkisiz olan kimselere kişisel bilgiler açılmamış ve yine veri güvenliğini sağlamak amacı ile şifre işlemlerinde şifreleme yöntemleri kullanılmıştır. Uygulamanın hızlı ve dinamik bir yapıya sahip olması için güncel teknolojiler kullanılmış olup mikroservis yapısından faydalanılmıştır. Kullanıcıların uygulama işlevlerini kolayca kullanabilmesi için sade ve kullanışlı arayüzler oluşturularak oluşabilecek karmaşıklığın önüne geçilmiştir.

Proje bu hedefler gerçekleştirilerek başarıyla sonuçlandırılmıştır.

5. KAYNAKLAR

1. ÇAĞDAŞ E. , BÜYÜKSARAÇ A. , IŞIK M.F. , 2022 , Early Intervention To Risk Groups With The QR Code System In Disasters.

2. Gionata, C., Francesco, F., Alessandro, F., Sabrina, I., Andrea, M. (2014). An Inertial And QR Code Landmarks-Based Navigation System For Impaired Wheelchair Users. In: Longhi, S., Siciliano, P., Germani, M., Monteriù, A. (Eds) Ambient Assisted Living. Springer, Cham. [Https://Doi.Org/10.1007/978- 3-319-01119-6\_21](https://doi.org/10.1007/978-%203-319-01119-6_21)

3. Al-Khalifa, H. S. (2008, July). Utilizing QR Code And Mobile Phones For Blinds And Visually İmpaired People. In International Conference On Computers For Handicapped Persons (Pp. 1065-1069). Springer, Berlin, Heidelberg.

4. ÇAĞDAŞ E. , BÜYÜKSARAÇ A. , IŞIK M.F. , 2022 , Early Intervention To Risk Groups With The QR Code System In Disasters.

5. SANAL, Aysel; ÖZTÜRKOĞLU, Yücel. HİZMET SEKTÖRÜNDE QR KOD KULLANIM ALANLARINA YÖNELİK BİR ALAN ÇALIŞMASI. Business & Management Studies: An International Journal, 2017, 5.4: 172-189.

6. Vivek, K., Nikhitha, N. R., Shaik, R. S., Anjali, K., & Saraswathi, M. (2022). Developing And Reading QR Codes. Journal Of Engineering Sciences, 13(12).

7. SWATHI KRISHNA, P. S.; MEERA, K. Enhanced QR: Approaches For QR Code Embedding Techniques. 2020.

17.   Stonebraker, M.; Rowe, L. A. (May 1986). [*The Design Of POSTGRES*](http://db.cs.berkeley.edu/papers/ERL-M85-95.pdf) (PDF). Proc. 1986 ACM [SIGMOD](https://tr.wikipedia.org/w/index.php?title=SIGMOD&action=edit&redlink=1) Conference On Management Of Data. Washington, DC. 6 Aralık 2019 Tarihinde Kaynağından [Arşivlendi](https://web.archive.org/web/20191206194718/http:/db.cs.berkeley.edu/papers/ERL-M85-95.pdf) (PDF). Erişim Tarihi: 6 Mart 2021

18. [Project Name – Statement From The Core Team"](https://web.archive.org/web/20071118121135/http:/archives.postgresql.org/pgsql-advocacy/2007-11/msg00109.php). Archives.Postgresql.Org. 16 Kasım 2007. 18 Kasım 2007 Tarihinde [Kaynağından](http://archives.postgresql.org/pgsql-advocacy/2007-11/msg00109.php) Arşivlendi.

19. Fielding, R.; Gettys, J.; Mogul, J.; Frystyk, H.; Masinter, L.; Leach, P.; Berners-Lee, T. (Haziran 1996). "Hypertext Transfer Protocol -- HTTP/1.1" (İngilizce): RFC2616. Doi:10.17487/Rfc2616. 9 Ağustos 2020 Tarihinde Kaynağından Arşivlendi.

**İnternet Siteleri:**

8. [Https://Www.Dogus.Edu.Tr/Ogrenci/Engelli-Ogrenci-Birimi/ Engelli-Tanimi#:~:Text=Bireyin%20ya%C5%9F%20ve%20geli %C5%9Fim%20d%C3%Bczeyine,Amerikan%20Psikiyatri%20Birli%C4%9Fi%2C%201994 (](Https://Www.Dogus.Edu.Tr/Ogrenci/Engelli-Ogrenci-Birimi/%20Engelli-Tanimi#:~:Text=Bireyin%20ya%C5%9F%20ve%20geli %C5%9Fim%20d%C3%Bczeyine,Amerikan%20Psikiyatri%20Birli%C4%9Fi%2C%201994 ()Erişim Tarihi: 27.12.2023)

9. <Https://Dent.Ege.Edu.Tr/Files/Dent/İcerik/Engelli.Pdf> (Erişim Tarihi: 27.12.2023)

10. <Https://Tr.Wikipedia.Org/Wiki/Engellili> (Erişim Tarihi: 27.12.2023)

11. <Https://Kilincgorkem.Wordpress.Com/Fizyolojik-Engeller-2/> (Erişim Tarihi: 27.12.2023)

12. <Https://Corumzubeydehanim.Meb.K12.Tr/İcerikler/Zihinsel-Engelliligin-Tanimi_7641819.Html> (Erişim Tarihi: 27.12.2023)

13. <Https://Www.Lodosozelegitim.Com/Bedensel-Engelli-Bireyler> (Erişim Tarihi: 27.12.2023)

14. <Https://Dent.Ege.Edu.Tr/Files/Dent/İcerik/Engelli.Pdf> (Erişim Tarihi: 27.12.2023)

15. <Https://Coderspace.İo/Sozluk/Apı>

16. <Https://Www.Linkedin.Com/Pulse/API-G%C3%Bcvenli%C4%9Fi-Temelleri-M%C3%Bccahit-İ%CC%87%C3%A7/?Originalsubdomain=Tr>

20. <Https://Medium.Com/Caglargul-Blog/Http-Protokol%C3%BC-Ve-I%CC%87ncelikleri-Ffd6ca3643f9>

21. [***Https://Www.Geeksforgeeks.Org/Spring-Repository-Annotation-With-Example/***](Https://Www.Geeksforgeeks.Org/Spring-Repository-Annotation-With-Example/)

22. https://tr.wikipedia.org/wiki/TypeScript

23. https://tr.wikipedia.org/wiki/HTTP

6. ÖZGEÇMİŞ

28.06.1999 yılında Kırşehir ilinde doğdum. İlk okulu Sırrı Kardeş İlk Öğretim okulunda ve orta okulu ise 30 Ağustos Zafer İlkokulu’nda tamamladım. 75. Yıl İ.M.K.B. Sağlık Meslek Lisesi’nde, hemşirelik bölümünde okudum. Şuanda ise Fırat Üniversitesinde Bilgisayar Mühendisliği bölümünde lisans eğitimi tahsil etmekteyim. Lisans eğitimime başladığım günden bu zamana kadar her zaman yeni teknolojiler öğrenmeye gayret ettim ve bu teknolojilerde el ile tutulur birer proje yapmak için çabaladım.

**İletişim:**

**GSM:** 05316960560

**E-posta:** ismail.yanik.1999@gmail.com