**Yöntem:** Geliştireceğimiz proje kullanımı kolay, sade ve anlaşılır olmalı. Bunun sebebi ise engelliliğin topluma kazandırılmasındaki en büyük etkenlerden birisinin eğitim seviyesi ve yoksulluk olduğunu söyleyebiliriz. Bilişimden ve teknolojiden uzak olan ailelerin faydalanabilmesi için, projede sitemize ilk girdiği andan itibaren kolay bir şekilde kullanabileceği, anlayabileceği, sade bir tasarıma sahip ve bu sadelik içerisinde işlevsel bir tasarım oluşturmamız gerekmektedir. Bunun için bahsedilen teknolojilerden frontend kısmında anguları kullanacağız. Angular içerisidne site yapısı için HTML, güzel ve kullanışlı bir tasarım oluşturmak için CSS ve Primeng kullanacağız. Angular ise sahip olduğu yapı sayesinde sitemizi işlevsel ve etkileşimli hale getirecek.

Ön yüze veri sağlayacak olan mikroservisi ise Java springboot ile geliştireceğiz. Bize sağlayacağı güvenlik imkanları ve veri iletişim hızı sayesinde acil vakalarda açık vermeden verilerin iletimini gerçekleştirebiliriz. Yine springboot ile geliştirilen api sayesinde tam entegre bir sistem hasline gelen projemiz geliştirmeye ve değiştirilmeye açık olacaktır.

Verileri depolayacağımız ve veriler üzerinde işlem yapacağımız veri tabanımız ise ücretsiz olarak bütün ihtiyaçlarımızı karşılayan postgresql olacak. Barındırdığı sql dili ile yapmak istediğimiz işlevleri hızlı ve güvenli bir şekilde yerine getirecektir.

projemiz api sayesinde iletişim kuracak olup belirtilen yollar dışında gelen her isteği red edecektir. Belirtilen yollar ile iletişim sağlamaya çalışan kullanıcıların ise yetkileri kontrol edilerek yetkisiz yapılan istekler yine red edilecektir. Geliştireceğimiz sitede 3 adet kullanıcı olacak bunlar:

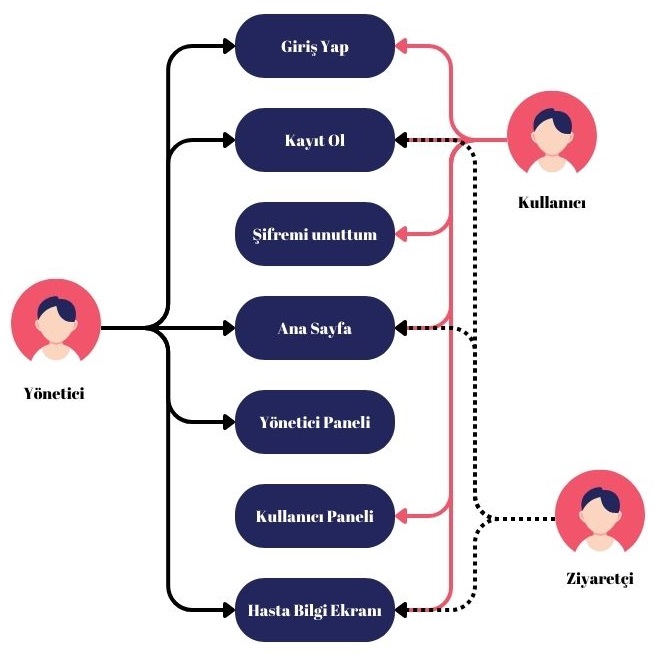
* Admin
* Engelli/Hasta
* Ziyaretçi

Her kulanıcı daha öncesinde yetkilendirilmiş bilgilere ulaşabilecektir. Kişisel verilerin korunması amacı ile kullannıcı giriş yapmadığı veriler umumi ortama açılmayacak ve açılan bilgiler ise yalnızca salt okunur yani değiştirilemez olacaktır. Her kullanıcının erişiebileceği servisler ve rolleri bellidir.

Admin: Yönetici yetkisi veri havuzumuza yeni hastalık veya engel ekleyebilir, veri havuzundaki eklenmiş olan engelleri veya hastalıkları güncelleyebilir veya silebilir. Hasta yada engelli vatandaşlarımızın eklediği hastalık veya engelleri kaldıramaz veya değiştiremez.

Engelli/Hasta yetkisi : Engelli veya hasta vatandaşlarımız kişisel bilgilerini ekleyebilir ve veri tabanındaki yönetici tarafından eklenmiş olan engelleri veya hastalıklardan sahip olduklarını kendisine tanımlayabilir. Diğer Engelli veya hastaların bilgilerini görüntüleyebilir ancak değiştiremez veya silemez. Veri tabanımıza yeni hastalık ekleyemez veya veri tabanında bulunan hastalık veya engelleri değiştiremez ya da silemez.

Ziyaretçi yetkisi : Ziyaretçiler qr kodu okutarak veya başka bir yöntem ile gönderdikleri url ile gönderilen hasta veya engelli idsine bağlı olan hastaların veya engellilerin verilerine ulaşabilir ve görüntüleyebilir. Engelli veya hastaların bilgilerini değiştiremez veya silemez. Veri tabanımıza yeni hastalık ekleyemez veya veri tabanında bulunan hastalık veya engelleri değiştiremez ya da silemez.

****

Bütün kullanıcılar sitemizin giriş yapma ve kayıt olma sayfasına erişebilir. Ziyaretçiler kayıtol sayfasından tc kimlik numarası, isim soy isim, telefon numarası girerek ve şifre oluşturarak sitemize kayıt olabilir. Girilen bilgiler kayıt edilmeden önce veri tabanından sorgulatılarak kontrol edilir. Eğer aynı tc numarasına sahip başka bir kişi var ise kayıt işlemi iptal edilerek kullanıcıya hata mesajı gösterilir. kayıt olan kullanıcılar Engelli/Hasta yetkisi kazanır.

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

Proje Geliştirme aşamaları: Projemiz 5 Aşamada gerçekleştirilecektir. Bunlar sıralı olmayıp birbirileri ile etkileşimde olacaktır. Aşamalar şu şekildedir:

* Veri tabanının tasarlanması ve geliştirilmesi
* Backend tarafının tasarlanması ve geliştirilmesi
* Frontend tarafının tasarlanması ve geliştirilmesi
* Test aşaması
* Hata giderme aşaması

***Veri Tabanının Tasarlanması ve geliştirilmesi:***

Veri tabanımız postgreSQL tarafından sağlacak olup verilerimiz sql dili ile ilenecektir. Veri tabanında tablolar oluşturularak ilişkileri kurulacaktır.

Hasta Tablosu : Her hasta veya engellinin bilgilerinin bulunduğu hasta adında bir tablo oluşturacağız. Siteye kayıt olurken ve kayıt olduktan sonraki Hasta veya engelli vatandaşlarımızın girdiği veriler bu tabloya yazılacak ve işlenecek. Kayıt olan her hasta veya engelli vatandaşa bir adet id verilecek. Veri çekme, kayıt etme, silme ve güncelleme işlemleri bu id ile yapılacak. Tabloda bulunan veriler tipleri ile aşağıda belirtirlmiştir.

* 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

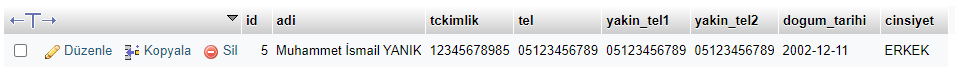
Hastalık tablosu : Yalnızca yönetici tarafından işlenebilen diğer kullanıcılara salt okunur olan hastalık veya engellerin bulunduğu tablodur. Yönetici hastalık ekleme, silme, güncelleme gibi işlemlerini bu tablo üzerinde gerçekleştirir. Eklenen her hastalık veya engele bir id verilir ve bütün işlemler bu id ile yapılır. Hastalık tablosunun barındırdığı stünlar veri itpleri 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

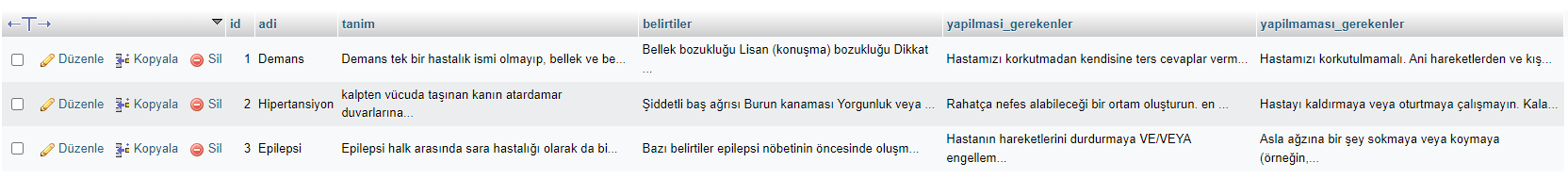
Hasta-hastalık tablosu: Hasta-hastalık tablosu Hasta veya engelli kullanıcıların sahip olduğu hastalıkları veya engelleri kendilerine tanımladığı tablodur. Bu tablo 3 adet veri barındırır. Bunlar aşağıda verilmiştir :

* 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

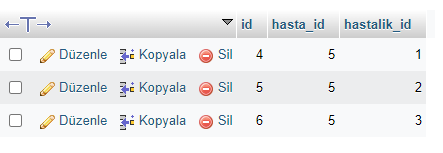
Hasta-hastalık tablosu hasta tablosu ile hastalık tablosunun id leri ile foregin key ilişkisi kurar. Her hasta veya engelli birden fazla hasta veya engele sahip olabilir bu neden ile hastalıkları hastaya tanımlamak için hastaların hastalıklarını tutan bir tabloya ihtiyacımız vardır. Burada bu gereksinimi hasta-hastalık tablosu karşılar. Aşağıdaki resimler ile bu durum örneklendirilmiştir.



HASTA KAYIT RESMİ



HASTALIK kayıt resmi



Hasta hastalık kayıt resmi

Resimdede görüldüğü gibi 4,5 ve 6 numaralı id ye sahip olan satırların hasta id veri kısmında 5 rakamı yazmakta bu 5 rakamı 5 numaralı id ye sahip olan hastamızı belirtmekte. 4,5 ve 6 rakamlı id ye sahip verilerin hastalık kısımları ise farklı hastalık id lerine sahip. Anlaşılacağı üzere 5 numaralı id ye sahip olan hastamız hastalık kısmında yazan idli hastalık veya engellerin tümüne sahiptir.

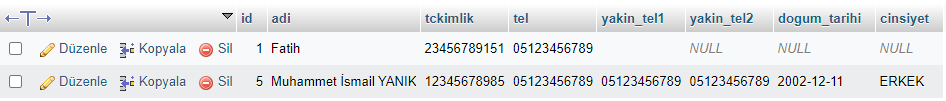
Kullanici Tablosu : Kullanıcı tablosu, uygulama kullanıcılarının bilgilerinin barındırıldığı tablodur. Yönetici ve hasta veye engelli kullanıcıların ortak olan bilgilerinin bulunduğu tablodur. Bu tablo genellikler şifre işlemleri ve elektronik posta işlemlerinde kullanılır. Güvenlik amaçlı tek tabloda tutulmayıp kullanıcı bilgileri iki tablo halinde ayrılmıştır. Böylece hasta veya engelli kullanıcıların hastalık bilgileri sorgulanır iken şifre ve mail bilgilerine erişilecek olan yollar ayrılmıştır. Bu yöntem ile siber saldırı tekniklerinden birisi olan SQL Injection tekniğine önlem alınmıştır. Kullanici tablosu veri 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

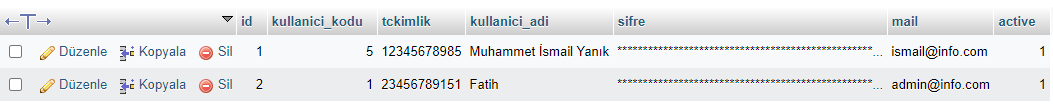
Kullanıcı tablosunun kullanici\_id sütunu ile kullaniciRole Tablosunun kullanici\_id sütunu birbirine yabancı anahtar ile bağlanmıştır. KullaniciRole tablosunda kullanıcılarımızın yetkilendirmeleri bulunmaktadır. Böylece bu tablo ile giriş esnasında yetki ve kimlik kontrolü yapabilmekteyiz.

KullaniciRole Tablosu : KullaniciRole tablosu, Kullanıcılarımızın yetkilerini barındıran tablodur. Bu tabloda iki adet veri bulunmakta. Birisi kullanıcı eşsiz kimliği bir diğeri ise yetki tipi. Böylece kullanıcılara yetkilendirme yapmış oluyoruz. Giriş esnasında yetkilendirme kontrolü bu veri tablosundan çekilen veri ile sağlanıyor. 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

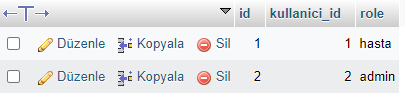


Admin kayıt resmi

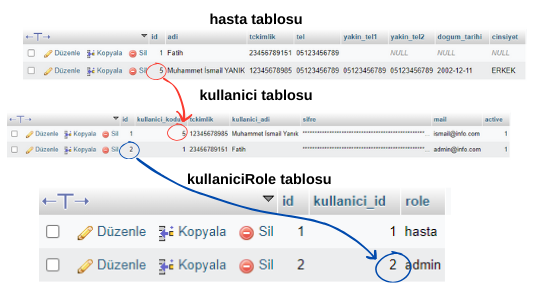
Resimde görüldüğü üzere fatih adlı kullanıcı hasta tablosunda 1 numaralı eşsiz kimlik ile kayıtlı.

Admin kullanıcı kaydı resmi

Resimde görüldğü üzere kullanici tablosunda 2 numaralı eşsiz kimliğe sahip olan satırın kullanici\_kodu sütunu 1 verisine sahip. Bu sütun hasta adlı tablonun id si ile yabancı key kullanrak ilişkilendirilmişti. Bu bağıntıdan kullanici adlı tablonun 2 numaralı eşsiz kimlik verisinin hasta adlı tablodaki 1 numaralı eşsiz kimliğe sahip kişi olduğunu anlıyoruz.



Kullanıcı rol resmi

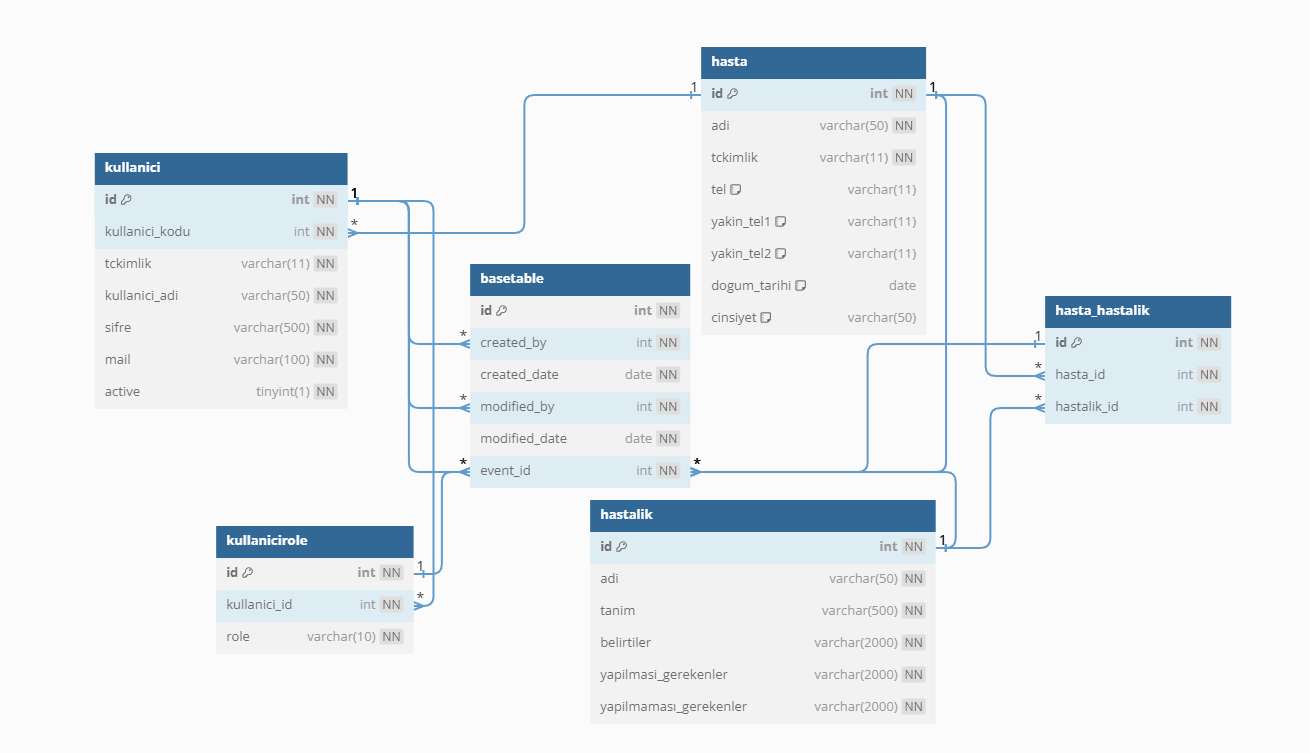
Yukarıdaki resimde görüldüğü üzere, kullaniciRole tablosunun id sütunu 2 olan satırın kullanici\_id sütununda 2 verisi bulunmakta. Ve aynı satırın role sütununda admin verisi bulunmakta. Bu sütunda kullanici\_id sütunundaki kullanıcıların yetkilerini belirtmekteyiz. 

Yukarıdaki resimde anlatılanlar şekil ile gösterilmiştir.

BaseTable tablosu : BaseTable tablosu, veri tabanında yapılan değişiklikleri tutarak bizlere yapılan değişiklikler hakkında bilgi verir. Bu tabloyu veri tabanının log tablosu olarak tanımlayabiliriz. Yapılan işlerin kimin tarafından ve nezaman yapıldığını değiştirilen verilerin kimin tarafından ve nezaman 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 kayıt edilmesi veya güncellenmesi projenin backend kısmı tarafından kontrol edilecek.

Veri tabanının oluşturulmuş tabloları ve ilişkilendirmeleri aşağıdaki şekilde gösterilmiştir. 

***VERİ TABANI İLİŞKİ GRAFİĞİ***

***Frontend tarafının tasarlanması ve geliştirilmesi :***

Projenin Arayüz tarfı Bahsedilen teknolojilerden olan angular ile geliştirildi.

Projede 7 adet sayfa bulunmakta. 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.

Anasayfa : Anasayfa sitenin tanımını yapan siteye ilk ulaştığımız yerdir. Bu sayfada projenin tanıtımı yapılmakta. Hakkımızda kısmında projenin ne işe yaradığı faydaları ve özellikleri bulunmakta. Hizmetimiz kısmında ise kullanıcılara sunduğumuz hizmet şablonlar ile açıklanmakta. Sık Sorulan Sorular kısmında ise kullanıcıların merak ettiği ve sıkça sorulan sorular cevapları ile birlikte yer almakta. İletişim kısmında kullanıcıların geliştiricilere e posta göndererk ulaşabileceği bir alan bulunmakta. Bunun dışında footer kısmında iletişim bilgileri ve sayfa haritası bulunmakta.

Kayıt ol sayfası: Kayıt ol sayfası Ad , soyad, e posta, ve şifre bilgilerinizi girerek yeni bir hesap açabileceğiniz bir sayfadır. Buradan kayıt olarak hasta/engelli profili oluşturulur.

Giriş yap sayfası : giriş yap sayfası Kayıtlı kullanıcıların eposta ve şifreleri ile kullanıcı panellerine giriş yapacakları sayfadır. Bu sayfa bilgilerinizi kontrol ettikten sonra sizi yetkilenirerek kullanıcı panelinize yönlendirir.

Şifre yenile sayfası : Şişfre yenile sayfası email adresi ve telefon numaranızı alarak sistemden dığruluklarını kontrol eder. Eğer bilgiler doğrulanır ise sistem kayıtlı e postanıza güvenlikli yeni bir şifre gönderir.

Kullanıcı paneli : Kullanıcı paneli Kullanıcıların bilgilerinin görünütlendiği değiştirilebildiği ve silinebildiği sayfadır. Bu sayfa sayesinde bilgilerinizi ekleyebilir değiştirebilir ve silebilirsiniz. Qr kod oluşturucusu ile qr kodunuzu indirebilirsiniz. Şifre değiştir kısmından eski şifrenizi doğrulayarak şifrenizi yenisi ile değiştirebilirsiniz. (MADDELER HALİNDE YAZILABLİİR). Tanımlı hastalıklar kısmında tanımlı hastalıklarını görüntüleyebilir ve silebilirsiniz. Hastalıklar tablosundan ise kendinize hastalık tanımlayabilirsiniz. Hastalık tablosundan tanımlı hastalıklar tablosuna tanımladığımız hastalıklar artık hastalıklar tablosunda görüntülenmeyeceğinden bir hastalık birden fazla kez tanımlanamaz.

Engelli/ hasta bilgi sayfası : Engelli/hasta bilgi sayfası ile id si url de gönderilen hastanın kayıtlı bilgilerine ve üzerindeki tanımlı hastalıklara ulaşabilirsiniz. Tek bir tıklama ile 112 acil çağrı butonuna tıklayarak acil çağrı merkezini arayabilirsiniz.

Yönetici paneli: Yönetici paneli Yöneticinin hastalık ekleyip silebileceği ve güncelleme yapabileceği paneldir.

Angular projelerinde api servislerinden veri çekme işleminde projeyi parçalara ayırmamız gerekir. Bu parçalar genel olarak model, view ve service olarak adlandırılır. Her bir parça farklı bir amaca hizmet eder. Böylece tekrar kullanılabilir, geliştirmeye açık bir proje ve düzenli kod yazımı elde ederiz.

Model : Angularda 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 formatı ile değil java nesneleri ile işleyebiliriz.

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 neden ile 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 halidir. Veri alış verişi 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.

 ***MODEL RESMİ***

Yukarıdaki resimde görüldüğü üzere veri tabanındaki tablo dan gelen verilerin java nesnesine çerliebilmesi için isimlerinin aynı olması ve veri tiplerinin uyumlu olması gerekmektedir.

Uygulama sayfalarında gösterilmek üzere hastalık bilgileri için istek yolladığımız zaman bize dönecek olan hastalık verileri için HastalikDTO adında bir model tanımlıyoruz.

Servisler : Servisler api iletişiminde sunuculara istek gönderdiğimiz katmandır. Hangi sunucuya ne istek gönderileceğini bu katmanda belirleriz. Gönderilen isteklerin sonuçları yine bu katmanda yakalanır ve işlenir.

Servis tanımlarken öncelikle gerekli importları yaparak başlıyoruz. Angular çekirdeğini içeri aktardıktan sonra sırası ile http protokolünü kullanmak için httpClient ve HttpErrorResponse moduüllerini içeri aktarıyoruz.

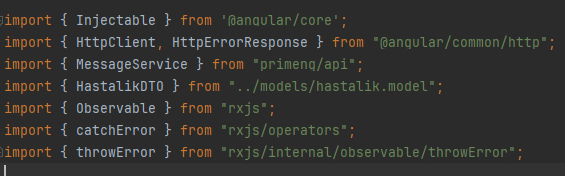
MessageService: Kullanıcıya bilgi mesajlarını göstermek için primeNG framewörkünün sahip olduğu MessageService modülünü içeri aktarıyoruz.

HastalikDTO: apiden gelen yanıt taki json formatındaki verileri java nesnesine dönüştürmek için HastalikDTO modelini yüklüyoruz.

Observable : observable mdülünü sunucu tarafına yaptığımız istekleri dinlemek ve değişikliklerdei yakalayabilmek için içeri aktarıyoruz.

catchError : sitek sonucu gelen hataları yakalamak için kullanılır.

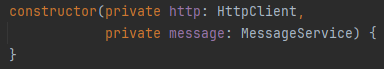
throwError : bazı hataları görmezden gelmek için kullanılır.



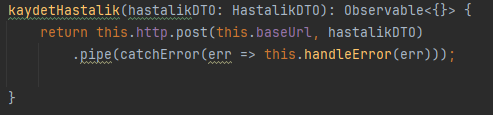
İMPORTLAR

baseUrl adında string tipinde bir değişken oluşturarak içerisine isteğin yapılacağı sunucu adresini yazıyoruz. 

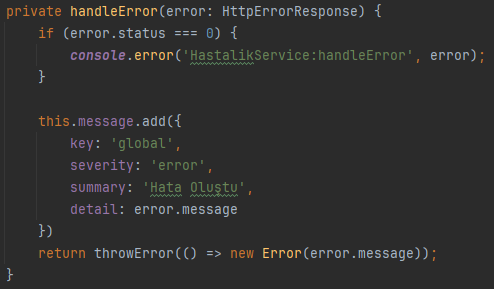
Servislerin bulunduğu sınıfın başlatıcısını http protokolünü kullanmak için HttpClient nesnesini parametre olarak gönderiyoruz ve başlatıcı içerisinde MessageService modülünü daha sonra kullanabilmek için nesneleştiriyoruz.



Örnek olarak oluşturduğumuz servislerden birisi olan hastalık kaydetmede kullanılan metodu gösterelim. Metot oluşturulurken metodun adını yazarak parametre olarak istekte bulunurken göndereceğimiz bilgilerin bulunduğu java nesnesini ekliyoruz. Observable ile metotumuzu dinamik dinleyici haline getiriyoruz. Return ile işlemin sonucu yakalayacağız ve çeğrılan yere göndereceğiz. This.http.post kodu ile bunun bir kayıt işlemi olduğunu ve http protokolü ile gönderileceğini belirttik. Post metodu birçok parametre içerir bu servis için iki parametre yeterli olacaktır. Öncelikle isteğin yapılacağı adresi belirtiyoruz ve ardından göndereceğimiz bilgilerin bulunduğu java nesnesini ekliyoruz. Geri dönüş verisini pipe ile dönüştürerek hata yakalama nesnesi için yazılmış handleError metoduna gönderiyoruz. Gelen veri ilk başta yazdığımız return ile metodun çağrıldığı yere geri dönecek.



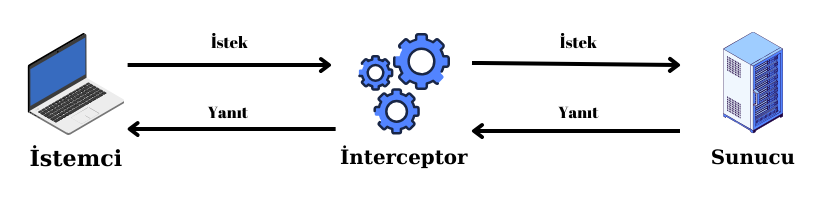
handleError metodukendisine gelen hata mesajını kontrol ederek başlatıcıda oluşturduğumuz message öğesi ile kullanıcı arayüzüne sunuyor.



Bir api 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 aşamadanda 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 dolayı anguların interceptor özelliğini kullanırız.

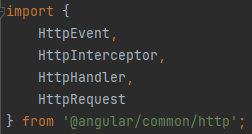
İnterceptor istemci ile sunucu arasında oluşan iletişimi kontrol eden bir yapıdır. İstemci ile sunucu iletişimin tam ortasında yer alır ve gelip giden bütün istek ve yanıtlar interceptor kontrolünde sağlanır. İnterceptor ile istemci ve sunucu arasındaki letişimde araya girerek gerekli kayıt, yönlendirme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



***INTERCEPTOR RESİM***

Yapılan isteklerde kimlik doğrulaması ve yetki kontrolü için göndereceğimiz header öğesini ınterceptor yapısına tanımlıyoruz böylece her seferinde göndermemiz gereken header öğelerini tek seferlik bir tanımlama ile her istek ile birlikte gönderebiliriz.

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

******

***INTERCEPTOR TANIMLAMASI***

Interceptor sınıfımızı oluşturmak için HttpInterceptor sınıfını implement ediyoruz. Böylece HttpInterceptor sınıfından yararlaabilir özelliklerini kullanabiliriz. Sınıfımızın başlatıcısı olan constructor metodunun içerisine interceptor metodumuzu tanımlıyoruz. Interceptor metodumuz http isteklerimizi yakalayacağı için parametre olarak http isteğini tanımlıyoruz. http olaylarını takip etmesi için observable özelliğini ekliyoruz. Header a token eklemek için sayfamıza erişim sağlayan kullanıcının sahip olduğu tokena ihtiyacımız var. Bunun için siteye erişim sağlayan kullanıcıların json formatında localstoreg’a depoladığımız bilgilerinin arasından tokenını almamız gerekiyor.

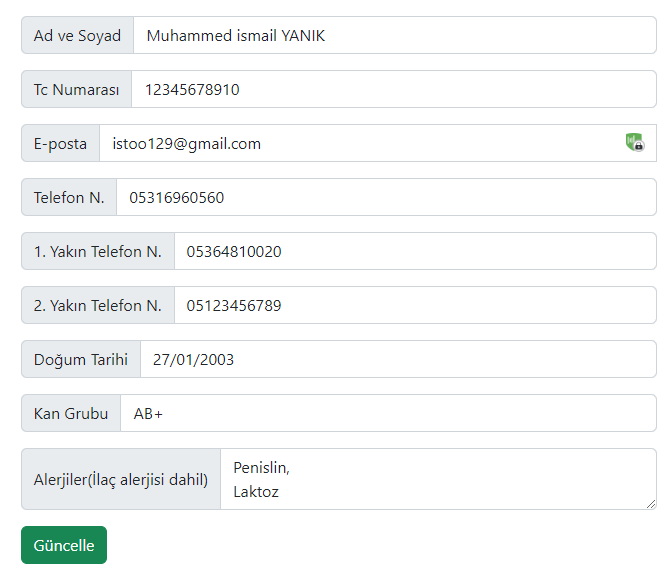
Currentuser adında bir değişken oluşturup içerisine tokenı kayıt edeceğiz. Tokenımızı localstorage.getItem metodunu kullanarak elde ediyoruz. Ve JSON.parse metodu ile elde ettiğimiz json formatındaki veriyi parçalayarak java nesnesi haline getiriyoruz. Eğer current user dolu ise isteğimizi clone metodu ile klonlayarak setHeaders ile header öğesini değiştiriyoruz. İsteğin headers öğesindeki Authorization tarafına tokenımızı ekliyoruz.



View : View katmanı elde ettiğimiz verilerin görüntüleme işleminin yapıldığı katmandır. View katmanında görüntülemek istediğimiz veriler için html css gibi teknolojileri kullanarak sayfalar oluştururz ve model ile gelen verileri anguların data binding özelliği ile gösterilmesini istediğimiz kısımlarda kullanıcıya sunabiliriz. Html ile oluşturulmuş sayfalar angular yardımı ile etkileşimli hale gelir. Sunucuya yapılan istekler ve dönen yanıtlar yine angular tarafından yönetilir.

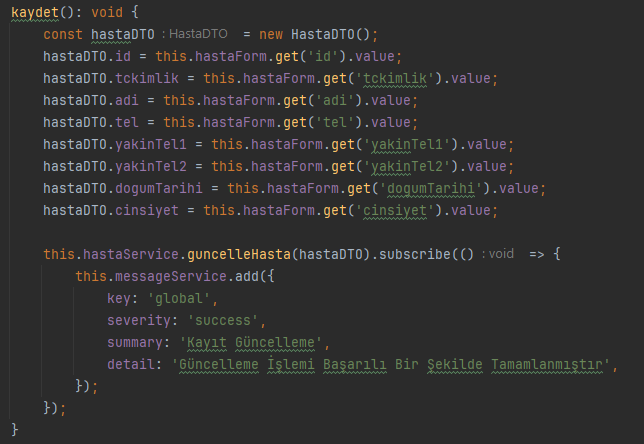
Projemizin view kısmını oluştururken html kodları ile sayfamızın elementlerini oluşturutoruz. Css ve primeng yardımı ile sayfamızın görselliğini geliştirerek daha kullanışlı hale getirebiliriz.

Kullanıcı paneli sayfasında kullanıcı bilgilerimizi görüntülememiz, güncellememiz ve kayıt edebilmemiz gerekiyor bunun için sayfamıza bir adet form öğesi ve buton ekliyoruz.



Form öğesini primeNG ve html elementleri ile oluşturuyoruz. Anguların farkı ise nesne tabanlı programlamaya destek verdiği için isteği form öğesi değil angular kendisi yapacak. Bunun için güncelle butonunun tipini submit yapıyoruz ve kaydet metodunu bağlıyoruz.



Kaydet metoduna yeni bir hastaDTO nesnesi oluşturuyoruz. Angular data bindin ile formdaki bilgileri alarak oluşturduğumuz hastaDTO nesnesine aktarıyoruz. Sunucuya istek yapmak için oluşturduğumuz observable özelliği olan metodumuzu burada çağırarak tetikleyeceğiz. Bunun için hastaservis adı ile nesneleştirdiğimiz servis sınıfının içerisindeki guncellehasta metoduna formdan bilgileri çekerek oluşturduğumuz hastaDTO nesnemizi parametre olarak verip observable özelliğini aktif etmek için subscribe metodu ile primeNG frameworkünün sağladığı bildirim mesajı özelliğini kullanrak çağırıyoruz.

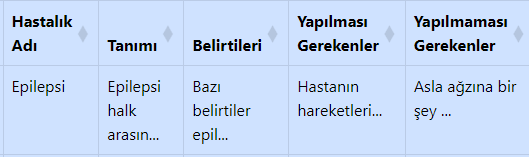
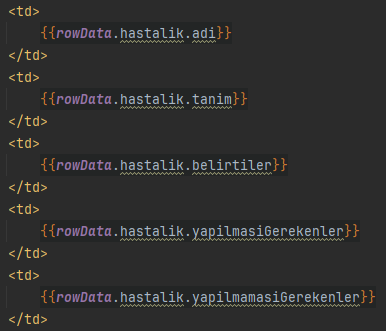
Kullanıcının qr kodunu oluşturması ve bilgi sayfasını görebilmesi için qr kod oluşturma aracı gerekli. Bunun için ise angular açık kaynak kütüphanelerinden olan angularx-qrcode kütüphanesini npm ile yükleyerek qrcode tagları ile kullanabiliyoruz.



Asdasdasd

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 özelliğini kullanacağız. Anguların bu özelliği sayesinde tek yöblü veri bağlama yapar isek typescript modülünde oluşturulan bir veriyi html dosyalarında görüntüleyebiliriz. Eğer iki yönlü veri bağlama yapılırsa typescript değişkenlerini html dosyalarına taşıdığımız gibi html inputları ile alınan verileride anlık olarak typescript değişkenine atayabiliriz.

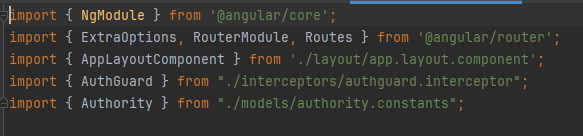
Tek yönlü veri bağlama işlemi html kodları arasına iki defa açılmış süslü parantez arasına yazılan değişken ile sağlanır. Böylece süslü parantezlerin açıldığı yere typescript değişkeninin değeri yazılmış olur.



Tek yönlü data binding örneği

**Angular Yönlendirmelerinin oluşturulması :**

Projemizde her sayfa için farklı modüller kullandık ve bu modülleri html ve css sayfalarına bağladık. Şimdi ise bu modüllerin adreslere yönlendirilmesi gereklidir. Angular ile yönlendirmeyi projemizde app-routing.ts adlı dosyamızda gerçekleştiriyoruz. Böylece sitedeki trafiği kontrol ederek gerekli adreslerin yakaladığı anda doğru yönlendirmeler ile modüllerimizi yükleyeceğiz. Yönlendirme yapmak ve trafiği kontrol etmek için routing modüllerini içeri aktarılmalıdır. Bunun için RouterModule, Routes modüllerini içeri aktarılmalı. Yetki kontrolü için ise AuthGuard ve Authority modülleri içeri aktarılmalı.



Rout için gerekli içeri aktarımlar

Const tipinde routes adında bir değişken oluştururarak gerkli yönlendirmeleri içerisine tanımlayacağız. Kullancağımız metother yönlendirmeyi modül paketine yapıp modül paketinde yeni yönlendirmeye tabi tutmak olacak. AuthGuard modülünü aktif hale getirerek yönetici ve kullanıcı panellerinin modüllerini yetkiye tabi tutuluyor. Kullanıcı giriş yaptığında giriş bilgileri konol edilerek gerekli yetkilendirme sunucu tarafından verilecek ardından verilen yetkilendirme kontrol edilerek yetkisinin giriş yapabileceği modüller yüklenecek. Eğer kullanıcı sunucudan ADMIN yetkisi almış ise views klasörünün içerisinde bulunan tanimlar klasöründe bulunan tanimlar.module modülü yüklenecek. Eğer kullanıcı USER yetkisi almış ise views kalsöründe bulunan işlemler klasöründeki işlemler.module modülü yüklenecek.

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



Yönlendirme örneği

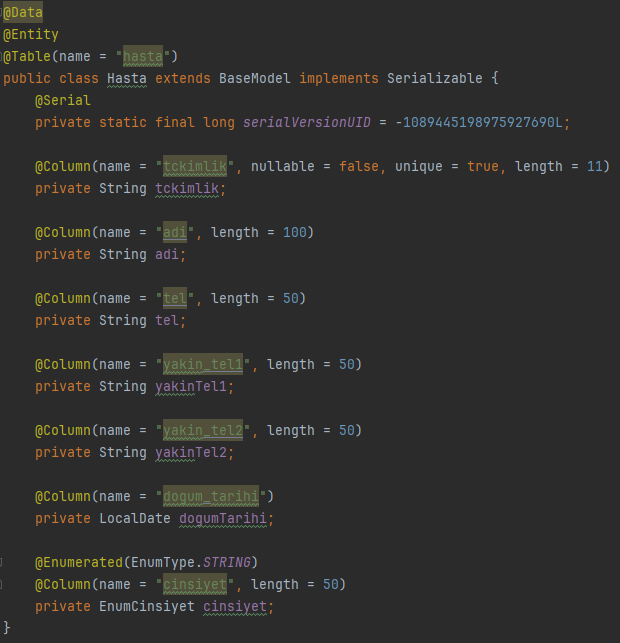
PROJENİN BACKEND TARAFININ Tasarlanması

Prjemizde backend kısmı bilgilerimizin tutulduğu, istemci tarafından yapılan istekleri yakalayarak bu istek doğrultusunda yanıt hazırlayıp hazırlanan yanıtı doğru formatta tekrar istemciye gönderecek olan kısımdır. Amacımız mikroservis mimarisi ile entegre olabilecek bir api oluşturup gelen istekleri yanıtlayabilmek.

Api servisimiz oluşturmak için hazırlamış olduğumuz veri tabanını en etkili şekilde kullanmak için java dilinde modellememiz gereklidir. Böylece veri tabanımızı javanın işleyebileceği şekilde nesneleştirerek daha etkin kullanabiliriz.

Veri tabanımızdaki tablolarımızı java nesnesi olarak tanımlama işleminde yapılması gereken tabloları sınıf olarak, sütunları ise @Column anotasyonu ile değişken olarak tanımlamaktır. @Column anotasyonundan sonra parantez içerisinde sütun ismini ve sütunun diğer kuralları tanımlanmalıdır. Bu kuralların birebir aynı olması karşılaşacağımız hataların önüne geçer. Projemizde her tabloyu bahsedilen şekilde java nesnesi olarak tanımlıyoruz.

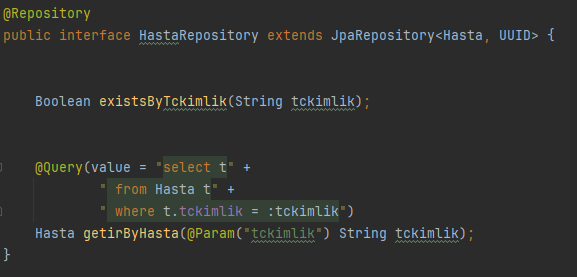
Eklediğimiz @Data anotasyonu java nesnelerinde güvenli veri alışverişi için kullandığımız get ve set metotları, hashcoding gibi kod fazlalığı oluşturacak metotları eklemeden sanki projemizde varmış gibi derleme zamanında ekleyerek iş yükünü azaltır. Bu sayede @Data anotasyonu ile bu metotları kullanabiliriz.



Hasta tablosu java nesnesine dönüşüm örnek

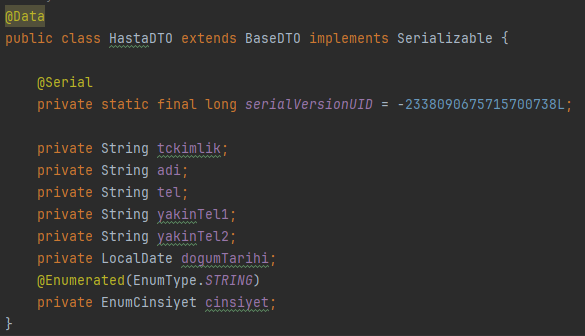
Nesneleştirilen tablo yardımı ile yapılacak sorgular java arayüzü olarak yazılır ve isteğin doğrultusunda servis bu arayüze ulaşarak istenilen sorguyu gerçekleştirebilir.

Sorgularımızı yazacağımız arayüze @Repository anotasyonu verilir. @Repository Annotation , sınıfın nesneler üzerinde depolama, alma, güncelleme, silme ve arama işlemi için mekanizma sağladığını belirtmek için kullanılır([***https://www.geeksforgeeks.org/spring-repository-annotation-with-example/***](https://www.geeksforgeeks.org/spring-repository-annotation-with-example/)***)***. Sorgularımızı @Query anotasyonu ile HQL dilinde yazılabilir.



Örnek sorgu

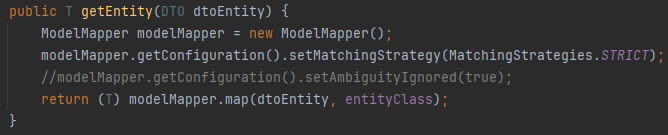
Springboot ile geliştirilen projelerde de angularda tercih edildiği gibi DTO yapıları çokça tercih edilmektedir. Gerek bilgi güvenliği gerek ise kaynak tasarrufu açısından bir çok faydası bulunur. Bazı sorgularda tablodaki bütün bilgilere ihtiyacımız olmadığı yerde entity ile değil DTO ile veri akışı 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 tasarlanmış. Bunun seberi varsayılan model mapper kurallarına uyarak varsayılan model mappırı projede kullanmaktır. Böylece model mapperda herhangi bir değişiklik yapmadan DTO ları Entitylere veya Entityleri DTO lara dönüştürebiliriz.



DTO ornek

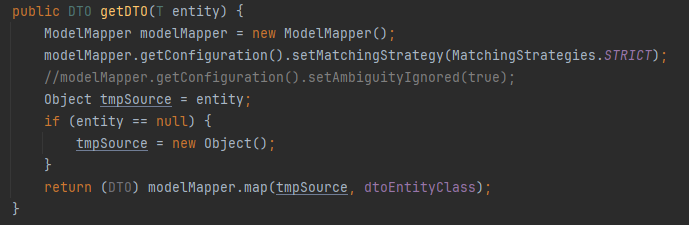
Springboot un sahip olduğu ModelMApper kütüphanesi ile Entityler ve dto lar arası dönüşüm sağlanır. Geliştiriciler model mapper özelliğini amaçlarına uygun şekilde geliştirerek sahip oldukları sınıfları birbirielrine dönüştürebilirler.

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 nesnesini entityedönüştürülebilir. Elde edilen entity nesnesi geri dönderilir.



Get entit metodu

Entity nesnelerini DTO nesnelerine dönüştüemke için getDTO adında DTO geri dönüş tipine sahip ve parametre olarak Entity nesnesi alan bir metot oluşturulur. Yine ModelMApper sınıfını nesneleştirilir. Bu nesne ile entity yi dtoya çevirme işlemi yapılır.

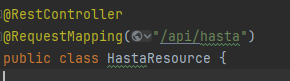


Getdto örnek

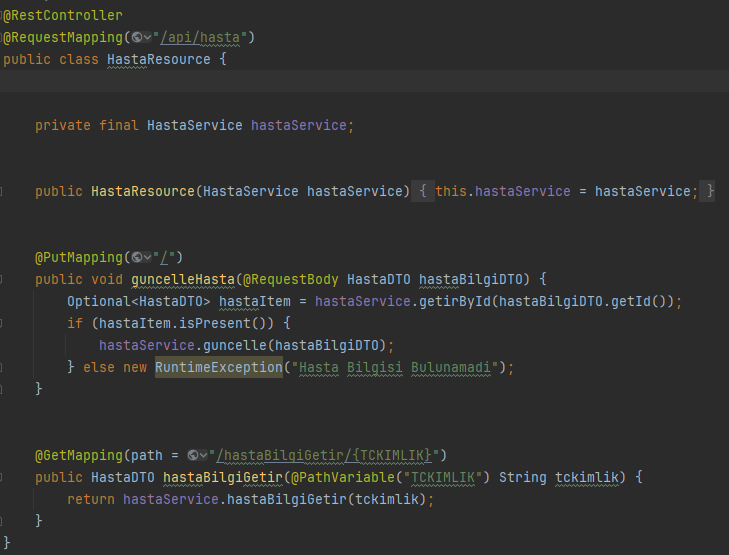
İstekte bulunan istemcinin isteğindeki yetkileri ve hasta bilgilerinin taşınmasındaki ve işlenmesindeki kolaylığı sağlamak için cinsiyet ve rol tanımlamarını sabit değerlere bağlamak için enum kullanılmıştır. EnumRoles sınıfımız admin ve user olan iki adetenum içerir. Enum Cinsiyet ise Erkek ve kadın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

APİ GELİŞTİRİLMESİ

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

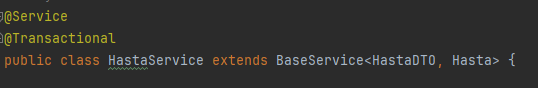


Böylece tanımlanan adrese gelen istekler bu sınıf tarafından karşılanacak ve işlenecek. Doğru adres ile doğru Mapping kullanılarak yapıan isteklerin metotları tetikleyebilmesi için metotlara @PutMapping anotasyonu verilerek kök url den sonra gelenlerin hangi metotları tetikleyeceği belirlenir. Bu metotlar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

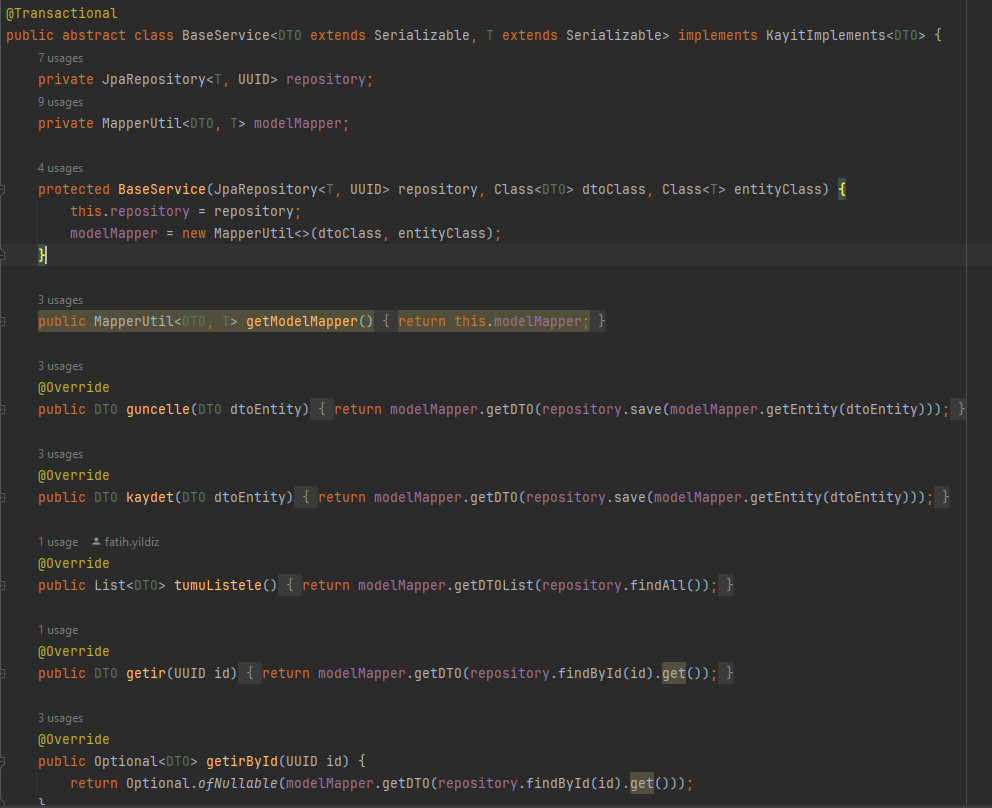


Hasta resource örnek

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

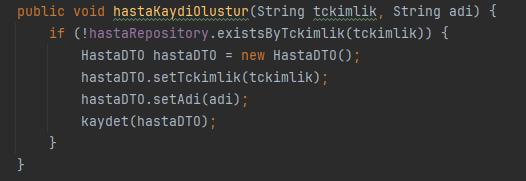


Service tanımlaası örneği



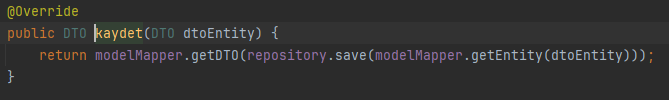
Base service

Örnek olarak hastaKaydiOlustur adındaki iki adet String parametre alan bu servis metodumuz hastaların kaydını oluşturmak için geliştirilmiştir. Bu metot çalıştırıldığında veri tabanına kayıdı yapılacak olan veriyi hazırlayarak kayıt kısmına gönderir. Öncelikle her tckimlik üzerine sadece bir hesap açılacağı için tckimlik ile daha önce kayıtlı kullanıcı olup olmadığına bakıyoruz. Eğer kayıt bulunamaz ise hasta veri taşıma nesnemiz olan hastaDTO sınıfımızı nesnleştirerek bir hastaDTO elde ediyoruz. Setter metotlarını kullanrak bu dtonun bilgilerini set ediyoruz. Base serviste kulunan kaydet metotduna parametre olarakgönderiyoruz.



Service metot örneği

@Overrşde anotasyonu ile kaydet metodunu miras alınan mtotlarda dahil olmak üzere ilk çalıştılacak metot olarak ayarlıyoruz. Parametre olarak alınan DTO nesnesini kayıt için ModelMApper ile Entity ye çevirerek reprositorydeki save metotduna gönderiyor. Geri dönen nesneyi tekrar ModelMApper ile dto ya çevirerek servise gönderiyor.



Bu reprository Springboot içerisinde bulunan JPA reprositorydir. Hazırlanan diğer reprositoryler JPA reprositoryden miras alır. Save, updateById, findAll, findById gibi crud işlemlerini barındıran bu reprository entityler ile hazırlanmış veritabanı modelleri ile etkileşime geçerek bu verileri veri tabanımaıza kayıt edebilirler. Böylece nonSQL bir kod yazmak mümkündür. Gerekli veri tabanı yaralaması java sınıfları olarak tanımlanırsa bu yöntem ile JPA reprository kullanılarak SQL yazmadan crud işlmelerini gerçekleştirebiliriz.

