

T.C. SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

WEB PROGRAMLAMA PROJE ÖDEVİ

HASTANE RANDEVU SİSTEMİ WEB SİTESİ

ABDULKADİR EREN YURTASLAN-G201210081-
1.ÖĞRETİM/A GRUBU

NECATİ BABACAN-G201210064-1.ÖĞRETİM/A
GRUBU

GITHUB LİNKİ: <https://github.com/ErenYurtaslan/HasteneRandevuSistemi>

Hastane Randevu Sistemi Web Programlama Ödevi Girizgahı

Admin girişı, hasta girişı ve girişsiz olarak 3 parça olan sitemizde; sunucu tarafında doktorların branşları, isimleri, fotoğrafları, çalışma gün ve saatleri, ekleme-güncelleme-silme işlemleri(CRUD), tüm bu bilgilerin çekildiği SQLServer veri tabanı bulunmakla beraber, hem hastaların hem de adminin randevu kaydı oluşturabileceği ve bunların veri tabanında aynı tabloya kaydedileceği bir algoritma düzenlenmiştir. Proje kodlarımızın, S.O.L.I.D. prensiplerine uygun olarak yazılmasına gayret edilmiştir.(esnek, clean code'a uygun, arabirimli, gereksiz koddan ve sınıftan kaçınma) Login yapılmadığı takdirde sadece "Anasayfa" ve "Bize Ulaşın" butonları aktif olurken, hasta olarak login yapıldığında bu ikisine ek olarak "Doktorlar" butonu da gelmektedir. Bu 3. butonun içeriğinde hasta, doktorun kendisini ilgilendiren verilerini görmekle beraber sadece randevu alma işlemi yapabilmektedir. Admin ise yukarıda belirtildiği gibi tüm verilere erişebilir ve değiştirebilir pozisyonodadır. Asp.Net Core 7.0 sürümüne uyumlu olarak yabancı dil dosyaları eklenmiş olup, hem Türkçe hem de İngilizce ayarı sağlanmıştır.

PROJE DETAYLARI

Projemizde 4 adet controller(DoktorBrans, Doktor, Home, Randevu), 13 adet model sınıfı, 4 DoktorBrans view'i, 3 Doktor view'i, 2 Home view'i ve 4 adet Randevu view'i, Kimlik'in olduğu Areas dosyası, db context sınıfının ve kullanıcı rollerinin olduğu Utility dosyası, çoklu dil entegrasyonu için de Resources, Languages ve Services dosyaları bulunmaktadır. Veri tabanı yönetim sistemi olarak Microsoft SQLServer kullanılmıştır. Projenin veri tabanına bağlantısı, Utility dosyası içindeki UygulamaDbContext öğeleri ve Models'in içindeki interfacerler ve onların implement edildiği Linq Repository sınıflarıyla sağlanmıştır.

Aşağıda appsettings.json dosyasıyla veri tabanı tablosunun adı belirlenmiştir:

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection": "Server=DESKTOP-EIEMKQ8\\SQLEXPRESS;Database=HastaneRandevuSistemi;Trusted_Connection=True;TrustServerCertificate=True"
  }
}
```

Sonrasında bu veri tabanı, db context aracılığıyla Program.cs'e tanımlandı:

```
builder.Services.AddDbContext<UygulamaDbContext>(options => options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));
```

Sonrasında migrationlar aracılığıyla, aşağıda tabloya dönüşecek verileri MSSQL'e tanımladık:

```
using HastaneRandevuSistemi.Models;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

//Veri Tabanında EF tablosu oluşturulması için ilgili model sınıflarımızı buraya ekliyoruz.
namespace HastaneRandevuSistemi.Utility
{
    //DbContext yerine IdentityDbContext sınıfı extend ettik ki admin paneli ve user paneli sistemini oluşturalım.
    public class UygulamaDbContext : IdentityDbContext
    {
        public UygulamaDbContext(DbContextOptions<UygulamaDbContext> options) : base(options)
        {
        }

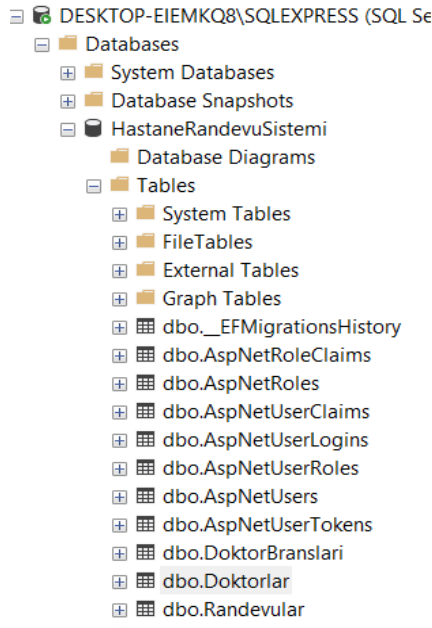
        public DbSet<DoktorBrans> DoktorBranslari { get; set; }

        public DbSet<Doktor> Doktorlar { get; set; }

        public DbSet<Randevu> Randevular { get; set; }

        public DbSet<ApplicationUser> ApplicationUsers { get; set; }
    }
}
```

Bunlara ek olarak tabi ki eklenen "Kimlik" özelliğiyle, aynı veri tabanına Areas dosyasındaki ek tablolar da eklendi, update-database komutu nuget konsoluna yazıldı ve tablo görünümü şöyle oldu:



Sonrasında bu veri tabanında elle yapılan değişiklikleri, front end olarak sitede görmek için controller ve viewlar kullanıldı. Controller içinde kullanılan fonksiyonlar, Models'teki interfacelerden temin edildi.

Örnek, Repository.cs ve IRepository.cs dosyaları:

```
namespace HastaneRandevuSistemi.Models
{
    [Serializable]
    public class Repository<T> : IRepository<T> where T : class
    {
        private readonly UygulamaDbContext _uygulamaDbContext;
        internal DbSet<T> dbSet;

        [Serializable]
        public Repository(UygulamaDbContext context)
        {
            _uygulamaDbContext = context;
            this.dbSet = _uygulamaDbContext.Set<T>();
            _uygulamaDbContext.Doktorlar.Include(k => k.DoktorBrans).Include(k => k.DoktorBransId);
        }

        [Serializable]
        public void Ekle(T entity)
        {
            dbSet.Add(entity);
        }

        [Serializable]
        public T Get(Expression
```

```

using System.Linq.Expressions;

namespace HastaneRandevuSistemi.Models
{
    4 başvuru
    public interface IRepository<T> where T : class
    {
        //T -> DoktorBrans

        8 başvuru
        IEnumerable<T> GetAll(string? includeProps = null);

        10 başvuru
        T Get(Expression<Func<T, bool>> filtre, string? includeProps = null);

        5 başvuru
        void Ekle(T entity);

        4 başvuru
        void Sil(T entity);

        1 başvuru
        void SilAralik(IEnumerable<T> entities);
    }
}

```

Burada kod tekrarından kaçınmak ve düzenli, esnek bir proje yapısına sahip olmak üzere interfacer kullanıldı, ayrıca Linq ile sorgu yapıldı. Her bir controllerın model sınıfı için(mesela RandevuRepository ve IRandevuRepository) bunları ayrı ayrı yazmak yerine, implement ve extendler kullanılarak daha yalın bir kod düzeni oluşturuldu. Sonra buradaki fonksiyonlar, controllerlarda kullanıldı, controllerlardaki db context nesneleri Program.cs'e dependency injection yoluyla tanımlandı ve onların da viewları oluşturuldu.

Örnek, RandevuController.cs

```

[Authorize(Roles = UserRoles.Role_Admin)]
[HttpPost]//bunu yazmazsan yukarıda aynı isimle action olduğu için "catch" çalışır, sayfayı göremezsin.
[ValidateAntiForgeryToken]
public IActionResult EkleGuncelle(Randevu randevu)
{
    if (ModelState.IsValid)
    {
        if (randevu.Id == 0)
        {
            _randevuRepository.Ekle(randevu);
            TempData["basarili"] = @localization.Getkey("Yeni randevu kaydedildi!").Value;
        }
        else
        {
            _randevuRepository.Guncelle(randevu);
            TempData["basarili"] = @localization.Getkey("Randevu güncellendi!").Value;
        }

        _randevuRepository.Kaydet();//bunu yapmazsan db'ye bilgiler eklenmez.

        return RedirectToAction("Index", "Randevu");
    }
    else
    {
        return View();
    }
}

```

Burada, Randevu/EkleGuncelle action metodunu görüyoruz. Hem ekleme hem güncelleme için ayrı ayrı metot ve görünüm yerine, bazı kıstaslar belirtilerek hangisinin kullanılacağını belirleyen bir düzen sağlanmıştır.

```
private readonly IRandevuRepository _randevuRepository;
private readonly IDoktorRepository _doktorRepository;
public readonly IWebHostEnvironment _webHostEnvironment;
private readonly LanguageService _localization;

0 bapuru
public RandevuController(IRandevuRepository randevuRepository, IDoktorRepository doktorRepository, IWebHostEnvironment webHostEnvironment, LanguageService localization)
{
    _randevuRepository = randevuRepository;
    _doktorRepository = doktorRepository;
    _webHostEnvironment = webHostEnvironment;
    _localization = localization;
}
```

```

@model Randevu
{
    ViewData["Title"] = Model == null ? "Randevu Oluştur" : " Randevu Güncelle";
}

<form method="post" enctype="multipart/form-data">

    <div class="row">
        <div class="col-10">
            <div class="border p-3 mt-5">

                <div class="row pb-3">
                    <h2 class="text-primary">@(Model == null ? @Localization.Getkey("Randevu Oluşturu").Value : @Localization.Getkey("Randevuyu Güncelle").Value)</h2>
                    <br>
                </div>

                <div class="mb-3">
                    <label asp-for="HastaId" class="p-0">@Localization.Getkey("HastaId").Value</label>
                    <input asp-for="HastaId" class="form-control" />
                    <span asp-validation-for="HastaId" class="text-danger"></span>
                </div>

                <div class="mb-3">
                    <label asp-for="DoktorId" class="p-0">@Localization.Getkey("Doktor Adı").Value</label>
                    <select asp-for="DoktorId" asp-items="ViewBag.DoktorList" class="form-select"></select>
                    <span asp-validation-for="DoktorId" class="text-danger"></span>
                </div>

                @if (Model != null)
                {
                    <button type="submit" class="btn btn-lg btn-primary" style="width:250px">@Localization.Getkey("Randevuyu Güncelle").Value</button>
                }
                else
                {
                    <button type="submit" class="btn btn-lg btn-primary" style="width:250px">@Localization.Getkey("Yeni Randevu Oluştur").Value</button>
                }

                <a asp-controller="Randevu" asp-action="Index" class="btn btn-lg btn-primary" style="width:250px">
                    @Localization.Getkey("Randevulara Giriş Dön").Value
                </a>

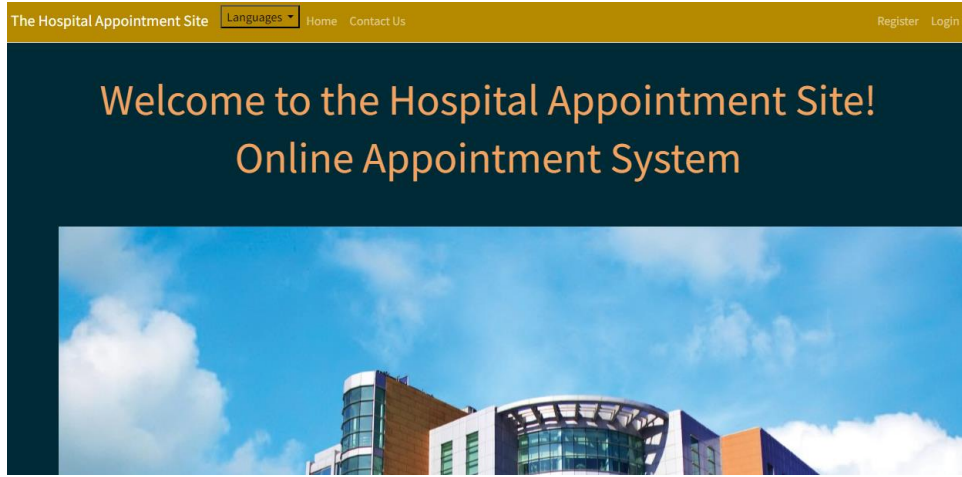
            </div>
        </div>
        <div class="col-2">
            @if (Model != null)
            {
                <br>
                <br />
                
            }
        </div>
    </div>

</form>

```

Örneklerle açıklama yapmaya gayret ettik. Şimdi admin paneline geliyoruz. Areas dosyası sayesinde register olan bir kişiye, MSSql'deki veri tabanımızın "Kimlik" ile beraber otomatik gelen tabloları içinde rol ve rol id tanımları yaparak yeni kaydolan kişiye rol ataması yapıyoruz. Bu şekilde giren kişi, admin mi hasta mı belli oluyor.

Şimdi sitenin bazı ekran görüntülerini paylaşarak raporu sonlandıralım:



Site ilk açıldığında, kullanıcı girişi yapılmadan sadece Anasayfa ve Bize Ulaşın butonları aktif oluyor. Ayrıca görüldüğü üzere default dili Türkçe olan projemizin, resx aracılığıyla İngilizce olarak da görüntülenmesini sağlıyoruz.



Admin olarak giriş yapılan sitede, buton düzeni yukarıdaki gibi olmaktadır. Örneğin Branşlarda, DoktorBranşlari isimli db tablosunda kayıtlı olan değerler ve id'leri görünmektedir ve bizi aşağıdaki gibi bir sayfa beklemektedir:

Doktor Branşları

Id	Doktor Branşları	Güncelle	Sil
1	KBB	Güncelle	Sil
2	Dahiliye	Güncelle	Sil
3	Göz Hastalıkları	Güncelle	Sil
4	Kardioloji	Güncelle	Sil
5	Ortopedi	Güncelle	Sil
6	Üroloji	Güncelle	Sil
7	Kardiyovasküler Cerrahi	Güncelle	Sil
8	Beyin Cerrahisi	Güncelle	Sil
9	Genel Cerrahi	Güncelle	Sil
10	Diş Cerrahisi	Güncelle	Sil
11	Onkoloji	Güncelle	Sil
12	Haciciye(Cildiye)	Güncelle	Sil
13	Psikiyatri	Güncelle	Sil
14	Nöroloji	Güncelle	Sil
15	Göğüs Hastalıkları	Güncelle	Sil
16	Radyoloji	Güncelle	Sil
17	Çocuk Hastalıkları	Güncelle	Sil
18	Çocuk Cerrahisi	Güncelle	Sil
19	Jinekoloji	Güncelle	Sil
20	Travmatoloji	Güncelle	Sil



Yeni Branş Ekle

Görüldüğü üzere, veri tabanında yapacağımız işlemleri burada sunucu taraflı kod yazarak daha estetik biçimde yapma şansına sahip oluyoruz. Güncelleme, silme ve ekleme işlemleri yapılacak butonlar resimde görülmektedir.

Bunun gibi diğer sayfalarda da düzenleme yaparak, onlarla alakalı db tablolarından değişiklikler yapmak mümkün. Şimdi de hasta rolüyle giriş yapalım:

Hastane Randevu Sistemi Diller [Anasayfa](#) [Bize Ulaşın](#) [Doktor Listesi](#) [Merhaba test1@gmail.com!](#) [Çıkış Yap](#)

Doktorlar

Doktorlar	Doktor Adı	Poliklinik	Çalışma Gün ve Saatleri	Branşı	Randevu Al
	Ersan Güllüoğlu	Pol-1	Hafta içi her gün 08.00-17.00	Göz Hastalıkları	Randevu Al
	Alara Kızılbaz	Pol-2	Pazar hariç her gün 10.00-15.00	Nöroloji	Randevu Al

Görüldüğü üzere hasta, sadece randevu alma işlemi yapabiliyor ve doktorun bilgilerine ulaşıp fotoğraflarını da görüntüleyebiliyor. Randevu al'a tıkladığımızda aşağıdaki görüntüyü görmekteyiz:

Hastane Randevu Sistemi **Diller** Anasayfa Bize Ulaşın Doktor Listesi Merhaba test1@gmail.com! Çıkış Yap

Randevu Al

Hastaid

Doktor Adı

Randevu Al **Doktor Listesine Geri Dön**

© 2023 - Hastane Randevu Web Programlama - İletişim

Burada rastgele bir numara girerek randevu kaydı yapıldığı vakit, bu kayıt admin panelinde de oluşturulan randevu kayıtlarıyla aynı tabloya kaydediliyor. Böylece alınan randevu, sunucu tarafından görülüyor.

TEŞEKKÜRLER....