

Hastane TMA

Karabük Üniversitesi Bilgisayar Mühendisliği %30 İngilizce Bölümü, 1.öğretim
Visual Programming Dersi
Nesrin Aydın Atasoy

YAZILIM RAPORU

A- YAZILIM TANITIMI

Programcıların Adı – Soyadı : Mehmet Ali Karadağ, Tülay Çelenk

Geliştirme Ortamları

Programlama Dili	: C#
Programın Adı	: TMA Hastane Sistemi
İşletim Sistemi	: Windows 7/8/10/11

B- KULLANILAN TEKNOLOJİLER

-Visual Studio 2022

Geliştiriciler için üretilmiş bir IDE.

-SQL Server Management Studio 2018

Database engine içeren bir server.

-Entity Framework ORM(Object Relational Mapping)

lişkisel veritabanı ile nesneye yönelik programlama(OOP) arasında bir köprü görevi gören bir framework. Entity Framework(EF) ADO.NET altyapısını kullanmaktadır.

C- YAZILIM TASARIMI

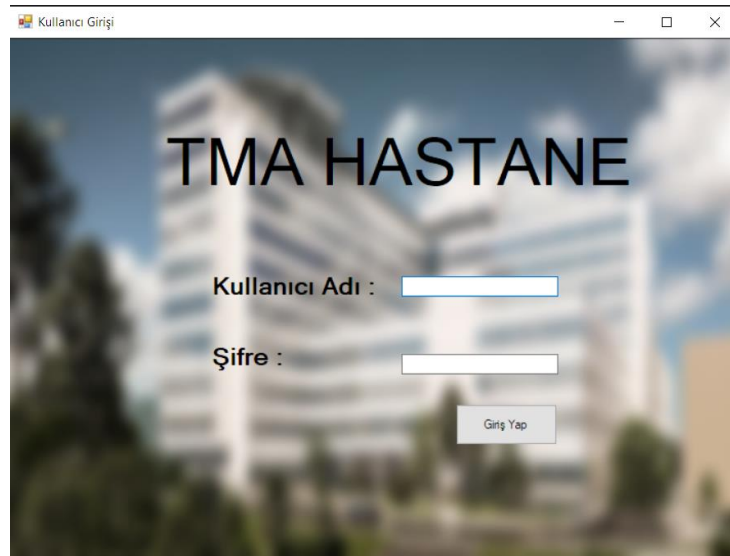
1- Problemin Tanımlanması /Programın Amacı

Yalnızca hastane içerisinde kullanılabilcek bu programın amacı doktorların, sekreterlerin ve yöneticilerin ihtiyacı olan bilgilere ve işlemlere kolayca ulaşabilmesini sağlamak ve genel anlamda hastanede çalışanların ve hasta olarak giriş yapan kişilerin kayıtlarını tutmaktır.

2- Problemin Çözümü / Çözüm Tasarımı

2.1- Son Kullanıcıya (Arabirime) Yönelik Tasarım

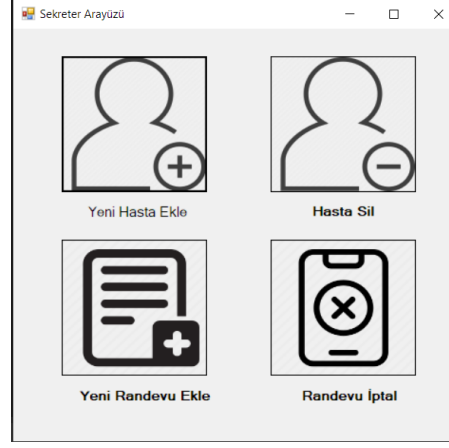
Kullanıcıyı ilk olarak giriş ekranı karşılayacaktır.



Kullanıcı adı ve şifresi ile yetkili olduğu alan için tasarlanan sayfaya yönlendirilecektir. Bu üç ara birim için kullanım ve özellikler şu şekildedir;

1. Sekreter

Sekreterin ekranı aşağıda gösterildiği şekildedir;

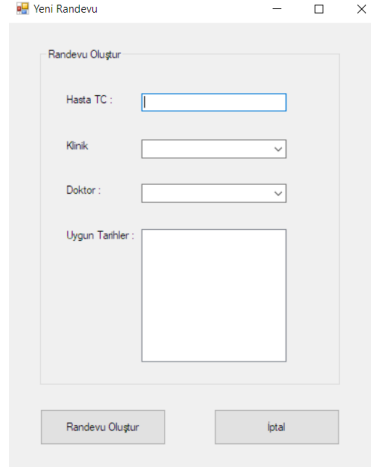


Sekreter ekranı için “Yeni Hasta Ekle”, “Yeni Randevu Ekle”, “Randevu İptal” ve “Hasta Sil” butonları oluşturulmuştur.

Yeni hasta eklemek için hastanın T.C. kimlik numarası, adı, soyadı, cinsiyeti, doğum tarihi ve adres bilgilerinin sekreter tarafından girilmesi istenir. Ardından sağ altta yer alan Yeni Hasta Ekle butonuyla kayıt sağlanır ya da yanında bulunan iptal butonuyla da sayfa kapatılır.

Hasta Sil butonuna tıklandığında gelen ekranda, sistemde kayıtlı olan hastanın T.C. kimlik numarasının girilerek Hastayı Sil butonuna tıklandığında hastanın silinmesi sağlanır.

Yeni randevu eklerken hasta daha önce sisteme eklenmediyse önce sekreterin hastayı “Yeni Hasta Ekle” butonu ile sisteme kaydetmesi gerekir. Eğer sistemde kayıtlıysa hastadan T.C. kimlik numarası, hangi klinikten, hangi doktora ve hangi tarihe randevu oluşturmak istediği bilgileri alınır ve gerekli alanlara girilir. Randevu Oluştur butonuna tıklandığında randevu oluşturulmuş olur.



Randevu İptal butonuyla, yeni açılacak pencerede sekreterin yalnızca hastanın T.C. kimlik numarasını girmesiyle randevuları listeleyebileceği ve iptal edilmesi istenen randevularından en az birini seçtikten sonra randevuyu silebileceği iki buton sayesinde iptal etme işlemi kolayca gerçekleştirilir.



2.Yönetici

Yönetici ekranında çalışanlar, klinikler, gelir ve gider tabloları hızlı ulaşım için konumlandırılmıştır. Bu tablolara yönelik işlemler, ekleme ve silme, sayfada yer alan ilgili yerler doldurularak ilgili butonlar ile yapılabilir. Ayrıca Kullanıcı bilgileri de benzer şekilde eklenip bir buton yardımıyla da görüntülenebilir.

The screenshot displays the 'Yönetim Ekranı' (Management Screen) with four main sections:

- Çalışanlar (Employees):** A table with columns: ID, Ad, Soyad, Onay, Doğum Tarihi, İstisna Tarihi, maaş. The first row is highlighted with ID 48, Ad Mehmet, Soyad Ali, Onay checked, Doğum Tarihi 3.03.2000, İstisna Tarihi 1.01.2015, maaş 5000.
- Çalışan Ekleme (Add Employee):** Form with fields for Ad, Soyad, Onay (radio buttons for Erkek, Kadın), Doğum Tarihi, and Maaş. Buttons: Çalışan Ekle, Çalışan Sil.
- Kullanıcı Ekleme (Add User):** Form with fields for Kullanıcı Adı, Rol No, Şifre, and Çalışan No. Buttons: Kullanıcı Ekle, Kullanıcı Sil, Kullanıcı Bilgisi.
- Klinikler (Clinics):** A table with columns: ID, Klinik Adı. The first row is highlighted with ID 1, Klinik Adı Beyin Cerrahisi.
- Gelir/Gider (Income/Expense):** Two tables. The 'Gelir' (Income) table has columns: Gelir Adı, Miktar, Tarih. The first row is highlighted with Gelir Adı Fecelana, Miktar 50000, Tarih 15.07.2021. The 'Gider' (Expense) table has columns: Gider Adı, Miktar, Tarih. The first row is highlighted with Gider Adı Otlu Muayene, Miktar 4500, Tarih 17.12.2021.

Kullanıcı bilgileri hassas bilgiler içerdiğinden dolayı ekstra bir buton ile gösterilmektedir.

ID	KullanıcıAdı	Şifre	RolNo	ÇalışanNo
21	xmehmetali	123	3	48
22	tulaycelnk	456	3	49
23	duygu_duran	123	1	50
24	can_yemen	123	2	51
25	erkan_dusunen	123	2	52
26	cem_yilmaz	123	2	53
27	enes_yurdatan	123	2	54
28	batikan_cimit	123	2	55
29	sevgican_kilic	123	2	56
30	kemaleddin_fat	123	2	57
33	furkan_yilmaz	123	2	58

Doktorlar ilk öncelikle sisteme çalışan olarak kaydedilmelidir. Çalışan olarak kaydedilen doktorların sisteme erişim sağlayabilmesi için hesap (kullanıcı) oluşturulması gerekmektedir. Her kullanıcının RolNo'su bulunur. Bu RolNo'su ile hesabın sekretere mi, doktora mı yoksa yöneticiye mi ait olduğu anlaşılır.

Hastaneye yeni bir çalışan olarak gelen doktorun kayıtları yapıldıktan sonra Doktor ilişkilendir butonu ile klinik ilişkilendirilmesi yapılır. Bu işlemi yönetici yapar.

Doktor İlişkilendir

İlişkilendirilecek Doktor:

CalisanNo	Ad	Soyad
51	Can	Yemen
52	Erkan	Düğünen
53	Cem	Yılmaz
54	Enes	Yurdatapan
55	Batikan	Cımbıt

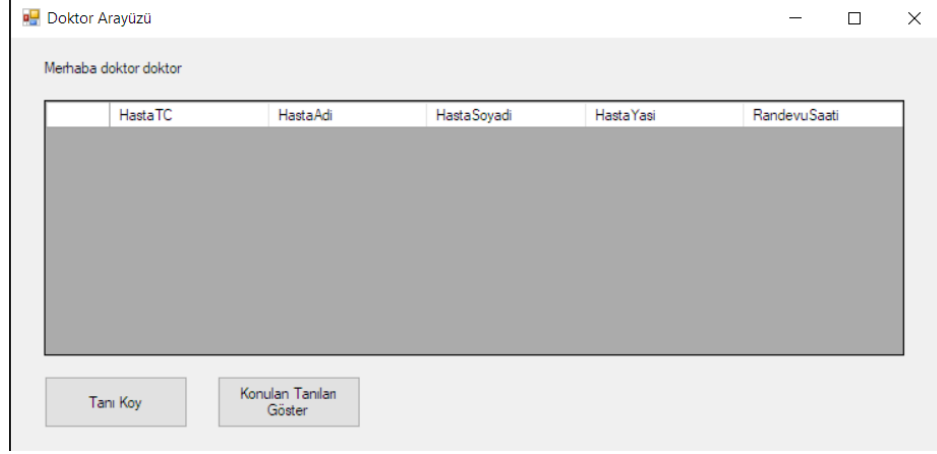
İlişkilendirilecek Klinik:

KlinikAdi
Beyin Cerrahisi
Çocuk Cerrahisi
Dermatoloji
Enfeksiyon Hastalıkları
İç Hastalıkları

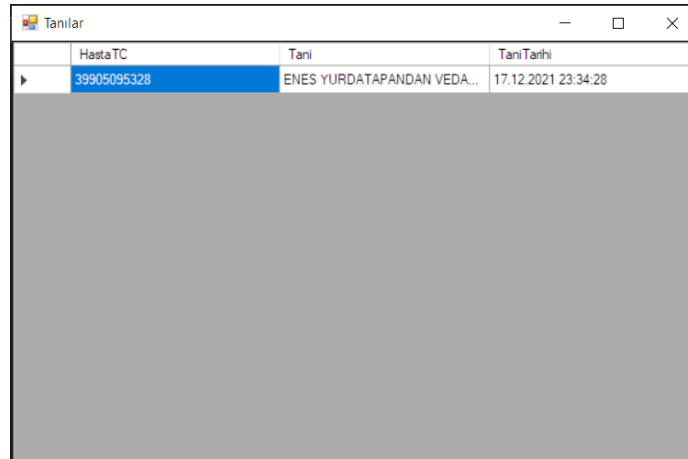
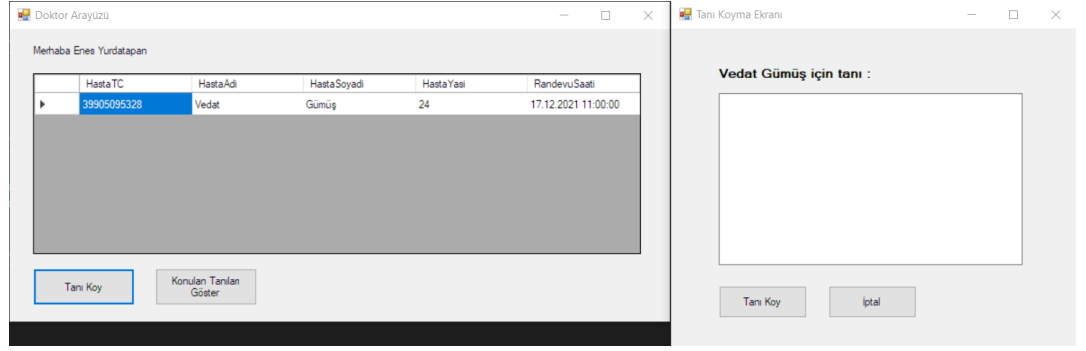
Doktor ve Kliniği İlişkilendir

3. Doktor

Doktorlar için oluşturulmuş ekranda, her doktor o gün muayene edeceği hastaların randevu saatlerini, isim soy isimlerini, T.C. kimlik numaralarını ve yaşlarını aktif olarak görebilecek. Aynı zamanda ekranın altında bulunan “Tanı Koy” butonuyla hastalarını muayene ettikten sonra tabloda görünen hastasının bulunduğu satırın herhangi bir hücreğine tıklayıp, ardından butona bastığında açılan ekran sayesinde hastanın durumunu, kullanması gereken ilaçları veya uygulayacağı tedaviyi kaydetmesi sağlanacaktır.



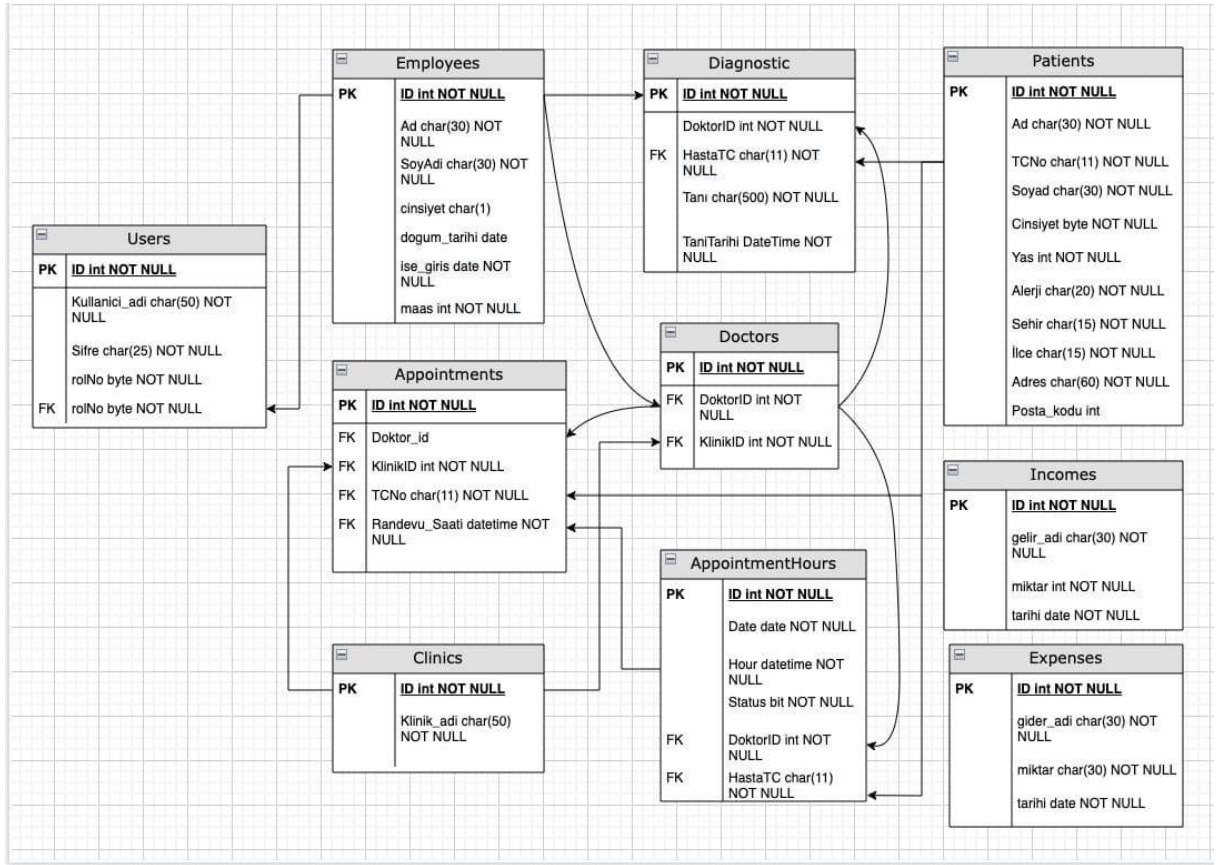
Tanı koyma ekranına geçebilmek için hali hazırda doktora randevu alınmış kişinin üstüne tıklanması ve tanı koy butonuna tıklanmış olması gerekiyor. Butona tıklandıktan sonra tanı koyma ekranı karşımıza çıkıyor. Tanılar Tanı tablosuna kaydedilip saklanıyor ve doktor daha önceden muayene ettiği hastalarının tanılarını görüntüleyebiliyor.



2.2- Programlamaya Yönelik Tasarım

2.2.1- Kod Yapısı

Database Yapısı :



Giriş ekranının kod yapısı:

Giriş ekranında 2 önemli kısım var. Bunlardan ilki sistem açılır açılmaz sisteme kayıtlı olan doktorlar için 1 hafta ötesine kadar günlük 10 saat boş randevu oluşturulmasıdır.

```
using (TMA_DBContext = new Model1()) {
    DateTime today = Convert.ToDateTime(DateTime.Now.ToString("yyyy-MM-dd"));
    DateTime hour = new DateTime(today.Year, today.Month, today.Day, 09, 00, 00);
    var Doctors = TMA_DBContext.Doctors.ToList();

    // --- DOKTOR SAYISI
    for (int k = 0; k < TMA_DBContext.Doctors.Count(); k++)
    {
        // --- GÜN
        for (int j = 0; j < 7; j++)
        {
            // --- SAATLER
            for (int i = 0; i < 10; i++)
            {
                AppointmentHour appointmentHour = new AppointmentHour();
                appointmentHour.Hour = hour;
                appointmentHour.Status = false;
                appointmentHour.HastaTC = null;

                var oneRow = Doctors.OrderBy(x => x.ID).Skip(k).Take(1).ToList();
                appointmentHour.DoktorID = oneRow.FirstOrDefault(r => r.ID == r.ID).DoktorID;

                var check = TMA_DBContext.AppointmentHours.Any(ah => ah.Hour == appointmentHour.Hour &&
                    ah.DoktorID == appointmentHour.DoktorID);

                if (check == false)
                {
                    TMA_DBContext.AppointmentHours.Add(appointmentHour);
                    TMA_DBContext.SaveChanges();
                }
                hour = hour.AddHours(1);
            }
            today = today.AddDays(1);
            hour = new DateTime(today.Year, today.Month, today.Day, 09, 00, 00);
        }
        today = Convert.ToDateTime(DateTime.Now.ToString("yyyy-MM-dd"));
        hour = new DateTime(today.Year, today.Month, today.Day, 09, 00, 00);
    }
}
```

Bu şekilde sisteme eklenmiş yeni bir doktor için otomatik olarak randevular oluşturulmuş olacaktır. Eğer programın her gün açıldığını varsayar isek hali hazırda sisteme kayıtlı olan doktorlar için ertesi güne boş randevu oluşturur.

Diğer önemli kısım ise kullanıcı girişidir. Kullanıcılar mesleklerine göre 3 gruba ayrılıyor.

1 → Sekreter , 2 → Doktor, 3 → Yönetici

Database'den çekilen RolNo verisi ile her bir meslek kendine ait ekranlara yönlendiriliyor.

```
private void btn_Giris_Click(object sender, EventArgs e)
{
    using (TMA_DBContext = new Model1())
    {
        string KullaniciAdi = txtBox_KullaniciAdi.Text;
        string Sifre = txtBox_Sifre.Text;

        var user = TMA_DBContext.Users.FirstOrDefault(u => u.KullaniciAdi == KullaniciAdi);
        if (user != null)
        {
            if (user.Sifre.ToString().Trim(' ') == Sifre)
            {
                switch (user.RolNo)
                {
                    // --- SEKRETER
                    case 1:
                        sekreterinEkranı sekreterinEkranı = new sekreterinEkranı(user.CalisanNo);
                        this.Hide();
                        sekreterinEkranı.Show();
                        break;

                    // --- DOKTOR
                    case 2:
                        doktorunEkranı doktorunEkranı = new doktorunEkranı(user.CalisanNo);
                        this.Hide();
                        doktorunEkranı.Show();
                        break;

                    // --- Yönetici
                    case 3:
                        yoneticininEkranı yoneticininEkranı = new yoneticininEkranı(user.CalisanNo);
                        this.Hide();
                        yoneticininEkranı.Show();
                        break;
                }
            }
            else
            {
                MessageBox.Show("Hatalı Giriş!");
            }
        }
        else
        {
            MessageBox.Show("Kullanıcı Bulunamadı!");
        }
    }
}
```

Yönlendirme yapılırken CalisanNo (Employees tablosundaki ID) verisinin diğer formlara gönderildiğine dikkat ediniz.

1-)Sekreter:

```
1 reference
private void btn_YeniHastaEkle_Click(object sender, EventArgs e)
{
    hastaEkle hastaEkle = new hastaEkle();
    hastaEkle.Show();
}

1 reference
private void btn_YeniRandevuEkle_Click(object sender, EventArgs e)
{
    yeniRandevu yeniRandevu = new yeniRandevu();
    yeniRandevu.Show();
}

1 reference
private void button1_Click(object sender, EventArgs e)
{
    hastaSil hastaSil = new hastaSil();
    hastaSil.Show();
}

1 reference
private void sekreterinEkranı_Load(object sender, EventArgs e)
{
}

1 reference
private void btn_RandevuIptal_Click(object sender, EventArgs e)
{
    randevuSil randevuSil = new randevuSil();
    randevuSil.Show();
}
```



Bu ekranda sadece gerekli pencerelere yönlendirme yapılıyor.

Gerekli veriler alınarak Database'deki Patients tablosunda tek bir kayda denk gelen Patient objesi oluşturulur. Bu objeye, alınan veriler yerleştirilip EntityFrameWork aracılığı ile Database'e kayıt sağlanır.

```
1 reference
private void btn_YeniHastaEkle_Click(object sender, EventArgs e)
{
    Model1 TMA_DBContext = new Model1();
    // --- HASTA BİLGİLERİ
    string TCNo = txtBox_TC.Text;
    string Ad=txtBox_Ad.Text;
    string Soyad=txtBox_Soyad.Text;
    bool Cinsiyet;
    if (rdBtn_CinsiyetErkek.Checked)
        Cinsiyet = false;
    else
        Cinsiyet=true;
    string yas = dateTime_Yas.Value.ToString("yyyy");

    // --- ADRES
    string İlce=txtBox_Ilce.Text;
    string Adres=txtBox_Adres.Text;
    string PostaKodu = txtBox_PostaKodu.Text;

    //-----Patient objesi :
    Patient patient=new Patient();
    patient.Ad = Ad;
    patient.Soyad = Soyad;
    patient.TCNo = TCNo;
    patient.Cinsiyet = Cinsiyet;
    patient.Yas = 2021-Convert.ToInt32(yas);
    patient.ilce = İlce;
    patient.Adres = Adres;
    patient.PostaKodu= PostaKodu;

    //try catch ekle buraya
    try {
        TMA_DBContext.Patients.Add(patient);
        TMA_DBContext.SaveChanges();
        MessageBox.Show("Hasta Eklendi!");
    }
    catch { MessageBox.Show("Hasta zaten kayıtlı!"); }
```

Hasta silme ekranında ise hastaya ait TC no gerekir. Bu TC no patients tablosunda aranıp hasta silinir. Hasta silinince hastaya ait tüm diğer verilerde silinir. Bunun nedeni ise Database'de foreign keyleri tanımlarken cascade deleting fonksiyonunun aktif edilmiş olmasıdır.

```
1 reference
private void btn_Sil_Click(object sender, EventArgs e)
{
    string TCNo = txtBox_HastaTC.Text;
    TMA_DBContext = new Model1();

    //Patient patient=new Patient() { TCNo=TCNo};

    var table = TMA_DBContext.Patients.ToList();
    var res = table.Where(s => s.TCNo==TCNo).ToList();

    TMA_DBContext.Patients.RemoveRange(res);
    TMA_DBContext.SaveChanges();

    MessageBox.Show( "HASTA SİLİNDİ");
}
```

Yeni randevu ekleme ekranında ise hastanın TC no'su alınır ve klinik, doktor ve uygun bir tarih seçilir. Ardından randevu oluşturulur. Klinik seçildikten sonra kliniğe ait doktorlar listelenir.

Klinik seçildikten sonra arka planda klinik combobox'ının indexi alınır ve doktorlar tablosundaki KlinikID sütunu, klinik combobox indexine eşit olanlar doktor combobox'ında gösterilir.

```
int[] dx;  
//reference  
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)  
{  
    cbo_Doktor.Items.Clear();  
    //https://hexacodein.com/makale/ling-include-kullanimi  
    var Doctors = TMA_DBContext.Doctors.Include("Employee").Where(d => d.KlinikID == cbo_Klinik.SelectedIndex + 1).ToList();  
    int boyut = Doctors.Count();  
  
    int l = 0;  
    dx=new int[boyut];  
    foreach (var d in Doctors)  
    {  
        cbo_Doktor.Items.Add(d.Employee.Ad.Trim(' ') + " " + d.Employee.Soyad.Trim(' '));  
        dx[l] = d.DoktorID;  
        l++;  
    }  
}
```

Randevu iptal ekranında ise ilk önce hastanın TC'si gerekiyor ki o hastaya ait randevuları görüntülenebilsin. Bu işlem, bir hasta birden fazla randevuya sahip olabileceğinden dolayı yapılmaktadır. Hastaya ait randevular listbox sayesinde gösterilmektedir. Gösterilen randevulardan iptal edilmesi istenen kayıt seçilir ve “Randevuyu Sil” butonuna tıklanır.

Try bloğunda Appointments tablosuna erişip girilen tc ile eşleşen kayıt var mı diye kontrol edilir (Model objesi oluşturulup Appointments tablosunun temsiline erişerek). Eğer kayıt/kayıtlar var ise bunları randevular değişkeninin içine atar. Bu randevuları listbox'ın içinde gösterir.

```
//reference  
private void btn_RandevulariListele_Click(object sender, EventArgs e)  
{  
    listBox_Randevular.Items.Clear();  
    TCNo = textBox_HastaTC.Text;  
    using (TMA_DBContext = new Model1())  
    {  
        try  
        {  
            randevular = TMA_DBContext.Appointments.Where(a => a.TCNo == TCNo).ToList();  
            foreach (var item in randevular)  
            {  
                listBox_Randevular.Items.Add(item.RandevuSaatl);  
            }  
            var AdSoyad = TMA_DBContext.Patients.Where(p => p.TCNo == TCNo).FirstOrDefault();  
            lbl_HastaAdSoyad.Text = AdSoyad.Ad.Trim() + " " + AdSoyad.Soyad.Trim() + "\n(için alınmış  
            randevular :";  
        }  
        catch  
        {  
            MessageBox.Show("Bir sorunla karşılaşıldı!");  
        }  
    }  
}
```

Listbox'ta seçilen değer (tarih olacaktır) string ifadeye dönüştürülüp bir değişkene atanır. Ardından try bloğunda TC no ve randevu tarihi ile eşleşen randevu var mı diye kontrol edilir (Appointments tablosunda). Eğer böyle bir kayıt var ise bu kayıt tablodan silinir.

```
private void btn_RandevuSil_Click(object sender, EventArgs e)
{
    string RandevuSaati = lstBox_Randevular.SelectedItem.ToString();

    using (TMA_DBContext = new Model1())
    {
        try
        {
            var tarih = Convert.ToDateTime(lstBox_Randevular.SelectedItem.ToString());
            var kaldır = TMA_DBContext.Appointments.Where(a => a.TCNo == TCNo && a.RandevuSaati == tarih).FirstOrDefault();
            TMA_DBContext.Appointments.Remove(kaldır);
            TMA_DBContext.SaveChanges();
            MessageBox.Show("Randevu Başarıyla Silindi!");
            txtBox_HastaTC.Clear();
            lstBox_Randevular.Items.Clear();
        }
        catch
        {
            MessageBox.Show("Bir sorunla karşılaşıldı.");
        }
    }
}
```

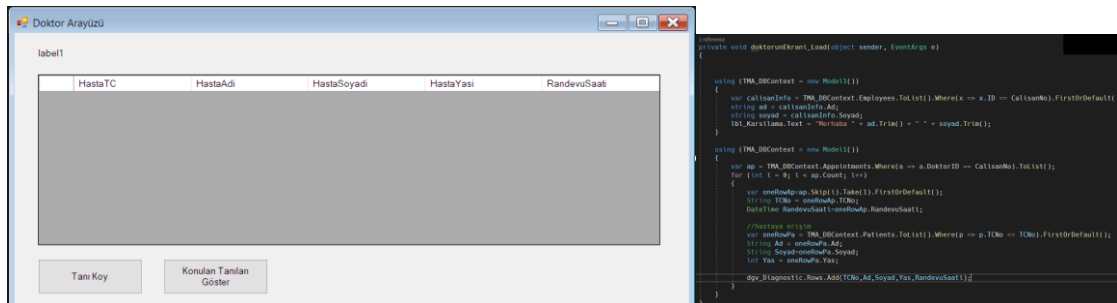
2-)Doktor:

Doktor giriş yaptıktan sonra doktor arayüzüne yönlendirilir.

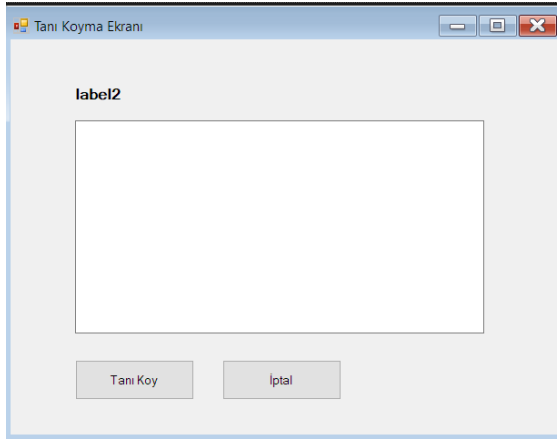
Doktor bu arayüzde bakacağı hastaların bilgilerini görür. Sırası gelen hastanın üstüne tıklayıp “Tanı Koy” butonuna tıklayarak tanı koyabilir. Daha önceden koyduğu tanıları görüntülemek için “Konulan Tanıları Göster” butonuna tıklaması yeterlidir.

Birinci using bloğunda doktoru karşılama mesajı oluşturulup ekranın sol üst kısmında gösteriliyor. Bu, giriş ekranından gelen CalisanNo değişkeni ile sağlanabiliyor. CalisanNo ile doktora ait bilgilerden isim ve soyisim bilgilerini elde ederek label'ın içeriğini “Merhaba isim soyisim” şeklinde değiştiriyoruz.

İkinci using bloğunda ise doktora ait hastaların bilgilerini görüntülüyoruz. Elde ettiğimiz randevu bilgilerinin listesini for döngüsüne sokuyoruz. İ değişkeni ile atlanacak satırı belirtip satırı take fonksiyonu ile alıyoruz. Bu bize doktora ait tek bir randevuyu veriyor. Bu satır ile datagridview'ın içindeki gerekli yerler dolduruluyor.

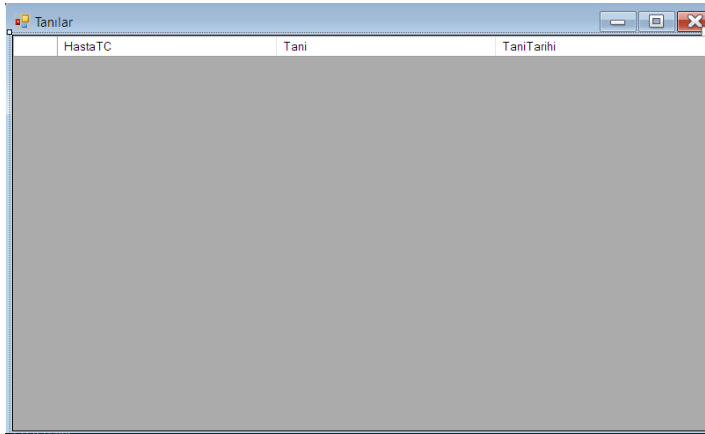


Gerekli veriler alındıktan sonra Diagnostic nesnesi oluşturulup database'e yükleme yapılır.



```
1 reference
private void btn_TaniKoy_Click(object sender, EventArgs e)
{
    Diagnostic diagnostic = new Diagnostic();
    diagnostic.DoktorID = CalisanNo;
    diagnostic.HastaTC = TCNo;
    diagnostic.Tani = txtBox_Tani.Text; //BURAYI GİRSİN MAX 500 KARAKTER
    diagnostic.TaniTarihi = DateTime.Now;

    using (TMA_DBContext=new Model1())
    {
        try{
            TMA_DBContext.Diagnostics.Add(diagnostic);
            TMA_DBContext.SaveChanges();
            MessageBox.Show("Tani oluşturuldu.");
            this.Hide();
        }
        catch
        {
            MessageBox.Show("Bir hata ile karşılaşıldı!");
        }
    }
}
```



```
public konulanTanilariGoster(int CalisanNo,string TCNo)
{
    InitializeComponent();
    this.CalisanNo = CalisanNo;
    this.TCNo = TCNo;

    using (TMA_DBContext = new Model1())
    {
        var tanilar = TMA_DBContext.Diagnostics.Where(t => t.DoktorID == CalisanNo).ToList();
        foreach (var t in tanilar)
            dgv_KonulanTanilar.Rows.Add(t.HastaTC, t.Tani, t.TaniTarihi);
    }
}
```

Aynı şekilde yükleme yapıldığı gibi veriler çekilerek gösterimi sağlanır.

3-)Yönetici:

Yönetim Ekranı

Çalışanlar

ID	Ad	Soyad	Cinsiyet	DogumTarihi	IseGirisTarihi	maas
48	Mehmet Ali	Karadağ	<input type="checkbox"/>	3.03.2000	1.01.2015	5000
49	Tülay	Çelenk	<input checked="" type="checkbox"/>	10.02.2001	2.01.2015	6000
50	Duygu	Duran	<input checked="" type="checkbox"/>	12.01.1999	30.01.2013	3000
51	Can	Yemen	<input type="checkbox"/>	12.12.1998	22.05.2014	3500
52	Ekran	Düğün	<input type="checkbox"/>	12.06.1995	11.11.2015	4500
53	Cem	Yılmaz	<input type="checkbox"/>	10.12.1972	16.08.2011	4500
54	Enes	Yurdatapan	<input type="checkbox"/>	14.06.1990	17.12.2021	5000
55	Batikan	Cımbıt	<input type="checkbox"/>	10.06.1990	17.12.2021	4000
56	Sevgi Can	Kılıç	<input checked="" type="checkbox"/>	5.07.1990	17.12.2021	5000
57	Kemal	Fatıma	<input type="checkbox"/>	12.07.1935	17.12.2021	22111
58	Furkan	Yılmaz	<input type="checkbox"/>	18.07.1986	17.12.2021	50
59	Ali Eren	Ergün	<input type="checkbox"/>	17.03.1989	17.12.2021	90222

1

Çalışan Ekleme

Ad : Soyad :

Cinsiyet : ☐ Erkek ☐ Kadın Doğum Tarihi : 19 Aralık 2021 Pazar

Maas :

Çalışan Ekle Çalışan Sil

Klinikler

3

ID	KlinikAdi
1	Beyin Cerrahisi
2	Çocuk Cerrahisi
3	Dermatoloji
4	Enfeksiyon Hastalıkları
5	İç Hastalıkları
6	Diyetetik

Klinik Adı : Klinik Ekle Klinik Sil Doktor İlgilidir

4

Gelir/Gider

GelirAdi	Miktar	Tarih
Forlana	50000	15.07.2021
Ahmet Çetin Baş	10000	5.08.2021

GiderAdi	Miktar	Tarih
Ofis Malzemeleri	4500	17.12.2021
Ambulans Muayene	15000	17.12.2021

Gelir Ekle Gelir Sil Gider Ekle Gider Sil

Gelir Adı : Gider Adı :

Miktar : Miktar :

Yönetici ekranı açılır açılmaz veri yüklemesi şu şekilde yapılmaktadır;

```
1 reference
private void yöneticininEkranı_Load(object sender, EventArgs e)
{
    this.usersTableAdapter.Fill(this.hastaneTMADatasetUsers.Users);
    this.expensesTableAdapter.Fill(this.hastaneTMADataset2.Expenses);
    this.incomesTableAdapter.Fill(this.hastaneTMADataset1.Incomes);
    this.clinicsTableAdapter.Fill(this.hastaneTMAClinics.Clinics);
    this.employeesTableAdapter.Fill(this.hastaneTMADataset.Employees);
}
```

Çalışanlar tablosuna çalışan ekleme;

```
// --- OBYENIN OLUŞTURULMASI VE DATABASE'E YÜKLENMESİ
if (textBox_Ad.Text != String.Empty && textBox_Soyad.Text != String.Empty && (rdBtn_Erkek.Checked || rdBtn_Kadin.Checked) && textBox_Maas.Text != String.Empty)
{
    Employee employee = new Employee();
    employee.Ad = textBox_Ad.Text;
    employee.Soyad = textBox_Soyad.Text;
    employee.Cinsiyet = rdBtn_Erkek.Checked ? false : true;
    employee.DogumTarihi = Convert.ToDateTime(dateTime_DogumTarihi.Value.ToString("yyyy-MM-dd"));
    employee.IseGirisTarihi = Convert.ToDateTime(DateTime.Now.ToString("yyyy-MM-dd"));
    employee.maas = Int.Parse(textBox_Maas.Text.ToString());

    TMA_DBContext.Employees.Add(employee);
    TMA_DBContext.SaveChanges();
    refreshData("Employees");
}
else
    MessageBox.Show("Lütfen Boş alanları doldurunuz!");
```

Kullanıcılar tablosuna kullanıcı ekleme;

```
private void btn_KullaniciEkle_Click(object sender, EventArgs e)
{
    string kullanicAdi = txtBox_KullaniciAdi.Text;
    string sifre = txtBox_Sifre.Text;
    int rolNo = int.Parse(txtBox_RolNo.Text);
    int calisanNo = int.Parse(txtBox_CalisanNo.Text);

    try
    {
        User user = new User();
        user.KullaniciAdi = kullanicAdi;
        user.Sifre = sifre;
        user.RolNo = rolNo;
        user.CalisanNo = calisanNo;

        TMA_DBContext.Users.Add(user);
        TMA_DBContext.SaveChanges();
        MessageBox.Show("Kullanıcı başarılı bir şekilde kaydedildi.");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Bir hata ile karşılaşıldı!\n" +
            "Lütfen kullanıcı oluşturmadan önce çalışan oluşturunuz.");
    }
}
```

Kullanıcı ID'si gösterilen textbox'tan alınır ve kullanıcıID değişkenine atanır. Bu ID'ye sahip kullanıcı users tablosundan bulunarak Context yardımı ile silme işlemi yapılır.

```
private void btn_KullaniciSil_Click(object sender, EventArgs e)
{
    int kullaniciID = int.Parse(txtBox_KullaniciSil.Text.ToString());

    using (TMA_DBContext = new Model1())
    {
        try
        {
            var kullanici = TMA_DBContext.Users.Where(ku => ku.ID == kullaniciID).ToList().FirstOrDefault();
            TMA_DBContext.Users.Remove(kullanici);
            TMA_DBContext.SaveChanges();
            MessageBox.Show("Kullanıcı başarılı bir şekilde silindi.");
            txtBox_KullaniciSil.Clear();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Bir hatayla karşılaşıldı!\n" + ex.ToString());
        }
    }
}
```

Kullanıcı ID :

Klinik ekleme için gerekli alanlardan veri alınıp klinik nesnesi oluşturulur ve bu nesne veri tabanına eklenir.

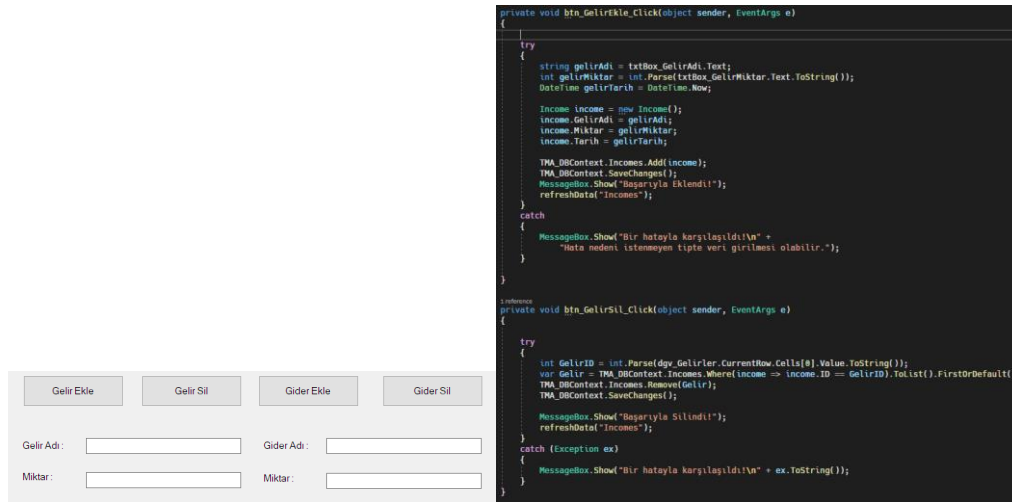
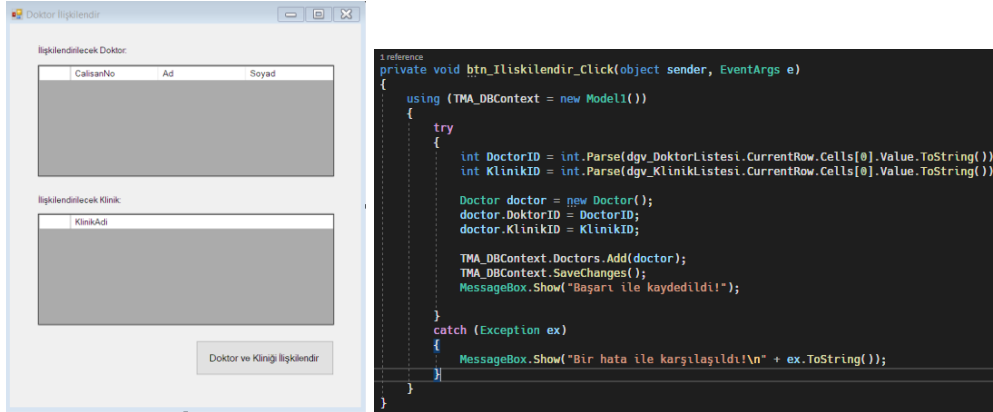
```
private void btn_KlinikEkle_Click(object sender, EventArgs e)
{
    if (txtBox_KlinikAdi != null)
    {
        Clinic clinic = new Clinic();
        clinic.KlinikAdi = txtBox_KlinikAdi.Text;
        TMA_DBContext.Clinics.Add(clinic);
        TMA_DBContext.SaveChanges();
        refreshData("Clinics");
    }
    else
    {
        MessageBox.Show("Lütfen Değer Giriniz!");
    }
}
```

Datagridview'da seçilen satırın id'si klinikID değişkenine atanır. Klinik değişkeni Klinikler tablosunda KlinikID'ye eşit olan kayıda denk gelir. Bu kayıt Context yardımı ile silinir.

```
private void btn_KlinikSil_Click(object sender, EventArgs e)
{
    try
    {
        var klinikID = int.Parse(dgv_Klinikler.CurrentRow.Cells[0].Value.ToString());
        var klinik = TMA_DBContext.Clinics.Where(k => k.ID == klinikID).ToList().FirstOrDefault();
        TMA_DBContext.Clinics.Remove(klinik);
        TMA_DBContext.SaveChanges();
        MessageBox.Show("Başarıyla Silindi!");
        refreshData("Clinics");
    }
    catch
    {
        MessageBox.Show("Bir hatayla karşılaşıldı!");
    }
}
```


Doktor ilişkilendirme için yeni bir pencere açılır. Bu ekranda sadece doktorları görüntülemek istediğimizden RolNo'su 2 olan kullanıcıların ad ve soyad bilgileri görüntülenir (kullanıcılar tablosunda ad ve soyad bilgileri tutulmaz ancak kullanıcılar tablosundaki ÇalışanNo foreign key'i ile Çalışanlar tablosuna erişip ad soyad bilgilerini elde ederiz).

İlişkilendirme için DoktorID ve KlinikID alınır. Doktorlar tablosunda DoktorID ve KlinikID sütunları bulunuyordu ve bu bize hangi doktor hangi kliniğe ait olduğunu bilgisini veriyor. Bu 2 veriyi doctor objesi oluşturup Context yardımı ile veri tabanına yüklüyoruz.



Gelir ekleme için obje oluşturulur Context yardımıyla veri tabanına yüklenir. Gelir silme işlemi için datagridview'de seçilen satırın ID'si alınır ve eşleşen kayıt veri tabanından silinir.

Aynı işlemler gider ekleme ve silme içinde geçerlidir.

```
private void btn_GiderSil_Click(object sender, EventArgs e)
{
    try
    {
        int GiderID = int.Parse(dgv_Giderler.CurrentRow.Cells[0].Value.ToString());
        var gider = TMA_DBContext.Expenses.Where(expense => expense.ID == GiderID).FirstOrDefault();
        TMA_DBContext.Expenses.Remove(gider);
        TMA_DBContext.SaveChanges();
        MessageBox.Show("Bakartıyla Silindi!");
        refreshData("Expenses");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Bir hatayla karşılaşıldı!" + ex.ToString());
    }
}

private void btn_GiderEkle_Click(object sender, EventArgs e)
{
    try
    {
        string giderAdi = txtBox_GiderAdi.Text;
        int giderMiktar = int.Parse(txtBox_GiderMiktar.Text.ToString());
        DateTime giderTarih = DateTime.Now;

        Expense expense = new Expense();
        expense.GiderAdi = giderAdi;
        expense.Miktar = giderMiktar;
        expense.Tarih = giderTarih;
        TMA_DBContext.Expenses.Add(expense);
        TMA_DBContext.SaveChanges();
        MessageBox.Show("Bakartıyla Eklendi!");
        refreshData("Expenses");
    }
    catch
    {
        MessageBox.Show("Bir hatayla karşılaşıldı!" +
            "Buna nedenli olmayan tipte veri girilemez olabilir.");
    }
}
```

PROGRAMIN KODLARI :

App.config:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit
http://go.microsoft.com/fwlink/?LinkId=237468 -->
    <section name="entityFramework"
type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection,
EntityFramework, Version=6.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089" requirePermission="false" />
  </configSections>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
  </startup>
  <entityFramework>
    <providers>
      <provider invariantName="System.Data.SqlClient"
type="System.Data.Entity.SqlServer.SqlProviderServices,
EntityFramework.SqlServer" />
      <provider invariantName="JetEntityFrameworkProvider"
type="JetEntityFrameworkProvider.JetProviderServices, JetEntityFrameworkProvider"
/>
    </providers>
  </entityFramework>
  <connectionStrings>
    <!--
```

SERVER :

```
    <add name="TMA_DBContext" connectionString="data
source=DESKTOP-ALI; initial catalog=HastaneTMA; integrated security=True"
providerName="System.Data.SqlClient" />
    <add name="TMA_DBContext_Config" connectionString="data source=DESKTOP-
ALI;initial catalog=HastaneTMA;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"
providerName="System.Data.SqlClient" />
    <add name="Model1" connectionString="data source=DESKTOP-ALI;initial
catalog=HastaneTMA;integrated
```

```
security=True;MultipleActiveResultSets=True;App=EntityFramework"
providerName="System.Data.SqlClient" />
```

```
*****
DATABASE'İN (HASTANETMA) MSSQL'E İMPORT EDİLMESİ GEREK
NOT : AŞAĞIDAKİ CONNECTION STRING PROGRAMIN AÇILACAĞI
BİLGİSAYARA GÖRE DEĞİŞTİRİLMELİ.
```

```
-->
<add name="TMA_DBContext" connectionString="data source=DESKTOP-ALI;
initial catalog=HastaneTMA; integrated security=True"
providerName="System.Data.SqlClient" />
<add name="TMA_DBContext_Config" connectionString="data source=DESKTOP-
ALI;initial catalog=HastaneTMA;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"
providerName="System.Data.SqlClient" />
<add name="Model1" connectionString="data source=DESKTOP-ALI;initial
catalog=HastaneTMA;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"
providerName="System.Data.SqlClient" />
```

```
</connectionStrings>
```

```
</configuration>
```

```
girisEkrani.cs
```

```
using HastaneTMA.Model;
using System;
using System.Linq;
using System.Windows.Forms;
using HastaneTMA.Doktor;

namespace HastaneTMA
{
    /*
        1-SEKRETER
        2-DOKTOR
        3-YÖNETİCİ

    */

    public partial class girisEkrani : Form
    {
        private Model1 TMA_DBContext;

        public girisEkrani()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {

```

```

        using (TMA_DBContext = new Model1()) {
            DateTime today = Convert.ToDateTime(DateTime.Now.ToString("yyyy-
MM-dd"));
            DateTime hour = new DateTime(today.Year, today.Month, today.Day,
09, 00, 00);
            var Doctors = TMA_DBContext.Doctors.ToList();

            // --- DOKTOR SAYISI
            for (int k = 0; k < TMA_DBContext.Doctors.Count(); k++)
            {
                // --- GÜN
                for (int j = 0; j < 7; j++)
                {
                    // --- SAATLER
                    for (int i = 0; i < 10; i++)
                    {
                        AppointmentHour appointmentHour = new
AppointmentHour();
                        appointmentHour.Hour = hour;
                        appointmentHour.Status = false;
                        appointmentHour.HastaTC = null;

                        var oneRow = Doctors.OrderBy(x =>
x.ID).Skip(k).Take(1).ToList();
                        appointmentHour.DoktorID
=oneRow.FirstOrDefault(r=>r.ID==r.ID).DoktorID;

                        var check = TMA_DBContext.AppointmentHours.Any(ah =>
ah.Hour == appointmentHour.Hour && ah.DoktorID==appointmentHour.DoktorID);

                        if (check == false)
                        {
                            TMA_DBContext.AppointmentHours.Add(appointmentHour);
                            TMA_DBContext.SaveChanges();
                        }
                        hour = hour.AddHours(1);
                    }
                    today = today.AddDays(1);
                    hour = new DateTime(today.Year, today.Month, today.Day,
09, 00, 00);
                }
                today= Convert.ToDateTime(DateTime.Now.ToString("yyyy-MM-
dd"));
                hour = new DateTime(today.Year, today.Month, today.Day, 09,
00, 00);
            }
        }

        private void label1_Click(object sender, EventArgs e)
        {
        }

        private void btn_Giris_Click(object sender, EventArgs e)
        {
            using (TMA_DBContext = new Model1())
            {

```

```

        string KullaniciAdi = txtBox_KullaniciAdi.Text;
        string Sifre = txtBox_Sifre.Text;

        var user = TMA_DBContext.Users.FirstOrDefault(u => u.KullaniciAdi
== KullaniciAdi);
        if (user != null)
        {
            if (user.Sifre.ToString().Trim(' ') == Sifre)
            {
                switch (user.RolNo)
                {
                    // --- SEKRETER
                    case 1:
                        sekreterinEkrani sekreterinEkrani = new
sekreterinEkrani(user.CalisanNo);
                        this.Hide();
                        sekreterinEkrani.Show();
                        break;

                    // --- DOKTOR
                    case 2:
                        doktorunEkrani doktorunEkrani = new
doktorunEkrani(user.CalisanNo);
                        this.Hide();
                        doktorunEkrani.Show();
                        break;

                    // --- Yönetici
                    case 3:
                        yoneticininEkrani yoneticininEkrani=new
yoneticininEkrani(user.CalisanNo);
                        this.Hide();
                        yoneticininEkrani.Show();
                        break;
                };
            }
            else
            {
                MessageBox.Show("Hatalı Giriş!");
            }
        }
        else
        {
            MessageBox.Show("Kullanıcı Bulunamadı!");
        }
    }
}
}
}

```

doktorunEkrani.cs

```

using HastaneTMA.Model;
using System;
using System.Data;
using System.Linq;
using System.Windows.Forms;
using HastaneTMA.Model;

```

```

namespace HastaneTMA.Doktor
{
    public partial class doktorunEkrani : Form
    {
        private Model1 TMA_DBContext;
        private int CalisanNo;
        string Ad, Soyad, TCNo;
        public doktorunEkrani(int CalisanNo)
        {
            InitializeComponent();
            this.CalisanNo = CalisanNo;
        }

        private void doktorunEkrani_Load(object sender, EventArgs e)
        {

            using (TMA_DBContext = new Model1())
            {
                var calisanInfo = TMA_DBContext.Employees.ToList().Where(x =>
x.ID == CalisanNo).FirstOrDefault();
                string ad = calisanInfo.Ad;
                string soyad = calisanInfo.Soyad;
                lbl_Karsilama.Text = "Merhaba " + ad.Trim() + " " + soyad.Trim();
            }

            using (TMA_DBContext = new Model1())
            {
                var ap = TMA_DBContext.Appointments.Where(a => a.DoktorID ==
CalisanNo).ToList();
                for (int i = 0; i < ap.Count; i++)
                {
                    var oneRowAp=ap.Skip(i).Take(1).FirstOrDefault();
                    String TCNo = oneRowAp.TCNo;
                    DateTime RandevuSaati=oneRowAp.RandevuSaati;

                    //hastaya erişim
                    var oneRowPa = TMA_DBContext.Patients.ToList().Where(p =>
p.TCNo == TCNo).FirstOrDefault();
                    String Ad = oneRowPa.Ad;
                    String Soyad=oneRowPa.Soyad;
                    int Yas = oneRowPa.Yas;

                    dgv_Diagnostic.Rows.Add(TCNo,Ad,Soyad,Yas,RandevuSaati);
                }
            }

            private void lbl_Karsilama_Click(object sender, EventArgs e)
            {

            }

            private void btn_KonulanTanilariGoster_Click(object sender, EventArgs e)
            {
                konulanTanilariGoster konulanTanilariGoster=new
konulanTanilariGoster(CalisanNo,TCNo);
                konulanTanilariGoster.Show();
            }

            private void dgv_Diagnostic_CellContentClick(object sender,
DataGridViewCellEventArgs e)

```

```

    {
    }

    private void btn_TaniKoy_Click(object sender, EventArgs e)
    {
        TCNo = dgv_Diagnostic.CurrentRow.Cells[0].Value.ToString();
        Ad= dgv_Diagnostic.CurrentRow.Cells[1].Value.ToString();
        Soyad= dgv_Diagnostic.CurrentRow.Cells[2].Value.ToString();
        taniKoy taniKoy=new taniKoy(CalisanNo,TCNo,Ad,Soyad);
        taniKoy.Show();
    }
}

```

konulanTanilariGoster.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using HastaneTMA.Model;

namespace HastaneTMA.Doktor
{
    public partial class konulanTanilariGoster : Form
    {
        int CalisanNo;
        string TCNo;

        private Model1 TMA_DBContext;
        public konulanTanilariGoster(int CalisanNo,string TCNo)
        {
            InitializeComponent();
            this.CalisanNo = CalisanNo;
            this.TCNo = TCNo;

            using (TMA_DBContext = new Model1())
            {
                var tanilar = TMA_DBContext.Diagnostics.Where(t => t.DoktorID ==
                CalisanNo).ToList();
                foreach (var t in tanilar)
                {
                    dgv_KonulanTanilar.Rows.Add(t.HastaTC, t.Tani, t.TaniTarihi);
                }
            }

            private void dgv_KonulanTanilar_CellContentClick(object sender,
            DataGridViewCellEventArgs e)

```

```

    {
    }
}

```

taniKoy.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using HastaneTMA.Model;

namespace HastaneTMA.Doktor
{
    public partial class taniKoy : Form
    {
        int CalisanNo;
        string TCNo, Ad, Soyad;
        private Model1 TMA_DBContext;
        private void btn_TaniKoy_Click(object sender, EventArgs e)
        {
            Diagnostic diagnostic = new Diagnostic();
            diagnostic.DoktorID = CalisanNo;
            diagnostic.HastaTC = TCNo;
            diagnostic.Tani = txtBox_Tani.Text;    //BURAYI GİRSİN MAX 500
            diagnostic.TaniTarihi = DateTime.Now;

            using (TMA_DBContext=new Model1())
            {
                try{
                    TMA_DBContext.Diagnostics.Add(diagnostic);
                    TMA_DBContext.SaveChanges();
                    MessageBox.Show("Tanı oluşturuldu.");
                    this.Hide();
                }
                catch
                {
                    MessageBox.Show("Bir hata ile karşılaşıldı!");
                }
            }

            private void btn_Iptal_Click(object sender, EventArgs e)
            {
                this.Hide();
            }

            public taniKoy(int CalisanNo, string TCNo, string Ad, string Soyad)
            {

```

KARAKTER


```

        InitializeComponent();
        this.CalisanNo = CalisanNo;
        this.TCNo = TCNo;
        this.Ad = Ad;
        this.Soyad = Soyad;
    }

    private void TaniKoy_Load(object sender, EventArgs e)
    {
        lbl_AdSoyad.Text = Ad.Trim()+" "+Soyad.Trim()+" için tanı :";
    }
}

```

sekreteriEkrani.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using HastaneTMA.Sekreter;

namespace HastaneTMA
{
    public partial class sekreterinEkrani : Form
    {
        private int CalisanNo;
        public sekreterinEkrani(int CalisanNo)
        {
            InitializeComponent();
            this.CalisanNo = CalisanNo;
        }

        private void button2_Click(object sender, EventArgs e)
        {
        }

        private void btn_YeniHastaEkle_Click(object sender, EventArgs e)
        {
            hastaEkle hastaEkle = new hastaEkle();
            hastaEkle.Show();
        }

        private void btn_YeniRandevuEkle_Click(object sender, EventArgs e)
        {
            yeniRandevu yeniRandevu = new yeniRandevu();
            yeniRandevu.Show();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            hastaSil hastaSil= new hastaSil();
        }
    }
}

```

```

        hastaSil.Show();
    }

    private void sekreterinEkranı_Load(object sender, EventArgs e)
    {

    }

    private void btn_Randevuİptal_Click(object sender, EventArgs e)
    {
        randevuSil randevuSil=new randevuSil();
        randevuSil.Show();
    }
}

```

hastaEkle.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using HastaneTMA.Model;

namespace HastaneTMA
{
    public partial class hastaEkle : Form
    {
        public hastaEkle()
        {
            InitializeComponent();
        }

        private void hastaEkle_Load(object sender, EventArgs e)
        {

        }

        private void btn_YeniHastaEkle_Click(object sender, EventArgs e)
        {
            Model1 TMA_DBContext = new Model1();
            // --- HASTA BİLGİLERİ
            string TCNo = txtBox_TC.Text;
            string Ad=txtBox_Ad.Text;
            string Soyad=txtBox_Soyad.Text;
            bool Cinsiyet;
            if (rdBtn_CinsiyetErkek.Checked)
                Cinsiyet = false;
            else
                Cinsiyet=true;
            string yas = dateTime_Yas.Value.ToString("yyyy");

            // --- ADRES
            string İlce=txtBox_Ilce.Text;
            string Adres=txtBox_Adres.Text;

```

```

string PostaKodu = txtBox_PostaKodu.Text;

//-----
//Patient objesi :

Patient patient=new Patient();
patient.Ad = Ad;
patient.Soyad = Soyad;
patient.TCNo = TCNo;
patient.Cinsiyet = Cinsiyet;
patient.Yas = 2021-Convert.ToInt32(yas);
patient.Ilce = Ilce;
patient.Adres = Adres;
patient.PostaKodu= PostaKodu;

//try catch ekle buraya
try {
    TMA_DBContext.Patients.Add(patient);
    TMA_DBContext.SaveChanges();
    MessageBox.Show("Hasta Eklendi!");
}
catch { MessageBox.Show("Hasta zaten kayıtlı!"); }

txtBox_TC.Clear();
txtBox_Ad.Clear();
txtBox_Soyad.Clear();

txtBox_Ilce.Clear();
txtBox_Adres.Clear();
txtBox_PostaKodu.Clear();

}

private void btn_Iptal_Click(object sender, EventArgs e)
{
    this.Hide();
}
}
}

```

hastaSil.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

using System.Windows.Forms;
using HastaneTMA.Model;

namespace HastaneTMA
{
    public partial class hastaSil : Form
    {
        private Model1 TMA_DBContext;
        public hastaSil()
        {
            InitializeComponent();

            private void hastaSil_Load(object sender, EventArgs e)
            {

            }

            private void btn_Sil_Click(object sender, EventArgs e)
            {
                string TCNo = txtBox_HastaTC.Text;
                TMA_DBContext = new Model1();

                //Patient patient=new Patient() { TCNo=TCNo};

                var table = TMA_DBContext.Patients.ToList();
                var res = table.Where(s => s.TCNo==TCNo).ToList();

                TMA_DBContext.Patients.RemoveRange(res);
                TMA_DBContext.SaveChanges();

                MessageBox.Show("HASTA SİLİNDİ");

            }
        }
    }
}

```

yeniRandevu.cs

```

using HastaneTMA.Model;
using System;
using System.Data;
using System.Linq;
using System.Windows.Forms;
using System.Collections.Generic;
using System.Collections;
using HastaneTMA.Model;

namespace HastaneTMA
{
    public partial class yeniRandevu : Form
    {
        private Model1 TMA_DBContext;
        public yeniRandevu()
        {
            InitializeComponent();

            TMA_DBContext = new Model1();

```

```

        var Clinics = TMA_DBContext.Clinics.ToList();
        foreach (Clinic Clinic in Clinics)
        {
            cbo_Klinik.Items.Add(Clinic.KlinikAdi);
        }
    }

    private void yeniRandevu_Load(object sender, EventArgs e)
    {

    }

    private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
    {
        lstBox_UygunTarihler.Items.Clear();

        //aşağıdaki doktor ID verir
        int dxSelected = dx[cbo_Doktor.SelectedIndex];

        var uygunTarihler = TMA_DBContext.AppointmentHours.Where(u =>
u.DoktorID==dxSelected && u.Status==false).ToList();

        for (int k = 0; k < uygunTarihler.Count; k++)
        {
            var oneRow = uygunTarihler.Skip(k).Take(1).ToList();
            if(oneRow.FirstOrDefault().Hour>=DateTime.Now.AddDays(-1))
                lstBox_UygunTarihler.Items.Add(oneRow.FirstOrDefault().Hour);
        }

    }

    int[] dx;
    private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
        cbo_Doktor.Items.Clear();

        //https://hexacodein.com/makale/linq-include-kullanimi

        var Doctors = TMA_DBContext.Doctors.Include("Employee").Where(d =>
d.KlinikID == cbo_Klinik.SelectedIndex + 1).ToList();
        int boyut = Doctors.Count();

        int i = 0;
        dx=new int[boyut];
        foreach (var d in Doctors)
        {
            cbo_Doktor.Items.Add(d.Employee.Ad.Trim(' ') + " " +
d.Employee.Soyad.Trim(' '));

            dx[i] = d.DoktorID;
            i++;
        }
    }

```

```

    }

    private void dgw_CellContentClick(object sender,
DataGridViewCellEventArgs e)
    {

    }

    private void listView1_SelectedIndexChanged(object sender, EventArgs e)
    {

    }

    private void lstBox_UygunTarihler_SelectedIndexChanged(object sender,
EventArgs e)
    {

    }

    private void btn_RandevuOlustur_Click(object sender, EventArgs e)
    {

        if (!String.IsNullOrEmpty(txtBox_TC.Text) && cbo_Klinik.SelectedItem
!= null && cbo_Doktor.SelectedItem != null && lstBox_UygunTarihler.SelectedItem
!= null)
        {
            Appointment appointment = new Appointment();
            appointment.TCNo = txtBox_TC.Text.ToString();
            appointment.DoktorID = dx[cbo_Doktor.SelectedIndex];
            appointment.KlinikID = cbo_Klinik.SelectedIndex + 1;
            appointment.RandevuSaati =
Convert.ToDateTime(lstBox_UygunTarihler.SelectedItem.ToString());

            var oneRow = TMA_DBContext.AppointmentHours.Where(h => h.Hour ==
appointment.RandevuSaati && h.DoktorID ==
appointment.DoktorID).ToList().FirstOrDefault();
            oneRow.Status = true;
            oneRow.HastaTC = appointment.TCNo;
            oneRow.DoktorID = appointment.DoktorID;

            // TRY CATCH KULLAN HATA ALIRSA İLK ÖNCE HASTAYI SİSTEME KAYIT
ETMEMİZ GEREK
            TMA_DBContext.Appointments.Add(appointment);
            TMA_DBContext.SaveChanges();
            MessageBox.Show("Randevu başarı ile oluşturuldu.");

        }
        else
        {
            MessageBox.Show("Lütfen boş alan bırakmayınız.");
        }

    }

    private void btn_Iptal_Click(object sender, EventArgs e)
    {
        this.Hide();
    }
}

```

```
}
```

randevuSil.cs

```
using HastaneTMA.Model;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Windows.Forms;

namespace HastaneTMA.Sekreter
{
    public partial class randevuSil : Form
    {
        private string TCNo;
        private Model1 TMA_DBContext;
        List<Appointment> randevular;
        public randevuSil()
        {
            InitializeComponent();

            private void txtBox_HastaTC_TextChanged(object sender, EventArgs e)
            {
            }

            private void randevuSil_Load(object sender, EventArgs e)
            {
            }

            private void btn_RandevuSil_Click(object sender, EventArgs e)
            {
                string RandevuSaati = lstBox_Randevular.SelectedItem.ToString();

                using (TMA_DBContext = new Model1())
                {
                    try
                    {
                        var tarih =
Convert.ToDateTime(lstBox_Randevular.SelectedItem.ToString());
                        var kaldir = TMA_DBContext.Appointments.Where(a => a.TCNo ==
TCNo && a.RandevuSaati == tarih).FirstOrDefault();
                        TMA_DBContext.Appointments.Remove(kaldir);
                        TMA_DBContext.SaveChanges();
                        MessageBox.Show("Randevu Başarıyla Silindi!");
                        txtBox_HastaTC.Clear();
                        lstBox_Randevular.Items.Clear();
                    }
                    catch
                    {
                        MessageBox.Show("Bir sorunla karşılaşıldı.");
                    }
                }
            }

            private void btn_RandevulariListele_Click(object sender, EventArgs e)
            {
                lstBox_Randevular.Items.Clear();
                TCNo = txtBox_HastaTC.Text;
            }
        }
    }
}
```

```

        using (TMA_DBContext = new Model1())
        {
            try
            {
                randevular = TMA_DBContext.Appointments.Where(a => a.TCNo ==
TCNo).ToList();
                foreach (var item in randevular)
                    lstBox_Randevular.Items.Add(item.RandevuSaati);

                var AdSoyad = TMA_DBContext.Patients.Where(p => p.TCNo ==
TCNo).FirstOrDefault();
                lbl_HastaAdSoyad.Text = AdSoyad.Ad.Trim() + " " +
AdSoyad.Soyad.Trim() + "\niçin alınmış randevular :";
            }
            catch
            {
                MessageBox.Show("Bir sorunla karşılaşıldı!");
            }
        }
    }
}

```

yoneticininEkrani.cs

```
using HastaneTMA.Yonetici;
```

```
using System;
```

```
using System.Data;
```

```
using System.Linq;
```

```
using System.Windows.Forms;
```

```
using HastaneTMA.Model;
```

```
using System.ComponentModel;
```

```
namespace HastaneTMA.Model
```

```
{
```

```
    public partial class yoneticininEkrani : Form
```

```
    {
```

```
        private Model1 TMA_DBContext = new Model1();
```



```
private int CalisanNo;

public yoneticininEkrani(int CalisanNo)
{
    InitializeComponent();

    this.CalisanNo = CalisanNo;

    //https://www.youtube.com/watch?v=TRH7rm8ozD4&ab_channel=FoxLearn
    // burda veri ekleme silme güncelleme anlatıyor
}

private void label1_Click(object sender, EventArgs e)
{
}

private void yoneticininEkrani_Load(object sender, EventArgs e)
{
    //this.employeesTableAdapter.Fill(this.hastaneTMADataset.Employees);

    BindingSource bi_employees = new BindingSource();

    BindingList <Employee> bi_emList= new BindingList <Employee>();

    foreach (var item in TMA_DBContext.Employees.ToList())
    {
        bi_emList.Add(item);
    }
}
```

```
bi_employees.DataSource = bi_emList;

dgv_Calisanlar.DataSource = bi_employees;

//-----

//this.usersTableAdapter.Fill(this.hastaneTMADatasetUsers.Users);

/*

BindingSource bi_users = new BindingSource();

BindingList<User> bi_usList = new BindingList<User>();

foreach (var item in TMA_DBContext.Users.ToList())

{

    bi_usList.Add(item);

}

bi_users.DataSource = bi_usList;

dgv_.DataSource = bi_users;

*/

//-----

//this.expensesTableAdapter.Fill(this.hastaneTMADataset2.Expenses);

BindingSource bi_expenses = new BindingSource();

BindingList<Expens> bi_exList = new BindingList<Expens>();

foreach (var item in TMA_DBContext.Expenses.ToList())

{

    bi_exList.Add(item);

}

bi_expenses.DataSource = bi_exList;

dgv_Giderler.DataSource = bi_exList;

//-----
```

```
//this.incomesTableAdapter.Fill(this.hastaneTMADataset1.Incomes);  
  
BindingSource bi_incomes = new BindingSource();  
  
BindingList<Income> bi_inList = new BindingList<Income>();  
  
foreach (var item in TMA_DBContext.Incomes.ToList())  
  
{  
  
    bi_inList.Add(item);  
  
}  
  
bi_incomes.DataSource = bi_inList;  
  
dgv_Gelirler.DataSource = bi_inList;  
  
//-----  
  
//this.clinicsTableAdapter.Fill(this.hastaneTMAClinics.Clinics);  
  
BindingSource bi_klinik = new BindingSource();  
  
BindingList<Clinic> bi_klList = new BindingList<Clinic>();  
  
foreach (var item in TMA_DBContext.Clinics.ToList())  
  
{  
  
    bi_klList.Add(item);  
  
}  
  
bi_klinik.DataSource = bi_klList;  
  
dgv_Klinikler.DataSource = bi_klList;  
  
  
}
```

```
private void grpBox_Calisanlar_Enter(object sender, EventArgs e)
```

```
{
```

```
}
```

```
private void grpBox_Klinikler_Enter(object sender, EventArgs e)
```

```
{
```

```
}
```

```
private void refreshData(string tablo)
```

```
{
```

```
    BindingSource bi = new BindingSource();
```

```
    switch (tablo)
```

```
    {
```

```
        case "Clinics":
```

```
            var query_Clinics = from t in TMA_DBContext.Clinics
```

```
                orderby t.ID
```

```
                select new { t.ID, t.KlinikAdi };
```

```
            bi.DataSource = query_Clinics.ToList();
```

```
            dgv_Klinikler.DataSource = bi;
```

```
            dgv_Klinikler.Refresh();
```

```
            break;
```

```
        case "Employees":
```

```
var query_Employees = from t in TMA_DBContext.Employees
                        orderby t.ID
                        select new { t.ID, t.Ad, t.Soyad, t.Cinsiyet, t.DogumTarihi,
t.IseGirisTarihi, t.maas };

bi.DataSource = query_Employees.ToList();

dgv_Calisanlar.DataSource = bi;

dgv_Calisanlar.Refresh();

break;

case "Incomes":

var query_Incomes = from t in TMA_DBContext.Incomes
                    orderby t.ID
                    select new {t.ID,t.GelirAdi,t.Miktar,t.Tarih };

bi.DataSource = query_Incomes.ToList();

dgv_Gelirler.DataSource = bi;

dgv_Gelirler.Refresh();

break;

case "Expenses":

var query_Expenses = from t in TMA_DBContext.Expenses
                    orderby t.ID
                    select new { t.ID, t.GiderAdi, t.Miktar, t.Tarih };

bi.DataSource = query_Expenses.ToList();

dgv_Giderler.DataSource = bi;

dgv_Giderler.Refresh();

break;
```

```
};
```

```
}
```

```
private void btn_KlinikEkle_Click(object sender, EventArgs e)
```

```
{
```

```
    if (txtBox_KlinikAdi != null)
```

```
    {
```

```
        Clinic clinic = new Clinic();
```

```
        clinic.KlinikAdi = txtBox_KlinikAdi.Text;
```

```
        TMA_DBContext.Clinics.Add(clinic);
```

```
        TMA_DBContext.SaveChanges();
```

```
        refreshData("Clinics");
```

```
    }
```

```
    else
```

```
        MessageBox.Show("Lütfen Değer Giriniz!");
```

```
}
```

```
private void btn_CalisanEkle_Click(object sender, EventArgs e)
```

```
{
```

```
    // --- OBJENİN OLUŞTURULMASI VE DATABASE'E YÜKLENMESİ
```

```
if (txtBox_Ad.Text != String.Empty && txtBox_Soyad.Text != String.Empty &&
(rdBtn_Erkek.Checked || rdBtn_Kadin.Checked) && txtBox_Maas.Text !=
String.Empty)
```

```
{
```

```
    Employee employee = new Employee();
```

```
    employee.Ad = txtBox_Ad.Text;
```

```
    employee.Soyad = txtBox_Soyad.Text;
```

```
    employee.Cinsiyet = rdBtn_Erkek.Checked ? false : true;
```

```
    employee.DogumTarihi =
Convert.ToDateTime(dateTime_DogumTarihi.Value.ToString("yyyy-MM-dd"));
```

```
    employee.IseGirisTarihi =
Convert.ToDateTime(DateTime.Now.ToString("yyyy-MM-dd"));
```

```
    employee.maas = int.Parse(txtBox_Maas.Text.ToString());
```

```
    TMA_DBContext.Employees.Add(employee);
```

```
    TMA_DBContext.SaveChanges();
```

```
    refreshData("Employees");
```

```
}
```

```
else
```

```
    MessageBox.Show("Lütfen Boş alanları doldurunuz!");
```

```
txtBox_Ad.Clear();
```

```
txtBox_Soyad.Clear();
```

```
rdBtn_Erkek.Invalidate();
```

```
rdBtn_Kadin.Invalidate();
```

```
        txtBox_Maas.Clear();  
    }
```

```
private void btn_KullaniciBilgileri_Click(object sender, EventArgs e)  
{  
    kullaniciBilgileri kullaniciBilgileri = new kullaniciBilgileri();  
    kullaniciBilgileri.Show();  
}
```

```
private void btn_CalisanSil_Click(object sender, EventArgs e)  
{  
    var calisanID =  
int.Parse(dgv_Calisanlar.CurrentRow.Cells[0].Value.ToString());
```

```
    using (TMA_DBContext = new Model1())  
    {  
        try  
        {  
            var employee = TMA_DBContext.Employees.Where(em => em.ID ==  
calisanID).ToList().FirstOrDefault();  
            TMA_DBContext.Employees.Remove(employee);  
            TMA_DBContext.SaveChanges();  
            refreshData("Employees");  
        }  
        catch (Exception ex)  
        {
```



```
        MessageBox.Show("Bir hata ile karşılaşıldı!");
    }
}

}

private void btn_KullaniciEkle_Click(object sender, EventArgs e)
{
    string kullaniciAdi = txtBox_KullaniciAdi.Text;

    string sifre = txtBox_Sifre.Text;

    int rolNo = int.Parse(txtBox_RolNo.Text);

    int calisanNo = int.Parse(txtBox_CalisanNo.Text);

    try
    {
        User user = new User();

        user.KullaniciAdi = kullaniciAdi;

        user.Sifre = sifre;

        user.RolNo = rolNo;

        user.CalisanNo = calisanNo;

        TMA_DBContext.Users.Add(user);

        TMA_DBContext.SaveChanges();

        MessageBox.Show("Kullanıcı başarılı bir şekilde kaydedildi.");
    }
}
```

```
catch (Exception ex)

{

    MessageBox.Show("Bir hata ile karşılaşıldı!\n" +

        "Lütfen kullanıcı oluşturmadan önce çalışan oluşturunuz.");

}

}
```

```
private void btn_KullaniciSil_Click(object sender, EventArgs e)

{

    int kullaniciID = int.Parse(txtBox_KullaniciSil.Text.ToString());

    using (TMA_DBContext = new Model1())

    {

        try

        {

            var kullanici = TMA_DBContext.Users.Where(ku => ku.ID ==

kullaniciID).ToList().FirstOrDefault();

            TMA_DBContext.Users.Remove(kullanici);

            TMA_DBContext.SaveChanges();

            MessageBox.Show("Kullanıcı başarılı bir şekilde silindi.");

            txtBox_KullaniciSil.Clear();

        }

        catch(Exception ex)

        {
```

```
        MessageBox.Show("Bir hatayla karşılaşıldı!\n"+ex.ToString());
    }
}
}

private void btn_KlinikSil_Click(object sender, EventArgs e)
{

    try
    {
        var klinikID =
int.Parse(dgv_Klinikler.CurrentRow.Cells[0].Value.ToString());

        var klinik = TMA_DBContext.Clinics.Where(k => k.ID ==
klinikID).ToList().FirstOrDefault();

        TMA_DBContext.Clinics.Remove(klinik);

        TMA_DBContext.SaveChanges();

        MessageBox.Show("Başarıyla Silindi!");

        refreshData("Clinics");
    }
    catch
    {
        MessageBox.Show("Bir hatayla karşılaşıldı!");
    }
}
```

```
}
```

```
private void btn_GelirEkle_Click(object sender, EventArgs e)
```

```
{
```

```
    try
```

```
    {
```

```
        string gelirAdi = txtBox_GelirAdi.Text;
```

```
        int gelirMiktar = int.Parse(txtBox_GelirMiktar.Text.ToString());
```

```
        DateTime gelirTarih = DateTime.Now;
```

```
        Income income = new Income();
```

```
        income.GelirAdi = gelirAdi;
```

```
        income.Miktar = gelirMiktar;
```

```
        income.Tarih = gelirTarih;
```

```
        TMA_DBContext.Incomes.Add(income);
```

```
        TMA_DBContext.SaveChanges();
```

```
        MessageBox.Show("Başarıyla Eklendi!");
```

```
        refreshData("Incomes");
```

```
    }
```

```
    catch
```

```
    {
```

```
        MessageBox.Show("Bir hatayla karşılaşıldı!\n" +
```

```
            "Hata nedeni istenmeyen tipte veri girilmesi olabilir.");
```

```
}
```

```
}
```

```
private void btn_GelirSil_Click(object sender, EventArgs e)
```

```
{
```

```
try
```

```
{
```

```
int GelirID = int.Parse(dgv_Gelirler.CurrentRow.Cells[0].Value.ToString());
```

```
var Gelir = TMA_DBContext.Incomes.Where(income => income.ID ==  
GelirID).ToList().FirstOrDefault();
```

```
TMA_DBContext.Incomes.Remove(Gelir);
```

```
TMA_DBContext.SaveChanges();
```

```
MessageBox.Show("Başarıyla Silindi!");
```

```
refreshData("Incomes");
```

```
}
```

```
catch (Exception ex)
```

```
{
```

```
MessageBox.Show("Bir hatayla karşılaşıldı!\n" + ex.ToString());
```

```
}
```

```
}
```

```
private void btn_GiderSil_Click(object sender, EventArgs e)
```

```
{
```

```

try
{
    int GiderID = int.Parse(dgv_Giderler.CurrentRow.Cells[0].Value.ToString());

    var Gider = TMA_DBContext.Expenses.Where(expense => expense.ID ==
GiderID).ToList().FirstOrDefault();

    TMA_DBContext.Expenses.Remove(Gider);

    TMA_DBContext.SaveChanges();

    MessageBox.Show("Başarıyla Silindi!");

    refreshData("Expenses");
}
catch (Exception ex)
{
    MessageBox.Show("Bir hatayla karşılaşıldı!\n" + ex.ToString());
}
}

```

```

private void btn_GiderEkle_Click(object sender, EventArgs e)

```

```

{

    try
    {

        string giderAdi = txtBox_GiderAdi.Text;

        int giderMiktar = int.Parse(txtBox_GiderMiktar.Text.ToString());

        DateTime giderTarih = DateTime.Now;
    }
}

```

```
    Expens expens = new Expens();

    expens.GiderAdi = giderAdi;

    expens.Miktar = giderMiktar;

    expens.Tarih = giderTarih;

    TMA_DBContext.Expenses.Add(expens);

    TMA_DBContext.SaveChanges();

    MessageBox.Show("Başarıyla Eklendi!");

    refreshData("Expenses");

}

catch

{

    MessageBox.Show("Bir hatayla karşılaşıldı!\n" +

        "Hata nedeni istenmeyen tipte veri girilmesi olabilir.");

}

}

private void txtBox_KullaniciSil_TextChanged(object sender, EventArgs e)

{

}

private void button1_Click(object sender, EventArgs e)

{
```

```
}
```

```
private void btn_DoktorIliskilendir_Click(object sender, EventArgs e)
```

```
{
```

```
    doktorIliskilendir doktorIliskilendir = new doktorIliskilendir();
```

```
    doktorIliskilendir.Show();
```

```
}
```

```
private void dgv_Calisanlar_CellContentClick(object sender,  
DataGridViewCellEventArgs e)
```

```
{
```

```
}
```

```
}
```

```
}
```

kullaniciBilgileri.cs

```
using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Drawing;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.Windows.Forms;  
using HastaneTMA.Model;  
  
namespace HastaneTMA.Yonetici  
{  
    public partial class kullaniciBilgileri : Form  
    {  
        private Model1 TMA_DBContext=new Model1();  
        public kullaniciBilgileri()  
        {  
            InitializeComponent();  
        }  
    }  
}
```



```

    }

    private void kullaniciBilgileri_Load(object sender, EventArgs e)
    {
        //this.usersTableAdapter.Fill(this.hastaneTMADatasetUsers2.Users);
        BindingSource bi_users = new BindingSource();
        BindingList<User> bi_usList = new BindingList<User>();
        foreach (var item in TMA_DBContext.Users.ToList())
        {
            bi_usList.Add(item);
        }
        bi_users.DataSource = bi_usList;
        dataGridView1.DataSource = bi_users;
    }

    private void btn_KullaniciyiSil_Click(object sender, EventArgs e)
    {
    }

    private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
    {
    }
}

```

doktorIliskilendir.cs

```

using HastaneTMA.Model;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Windows.Forms;

namespace HastaneTMA.Yonetici
{
    public partial class doktorIliskilendir : Form
    {
        private Model1 TMA_DBContext;
        public doktorIliskilendir()
        {
            InitializeComponent();
        }

        private void doktorIliskilendir_Load(object sender, EventArgs e)
        {
            this.clinicsTableAdapter.Fill(this.hastaneTMADataset3.Clinics);

            using (TMA_DBContext = new Model1())
            {
                var calisanIDsFromUsers=TMA_DBContext.Users.Where(u => u.RolNo ==
2).ToList();

                List<Employee> employees=new List<Employee> ();
            }
        }
    }
}

```

```

        foreach (var item in calisanIDsFromUsers)
        {
            dgv_DoktorListesi.Rows.Add(item.Employee.ID,
            item.Employee.Ad, item.Employee.Soyad);
        }
    }

    private void dgv_DoktorListesi_CellContentClick(object sender,
DataGridViewCellEventArgs e)
    {
    }

    private void btn_Iliskilendir_Click(object sender, EventArgs e)
    {
        using (TMA_DBContext = new Model1())
        {
            try
            {
                int DoctorID =
                int.Parse(dgv_DoktorListesi.CurrentRow.Cells[0].Value.ToString());
                int KlinikID =
                int.Parse(dgv_KlinikListesi.CurrentRow.Cells[0].Value.ToString());

                Doctor doctor = new Doctor();
                doctor.DoktorID = DoctorID;
                doctor.KlinikID = KlinikID;

                TMA_DBContext.Doctors.Add(doctor);
                TMA_DBContext.SaveChanges();
                MessageBox.Show("Başarı ile kaydedildi!");
            }
            catch (Exception ex)
            {
                MessageBox.Show("Bir hata ile karşılaşıldı!\n" +
                ex.ToString());
            }
        }
    }
}

```

NOT : Model dosyalarına proje kaynak kodlarından erişebilirsiniz. Rar dosyasının içerisinde bulabilirsiniz.