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

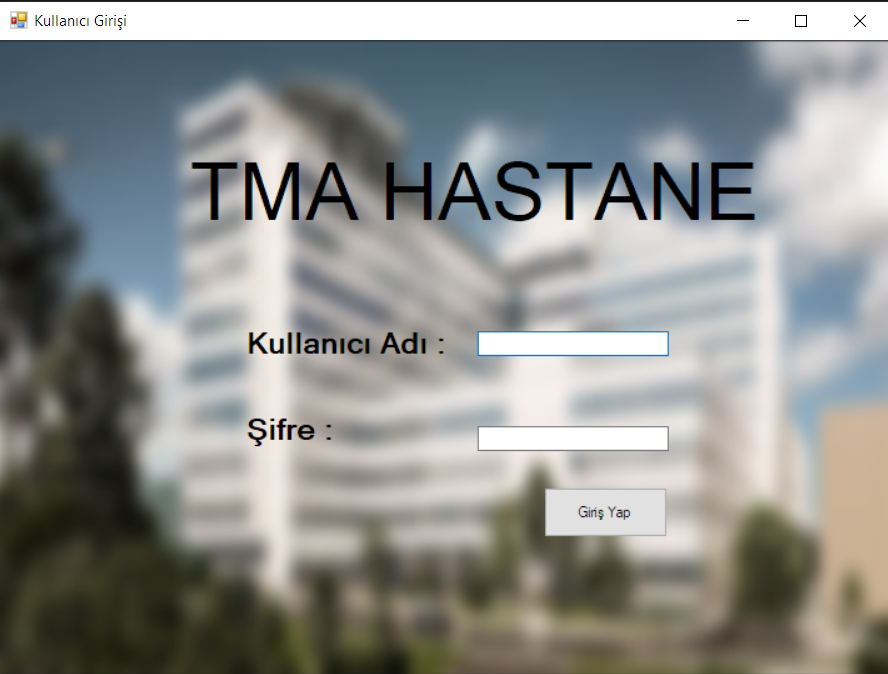
1. Problemin Tanımlanması /Programın Amacı

Yalnızca hastane içerisinde kullanılabilecek 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

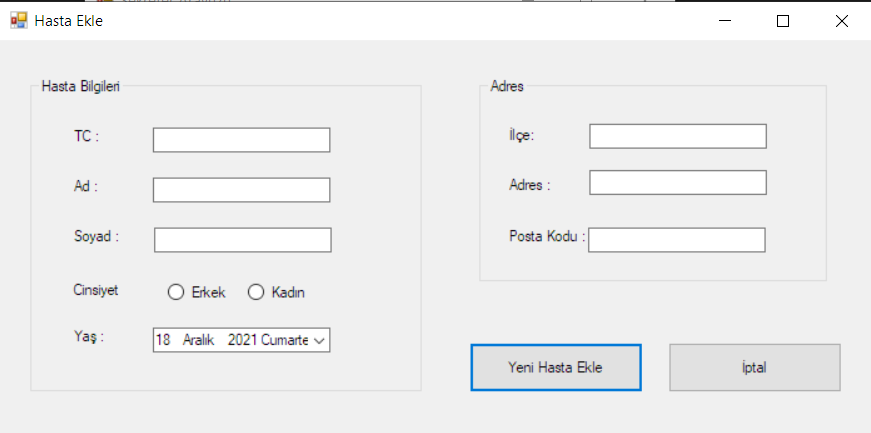
Sekreterin ekranı aşağıda gösterildiği şekildedir;

metin, ekran görüntüsü içeren bir resim

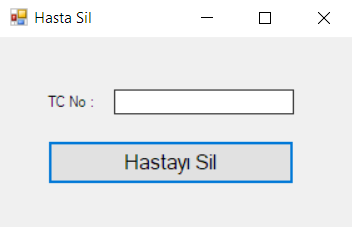
Açıklama otomatik olarak oluşturuldu

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

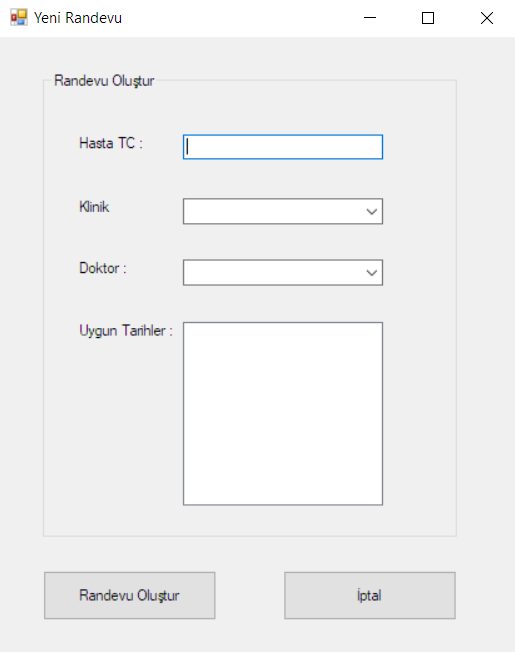
Yeni hasta eklemek için hastanın T.C. kimlik numarası, adı, soyadı, cinsiyeti, doğum tarihi ve adres bilgilerinin sekreter tarafından girilmesi istenir. Ardından sağ altta yer alan Yeni Hasta Ekle butonuyla kayıt sağlanır ya da yanında bulunan iptal butonuyla da sayfa kapatılır.



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 alınar 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.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

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.

metin, ekran görüntüsü, iç mekan içeren bir resim

Açıklama otomatik olarak oluşturuldu

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

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

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.

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

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 hücresine tıklayıp, ardından butona bastığında açılan ekran sayesinde hastanın durumunu, kullanması gereken ilaçları veya uygulayacağı tedaviyi kaydetmesi sağlanacaktır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

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.

metin içeren bir resim

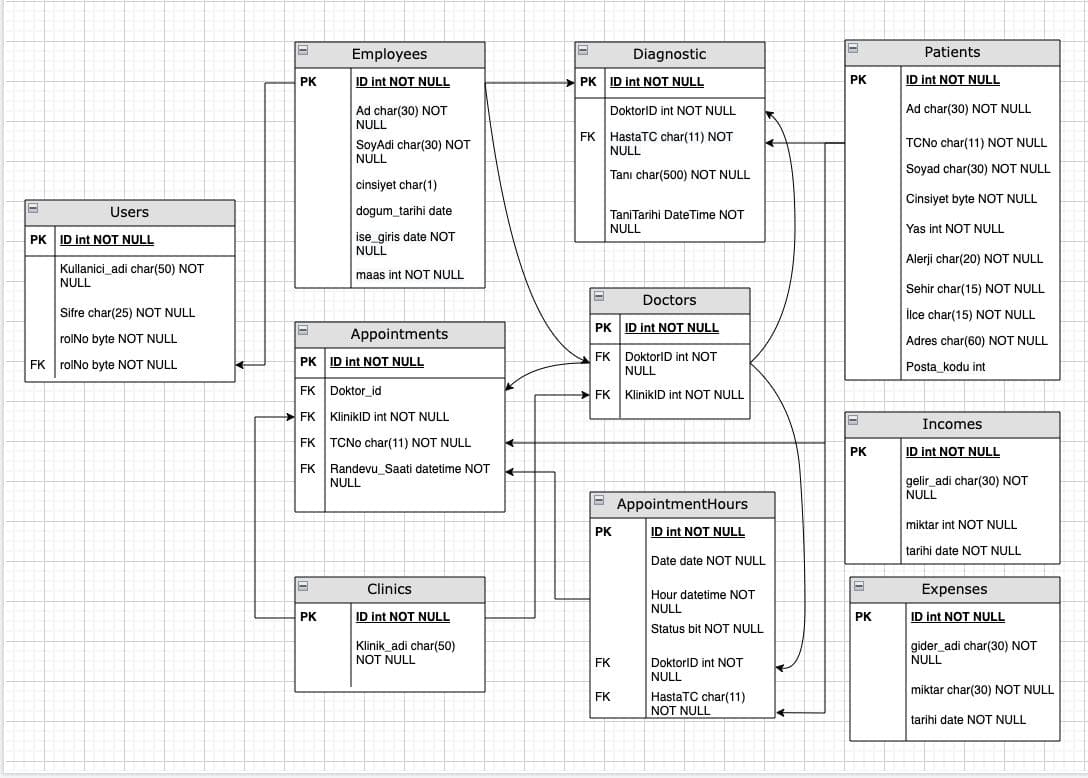
Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

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.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

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.

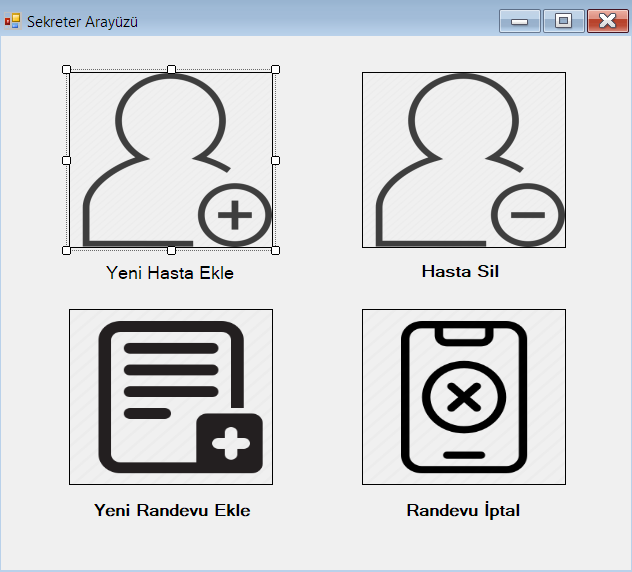
metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

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

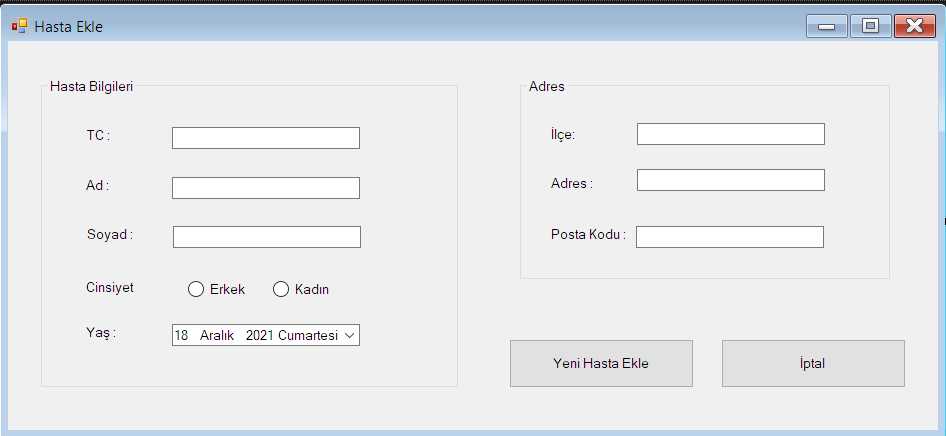
metin içeren bir resim

Açıklama otomatik olarak oluşturuldu1-)Sekreter:



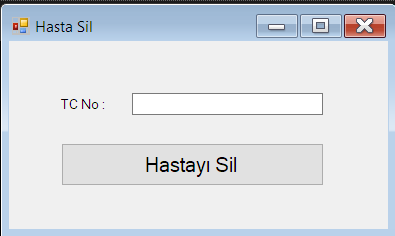
Bu ekranda sadece gerekli pencerelere yönlendirme yapılıyor.

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.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

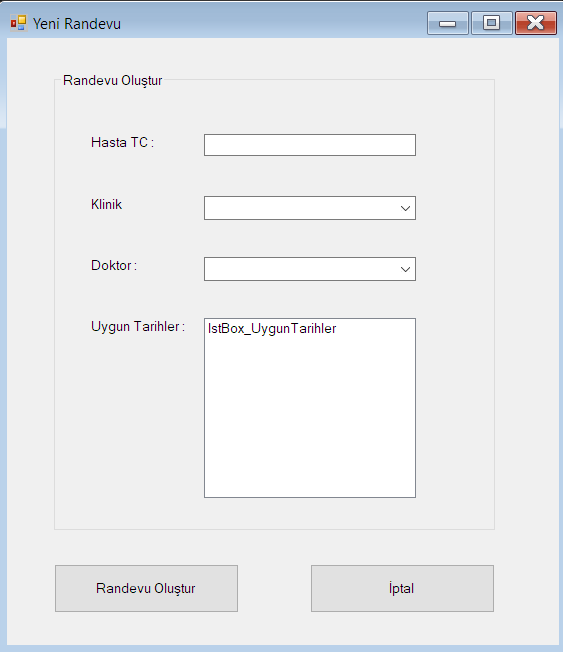
Hasta silme ekranında ise hastaya ait TC no gerekir. Bu TC no patients tablosunda aranıp hasta silinir. Hasta silinince hastaya ait tüm diğer verilerde silinir. Bunun nedeni ise Database’de foreign keyleri tanımlarken cascade deleting fonksiyonunun aktif edilmiş olmasıdır.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

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.

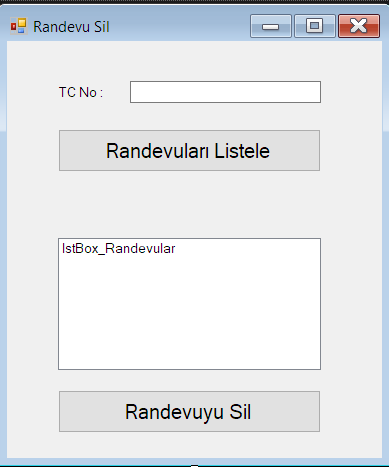
Klinik seçildikten sonra arka planda klinik combobox’ının indexi alınır ve doktorlar tablosundaki KlinikID sütunu, klinik combobox indexine eşit olanlar doktor combobox’ında gösterilir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

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.

 metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

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.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

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.

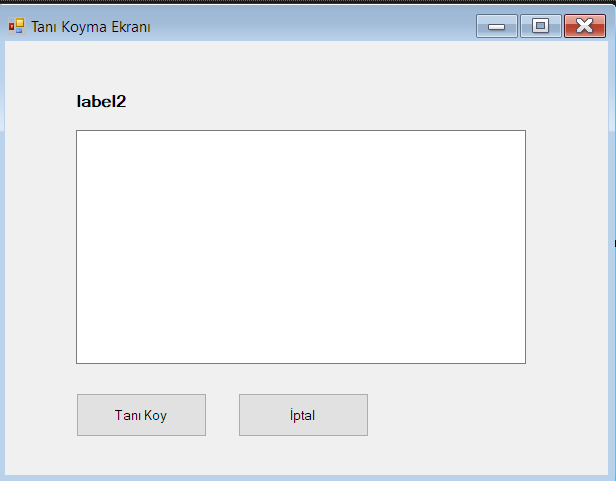
İkinci using bloğunda ise doktora ait hastaların bilgilerini görüntülüyoruz. Elde ettiğimiz randevu bilgilerinin listesini for döngüsüne sokuyoruz. İ değişkeni ile atlanacak satırı belirtip satırı take fonksiyonu ile alıyoruz. Bu bize doktora ait tek bir randevuyu veriyor. Bu satır ile datagridview’ın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

metin içeren bir resim

Açıklama otomatik olarak oluşturuldumetin içeren bir resim

Açıklama otomatik olarak oluşturuldu

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

 metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, ekran görüntüsü, ekran, kapat içeren bir resim

Açıklama otomatik olarak oluşturuldu

Aynı şekilde yükleme yapıldığı gibi veriler çekilerek gösterimi sağlanır.

3-)Yönetici:

metin, ekran görüntüsü, iç mekan içeren bir resim

Açıklama otomatik olarak oluşturuldu

Yönetici ekranı açılır açılmaz veri yüklemesi şu şekilde yapılmaktadır;

metin, iç mekan içeren bir resim

Açıklama otomatik olarak oluşturuldu

Çalışanlar tablosuna çalışan ekleme;

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

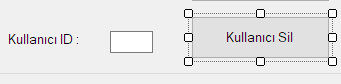
Kullanıcılar tablosuna kullanıcı ekleme;

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

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

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

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

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

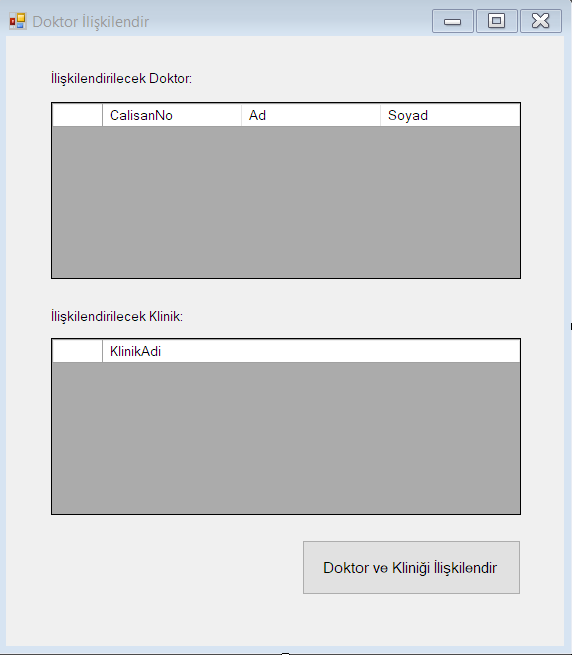
Datagridview’da seçilen satırın id’si klinikID değişkenine atanır. Klinik değişkeni Klinikler tablosunda KlinikID’ye eşit olan kayıda denk gelir. Bu kayıt Context yardımı ile silinir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

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 CalışanNo 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 olduğunun bilgisini veriyor. Bu 2 veriyi doctor objesi oluşturup Context yardımı ile veri tabanına yüklüyoruz.

 metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

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.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

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 CONNECTİON STRİNG 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 KARAKTER

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)

{

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 sekreterinEkrani\_Load(object sender, EventArgs e)

{

}

private void btn\_RandevuIptal\_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 Ilce=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.**