



T.C. DÜZCE ÜNİVERSİTESİ MÜHENDİSLİK
FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BM309-Veritabanı Yönetim Sistemleri

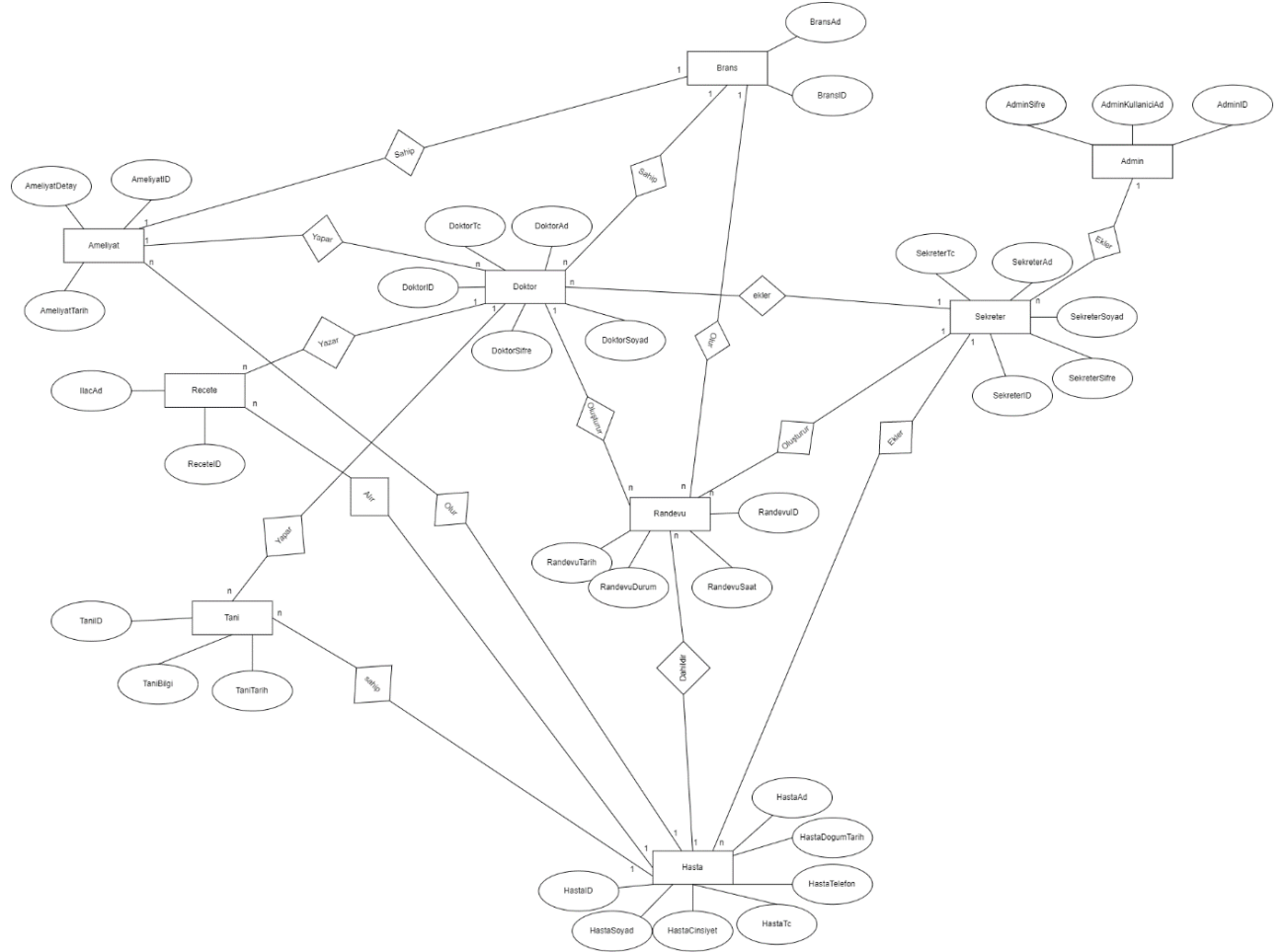
Konu: Bir hastanede yapılan muayene, reçete, ameliyat... gibi işlemlerin takibini yapmak amaçlı kullanılacak bir veritabanı projesi

Teslim Eden:

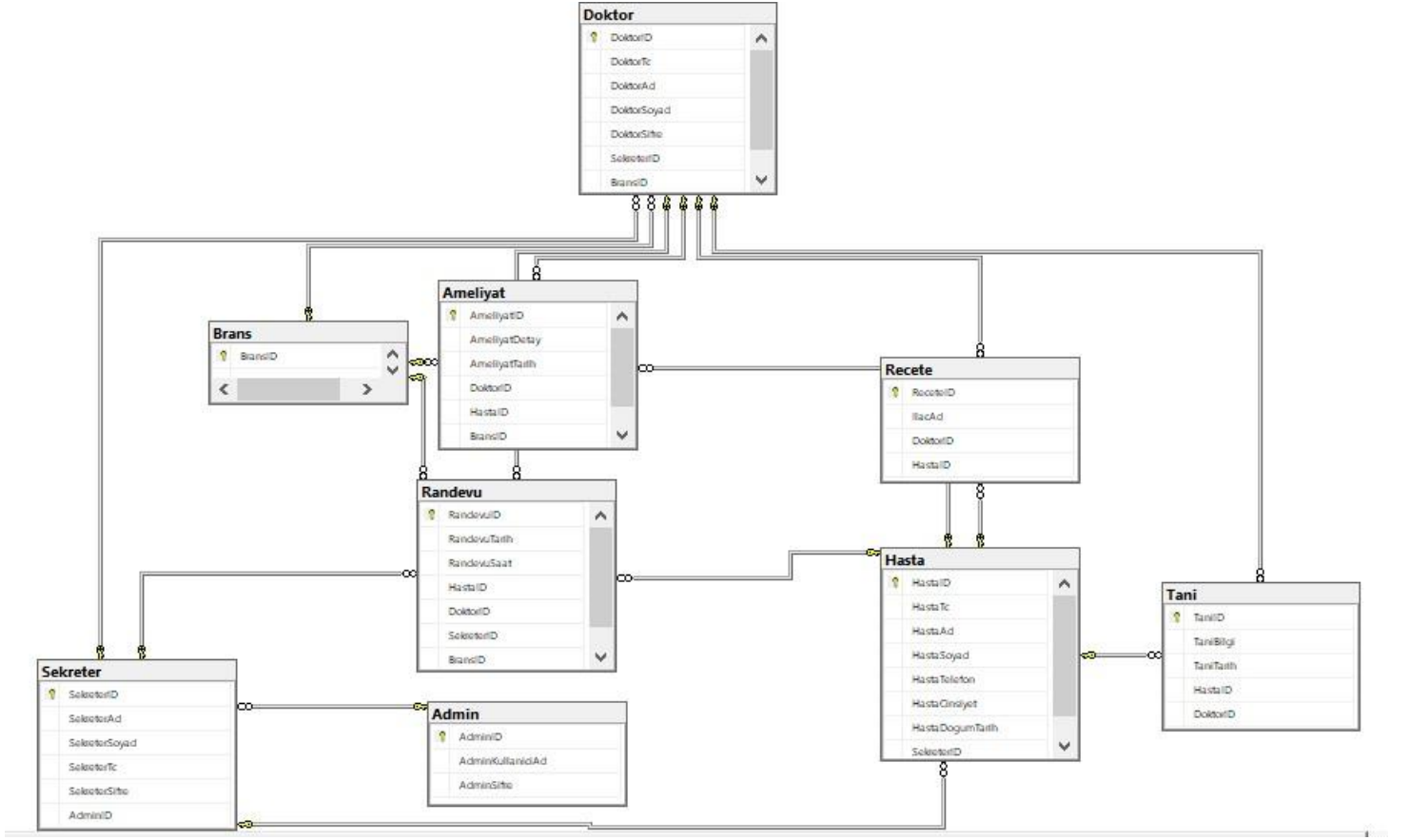
Ad Soyad	Öğrenci Numarası	Öğretim Türü
Büşra Özdemir	201001082	Normal Öğretim
Sudenur Çelik	201001018	Normal Öğretim
Murat Civek	191001045	Normal Öğretim
Muhammet Emin Üçkan	191001054	Normal Öğretim

ER DİYAGRAMLARI

Varlık- İlişkilerinin Tanımlanması



VERİTABANI ŞEMASI



VERİTABANI TABLOLARI

Column Name	Data Type	Allow Nulls
AdminID	smallint	<input type="checkbox"/>
AdminKullaniciAd	varchar(20)	<input checked="" type="checkbox"/>
AdminSifre	varchar(10)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Column Name	Data Type	Allow Nulls
AmeliyatID	smallint	<input type="checkbox"/>
AmeliyatDetay	varchar(500)	<input checked="" type="checkbox"/>
AmeliyatTarih	date	<input checked="" type="checkbox"/>
DoktorID	smallint	<input checked="" type="checkbox"/>
HastaID	smallint	<input checked="" type="checkbox"/>
BransID	smallint	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Column Name	Data Type	Allow Nulls
BransID	smallint	<input type="checkbox"/>
BransAd	varchar(100)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Column Name	Data Type	Allow Nulls
BransID	smallint	<input type="checkbox"/>
BransAd	varchar(100)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Column Name	Data Type	Allow Nulls
DoktorID	smallint	<input type="checkbox"/>
DoktorTc	char(11)	<input checked="" type="checkbox"/>
DoktorAd	varchar(30)	<input checked="" type="checkbox"/>
DoktorSoyad	varchar(30)	<input checked="" type="checkbox"/>
DoktorSifre	varchar(10)	<input checked="" type="checkbox"/>
SekreterID	smallint	<input checked="" type="checkbox"/>
BransID	smallint	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Column Name	Data Type	Allow Nulls
DoktorID	smallint	<input type="checkbox"/>
DoktorTc	char(11)	<input checked="" type="checkbox"/>
DoktorAd	varchar(30)	<input checked="" type="checkbox"/>
DoktorSoyad	varchar(30)	<input checked="" type="checkbox"/>
DoktorSifre	varchar(10)	<input checked="" type="checkbox"/>
SekreterID	smallint	<input checked="" type="checkbox"/>
BransID	smallint	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Column Name	Data Type	Allow Nulls
HastaID	smallint	<input type="checkbox"/>
HastaTc	char(11)	<input checked="" type="checkbox"/>
HastaAd	varchar(30)	<input checked="" type="checkbox"/>
HastaSoyad	varchar(30)	<input checked="" type="checkbox"/>
HastaTelefon	varchar(15)	<input checked="" type="checkbox"/>
HastaCinsiyet	varchar(10)	<input checked="" type="checkbox"/>
HastaDogumTarih	date	<input checked="" type="checkbox"/>
SekreterID	smallint	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Column Name	Data Type	Allow Nulls
HastaID	smallint	<input type="checkbox"/>
HastaTc	char(11)	<input checked="" type="checkbox"/>
HastaAd	varchar(30)	<input checked="" type="checkbox"/>
HastaSoyad	varchar(30)	<input checked="" type="checkbox"/>
HastaTelefon	varchar(15)	<input checked="" type="checkbox"/>
HastaCinsiyet	varchar(10)	<input checked="" type="checkbox"/>
HastaDogumTarih	date	<input checked="" type="checkbox"/>
SekreterID	smallint	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Column Name	Data Type	Allow Nulls
RandevuID	smallint	<input type="checkbox"/>
RandevuTarih	date	<input checked="" type="checkbox"/>
RandevuSaat	time(7)	<input checked="" type="checkbox"/>
HastaID	smallint	<input checked="" type="checkbox"/>
DoktorID	smallint	<input checked="" type="checkbox"/>
SekreterID	smallint	<input checked="" type="checkbox"/>
BransID	smallint	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Column Name	Data Type	Allow Nulls
RandevuID	smallint	<input type="checkbox"/>
RandevuTarih	date	<input checked="" type="checkbox"/>
RandevuSaat	time(7)	<input checked="" type="checkbox"/>
HastaID	smallint	<input checked="" type="checkbox"/>
DoktorID	smallint	<input checked="" type="checkbox"/>
SekreterID	smallint	<input checked="" type="checkbox"/>
BransID	smallint	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Column Name	Data Type	Allow Nulls
SekreterID	smallint	<input type="checkbox"/>
SekreterAd	varchar(30)	<input checked="" type="checkbox"/>
SekreterSoyad	varchar(30)	<input checked="" type="checkbox"/>
SekreterTc	char(11)	<input checked="" type="checkbox"/>
SekreterSifre	varchar(10)	<input checked="" type="checkbox"/>
AdminID	smallint	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Column Name	Data Type	Allow Nulls
SekreterID	smallint	<input type="checkbox"/>
SekreterAd	varchar(30)	<input checked="" type="checkbox"/>
SekreterSoyad	varchar(30)	<input checked="" type="checkbox"/>
SekreterTc	char(11)	<input checked="" type="checkbox"/>
SekreterSifre	varchar(10)	<input checked="" type="checkbox"/>
AdminID	smallint	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Column Name	Data Type	Allow Nulls
ReceteID	smallint	<input type="checkbox"/>
IlacAd	varchar(50)	<input checked="" type="checkbox"/>
DoktorID	smallint	<input checked="" type="checkbox"/>
HastaID	smallint	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Column Name	Data Type	Allow Nulls
TaniID	smallint	<input type="checkbox"/>
TaniBilgi	varchar(500)	<input checked="" type="checkbox"/>
TaniTarih	date	<input checked="" type="checkbox"/>
HastaID	smallint	<input checked="" type="checkbox"/>
DoktorID	smallint	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

- Admin sekreter ekleme/silme/güncelleme kabiliyetinde olduğu için aralarında bire çok ilişki vardır. Yani admin birden fazla sekreter ekleyebilir.
- Sekreter branş, hasta, doktor için ekleme/silme/güncelleme işlemlerini yapabilir. Sekreter bir veya birden fazla hastaya randevu oluşturabilir, veya randevuyu iptal edebilir. Bu yüzden Sekreter tablosu Branş, Hasta, Doktor ve Randevu tablolarıyla aralarında bire çok ilişki vardır.
- Doktor bir veya birden fazla hastaya tanı, reçete oluşturup eğer gerekiyorsa ameliyat randevusu da oluşturabilir. Doktor tablosuyla Tanı, Reçete ve Ameliyat tablosu arasında bire çok ilişki vardır.
- Admin, Sekreter veya Doktor yetkisine sahip kullanıcı herhangi bir kayıt sildiğinde silinen kayıda ait isimdeki ilgili tabloya (....._Silinenler) silinmiş kayıtlar kaydolmaktadır.



Şekil 1: Veritabanı Tablo Yapısı

VERİTABANI KODLARI, DEFAULT VE CONSRİİNTLER

```
CREATE DATABASE HastaneOtomasyon

CREATE TABLE Admin(
    AdminID smallint primary key not null identity(1,1),
    AdminKullaniciAd varchar(20),
    AdminSifre varchar(10)
)

CREATE TABLE Brans(
    BransID smallint primary key not null identity(1,1),
    BransAd varchar(100)
)

CREATE TABLE Sekreter(
    SekreterID smallint primary key not null identity(1,1),
    SekreterAd varchar(30),
    SekreterSoyad varchar(30),
    SekreterTc char(11),
    SekreterSifre varchar(10),
    AdminID smallint foreign key references Admin(AdminID)
)

CREATE TABLE Doktor(
    DoktorID smallint primary key not null identity(1,1),
    DoktorTc char(11),
    DoktorAd varchar(30),
    DoktorSoyad varchar(30),
    DoktorSifre varchar(10),
    SekreterID smallint foreign key references Sekreter(SekreterID),
    BransID smallint foreign key references Brans(BransID)
)
```



```

Create table Hasta(
HastaID smallint primary key not null identity(1,1),
HastaTc char(11),
HastaAd varchar(30),
HastaSoyad varchar(30),
HastaTelefon varchar(15),
HastaCinsiyet varchar(5),
HastaDogumTarih date
Constraint ck_HastaDogumTarih check(HastaDogumTarih<getdate()),
SekreterID smallint foreign key references Sekreter(SekreterID)
)

Create table Ameliyat(
AmeliyatID smallint primary key not null identity(1,1),
AmeliyatDetay varchar(500) default ('Tani Girilmedi.'),
AmeliyatTarih date
Constraint ck_AmeliyatTarih check(AmeliyatTarih>=getdate()),
DoktorID smallint foreign key references Doktor(DoktorID),
HastaID smallint foreign key references Hasta(HastaID),
BransID smallint foreign key references Brans(BransID)
)

Create table Randevu(
RandevuID smallint primary key not null identity(1,1),
RandevuTarih date
Constraint ck_RandevuTarih check(RandevuTarih>=getdate()),
RandevuSaat time(7),
HastaID smallint foreign key references Hasta(HastaID),
DoktorID smallint foreign key references Doktor(DoktorID),
SekreterID smallint foreign key references Sekreter(SekreterID),
BransID smallint foreign key references Brans(BransID)
)

Create table Recete(
ReceteID smallint primary key not null identity(1,1),
IlacAd varchar(50),
DoktorID smallint foreign key references Doktor(DoktorID),
HastaID smallint foreign key references Hasta(HastaID)
)

Create table Tani(
TaniID smallint primary key not null identity(1,1),
TaniBilgi varchar(500),
TaniTarih date
Constarint ck_TaniTarih check(TaniTarih<=getdate()),
HastaID smallint foreign key references Hasta(HastaID),
DoktorID smallint foreign key references Doktor (DoktorID)
)

```

```

Create table Brans_Silinenler(
    BransID smallint primary key not null identity(1,1),
    BransAd varchar(100)
)

Create table Sekreter_Silinenler(
    SekreterID smallint primary key not null identity(1,1),
    SekreterAd varchar(30),
    SekreterSoyad varchar(30),
    SekreterTc char(11),
    SekreterSifre varchar(10),
    AdminID smallint foreign key references Admin(AdminID)
)

Create table Doktor_Silinenler(
    DoktorID smallint primary key not null identity(1,1),
    DoktorTc char(11),
    DoktorAd varchar(30),
    DoktorSoyad varchar(30),
    DoktorSifre varchar(10),
    SekreterID smallint foreign key references Sekreter(SekreterID),
    BransID smallint foreign key references Brans(BransID)
)

Create table Hasta_Silinenler(
    HastaID smallint primary key not null identity(1,1),
    HastaTc char(11),
    HastaAd varchar(30),
    HastaSoyad varchar(30),
    HastaTelefon varchar(15),
    HastaCinsiyet varchar(5),
    HastaDogumTarih date
    Constraint ck_HastaDogumTarih check(HastaDogumTarih<getdate()),
    SekreterID smallint foreign key references Sekreter(SekreterID)
)

Create table Randevu_Silinenler(
    RandevuID smallint primary key not null identity(1,1),
    RandevuTarih date
    Constraint ck_RandevuTarih check(RandevuTarih>=getdate()),
    RandevuSaat time(7),
    HastaID smallint foreign key references Hasta(HastaID),
    DoktorID smallint foreign key references Doktor(DoktorID),
    SekreterID smallint foreign key references Sekreter(SekreterID),
    BransID smallint foreign key references Brans(BransID)
)

```


INDEXLER

```
Indexs.sql - DESK...-G8IS3SP\ömer (59)) + X
Create Unique Index Clst_DoktorTc
ON Doktor(DoktorTc)

Create Unique Index Clst_HastaTc
ON Hasta(HastaTc)

Create Unique Index Clst_SekreterTc
ON Sekreter(SekreterTc)
```

Şekil 2: Tanımlanan Index'ler

```
Create Unique Index Clst_DoktorTc
ON Doktor(DoktorTc)

Insert Into Doktor(DoktorTc,DoktorAd) Values('3333333333','Büşra')
```

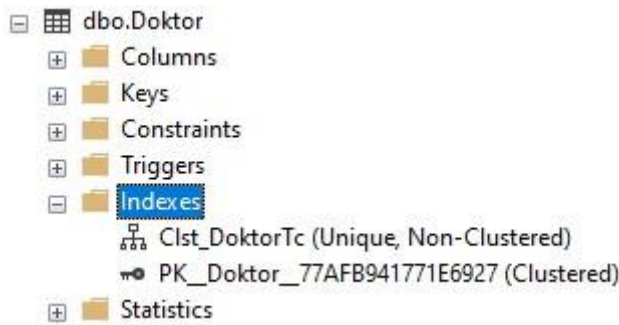
100 %

Messages

Msg 2601, Level 14, State 1, Line 4
Cannot insert duplicate key row in object 'dbo.Doktor' with unique index 'Clst_DoktorTc'. The duplicate key value is (3333333333). The statement has been terminated.

Completion time: 2023-01-08T13:35:04.5238016+03:00

Şekil 3: Doktor tablosunda DoktorTc alanına eklenen TC'lerin benzersiz olması için Unique Index tanımladık. Yukarıda da örnek kayıt girişi sırasında tabloda var olan bir TC eklendiğinde hata verdiğini gördük.



Şekil 4: Doktor tablosuna ait tanımlı olan Index'ler

TRIGGERLAR

```
CREATE TRIGGER trTaniTarihGuncelle
ON Tani
FOR INSERT
AS
    DECLARE @tanitarih date
    SELECT @tanitarih=TaniTarih From inserted
    IF (@tanitarih=NULL or @tanitarih=' ' )
    BEGIN
        DECLARE @taniId int
        SELECT @taniId=TaniID From Tani
        UPDATE Tani Set TaniTarih=GETDATE()
        WHERE TaniID=@taniId
    END

--Tani tablosuna ekleme yaptığımızda tarih girmezsek varsayılan olarak sistem tarihini atayacak
INSERT Tani (TaniBilgi,TaniTarih) VALUES('baş ağrısı',' ')
```

Şekil 5: Tani tablosuna herhangi bir kayıt eklediğimizde eğer TaniTarih alanını girmezsek otomatik sistem tarihini atayacak olan trigger.

```
CREATE TRIGGER trRandevuTarihKontrol
ON Randevu
INSTEAD OF INSERT
AS
    DECLARE @randevutarih date
    SELECT @randevutarih=RandevuTarih from INSERTED
    IF(@randevutarih<getdate())
    BEGIN
        RAISERROR('Randevu tarihi sadece bugün veya daha ileri bir tarihi içerebilir!',0,1)
        ROLLBACK
    END

INSERT INTO Randevu(RandevuTarih,RandevuSaat) Values ('2022-08-20','11:30')
```

100 %

Messages

Randevu tarihi sadece bugün veya daha ileri bir tarihi içerebilir!
Msg 3609, Level 16, State 1, Line 20
The transaction ended in the trigger. The batch has been aborted.

Şekil 6: Randevu tablosuna sistem tarihinden önce bir tarih değeri girilirse ‘Randevu tarihi sadece bugün veya daha ileri bir tarihi içerebilir!’ uyarısını bize vererek kaydı eklememizi engelleyecek olan trigger.

```
CREATE TRIGGER trHastaBilgiSil
ON Hasta
FOR DELETE
AS
    DECLARE @hastaaid int
    DECLARE @hastatc char(11)
    DECLARE @hastaad varchar(20)
    DECLARE @hastasoyad varchar(20)
    SELECT @hastaaid=HastaID from DELETED
    SELECT @hastatc=HastaTc from DELETED
    SELECT @hastaad=HastaAd from DELETED
    SELECT @hastasoyad=HastaSoyad from DELETED
    PRINT cast(@hastatc as char(11)) + ' TC li ' + cast(@hastaad as varchar(20))+
    cast(@hastasoyad as varchar(20))+ ' adındaki hasta silinmiştir.'

INSERT Hasta (HastaTc,HastaAd,HastaSoyad) VALUES('0000000000','Defne','Yıldırım')

DELETE FROM Hasta Where HastaTc='0000000000'
```

100 %

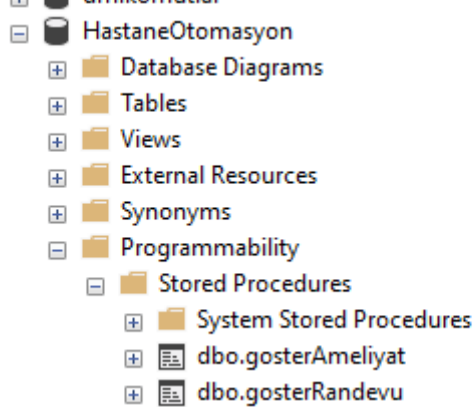
Messages

0000000000 TC li DefneYıldırım adındaki hasta silinmiştir.

(1 row affected)

Şekil 7: Hasta tablosuna herhangi kayıt ekleyip ardından kaydı TC'sine göre silince silinmiş olduğu mesajını gösteren trigger.

STORED PROCEDURE



Şekil 8: Veri tabanımızda kullandığımız stored procedureler.

```
CREATE PROCEDURE [dbo].[gosterRandevu]
AS
    select * from Randevu
GO
```

Şekil 9: gosterRandevu stored procedure.

```

CREATE PROCEDURE [dbo].[gosterAmeliyat]
AS
select * from Ameliyat
GO

```

Şekil 10: gosterAmeliyat stored procedure.

```

1 reference
private void btnRaporla_Click(object sender, EventArgs e)
{
    DataTable dt = new DataTable();
    SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("gosterAmeliyat", baglan.baglanti());
    sqlDataAdapter.Fill(dt);
    dtgrdAmeliyat.DataSource = dt;
}
1 reference
private void btnRandevuRaporla_Click(object sender, EventArgs e)
{
    DataTable dt = new DataTable();
    SqlDataAdapter sqlDataAdapter = new SqlDataAdapter("gosterRandevu", baglan.baglanti());
    sqlDataAdapter.Fill(dt);
    dtgrdRandevu.DataSource = dt;
}

```

Şekil 11: Stored procedureleri kod kısmında kullandığımız alan.

RAPORLAMA İŞLEMLERİ

Ameliyat Bilgi

	AmeliyatID	AmeliyatDe	AmeliyatTa	DoktorID	HastaID	BransID
1	1	Mide Has...	4.11.20...	1	10	10
2	2	Apandisit	2.02.20...	8	9	6
3	3	Bel ve Bo...	28.01.2...	21	16	7
4	4	Beyin Ka...	8.03.20...	18	22	8
5	5	Parkinson	6.01.20...	10	24	16
6	6	Burun a...	7.05.20...	13	16	18
7	7	El ayak ş...	18.02.2...	17	2	5

Raporla PDF

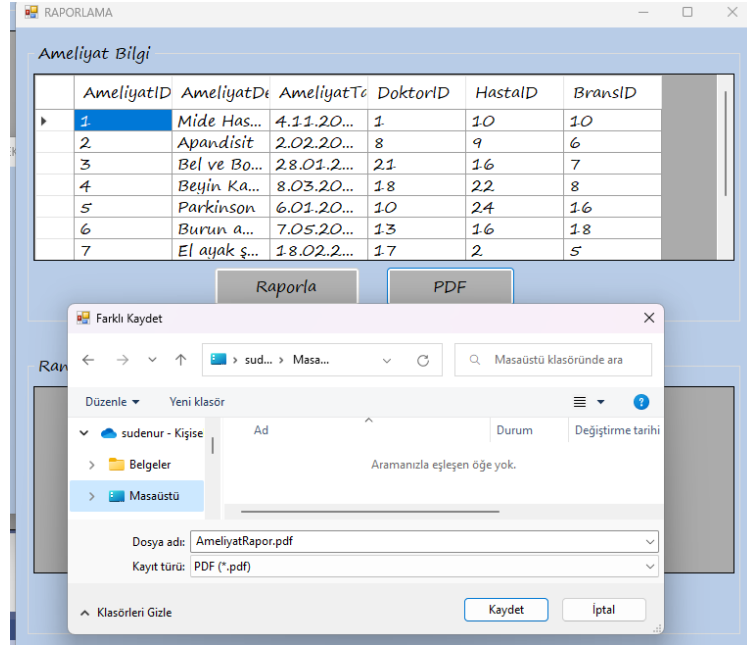
Randevu Bilgi

	RandevuID	RandevuTa	RandevuSa	HastaID	DoktorID	SekreterID	Brans
1	1	20.01.2...	15:20:00	10	12	1	11
2	2	4.02.20...	10:15:00	2	19	1	9
3	3	12.03.2...	10:20:00	26	14	1	1
4	4	18.03.2...	09:50:00	10	15	1	4
5	5	6.03.20...	10:10:00	23	15	1	4
6	6	8.03.20...	11:15:00	9	6	1	20

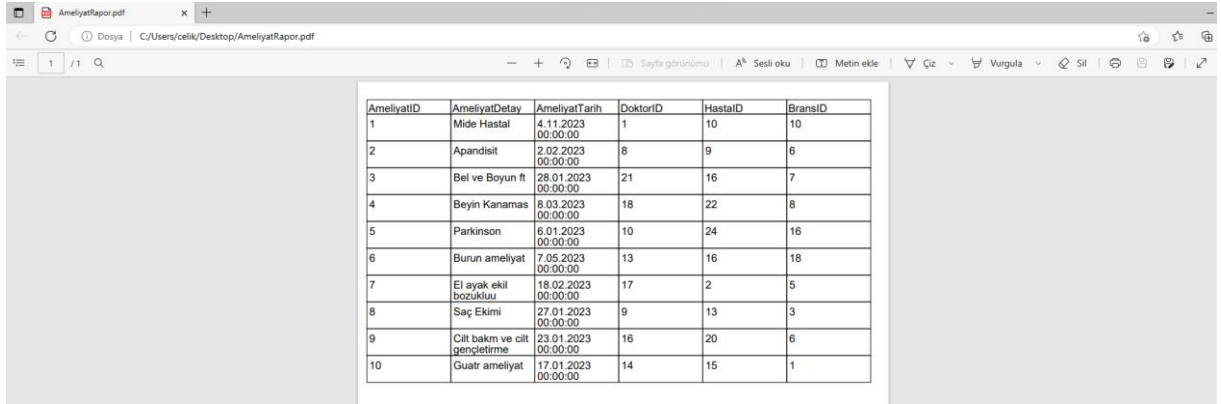
Raporla PDF

Şekil 12: Projedeki raporlama ekranımız. Ameliyat ve Randevu için ayrı ayrı Raporla butonuna bastığımızdaki görüntüler yukarıdaki gibidir.

Ekranada Listelenmekte Olan O Anki Raporun .pdf Uzantılı Olarak Kaydedilebilmesi



Şekil 13: Ameliyat Bilgi kısmında Raporlama yaptıktan sonra PDF olarak kaydetmek istersek PDF butonuna tıkladığımızdaki görüntü yukarıdaki gibidir. Varsayılan olarak AmeliyatRapor adında kaydedecektir.



Şekil 14: AmeliyatRapor adında pdf olarak kaydettikten sonra oluşan PDF'in görüntüsü

```

private void btnRandevupdf_Click(object sender, EventArgs e)
{
    if (dtgrdRandevu.Rows.Count > 0)
    {
        SaveFileDialog sfd = new SaveFileDialog();
        sfd.Filter = "PDF (*.pdf)|*.pdf";
        sfd.FileName = "RandevuRapor.pdf";
        bool fileError = false;
        if (sfd.ShowDialog() == DialogResult.OK)
        {
            if (File.Exists(sfd.FileName))
            {
                try
                {
                    File.Delete(sfd.FileName);
                }
                catch (IOException ex)
                {
                    fileError = true;
                    MessageBox.Show("Hata Oluştı. Tekrar Deneyiniz." + ex.Message);
                }
            }
            if (!fileError)
            {
                try
                {
                    PdfPTable pdfTable = new PdfPTable(dtgrdRandevu.Columns.Count);
                    pdfTable.DefaultCell.Padding = 3;
                    pdfTable.WidthPercentage = 100;
                    pdfTable.HorizontalAlignment = Element.ALIGN_LEFT;

                    foreach (DataGridViewColumn column in dtgrdRandevu.Columns)
                    {
                        PdfPCell cell = new PdfPCell(new Phrase(column.HeaderText));
                        pdfTable.AddCell(cell);
                    }

                    foreach (DataGridViewRow row in dtgrdRandevu.Rows)
                    {
                        foreach (DataGridViewCell cell in row.Cells)
                        {
                            pdfTable.AddCell(cell.Value.ToString());
                        }
                    }

                    using (FileStream stream = new FileStream(sfd.FileName, FileMode.Create))
                    {
                        Document pdfDoc = new Document(PageSize.A4, 10f, 20f, 20f, 10f);
                        PdfWriter.GetInstance(pdfDoc, stream);
                        pdfDoc.Open();
                        pdfDoc.Add(pdfTable);
                        pdfDoc.Close();
                        stream.Close();
                    }

                    MessageBox.Show("İşlem Başarılı.", "Info");
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Hata : " + ex.Message);
                }
            }
        }
        else
        {
            MessageBox.Show("Kayıt Başarısız", "Info");
        }
    }
}

```

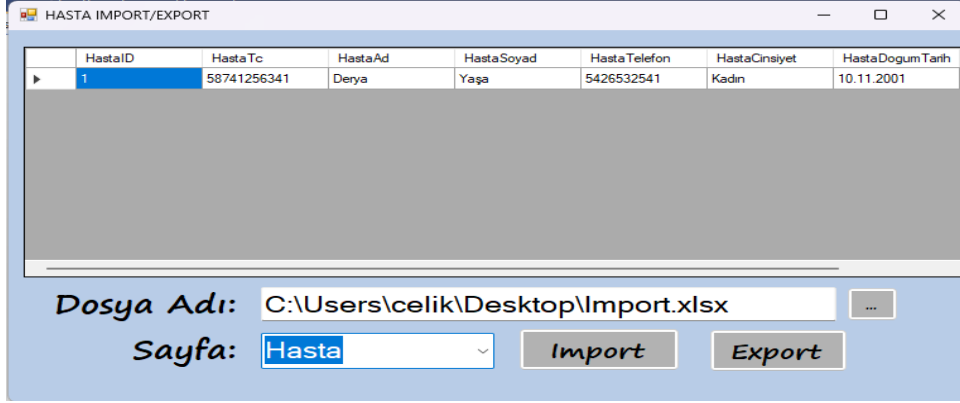
Şekil 15: PDF butonunun işlemleri yaptığı kod bloğu

- ✓ Yukarıdaki işlemleri Randevu Bilgi kısmında Raporlama işlemi yaparak da gerçekleştirebiliriz.

IMPORT/EXPORT İŞLEMLERİ

HastaID	HastaTc	HastaAd	HastaSoyad	HastaTelefon	HastaCinsiyet	HastaDogumTarih	SekreterID
1	58741256341	Derya	Yaşa	5426532541	Kadın	10.11.2001	1

Şekil 16: İmport yapmak istediğimiz verileri excele gireriz.



Şekil 17: Kaydettiğimiz excel belgesini dosya adından seçeriz ve excelde hangi sayfada olduğunu seçerek exceldeki verileri import yaparız.

HastaID	HastaTc	HastaAd	HastaSoyad	HastaTelefon	HastaCinsiyet	HastaDogumT...	SekreterID
2	44444444444	Cemil	Er	5425321520	Erkek	2010-11-10	1
5	12141251255	Sevda	Yüce	5421253256	Kadın	2000-10-02	1
10	58741256341	Derya	Yaşa	5426532541	Kadın	2001-10-11	1

Şekil 18: İmport sonucunda exceldeki veri, veri tabanımıza kayıt olmuştur.

```
1 reference
private void btnImport_Click(object sender, EventArgs e)
{
    int satir = dtgrHasta.Rows.Count;
    if (satir > 0)
    {
        String tc, ad, soyad, telefon, cinsiyet, dogumtarih, sekreterid;

        for (int i = 0; i < satir; i++)
        {
            tc = dtgrHasta.Rows[i].Cells[1].Value.ToString();
            ad = dtgrHasta.Rows[i].Cells[2].Value.ToString();
            soyad = dtgrHasta.Rows[i].Cells[3].Value.ToString();
            telefon = dtgrHasta.Rows[i].Cells[4].Value.ToString();
            cinsiyet = dtgrHasta.Rows[i].Cells[5].Value.ToString();
            dogumtarih = dtgrHasta.Rows[i].Cells[6].Value.ToString();
            sekreterid = dtgrHasta.Rows[i].Cells[7].Value.ToString();

            SqlCommand komut = new SqlCommand("Insert into Hasta(HastaTc,HastaAd,HastaSoyad,HastaTelefon,HastaCinsiyet,HastaDogumTarih,SekreterID) Values (@tc,@ad,@soyad,@telefon,@cinsiyet,@dogumtarih,@sekreterid)", baglan.baglanti());
            komut.Parameters.AddWithValue("@tc", tc);
            komut.Parameters.AddWithValue("@ad", ad);
            komut.Parameters.AddWithValue("@soyad", soyad);
            komut.Parameters.AddWithValue("@telefon", telefon);
            komut.Parameters.AddWithValue("@cinsiyet", cinsiyet);
            komut.Parameters.AddWithValue("@dogumtarih", dogumtarih);
            komut.Parameters.AddWithValue("@sekreterid", sekreterid);
            komut.ExecuteNonQuery();
        }
        baglan.baglanti().Close();
        MessageBox.Show("İmport işlemi başarıyla gerçekleşti.");
    }
}
```

Şekil 19: İmport butonunun işlemleri yaptığı kod bloğu

HASTA IMPORT/EXPORT

	HastaID	HastaTc	HastaAd	HastaSoyad	HastaTelefon	HastaCinsiyet	HastaDogumTarih
▶	2	44444444444	Cemil	Er	5425321520	Erkek	10.11.2010
	5	12141251255	Sevda	Yüce	5421253256	Kadın	2.10.2000
	10	58741256341	Derya	Yaşa	5426532541	Kadın	11.10.2001

Dosya Adı: ...

Sayfa: Import Export

Şekil 20: Bu ekran üzerinden sistemde bulunan hastaların bilgilerini export ederiz.

Farklı Kaydet

← → ↕ ↻ Giriş

Giriş klasöründe ara

Düzenle

Giriş

sudenur - Kişise

Belgeler

Ad

Masaüstü

Değiştirme tarihi

29.12.2022 23:43

Tür

Sistem Klasörü

Dosya adı: Export

Kayıt türü: Excel Workbook (*.xlsx)

Klasörleri Gizle

Kaydet İptal

Şekil 21: Export ettiğimiz verilerin .xlsx formunda kaydettiğimiz sayfa.

	A	B	C	D	E	F	G	H
1	HastaID	HastaTc	HastaAd	HastaSoyad	HastaTelefon	HastaCinsiyet	HastaDogumTarih	SekreterID
2	2	44444444444	Cemil	Er	5425321520	Erkek	10.11.2010	1
3	5	12141251255	Sevda	Yüce	5421253256	Kadın	2.10.2000	1
4	10	58741256341	Derya	Yaşa	5426532541	Kadın	11.10.2001	1

Şekil 22: Export işlemi sonucunda sistemdeki verilerin excele kaydedilmiş hali.

```

1 reference
private void btnExport_Click(object sender, EventArgs e)
{
    using (SaveFileDialog sfd = new SaveFileDialog() { Filter = "Excel Workbook|*.xlsx" })
    {
        if (sfd.ShowDialog() == DialogResult.OK)
        {
            try
            {
                using (XLWorkbook workbook = new XLWorkbook())
                {
                    workbook.Worksheets.Add(this.appData.Hasta.CopyToDataTable(), "Hasta");
                    workbook.SaveAs(sfd.FileName);
                }
                MessageBox.Show("export işlemi başarıyla gerçekleşti.");
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Message", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
}

```

Şekil 23: Export butonunun işlemleri yaptığı kod bloğu

UYGULAMAMIZDA KULLANDIĞIMIZ “YEDEK ALMA” VE “YEDEKTEN DÖN” İŞLEMLERİ

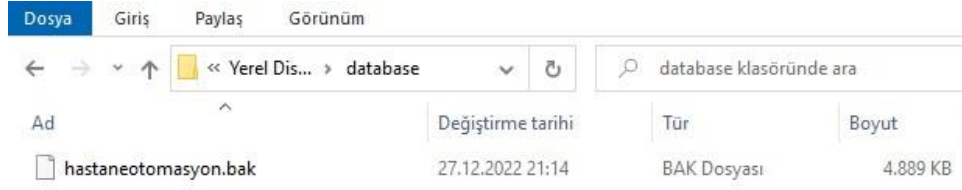
1-Yedek Alma



Şekil 24: Yedek alma işlemini yaptığımız bu arayüzde yedeklemek istediğimiz veritabanının adını girip ardından yedekle butonuna bastığımızda C:\database\ yolu içerisinde yedeklediğimiz .bak uzantılı dosyayı ekliyor.



Şekil 25: Yedekleme işlemini başarıyla gerçekleştirirse bu mesaj ekranda beliriyor.



Şekil 26: Yedekleme işlemi tamamlandığında yukarıdaki gibi .bak uzantılı dosyayı elde ettik.

```
private void btnYedekle_Click(object sender, EventArgs e)
{
    string serverName = txtSunucu.Text;
    string dbName = txtDB.Text;
    string connectionstr = @"Data Source=" + serverName + ";Initial Catalog=" + dbName + ";Integrated Security=True";
    SqlConnection baglan = new SqlConnection(connectionstr);
    baglan.Open();

    string str1 = "USE " + dbName + ";";
    string str2 = "BACKUP DATABASE " + dbName +
        " TO DISK = 'C:\\database\\" + dbName +
        ".bak' WITH FORMAT, MEDIANAME = 'Z_SQLServerBackups', NAME = 'Full Backup of " + dbName + "'";
    SqlCommand cmd1 = new SqlCommand(str1, baglan);
    SqlCommand cmd2 = new SqlCommand(str2, baglan);
    cmd1.ExecuteNonQuery();
    cmd2.ExecuteNonQuery();
    MessageBox.Show("Yedekleme işlemi başarıyla gerçekleşti. Yedeklenen dosyaya ulaşmak için (DB ismi.Bak) uzantısını Disk C:\\database\\(DB ismi.Bak) yolunda bulabilirsiniz");
    baglan.Close();
}
```

Şekil 27: Yedekle butonundaki işlemleri gerçekleştiren kodlar

2-Yedekten Dön

Şekil 28: Dosya yolundan yedekten döndürmek istediğimiz C:\database\ yolu içerisinde yedeklediğimiz .bak uzantılı dosyayı seçiyoruz. Ardından sunucu ve veritabanı adını yazdıktan sonra yedekten dön işlemini gerçekleştirmiş oluyoruz.

```

private void btnDosyaYol_Click(object sender, EventArgs e)
{
    OpenFileDialog openFileDialog = new OpenFileDialog();

    openFileDialog.InitialDirectory = @"C:\\";
    openFileDialog.Title = "Browse Text Files";

    openFileDialog.CheckFileExists = true;
    openFileDialog.CheckPathExists = true;

    openFileDialog.DefaultExt = ".bak";
    openFileDialog.Filter = "Text files (*.bak)|*.bak";
    openFileDialog.FilterIndex = 2;
    openFileDialog.RestoreDirectory = true;

    openFileDialog.ReadOnlyChecked = true;
    openFileDialog.ShowReadOnly = true;

    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        txtDosyaYol.Text = openFileDialog.FileName;
    }
}

```

Şekil 29: Dosya Yolunu seçerek textbox'a yazdıran tıkladığımız 3 nokta butonunda yer alan kodlar

```

private void btnYedektenDon_Click(object sender, EventArgs e)
{
    string servername = txtSunucu.Text;
    string dbname = txtDB.Text;

    string connectionstr = @"Data Source=" + servername + ";Initial Catalog=" + dbname + ";Integrated Security=True";
    SqlConnection baglan = new SqlConnection(connectionstr);

    baglan.Open();

    string str1 = "USE master;";
    string str2 = "ALTER DATABASE " + dbname + " SET SINGLE_USER WITH ROLLBACK IMMEDIATE;";
    string str3 = "RESTORE DATABASE " + dbname + " FROM DISK = '" + txtDosyaYol.Text + "' WITH REPLACE ";
    SqlCommand cmd1 = new SqlCommand(str1, baglan);
    SqlCommand cmd2 = new SqlCommand(str2, baglan);
    SqlCommand cmd3 = new SqlCommand(str3, baglan);
    cmd1.ExecuteNonQuery();
    cmd2.ExecuteNonQuery();
    cmd3.ExecuteNonQuery();
    MessageBox.Show("Veritabanınızı yedekten döndürme işlemi başarıyla gerçekleşti.");

    baglan.Close();
    Application.Exit();
    this.Hide();
}

```

Şekil 30: Yedekten dönme işlemini gerçekleştiren Yedekten Dön butonu içerisinde yer alan kodlar

PROJEMİZDE SİLİNER KAYITLARI ...Silinenler ADLI TABLOYA EKLEME

- Her bir tablo için aynı alan ve türlerini içeren ..._Silinenler tablosu oluşturduk. Yani örneğin Brans tablosundan kayıt silme işlemi sonrasında silinenleri başka tabloya kaydedebilmek için veritabanında Brans_Silinenler adında başka bir tablo oluşturup silinen kayıtların buraya aktarılmasını sağladık. Bu işlemi Doktor, Hasta, Randevu ve Sekreter tabloları için de tekrarladık.

BransID	BransAd
1	dahiliye
2	Cildiye
3	Göz hastalıkları
4	Dahiliye
NULL	NULL

Şekil 31: Projemizde örnek birkaç tane brans ekleyip ardından bazılarını sildikten sonra Brans_Silinenler tablosunda oluşan görüntü yukarıdaki gibidir.

```

1 reference
private void btnSil_Click(object sender, EventArgs e)
{
    SqlCommand insertKomut = new SqlCommand( "INSERT INTO dbo.Brans_Silinenler (BransAd) SELECT BransAd FROM Brans WHERE BransAd=@ad SET " +
        "IDENTITY_INSERT dbo.Brans_Silinenler OFF;" , baglan.baglanti() );
    insertKomut.Parameters.AddWithValue( "@ad" , txtBransAd.Text );
    insertKomut.ExecuteNonQuery();
    SqlCommand komut = new SqlCommand("Delete from Brans where BransAd=@ad", baglan.baglanti());
    komut.Parameters.AddWithValue("@ad", txtBransAd.Text);
    komut.ExecuteNonQuery();
    baglan.baglanti().Close();
    MessageBox.Show("Kayıt silindi.", "Uyarı", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
    listele();
    lblToplamBrans.Text = Convert.ToString(dtgrBrans.RowCount - 1);
}

```

Şekil 32: Silinmiş kayıtları Brans_Silinenler tablosuna aktardığımız işlemi Brans ekranında yer alan Sil butonu içerisinde yukarıdaki kodlarla gerçekleştirdik.

PROJEMİZİN ARAYÜZ SAYFASI



Şekil 33: Giriş Sayfamızda Admin, Sekreter ve Doktor için ayrı ayrı girişler mevcuttur. Her kullanıcı kendisine ait işlemleri gerçekleştirebilmektedir.

ADMIN GİRİŞ

Admin Giriş

TC:

Şifre:

Şekil 34: Admin Giriş Sayfası. Sekreter ve Doktor kullanıcıları için de giriş sayfaları benzerdir. Projemizdeki senaryoda ağırlıklı olarak TC ile işlemleri gerçekleştirdiğimizden ötürü kullanıcı ad yerine TC ile girişlerini sağladık.

SEKRETER EKLE

Sekreter Ekleme Sayfası

TC:

Ad:

Soyad:

Şifre:

Sekreter Listesi

	SekreterID	SekreterAd	SekreterSoy	SekreterTc	SekreterSif	AdminID
▶	1	Can	dakkdjak	111111...	123	1
	2	Sude	lkdl	222222...	1234	1
*						

Şekil 35: Admin'in kullanacağı sekreter ekleme sayfası. Bu sayfa üzerinden Sekreter ekleyebilir, gerektiğinde sekreterin bilgilerini güncelleyebilir veya direkt olarak sekreteri silebilir.

SEKRETER EKRANI

Sekreter Ekranı



Doktor
Detay



Randevu
Oluştur



Branş
Ekle



Hasta
Ekle



Raporlama

Şekil 36: Sekreter kullanıcısı TC ve Şifre ile sisteme girdikten sonra karşısına çıkacak olan ekran.

DOKTOR EKLE

Doktor Ekle


TC:

Ad:

Soyad:

Şifre:


Branş:



Doktor Listeleme ve Arama

Doktor TC Ara:

Doktor Adı Ara:



	DoktorID	DoktorTc	DoktorAd	DoktorSoyad	DoktorBranş
▶	1	333333...	seda	ak	123
*					

Listelenen Doktor Sayısı: 1

Şekil 37: Sekreter kullanıcısının doktor eklemek için kullanacağı sekreter ekleme sayfasıdır. Bu sayfa üzerinden sekreter, doktor ekleyebilir, gerektiğinde doktorun bilgilerini güncelleyebilir veya direkt olarak doktoru silebilir. Bu ekranda aynı zamanda doktorun TC ve adına göre de arama yapılabilir aradığı bilgilere uyan kişiler DataGridView ekranında listelenmektedir. Ve toplam doktor sayısı da ekranda listelenmektedir.

HASTA EKLE

Hasta Bilgi

TC:


Ad:

Soyad:

Telefon: () -

Doğum Tarihi: 21.11.2022


Cinsiyet: ☐ K ☐ E



Hasta Listeleme ve Arama

Hasta TC Ara:

Hasta Telefon Ara:



	HastaID	HastaTc	HastaAd	HastaSoyad
▶	2	444444...	Cemil	Er
	5	121412...	Sevda	Yüce
*				

Kadın Hasta Sayısı: 1 Listelenen Hasta Sayısı: 2
Erkek Hasta Sayısı: 1

Şekil 38: Sekreter kullanıcısının eklemek için kullanacağı hasta ekleme sayfasıdır. Bu sayfa üzerinden Sekreter, hasta ekleyebilir, gerektiğinde hastayı silebilir. Bu ekranda aynı zamanda hastanın TC ve adına göre de arama yapılabilir aradığı bilgilere uyan kişiler DataGridView ekranında listelenmektedir. Ve ekranda kadın hasta, erkek hasta ve toplam bulunan hasta sayısı listelenmektedir.

RANDEVU OLUŞTUR

Hasta Bilgisi Seçin

Hasta ID:

Hasta TC Ara:

	HastaID	HastaTc	HastaAd	HastaSot
▶	2	444444...	Cemil	Er
	5	121412...	Sevda	Yüce
*				


Randevu Bilgi

Branş:

Doktor:

Randevu Tarihi: 29 Aralık 2022

Randevu Saat:



Randevular

Randevu ID:


	RandevuID	RandevuTa	RandevuSa	HastaID	DoktorID	SekreterID	BransID
▶	2	30.12.2...	10:20:00	2	1	1	1
*							

Şekil 39: Sekreter kullanıcısının randevu oluşturma sayfasıdır. Bu sayfa üzerinden sekreter, randevu oluşturmak istediği hastayı seçerek randevu ekler ve gerektiği zaman oluşturduğu randevulardan istediğini silebilir. Bu ekranda aynı zamanda hastanın TC'sine göre de arama yapılabilir aradığı bilgilere uyan kişiler DataGridView ekranında listelenmektedir.

BRANŞ EKLE


Branş Ekle

Branş Ad:



Branş Arama

Branş Adı Ara:



	BransID	BransAd
▶	1	göz
*		

Listelenen Branş Sayısı: 1

Şekil 40: Sekreter kullanıcısının branş ekleme sayfasıdır. Bu sayfa üzerinden sekreter, branş ekleyebilir, gerektiğinde branşı güncelleyebilir veya branşı silebilir. Bu ekranda aynı zamanda branşın adına göre de arama yapılabilir aradığı bilgilere uyan branşlar DataGridView ekranında listelenmektedir. Ve toplam branş sayısı da ekranda listelenmektedir.

DOKTOR İŞLEMLERİ

Hasta Bilgisi Seçin

Hasta TC Ara:

	HastaID	HastaTc	HastaAd	HastaSoyad	H
▶	2	4444444...	Cemil	Er	5
*	5	121412...	Sevda	Yüce	5
*					

Tanı ve Reçete


Hasta ID:

Tarih: 29 Aralık 2022 ▾

Tanı Bilgi:

İlaç Ad:

☐ İlaç Yok



Randevular


	RandevuID	RandevuTa	RandevuSa	HastaID	H
▶	2	30.12.2...	10:20:00	2	2
*					

Ameliyat

Hasta ID:

Tarih: 29 Aralık 2022 ▾

Detay:



Şekil 41: Doktor kullanıcısının işlemlerini yaptığı sayfadır. Bu ekranda doktor hastalarına tanı ve reçete bilgilerini girebilmektedir. Ve aynı zamanda doktor randevularında bulunan hastalardan istediğine ameliyat ekleyebilmektedir.