

21

ÖĞRENCİ NOT KAYIT SİSTEMİ

Bu Bölümde

Proje Amacı	348
Veritabanı ve Tablolarn Oluşturulması	348
Tetikleyicilerin Oluşturulması	353
Prosedürlerin Oluşturulması	356
Giriş Formunun Tasarlanması	357
Bağılantı Sınıfının Oluşturulması ve Giriş Formu Kodları	360
Öğretmen Formu Tasarımı	369
Duyuru İşlemleri Oluşturma Formu	387
Duyuru Listesi Formu	394
Mesajlar Formu	397
Öğrenci Formu	405
Neler Öğrendik?	414

Bu bölümde C# form ile ADO.NET çerçevesinde temel SQL sorguları olan Ekleme, Silme, Güncelleme ve Listeleme başlığına bu 4 temel başlığa ek olarak prosedür, tetikleyici, alt sorgu ve birleştirme işlemlerini bir proje altında birebirştireceğiz.

Tamamıyla SQL alt yapısında gerçekleştireceğimiz projemizde kullanıcılar arası mesajlaşma, sisteme giriş yapan kullanıcının bilgilerini getirme işlemini yani biraz daha terim biçiminde ifade edecek olursak "User Login Management" (Kullanıcı Giriş Yönetimi) sistemini, duyuru oluşturmayı, alınan sınav notlarına göre öğrencinin durumunu SQL üzerinde değiştirmeye gibi pek çok farklı başlığı modüler ve geliştirmeye çok müsaait bir proje çerçevesinde öğreneceğiz.

PROJE AMACI

İlk olarak projemizde neler yapacağımızla başlamak istedim. Bu projemizde amacımız kitabımızın 19 bölümünün tamamında kullanıldığımda başlıkları derleip harmanlayarak ortaya somut ve daha kapsamlı ayrıca geliştirmeye çok müsait bir veritabanı uygulaması koyabilmektir. Projemiz içerisinde kişilerin kendi verilerini görebilecekleri, mesajlaşabilecekleri, profil bilgilerini düzenleyebileceğii alanlara da yer vermek istedim. Bunu beraber kolaylıkla yapabiliyoruz. Amacımız öğrencilerein tipki e-okul gibi kendi numara ve şifreleri ile sisteme giriş yaptıktan zaman notlarını görüntülediği Fotoğraf, Ad, Soyad, Şifre gibi bilgilerini düzenleyebildikleri, diğer öğrencilere ve öğretmenlere mesaj atabildikleri bir sistem oluşturmak. Projemizde tablo olarak;

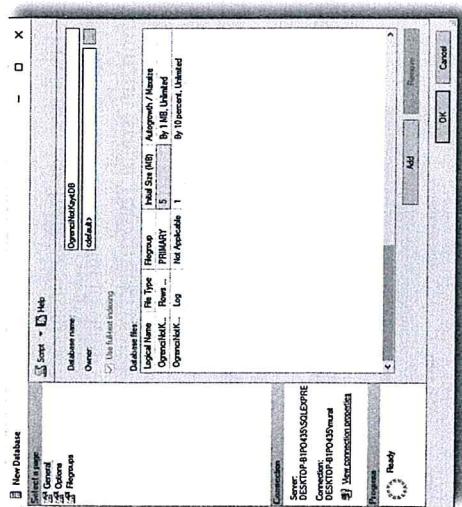
- » Öğrenci Bilgileri Tablosu
- » Öğretmen Bilgileri Tablosu
- » Sınav Notları Tablosu
- » Mesajlar Tablosu
- » Duyurular Tablosu

Olmak üzere 5 temel tablo kullanacağız. Form ara yüzü olarak da;

- » Giriş Formu
 - » Öğrenci Kayıt Formu
 - » Öğretmen Bilgi Detayları Formu
 - » Öğrenci Listesi Formu
 - » Mesajlar Formu
 - » Duyurular Formu
- Olmak üzere 7 temel form kullanacağız. Dilerseniz proje bittiğinden sonra yeni tablo ve veritabanları ile içeriğini zenginleştirbilirisiniz.

VERİTABANI VE TABLOLARIN OLUŞTURULMASI

SQL Server'ı açalım **ve** yeni bir veritabanı oluşturarak işlemlerimize başlayalım. Projemizin ve veritabanımızın isimlerini birbirine paralel olarak adlandırıralım. Veritabanı adı olarak **OgrenciNotKayitDB** verebiliriz.



Şimdi ilk tablomuz olan öğrenci bilgilerini tutacağımız tablomuzu oluşturarak başlayalım. Sırasıyla tablolarmızın alanlarını yazalım içeriğini detaylandıralım.

» Öğrenci Tablosu

- » ID
- » Ad
- » Soyad

Column Name	Data Type	Allow Nulls
ID	smallint	<input type="checkbox"/>
AD	varchar(20)	<input checked="" type="checkbox"/>
SOYAD	varchar(20)	<input checked="" type="checkbox"/>
NUMARA	char(4)	<input checked="" type="checkbox"/>
ŞİFRE	varchar(10)	<input checked="" type="checkbox"/>
FOTOĞRAF	varchar(100)	<input checked="" type="checkbox"/>

ID alanımızı notlar tablosundaki değerlerle ilişkilendireceğimiz için birinci anahtar olarak belirledik. Ayrıca yine ID alanının değerini otomatik artan yaptı. Otomatik artan işlemini önceki bölgülerde anlatmıştık. Numara için char(4) ifadesi kullanmanız sebebi öğrencilerein numarasının 4 karakterden oluşacak olmasındadır. Siz dillerseniz daha geniş kapsamlı bir proje için daha geniş aralıklar kullanabilirsiniz. Şifre alanını da en fazla 10 karakter olacak

Şekilde ayırdık. Fotoğraf kısmı için Image kullanmadık. Çünkü Image veri tipi veritabanı boyutunu çok fazla artırdığından bunun yerine resim yolunu tut.1) cağımız bir string ifade kullanmanın daha optimal bir çözüm olacağını düşünü dük. Tablomuzun ismini TbOGrenci olarak belirleyip kaydederek ilk tablonu sun işlemlerini tamamlamış olduk.

Notlar Tablosu

» ID

» Ad

» Soyad

» Numara

» Şifre

Column Name	Data Type	Allow Nulls
ID	tinyint	<input type="checkbox"/>
AD	varchar(20)	<input checked="" type="checkbox"/>
SOYAD	varchar(20)	<input checked="" type="checkbox"/>
NUMARA	char(4)	<input checked="" type="checkbox"/>
SIFRE	varchar(5)	<input type="checkbox"/>

5 temel alanımız var. Tipki öğrenci tablosunda olduğu gibi her öğretmenin 1'e birer ID değeri, Adı, Soyadı, Numarası ve Şifresi bulunacak. Ancak fotoğraf,1.1 ihtiyaca duymadık. Çünkü fotoğraf alanı öğretmeni öğrenci hatırlaması için mevcuttur. Numarayı yine 4 karakter olarak belirleyip şifreyi bu kez daha krt.,1 tuttuk. Öğretmen ve öğrenci tabolarına girişler farklı şekilde olacak. Bu det.,1 yları projemizin form kısmına geçtiğimiz zaman göreceğiz.

Notlar Tablosu

» ID

» Sınav1

» Sınav2

» Sınav3

» Proje

» Ortalama

» Durum

Duyuru Tablosu

» Duyuru ID

» Duyuru İçerik

Column Name	Data Type	Allow Nulls
OGRID	smalldint	<input type="checkbox"/>
SINAV1	tinyint	<input checked="" type="checkbox"/>
SINAV2	tinyint	<input checked="" type="checkbox"/>
SINAV3	tinyint	<input checked="" type="checkbox"/>
PROJE	tinyint	<input checked="" type="checkbox"/>
ORTALAMA	decimal(18, 2)	<input checked="" type="checkbox"/>
DURUM	bit	<input checked="" type="checkbox"/>

Notlar tablomuzda ID alanı dışında Sınav notlarını gireceğimiz 1, 2 ve 3. sınavlar, proje notunu gireceğimiz proje sütunu, notların toplandığı bölümnesiyle hesaplanacak ortalamaya değerini ve öğrencinin geçti mi, yoksa kaldı mı? sonucunun yazarlığı durum sütunları bulunmaktadır. Sınav notları 0-100 arasında birer tam sayıdan oluşacağı için tınyint veri tipini kullandı. Ortalama değerlerini ondalıklı çıkabileceğinden decimal veri tipini virgülüden sonra en fazla iki basamak alacak şekilde tercih ettik. Durumun iki sonucu olduğunu bu veri tipi için de bit kullanarak tablomuzu oluşturduk.

Mesajlar Tablosu

» ID

» Gönderen

» Alıcı

» Başlık

» İçerik

Column Name	Data Type	Allow Nulls
ID	smalldint	<input type="checkbox"/>
GONDEREN	char(4)	<input checked="" type="checkbox"/>
ALICI	char(4)	<input checked="" type="checkbox"/>
BASLIK	varchar(50)	<input checked="" type="checkbox"/>
ICERIK	varchar(250)	<input checked="" type="checkbox"/>

Her mesajın bir ID numarası olsun istedğimizden ilk olarak ID sütununu ekledik ve sütün özelliklerini penceresinden bu alanımızı otomatik artan yaptıkt. Gönderici ve alıcı değerleri numaraya göre yapılacağı için bu başlıklarını char(4) olarak belirledik. Mesajımızın bir başlığı ya da konusu olsun istedğimiz için başlık son olarak da mesajın içeriğini tutulacağının tutulacağı içerik sütunumuzu da ekleyerek tablo-muzu tamamladık.

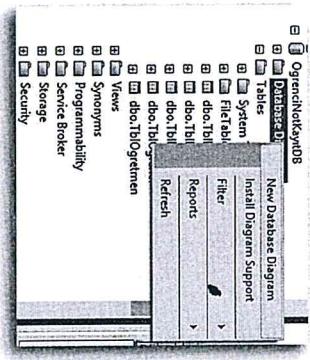
Duyurular Tablosu

» Duyuru ID

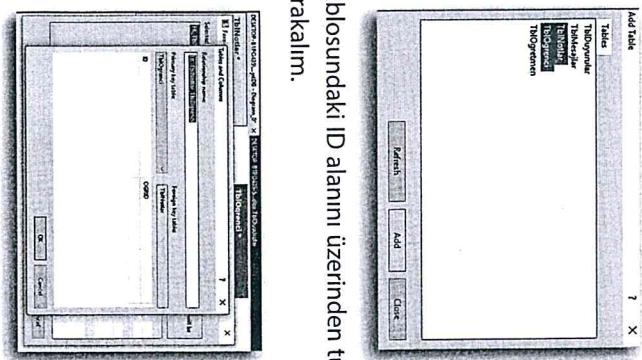
» Duyuru İçerik

Column Name	Data Type	Allow Nulls
ID	smalldint	<input type="checkbox"/>
İçerik	varchar(100)	<input checked="" type="checkbox"/>

Duyuru tablomuz öğretmen tarafından oluşturulacak duyuruları tutacaktır. Bu duyurular tüm öğrencilere ulaşacağı için herhangi bir kısıtlama yoktur. Yani dileyim ki öğretmenin yarın sabah dersin 09:30 yerine 10:15'de başlayacağını tüm öğrencilere haber vermek istiyor. Bunun için öğrencilerin tamamına tek tek mesaj atmak çok zahmetli olacağından basit bir duyuru oluşturma yöntemiyle tek bir Button tıklamasıyla tüm öğrencilere ulaşabilir. Böylece tablolarmızı tamamlandı. Son olarak öğrenci tablomuzla notlar tablomuz arasında ilişkili olduğundan dolayı bu iki tabloyu birbirine bağlayalım. Database Diagrams üzerinde sağ tuşa tıklayalım.



Açılan pencerede CTRL tuşuna basılı tutarak öğrenci tablosunu ve notlar tablosunu seçelim.



Böylelikle ilişkimizi de tamamlamış olduk. Bu pencere üzerindeyken Ctrl+S tuşlarına basarak ilişkimizi kaydededelim.

Şimdi çok önemli olan bir başka durum var. Bir öğrenci sisteme kaydedildiği anda o öğrencinin ID değeri notlar tablosuna da otomatik olarak kaydolmalıdır. Bunun için tetikleyici kullanabiliriz. Tetikleyicilerin ne işe yaradığını hatırlatalım: "Bir tablodaki olayın bir başka tabloya etkilemesidir." Örneğin; biz ürün tablosundan bir ürünü sildiğimiz anda ilgili ürünün değeri stok tablosunda bir azalısın gibi...

TETİKLEYİCİLERİN OLUŞTURULMASI

SQL'de daha önce basit bir şekilde nasıl tetikleyici oluşturacağımızı öğrenmiştim. Şimdi biraz daha kapsamlı bir tetikleyici oluşturacağız. Sorgumuzu yapıp açıklayalım.



Öğrenci tablosuna herhangi bir öğrenci eklendiği anda ilgili öğrencinin ID değerini otomatik olarak notlar tablosuna atayan sorguyu yazalım.

```
Create trigger IdEkle
on TblOgrenci
After Insert
as
Declare @ID smallint
select @ID=ID from inserted
insert into TblNotlar(06RID)values(@ID)
```

Daha sonra öğrenci tablosundaki ID alanını üzerinden tutup sürükleyip notlar tablosunun üzerine bırakalım.

Aşağıda bulunan resimdeki gibi birebir formatta bir ilişki oluşturacaktır. Birebir formatta olması şu anlamına geliyor; bir öğrencinin yalnızca bir kez notları girişi olabilir. Yani 4 nolu ID değerine sahip öğrenci birden fazla kez notları girilemez.

Sorgumuzu yapıp çalıştırduğumuz zaman alt ekranın içeriğinin başsanlı bir şekilde gerçekleştiğine dair bir mesaj almamız gerekecektir.

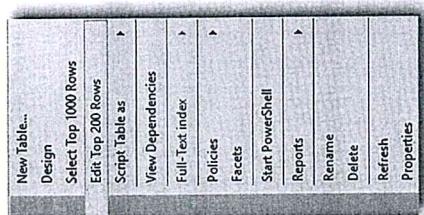
Create ifadesi ile önce ne oluşturacağımızı sorgumuza bildirdik. Create komutundan sonra tablo, prosedür, veritabanı ve daha pek çok şey oluşturulabilir. Bu tetikleyici oluşturacağımız için trigger ifadesini kullandık ve tetikleyicimizin ismini IdEkle olarak belirledik. On komutu ile tetikleyicimizin hangi tablo üzerinde çalışacağını belirttik. Daha sonra tetikleyicimizin ne zaman çalışacağıını bildirdik. After Insert sorgusu ile tetikleyicimiz herhangi bir kayıt yapıldıktan sonra çalışacaktır. As komutu ile tetikleyici çalışmına neler olacağına yazmayıp başladık. Biz bir tablodaki değeri diğer tabloya taşımak istedığımız için taşımak istediğimiz değerler için birer değişken oluşturduk. Sadece ID alanını taşıyacağımız için @ID ismindede bir değişken tanımlaması yaptık.

Bu değişkenin veri türünü tablomuzdaki veri türleri ile aynı olmasını gözden kaçırılmayalım.

Değişken tanımlamaları SQL'de Declare komutu ile başladığından sorgumuzu başına bu komutu ekledik. Alt satırda yani Select ile başlayan satırda tanımladığımız değerimize atayacağımız değer verdik. Bu değer öğrenci tablosunu son eklenen öğrencinin ID değeridir. Daha sonra hafızaya aldığımız @ID değişkenini Insert sorgusuya notlar tablosuna ekleyerek sorgumuzu tamamladık.

Öğrenci tablomuza bir kayıt eklemeden önce sınav notlarının tutacağımız tablo da tüm notlarımızın varsayılan değerini 0 yapalım ki herhangi bir öğrenci eklemesi yaptığımda zaman notları null değer olarak kalmasın.

Tablo sütunlarınızın default value alanından tüm sınav notları ve durum alanınızın başlangıç değerini 0 yaptık. Şimdi öğrenci tablomuzaki tane veri girişini yapalım. Veri giriş'i için tablomuz üzerinde sağ tuşla tıklayıp Edit Top 200 Rows alanını seçiyoruz.



İki tane veri giriş'i yapalım.

ID	AD	SOYAD	NUMARA	SİFRE	FOTOGRAF
1	Mehmet	Yılmaz	1226	11	NULL
2	Caren	Çetink	2568	a1	NULL
•	NULL	NULL	NULL	NULL	NULL

OGRID	SINAV1	SINAV2	SINAV3	PROJE	ORTALAMA	DURUM
1	0	0	0	0	0,00	False
2	0	0	0	0	0,00	False
•	NULL	NULL	NULL	NULL	NULL	NULL

Biz öğrenci tablosuna verilerimizi girdiğimiz anda notlar tablosunda ID değerleri otomatik olarak eklendi. Böylece tetikleyicimizi de tamamlandı. Şimdi gelelim bir diğer başlığıımız olan prosedür yazımıma.

Column Name	Data Type	Allow Nulls	Column Properties
OGRID	smallint	<input type="checkbox"/>	
SINAV1	tinyint	<input checked="" type="checkbox"/>	
SINAV2	tinyint	<input checked="" type="checkbox"/>	
SINAV3	tinyint	<input checked="" type="checkbox"/>	
PROJE	decimal(18, 2)	<input checked="" type="checkbox"/>	
ORTALAMA	bit	<input checked="" type="checkbox"/>	
DURUM		<input type="checkbox"/>	

General	SINAV1	Yes
Name		tinyint
Allow Nulls		(0)
Data Type		
Default Value or Binding		
Table Designer		

PROSEDÜRLERİN OLUŞTURULMASI

Prosedürler başlığını önceki bölmelerimizde detaylı bir şekilde anlatmıştık. Prosedürleri genellikle ilişkili tablolarındaki bazı sütunları birleştirme sonrasında uzun SQL sorgularını kısaltmak için kullanıyoruz. Projemizin içeriğine göre prosedür kullanıacak alanlar arıtmalı. Ancak biz yalnızca notlar tablosu için bir prosedür yazacağız.

Üyelik2

Notlar tablosundaki verileri ID değerine karşılık gelen Ad ve Soyad bilgisini göre listeleyen soruyu prosedür içinde yazalım.

Create Procedure Ogrenciler

as

```
Select Ad,Soyad,Sinav1,Sinav2,Sinav3,Proje,Ortalama,Durum From TblNotlar
inner join TblGrenci
on
```

```
TblGrenci.ID=TblNotlar.OGRID
```

Yukarıdaki sorumuzu yapıp çalıştığımız zaman prosedürümüz oluşmuş olacaktır. Prosedürümüzü Çalıştırmak için başına Execute yazmamız yeteरildi. Bu prosedür ile notlar tablosuyla öğrenciler tablosu arasında birleştirme yapmış olduk.

SQL üzerinde yapacağımız temel işlemleri bitirdiğimize göre **Visual Studio** derleyicimizi açıp projemizi oluşturabiliriz. Projemizi şu adımlara ayıracagız;

» Giriş formunun tasarılanması

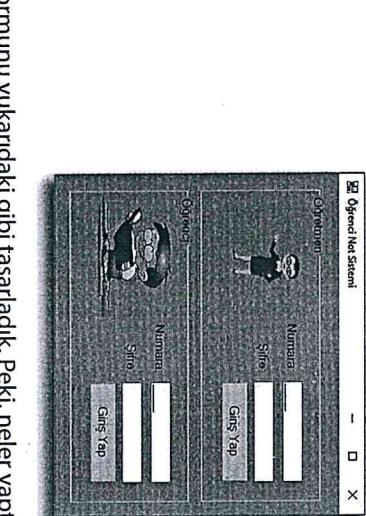
» Bağlantı sınıfının oluşturulması

» Öğrenci bilgileri formu

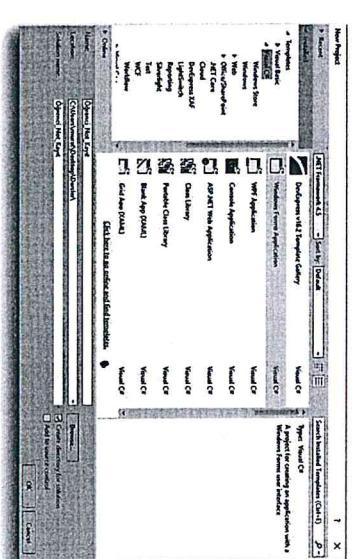
» Öğretmen formu

» Mesajlar formu

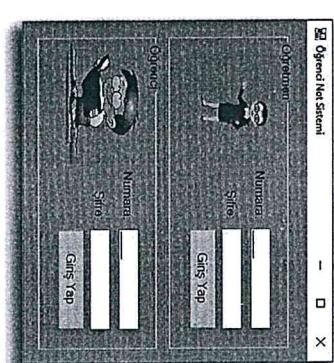
» Duyurular formu



İlk olarak yeni bir proje oluşturalım. Projemizin ismini veritabanımızın ismiyle uyumlu bir şekilde adlandırmaya özen gösterelim.



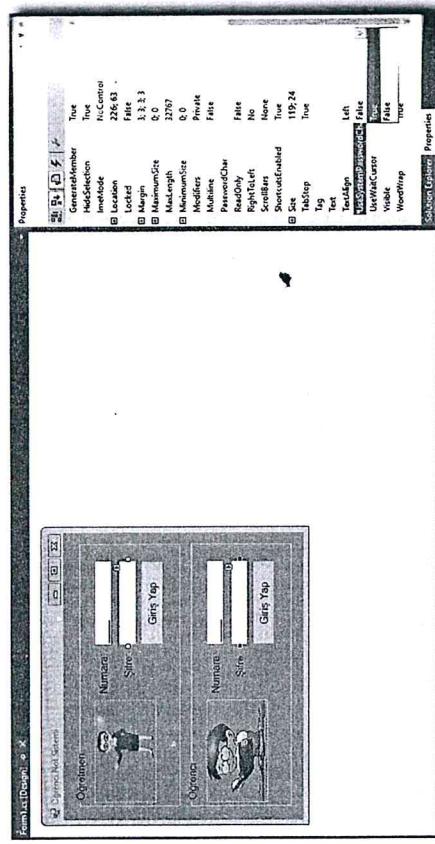
Projemizde ilk olarak giriş formunu hazırlayacağız. Giriş formunda iki ayrı giriş ekranı kullanalım. Bunlardan birisi Öğretmen içeriği de öğrenci içi olsun. Alanları birbirinden ayırmak için kullanılabilen birden fazla araç mevcut. Biz Groupbox tercih edeceğiz.



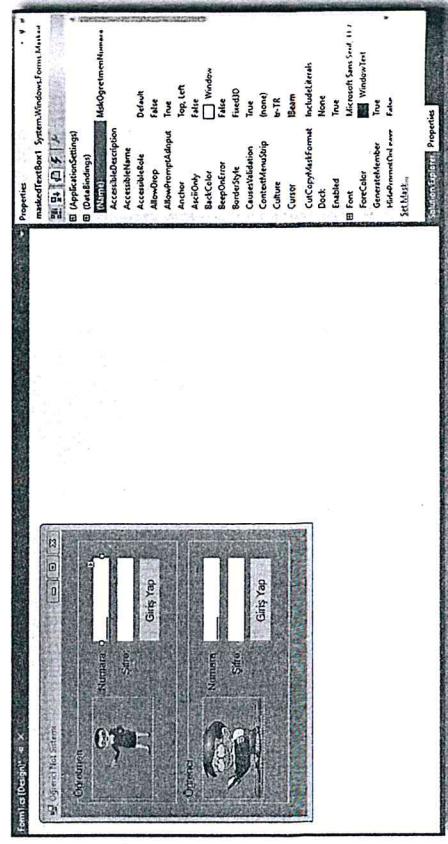
Giriş formunu yukarıdaki gibi tasarladık. Peki, neler yaptı? İlk olarak öğretmen ve öğrenci isminde iki ayrı giriş alanı oluşturduk. Bu alanları Groupbox araçları ile birbirinden ayırdık. Her bir Groupbox aracının içerisinde bir tane Picturebox yanı resim kutusu ekledik. Bu resim kutularında öğretmen ve öğrenciyi temsil edecek iki tane resmi Google görsellerde PNG formatında bulup resim kutularımız içinde kullandık. PNG tercih etmemizin sebebi arkasının transparan yanı şeffaf olmasını istediğimiz içindir. Öğretmen ve öğrenciler sisteme numara ve şifreleri ile giriş yapacakları için birer tane MaskedTextBox ve TextBox araçları kullandık. MaskedTextBox aracımızın giriş modunu 4 karakter olarak belirledik.

GİRİŞ FORMUNUN TASARLANMASI

Bu aracımızın kullanımını kitabımızın araçlar bölümünde detaylı olarak anlatılmıştır. Daha sonra aşağıda bulunan görseldeki gibi TextBox araçlarımızın `Numara` ve `Sıfır` özellikleri özelliğini açtıktan böylece girilen değerler sıfır karakteri şeklinde gelecektir.



Böylece giriş formumuzu tamamlamış olduk. Unutulmaması gereken en önemlili durumlardan birisi de araçların isimlendirilmeleridir. Çünkü birden fazla biliniyi tekrar eden araç için hangi aracın ne işe yaradığını kod kısmında karıştırılabilir. Bundan dolayı araçlarımızı en doğru şekilde adlandırmaya çalişalım. Araçların adlandırılması işimizi de kitabımızın geçmiş böümlerinde çok kez yapmıştık. Örneğin öğretmen numarasının girileceği aracımızı aşağıdaki bulunan görseldeki gibi adlandırabiliriz.



Diger araçlarımız için de şu isimleri kullanalım:

Öğretmen Numara: MskOgretmenNumara

Öğretmen Şifre: TxtOgretmenSifre

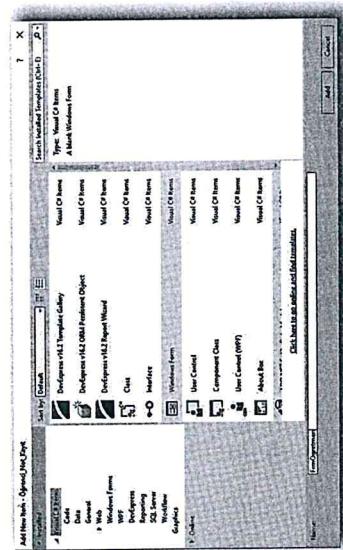
Öğrenci Numara: MskOgrenciNumara

Öğrenci Şifre: TxtOgrenciSifre

Öğretmen Giriş Yap: BtnOgretmenGiris

Öğrenci Giriş Yap: BtnOgrGiris

Diger araçlarımızı kod kısmında kullanmayağımız için onların isimlerini değiştirmemiz gerekmeyecektir. Şimdi ikinci form olarak öğretmen formunu eleyelim. Öğretmen formunu ekleyp formlar arasında geçiş işlemi yapıp akabinde öğretmen formunun tasarımını detaylı bir şekilde yağalım. Öğretmen formumuza ekleyelim.



Project menüsünden **add Windows form** seçenekine tıklayarak yeni form eklemme penceresini açtık. Açılan pencerede formumuzu ismini **FrmOgretmen** olarak belirleyip **OK** yanı tamam seçenekine tıklayarak formumuzu oluşturmuş olduk.

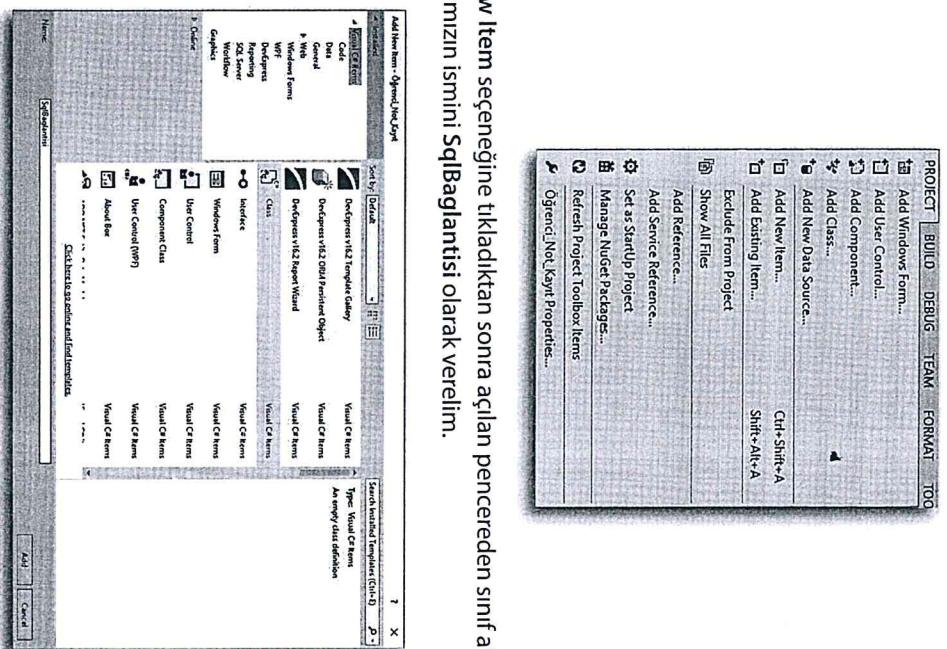
Formlar arasında geçiş yapmadan önce halletmemiz gereken önemli bir problem var. Projemizde 5 belki de daha fazla form bulunacak. Her formumuzun altında mutlaka bir veritabanı işlemi yer alacak. Formlarımıza altında veritabanı işlemi olduğu için bir veritabanı bağlantısına ihtiyaç duyacağız.

Sorun şu: diyelim ki projeyi başka bir bilgisayara taşımak istedik ve projemizde 5 değil de 55 tane form var. Başka bilgisayarda çalıştmak için tüm bağlantı adreslerine ilgili bilgisayarın SQL sunucusu adının yazılması gerekiyor. Ancak 55 defa tek tek değiştirmek hem zaman kaybi hem de algoritmik olarak yanlış bir yaklaşım olur. Ne yapmalıyız?

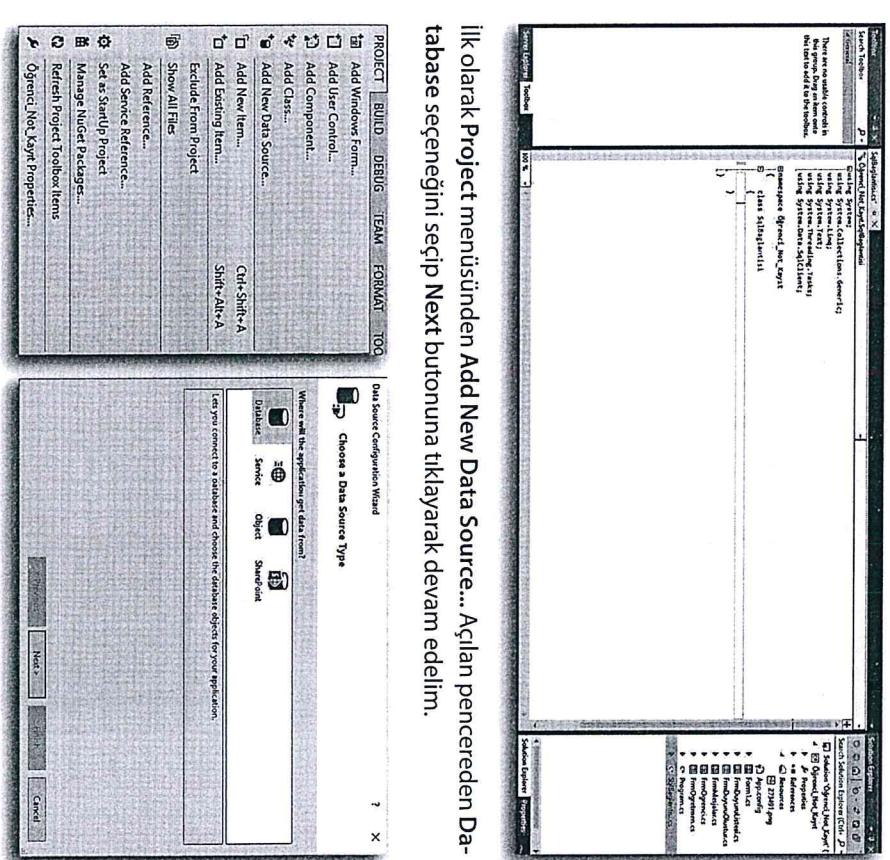
İşte bu noktada devreye sınıf ve metod yapısı girecek. Oluşturacağımız sınıfın de adresimizi tutup her yeni formda sadece bir nesne türeterek bağlantı adresini taşıyabileceğiz. O halde gelin bu başlığı beraber inceleyelim.

BAĞLANTI SINIFININ OLUŞTURULMASI VE GİRİŞ FORMU KODLARI

Bağlantı sınıflar sadece C# formda değil ASP gibi web tabanlı platformlarda da büyük önem taşır. Bağlantı sınıfını oluşturmak için ihtiyacımız olan ilk şey boyut bir tane sınıf. Bunun için Project menüsünü kullanabiliriz.



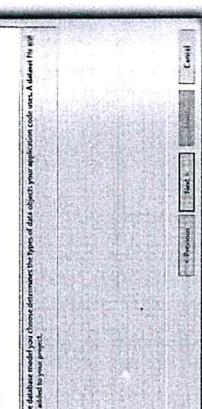
Add New Item seçeneğine tıkladıkten sonra açılan pencereden sınıf alanını seçip sınıfımızın ismini SqlBaglantisi olarak verelim.



ilk olarak Project menüsünden Add New Data Source... Açılan pencereden Database seçeneğini seçip Next butonuna tıklayarak devam edelim.

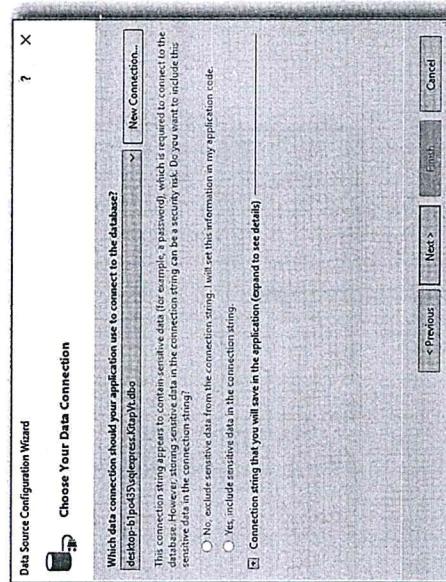
Karşımıza aşağıda bulunan görseldeki gibi boş bir kod sınıfı alanı gelecektir. Sınıf içinde bir metod tanımlayıp bu metod aracılığı ile değerlerimize ulaşabiliyoruz. Sınıf içerisinde metod tanımlanmadan önce üzerinde işlem yapacağımız SQL bağlantı adresini yanı Connection String değerini almamız gerekiyor. Bunun için bağlantı şıhirdini kullanabiliyoruz.

Sunucu adını yeni bağlantı penceresinin Server Name alanına yapıştırıldığımız zaman aşağıda bulunan veritabanı seçim alanı aktif hale gelecektir. Oradan kullanacağımız veritabanını seçiyoruz ve devam ediyoruz.

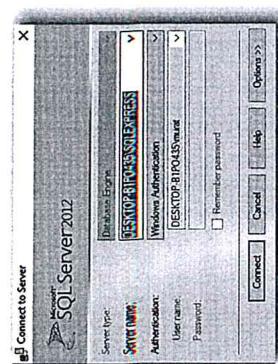


Çıkan ekranın Dataset alanını seçip Next buttonuna tıklyoruz.

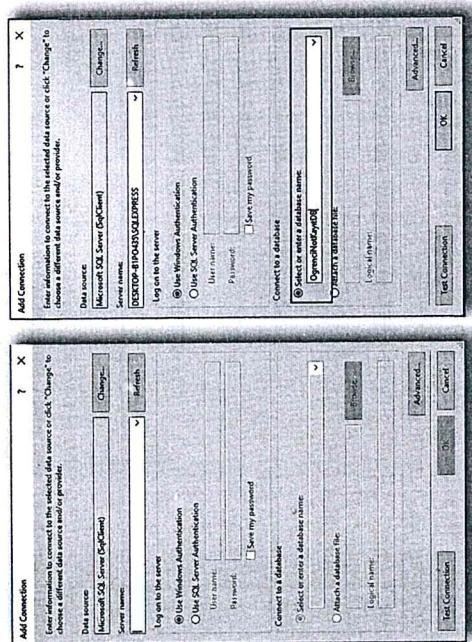
Daha sonra açılan pencerede daha önce kullandığımız herhangi bir bağlantı adresi olabilir. Biz yeni bir bağlantı oluşturmak için New Connection... seçeneğine tıklyoruz.



Tamam butonuna tıklayıp tekrar silhırbaş pencereimize döndük. Burada yapmanız gereken şey Connection String yazan yanındaki tutan alanın sol tarafındaki artıya tıklayıp bağlantılı adresini almak. Bağlılık adresini kopyalyoruz ve Cancel seçeneğine tıklayarak silhırbaşı kapatıyoruz. Armacımız yalnızca bağlantılı adresini alabilmekti.



Yeni bağlantı penceresinde sunucu adı yani Server Name yazan alana SQL'e bağlanırken karşımıza gelen sunucu adını kopyalayıp yapıştırıyoruz.



Tekrar sınıfımıza dönelim. Önce kodlarımızı yazalım daha sonra kodlarımız üzerinde açıklamalarımızı yapalım.

Uygulama 3

SQL bağlantısı sınıfı içeresine bir tane SQL bağlantı metodu tanımlayan kodu yazalım.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient;

namespace Öğrenci_Not_Kayıt
{
    class SqlBaglantisi
    {
        public SqlConnection baglanti()
        {
            SqlConnection baglan = new SqlConnection(@"Data Source=DESKTOP-B1P043;\\SQLEXPRESS;Initial Catalog=ÖğrenciNotKayıtDB;Integrated Security=True");
            baglan.Open();
            return baglan;
        }
    }
}
```

SqlBaglantisi sınıfı içeresine bir tane bağlantı metodunu tanımlayın daha sonra kodlarınıza ekleyin.

Metodumuzu açıp metodumuzun işlevini yazmaya başladık. İlk olarak metodumuz aracılığı ile ulaşacağımız SQL bağlantı adresimizi tanımladık. **Bağlantı() metodunda**

açtık ve geriye bağlantımızdan türöttüğümüz bağlan nesnesini döndürdük. bağlantı

ıan nesnesi bağlantı adresimizi tutacaktır.

Bu kodu çalıştırıp kontrol etmek için küçük bir tablo verecek olursak:

Sqlbaglantisi	Sınıfımızın adı
Baglanti	Metodumuzun adı
Baglan	Metodumuz aracılığı ile ulaşacağımız SQL bağlantı nesnemizin adı

Şeklinde toparlayabiliriz. Şimdi şunu yapalım.

Eğer öğretmen Kullanıcı Adı olan Numarayı ve Şifresini doğru bir şekilde giriyorsa daha önce oluşturmuş olduğumuz boş öğretmen formunu açalım.

Kodlarımızı yazmadan önce kodlarımızın doğruluğunu kontrol etmek için öğretmen tablomuzu açıp bir tane kayıt giriş yapalım ve hemen akabinde kod bloğumuza geçelim.

DESKTOP-B1P043\...\dbo.TblÖğretmen		DESKTOP-B1P043\...\dbo.TblÖğretmen X		
ID	AD	SOYAD	NUMARA	SİFRE
1	Murat	Yıldız	1582	011
2	NULL	NULL	NULL	NULL

Uygulama 4

Öğretmen numarasını ve şifresini doğru bir şekilde girerse öğretmenin formunu açan kodları yazalım.

```
SqlBaglantisi bgl = new SqlBaglantisi();
private void BtnögretmenGiris_Click(object sender, EventArgs e)
{
    SqlCommand komut = new SqlCommand("Select * From TblÖğretmen where
Numara=@p1 and Sifre=@p2", bgl.baglanti());
komut.Parameters.AddWithValue("@p1", MskögretmenNumara.Text);
komut.Parameters.AddWithValue("@p2", TxtögretmenSifre.Text);
SqlDataReader dr = komut.ExecuteReader();
if (dr.Read())
```

Metodumuzu tüm formlarımızdan sorunsuz bir şekilde ulaşabilmek için metodumuzun erişim türü Public olarak belirledik. Sonra süslü parantezimizi açıp metodumuzun işlevini yazmaya başladık. İlk olarak metodumuz aracılığı ile ulaşacağımız SQL bağlantı adresimizi tanımladık. **Baglantı() metodunda**

açtık ve geriye bağlantımızdan türöttüğümüz bağlan nesnesini döndürdük. bağlantı

ıan nesnesi bağlantı adresimizi tutacaktır.

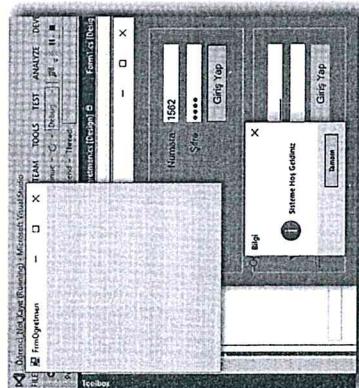
```

    {
        Frm0gretmen frm = new Frm0gretmen();
        frm.Show();
        this.Hide();
    }
    else
    {
        MessageBox.Show("Hatalı Numara ya da Şifre", "Uyarı!");
        MessageBoxButtons.OK, MessageBoxIcon.Stop;
    }
    bg1.baglanti().Close();
}

```

MessageBox.Show("Hatalı Numara ya da Şifre", "Uyarı!", MessageBoxButtons.OK, MessageBoxIcon.Stop);

bg1.baglanti().Close();



Global alanda SQL bağlantısı sınıfımız içeriğine ulaşabilmek için bg1 isminden bir nesne türrettik. Nesne isimlerinde aynılarını yazmak zorunluluğu yoktur. Dildeğiniz ismi verebilirsiniz. Daha sonra öğretmen giriş butonumuza çift tıkladık ve kodlarımızı yazmaya başladık. İlk olarak 4 temel sorgumuzu olan Listele, Ekle, Sil, Güncelle komutlarından hangisini kullanacağımızı belirlememiz gerekiyor.

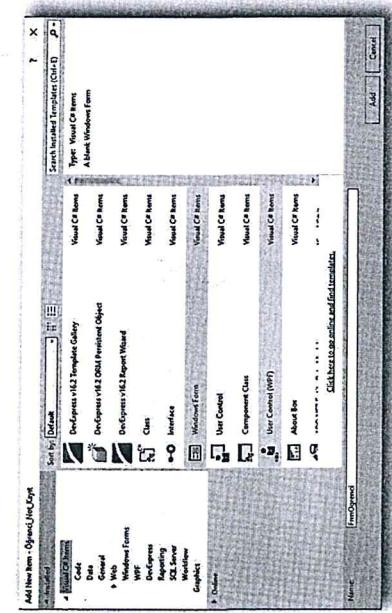
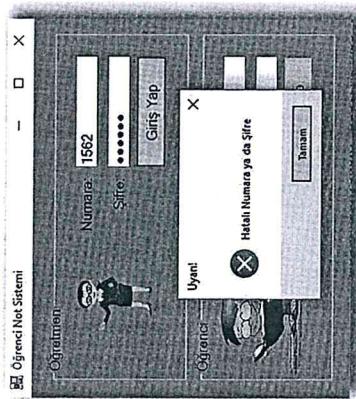
Yapacağımız işlemede amacımız girdiğimiz bilginin doğru olup olmadığını okutmak. Eğer bir yerde okutma işlemi varsa orada mutlaka Select sorgusu kullanılır. Biz de SQL sorgularını yazabilmek için ihtiyacımız olan SqlCommand sınıfından komut isminde bir nesne türettik. Bu satırımızın içine yapmayı istediğiniz SQL sorgusunu yazdık.

Sorgumuzda “**“öğretmen tablosundan bütün bilgileri çek ancak numarası 1. parametreye ve şifresi 2. parametreye eşitse”** anlamı taşıyan bir ifade var. Altında bulunan satırlarda parametrelerinize değer ataması işlemi yaptıktı. 1. parametre öğretmen numara aracından, 2. parametre öğretmen şifre kutucuğuna alınacak. Sonra bu soru ifadesini SqlDataReader ile okuttuk.

Hemen altındaki satırda eğer okuma işlemi başarılı bir şekilde gerçekleşiyorsa yani parametreler ile kutucuktaki ifadeler birbirile uyumluysa o zaman öğretmen formunu aç aksi durumda yani veritabanındaki ifadeyle parametre değerleri birbirine eşleşmiyorsa o zaman bir uyarı ver dedik. İşlemlerimiz bitince bağlantımızı da kapatarak kod bloğumuzu tamamladık.

Ödev

Eğer TextBox ya da MaskedTextBox araçları boş geçilirse lütfen boş geçmeyin şeklinde uyarı veren kod bloğunu projenize ekleyin.
Eğer öğretmenin numarası ya da şifresi yanlışsa o zaman aşağıdaki gibi bir uyarı ekranı gelecektir.



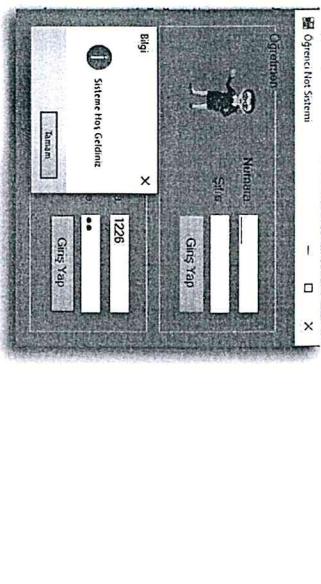
Projemize bir tane daha form ekleyelim. Bu form öğrenci bilgilerini tutacak olan form olsun. Şu an içerisinde herhangi bir araç eklememize gerek yok. Sadece formumu eklememiz yeterli. İlgili formların tasarımını ve kodlamasını giriş formunun işlemleri bittiği zaman yapacağız.

Bos bir öğrenci formu eklediğimizde göre artık öğrencilerimiz için de kodlarımları yazabiliyoruz.

Uygulama5

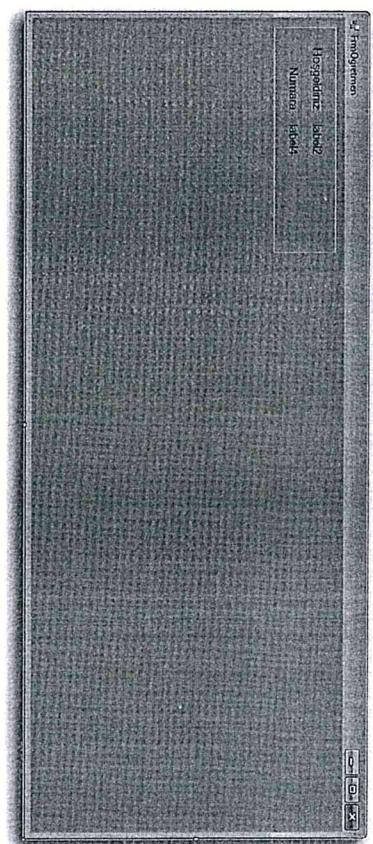
Klavilden numarası ve şifresini giren öğrencinin bilgileri doğru ise sisteme girip kendi bilgilerini listeleyen kodu yazalım.

```
private void Btn0grenciGiris_Click(object sender, EventArgs e)
{
    SqlCommand komut = new SqlCommand("Select * From Tbl0grenci where
    Numara=@p1 and Sifre=@p2", bgl.baglanti());
    komut.Parameters.AddWithValue("@p1", Msk0grenciNumara.Text);
    komut.Parameters.AddWithValue("@p2", Txt0grenciSifre.Text);
    SqlDataReader dr = komut.ExecuteReader();
    if (dr.Read())
    {
        Frm0grenci frm = new Frm0grenci();
        frm.Show();
    }
    MessageBox.Show("Sisteme Hoş Geldiniz", "Bilgi",
    MessageBoxButtons.OK, MessageBoxIcon.Information);
    this.Hide();
}
else
{
    MessageBox.Show("Hatalı Numara ya da Şifre", "Uyarı!",
    MessageBoxButtons.OK, MessageBoxIcon.Stop);
}
bgl.baglanti().Close();
}
```



ÖĞRETMEN FORMU TASARIMI

Öğretmen formumuzu açıp dikdörtgen şeklinde bir boyut ayarlaması yapalım. Sol üst tarafa bir tane Groupbox aracı ekleyelim. Groupbox aracımız içerişine 4 tane Label ekliyoruz. Label aracımızın ikisi etiket görevi görecek, 3. label aracımız giriş formundaki numarayı tutacak son Label aracımız ise sisteme giriş yapan öğretmenin numarasına göre adını ve soyadını yazdıracak.

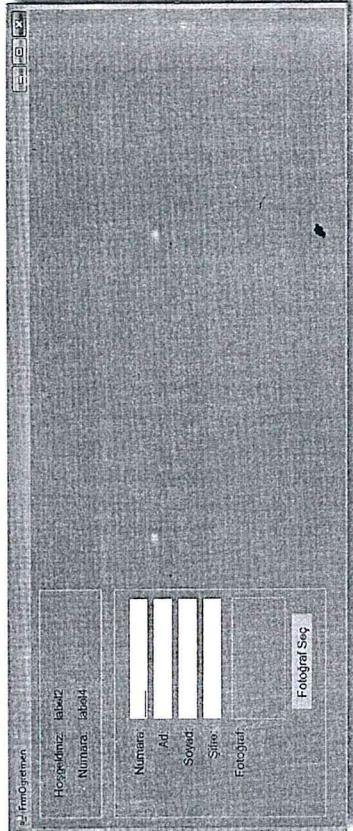


İkinci adımda öğrencinin bilgilerinin kaydedilmesi için formumuz üzerinde küçük bir alan oluşturduk. Bu alanda öğrencinin adı, soyadı, şifresi TextBoxlarına, numarası MaskedTextBox aracına ve fotoğrafı da PictureBox aracından kaydedilecektir.

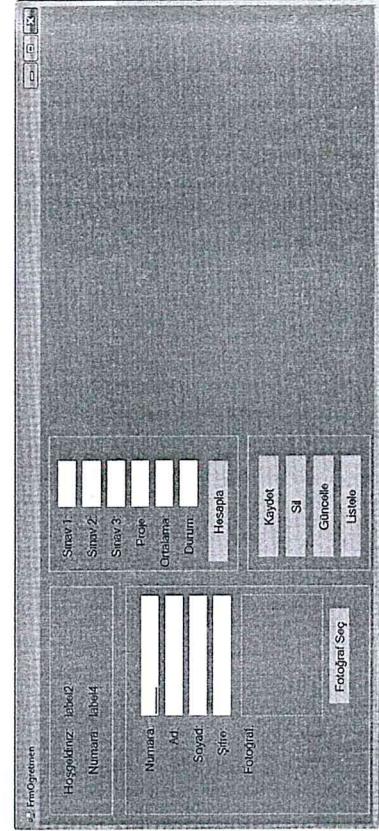
Bir önceki öğretmenin girişiyile yüzde 90 ölçüde aynı kodlara sahip blogumuzda değiştirdiğimiz 3 yer oldu.

Bunların birincisi tablo adı bu kez öğretmen değil öğrencidir. Çünkü bilgileri Tbl0grenci tablosundan çekiyoruz. İkincisi 1. parametre değerini öğretmenin numara kutucuğundan değil öğrencinin numara kutucuğundan aldık. Üçüncüsüne diğer parametremizi öğretmen şifre kutucuğundan değil öğrenci şifre kutucuğundan alarak kod blogumuzu tamamladık. Böylelikle hem öğrenci hem de öğretmen için giriş değerlerimizi yazmış olduk. Öyleyse bir diğer formumuz olan öğretmen formuna geçiş yapabiliriz.

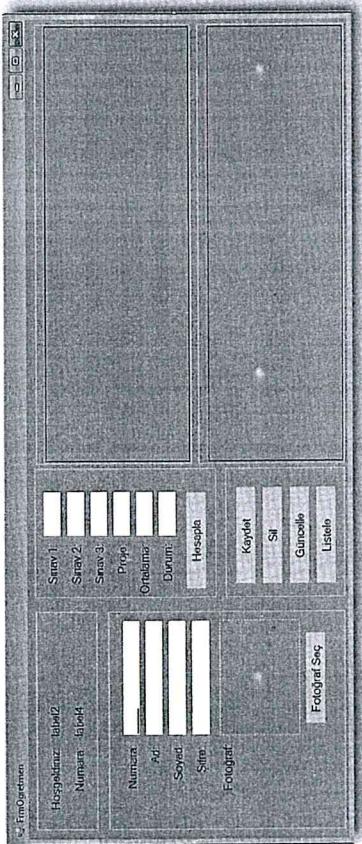
Araçlarınıza adlandırmayı unutmayalım ve diğer panelimiz olan notlar kısmını geçelim.



Üçüncü adımlımızda notlar panelini oluşturuyoruz. Bunun için aşağıda bulunan görseldeki gibi bir tane Groupbox aracı ekleyip içinde 6 tane TextBox ekledik. TextBoxlarımız öğrencilerin sınav notlarını tutacak. Notlar panelimizin değişiklikleri öğrenci panelinin numara bilgisine göre yapacağız. Notlar panelimizin altına işlemleri tutacak Groupboxımızı ekledik. 4 Temel işlem olan ekle, sil, güncelle ve listele bu alandan gerçekleştirilecek.



Son olarak dördüncü adımda formumuzun sağ tarafına iki tane Groupbox ekleyip her birinin içine birer tane DataGridView aracını dahil ettiğimiz. Bu DataGridView araçlarınızın birisi öğrenci listesini diğeride notlar listesini tutacaktır. Böylece form tasarımımızı bitirmemeler için de bitirmiş olduk.



Öğretmenler formu için yapmamız gereken işlemi adım adım yazalım;

- » Formlar arası veri yaşıma işlemiyle numaranın taşınması
- » Taşınan numaraya göre öğretmenin ad soyad bilgisini getirme
- » Öğrencileri listeleye
- » Notları listeleye
- » Yeni öğrenci kaydi
- » Öğrenci bilgilerinin araçlara taşınaması
- » Öğrenci silme
- » Öğrenci güncelleme

FORMLAR ARASI VERİ TAŞIMA İŞLEMİYLE NUMARANIN TAŞINMASI

Bir formdan diğerine veri taşıma işlemi gerçekleştireceğiz. Sisteme giriş yapan öğretmenin numarasını üzerinde çalıştığımız FrmÖğretmen formuna taşıma-mız gerekiyor. Formlar arası veri taşıma işlemlerinde devreye erişim belirteçleri girecek. Bir veriyi diğer forma taşıtmamız için ilk olarak tasınacak formda bir adet global ve public değişkene ihtiyacımız var. Önce öğretmen formumuz için son- ra da giriş formu için kodlarımızı yazalım.

NOT Label2 ve Label4 araçlarını adlandırmayı unutturmayalım. Bu araçların hangi olduğunu hatırlamak için önceki sayfada bulunan tasarım görseline bakabilirsiniz.

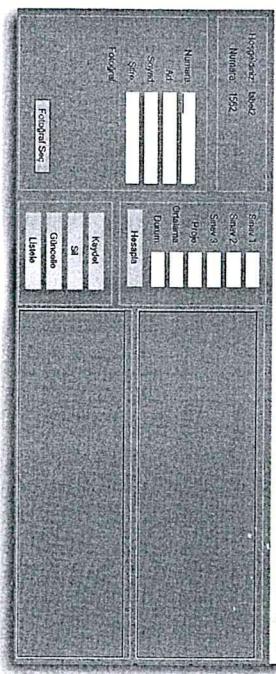
Üyulana6
Öğretmen formu için değer ataması yapacağımız global bir değişken kodunu yazalım.

```
public string numara;
private void FrmOgretmen_Load(object sender, EventArgs e)
{
    LblNumara.Text = numara;
}
```

Öğretmen formumuza çift tıklayıp global alanda yani herkes tarafından erişim sağlanacak kapşam dışı noktada string türünde public olarak bir değişken tanımladık. Tanımladığımız bu değişkenimizin alacağı değeri yazdırmak için formumuzun load yani yüklenme kod bloğu içinde LblNumara aracımızın Text özelliğine numara değişkenimizden gelen değerini yazdırdık.

Üyulana7

Öğretmenin numara bilgisini giriş formundan öğretmen formuna gönderecek kodu yazalım.



NOT
Bir formdan diğerine veri taşıma işlemi; veri nereye taşınacağı o forma ait bir nesne türetilecek gerçekleştirilecektir.

TAŞINAN NUMARAYA GÖRE ÖĞRETMENİN AD SOYAD BİLGİSİNİ GETİRME

Öğretmenin numara bilgisi sisteme taşındı. Şimdi taşınan numaraya göre adı ve soyadı getirmek işlemi yapacağız. Yani tipki herhangi bir sosyal medyanın oturum açma yöntemi gibi; Mail adresi ve şifremiz yazıldıkları sonra sisteme giriş yapınca kendi ad – soyad bilgimizin karşımıza çıkması gibi işlemler yapacağız. Ad ve soyad bilgisi veritabanından Select sorgusunun numara şartına göre çekeceğiz.

Üyulana8

Sisteme giriş yapan öğretmenin numarasına göre ad ve soyad bilgisini yazdırın kodu yazalım.

```
if (dr.Read())
{
    FrmOgretmen frm = new FrmOgretmen();
    frm.numara = MskOgretmenNumara.Text;
    frm.Show();
}

MessageBox.Show("Sisteme Hoş Geldiniz", "Bilgi",
MessageBoxButtons.OK, MessageBoxIcon.Information);
this.Hide();
```

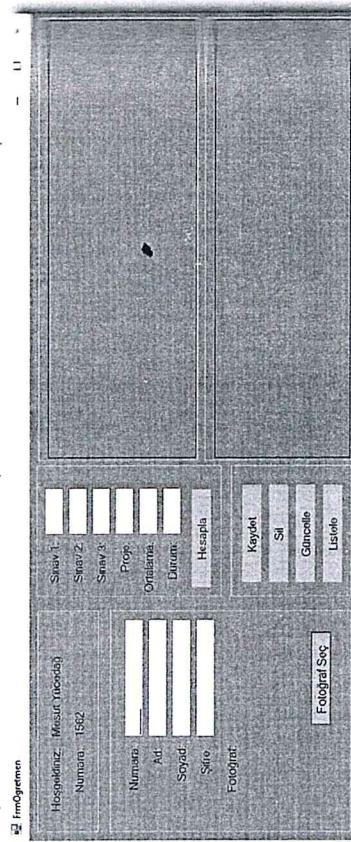
Giriş formumuzda daha önce öğretmen ve öğrenci için giriş kodlarını yazmıştık. Öğretmen girişi içinde diğer forma yönlendirme kod bloğunda bulunan Frm.Show() metodundan önce veri taşıma görevini üstlenecek frm. numara=MskOgretmenNumara.Text komutu ile taşıma işlemini gerçekleştirdik.

Numara değişkenimize ulaşmak için öğretmen formumuza ait bir nesneye ihtiyacımız vardır. Biz zaten frmOgretmen frm=new frmOgretmen komutunu daha önce yazdığımız için ekstra bir nesne türetme gerekii duymadık. Bunun yerine türüttüğümüz frm nesnesine değer ataması yaparak kod bloğumuzu tamamladık. O zaman şöyle bir not düşerek bu başlığını kapatalım;

```

SqlDataReader dr = komut.ExecuteReader();
while (dr.Read())
{
    LblAdSoyad.Text = dr[1] + " " + dr[2];
    bg1.baganti().Close();
}

```



İlk olarak kütüphaneler kısımına SQL kütüphanemizi ekleyerek kodlarımızı yazmaya başladık. Global alanda bağlantı sınıfımızdan bir tane nesne tırtıltık. Bu nesneyi öğretmen formumuzun hemen hemen her yerinde kullanacağız. Formumuza load kısmına geri döndük. Daha önce yazmış olduğumuz numara bilgiyi taşıma komutlarının hemen altında SQL için kodlarımızı yazmaya başladık. SqlCommand sınıfından komut isminde bir nesne türeterek bu kod沼umuzun içine SQL sorgumuzu yazdık. SQL sorgumuzda şartımız öğretmen formundaki numarası parametre1'e eşit olan kaydı getir olsun dedik. Sonra alt satırda yanılı komut.parameters ile başlayan satırda numara değerimiz tekbül edecek olup ifadesinin değerini nereden alacağımı bildirdik.

Burada değerimizi numara değişkenimizden alacak. Numara değişkenimi/değerini giriş formundaki **MaskedTextBox** aracından almaktı. Sonra SqlDataReader komutıyla dr isminde bir nesne türetip bu nesnemizi komut sorgumuzla ilişkilendirdik. Son olarak dr komutu okuma işlemi süreci **LblAdSoyad Label** aracımıza dr nesnemizden gelen 1 ve 2. sırada bulunan indeksteki değerlerimizi yazdırıktı. Burada 1 ve 2. sıradaki değerler SQL tablomuzdaki sıralardır.

Örneğin 0.indexte id alanı, 1'de ad, 2'de soyad, 3'de numara ve 4'de şifre alanları bulunmaktadır. Bağlantımızı kapatarak kod沼umuzu sonlandırdık.

ÖĞRENCİLERİ LISTELEME

Öğretmen formuna giriş yapıldığı zaman form yüklenince gerçekleşmesini istediğiniz bir diğer başlık öğrenci listeini görüntülemektir. Öğrencileri DataGridView aracımızda listeleyebiliriz. Öğrenci listelemeye işlemi Select sorgusuya yapılacaktır. Oldukça kolay bir kullanım yapısı olan listeleme kodlarını beraber yazalım.

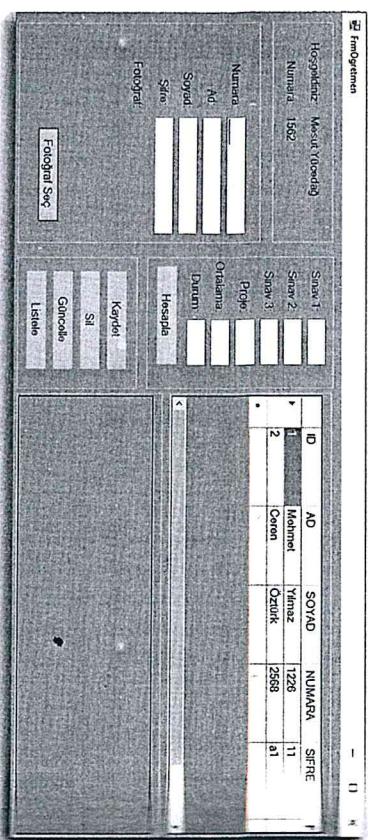
NOT Projelerinizde özellikle ödev veya dönem sonu bitirme uygulaması gibi projelerinizdeki verilerin içine olabildiğince çok veri girişi yapın. Ne kadar çok veri olursa projeniz o kadar dolu gözükecektir.

```

void OgrenciListesi()
{
    SqlCommand komut = new SqlCommand("Select * From Tbl0grenci",
        bg1.baganti());
    SqlDataAdapter da = new SqlDataAdapter(komut);
    DataTable dt = new DataTable();
    da.Fill(dt);
    dataGridView1.DataSource = dt;
}

private void FrmOgretnmen_Load(object sender, EventArgs e)
{
    //Form Loadda metodumuzu çağırıyoruz.
    OgrenciListesi();
}

```



Öğrenci Listesi isminde bir metot oluşturduk. Burada metot oluşturmadan sebebi ilerleyen sayfalarда yapacağımız ekleme, silme ve güncelleme işlemelerinden sonra da listeleye işlemini yapacağımız içindir.

SqlCommand sınıfından komut isminden bir nesne türetip bu nesnemiz aracılığı ile sorgumuzu yazdık. SqlDataAdapter komutu DataGridView'aracımıza verilecektir yazdırma için sorgumuzu üzerinde tutacak olan köprü görevi gören sınıfır. Bu sınıfımızı komut nesnemizle ilişkilendirdik. Geçtiğimiz bölgülerde tüm bu kavramları detaylı olarak açıkladığımız için burada biraz daha yalın bir şekilde geçiyoruz. Bir tane veri tablosu nesnesi oluşturup verileri tablonun içine atıp Datagridview aracımızın veri kaynağı olarak bu veri tablomuzu göstererek kodlarımızı tamamlamış olduk. Son olarak formumuzu çağrırdık.

NOTLARI LISTELEME

Öğrencileri listeledik. Şimdi de diğer DataGridView aracımıza öğrencilerimizin notlarını listeleyeceğiz. Notları listelemek için Select sorgusu kullanmayaçğız bunun yerine önceki sayfalarda oluşturmuş olduğumuz prosedürü kullanmak yeterli olacaktır. Çünkü notlar tablosunda ilişki var ve öğrenci ID değerini öğretmenin anlamayacağından düşünerek ID değerlerine karşılık gelen ad soyad bilgilerini prosedürümüz içinde birleştirme sorgusuya oluşturmuştu.

İşlevlər 10
Öğretmen formuna giriş yaptığı zaman 2. DataGridView aracımızda öğrenci notlarını listeleyelim.

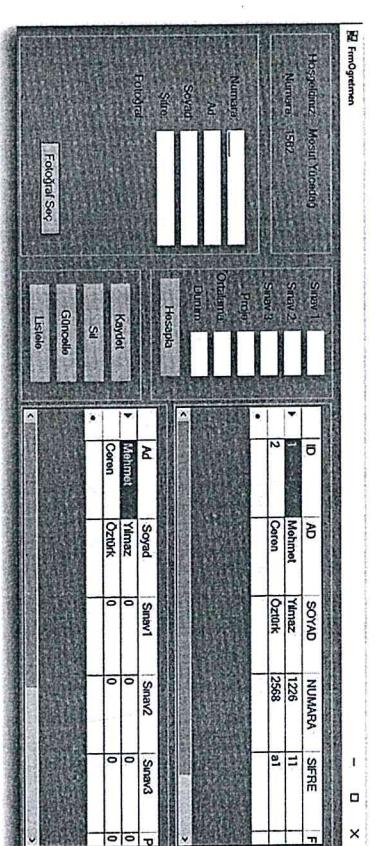
```

void NotListesi()
{
    SqlCommand komut = new SqlCommand("Execute Öğrenciler", bgl);
    SqlDataAdapter da = new SqlDataAdapter(komut);
    DataTable dt = new DataTable();
    da.Fill(dt);
    dataGridView2.DataSource = dt;
}

private void FrmÖğretmen_Load(object sender, EventArgs e)
{
    //Form Loadda 2.metodumuzu çağırıyoruz.

    NotListesi();
}

```



Öğrenci Listesinde olduğu gibi aynı formatta kodlarımızı yazdık. Metodumuzun ismini Not Listesi olarak belirledik. Tek bir farkı var. Bu kez sorgu alanı içerisinde Select sorgusu yerine direkt prosedürümüzü yazıp çalıştmak istediğimiz için Execute Öğrenciler yazarak kod bloğumuzu tamamlamış olduk. Yani SQL üzeinde sorgularımızı nasıl yaziyorsak C#da da aynı formatta yazdığımız zaman sorgumuz sorunsuz bir şekilde çalışacaktır.

YENİ ÖĞRENCİ KAYDI

Formumuza yeni öğrenci eklemesi gerçekleştireceğiz. Biz öğrenci eklediğimiz anda tetkileyicimiz çalışacak ve notlar tablosuna ilgili öğrencimizin id değeri otomatik olarak eklenecektir. Ekleme işlemi için Insert sorgusunu kullanacağız. Kodlarımıza yazmaya başlamadan önce formumuza bir tane OpenFileDialog aracı ekleyip öğrencimizin resmini hafızaya alalım. Daha sonra da Insert sorgumuzu yazalım.

Uygulama 12

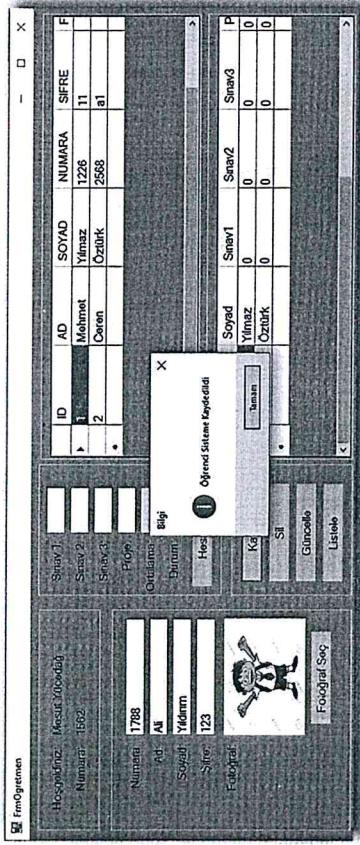
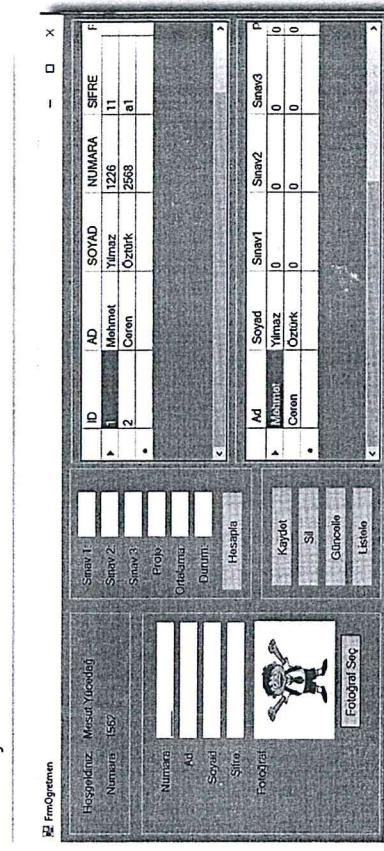
Öğrenci bilgilerini girip kaydetme işlemini gerçekleştirelim.

```
private void BtnKaydet_Click(object sender, EventArgs e)
{
    SqlCommand komut = new SqlCommand("insert into tbogrenci
(ad,soyad,numara,sifre,fotograf) values (@p1,@p2,@p3,@p4,@p5)", bgl.baglanti());
    komut.Parameters.AddWithValue("@p1", TxtAd.Text);
    komut.Parameters.AddWithValue("@p2", TxtSoyad.Text);
    komut.Parameters.AddWithValue("@p3", NskNumara.Text);
    komut.Parameters.AddWithValue("@p4", TxtSifre.Text);
    komut.Parameters.AddWithValue("@p5", Fotograf);
    komut.ExecuteNonQuery();

    MessageBox.Show("Öğrenci Sisteme Kaydedildi", "Bilgi",
MessageBoxButtons.OK, MessageBoxIcon.Information);

    bgl.baglanti().Close();
    OgrenciListesi();
    NotListesi();
}

string Fotograf;
private void BtnFotograflSec_Click(object sender, EventArgs e)
{
    openFileDialog1.ShowDialog();
    OpenFileDialog1.FileName;
    Fotograf= openFileDialog1.FileName;
    pictureBox1.ImageLocation = Fotograf;
}
```



Global alanda fotoğraf isminde bir string değişken tanımladık. Bu değişkeni hem fotoğraf sec butonu içinde hem de kaydetme komutu içinde kullanacağız. Fotoğraf seç butonumuza çift tıkladık ve kodlarımıza yazmaya başladık. İlk önce dosya açma işlemlerinde kullandığımız OpenFileDialog aracımız yardımıyla dialog penceremizi açtık. Daha sonra fotoğraf değişkenimize OpenFileDialog aracımızdan seçmiş olduğumuz dosyanın yolunu atadık. Son olarak PictureBox aracımızda fotoğraf değişkeninin hafızasının tuttuğu sırasında tuttuğu görüntünüledik.

Kaydet butonumuza çift tıkladık ve komut isminde bir nesne oluşturarak kodlarımıza yazmaya başladık. Ekleme işlemi yapacağımız için Insert DML komutuzu kullandık. Daha sonra tablo adımızı yazıp hangi sütunlara ekleme yapacağımız o sütunları bir parantez bloğu içine yazdık. Bu sütunlar değerlerini alabilmesi için hiyerarşik bir yapıda atama sağlayan parametrelerimizi yazdık. Böylece values komutunun sol tarafındaki değerler SQL sütunlarını, sağ tarafındaki değerler ise bu sütunlara tekabül edecek parametreleri ifade eder.

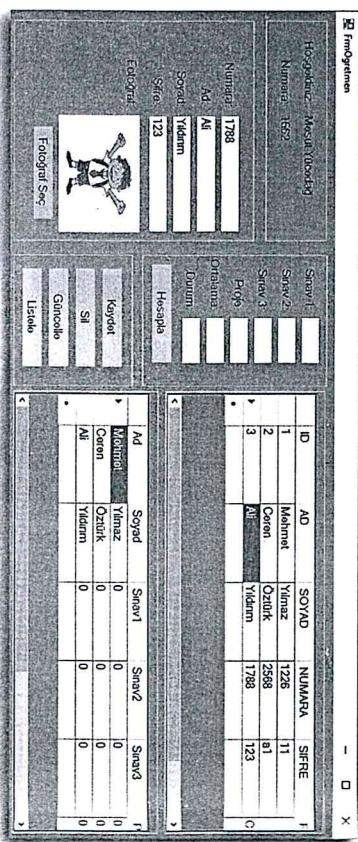
Daha sonra her bir parametreye değer ataması yapıp ExecuteNonQuery metodunu ile komutlarımıza çalıştırmasını programa bildirdik. Kaydetme işleminden sonra hem öğrenci hem de not listesi metotlarımıza çağrırdık.

ÖĞRENCİ BİLGİLERİNİ ARAÇLARA TAŞIMA

Öğrencilerin ad, soyad veya benzeri bilgilerinin güncellenmesi için araçlara taşınması şarttır. Ayrıca aynı şekilde notların hesaplanması işleminde de güncellemeye ihtiyacımız olacak. Bunun iki adımı var. Birincisi tıklanan kişiye ait not bilgilerinin araçlara taşınması. Diğer iki adımı var. Birincisi tıklanan kişiye ait not bilgilerinin notlu alanına taşınması şeklinde olacaktır. O halde ilk önce tıklanan hücrenin değerini araçlara taşıyalım. Bunun için Datagridview aracımızın olayları kısmındaki Cell Click yani hücre tıklanması alanına çift tıklyoruz.

Uygulama 13

Datagridview aracına tıkladığımız zaman tıklanan hücredeki verileri araçlarla taşıyan komutu yazalım.



```
private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
    int secilen = dataGridView1.SelectedCells[0].RowIndex;
    TxtAd.Text = dataGridView1.Rows[secilen].Cells[1].Value.ToString();
    TxtSoyad.Text = dataGridView1.Rows[secilen].Cells[2].Value.ToString();
    MskNumara.Text = dataGridView1.Rows[secilen].Cells[3].Value.ToString();
    TxtSifre.Text = dataGridView1.Rows[secilen].Cells[4].Value.ToString();
    pictureBox1.ImageLocation = dataGridView1.Rows[secilen].Cells[5].Value.ToString();
}
```

Seçilen isminde bir değişken tanımlayıp bu değişkene değer olarak satır indeksini atadık. Yani tıklamış olduğumuz satırın indeksini seçilen adlı değişkenimize atadık. Daha sonra seçilen alanımıza göre ad, soyad, numara ve şifre bilgilerini araçlara atadık. Tüm araçların **Text** özelliğini **u****k****ü****n****a****l****a****r****a****k****y****a****z****d****i****r****ü****r****m** işlemi yaparken Picturebox aracımızın **ImageLocation** özelliğini kullandık. Böylece tıkladığımız değerin adresinde bulunan resmi Picturebox aracımızda göstermiş olduk.

Uygulama 14

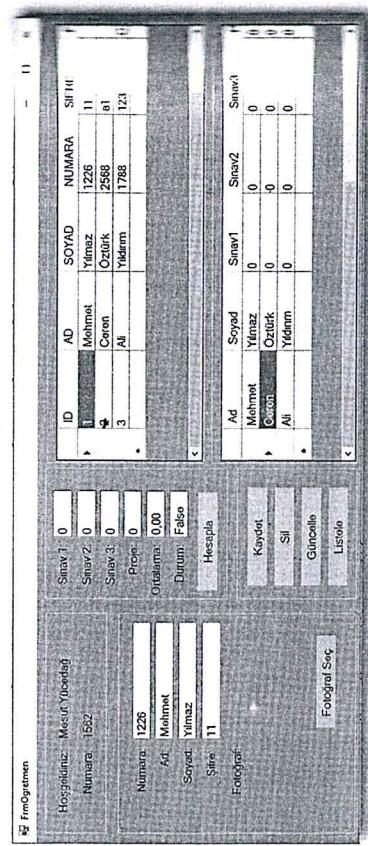
Seçilen numaraya ait not bilgilerini ilgili araçlara aktaralım.

```
private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
    int secilen = dataGridView1.SelectedCells[0].RowIndex;
    TxtAd.Text = dataGridView1.Rows[secilen].Cells[1].Value.ToString();
    TxtSoyad.Text = dataGridView1.Rows[secilen].Cells[2].Value.ToString();
    MskNumara.Text = dataGridView1.Rows[secilen].Cells[3].Value.ToString();
    TxtSifre.Text = dataGridView1.Rows[secilen].Cells[4].Value.ToString();
```

```
    pictureBox1.ImageLocation = dataGridView1.Rows[secilen].Cells[5].Value.ToString();
}

SqlCommand komut = new SqlCommand("Select * From TblNotlar where
        NotId=(Select ID From TblOgrenci Where Numara=@p1)", bg1.baglanti());
komut.Parameters.AddWithValue("@p1", MskNumara.Text);
SqlDataReader dr = komut.ExecuteReader();
while (dr.Read())
{
    TxtSınav1.Text = dr[1].ToString();
    TxtSınav2.Text = dr[2].ToString();
    TxtSınav3.Text = dr[3].ToString();
    TxtProje.Text = dr[4].ToString();
    TxtOrtalama.Text = dr[5].ToString();
    TxtDurum.Text = dr[6].ToString();
}
```

```
    bgl.baglanti().Close();
}
```



Bir önceki kod blogümüz içinde kodlarınıza yazmaya devam ediyoruz. Ama kımız seçmiş olduğumuz öğrencinin not bilgilerini listelemek. Bu noktada bir all sorğu kullanmanız gerekecek çünkü notlar ile öğrenciler arasında ortak olan tek alan ID alanıdır. Öğrenciler tablosundaki ID değeri ile notlar tablosunda ID değerleri aynı olan değerini listelemiş olduk. Burada şartımız parametre 1'te karşılık gelen yanı numara sütununun değerinin MskNumara aracından alınacak olmalıdır. Araçlarımıza not değerlerini aktarır kod bloğumuzu tamamladık.

ÖĞRENCİ BİLGİLERİNI SILME

Silme işlemini kitabımızın SQL böümlerinde detaylı bir şekilde vermiştık. Bu projemizde silme işlemi kullanılmayacağz. Çünkü içerisinde ilişki bulunan tablolarda silme işlemi tercih edilen bir durum değildir. Bunun yerine aktif pasif işlemi yapılır. Şöyle ki; her öğrencinin yanına aktiflik durumunu belirten bir stütun eklenir. Eğer öğrenci silinmek istenirse delete komutu kullanılmaz bunun yerine update komutu ile aktiflik durumu değiştirilir. Öğrenci pasif hale getirili. Zira öğrencinin silinmesi ilişkide hata doğuracağından dolayı aktif ve pasif hal getirme işlemi daha çok tercih edilir.

ÖĞRENCİ BİLGİLERİNI GÜNCELLEME

Güncelleme işlemi hatalı bilgi girişi durumunda çokça kullanılan bir yapıdır. Kişiin şifresinin değiştirilmesi istenebilir veya soyadı yanlış girilmiş olabilir. Böyle durumlarda güncelleme kullanılması gerekmektedir. Güncelleme işlemi için DML komutları içindeki update komutu kullanılmaktadır.

Üyelik 15

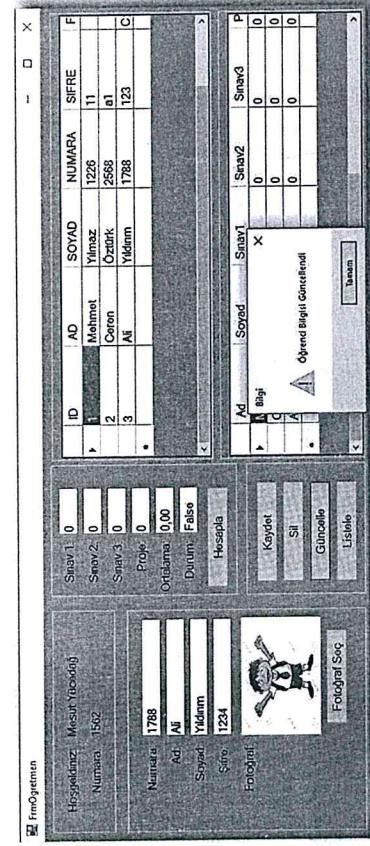
```
Seçilen öğrencinin bilgilerini güncelleyen komutu yazalım.
```

```
private void BtnGuncelle_Click(object sender, EventArgs e)
{
    //Öğrenci Bilgilerini Güncelleme

    SqlCommand komut = new SqlCommand("update tbloğrenci set ad=@
p1,soyad=@p2,sifre=@p3,fotograf=@p4 where numara=@p5", bg1.baglanti());
    komut.Parameters.AddWithValue("@p1", TxtAd.Text);
    komut.Parameters.AddWithValue("@p2", TxtSoyad.Text);
    komut.Parameters.AddWithValue("@p3", TxtSifre.Text);
    komut.Parameters.AddWithValue("@p4", Fotograf);
    komut.Parameters.AddWithValue("@p5", MskNumara.Text);

    komut.ExecuteNonQuery();
}

MessageBox.Show("Öğrenci Bilgisi Güncellendi", "Bilgi",
 MessageBoxButtons.OK, MessageBoxIcon.Warning);
bg1.baglanti().Close();
ÖgrenciListesi();
```



ilk önce öğrencinin kişisel bilgilerini güncelleterek başladık. Bir sonraki adımda öğrencinin not bilgisini güncelleyeceğiz. Güncelleme işlemi için update sorgusu kullanıyoruz. Her bir sütunu bir parametreden alıyoruz. Güncelleme işleminde şart sütununu numara olarak belirledik. Yani numarası parametre 5'e eşit olan alana göre güncelleme işlemi gerçekleştirildik. Fotoğraf sütununu da global alanda tanımlımiş olduğumuz fotoğraf değişkeninden alarak güncelleme işlemini tamamladık. İlk noktada yalnızca öğrencinin profil bilgilerini güncelleyoruz. Not bilgisini güncellemeden önce hesaplama işlemi yapacağız.

Uygulama 16

Hesapla buttonuna tıkladığımız zaman öğrencinin notlarını hesaplayıp durumunu yazdırın kodu yazalım. (Durum 50'den büyükse veya eşitse geçti, aksi durumda kaldı şeklinde düzenlenecektir)

```
private void BtnHesapla_Click(object sender, EventArgs e)
{
    double sınav1, sınav2, sınav3, proje, ortalama;
    sınav1 = Convert.ToDouble(txtSınav1.Text);
    sınav2 = Convert.ToDouble(txtSınav2.Text);
    sınav3 = Convert.ToDouble(txtSınav3.Text);
    proje = Convert.ToDouble(txtProje.Text);
    ortalama = (sınav1 + sınav2 + sınav3 + proje) / 4;
    TxtOrtalama.Text = ortalama.ToString();
    if (ortalama >= 50)

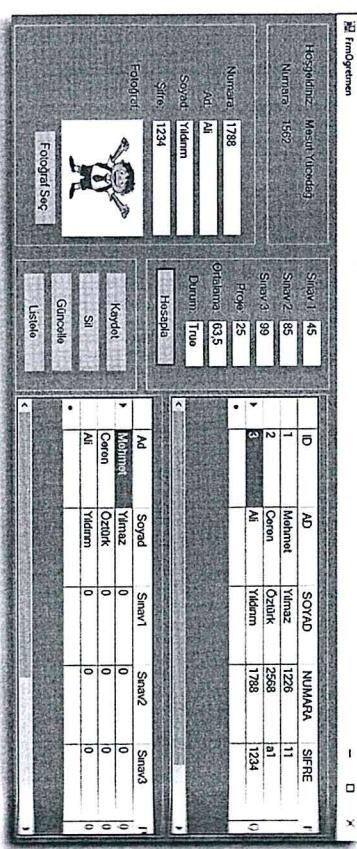
    {
        TxtDurum.Text = "True";
    }
    else
    {
        TxtDurum.Text = "False";
    }
}

//Öğrenci Bilgilerini Güncelleme

private void Btngüncelle_Click(object sender, EventArgs e)
{
    SqlCommand komut = new SqlCommand("update tblogrenci set ad=@
    soyad=@p2,sıfır=@p3,fotograf=@p4 where numara=@p5", bgl.bağlanti());
    komut.Parameters.AddWithValue("@p1", TxtAd.Text);
    komut.Parameters.AddWithValue("@p2", TxtSıfır.Text);
    komut.Parameters.AddWithValue("@p3", Fotograf);
    komut.Parameters.AddWithValue("@p4", MskNumara.Text);
    komut.ExecuteNonQuery();
    bgl.bağlanti().Close();
}

//Not Bilgilerini Güncelleme

SqlCommand komut2 = new SqlCommand("update TblNotlar set Sınav1=@
    p1,Sınav2=@p2,Sınav3=@p3,Proje=@p4,Ortalama=@p5,Durum=@p6 where 0RID=(Select
    ID From TblOgrenci Where Numara=@p7)", bgl.bağlanti());
komut2.Parameters.AddWithValue("@p1", txtSınav1.Text);
komut2.Parameters.AddWithValue("@p2", txtSınav2.Text);
komut2.Parameters.AddWithValue("@p3", txtSınav3.Text);
komut2.Parameters.AddWithValue("@p4", txtProje.Text);
komut2.Parameters.AddWithValue("@p5", Convert.ToDecimal(TxtOrtalama.Text));
komut2.Parameters.AddWithValue("@p6", TxtDurum.Text);
komut2.Parameters.AddWithValue("@p7", MskNumara.Text);
komut2.ExecuteNonQuery();
```



Hesapla buttonumuza çift tıklayıp kodlarımızı yazmaya başladık. İlk olarak double türünde değişkenlerimizi tanımladık. Burada double kullanmamızın sebebi ortalamanın ondalıklı olarak sonuç verme ihtimalindendir. Değişkenlerimizi double olarak tanımladığımız için klavyeden alınacak dönüşüm işlemleri içinde ToDouble dönüştürmünü tercih ettilik.

Daha sonra ortalamayı hesaplamak için sınavlarımızı ve proje notumuzu toplayıp 4'e bölecek ortalamamızı hesapladık. Eğer ortalama 50'den büyük veya eşitse durum TextBox aracına True değerini yazdırırdık. True yazmamızın sebebi durum alanını SQL'de bit olarak tuttuğumuz içindir. Bit veri tipi yalnızca 1-0 ya da True-False şeklinde değer alındıdan biz de durum alanını geçti kaldı yerine True - False olarak kullanmayı tercih ettik. Böylece hesaplama işlemini tamamladık. Şimdi notları da güncelleme işlemine ekleyelim.

Uygulama 17

Güncelle buttonuna tıkladığımız zaman hem profili bilgilerini hem de not bilgilerini güncelleyen kod yazalım.

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

```
{ //Öğrenci Bilgilerini Güncelleme
```

```
p1,soyad=@p2,sıfır=@p3,fotograf=@p4 where numara=@p5", bgl.bağlanti());
```

```
komut.Parameters.AddWithValue("@p1", TxtAd.Text);
komut.Parameters.AddWithValue("@p2", TxtSıfır.Text);
komut.Parameters.AddWithValue("@p3", Fotograf);
komut.Parameters.AddWithValue("@p4", MskNumara.Text);
komut.ExecuteNonQuery();
bgl.bağlanti().Close();

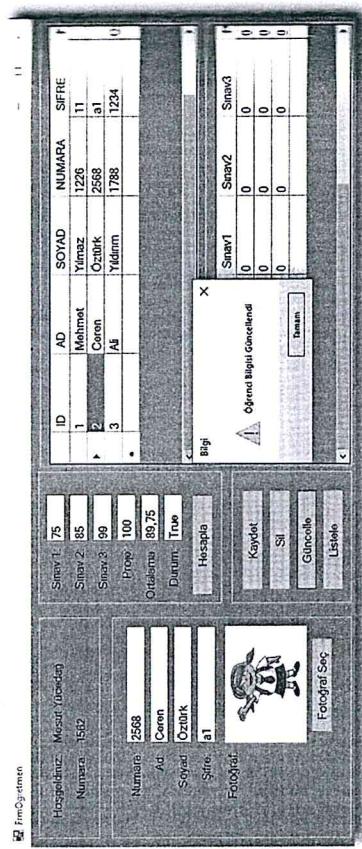
//Not Bilgilerini Güncelleme

SqlCommand komut2 = new SqlCommand("update TblNotlar set Sınav1=@
    p1,Sınav2=@p2,Sınav3=@p3,Proje=@p4,Ortalama=@p5,Durum=@p6 where 0RID=(Select
    ID From TblOgrenci Where Numara=@p7)", bgl.bağlanti());
komut2.Parameters.AddWithValue("@p1", txtSınav1.Text);
komut2.Parameters.AddWithValue("@p2", txtSınav2.Text);
komut2.Parameters.AddWithValue("@p3", txtSınav3.Text);
komut2.Parameters.AddWithValue("@p4", txtProje.Text);
komut2.Parameters.AddWithValue("@p5", Convert.ToDecimal(TxtOrtalama.Text));
komut2.Parameters.AddWithValue("@p6", TxtDurum.Text);
komut2.Parameters.AddWithValue("@p7", MskNumara.Text);
komut2.ExecuteNonQuery();
```

```

MessageBox.Show("Öğrenci Bilgisi Güncellendi", "Bitti!");
bgl.baglanti().Close();
OgrenciListesi();
NotListesi();
}

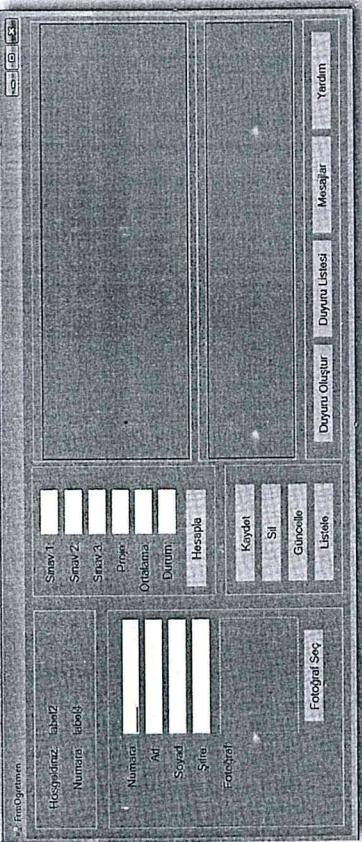
```



Her iki kod bloğunu yanı hem profil bilgileri güncelleme işlemini hem de not güncelleme işlemini aynı buton içinde gerçekleştirdik. Profil bilgileri güncelleme kısmından öğrenci bilgilerini listeleye metodunu ve mesaj kutusunu kaldırık çünkü bu işlemi notları güncelledikten sonra yapmak daha doğru olacaktır.

Not bilgisi güncelleme açıklaması satır ile başlayan alanımızda komut2 isminde bir SqlCommand nesnesi oluşturduk. Bu nesneye notları tablomuzun güncellenmesi için gerekli olan sorgu değerini atadık. Parametre 5de sadece küçük bir dönüştürme karşılaşacaktır. Yani ID değeri öğrenci tablosundaki ID ile uyan öğrencinin numarasına göre güncelleme yaptıktı. Böylece alt sorguyu bir kez daha kullanarak bir adım daha fazla pekiştirmiş olduk. Sırasıyla sütunlarımıza ait parametrelere araçlara karşılık gelen değerleri atadık. Parametre 5de decimal olduğu için C# Form üzerinde decimal türüne dönüştürdük. Aksi durumda hata vermesi muhtemeldir. Son olarak bağlantıyı kapatıp not ve profil bilgilerini listeleyip mesaj kutusunda güncelle şeklinde bir uyarı vererek kod bloğumuzu tamamlamış olduk.

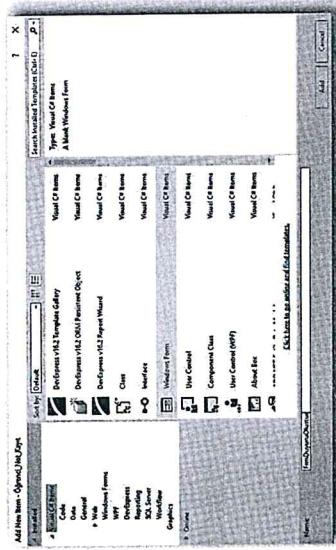
Öğretmen ile ilgili temel öğrenci – not ilişkisini tamamladığımız göre şimdiden öğretmen formunda küçük bir tasarım değişikliğine gideceğiz. Öğretmenimiz için bir tane hızlı erişim menüsü oluşturalım. Bunun için 2. Daatagridview aracımızın hemen altına şöyle bir tasarım eklemesi yapalım.



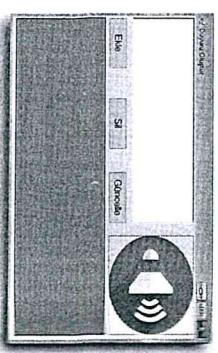
4 tane yeni buton eklemesi yaptıktı. Bu butonlarımız sırasıyla duyuru oluştur, duyuru listesi, mesajlar ve yardım menülerini açacaktır. Öyleyse öğretmen formunda ilk olarak duyuru oluşturmaya başlayalım.

DUYURU İŞLEMLERİ OLUŞTURMA FORMU

Öğretmenin tüm öğrencilere ileteceği mesajlar olabilir. Örneğin; dersin iptali, ders saatini değiştirmesi, sınav sonuçları vs. böyle durumlarda her bir öğrenci tek tek mesaj atmak yerine bir duyuru oluşturarak işlemlerini çok daha pratik bir şekilde çözebilir. Project menüsünden Add Windows Form seçeneğine tıklayarak yeni bir form ekleyelim.



Formumuzu ekledikten sonra küçük bir tasarım yapalım. Formumuz ekleme, silme ve güncelleme blokları üzerine çalışacaktır. O halde şöyle bir tasarım yapabiliyoruz.



Duyuru oluştur butonumuza tıklayıp duyurular formundan frm isminde bir nesne tırtıltı. Daha sonra bu nesnenin show özelliğini kullanarak yeni formumuzu açtık.

Uygulama 19

Duyurular formu açıldığı zaman duyuruları SQL'den DataGridView aracımıza çekmek komutu yazalım.

```
SqlBaglantisi bgl = new SqlBaglantisi();
void Liste()
{
    SqlCommand komut = new SqlCommand("Select * From TblDuyurular",
        bgl.baglanti());
    SqlDataAdapter da = new SqlDataAdapter(komut);
    DataTable dt = new DataTable();
    da.Fill(dt);
    dataGridView1.DataSource = dt;
}
```

Duyuru oluşturma formumuzda sol üst köşede duyuru methinin yazılacağı bli RichTextBox aracı ekledik. Altına ekleme, silme ve güncelleme işlevlerini teliklemek için birer Button, sağ tarafa boş kalmasın ve biraz daha görsel olarak sık dursun diye bir duyuru resmi ve alt kısma da duyuruları listeleyeceğimiz bir TagGridview aracı ekleyerek tasarımımızı tamamladık.

İlk olarak formumuz yüklendiği zaman duyurularımızı DataGridView aracımızdırla listeleyerek kodlarımızı yazmaya başlayalım. Ancak bir adım öncesinde duyurular formuna geçiş yapmak için öğretmen formuna eklediğimiz duyuru oluştur butonuna tıklayıp bu formumuza geçiş yapalım.

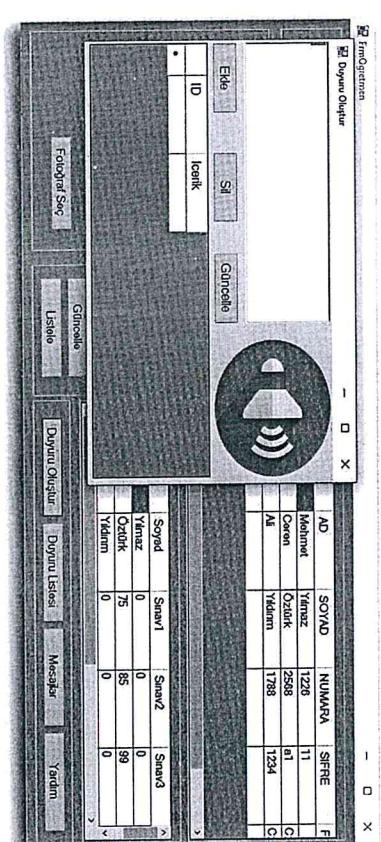
Uygulama 18

Duyuru oluşturma buttonuna tıkladığımız zaman duyuru oluşturma ve düzenlemeye formunu açan kodu yazalım.

```
private void BtnDuyuruOlustur_Click(object sender, EventArgs e)
{
    FrmDuyuruOlustur frm = new FrmDuyuruOlustur();
    frm.Show();
}

private void FrmDuyuruOlustur_Load(object sender, EventArgs e)
{
    Liste();
}

private void BtnDuyuruOlustur_Click(object sender, EventArgs e)
{
    FrmDuyuruOlustur frm = new FrmDuyuruOlustur();
    frm.Show();
}
```



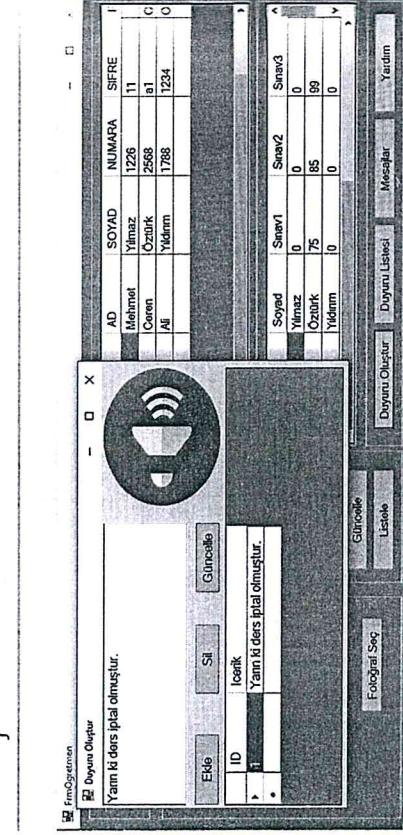
SQL Bağlantısı sınıfımızı çağırarak kodlarımızı yazmaya başladık. Kütüphaneler kismına SQLClient kütüphanesini eklemeyi unutmayalım. Liste isminde bir metod oluşturup bu metodun içine sorgu değerleri olarak duyurular tablomuzda ki verileri listelemek için "Select * From TblDuyurular" ifadesini ekledik. Daha sonra SqlCommand sınıfımızdan türettiğimiz komut nesnemizi SqlDataAdapter sınıfı içine alıp dt isminde bir veri tablosu oluşturduk.

Tablomuzun içine doldurup DataGridView aracımızın veri kaynağı kısımına dt adlı nesnemizi atayarak kod bloğumuzu tamamladık ve metodumuzu formumuz yüklenince çağırıldı.

Uygulama 20

```
private void BtnEkle_Click(object sender, EventArgs e)
{
    SqlCommand komut = new SqlCommand("insert into TblDuyurular
(icerik) values (@p1)", bgl.baglanti());
    komut.Parameters.AddWithValue("@p1", RichDuyuru.Text);
    komut.ExecuteNonQuery();
    MessageBox.Show("Duyuru Oluşturuldu", "Bilgi", MessageBoxButtons.OK,
    MessageBoxIcon.Information);
    bgl.baglanti().Close();
    Listele();
}
```

```
private void BtnEkle_Click(object sender, EventArgs e)
{
    int secilen = dataGridView1.SelectedCells[0].RowIndex;
    id = dataGridView1.Rows[secilen].Cells[0].Value.ToString();
    RichDuyuru.Text = dataGridView1.Rows[secilen].Cells[1].Value.
    ToString();
    this.Text = id;
}
```



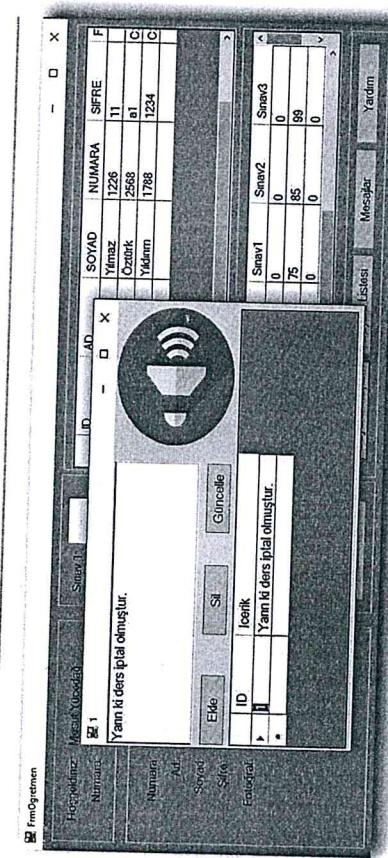
Ekle butonumuza tıklayıp SqlCommand sınıfımızdan komut isminden bir nesne tıktıktı ve bu nesnemize sorgu değeri olarak Insert ataması yaptı. Daha sonra sorgu içindeki tek parametre ifadesi olan parametre 1'in değerinin RichTextBox aracımızdan aldık. ExecuteronQuery ifadesiyle sorguyu çalıştırıp kaydetme işlevini gerçekleştirdik ve bir mesaj kutusuyla kaydetme işeminin gerçekleştirildiğini dair uyan mesajı verdik. DataGridView'de tıkladığımız hücredeki değeri tekrar RichTextBox aracına gönderelim.

Uygulama 21

Datagridview'de seçilen hücredeki duyuru içeriğini RichTextBox aracına aktaran kodu yazalım.

```
string id;
```

```
private void dataGridView1_CellClick(object sender,
EventArgs e)
{
    int secilen = dataGridView1.SelectedCells[0].RowIndex;
    id = dataGridView1.Rows[secilen].Cells[0].Value.ToString();
    RichDuyuru.Text = dataGridView1.Rows[secilen].Cells[1].Value.
    ToString();
    this.Text = id;
}
```



Global alanda id isminde string türünde bir değişken oluşturduk. Bu değişken duyurumuzun ID değerini tutacak. Böylece silme ve güncelleme işlemlerini ID değerine göre kolaylıkla yapabileceğiz. Daha sonra seçtiğimiz satırın indeks değerini seçilen değişkenine atadık. Bu indeks değerine göre tıkladığımız hücredeki duyuru içeriğini RichTextBox aracımıza kolaylıkla yazdırıldı. Ayrıca formun sol üst köşesine de seçtiğimiz duyurunun ID değerini görüntülemek için id değerini yazdırıldı.

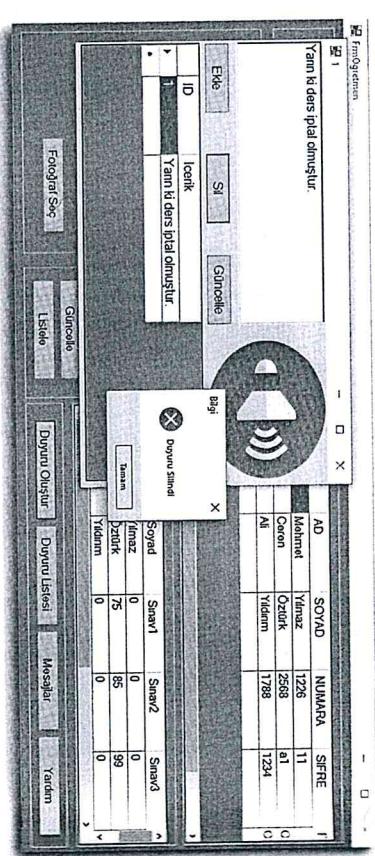
Uygulama 22

Seçilen duyuruyu silen kodu yazalım.

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

```
{
    SqlCommand komut = new SqlCommand("Delete From TblDuyurular where
ID=@p1", bgl.baglanti());
komut.Parameters.AddWithValue("@p1", id);
komut.ExecuteNonQuery();

MessageBox.Show("Duyuru Silindi", "Bilgi", MessageBoxButtons.OK,
MessageBoxIcon.Stop);
bgl.baglanti().Close();
Listele();
}
```

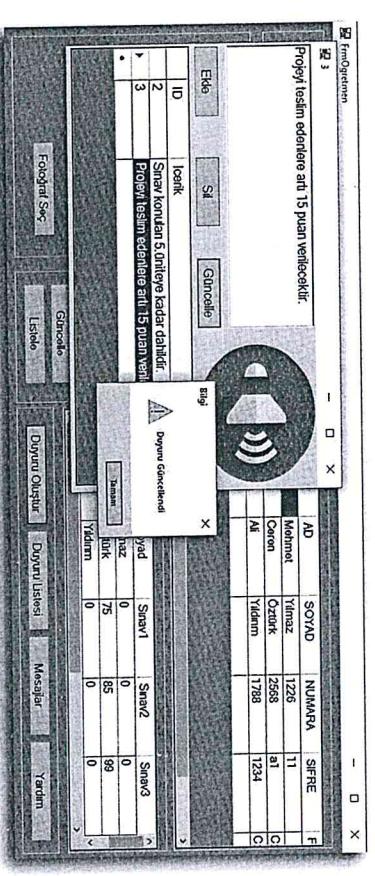
**Uygulama 23**

Seçilen duyuruyu güncelleleyen kodu yazalım.

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

```
{
    SqlCommand komut = new SqlCommand("Update TblDuyurular set icerik=@
p1 where ID=@p2", bgl.baglanti());
komut.Parameters.AddWithValue("@p1", RchDuyuru.Text);
komut.Parameters.AddWithValue("@p2", id);
komut.ExecuteNonQuery();

MessageBox.Show("Duyuru Güncellendi", "Bilgi", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
bgl.baglanti().Close();
Listele();
}
```



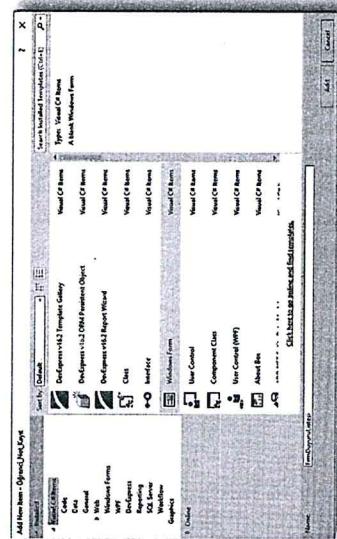
Sil butonumuza çift tıklayıp kodlarımı yazmaya başladık. Oluşturduğumuz komut nesnesine delete görevini vererek ID değerini parametre'ye eşit olan duyuruyu sil dedik. Parametre değerimizi de global alanda tanımladığımız id adlı değişkenden aldık. Daha sonra işlemi tamamlayıp mesaj verip son olarak duyuruları listeleyerek kodlarımızı tamamladık. Güncelleme işlemi yapabilmek için bir iki tane duyuru eklemesi yapalım çünkü veritabanımızda duyurumuz kalmadı.

Güncelle butonuna çift tıklayıp kodlarımı yazmaya başladık. İlk olarak SqlCommand sınıfından komut isminde bir nesne türettik. Daha sonra bu nesnemizin alacağı SQL sorgusu olan update sorgusunu yazıp sırasıyla id ve duyuru içeriği parametrelerimizi tanımladık. ExecuteNonQuery ifadesini kullanıp hemen akabinde bir mesaj vererek sorgumuzu güncellendirdiğini öğretmene bildirmiş olduk.

Son olarak bağlantımızı kapatıp verileri listeleyerek kodbloğumuzu tamamladık. Böylece duyuru işlemleri formumuzu bitirdik. Duyuru listesi formu öğrenciler tarafından gözükecek ve sadece okunur bir form olacak. O formda amacımız dinamik araç kullanmak olduğundan biraz daha farklı bir yol izleyeceğiz.

DUYURU LİSTESİ FORMU

Böyle bir formu kullanacak olmamızın en büyük sebebi dinamik olarak oluşturacağımız bir araca veritabanından atama yapabilmektir. İçerisine herhangi bir araç eklemeden duyuru listesi altında bir form oluşturallım.

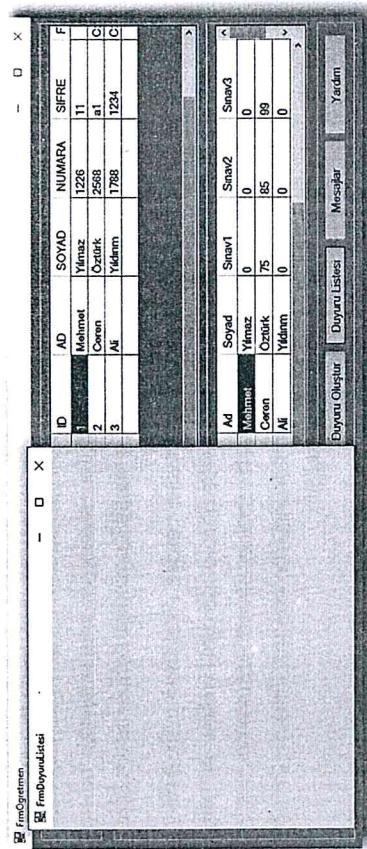


Duyurulisteformumuza oluşturduk. Formumuza araç eklemeyip yapmayaçğız. Yalnızca dinamik bir tane ListBox aracı oluşturup bu aracımızın içine veritabanından duyularımızı çekeceğiz. Dilerseniz araç kutusundan ListBox veya benzeri bir araç ekleyerek daha pratik bir yöntemle ilgili işlemi yapabilirsiniz. Burada yazar olarak benim amacım sizlere olabidiğince farklı başlıklar gösterebilmek. Öyleyse ilk önce dinamik araç oluşturma sonra da bu aracın içine duyuruları çekme işlemini yapalım. Fakat hepsinden önce öğretmem formundan bu forma geçiş sağlayalım.

Üygulama 24:

Öğretmen formunda duyuru listesi butonuna tıkladığımız duyuru listesi formunu açan kodu yazalım.

```
private void BtnDuyuruliste_Click(object sender, EventArgs e)
{
    FrmDuyuruliste frm = new FrmDuyuruliste();
    frm.Show();
}
```

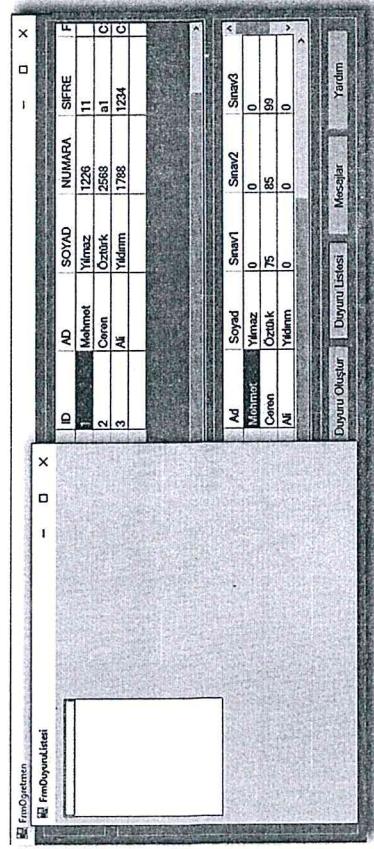


Duyuru listesi formumuzdan frm isminde bir nesne türettik ve frm nesnemizin show özelliğini kullanarak diğer forma geçiş işlemini tamamladık.

Üygulama 25:

Duyuru listesi formu açıldığı zaman formda dinamik bir ListBox aracı oluşturan kodu yazalım.

```
private void FrmDuyuruliste_Load(object sender, EventArgs e)
{
    ListBox lst = new ListBox();
    Point lstKonum = new Point(10, 10);
    lst.Name = "Listbox1";
    lst.Location = lstKonum;
    lst.Width = 200;
    lst.Height = 150;
    lst.Controls.Add(lst);
}
```

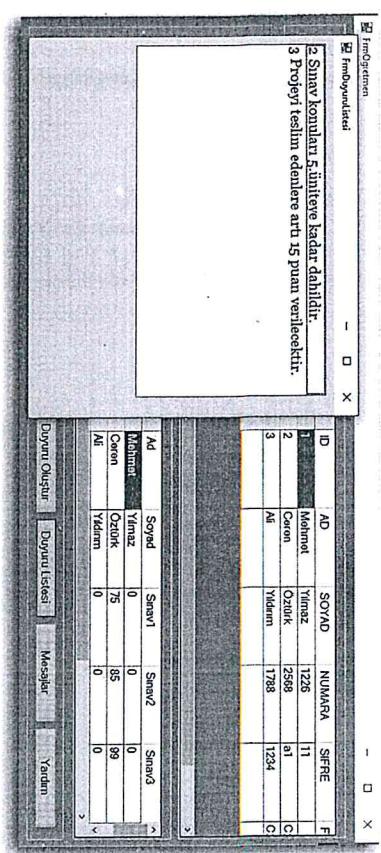


Duyuru listesi formu yüklediğim zaman ListBox aracı oluşturmak istediğim için kodlarımı form load içine yazdım. ListBox sınıfından 1st isminde bir nesne türrettik. Point sınıfından da 1stkonum isminde bir nesne türetip bu nesnemize X ve Y koordinatlarında kaçıp piksel boşluk bırakarak konumlanmasını belirttik.

Daha sonra ListBox aracımıza bir isim verdik. Akabinde yükseklik ve genişlik değerlerini belirledik. Üstteki resimde görüldüğü gibi ListBox aracımız epes kükük kaldı genişliğini biraz daha artturabiliyoruz. Sonra aracımızın lokasyon değerini olarak 1stkonum nesnemizle ilişkilendirip formda gözükmesi için formun içine ekleyerek kod bloğumuzu tamamladık.

Uygulama 26

Duyuruları dinamik olarak oluşturduğumuz ListBox aracımızda gösterelim.



En üst kısma kütüphanemizi ekleyerek kodlarımızyazmaya başladık. Daha sonra SQL bağlantı sınıfımızı çağırıldı. Bir önceki sayfada yazmış olduğumuz daimik araç oluşturma kodunda bir iki değişikliğe gittik. ListBox aracımızın genişliğini biraz daha arttırdık ve yazı stilini de new font özelliği ile özelleştirdik. New fon komutundan sonra 3 tane parametre kullandık.

Bunların birincisi yazının font türü, ikincisi boyutu ve üçüncüsü de ekstra özel leşitirme durumu yani kalın, italik vs. daha sonra duyuruları listelemek için bir Select sorgusu yazdık. Yazdığımız sorgu ile hem duyuru id değerlerini hem de içerikleri çekebiliyoruz.

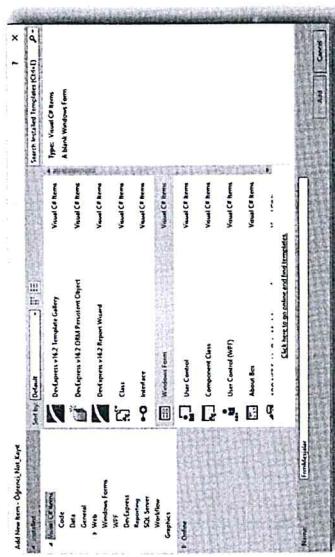
İkisini de aralarında bir karakterlik boşluk bırakarak ListBox aracımızda gösterdik. ListBox aracımıza bu ifadeleri yazdırma için ListBox aracımız için türetilmiş 1st.Font = new Font("Georgia", 14, FontStyle.Regular); 1st.Location = 1stkonum; this.Controls.Add(1st);

//Duyuruları Listeleme Kodları

```
SqlCommand komut = new SqlCommand("Select * From Tblduyurular",
    bgl.baglanti());
SqlDataReader dr = komut.ExecuteReader();
while (dr.Read())
{
    1st.Items.Add(dr[0] + " " + dr[1]);
}
bgl.baglanti().Close();
```

MESAJLAR FORMU

Mesajlar formumuz hem öğrenciler hem de öğretmen için ortak olacak. Yalnızca geçişlerde biraz daha farklı bir yöntem uygulayacağız. Mesajlar formumuz gelen mesajlar ve giden mesajlar şeklinde ikiye ayrılacak. Ayrıca mesaj gönderme formunu da yine aynı form içinde oluşturabiliriz. Öncelikle yeni bir form ekleyelim.



Formumuza ekledik. Şimdi bu formda olması gerekenleri sıralayacak olursak;

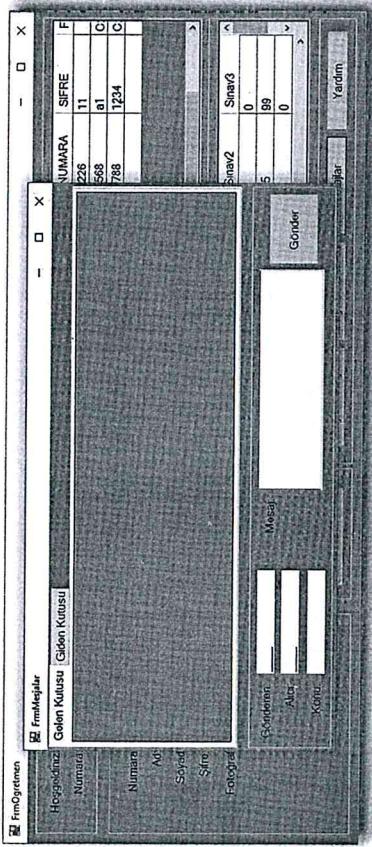
- » Mesaj gönderme alanı
- » Gelen kutusu
- » Giden kutusu
- » Mesaja tıklandığı zaman mesaj detaylarını görebilme

Bu 4 başlığına göre işlemlerimizi gerçekleştiriyoruz. İlk önce form tasarımını yapalım.

Formumuzun üst kısmına bir tane Tabcontrol aracı ekledik. Bu araç sekme kontrol anlamındadır ve tasarımda sekmelere ayırmamız gereken durumlarda kullanılmaktadır. Tabcontrol aracımızın her bir sekmesinin ismini değiştirmek için önce ilgili sekmeye tıkıyoruz sonra ilgili sekmenin bulunduğu alanın içine tıkladığınızınız özelliginiz pencerelerinde Tab Page yanı sekme sayfasının özellilikleri gelecektir. Buradan isim değiştirme gerçekleştirebilir. Sekmelerimizin birisi gelen dğeri de giden kutusunu listeleyecektir. Alt kısmı bir tane Groupbox aracı ekledik. Groupbox eklediğimiz alanı mesaj gönderme paneli şeklinde düzenledik. Mesaj gönderme işlemi numaraya göre yapılacak için gönderici ve alıcı değerlerini numara olarak belirledik. Daha sonra mesaj başlığını, içeriğini ve gönderme butonunu ekledik. Şimdi Datagridview aracımızı da ekleyelim

Gelen ve giden kutusu alanlarına birer tane Datagridview aracı ekledik. Bu alanlarda mesajlarımızı listeleyeceğiz. Form tasarımımızı bitirdiğimize göre artık kodlama kısmına geçebiliriz. İlk önce Öğretmen formunda bu forma geçiş işlemi yapalım.

```
private void BtnMesajlar_Click(object sender, EventArgs e)
{
    FrmMesajlar frm = new FrmMesajlar();
    frm.Show();
}
```

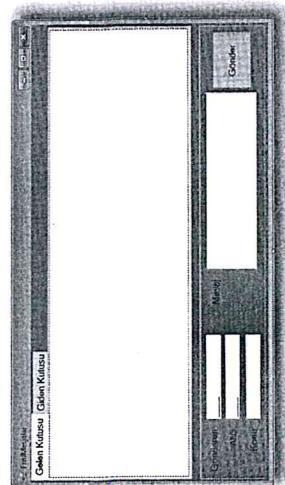


Öğretmen formundan mesajlar formuna geçiş yapan kodu yazalım.

Mesajlar butonumuza tıkladık ve formumuzu açtık. Yalnız küçük bir değişiklik gidelim. İlk etapta rastgele olarak belirlediğimiz öğrencimen numarası olan 1562 değerini değiştirelim. Çünkü bu değer sıradan bir öğrenci numarası gibi kullanılabilir. Onun yerine 0000 seçeneğini ekleyelim. Böylece bu kişinin bir öğretmen ya da sistem yetkilisi olduğu biraž daha belli olacaktır.



SQL'de bulunan öğrencinin tablomuzdaki numarayı 0000 olarak değiştirdik. İşlemlerimize devam edelim. İlk olarak gelen kutusu ve giden kutusu listelememiz gerekecek. Ancak öncesinde yapmamız gereken bir işlem daha var. Formlar arası geçişte sisteme kimin giriş yaptığınumara değiştikenin diğer forma taşımak. Bunu beraber uygulayalım.



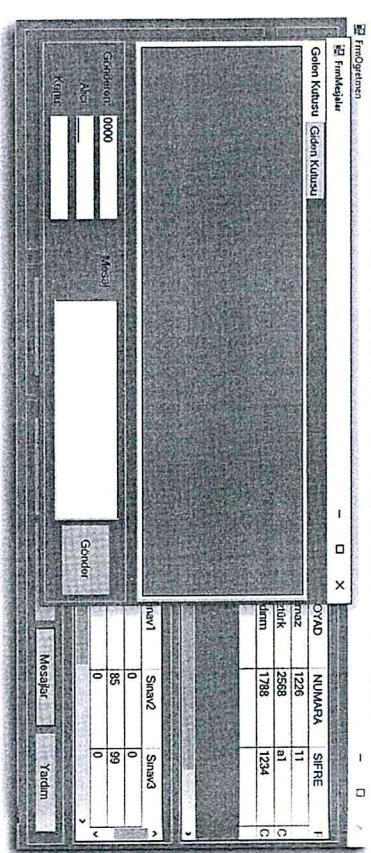
Üyelikler 28:

Mesajlar formuna giriş yapan öğretmenin numarasını öğretmen formundan mesajlar formuna taşıyalım.

```
//Öğretmen Formu

private void BtnMesajlar_Click(object sender, EventArgs e)
{
    FrmMesajlar frm = new FrmMesajlar();
    frm.numara = LblNumara.Text;
    frm.Show();
}

//Mesajlar Formu
```

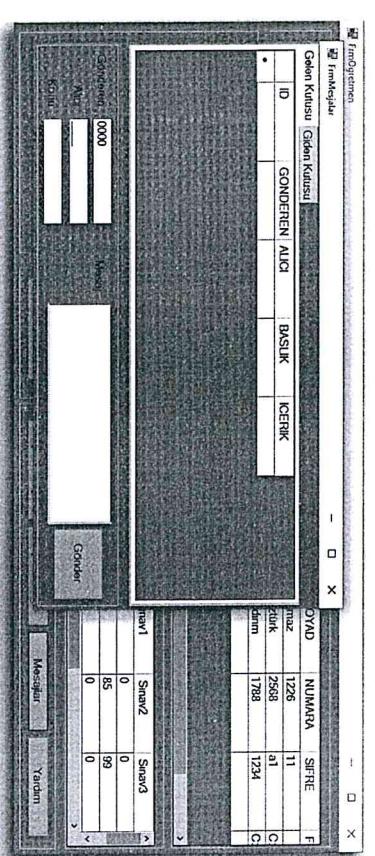
**Üyelikler 29:**

Mesajlar formunda öğretmene girmiş olan mesajları listeleyelim.

```
void GelenMesajlar()
{
    SqlCommand komut = new SqlCommand("Select * From TblMesajlar Where
    ALICI=@p1", bgl.baglanti());
    komut.Parameters.AddWithValue("@p1", numara);
    SqlDataAdapter da = new SqlDataAdapter(komut);
    DataTable dt = new DataTable();
    da.Fill(dt);
    dataGridView1.DataSource = dt;
}
```

```
private void FrmMesajlar_Load(object sender, EventArgs e)
{
    MskGonderen.Text = numara;
}
```

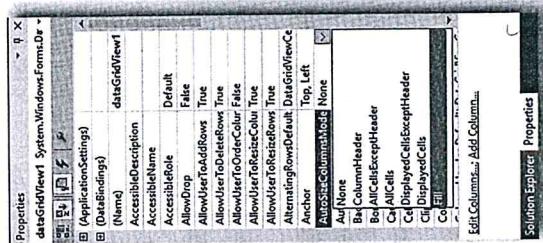
```
GelenMesajlar();
```



Önce öğretmen formuna gelip mesajlar formunun açılma kısmında mesajlar formunda tanımlamış olduğumuz numara değişkenine değer atamayı yaptık. Bu değişkenimiz değerini sisteme giriş yapan öğretmenin numarasından alacaktır. Daha sonra mesajlar formunda global alanda public static sınıfımızda bir değişken oluşturduk ve bu değişkenimizi öğretmen formundan değer aldıktan sonra tekrar mesajlar formuna dönünce aldığı değer göstermesi için MskGonderen MaskedTextBox aracımıza yazdırıldık.

En üst kısma SQL kütüphanemizi ekleyerek kodlarımızı yazmaya başladık. Gelen mesajları bir metod olarak tanımlayıp metodumuzda SqlCommand sınıfından komut isminde bir nesne türettik. komut nesnemizin sorgusu için mesajlar tablosunda alıcı değeri parametre'ye eşit olan değerleri getirmesi istedik. parametre1 değerini olarak diğer formdan taşdıgımız numara değerini verdik. Bu numara değeri öğrenciler için de aynı yöntemle alınacaktır yani bundan dolayı öğrenciler için yeni bir değer ataması yapmak zorunda değiliz. Daha sonra parametremizi tanımlayıp sanal tablomuzu oluşturup bunun içini doldurup verileri DataGridView aracımızda gösterdik ve metodumuzu formda çağrıdık.

Datagridview aracımızda hiç gri alan kalmaması ve tüm boşluklara eşit şekilde hücre verilerinin dolması için özellikler penceresinde her iki Datagridview aracımızın da AutoSizeColumnsMode özelliğini Fill yani doldur seçeneği olarak belirliyoruz.



Üyelik 30

Mesajlar formunda öğretmenin göndermiş olduğu mesajları listeleyelim.

void GidenMesajlar()

```

{
    SqlCommand komut = new SqlCommand("insert into TblMesajlar
(Gonderen,Alıcı,Baslı,Icerik) values (@p1,@p2,@p3,@p4)", bgl.baglanti());
    komut.Parameters.AddWithValue("@p1", MskGonderen.Text);
    komut.Parameters.AddWithValue("@p2", MskAlıcı.Text);
    komut.Parameters.AddWithValue("@p3", TxtKonu.Text);
    komut.Parameters.AddWithValue("@p4", RchMesaj.Text);
    komut.ExecuteNonQuery();
}

MessageBox.Show("Mesajınız İletildi...", "Bilgi", MessageBoxButtons.OK, MessageBoxIcon.Information);
bgl.baglanti().Close();

```

```

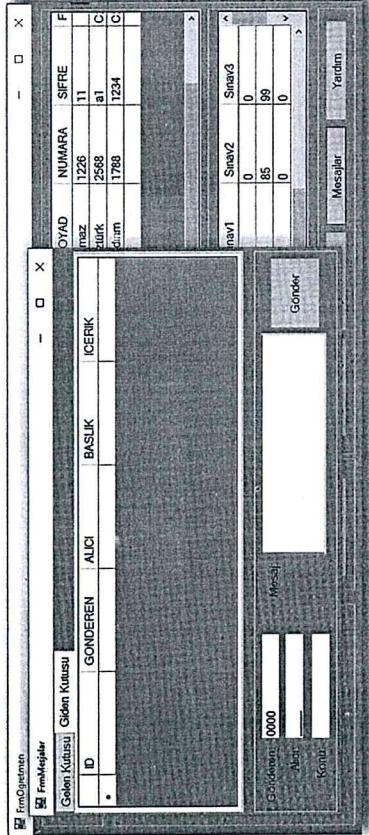
private void FrmMesajlar_Load(object sender, EventArgs e)
{
    MskGonderen.Text = numara;
}

```

```

GelenMesajlar();
GidenMesajlar();
}

```



Gelen mesajlar metodundaki yöntemin ayını giden yanı gönderilmiş olan mesajları için de izledik. Bu kez tablo sütunumuzda şartımızı parametre değerimizinden değerini olmasından. Yani sisteme giriş yapan numaranın göndermiş olduğunu mesajları listeledik. Listeleme işlemi için 2. datagridview aracımızı kullandık. Son olarak metodumuza formda çağrıarak kod bloğumuzu tamamladık.

Üyelik 31

Yeni mesaj gönderme işlemini gerçekleştirelim.

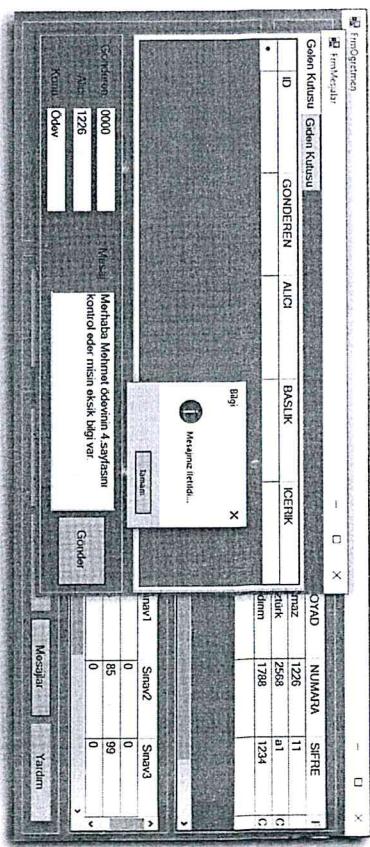
private void BtnGonder_Click(object sender, EventArgs e)

```

{
    SqlCommand komut = new SqlCommand("insert into TblMesajlar
(Gonderen,Alıcı,Baslı,Icerik) values (@p1,@p2,@p3,@p4)", bgl.baglanti());
    komut.Parameters.AddWithValue("@p1", MskGonderen.Text);
    komut.Parameters.AddWithValue("@p2", MskAlıcı.Text);
    komut.Parameters.AddWithValue("@p3", TxtKonu.Text);
    komut.Parameters.AddWithValue("@p4", RchMesaj.Text);
    komut.ExecuteNonQuery();

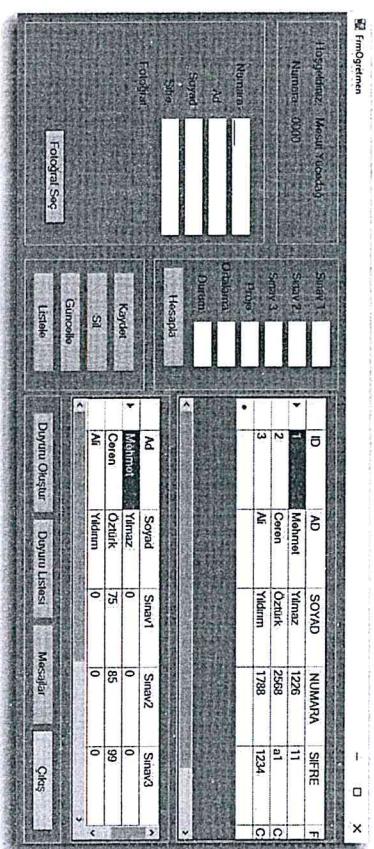
    MessageBox.Show("Mesajınız İletildi...", "Bilgi", MessageBoxButtons.OK, MessageBoxIcon.Information);
    bgl.baglanti().Close();
    GelenMesajlar();
    GidenMesajlar();
}

```



```
private void BtnKisik_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

Üygulama 32
Yardım butonunun ismini çıkış olarak değiştirdip, tıkladığı zaman sistemi kapatıp kodları yazalım.



Ödev

Gönder buttonumuza çift tıkladık ve kodlarınıza yazmaya başladık. İlk olarak SqlCommand sınıfımızdan komut isminde bir nesne türettiğimiz. Daha sonra bu nesnemize Insert sorgusunu gönderdik. Insert sorgusu aracılığı ile mesajlar tablomuza eklemeye yaparak mesaj gönderimi sağlayacağımız. Gönderen, alıcı, başlık ve mesaj içerikleri için parametrelerimizi tanımlayıp değer atamalarını gerçekleştirdik. Sonra mesaj kutusunda mesajımızı illetildiğine dair bir mesaj gösterdik. Bağlantımızı kapatıp gelen ve giden kutusunu mesajları listeleyen metodlarınıza çağrıarak kod bloğumuzu sonlandırdık.

Mesajların listelediği DataGridView araçlarına çift tıklandığı zaman ilgili mesajın detaylarını ve numarası yerine ad soyad bilgisi listeleyen kodları yazın. (Birleştirme kullanılsın)

Öğretmen formunun son başlığı olan yardım menüsünde de küçük bir değişiklik yapıyoruz. Dilerseniz burası yardım menüsü olarak kalabilir ve siz küçük bir pencerede öğretmen formunun nasıl kullanılabileceğini dair bir açıklama metni yazıp yönlendirmeleri verebilirisiniz. Ben burayı çıkış olarak değiştirmeye karar verdim. Böylece öğretmen işini tamamlayınca sisteme buradan direkt olarak kapabileceksiniz.

ÖĞRENCİ FORMU

Üstte bulunan görseldeki gibi yardım butonunun değerini çıkış olarak değiştirdik. Butonumuza çift tıklayıp uygulamayı sonlandı anlamına gelen Application Exit komutunu yazarak kod bloğumuzu tamamladık. Böylece öğretmen formu tamamen bitmiş oldu. Sizlerin bu formu daha çok geliştirebilir daha optimize bir hale getireceğinden şüphem yok. Artık öğrenci formuna geçip öğrenci ile ilgili işlemleri yapabiliyoruz.

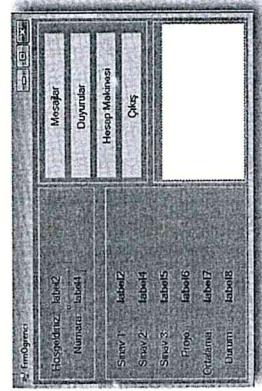
ÖĞRENCİ FORMU

Öğretmen formunu tamamen bitirdik. Şimdi öğrenci formuna geçiş yapacağız. Projemiz ilk kısımlarda öğrenci formunu oluşturmuş ancak tasarımını yapmadıktır. Önce öğrenci formumuzu tasarlayalım. Öğrenci formunda izleyeceğimiz adımlar şöyle olacak;

- » Form tasarımlı
- » Numaranın taşınması
- » Numaraya göre isim bilgisi getirme
- » Numaraya göre sınav bilgisi getirme
- » Çıkış
- » Duyurular formu
- » Mesajlar formu
- » Hesap makinesi

ÖĞRENCİ FORMU TASARIMI

Form tasarımda aşağıdaki gibi bir tasarım uyguladık. Alanlarımızı görselimiz üzerinde anlatalım.



Sol üst tarafta bulunan **Hosgeldiniz** kısmının karşısına sisteme giriş yapan kişinin ad ve soyad bilgisini yazdıracağız. Hemen altına da giriş formundaki numara bilgisini taşıyacağız. Sol alt kısımda bulunan Groupbox aracı içində sisteme giriş yapan öğrencinin sınav notları yer alacak. Sağ üst tarafta öğrencinin erişim sağlayabileceğii mesajlar, duyurular menüsü ile hesap makinesi ve çıkış alanları yer alacak. Sağ alt tarafta da sisteme giriş yapan öğrencinin fotoğralını Picturebox aracı içinde göstereceğiz.

ÖĞRENCİ FORMUNA GİRİŞ VE NUMARANIN TAŞINMASI

Giriş formunda işlememiz bu kez öğrenciye göre gerçekleştireceğiz. Eğeli numara ve şifre değeri doğruysa bu numara ve şifre değerini öğrenci formunu taşıyarak buradan da numaraya göre işlemlerimiz devam edeceğiz. Bu işlemi zaten daha önce yaptığımiz için yalnızca numara taşıma işlemini gerçekleştireceğiz.

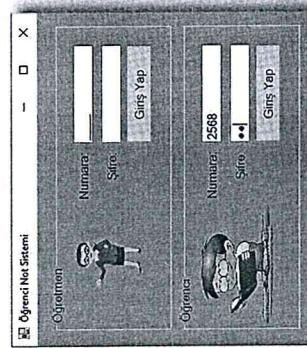
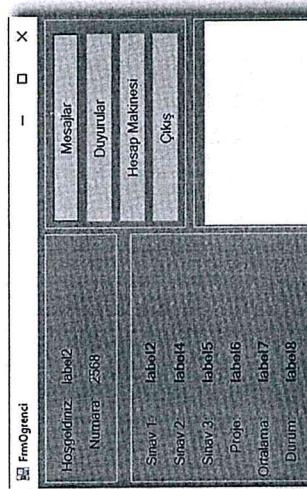
Üyelama 33
Giriş formunda numara ve şifresini giren kişiyi öğrenci formuna yönlendiren kodu yazalım.

```
// Giriş Formu
private void Btn0grenciGiris_Click(object sender, EventArgs e)
{
    SqlCommand komut = new SqlCommand("Select * From Tbl0grenci where
Numara=@p1 and Şifre=@p2", bg1.bağılanti());
Parameters.AddWithValue("@p1", Msk0grenciNumara.Text);
komut.Parameters.AddWithValue("@p2", Txt0gencisifre.Text);
SqlDataReader dr = komut.ExecuteReader();
```

ÖĞRENCİ FORMU KODLAMA

```
if (dr.Read())
{
    Frm0grenci frm = new Frm0grenci();
    frm.numara = Msk0grenciNumara.Text;
    frm.Show();
    this.Hide();
}
else
{
    MessageBox.Show("Hatalı Numara ya da Şifre", "Uyarı!",
MessageBoxButtons.OK, MessageBoxIcon.Stop);
}
bg1.bağılanti().Close();
}

//Öğrenci Formu
public string numara;
private void Frm0grenci_Load(object sender, EventArgs e)
{
    LblNumara.Text = numara;
}
```



Kodlarımıza daha önce yazmıştık. Kod bloğumuzda bir iki değişiklik yaptıktır. Mesaj kutusunu sadecə hatalı durumlarda vermemesini istedik. Sisteme giriş doğru bir şekilde yapılıyorsa mesaj vermesine gerek yok. Sisteme giriş yapmadan önce öğrenci formunda global ve erişimi herkese açık bir değişken tanımladık. Bu değişkenimizi LblNumara aracına yazdırıldı. Daha sonra giriş formunda frm nesnemizin show özelliğinden önce numara değişiksenine atama işlemimi gerçekleştirdik. Daha sonra show metoduya öğrenci formunu açıp işlemlerimizi tamamladık. Kod bloğumuzu önceki sayfalarda açıkladığımız için yeniden açıklamaya ihtiyaç duymadık.

NUMARAYA GÖRE İSİM BİLGİSİNİN GETİRİLMESİ

Sisteme giriş yapan öğrencinin numarasına göre yani global alanda tanımladığımız numara değişkenin aldığı değere göre ad, soyad ve ilgili öğrencinin fotoğrafını listeleyelim. Tipki öğretmen formunda öğretmenin ad ve soyad bilgisini getirdiğimiz gibi burada da öğrencinin kişisel bilgilerini görüntüleyeceğiz.

Uygulama 34

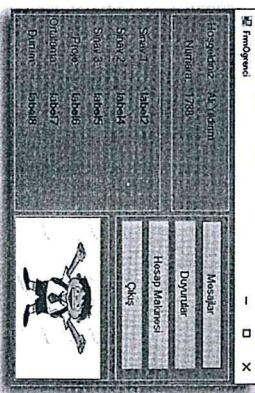
Öğrenci formuna giriş yapan öğrencinin numarasına göre ad, soyad ve fotoğrafını getiren kod yazalım.

```
SqlBaglantisi bgl = new SqlBaglantisi();
private void FrmOgrenci_Load(object sender, EventArgs e)
{
    LblNumara.Text = numara;

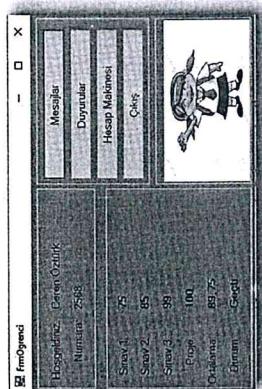
    //Numaraya göre isim bilgisi getirme
    SqlCommand komut = new SqlCommand("Select * From TblOgrenci where
    Numara=@p1", bgl.baglanti());
    komut.Parameters.AddWithValue("@p1", numara);
    SqlDataReader dr = komut.ExecuteReader();
    while (dr.Read())
    {
        LblAdSoyad.Text = dr[1] + " " + dr[2];
        pictureBox1.ImageLocation = dr[5].ToString();
        bgl.baglanti().Close();
    }

    //Not Listesi
    SqlCommand komut2 = new SqlCommand("Select ID From TblNotlar where
    0gr1d=(Select ID From TblOgrenci Where Numara=@p1)", bgl.baglanti());
    komut2.Parameters.AddWithValue("@p1", LblNumara.Text);
    SqlDataReader dr2 = komut2.ExecuteReader();
    while (dr2.Read())
    {
        LblSınav1.Text = dr2[1].ToString();
        LblSınav2.Text = dr2[2].ToString();
        LblSınav3.Text = dr2[3].ToString();
        LblProje.Text = dr2[4].ToString();
        LblOrtalama.Text = dr2[5].ToString();
    }
    bgl.baglanti().Close();

    if (Convert.ToDouble(LblOrtalama.Text) >= 50)
    {
        LblDurum.Text = "Geçti";
    }
    else
    {
        LblDurum.Text = "Kaldı";
    }
}
```



Global alanda ilk olarak SQL Bağlantısı sınıfımızdan bir nesne türettiğimiz. Form Load bloğu içinde SqlCommand sınıfımızdan komut adında bir nesne türeterek bu nesnemize sorgu değerini olarak Select * From TblOgrenci ifadesini yazdık.



Üyelik 36

Öğrenci formundan mesajlar formuna geçiş yapalım.

```
//Mesajlar formu

public string numara;

private void FrmMesajlar_Load(object sender, EventArgs e)
{
    MskGonderen.Text = numara;
    GelenMesajlar();
    GidenMesajlar();
}

//öğrenci formu
private void BtnMesajlar_Click(object sender, EventArgs e)
{
    FrmMesajlar frm = new FrmMesajlar();
    frm.numara = LblNumara.Text;
    frm.Show();
}
```

SqlCommand sınıfından komut2 isminden bir nesne türettiğimiz komut nesnesini daha önce öğrenci bilgilerini çektiğimiz alanda kullandığımız için türettiğimiz nesne adını farklı bir değer olarak verdik.

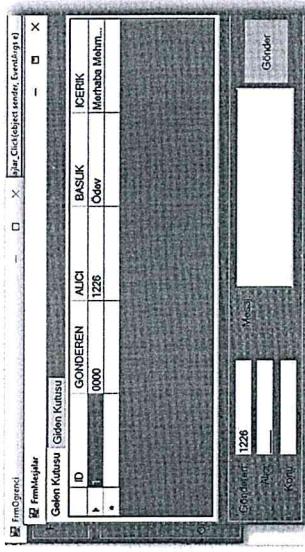
Öğretmen bölümünden öğrenciye ait not bilgisini getirmek için kullandığımız alt sorguyu burada yeniden kullandık. Notlar ile öğrenciler tablosu arasında ilişkili bulunduğundan ve ilişkili alanımız ID değerine göre olduğundan dolayı bizde alt sorguda anahtar sütunu ID alanı olarak belirledik. Sisteme giriş yapan öğrenci numarasına göre notlarını listeleyip değerlerimizi Label araçlarımıza yazdırıldı. Son olarak eğer ortalamama 50'den büyükse veya eşitse durum Label aracımıza geçti, aksi durumda kalıcı yazdırarak kodlarımızı tamamladık. Öğrencimizin kişisel bilgilerinin ve notlarının listelenmesi işlemi tamamlandı.

Öğrenci

Öğrencinin şifresini değiştirebilmesi için gerekli olan form tasarnı ve kodlamayı yapın.

MESAJLAR FORMU

Mesaj formumuzu öğretmen formunu oluştururken tamamlamıştık. Burada öğrenci için mesaj formu hazırlarken yeniden bir form tasarnı yapmaya ya da yeniden mesaj formunu kodlamaya ihtiyacımız yoktur. Sadece ilgili öğrencinin numara değerini mesaj formuna taşımadım gerekecek. Böylece ilgili öğrenciye gönderilen ya da öğrencimizin gönderdiği mesajları görebileceğiz.



Üst kısımda yazan mesajlar formu kod bloğunu yeniden yazmadık. Bu zaten mesajlar formunda var olan kodlarımızı sadece burada da hatırlamak adına yeniden gösterdik. Öğrenci formundaki mesajlar butonumuza çift tıklayıp kodlarımızı yazmaya başladık. Mesajlar formundan form isminde bir nesne türrettik. Türettiğimiz bu nesne aracılığı ile mesajlar formunda bulunan numara değişkenine erişim sağladık. Daha sonra bu değişkenimize LblNumara üzerinde bulunan değerini atadık. Son olarak show metodunu kullanarak mesajlar formumuzu açtık. Böylece ilgili öğrenci numara bilgisi ile mesajlar formuna taşınmış oldu.

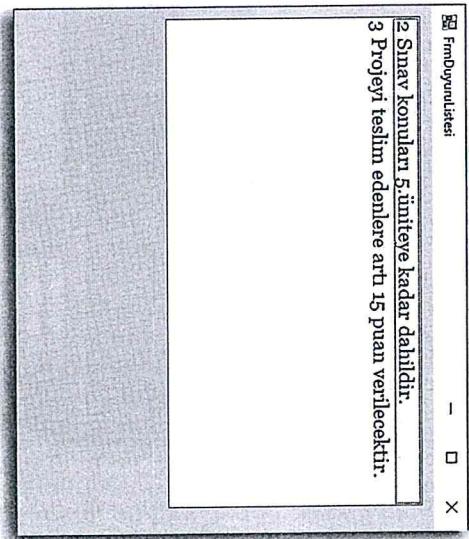
DUYURULAR FORMU

Öğrencim tarafından oluşturulan duyuruları listeleyeceğimiz form olacak. Duyurular için daha önce dinamik bir ListBox aracı üzerinden listelediğimiz bir form olduğu için yeni bir form oluşturmaya gerek duymuyoruz. Öyleyse öğrenci formundan duyurulara geçiş yapalım.

Üygulama 37

Öğrenci formundan duyurular formuna geçiş yapan kodu yazalım.

```
private void BtnDuyurular_Click(object sender, EventArgs e)
{
    FrmDuyuruliste frm = new FrmDuyuruliste();
    frm.Show();
}
```



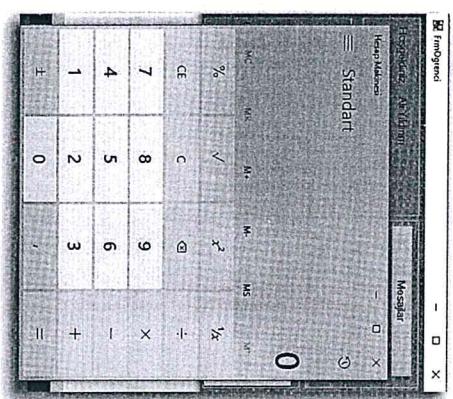
Duyurular formundan frm isminde bir nesne türettiğim. Daha sonra show metodu aracılığı ile duyuru listesi formumuzu görüntüleyerek kod bloğumuzu tamamladık.

HESAP MAKİNESİ

Öğrenci kaç puan alırsam sınavı geçerim şeklinde hesaplama yapmak istediği durumlarda başlat menüsünden hesap makinesini açmak yerine çok daha pratik bir yöntem kullanarak hesap makinesine erişim sağlamak için tek bir Button tıklaması ile gerçekleştireceğimiz komut alanıdır.

Üygulama 38
Öğrenci formunda bulunan hesap makinesi butonuna tıkladığımız zaman hesap makinesini çalıştırın kodu yazalım.

```
private void BtnHesapMakinesi_Click(object sender, EventArgs e)
{
    System.Diagnostics.Process.Start("Calc.Exe");
}
```



Butonumuza çift tıklayıp kodlarımıza yazmaya başladık. cmd ile gelen çalıştır alandaki tüm uzantıları System Diagnostics Process Start komutu ile çalıştırabiliriz.

System; sistem anlamına gelmektedir.

Diagnostics; tanılamak anlamını taşyor.

Process; start ise başlat anlamındadır.

Calc.Exe komutu hesap makinesini çalıştırmak için gerekli olan uzantıdır. Yani toparlarsak kod bloğumuz, sisteme tanımlı olduğumuz şu işlemi çalıştır, anlamına gelmektedir.

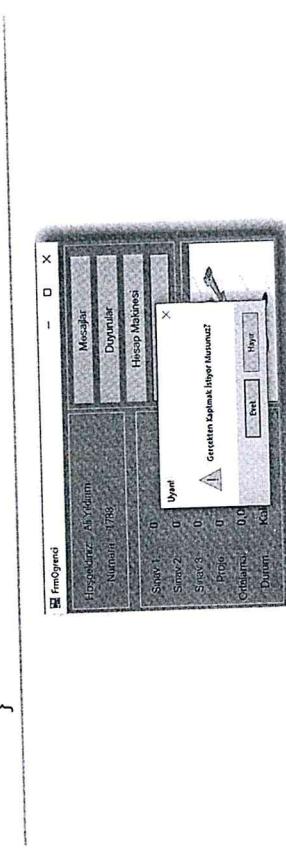
ÇIKIŞ

İşlemimizi bitirdikten sonra formu kapatmak için çıkış seçenekine tıklayarak projemizi kapatalım.

Üyelik 39

Öğrenci formunda çıkış seçenekine tıklayınca formu kapatıp kodu yazalım.

```
private void BtnCikis_Click(object sender, EventArgs e)
{
    DialogResult dr = MessageBox.Show("Gerektiken Kapıtmak İstiyor Musunuz?", "Uyarı!", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
    if (dr == DialogResult.Yes)
    {
        Application.Exit();
    }
}
```



Bu kez direkt olarak Application.Exit komutu yerine bir diyalog penceresi aracılığı ile soru sorarak çıkış sağladık. DialogResult sınıfından dr isminde bir nesne oluşturduk ve bu nesnemize bir mesaj kutusu görevini gönderdim. Eğer kişi ile gelecek olan pencerede Yes yani Evet seçeneğine tıklarsa uygulamayı kapatmasını istedik Hayır seçeneğine tıklarsa herhangi bir işlem yapmayacak şekilde diyalog penceresini kapatacaktır. Böylece uygulamamızın sonuna geldik.

NELER ÖĞRENDİK?

Kitabımızın şu ana kadarki kapsamlı bölümü olan projenin bölümünde olabildiğince farklı başlıkların bir araya getirip tek bir proje üzerinde kullanılmaya çalıştık. SQL'de bir veritabanında birden fazla tablo oluşturmayı, tablolar arasında birebir ilişkisi, prosedür oluşturmayı, bir tablodaki kaydın diğer tabloyu etkilemesinden ötürü ilgili tablomuza tetikleyici yazmayı, 4 temel SQL sorgusu olan "Select, Insert, Delete ve Update" komutlarını, birleştirme işlemleri, alt sorguları, sorgu içinde birden fazla sorgu yazmayı, şartlı sorgulamaları, kullanıcı adı ve şifre değerine göre sisteme giriş yaptırmayı, kullanıcı adına göre bilgi getirmeyi, sisteme giriş yapan kişinin fotoğrafını görüntülemeyi, SQL bağlantı sınıf ve metodları oluşturmaya, neden sınıflara ve metodlara bu kadar ihtiyacımız olduğunu ve sıfırdan adım adım ilerleyerek projeye yapmak için gerekli süreci öğrenmiş olduk.

17.2³
17.2⁴