



DÜZCE
ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

MÜHENDİSLİK FAKÜLTESİ

**2023-2024 AKADEMİK YILI
GÜZ DÖNEMİ**

GÖRSEL PROGRAMLAMA

Ders Sorumlusu:

DERNEK TAKİP OTOMASYONU

Hazırlayanlar:

Ad Soyad

Öğrenci Numarası

Öğretim Türü

İçindekiler

1.	GİRİŞ	3
2.	AMAÇ.....	3
3.	TANIM.....	3
4.	GEREKSİNİMLER VE İHTİYAÇLAR.....	3
5.	ANALİZ	5
6.	PROJE.....	6
6.1.	Models Katmanı.....	6
6.2.	Data Access Katmanı	9
6.2.1	Abstract	9
6.2.2	Concrete	11
6.2.3	Context	13
6.3.	Business Katmanı.....	14
6.3.1	Abstract	14
6.3.2	Concrete	16
6.4.	Presentation Katmanı.....	23
6.4.1	Giriş Ekranı.....	24
6.4.2	Kayıt Ekranı.....	25
6.4.3	Ana Sayfa	27
6.4.4	Üye İşlemleri Ekranı.....	28
6.4.5	Aidat İşlemleri Ekranı.....	33
6.4.6	Aidat Düzenleme Ekranı	38
6.4.7	Borçlular Listesi Ekranı.....	40
6.4.8	Duyuru Ekranı	45
6.4.9	Raporlama Ekranı	47
6.4.10	Grafikler.....	51

1. GİRİŞ

Dernek Takip Otomasyonu; üye işlemleri, aidat işlemleri, borcu olanların listelenmesi, duyuru yayınılama ekranı, raporlama ve grafik ekranlarını içermektedir. Dernek yetkilileri gerekli bilgilerle sisteme üye olabilirler. Üye olan yetkililer sisteme giriş yapıp programda bulunan ekranlara erişim sağlayarak bu ekranlar üzerinde ekleme, silme, güncelleme işlemlerini gerçekleştirebilirler.

2. AMAÇ

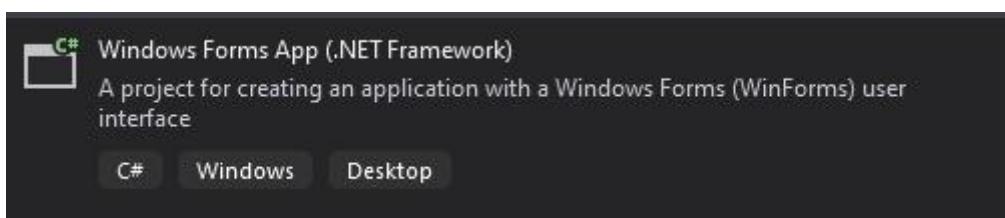
Projenin amacı; dernek yetkililerinin kullanabilmesi için tasarlanmış olup üye ve yetkililerin bilgilerinin tutulduğu kolay ve her türden insanın kullanabileceği otomasyon yapmaktadır. Yetkili, giriş ekranı üzerinden girdiği bilgileri doğru olduğu taktirde sisteme erişerek işlemleri gerçekleştirebilir. Projemiz yetkililerin dernekteki işlemleri daha hızlı gerçekleştirerek kullanım kolaylığı sunmaktadır .

3. TANIM

Dernek Takip Programı C# Windows Form Entity Framework kullanılarak geliştirilmiştir. Veri tabanı işlemleri için ilişkisel veri tabanı olan Microsoft SQL Server kullanılmıştır. Veri tabanı kontrolleri ve sorgulamaları Sql kullanarak yapılmıştır. C# projesi içinde kullanılacak olan katmanlar oluşturularak projede katmanlı mimari yaklaşımı benimsenmiştir. Sistemde borcu olan üyelerin bilgisinin pdf olarak belgelenmesi işleminde de ilgili paket indirilerek üzerinde gerekli işlemler yapılmıştır.

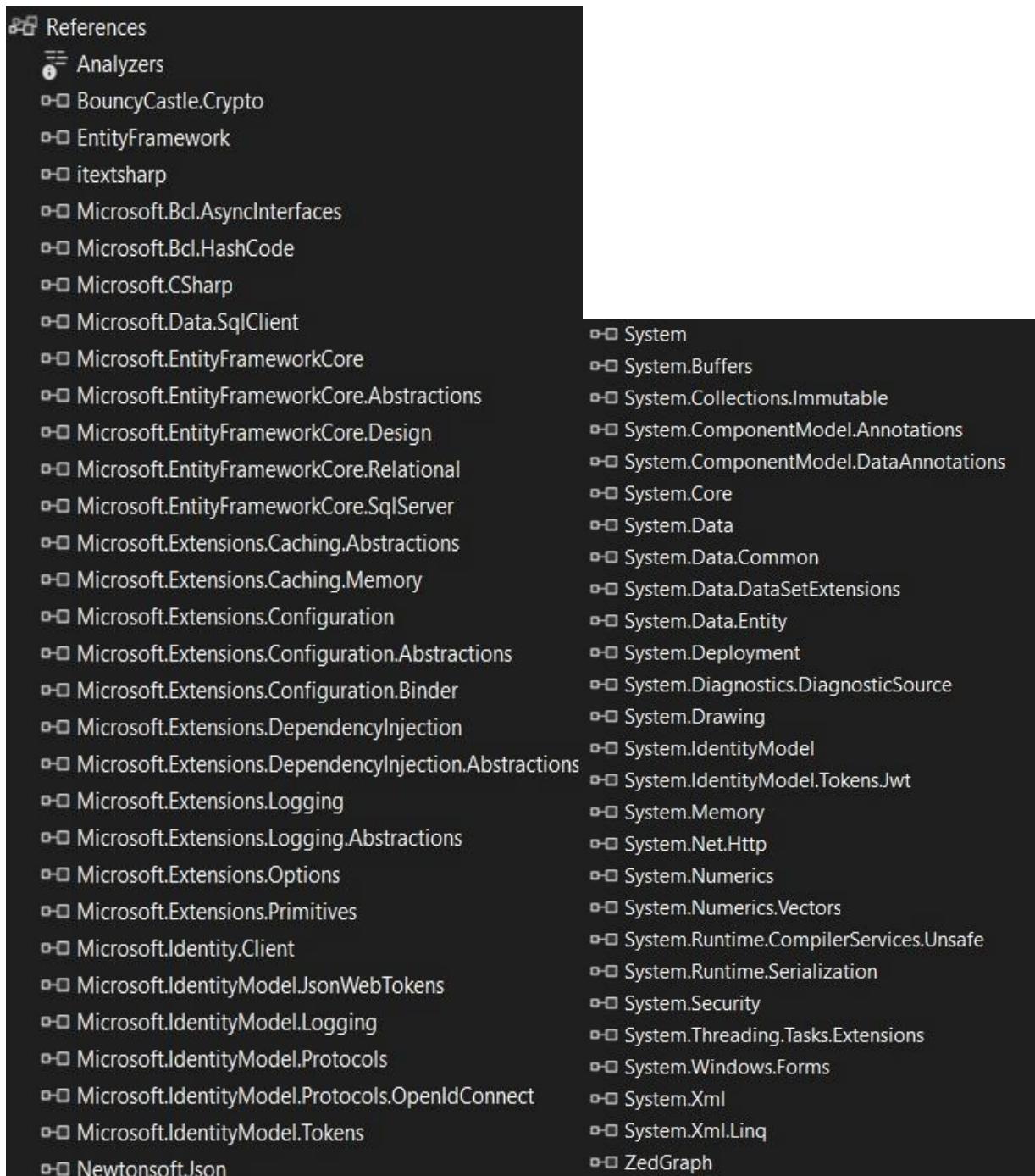
4. GEREKSİNİMLER VE İHTİYAÇLAR

Proje Visual Studio 2019 idesinde masaüstü geliştirme template'si kullanılmıştır.



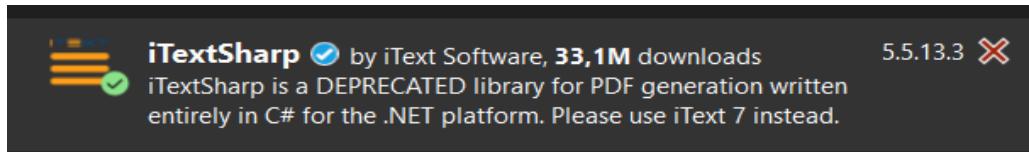
Şekil 4.1. Windows Forms App

Gerekli referanslar ve paketler eklenmiştir.



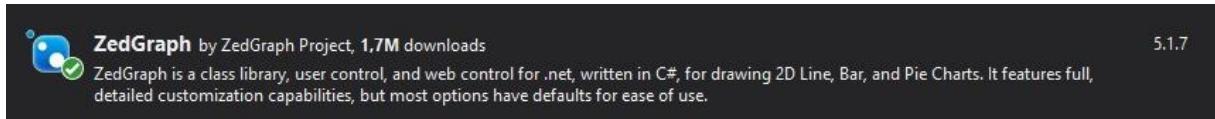
Şekil 4.2. Projede kullanılan referanslar

Pdf raporu oluşturmak için eklenen paket;



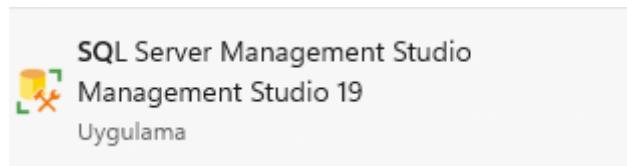
Şekil 4.3. iTextSharp paketi

Grafik işlemlerini görselleştirmek için eklenen paket;



Şekil 4.4. ZedGraph paketi

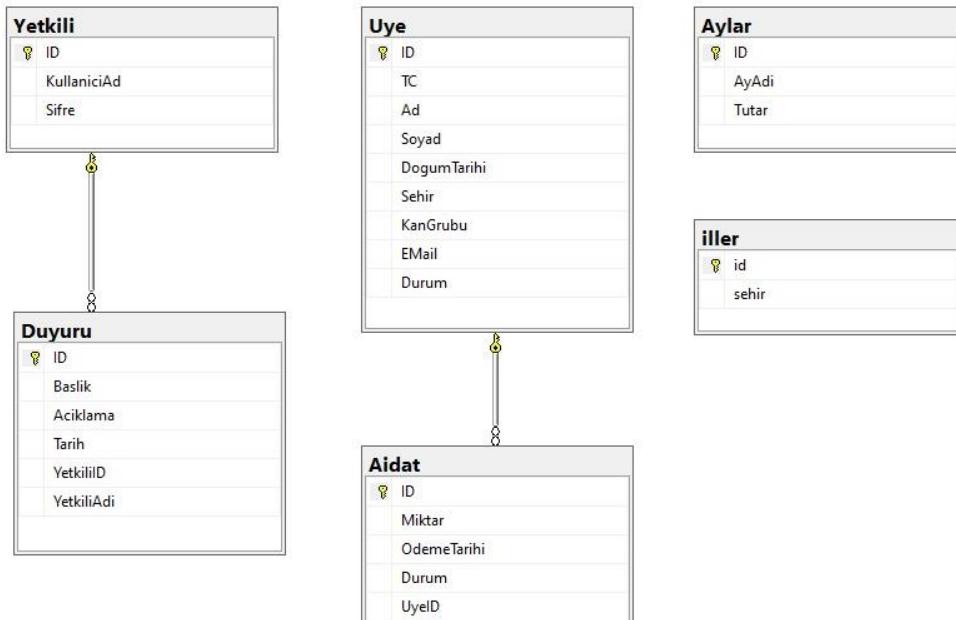
Veri tabanı işlemleri için kullandığımız Sql Server programı;



Şekil 4.5. SQL Server

5. ANALİZ

Projenin tasarım aşamasından önce analiz süreci gerçekleştirılmıştır. Proje üzerinde oluşturulacak sayfalar, gerekli veri tabanı tabloları ve oluşturulacak ilişkiler belirlenmiştir.



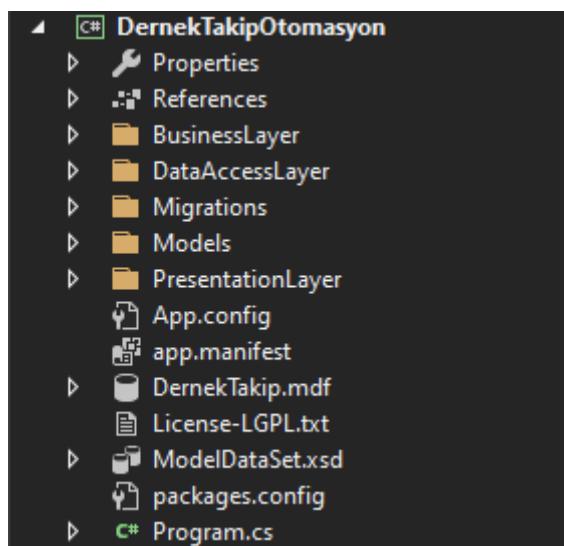
Şekil 5.1. Veri tabanı diyagramı

Üye tablosunda yer alan TC alanını benzersiz yapmak için unique olarak tanımlandı. Bu işlemi aşağıdaki kod aracılığıyla Sql Server üzerinden gerçekleştirdik.

```
Use DernekTakip
Go
Alter Table Uye
Add Constraint CK_unique UNIQUE(TC)
```

Şekil 5.2. TC alanı unique kodu

Proje katmanlı mimari üzerine tasarlanmış olup; Models, Business, Data Access ve Presentation katmanlarından oluşmaktadır. Migrations klasörü içerisindeki dosyalar veri tabanı değişiklikleri adım adım kaydeden ve uygulayan kod parçalarını içermektedir.

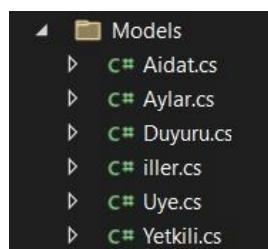


Şekil 5.3. Projede kullanılan katmanlı mimari

6. PROJE

6.1. Models Katmanı

Proje Code First yaklaşımı esas alınarak oluşturulmuştur. Models katmanı, uygulamanın veri tabanındaki tabloları temsil eden nesnelerini içerir ve bu nesneler, veri tabanındaki verileri almak ve kaydetmek için kullanılır.



Şekil 6.1.1. Models katmanı

Models katmanı içerisinde bulunan sınıfların içeriği aşağıdaki gibidir;

```
public class Aidat
{
    [Key]

    public int ID { get; set; }
    [StringLength(11, ErrorMessage = "TC alanı 11 karakterden az veya daha fazla olamaz.")]
    3 references
    public decimal Miktar { get; set; }
    5 references
    public DateTime OdemeTarihi { get; set; }
    3 references
    public bool Durum { get; set; }
    2 references
    public int UyeID { get; set; }
    [NotMapped]
    0 references
    public Uye Uye { get; set; }
}
```

Şekil 6.1.2. Aidat sınıfı

```
public class Aylar
{
    [Key]

    public int ID { get; set; }

    public string AyAdi { get; set; }
    14 references
    public decimal Tutar { get; set; }
}
```

Şekil 6.1.3. Aylar sınıfı

```
public class Duyuru
{
    [Key]

    public int ID { get; set; }

    public string YetkiliAdi { get; set; }

    public string Baslik { get; set; }

    public string Aciklama { get; set; }

    public DateTime Tarih { get; set; }

    public int YetkiliID { get; set; }
    [NotMapped]
    0 references
    public Yetkili Yetkili { get; set; }
}
```

Şekil 6.1.4. Duyuru sınıfı

```
public class iller
{
    [Key]

    public int id { get; set; }

    public string sehir { get; set; }
}
```

Şekil 6.1.5. İller sınıfı

```
public class Üye
{
    [Key]
    5 references
    public int ID { get; set; }

    [StringLength(11, ErrorMessage = "TC alanı 11 karakterden az veya daha fazla olamaz.")]
    7 references
    public string TC { get; set; }
    2 references
    public string Ad { get; set; }
    2 references
    public string Soyad { get; set; }
    2 references
    public DateTime DogumTarihi { get; set; }
    4 references
    public string Sehir { get; set; }
    3 references
    public string KanGrubu { get; set; }
    3 references
    public string EMail { get; set; }
    5 references
    public bool Durum { get; set; }
    0 references
    public List<Aidat> Aidats { get; set; }
}
```

Şekil 6.1.6. Üye sınıfı

```
public class Yetkili
{
    [Key]

    public int ID { get; set; }

    public string KullaniciAd { get; set; }

    public string Sifre { get; set; }
    0 references
    public List<Duyuru> Duyurus { get; set; }

}
```

Şekil 6.1.7. Yetkili sınıfı

[Key] Attribute'u ID alanları için primary key olarak tanımlamayı sağlar.

[NotMapped] Attribute'u ise veri tabanı tablosunda olmasını istemediğimiz alanı tanımlamak için kullanılır.

Üye ve Aidat tablosu arasındaki bire-çok ilişkiyi tanımlamak için gerekli kodlar sınıf içerisinde yazılırak ilişki oluşturulmuştur.

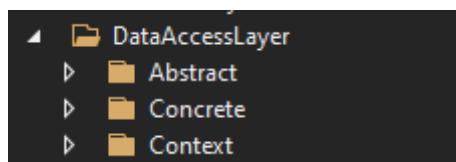
```
//Üye tablosunda ilişki için gerekli olan kod parçası
public List<Aidat> Aidats { get; set; }

//Aidat tablosunda ilişki için gerekli olan kod parçası
public int UyeID { get; set; }

[NotMapped]
public Uye Uye { get; set; }
```

6.2. Data Access Katmanı

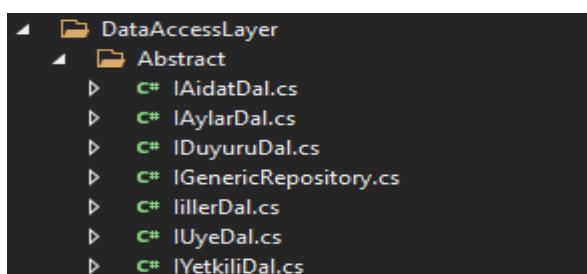
Data Access katmanını programımız ve veri tabanı arasındaki bağlantıyı sağlamak için oluşturduk. Bu katman içerisinde Abstract, Concrete ve Context adında 3 klasör oluşturduk.



Şekil 6.2.1. Data Access katman klasörleri

6.2.1 Abstract

Abstract klasörü, Data Access katmanı içerisinde bulunan interface'leri barındırmaktadır. Soyut sınıflar bu alanda bulunur.



Şekil 6.2.1.1. Abstract klasörü içerisinde bulunan interfaceler

Abstract klasörü içerisinde ilk olarak IGenericRepository sınıfını oluşturduk bu sınıf içerisinde temel CRUD (oluşturma, okuma, güncelleme, silme) işlemlerini tanımladık.

`IGenericRepository<T>` kodunda bulunan T bir classa karşılık gelmektedir. Bu class sayesinde CRUD işlemleri için yazdığımız metodların hepsi T türünde değer döndürür. T sınıfı aslında

bizim veri tabanımızda bulunan tablolara karşılık gelmektedir. Bu sayede kod tekrarına düşmeden işlemleri gerçekleştirmektedir.

```
// references
public interface IGenericRepository<T>
{
    7 references
    T GetById(int id); //Belirli ID'ye sahip olan varlığı getirir
    7 references
    List<T> GetAll(Expression<Func<T, bool>> filter = null); //Belirli filtreleme kriterine göre varlık getirir
    6 references
    void Add(T entity); //Yeni varlık ekleme
    6 references
    void Update(T entity); //Mevcut bulunan varlık üzerinde güncelleme işlemi
    6 references
    void Delete(T entity); //Mevcut bulunan varlığı silmek için kullanılır
}
```

Şekil 6.2.1.2. IGenericRepository sınıfı kodları

Data Access katmanında Abstract klasörü içerisinde bulunan IAidatDal, IAylarDal, IDuyuruDal, IillerDal, IUyeDal ve IYetkiliDal sınıflarının hepsi IGenericRepository sınıfından miras alırlar. Bu sayede IGenericRepository sınıfında bulunan metodların hepsini kullanabilmektedirler. Eğer bu genel metodlar haricinde sınıfı özgü bir metot tanımlanmak istenirse ilgili sınıf içerisinde o sınıfı özgü olarak tanımlanabilmektedir.

```
public interface IAidatDal : IGenericRepository<Aidat>
{
    //başlangıç ve bitiş parametrelerini alarak bu tarih arasındaki ödemeleri liste olarak döndürür
    2 references
    List<Aidat> TariheGoreOdemeler(string baslangic, string bitis);
}
```

Şekil 6.2.1.3. IAidatDal sınıfı kodları

```
public interface IAylarDal : IGenericRepository<Aylar>
{
    //Aylar varlığında bulunan tüm ayların listelenmesini sağlar
    2 references
    List<string> GetAylar();
}
```

Şekil 6.2.1.4. IAylarDal sınıfı kodları

```
public interface IDuyuruDal : IGenericRepository<Duyuru>
{
    //Bu sınıfı özgü herhangi bir metot yazılmamıştır.
}
```

Şekil 6.2.1.5. IDuyuruDal sınıfı kodları

```
public interface IillerDal : IGenericRepository<iller>
{
    //iller varlığında bulunan tüm illerin listelenmesini sağlar
    2 references
    List<string> GetSehir();
}
```

Şekil 6.2.1.6. IillerDal sınıfı kodları

```
public interface IUyeDal : IGenericRepository<Uye>
{
    //Belirli şehrə ait olan üyeleri liste olarak döndürür
    2 references
    List<Uye> SehreGoreUyeler(string sehir);
    //Belirli kan grubuna ait olan üyeleri liste olarak döndürür
    2 references
    List<Uye> KanGrubunaGoreUyeler(string kanGrubu);
    //Durumun aktif veya pasif olmasına göre üyeleri liste olarak döndürür
    2 references
    List<Uye> DurumaGoreUyeler(bool durum);
}
```

Şekil 6.2.1.7. IUyeDal sınıfı kodları

```
public interface IYetkiliDal : IGenericRepository<Yetkili>
{
    //Bu sınıfın özgü herhangi bir metod yazılmamıştır.
}
```

Şekil 6.2.1.8. IYetkiliDal sınıfı kodları

6.2.2 Concrete

Concrete klasöründe aslında Abstract klasöründe yaptığımız işlemlerin tam tersi işlemleri uygularız. Abstract da oluşturduğumuz soyut sınıfları bu klasörde somut olarak tanımlarız.



Şekil 6.2.2.1. Concrete klasörü içerisinde bulunan sınıflar

IGenericRepository interface'sinden miras alınarak GenericRepository kodları yazılır. Interfacede tanımlanan gövdesiz olarak verilen metodların hepsini bu somut sınıf içerisinde gövdeli olarak tanımlayarak gerekli kod blokları yazılır.

```

public class GenericRepository<T>: IGenericRepository<T> where T: class
{
    //Veri tabanına erişmek için db nesnesi oluştururuz.
    DernekTakipEntities db = new DernekTakipEntities();

    7 references
    public T GetByID(int id)
    {
        //veri tabanında belirli bir varlık türü içerisinde belirtilen ID'ye göre belirtilen varlığı bulur
        return db.Set<T>().Find(id);
    }

    7 references
    public List<T> GetAll(Expression<Func<T, bool>> filter = null)
    {
        //Veritabanındaki belirli bir varlık türünün tüm varlıklarını liste olarak getirir.
        return db.Set<T>().ToList();
    }

    6 references
    public void Add(T entity)
    {
        // Belirtilen varlığı veritabanına ekleyerek veri tabanına kaydeder.
        db.Set<T>().Add(entity);
        db.SaveChanges();
    }

    6 references
    public void Update(T entity)
    {
        //Belirtilen varlığı veritabanında günceller ve veri tabanında olan değişiklikleri kaydeder.
        db.Set<T>().Update(entity);
        db.SaveChanges();
    }

    6 references
    public void Delete(T entity)
    {
        // Belirtilen varlığı veritabanından siler ve veri tabanının son halini kaydeder.
        db.Set<T>().Remove(entity);
        db.SaveChanges();
    }
}

```

Şekil 6.2.2.2. GenericRepository sınıf kodları

```

public class AidatRepository : GenericRepository<Aidat>, IAidatDal
{
    DernekTakipEntities db = new DernekTakipEntities();

    2 references
    public List<Aidat> TariheGoreOdemeler(string baslangic, string bitis)
    {
        // LINQ sorgusu ile veritabanından ödemeleri tarih aralığına göre filtreleyip liste olarak getiriyor.
        return db.Aidat.Where(x => x.OdemeTarihi >= Convert.ToDateTime(baslangic) && x.OdemeTarihi <= Convert.ToDateTime(bitis)).ToList();
    }
}

```

Şekil 6.2.2.3. AidatRepository sınıf kodları

```

public class AylarRepository : GenericRepository<Aylar>, IAylarDal
{
    DernekTakipEntities db = new DernekTakipEntities();

    2 references
    public List<string> GetAylar()
    {
        // LINQ sorgusu ile veritabanından Aylar tablosundaki ay adlarını çekip liste olarak getiriyor.
        return db.Aylar.Select(y=>y.AyAdi).ToList();
    }
}

```

Şekil 6.2.2.4. AylarRepository sınıf kodları

```

public class DuyuruRepository : GenericRepository<Duyuru>, IDuyuruDal
{
    //Duyuru interface'de herhangi bir metot olmadığı için burada da herhangi
    //bir metot yazılmamıştır sadece GenericRepository miras alınır
}

```

Şekil 6.2.2.5. DuyuruRepository sınıf kodları

```

public class illerRepository : GenericRepository<iller>, IillerDal
{
    DernekTakipEntities db = new DernekTakipEntities();
    2 references
    public List<string> GetSehir()
    {
        // LINQ sorgusu ile veritabanından iller tablosundaki şehir adlarını çekip liste olarak getiriyor.
        return db.iller.Select(y => y.sehir).ToList();
    }
}

```

Şekil 6.2.2.6. illerRepository sınıf kodları

```

public class UyeRepository : GenericRepository<Uye>, IUyeDal
{
    DernekTakipEntities db = new DernekTakipEntities();

    2 references
    public List<Uye> DurumaGoreUyeler(bool durum)
    {
        // LINQ sorgusu ile veritabanından duruma göre üyeleri filtreleyip, liste olarak getiriyor.
        return db.Uye.Where(x => x.Durum.Equals(durum)).ToList();
    }

    2 references
    public List<Uye> KanGrubunaGoreUyeler(string kanGrubu)
    {
        // LINQ sorgusu ile veritabanından kan grubuna göre üyeleri filtreleyip, liste olarak getiriyor.
        return db.Uye.Where(x => x.KanGrubu.Equals(kanGrubu)).ToList();
    }

    2 references
    public List<Uye> SehreGoreUyeler(string sehir)
    {
        // LINQ sorgusu ile veritabanından şehrre göre üyeleri filtreleyip, liste olarak getiriyor.
        return db.Uye.Where(x => x.Sehir.Equals(sehir)).ToList();
    }
}

```

Şekil 6.2.2.7. UyeRepository sınıf kodları

```

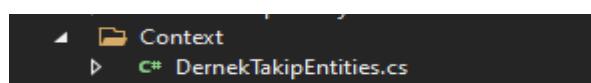
public class YetkiliRepository : GenericRepository<Yetkili>, IYetkiliDal
{
    // Yetkili interface'de herhangi bir metot olmadığı için burada da herhangi
    // bir metot yazılmamıştır sadece GenericRepository miras alınır
}

```

Şekil 6.2.2.8. YetkiliRepository sınıf kodları

6.2.3 Context

Context klasörünün içerisinde bulunan DernekTakipEntities sınıfı ile veri tabanı bağlantımızı burada gerçekleştirmekteyiz.



Şekil 6.2.3.1. Context klasörü içerisinde bulunan veri tabanı bağlantı sınıfı

Bağlantı metni içerisinde yazan Server kısmı kendi Ms Sql Server'ımıza bağlanmak için gerekli olan Server name'ini belirtmektedir.

```

public class DernekTakipEntities : DbContext
{
    0 references
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        // Localde veritabanını oluşturduk
        // optionsBuilder.UseSqlServer(@"Server=(localdb)\mssqllocaldb; Database=DernekTakip; Trusted_Connection=true");

        optionsBuilder.UseSqlServer(@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\DernekTakip.mdf;Integrated Security=True");
    }
    // Veritabanındaki alanları temsil eden DbSetler.
    4 references
    public DbSet<Aidat> Aidat { get; set; }
    3 references
    public DbSet<Aylar> Aylar { get; set; }
    0 references
    public DbSet<Duyuru> Duyuru { get; set; }
    1 reference
    public DbSet<iller> iller { get; set; }
    11 references
    public DbSet<Uye> Uye { get; set; }
    5 references
    public DbSet<Yetkili> Yetkili { get; set; }

    0 references
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        // Veritabanıoluştuğu anda aşağıdaki veriler Aylar tablosunda otomatik oluşturulacak.

        modelBuilder.Entity<Aylar>().HasData(
            new Aylar { ID = 1, AyAdi = "OCAK", Tutar = 100 },
            new Aylar { ID = 2, AyAdi = "SUBAT", Tutar = 200 },
            new Aylar { ID = 3, AyAdi = "MART", Tutar = 300 },
            new Aylar { ID = 4, AyAdi = "NİSAN", Tutar = 400 },
            new Aylar { ID = 5, AyAdi = "MAYIS", Tutar = 500 },
            new Aylar { ID = 6, AyAdi = "HAZİRAN", Tutar = 600 },
            new Aylar { ID = 7, AyAdi = "TEMMUZ", Tutar = 700 },
            new Aylar { ID = 8, AyAdi = "AĞUSTOS", Tutar = 800 },
            new Aylar { ID = 9, AyAdi = "EYLÜL", Tutar = 900 },
            new Aylar { ID = 10, AyAdi = "EKİM", Tutar = 1000 },
            new Aylar { ID = 11, AyAdi = "KASIM", Tutar = 1100 },
            new Aylar { ID = 12, AyAdi = "ARALIK", Tutar = 1200 });
    }

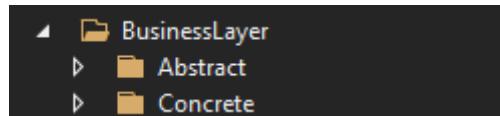
    // Yetkili tablosuna varsayılan bir admin adında yetkili ekleniyor.
    modelBuilder.Entity<Yetkili>().HasData(
        new Yetkili { ID=1, KullaniciAd="admin", Sifre="000" });
}

```

Şekil 6.2.3.2. DernekTakipEntities Sql bağlantı sınıfı kodları

6.3. Business Katmanı

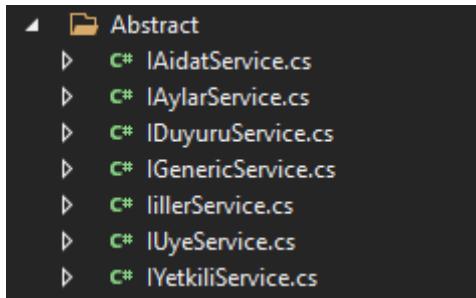
Uygulamanın akışını anlatan katmandır. Data Access ve Presentation katmanları arasında ara katman olarak tanımlanır. Bu katmanı, Data Access katmanı ile veri tabanından projemize çekmiş olduğumuz verileri alıp işledikten sonra kullanabiliriz. Kullanıcıdan gelen veriler Business katmanına gelir bu katmanda veriler işlenerek Data Access katmanına aktarılır.



Şekil 6.3.1. Business katmanı klasörleri

6.3.1 Abstract

İş akışında yer alan arayüzler bu klasörde tanımlanır.



Şekil 6.3.1.1. Abstract klasöründe bulunan interfaceler

Abstract klasörü içerisinde ilk olarak IGenericService sınıfını oluşturduk bu sınıf, genel CRUD (oluşturma, okuma, güncelleme, silme) işlemlerini yönetmek üzere oluşturulmuştur. Bu arayüz sayesinde iş mantığını uygulamadan önce temel veri tabanı işlemleri soyutlaştırarak standartlaştırırız. `IGenericService<T>` kodunda bulunan T bir classa karşılık gelmektedir. Bu class sayesinde CRUD işlemleri için yazdığımız metodların hepsi T türünde değer döndürür. T sınıfı veri tabanımızda bulunan tablolara karşılık gelmektedir.

```
public interface IGenericService<T>
{
    //verilen bir varlığı oluşturup veri tabanına ekler.
    void TAdd(T t);
    //verilen varlığın silinip veri tabanından da kaldırılmasını sağlar.
    void TDelete(T t);
    //verilen varlığı güncelleyip veri tabanı üzerinde de güncellenmesini sağlar.
    void TUpdate(T t);
    //veri tabanında bulunan belirli varlık türüne ait tüm varlıklarını getirmesini sağlar.
    List<T> GetList();
    //veri tabanında bir varlığı belirtlen ID değerine göre getirmesini sağlar
    T T GetById(int id);
}
```

Şekil 6.3.1.2. IGenericService genel ara yüz kodları

Business katmanında Abstract klasörü içerisinde bulunan IAidatService, IAylarService, IDuyuruService, IillerService, IUyeService ve IYetkiliService sınıflarının hepsi IGenericService sınıfından miras alırlar. Bu sayede IGenericService sınıfında bulunan metodların hepsini kullanabilmektedirler. Eğer bu genel metodlar haricinde sınıfı özgü bir metod tanımlanmak istenirse ilgili sınıf içerisinde o sınıfı özgü olarak tanımlanabilmektedir.

```
public interface IAidatService:IGenericService<Aidat>
{
    // Belirli bir tarih aralığına göre ödemeleri getirmek için kullanılır.
    List<Aidat> TariheGoreOdemeler(string baslangic, string bitis);
}
```

Şekil 6.3.1.3. IAidatService sınıfı kodları

```
public interface IAylarService:IGenericService<Aylar>
{
    // Aylar tablosundan ay adlarını liste olarak getirir.
    3 references
    List<string> GetAylar();
}
```

Şekil 6.3.1.4. IAylarService sınıfı kodları

```
public interface IDuyuruService:IGenericService<Duyuru>
{
    //bu sınıfı özel bir işlev tanımlanmamıştır.
}
```

Şekil 6.3.1.5. IDuyuruService sınıfı kodları

```
public interface IIllerService : IGenericService<iller>
{
    // iller tablosunda bulunan şehir adlarını liste olarak getirir
    3 references
    List<string> GetSehir();
}
```

Şekil 6.3.1.6. IIllerService sınıfı kodları

```
2 references
public interface IUyeService:IGenericService<Uye>
{
    //Belirtilen şehrre göre üyeleri listeler
    2 references
    List<Uye> SehreGoreUyeler(string sehir);
    //Belirtilen kangrubuna göre üyeleri listeler
    2 references
    List<Uye> KanGrubunaGoreUyeler(string kanGrubu);
    //Belirtilen duruma göre üyeleri listeler
    3 references
    List<Uye> DurumaGoreUyeler(bool durum);
}
```

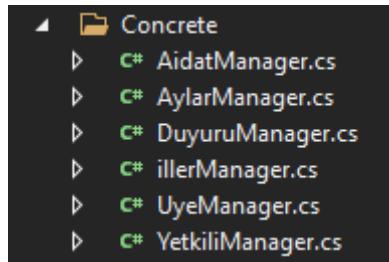
Şekil 6.3.1.7. IUyeService sınıfı kodları

```
1 reference
public interface IYetkiliService:IGenericService<Yetkili>
{
    //bu sınıfı özel bir işlev tanımlanmamıştır.
}
```

Şekil 6.3.1.8. IYetkiliService sınıfı kodları

6.3.2 Concrete

Concrete klasöründe aslında Abstract klasöründe yaptığımız işlemlerin tam tersi işlemleri uygularız. Abstract'da oluşturduğumuz soyut sınıfları bu klasörde somut olarak tanımlayarak gerçek iş akışını uygulamış oluruz.



Şekil 6.3.2.1. Concrete klasörü içerisindeki sınıflar

Aidat sınıfının işlemlerini yürütmek için AidatManager sınıfı oluşturulur. AidatManager sınıfı IAidatService arayüzüni uygulamaktadır. IAidatDal arayüzüne kullanarak veri tabanında aidat verileriyle ilgili işlemleri gerçekleştirir. Bu sınıf, Aidat nesnelerini ekleme, güncelleme, silme, listeleme, ID' ye göre listeleme işlemlerini yapmaktadır. Bu işlemler haricinde bu sınıfın özel olarak belirtilen metod tarihe özel olarak listeleme yapmaktadır.

```

public class AidatManager : IAidatService
{
    //veri tabanı erişim işlemleri için kullanılacak veri erişim katmanı
    IAidatDal _aidatDal;

    2 references
    public AidatManager(IAidatDal aidatDal)
    {
        _aidatDal = aidatDal;
    }

    2 references
    public List<Aidat> GetList()
    {
        // Veritabanındaki tüm Aidat varlıklarının listesi.
        return _aidatDal.GetAll();
    }

    2 references
    public void TAdd(Aidat t)
    {
        // - t: Eklenecek istenen Aidat varlığı veri tabanına eklenir.
        _aidatDal.Add(t);
    }

    2 references
    public List<Aidat> TariheGoreOdemeler(string baslangic, string bitis)
    {
        //Belirtilen tarih aralığına uygun Aidat varlıklarının listesi.
        return _aidatDal.TariheGoreOdemeler(baslangic, bitis);
    }
}

```

```

1 reference
public void TDelete(Aidat t)
{
    //t: Silinmek istenen Aidat varlığı veri tabanından silinir.
    _aidatDal.Delete(t);
}

1 reference
public Aidat T GetById(int id)
{
    //Belirtilen ID değerine sahip Aidat varlığı.
    return _aidatDal.GetById(id);
}

1 reference
public void TUpdate(Aidat t)
{
    //Belirtilen kimlik değerine sahip Aidat varlığı veri tabanında güncellenir.
    _aidatDal.Update(t);
}

```

Şekil 6.3.2.2. – Şekil 6.3.2.3. AidatManager sınıf kodları

Aylar sınıfının işlevlerini执行mek için AylarManager sınıfı oluşturulur. AylarManager sınıfı IAylarService arayüzüne uygulamaktadır. IAylarDal arayüzüne kullanarak veri tabanında aylar verileriyle ilgili işlemleri gerçekleştirir. Bu sınıf, Aylar nesnelerini ekleme, güncelleme, silme, listeleme, ID' ye göre listeleme işlevlerini yapmaktadır. Bu işlemler haricinde bu sınıfa özel olarak oluşturulan metod veri tabanında bulunan tüm Ay kayıtlarını listeleme yapmaktadır.

```

public class AylarManager : IAylarService
{
    IAylarDal _aylarDal;

    2 references
    public AylarManager(IAylarDal aylarDal)
    {
        _aylarDal = aylarDal;
    }

    3 references
    public List<string> GetAylar()
    {
        // Veritabanındaki tüm iller varlıklarının listesi.
        return _aylarDal.GetAylar();
    }

    3 references
    public List<Aylar> GetList()
    {
        //veri tabanında bulunan tüm ay varlıklarını listeler
        return _aylarDal.GetAll();
    }
}

```

```

1 reference
public void TAdd(Aylar t)
{
    // - t: Eklenecek istenen aylar varlığı veri tabanına eklenir.
    _aylarDal.Add(t);
}

1 reference
public void TDelete(Aylar t)
{
    //t: Silinmek istenen aylar varlığı veri tabanından silinir.
    _aylarDal.Delete(t);
}

2 references
public Aylar TGetById(int id)
{
    //Belirtilen ID değerine sahip aylar varlığı.
    return _aylarDal.GetByID(id);
}

2 references
public void TUpdate(Aylar t)
{
    //Belirtilen kimlik değerine sahip aylar varlığı veri tabanında güncellenir.
    _aylarDal.Update(t);
}

```

Şekil 6.3.2.4. – Şekil 6.3.2.5. AylarManager sınıf kodları

Duyuru sınıfının işlemlerini执行mek için DuyuruManager sınıfı oluşturulur.

DuyuruManager sınıfı IDuyuruService arayüzüünü uygulamaktadır. IDuyuruDal arayüzüünü kullanarak veri tabanında duyuru verileriyle ilgili işlemleri gerçekleştirir. Bu sınıf, Duyuru nesnelerini ekleme, güncelleme, silme, listeleme, ID' ye göre listeleme işlemlerini yapmaktadır. Bu işlemler haricinde bu sınıfa özel olarak tanımlanan bir metot bulunmamaktadır.

```

1 reference
public class DuyuruManager : IDuyuruService
{
    IDuyuruDal _duyuruDal;

    1 reference
    public DuyuruManager(IDuyuruDal duyuruDal)
    {
        _duyuruDal = duyuruDal;
    }
    2 references
    public List<Duyuru> GetList()
    {
        // Veritabanındaki tüm duyuru varlıklarının listesi.
        return _duyuruDal.GetAll();
    }

    2 references
    public void TAdd(Duyuru t)
    {
        // - t: Eklenecek istenen duyuru varlığı veri tabanına eklenir.
        _duyuruDal.Add(t);
    }

    1 reference
    public void TDelete(Duyuru t)
    {
        //t: Silinecek istenen duyuru varlığı veri tabanından silinir.
        _duyuruDal.Delete(t);
    }

    1 reference
    public Duyuru T GetById(int id)
    {
        //Belirtilen ID değerine sahip duyuru varlığı.
        return _duyuruDal.GetByID(id);
    }

    1 reference
    public void TUpdate(Duyuru t)
    {
        //t: güncellenmek istenen duyuru varlığı veri tabanında güncellenir.
        _duyuruDal.Update(t);
    }
}

```

Şekil 6.3.2.6. DuyuruManager sınıf kodları

İller sınıfının işlemlerini执行mek için illerManager sınıfı oluşturulur. illerManager sınıfı IllerService arayüzü uygulamaktadır. illerDal arayüzü kullanarak veri tabanında iller verileriyle ilgili işlemleri gerçekleştirir. Bu sınıf, iller nesnelerini ekleme, güncelleme, silme, listeleme, ID' ye göre listeleme işlemlerini yapmaktadır. Ekleme, silme ve güncelleme işlemleri IllerService sınıfından miras almaktadır fakat kullanılmamaktadır. Bu işlemler haricinde bu sınıfa özel olarak tanımlanan metod veri tabanında bulunan tüm şehir varlıklarını listeleme yapmaktadır.

```

3 references
public class illerManager : IillerService
{
    IillerDal _illerDal;

    2 references
    public illerManager(IillerDal illerDal)
    {
        _illerDal = illerDal;
    }

    1 reference
    public List<iller> GetList()
    {
        return _illerDal.GetAll();
    }

    3 references
    public List<string> GetSehir()
    {
        return _illerDal.GetSehir();
    }

    1 reference
    public void TAdd(iller t)
    {
        throw new NotImplementedException();
    }

    1 reference
    public void TDelete(iller t)
    {
        throw new NotImplementedException();
    }

    1 reference
    public iller T GetById(int id)
    {
        return _illerDal.GetById(id);
    }

    1 reference
    public void TUpdate(iller t)
    {
        throw new NotImplementedException();
    }
}

```

Şekil 6.3.2.7. illerManager sınıf kodları

Üye sınıfının işlemlerini yürütmek için UyeManager sınıfı oluşturulur. UyeManager sınıfı IUyeService arayüzüünü uygulamaktadır. IUyeDal arayüzüünü kullanarak veri tabanında üye verileriyle ilgili işlemleri gerçekleştirir. Bu sınıf, üye nesnelerini ekleme, güncelleme, silme, listeleme, ID' ye göre listeleme işlemlerini yapmaktadır. Bu işlemler haricinde bu sınıfın özel

olarak kan grubuna, duruma ve şehrre göre üyeleri listeleme işlemleri gerçekleştirmek için metodlar tanımlanmıştır.

```
public class UyeManager : IUyeService
{
    IUyeDal _uyeDal;

    2 references
    public UyeManager(IUyeDal uyeDal)
    {
        _uyeDal = uyeDal;
    }

    3 references
    public List<Uye> DurumaGoreUyeler(bool durum)
    {
        return _uyeDal.DurumaGoreUyeler(durum);
    }

    2 references
    public List<Uye> GetList()
    {
        return _uyeDal.GetAll();
    }

    2 references
    public List<Uye> KanGrubunaGoreUyeler(string kanGrubu)
    {
        return _uyeDal.KanGrubunaGoreUyeler(kanGrubu);
    }

    2 references
    public List<Uye> SehreGoreUyeler(string sehir)
    {
        return _uyeDal.SehreGoreUyeler(sehir);
    }

    2 references
    public void TAdd(Uye t)
    {
        _uyeDal.Add(t);
    }

    2 references
    public void TDelete(Uye t)
    {
        _uyeDal.Delete(t);
    }

    3 references
    public Uye T GetById(int id)
    {
        return _uyeDal.GetById(id);
    }

    2 references
    public void TUpdate(Uye t)
    {
        _uyeDal.Update(t);
    }
}
```

Şekil 6.3.2.8. UyeManager sınıf kodları

Yetkili sınıfının işlemlerini执行为了执行 YetkiliManager sınıfı oluşturulur. YetkiliManager sınıfı IYetkiliService arayüzüne uygulamaktadır. IYetkiliDal arayüzüne kullanarak veri

tabanında yetkili verileriyle ilgili işlemleri gerçekleştirir. Bu sınıf, yetkili nesnelerini ekleme, güncelleme, silme, listeleme, ID' ye göre listeleme işlemlerini yapmaktadır. Bu işlemler haricinde bu sınıfa özel olarak tanımlanan bir metod bulunmamaktadır.

```
public class YetkiliManager : IYetkiliService
{
    IYetkiliDal _yetkiliDal;

    public YetkiliManager(IYetkiliDal yetkiliDal)
    {
        _yetkiliDal = yetkiliDal;
    }

    public List<Yetkili> GetList()
    {
        return _yetkiliDal.GetAll();
    }

    public void TAdd(Yetkili t)
    {
        _yetkiliDal.Add(t);
    }

    public void TDelete(Yetkili t)
    {
        _yetkiliDal.Delete(t);
    }

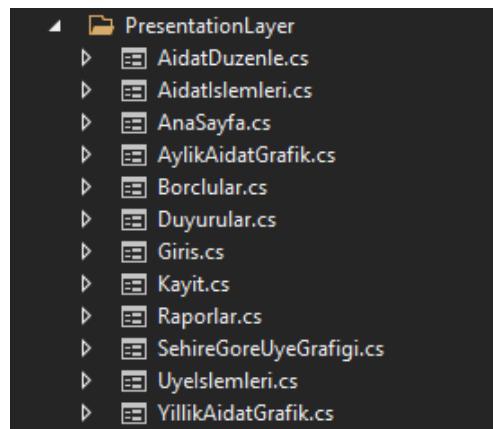
    public Yetkili TGetById(int id)
    {
        return _yetkiliDal.GetById(id);
    }

    1 reference
    public void TUpdate(Yetkili t)
    {
        _yetkiliDal.Update(t);
    }
}
```

Şekil 6.3.2.9. YetkiliManager sınıf kodları

6.4. Presentation Katmanı

Presentation katmanı uygulamanın kullanıcıyla etkileşime geçtiği katmandır. Dernek takip otomasyon projesini Windows Forms template’i kullanarak oluşturduğumuz için bizim projemizde kullanıcıyla etkileşime geçecek sayfalar form ekranları olmaktadır. Bu katmanda amacımız kullanıcıdan verileri alarak Business katmanında işlenerek daha sonra Data Access katmanına iletmektir. Data Access’de olan veriler de Business katmanı aracılığıyla Presentation katmanında kullanıcıya gösterilir. Bu katmanın doğrudan veriye erişimi yoktur.



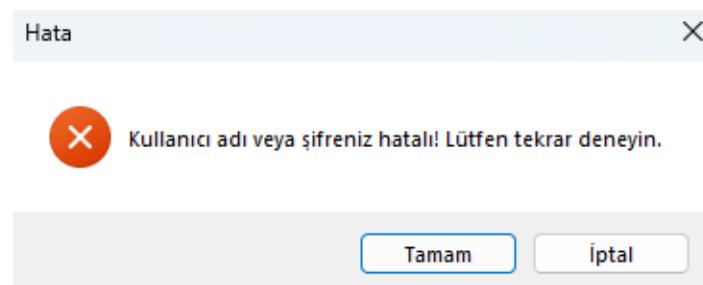
Şekil 6.4.1. Presentation katmanında bulunan form ekranları

6.4.1 Giriş Ekranı

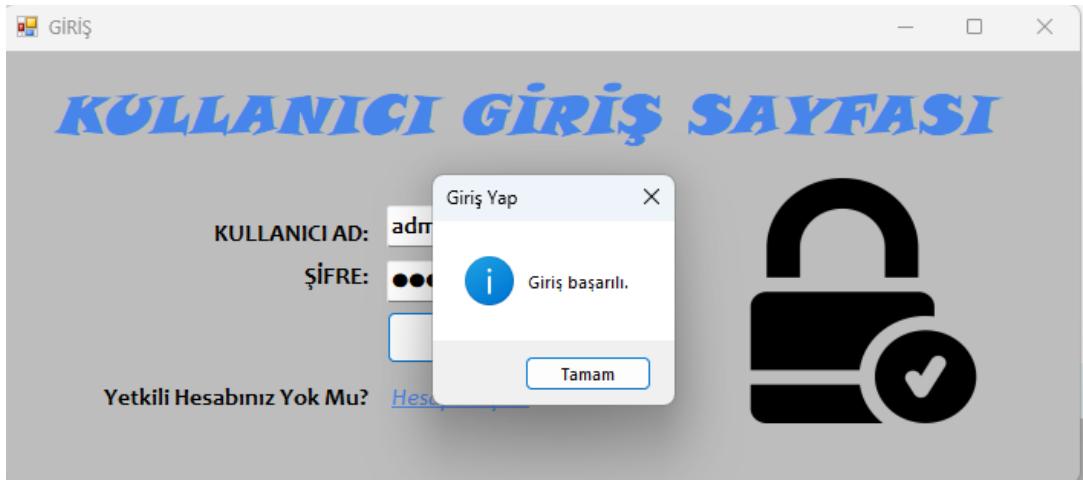
Otomasyon projemize ilk olarak giriş ekranı üzerinden sayfaya erişim sağlarız. Bu alanda sisteme üye olan yetkili kişiler kullanıcı adı ve şifrelerini doğru girdikleri takdirde sisteme giriş yaparlar. Yanlış kullanıcı adı ya da şifreyle giriş yapan yetkililer için ise uyarı mesajı verir ve tekrardan bilgilerini doğru bir şekilde girmelerini ister. Sisteme üye olmayan yetkililer için ise yine aynı ekran üzerinde bulunan hesap oluştur butonunu kullanarak bilgileri eksiksiz doldurduktan sonra hesap oluşturabilirler.



Şekil 6.4.1.1. Form Giriş ekranı



Şekil 6.4.1.2. Kullanıcı adı ve şifre yanlış girildiğinde verilen hata mesajı



Şekil 6.4.1.3. Kullanıcı adı ve şifre doğru girildiğinde verilen mesaj

```
private void btnGiris_Click(object sender, EventArgs e)
{
    DernekTakipEntities db = new DernekTakipEntities();

    //Kullanıcı adı ve şifre birbirine uyuyor mu diye kontrol eden LINQ sorusu
    var yetkili = db.Yetkili.FirstOrDefault(x => x.KullaniciAd == txtKullaniciAd.Text && x.Sifre == txtSifre.Text);

    if (yetkili != null)
    {
        MessageBox.Show("Giriş başarılı.", "Giriş Yap", MessageBoxButtons.OK, MessageBoxIcon.Information);
        AnaSayfa anaSayfa = new AnaSayfa();
        anaSayfa.Show();
    }
    else
    {
        MessageBox.Show("Kullanıcı adı veya şifreniz hatalı! Lütfen tekrar deneyin.", "Hata", MessageBoxButtons.OKCancel, MessageBoxIcon.Error);
    }
}
```

Şekil 6.4.1.4. Giriş yap butonuna basıldığında yapılması gereken işlemlerin kodu

```
private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    //Hesap oluştur butonuna basıldığında bizi kayıt sayfasına yönlendirir
    Kayit kayit = new Kayit();
    kayit.Show();
}
```

Şekil 6.4.1.5. Hesap oluştur butonuna basıldığında yapılması gereken işlemlerin kodu

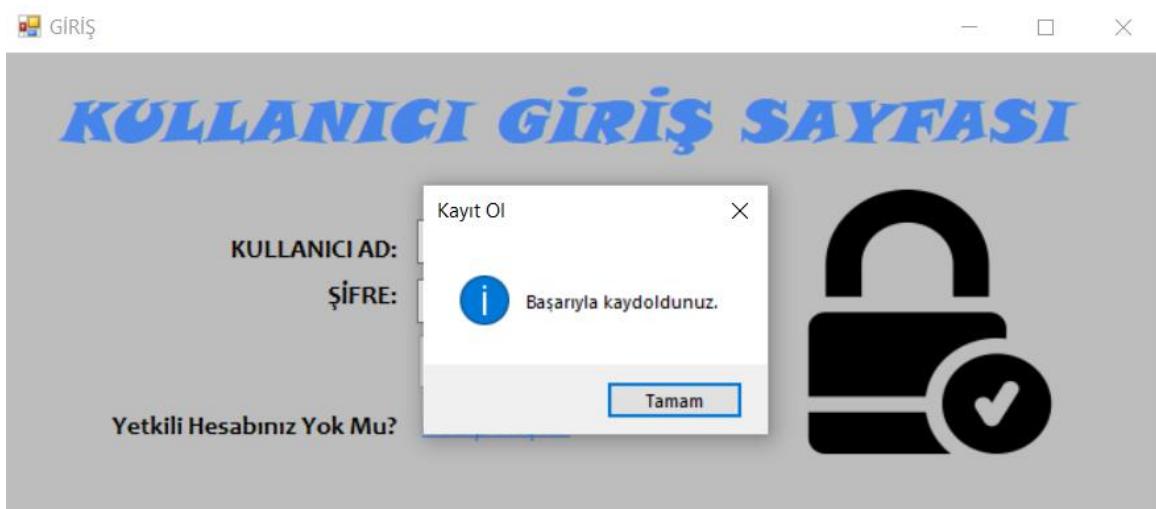
6.4.2 Kayıt Ekranı

Dernek Takip uygulamamızda hesabı olmayan yetkililer için kayıt sayfası yapılmıştır. Kayıt paneli üzerinden yetkililer kendilerine bir kullanıcı ad ve şifre belirleyerek sisteme kayıt olabilirler. Şifre tekrar kısmını hatalı ya da belirledikleri şifreyle eşit girmedikleri takdirde yetkiliye bir mesaj ile uyarıda bulunulmaktadır. Ayrıca eğer veri tabanında kayıtlı olan bir kullanıcı ad ile sisteme kayıt olmaya çalışırsa bu kullanıcı adının zaten alınmış olduğunu belirten bir mesaj ile uyarıda bulunulmaktadır. Tüm alanları doğru ve eksiksiz girdiği

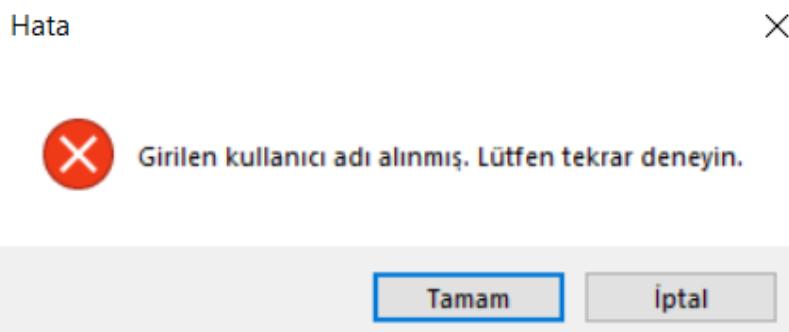
zamanda da başarıyla kaydolduğunu belirten mesaj döndürülmekte olup kullanıcı ad ve şifresini girerek sisteme erişim sağlayabilmesi için karşısına giriş paneli getirilmektedir.



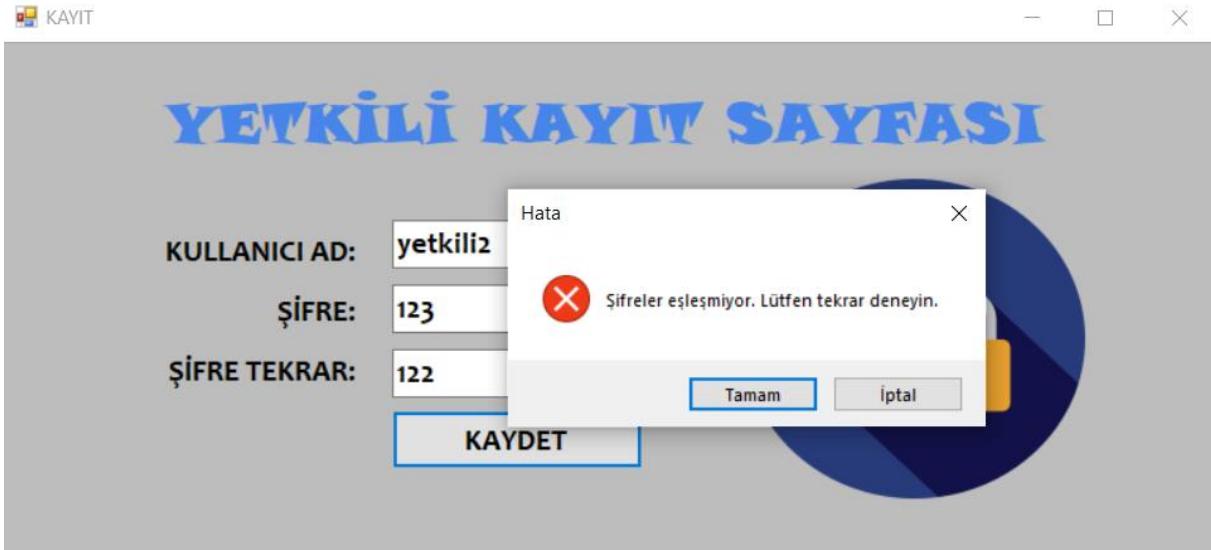
Şekil 6.4.2.1. Form Kayıt Ekranı



Şekil 6.4.2.2. Kullanıcı adı ve şifre bilgileri hatasız girildiğinde verilen mesaj



Şekil 6.4.2.3. Veri tabanında kayıtlı olan kullanıcı ad ile kayıt olmaya çalışıldığında verilen uyarı mesajı



Şekil 6.4.2.4. Şifreler eşleşmediği takdirde döndürulen hata mesajı

```
// Yetkili tablosu için oluşturulan mimariye erişebilmek adına nesne üretildi.
YetkiliManager yetkiliManager = new YetkiliManager(new YetkiliRepository());

// Veri tabanından direkt olarak erişmemiz gerekiği durumlarda nesne üretildi.
DernekTakipEntities db = new DernekTakipEntities();
private void btnKaydet_Click(object sender, EventArgs e)
{
    Yetkili yetkili = new Yetkili();
    yetkili.KullaniciAd = txtkullanici.Text;

    var kullaniciAdVar = db.Yetkili.Where(x => x.KullaniciAd.Equals(txtkullanici.Text)).FirstOrDefault();

    // Girilen kullanıcı adı veri tabanında varsa uyarı mesajı döndüren kod bloğu.
    if (kullaniciAdVar != null)
    {
        MessageBox.Show("Girilen kullanıcı adı alınmış. Lütfen tekrar deneyin.", "Hata", MessageBoxButtons.OKCancel, MessageBoxIcon.Error);
        txtkullanici.Text = "";
        txtSifre.Text = "";
        txtSifreTekrar.Text = "";
    }

    // Kullanıcı adı önceden alınmamışsa çalışacak olan kod bloğu.
    else
    {
        if (txtSifre.Text == txtSifreTekrar.Text) // Şifre alanları eşit girilmişse çalışır
        {
            yetkili.Sifre = txtSifre.Text;
            yetkiliManager.Add(yetkili);

            Giris giris = new Giris();
            giris.Show();

            MessageBox.Show("Başarıyla kaydoldunuz.", "Kayıt Ol", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        else
        {
            MessageBox.Show("Şifreler eşleşmiyor. Lütfen tekrar deneyin.", "Hata", MessageBoxButtons.OKCancel, MessageBoxIcon.Error);
        }
    }
}
```

Şekil 6.4.2.5. Kayıt ol butonuna basıldığında çalışacak olan kod bloğu

6.4.3 Ana Sayfa

Sisteme giriş sağlayan yetkili kullanıcıların karşısına ana sayfa çıkmaktadır. Bu sayfada üye ve aidat işlemleri, raporlamalar veya grafiksel gösterimler gibi ekranlar yer almaktadır.



Şekil 6.4.3.1. Ana Sayfa

6.4.4 Üye İşlemleri Ekranı

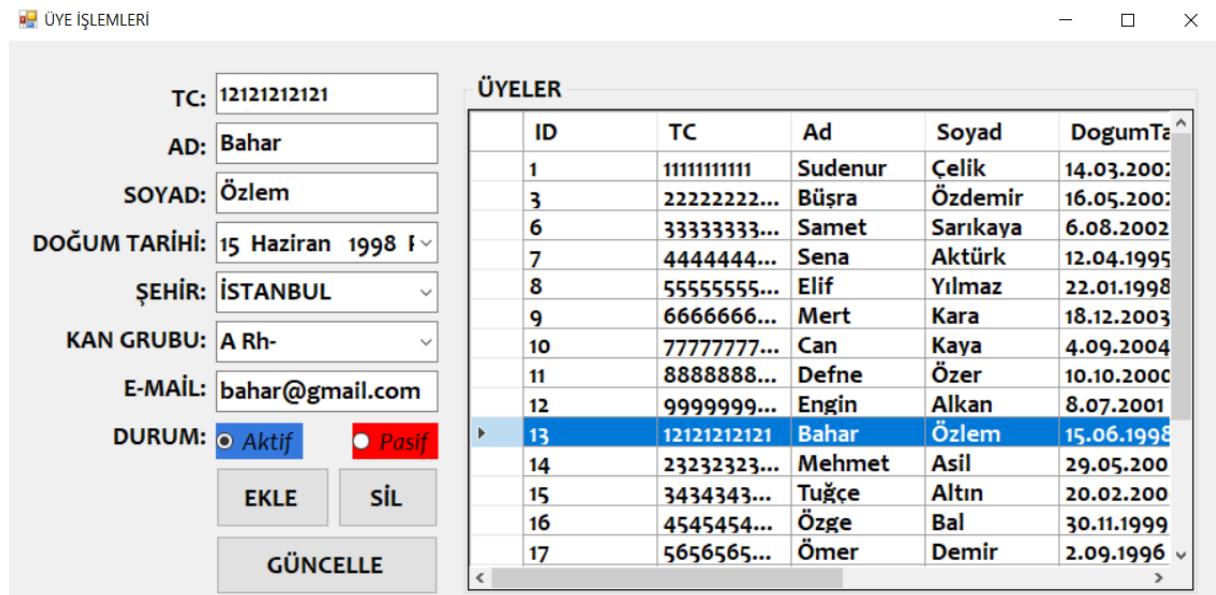
Üye işlemleri ekranında yetkililer derneğe üye olmak isteyen kişilerin bilgilerini kaydedebilir ya da bilgileri kayıtlı olan üyeleri üzerinde güncelleme işlemleri yapabilir. Dernek üyeliğinden çıkmak isteyenlerin kaydını seçip direkt silebilir ya da üyeliğini pasif hale getirebilir. Dernek sistemi üzerinde silme, güncelleme ya da ekleme işlemi sırasında kullanıcıya gerekli uyarı ya da mesajlarla yetkili kullanıcı bilgilendirilmiştir.

Üye TC alanı için MaskedTextBox aracı kullanılmış olup TC alanına 11 karakterden fazla veri girişi yasaklanmıştır.

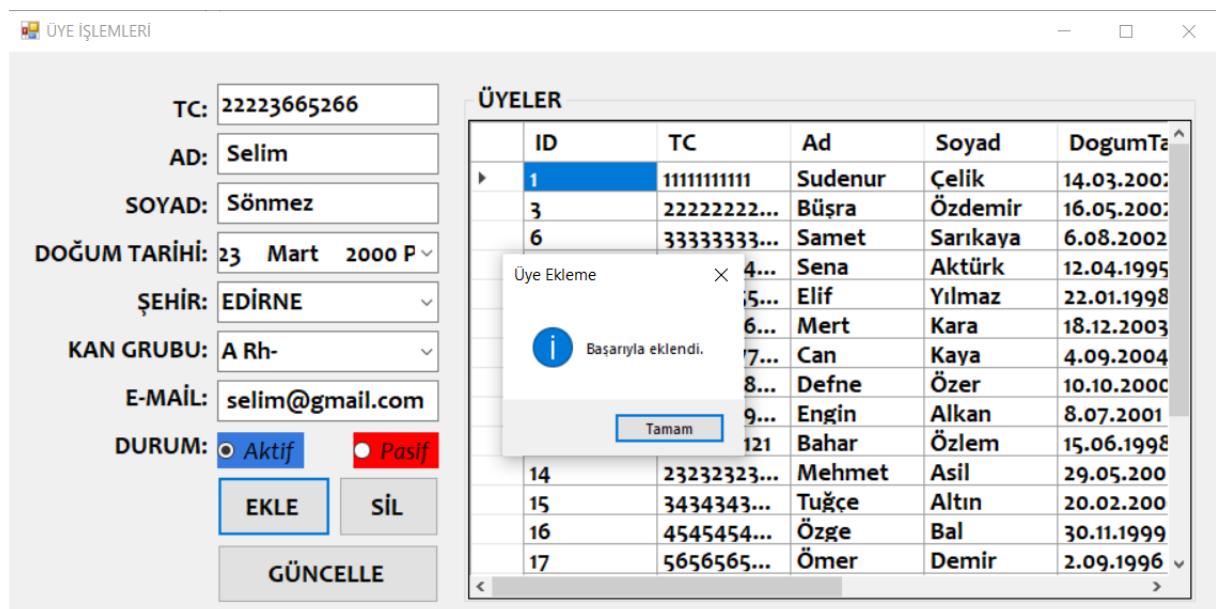
ÜYELER				
ID	TC	Ad	Soyad	DogumTarihi
1	11111111111	Sudenur	Celik	14.03.2001
3	222222222...	Büşra	Özdemir	16.05.2000
6	333333333...	Samet	Sarıkaya	6.08.2002
7	44444444...	Sena	Aktürk	12.04.1995
8	55555555...	Elif	Yılmaz	22.01.1998
9	6666666...	Mert	Kara	18.12.2003
10	77777777...	Can	Kaya	4.09.2004
11	8888888...	Defne	Özer	10.10.2000
12	9999999...	Engin	Alkan	8.07.2001
13	12121212121	Bahar	Özlem	15.06.1998
14	23232323...	Mehmet	Asıl	29.05.2000
15	3434343...	Tuğçe	Altın	20.02.2000
16	4545454...	Özge	Bal	30.11.1999
17	5656565...	Ömer	Demir	2.09.1996

Şekil 6.4.4.1. Üye işlemleri ekranı

Aşağıda listelenen üyeleri içerisinden seçilen bir kişinin bilgilerinin textbox ya da ilgili alanlara aktarıldığı gösterilmektedir. Yetkili kullanıcı aktarılan bu bilgiler vasıtasıyla ilgili kişinin bilgileri üzerinde güncelleme yapabilir ya da direkt bu üyeyi silme işlemi yapabilir.

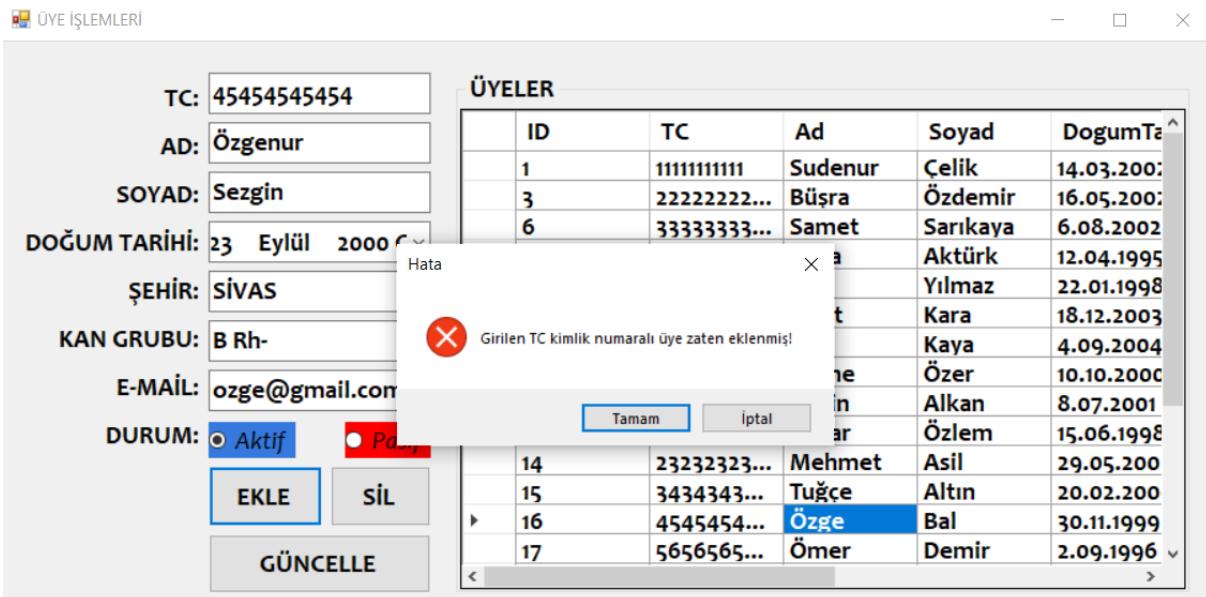


Şekil 6.4.4.2. Seçilen kişi bilgilerinin textbox'lara aktarımı



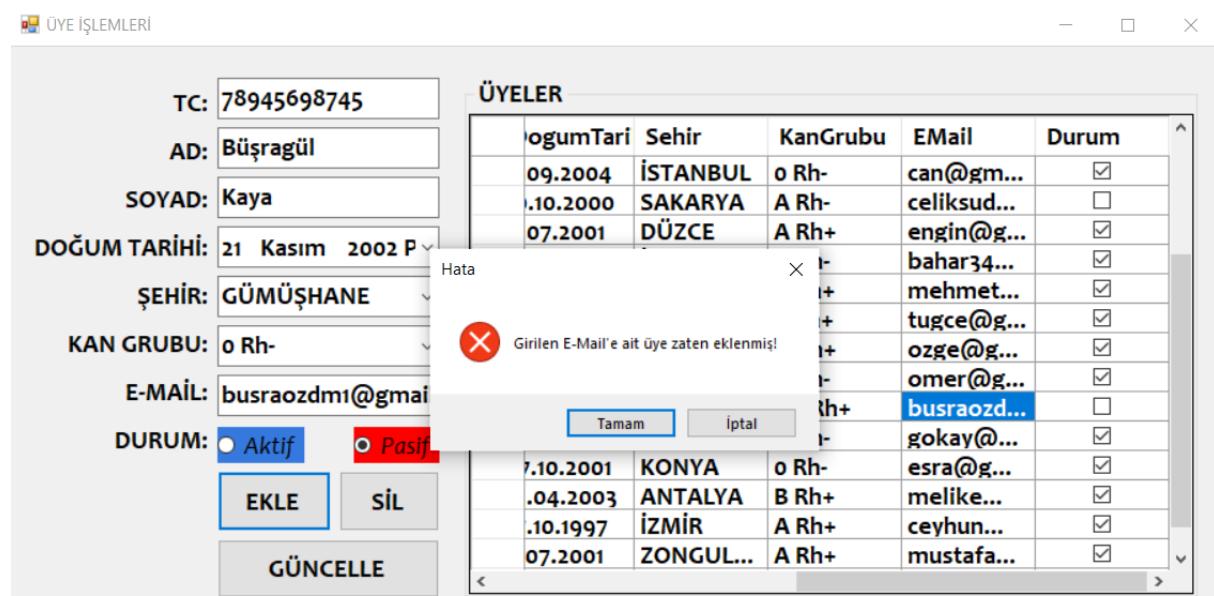
Şekil 6.4.4.3. Üye ekleme işlemi

Üye ekleme işlemi sırasında eğer önceden eklenmiş ve veri tabanında kayıtlı olan bir TC girilirse yetkili kullanıcıya mesaj ile uyarıda bulunarak o kaydın eklenmesi engellenmektedir.



Şekil 6.4.4.4. Aynı TC alanına sahip üye ekleme işlemi sırasında döndürülen hata mesajı

Veri tabanında kayıtlı bir e-mail ile üye eklenmeye çalışılırsa yetkiliye uyarıda bulunarak o kaydın eklenmesi engellenmektedir.



Şekil 6.4.4.5. Aynı E-Mail alanına sahip üye ekleme işlemi sırasında döndürülen hata mesajı

```

// Üye tablosu için oluşturulan mimariye erişebilmek adına nesne üretildi.
UyeManager uyeManager = new UyeManager(new UyeRepository());

// Veri tabanından direkt olarak erişmemiz gerekiği durumlarda nesne üretildi.
DernekTakipEntities db = new DernekTakipEntities();

// Ekleme işlemi için Üye tablosundan nesne üretildi
Uye üye = new Uye();

2 references
void Temizle() // Güncelleme ve silme işlemi sonrası ilgili alanlarda bulunan bilgilerin temizlenmesi için
{
    mskdTc.Text = " ";
    txtAd.Text = " ";
    txtSoyad.Text = " ";
    dateTimePicker1.Value= DateTime.Now;
    cmbSehir.Text = " ";
    cmbKanGrup.Text = " ";
    txtEmail.Text = " ";
    radioAktif.Checked = false;
    radioPasif.Checked = false;
}

4 references
void Liste() // Veri ekleme, silme, güncelleme işlemi sonrasında datagridview'daki bilgiler tekrardan listelensin
{
    dataGridViewUye.DataSource = uyeManager.GetList();
}

```

Şekil 6.4.4.6. Üye işlemleri için Temizle ve Liste metotları

```

1 reference
private void UyeIslemleri_Load(object sender, EventArgs e)
{
    Liste();

    illerManager illermanager = new illerManager(new illerRepository());

    var sehirler = illermanager.GetSehir();

    // Sayfa yüklenliğinde combobox'ta şehirler listelensin
    for (int i = 0; i < sehirler.Count; i++)
    {
        cmbSehir.Items.Add(sehirler[i]);
    }
}

```

Şekil 6.4.4.7. Sayfa yükleniği anda çalışacak olan kod bloğu

```

private void btnEkle_Click(object sender, EventArgs e)
{
    uye.TC = mskdTC.Text;
    uye.Ad = txtAd.Text;
    uye.Soyad = txtSoyad.Text;
    uye.DogumTarihi = dateTimePicker1.Value.Date;
    uye.Sehir = cmbSehir.Text;
    uye.KanGrubu = cmbKanGrup.Text;
    uye.EMail = txtEmail.Text;
    if (radioAktif.Checked)
    {
        uye.Durum = true;
    }
    else if(radioPasif.Checked)
    {
        uye.Durum = false;
    }

    var kisiTCVar = db.Uye.Where(x => x.TC == mskdTC.Text).FirstOrDefault(); // Girilen TC veri tabanında varsa o veriyi çek
    var kisiMailVar = db.Uye.Where(x => x.EMail== txtEmail.Text).FirstOrDefault();

    // kisiTCVar alanı null dönmüyorsa yani girilen TC veri tabanında kayıtlıysa çalışacak olan kod bloğu
    if (kisiTCVar != null)
        MessageBox.Show("Girilen TC kimlik numaralı üye zaten eklenmiş!", "Hata", MessageBoxButtons.OKCancel, MessageBoxIcon.Error);

    // kisiMailVar alanı null dönmüyorsa yani girilen E-Mail veri tabanında kayıtlıysa çalışacak olan kod bloğu
    else if (kisiMailVar != null)
        MessageBox.Show("Girilen E-Mail'e ait üye zaten eklenmiş!", "Hata", MessageBoxButtons.OKCancel, MessageBoxIcon.Error);

    else // Tüm işlemler doğru bir şekilde ilerlediyse ekleme işlemi yapılacak
    {
        uyeManager.TAdd(uye);
        Listele();
        MessageBox.Show("Başarıyla eklendi.", "Üye Ekleme", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }

    Temizle();
}

```

Şekil 6.4.4.8. Ekle butonuna basıldığında çalışacak olan kod bloğu

```

private void btnSil_Click(object sender, EventArgs e)
{
    // Seçilen kişinin bilgileri ilgili alanlara aktarıldığından TC alanında yer alan kişinin ID bilgisini veri tabanından çeker
    var kisiID = db.Uye.Where(x => x.TC.Equals(mskdTC.Text)).Select(y => y.ID).FirstOrDefault();

    var silinecekID = uyeManager.T GetById(kisiID); // Silinecek kişinin ID'sini Üye tablosunda arayarak o kişinin tüm bilgilerini çek
    uyeManager.TDelete(silinecekID);

    Listele();

    MessageBox.Show("Başarıyla silindi.", "Üye Silme", MessageBoxButtons.OK, MessageBoxIcon.Information);
}

```

Şekil 6.4.4.9. Sil butonuna basıldığında çalışacak olan kod bloğu

```

private void btnGuncelle_Click(object sender, EventArgs e)
{
    // Seçilen kişinin bilgileri ilgili alanlara aktarıldığından TC alanında yer alan kişinin ID bilgisini veri tabanından çeker.
    var kisiID = db.Uye.Where(x => x.TC.Equals(mskdTC.Text)).Select(y => y.ID).FirstOrDefault();

    // Güncelleme yapılacak kişinin ID'sini Üye tablosunda arayarak o kişinin tüm bilgilerini çek.
    var guncelleme = uyeManager.T GetById(kisiID);

    guncelleme.TC = mskdTC.Text;
    guncelleme.Ad = txtAd.Text;
    guncelleme.Soyad = txtSoyad.Text;
    guncelleme.DogumTarihi = dateTimePicker1.Value.Date;
    guncelleme.Sehir = cmbSehir.Text;
    guncelleme.KanGrubu = cmbKanGrup.Text;
    guncelleme.EMail = txtEmail.Text;
    if (radioAktif.Checked)
        guncelleme.Durum = true;
    else if (radioPasif.Checked)
        guncelleme.Durum = false;

    uyeManager.TUpdate(guncelleme);

    Listele();
    Temizle();
    MessageBox.Show("Başarıyla güncellendi.", "Üye Güncelleme", MessageBoxButtons.OK, MessageBoxIcon.Information);
}

```

Şekil 6.4.4.10. Güncelle butonuna basıldığında çalışacak olan kod bloğu

DataGridView içerisinde listelenen kayıtlar içerisindeki seçilen bir kişinin bilgilerini ilgili alanlara aktarılmasını sağlayacak olan kod bloğu aşağıda verilmektedir.

```
1 reference
private void dataGridUye_CellClick(object sender, DataGridViewCellEventArgs e)
{
    int secilen = dataGridUye.SelectedCells[0].RowIndex;
    mskdTC.Text = dataGridUye.Rows[secilen].Cells[1].Value.ToString();
    txtAd.Text = dataGridUye.Rows[secilen].Cells[2].Value.ToString();
    txtSoyad.Text = dataGridUye.Rows[secilen].Cells[3].Value.ToString();
    dateTimePicker1.Value = Convert.ToDateTime(dataGridUye.Rows[secilen].Cells[4].Value);
    cmbSehir.Text = dataGridUye.Rows[secilen].Cells[5].Value.ToString();
    cmbKanGrup.Text = dataGridUye.Rows[secilen].Cells[6].Value.ToString();
    txtEmail.Text = dataGridUye.Rows[secilen].Cells[7].Value.ToString();
    if (dataGridUye.Rows[secilen].Cells[8].Value.ToString() == "True") // seçilen kaydın Durum bilgisi True ise
    {
        radioAktif.Checked = true; // Aktif olan radiobuttonu seçili hale getir.
    }
    else if (dataGridUye.Rows[secilen].Cells[8].Value.ToString() == "False") // seçilen kaydın Durum bilgisi False ise
    {
        radioPasif.Checked = true; // Pasif olan radiobuttonu seçili hale getir.
    }
}
```

Şekil 6.4.4.11: DataGridView CellClick özelliği ile seçilen kaydın bilgilerini alanlara aktarma

6.4.5 Aidat İşlemleri Ekranı

Aidat işlemleri ekranı ile yetkili dernekte bulunan üyeleri için her ay aidat ödeyip ödememe durumunu kaydedebilir. Aidatı ödemeyen yani aidat kaydı pasif olan üye, daha sonradan aidat borcunu ödemesi durumunda yetkili, üyenin o ayki aidat kaydını aktif olarak güncelleyebilmektedir.

Herhangi bir üye için aidat kaydı eklemeye işlemi sırasında üyenin TC'si manuel olarak ilgili textbox'a girilmelidir. Ya da pratik olması açısından ekranda yer alan üye listesinden hangi üye için aidat kaydı eklenecekse o üye seçilmelidir. Bu işlem ile arka planda seçilen üyenin TC'si otomatik olarak ilgili textbox'a aktarılması sağlanmaktadır. Ardından aidat kaydı eklenmek istenen ay bilgisi ilgili combobox'tan seçilir. Miktar kısmında aidat kaydı yapılacak olan ayın yani combobox'tan seçilen ay bilgisinin o ay için belirlenen ücret tutarı otomatik olarak gösterilmektedir. Bu veri ilgili tablodan otomatik olarak çekildiği için değiştirilmeye izin verilmemektedir. Her ay için belirlenen ücret tutarı Aidat Düzenleme Ekranı üzerinden güncellenebilmektedir. Ödeme tarihi ve aidat ödeme durumu da seçildikten sonra ilgili kaydın eklemesi yapılmaktadır.

Aşağıda ise bir üye için eklenen aidat kaydı sonrasında verilen bilgi mesajı gösterilmektedir.

AİDAT İŞLEMLERİ

AİDAT LİSTESİ

ID	Miktar	OdemeTari	Durum	UserId
1	1150,00	3.11.2023	<input type="checkbox"/>	1
2	1150,00	15.11.2023	<input checked="" type="checkbox"/>	7
3	1150,00	11.11.2023	<input checked="" type="checkbox"/>	10
4	1150,00	27.11.2023	<input checked="" type="checkbox"/>	12
5	1150,00	14.11.2023	<input checked="" type="checkbox"/>	21
6	1150,00	30.11.2023	<input type="checkbox"/>	8
7	1150,00	29.11.2023	<input type="checkbox"/>	3
8	1150,00	16.11.2023	<input checked="" type="checkbox"/>	24
9	1150,00	1.11.2023	<input checked="" type="checkbox"/>	23
10	1150,00	13.11.2023	<input checked="" type="checkbox"/>	22
11	1150,00	9.11.2023	<input checked="" type="checkbox"/>	6

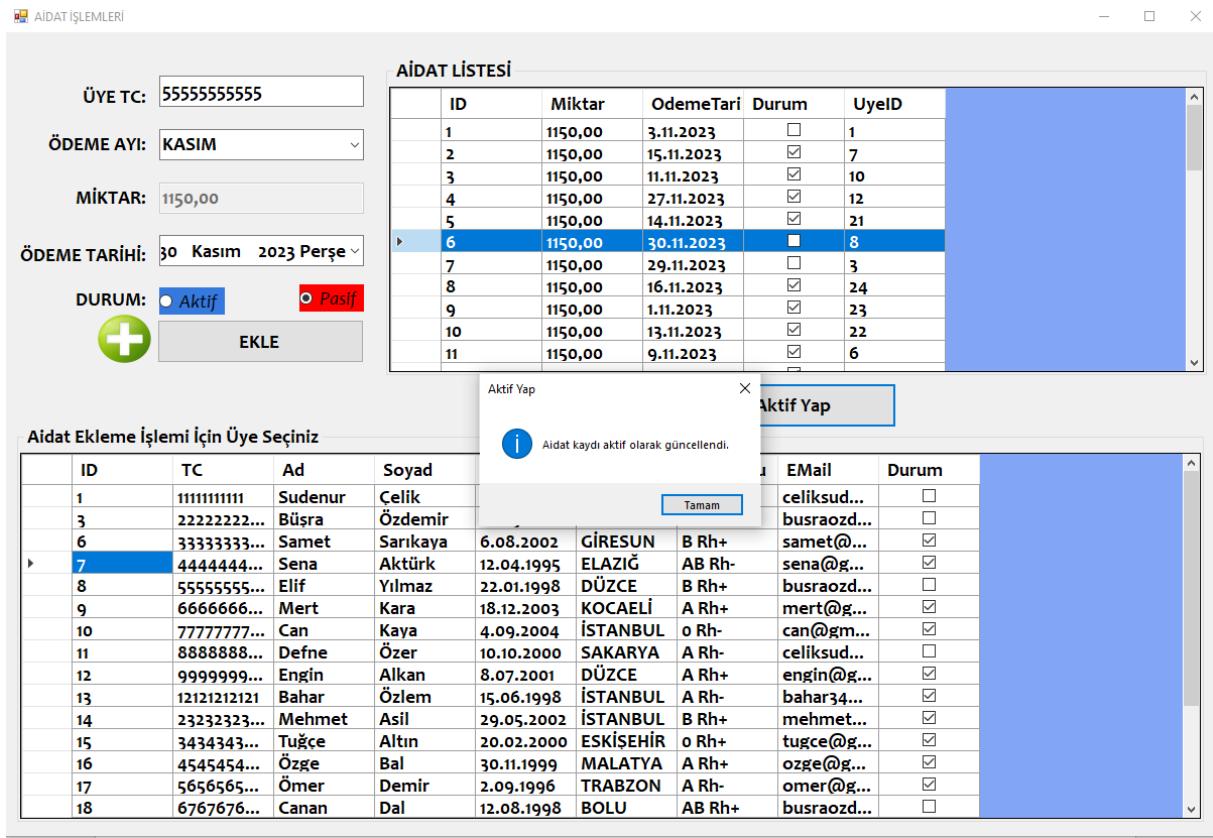
Aidat Bilgisi Ekleme × **Aktif Yap**

Başarılı eklendi.

ID	TC	Ad	Soyad	Dogum	İGrubu	EMail	Durum
1	1111111111	Sudenur	Çelik	14.03.1990	<input type="checkbox"/>	celiksud...	<input type="checkbox"/>
3	22222222...	Büşra	Özdemir	16.05.1990	<input type="checkbox"/>	busraozd...	<input type="checkbox"/>
6	33333333...	Samet	Sarıkaya	6.08.2002	<input type="checkbox"/>	GİRESUN	<input checked="" type="checkbox"/>
7	4444444...	Sena	Aktürk	12.04.1995	<input type="checkbox"/>	ELAZIĞ	<input checked="" type="checkbox"/>
8	55555555...	Elif	Yılmaz	22.01.1998	<input type="checkbox"/>	DÜZCE	<input checked="" type="checkbox"/>
9	6666666...	Mert	Kara	18.12.2003	<input type="checkbox"/>	KOCAELİ	<input checked="" type="checkbox"/>
10	77777777...	Can	Kaya	4.09.2004	<input type="checkbox"/>	İSTANBUL	<input checked="" type="checkbox"/>
11	8888888...	Defne	Özer	10.10.2000	<input type="checkbox"/>	SAKARYA	<input checked="" type="checkbox"/>
12	9999999...	Engin	Alkan	8.07.2001	<input type="checkbox"/>	DÜZCE	<input checked="" type="checkbox"/>
13	12121212121	Bahar	Özlem	15.06.1998	<input type="checkbox"/>	İSTANBUL	<input checked="" type="checkbox"/>
14	23232323...	Mehmet	Asıl	29.05.2002	<input type="checkbox"/>	İSTANBUL	<input checked="" type="checkbox"/>
15	3434343...	Tüngle	Altın	20.02.2000	<input type="checkbox"/>	ESKİSEHIR	<input checked="" type="checkbox"/>
16	4545454...	Özge	Bal	30.11.1999	<input type="checkbox"/>	MALATYA	<input checked="" type="checkbox"/>
17	5656565...	Ömer	Demir	2.09.1996	<input type="checkbox"/>	TRABZON	<input checked="" type="checkbox"/>
18	6767676...	Canan	Dal	12.08.1998	<input type="checkbox"/>	BOLU	<input checked="" type="checkbox"/>

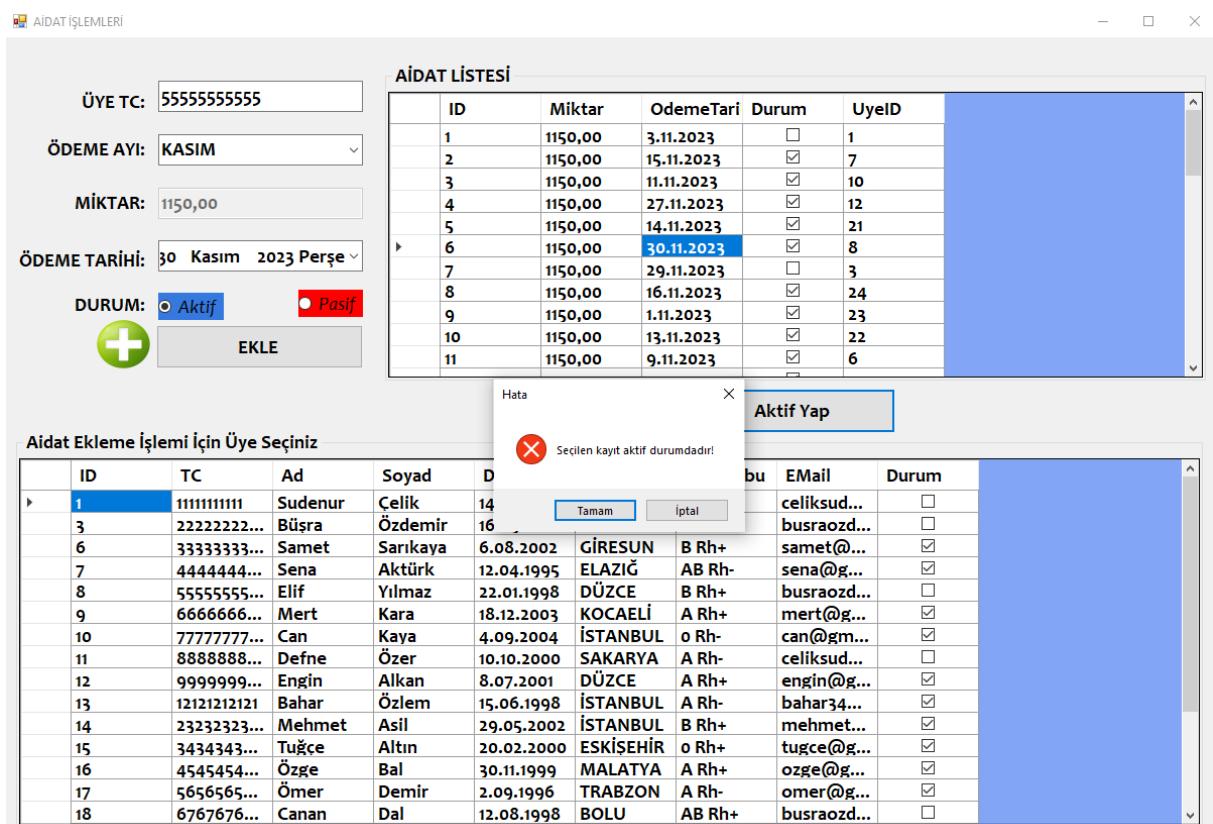
Şekil 6.4.5.1. Aidat kaydı ekleme işlemi

Üye'nin ödediği aidatlar her ay için ayrı ayrı Aidat tablosunda tutulmaktadır. Herhangi bir ay için aidat ücretini ödediği durumda bu kayıt yetkili kişi tarafından pasif olarak veri tabanına kaydedilmektedir. Eğer üye, pasif olan aidat borcunu daha sonraki zamanlarda ödemisse o kayıt aktif hale getirilmesi gereklidir. Bu işlem için de aşağıda bulunan aidat listesinden durumu aktif yapmak istenen kayıt seçilerek aktif yap butonu tıklanmaktadır ve sonrasında ilgili bilgi mesajı otomatik olarak kullanıcıya bildirilmektedir.



Şekil 6.4.5.2. Aidat kaydı durumunu aktif yapma

Eğer seçilen kayıt zaten aktif haldeyse aşağıdaki gibi bir uyarı mesajı verilecektir.



Şekil 6.4.5.3. Aidat kaydı aktif yapma işlemindeki uyarı mesajı

```

// Aidat tablosu için oluşturulan mimariye erişebilmek adına nesne üretildi.
AidatManager aidatManager = new AidatManager(new AidatRepository());

// Veri tabanından direkt olarak erişmemiz gerektiği durumlarda nesne üretildi.
DernekTakipEntities db = new DernekTakipEntities();

// Ekleme işlemi için Uye tablosundan nesne üretildi
Aidat aidat = new Aidat();
2 references
void Temizle() // Güncelleme ve silme işlemi sonrası ilgili alanlarda bulunan bilgilerin temizlenmesi için
{
    msktxtTc.Text = " ";
    cmbAylar.Text = "";
    txtMiktar.Text = " ";
    rdbtnAktif.Checked = false;
    rdbtnPasif.Checked = false;
    dateTimePicker1.Value = DateTime.Now;
}

3 references
void Listele() // Veri ekleme, silme, güncelleme işlemi sonrasında datagridview'daki bilgiler tekrardan listelensin
{
    dataGridViewAidat.DataSource = aidatManager.GetList();

    // Datagrid'de listelenen alanlar içerisinde gözükmemesini istemediğimiz alanların görünümü false yapıldı.
    dataGridViewAidat.Columns["Uye"].Visible = false;
}

```

Şekil 6.4.5.4. Temizle ve Listele metodları

```

1 reference
private void AidatIslemleri_Load(object sender, EventArgs e)
{
    Listele();

    // Sayfa yüklenliğinde aylar combobox'a gelmesi için
    AylarManager aylarManager = new AylarManager(new AylarRepository());

    var aylar = aylarManager.GetAylar();

    for (int i = 0; i < aylar.Count; i++)
    {
        cmbAylar.Items.Add(aylar[i]);
    }

    // Aidat bilgisi için listelenen üyeleri arasından seçim yaparak ekleme yapılabilir.
    UyeManager uyeManager = new UyeManager(new UyeRepository());
    dataGridViewUye.DataSource = uyeManager.GetList();
}

```

Şekil 6.4.5.5. Sayfa yüklenliğinde yapılması gereken işlemlerin kodu

```

1 reference
private void btnEkle_Click(object sender, EventArgs e)
{
    var uyeTC = msktxtTc.Text;

    // Girilen TC'ye ait kişinin ID bilgisini çek
    var uyeID = db.Uye.Where(x => x.TC.Equals(uyeTC)).Select(y => y.ID).FirstOrDefault();

    // Eğer girilen TC veri tabanında yoksa uyeID 0 dönecektir ve 0 dönüyorsa da ekleme işlemine izin verilmemektedir.
    if (uyeID == 0)
    {
        MessageBox.Show("Girilen TC kimlik numaralı üye veri tabanında bulunmamaktadır!", "Hata",
                        MessageBoxButtons.OKCancel, MessageBoxIcon.Error);
        Temizle();
    }
    else
    {
        aidat.UyeID = uyeID;
        aidat.OdemeTarihi = dateTimePicker1.Value.Date;
        aidat.Miktar = Convert.ToDecimal(txtMiktar.Text);
        if (rdbtnAktif.Checked) // Aktif adlı buton seçiliyse
        {
            aidat.Durum = true; // Durum alanına true değerini kaydet
        }
        else // Pasif adlı buton seçiliyse
        {
            aidat.Durum = false; // Durum alanına false değerini kaydet
        }

        aidatManager.TAdd(aidat);

        Listele();

        Temizle();

        MessageBox.Show("Başarıyla eklendi.", "Aidat Bilgisi Ekleme",
                        MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}

```

Şekil 6.4.5.6. Aidat ekleme işlemi sırasında çalışan kod bloğu

```

1 reference
private void cmbAylar_SelectedIndexChanged(object sender, EventArgs e)
{
    // Combobox'ta listelenen aylar arasında hangisi seçildiyse o aya ait aidat miktarının değeri+
    //txtMiktar adlı textbox'a otomatik getirilecektir.

    string ayAdi = cmbAylar.Text;
    var miktar = db.Aylar.Where(x => x.AyAdi == ayAdi).Select(y => y.Tutar).FirstOrDefault();
    txtMiktar.Text = miktar.ToString();
}

```

Şekil 6.4.5.7. Ay bilgisi seçildikten sonra çalışması gereken kod bloğu

```

1 reference
private void dataGridAidat_CellClick(object sender, DataGridViewCellEventArgs e)
{
    // Datagrid'de listelenen alanlar içerisinde gözükmeyi istemediğimiz alanların görünümü false yapıldı.
    dataGridAidat.Columns["Uye"].Visible = false;

    int secilen = dataGridAidat.SelectedCells[0].RowIndex;
    txtMiktar.Text = dataGridAidat.Rows[secilen].Cells[1].Value.ToString();
    datepicker1.Value = Convert.ToDateTime(dataGridAidat.Rows[secilen].Cells[2].Value);

    // Listelenen Aidat kayıtları arasından seçilen kaydın Ay bilgisini combobox'da gösterme
    var ay = Convert.ToDateTime(dataGridAidat.Rows[secilen].Cells[2].Value);
    int ayNo = ay.Month;
    var ayAdi = db.Aylar.Where(x => x.ID == ayNo).Select(y => y.AyAdi).FirstOrDefault();
    cmbAylar.Text = ayAdi.ToString();

    // Listelenen Aidat kayıtları arasından seçilen kaydın TC bilgisini TC için ayrılmış alanda gösterme
    var uyeID = Convert.ToInt32(dataGridAidat.Rows[secilen].Cells[4].Value);
    var kisiTC = db.Uye.Where(x => x.ID.Equals(uyeID)).Select(y => y.TC).FirstOrDefault();
    msktxtTc.Text = kisiTC.ToString();

    if (dataGridAidat.Rows[secilen].Cells[3].Value.ToString() == "True")
    {
        rdbtnAktif.Checked = true;
    }
    else if (dataGridAidat.Rows[secilen].Cells[3].Value.ToString() == "False")
    {
        rdbtnPasif.Checked = true;
    }
}

```

Şekil 6.4.5.8. Aidat listeleme işlemi sırasında CellClick özelliğinin kodları

```

1 reference
private void dataGridUye_CellClick(object sender, DataGridViewCellEventArgs e)
{
    int secilen = dataGridUye.SelectedCells[0].RowIndex;
    msktxtTc.Text = dataGridUye.Rows[secilen].Cells[1].Value.ToString();
}

```

Şekil 6.4.5.9. Üye listeleme işlemi sırasında CellClick özelliğinin kodları

```

1 reference
private void btnAktif_Click(object sender, EventArgs e)
{
    var aidatID = dataGridAidat.SelectedCells[0].RowIndex;

    var aidat = aidatManager.T GetById(aidatID+1); // Seçilen kaydın indexi 0'dan başladığı için +1 eklendi.

    if (!aidat.Durum) // Aidat durumu false ise çalışacak
    {
        aidat.Durum = true;
        aidatManager.TUpdate(aidat);

        MessageBox.Show("Aidat kaydı aktif olarak güncellendi.", "Aktif Yap", MessageBoxButtons.OK, MessageBoxIcon.Information);
        Listele();
    }
    else
    {
        MessageBox.Show("Seçilen kayıt aktif durumdadır!", "Hata", MessageBoxButtons.OKCancel, MessageBoxIcon.Error);
    }
}

```

Şekil 6.4.5.10. Aktif Yap butonuna basıldığında çalışacak olan kod bloğu

6.4.6 Aidat Düzenleme Ekranı

Aidat düzenleme ekranı sayesinde yetkili kullanıcı aylık aidat bilgisini güncelleyebilir.

Her ay için farklı aidat miktarı belirlenmişse ya da aidat miktarı arttırılmak, azaltılmak

istenmesi durumundaki işlemler bu ekran vasıtasıyla gerçekleştirilebilir.

The screenshot shows a Windows application window titled "AİDAT DÜZENLE". On the left, there are two input fields: "AY SEÇİNİZ" (Month Selection) with a dropdown menu showing "OCAK" as the selected item, and "TUTAR" (Amount) with a text input field containing "250,00". Below these is a button labeled "Ücret Güncelle" (Update Amount). To the right is a data grid table with columns "ID", "AyAdı" (Month), and "Tutar" (Amount). The table contains 12 rows, each representing a month from OCAK to EYLÜL with its corresponding value. The first row (OCAK) is highlighted with a blue background.

ID	AyAdı	Tutar
1	OCAK	250,00
2	ŞUBAT	200,00
3	MART	300,00
4	NİSAN	400,00
5	MAYIS	500,00
6	HAZİRAN	600,00
7	TEMMUZ	700,00
8	AĞUSTOS	800,00
9	EYLÜL	900,00

Şekil 6.4.6.1. Aidat Düzenleme Ekranı

Listelenen Aidat değerleri ve ay bilgisi arasından güncelleme yapılmak istenen kayıt seçilerek değerleri ilgili alana otomatik gelmesi sağlanabilir. Ya da combobox'tan ay seçilipt tutar bilgisi de yazılarak manuel bir şekilde kayıt güncelleme işlemi yapılabilir.

This screenshot shows the same application window after an update. The "AY SEÇİNİZ" dropdown now shows "MAYIS". The "TUTAR" input field still contains "550,00". The "Ücret Güncelle" button is highlighted with a blue border. A message box in the center says "Aidat Ücret Güncelleme" and "Başarıyla güncellendi." (Updated successfully). Below the message are "Tamam" (OK) and "Kapat" (Close) buttons. The data grid table on the right remains the same as in the previous screenshot.

Şekil 6.4.6.2. Aidat ücreti güncellendikten sonra ekrana gelen mesaj

```

// Aylar tablosu için oluşturulan mimariye erişebilmek adına nesne üretildi.
AylarManager aylarManager = new AylarManager(new AylarRepository());

// Veri tabanından direkt olarak erişmemiz gerektiği durumlarda nesne üretildi.
DernekTakipEntities db = new DernekTakipEntities();

1 reference
void Temizle() // Güncelleme işlemi sonrası ilgili alanlarda bulunan bilgilerin temizlenmesi için
{
    cmbAylar.Text = "";
    txtTutar.Text = "";
}

1 reference
private void AidatDuzenle_Load(object sender, EventArgs e)
{
    dataGridAidat.DataSource = aylarManager.GetList();

    // Sayfa yüklenliğinde aylar combobox'a gelmesi için
    var aylar = aylarManager.GetAylar();

    for (int i = 0; i < aylar.Count; i++)
    {
        cmbAylar.Items.Add(aylar[i]);
    }
}

```

Şekil 6.4.6.3. Temizle metodu ve sayfa yüklenliğinde yapılması gereken işlemlerin kodu

```

1 reference
private void dataGridAidat_CellClick(object sender, DataGridViewCellEventArgs e)
{
    int secilen = dataGridAidat.SelectedCells[0].RowIndex;
    cmbAylar.Text = dataGridAidat.Rows[secilen].Cells[1].Value.ToString();
    txtTutar.Text = dataGridAidat.Rows[secilen].Cells[2].Value.ToString();
}

```

Şekil 6.4.6.4. Listelenen aidat ve ay kayıtlarından herhangi birinin seçildiğinde ilgili alanlara aktarım yapan kod

```

1 reference
private void btnGuncelle_Click(object sender, EventArgs e)
{
    var secilen = cmbAylar.SelectedIndex; // Combobox'tan seçilen ay isminin index değeri secilen'e atandı.

    var guncelleme = aylarManager.T GetById(secilen + 1); // Combobox'tan seçilen index 0'dan başladığı için +1 eklenerek verinin getirilmesi.

    guncelleme.AyAdi = cmbAylar.Text;
    guncelleme.Tutar = Convert.ToDecimal(txtTutar.Text);

    aylarManager.TUpdate(guncelleme);

    dataGridAidat.DataSource = aylarManager.GetList();

    MessageBox.Show("Başarıyla güncellendi.", "Aidat Ücret Güncelleme", MessageBoxButtons.OK, MessageBoxIcon.Information);

    Temizle();
}

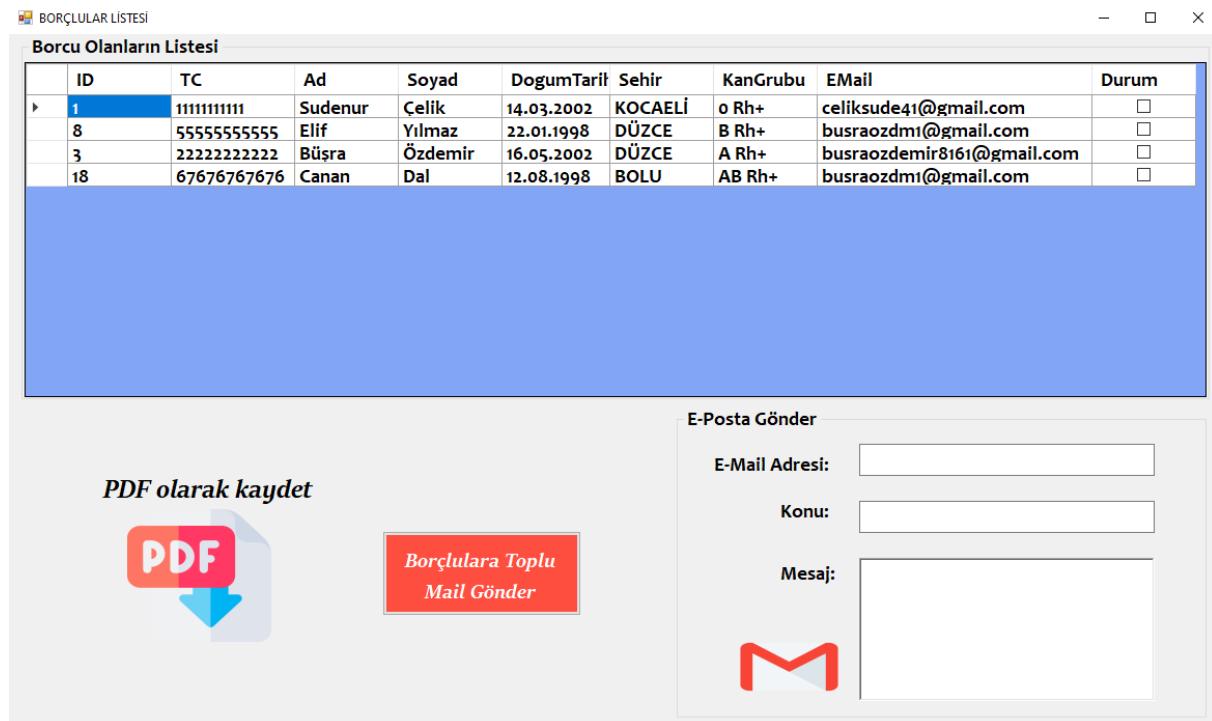
```

Şekil 6.4.6.5. Ücret Güncelle butonuna tıklandığında çalışacak olan kod bloğu

6.4.7 Borçlular Listesi Ekranı

Borçlular listesi ekranı ile yetkili kişiler borcu olanları liste halinde görebilir ve dilerse bu listeyi pdf olarak cihazına indirebilmektedir. Borcu olan üyelere bir butona basılarak otomatik olarak mail gönderme işlemi yapılmaktadır. Herhangi bir üyeye ayrı olarak yani o üyeye özel mail gönderilmek istenirse mail adresi ve mesaj içeriği ilgili alanlar vasıtasiyla

istenildiği gibi değiştirilip gönderilebilmektedir.



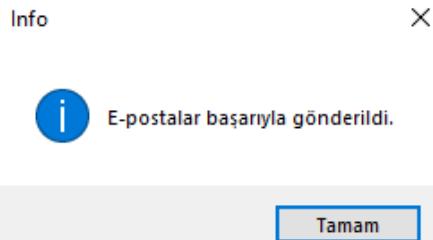
Şekil 6.4.7.1. Borçlular listesi ekranı

Borcu olan üyeleri PDF olarak kaydet butonuna basarak pdf halinde kaydetme işlemi sağlanabilir. Kaydedildiği takdirde başarılı olarak kayıt olduğuna dair mesaj verilir. Eğer PDF'i kaydederken hata oluştussa da ilgili hata yetkili kullanıcıya gösterilmektedir.

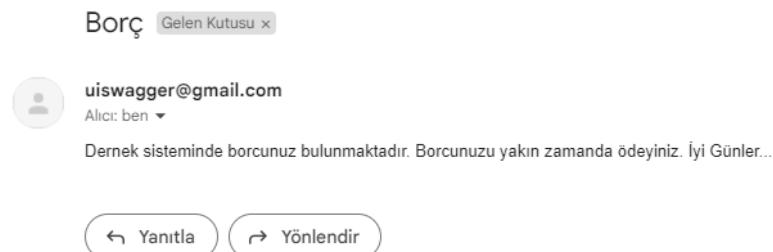
ID	TC	Ad	Soyad	DogumTari̇t	Sehir	KanGrubu	EMail	Durum
1	1111111111 111	Sudenur	Celik	14.03.2002 20:00:00:00	KOCAEL	O Rh+	celiksude41@gmail.com	False
8	5555555555 555	Elif	Yılmaz	22.01.1998 00:00:00:00	DÜZCE	B Rh+	busraozdm1@gmail.com	False
3	2222222222 222	Büşra	Özdemir	16.05.2002 20:00:00:00	DÜZCE	A Rh+	busraozdemir8161@gmail.com	False
18	67676767676 676	Canan	Dal	12.08.1998 00:00:00:00	BOLU	AB Rh+	busraozdm1@gmail.com	False

Şekil 6.4.7.2. Borçlular listesi'nin PDF olarak kaydedildikten sonraki PDF içeriği

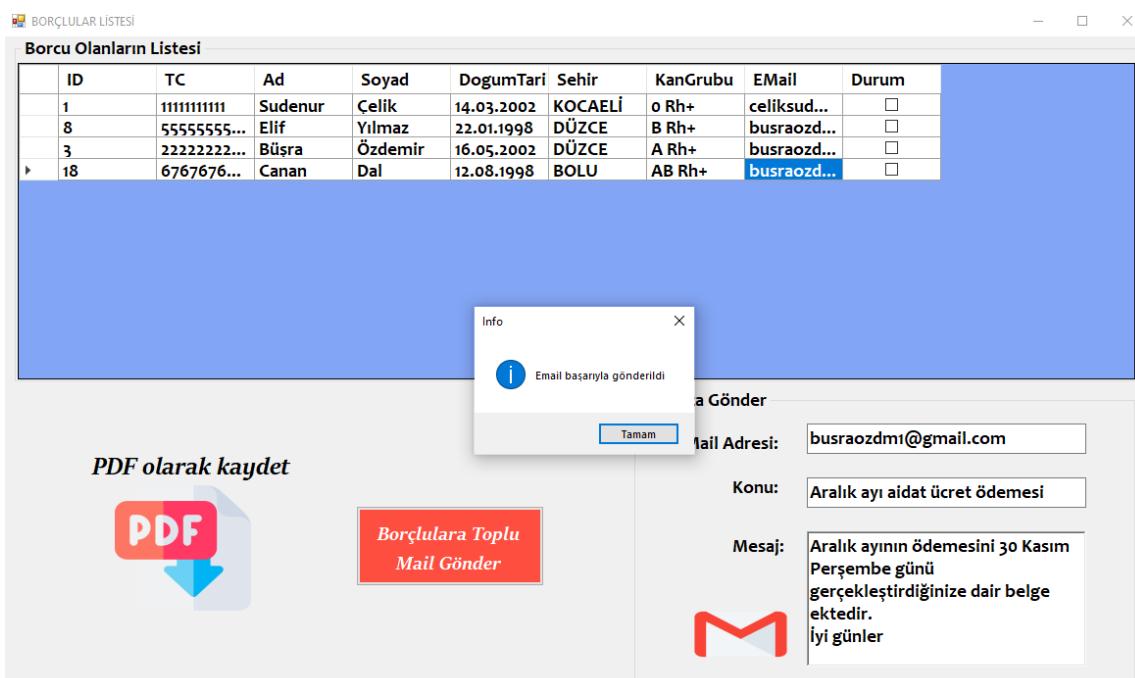
Borçlulara Toplu Mail Gönder butonuna tıklandığında listede olan tüm borçlulara otomatik mail gitmektedir. Mailler başarılı bir şekilde iletilince ekrana yetkili bilgilendiren mesaj da verilmektedir.



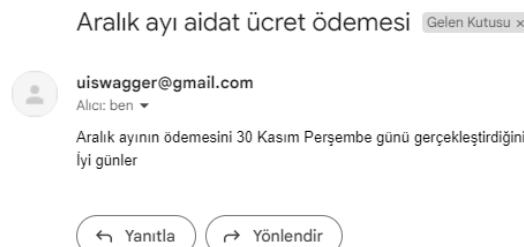
Şekil 6.4.7.3. Toplu mail gönderildikten sonra gelen bilgi mesajı



Şekil 6.4.7.4. Toplu mail ile hesaba gelen otomatik mail içeriği



Şekil 6.4.7.5. Kişiye özel mail gönderme



Şekil 6.4.7.6. Mail hesabına gelen mesaj

```

// Veri tabanından direkt olarak erişmemiz gerektiği durumlarda nesne üretildi.
DernekTakipEntities db = new DernekTakipEntities();

1 reference
private void Borclular_Load(object sender, EventArgs e)
{
    // Aidat tablosunda ödeme durumu false olan üyelerin bilgilerini sayfa yüklenliğinde listeleme işlemi

    var uyeler = db.Aidat.Where(x => x.Durum == false).Select(y => y.UyeID).ToList();

    List<Uye> borclular = new List<Uye>();

    foreach (int uyeID in uyeler)
    {
        var borclu = db.Uye.Where(x => x.ID == uyeID).FirstOrDefault();

        if (borclu != null)
        {
            borclular.Add(borclu);
        }
    }

    dtgrBorclular.DataSource = borclular;
}

```

Şekil 6.4.7.7. Sayfa yüklenliğinde borçlu olan üyeleri listelemek için olan kod blogu

```

private void pctrbxPdfKaydet_Click(object sender, EventArgs e)
{
    if (dtgrBorclular.Rows.Count > 0)
    {
        SaveFileDialog sfd = new SaveFileDialog(); // Dosyayı kaydetmek için iletişim kutusu açabilmek adına nesne oluşturuldu.
        sfd.Filter = "PDF (*.pdf)|*.pdf"; // Dosyanın formatı (pdf)
        sfd.FileName = "BorclularListesi.pdf"; // Dosyanın adı
        bool fileError = false;
        if (sfd.ShowDialog() == DialogResult.OK)
        {
            if (File.Exists(sfd.FileName))
            {
                try
                {
                    File.Delete(sfd.FileName);
                }
                catch (IOException ex)
                {
                    fileError = true;
                    MessageBox.Show("Hata Oluştı. Tekrar Deneyiniz." + ex.Message, "Hata", MessageBoxButtons.OKCancel, MessageBoxIcon.Error);
                }
            }
            if (!fileError)
            {
                try
                {
                    PdfPTable pdfTable = new PdfPTable(dtgrBorclular.Columns.Count);
                    pdfTable.DefaultCell.Padding = 3;
                    pdfTable.WidthPercentage = 100;
                    pdfTable.HorizontalAlignment = Element.ALIGN_LEFT;

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

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

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

                    MessageBox.Show("İşlem Başarılı.", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Hata:" + ex.Message, "Hata", MessageBoxButtons.OKCancel, MessageBoxIcon.Error);
                }
            }
        }
    }
}

```

Şekil 6.4.7.8. PDF olarak kaydet butonuna tıklandığında çalışacak kodlar

```

private void btnTopluMail_Click(object sender, EventArgs e)
{
    try
    {
        // E-posta gönderen hesap bilgileri
        string senderEmail = "uiswagger@gmail.com";
        string senderPassword = "uzypqmelvqfrisiu";

        // Alıcı e-posta adresleri (toplu mail gönderileceği için List formatında tutuldu)
        List<string> recipientEmails = new List<string>();

        // Belirli sütunun indeksini bul (maillerin yer aldığı sütun)
        int columnIndex = dtgrBorcular.Columns[7].Index;

        // Her bir satırdaki belirli hücrenin değerini recipientEmails listesine ekle
        foreach (DataGridViewRow row in dtgrBorcular.Rows)
        {
            object cellValue = row.Cells[columnIndex].Value;
            if (cellValue != null)
            {
                recipientEmails.Add(cellValue.ToString());
            }
        }

        // E-posta başlığı ve içeriği
        string subject = "Borç";
        string body = "Dernek sisteminde borcunuz bulunmaktadır. Borcunuza yakın zamanda ödeyiniz. İyi Günler...";

        // SMTP sunucusu ve port bilgileri
        string smtpServer = "smtp.gmail.com";
        int smtpPort = 587;

        // E-posta gönderme işlemi
        using (SmtpClient smtpClient = new SmtpClient(smtpServer, smtpPort))
        {
            smtpClient.Credentials = new NetworkCredential(senderEmail, senderPassword);
            smtpClient.EnableSsl = true;

            foreach (var recipientEmail in recipientEmails)
            {
                using (MailMessage mailMessage = new MailMessage(senderEmail, recipientEmail, subject, body))
                {
                    smtpClient.Send(mailMessage);
                }
            }
            MessageBox.Show("E-postalar başarıyla gönderildi.", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Hata: {ex.Message}", "Hata", MessageBoxButtons.OKCancel, MessageBoxIcon.Error);
    }
}

```

Şekil 6.4.7.9. Toplu mail gönderme işleminde çalışacak olan kodlar

```

1 reference
private void pctrbxGonder_Click(object sender, EventArgs e)
{
    try
    {
        MailMessage mail = new MailMessage();
        SmtpClient SmtpServer = new SmtpClient();

        mail.From = new MailAddress("uiswagger@gmail.com"); // Mail uiswagger@gmail.com adlı hesaptan gönderilecek
        mail.To.Add(txtEmail.Text.ToString());
        mail.Subject = txtKonu.Text.ToString();
        mail.Body = txtMesaj.Text.ToString();

        // Mail'in gönderileceği mail hesabı belirtildi ve şifresi hashlenerek tutuldu
        SmtpServer.Credentials = new System.Net.NetworkCredential("uiswagger@gmail.com", "uzypqmelvqfrisiu");

        SmtpServer.Host = "smtp.gmail.com";
        SmtpServer.EnableSsl = true;
        SmtpServer.Port = 587;
        SmtpServer.Send(mail);
        MessageBox.Show("Email başarıyla gönderildi", "Info", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Hata: {ex.Message}", "Hata", MessageBoxButtons.OKCancel, MessageBoxIcon.Error);
    }
}

```

Şekil 6.4.7.10. Kişiye özel mail gönderme işleminde çalışacak olan kodlar

```

1 reference
private void dtgrBorcular_CellClick(object sender, DataGridViewCellEventArgs e)
{
    int secilen = dtgrBorcular.SelectedCells[0].RowIndex;
    // Listededen seçilen kişi mail bilgisinin Email textbox'ına otomatik gelmesi için
    txtEmail.Text = dtgrBorcular.Rows[secilen].Cells[7].Value.ToString();
}

```

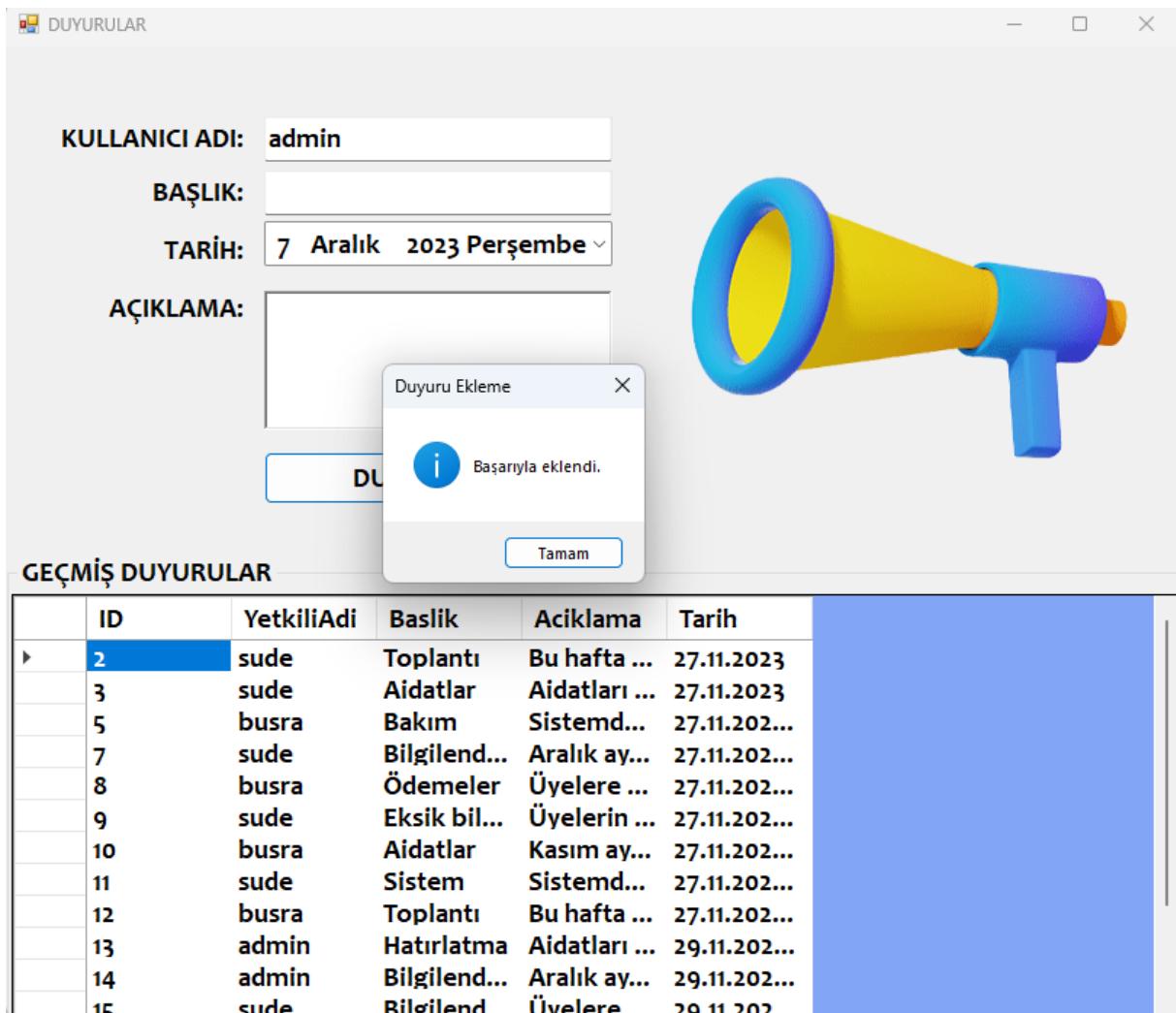
Şekil 6.4.7.11. Seçilen kişi mail bilgisinin textbox'a gelmesi için kullanılan kod

6.4.8 Duyuru Ekranı

Duyuru ekranı sayfası ile bir yetkilinin diğer yetkililer için bildirmek istediği mesajı burada paylaşılmasına olanak sağlar. Bu sayfada bulunan DataGridView sayesinde bütün yetkililerin eklediği duyurular liste halinde gösterilmektedir. Bu listede hangi duyuruyu hangi yetkilin yaptığı ve duyurunun yapılmama tarihi de belirtilmektedir.

ID	YetkiliAdı	Baslik	Açıklama	Tarih
2	sude	ghhh	ıgvıghv	27.11.2023
3	sude	jhvgigf	ıghıgh	27.11.2023
5	busra	ıgfıhg	jhgjhg h ...	27.11.202...
7	sude	gfdhfd	fhgfdhgf	27.11.202...
8	busra	aaaa	gdsgdsg...	27.11.202...
9	sude	merhaba	merhaba...	27.11.202...
10	busra	agdhadsgf	dhgdsjfhgs	27.11.202...
11	sude	fdjhf	sfgdgnrk	27.11.202...
12	busra	gdfs	aaaaaaaa...	27.11.202...
13	admin	merhaba...	fdsafdsf...	29.11.202...
14	admin	aaaaaa	bbbbbb...	29.11.202...
15	sude	ooooooooo	vvvvvvvv	29.11.202...

Şekil 6.4.8.1. Duyuru ekranı



Şekil 6.4.8.2. Yeni duyuru ekleme ile verilen bilgilendirme mesajı

```

private void btnEkle_Click(object sender, EventArgs e)
{
    // Girilen kullanıcı adı Yetkili tablosunda bulunan kullanıcı adlarından biriyse o kişinin ID'sini al.
    var yetkiliID = db.Yetkili.Where(x => x.KullaniciAd.Equals(txtKullaniciAd.Text)).Select(y => y.ID).FirstOrDefault();

    // Girilen kullanıcı adı Yetkili tablosunda bulunan kullanıcı adlarından biriyse o kişinin kullanıcı adını al.
    var yetkiliAdı = db.Yetkili.Where(x => x.KullaniciAd.Equals(txtKullaniciAd.Text)).Select(y => y.KullaniciAd).FirstOrDefault();

    if (yetkiliAdı != null) // Kullanıcı adı veritabanında varsa yani yetkiliAdı değişkeni boş değer döndürmüyorsa çalışacak
    {
        duyuru.YetkiliAdı = yetkiliAdı;
        duyuru.Başlık = txtBaslik.Text;
        duyuru.Tarih =.dateTimePicker1.Value.Date;
        duyuru.Acıklama = txtAcıklama.Text;
        duyuru.YetkiliID = yetkiliID;

        duyuruManager.TAdd(duyuru);

        Listele();
        Temizle();

        MessageBox.Show("Başarıyla eklendi.", "Duyuru Ekleme", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    else // Girilen kullanıcı adına ait veritabanında kayıt yoksa çalışacak
    {
        MessageBox.Show("Kullanıcı adını hatalı. Lütfen tekrar deneyin.", "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
        txtKullaniciAd.Text = "";
        txtBaslik.Text = "";
        txtAcıklama.Text = "";
    }
}

```

Şekil 6.4.8.3. Duyuru ekle butonuna basıldığında çalışacak olan kodlar

```

private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
    // Datagrid'de listelenen alanlar içerisinde gözükmemesini istemediğimiz alanların görünümü false yapıldı.
    dataGridView1.Columns["Yetkili"].Visible = false;
    dataGridView1.Columns["YetkiliID"].Visible = false;

    // Datagrid'de seçilen duyuru bilgilerinin gerekli textboxlarda görüntülenmesi
    int secilen = dataGridView1.SelectedCells[0].RowIndex;
    txtKullaniciAd.Text = dataGridView1.Rows[secilen].Cells[1].Value.ToString();
    txtBaslik.Text = dataGridView1.Rows[secilen].Cells[2].Value.ToString();
    txtAciklama.Text = dataGridView1.Rows[secilen].Cells[3].Value.ToString();
    dateTimePicker1.Value = Convert.ToDateTime(dataGridView1.Rows[secilen].Cells[4].Value);
}

```

Şekil 6.4.8.4. Listelenen duyuru kayıtlarından herhangi biri seçildiğinde ilgili alanlara aktarım yapan kod

```

// Duyuru tablosu için oluşturulan mimariye erişebilmek adına nesne üretildi.
DuyuruManager duyuruManager = new DuyuruManager(new DuyuruRepository());

// Veri tabanından direkt olarak erişmemiz gerekiği durumlarda nesne üretildi.
DernekTakipEntities db = new DernekTakipEntities();

// Ekleme işlemi için Duyuru tablosundan nesne üretildi
Duyuru duyuru = new Duyuru();

2 references
void Listele() // Veri ekleme işlemi sonrasında datagridview'daki bilgiler tekrardan listelensin
{
    dataGridView1.DataSource = duyuruManager.GetList();

    // Datagrid'de listelenen alanlar içerisinde gözükmemesini istemediğimiz alanların görünümü false yapıldı.
    dataGridView1.Columns["Yetkili"].Visible = false;
    dataGridView1.Columns["YetkiliID"].Visible = false;
}

1 reference
void Temizle() // Ekleme işlemi sonrası ilgili alanlarda bulunan bilgilerin temizlenmesi için
{
    txtBaslik.Text = " ";
    txtAciklama.Text = " ";
    dateTimePicker1.Value = DateTime.Now;
}

```

Şekil 6.4.8.5. Duyuru işlemleri için Temizle ve Listele metotları

```

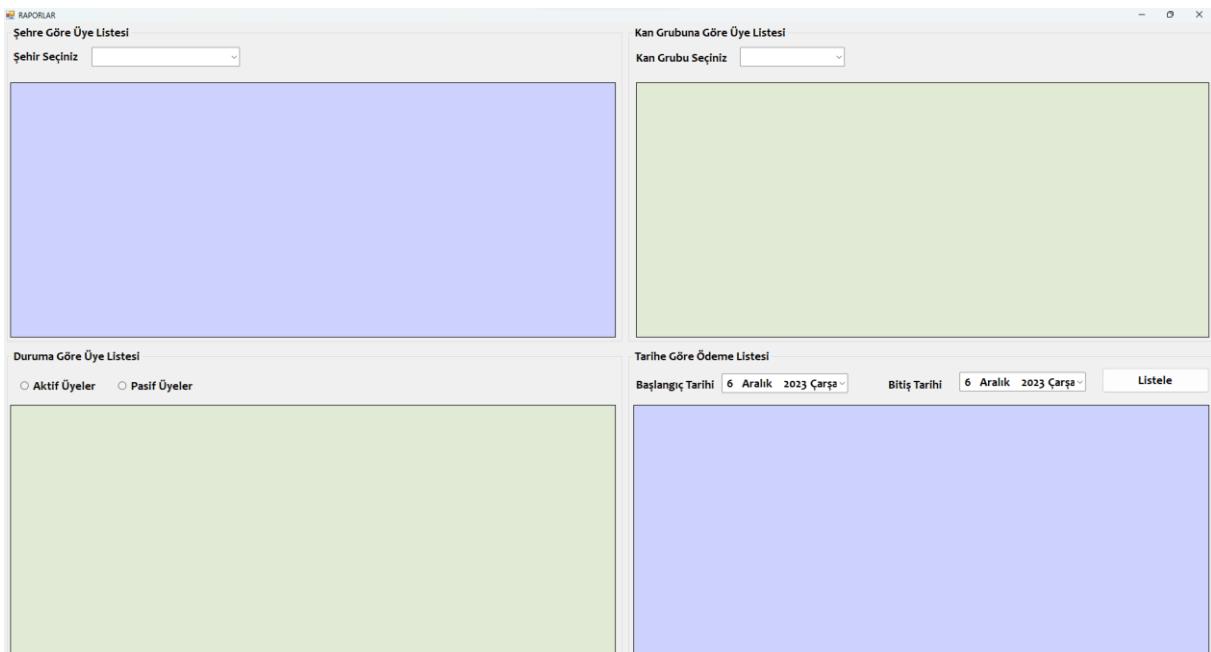
1 reference
private void Duyurular_Load(object sender, EventArgs e)
{
    Listele();
}

```

Şekil 6.4.8.6. Duyuru sayfası yüklenliğinde Listele metot çağrıması

6.4.9 Raporlama Ekranı

Raporlama sayfasında sistemde bulunan veriler için istenilen filtrelemeler yapılarak üyelerin datagridview üzerinde listelenmesi sağlanır. Bu filtrelemeler şehir seçilerek o şehrde ait olan üyelerin listelenmesi, kan grubu seçilerek o kan grubuna ait olan üyelerin listelenmesi, duruma göre(aktif-pasif) üyelerin listelenmesi ve seçilen iki tarih aralığında olan ödemelerin listelenmesi işlemleri bu raporlama sayfasında yapılmaktadır. Bu sayfa, yetkilinin istediği filtrelemeleri yaparak sistemi kullanmasına olanak sağlamaktadır.



Şekil 6.4.9.1. Raporlama ekranı

```

DernekTakipEntities db = new DernekTakipEntities();
UyeManager uyeManager = new UyeManager(new UyeRepository());
AidatManager aidatManager = new AidatManager(new AidatRepository());
1 reference
private void Raporlar_Load(object sender, EventArgs e)
{
    illerManager illermanager = new illerManager(new illerRepository());

    //comboboxta illerin listelenmesi
    var sehirler = illermanager.GetSehir();

    for (int i = 0; i < sehirler.Count; i++)
    {
        cmbSehir.Items.Add(sehirler[i]);
    }
}

```

Şekil 6.4.9.2. Raporlama ekranı yüklenliğinde çalışacak olan kod

Şehre göre üyelerin listelenmesi kısmında verilen combobox'ta; iller tablosunda bulunan bütün şehirler listelenir. Bu combobox'ta listelenen şehirlerden istenilen şehir seçilerek o şehrə ait olan üyelerin datagridview üzerinde listelenmesi sağlanır.

```

private void cmbSehir_SelectedIndexChanged(object sender, EventArgs e)
{
    //comboboxta bulunan illerden istediğimizi seçince o ile ait olan üyelerin listelenmesi
    var uyeeler = uyeManager.SehreGoreUyeler(cmbSehir.Text);
    dtgrdSehir.DataSource = uyeeler;
}

```

Şekil 6.4.9.3. Combobox SelectedIndexChanged özelliği kodları

Şehre Göre Üye Listesi									
Şehir Seçiniz KOCAELİ									
	ID	TC	Ad	Soyad	DogumTari	Sehir	KanGrubu	EMail	Durum
▶	1	11111111111	Sudenur	Celik	14.03.2002	KOCAELİ	O Rh+	celiksud...@g...	<input type="checkbox"/>
	9	6666666...	Mert	Kara	18.12.2003	KOCAELİ	A Rh+	mert@g...	<input checked="" type="checkbox"/>

Şekil 6.4.9.4. Seçilen şehre göre üyelerin listelenmesi

Kan grubuna göre üye listeleme de combobox'da bulunan kan gruplarından birinin seçilmesi halinde bu kan grubuna ait olan üyeleri DataGridView üzerinde listelenmektedir.

```
private void cmbKanGrup_SelectedIndexChanged(object sender, EventArgs e)
{
    //comboboxda bulunan kan gruplarından birini seç, nce o kan grubuna ait olan üyelerin listelenmesi
    var üyeler = uyeManager.KanGrubunaGoreUyeler(cmbKanGrup.Text);
    dtgrdKanGrubu.DataSource = üyeler;
}
```

Şekil 6.4.9.5. Combobox SelectedIndexChanged özelliği kodları

Kan Grubuna Göre Üye Listesi									
Kan Grubu Seçiniz A Rh+									
	ID	TC	Ad	Soyad	DogumTari	Sehir	KanGrubu	EMail	Durum
▶	3	22222222...	Büşra	Özdemir	16.05.2002	DÜZCE	A Rh+	busraozd...@g...	<input type="checkbox"/>
	9	6666666...	Mert	Kara	18.12.2003	KOCAELİ	A Rh+	mert@g...	<input checked="" type="checkbox"/>
	12	9999999...	Engin	Alkan	8.07.2001	DÜZCE	A Rh+	engin@g...	<input checked="" type="checkbox"/>
	16	4545454...	Özge	Bal	30.11.1999	MALATYA	A Rh+	ozge@g...	<input checked="" type="checkbox"/>
	22	12345678...	Ceyhun	Kayalı	17.10.1997	İZMİR	A Rh+	ceyhun...	<input checked="" type="checkbox"/>
	23	9876543...	Mustafa	Öz	3.07.2001	ZONGUL...	A Rh+	mustafa...	<input checked="" type="checkbox"/>

Şekil 6.4.9.6. Seçilen kan grubuna göre üyelerin listelenmesi

Duruma göre üyelerin listelenmesi kısmı radio buttonlar kullanılarak yapılmıştır. Eğer aktif olan üyelerin listelenmesi isteniyorsa aktif üyeleri yazan radio button işaretlenir ve DataGridView üzerinde bu filtreye uygun üyeleri listelenir. Pasif üyelerin görüntülenmesi de aynı şekilde yapılır. Bu sefer pasif üyeler butonu işaretlenerek üyelerin listelenmesi sağlanır.

```
private void rdbtnAktif_CheckedChanged(object sender, EventArgs e)
{
    //radio buttonlardan aktif olan seçildiğinde durumu aktif olan üyelerin listelenmesi
    var üyeler = uyeManager.DurumaGoreUyeler(true);
    dtgrdDurum.DataSource = üyeler;
}
```

Şekil 6.4.9.7. Aktif butonu seçildiğinde gerekli olan kod

Duruma Göre Üye Listesi

Aktif Üyeler Pasif Üyeler

ID	TC	Ad	Soyad	DogumTari	Sehir	KanGrubu	EMail	Durum
6	33333333...	Samet	Sarıkaya	6.08.2002	GİRESUN	B Rh+	samet@g...	<input checked="" type="checkbox"/>
7	4444444...	Sena	Aktürk	12.04.1995	ELAZIĞ	AB Rh-	sena@g...	<input checked="" type="checkbox"/>
9	6666666...	Mert	Kara	18.12.2003	KOCAELİ	A Rh+	mert@g...	<input checked="" type="checkbox"/>
10	77777777...	Can	Kaya	4.09.2004	İSTANBUL	O Rh-	can@gm...	<input checked="" type="checkbox"/>
12	9999999...	Engin	Alkan	8.07.2001	DÜZCE	A Rh+	engin@g...	<input checked="" type="checkbox"/>
13	12121212121	Bahar	Özlem	15.06.1998	İSTANBUL	A Rh-	bahar34...	<input checked="" type="checkbox"/>
14	23232323...	Mehmet	Asil	29.05.2002	İSTANBUL	B Rh+	mehmet...	<input checked="" type="checkbox"/>
15	3434343...	Tuğçe	Altın	20.02.2000	ESKİŞEHİR	O Rh+	tugce@g...	<input checked="" type="checkbox"/>
16	4545454...	Özge	Bal	30.11.1999	MALATYA	A Rh+	ozge@g...	<input checked="" type="checkbox"/>
17	5656565...	Ömer	Demir	2.09.1996	TRABZON	A Rh-	omer@g...	<input checked="" type="checkbox"/>
19	7878787...	Gökay	Durak	25.09.2002	DÜZCE	A Rh-	gokay@g...	<input checked="" type="checkbox"/>
20	8989898...	Esra	Kaba	27.10.2001	KONYA	O Rh-	esra@g...	<input checked="" type="checkbox"/>
21	9090909...	Melike	Göksoy	21.04.2003	ANTALYA	B Rh+	melike...	<input checked="" type="checkbox"/>
22	12345678...	Ceyhun	Kayalı	17.10.1997	İZMİR	A Rh+	ceyhun...	<input checked="" type="checkbox"/>
23	9876543...	Mustafa	Öz	3.07.2001	ZONGUL...	A Rh+	mustafa...	<input checked="" type="checkbox"/>
24	22223665...	Selim	Sönmez	23.03.2000	EDİRNE	A Rh-	selim@g...	<input checked="" type="checkbox"/>

Şekil 6.4.9.8. Seçilen duruma (aktif) göre üyelerin listelenmesi

```
private void rdbtnPasif_CheckedChanged(object sender, EventArgs e)
{
    //radio buttonlardan pasif olan seçiliince durumu aktif olan üyelerin listelenmesi
    var uyeler = uyeManager.DurumaGoreUyeler(false);
    dtgrdDurum.DataSource = uyeler;
}
```

Şekil 6.4.9.9. Pasif butonu seçildiğinde gerekli olan kod

Duruma Göre Üye Listesi

Aktif Üyeler Pasif Üyeler

ID	TC	Ad	Soyad	DogumTari	Sehir	KanGrubu	EMail	Durum
1	1111111111	Sudenur	Celik	14.03.2002	KOCAELİ	O Rh+	celiksud...	<input type="checkbox"/>
3	22222222...	Büşra	Özdemir	16.05.2002	DÜZCE	A Rh+	busraozd...	<input type="checkbox"/>
8	55555555...	Elif	Yılmaz	22.01.1998	DÜZCE	B Rh+	busraozd...	<input type="checkbox"/>
11	8888888...	Defne	Özer	10.10.2000	SAKARYA	A Rh-	celiksud...	<input type="checkbox"/>
18	6767676...	Canan	Dal	12.08.1998	BOLU	AB Rh+	busraozd...	<input type="checkbox"/>

Şekil 6.4.9.10. Seçilen duruma (pasif) göre üyelerin listelenmesi

Tarihe göre aidat ödeme listelemesi yapılrken ekranın bulunan başlangıç ve bitiş tarihleri dikkate alınır. Yetkilinin buradan seçmiş olduğu tarihler arasında bulunan aidat ödeme kayıtları DataGridView üzerinde listelenir.

```
private void button1_Click(object sender, EventArgs e)
{
    //datetimepickerlarda seçilen iki tarih arasında olan üyelerin listelenmesi
    var tarih = aidatManager.TariheGoreOdemeler(dateTimePicker1.Value.ToString(),dateTimePicker2.Value.ToString());
    dtgrdTarih.DataSource = tarih;
}
```

Şekil 6.4.9.11. Listele butonuna tıklandığında yapılması gereken kod

Tarihe Göre Ödeme Listesi						
	Başlangıç Tarihi	14 Kasım 2023	Sa	Bitiş Tarihi	30 Kasım 2023 Perş	Listele
ID	Miktar	OdemeTari	Durum	UyeID	Uye	
2	1150,00	15.11.2023	<input checked="" type="checkbox"/>	7		
4	1150,00	27.11.2023	<input checked="" type="checkbox"/>	12		
6	1150,00	30.11.2023	<input checked="" type="checkbox"/>	8		
7	1150,00	29.11.2023	<input type="checkbox"/>	3		
8	1150,00	16.11.2023	<input checked="" type="checkbox"/>	24		
12	1150,00	22.11.2023	<input checked="" type="checkbox"/>	9		
13	1150,00	24.11.2023	<input type="checkbox"/>	11		
16	1150,00	30.11.2023	<input checked="" type="checkbox"/>	16		
17	1150,00	24.11.2023	<input checked="" type="checkbox"/>	17		
18	1150,00	20.11.2023	<input type="checkbox"/>	18		
20	1150,00	29.11.2023	<input checked="" type="checkbox"/>	20		
21	1150,00	29.11.2023	<input checked="" type="checkbox"/>	22		

Şekil 6.4.9.12. Seçilen tarih aralıklarına göre aidat kayıtlarının listelenmesi

6.4.10 Grafikler

Projemizde grafik alanlarını temsil etmek için Zedgraph'dan faydalandık. Aylık aidat gelirleri, yıllık aidat gelirleri, şehirlere göre üye grafikleri olmak üzere 3 ayrı alan için zedgraph kullanılarak grafikler tasarlanmıştır.



Şekil 6.4.10.1. toolStripSplitButton da açılan alt alanlar

Aylık Aidat Gelir Grafiği

Aylık aidat gelir grafiğini tasarlarken hangi ay toplam kaç üyenin ödeme yaptığını daha iyi görselleştirmek için zedgraph kullanılmıştır. Bu grafiğin X ekseninde aylar sayısal olarak sırayla gösterilmiştir. Örneğin 1 Ocak, 2 Şubat ... 12 ise Aralığa karşılık gelmektedir. Y ekseninde ise aylara karşılık gelecek olan toplam üye sayısı bulunmaktadır. Üye sayısını bulmak için ise LINQ sorgusu kullanılarak her aya karşılık gelen üyelerin toplamı bulunmuştur.

```

private void BtnAyListele_Click(object sender, EventArgs e)
{
    DernekTakipEntities db = new DernekTakipEntities();
    // LINQ sorgusu
    var aylıkToplamlar = db.Aidat
        .GroupBy(a => new { Ay = a.OdemeTarihi.Month })
        .Select(g => new { Ay = g.Key.Ay, ToplamMiktar = g.Sum(a => a.Miktar) });

    GraphPane graphPane = zedGraphControl1.GraphPane;
    graphPane.CurveList.Clear(); // Önceki grafikleri temizle

    // Grafik başlıklar
    graphPane.Title.Text = "Aylık Aidat Miktarları";
    graphPane.XAxis.Title.Text = "Aylar";
    graphPane.YAxis.Title.Text = "Toplam Miktar (TL)";

    // X Eksenini
    string[] xLabels = aylıkToplamlar.Select(item => $"{item.Ay}").ToArray();
    double[] xValues = Enumerable.Range(1, aylıkToplamlar.Count()).Select(Convert.ToDouble).ToArray();

    // Y Eksenini
    double[] yValues = aylıkToplamlar.Select(item => Convert.ToDouble(item.ToplamMiktar)).ToArray();

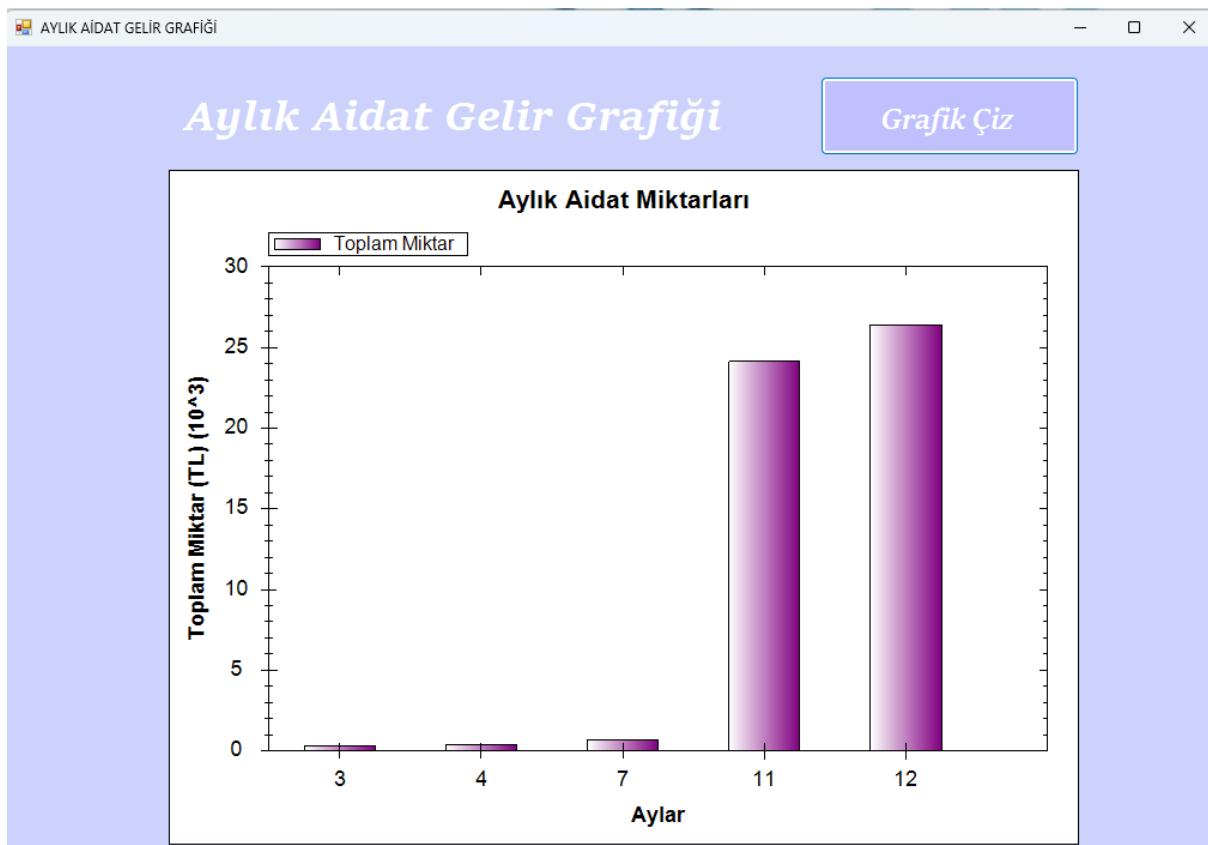
    // Çubuk grafik oluştur
    BarItem barItem = graphPane.AddBar("Toplam Miktar", null, yValues, System.Drawing.Color.Purple);

    // X eksenini etiketlerini ayarla
    graphPane.XAxis.Scale.TextLabels = xLabels;
    graphPane.XAxis.Type = AxisType.Text;
    graphPane.XAxis.Scale.Max = xValues.Max() + 1;

    // Grafiği güncelle
    zedGraphControl1.AxisChange();
    zedGraphControl1.Invalidate();
}

```

Şekil 6.4.10.2. Aylık aidat gelir grafiği çizmek için gerekli kodlar



Şekil 6.4.10.3. Aylık aidat gelir grafik ekranı

Yıllık Aidat Gelir Grafiği

Yıllık aidat grafiğini tasarlarken hangi yıl toplam kaç üyenin ödeme yaptığını daha iyi görselleştirmek için zedgraph kullanılmıştır. Bu grafiğin X ekseninde yıllar sırayla gösterilmiştir. Y ekseninde ise yıllara karşılık gelecek olan toplam üye sayısı bulunmaktadır. Üye sayısını bulmak için ise LINQ sorgusu kullanılarak her yıla karşılık gelen üyelerin toplamı bulunmuştur.

```
private void BtnAyListele_Click(object sender, EventArgs e)
{
    DernekTakipEntities db = new DernekTakipEntities();

    // LINQ sorgusu
    var yillikToplamlar = db.Aidat
        .GroupBy(a => new { Yil = a.OdemeTarihi.Year })
        .Select(g => new { Yil = g.Key.Yil, ToplamMiktar = g.Sum(a => a.Miktar) });

    GraphPane graphPane = zedGraphControl1.GraphPane;
    graphPane.CurveList.Clear(); // Önceki grafikleri temizle

    // Grafik başlıklarını
    graphPane.Title.Text = "Yıllık Aidat Miktarları";
    graphPane.XAxis.Title.Text = "Yıllar";
    graphPane.YAxis.Title.Text = "Toplam Miktar (TL)";

    // X Eksenini
    string[] xLabels = yillikToplamlar.Select(item => $"{item.Yil}").ToArray();
    double[] xValues = Enumerable.Range(1, yillikToplamlar.Count()).Select(Convert.ToDouble).ToArray();

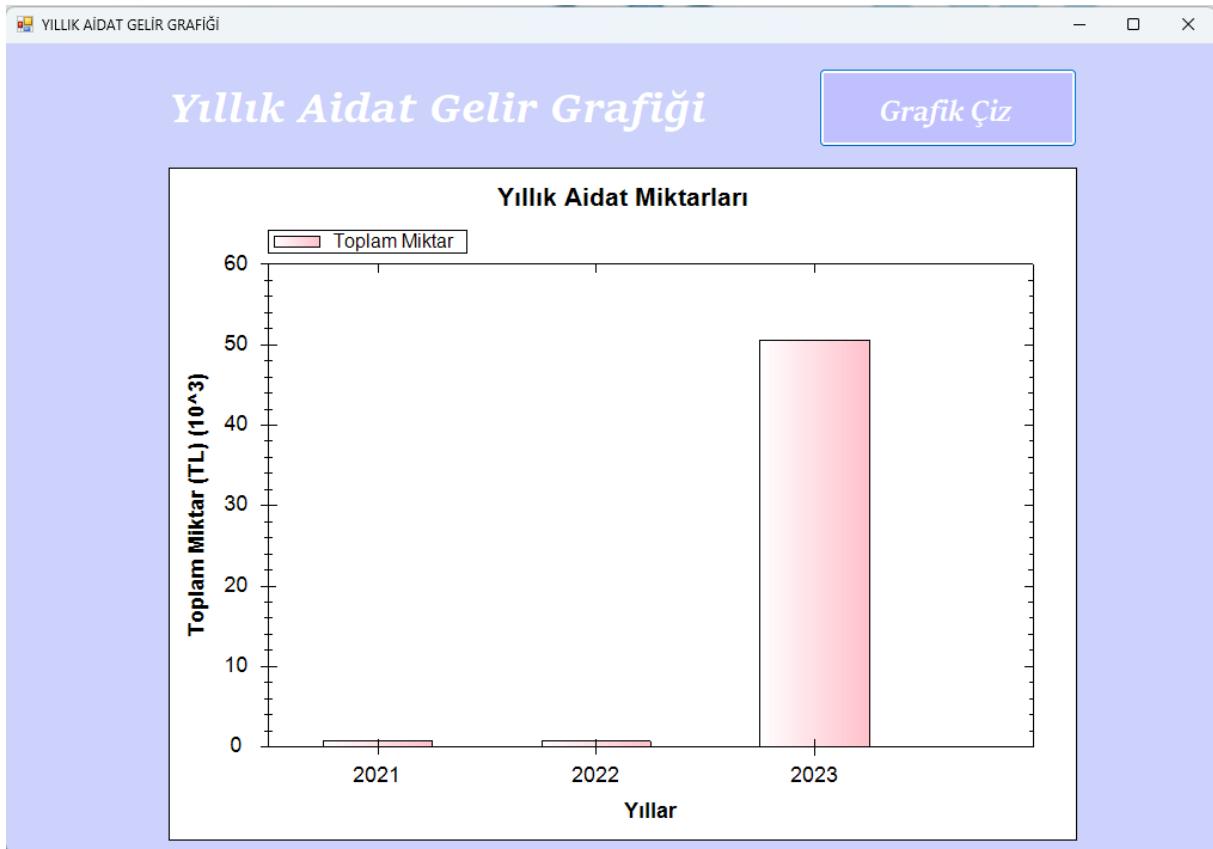
    // Y Eksenini
    double[] yValues = yillikToplamlar.Select(item => Convert.ToDouble(item.ToplamMiktar)).ToArray();

    // Çubuk grafik oluştur
    BarItem barItem = graphPane.AddBar("Toplam Miktar", null, yValues, System.Drawing.Color.Pink);

    // X eksenini etiketlerini ayarla
    graphPane.XAxis.Scale.TextLabels = xLabels;
    graphPane.XAxis.Type = AxisType.Text; // X eksenini metin olarak ayarla
    graphPane.XAxis.Scale.Max = xValues.Max() + 1;

    // Grafiği güncelle
    zedGraphControl1.AxisChange();
    zedGraphControl1.Invalidate();
}
```

Şekil 6.4.10.4. Yıllık aidat gelir grafiği çizmek için gerekli kodlar



Şekil 6.4.10.5. Yıllık aidat gelir grafik ekranı

Şehirlere Göre Üye Grafiği

Şehirlere göre üye grafiğini tasarlarken hangi şehirde kaç üye olduğunu daha iyi görselleştirmek için zedgraph kullanılmıştır. Bu grafiğin X ekseninde şehirler listelenmektedir. Y ekseninde ise bu şehrle karşılık gelen üye miktarı bulunmaktadır. Şehirlere özel üye sayısını bulmak için ise LINQ sorgusu kullanılmıştır. Bu grafik ile her şehrle karşılık gelen üyelerin toplamı grafiksel olarak gösterilmektedir.

```

private void BtnAyListele_Click(object sender, EventArgs e)
{
    DernekTakipEntities db = new DernekTakipEntities();

    // LINQ sorgusu
    var sonuc = db.Uye.GroupBy(u => u.Sehir).Select(g => new { Sehir = g.Key, KisiSayisi = g.Count() });

    GraphPane graphPane = zedGraphControl1.GraphPane;
    graphPane.CurveList.Clear(); // Önceki grafikleri temizle

    graphPane.Title.Text = "Şehirdeki Toplam Kişi Sayısı";
    graphPane.XAxis.Title.Text = "Şehirler";
    graphPane.YAxis.Title.Text = "Toplam Kişi Sayısı";

    // X eksenin etiketleri ve değerleri
    string[] xLabels = sonuc.Select(item => item.Sehir).ToArray();
    double[] xValues = Enumerable.Range(1, sonuc.Count()).Select(Convert.ToDouble).ToArray();

    // Y eksenin değerleri
    double[] yValues = sonuc.Select(item => Convert.ToDouble(item.KisiSayisi)).ToArray();

    // Çubuk grafik oluşturma
    BarItem barItem = graphPane.AddBar("Toplam Kişi Sayısı", null, yValues, System.Drawing.Color.DarkBlue);

    // X eksenin etiketlerini ayarla
    graphPane.XAxis.Scale.TextLabels = xLabels;
    graphPane.XAxis.Type = AxisType.Text; // X eksenin tipini metin olarak ayarla
    graphPane.XAxis.Scale.Max = xValues.Max() + 1;

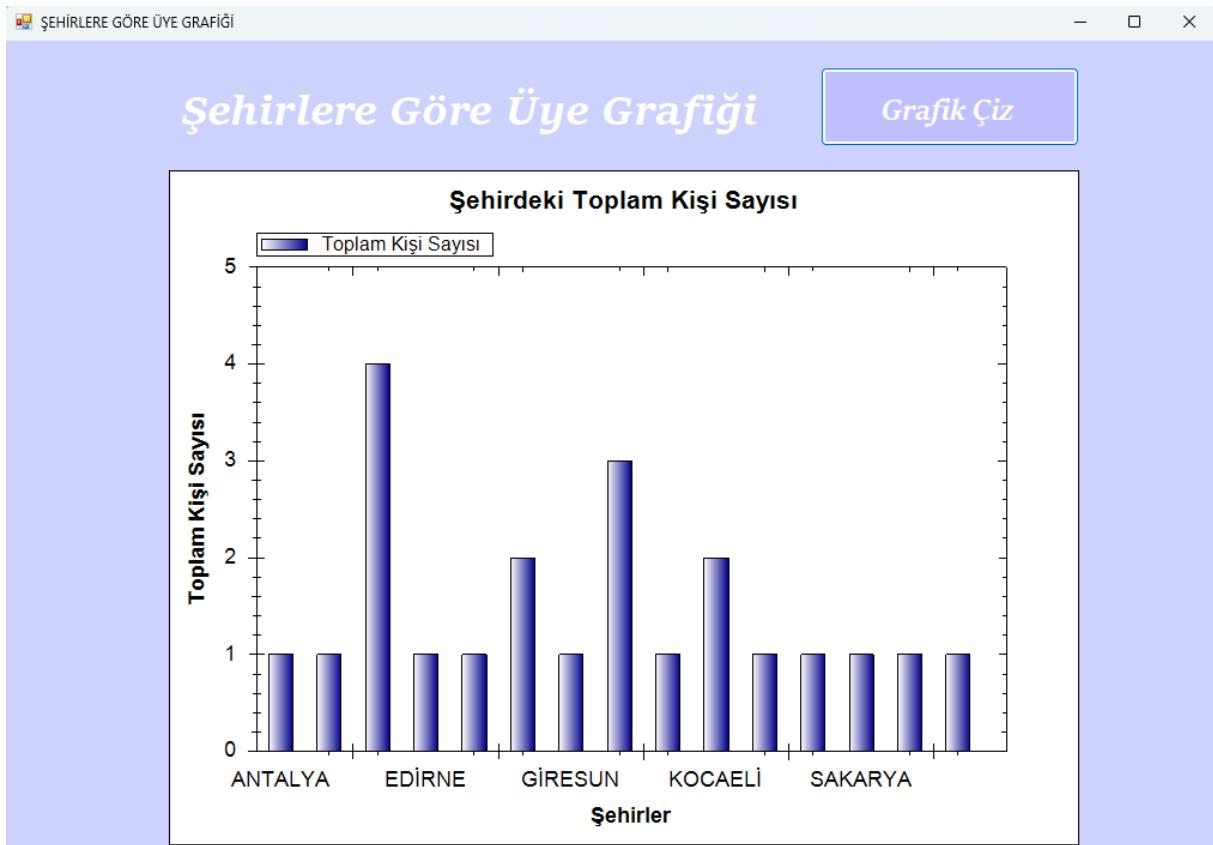
    // X eksenin etiketlerini daha iyi görünür hale getirme
    graphPane.XAxis.MajorTic.IsBetweenLabels = true;
    graphPane.XAxis.Scale.Align = AlignP.Inside;

    // Grafiği güncelle
    zedGraphControl1.AxisChange();
    zedGraphControl1.Invalidate();
}

}

```

Şekil 6.4.10.6. Şehirlere göre üye grafiği çizmek için gerekli kodlar



Şekil 6.4.10.7. Şehirlere göre üyelerin grafik ekranı