


T.C.
Zonguldak Bülent Ecevit Üniversitesi
Mühendislik Fakültesi
Öğrenci Staj Raporu

ÖĞRENCİNİN		
TC Kimlik No.	180106109046	
Adı - Soyadı	Erdoğan IŞIK	
Baba Adı	Yusuf	
Anne Adı	Emine	
Doğum Tarihi	02.09.1999	
Doğum Yeri	İstanbul	
Nüfusa Kayıtlı Olduğu	İl	İstanbul
	İlçe	Başakşehir
	Mahalle / Köy	Başak
	Cilt No.	YENİ KİMLİK KARTI KULLANIYORUM
	Aile Sıra No.	YENİ KİMLİK KARTI KULLANIYORUM
	Sıra No.	YENİ KİMLİK KARTI KULLANIYORUM



ÖĞRENCİNİN		
Bölümü	Bilgisayar Mühendisliği	
Okul Numarası	180106109046	
Sınıfı	Okuduğu	3
	Stajını Yaptığı	3
Stajını Yaptığı Kurum		
Stajını Yaptığı İl / İlçe		
Staja Başlama Tarihi	14 / 06 / 2021	
Stajı Bitirme Tarihi	09 / 07 / 2021	
Staj Süresi	20 İş günü	

STAJ YERİ YETKİLİSİ MÜHENDİSİN	
Adı Soyadı	Onay Tarihi ... / ... / 20...
İş ve Meslek Ünvanı (Kaşe)	İmzası

STAJ YERİ BİLGİLERİ

Staj Yapılan Kurum/Kuruluş/Firmanın

Adı:	
Adresi:	
Telefonu:	
Faksı:	
Web Sitesi:	
Çalışan Sayısı:	
Çalışma Alanı:	
Kapasitesi:	

Vaziyet Planı: (Tesisin görülebilir boyuttaki uydu görüntüsü veya çizimi eklenecek)

--

Stajı Onaylayan Mühendisin
Adı Soyadı:

İmzası:

Tarih: ... / ... / 20 ...

İÇİNDEKİLER

ÖNSÖZ	5
VERİTABANI	7
ADMIN PANELİ	11
Admin Login	11
Admin Layout	14
Anasayfa	15
Kategoriler	17
1-)Kategori Ekle	18
2-)Kategori Güncelleme	20
2-)Kategori Sil	21
Yazarlar	22
1-)Yazar Kitapları	22
Kitaplar	23
Üyeler	26
1-)Üye Hareketler	27
Personeller	28
Ödünç Kitap Ver	28
Ödünç Kitap Al	29
1-)Teslim Et	29
İşlemler	31
Duyurular	32
Ayarlar	32

İÇİNDEKİLER

VİTRİN	35
ShowcaseLayout	35
Yeni Eklenen	35
Tüm Kitaplar	36
KULLANICI PANELİ	37
Kullanıcı Login-Register	37
Membersdashboard	38
Profilim	39
Mesajlar	42
Kitap Hareketlerim	43
Vitrin	44
Error Sayfası	45

ÖNSÖZ

Kütüphane otomasyon sisteminde Admin paneli, Vitrin ve Kullanıcı paneli olmak üzere 3 adet panel bulunmaktadır.

Admin paneli;

Admin panelinin giriş sayfası kullanıcı panelinin giriş sayfasına nazaran bir buton yada link tarafından yönlendirilmemektedir. Url kısmına “Domainadı/Admin/Login” şeklinde giriş yapılmaktadır. Admin belirlemiş olduğu Mail ve şifresini yazarak sisteme giriş yapmaktadır. Admin panelinde Anasayfa, Kategoriler, Yazarlar, Kitaplar, Üyeler, Personeller, Ödünç Kitap Ver, Ödünç Kitap Al, İşlemler, Duyurular, Vitrin ve Ayarlar bölümleri bulunmaktadır.

Anasayfa: Burada emanet edilen kitabın vaktinde getirilmediğinde (7 gün) işlenen cezaların (Geciktiği gün başına 1 TL) toplamını, Toplam üye sayısını, Toplam kitap sayısını ve Emanet olarak verilen kitap sayısını gösteren kartlar bulunmaktadır.

Kategoriler: Kitap kategorisi ekleme, silme ve güncelleme işlemlerinin yapıldığı yerdir.

Yazarlar: Kitap yazarı ekleme, silme ve güncelleme işlemlerinin yapıldığı yerdir. Ayrıca bu kısımda yazarlara ait olan kitaplar da listelenmektedir.

Kitaplar: Kitap ekleme, silme ve güncelleme işlemlerinin yapıldığı yerdir. Ayrıca bu kısımda emanet olarak alınmış kitapların altında kırmızı, kütüphaneden ödünç alınmamış kitapların altında yeşil bir gösterge bulunmaktadır ve bu sayede hangi kitabın ödünç alınıp alınmadığı daha rahat görülmektedir.

Üyeler: Üye ekleme, silme ve güncelleme işlemlerinin yapıldığı yerdir. Ayrıca burada Hareketler kısmında belirtilen üyenin kitap hareketleri bulunmaktadır.

Personeller: Personel ekleme, silme ve güncelleme işlemlerinin yapıldığı yerdir.

Ödün Kitap Ver: Kütüphanemizin üyelerine kitap ödünç verildiği yerdir. Bu kısımda ödünç verirken hali hazırda ödünç olarak alınmış kitaplar gözükmemektedir. Kitap alış tarihinden 7 gün sonraya zorunlu kitap iade tarihi bulunmaktadır ve bu tarihin aşılması halinde günlük 1TL üyeye ceza verilmektedir.

Ödünç Kitap Al: Üyeye verilen ödünç kitabın alındığı kısımdır. Teslim et butonuna tıklandığında ödünç işleminin ayrıntıları görünür ve eğer üye zorunlu iade tarihinden sonra kitabı geri getirmişse kaç gün geç getirdiği belirtilir ve ona göre ceza işlemi uygulanır.

İşlemler: Başından sonuna kadar tamamlanan tüm ödünç işlemlerinin listelendiği kısımdır.

Duyurular: Duyuru ekleme, silme ve güncelleme işlemlerinin yapıldığı yerdir. Kullanıcı panelinde kullanıcıların görebileceği duyurular buradan gönderilir.

Vitrin: Kütüphanede bulunan kitapların listelendiği web sayfasıdır.

ÖNSÖZ

Ayarlar: Admin ekleme, silme ve güncelleme işlemlerinin yapıldığı yerdir. Adminlerde yetkilendirme sistemi bulunmaktadır ve sadece yetkisi olan admin giriş yaptığında bu bölüm gözükür. Yalnızca yetkisi “A” olan admin bu işlemleri gerçekleştirebilmektedir.

Vitrin;

Bu kısım Kullanıcıların göreceği ve projenin çalıştığı ana kısımdır. Admin panelinden eklenen kitaplar burada sergilenir. Ayrıca Toplam Üye, Toplam Kitap ve Personel sayısı burada kullanıcılara sunulur. Yeni eklenen, Tüm kitaplar ve giriş bölümleri bulunmaktadır.

Yeni eklenen: Bu bölümde kütüphaneye admin panelinden eklenen son 6 kitap listelenir.

Tüm kitaplar: Kütüphanede bulunan tüm kitaplar listelenir.

Giriş: Kullanıcı paneline giriş buradan yönlendirilir. Açılan sayfadan eğer üye değilse üye olup giriş yapar veyahut üye ise Mail ve şifresini girerek kullanıcı paneline giriş yapar.

Kullanıcı paneli;

Vitrin kısmından giriş yapan kullanıcıyı karşılayan paneldir. Kullanıcı panelinde Profilim, Mesajlar, Kitap Hareketlerim ve Vitrin bölümleri bulunmaktadır.

Profilim: Panele giriş yapıldığında Üyenin fotoğrafı, adı, soyadı, aldığı toplam kitap sayısı, mesaj sayısı ve duyuru sayısı bilgilerinin bulunduğu bir kart karşılar. Kartın hemen altında Üyenin Üniversite, telefon, kullanıcı adı, mail bilgilerinin bulunduğu başka bir kart bulunmaktadır. Ayrıca admin panelinden yayınlanan duyurular burada görünmektedir. Üyenin bilgilerini güncelleyebileceği ayarlar kısmı da burada bulunmaktadır.

Mesajlar: Üyelerin birbirleriyle mesajlaşabilecekleri bölümdür. Bu bölümde aktif olarak çalışan Gelen Kutusu, Giden Kutusu ve Yeni Mesaj kısımları bulunmaktadır. Üyeler birbirleri ile panel üzerinden mesajlaşmak istediklerinde mesajlaşmak istediği üyenin mail adresini bilmesi yeterlidir.

Kitap hareketlerim: Üyenin ödünç olarak aldığı kitapların listelendiği kısımdır. Üyenin kitabı aldığı tarih, eğer teslim etmişse teslim ettiği tarih, aldığı kitabın adı gibi bilgiler bulunmaktadır.

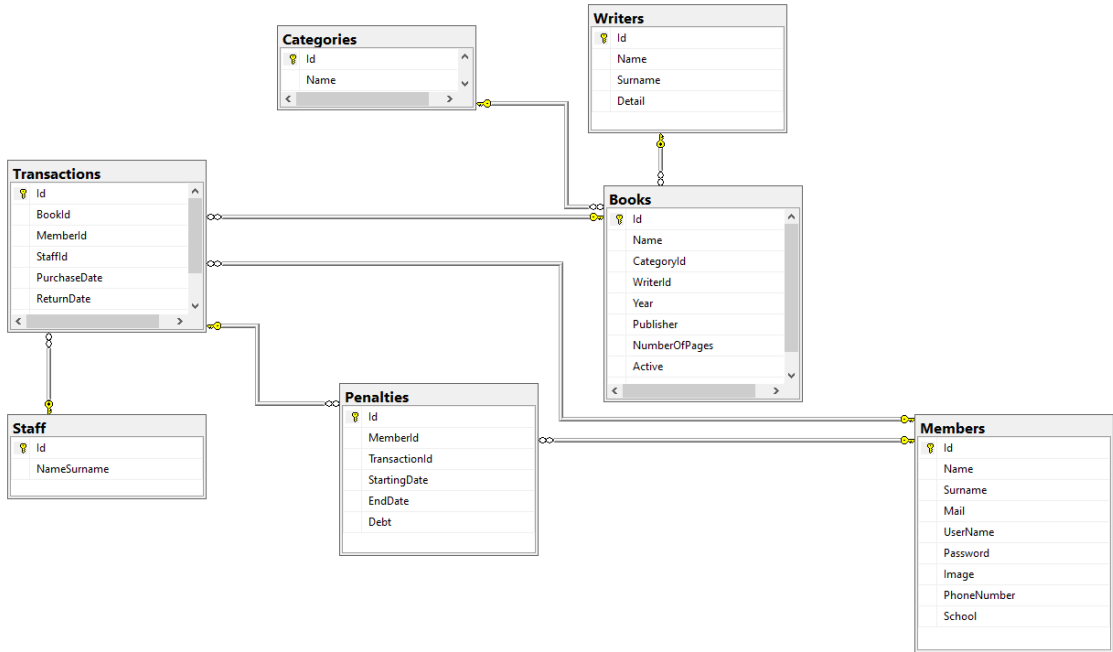
Vitrin: Vitrine geri döner.

VERİTABANI

+	dbo.Admins
+	dbo.Books
+	dbo.Categories
+	dbo.Members
+	dbo.Messages
+	dbo.MoneyCase
+	dbo.Notifications
+	dbo.Penalties
+	dbo.Staff
+	dbo.Transactions
+	dbo.Writers

Şekil 1:Veritabanı Tabloları

Şekil 1 'de görülen tablolar oluşturuldu.



Şekil 1.1:Veritabanı Diagramı

Oluşturulan tabloların ilişkilendirme işlemleri Şekil 1.1 'de görüldüğü gibi yapıldı ve Models/Entity klasörünün altına Model1 ismiyle projeye eklendi(Şekil 1.2).

VERİTABANI



Şekil 1.2:Veritabanının projeye eklenmesi

Ayrıca Transactions tablosunda 3 adet trigger yazıldı(Şekil1.3).

trigger
BOOKACTIVE
BOOKACTIVE2
DEBT

Şekil1.3:Trigger

Stajı Onaylayan Mühendisin
Adı Soyadı:

İmzası:

Tarih: ... / ... / 20 ...

VERİTABANI

BOOKACTIVE: Transactions(İşlemler) tablosunda bulunan BookId sütununda bir değişiklik olursa Books(Kitaplar) tablosunda karşılığını bulup o tabloda bulunan ve kitabın ödünç alınıp alınmadığını belirten Active sütununu 0 değerine eşitler. Bu sayede kitap ödünç alındığında trigger harekete geçer ve ödünç verilen kitap Active=0 durumuna geçer. Bu sayede kitabın ödünç alındığı belirlenir(Şekil 1.4).

```
USE [Kutuphane]
GO
/***** Object: Trigger [dbo].[BOOKACTIVE]
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [dbo].[BOOKACTIVE]
ON [dbo].[Transactions]
AFTER INSERT
AS
DECLARE @BookId INT
SELECT @BookId=BookId From inserted
UPDATE Books SET Active=0 WHERE Id=@BookId
```

Şekil 1.4:BOOKACTIVE

BOOKACTIVE2: BOOKACTIVE triggerının yaptığı işlemin tam tersini yapar. Bu sayede ödünç alınan kitap geri getirildiğinde kitap ACTIVE=1 durumuna geçerek başka birinin ödünç alabileceği belirlenir(Şekil 1.5).

```
USE [Kutuphane]
GO
/***** Object: Trigger [dbo].[BOOKACTIVE2]
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [dbo].[BOOKACTIVE2]
ON [dbo].[Transactions]
AFTER UPDATE
AS
DECLARE @BookId INT
SELECT @BookId=BookId FROM inserted
UPDATE Books SET Active=1 WHERE Id=@BookId
```

Şekil 1.5:BOOKACTIVE2

VERİTABANI

DEBT: Bu trigger üye kitabı geri getirdiğinde eğer geç getirmişse Penalties(Cezalar) tablosuna üyenin id(MemberId)'si, işlem id(TransactionId)'si ve son olarak Borç(Debt) kısımlarını yazar. Zorunlu iade tarihinden (ReturnDate) üyenin getirdiği tarihi (MemberReturnDate) çıkartır. Kütüphanemizde her geç gün başına 1TL ceza uygulandığından bu sonuç üyenin ceza yediği miktardır. Fakat üye zorunlu iade tarihinden erken getirirse Debt sütununa “-” bir değer yazılmaması için bir if bloğu kullanıldı, eğer erken getirmişse “-” bir değer yerine “0” eğer geç getirmişse ise geç getirdiği gün kadar ceza Debt sütununa yazılmaktadır(Şekil 1.6).

```
USE [Kutuphane]
GO
/***** Object: Trigger [dbo].[DEBT]    Script Date: 23.06.2021 23:47:46 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [dbo].[DEBT]
ON [dbo].[Transactions]
AFTER UPDATE
AS
DECLARE @MemberId int
DECLARE @Id int
DECLARE @Debt int
SELECT @MemberId=MemberId,@Id=Id FROM inserted
SELECT @Debt = (SELECT DATEDIFF(DAY,ReturnDate,MemberReturnDate) FROM Transactions WHERE Id=@Id)
IF @Debt>=0 BEGIN
INSERT INTO Penalties (MemberId,TransactionId,Debt) VALUES (@MemberId,@Id,@Debt)
END
ELSE BEGIN
INSERT INTO Penalties (MemberId,TransactionId,Debt) VALUES (@MemberId,@Id,0)
END
```

Şekil 1.6:DEBT

ADMIN PANELİ

Admin Login

Admin giriş paneli için bir tane "AdminController.cs" isminde controller oluşturuldu. İlk olarak bir veri tabanı nesnesi yazıldı(Şekil 2).

```
// GET: Admin
KutuphaneEntities kutuphaneEntities = new KutuphaneEntities();
0 references
```

Şekil 2:Veritabanı Nesnesi

Daha sonra biri sayfa açıldığında diğeri ise giriş yap butonuna tıklandığında çalışacak olan iki adet Login() metodu oluşturuldu(Şekil 2.1).

```
public ActionResult Login()
{
    return View();
}

[HttpPost]
0 references
public ActionResult Login(Admins admins)
{
    var result = kutuphaneEntities.Admins.FirstOrDefault(p => p.Mail == admins.Mail && p.Password == admins.Password);
    if (result != null)
    {
        FormsAuthentication.SetAuthCookie(result.Mail, false);
        Session["Mail"] = result.Mail.ToString();
        return RedirectToAction("Index", "Statistics");
    }
    return View();
}
```

Şekil 2.1:Login()

Login() metoduna View eklendi ve bu View içine login sayfasının tasarım kodları yazıldı(Şekil 2.2).

```
<body>
<div class="limiter">
<div class="container-login100">
<div class="wrap-login100">
<div class="login100-pic js-tilt" data-tilt>
<a href="/Showcase/Index/">

</a>
</div>
<form class="login100-form validate-form" method="post">
<span class="login100-form-title">
Admin Giriş
</span>
<div class="wrap-input100 validate-input" data-validate="Geçerli e-posta gerekli: ex@abc.xyz">
<input class="input100" type="text" name="Mail" placeholder="Mail">
<span class="focus-input100"></span>
<span class="symbol-input100">
<i class="fa fa-envelope" aria-hidden="true"></i>
</span>
</div>
<div class="wrap-input100 validate-input" data-validate="Şifre gerekli">
<input class="input100" type="password" name="Password" placeholder="Şifre">
<span class="focus-input100"></span>
<span class="symbol-input100">
<i class="fa fa-lock" aria-hidden="true"></i>
</span>
</div>
<div class="container-login100-form-btn">
<button class="login100-form-btn">
Giriş
</button>
</div>
</form>
</div>
</div>
</div>
</div>
```

Şekil 2.2:Admin Login View

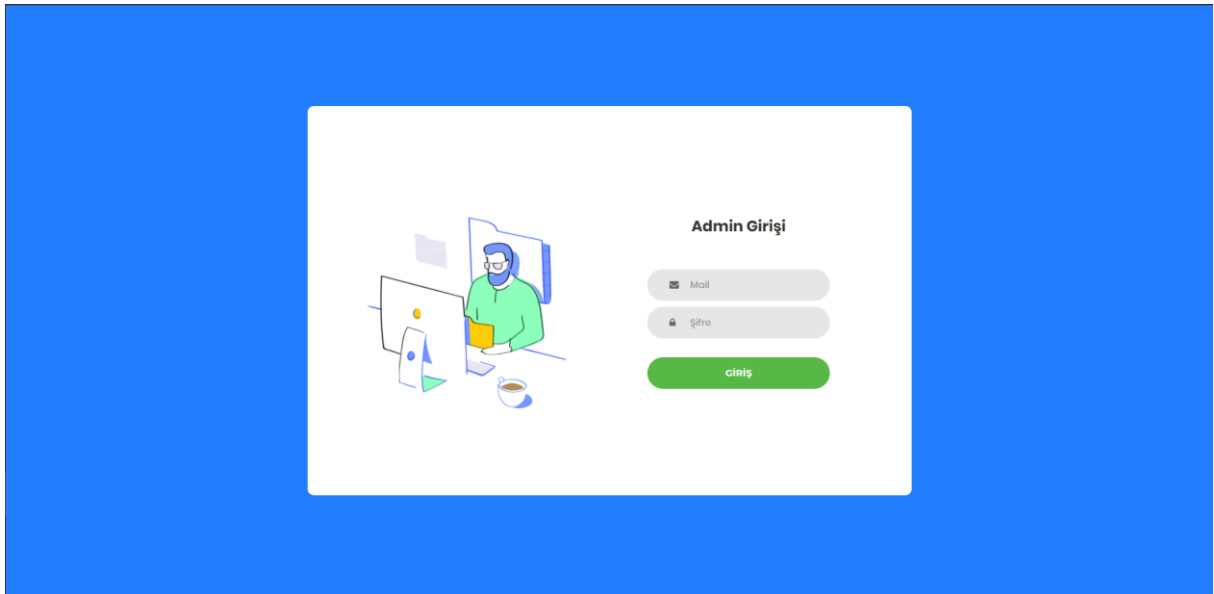
Stajı Onaylayan Mühendisin
Adı Soyadı:

İmzası:

Tarih: ... / ... / 20 ...

ADMİN PANELİ

Şekil 2.2’de görüldüğü gibi view içinde method=”post” olan bir form oluşturuldu ve input içinde bulunan name=”” kısımlarına Admins tablosundaki sütunların adı yazıldı. Bu şekilde Giriş butonuna tıklandığında Şekil 2.1’de bulunan Post Login() metoduna gitmekte. Ayrıca Şekil 2.2 de fotoğraf kısmına Showcase/Index/ yönlendirilmesi yapıldı. Fotoğrafa tıklandığında Vitrin sayfasına yönlendirilme sağlandı. Şekil 2.1 ‘de ki post Login() metodu Admins türünde admins adında bir değişken almakta bu değişken Admin Login ekranında girilen bilgileri tutmakta olup Maile göre sorgulanmaktadır. Eğer girilen Mail ve şifre ye ait Admins tablosunda bir admin var ise o satırı yok ise null döndürmesi sağlandı. Eğer girilen değerlere ait bir sütun yok ise return View() komutu ile Admin Login sayfası yeniden yüklenir ve herhangi bir yönlendirme yapılmaz. Fakat sütunda böyle bir admin var ise Web.config dosyasında (Şekil 2.4) ayarını yaptığımız oturum açma işlemi gerçekleştirilir. Bu oturum açma işlemi Mail üzerinden yapılmaktadır. Bu oturum açma işlemi System.Web.Security kütüphanesi sayesinde yapılmaktadır. Bu işlemten sonra admin panelinin anasayfası olan ve Statistics controllerında bulunan Index sayfasına yönlendirme yapılmaktadır. Ayrıca bu Authorize işleminde kullanıcı Url kısmına Statistics/Index yazıldığında admin panelinin açılmaması ve Login sayfasına yönlendirilmesi sağlandı. Bunun sağlanması için proje bazında Authorize yapıldı (Şekil 2.5). Bu işlem yapıldığında proje içindeki her şeye erişim için Authorize istendiğinden dolayı bazı istisnalar belirlenmesi gerekti (Örn. Admin Login, Login, Register, Showcase). Bu istisnalar belirtilen Controllerların başına [AllowAnonymous] etiketi yazılarak sağlandı (Şekil 2.6). Aynı zamanda admin yetki sistemi için de Web.config ‘de ayarlamalar yapıldı. Roles klasörü oluşturulup içine AdminRoleProvide class’ı oluşturuldu. Bu class’ın içine RoleProvider implementasyonu gerçekleştirildi. Daha sonra gerekli satıra yetkilendirme sistemi için kod yazıldı (Şekil 2.7). Son olarak admin panelinde bulunan oturum kapatma butonu için bir metot daha yazıldı ve admin panelinden bu tuşa bağlı yönlendirme bu controller’a yapıldı (Şekil 2.8) ve çıkış yapılma işleminden sonra tekrar admin login sayfasına yönlendirildi.



Şekil 2.3:Admin Login Sayfa Tasarımı

ADMİN PANELİ

```
<authentication mode="Forms">
  <forms loginUrl="/Login/Login/"></forms>
</authentication>
<roleManager enabled="true" defaultProvider="TestProvider">
  <providers>
    <add name="TestProvider" type="Kutuphane.Roles.AdminRoleProvider"/>
  </providers>
</roleManager>
</system.web>
```

Şekil 2.4:Web.config

```
GlobalFilters.Filters.Add(new AuthorizeAttribute());
```

Şekil 2.5:Global.asax

```
[AllowAnonymous]
0 references
public class AdminController : Controller
{
    // GET: Admin
    KutuphaneEntities kutuphaneEntities = new KutuphaneEntities();
    0 references
    public ActionResult Login()
```

Şekil 2.6:AdminController.cs

```
0 references
public override string[] GetRolesForUser(string username)
{
    KutuphaneEntities kutuphaneEntities = new KutuphaneEntities();
    var result = kutuphaneEntities.Admins.FirstOrDefault(p => p.Mail == username);
    return new string[] { result.Authorize };
}
```

Şekil 2.7:AdminRoleProvider.cs

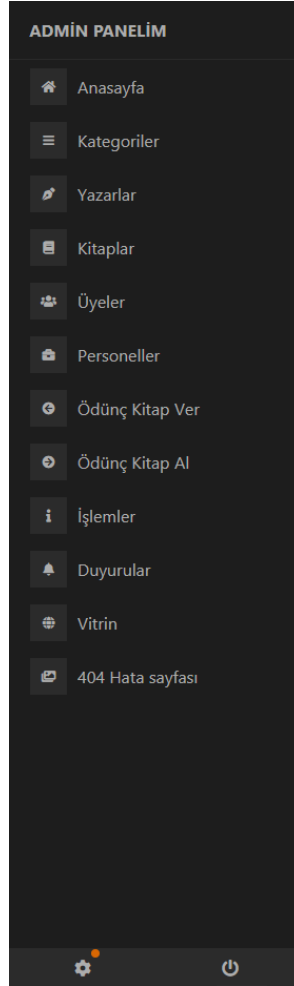
```
0 references
public ActionResult Logout()
{
    FormsAuthentication.SignOut();
    return RedirectToAction("Login", "Admin");
}
```

Şekil 2.8:AdminController, Logout

ADMİN PANELİ

Admin Layout

Views/Shared klasörünün altına yeni bir Layout.cshtml adında bir layout oluşturuldu. Bu layout admin panelinin değişmeyecek olan menü kısmını oluşturmaktadır (Şekil 2.9).



Şekil 2.9:Admin Menü Tasarım

Gerekli yönlendirmeler yapıldı. Bu sayfada değişecek olan içerik kısmına html kodları içinde @RenderBody() etiketi ile belli edildi (Şekil 2.10).

```
<!-- page-content -->
<main class="page-content pt-2">
  <div id="overlay" class="overlay"></div>
  <div class="container-fluid p-5">
    @RenderBody()
  </div>
</main>
<!-- page-content -->
```

Şekil 2.10: Layout.cshtml, @RenderBody()

ADMIN PANELİ

Yapılan yetkilendirme sisteminde ayarlar kısmını belirlenen yetkili dışında görünümüne kapanması sağlanacak kodlar yazıldı (Şekil 2.11).

```
<div class="sidebar-footer">
  @if (User.IsInRole("A"))
  {
    <div class="dropdown">
      <a href="#" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
        <i class="fa fa-cog"></i>
        <span class="badge-sonar"></span>
      </a>
      <div class="dropdown-menu" aria-labelledby="dropdownMenuMessage">

        <a class="dropdown-item" href="/Settings/Index/">Ayarlar</a>

      </div>
    </div>
  }
</div>
```

Şekil 2.11:Layout.cshtml, Role

Anasayfa

Statistics controller ı oluşturuldu ve içinde bulunan Index 'e View eklendi. View eklenirken oluşturduğumuz layout sayfası kullanıldı.

```
KutuphaneEntities kutuphaneEntities = new KutuphaneEntities();
0 references
public ActionResult Index()
{
    var result = kutuphaneEntities.Members.Count();
    ViewBag.MembersCount = result;
    var result2 = kutuphaneEntities.Books.Count();
    ViewBag.BooksCount = result2;
    var result3 = kutuphaneEntities.Books.Where(p => p.Active == false).Count();
    ViewBag.LoansCount = result3;
    var result4 = kutuphaneEntities.Penalties.Sum(p => p.Debt);
    ViewBag.Money = result4;
    return View();
}
```

Şekil 2.12:StatisticsController, Index

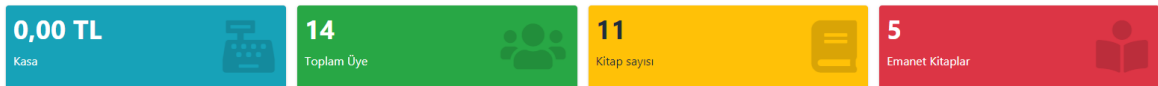
Şekil 2.12 'de görülen Index metodunda kitap, üye, emanet kitap ve kasadaki toplam para Linq sorguları ile alındı ve ViewBag'ler yardımıyla Index View'ine taşındı.


```
<div class="row">
  <div class="col-lg-3 col-6">
    <!-- small card -->
    <div class="small-box bg-info">
      <div class="inner">
        <h3>@ViewBag.Money TL</h3>

        <p>Kasa</p>
      </div>
      <div class="icon">
        <i class="fas fa-cash-register"></i>
      </div>
    </div>
  </div>
  <!-- ./col -->
  <div class="col-lg-3 col-6">
    <!-- small card -->
    <div class="small-box bg-success">
      <div class="inner">
        <h3>@ViewBag.MembersCount </h3>

        <p>Toplam Üye</p>
      </div>
      <div class="icon">
        <i class="fas fa-users"></i>
      </div>
    </div>
  </div>
</div>
```

Şekil 2.13:Statistics Index.cshtml



Şekil 2.14:Anasayfa Tasarımı

ADMIN PANELİ

Kategoriler

Categories controller ı oluşturuldu ve içinde bulunan Index 'e View eklendi. View eklenirken oluşturduğumuz layout sayfası kullanıldı. Veritabanı nesnesi oluşturuldu. Sayfalama yapısı için PagedList kütüphanesi kullanıldı. Ayrıca arama yapabilmek için Linq sorgusu yazıldı. Arama yapılmadığında yani serach kısmı boş geldiğinde tüm kategorilerin listelenmesi için if ile şartı kodlandı (**Şekil 2.15**).

```
// GET: Categories
KutuphaneEntities kutuphaneEntities = new KutuphaneEntities();
0 references
public ActionResult Index(string search, int page = 1)
{
    var result = from c in kutuphaneEntities.Categories select c;
    if (!string.IsNullOrEmpty(search))
    {
        result = result.Where(p => p.Name.Contains(search));
    }
    return View(result.Where(p => p.Active == true).ToList().ToPagedList(page, 16));
}
```

Şekil 2.15:CategoriesController, Index

Oluşturulan View içine Categories tipinde listeleme için model oluşturuldu ve foreach döngüsü ile tüm kategoriler listelendi (**Şekil 2.16**).

```

using Kutuphane.Models.Entity;
using PagedList;
using PagedList.Mvc;
using PagedList.IPagedList<Categories>

ViewBag.Title = "Index";
Layout = "~/Views/Shared/Layout.cshtml";

using (Html.BeginForm("Index", "Categories", FormMethod.Get))
{
    <div style="margin-bottom:15px; margin-top:15px">
        @Html.TextBox("search", null, new { @class = "form-control", placeholder = "Kategori Ara.." })
    </div>

    <button class="btn btn-outline-success btn-lg btn-block" data-toggle="modal" data-target="#Modal1">Kategori ekle</button>

    <div class="container" style="padding-top: 15px">
        <div class="row">
            @foreach (Categories result in Model)
            {
                <div class="col-3">
                    <div class="card border-dark mb-3" style="max-width: 18rem;">
                        <div class="card-body text-dark" style="text-align:center">
                            
                            <h5 class="card-title">@result.Name</h5>
                            <h6 class="card-subtitle mb-2 text-muted">Id: @result.Id</h6>
                            <br />
                            <div>
                                <a href="~/Categories/FetchCategory/@result.Id" class="btn btn-primary btn-sm">Güncelle</a>
                                @Html.ActionLink("Sil", "CategoryDelete", new { id = result.Id }, new { @class = "btn btn-danger btn-sm", onclick = "return confirm('Silmek istiyor musunuz?')"} )
                            </div>
                        </div>
                    </div>
                </div>
            }
        </div>
    </div>
    @Html.PagedListPager((IPagedList)Model, page => Url.Action("Index", new { page }))
</div>

```

Şekil 2.16:Categories Index.cshtml

Kategori Ara...

Kategori ekle

Macera id: 1 Güncelle Sil	Bilimkurgu id: 2 Güncelle Sil	Fantastik id: 3 Güncelle Sil	Şiir id: 4 Güncelle Sil
Tiyatro id: 5 Güncelle Sil	Çocuk Kitapları id: 6 Güncelle Sil	Rus Klasikleri id: 7 Güncelle Sil	Hikaye id: 8 Güncelle Sil
Güldürü id: 9 Güncelle Sil	Kişisel Gelişim id: 10 Güncelle Sil	Biyografi id: 11 Güncelle Sil	Akademik id: 12 Güncelle Sil
Dergi id: 15 Güncelle Sil	Din id: 2002 Güncelle Sil	Anı id: 2003 Güncelle Sil	Tarih id: 2004 Güncelle Sil

12>

Şekil 2.17:Kategoriler Tasarım

Di

Kategori ekle

Din id: 2002 Güncelle Sil	Dilbilim id: 2007 Güncelle Sil
---------------------------------	--------------------------------------

1

Şekil 2.18:Kategoriler Tasarım

1-)Kategori Ekle

Kategori ekleme için öncelikle controlller kısmında biri get biri post olmak üzere iki adet metot oluşturuldu (Şekil 2.19). Ekleme işlemi için yeni bir View oluşturmak yerine pop-up kullanıldı bundan dolayı get metodundan kategoriler listelendi. Post metodunda ilk önce eklenecek olan kategorinin durumu true yapıldı, veri tabanına eklendi, veri tabanı kaydedildi ve en son Index sayfasına yönlendirme sağlandı.

```
[HttpGet]
0 references
public ActionResult CategoryAdd()
{
    var result = kutuphaneEntities.Categories.ToList();
    return View(result);
}

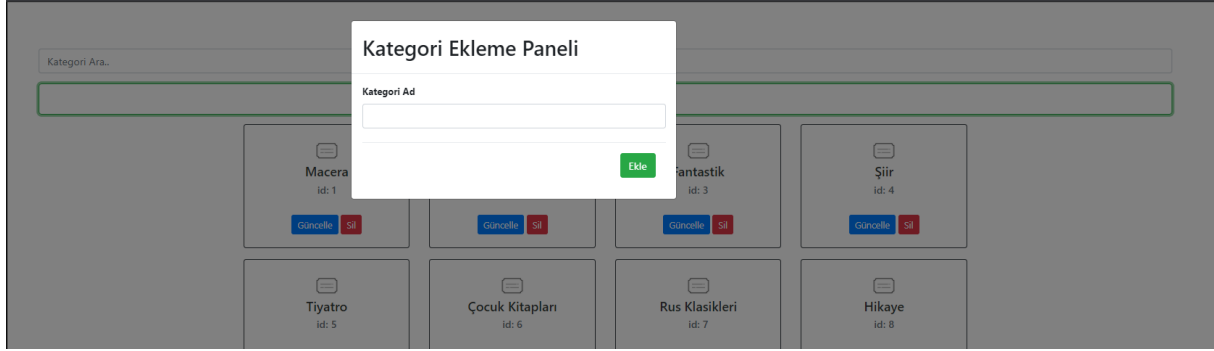
[HttpPost]
0 references
public ActionResult CategoryAdd(Categories categories)
{
    categories.Active = true;
    kutuphaneEntities.Categories.Add(categories);
    kutuphaneEntities.SaveChanges();
    return RedirectToAction("Index");
}
```

Şekil 2.19: CategoriesController, CategoryAdd

Pop-up için Index View'e kodlar yazıldı (Şekil 2.20).

```
<div class="modal" id="Modal1">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h2 class="modal-title">Kategori Ekleme Paneli</h2>
      </div>
      <form action="/Categories/CategoryAdd/" method="post">
        <div class="modal-body">
          <label><b>Kategori Ad</b></label>
          <input type="text" name="Name" class="form-control" />
          <br />
          <div class="modal-footer">
            <button class="btn btn-success">Ekle</button>
          </div>
        </div>
      </form>
    </div>
  </div>
</div>
```

Şekil 2.20



Şekil 2.21:Kategori Ekle Tasarım

2-)Kategori Güncelleme

Güncelleme işlemi için ilk önce güncellenecek olan kategorinin bilgileri Textbox'lara yazdırılması için o kategoriye ait bilgileri bulacak olan FetchCategory adında bir metot yazıldı. Yazılan metot Index'de güncelle butonuna tıklandığında tıklanan kategorinin id sini bu metoda gönderiyor. Gönderilen id tablodan bulunup oluşturulan View'e gönderiliyor aynı zamanda oluşturulan View'e yönlendirme yapılıyor (Şekil 2.22). Oluşturulan View'de Categories veri tipinde bir model oluşturuldu ve html helper yardımıyla textbox'lara yazıldı (Şekil 2.23). Kullanıcı değişiklik yapmak istediğinde Güncelle butonuna bastığında çalışacak olan CategoryUpdate metodu yazıldı. Gelen değeri id'sine göre bulup tablodaki değerleri gelen değerlerle değiştirdikten sonra veri tabanına kaydedip tekrar Index sayfasına yönlendirme sağlandı (Şekil 2.22).

```
0 references
public ActionResult FetchCategory(int id)
{
    var result = kutuphaneEntities.Categories.Find(id);
    return View("FetchCategory", result);
}

0 references
public ActionResult CategoryUpdate(Categories categories)
{
    var result = kutuphaneEntities.Categories.Find(categories.Id);
    result.Name = categories.Name;
    kutuphaneEntities.SaveChanges();
    return RedirectToAction("Index");
}
```

Şekil 2.22: CategoriesController, FetchCategory-CategoryUpdate

```
@model Kutuphane.Models.Entity.Categories
@{
    ViewBag.Title = "FetchCategory";
    Layout = "~/Views/Shared/Layout.cshtml";
}

@using (Html.BeginForm("CategoryUpdate", "Categories", FormMethod.Post))
{
    <div style="margin-top:15px"></div>
    <div class="form-group">
        @Html.LabelFor(p=>p.Id)
        @Html.TextBoxFor(p=> p.Id,new { @class="form-control", @readonly = "readonly" })
        <br />
        @Html.LabelFor(p=> p.Name)
        @Html.TextBoxFor(p=> p.Name,new { @class = "form-control" })
        <br />
    </div>
    <div class="modal-footer">
        <button class="btn btn-primary">Güncelle</button>
    </div>
}
```

Şekil 2.23:Categories, FetchCategory.cshtml



Şekil 2.24:Kategori Güncelle Tasarım

3-)Kategori Sil

Silme işleminde Categories tablosu başka tablolara da bağlı olduğundan Active kısmını false yaparak silme işlemi uygulandı. Tablodan silinmedi. Veri tabanı kaydedildi ve Index yönlendirmesi yapıldı (Şekil 2.24). Ayrıca sil butonuna tıklandığında Silmek istiyor musunuz? Şeklinde uyarı mesajı çıkması Index tarafından sağlandı.

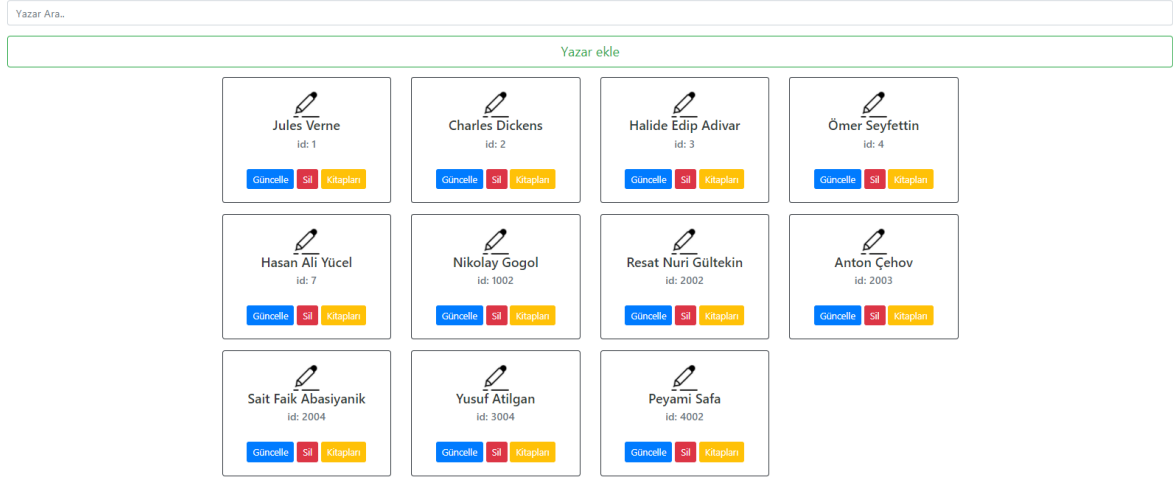
```
0 references
public ActionResult CategoryDelete(int id)
{
    var result = kutuphaneEntities.Categories.Find(id);
    result.Active = false;
    kutuphaneEntities.SaveChanges();
    return RedirectToAction("Index");
}
```

Şekil 2.25: Categories, CategoryDelete

ADMİN PANELİ

Yazarlar

Kategoride yapılan işlemler yazarlar için de yapıldı.



Şekil 2.26:Yazarlar Tasarım

1-)Yazar Kitapları

Kategorilerden farklı olarak Yazarların yazdığı kitaplar bulunmaktadır. Bunun için WriterBooks adında bir metot oluşturuldu ve View eklendi. Books tablosunda id si gönderilen geçerli yazara ait kitaplar listelendi. Son olarak yazarın adı ViewBag yardımıyla View'e taşındı (Şekil 2.27).

```
0 references
public ActionResult WriterBooks(int id)
{
    var result = kutuphaneEntities.Books.Where(p => p.WriterId == id).ToList();
    var writerName = kutuphaneEntities.Writers.Where(p => p.Id == id).Select(p => p.Name + " " + p.Surname).FirstOrDefault();
    ViewBag.NameSurname = writerName;
    return View(result);
}
```

Şekil 2.27:WritersController, WriterBooks

```
@using Kutuphane.Models.Entity
@model List<Books>

@{
    ViewBag.Title = "WriterBooks";
    Layout = "~/Views/Shared/Layout.cshtml";
}

<h2 style="text-align:center">@ViewBag.NameSurname Kitapları</h2><br />
<table class="table table-striped">
<tr>
    <th>Id</th>
    <th>Kitap Ad</th>
    <th>Kategori Ad</th>
    <th>Yayınevi</th>
</tr>
<tr>
    @foreach (Books result in Model)
    {
        <tr>
            <td>@result.Id</td>
            <td>@result.Name</td>
            <td>@result.Categories.Name</td>
            <td>@result.Publisher</td>
        </tr>
    }
</table>
```

Şekil 2.28:Writers, WriterBooks.cshtml

Jules Verne Kitapları

Id	Kitap Ad	Kategori Ad	Yayınevi
1	Tuna Kılavuzu	Macera	Yıldız
7	80 Günde Devri Alem	Macera	Plüton
1002	Doktor Oxun Deneyi	Fantastik	Mars

Şekil 2.29:Yazar Kitapları Tasarım

Kitaplar

Kategoride yapılan işlemler kitaplar için de yapıldı. Farklı olarak Kitap ekleme kısmı pop-up değil de yeni bir View türeterek (Şekil 2.30) yapıldı ve kategori ve yazar açılır listeden seçilir hale getirildi (Şekil 2.31) . Ayrıca Books controllerında bulunan Index.cshtml 'in içinde basit bir if şart bloğuyla kitabın altında alınmış ise kırmızı alınmamış ise yeşil renk yanması sağlandı (Şekil 2.32). Kitap fotoğrafları ve projede kullanılan dinamik her fotoğraf, depolamadan tasarruf edilebilmesi için dosya olarak değil de kısa link şeklinde tutuldu.


```
@model Kutuphane.Models.Entity.Books
{
    ViewBag.Title = "BookAdd";
    Layout = "~/Views/Shared/Layout.cshtml";

    <form class="form-group" method="post">
        <div></div>
        <div style="margin-top:15px">
            <label><b>Kitap Ad</b></label>
            @Html.TextBoxFor(p => p.Name, new { @class = "form-control" })
        </div>
        <div style="margin-top:15px">
            <label><b>Kategori</b></label>
            @Html.DropDownListFor(p => p.Categories.Id, (List<SelectListItem>)ViewBag.result, new { @class = "form-control" })
        </div>
        <div style="margin-top:15px">
            <label><b>Yazar</b></label>
            @Html.DropDownListFor(p => p.Writers.Id, (List<SelectListItem>)ViewBag.result2, new { @class = "form-control" })
        </div>
        <div style="margin-top:15px">
            <label><b>Basım yılı</b></label>
            @Html.TextBoxFor(p => p.Year, new { @class = "form-control" })
        </div>
        <div style="margin-top:15px">
            <label><b>Yayınevi</b></label>
            @Html.TextBoxFor(p => p.Publisher, new { @class = "form-control" })
        </div>
        <div style="margin-top:15px">
            <label><b>Sayfa sayısı</b></label>
            @Html.TextBoxFor(p => p.NumberOfPages, new { @class = "form-control" })
        </div>
        <div style="margin-top:15px">
            <label><b>Kitap Fotoğraf Linki</b></label>
            @Html.TextBoxFor(p => p.Image, new { @class = "form-control" })
            <br />
        </div>
        <div class="modal-footer">
            <button class="btn btn-success">Ekle</button>
        </div>
        <input type="hidden" name="Active" value="True" />
    </form>
}
```

Şekil 2.30:Books, BooksAdd.cshtml

Kitap Ad

Kategori

Macara

Yazar

Jules Verne

Basım yılı

Yayınevi

Sayfa sayısı

Kitap Fotoğraf Linki

Ekle

Şekil 2.31:Kitap Ekle Tasarım

```
[HttpGet]
0 references
public ActionResult BookAdd()
{
    List<SelectListItem> category = (from i in kutuphaneEntities.Categories.ToList()
                                     select new SelectListItem
                                     {
                                         Text = i.Name,
                                         Value = i.Id.ToString()
                                     }).ToList();

    ViewBag.result = category;
    List<SelectListItem> writer = (from i in kutuphaneEntities.Writers.ToList()
                                    select new SelectListItem
                                    {
                                        Text = i.Name + ' ' + i.Surname,
                                        Value = i.Id.ToString()
                                    }).ToList();

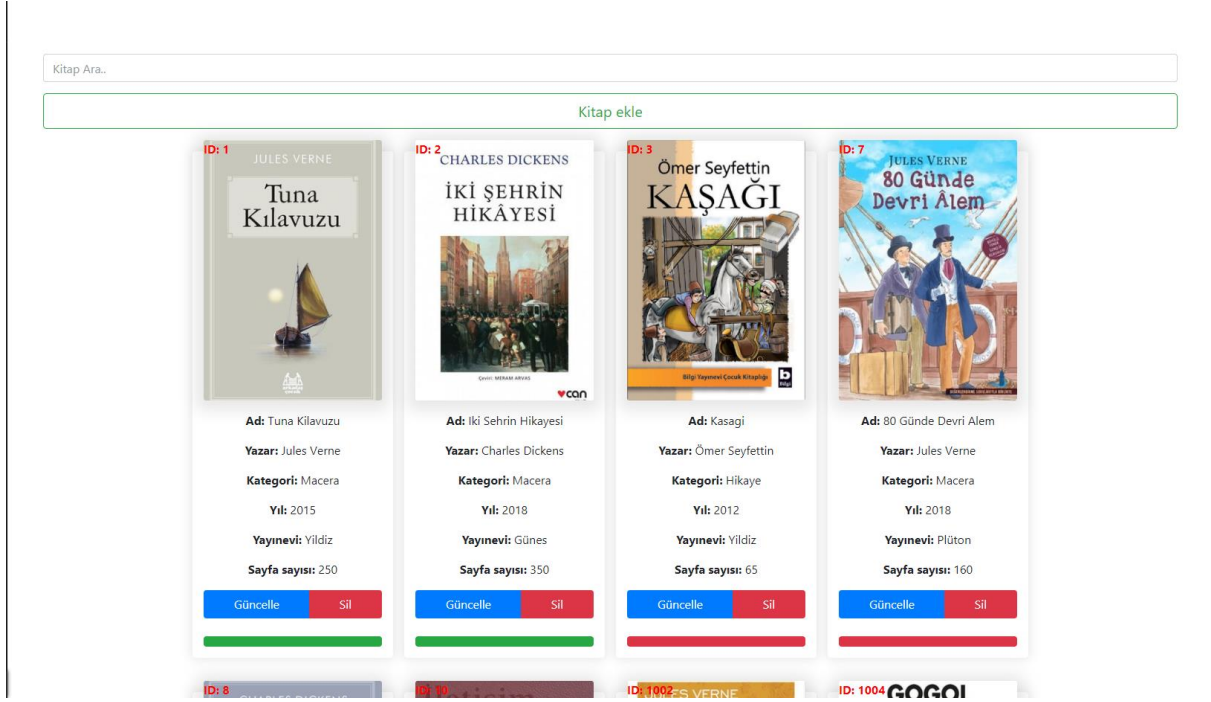
    ViewBag.result2 = writer;
    return View();
}
```

Şekil 2.32:BooksController, BookAdd

```
foreach (Books result in Model)
{
    if (result.Active == true)
    {
        astatus = "btn btn-success";
        spanstatus = "bg-success";
    }
    else
    {
        astatus = "btn btn-danger";
        spanstatus = "bg-danger";
    }

    <div class="col-3">
        <div class="card c">
            <div class="top-sec" style="position:relative">
                <p style="position:absolute; top:0px; color:red"><b>ID:</b></p>
                
            </div>
            <div class="bottom-sec" style="text-align:center">
                <p><b>Ad:</b> @result.Name</p>
                <p><b>Yazar:</b> @result.Writers.Name @result.Writers.Surname</p>
                <p><b>Kategori:</b> @result.Categories.Name</p>
                <p><b>Yıl:</b> @result.Year</p>
                <p><b>Yayınevi:</b> @result.Publisher</p>
                <p><b>Sayfa sayısı:</b> @result.NumberOfPages</p>
            </div>
            <div class="btn-group text-white role="group" aria-label="Basic mixed styles example">
                <a href="/Books/FetchBook/@result.Id" class="btn btn-primary">Güncelle</a>
                <Html.ActionLink("Sil", "BookDelete", new { id = result.Id }, new { @class = "btn btn-danger", onclick = "return confirm('Silmek istiyor musunuz?')"} )>
            </div>
            <br />
            <a>@astatus</a>
            <span class="@spanstatus">
            </span>
        </a>
    </div>
</div>
</div>
```

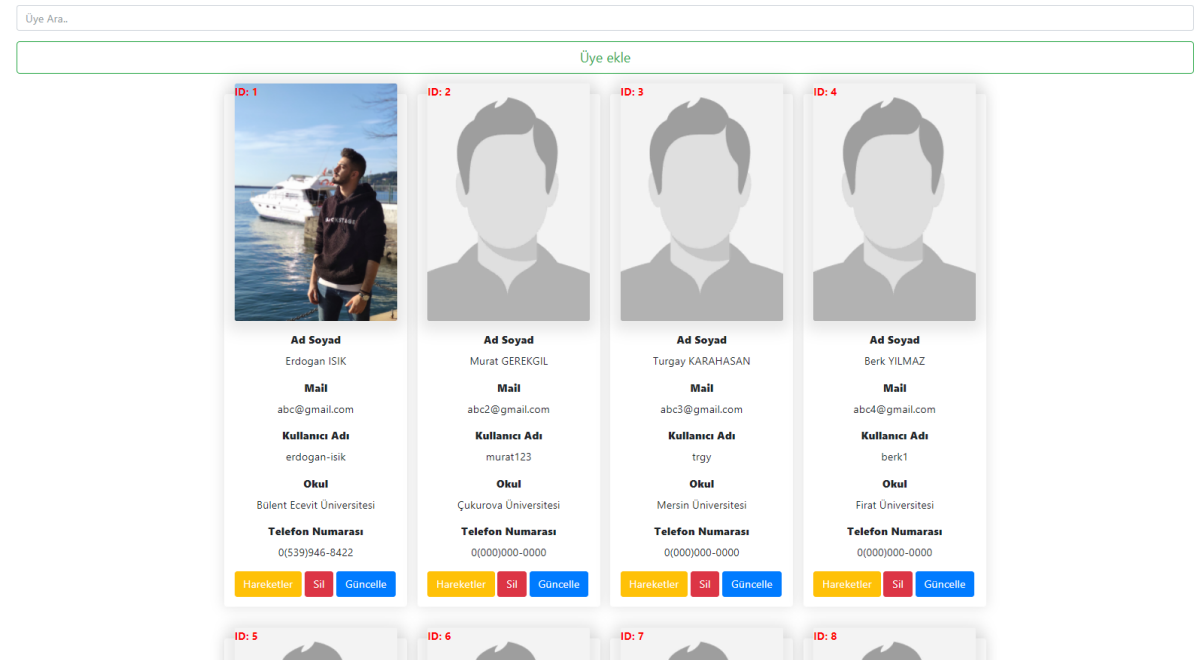
Şekil 2.33:Books, Index.cshtml



Şekil 2.34: Kitaplar Tasarım

Üyeler

Kategoride yapılan işlemler üyeler için de yapıldı.



Şekil 2.35: Üyeler Tasarım



Şekil 2.38:Üye Hareketler Tasarım

Personeller

Kategoride yapılan işlemler personeller için de yapıldı.

Personel Ara...

Personel ekle

 Sadi MERMER id: 1 Güncelle Sil	 Mahmut SEFER id: 2 Güncelle Sil	 Enes TOPLU id: 3 Güncelle Sil	 Mehmet ÇINAR id: 4 Güncelle Sil
 Hakan TOPÇU id: 5 Güncelle Sil	 Mehmet Erdem ISIK id: 1002 Güncelle Sil		

1

Şekil 2.39:Personeller Tasarım

Ödünç Kitap Ver

LoansController oluşturuldu. Kitap eklemede yapılan işlemler Ödünç Kitap Ver için de yapıldı.

Üye
Erdogan ISIK

Kitap
Tuna Kilavuzu

Personel
Sadi MERMER

Kitap alış tarihi
27.06.2021

Kitap iade tarihi
4.07.2021

Ödünç ver

Şekil 2.40:Ödünç Kitap Ver Tasarım

ADMİN PANELİ

Ödünç Kitap Al

LoansController da bulunan Index'e CategoryController da ki Index de yapılan işlemler ödünç alma için de yapıldı (Şekil 2.41).

Ara...						
Id	Ad	Üye	Personel	Alış Tarihi	Zorunlu İade Tarihi	İşlem
4009	Aylak Adam	Seval ISIK	Mahmut SEFER	22.06.2021	29.06.2021	Teslim et
4007	Kasagi	Emel YILDIRIM	Mahmut SEFER	22.06.2021	29.06.2021	Teslim et
2008	80 Günde Devri Alem	Erdogan ISIK	Enes TOPLU	17.06.2021	24.06.2021	Teslim et
2007	Geçtiğim Günlerden	Erdogan ISIK	Sadi MERMER	17.06.2021	24.06.2021	Teslim et
5	Müfettis	Murat GEREKGIL	Sadi MERMER	14.06.2021	21.06.2021	Teslim et

Şekil 2.41:Ödünç Kitap Al Tasarım

1-)Teslim Et

Kategorilerden farklı olarak üyelerin aldığı kitapları geri almak için teslim et butonu konuldu. Controller kısmına LoansReturn metot oluşturuldu ve View eklendi. Transactions tablosunda id si gönderilen geçerli üyeye ait alınmış kitap listelendi. Son olarak zorunlu iade tarihinden teslim etme tarihini yani butona tıklandığı tarih çıkartıldı. Bu sayede üyenin geç getirdiyse kaç gün geç getirdiği ViewBag yardımıyla View'e taşındı (Şekil 2.42).

```
0 references
public ActionResult LoanReturn(Transactions transactions)
{
    var result = kutuphaneEntities.Transactions.Find(transactions.Id);
    DateTime date1 = DateTime.Parse(result.ReturnDate.ToString());
    DateTime date2 = Convert.ToDateTime(DateTime.Now.ToShortDateString());
    TimeSpan date3 = date2 - date1;
    ViewBag.toValue = date3.TotalDays;
    return View("LoanReturn", result);
}
```

Şekil 2.42:LoansController, LoanReturn

ViewBag ile taşınan geç kaldığı gün sayısı eğer erken getirmiş ise "-" bir değer dönmemesi için if şart bloğu yazıldı (Şekil 2.43).

```
@model Kutuphane.Models.Entity.Transactions
@{
    ViewBag.Title = "LoanReturn";
    Layout = "~/Views/Shared/Layout.cshtml";
    int toValue;
}

using (Html.BeginForm("LoanReturnUpdate", "Loans", FormMethod.Post))
{
    if (@ViewBag.toValue > 0)
    {
        toValue = (int)@ViewBag.toValue;
    }
    else
    {
        toValue = 0;
    }

    <div style="margin-top:15px"></div>
    <div class="form-group">
        <label><b>Id</b></label>
        <input type="text" value="@ViewBag.Id" class="form-control" />
        <br />
        <label><b>Ad</b></label>
        <input type="text" value="@ViewBag.Ad" class="form-control" />
        <br />
        <label><b>Üye</b></label>
        <input type="text" value="@ViewBag.MemberName" class="form-control" />
        <br />
        <label><b>Personel</b></label>
        <input type="text" value="@ViewBag.StaffNameSurname" class="form-control" />
        <br />
        <label><b>Alış tarihi</b></label>
        <input type="text" value="@ViewBag.PurchaseDate" class="form-control" />
        <br />
        <label><b>Son iade tarihi</b></label>
        <input type="text" value="@ViewBag.ReturnDate" class="form-control" />
        <br />
        <label><b>Üyenin iade ettiği tarih</b></label>
        <input type="text" value="@ViewBag.MemberReturnDate" class="form-control" />
        <br />
        <label><b>Son iade tarihi aşımı</b></label>
        <input type="text" value="@ViewBag.BeLate" class="form-control" />
    </div>
    <div class="modal-footer">
        <button class="btn btn-group" style="background-color:#ff8500;color:white">Kitabı geri al</button>
    </div>
}
```

Şekil 2.43:Loans, LoanReturn.cshtml

Kitabı geri al butonuna tıklandığında çalışacak olan LoanReturnUpdate metodu yazıldı (Şekil 2.44). Üyenin getirdiği tarih yazıldı ve durum true yapıldı (Durum bir sonraki sayfada bulunan işlemlerde işlem durumu true olanları listelemek için kullanılacak). Veri tabanı kaydedildi ve Index' e yönlendirme sağlandı.

```
0 references
public ActionResult LoanReturnUpdate(Transactions transactions)
{
    var result = kutuphaneEntities.Transactions.Find(transactions.Id);
    result.MemberReturnDate = transactions.MemberReturnDate;
    result.Status = true;
    kutuphaneEntities.SaveChanges();
    return RedirectToAction("Index");
}
```

Şekil 2.44:Loans, LoanReturnUpdate

ADMIN PANELİ

Id	5
Ad	Müfettis
Üye	Murat
Personel	Sadi MERMER
Alış tarihi	14.06.2021
Son lade tarihi	21.06.2021
Üyenin lade ettiği tarih	28.06.2021
Son lade tarihi aşımı	7 Gün

Kitabı geri al

Şekil 2.45: Teslim Et Tasarım

İşlemler

TransactionsController oluşturuldu View eklendi ve Transactions tablosunda durumu true olan tüm işlemler listelendi. Category Index de yapılan işlemler Transactions için de yapıldı.

Tüm Hareketler

Ara...

Id	Ad	Üye	Personel	Alış Tarihi	Üye İade Tarihi
4008	İki Sehrin Hikayesi	Seval İSİK	Hakan TOPÇU	22.06.2021	22.06.2021
3008	Yalnız Efe	Erdogan İSİK	Enes TOPLU	20.06.2021	20.06.2021
3007	Yalnız Efe	Rümeysa DAGDELEN	Mehmet ÇINAR	20.06.2021	20.06.2021
2009	Kasagi	Rümeysa DAGDELEN	Mahmut SEFER	17.06.2021	17.06.2021
1007	Yalnız Efe	Murat GEREKGİL	Mahmut SEFER	15.06.2021	15.06.2021
1006	Yalnız Efe	Berk YILMAZ	Mehmet ÇINAR	15.06.2021	15.06.2021
1005	Yalnız Efe	Berk YILMAZ	Mahmut SEFER	15.06.2021	15.06.2021
6	Vurun Kahpeye	Aylin ÖZTÜRK	Sadi MERMER	14.06.2021	14.06.2021
4	Doktor Oxun Deneyi	Emel YILDIRIM	Enes TOPLU	14.06.2021	14.06.2021
3	İki Sehrin Hikayesi	Mehmet GÜMBÜR	Mahmut SEFER	14.06.2021	14.06.2021

1

Şekil 2.46:İşlemler Tasarım

Stajı Onaylayan Mühendisin
Adı Soyadı:

İmzası:

Tarih: ... / ... / 20 ...

ADMİN PANELİ

Duyurular

Yazarlar için yapılan işlemler duyurular için de yapıldı.

Duyuru ekle

Yeni Kitap

Id: 5

Tarih: 18.06.2021

Güncelle

Sil

İçerik

Diğer

Id: 3

Tarih: 10.06.2021

Güncelle

Sil

İçerik

Kitapları Geciktirmeyin

Id: 2

Tarih: 15.06.2021

Güncelle

Sil

İçerik

Yeni Kitap

Id: 1

Tarih: 18.06.2021

Güncelle

Sil

İçerik

Şekil 2.47:Duyurular Tasarım

Ayarlar

Kategoriler için yapılan işlemler ayarlar için de yapıldı.

Admin ekle

erdogani0202@gmail.com

Yetki: A

id: 1

Güncelle

Sil

admin2@gmail.com

Yetki: B

id: 2

Güncelle

Sil

admin3@gmail.com

Yetki: B

id: 3

Güncelle

Sil

admin4@gmail.com

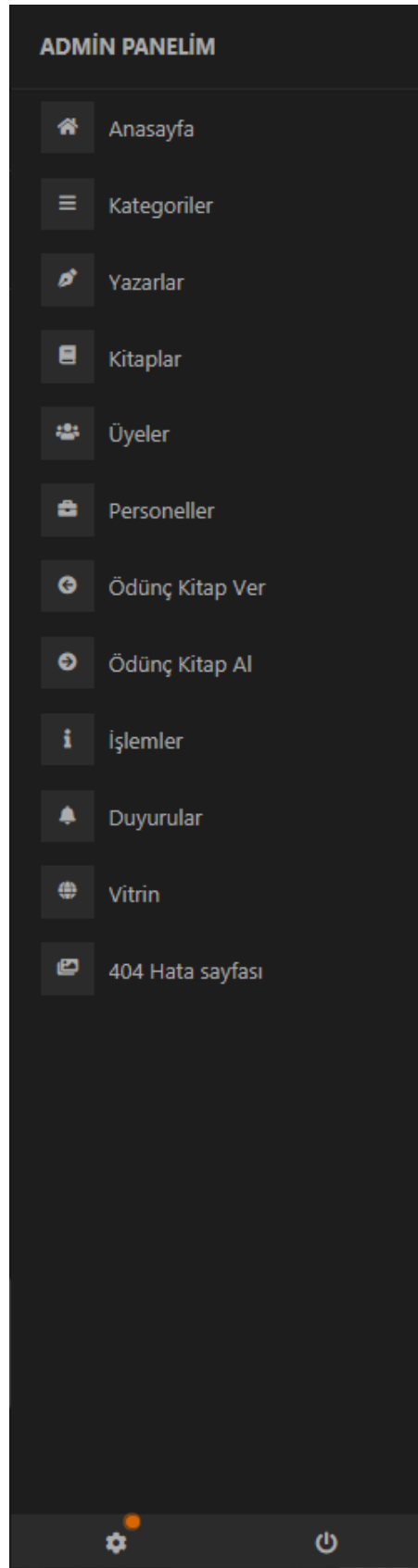
Yetki: B

id: 2002

Güncelle

Sil

Şekil 2.48:Ayarlar Tasarım

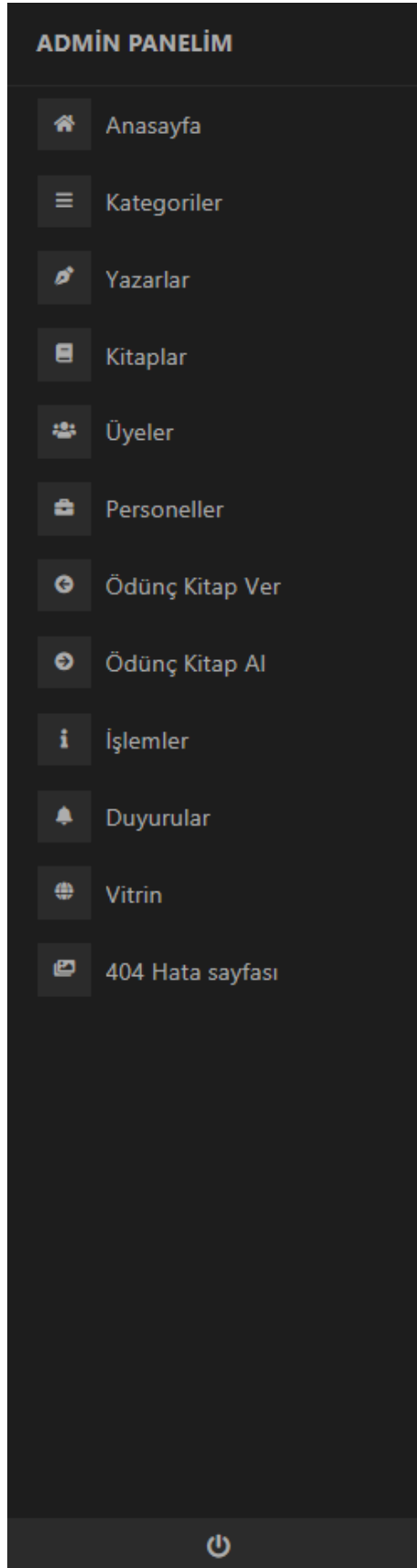


Şekil 2.49:A Yetkili Admin

Stajı Onaylayan Mühendisin
Adı Soyadı:

İmzası:

Tarih: ... / ... / 20 ...



Şekil 2.50:B Yetkili Admin

Stajı Onaylayan Mühendisin
Adı Soyadı:

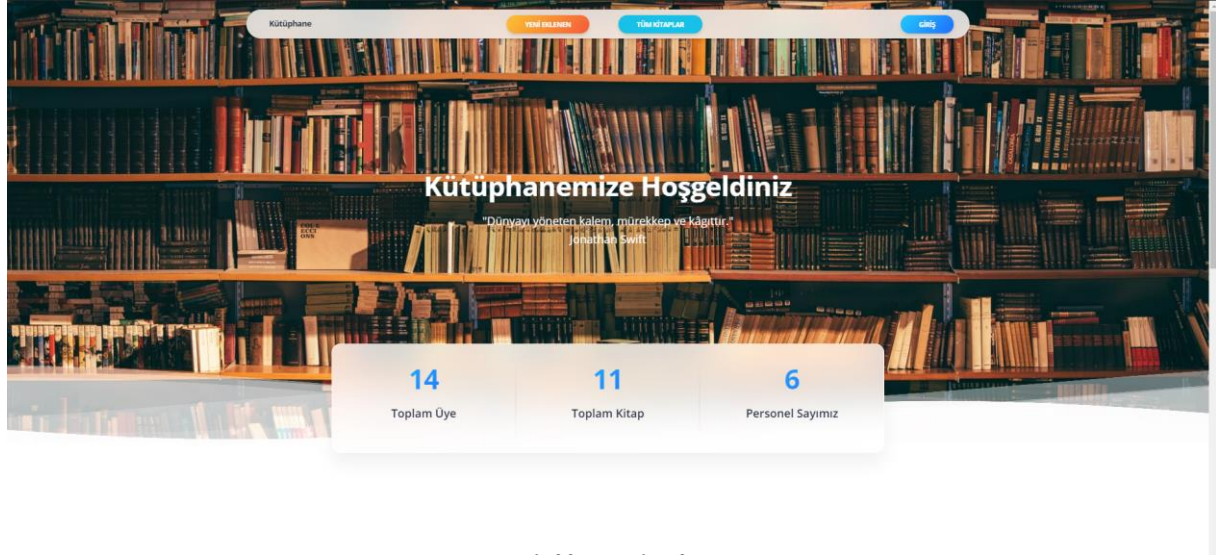
İmzası:

Tarih: ... / ... / 20 ...

VİTRİN

ShowcaseLayout

Views/Shared klasörünün altına yeni bir ShowcaseLayout.cshtml adında bir layout oluşturuldu. Bu layout vitrin de değişmeyecek olan ana kısmını oluşturmaktadır (Şekil 3). Admin layout da yapılan işlemler bu layout için de yapıldı.

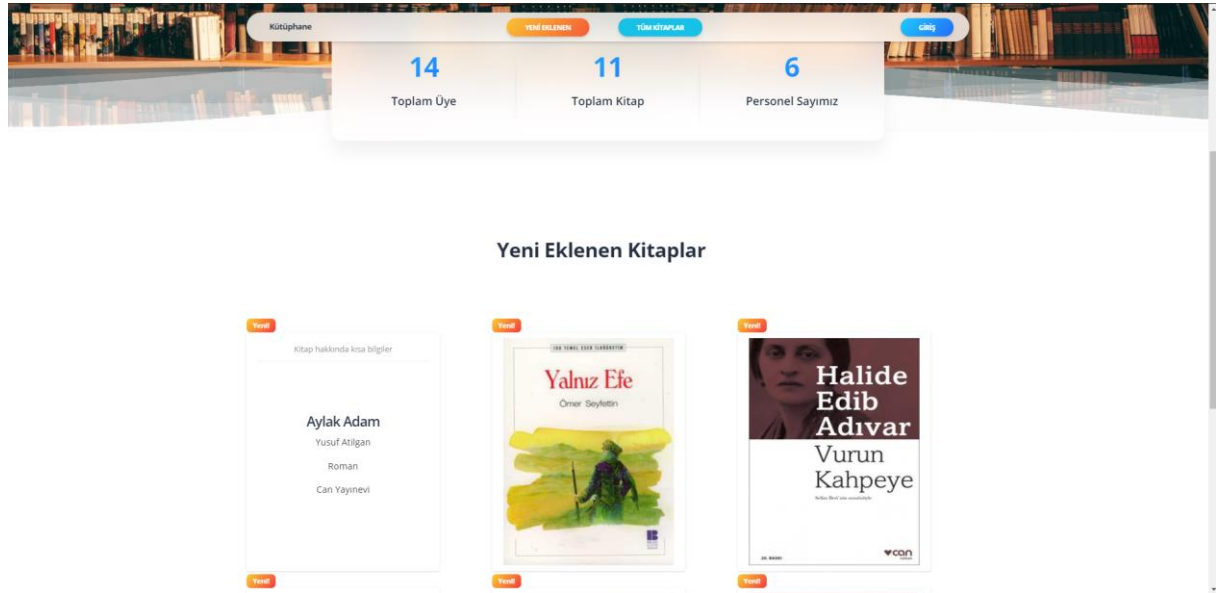


Şekil 3:Vitrin Tasarım

ShowcaseController oluşturuldu.

Yeni Eklenen

Kitap, üye ve personel sayısı ViewBag ler yardımıyla Index'e taşındı ve Index de ki foreach döngüsünde Books tablosuna eklenen son 6 kitap listelendi (Şekil 3.1).

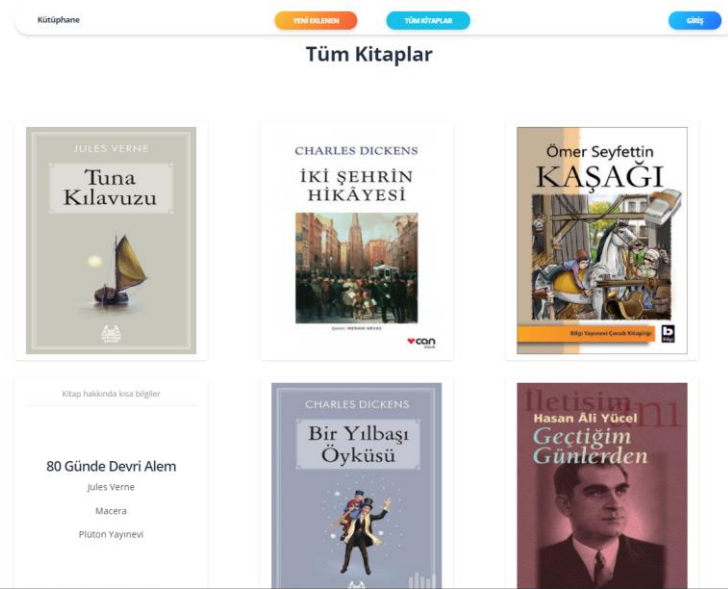


Şekil 3.1:Yeni Eklenen Tasarım

VİTRİN

Tüm Kitaplar

Tüm kitaplar listelendi (Şekil3.2).

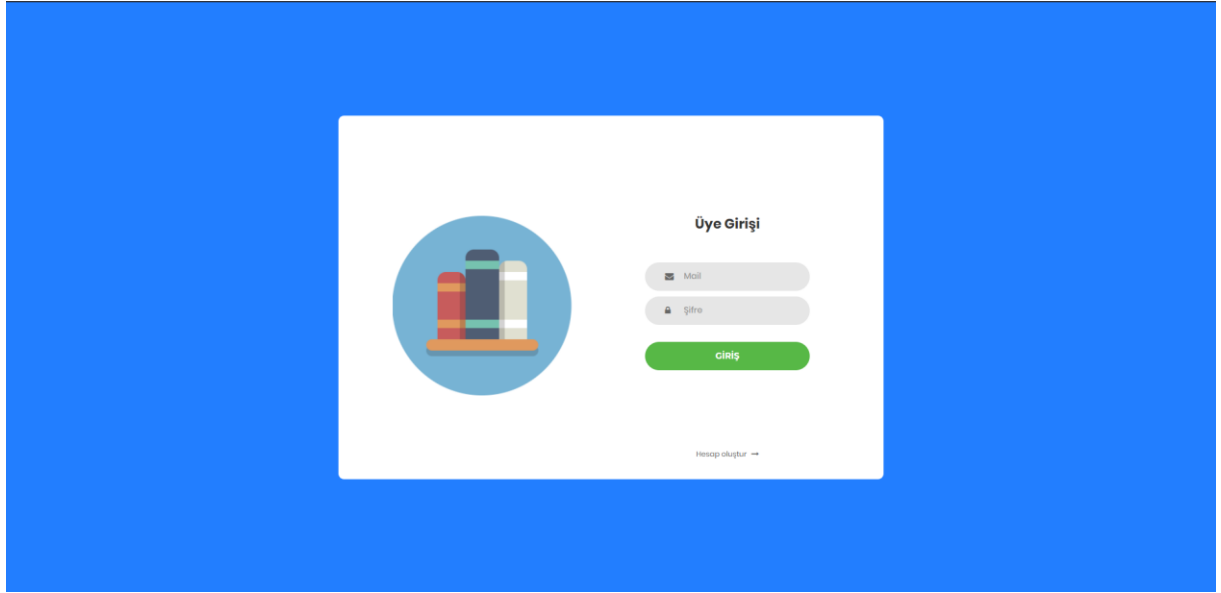


Şekil 3.2:Tüm Kitaplar Tasarım

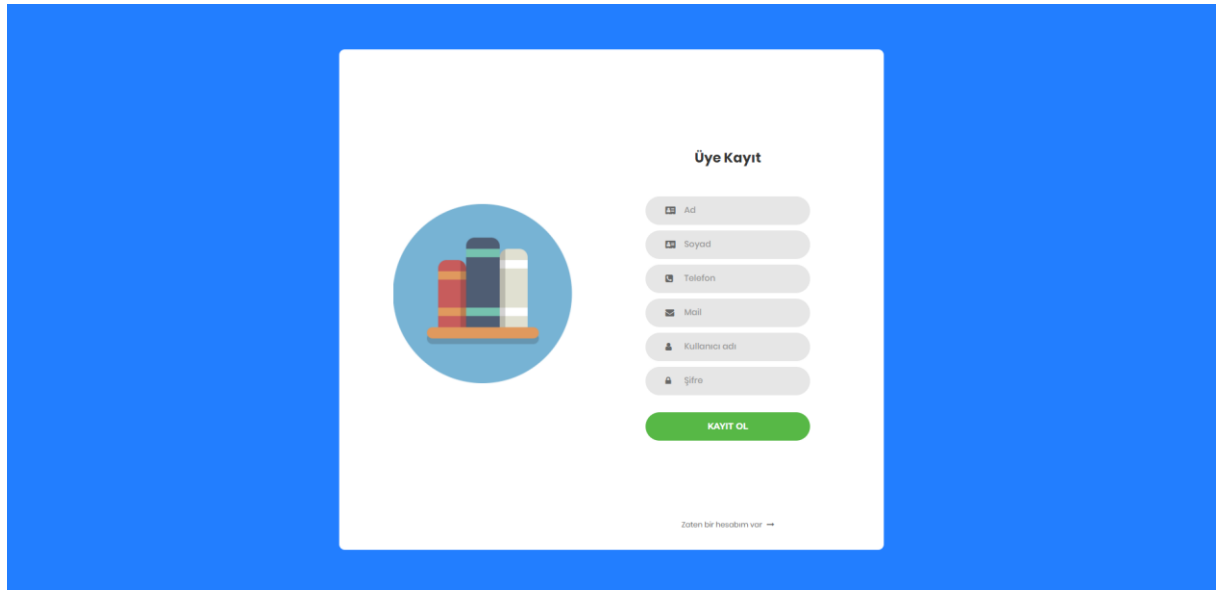
KULLANICI PANELİ

Kullanıcı Login-Register

Admin login de yapılan işlemler kullanıcı için de yapıldı (Şekil 4,4.1).



Şekil 4:Kullanıcı Login Tasarım

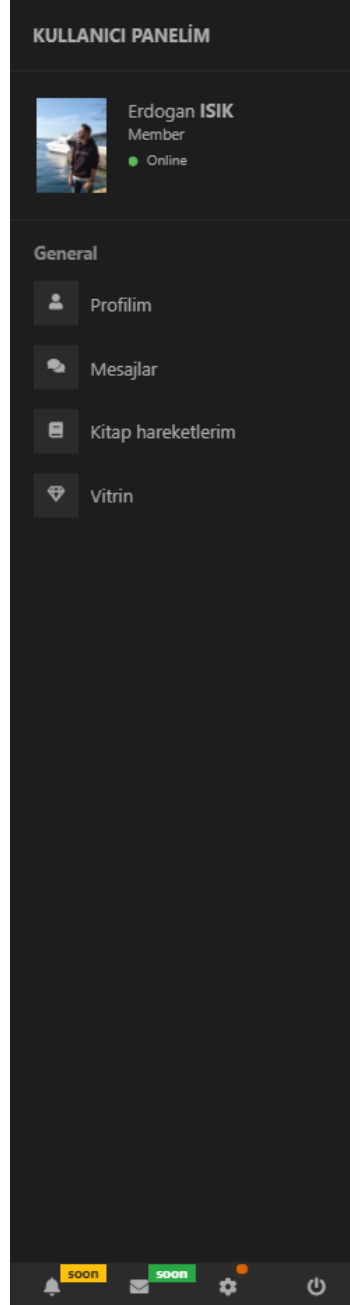


Şekil 4.1:Kullanıcı Register Tasarım

KULLANICI PANELİ

Membersdashboard

Views/Shared klasörünün altına yeni bir Membersdashboard.cshtml adında bir layout oluşturuldu. Bu layout kullanıcı panelinde değişmeyecek olan ana kısmını oluşturmaktadır (Şekil 4.2). Admin layout da yapılan işlemler bu layout için de yapıldı.



Şekil 4.2:Kullanıcı Menü Tasarım

Giriş yapan kullanıcının adı soyadı ve fotoğrafı Session'lar yardımıyla taşındı.

KULLANICI PANELİ

Profilim

MemberdashboardController oluşturuldu. Home metodu ile kullanıcının tüm bilgilerini ViewBag ler yardımıyla Home View'ine taşındı. Diğer sayfa yapılarından farklı olarak tek sayfada iki farklı sayfa yapısı kullanıldığından (Duyurular ve Ayarlar) dolayı Home.cshtml 'de Partial View kullanıldı (**Şekil 4.3**). Partial1 (Duyurular) ve Partial2 (Ayarlar) metodları yazıldı (**Şekil 4.4**) ve bu partialların tasarım kodları yazıldı. Ayalar kısmında ki güncelle butonunun işlevini yerine getirebilmesi için http post olarak HomeProfile metodu oluşturuldu (**Şekil 4.5**).

```
<div class="col-md-9">
  <div class="card">
    <div class="card-header p-2">
      <ul class="nav nav-pills">
        <li class="nav-item"><a class="nav-link active" href="#activity" data-toggle="tab">Duyurular</a></li>
        <li class="nav-item"><a class="nav-link" href="#settings" data-toggle="tab">Ayarlar</a></li>
      </ul>
    </div><!-- /.card-header -->
    <div class="card-body">
      <div class="tab-content">
        <div class="active tab-pane" id="activity">
          <!-- Post -->
          @Html.Partial("Partial1")
          <!--Post-->
        </div>
        <div class="tab-pane" id="settings">
          @Html.Action("Partial2", "Memberdashboard")
        </div>
      <!-- /.tab-pane -->
    </div>
  </div>
</div>
```

Şekil 4.3:Home.cshtml

KULLANICI PANELİ

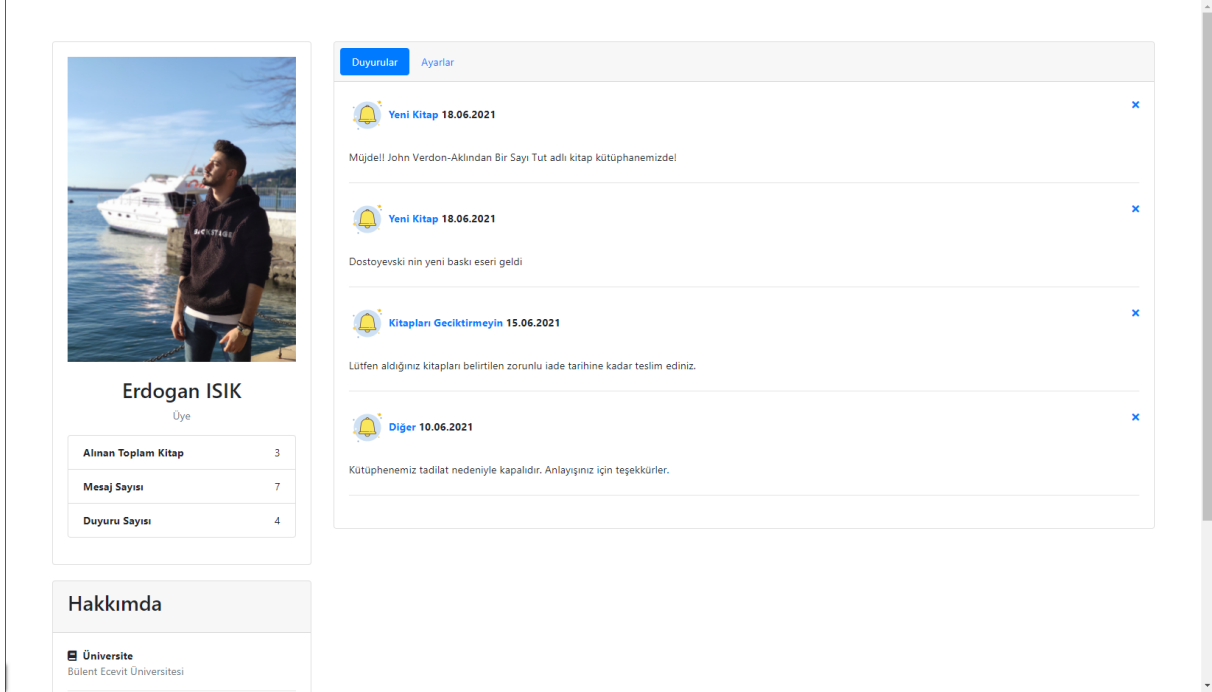
```
0 references
public PartialViewResult Partial1()
{
    return PartialView();
}

0 references
public PartialViewResult Partial2()
{
    var mail = (string)Session["Mail"];
    var memberId = kutuphaneEntities.Members.Where(p => p.Mail == mail).Select(p => p.Id).FirstOrDefault();
    var result = kutuphaneEntities.Members.Find(memberId);
    return PartialView("Partial2", result);
}
```

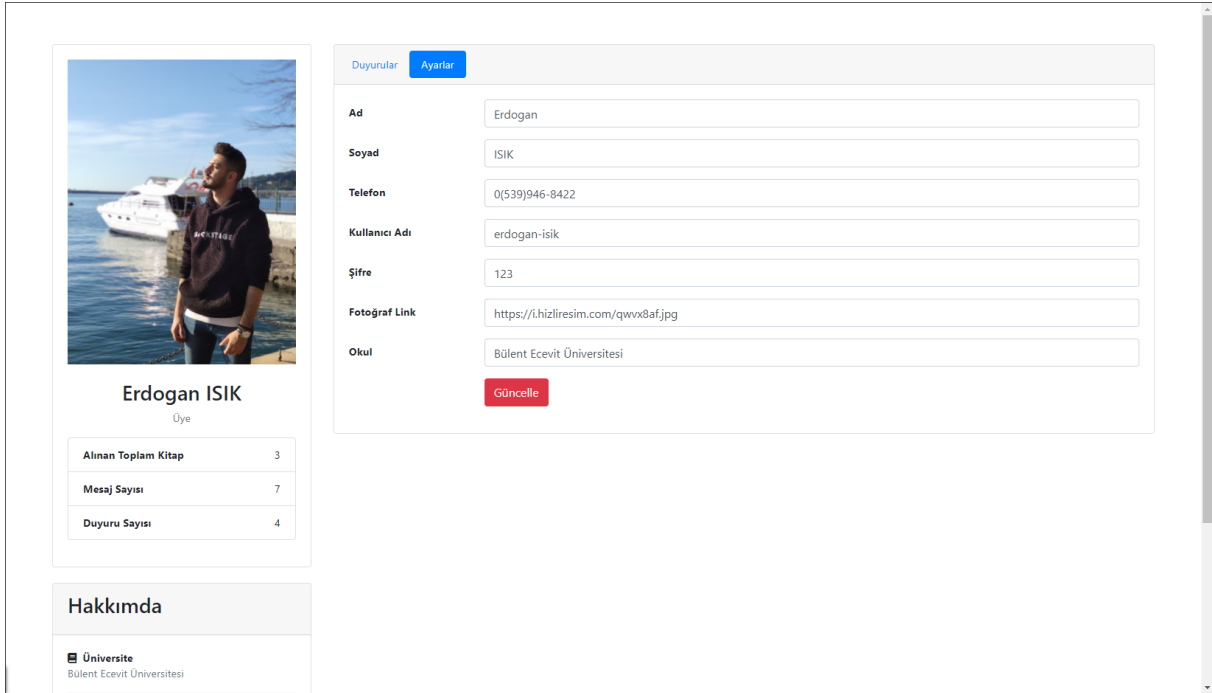
Şekil 4.4:Memberdashboard, Partial1-Partial2

```
[HttpPost]
0 references
public ActionResult HomeProfile(Members members)
{
    if (!ModelState.IsValid)
    {
        return View("Home");
    }
    var mail = (string)Session["Mail"];
    var user = kutuphaneEntities.Members.FirstOrDefault(p => p.Mail == mail);
    user.Password = members.Password;
    user.Name = members.Name;
    user.Surname = members.Surname;
    user.PhoneNumber = members.PhoneNumber;
    user.UserName = members.UserName;
    user.School = members.School;
    user.Image = members.Image;
    kutuphaneEntities.SaveChanges();
    return RedirectToAction("Home");
}
```

Şekil 4.5:Memberdashboard, HomeProfile



Şekil 4.6:Profilim, Duyurular Tasarım

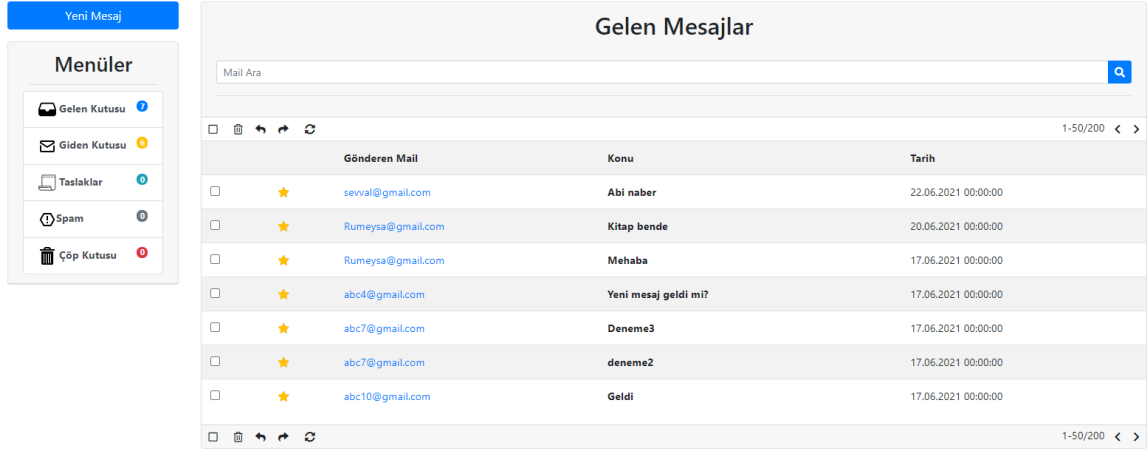


Şekil 4.7:Profilim, Ayarlar Tasarım

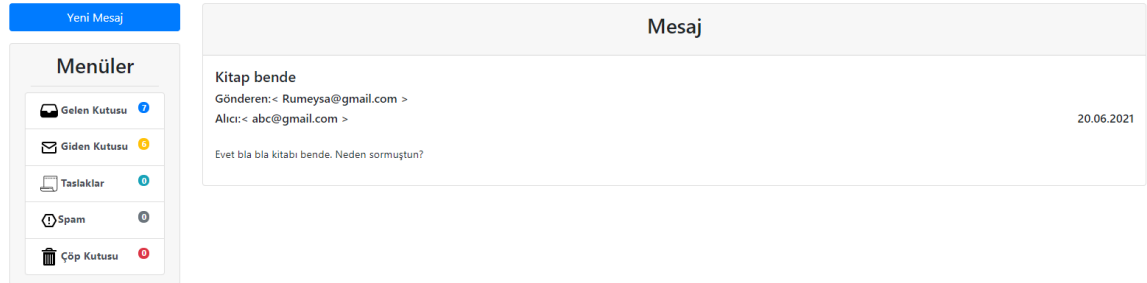
KULLANICI PANELİ

Mesajlar

MessagesController oluşturuldu. Profilimde yapılan işlemler mesajlar için de yapıldı.



Şekil 4.8:Mesajlar Tasarım



Şekil 4.9:Mesaj Okuma Tasarım

KULLANICI PANELİ

Yeni Mesaj Oluştur

Kime:

Konu:

İçerik:

Gönder

Şekil 4.10:Yeni Mesaj Tasarım

Kitap Hareketlerim

MemberdashboardControllera BooksAction metodu (Şekil 4.11) yazıldı ve View oluşturularak tasarım kodları yazıldı.

```
0 references
public ActionResult BooksAction()
{
    var mail = (string)Session["Mail"];
    var id = kutuphaneEntities.Members.Where(p => p.Mail == mail.ToString()).Select(p => p.Id).FirstOrDefault();
    var result = kutuphaneEntities.Transactions.Where(p => p.MemberId == id).OrderByDescending(p => p.Id).ToList();
    return View(result);
}
```

Şekil 4.11:Memberdashboard, BooksAction

Kitap Hareketlerim

20.06.2021	→	20.06.2021	→	✓
Yalnız Efe - Ömer Seyfettin				
17.06.2021	→	Teslim Edilmedi	→	⚠
80 Günde Devri Alem - Jules Verne				
17.06.2021	→	Teslim Edilmedi	→	⚠
Geçtiğim Günlerden - Hasan Ali Yücel				

Şekil 4.12:Kitap Hareketlerim Tasarım

Stajı Onaylayan Mühendisin
Adı Soyadı:

İmzası:

Tarih: ... / ... / 20 ...

KULLANICI PANELİ

Vitrin

Tıklandığında Vitrine yönlendirme sağlandı.

Son olarak çıkış işlemi için MemberdashboardControllera LogOut metodu yazıldı (**Şekil 4.13**) ve oturum kapatma tuşuna bu metodun yönlendirilmesi yapıldı.

```
0 references
public ActionResult Logout()
{
    FormsAuthentication.SignOut();
    return RedirectToAction("Login","Login");
}
```

Şekil 4.13:Memberdashboard, Logout

Error Sayfası

Error sayfasını özelleştirebilmek için ilk önce ErrorController oluşturuldu ve hata kodlarına göre (Örn. 404) metodlar yazıldı (Şekil 5). Yazılan metodlara View eklendi ve bu View'lere hata sayfasının tasarım kodları yazıldı (Şekil 5.1). Son olarak Web.config ayarlaması yapıldı (Şekil 5.2).

```
[AllowAnonymous]
0 references
public class ErrorController : Controller
{
    // GET: Error
    0 references
    public ActionResult Page400()
    {
        Response.StatusCode = 400;
        Response.TrySkipIisCustomErrors = true;
        return View();
    }

    0 references
    public ActionResult Page403()
    {
        Response.StatusCode = 403;
        Response.TrySkipIisCustomErrors = true;
        return View();
    }

    0 references
    public ActionResult Page404()
    {
        Response.StatusCode = 404;
        Response.TrySkipIisCustomErrors = true;
        return View();
    }
}
```

Şekil 5:ErrorController

Error Sayfası

```
Layout = null;

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">

<title>404-Sayfa bulunamadı</title>
<link href="#" rel="shortcut icon" />
<!-- Google font -->
<link href="https://fonts.googleapis.com/css?family=Montserrat:400,700,900" rel="stylesheet">

<!-- Custom stylesheet -->
<link type="text/css" rel="stylesheet" href="~/Error404/css/style.css" />
</head>
<body>
<div id="notfound">
<div class="notfound">
<div class="notfound-404">
<h1>Oops!</h1>
</div>
<h2>404 - Sayfa bulunamadı</h2>
<p>Aradığınız sayfanın adı değiştirilmiş veya geçici olarak kullanılmıyor olabilir.</p>
<a href="/Showcase/Index/">Anasayfa'ya Geri Dön</a>
</div>
</div>
</body>
</html>
```

Şekil 5.1:Page404.cshtml

```
<customErrors mode="On">
<error statusCode="404" redirect="/Error/Page404/" />
</customErrors>
```

Şekil 5.2:Web.config



Şekil 5.3:Page404 Tasarım