

STUDENT MANAGEMENT SYSTEM

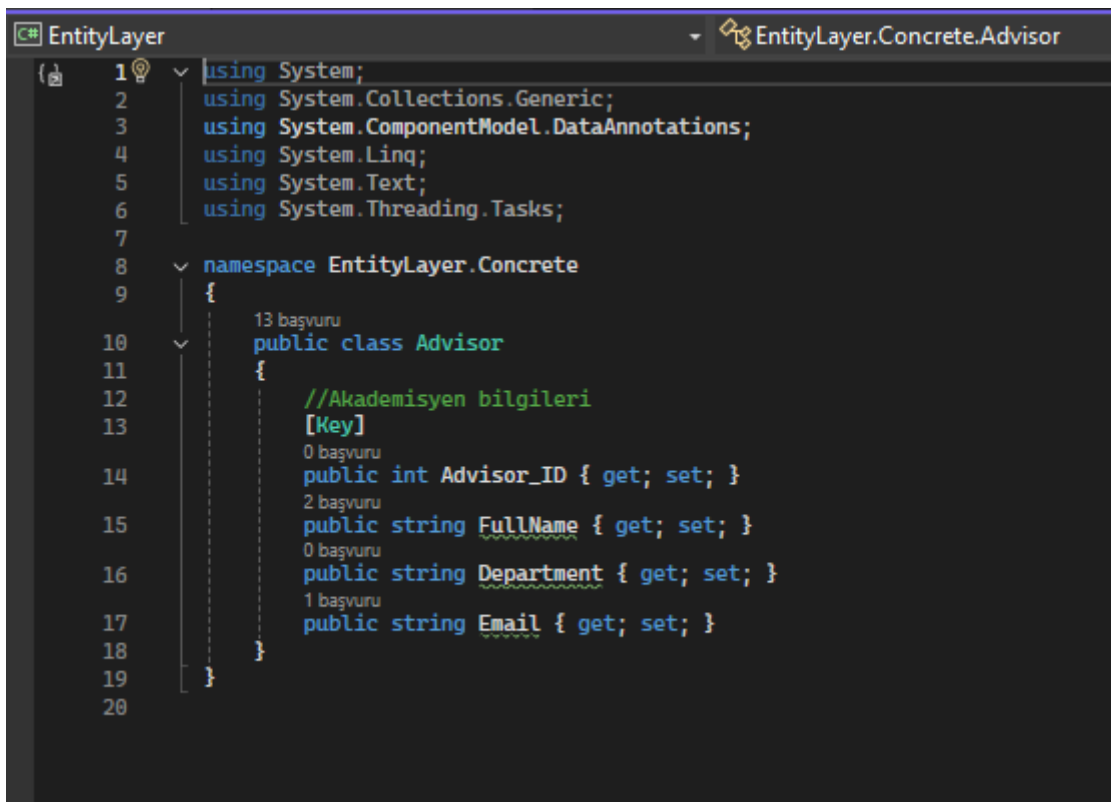
(Core_ManagementSystem)

Çok katmanlı mimari yapısı

- EntityLayer
- DataAccessLayer
- BuisnessLayer
- Core_ManagementSystem

1)EntityLayer'da Tablo İçeriklerinin Oluşturulması

Advisor Bilgileri:



```
EntityLayer EntityLayer.Concrete.Advisor
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel.DataAnnotations;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace EntityLayer.Concrete
9 {
10     public class Advisor
11     {
12         //Akademisyen bilgileri
13         [Key]
14         public int Advisor_ID { get; set; }
15         public string FullName { get; set; }
16         public string Department { get; set; }
17         public string Email { get; set; }
18     }
19 }
20
```

Course Bilgileri:

```
C# EntityLayer EntityLayer.Concrete
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel.DataAnnotations;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace EntityLayer.Concrete
9 {
10     public class Course
11     {
12         //Ders bilgileri
13         [Key]
14         public int Course_ID { get; set; }
15         public string Course_Name { get; set; }
16         public string Course_Codes { get; set; }
17         public int Credits { get; set; }
18         public int Advisor_ID { get; set; }
19         public string Department { get; set; }
20         public bool IsMandatory { get; set; }
21     }
22 }
23
```

Course Selection Bilgileri:

```
C# EntityLayer EntityLayer.Concrete.CourseS
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel.DataAnnotations;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace EntityLayer.Concrete
9 {
10     public class CourseSelection
11     {
12         //Ders seçimi yapma kısmı
13         [Key]
14         public int Selection_ID { get; set; }
15         public int Student_ID { get; set; }
16         public int Course_ID { get; set; }
17         public bool IsApproved { get; set; } //onaylandı-onaylanmadı
18     }
19 }
20
```

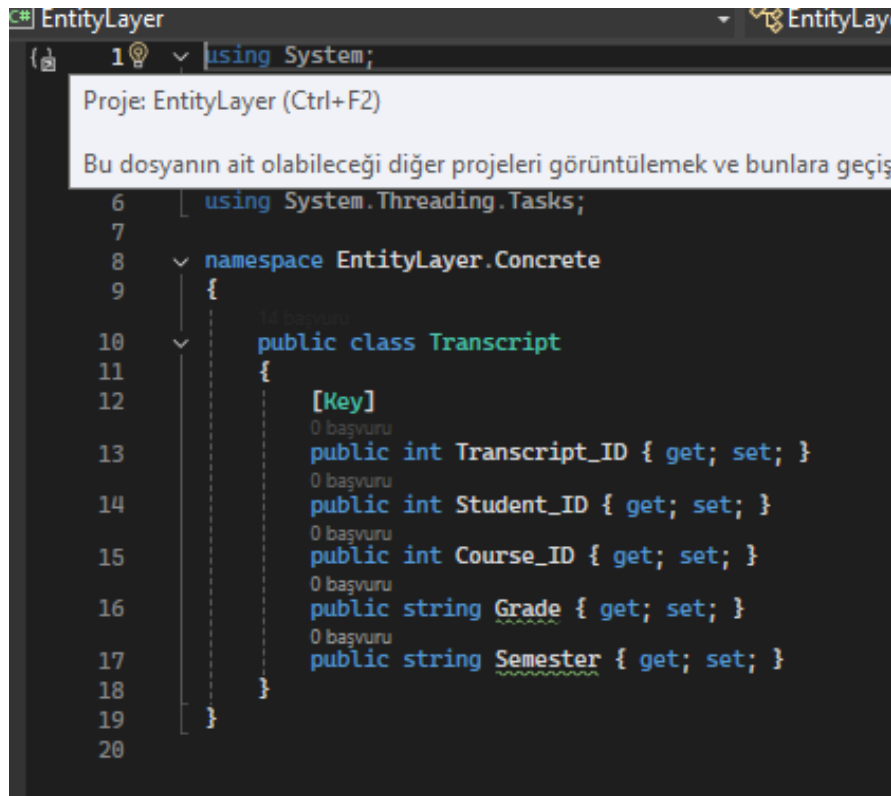
Role Bilgileri:

```
C# EntityLayer EntityLayer.Concrete.Role
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel.DataAnnotations;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace EntityLayer.Concrete
9 {
10     public class Role
11     {
12         //akademisyen ve öğrencilerin bilgi ve görevlerinin açıklamaları
13         [Key]
14         public int Role_ID { get; set; }
15         public string Role_Name { get; set; }
16         public string Description { get; set; }
17     }
18 }
19
```

Student Bilgileri:

```
C# EntityLayer EntityLayer.Co
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel.DataAnnotations;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace EntityLayer.Concrete
9 {
10     public class Student
11     {
12         //Temada öğrenci bilgileri
13         [Key]
14         public int Student_ID { get; set; }
15         public string FirstName { get; set; }
16         public string LastName { get; set; }
17         public string Email { get; set; }
18         public string Department { get; set; }
19         public DateTime EnrollmentDate { get; set; }
20         public int Advisor_ID { get; set; }
21     }
22 }
23
```

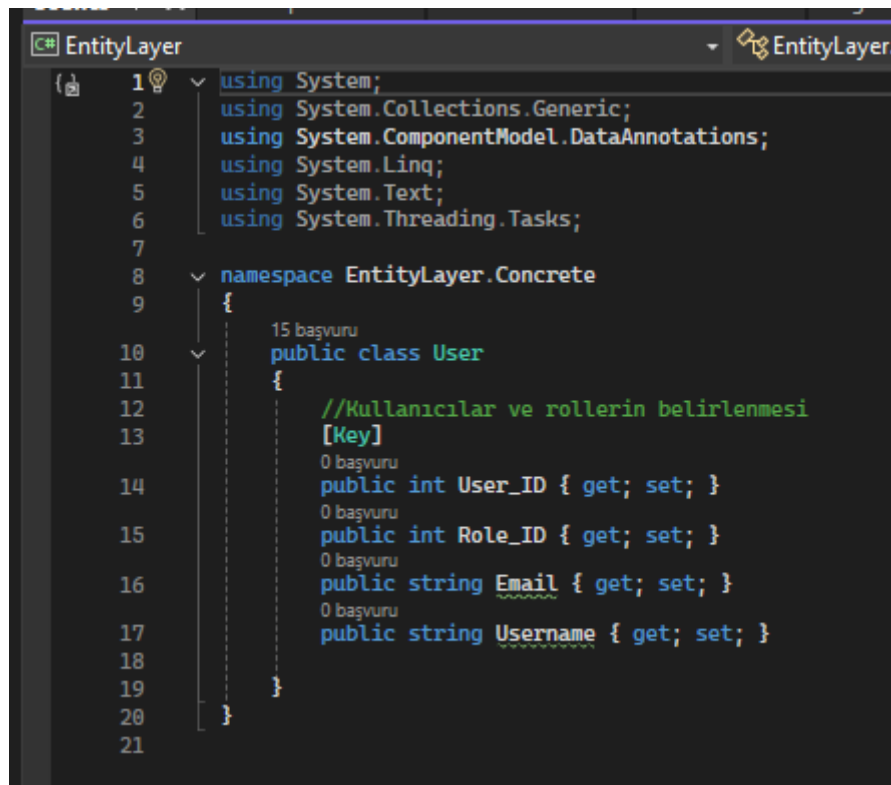
Transcript Bilgileri:



The screenshot shows the EntityLayer project in Visual Studio. A tooltip is visible at the top, indicating the project name and a keyboard shortcut. The code defines a public class Transcript within the EntityLayer.Concrete namespace. The class has a primary key attribute [Key] and six public properties: Transcript_ID, Student_ID, Course_ID, Grade, and Semester, each with get and set methods. The code is as follows:

```
1 using System;
2
3 namespace EntityLayer
4 {
5     14 başvuru
6     public class Transcript
7     {
8         [Key]
9         0 başvuru
10        public int Transcript_ID { get; set; }
11        0 başvuru
12        public int Student_ID { get; set; }
13        0 başvuru
14        public int Course_ID { get; set; }
15        0 başvuru
16        public string Grade { get; set; }
17        0 başvuru
18        public string Semester { get; set; }
19    }
20 }
```

User Bilgileri:



The screenshot shows the EntityLayer project in Visual Studio. The code defines a public class User within the EntityLayer.Concrete namespace. The class has a comment indicating its purpose for determining users and roles, a primary key attribute [Key], and five public properties: User_ID, Role_ID, Email, and Username, each with get and set methods. The code is as follows:

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel.DataAnnotations;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace EntityLayer.Concrete
9 {
10     15 başvuru
11     public class User
12     {
13         //Kullanıcılar ve rollerin belirlenmesi
14         [Key]
15         0 başvuru
16        public int User_ID { get; set; }
17        0 başvuru
18        public int Role_ID { get; set; }
19        0 başvuru
20        public string Email { get; set; }
21        0 başvuru
22        public string Username { get; set; }
23    }
24 }
```

Forgot Password sayfası için kullanılacak bilgiler:

```
C# EntityLayer
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel.DataAnnotations;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace EntityLayer.Concrete
9 {
10     public class ForgotPassword
11     {
12         [Required]
13         [EmailAddress]
14         public string Email { get; set; }
15     }
16 }
17
```

Login sayfası için kullanılacak bilgiler:

```
C# EntityLayer
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel.DataAnnotations;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace EntityLayer.Concrete
9 {
10     public class Login
11     {
12         //Kullanıcı girişi
13         public int User_ID { get; set; }
14         [Required(ErrorMessage = "Kullanıcı adı gereklidir.")] public required string User_Name { get; set; }
15         [Required(ErrorMessage = "Kullanıcı şifresi gereklidir.")] public required string PasswordHash { get; set; }
16         public int Role_ID { get; set; }
17     }
18 }
19
```

2)DataAccessLayer Katmanı

DataAccessLayer

- Abstract
- Repository
- EntityFramework
- Migrations
- Concrete
 - Context.cs

->Context.cs içinde veritabanı bağlantı yapılandırılması ve tabloların oluşturulması

```
Context.cs
C:\Users\SERRA NUR\source\repos\Core_ManagementSystem\DataAccessLayer\Concrete\Context.cs
using Microsoft.EntityFrameworkCore;
using Microsoft.Identity.Client;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DataAccessLayer.Concrete
{
    public class Context : DbContext
    {
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlServer("server=LAPTOP-BBPSTK82\\SQLEXPRESS;database=CoreProjeDB;integrated security=true;TrustServerCertificate=True");
        }

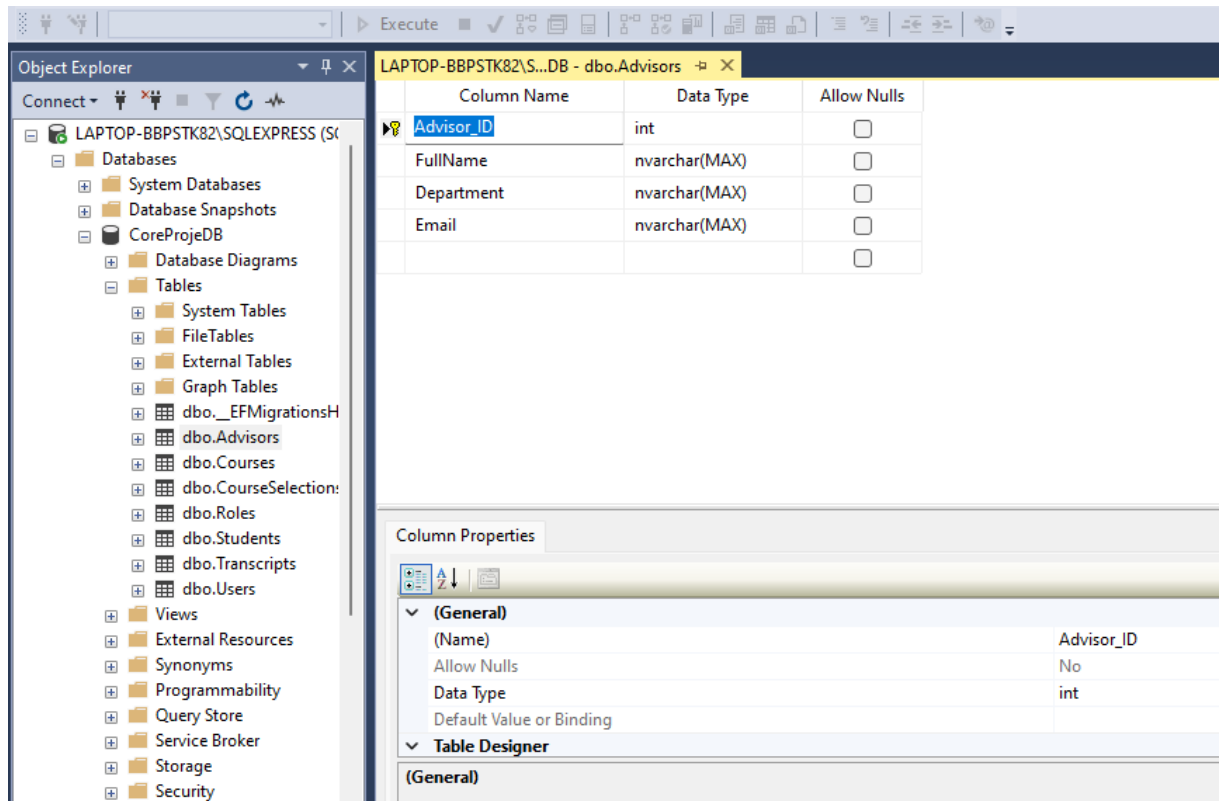
        public DbSet<User> Users { get; set; }
        public DbSet<Course> Courses { get; set; }
        public DbSet<CourseSelection> CourseSelections { get; set; }
        public DbSet<Advisor> Advisors { get; set; }
        public DbSet<Student> Students { get; set; }
        public DbSet<Role> Roles { get; set; }
        public DbSet<Transcript> Transcripts { get; set; }
    }
}
```

Migrations işlemi ve veritabanına tabloların aktarımı

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
    optionsBuilder.UseSqlServer("server=LAPTOP-BBPSTK82\\SQLEXPRESS;database=CoreProjeDB;integrated security=true;TrustServerCertificate=True");
}

public DbSet<User> Users { get; set; }
public DbSet<Course> Courses { get; set; }
public DbSet<CourseSelection> CourseSelections { get; set; }

Paket Yöneticisi Konsolu
Paket kaynağı: Tümü Varsayılan proje: Core_ManagementSystem
Her paket size sahibi tarafından lisanslanır. NuGet üçüncü taraf paketlerden sorumlu değildir ve bunlar için lisans vermez. Bazı paketler ek lisanslar tarafından yönetilen bağımlılıklar içerir. Bağımlılıkları belirlemek için paket kaynağı (akış) URL'sini izleyin.
Paket Yöneticisi Konsolu Konak Sürümü 6.11.1.2
Kullanılabilir tüm NuGet komutlarını görmek için 'get-help NuGet' yazın.
PM> add-migrations mig1
```

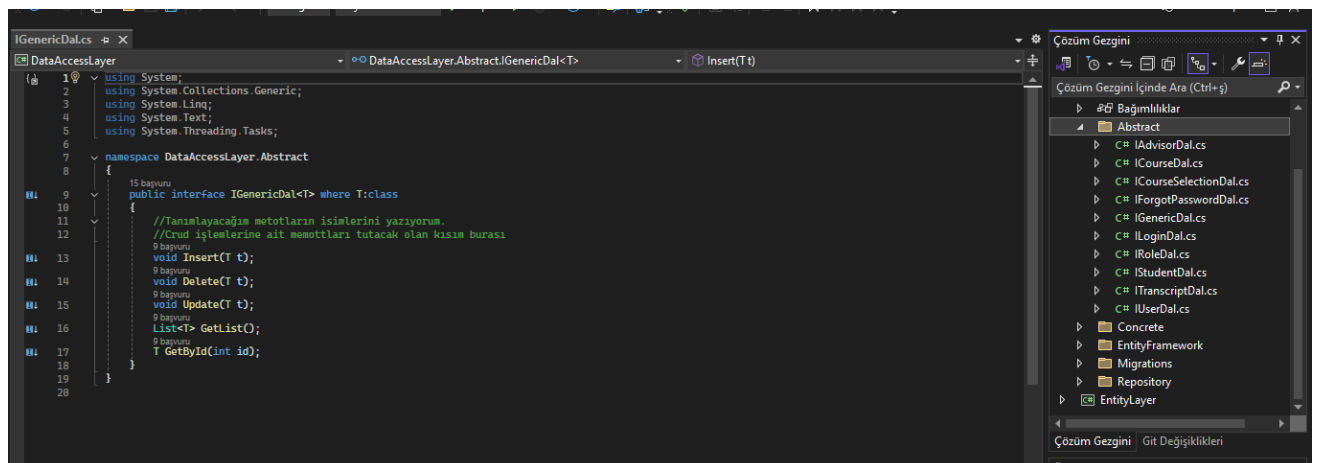


DataAccessLayer Katmanında Abstract içine Interfacelerin tanımlanması

└─ Abstract

└─ IAdvisorDal.cs

...



Generic Repository'den miras alan EF sınıflarının oluşturulması

└─ EntityFramework

└─ EFAdvisorDal.cs

...

```
EFAdvisorDal.cs -p X
C# DataAccessLayer
1 using DataAccessLayer.Abstract;
2 using DataAccessLayer.Repository;
3 using EntityLayer.Concrete;
4 using System;
5 using System.Collections.Generic;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9
10 namespace DataAccessLayer.EntityFramework
11 {
12     2 başvuru
13     public class EFAdvisorDal:GenericRepository<Advisor>,IAdvisorDal
14     {
15     }
16 }
```

└─ Repository

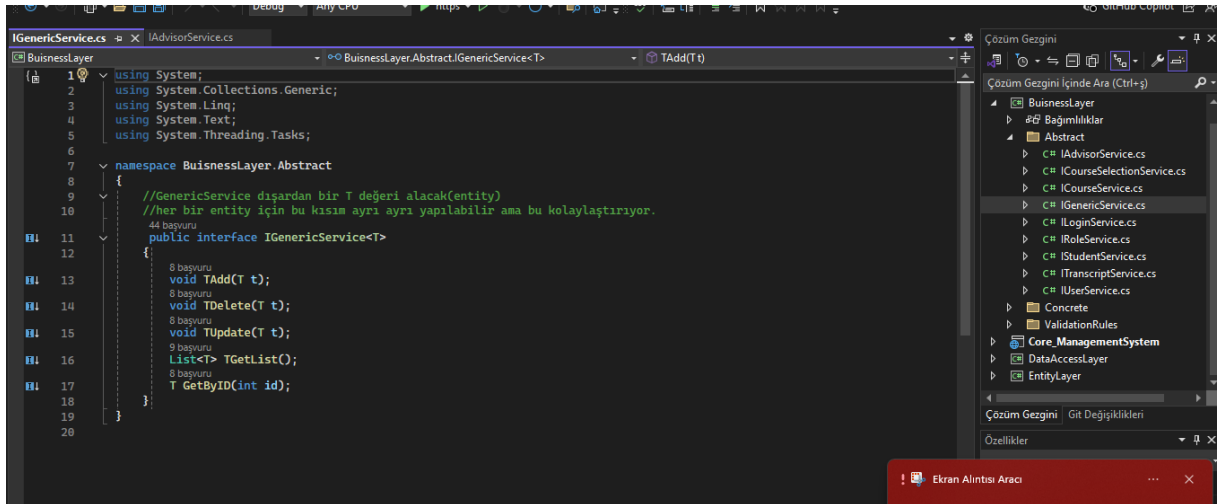
└─ GenericRepository.cs

```
GenericRepository.cs -p X EFAdvisorDal.cs
C# DataAccessLayer
1 using DataAccessLayer.Abstract;
2 using DataAccessLayer.Concrete;
3 using System;
4 using System.Collections.Generic;
5 using System.Linq;
6 using System.Text;
7 using System.Threading.Tasks;
8
9 namespace DataAccessLayer.Repository
10 {
11     9 başvuru
12     public class GenericRepository<T> : IGenericDal<T> where T : class
13     {
14     9 başvuru
15     void IGenericDal<T>.Delete(T t)
16     {
17         using var c = new Context();
18         c.Remove(t); //t dn gelen deęeri silme ięlemi tamamlandı
19         c.SaveChanges(); //Veri tabanına yansıması için deęişiklikleri kaydet
20     }
21
22     9 başvuru
23     T IGenericDal<T>.GetById(int id)
24     {
25         using var c = new Context();
26         return c.Set<T>().Find(id);
27     }
28
29     9 başvuru
30     List<T> IGenericDal<T>.GetList()
31     {
32         using var c = new Context();
33         return c.Set<T>().ToList();
34     }
35
36     9 başvuru
37     void IGenericDal<T>.Insert(T t)
38     {
39         using var c = new Context();
40         c.Add(t);
41         c.SaveChanges();
42     }
43
44     9 başvuru
45     void IGenericDal<T>.Update(T t)
46     {
47         using var c = new Context();
48         c.Update(t);
49         c.SaveChanges();
50     }
51 }
```

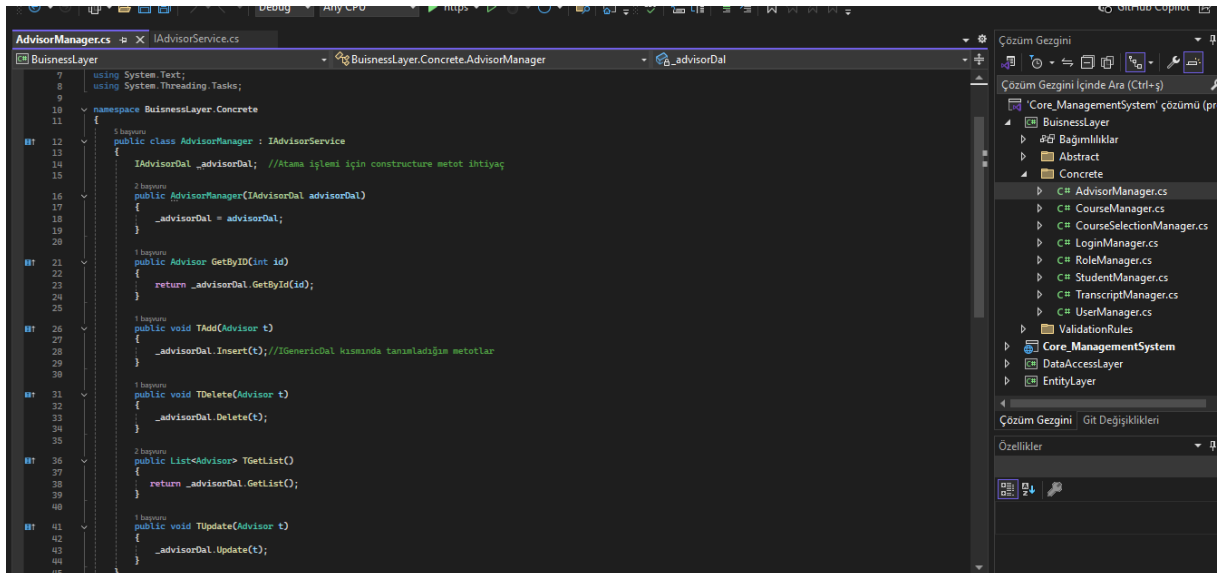

3)BuisnessLayer Katmanının oluşturulması

- Abstract
- Concrete
- ValidationRules

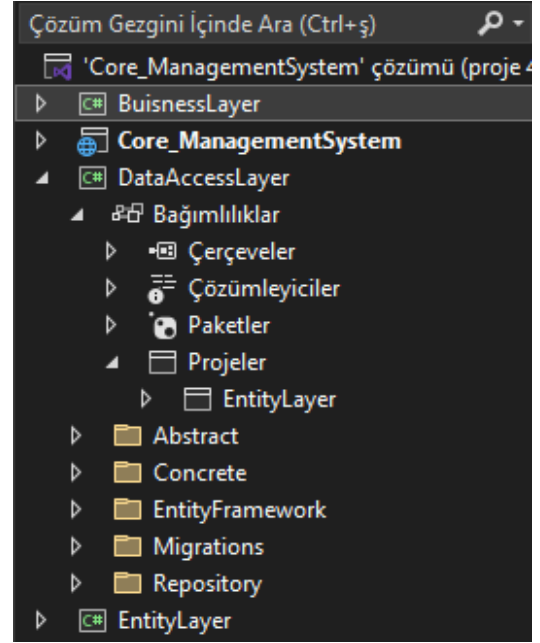
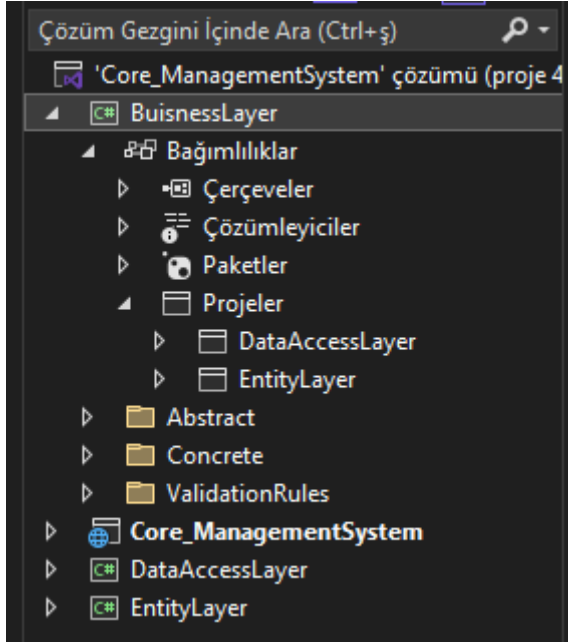
Abstract'ın içindeki Service sınıfları



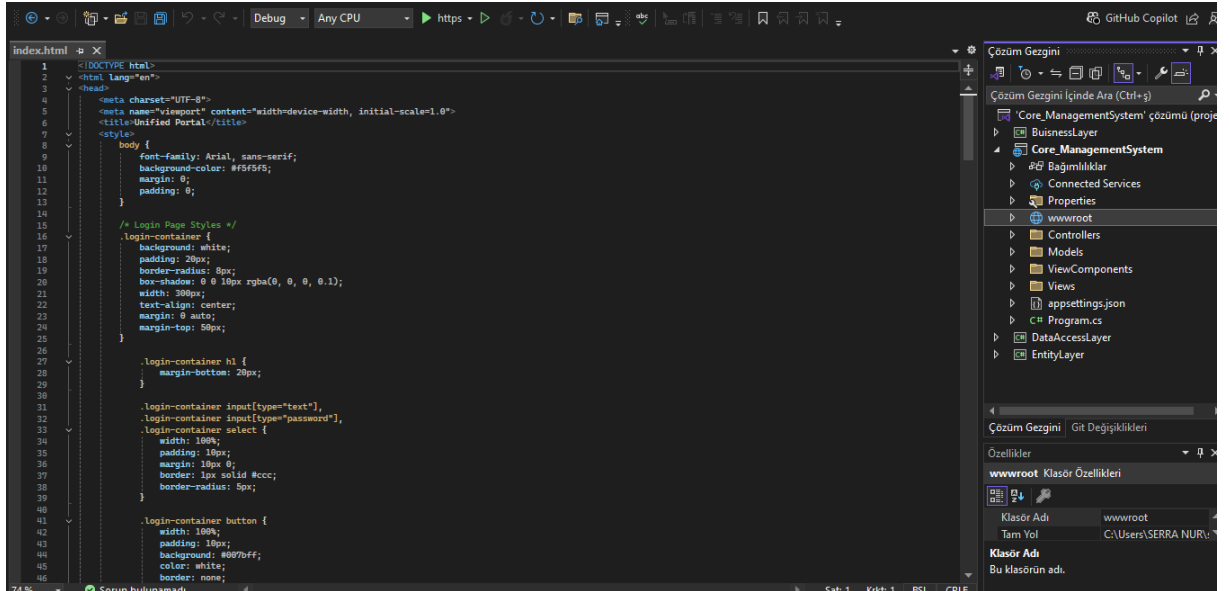
Concrete'in içindeki Manager sınıfları



4)Projelerin Birbirine Bağlanması



5)Core_ManagementSystem Katmanı ve Temanın oluşturulup yüklenmesi



Login

Student

▼

Username

Password

Login

[Forgot Password?](#)

Ders Seçimi

☐ Matematik

☐ Fizik

Dersleri Seç

(Eski ders seçimi sayfa görseli)

Daha kolay ve anlaşılır kod yönetimi bakımından kodların partial kullanılarak ayrılması

```
HeaderPartial.cshtml | Index.cshtml -p X
Core_ManagementSystem

1
2 {
3     Layout = "~/Views/Shared/_Layout.cshtml";
4 }
5 <!DOCTYPE html>
6 <html lang="en">
7     @await Html.PartialAsync("HeaderPartial")
8 </html>
9 <body>
10     <br />
11
12     <!-- Login Page -->
13     <div id="loginPage" class="page">
14         <div class="login-container">
15             @await Component.InvokeAsync("LoginList")
16         </div>
17     </div>
18
19     <!-- Student Panel Page -->
20     <div id="studentPage" class="page">
21         @await Component.InvokeAsync("CourseSelectedList")
22     </div>
23
24     <!-- Advisor Panel Page -->
25     <div id="advisorPage" class="page">
26         @await Component.InvokeAsync("AdvisorList")
27     </div>
28
29     <!-- Transcript Page -->
30     <div id="transcriptPage" class="page">
31         @await Component.InvokeAsync("TranscriptList")
32     </div>
33
34     <script>
35         function showPage(page) {
36             // Hide all pages
37             const pages = document.querySelectorAll('.page');
38             pages.forEach(p => p.style.display = 'none');
39
40             // Show the selected page
41         }
42     </script>
43
44 }
```

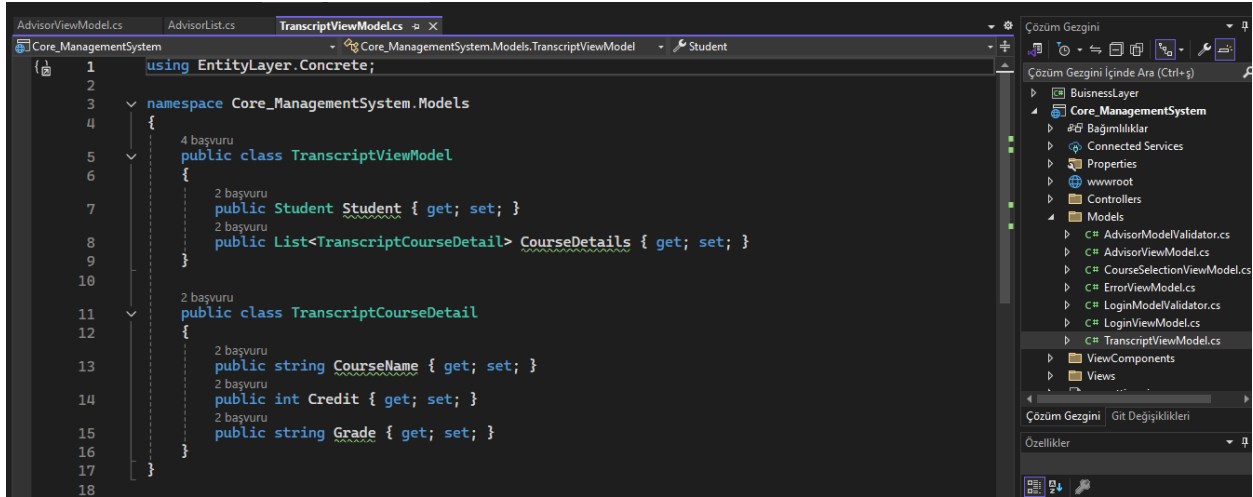
Bu işlem için kodların ayrılması

(Views->Shared->Components klasörü içinde her sayfanın kodunu içeren sınıflar oluşturuldu.)

```
Default.cshtml | Default.cshtml
Core_ManagementSystem

1 @model List<Core_ManagementSystem.Models.TranscriptViewModel>
2
3 <div class="container">
4     @foreach (var transcript in Model)
5     {
6         <h1>Transcript for @transcript.Student.First_Name</h1>
7         <table>
8             <thead>
9                 <tr>
10                     <th>Course Name</th>
11                     <th>Credit</th>
12                     <th>Grade</th>
13                 </tr>
14             </thead>
15             <tbody id="transcriptData">
16                 @foreach (var courseDetail in transcript.CourseDetails)
17                 {
18                     <tr>
19                         <td>@courseDetail.CourseName</td>
20                         <td>@courseDetail.Credit</td>
21                         <td>@courseDetail.Grade</td>
22                     </tr>
23                 }
24             </tbody>
25         </table>
26
27         <a href="studentDashboard.html" class="btn">Back to Dashboard</a>
28     }
29 </div>
30
31 }
```

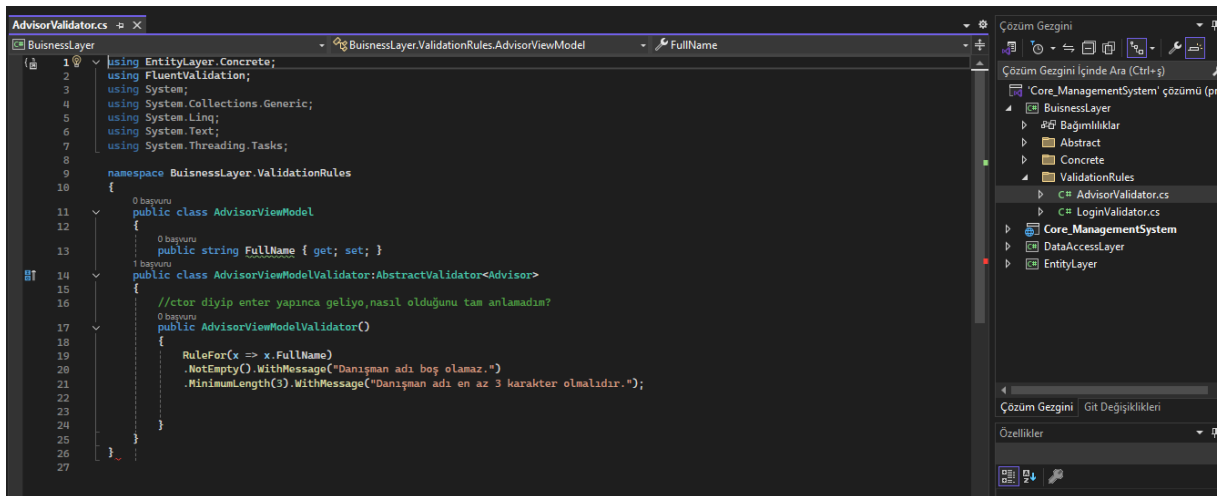
Tabloların bilgilerinin sayfada birbirleri için kullanımını sağlayan ViewModeller



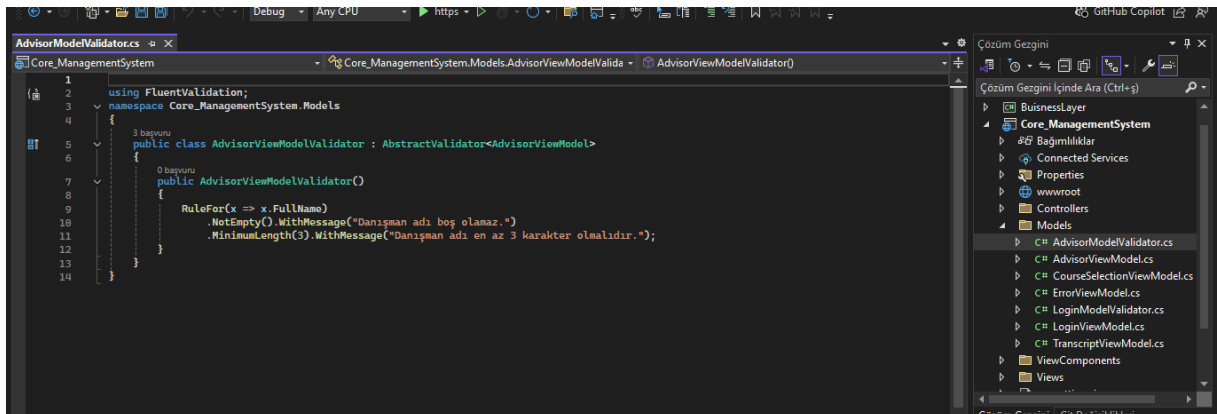
```
1 using EntityLayer.Concrete;
2
3 namespace Core_ManagementSystem.Models
4 {
5     4 başvuru
6     public class TranscriptViewModel
7     {
8         2 başvuru
9         public Student Student { get; set; }
10        2 başvuru
11        public List<TranscriptCourseDetail> CourseDetails { get; set; }
12    }
13
14    2 başvuru
15    public class TranscriptCourseDetail
16    {
17        2 başvuru
18        public string CourseName { get; set; }
19        2 başvuru
20        public int Credit { get; set; }
21        2 başvuru
22        public string Grade { get; set; }
23    }
24 }
```

Fluent Validation paketlerinin Buisness Katmanına yüklenmesi ve Validation Rules(kuralların) belirlenmesi

->(Validator Rules Sınıflarının ve Modellerinin oluşturulması)

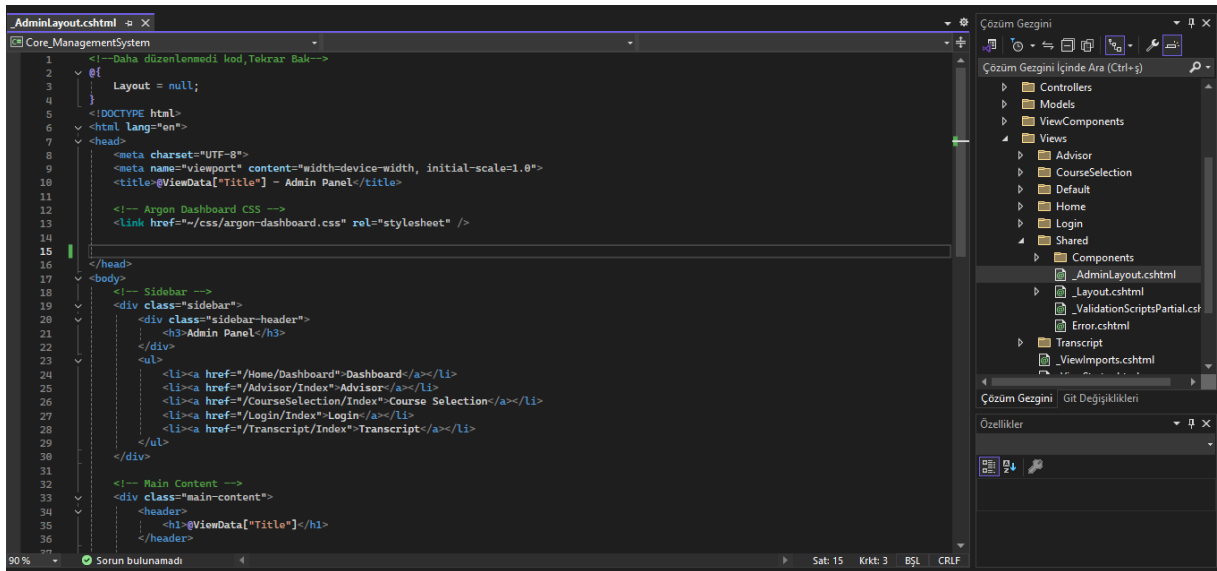
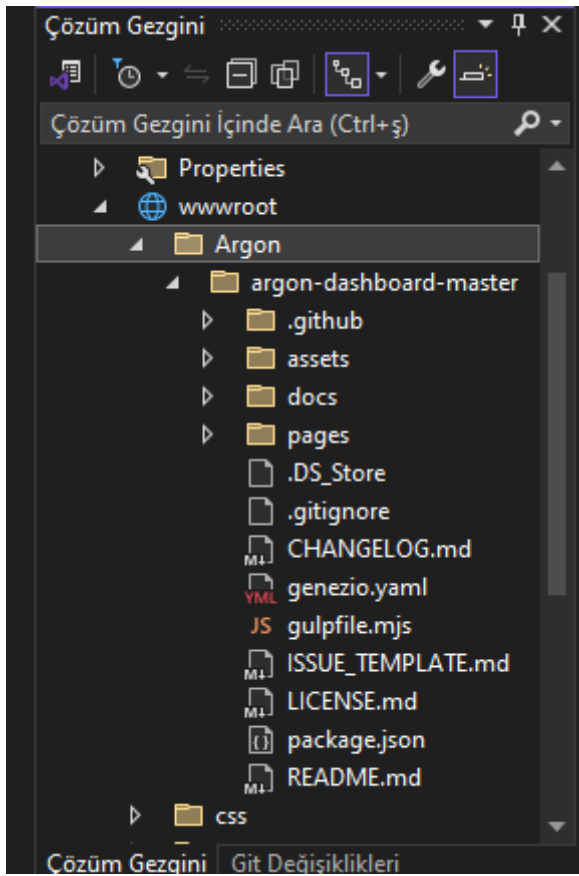


```
1 using EntityLayer.Concrete;
2 using FluentValidation;
3 using System;
4 using System.Collections.Generic;
5 using System.Linq;
6 using System.Text;
7 using System.Threading.Tasks;
8
9 namespace BusinessLayer.ValidationRules
10 {
11     0 başvuru
12     public class AdvisorViewModel
13     {
14         0 başvuru
15         public string FullName { get; set; }
16     }
17
18     1 başvuru
19     public class AdvisorViewModelValidator:AbstractValidator<Advisor>
20     {
21         //ctor diyip enter yapınca geliyo,nasıl olduğunu tam anlamadım?
22         0 başvuru
23         public AdvisorViewModelValidator()
24         {
25             RuleFor(x => x.FullName)
26                 .NotEmpty().WithMessage("Danışman adı boş olamaz.")
27                 .MinimumLength(3).WithMessage("Danışman adı en az 3 karakter olmalıdır.");
28         }
29     }
30 }
```



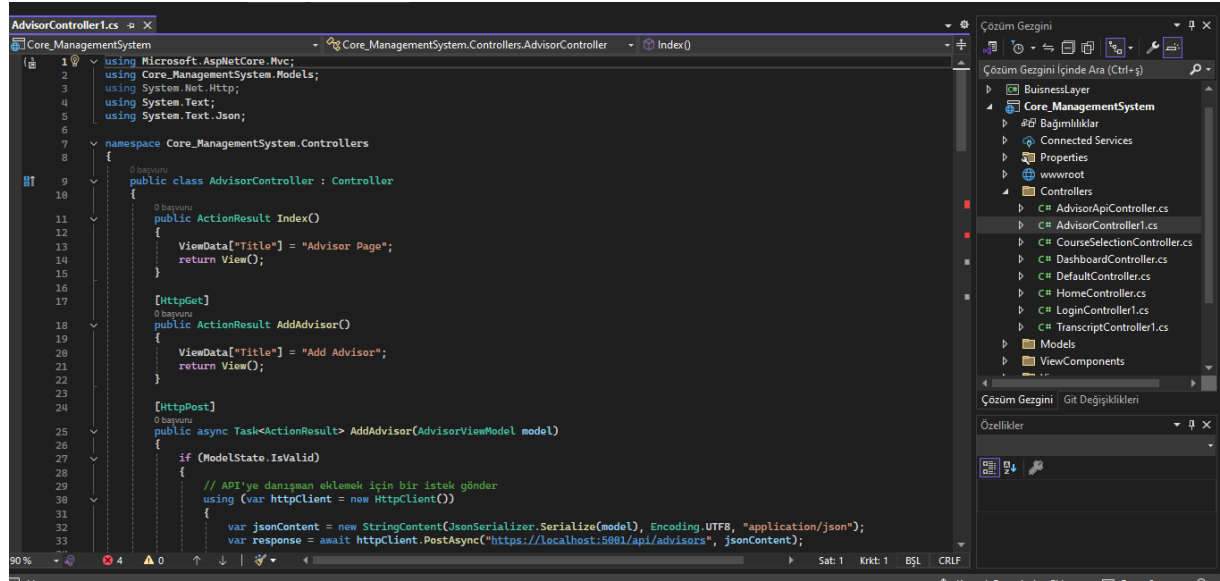
```
1 using FluentValidation;
2
3 namespace Core_ManagementSystem.Models
4 {
5     3 başvuru
6     public class AdvisorModelValidator : AbstractValidator<AdvisorViewModel>
7     {
8         0 başvuru
9         public AdvisorModelValidator()
10        {
11            RuleFor(x => x.FullName)
12                .NotEmpty().WithMessage("Danışman adı boş olamaz.")
13                .MinimumLength(3).WithMessage("Danışman adı en az 3 karakter olmalıdır.");
14        }
15    }
16 }
```

Admin Temasınının eklenmesi



Admin için Controller sınıfları

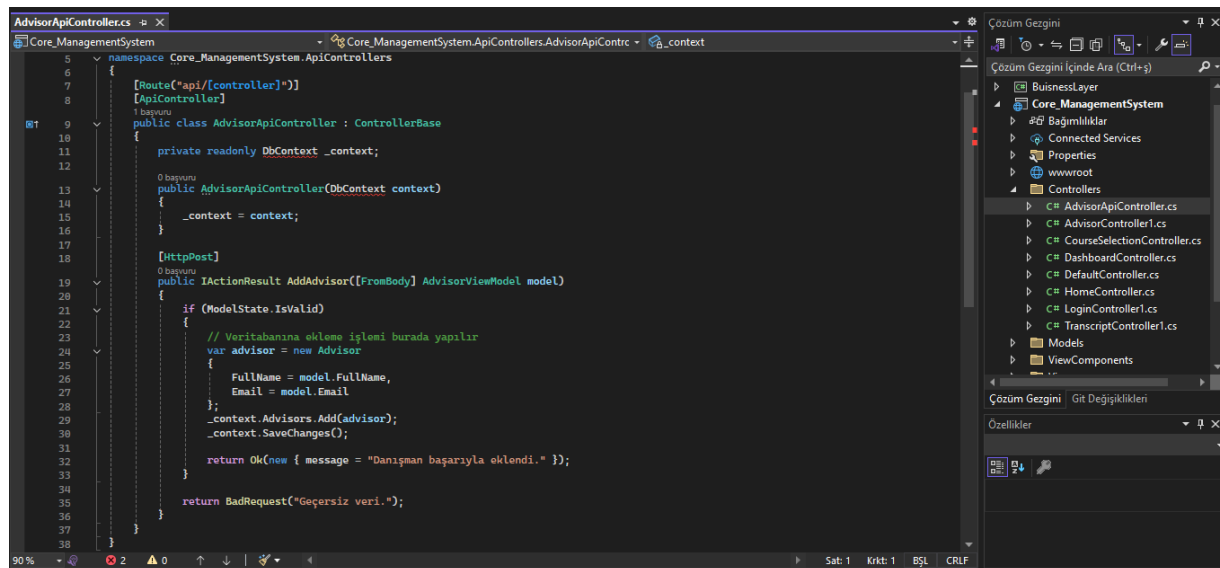
(Advisor ekleme...vb işlemleri yapabilmek için)



```
1 using Microsoft.AspNetCore.Mvc;
2 using Core_ManagementSystem.Models;
3 using System.Net.Http;
4 using System.Text;
5 using System.Text.Json;
6
7 namespace Core_ManagementSystem.Controllers
8 {
9     public class AdvisorController : Controller
10     {
11         public IActionResult Index()
12         {
13             ViewData["Title"] = "Advisor Page";
14             return View();
15         }
16
17         [HttpGet]
18         public IActionResult AddAdvisor()
19         {
20             ViewData["Title"] = "Add Advisor";
21             return View();
22         }
23
24         [HttpPost]
25         public async Task<ActionResult> AddAdvisor(AdvisorViewModel model)
26         {
27             if (ModelState.IsValid)
28             {
29                 // API'ye danışman eklemek için bir istek gönder
30                 using (var httpClient = new HttpClient())
31                 {
32                     var jsonContent = new StringContent(JsonSerializer.Serialize(model), Encoding.UTF8, "application/json");
33                     var response = await httpClient.PostAsync("https://localhost:5981/api/advisors", jsonContent);
34                 }
35             }
36         }
37     }
38 }
```

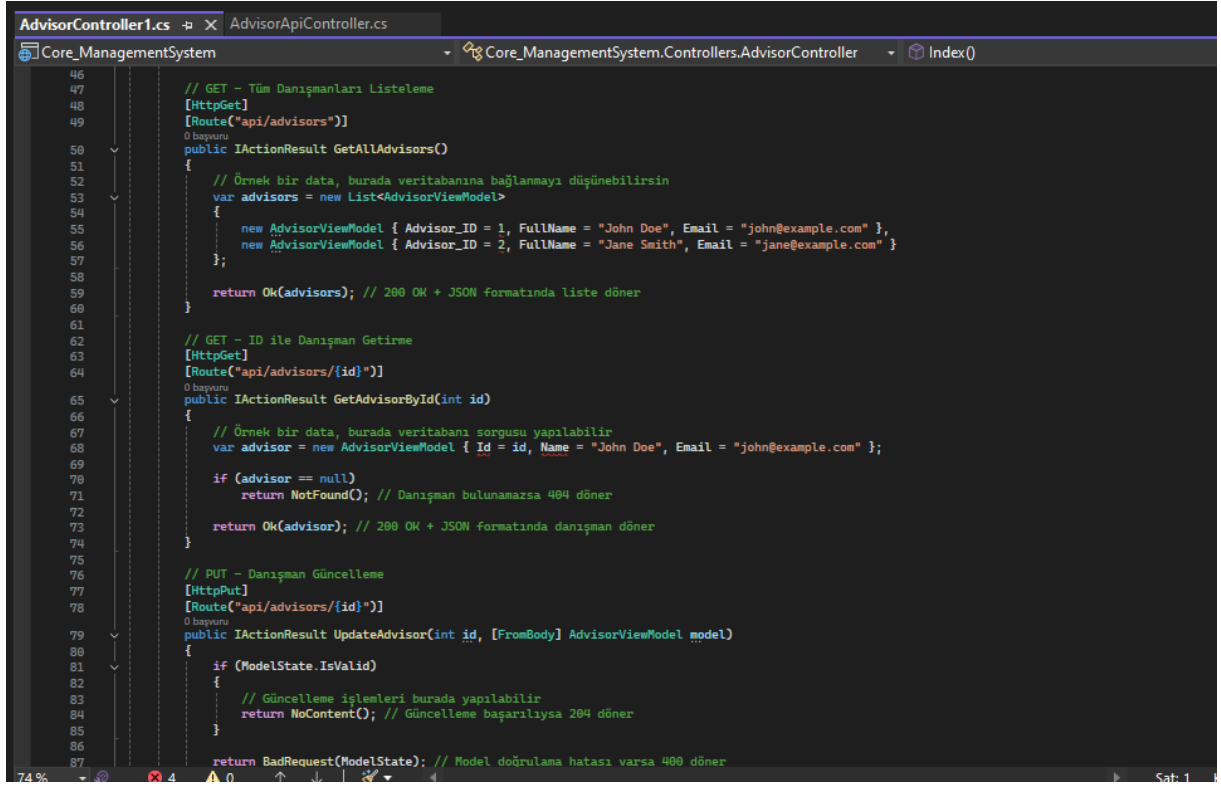
API

ApiController kısımları



```
1 namespace Core_ManagementSystem.ApiControllers
2 {
3     [Route("api/[controller]")]
4     [ApiController]
5     public class AdvisorApiController : ControllerBase
6     {
7         private readonly DbContext _context;
8
9         public AdvisorApiController(DbContext context)
10         {
11             _context = context;
12         }
13
14         [HttpPost]
15         public IActionResult AddAdvisor([FromBody] AdvisorViewModel model)
16         {
17             if (ModelState.IsValid)
18             {
19                 // Veritabanına ekleme işlemi burada yapılır
20                 var advisor = new Advisor
21                 {
22                     FullName = model.FullName,
23                     Email = model.Email
24                 };
25                 _context.Advisors.Add(advisor);
26                 _context.SaveChanges();
27                 return Ok(new { message = "Danışman başarıyla eklendi." });
28             }
29             return BadRequest("Geçersiz veri.");
30         }
31     }
32 }
```

Önceki Controller kısımlarına ekleme yaptım.



```
46
47 // GET - Tüm Danışmanları Listeleme
48 [HttpGet]
49 [Route("api/advisors")]
50 public IActionResult GetAllAdvisors()
51 {
52     // Örnek bir data, burada veritabanına bağlanmayı düşünebilirsin
53     var advisors = new List<AdvisorViewModel>
54     {
55         new AdvisorViewModel { Advisor_ID = 1, FullName = "John Doe", Email = "john@example.com" },
56         new AdvisorViewModel { Advisor_ID = 2, FullName = "Jane Smith", Email = "jane@example.com" }
57     };
58     return Ok(advisors); // 200 OK + JSON formatında liste döner
59 }
60
61 // GET - ID ile Danışman Getirme
62 [HttpGet]
63 [Route("api/advisors/{id}")]
64 public IActionResult GetAdvisorById(int id)
65 {
66     // Örnek bir data, burada veritabanı sorgusu yapılabilir
67     var advisor = new AdvisorViewModel { Id = id, Name = "John Doe", Email = "john@example.com" };
68
69     if (advisor == null)
70     {
71         return NotFound(); // Danışman bulunamazsa 404 döner
72     }
73     return Ok(advisor); // 200 OK + JSON formatında danışman döner
74 }
75
76 // PUT - Danışman Güncelleme
77 [HttpPut]
78 [Route("api/advisors/{id}")]
79 public IActionResult UpdateAdvisor(int id, [FromBody] AdvisorViewModel model)
80 {
81     if (ModelState.IsValid)
82     {
83         // Güncelleme işlemleri burada yapılabilir
84         return NoContent(); // Güncelleme başarılıysa 204 döner
85     }
86
87     return BadRequest(ModelState); // Model doğrulama hatası varsa 400 döner
88 }
```

(Api kısmı tamamen bitmedi)

->Admin tema şablonunu Argon'un sunduğu klaösrden indirdim.