AZİME NUR ÖZKAYA

Ders Sunum Sistemi

Proje Hakkında

Ders Sunum Sistemi, eğitim kurumları ve öğretim görevlileri için geliştirilmiş bir web uygulamasıdır. Bu sistem, ders materyallerinin kategorize edilmiş ve düzenli bir şekilde saklanmasını, yönetilmesini ve öğrencilere sunulmasını sağlar.

MVC Mimarisi Nedir?

****MVC (Model-View-Controller)****, yazılım geliştirmede kullanılan bir mimari tasarım desenidir. Uygulamayı üç ana bileşene ayırarak kodun daha düzenli, sürdürülebilir ve test edilebilir olmasını sağlar:

1. Model (Veri Katmanı)

- ****Görev****: Uygulamanın verilerini ve iş mantığını temsil eder

- ****Projemizde****:

  - `User.cs` - Kullanıcı bilgilerini tutar (Id, Kullanıcı Adı, Şifre, Tam İsim, Admin yetkisi)

  - `Category.cs` - Ders kategorilerini temsil eder (Matematik, Fizik, vb.)

  - `Course.cs` - Dersleri temsil eder (İlişkili kategori, eğitmen bilgisi)

  - `Presentation.cs` - Sunum dosyalarını temsil eder (Başlık, açıklama, dosya yolu)

2. View (Görünüm Katmanı)

- ****Görev****: Kullanıcıya gösterilecek arayüzü oluşturur

- ****Projemizde****:

  - `Views/Home/Index.cshtml` - Ana sayfa, kategorileri listeler

  - `Views/Home/Category.cshtml` - Seçilen kategorideki dersleri gösterir

- `Views/Home/Course.cshtml` - Ders detayları ve sunumları listeler

- `Views/Admin/Login.cshtml` - Admin giriş sayfası

- `Views/Categories/`, `Courses/`, `Presentations/` - CRUD işlemleri için sayfalar

3. Controller (Kontrol Katmanı)

- ****Görev****: Model ve View arasında köprü görevi görür, kullanıcı isteklerini yönetir

- ****Projemizde****:

- `HomeController.cs` - Ana sayfa ve gezinme işlemlerini yönetir

- `AdminController.cs` - Admin girişi ve yetkilendirme

- `CategoriesController.cs` - Kategori CRUD işlemleri

- `CoursesController.cs` - Ders CRUD işlemleri

- `PresentationsController.cs` - Sunum yükleme ve yönetimi

MVC'nin Avantajları

1. ****Separation of Concerns (Sorumlulukların Ayrılması)****: Her katman kendi işine odaklanır

2. ****Test Edilebilirlik****: Her katman bağımsız test edilebilir

3. ****Yeniden Kullanılabilirlik****: Aynı Model farklı View'larda kullanılabilir

4. ****Paralel Geliştirme****: Ekip üyeleri farklı katmanlarda eş zamanlı çalışabilir

5. ****Bakım Kolaylığı****: Değişiklikler sadece ilgili katmanda yapılır

Proje Yapısı ve İşleyiş

Veritabanı Tasarımı (Entity Framework Core)

```
ApplicationDbContext.cs (Data Katmanı)
 ├— Users (Kullanıcılar tablosu)
 ├— Categories (Kategoriler tablosu)
```

```
│   └── Courses (1-N ilişki: Bir kategorinin birçok dersi var)
├── Courses (Dersler tablosu)
│   ├── CategoryId (Foreign Key)
│   └── Presentations (1-N ilişki: Bir dersin birçok sunumu var)
└── Presentations (Sunumlar tablosu)
    └── CourseId (Foreign Key)
```

Kullanılan Teknolojiler

- **ASP.NET Core 8.0**: Web framework
- **Entity Framework Core 9.0**: ORM (Object-Relational Mapping)
- **SQL Server**: Veritabanı
- **Bootstrap 5**: UI framework
- **Razor Pages**: View engine

Özellikler

Kullanıcı Tarafı

1. **Ana Sayfa**: Tüm kategorileri görüntüleme
2. **Kategori Sayfası**: Seçilen kategorideki dersleri listeleme
3. **Ders Sayfası**: Ders detayları ve sunumları görüntüleme
4. **Sunum İndirme**: Dosyaları indirme imkanı

Admin Paneli

1. **Güvenli Giriş**: Session tabanlı kimlik doğrulama
2. **Kategori Yönetimi**: Ekleme, düzenleme, silme (CRUD)
3. **Ders Yönetimi**: Kategorilere ders ekleme/düzenleme

4. **Sunum Yönetimi**: PDF, PowerPoint dosyalarını yükleme ve yönetme

Nasıl Çalıştırılır?

Gereksinimler

- .NET 8.0 SDK

- SQL Server veya SQL Server Express

- Visual Studio 2022 veya VS Code

Adımlar

1. **Projeyi Klonlayın**

Github adresimde mevcut azime57

2. **Veritabanını Oluşturun**

```bash

cd DersSunumSistemi

dotnet ef database update

```

3. **Uygulamayı Çalıştırın**

```bash

dotnet run

```

4. **Tarayıcıda Açın**

```

http://localhost:5178

```

İlk Admin Girişi

- **Kullanıcı Adı**: admin

- **Şifre**: admin123

Projenin Teknik Detayları

1. Routing Yapısı

```csharp
// Program.cs içinde
app.MapControllerRoute(
   name: "default",
   pattern: "{controller=Home}/{action=Index}/{id?}");
```

- `/` → Ana sayfa (Kategoriler)

- `/Home/Category/1` → 1 numaralı kategori

- `/Home/Course/1` → 1 numaralı ders

- `/Admin` → Admin paneli

2. Veri İlişkileri (Entity Framework)

Include ve ThenInclude Kullanımı:

```csharp
// HomeController.cs:22-24
var categories = await _context.Categories
   .Include(c => c.Courses)           // Kategorinin derslerini dahil et
   .ThenInclude(c => c.Presentations)   // Her dersin sunumlarını dahil et
   .ToListAsync();
```

```

Bu yapı sayesinde N+1 sorunu önlenir ve tek sorguda tüm ilişkili veriler çekilir.

### 3. Dosya Yükleme Sistemi

```csharp
// PresentationsController.cs içinde
public async Task<IActionResult> Create(Presentation presentation, IFormFile file)
{
  if (file != null)
  {
    var uploadsFolder = Path.Combine("wwwroot", "uploads");
    var uniqueFileName = Guid.NewGuid().ToString() + "_" + file.FileName;
    var filePath = Path.Combine(uploadsFolder, uniqueFileName);

    using (var stream = new FileStream(filePath, FileMode.Create))
    {
      await file.CopyToAsync(stream);
    }

    presentation.FileName = uniqueFileName;
    presentation.FilePath = "/uploads/" + uniqueFileName;
  }
}
```

4. Session Yönetimi (Admin Kontrolü)
```

```csharp
// AdminController.cs içinde giriş kontrolü
HttpContext.Session.SetString("AdminId", user.Id.ToString());
HttpContext.Session.SetString("AdminName", user.FullName);

// Diğer controller'larda yetki kontrolü
if (HttpContext.Session.GetString("AdminId") == null)
{
    return RedirectToAction("Login", "Admin");
}
```

Öğrendiklerim ve Geliştirme Süreci

1. MVC Pattern Anlayışı

- Controller'ların nasıl çalıştığını öğrendim
- Action metodlarının View'lara veri aktarımını kavradım
- Routing mekanizmasını anladım

2. Entity Framework Core

- Code-First yaklaşımı ile veritabanı oluşturmayı öğrendim
- Navigation properties ile ilişkisel veritabanı tasarımını uyguladım
- LINQ sorguları ile veri çekme işlemlerini gerçekleştirdim

3. Asenkron Programlama

- `async/await` kullanımını öğrendim
- `Task<IActionResult>` dönen metodlar yazdım

- Veritabanı işlemlerinde asenkron metodlar kullandım

4. Dosya İşlemleri

- `IFormFile` ile dosya yükleme
- `wwwroot` klasöründe statik dosya yönetimi
- Güvenli dosya adlandırma (GUID kullanımı)

Karşılaşılan Sorunlar ve Çözümler

Sorun 1: Null Reference Uyarıları

****Çözüm****: Navigation property'lerde nullable (`?`) kullanımı

```csharp
public Category? Category { get; set; } // Nullable referans
```

Sorun 2: Cascade Delete Problemleri

****Çözüm****: Entity Framework'ün otomatik cascade delete davranışını kullanma

Sorun 3: Session Yönetimi

****Çözüm****: Program.cs'de session middleware'i ekleme

```csharp
builder.Services.AddSession();
app.UseSession();
```

Geliştirme Fikirleri

- [ ] Kullanıcı kayıt sistemi

- [ ] Öğrenci ve öğretmen rolleri
- [ ] Sunum dosyalarına yorum yapabilme
- [ ] Arama ve filtreleme özellikleri
- [ ] Responsive tasarım iyileştirmeleri
- [ ] API desteği (RESTful)