**KTUBYS Okul Yönetim Sistemi Proje Raporu**

1.Giriş

**KTUBYS Okul Yönetim Sistemi**, üniversite öğrencileri, dersler ve danışmanların yönetilmesini sağlayan kapsamlı bir çözümdür. Proje, duyarlı (responsive) ve ölçeklenebilir bir yönetim sistemi oluşturmak amacıyla front-end, back-end, veritabanı ve API servislerini entegre eder. Bu rapor, geliştirme sürecinin her aşamasını ayrıntılı olarak anlatmakta ve önemli teknik uygulamalar ile tasarım kararlarını vurgulamaktadır.

**2.Frontend Geliştirme**

**2.1 HTML ve Razor Görünümleri**

Projenin ön yüzü, **Razor View Engine** kullanılarak ASP.NET Core ile oluşturulmuştur. Her varlık (öğrenciler, dersler, danışmanlar) için CRUD işlemlerini gerçekleştiren görünümler oluşturulmuştur:

* **Index.cshtml**: Kayıtların listesini görüntüler.
* **Details.cshtml**: Belirli bir kaydın detaylarını gösterir.
* **Create.cshtml**: Yeni kayıt eklemek için form sağlar.
* **Edit.cshtml**: Mevcut kayıtların düzenlenmesine olanak tanır.
* **Delete.cshtml**: Kayıtların silinmesi için onay ekranı sağlar.

**2.2 Duyarlı Tasarım**

Proje, **CSS** ve **medya sorguları** kullanılarak duyarlı bir tasarım ile geliştirildi. Tasarım, sistemin mobil cihazlar da dahil olmak üzere çeşitli ekran boyutlarında iyi çalışmasını sağlamaktadır. site.css dosyasında temel stil ve duyarlılık için şunlar yer aldı:

* Flexbox ile navigasyon ve içerik düzeni sağlandı.
* 768px'den küçük ekranlar için medya sorguları kullanılarak düzen mobil cihazlara uygun hale getirildi.

**2.3 Layout ve Scriptler**

Tüm sayfalar için ana layout dosyası olarak \_Layout.cshtml oluşturuldu. Bu layout şunları içeriyordu:

* Öğrenciler, Dersler ve Danışmanlar bölümlerine bağlantılar içeren tutarlı bir navigasyon menüsü.
* Alt bilgi (footer) ve ana içerik alanı (RenderBody) gibi ortak bölümler. Ayrıca, form doğrulama ve kullanıcı etkileşimlerini sağlamak amacıyla script.js dosyası oluşturuldu.

**3. Backend Geliştirme**

**3.1 Controller'lar**

Her varlık (Öğrenciler, Danışmanlar, Dersler, StudentCourseSelections) için ayrı bir controller oluşturuldu ve **ASP.NET MVC** deseni kullanıldı. Her controller, CRUD işlemlerini yönetmek için gerekli olan yöntemleri içerir:

* **Index**: Tüm kayıtları getirir ve listeler.
* **Details**: Belirli bir kaydı ID ile getirir.
* **Create**: Yeni kayıt ekler.
* **Edit**: Mevcut kayıtları günceller.
* **Delete**: Kayıtları siler.

Controller'lar, veritabanı işlemleri için **Entity Framework (EF) Core** kullanılarak geliştirildi. Veritabanı işlemlerini yönetmek için DbContext sınıfı (KTUBYSContext) controller'lara enjekte edildi.

**3.2 API Entegrasyonu**

Sistemi genişletmek amacıyla, tüm ana varlıklar için bir **RESTful API** oluşturuldu. API, en iyi uygulamalara uygun olarak GET, POST, PUT ve DELETE gibi standart HTTP metodlarını kullandı. API belgelerini kolayca görüntülemek ve test etmek için **Swagger** entegre edildi. Başlıca adımlar şunlardı:

* Controller'lara API rotaları eklendi.
* JSON verilerinin serileştirilmesi ve serileştirilmemesi sağlandı.

**4. Veritabanı Yönetimi**

**4.1 Veritabanı Kurulumu**

Projede **SQL Server** veritabanı yönetim sistemi kullanıldı. Veritabanı şeması şu tabloları içeriyordu:

* Öğrenciler (Students)
* Danışmanlar (Advisors)
* Dersler (Courses)
* StudentCourseSelections (Öğrenci-Ders Seçimleri)

Her tablo, uygun birincil anahtar ve yabancı anahtar ilişkilerine sahipti. **Entity Framework Core**, veritabanı yönetimi için kullanıldı ve veritabanı şemasını güncellemek ve sürümlemek için **Migrations** aracı kullanıldı.

#### 4.2 Migrations

Veritabanı, **EF Core Migrations** kullanılarak oluşturuldu ve güncellendi. Add-Migration InitialCreate komutu ile ilk migration oluşturuldu ve Update-Database komutu kullanılarak migrationlar uygulandı. Bu süreç, veritabanı şeması üzerinde versiyon kontrolü sağlamaya yardımcı oldu.

#### 4.3 Bağlantı Dizgisi (Connection String)

Veritabanı ile proje arasındaki bağlantıyı sağlamak için appsettings.json dosyası, SQL Server bağlantı dizgisiyle yapılandırıldı.

{

"ConnectionStrings": {

"KTUBYSContext": "Server=your\_server\_name;Database=KTUBYS;Trusted\_Connection=True;"

}

}

### 5. API Geliştirme

#### 5.1 RESTful API Endpoint'leri

API, her bir varlık için CRUD işlemleri sağlamak amacıyla endpoint'ler sundu. Bu endpoint'ler, standart HTTP istekleri ile verilerin yönetilmesine olanak sağladı. Örnek API rotaları şunlardı:

* GET /api/Students
* GET /api/Students/{id}
* POST /api/Students
* PUT /api/Students/{id}
* DELETE /api/Students/{id}

#### 5.2 Swagger Entegrasyonu

Swagger, API belgelerini otomatik olarak oluşturmak amacıyla projeye eklendi. Bu sayede, tarayıcı arayüzü üzerinden API endpoint'lerinin kolayca test edilmesi ve API'nin standartlara uygun olup olmadığının kontrol edilmesi sağlandı.

services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "KTUBYS API", Version = "v1" });

});