**Genel Bilgiler**

Proje bir .net core web api projesidir. Generic Repository tasarım deseni kullanılarak, **Code First** yöntemi ile oluşturulmuştur.

Proje tek katmandan oluşmaktadır.

Validasyon için FluentValidation kütüphanesinden yararlanılmıştır.

Veritabanı olarak SqLite veritabanı kullanılmıştır.

Projede öğrenci, ders ve not modelleri için ekleme güncelleme, silme ve listeleme işlemlerini yapabilirsiniz.

**Not : Veritabanı dosyasını ana dizinde bulabilirsiniz. (SRMAPI.db)**

**Kullanılan teknolojiler**

• AspNetCore 7.0.10

• Microsoft.EntityFrameworkCore 7.0.10

• Microsoft.EntityFrameworkCore.Sqlite 7.0.10

• FluentValidation 11.7.1

• FluentValidation.AspNetCore 11.3.0

**Klasörler**

1. **Data**
2. **Entities**
3. **Extensions**
4. **Filters**
5. **Models**
6. **Validations**

**1 -) Data Klasörü**

Veri tabanı ile işlem yapmamızı sağlayan sınıfların ve interface’lerin barındırıldığı klasördür.

İnterface’ler Abstract klasöründe, Bunların doldurulduğu sınıflar ise Concrete klasöründe toplanmıştır.

Veri tabanı Context nesnesi olan Db.cs sınıfı burada tutulur.

**2-) Entities Kalsörü**

Veri tabanı nesnelerinin tutulduğu klasördür.

Not: Tüm veri tabanı nesneleri **EntityBase** nesnesinden kalıtım almaktadır.

**3-) Extensions Klasörü**

Extension sınıflarının tutulduğu klasördür.

Dependecy Injection talepleri için ServiceRegistration sınıfı kullanılmaktadır. Yeni bir repository vs. eklendiğinde **kullanılabilmesi için burada tanımlanmalıdır**.

**4-) Filters Klasörü**

Requestlerde araya girip işlem yapmak istediğimiz filter nesnelerimizi tanımladığımız klasördür.

**5-) Models Klasörü**

Ekleme, güncelleme lisetleme gibi işlemleri yaparken kullanacağımız nesneleri tuttuğumuz klasördür.

Örneğin öğrenci eklemek için kullandığımız view model burada tanımlanmıştır.

**6-) Validations Klasörü**

Fluent validation nesnelerimizi tuttuğumuz klasördür.

**Yeni Bir Repo Ekleme**

Veri tabanına yeni bir tablo eklediğinizde, controllerda bu tablo ile işlem yapabilmek için önce o tabloya ait IRepository<T> interface’sinden kalıtım alan bir interface tanımlamanız gerekmektedir. Ardından bu interface’yi doldurarak ServiceRegistration sınıfındaki AddServices() metodunda tanımlamanız gerekmektedir.

**Veri Tabanı**

Veri tabanı olarak SqLite kullanılmaktadır. Veri tabanı context nesnesi AddServices() metodunda IOC container’e eklenmiştir. Connection string bilgisini değiştirmek istiyorsanız veya farklı bir provider kullanmak istiyorsanız AddServices() metounda ilgili düzenlemeleri yapabilirsiniz.

**Öğrencilerin Listelenmesi**

Öğrenciler profillerindeki doluluk oranına göre büyükten küçüğe doğru sıralanırlar.

Bu sıralama algoritması öğrenciye tanımlı ders sayısı ve bu derslere girilen not bilgisini kontrol eder. Notlandırılmış ders ve Tanımlı ders oranı doluluk yüzdesini verir.

Örneğin birinci öğrenciye 3 ders tanımlanmış olsun bu öğrenciye bu derslerden sadece bir tanesi için not kaydedilmiş olsun. Bu durumda ((1/3)\*100) bu öğrencinin doluluk yüzdesi olacaktır.

**Veri Tabanı Tabloları**

1. **Student**
2. **Lesson**
3. **Grade**
4. **StudentLesson**

**Student**

Öğrenci Nesnesidir

**Lesson**

Ders Nesnesidir

**Grade**

Notların tutulduğu nesnesidir

**StudentLesson**

Öğrencilere ders tanımlamasının yapıldığı nesnedir. Öğrenci ve Ders id değerlerinin verilmesi yeterlidir.