İLERİ VERİTABANI SİSTEMLERİ

FİNAL ÖDEVİ

ZEHRA BELLİBAŞ - 160401061

- Blazor WebAssembly, WebAssembly'i kullanır. WebAssembly, çoğu modern tarayıcıda desteklenir. Sadece IE11'de desteklenmez.

- Blazor Server'da olduğu gibi, Razor bileşenlerini veya Razor sayfalarını kullanarak istemci tarafı sayfaları oluşturabiliriz.

Projemde Blazor WebAsembly kullanmayı tercih ettim. Bunun en önemli nedeni sürkeli sunucu ile bağlantıya gerek olmamasıdır.

- WebAssembly'de tarayıcının Blazor uygulamasını indirmesine izin verilir. Bu, Blazor uygulamasının web tarayıcısında çalıştığı anlamına gelir. Kısacası uygulamanın çalışması için bir sunucu ile sürekli bağlantıya gerek yoktur. Ancak, yalnızca istemci tarafı bir uygulama olduğundan, herhangi bir sunucu tarafı işlevselliğini Blazor uygulamasına doğrudan entegre edemeyiz. Sunucu tarafı işlevselliği için, onu ASP.NET Core Web API gibi bir sunucu tarafı uygulamasıyla bağlamamız gerekir.

Projemde veritabanı olarak SQLite kullanarak Blazor WebAssembly'de temel CRUD işlemlerini uyguladım. SQLite kullanmamın temel nedeni; herhangi bir ana sunucu işlemi gerektirmemesidir.

- Sqlite kullanmak için herhangi bir sunucu işlemini başlatmak, durdurmak ya da yapılandırmak gerekmez. Veritabanına erişmek için sunucuya istek göndermek ve sonuç almak için ara işlem iletişimi kurmak gerekmez.

Leaflet'ten faydalanarak veritabanında bulunan harita bilgileri çekebildim. Ayrıca projemde harita kullandım.

- JavaScript ve Css dosyalarını HTML tarafına import ettikten sonra kendi iç yapısındaki işleyişine müdahale etmediğimiz açık kaynak bir kitaplık olduğu için Lesaflet'i kullanmayı tercih ettim.

Projede Cargo'ların güncel konumlarının takibi yapılmaktadır. Ayrıyeten kargoların özellikleri de konumları ile birlikte veritabanında tutulmaktadır.

Projeyi uygulamaya koymak için en başta Visual Studio'da yeni bir proje oluşturdum.

- Visual Studio -> Create a new Project -> Blazor App -> Next -> Project Name = BlazorLeaflet -> Create -> Blazor WebAssembly ->Create adımlarını izleyerek yeni projemi oluşturmuş oldum.

Projeye başlarken öncelikle Leaflet'ten almam gereken kodları uygulamaya çektim.

Sonra projenin doğru şekilde ilerlemesi için gerekli olan aşağıdaki NuGet Package'ları indirdim.

- Microsoft.EntityFrameworkCore
- Microsoft.EntityFrameworkCore.Tools
- Microsoft.EntityFrameworkCore.Sqlite

BlazorLeaflet.Samples'teki Data klasörüne kargo varlığını Cargo.cs ismi ile class olarak ekledim. Burada veritabanındaki kargo tablosunun column'larının C# kodu ile yazılmış hali var.

- Id'yi 'Primary Key' olarak oluşurdum. Ayrıca kargonun ağırlık, ücret, teslimat şehri, konum özellikleri de vardır. Böylece bir kargoyu takip etmek için lazım olan özellikleri oluşturmuş oldum.

Sonra CargoDbContext.cs sınıfını oluşturdum.

- Bu, etkileşim kurmamıza ve veritabanı işlemlerini gerçekleştirmemize yardım eden bir DbContext sınıfıdır. Bu sınıfı oluştururken birkaç tanede cargo varlığı oluşturarak özelliklerini girdim.

CargoDbContext'i oluşturunca Startup.cs sınıfında bulunan ConfigureServices() kısmına veritabanının adını yani 'Cargos.db'yi içeren bağlantı dizesini ekledim. Eklediğim kod bloğu aşağıdaki gibidir:

```
services.AddDbContext<CargoDbContext>( options =>
{
        options.UseSqlite("Data Source = Cargos.db");
});
```

CargoServices.cs sınıfını da BlazorLeaflet.Samples'teki Data klasörüne ekledim. Bu sınıf, temel CRUD işlemlerini sağlayan hizmet sınıfıdır. Kargo ekleme, güncelleme, silme ve listeleme işlemleri bulunmaktadır.

Blazor.Leaflet.Samples'teki Pages klasörüne CRUD için kullanıcı arayüzü kodunuve mantığını ekledim. Bu dosyanın adını CargoCrud.razor olarak belirledim.

Bu sayfada kargo ekleme, silme, güncelleme, listeleme işlemleri ve bunların yanında kargoların konumlarını görüntüleme işlemleri yapılabilir.

Blazor.Leaflet.Samples'teki Pages klasörüne eklediğimiz CargoTracking.razor ise kargonun map'ten takibini yapmamızı sağlıyor.

- Eklediğimiz kargoların konumlarını haritada Marker'lar ile görürüz. Aynı zamanda CargoCrud.razor'da da kargoların konumlarını gösteren bir harita bulunmaktadır.

Index.razor ana sayfamızdır. Projeyi çalıştırdığımızda direkt bu sayfaya yönlendiririliriz. Burada veritabanımızda tutulan kargoların özelliklerini görebiliriz. Böylece herhangi bir işlem yapmayacağımız zaman sadece tüm kargoları listelemek için Index.razor sayfasına gitmemiz gerekir.

Shapes.razor sayfasında ise bir harita bulunuyor. Merkezi nokta olarak Türkiye'yi gösteren bu haritada Marker'ı istediğimiz gibi hareket ettirerek bulunduğu konumun Latitude ve Longitude değelerini öğrenebiliriz.