SAKARYA
ÜNİVERSİTESİ

Web Programlama Proje Ödevi

Uçak Koltuk Rezervasyon Sistemi

Y225012058

İhsan Eren Delibaş

A Gurubu

Github: https://github.com/eren5854/FlightReservationProject.git

UÇAK KOLTUK REZERVASYON SİSTEMİ

İhsan Eren DELİBAŞ[1],

Özet

Bu çalışmada ulaşım hizmeti veren uçak firmalarının, müşterilerinin güvenli ve hızlı bilet alması için oluşturmuş oldukları websitelerinin nasıl çalıştığı incelenerek örnek bir websitesi oluşturulmuştur.

Giriş

Proje Microsoft şirketinin Asp.Net framework'ü içerisinde bulunan MVC yaklaşımı kullanılarak oluşturulmuştur. MVC model-view-controller kelimelerinin kısaltmasından oluşur. Model verilerin depolandığı class'lardır. View kullanıcı arayüzünün oluşturduğu kısımdır. Controller kullanıcının yaptığı isteklere göre model ile view arasında bağlantı oluşturur bir nevi köprü gibidir.

MVC deseni, yazılımın modüler ve sürdürülebilir olmasına yardımcı olur. Her bir bileşenin belirli bir sorumluluğu vardır ve bu bileşenler birbirinden bağımsızdır. Bu, uygulamanın bakımını kolaylaştırır, kodun daha okunabilir ve sürdürülebilir olmasını sağlar. MVC deseni, özellikle web uygulamalarında ve masaüstü uygulamalarında sıkça kullanılır. Bu desen, birçok programlama dilinde ve çeşitli platformlarda uygulanabilir.

Projede kullanılan teknolojiler ve kütüphaneler; Asp.Net 7 MVC, MS Sql Server, Entity Faramework SqlServer, Entity Framework Tools.

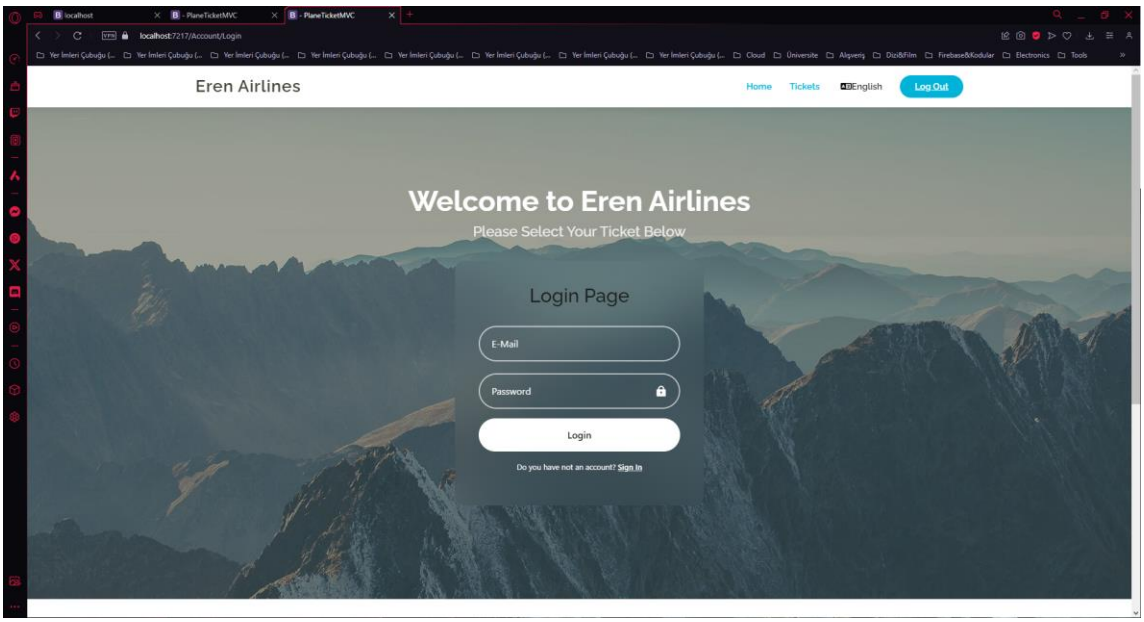
Bu projede karşılanmış olan beklentiler;

1- Kullanıcıların bilet seçiminde rahatlık sağlayan kullanıcı arayüzü (Front-End).
2- Yönetici rolünde (Admin) olan kişinin uçak tipi ve güzergâh eklemesini sağlayan Admin paneli.
3- Kullanıcıların kayıt yapmasını sağlayan kayıt sayfası.
4- Türkçe ve ingilizce olmak üzere iki adet dil desteği.
5- Yetki kontrolü yaparak kayıtlı olmayan kullanıcıların erişim kısıtlaması ve kayıtlı olan kullanıcıların admin paneline erişim engeli sağlanmıştır.
6- Uygulamanın windows makinede ilk defa çalışması durumunda kayıt kontrolü yaparak varsayılan admin ve güzergâh kayıtları ataması yapılır (Program.cs içerisinde anlatılacaktır).
7- Uygulama içerisinde bir adet API hizmeti bulunmaktadır (Ticket bölümünde detaylı anlatılacaktır.).

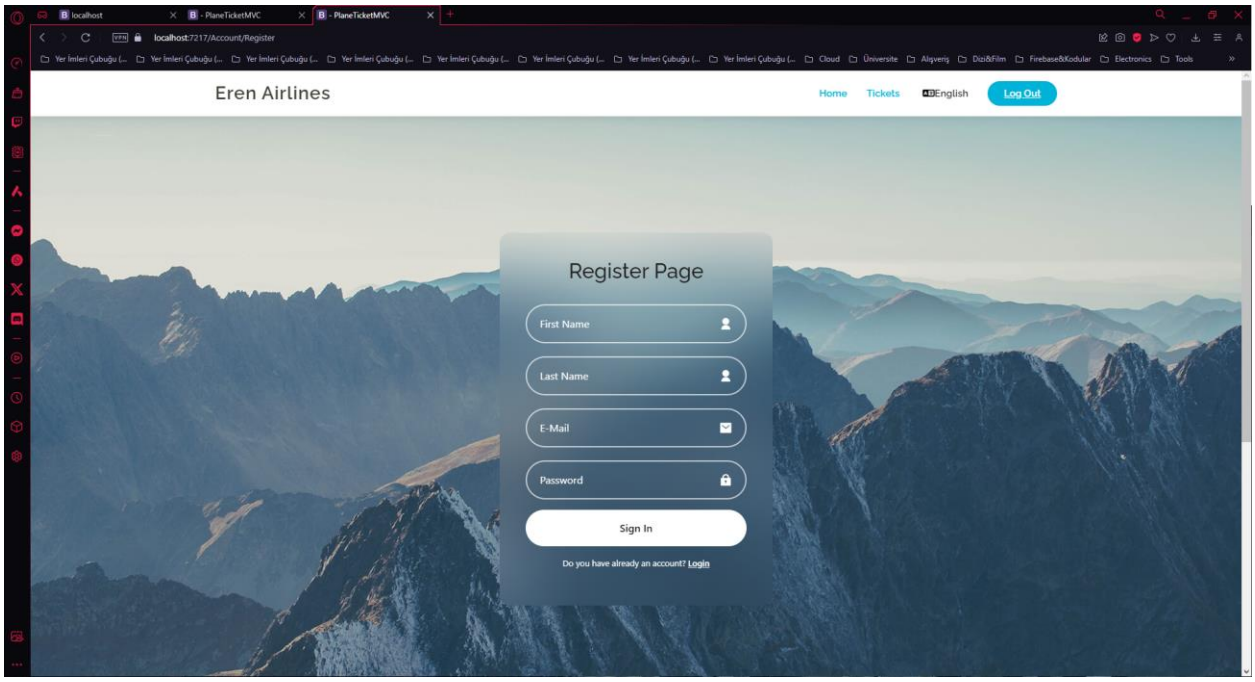
Bölüm 1

Bu bölümde projenin UI ve Admin paneli kısaca bahsedilecektir.

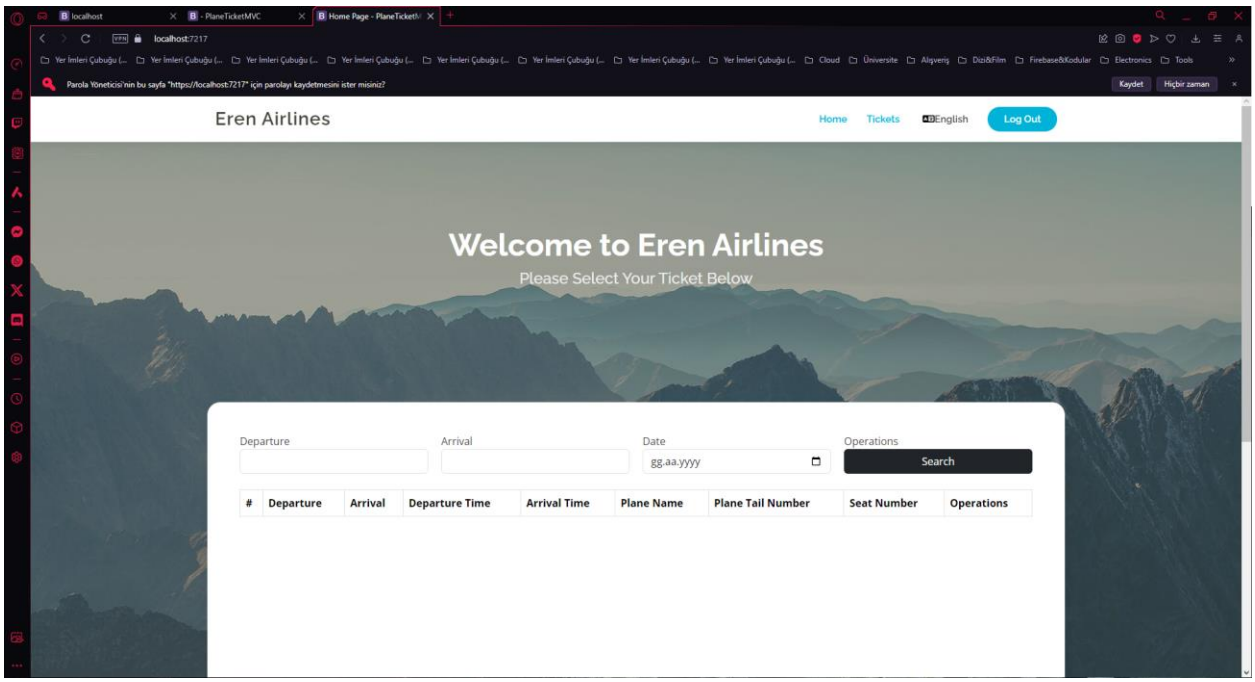
[1] Bilgisayar Müh. Yüksek Lisans Sakarya Üniversitesi, eren.delibas@ogr.sakarya.edu.tr



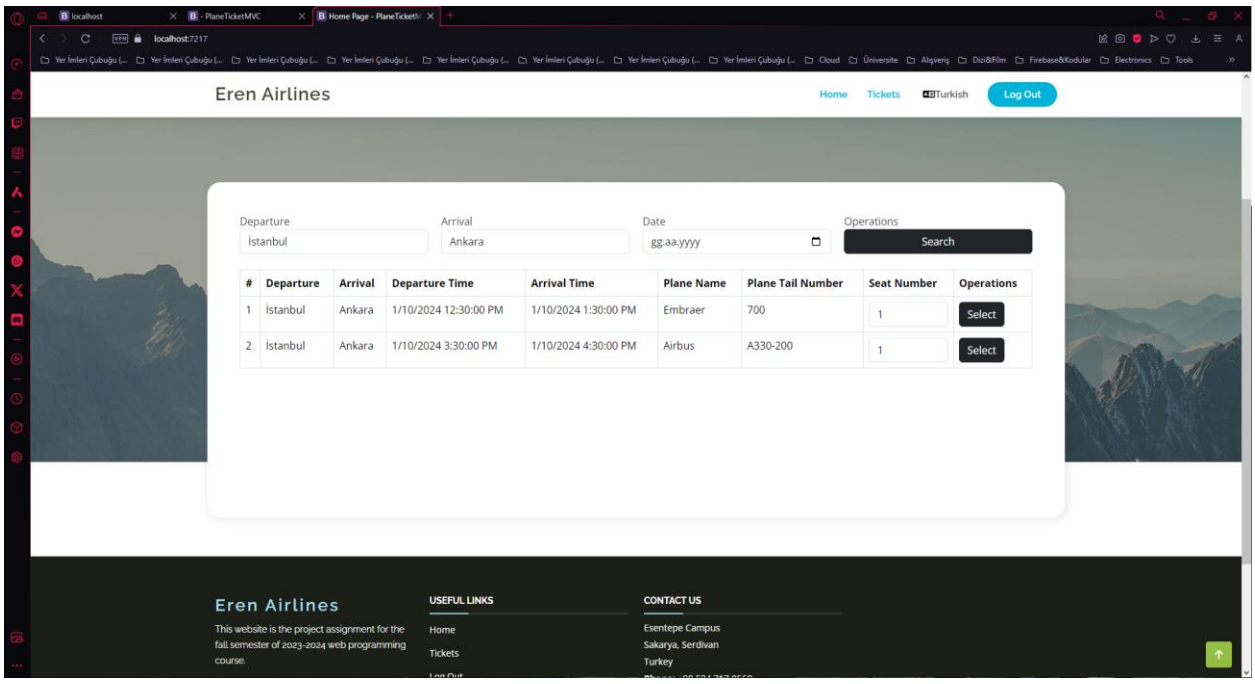
Uygulama çalıştığında ilk olarak AccountController'a bağlı olan Login sayfası açılır. Kullanıcı burada eğer kayıtlıysa, e-posta adresini ve şifresini doğru girdiyse anasayfaya yönlendirilir. Admin kullanıcısı bilgilerini girdiyse admin paneline yönlendirilir burada kullanıcıların rolüne göre bir yetkilendirme yapıldığı görülür. Eğer kayıtlı bir kullanıcı yoksa aşağıda bulunan signin etiketiyle kayıt olma sayfasına yönlendirilir.



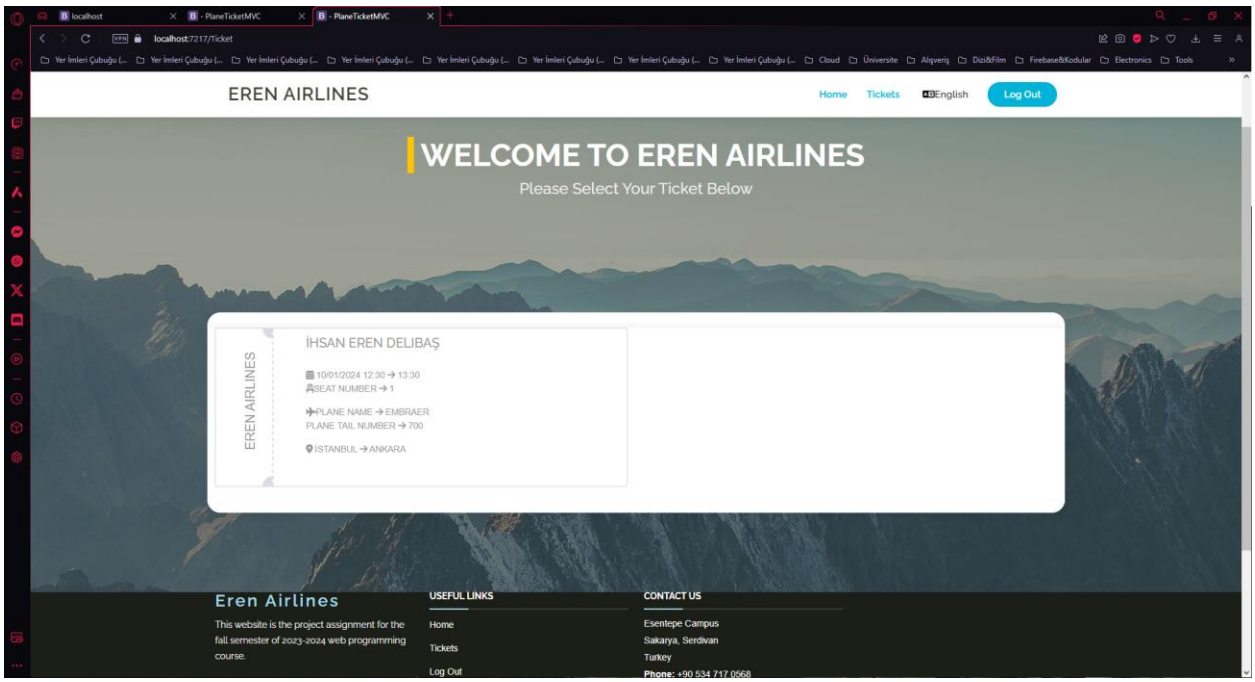
Kayıt sayfasında kullanıcı adını, soyadını, e-posta adresini, şifresini girerek kayıt olur. Kullanıcı tüm inputları doldurmalıdır ve e-posta adresini "@" sembolü ile yazmalıdır. Aksi halde kayıt yapılamamaktadır. Kayıt butonuna tıklandığında bir kayıt tamamlandı ekranı ile karşılanır ve login sayfasına gidilmesi sağlanır. Eğer girmiş olduğu e-posta adresi zaten kayıtlıysa en alt kısımda kırmızı yazıyla uyarılır. Kayıtlı olduğunu gören kullanıcı altta bulunan "Giriş Yap" etiketiyle tekrar giriş ekranına dönebilir.



Giriş yapıldıktan sonra HomeController'a bağlı Index sayfası karşımıza gelir. Burada kullanıcıdan/müşteriden kalkış yeri, varış yeri ve hangi tarihte gideceğini isteyen girişler bulunmaktadır. Arama butonuna tıklandığında girilen bilgiler doğrultusunda güzergâh kayıtları getirilir.

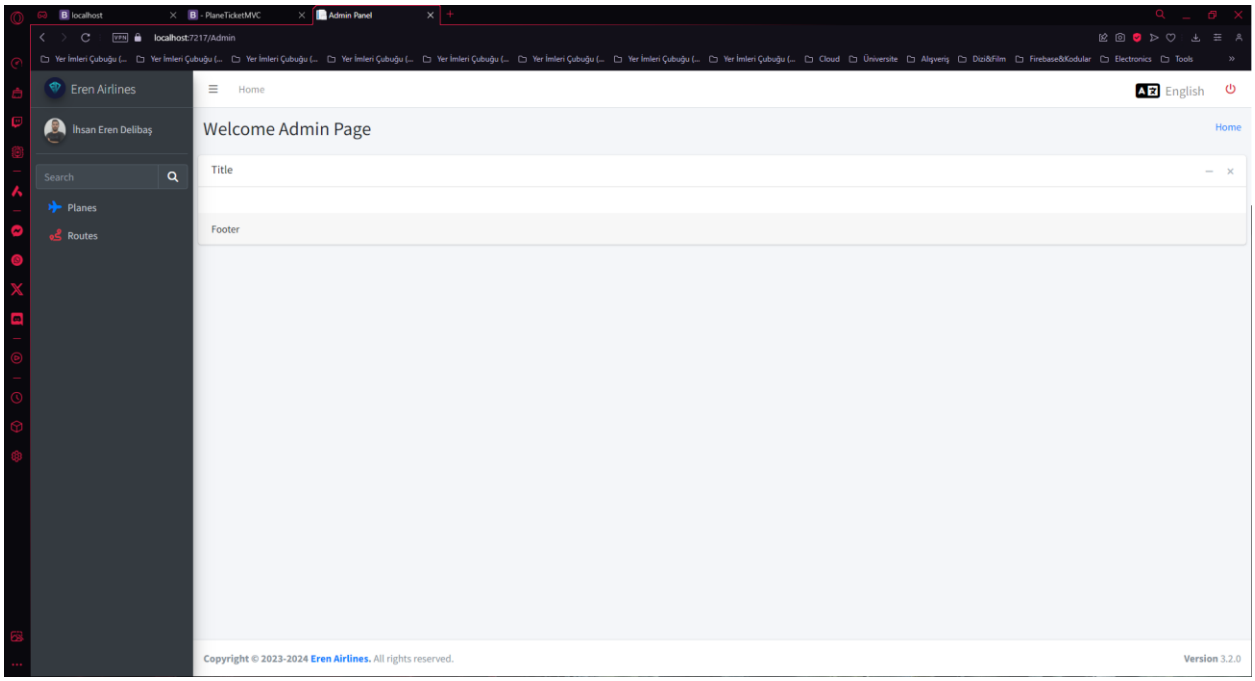


Aynı gün içerisinde farklı saatlerde kayıtlı güzergahlar varsa zamana göre sıralanarak kayıtlar getirilir. Kullanıcı koltuk numarasını seçtikten sonra "Select" butonu ile istediği güzergahı seçer ve ardından Ticket sayfasına yönlendirilir. Uçuş kaydı tamamlanır ve bilet oluşturulur.

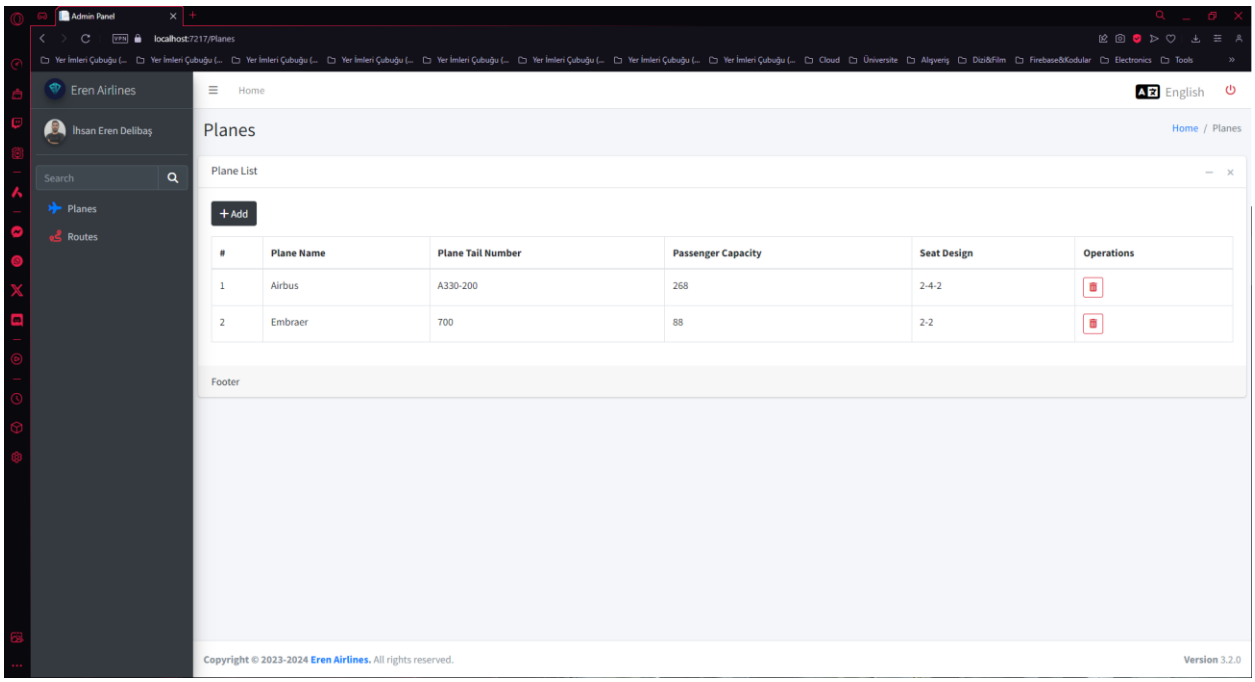


TicketController'a bağlı Index sayfası yukarıdaki gibidir. Kullanıcının ad, soy ad ve seçmiş olduğu uçuş bilgileri bir bilet tasarımı ile kullanıcıya bildirilir.

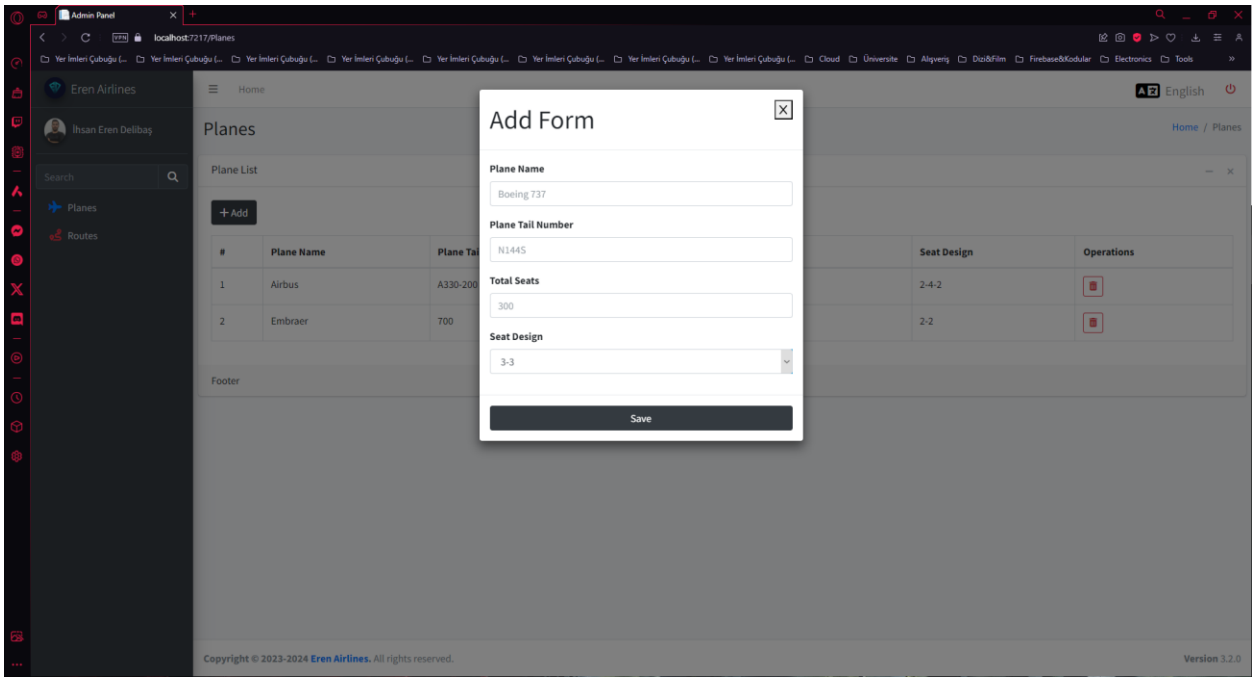
Kullanıcı sağ yukarıda bulunan dil değiştirme seçeneği ile dili değiştirebilir. Çıkış yapmak için "Çıkış Yap" butonuna tıklanır ardından tekrar giriş yap ekranına yönlendirilir.



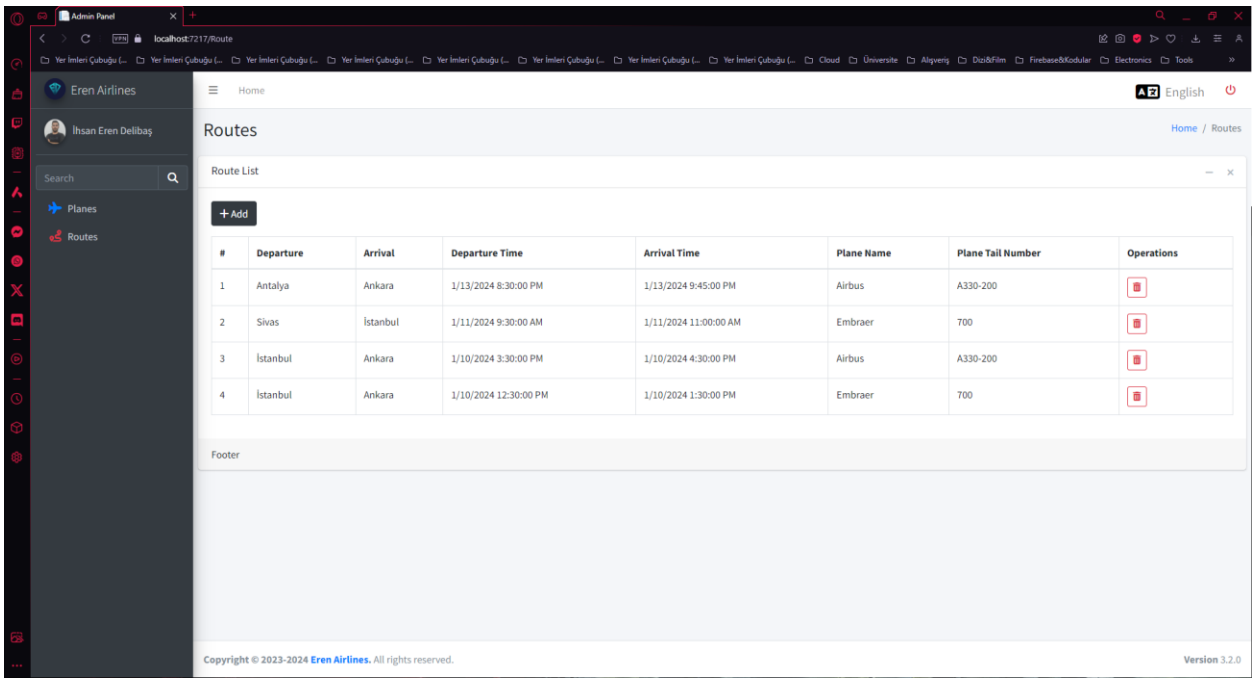
"Giriş Yap" sayfasında eğer admin bilgileri girilerek giriş yapıldıysa AdminController'a bağlı Index sayfasına yönlendirilir. Sol tarafta bulanan menü ile uçaklar ve güzeregah sayfalarına geçiş yapılır.



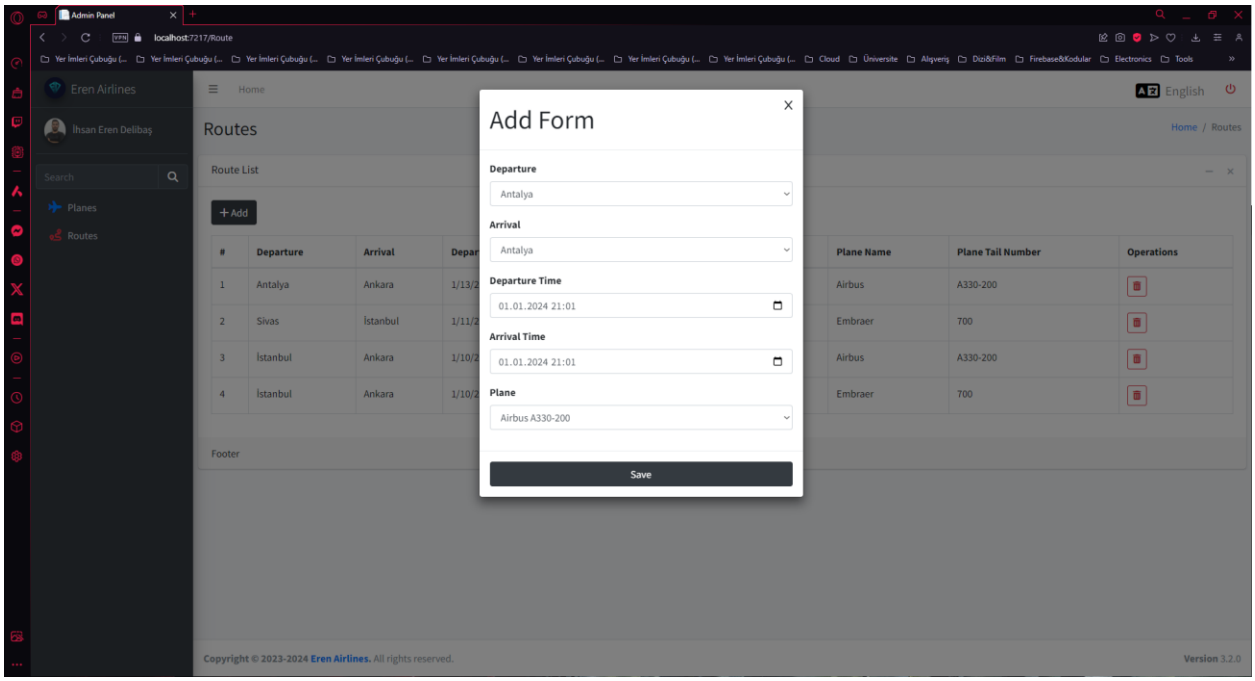
Uçaklar sayfası açıldığında kayıtlı uçaklar listesi tablosu karşımıza gelir. Tablonun sol üstünde bulunan ekle butonu ile bir modal açılır ve kayıtlar oraya yapılır.



Açılan modal'a uçak ismi, kuyruk numarası, toplam koltuk ve koltuk düzeni bilgisi girilerek kaydedilir. Kayıt işleminden sonra tekrar uçak listesi tablosu güncellenmiş şekilde karşımıza gelir.



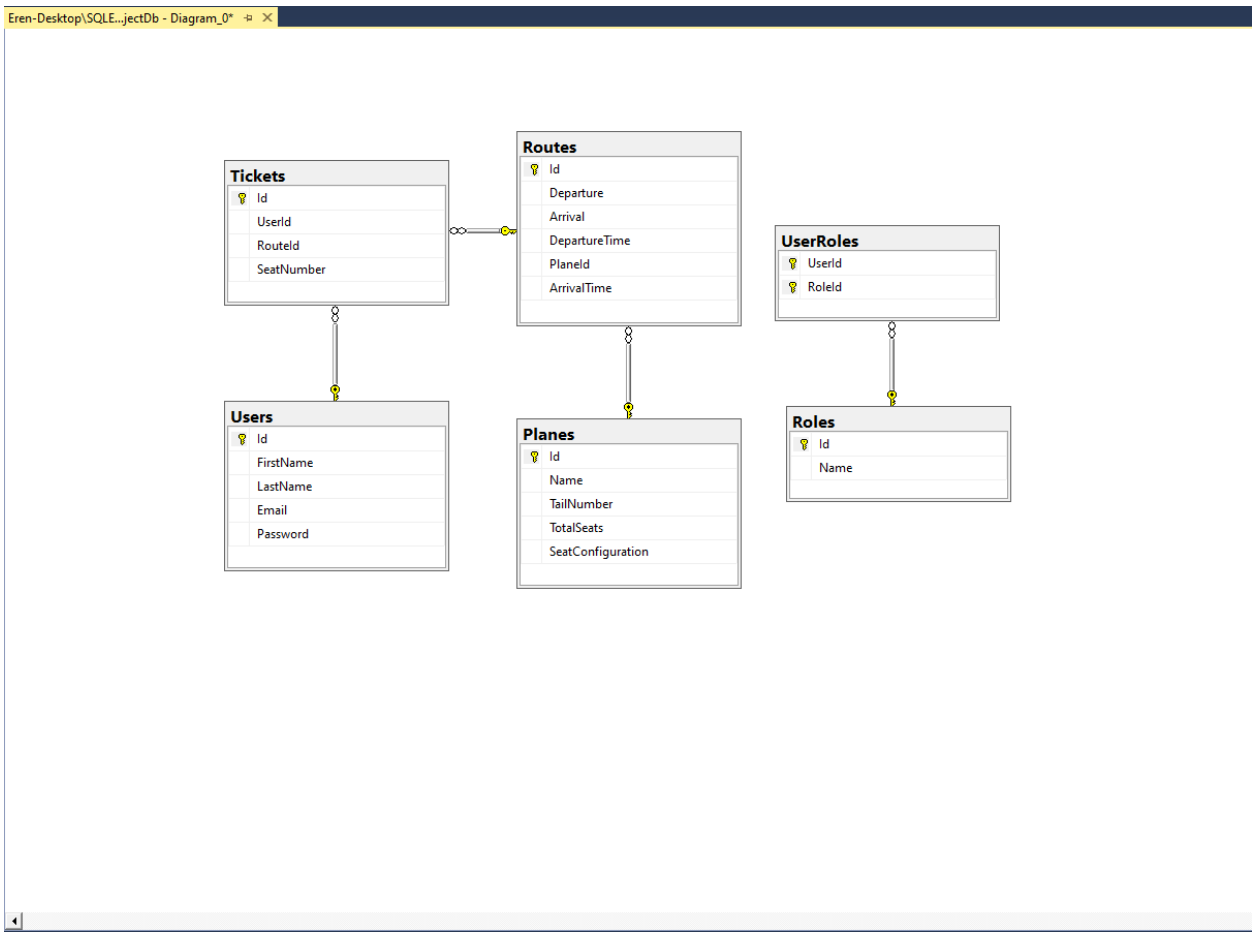
Güzergâh sayfası açıldığında kayıtlı güzergâh listesi tablosu karşımıza gelir. Tablonun sol üstünde bulunan ekle butonu ile bir modal açılır ve kayıtlar oraya yapılır.



Açılan modal'a kalkış yeri, varış yeri, kalkış tarihi ve saati, varış tarihi ve saati ve uçak bilgisi girilerek kaydedilir. Kayıt işleminden sonra tekrar güzergâh listesi tablosu güncellenmiş şekilde karşımıza gelir.

Bölüm 2

Bu bölümde projenin veri tabanı yapısı kısaca bahsedilecektir.



Veri tabanı diyagramı yukarıdaki gibidir. Toplam 6 tablodan oluşur. Güzergâh ataması esnasında uçak bilgilerine de ihtiyaç duyulduğu için Planes tablosu ile Routes tablosu ilişkilidir. Bilet alındıktan sonra bilet bilgileri Tickets tablosuna kaydedilir. Bilet içeriğinde güzergâh bilgileri ve kullanıcı bilgileri de ihtiyaç duyulduğu için Ticket tablosu ile Routes ve Users tabloları ilişkilidir.

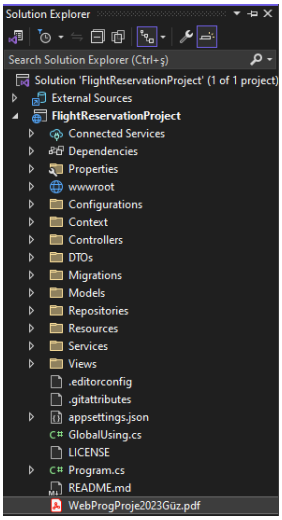
Roles tablosuna yetkilendirme için ihtiyaç duyulur. Burada admin ya da user olmak üzere iki adet veri tutulur. UserRoles tablosu ile Roles tablosu ilişkilidir çünkü kullanıcının yetkilendirilmesi bu tablolar ile gerçekleştirilir.

Users tablosunda kullanıcının kayıt yapılması için gereken standart bilgilerin sütunları eklenmiştir.

Planes tablosuna uçak bilgilerini detaya inmeden gerekli sütunları eklenmiştir.

Bölüm 3

Bu bölümde projenin back-end kısmı kısaca bahsedilecektir.



Yukarıdaki görselde projenin çözüm gezginindeki dosya yapısı görülmektedir. Toplam 10 adet önemli dosya vardır. Bu dosyalardan kısaca sırasıyla aşağıda açıklanmıştır.

1- wwwroot: Bu klasörde projenin css, js, img dosyalarının tutulduğu alandır. Frontend için önemli bir klasördür. Views de oluşturulan html yapıları kütüphaneleri ve css, js gibi dosyaları buradan çeker.
2- Configurations: Veri tabanında oluşturulacak tabloların ayarlamaları bu klasörde yapılır.
3- Context: Veri tabanı bağlantısı için gerekli dosya burada tutulur. Constructor yapısı içerisinde kurulur daha sonra dependency injection ile program.cs'e dahil edilir.
4- Controllers: Models içerisinde bulunan class'lar ve Repositoryler içerisinde yapılan işlemlerle Views arasında köprü oluşturan dosyaların bulunduğu klasör.
5- DTOs: Data Transfer Object front-end(views) den gelen input verileri önce dto dosyasında ilgili alanlara geçici olarak kaydedilir. Daha sonra controllerda bu veriler çekilerek models deki classlara aktarılır. Best practice olduğu için uygulanan bir yöntemdir.
6- Migrations: Veri tabanına tablo yapıları gönderilmeden önce packet manager konsolunda add-migration uygulanır oluşan migration yapısı veri tabanına aktarılır.
7- Models: Verilerin depolandığı kısımdır. Veri tabanına gönderilen yada getirilen veriler buradaki değişkenlere depolanır.
8- Repositories: Burada gerekli işlemler yapılır. Controllerda yapılacak işlemler burada yapılarak controller'a aktarılır. Best practice olduğu için uygulanır.
9- Resources ve Services: Bu projede bu klasörler dil yapısını ayarlamak ve kullanmak için oluşturulmuştur.
10- Views: Front-end kısmı için gerekli olan html kodları buradaki dosyalara yazılır.

Program.cs dosyasında uygulamanın ayağa kalktığı esnada başlangıçta çalışması gereken yapılar eklenir.

```
#region Localization
//Localization yapısı
builder.Services.AddSingleton<LanguageService>();
builder.Services.AddLocalization(options => options.ResourcesPath = "Resources");

builder.Services.AddMvc()
    .AddViewLocalization()
    .AddDataAnnotationsLocalization(options =>
        options.DataAnnotationLocalizerProvider = (type, factory) =>
        {
            var assemblyName = new AssemblyName(typeof(SharedResource).GetTypeInfo().Assembly.FullName ?? "");
            return factory.Create(nameof(SharedResource), assemblyName.Name ?? "");
        });
builder.Services.Configure<RequestLocalizationOptions>(options =>
{
    var supportCultures = new List<CultureInfo>
    {
        new CultureInfo("en-US"),
        new CultureInfo("tr-TR")
    };

    options.DefaultRequestCulture = new RequestCulture(culture: "tr-TR", uiCulture: "tr-TR");
    options.SupportedCultures = supportCultures;
    options.SupportedUICultures = supportCultures;
    options.RequestCultureProviders.Insert(0, new QueryStringRequestCultureProvider());
});
#endregion
```

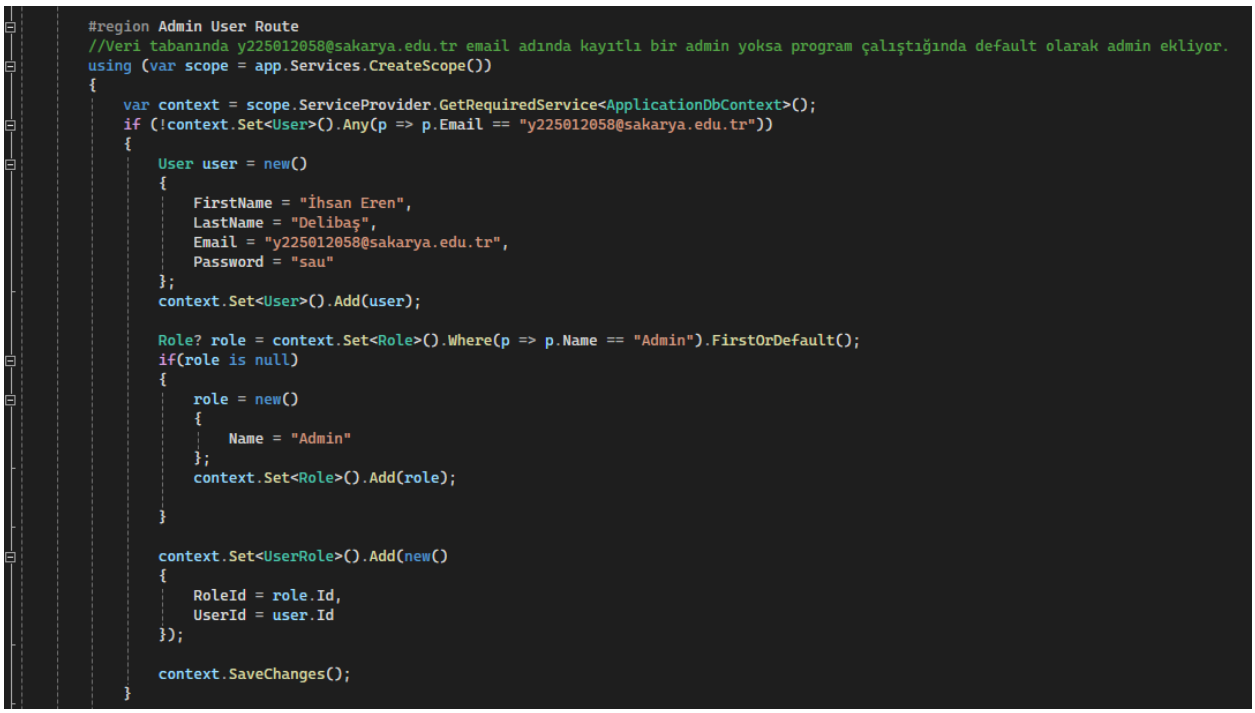
Localization: Dil için gerekli ayarlamalar yapılır. Projede 2 adet dil kullanılmıştır. Resources klasöründen ilgili dil tabloları program başlangıcında getirilir.

```
#region Context
//Sql Server connection.
builder.Services.AddDbContext<ApplicationDbContext>(opt =>
{
    opt.UseSqlServer(builder.Configuration.GetConnectionString("SqlServer"));
});
#endregion
```

Veri tabanı ile bağlantı kurmak için gerekli olan “Connnection String” appsettings.Development.json dosyasından çağırılır.

```
#region Authentication
//Authentication yapısı. Yetkisiz bir durumda otomatik olarak login sayfasına yönlendirilecek.
builder.Services.AddAuthentication("CookieAuth").AddCookie("CookieAuth", configuration =>
{
    configuration.AccessDeniedPath = "/Account/Login";
    configuration.LogoutPath = "/Account/Login";
    configuration.ExpireTimeSpan = TimeSpan.FromHours(1);
    configuration.Cookie.Name = "UserLoinCookie";
});
#endregion
```

Authentication yapısı bir sayfaya yetkisiz erişim talebinde bulunması durumunda varsayılan sayfaya yönlendirmek için kurulmuştur. Aynı zamanda çıkış yapıldığında gidilecek sayfa ve giriş yapan kullanıcının ne kadar süre sistemde duracağı da ayarlanmıştır.



```
#region Admin User Route
//Veri tabanında y225012058@sakarya.edu.tr email adında kayıtlı bir admin yoksa program çalıştığında default olarak admin ekliyor.
using (var scope = app.Services.CreateScope())
{
    var context = scope.ServiceProvider.GetRequiredService<ApplicationDbContext>();
    if (!context.Set<User>().Any(p => p.Email == "y225012058@sakarya.edu.tr"))
    {
        User user = new()
        {
            FirstName = "İhsan Eren",
            LastName = "Delibaş",
            Email = "y225012058@sakarya.edu.tr",
            Password = "sau"
        };
        context.Set<User>().Add(user);

        Role? role = context.Set<Role>().Where(p => p.Name == "Admin").FirstOrDefault();
        if(role is null)
        {
            role = new()
            {
                Name = "Admin"
            };
            context.Set<Role>().Add(role);

        }

        context.Set<UserRole>().Add(new()
        {
            RoleId = role.Id,
            UserId = user.Id
        });

        context.SaveChanges();
    }
```

Bu kısımda da program başlangıcında eğer veri tabanında kayıtlı bir admin, kullanıcı, uçak bilgisi ve güzergâh yoksa oluşturulması sağlanıyor.

SONUÇ

Sonuç olarak piyasada bulunan uçak bileti satın alma işlemini gerçekleştiren firmaların tasarlamış oldukları websitelerinin bir örneği bu projede uygulanmaya çalışılmıştır.