2024

BAU-WISSEN AKADEMIE & ISTKA
Full Stack Programlama Eğitimi
Bitirme Projesi

ASP .NET CORE MVC KULLANARAK REPOSİTORY PATTERN İLE N-KATMANLI WEB PROJESİ GELİŞTİRME

Hazırlayan:
Emir Öztürk

TEŞEKKÜR

İSTKA: Yazılım, Yetenek, Eğitim Ve İstihdam Merkezi Projesi & BAU Wissen Akademi kapsamında yürütülen ve %100 burslu olarak katıldığım Full Stack Developer eğitimimi tamamlamamda emeği geçen Ercan Öztürk ve Selkan Larlar hocalarıma sonsuz teşekkürlerimi sunarım.

16 Ağustos 2024 **Emir Öztürk**

İÇİNDEKİLER

Sayfa No

1. ÖZET

ASP .NET CORE MVC KULLANARAK
REPOSİTORY PATTERN İLE N-KATMANLI WEB PROJESİ GELİŞTİRME
Emir Öztürk
BAU-Wissen Akademie & ISTKA
Full Stack Developer Programı Bitirme Projesi
Ağustos 2024, 9 sayfa

Bu bitirme projesi, ASP.NET Core MVC kullanılarak geliştirilen ve Repository pattern ile N-katmanlı bir mimariye sahip bir web uygulamasıdır. Projemin temel konusu, günümüzün en büyük sorunlarından biri haline gelen yüksek kira ücretlerini hafifletmeye yönelik bir çözüm sunmak amacıyla bir ev ve oda paylaşım platformu oluşturmaktır. Bu platform, kullanıcıların üye olabileceği, üye olurken güvenli bir şekilde kişisel bilgilerini paylaşabileceği, ilan verebileceği ve mevcut ilanlara göz atabileceği bir yapı sunar.

Site, iller, ilçeler ve semtler gibi çeşitli kriterlere göre arama yapmayı sağlayan gelişmiş bir filtreleme sistemi içerir. Bu özellik, kullanıcıların istedikleri konumda ve bütçede uygun bir oda veya ev bulmalarını kolaylaştırır. Ayrıca, üyelerin ilanlarını yönetmelerine ve görüntülemelerine olanak tanıyan kullanıcı dostu bir arayüze sahiptir.

Proje, SOLID prensiplerine uygun olarak tasarlanmış beş ayrı katmandan oluşmaktadır: Veri Erişim Katmanı, İş Mantığı Katmanı, Servis Katmanı, Sunum Katmanı ve Ortak Katman. Her bir katman, projenin modülerliğini ve test edilebilirliğini artırmak amacıyla ayrı ayrı ele alınmıştır. Veri erişim işlemleri, Repository pattern kullanılarak izole edilmiş ve iş mantığı katmanı ile veri erişim katmanı arasındaki bağımlılıkları en aza indirilmiştir.

Bu projeyi hayata geçirmemdeki asıl amaç, 480 saatlik yoğun bir eğitim sürecini başarıyla tamamladığımı ve bu süreçte kazandığım bilgi ve becerilerle mezun olmaya ve başarı sertifikasını almaya hak kazandığımı göstermektir. Proje boyunca, öğrendiğim

teorik bilgileri pratikte uygulama fırsatı buldum ve yazılım geliştirme süreçlerindeki yetkinliğimi daha da ileriye taşıdım.

2. GİRİŞ

Projemin daha kolay test edilebilir, kullanıcı dostu ve yararlı olması çeşitleri teknolojileri kullandım. Aşağıda, projede kullandığım başlıca teknolojiler ve yapılar kısa açıklamalarıyla birlikte sunulmuştur:

ASP.NET Core MVC:

Projemde ASP.NET Core MVC'yi kullanarak Model-View-Controller (MVC) mimarisi ile web uygulaması geliştirdim. Bu mimari, projede veri işleme (Model), kullanıcı arayüzü (View) ve iş mantığını (Controller) birbirinden ayırarak kodun daha organize ve modüler olmasını sağladı.

Entity Framework Core:

Veritabanı işlemlerini yönetmek için Entity Framework Core'u kullandım. Bu ORM (Object-Relational Mapping) aracı, veritabanı ile .NET nesneleri arasında köprü kurarak veri erişim kodlarını daha kolay ve anlaşılır hale getirdi. Ayrıca, veritabanı işlemlerini code first yaklaşımıyla yöneterek, veritabanı şemasını kod üzerinden oluşturup güncelleyebildim.

JavaScript:

Kullanıcı deneyimini geliştirmek için JavaScript kullanarak sayfalarda dinamik ve etkileşimli bileşenler oluşturdum. JavaScript ile form doğrulama, veri filtreleme ve sayfa içi veri güncellemeleri gibi işlemleri gerçekleştirdim.

CSHTML Sayfalar:

Kullanıcı arayüzü tasarımı için Razor View Engine ile oluşturulan CSHTML sayfalarını kullandım. Bu sayfalar, HTML, CSS ve C# kodlarını bir araya getirerek dinamik web içerikleri oluşturmamı sağladı.

Identity Scaffold:

Kullanıcı kimlik doğrulama ve yetkilendirme işlemleri için ASP.NET Core Identity scaffold kullanarak, kullanıcı yönetimi ve kimlik doğrulama mekanizmalarını projeme entegre ettim. Bu sayede, kullanıcıların güvenli bir şekilde üye olmasını, oturum

açmasını ve yetkilendirilmiş işlemleri gerçekleştirmesini sağladım.

Areas ile Sayfa Yönetimi:

Projemde farklı modülleri ve işlevleri organize etmek amacıyla Areas yapısını kullandım. Bu yapı, projenin modülerliğini artırarak, farklı bölümlerin (örneğin, kullanıcı ve yönetici bölümleri) ayrı ayrı yönetilmesini sağladı.

Repository Design Pattern:

Veritabanı erişimini yönetmek ve iş mantığını daha bağımsız hale getirmek için Repository design pattern'i kullandım. Bu pattern, veri erişim işlemlerini soyutlayarak, kodun test edilebilirliğini ve sürdürülebilirliğini artırdı.

İkonlar:

Projede kullanıcı arayüzünü zenginleştirmek ve daha görsel bir deneyim sunmak amacıyla çeşitli ikon setlerini kullandım. İkonlar, butonlar ve menüler gibi kullanıcı etkileşim noktalarını daha anlaşılır hale getirdi.

Code First:

Veritabanı tasarımında Code First yaklaşımını benimsedim. Bu yöntemle, veri modellerini kod olarak tanımlayıp, Entity Framework Core üzerinden veritabanını otomatik olarak oluşturdum ve yönettim. Bu yaklaşım, veritabanı tasarımı üzerinde tam kontrol sağlamamı ve gerektiğinde kolayca değişiklik yapmamı mümkün kıldı.

3. PROJE İÇERİĞİ

Projemin yaşam döngüsü şu şekilde işlemektedir. Bir kullanıcı siteye girdiğinde yayınlanmış ilanları görebilir filtreleme yapabilir fakat ilanı yayınlayan kişinin bilgilerini göremez. Siteye üye olduktan sonra ilanın detaylarında ilanı yayınlayan kişinin detaylarını görebilir. Üye olmayan kişi ilan oluşturamaz. İlan oluşturmak için üye olmanız gerekmektedir. Siteye giriş yapan kişi profil kısmından yayınladığı ilanları inceleyebilir, güncelleyebilir veya silebilir. Profil sayfasından ilgili iletişim bilgilerini ve sosyal medya bilgilerini güncelleyebilir.

Admin kişisi sistem tarafından girilmiştir ve üye olan her kullanıcı normal user

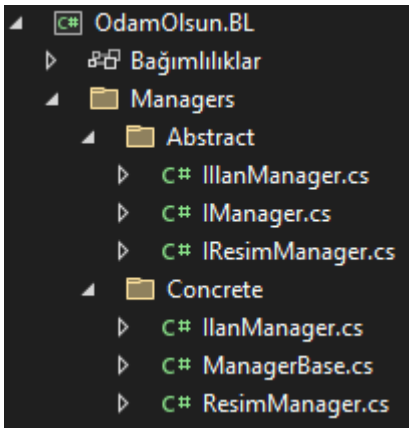
olarak kaydedilmektedir. Admin yayınlanan bütün ilanları görebilir, güncelleyebilir veya silebilir. Normal kullanıcı sadece kendi yayınladığı ilanlar üzerinde bu yetkiye sahiptir.

İlan yayınlarken çoklu fotoğraf eklemesi yapılabilir. İl, ilçe ve semt verileri API den çekilmektedir. İlanların görüntülenmesi sırasında sistemin yorulmaması için sayfalama kullanılmıştır. UI ve UX olarak daha etkili olması için popup uyarılar ve toastr mesajlar kullanılmıştır.

Kullanıcının sistemde sürekli aktif kalabilmesi için 'Remember Me' alanı konulmuş ve aktif bir şekilde çalışmaktadır.

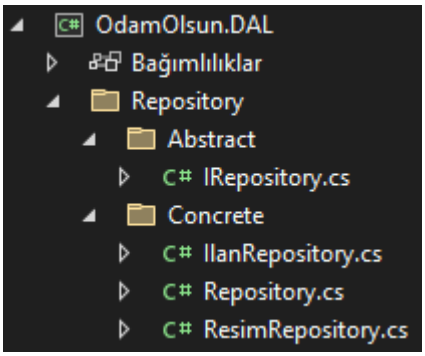
4. PROJEDE KULLANILAN KATMANLAR

BL (Business Logic) Katmanı, yazılım projelerinde iş kurallarının ve iş mantığının yer aldığı katmandır. Bu katman, veri erişim (DAL) ile kullanıcı arayüzü (UI) arasında köprü görevi görür. İşlemler burada tanımlanır, veri işlenir ve gerekli olan her türlü mantık burada uygulanır.



Projemde BL katmanı resimde görüldüğü gibi kullanılmıştır ve SOLID prensipleri çerçevesinde uygulanmıştır.

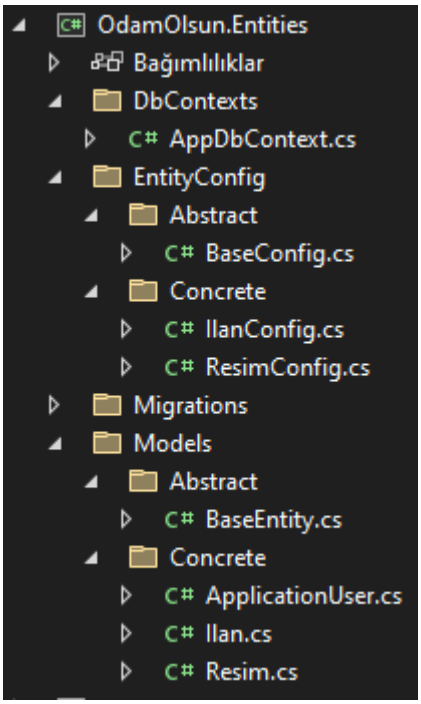
DAL (Data Access Layer) Katmanı, yazılım projelerinde veritabanına erişim işlemlerinin yönetildiği katmandır. Bu katmanda, veritabanı ile ilgili CRUD (Create, Read, Update, Delete) işlemleri yapılır. DAL, veritabanı ile doğrudan iletişim kurarak veri alır, günceller, ekler veya siler, bu işlemleri gerçekleştirirken iş mantığına göre belirlenmiş olan kuralları uygular.



Projemde DAL katmanı resimde görüldüğü gibi kullanılmıştır ve CRUD işlemleri için asenkron methodlar içermektedir.

Entities Katmanı, yazılım projelerinde veritabanında saklanan veri modellerinin tanımlandığı katmandır. Bu katmanda, uygulamanın veri yapıları ve ilişkileri, genellikle sınıflar (classes) olarak temsil edilir. Entity sınıfları, veritabanı tablolarını ve bu tablolar arasındaki ilişkileri karşılar. Bu katman, veri modellemesinin temelini oluşturur ve uygulamanın veri tabanıyla olan etkileşimlerinde kullanılır.

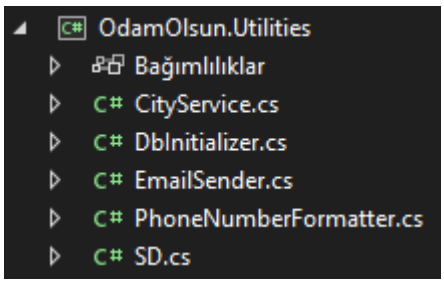
Code First yaklaşımı, Entity Framework Core'da veritabanı şemasını kod üzerinden tanımlamanızı sağlar. Bu yaklaşımda, veritabanı yapısı ve ilişkileri, doğrudan C# sınıfları (entity'ler) ve bu sınıfların özellikleri aracılığıyla tanımlanır. Veritabanı şeması oluşturulurken veya güncellenirken, Entity Framework Core bu sınıfları ve özellikleri kullanarak otomatik olarak veritabanı tabloları ve ilişkileri oluşturur.



Projemde Entities katmanı resimde görüldüğü gibi kullanılmıştır ve modeller arası

bire çok veya çoka çok gibi ilişkisel bağlantıları içermektedir. Code First yaklaşımı ile geliştirilmiştir.

Utilities Katmanı, yazılım projelerinde genellikle yardımcı sınıflar ve işlevlerin bulunduğu katmandır. Bu katman, uygulamanın çeşitli yerlerinde tekrar eden veya genel amaçlı işlevleri sağlayan yardımcı kodları içerir. Bu tür işlevler, veri dönüştürme, yardımcı metodlar, ortak işlemler ve genellikle uygulama genelinde kullanılan kod parçacıkları olabilir.



Projemde Utilities katmanı resimde görüldüğü gibi kullanılmıştır. Bu katmanda projemde telefon numaralarını UI tarafına düzgün göndermem için kullanılan, veritabanından plaka kodu ile gelen ilin şehire string olarak dönüştürülmesini sağlayan, Rollerin tutulduğu sınıf vb. gibi yardımcı sınıflar kullandım.

Web Katmanı, modern yazılım projelerinde, kullanıcıların uygulama ile etkileşimde bulunmasını sağlayan önemli bir katmandır. Bu katman, genellikle web tabanlı uygulamalarda, kullanıcı arayüzü (UI), kullanıcı deneyimi (UX) ve kullanıcı etkileşimleri ile ilgili tüm bileşenleri içerir. Web katmanının temel işlevleri ve bileşenleri şunlardır:

Kontrolörler (Controllers):

Web katmanının en önemli bileşenlerinden biri olan kontrolörler, kullanıcıdan gelen istekleri işleyerek uygun yanıtları üretir. ASP.NET Core MVC'de kontrolörler, uygulamanın iş mantığını ve iş akışını yöneten sınıflardır. Her bir kontrolör, belirli bir işlevi yerine getiren yöntemlere (action methods) sahip olabilir. Bu yöntemler, HTTP isteklerini alır, ilgili verileri işleyerek uygun görünüme (View) veya API yanıtına yönlendirir.

Görünümler (Views):

Görünümler, kullanıcı arayüzünü oluşturan ve veriyi kullanıcıya sunan HTML sayfalarıdır. ASP.NET Core MVC'de Razor View Engine kullanılarak oluşturulan

CSHTML dosyaları, dinamik içerikler üretir ve verileri kullanıcıya sunar. Görünümler, kontrolörlerden gelen verileri alır ve HTML ile CSS kullanarak kullanıcıya görsel olarak sunar.

Modeller (Models):

Web katmanında, genellikle kullanıcıya sunulacak verilerin yapısını belirleyen modeller bulunur. Modeller, genellikle view'lerde kullanılacak veri yapılarını temsil eder ve kullanıcı arayüzü ile iş mantığı arasındaki veri aktarımını sağlar. ASP.NET Core MVC'de bu modeller, kullanıcıdan alınan verilerin doğruluğunu ve geçerliliğini kontrol etmek için kullanılır.

JavaScript ve CSS:

Dinamik ve etkileşimli kullanıcı arayüzleri oluşturmak için JavaScript kullanılır. Web katmanında, kullanıcı etkileşimlerine yanıt veren ve sayfa içi dinamik işlemleri gerçekleştiren JavaScript kodları bulunur. CSS ise, görsel tasarımı ve stil uygulamalarını yönetir, böylece kullanıcı arayüzü estetik ve kullanıcı dostu hale gelir.

API Endpoint'leri:

Web katmanında, uygulamanın dış dünya ile veri alışverişi yapmasını sağlayan API endpoint'leri bulunur. RESTful API'ler, genellikle HTTP isteklerini işleyerek JSON veya XML formatında veri sağlar ve istemcilerle etkileşimde bulunur.

Routing:

Web katmanında, kullanıcıların belirli URL'lere erişimini sağlayan routing (yönlendirme) yapılandırmaları bulunur. Routing, belirli URL desenlerine göre uygun kontrolör ve eylem yöntemlerini çağırır, böylece kullanıcı istekleri doğru şekilde yönlendirilir.

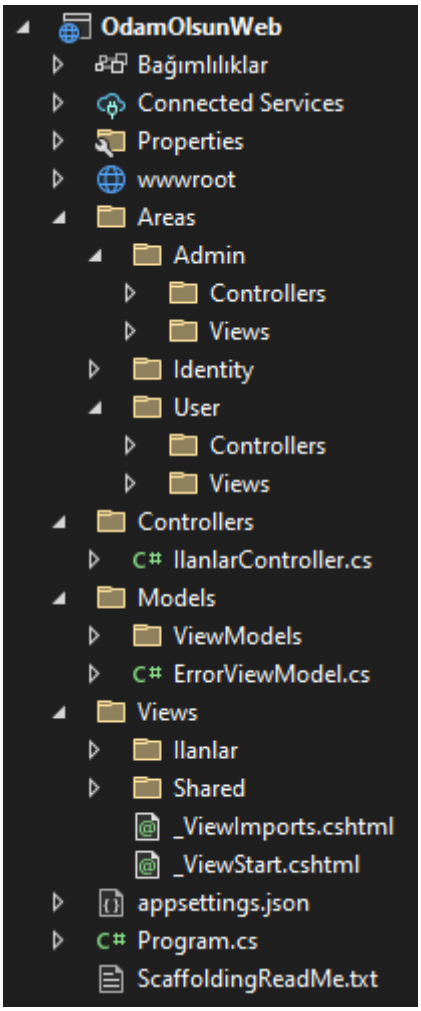
Doğrulama ve Güvenlik:

Web katmanında, kullanıcı girişi ve form doğrulama işlemleri yapılır. ASP.NET Core Identity gibi güvenlik özellikleri, kullanıcı kimlik doğrulama ve yetkilendirme işlemlerini yönetir, böylece uygulamanın güvenliği sağlanır.

Projemde, Web katmanını, kullanıcı arayüzünü yönetmek ve etkileşimleri yönlendirmek için geniş bir şekilde kullandım. Kontrolörler aracılığıyla kullanıcı isteklerini işledim, görünümlerle veriyi dinamik olarak sundum ve JavaScript ile kullanıcı

etkileşimlerini yönettim. Ayrıca, CSS ile estetik tasarımı sağladım ve API endpoint'leri ile veri alışverişini gerçekleştirdim. Web katmanı, kullanıcı deneyimini optimize etmek ve uygulamanın tüm görsel ve etkileşimli bileşenlerini yönetmek için merkezi bir rol oynar.

Projemde bulunan üyelerin şifreleri hashlenmektedir. Bu sayede kötü taraflı yazılımlardan ve saldırılardan üyelerin hesap bilgileri korunmayı amaçlanmaktadır.



Projemde Web katmanı resimde görüldüğü gibi kullanılmıştır.

Projede veritabanı bağlantısı için gereken connection string datası appsetting.json dosyasında tutulmuştur.

5. PROJEYİ ÇALIŞTIRMAK İÇİN GEREKENLER

Öncelikle projede veritabanı olarak MYSQL kullanılmıştır, codefirst yaklaşımı bulunmaktadır ve Asp .NET Core 8 kullanılmıştır. Projede bulunan resim vb. dosya yolları MacOS ve Linux cihazlar için ayarlanmıştır. Windows cihazda çalışması için dosya yollarının düzenlenmesi gerekmektedir. MySql localimizde bulunan databaseler arasında appsettings.json ve ApplicationDbContext dosyalarındaki bağlantı yollarında bulunan Database alanındaki ile aynı isimde bir database olmadığın emin olmalı ve ilgili bağlantı yolundaki şifreyi localimiz olan mysql şifresiyle değiştirmemiz gerekmektedir. Projenin çalışması için eğer migration bulunmuyorsa Entities katmanını terminalde açarak 'dotnet ef migrations add initDb' yazarak migration eklememiz gerekmektedir. Eklediğimiz migration'ı veritabanına yansıtmamız için aynı terminalde 'dotnet ef database update' yazmamız gerekmektedir.

6. GİRİŞ İÇİN GEREKLİ LOGİN BİLGİLERİ

Projede 2 adet rol bulunmaktadır. Bu roller Admin ve Customer rolleridir.

Admin rolü ile giriş yapmak için gereken kullanıcı bilgileri:

Email: admin@site.com

Password: Admin123!

Customer rolü ile giriş yapmak için gereken kullanıcı bilgileri:

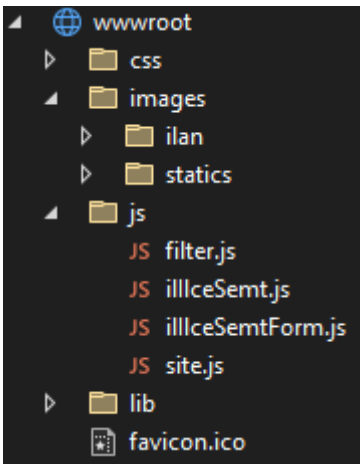
Email: emirozturk@site.com

Password: User123!

Customer rolü ile giriş yapmanın bir diğer yolu da sağ üstte bulunan register butonuna tıklayıp ilgili sayfadan üye olup giriş yapmaktır.

Seed data olarak bulunan ilanların userlarına bakmak için Utilities katmanındaki DbInitializer sınıfına göz atabilirsiniz. 6 farklı User bulunmaktadır. Bu şekilde İlanlarım sayfasının çalışıp çalışmadığını görebilirsiniz.

7. EKLER



Projedeki javascript dosyaları yukarıdaki resimde görüldüğü üzere wwwroot dosyasında tutulmaktadır. Aynı şekilde resimler de localde tutulmakta olup dosya yolu wwwroot içindeki images klasöründedir.

Projede kullanılan Api'nin dokümantasoyunu aşağıdaki linkte belirtilmiştir.

https://api.kadircolak.com/Konum/