

**YALOVA ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**- BİTİRME TEZİ -**

**TADAA! bir görev websitesi**

**MUHAMMED BERKAY ÇINAR**

**İLYAS SÜZMEN**

**Bitirme Tezi Danışmanı: DR. ÖĞR. ÜYESİ, GÜNEŞ HARMAN**

**YALOVA, 2024**

**YALOVA ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**TADAA! bir görev websitesi**

**MUHAMMED BERKAY ÇINAR**

**190101051**

**İLYAS SÜZMEN**

**190101086**

**1. Bitirme Tezi** **Danışmanı : Dr. Öğr. Üyesi Güneş Harman**

**2. Jüri Üyesi : Doç. Dr. Adem TUNCER**

**3. Jüri Üyesi : Dr. Öğr. Üyesi Yunus ÖZEN**

**Bitirme Tezinin Dönemi: 2023 – 2024 Bahar Yarıyılı**

**İÇİNDEKİLER**

Sayfa

KISALTMA LİSTESİ………………………………………………………………………...iii

ŞEKİL LİSTESİ……………………………………………………………………………….iv

ÖNSÖZ………………………………………………………………………………………...v

ÖZET…………………………………………………………………………….……………vi

ABSTRACT………………………………………………………………………………….vii

1. GİRİŞ……………………………………………………………………………1

1.1 Tez Çalışmasının Amacı…………………………………………………….......2

1.2 Literatür Taraması……………………………………………………………….3

2. MATERYAL VE METODOLOJİ……………………………………………....4

2.1 Araştırma Tasarımı……………………………………………………………....4

2.2 Veri Toplama Yöntemleri………………………………………………………..5

2.3 Analiz Yöntemleri………………………………………………………….…....5

2.4 Kullanılan teknolojiler ve Altyapı…………………………………………….....6

3. BACKEND TASARIMI VE DÜZENLENMESİ…………………………….…7

3.1 Neden Katmanlı Mimari Kullanılır……………………………………………...7

3.2 Katmanlı Mimarinin Oluşturulması……………………………………………...7

3.3 Projelerin Kodlanması………………………………………………………….10

3.4 Veri tabanı oluşturulması ve NuGet Paketleri Yüklenimi……………………...17

3.5 API servisinin Swagger ile test edilmesi……………………………………….19

4. ARAYÜZÜN OLUŞTURULMASI VE KULLANIMI………………………..24

4.1 Anasayfanın Oluşturulması…………………………………………………….24

4.2 Giriş Sayfalarının Oluşturulması……………………………………………….25

4.3 Kayır Sayfalarının Oluşturulması........................................................................33

4.4 Profil Sayfasının Oluşturulması………………………………………………..36

4.5 Takvim Sayfalarının Oluşturulması……………………………………………41

4.6 Not Sayfalarının Oluşturulması….………………………………………..........48

5. BULGULAR VE TARTIŞMA…………………………………………………60

6. SONUÇLAR VE ÖNERİLER………………………………………………….61

KAYNAKLAR

**KISALTMA LİSTESİ**

MSSQL Microsoft Server Structered Query Language

UPA Uygulama Programlama Arayüzü

TGO Tümleşik Geliştirme Ortamı

CSS Cascading Styles Sheets

HTML HyperText Markup Languge

ToDo Yapılacaklar Listesi

Vb. ve benzeri

##### ŞEKİL LİSTESİ

Sayfa

Şekil 3.1……………………………………………………………………………………… .8

Şekil 3.2....…………………………………………………………………………………….10

Şekil 3.3……………………………………………………………………………………… 11

Şekil 3.4…...…………………………………………………………………………………. 12

Şekil 3.5…...…………………………………………………………………………………..12

Şekil 3.6…..…………………………………………………………………………………...13

Şekil 3.7…..…………………………………………………………………………………...14

Şekil 3.8……………………………………………………………………………………... .15

Şekil 3.9…...…………………………………………………………………………………..16

Şekil 3.10.....…………………………………………………………………………………..17

Şekil 3.11..…………………………………………………………………………………….18

Şekil 3.12..…………………………………………………………………………………….19

Şekil 3.13..…………………………………………………………………………………….19

Şekil 3.14..…………………………………………………………………………………….19

Şekil 3.15..…………………………………………………………………………………….20

Şekil 3.16..…………………………………………………………………………………….20

Şekil 3.17..…………………………………………………………………………………….20

Şekil 3.18..…………………………………………………………………………………….21

Şekil 3.19..…………………………………………………………………………………….21

Şekil 3.20..…………………………………………………………………………………….21

Şekil 3.21..…………………………………………………………………………………….22

Şekil 3.22..…………………………………………………………………………………….22

Şekil 4.1....…………………………………………………………………………………….23

Şekil 4.2....…………………………………………………………………………………….23

Şekil 4.3....…………………………………………………………………………………….24

Şekil 4.4....…………………………………………………………………………………….24

Şekil 4.5....…………………………………………………………………………………….25

Şekil 4.6....…………………………………………………………………………………….26

Şekil 4.7....…………………………………………………………………………………….26

Şekil 4.8....…………………………………………………………………………………….27

Şekil 4.9....…………………………………………………………………………………….27

Şekil 4.10..…………………………………………………………………………………….28

Şekil 4.11..…………………………………………………………………………………….29

Şekil 4.12..…………………………………………………………………………………….29

Şekil 4.13..…………………………………………………………………………………….30

Şekil 4.14..…………………………………………………………………………………….30

Şekil 4.15..…………………………………………………………………………………….31

Şekil 4.16..…………………………………………………………………………………….31

Şekil 4.17..…………………………………………………………………………………….32

Şekil 4.18..…………………………………………………………………………………….33

Şekil 4.19..…………………………………………………………………………………….34

Şekil 4.20..…………………………………………………………………………………….35

Şekil 4.21..…………………………………………………………………………………….36

Şekil 4.22..…………………………………………………………………………………….37

Şekil 4.23..…………………………………………………………………………………….37

Şekil 4.24..…………………………………………………………………………………….38

Şekil 4.25..…………………………………………………………………………………….38

Şekil 4.26..…………………………………………………………………………………….38

Şekil 4.27..…………………………………………………………………………………….39

Şekil 4.28..…………………………………………………………………………………….40

Şekil 4.29..…………………………………………………………………………………….40

Şekil 4.30..…………………………………………………………………………………….41

Şekil 4.31..…………………………………………………………………………………….42

Şekil 4.32..…………………………………………………………………………………….42

Şekil 4.33..…………………………………………………………………………………….43

Şekil 4.34..…………………………………………………………………………………….44

Şekil 4.35..…………………………………………………………………………………….45

Şekil 4.36..…………………………………………………………………………………….45

Şekil 4.37..…………………………………………………………………………………….46

Şekil 4.38..…………………………………………………………………………………….47

Şekil 4.39..…………………………………………………………………………………….48

Şekil 4.40..…………………………………………………………………………………….49

Şekil 4.41..…………………………………………………………………………………….50

Şekil 4.42..…………………………………………………………………………………….51

Şekil 4.43..…………………………………………………………………………………….51

Şekil 4.44..…………………………………………………………………………………….52

Şekil 4.45..…………………………………………………………………………………….52

Şekil 4.46..…………………………………………………………………………………….53

Şekil 4.47..…………………………………………………………………………………….54

Şekil 4.48..…………………………………………………………………………………….54

Şekil 4.49..…………………………………………………………………………………….55

Şekil 4.50..…………………………………………………………………………………….55

Şekil 4.51..…………………………………………………………………………………….56

Şekil 4.52..…………………………………………………………………………………….57

Şekil 4.53..…………………………………………………………………………………….57

Şekil 4.54..…………………………………………………………………………………….58

Şekil 4.55..…………………………………………………………………………………….58

Şekil 4.56..…………………………………………………………………………………….59

Şekil 4.57..…………………………………………………………………………………….59

ÖNSÖZ

Bitirme tezi süresince benden yardımlarını esirgemeyen, akademik anlamda bize her zaman destek olan ve doğru yolu gösteren sayın danışman hocam Güneş Harman’a teşekkür ederiz. Sayın Harman, akademik bilgisi, tecrübesi ve özverisiyle beni her adımda destekleyerek bu tezin başarıyla tamamlanmasına önemli bir katkı sağlamıştır. Tez çalışmam sürecindeki destekleri ve katkıları için tüm bu sevgili insana minnettarlığımızı sunar, bize verdikleri güven ve destekle bu önemli aşamayı başarıyla tamamlamamızı sağlayan herkese içtenlikle teşekkür ederiz. Bu başarı, sizinle paylaşmanın gururunu taşıyoruz.

MUHAMMED BERKAY ÇINAR

İLYAS SÜZMEN

##### ÖZET

Bu Tez çalışması TODO (Yapılacaklar Listesi) web sitesi geliştirmeyi amaçlamaktadır. İçerisinde anasayfa: Kullanıcılar web sitesine girdiklerinde karşılarına çıkacak olan ilk sayfa, kullanıcı dostu ve erişilebilir bir tasarıma sahip olacak. Bu sayfada, kullanıcılara genel bir bakış sunulacak ve siteye üye olma, giriş yapma gibi işlemler kolayca gerçekleştirilebilecektir. Not alma: Kullanıcılar, görevleriyle ilgili notlar alabilecekleri özel bir alan bulacaklar. Bu bölüm, kullanıcıların görevleriyle ilgili detaylı bilgi girebilmelerini ve bu bilgileri düzenleyebilmelerini sağlayacaktır. Takvim sayfası, kullanıcı girişi, yapılacaklar listesi ve bunların tarihe göre seçimi yapılan bölümler bulunacaktır. Kullanıcıların kaydolmaları, giriş yapmalarını ve kişisel görev listelerini oluşturmalarını sağlayan bir platform oluşturulacaktır. Kullanıcılar, kişisel görev listelerini oluşturabilecek ve bu listeleri yönetebilecekler. Bu bölümde, yeni görev ekleme, mevcut görevleri düzenleme, tamamlama ve silme gibi işlemler kolaylıkla yapılabilecektir. Kullanıcılar, görevlerin durumlarını belirleyebilecek ve tamamlanan görevleri arşivleyebilecekler. Güvenlik açısından, kullanıcıların kişisel bilgileri şifrelenmiş bir şekilde saklanacak ve güvenli bir oturum yönetimi uygulanacaktır. Ana fonksiyonlar arasında kullanıcı istediğinde yeni görev ekleme, mevcut görevleri düzenleme, tamamlama ve silme gibi işlemler bulunacaktır. Ayrıca, kullanıcıların görevlere öncelik atama ve takvimle entegrasyon gibi gelişmiş özellikleri kullanmalarına imkan tanınacaktır. Arayüz ve arka uç tarafında kullanılacak teknolojiler de belirlenecek ve bu teknolojilerin kullanımıyla ilgili açıklamalar yapılacaktır. Tez çalışması işlevsel bir arka plan programlanması ve kullanıcı dostu bir arayüz tasarımına odaklanmalıdır.

##### ABSTRACT

This thesis project aims to develop a ToDo (Task List) website. The website will include the following components: The initial page users will encounter upon visiting the website will feature a user-friendly and accessible design. This page will provide an overview and allow users to easily perform actions such as signing up and logging in. There will be a section for taking notes, where users can record notes related to their tasks. This section will allow users to enter detailed information about their tasks and manage this information. The site will include a calendar page, user login, to-do list, and sections for selecting tasks by date. The platform will enable users to register, log in, and create personal task lists. In this section, users will be able to add new tasks, edit existing tasks, mark them as complete, and delete them with ease. Users will also be able to set the status of their tasks and archive completed ones. For security, users' personal information will be stored in an encrypted form and secure session management will be implemented. The main functions will include adding new tasks, editing existing tasks, marking tasks as complete, and deleting tasks. Additionally, users will have the ability to assign priorities to tasks and use advanced features such as calendar integration. The front-end and back-end technologies to be used will also be determined, and explanations will be provided on the usage of these technologies. The thesis project should focus on functional backend programming and user-friendly interface design.

## Tez Metni

## 1. GİRİŞ

Yapılacaklar Listesi ve Takvim uygulaması web sitesi geliştirmeyi amaçlayan bu tez çalışması, modern yaşamın hızlı tempolu ve karmaşık yapısına uygun olarak bireylerin günlük görevleriyle daha etkili bir şekilde başa çıkmalarını desteklemek amacını taşımaktadır. Bu bağlamda, tez kapsamında, anasayfa, not alma, not silme not güncelleme, notlara öncelik atama takvim sayfası, kullanıcı girişi, kullanıcı ayarları ve profili, yapılacaklar listesi ve bunların tarihe göre seçimi yapılan bölümler içeren bir görev web sitesi geliştirilecektir.

Tezin amacı, kullanıcıların kişisel görev listelerini oluşturabilmeleri ve bu görevleri etkili bir şekilde yönetebilmeleri için kullanıcı dostu bir platform sağlamaktır. Ayrıca, güvenlik açısından, kullanıcıların kişisel bilgileri şifrelenmiş bir şekilde saklanacak ve güvenli bir oturum yönetimi uygulanacaktır, böylece kullanıcıların gizliliği ve veri bütünlüğü korunacaktır.

**1.1 Tez Çalışmasının Amacı**

Bu çalışma, önceki benzer çalışmaların değerlendirilmesini içermekte olup, literatürdeki boşlukları doldurarak ve önceki çalışmalara yeni bir bakış açısı getirerek bu alandaki bilgi birikimine katkıda bulunmayı hedeflemektedir. Ayrıca, yapılacaklar listesi web sitesinin kullanıcı ara yüzü tasarımı ve işlevsel arka plan programlaması üzerine odaklanarak, kullanıcı deneyimini en üst düzeye çıkarmayı amaçlamaktadır.

Bu tez çalışmasının kapsamı, kullanıcıların yeni görev ekleme, mevcut görevleri düzenleme, tamamlama ve silme gibi temel işlemleri gerçekleştirebilecekleri bir platform sunmayı içermektedir. Ayrıca, kullanıcıların görevlere öncelik atama ve takvimle entegrasyon gibi gelişmiş özellikleri kullanmalarına imkan tanıyarak, yapılacaklar listesi web sitesini kullanıcıların ihtiyaçlarına daha fazla uygun hale getirmeyi amaçlamaktadır.

Tez çalışmasının metodolojisi, arayüz ve arka uç tarafında kullanılacak teknolojilerin belirlenmesi ve bu teknolojilerin nasıl kullanılacağına dair detaylı açıklamalar içerecektir. Bu teknolojik altyapı, kullanıcılara güvenilir, hızlı ve etkileşimli bir deneyim sunmak için seçilecek ve detaylı bir şekilde açıklanacaktır.

**1.2 Literatür Taraması**

Görev web siteleri ve benzeri uygulamalar konusundaki literatür, kullanıcıların görevleri organize etme ve takip etme ihtiyacını karşılamak üzere çeşitli platformları içermektedir. Önceki çalışmalardan örnek olarak ‘Notion’, ‘Wunderlist’ ve ‘Todoist’ gibi platformlar, kullanıcıların görevlerini düzenleme, takvimle entegrasyon, öncelik atama gibi temel özelliklere odaklanmış ve başarılı modeller sunmuştur.

Ancak, mevcut literatürde belirli eksiklikler ve yeni ihtiyaçlar da ortaya çıkmaktadır. Örneğin, bazı platformlar kullanıcı dostu arayüz tasarımlarına odaklanırken, güvenlik önlemleri yeterince vurgulanmamıştır. Bu çalışma, kullanıcı deneyimini ve güvenliği bir araya getirerek daha kapsamlı bir çözüm sunmayı hedeflemektedir.

Literatürdeki Boşluklar ve Gereklilikler: Önceki çalışmalarda belirli platformlar üzerine odaklanılmış olsa da kullanıcı dostu arayüz tasarımları ile güvenlik önlemleri arasında denge sağlanmamıştır. Bazı uygulamalar, estetik açıdan kullanıcı dostu olabilirken, güvenlik konusunda eksiklikler barındırmaktadır. Diğer taraftan, güvenliği ön planda tutan uygulamalar ise kullanıcı dostu arayüzlerden ödün vermiş olabilir.

Bu boşluklar önemlidir çünkü bireyler, günlük yaşamlarında etkili bir şekilde görevlerini yönetmeye çalışırken, kullanıcı deneyimi ile güvenlik arasında bir denge kurmaya ihtiyaç duyarlar. Eksik veya dengesiz bir yaklaşım, kullanıcıların platformu etkili bir şekilde kullanmalarını sınırlayabilir ve günlük görev yönetimini zorlaştırabilir.

Bu tez çalışması, literatürdeki bu boşluklara odaklanarak, kullanıcı dostu bir arayüz tasarımı ile güvenli bir altyapıyı birleştirmeyi amaçlamaktadır. Kullanıcıların görev yönetiminde eksik kalan unsurları tamamlamak, literatürdeki bu boşluğu doldurmak ve etkili bir görev web sitesi modeli sunmak bu çalışmanın temel hedefidir.

Araştırmanın Konsept Çerçevesi:Tezin temel yaklaşımları, kullanıcı dostu bir arayüz tasarımı ve güvenlik odaklı bir altyapıyı bir araya getirmek üzerine odaklanmaktadır. Kullanıcıların görevlerini etkili bir şekilde yönetmelerini sağlamak için arayüz tasarımında berraklık, erişilebilirlik ve kullanılabilirlik ön planda tutulacaktır. Aynı zamanda, güvenlik, kullanıcı verilerinin korunması ve güvenli oturum yönetimi gibi önemli unsurlar üzerine odaklanacaktır.

Tezin konsept çerçevesi, görev yönetimindeki en iyi uygulamaları ve güvenlik standartlarını içerecek şekilde tasarlanmıştır. Bu çerçeve, bireylerin günlük yaşamlarını daha etkili bir şekilde planlamalarına yardımcı olacak bir görev web sitesinin temel ilkelerini sağlamaktadır.

Araştırmanın teorik temelleri, görev yönetimi, kullanıcı deneyimi tasarımı, güvenlik algoritmaları ve web tabanlı uygulama geliştirme üzerine odaklanan literatürden türetilmiştir. Bu temeller, görev web sitesi geliştirmenin temel prensiplerini destekleyerek, kullanıcıların günlük görevlerini daha etkili bir şekilde yönetmelerini sağlamayı hedeflemektedir.

**2. MATERYAL VE METHOD**

**2.1 Araştırma Tasarımı**

Araştırmanın temel tasarımı, tasarım ve geliştirme odaklı bir çalışma olarak belirlenmiştir. Bu doğrultuda, kullanıcıların ihtiyaçları ve beklentileri üzerinde odaklanmak amacıyla nitel araştırma yöntemleri tercih edilmiştir. Anketler ve kullanıcı görüşmeleri, geniş bir katılımcı kitlesiyle kullanıcı tercihleri ve beklentileri hakkında genel bir perspektif sağlamak amacıyla tercih edilmiştir.

Yöntem seçimindeki temel faktör, kullanıcıların günlük yaşamlarında etkili bir şekilde görevlerini yönetebilmeleri için ne tür özelliklere ihtiyaç duyduğunu anlamaktır. Bu nedenle, nitel araştırma yöntemleri, kullanıcıların gerçek deneyimlerini ve beklentilerini doğrudan ele almaya yönelik en uygun seçenekler olarak değerlendirilmiştir.

Bu tasarımın seçilme nedenleri, elde edilecek verilerin tezin amacına uygun, güvenilir ve geçerli olmasını sağlamak, geliştirilecek görev web sitesinin kullanıcı odaklı ve ihtiyaçlara uygun olmasını temin etmek ve literatürdeki boşlukları doğru bir şekilde doldurabilmek üzerine odaklanmaktadır.

**2.2 Veri Toplama Yöntemleri**

Kullanıcı görüşmeleri derinlemesine anlayış elde etmek ve kullanıcıların önerilerini daha detaylı bir şekilde alabilmek için gerçekleştirilecektir. Görüşmeler, belirli kullanıcı gruplarıyla gerçekleştirilerek, kullanıcıların platformla ilgili detaylı düşüncelerini ve geri bildirimlerini almayı hedeflemektedir.

Bu yöntemin seçilme nedeni, görev web sitesinin kullanıcı dostu bir arayüz tasarımına ve güvenlik odaklı bir altyapıya sahip olabilmesi için kullanıcı beklenti ve ihtiyaçlarını en doğru şekilde anlamak ve değerlendirmektir. Bu veri toplama yöntemi, kullanıcı odaklı bir geliştirme sürecini destekleyerek, elde edilen verilerin tezin amacına uygun, güvenilir ve anlamlı olmasını sağlayacaktır.

**2.3 Analiz Yöntemleri**

Araştırma verilerinin analiz süreci, görüşme verilerinin derlenmesi ve düzenlenmesi ile başlamıştır. Bu aşama, katılımcıların ifadelerini, duygusal yüklemelerini ve ana temaları anlamak amacıyla detaylı bir analizini içermiştir. Görüşmelerden elde edilen metin verileri titizlikle incelenerek, katılımcıların söylemlerindeki öne çıkan kavramlar ve anahtar ifadeler belirlenmiştir.

Belirlenen anlam birimleri daha sonra ana konu başlıklarına atanmış ve ilişkileri anlamak amacıyla kategorilere ayrılmıştır. Bu aşamada, her bir kategori içindeki anlam birimleri arasındaki benzerlikler ve bağlantılar dikkate alınarak tematik gruplamalar yapılmıştır.

Analiz sürecinin bu aşamasında, katılımcıların ifadelerinin ötesine geçilerek duygusal yüklemeler ve ana temaların derinlemesine bir anlayışı sağlamak hedeflenmiştir. Verilerin bu şekilde analizi, görüşme süreçlerinden elde edilen bilgilerin daha anlamlı ve bütünsel bir şekilde değerlendirilmesini sağlamıştır.

**2.4 Kullanılan Teknolojiler ve Altyapı**

Arayüz Geliştirme:HTML ve CSS, temel yapı ve stil için kullanılmış, sayfaların düzeni, görünümü ve kullanılabilirliği optimize edilmiştir. Tailwind CSS, hızlı ve esnek bir CSS çerçevesi olarak tercih edilmiş, kullanıcı arayüzündeki öğelerin tasarımını daha verimli bir şekilde yönetmeyi sağlamıştır. JavaScript ve ReactJS, kullanıcı arayüzünde dinamik ve etkileşimli öğeler sağlamak amacıyla kullanılmıştır. Bu teknolojiler, sayfalar arasında hızlı geçişler, canlı güncellemeler ve kullanıcı dostu bir etkileşim sunmaktadır.

Arka Uç Geliştirme:C# programlama dili ASP.NET Core 8.0 sürümüyle birlikte Entity Framework, NuGet Paketleri ve gerekli SQL paketleri de hızlı ve ölçeklenebilir bir arka uç geliştirmek amacıyla kullanılmıştır. Bu teknolojiler, veri işleme, yönlendirme ve sunucu tarafı mantık için uygun bir çerçeve sağlamaktadır. MSSQL veri tabanı, kullanıcı verilerini güvenli bir şekilde depolamak ve yönetmek için tercih edilmiştir. İlişkisel verit abanı yönetim sistemi, veri bütünlüğü ve performans açısından avantajlar sunmaktadır.

**3. BACKEND TASARIMI VE DÜZENLEMESİ**

**3.1 Neden Katmanlı Mimari kullanılır**

Katmanlı mimari, büyük ve karmaşık yazılım sistemlerinde yapı ve işlevsellik sağlamak için yaygın olarak kullanılan bir tasarım modelidir. Bu mimari, uygulamanın farklı işlevsel bölümlerini belirli katmanlara ayırarak geliştirir ve yönetir. Sağlam bir arka uç sahip web sitelerinde veya uygulamalarda katmanlı mimarinin kullanılmasının birçok nedeni ve faydası vardır.

1-Modülerlik: Katmanlı mimari, uygulamanın farklı işlevsel bölümlerini ayrı katmanlara ayırarak modülerlik sağlar. Bu sayede her katman belirli bir görevi yerine getirir ve diğer katmanlardan bağımsız olarak geliştirilip test edilebilir. Örneğin, veri erişim katmanı veri tabanı işlemlerini yönetirken, iş mantığı katmanı iş kurallarını ve lojik işlemleri gerçekleştirir.

2-Bakım kolaylığı: Kodun belirli katmanlara ayrılması, bakım işlemlerini kolaylaştırır. Bir katmanda yapılan değişiklikler genellikle diğer katmanları etkilemez. Bu da hata ayıklama, güncelleme ve yeni özelliklerin eklenmesi gibi işlemleri daha hızlı ve güvenilir hale getirir.

3-Sürdürülebilir olması: Katmanlı mimari, uzun vadede yazılımın sürdürülebilirliğini artırır. Uygulamanın büyümesi ve yeni özelliklerin eklenmesi gerektiğinde, mevcut yapıyı bozmadan bu işlemler gerçekleştirilebilir.

4-Performans optimizasyonu: Katmanlar arası iletişim ve veri işlemleri optimize edilerek uygulamanın performansı iyileştirilebilir. Örneğin, veri erişim katmanında yapılan iyileştirmeler doğrudan veri tabanı performansını artırabilir.

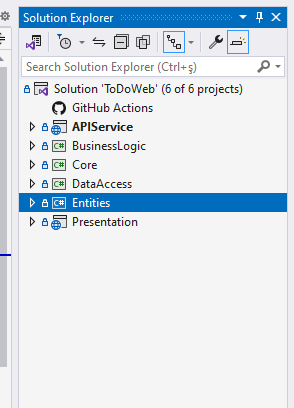
5-SOLID prensiplerine uygunluk: Farklı katmanlar üzerinde çalışan geliştirici ekipler, birbirlerinden bağımsız olarak çalışabilirler. Bu, paralel geliştirme süreçlerini destekler ve projenin genel ilerleme hızını artırır.

6-Güvenlik: Katmanlı yapı, güvenliği artırabilir. Özellikle veri erişim katmanı, veri tabanı işlemlerini güvenli bir şekilde yönetebilir ve sadece iş mantığı katmanının erişimine izin verebilir. Bu, SQL enjeksiyonu gibi saldırılara karşı ek bir güvenlik katmanı sağlar.

Özetle Katmanlı mimari, modülerlik, bakım kolaylığı, yeniden kullanılabilirlik, bağımsız geliştirme, test edilebilirlik, esneklik, güvenlik, dağıtılabilirlik, performans optimizasyonu ve sürdürülebilirlik gibi birçok avantaj sağlar.

**3.2 Katmanlı Mimari Oluşturulması**

Tadaa! Web sitesi tasarlanırken toplam 5 adet katmana ayrılmıştır. BusinessLogic (iş mantığı), DataAccess (veri erişimi), Entities (nesneler), API Service (upa servisi) ve Core (çekirdek). Her bir modül ayrı bir C# projesi olarak çözüme eklenmiştir. Bu katmanların her biri kendine özel görevler ve düzenlemeler içermektedir.



Şekil 3.1 Katmanlı mimari

Business Logic, iş mantığı katmanı, katmanlı mimaride oldukça kritik bir role sahiptir. Bu katman, uygulamanın iş kurallarını, veri işleme mantığını ve karar verme süreçlerini kapsar. İş mantığı katmanının ana görevleri ve sorumlulukları iş kurallarının uygulanması, veri işleme ve dönüştürme, veri erişiminin yönetimi, işlem koordinasyonu ve akış yöntemi, hata yönetimi, servis tabanlı mimari ile entegrasyon ve iş akışını yönetmektir.

Data Access, veri erişimi katmanı uygulamanın veri tabanı veya diğer veri depolama sistemleri ile etkileşim kuran katmanıdır. Bu katman, veri tabanı işlemlerini soyutlayarak iş mantığı katmanının doğrudan veri tabanı ile etkileşime girmesini engeller ve böylece kodun daha temiz, modüler ve sürdürülebilir olmasını sağlar. Bağlantı oluşturma, bağlantıyı açma, kapama ve yönetme gibi işlemleri içerir. Veri tabanı bağlantı havuzu kullanılarak performans ve verimlilik artırılabilir. İş mantığı katmanının ihtiyaç duyduğu veri erişim metotlarını sağlar. Bu metotlar, belirli sorguların yürütülmesi, veri tabanından belirli kriterlere göre veri çekilmesi ve karmaşık sorguların çalıştırılması gibi işlemleri içerir. Örneğin, bir ürün listesini kategoriye göre getirme veya belirli tarih aralığındaki siparişleri çekme işlemleri bu katmanda tanımlanır.

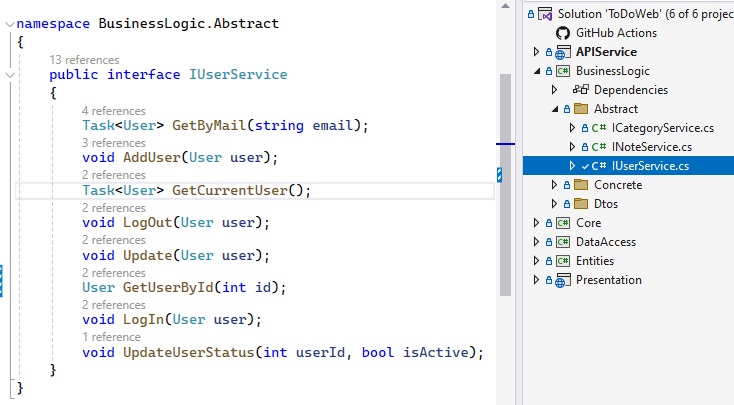
Core, çekirdek katman uygulamanın temel yapı taşlarını ve çekirdek bileşenlerini içeren katmandır. Bu katman, uygulamanın iş mantığı, veri modelleri, arabirimler ve genellikle uygulamanın diğer katmanları tarafından sıkça kullanılan yardımcı sınıfları barındırır. Uygulamanın temel bileşenlerini ve ortak yapılarını içerir, bu sayede diğer katmanların daha esnek, yeniden kullanılabilir ve sürdürülebilir olmasına katkıda bulunur.

Entities, modellerin Katmanı da denebilir. Uygulamanın veri modellerini ve varlık sınıflarını içeren katmandır. Bu katman, veri tabanı tablolarının karşılıklarını ve uygulamanın temel veri yapılarının tanımlarını barındırır. veri erişim katmanı tarafından kullanılan veri yapılarının ve ilişkilerin merkezi bir şekilde tanımlandığı yerdir. Bu sayede, veri tabanı ile uygulama arasındaki veri alışverişi tutarlı ve organize bir şekilde gerçekleştirilir. Varlık sınıflarında, veri doğrulama ve kısıtlamaları tanımlamak için veri dipnotları kullanılabilir. Örneğin, zorunlu alanlar, maksimum uzunluklar ve ilişkiler (bir çoktan-bire) burada belirtilir.

UPA katmanı, uygulamanın dış dünyayla etkileşime geçtiği ve istemci uygulamalara veri sağladığı katmandır. Bu katman, genellikle RESTful veya GraphQL API'ler aracılığıyla çalışır ve iş mantığını dış dünyaya sunar. Bu katman iş mantığı katmanını çağırarak gerekli işlemleri gerçekleştirir. İş mantığı katmanı ile doğrudan etkileşim kurarak iş kurallarının uygulanmasını ve veri işlemlerinin yapılmasını sağlar. UPA'ların doğru bir şekilde kullanılabilmesi için dokümantasyon sağlar. Swagger/OpenAPI gibi araçlar kullanılarak UPA dokümantasyonu otomatik olarak oluşturulur. Ayrıca, sürümleme stratejileri (v1, v2 gibi) ile geriye dönük uyumluluk korunur.

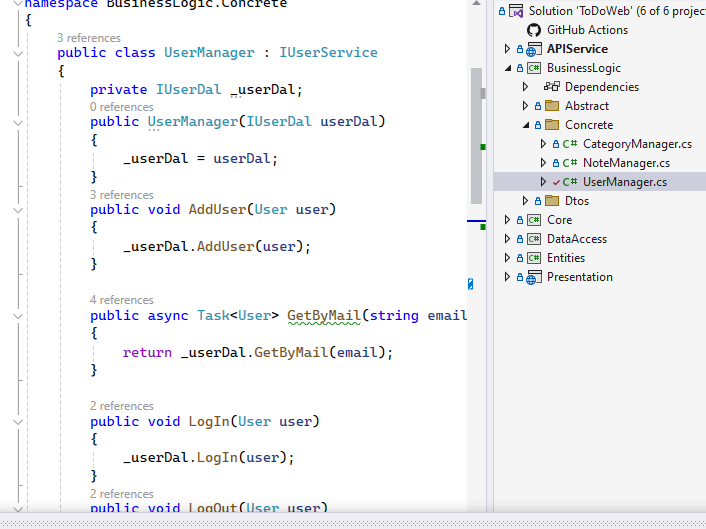
**3.3 Projelerin Kodlanması**

BusinessLogic: Soyut (abstract), Somut (concrete) ve VTO (veri transfer objesi) olmak üzere 3 adet dosyalaması yapılmıştır. Soyut (abstract) klasöründe iş mantığı ile ilgili soyut sınıflar yani ara sınıf kodlanmıştır. Aşağıda kodların görseli mevcuttur:



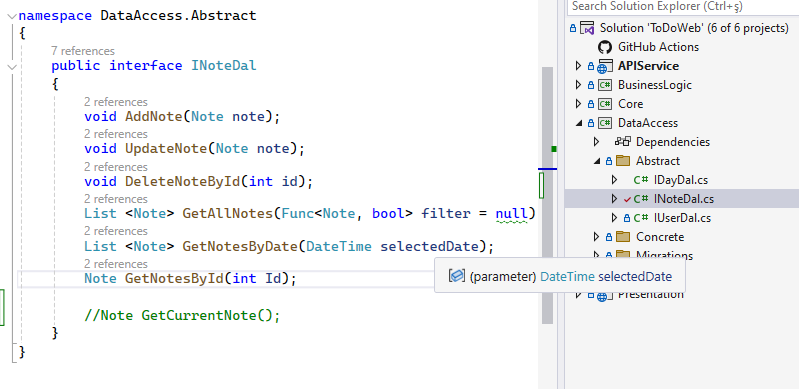
Şekil 3.2 Kullanıcı servis sınıfı

Katmanın soyut sınıfları ise arayüz sınıfların implementasyonu yapılarak oluşturulmuştur. Bu sınıflar kendini yinelemeli fonksiyonlar içerir ve veri erişimi katmanından fonksiyonlara çağrı gönderir. Örnek bir somut sınıf görseli aşağıda mevcuttur:



Şekil 3.3 Kullanıcı yönetici sınıfı

Data Access, soyut (absrtact), Somut (concrete) ve veri tabanı değişikliği (migration) olmak üzere 3 adet dosyalaması yapılmıştır. Soyut klasöründe veri erişimi ve verilerle ilgili değişikliklerin yapıldığı soyut sınıflar yani ara sınıf kodlanmıştır. Aşağıda kodların görseli mevcuttur:

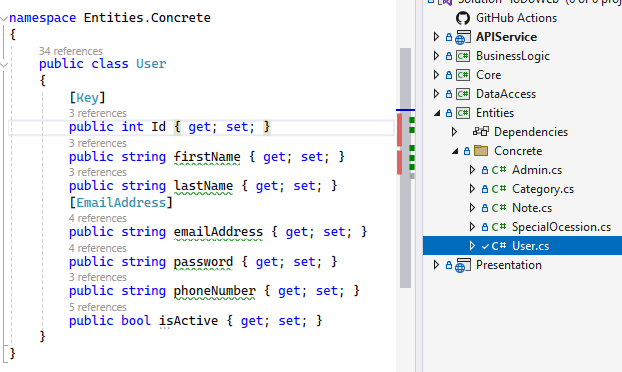
****

Şekil 3.4 ‘INoteDal’ sınıfı

Katmanın soyut dosyasında ise ara sınıf sınıfların implementasyonları yapılarak oluşturulan normal sınıflar kodlanmıştır. Bu sınıfta veri iletişimi, veri tabanı nesneleri ile ilgili işlemler, EntityFramework çerçevesi ile veri tabanı değişiklikleri içeren gerekli fonksiyonlar kodlanmıştır. Aşağıda görseli mevcuttur:

Şekil 3.5 ‘NoteDal’ sınıfı

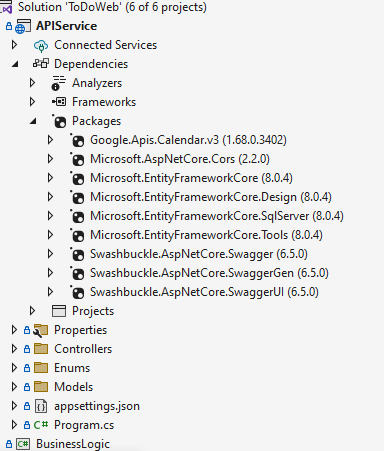
Entities, varlıklar katmanı veri tabanında kullanılacak ve verileri tutacak nesnelerin modellerinin barındığı katmandır. Bu katman aynı zamanda ana ala olarak da geçer. Bu katmanda sadece verit tabanı nesneleri ve projede kullanılacak diğer modeller kodlanır. Aşağıda örnek bir model sınıfı görseli mevcuttur:



Şekil 3.6 Kullanıcı modeli sınıfı

Aynı zamanda bu model sınıflar da ASP.NET Core ve EntityFramework çalışmalarının çok önemli bir noktası olan veri açıklamarı kullanılmıştır. Bu özellik veri formatlama ve veri ilişkilerini belirtmek için kullanılırlar. Veri açıklamaları, veri modellerine eklenen metadata ile model doğrulama ve veri yönetimi işlemlerini kolaylaştırır.

EntityFramework Paketleri: Aşağıda projede kullanılan EntityFramework paketleri verilmiştir. Bu paketlerin kullanım amacı veri tabanı işlemleri ve güvenlik önlemleridir.



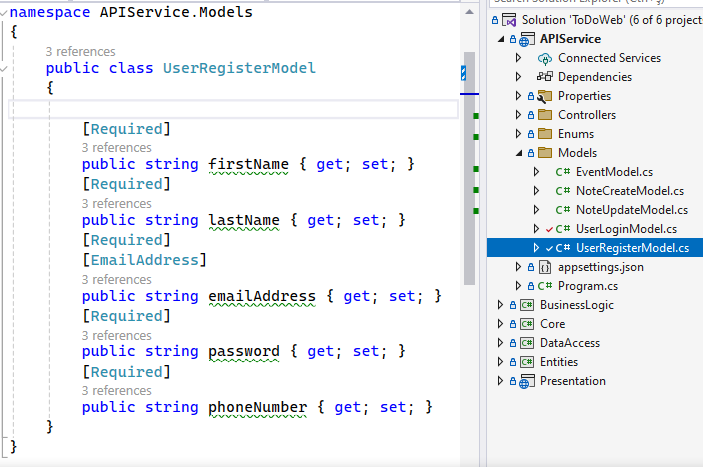
Şekil 3.7 NuGet paketleri

API Service, UPA servisi HTTP isteklerini (GET, POST, PUT, DELETE gibi) karşılar ve bu istekleri ilgili iş mantığına veya servis metoduna yönlendirir. Kontrolörler bu işlemleri yönetir ve her bir isteği uygun iş mantığı metoduna yönlendirir. Servislerdeki ve Data Access katmanındaki sınıflarda bulunan Kullanıcı, Not vb. nesneler için Kontrolörler oluşturulmuştur. Aşağıda görseli mevcuttur:



Şekil 3.8 Not kontrolör sınıfı

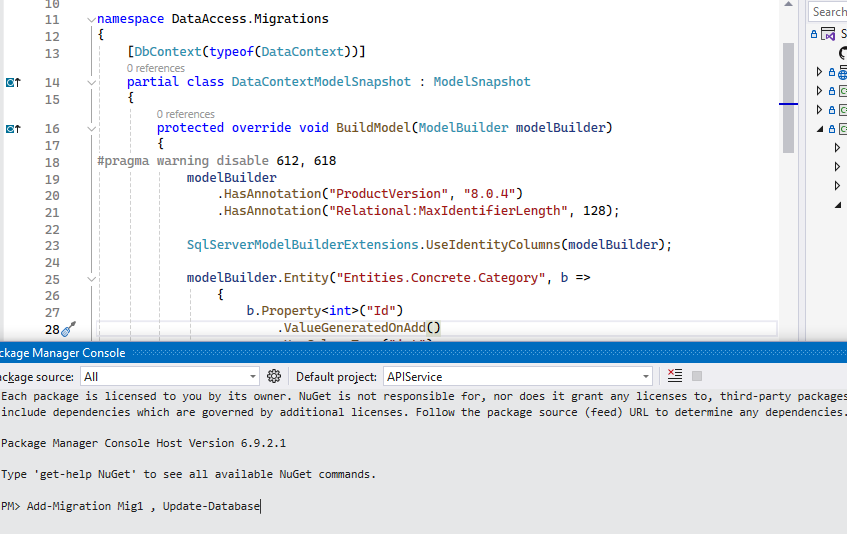
Yukarıda veri tabanında not oluşturan Oluştur (create) fonksiyonu verilmiştir. Bu fonksiyon direk çağırıldığında direk veri tabanında bir not nesnesi oluşturur ve artık bir notumuz olmuş olur. Ayrıca bu katmanda bazı VTO (veri transfer objesi) modelleri yazılmıştır. Bu model sınıfların yazılma sebebi modülerliği ve veri güvenliğini korumaktır. Örneğin bir ‘UserRegisterModel’ sınıfı yazılmıştır çünkü kullanıcı kaydolurken sadece belirli bilgiler istenmektedir. Buna uygun modeli yazmak veri kaybını engellemek için daha uygundur. Aşağıda görseli mevcuttur:



Şekil 3.9 Veri Transferi Nesnesi sınıfı

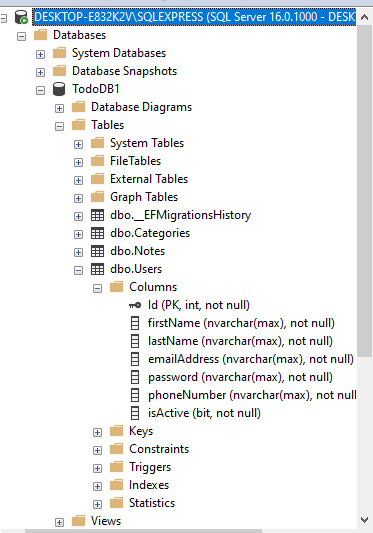
**3.4 Veri Tabanı Oluşturma ve Düzenleme**

Migration işlemleri arka uç programlama için özellikle NTP(nesne tabanlı programlama)) tabanlı işlemler ve uygulamalar yaparken günümüzde olmazsa olmaz bir işlemdir denebilir. Migration işlemi, veri tabanı şemasının yönetilmesi ve versiyonlanması sürecidir. Birkaç satır komut yazarak projedeki modellere göre veri tabanı nesnelerinizi ve diyagramınızı oluşturur. Yazılım geliştirme sürecinde, veri tabanı şemasında yapılan değişikliklerin (örneğin, yeni tablolar eklemek, mevcut tabloları değiştirmek, kolon eklemek veya silmek gibi) tutarlı ve takip edilebilir bir şekilde uygulanmasını sağlar. Aşağıda nasıl Migration yapılacağı ile görsel mevcuttur. Komut satırı için NuGet paket yönetici konsolu kullanılmaktadır.



Şekil 3.10 Migration işlemi komutu

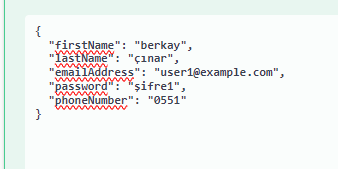
İşlem bize hem zamandan hem maaliyetten çok fazla fayda sağlamaktadır. Bu işlemden sonra veri tabanı aşağıda gösterildiği gibi oluşmuştur:



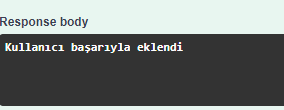
Şekil 3.11 MSSQL veri tabanı tabloları

**3.5 API Servisinin Swagger ile Test Edilmesi**

Bu bölümde web sitesinin UPA’sından birkaç örnek işlev test edilmiştir ve bunların görseli aşağıya konulmuştur.

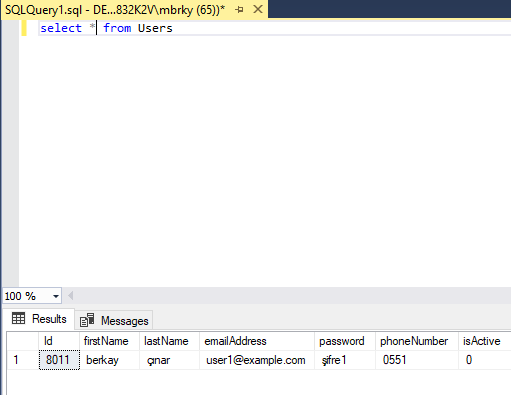


Şekil 3.12 Kullanıcı oluşturma fonksiyonu test



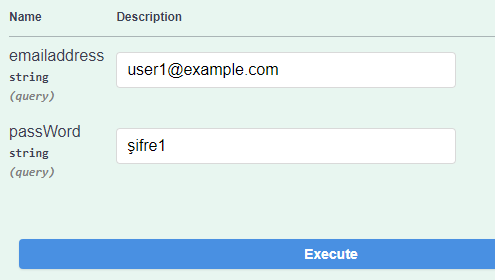
Şekil 3.13 Fonksiyon cevabı

Veri tabanında da gördüğümüz gibi kullanıcı başarılı bir şekilde kaydedilmiştir. Gerekli doğrulamalar ve adımlar tek tek test edilmiştir.

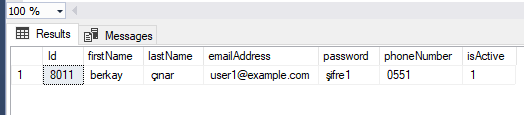


Şekil 3.14 Veri tabanı kontrolü

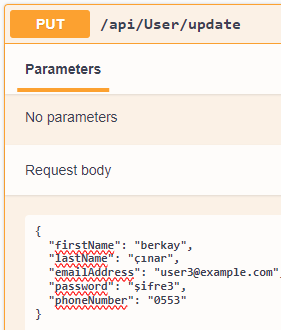
Kullanıcı girişi işlemi ‘isActive’(aktif mi) özelliği üzerinden yapılmaktadır. Özellik başlangıç halinde 0’dır. Kullanıcı giriş bilgilerini doğru girdiği takdirde bilgileri güncellenir ve ‘isActive’(aktif mi) özelliği 1 olur. Aşağıda testi mevcuttur.



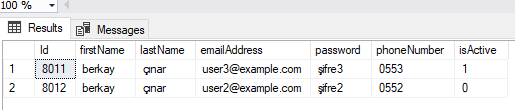
Şekil 3.15 Giriş işlemi Swagger

  
Şekil 3.16 Veri tabanı kontrol

Kullanıcı güncelleme işlemi:

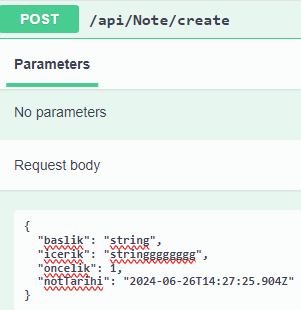


Şekil 3.17 Kullanıcı bilgi güncelleme

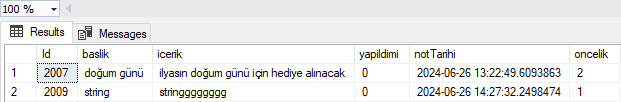


Şekil 3.18 Veri tabanı kontrol

Kullanıcı hesabına girdikten sonra yapılacaklar listesine not ekleyebilmektedir.



Şekil 3.19 Swagger not ekleme

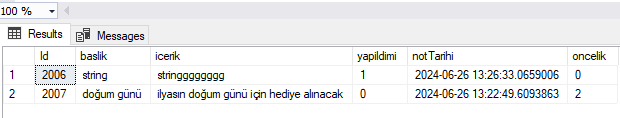


Şekil 3.20 Veri tabanı kontrol

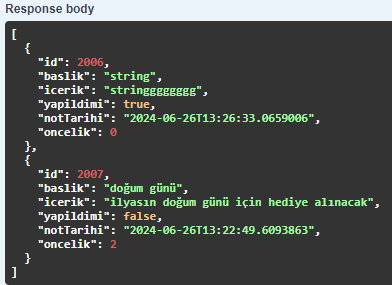
Görüldüğü üzere not başarıyla eklenmiştir.

Aynı şekilde not silme, id özelliği ile not döndürme, notun önceliğini artırma, not bilgilerini güncelleme ve notları listeleme gibi işlemler de UPA’da yapılabilmektedir.

Profil kısmından kullanıcı notlarına ulaşılabilmektedir.



Şekil 3.21 Veri tabanı



Şekil 3.22 Swagger cevabı

Görüldüğü gibi veri tabanında kayıtlı notlar kullanıcı tarafından görüntülenebilmektedir.

**4. FRONTEND OLUŞTURULMASI VE DÜZENLENMESİ**

**4.1 Ana Sayfaların Oluşturulması**

Anasayfa.html:Bu sayfa, kullanıcıya site hakkında ilk izlenimi verir ve genel site navigasyonunu sağlar.

Sayfanın en üst kısmında başlık bulunur. Başlık içerisinde site logosu, başlık ve ana navigasyon menüsü bulunur. Kullanıcıların siteyi gezinmelerini sağlar.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.1 Header kodu

Sayfanın merkezinde main bulunur. Burada hoş geldiniz mesajı, site hakkında genel bilgiler ve motivasyon bölümü yer alır.

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturulduŞekil 4.2 ‘Main’

Sayfanın en altında ise alt bilgi bölümü içerisinde telif hakkı bilgisi bulunur.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

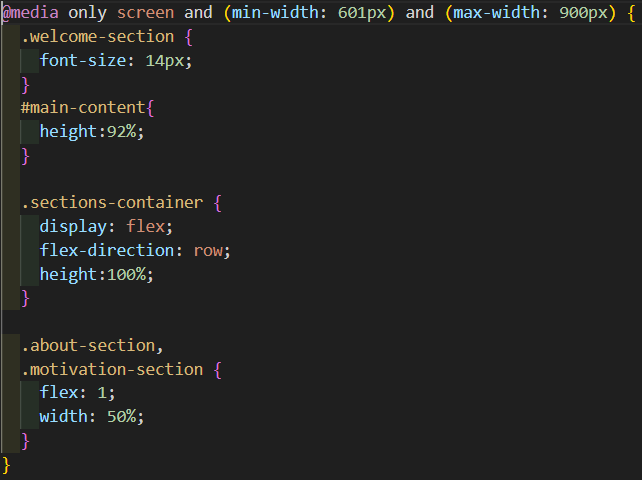
Şekil 4.3 Alt bilgi

Style.css: Arka plan rengi, yazı tipi seçimi ve buton stillemeleri gibi görsel öğelerle kullanıcı deneyimini zenginleştirir. Ayrıca, mobil uyumluluk için medya sorguları ile farklı ekran boyutlarına uyum sağlar.  
  
**metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Şekil 4.4   
Navbar içindeki öğeler ve içeriğin düzenlenmesi için Flexbox kullanılmıştır. Örneğin, .navbar içindeki öğeler justify-content: space-between ve align-items: center ile hizalanmıştır.  
Farklı ekran boyutları için medya sorguları kullanılarak (@media queries) sayfanın responsive (duyarlı) olması sağlanmıştır. Bu medya sorguları, mobil, tablet, dizüstü bilgisayar ve büyük monitör gibi farklı cihazlara uyumlu görünümler sağlar.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.5 Medya sorguları

Script.js: Bu dosya, kullanıcı etkileşimlerini yönetir, sayfa animasyonlarını kontrol eder ve motivasyon alıntılarını dinamik olarak değiştirir.

Bu bölümde sayfa yüklendiğinde DOMContentLoaded eventi çalışmaya başlar. startButton adlı buton seçilir ve tıkla olay dinleyicisi eklenir. Kullanıcı butona tıkladığında, kullanıcı login sayfasına yönlendirilir ve SweetAlert2 kütüphanesi kullanılarak bilgilendirici bir mesaj gösterilir.

metin, ekran görüntüsü, ekran, görüntüleme, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.6 Olay dinleyicisi

Bu kod bloğu, sayfa yüklendiğinde giriş yap ve kayıt ol butonlarına tıklama olay dinleyicileri ekler. Giriş yap butonuna tıklanıldığında kullanıcı login sayfasına, kayıt ol butonuna tıklanıldığında ise kayıt sayfasına yönlendirilir. Bu işlemler sırasında kullanıcıya bilgilendirici mesajlar gösterilir.

metin, ekran görüntüsü, ekran, görüntüleme, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.7 Olay dinleyicisi

Bu bölümde, sayfa yüklendiğinde GSAP (GreenSock Animation Platform) kullanılarak çeşitli animasyonlar uygulanır. Navigasyon çubuğu yukarıdan aşağıya doğru “bounce” efekti ile gelir, hoş geldiniz bölümü, hakkında bölümü ve motivasyon bölümü sırasıyla opaklığı artarak görünür hale gelir.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.8

Bu bölümde, bir dizi motivasyon alıntısı ve yazarları tanımlanır. updateQuote fonksiyonu, belirli aralıklarla (3 saniyede bir) alıntıları günceller. GSAP kullanılarak alıntı metni ve yazarının opaklığı sıfıra düşürülüp yeni alıntı ve yazar bilgisi ile değiştirilir ve tekrar opaklığı artırılarak görünür hale getirilir. Bu süreç, alıntıların dinamik ve görsel olarak çekici bir şekilde değişmesini sağlar.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.9 Motivasyon JavaScript kodu

**metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Şekil 4.10 Anasayfa

**4.2 Giriş Sayfalarının Oluşturulması**

Loginpage.html: Bu sayfa kullanıcıların e-posta adresleri ve şifreleri ile giriş yapmalarını sağlamak amacıyla tasarlanmıştır. Ayrıca, kayıt olmak isteyen kullanıcılar için bir yönlendirme bağlantısı içerir. Sayfa, modern web teknolojileri kullanılarak oluşturulmuş olup, kullanıcı deneyimini artırmak için çeşitli kütüphaneler ve doğrulama mekanizmaları içermektedir.

Tailwind CSS ve AOS kütüphaneleri stil dosyası olarak dahil edilmiştir. Genel stil ayarları da style etiketi içerisinde tanımlanmıştır.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.11 Giriş sayfası CSS kodu

AOS ve GSAP kütüphaneleri için senaryo dosyaları eklenmiştir. Ayrıca login.js dosyası dahil edilmiştir.

Body’de kullanıcıya hitap eden başlık ve alt başlıklar, giriş formu ve kayıt ol butonu tanımlanmıştır.

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.12 HTML kodu

Login.js: loginpage.html dosyasıyla entegre çalışarak kullanıcı tarafından girilen e-posta ve şifrenin doğrulanması, sunucuya gönderilmesi ve yanıtın işlenmesi gibi işlemleri gerçekleştirir.

Öncelikle sayfa yüklendiğinde AOS animasyonlarının başlatılmasını sağlayan AOS.init() komutu içerir.

E-posta doğrulama fonksiyonu kullanıcının girdiği e-posta adresinin belirli bir formata uygun olup olmadığını kontrol eder. Sadece gmail.com, hotmail.com ve outlook.com alan adlarını kabul eder.

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.13 isValidEmail fonksiyonu

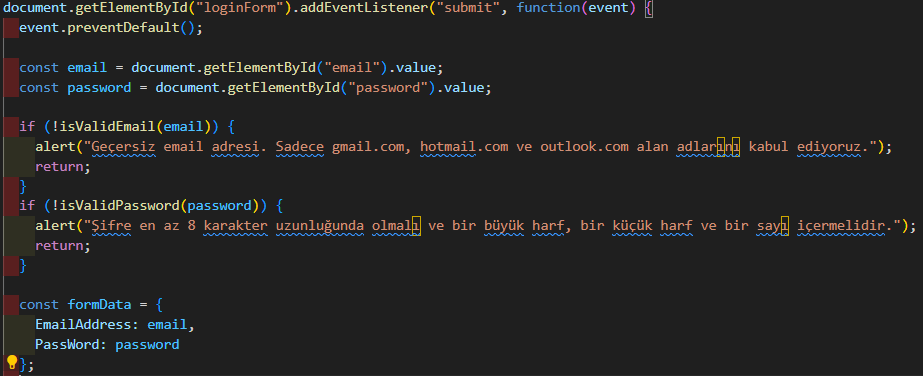
Şifre doğrulama fonksiyonu, kullanıcının girdiği şifrenin en az 8 karakter uzunluğunda olmasını ve bir büyük harf, bir küçük harf ve bir sayı içermesini sağlar.

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.14 Fonksiyon

Form abone olayı dinleyicisi, form gönderildiğinde çalışır. Kullanıcının girdiği e-posta ve şifre bilgileri doğrulanır. Geçerli bilgiler, ‘fetch UPA’ kullanılarak sunucuya gönderilir ve yanıt işlenir. Giriş başarılı olduğunda kullanıcı notlar sayfasına yönlendirilir, hata durumlarında ise uygun mesajlar gösterilir.



Şekil 4.15 GetElementById

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.16 UPA bağlama

metin, ekran görüntüsü, tasarım içeren bir resim

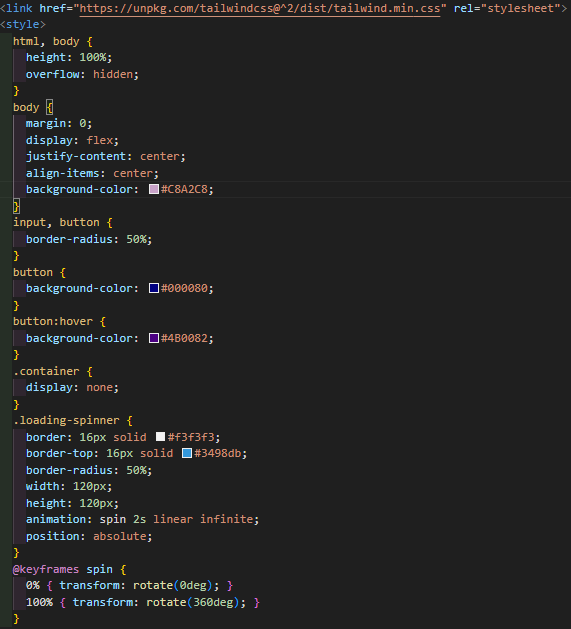
Açıklama otomatik olarak oluşturuldu

Şekil 4.17 Giriş ekranı

**4.3 Kayıt Sayfasının Oluşturulması**

Registerpage.html: Kullanıcıların ad, soy ad, e-posta adresi, şifre ve telefon numarası gibi bilgilerini girerek hesap oluşturmasını sağlar. Ayrıca, kullanıcı kaydının başarılı veya başarısız olması durumunda ilgili bilgilendirme mesajlarını gösterir.

Bu bölüm, gerekli stil ve kütüphanelerin sayfaya dahil edilmesini sağlar. Tailwind CSS kütüphanesi dahil edilmiştir. Ayrıca, sayfanın arka plan rengi, buton renkleri ve genel düzeni için özel stil kuralları tanımlanmıştır. Sayfa yüklendiğinde gösterilen bir yükleme animasyonu (loading-spinner) ve kullanıcı formunun başlangıçta gizli olması için stil kuralları da eklenmiştir.



Şekil 4.18 Kayıt ekranı CSS kodu

Kullanıcıya hitap eden başlık ve form alanları(ad, soy ad, e-posta, şifre, telefon numarası) body alanında tanımlanmıştır.

Kullanıcı formu gönderdiğinde, e-posta, şifre ve telefon numarası doğrulama işlemleri yapılır. Geçerli bilgiler sunucuya gönderilir ve yanıt işlenir. Kayıt başarılı olduğunda kullanıcı giriş sayfasına yönlendirilir, hata durumlarında ise uygun mesajlar gösterilir.

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.19 Olay dinleyicisi

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.20 UPA istek atma

**ekran görüntüsü, metin, çizgi, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Şekil 4.21 Kayıt ekranı

**4.4 Profil Sayfalarının Oluşturulması**

Profile.html:Bu bölüm, gerekli stil ve kütüphanelerin sayfaya dahil edilmesini sağlar. Tailwind CSS kütüphanesi dahil edilmiştir. Ayrıca, sayfanın arka plan rengi ve genel düzeni için özel stil kuralları tanımlanmıştır.

metin, ekran görüntüsü, ekran, görüntüleme, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu  
Şekil 4.22 Tailwind import kodu

Kullanıcıya hitap eden başlık ve form alanları body kısmında tanımlanmıştır. Ayrıca profile.js dosyası sayfaya dahil edilmiştir.  
metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.23 Profil güncelleme sayfası kodu

Profile.js: profile.html dosyasıyla entegre çalışarak kullanıcı profil bilgilerini günceller. Bu dosya, kullanıcı tarafından girilen bilgilerin doğrulanması, sunucuya gönderilmesi ve yanıtın işlenmesi gibi işlemleri gerçekleştirir.

Bu satır, sayfanın tamamı yüklendiğinde çalışacak olan kodları içerir. Bu olay dinleyicisi, sayfa yükleme işlemi tamamlandığında belirtilen işlevi yürütür.



Şekil 4.24

isValidEmail(email), e-posta adresinin gmail.com, outlook.com veya hotmail.com alan adlarına sahip olup olmadığını kontrol eder. isValidPassword(password), şifrenin en az 8 karakter uzunluğunda olup olmadığını ve büyük harf, küçük harf ve rakam içerip içermediğini kontrol eder. isValidPhoneNumber(phoneNumber), telefon numarasının 10 haneli olup olmadığını kontrol eder.

metin, ekran görüntüsü, yazı tipi, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.25 Fonksiyonlar

Form sunma olayıyla profileForm formunun sunma olayını dinler. Form gönderildiğinde, varsayılan form gönderme işlemini durdurur (event.preventDefault()). Form alanlarından girilen bilgiler alınır (ad, soy ad, e-posta adresi, şifre, telefon numarası). Girilen bilgilerin geçerli olup olmadığını kontrol edilir. Geçersiz bilgiler varsa kullanıcıya bir uyarı mesajı gösterir ve işlemi durdurur. Geçerli bilgiler bir kullanıcı nesnesine dönüştürülür. fetch API kullanılarak bu bilgiler sunucuya gönderilir. PUT yöntemi kullanılarak güncelleme işlemi gerçekleştirilir. Sunucudan gelen yanıt işlenir. Yanıt başarılıysa (response.ok), başarı mesajı gösterilir. Hata durumunda uygun hata mesajları gösterilir.

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.26 Profil JavaScript sınıfı kodu

metin, ekran görüntüsü, yazılım, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.27 UPA istek atma

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.28 Profil güncelleme sayfası

**4.5 Takvim Sayfalarının Oluşturulması**

Index.html: Index.html dosyası, kullanıcıların tarihleri görüntüleyip not ekleyebileceği bir takvim uygulamasının ana sayfasını içerir. Başlık ve stil bilgileriyle donatılmış bu sayfa, kullanıcı dostu bir arayüz sunar. Takvimde aylar ve yıllar arasında geçiş yapılabilir, belirli tarihlere not eklenebilir ve not ekleme kutusu kullanılarak başlık, açıklama ve öncelik girilebilir. Kullanıcılar bu sayfa üzerinden etkinliklerini ve notlarını kolayca yönetebilir.

Aynı zamanda script.js de sayfaya dahil edilmiştir.

metin, ekran görüntüsü, ekran, görüntüleme, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.29 Konteynır HTML kodu

Style.css:Takvim ve tarih/saat bileşenlerinin çeşitli animasyonları örneğin; to-top, to-left, to-right, showtime ve iddetime animasyonları farklı hareket efektleri sağlar.

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldumetin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.30 ‘keyframes’ CSS kodu

Script.js:Script.js dosyası, takvim uygulamasının işlevselliğini sağlar. Artık yıl kontrolü, takvim oluşturma, tarih ve zaman gösterimi, not ekleme ve not kutusunu gösterme/gizleme gibi işlevleri içerir. Bu dosya, kullanıcıların tarihler arasında gezinebilmesini ve not ekleyebilmesini sağlar.

**metin, ekran görüntüsü, yazı tipi, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Şekil 4.31

Ay seçiciye tıklanınca ay listesini gösterir ve diğer formatları gizler.

**metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Şekil 4.32

Verilen ay ve yıl için takvimi oluşturur. Günleri hesaplar ve takvim günlerine ekler. Geçerli günü vurgular ve tıklanabilir yapar.

**metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Şekil 4.33 Takvim JavaScript kodu

Geçerli tarihi ve zamanı formatlar ve gösterir. Zaman her saniye güncellenir.

**metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Şekil 4.34

Not ekleme formu gönderildiğinde verileri toplar, doğrular ve sunucuya gönderir. Başarılı olursa not verilerini yedek depolamaya kaydeder ve not sayfasına yönlendirir.

**ekran görüntüsü, metin, sayı, numara, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Şekil 4.35 Not ekleme sayfası

**4.6 Not Sayfalarının Oluşturulması**

Note.html:Kullanıcılara not ekleme, düzenleme ve silme gibi işlemleri gerçekleştirebilecekleri kullanıcı dostu bir arayüz sunar.

**Popup-box ve popup,** not ekleme kutusunu içerir. Kullanıcının yeni not ekleyebilmesi için bir form sağlar. Sınıfı wrapper olan elementin altındaki add-box, yeni not ekleme butonunu içerir. Ayrıca bu kodda çeşitli senaryo bağlantıları yapılmıştır:

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.36 Not sayfası HTML kodu

Notescript.js:Bu dosya, kullanıcıların not yönetimini kolaylaştırmak için dinamik etkileşimleri ve API işlemlerini içerir. Notların sunucuya kaydedilmesi, güncellenmesi ve silinmesi işlemlerini içerir. Ayrıca, notların yerel depolamada saklanmasını ve sayfa yüklendiğinde bu notların gösterilmesini sağlar. Bu işlevsellikler, kullanıcıların notlarını kolayca yönetmelerine olanak tanır.

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.37 Not sayfası JavaScript kodu

addBox butonuna tıklandığında, popup kutusu açılır ve kullanıcı yeni bir not ekleyebilir. closeIcon butonuna tıklandığında, popup kutusu kapanır ve input alanları temizlenir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.38 Olay dinleyicisi

Yerel depolamadan notları alır ve sayfada gösterir. Her not için HTML oluşturur ve sayfaya ekler.

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.39

Menü simgesine tıklandığında menüyü gösterir ve dışına tıklandığında menüyü gizler.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.40 ShowMenü

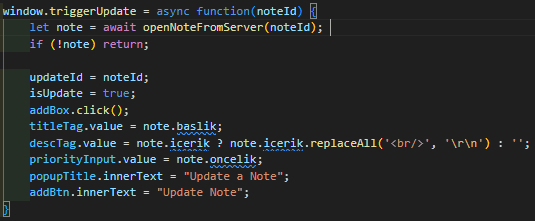
Kullanıcı bir notu silmek istediğinde çalışır. Önce kullanıcıdan onay alır, sonra notu yerel depolama ve sunucudan siler.

metin, ekran görüntüsü, ekran, görüntüleme, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.41 Not silme fonksiyonu

Bir notu güncellemek için kullanılır. Notun bilgilerini girdi alanlarına yükler ve popup kutusunu açar.



Şekil 4.42 Fonksiyon tetikleyicisi

Kullanıcı yeni bir not eklediğinde veya var olan bir notu güncellediğinde çalışır. Notu sunucuya gönderir ve işlemin sonucuna göre kullanıcıya bildirim gösterir.

metin, ekran görüntüsü, ekran, görüntüleme, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.43 Olay dinleyicisi

Style.css:Bu dosya, sayfanın genel düzenini, not ekleme kutusunu, pop-up kutularını ve notların detaylarını içeren stil kurallarını içerir. Ayrıca, çeşitli kullanıcı etkileşimlerine yönelik görsel geri bildirimleri de düzenler.

Getnote.js:Notları sunucudan almak ve yerel depolamayı güncellemek için kullanılan fonksiyonları içerir. Bu dosya, belirli bir notu veya tüm notları sunucudan getirir, getirilen notları yerel depolamaya kaydeder ve notların görüntülenmesini sağlar. Ayrıca, hata durumlarında uygun mesajlar vererek kullanıcıya bilgi sağlar.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.44 UPA istek atma

metin, ekran görüntüsü, yazılım, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.45 UPA istek atma

Bu fonksiyon, alınan notları yerel depolamaya kaydeder ve ‘showNotes’ fonksiyonunu çağırarak notları görüntüler.

Notları yerel depolamaya JSON formatında kaydeder. ‘showNotes’ fonksiyonunu çağırarak notların güncellenmesini ve görüntülenmesini sağlar.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.46

Bu fonksiyon, notları sunucudan alır ve yerel depolamayı günceller.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.47

Notecreate.js:Yeni bir not oluşturmak için kullanılan fonksiyonları içerir. Bu dosya, kullanıcıdan alınan not verilerini sunucuya gönderir ve başarılı bir şekilde oluşturulan notu yerel depolamaya kaydeder. Ayrıca, hata durumlarında uygun mesajlar vererek kullanıcıya bilgi sağlar. Bu fonksiyon, sunucuya yeni bir not oluşturma isteği gönderir.

Kullanıcıdan alınan not verilerini (title, description, priority) kullanarak noteData adında bir nesne oluşturur. Bu nesneye notun başlığı, içeriği, yapılma durumu (false), oluşturulma tarihi (şu anki tarih ve saat) ve öncelik derecesi eklenir. Sunucuya bir POST isteği gönderir ve not oluşturma isteğinde bulunur. İstek başarılı olursa, yanıtı JSON formatında çözümleyerek notun ID’sini alır ve noteData nesnesine ekler. NoteData nesnesini yerel depolamaya kaydeder (addNoteToLocal fonksiyonunu çağırarak).

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.48 Not oluşturma fonksiyonu

Bu fonksiyon, yeni oluşturulan notu yerel depolamaya kaydeder ve notları görüntüler. noteData, yerel depolamaya kaydedilecek not verileridir.

Yerel depolamada mevcut olan notları alır ve notlar dizisine ekler. Yeni notu (noteData) notlar dizisine ekler. Güncellenmiş notlar dizisini JSON formatında yerel depolamaya kaydeder. showNotes fonksiyonunu çağırarak notların güncellenmesini ve görüntülenmesini sağlar.

Popup kutusunu kapatır ve sayfanın kaydırılabilir olmasını sağlar.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.49

Notedelete.js:Belirli bir notun sunucudan ve yerel depolamadan silinmesi için gerekli fonksiyonları içerir. Kullanıcı bir notu sildiğinde, bu işlemi hem sunucu hem de yerel depolama seviyesinde gerçekleştirir. Bu fonksiyon, belirli bir notun sunucudan silinmesi için DELETE isteği gönderir.

Sunucuya notu silme isteği (DELETE isteği) gönderir. Sunucudan yanıt bekler ve yanıt başarılı olursa, notun sunucudan başarıyla silindiğini belirten mesajı konsola yazar.

Yanıt JSON formatında değilse, yanıt metnini işleyerek konsola yazdırır. Not sunucudan başarıyla silindiğinde, notu yerel depolamadan silmek için deleteNoteFromLocal fonksiyonunu çağırır. Hata durumunda, hata mesajlarını konsola yazar.

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.50

Bu fonksiyon, belirli bir notu yerel depolamadan siler. Yerel depolamada mevcut olan notları alır ve notlar dizisine ekler. Belirtilen noteId ile eşleşmeyen notları filtreleyerek updatedNotes dizisini oluşturur. Güncellenmiş notlar dizisini JSON formatında yerel depolamaya kaydeder. Silme işleminin başarıyla tamamlandığını konsola yazar. Notların güncellenmiş halini kullanıcıya göstermek için showNotes fonksiyonunu çağırır.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.51

Noteupdate.js:Kullanıcıların mevcut notlarını güncelleme işlemlerini yönetir. Bu dosya, notun hem sunucuda hem de yerel depolamada güncellenmesi için gerekli fonksiyonları içerir. Kullanıcı bir notu güncellediğinde, bu işlemler sunucuya gönderilir ve aynı zamanda yerel depolama da güncellenir.

Bu fonksiyon, belirli bir notun sunucuda güncellenmesi için PUT isteği gönderir.

Güncellenmiş not verilerini içeren bir noteData nesnesi oluşturur. Sunucuya notu güncelleme isteği (PUT isteği) gönderir. Sunucudan yanıt bekler ve yanıt başarılı olursa, notun başarıyla güncellendiğini belirten mesajı konsola yazar. Yanıt JSON formatında değilse, yanıt metnini işleyerek konsola yazdırır. Not sunucuda başarıyla güncellendiğinde, notu yerel depolamadan güncellemek için updateNoteLocal fonksiyonunu çağırır. Hata durumunda, hata mesajlarını konsola yazar.

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.52

Bu fonksiyon, belirli bir notu yerel depolamada günceller, notData güncellenmiş not verilerini içeren nesnedir:

Yerel depolamada mevcut olan notları alır ve notlar dizisine ekler. Belirtilen noteData.id ile eşleşen notu bulur ve günceller. Eğer not bulunamazsa, yeni not olarak ekler. Güncellenmiş notlar dizisini JSON formatında yerel depolamaya kaydeder. Güncelleme işleminin başarıyla tamamlandığını konsola yazar. Notların güncellenmiş halini kullanıcıya göstermek için showNotes fonksiyonunu çağırır. Güncelleme çerezleri kapatır ve sayfa kaydırmasını tekrar aktif hale getirir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.53

Opennote.js:Belirli bir notun sunucudan alınmasını ve kullanıcının bu notun detaylarını bir modal pencere içinde görmesini sağlar. Bu işlem sırasında, GSAP kullanılarak açma ve kapama animasyonları uygulanır.

Bu fonksiyon, sunucudan belirli bir notu almak için GET isteği gönderir.

Sunucuya notu açma isteği (GET isteği) gönderir. Yanıt başarılı olursa, notun detaylarını JSON formatında alır ve konsola yazar. Yanıt başarısız olursa, hata mesajlarını ve detaylarını konsola yazar. Hata durumunda, hata mesajlarını konsola yazar ve sıfır döner.

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.54

Bu fonksiyon, belirli bir notu açmak ve detaylarını göstermek için kullanılır.

openNoteFromServer fonksiyonunu çağırarak notun detaylarını sunucudan alır. Not başarıyla alındıysa, activeNoteId değerini günceller ve notun detaylarını göstermek için showNoteDetails fonksiyonunu çağırır.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.55

Bu fonksiyon, notun detaylarını bir modal pencerede göstermek için kullanılır. Not, gösterilecek notun detaylarını içeren nesnedir.

Bir div öğesi oluşturur ve detail-popup ve show(göster) sınıflarını ekler. Notun başlığı ve içeriği ile modal içeriğini oluşturur.Bu içeriği detailPopup öğesine ekler ve document.body'ye yerleştirir. GSAP animasyonu kullanarak modal'ı açar.

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.56

Bu fonksiyon, not detaylarını gösteren modal pencereyi kapatır. detail-popup öğesi seçilir.GSAP animasyonu kullanarak modal'ı kapatılır ve animasyon tamamlandığında öğe kaldırılır.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şekil 4.57

**5. BULGULAR VE TARTIŞMA**

**5.1 Performans ve Ölçeklenebilirlik**

Bulgular: Uygulamanın performans testlerinde, ASP.NET Core 8.0'in sağladığı verimli çalışma yapısı ve Entity Framework'ün etkin veri yönetimi özellikleri sayesinde yüksek performans elde edilmiştir. Arka uç ve arayüz arasındaki iletişimin hızlı ve kesintisiz olması, uygulamanın genel hızını ve yanıt verme süresini olumlu yönde etkilemiştir.

Tartışma: Bu bulgu, ASP.NET Core 8.0'in modern web uygulamaları geliştirme sürecinde güçlü bir platform olduğunu göstermektedir. Ölçeklenebilirlik açısından, uygulamanın daha büyük veri setleri ve artan kullanıcı yükleri altında nasıl performans göstereceği, daha kapsamlı testlerle doğrulanmalıdır. Gelecekteki çalışmalar, özellikle yüksek trafikli ortamlarda uygulamanın kararlılığını ve performansını optimize etmeye odaklanmalıdır.

**5.2 Güvenlik ve Veri Koruma**

Bulgular: Uygulamanın güvenlik testleri, temel güvenlik önlemlerinin etkin bir şekilde uygulandığını göstermektedir. Kullanıcı verilerinin korunması ve veri güvenliği sağlanmıştır. Ancak, ileri düzey saldırılara karşı daha kapsamlı güvenlik önlemleri gerekmektedir.

Tartışma: Güvenlik, web uygulamaları için kritik bir faktördür. Bu bulgular, temel güvenlik önlemlerinin yeterli olduğunu gösterse de, güvenlik tehditlerine karşı sürekli güncellemeler ve iyileştirmeler yapılması gerekmektedir. Özellikle kullanıcı verilerinin gizliliğini korumak için daha gelişmiş şifreleme teknikleri ve güvenlik protokollerinin entegrasyonu gelecekteki çalışmaların odak noktası olmalıdır.

Sonuç olarak, bu tez çalışması boyunca elde edilen bulgular, ASP.NET Core 8.0 ve React/JavaScript kullanılarak geliştirilen web uygulamasının performans, kullanıcı deneyimi ve güvenlik açısından başarılı olduğunu göstermektedir. Ancak, gelecekte yapılacak çalışmalar ve iyileştirmeler, uygulamanın daha da geliştirilmesine ve genişletilmesine olanak tanıyacaktır.

# 

# 6. SONUÇLAR VE ÖNERİLER

##### Bu tez çalışması, modern web teknolojilerini kullanarak kullanıcı dostu ve işlevsel bir yapılacaklar listesi ve takvim uygulaması geliştirme sürecini başarılı bir şekilde tamamlamıştır. ASP.NET Core 8.0'nin sağlam altyapısı ve EntityFramework'ün esnek veri yönetimi yetenekleri, uygulamanın güvenilir ve ölçeklenebilir olmasını sağlamıştır. JavaScript ve React kullanılarak geliştirilen arayüz, kullanıcı deneyimini optimize ederken, aynı zamanda uygulamanın hızlı ve dinamik olmasını sağlamıştır.

Tez çalışması boyunca kullanılan teknolojiler ve yöntemler, projenin gereksinimlerini karşılamada yeterli olmuştur. ASP.NET Core 8.0 ve EntityFramework, arka uç tarafında güçlü ve güvenilir bir yapı sunarken, React ve JavaScript ise arayüz gelişiminde esneklik ve hız sağlamıştır. Ancak, bazı durumlarda daha spesifik teknolojilerin veya ek kütüphanelerin kullanımı, projenin belirli alanlarını daha da geliştirebilir.

Gelecekte yapılması önerilen çalışmalar, mobil uygulama entegrasyonu, veri analizi ve raporlama, ek güvenlik önlemleri, test sunucuları vb. çalışmalar olabilir.

Ölçeklenebilirlik Sorunları: Geliştirilen uygulamanın çok sayıda kullanıcıya hizmet verirken performansının nasıl olacağı tam olarak test edilmemiştir. Gelecekte yapılacak çalışmalar, uygulamanın yüksek kullanıcı yüklerinde nasıl performans göstereceğini değerlendirmelidir.

Kullanıcı Davranışları: Kullanıcıların uygulamayı nasıl kullanacaklarına dair daha fazla veri ve geri bildirim toplanması, uygulamanın daha kullanıcı odaklı geliştirilmesine yardımcı olabilir.

##### KAYNAKLAR

<https://learn.microsoft.com/en-us/aspnet/core/getting-started/?view=aspnetcore-8.0>

<https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-8.0&WT.mc_id=dotnet-35129-website>

<https://learn.microsoft.com/tr-tr/ef/core/get-started/overview/first-app?tabs=netcore-cli>

<https://www.kodlama.io/p/yazilim-gelistirici-yetistirme-kampi-2024-1>

##### EKLER

Ek 1 Dış Kapak Biçimi

Ek 2 İç Kapak Biçimi