

**T.C.
ERZİNCAN BİNALİ YILDIRIM ÜNİVERSİTESİ
MÜHENDİSLİK-MİMARLIK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

LİSANS BİTİRME PROJESİ RAPORU

**Üniversite Kırtasiyelerinde Fotokopi Hizmetleri İçin Dijital Sipariş
ve Yönetim Sistemi**

Bayram GÜNGÖR

Danışman: Dr. Öğr. Üyesi Funda AKAR

Haziran 2024

ÖZET

Lisans Bitirme Projesi

Üniversite Kırtasiyelerinde Fotokopi Hizmetleri İçin Dijital Sipariş ve Yönetim Sistemi

Bayram GÜNGÖR

Erzincan Binali Yıldırım Üniversitesi
Mühendislik-Mimarlık Fakültesi
Bilgisayar Mühendisliği Bölümü

Danışman: Dr. Öğr. Üyesi Funda AKAR

Bu çalışma da sınav dönemlerinde fotokopi işlemleri sırasında oluşan yoğunluk nedeniyle uzun süre beklemek zorunda kalan öğrencilerin ve yalnızca günün belirli saatlerinde yoğunluk yaşayan kırtasiyecilerin karşılaştığı sorunlara çözüm sunmayı amaçlayan bir uygulama geliştirilmesini kapsamaktadır.

Öğrencilerin mevcut konumlarına yakın olan kırtasiyeleri listeleyebilmek için seçilen kırtasiyede bulunan mevcut fotokopileri incelemelerine veya kendi dosyalarını sisteme yükleyerek kırtasiyeye göndermelerine imkân sağlayan bir proje geliştirilmiştir. Uygulamada öğrencilerin kırtasiyelerdeki fotokopi fiyatları hakkında bilgi olabilmeleri sağlanarak, ekonomik açıdan en uygun seçimi yapmalarına yardımcı olmaktadır. Ayrıca, öğrencilerin sıkça karşılaştığı, belirli bir öğretim üyesine ait notların hangi kırtasiyede bulunduğu sorunu da bu uygulama ile çözüme kavuşturulacaktır. Kırtasiyeciler, sisteme yüklenen not veya belgelere erişim sağlayarak baskı işlemini gerçekleştirebilecektir.

Proje yazıldığı tarih de ki mevcut olan en güncel teknolojiler ve kütüphaneler kullanılarak geliştirilmiştir. Proje kapsamında, .NET Core 8.0 ile oluşturulmuş bir backend uygulaması ve kırtasiyeler ile müşteriler için ayrı ayrı olmak üzere Angular 17.3.1 ile yazılmış iki adet frontend uygulaması bulunmaktadır.

Anahtar Kelimeler: .Net Core 8.0 ile Web Uygulaması Geliştirme, Fotokopi işlemlerinin sanallaştırılması.

ABSTRACT

Undergraduate Graduation Project

Digital Order and Management System for Photocopy Services in University Copy Centers

Bayram GÜNGÖR

Erzincan Binali Yıldırım University
Faculty of Engineering and Architecture
Department of Computer Engineering

Supervisor: Assist. Prof. Funda AKAR

This study aims to develop an application to address the problems faced by students who have to wait for long periods during exam times due to the high demand for photocopy services and by copy center owners who experience peak demand only during certain hours of the day.

The project enables students to list copy centers near their current location, review existing photocopies available at the selected copy center, or upload their own documents to be sent to the copy center. The application also provides information on photocopy prices at different copy centers, helping students make the most economical choice. Additionally, it resolves the common issue students face of identifying which copy center has notes from a specific instructor. Copy center owners can access the notes or documents uploaded to the system and perform the printing process.

The project has been developed using the latest available technologies and libraries at the time of writing. It includes a backend application created with .NET Core 8.0 and two separate frontend applications for copy centers and customers, both written in Angular 17.3.1.

Keywords: Web Application Development with .NET Core 8.0, Virtualization of Photocopy Processes

TEŞEKKÜR

Lisans eğitimim süresince ve bitirme projemin tamamlanmasında bana maddi ve manevi her türlü desteği sağlayan, yol gösterici rehberliği ile yanımda olan saygıdeğer danışman hocam Dr. Öğr. Üyesi Funda Akar'a en içten teşekkürlerimi sunarım. Hocamın yönlendirmeleri, bilgi ve tecrübesiyle bana kattığı değerler, eğitimim boyunca karşılaştığım zorlukların üstesinden gelmemde ve kişisel gelişimimde önemli bir rol oynamıştır. Projemin her aşamasında verdiği destek ve sağladığı motivasyon sayesinde hem bireysel hem de akademik anlamda büyük bir ilerleme kaydettim. Hocamın değerli katkıları, eğitim sürecime olan inancımı pekiştirmiş ve bana ilham kaynağı olmuştur. Onun rehberliğinde edindiğim bilgi ve deneyimler, gelecekteki çalışmalarım için sağlam bir temel oluşturmuştur.

Eğitimimi sürdürebilmem için temel motivasyon kaynağım olan ve her an yanımda olan aileme de sonsuz teşekkürlerimi sunarım. Onların sürekli desteği, sevgi dolu yaklaşımları ve bana olan inançları, bu süreçteki en büyük güç kaynağım olmuştur.

Ek olarak projemde kullandığım teknolojilerin eğitimlerinde en çok faydalandığım kaynak üreticisi olan Gençay Yıldız'a da teşekkür etmek isterim. Onun hazırladığı eğitim materyalleri ve kaynaklar projede kullandığım teknolojileri derinlemesine anlamamı sağladı ve uygulamada karşılaştığım teknik sorunları aşmamda büyük katkı sağladı.

Bayram GÜNGÖR

HAZİRAN, 2024

İÇİNDEKİLER

	Sayfa
ÖZET.....	i
ABSTRACT	ii
TEŞEKKÜR	iii
İÇİNDEKİLER	iv
ŞEKİLLER LİSTESİ.....	vi
KISALTMALAR	ix
1. GİRİŞ.....	1
1.1. Bilgi Gereksinim Belirleme, Problemin Tanımlanması	2
Amaç	2
1.1.1. Konu ve Kapsam	2
1.2. Beklenen Fayda	2
1.3. Çalışma Takvimi	3
1.3.1. Kişi- İş Açıklaması (Projede kim hangi işte görev alacak):.....	3
1.4. Sistem Gereksinimlerini Ortaya Çıkarma Yöntem ve Teknikleri	4
1.5. Sistemsel Gereksinimler	5
1.5.1. İşlevsel Gereksinimler	5
1.5.2. Sistem ve Kullanıcı Ara yüzleri ile ilgili Gereksinimler	6
1.5.3. Veriyle İlgili Gereksinimler.....	6
1.5.4. Kullanıcılar ve İnsan Faktörü Gereksinimleri, Güvenlik Gereksinimleri	6
1.5.5. Teknik ve Kaynak Gereksinimleri, Fiziksel Gereksinimler	6
2. LİTERATÜR ÖZETİ.....	7
3. MATERYAL ve YÖNTEM.....	9
3.1. Materyal	9
3.1.1. .NET Core.....	10
3.1.2. .NET Core Web API.....	12
3.1.3. Onion Architecture	14
3.1.4. MSSQL	16
3.1.5. Entity Framework Core	17
3.1.6. Generic Repository	19
3.1.7. SQRS	21

3.1.8. Microsoft Identity	22
3.1.9. JWT Bearer	23
3.1.10. Refresh Token	25
3.1.11. Serilog	27
3.1.12. iTextSharp	27
3.1.13. Angular	28
3.1.14. Bootstrap	29
3.2. Yöntem	31
3.2.1. Sistem Tasarımı	31
3.2.2. Kullanıcı ve Sistem Arayüzü Tasarımları	37
3.2.2.1. Müşteri Arayüz Tasarımı	37
3.2.2.2. Satıcı Arayüz Tasarımı	47
3.2.3. Yazılım Tasarımı	54
3.2.3.1. Backend Tasarımı	54
3.2.3.2. Müşteri Frontend Uygulamasının Tasarımı	61
3.2.3.2. Satıcı Frontend Uygulamasının Tasarımı	63
3.2.4. Veri Tabanı Tasarımı	65
4. SONUÇ ve ÖNERİLER	67
KAYNAKLAR	70
EKLER	72
Ek-1. Backend Uygulaması GitHub Adresi	72
Ek-2. Satıcı Frontend Uygulaması GitHub Adresi	72
Ek-2. Müşteri Frontend Uygulaması GitHub Adresi	72

ŞEKİLLER LİSTESİ

	Sayfa
Şekil 1. .NET Core	10
Şekil 2. .NET Core Web API	12
Şekil 3. Onion Architecture	14
Şekil 4. Entity Framework	18
Şekil 5. Generic Repository	19
Şekil 6. SQRS	21
Şekil 7. JWT.....	24
Şekil 8. Refresh Token.....	26
Şekil 9. Bootstrap.....	30
Şekil 10. Sipariş Oluşturma - Akış Şeması	31
Şekil 11. Satıcıları Listeleme - Akış Şeması.....	31
Şekil 12. Satıcı Bilgilerini Getirme - Akış Şeması	32
Şekil 13. Sepete Ürün Ekleme - Akış Şeması	32
Şekil 14. Sepete Özel Ürün Ekleme - Akış Şeması	33
Şekil 15. Satıcı Dosya Ekleme - Akış Şeması.....	33
Şekil 16. Satıcı Paket Ekleme – Akış Şeması	34
Şekil 17. Satıcının Siparişleri Görüntülemesi - Akış Şeması.....	34
Şekil 18. Satıcı Sipariş Durumunu Değiştirme - Akış Şeması.....	35
Şekil 19. Kullanıcı Girişi - Akış Şeması	35
Şekil 20. Kullanıcı Doğrulama - Akış Şeması	36
Şekil 21. Fiyat Belirleme Algoritması – Akış Şeması	36
Şekil 22. Müşteri Sistemi - Header ve Footer Tasarımı.....	37
Şekil 23. Müşteri Sistemi - Anasayfa Tasarımı 1.Kısım.....	37
Şekil 24. Müşteri Sistemi - Anasayfa Tasarımı 2.Kısım.....	38
Şekil 25. Müşteri Sistemi - Kullanıcı Girişi Tasarımı	38
Şekil 26. Müşteri Sistemi - Kullanıcı Kayıt Formu Tasarımı	40
Şekil 27. Müşteri Sistemi - Kullanıcı Kayıt Formu Doğrulama Tasarımı	40
Şekil 28. Müşteri Sistemi - Satıcı Seçimi Tasarımı	41
Şekil 29. Müşteri Sistemi - Satıcı Sayfası Tasarımı.....	42

Şekil 30. Müşteri Sistemi - Dosya Filtreleme Menü Tasarımı.....	42
Şekil 31. Müşteri Sistemi – Satıcı Dosyalarının Görüntülenmesi İçin Tasarım	42
Şekil 32. Müşteri Sistemi - Satıcı Paketlerinin Görüntülenmesi İçin Tasarım	43
Şekil 33. Müşteri Sistemi - Müşteri Dosya Yükleme İşlemi Modal Tasarımı.....	43
Şekil 34. Müşteri Sistemi - Satıcı Dosyalarını Görüntüleme Modal Tasarımı	44
Şekil 35. Müşteri Sistemi Ürün Paket Seçimi Modal Tasarımı	45
Şekil 36. Müşteri Sistemi - Ürünün Sepete Eklenmesi Modal Tasarımı	45
Şekil 37. Müşteri Sistemi - Alışveriş Sepeti Tasarımı	46
Şekil 38. Müşteri Sistemi - Sipariş Sayfasının Tasarımı	46
Şekil 39. Satıcı Sistemi - Kullanıcı Girişi Tasarımı.....	47
Şekil 40. Satıcı Sistemi - Kullanıcı Başvurusu Tasarımı	47
Şekil 41. Satıcı Sistemi - Satıcı Dosya Yönetim Sayfası Tasarımı.....	48
Şekil 42. Satıcı Sistemi – Satıcı Dosyalarının Görüntülenmesi İçin Tasarım.....	48
Şekil 43. Satıcı Sistemi - Satıcı Dosya Silmeyi Onaylama Modalının Tasarımı	48
Şekil 44. Satıcı Sistemi - Satıcı Dosya Güncelleme İşlemleri İçin Modal Tasarımı	49
Şekil 45. Satıcı Sistemi - Satıcı Dosya Ekleme Onay Modalının Tasarımı	49
Şekil 46. Satıcı Sistemi – Satıcı Paket Yönetim Sayfasının Tasarımı	50
Şekil 47. Satıcı Sistemi - Satıcı Paket Güncelleme Modalının Tasarımı	50
Şekil 48. Satıcı Sistemi – Satıcı Paket Güncelleme Onaylama Modalının Tasarımı.....	51
Şekil 49. Satıcı Sistemi - Satıcı Paket Silmeyi Onaylama Modalının Tasarımı	51
Şekil 50. Satıcı Sistemi - Satıcı Sipariş Yönetim Sayfasının Tasarımı.....	52
Şekil 51. Satıcı Sistemi - Satıcı Siparişteki Ürünleri Görüntülemesi İçin Tasarım	52
Şekil 52. Satıcı Sistemi - Satıcı Hesap Bilgileri Adres Değişikliği Tasarımı	53
Şekil 53. Satıcı Sistemi - Satıcı Hesap Bilgileri Şirket Değişikliği Tasarımı	53
Şekil 54. Satıcı Sistemi - Satıcı Hesap Bilgileri Görsel Değişikliği Tasarımı	53
Şekil 55. Onion Mimaride Proje Yapısı.....	54
Şekil 56. Backend Proje Yapısı.....	54
Şekil 57. Domain Katmanı.....	55
Şekil 58. Domain Katmanının da Kullanılan Nuget Paketleri	55
Şekil 59. Application Katmanının da Kullanılan Nuget Paketleri	56
Şekil 60. Application Katmanı Abstraction Klasörü.....	57
Şekil 61. Application Katmanı DTOs Klasörü	57

Şekil 62. Application Katmanı Features Klasörü.....	57
Şekil 63. Application Katmanı Repositories Klasörü	57
Şekil 64. Application Katmanı ViewModels Klasörü.....	57
Şekil 65. Infrastructure Katmanı	58
Şekil 66. Infrastructure Katmanında Kullanılan Nuget Paketi.....	58
Şekil 67. Persistence Katmanı.....	59
Şekil 68. Persistence Katmanında Kullanılan Nuget Paketleri	59
Şekil 69. API Katmanı	60
Şekil 70. API Katmanında Kullanılan Nuget Paketleri.....	60
Şekil 71. Müşteri Frontend Uygulamasının Module - Component İlişkisi.....	61
Şekil 72. Müşteri Frontend Uygulamasının Route Yapısı	62
Şekil 73. Müşteri Frontend Uygulamasının Klasör Yapısı	62
Şekil 74. Satıcı Frontend Uygulamasının Module - Component İlişkisi	63
Şekil 75. Satıcı Frontend Uygulamasının Route Yapısı.....	64
Şekil 76. Satıcı Frontend Uygulamasının Klasör Yapısı	64
Şekil 77. Veri Tabanındaki İlişkiler	65
Şekil 78. Veri Tabanı Tablo İçerikleri Kısım 1.....	65
Şekil 79. Veri Tabanı Tablo İçerikleri Kısım 2.....	66
Şekil 80. Veri Tabanı Tablo İçerikleri Kısım 3.....	66

KISALTMALAR

Kisaltmalar

API	Application Programming Interface
CLI	Command Line Interface
ORM	Object to Relational Mapping
EF	Entity Framework
MVC	Model-View-Controller
LINQ	Language Integrated Query
JWT	JSON Web Token
SPA	Single Page Application
DI	Dependency Injection
CRUD	Create-Read-Update-Delete
Mig	Migration
DTO	Data Transfer Object
SQL	Structured Query Language
Db	Database
Ms	Microsoft
ng	Angular

1. GİRİŞ

Üniversite kırtasiyeleri sınav dönemlerinde yoğunluk yaşamaktadır. Yaşanan yoğunluk genellikle aynı zaman diliminde bulunan ders araları ve öğle araları gibi zamanlardır. Bu zamanlarda kırtasiyeye gelen öğrenciler bazen uzun süre sıra bekleyerek fotokopilerini alabilmekte bazense beklemelerine rağmen süreleri yetişmediği için fotokopilerini hiç alamamaktadır. Fotokopiler ise bu zaman diliminde yetişemedikleri müşterilerin bir kısmını ileri bir tarihe kadar ertelemekte bir kısmını ise kaybetmektedir. Öğrencilerin notlarını daha geniş bir zaman dilimi içerisinde internetten yükleyebilecekleri ve kırtasiyeye gitmeden önce fotokopisinin hazır gelmesini sağlayacak bir sistem gerekmektedir. Belgeler sisteme yüklenirken yazdırma seçeneklerini (Siyah-Beyaz, Yatay vb.) öğrenerek fotokopinin onayına gönderir. Fotokopici onaylama yaptıktan sonra işlem adımlarını (Onaylandı, Hazırlanıyor, Hazır) sisteme girerek fotokopi durumunu müşteriye bildirir. Öğrenciler fotokopi durumu hazır hale geldikten sonra fotokopilerini teslim almak için kırtasiyeye gider.

Böyle bir sistemin bulunması aynı zamanda kırtasiyelerdeki belge transferi için harcanan zamanı azaltarak yoğunluğu düşürür ve öğrenci güvenliğini artırır. Bazı öğrenciler gerekli belgeleri USB aracılığı ile transfer ederek hızlandırmaktadır fakat bu yöntem günde yüzlerce USB'nin ve kaynağı belirsiz dosyanın açıldığı bilgisayarlarda kullanmak için tehlikelidir. Diğer transfer yöntemi genelde üniversitelerin sistemlerinde bulunan notları indirmek için sisteme giriş yapmaktır. Bu yöntem ise sisteme giriş yaptıktan sonra gerekli belgeyi bulana kadar fazlaca zaman kaybettirmektedir. Kaybettirdiği zamanın yanı sıra keylogger gibi zararlı yazılımları barındırma ihtimali çok yüksek olan bu bilgisayarlarda tehlikelidir. Günümüzde en çok kullanılan yöntem ise Whatsapp aracılığı ile belgeleri göndermektir. Bu yöntem ise kırtasiyelerdeki bilgisayar ekranlarının müşterilerin görebileceği şekilde konumlandırılması sebebiyle telefon numaranızın yayılmasına sebep olabilir. Geliştirilecek olan uygulama hem giriş yapmak için harcanan zaman kaybını engelleyerek daha hızlı fotokopi baskısı yapılmasını sağlayacak hem de öğrenci ile kırtasiye arasındaki veri alışverişi sistemle kesileceği için daha güvenli hale gelecektir.

Fotokopi çektmek için az tercih edilse de internet üzerinden Whatsapp ile sipariş alan bazı internet sitelerini kullanmaktır. Bu siteler uygun fiyatta fotokopi hizmeti verdiği için

tercih edilirler. Az tercih edilme sebepleri ise kargo bekleme süreleri, kargo ücreti ödemeleri ve belirli bir konumlarının olmamasıdır. Bunların yanı sıra normal bir kırtasiyede yanlışlık olması durumunda birkaç dakika içinde düzeltilebilecek küçük bir hatanın bile hem zaman maliyeti hem de kargo maliyeti yüksektir. Geliştirilecek olan sistemin diğer sistemlerden farkı öğrenciler fotokopi siparişlerini daha önce bildikleri, konumu yakın kırtasiyelerden sipariş verecekleri için kargo sürecini beklemeyecek ve hata durumunda anlık müdahalede bulunabileceklerdir.

1.1. Bilgi Gereksinim Belirleme, Problemin Tanımlanması

Amaç

Bu projenin amacı, üniversite öğrencilerinin sınav dönemlerinde fotokopi işlemleri sırasında yaşadıkları uzun bekleme sürelerini azaltmak ve kırtasiyecilerin belirli saatlerde yaşadığı yoğunluk problemlerine çözüm sunmaktır.

1.1.1. Konu ve Kapsam

Proje, öğrencilerin konumlarına yakın kırtasiyeleri listeleyerek seçilen kırtasiyede mevcut fotokopi belgelerini incelemelerini veya kendi belgelerini sisteme yükleyerek kırtasiyeye göndermelerini sağlar. Ayrıca, öğrenciler fotokopi fiyatları hakkında bilgi edinerek ekonomik açıdan en uygun seçimi yapabilirler. Proje, .NET Core 8.0 backend ve Angular 17.3.1 frontend teknolojileri kullanılarak geliştirilmiştir.

1.2. Beklenen Fayda

- Öğrencilerin sınav dönemlerinde uzun süre beklemelerini önlemek.
- Kırtasiyecilerin yoğun saatlerdeki iş yükünü azaltmak.
- Belgelerin güvenli bir şekilde transfer edilmesini sağlamak.
- Öğrencilerin ekonomik olarak en uygun kırtasiyeyi seçmelerine yardımcı olmak.
- Öğrencilerin belirli öğretim üyelerine ait notları kolayca bulmalarını sağlamak

1.3.Çalışma Takvimi

İŞ-ZAMAN ÇİZELGESİ

İş Paketi Ad/Tanım	HAFTALAR															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Proje Başlangıcı ve Planlama	x															
Gereksinim Analizi		x	x													
Sistem Tasarımı				x	x	x										
Backend Geliştirme						x	x	x	x	x	x	x				
Frontend Geliştirme							x	x	x	x	x	x	x			
Test ve Entegrasyon										x	x	x	x	x		
Proje Kapanışı ve Raporlama															x	x

1.3.1. Kişi- İş Açıklaması (Projede kim hangi işte görev alacak):

Yürütücü : Bayram GÜNGÖR

Danışman : Dr. Öğr. Funda AKAR

1.4. Sistem Gereksinimlerini Ortaya Çıkarma Yöntem ve Teknikleri

Yüz Yüze Görüşme

Sistem gereksinimlerini belirlemek amacıyla, doğrudan kullanıcılarla yapılan yüz yüze görüşmeler önemli bir bilgi kaynağı olmuştur. Bu kapsamda:

- **Öğrencilerle Görüşmeler:** Çeşitli bölümlerde okuyan ve fotokopi hizmetlerinden sıkça faydalanan öğrencilerle yapılan görüşmelerde, fotokopi işlemleri sırasında karşılaştıkları zorluklar, ihtiyaç duydukları hizmetler ve mevcut sistemlerdeki eksiklikler hakkında geri bildirimler toplandı. Öğrenciler, özellikle sınav dönemlerinde yaşadıkları yoğunluk ve bekleme süreleri hakkında detaylı bilgi verdiler.
- **Kırtasiye Sahipleri ile Görüşmeler:** Üniversite yakınındaki kırtasiyecilerle yapılan görüşmelerde, iş süreçleri, yoğun saatlerde yaşanan problemler ve öğrencilerin ihtiyaçlarına nasıl daha iyi hizmet verebilecekleri üzerine tartışmalar yapıldı. Kırtasiye sahipleri, fotokopi işlemlerinin yönetimi ve müşteri memnuniyetini artırmak için önerilerde bulundular.

Gözlem

Doğrudan kırtasiye ve öğrenci davranışlarını yerinde gözlemlemek, sistem gereksinimlerini anlamak için kritik bir rol oynamıştır. Bu gözlemler sırasında:

- **Kırtasiyelerdeki Yoğun Saatlerin İncelenmesi:** Özellikle sınav dönemlerinde kırtasiyelerde yaşanan yoğunluk gözlemlendi. Öğrencilerin hangi saatlerde daha fazla geldiği, hangi tür belgeleri daha çok fotokopi çektirdikleri ve kırtasiyecilerin bu yoğunlukla nasıl başa çıktıkları not edildi.
- **Öğrencilerin Fotokopi İşlemleri:** Öğrencilerin fotokopi işlemlerini nasıl gerçekleştirdikleri, belge transferi için hangi yöntemleri kullandıkları (USB, Email, WhatsApp vb.), sırada beklerken yaşadıkları sıkıntılar ve genel kullanıcı deneyimleri yerinde gözlemlendi. Bu gözlemler, kullanıcı ihtiyaçlarının daha iyi anlaşılmasını sağladı.

Veri Akış Şemaları

Kavramsal ve mantıksal veri akış şemaları, sistemin nasıl çalışacağını ve veri hareketlerinin nasıl olacağını görselleştirmek için kullanıldı:

- **Kavramsal Veri Akış Şemaları:** Sistemin genel işleyişini gösteren şemalar oluşturuldu. Bu şemalar, öğrencilerin ve kırtasiyecilerin sistemdeki etkileşimlerini, belge yükleme ve onaylama süreçlerini görselleştirdi.
- **Mantıksal Veri Akış Şemaları:** Kavramsal şemalardan daha detaylı olarak, her bir işlevin nasıl gerçekleştirileceği, hangi verilerin hangi süreçlerden geçeceği ve bu süreçlerdeki karar noktaları belirlendi. Bu şemalar, sistem tasarımında rehberlik etti ve yazılım geliştirme sürecinde kullanıldı.

İş Akış Şeması

Fotokopi işlemleri için detaylı iş akış şemaları çizildi:

- **Fotokopi Siparişi İş Akışı:** Öğrencilerin sisteme belge yüklemesi, baskı seçeneklerini seçmesi ve kırtasiyeye sipariş göndermesi ile başlayan süreç kırtasiyecinin siparişi onaylaması, baskıyı hazırlaması ve öğrenciye teslim etmesi adımlarını içerir. Her adımda sistemin ve kullanıcıların yapması gereken işlemler detaylı olarak gösterildi.
- **Onay ve Durum Güncellemeleri:** Kırtasiyecinin belgeyi onaylaması, baskı işleminin durumunu güncellemesi ve bu bilgilerin öğrenciye iletilmesi süreçleri ayrıntılı olarak şemalar çizildi.

Bu yöntemler ve teknikler, sistem gereksinimlerinin kapsamlı bir şekilde belirlenmesine ve doğru bir şekilde dokümanite edilmesine yardımcı olmuştur. Böylece geliştirilecek uygulamanın hem öğrenciler hem de kırtasiyeciler için verimli ve kullanışlı olması sağlanmıştır.

1.5. Sistemsel Gereksinimler

1.5.1. İşlevsel Gereksinimler

- Öğrencilerin kırtasiyeleri listeleyebilmesi

- Belgelerin sisteme yüklenmesi ve baskı seçeneklerinin belirlenmesi
- Kırtasiyelerin baskı taleplerini onaylaması ve durum güncellemeleri
- Fotokopi fiyatlarının sayfa sayısına göre hesaplanması

1.5.2. Sistem ve Kullanıcı Ara yüzleri ile ilgili Gereksinimler

- Kullanıcı dostu ve erişilebilir arayüz tasarımı
- Kırtasiyeler için yönetim paneli
- Öğrenciler için sipariş takip ekranı

1.5.3. Veriyle İlgili Gereksinimler

- Güvenli belge transferi ve depolama
- Öğrenci ve kırtasiye bilgileri için veri tabanı yönetimi
- Fotokopi sipariş verilerinin yönetimi

1.5.4. Kullanıcılar ve İnsan Faktörü Gereksinimleri, Güvenlik Gereksinimleri

- Kullanıcı eğitimi ve destek dokümantasyonu
- Kullanıcı doğrulama ve yetkilendirme mekanizmaları
- Verilerin gizliliği ve güvenliği

1.5.5. Teknik ve Kaynak Gereksinimleri, Fiziksel Gereksinimler

- .NET Core 8.0 ve Angular 17.3.1 kullanımı
- Sunucu ve veri tabanı altyapısı
- Yeterli donanım ve ağ bağlantısı
- Gerekli yazılım lisansları ve kütüphaneler

2. LİTERATÜR ÖZETİ

Fotokopi işlemleri için satıcılar ile müşteriler arasında bağlantı kuran e-ticaret sisteminin oluşturulması hakkında literatür araştırılması yapıldığında, konu hakkında hem yerli ve hem de yabancı kaynaklarda yeteri kadar bilgi olmadığı görülmüştür. Bu kapsamda hazırlanan projenin literatüre katkı sağlayacaktır.

Türkiye'de e-ticaret, 6563 sayılı Elektronik Ticaretin Düzenlenmesi Hakkında Kanun'da "fiziksel olarak karşı karşıya gelmeden, elektronik ortamda gerçekleştirilen çevrimiçi iktisadi ve ticari her türlü faaliyet" şeklinde tanımlanmaktadır (T.C. Ticaret Bakanlığı, 2019). Elektronik ticaret çağımızın yaşam biçimini dönüştüren önemli bir olgudur. Bilgi teknolojilerindeki ilerlemeler ve ekonomik iletişimle ortaya çıkan e-ticaret, geleneksel ticaretin birçok sınırlamasını ortadan kaldırmıştır. Fiziksel mağaza gereksinimini elimine eden sanal pazarlar, kullanıcılara her an ve her yerden alışveriş yapma imkânı sunmaktadır. E-ticaret, ürün seçimi, sipariş, reklam ve elektronik ödeme süreçlerini kapsar (Nanehkaran, 2013). E-ticaret sitelerinin kullanımı her yıl önemli ölçüde artış göstermektedir. 2019 yılında e-ticaret hacmi 136 milyar TL olarak kaydedilirken, 2020 yılında bu rakam 226,2 milyar TL'ye ulaşmıştır. Bu hızlı büyüme trendi, sonraki yıllarda da devam etmiş ve 2022 yılı itibarıyla e-ticaret hacmi 800,7 milyar TL'ye kadar yükselmiştir (Ateş, 2024). Bu veriler, e-ticaretin ekonomik büyüklüğünün ve dijital alışveriş alışkanlıklarının ne denli hızla değiştiğini ve geliştiğini açıkça ortaya koymaktadır. E-ticaret sektöründeki bu dramatik artış, dijital platformların ekonomik faaliyetlerdeki önemini ve etkisini vurgulamakta, aynı zamanda gelecekte bu alanda daha fazla yenilik ve gelişme beklenebileceğini göstermektedir.

E-ticaretin ortaya çıkışı ve gelişimi, baskı endüstrisini de diğer endüstri kolları gibi derinden etkilemiş ve bu bağlamda Web to Print (W2P) kavramının ortaya çıkmasına neden olmuştur. Günümüzde tüm üretim sistemleri bu yeni kavrama uyum sağlamaya çalışmaktadır. Ofset, gravür, flekso, serigrafi ve dijital baskı yöntemlerinin internet entegrasyonu olmadan sürdürülebilirliği artık mantıklı görünmemektedir. Bu noktada, matbaacılık endüstrisinin maliyetleri azaltmak ve verimliliği artırmak amacıyla Web to Print platformları geliştirme eğilimi, stratejik bir yaklaşım olarak değerlendirilmektedir. Bu sektörel değişimi başarıya dönüştürmek ise dijital çözümleri doğru tüketicilere

sunmakla mümkündür. Baskı işlemleri bir kez elektronik ortama taşındığında, yeni baskı yöntemleri ile iş yönetimi ve takibine yönelik yeni yaklaşımlar mümkün hale gelmektedir. Basılacak işler ve iş akış süreçleri dijitalleşmekte ve matbaacılık sektörü için yeni bir dönemin kapıları aralanmaktadır. Bu bağlamda, baskı üretim sistemleri günümüzde iki ana kategoriye ayrılmaktadır: web tabanlı iş akış sistemleri ve geleneksel iş akış sistemleri. Web tabanlı baskı üretim sistemleri, basılı ürünlerin yönetimi ve üretimi ile baskıya dayalı hizmetlerin sunumunda internet tabanlı süreçler kullanmaktadır. Bu web destekli hizmetler, basit dosya yönetimi hizmetlerinden (örneğin, dosya aktarım protokolü - FTP) tamamen web tabanlı belge oluşturma ve yerine getirme yönetimi çözümlerine kadar geniş bir yelpazeyi kapsamaktadır (Mutlu et al., 2018).

Baskı endüstrisi, baskı talebinin azalması ve müşteri beklentilerini karşılama zorlukları ile karşı karşıyadır. Günümüzde, baskı operasyonlarını tanımlayan başlıca unsurlar arasında daha kısa teslim süreleri, maliyet kontrolü ve yüksek kaliteli ürün talebi yer almaktadır. Ayrıca, işletmeler, yetenekli iş gücünün azalması, işletme içi maliyet kontrolü ve üretim materyalleri ile dış kaynakların yönetimi gibi zorluklarla karşılaşmaktadır. Müşterilere ilişkin bilgilerin hızlı bir şekilde iletilmesi gerekliliği de bu zorluklar arasında bulunmaktadır. Bu eğilimler, endüstri içinde otomatikleştirilmiş ve entegre süreçlerin uygulanmasına yönelik yeni yaklaşımların ortaya çıkmasına neden olmuştur. Günümüzde, baskı üretimi, çoğu adımın izole edildiği ve otomatik kontrollerle donatıldığı süreçlerle gerçekleştirilmektedir. Otomatik üretimdeki bir sonraki adım, tüm operasyonların sürece entegre edilmesidir. Sürekli bir iş akışında, işlem müşterinin talebiyle başlayacak, baskı öncesi aşamaya getirilecek ve ardından baskı, stok kontrolü, dağıtım ve sonlandırma aşamalarını içeren baskı sonrası aşaması ile tamamlanacaktır (Özer, 2008). Yoğun rekabetin hüküm sürdüğü baskı sektöründe, yeni iletişim teknolojilerinin ve kanallarının etkin kullanımı, işletmelerin büyüme potansiyelini önemli ölçüde artırabilir. Matbaa işletmelerinin mevcut ekipmanları, iletişim kapasiteleri, hedefleri ve yatırım olanaklarına bağlı olarak en uygun Web2Print sistemini seçmeleri ve kullanıcı dostu çevrimiçi sistemlerle entegrasyon sağlamaları zorunludur. Yeni faaliyete geçen veya yeni iletişim teknolojilerine uyum sürecini tamamlamamış işletmeler, başlangıçta basit işlem odaklı e-iş stratejileri benimseyebilir. Bu işletmeler, e-iş yeteneklerini geliştirdikçe, yeni stratejiler belirleyerek sürdürülebilir bir büyüme sağlayabilirler (Mutlu et al., 2018).

3. MATERYAL ve YÖNTEM

3.1. Materyal

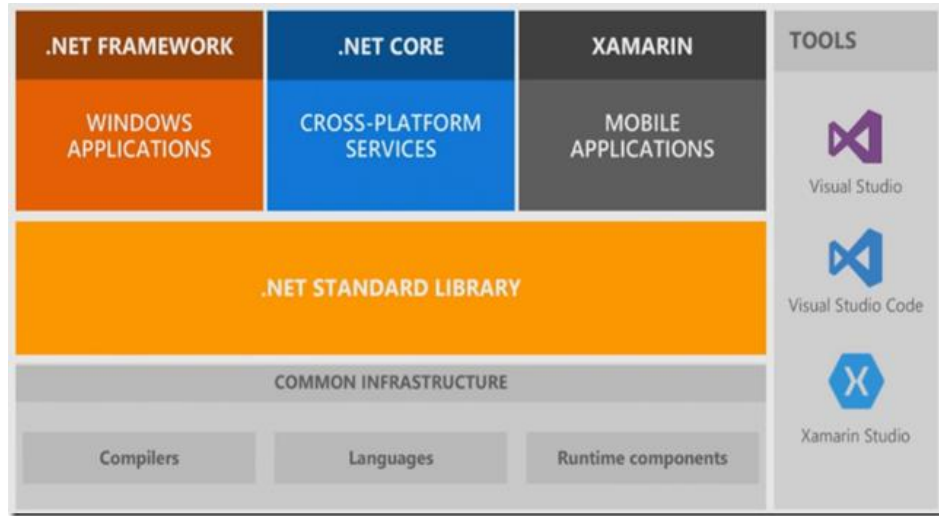
Projede Kullanılan Teknolojiler ve Yapılar	
Adı/Modeli	Projede Kullanım Amacı
.NET Core 8.0	Web API geliştirme için temel platform.
.NET Core Web API	REST API'ler oluşturmak ve sunmak için kullanılmıştır.
Onion Architecture	Katmanlı ve bağımsız bir uygulama mimarisi oluşturmak için kullanılmıştır.
MSSQL	Veri tabanı yönetim sistemidir.
EF Core	ORM aracı olarak kullanılmıştır.
Generic Repository	Veri erişim katmanını soyutlamak için kullanılmıştır.
SQRS (MediatR)	Komut ve sorgu modellerini ayırmak için kullanılmıştır.
Microsoft Identity	Kullanıcı kimlik doğrulaması ve yetkilendirme için kullanılmıştır.
JwtBearer	JSON Web Token tabanlı kimlik doğrulaması için kullanılmıştır.
Refresh Token	Kullanıcı oturumlarını yenilemek için kullanılmıştır.
Serilog	Günlük log kaydı ve hata ayıklama için kullanılmıştır.
iTextSharp	PDF belgeleri yönetmek için kullanılmıştır.
Angular 17.3.1	Web uygulama arayüzünü geliştirme için kullanılmıştır.
Bootstrap 5.3	Web arayüzü tasarımında hızlandırmak, modern ve duyarlı bir web kullanıcı arayüzü oluşturmak için kullanılmıştır.

WebFotokopi sistemi geliştirme sürecinde çeşitli teknolojiler, tasarım desenleri, kütüphaneler ve yapılar kullanılarak oluşturulmuştur. Bu sistemde temel olarak .NET Core 8.0 platformu tercih edilmiş ve bu platform üzerinde REST API'lerin geliştirilmesi için .NET Core Web API kullanılmıştır. Uygulama mimarisi olarak Onion Architecture tercih edilmiş, bu sayede katmanlı ve bağımsız bir yapı oluşturulmuştur. Veri tabanı yönetimi için MSSQL kullanılmış ve veri erişim işlemlerini yönetmek amacıyla Entity Framework Core (EF Core) ORM aracı kullanılmıştır. Ayrıca veri erişim katmanını soyutlamak ve genel CRUD operasyonlarını gerçekleştirmek için Generic Repository kullanılmıştır. Sistem içinde komut ve sorgu modellerinin ayrılması amacıyla SQRS (MediatR) kullanılmıştır. Kullanıcı kimlik doğrulaması ve yetkilendirme işlemleri için Microsoft Identity ve JSON Web Token (JwtBearer) tabanlı kimlik doğrulaması

kullanılmıştır. Kullanıcı oturumlarını yönetmek ve yenilemek amacıyla Refresh Token mekanizması entegre edilmiştir. Hata ayıklama ve günlük log kaydı için Serilog kullanılmıştır. PDF belgelerini yönetmek için iTextSharp kütüphanesi tercih edilmiştir. Web uygulama arayüzünün geliştirilmesi için Angular 17.3.1 kullanılmış, bu sayede modern ve kullanıcı dostu bir arayüz oluşturulmuştur. Arayüz tasarımında hız kazanmak ve duyarlı bir web kullanıcı arayüzü sağlamak amacıyla Bootstrap 5.3 kullanılmıştır.

3.1.1. .NET Core

.NET Core, Microsoft tarafından geliştirilen ve açık kaynak kodlu, çapraz platform desteğine sahip bir uygulama geliştirme çerçevesidir. Bu modern çerçevenin kökenleri, Microsoft'un 2002 yılında piyasaya sürdüğü ve büyük ölçüde Windows işletim sistemi üzerinde çalışan .NET Framework'e dayanmaktadır. Ancak, yazılım geliştirme dünyasında hızla değişen ihtiyaçlar ve çeşitli platformlarda çalışabilen uygulamalara duyulan gereksinim, Microsoft'u yeni bir strateji geliştirmeye yöneltti. 2014 yılında, bu ihtiyaçlara cevap verebilmek için .NET Core duyuruldu.



Şekil 1. .NET Core (Microsoft Developer Support, 2017)

.NET Core, başlangıçta .NET Framework'ün yeniden yapılandırılmış bir versiyonu olarak ortaya çıktı, ancak zamanla kendi kimliğini kazandı ve geliştiriciler arasında hızla popülerlik kazandı. .NET Core'un evrimi, sürekli olarak yeni sürümler ve özellikler ile desteklenmiştir. 2016 yılında piyasaya sürülen ilk stabil sürümden sonra, .NET Core hızla gelişti ve her yeni sürümde daha fazla özellik, performans iyileştirmeleri ve geliştirilmiş

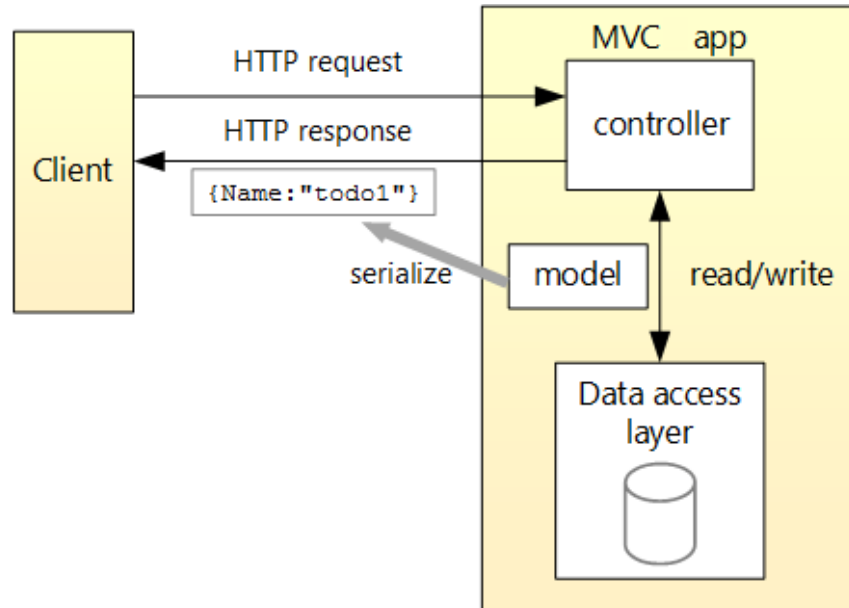
kullanıcı deneyimi sundu. 2020 yılında .NET 5.0 ile birlikte, Microsoft tüm .NET ekosistemini birleştirme yolunda önemli bir adım attı ve .NET Core, .NET 5.0 çatısı altında birleşti. Bu birleşme, geliştiricilere daha tutarlı ve birleşik bir geliştirme deneyimi sundu. 2021 yılında .NET 6.0 ve 2022 yılında .NET 7.0 sürümleriyle devam eden bu süreç, .NET Core'un sürekli gelişimini ve iyileştirilmesini sağladı. Son olarak, 2023 yılında duyurulan .NET 8.0 sürümü ile, platform daha da güçlenmiş ve modern yazılım geliştirme ihtiyaçlarına daha fazla cevap verebilir hale gelmiştir.

.NET Core'un temel özelliklerinden biri, çapraz platform desteği sunmasıdır. Bu sayede geliştiriciler, aynı kod tabanını kullanarak uygulamalarını Windows, macOS ve Linux gibi farklı işletim sistemlerinde çalıştırabiliyorlar. Bu, yazılım geliştirme süreçlerinde büyük bir esneklik sağlar ve uygulamaların daha geniş bir kullanıcı kitlesine ulaşmasını mümkün kılar. Ayrıca, .NET Core'un açık kaynak kodlu olması, dünya çapındaki geliştiricilerin katkıda bulunabilmesine ve çerçevenin daha hızlı bir şekilde gelişmesine olanak tanır. GitHub üzerinde açık kaynak olarak geliştirilen .NET Core, sürekli olarak topluluktan gelen geri bildirimler ve katkılar ile evrim geçirir, bu da onu dinamik ve güncel bir platform haline getirir. .NET Core'un performans ve ölçeklenebilirlik açısından sunduğu avantajlar da dikkate değerdir. Yüksek performanslı ve ölçeklenebilir uygulamalar geliştirmek için optimize edilmiştir. Özellikle web uygulamaları için kullanılan Kestrel web sunucusu, hafif ve hızlı yapısıyla dikkat çeker ve .NET Core uygulamalarının yüksek trafik altında bile sorunsuz çalışmasını sağlar. Ayrıca, .NET Core'un modüler yapısı, geliştiricilere sadece ihtiyaç duydukları bileşenleri projelerine dahil etme esnekliği sunar. Bu modüler yapı, uygulamaların daha hafif ve hızlı olmasını sağlar çünkü gereksiz bileşenler projeye dahil edilmez. .NET Core'un CLI desteği, geliştiricilere büyük kolaylık sağlar. CLI araçları, projelerin komut satırından yönetilmesine olanak tanır. Proje oluşturma, derleme, test etme ve yayımlama gibi işlemler, CLI araçları sayesinde hızlı ve verimli bir şekilde gerçekleştirilebilir. Bu, özellikle otomasyon süreçlerinde ve sürekli CI/CD süreçlerinde büyük bir avantaj sağlar. Ayrıca .NET Core'un modern yazılım geliştirme yöntemleri ile uyumlu olması da önemli bir avantajdır. Visual Studio Code gibi hafif ve güçlü geliştirme ortamları ile sorunsuz çalışır ve geliştiricilere modern araçlarla çalışma imkânı sunar. Mikro hizmet mimarisi de .NET Core'un güçlü yönlerinden biridir. Mikro hizmet mimarisi, büyük ve karmaşık uygulamaların daha küçük, bağımsız hizmetlere bölünerek yönetilmesini sağlar. Bu

hizmetler bağımsız olarak geliştirilebilir, dağıtılabilir ve ölçeklenebilir. .NET Core'un hafif ve performans odaklı yapısı, mikro hizmetlerin etkin bir şekilde geliştirilmesine ve yönetilmesine olanak tanır. Geniş kütüphane ve araç desteği ise geliştiricilerin yaygın ihtiyaçlarını hızlıca karşılamalarına yardımcı olur. Bu kütüphane ve araçlar, tekrar tekrar aynı tekerleği icat etme zorunluluğunu ortadan kaldırır ve daha hızlı, güvenilir uygulamalar geliştirilmesini sağlar.

3.1.2. .NET Core Web API

.NET Core Web API, Microsoft tarafından geliştirilen ve .NET Core platformunda çalışan bir web API (Application Programming Interface) çerçevesidir. Web API'ler, farklı platformlar arasında veri iletişimini sağlayan, HTTP protokolü üzerinde çalışan ve genellikle JSON veya XML formatında veri alışverişi yapabilen uygulama arayüzleridir. .NET Core Web API, .NET Core'un güçlü özelliklerini kullanarak RESTful (Representational State Transfer) web servislerinin hızlı ve kolay bir şekilde geliştirilmesini sağlar. RESTful mimarinin prensiplerine uygun olarak tasarlanmış olan Web API'ler, HTTP protokolünü kullanarak kaynaklara erişimi sağlar ve CRUD operasyonlarını destekler. Bir .NET Core Web API projesi, ASP.NET Core MVC çerçevesi üzerine inşa edilmiştir. Şekil 2'de .NET Core Web API'nin yapısı gösterilmiştir.



Şekil 2. .NET Core Web API (Microsoft Learn, 2024)

.NET Core Web API geliştiricilere MVC tasarım desenini kullanarak uygulama mantığını, sunum katmanını ve veri modelini ayrı tutma esnekliği sağlar. Ayrıca ASP.NET Core'un middleware yapısı, isteği işlemek ve yanıt üretmek için kullanılabilir, bu da geliştiricilere isteklerin nasıl yönlendirileceği ve nasıl işleneceği konusunda büyük bir kontrol sağlar.

.NET Core Web API'nin özelliklerinden bazıları aşağıdaki gibidir;

Routing: ASP.NET Core MVC'nin güçlü yönlendirme motoru, gelen istekleri belirli bir işlem yöntemine yönlendirir. Bu, birden çok isteği tek bir uygulama noktasına yönlendirmek ve doğru işlem metodunu çağırmak için kullanılır.

Model Binding ve Validation: Gelen isteklerin gövdesinden veya sorgu parametrelerinden veri almak ve bu veriyi bir model nesnesine bağlamak için model bağlama (model binding) ve doğrulama (validation) işlemleri kullanılır. Bu, gelen verinin doğrulanması ve işleme alınması için önemlidir.

Dependency Injection (Bağımlılık Enjeksiyonu): ASP.NET Core, içinde bulunan yerleşik DI konteyneri sayesinde bağımlılık enjeksiyonunu destekler. Bu, bileşenler arasındaki bağımlılıkları azaltır ve kodun test edilebilirliğini artırır.

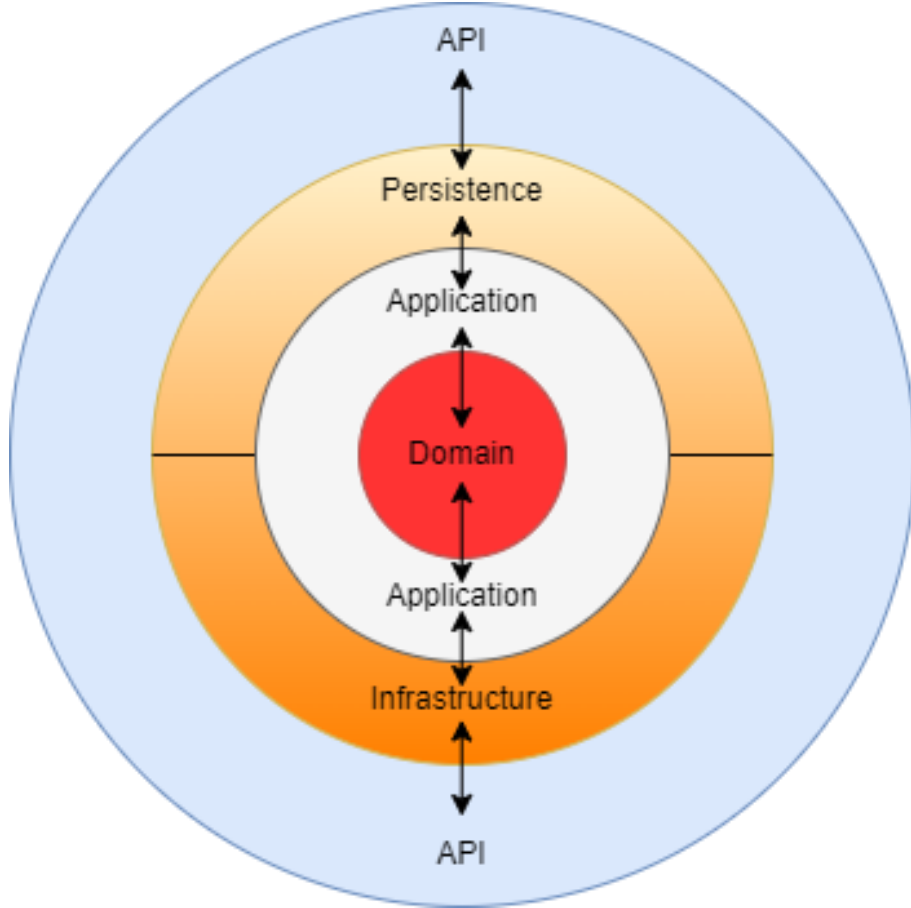
Middleware Pipeline: ASP.NET Core, HTTP isteklerini işlemek için bir middleware pipeline sağlar. Bu pipeline, isteği işlemek ve yanıtı oluşturmak için kullanılan bir dizi ara katmanı içerir. Geliştiriciler, middleware'leri uygulamalarına ekleyerek istekleri işlemek ve yanıtları oluşturmak için gereken özelleştirmeleri yapabilirler.

Swagger Desteği: Swagger veya OpenAPI belgeleri, API'lerin tanımlanması, belgelenmesi ve test edilmesi için standart bir formattır. ASP.NET Core, Swagger UI'ı kolayca entegre edebilir, bu da geliştiricilere API'lerini otomatik olarak belgeleme ve test etme imkânı sunar.

.NET Core Web API, geliştiricilere hızlı, güvenilir ve ölçeklenebilir web servislerinin geliştirilmesini sağlar. Ayrıca, .NET Core'un çapraz platform desteği sayesinde, Web API'ler farklı platformlarda da çalışabilir ve geniş bir kullanıcı kitlesine erişim sağlayabilir.

3.1.3. Onion Architecture

Onion Architecture (Soğan Mimarisi) yazılım sistemlerinin katmanlarını düzenleyen ve iş mantığını dış dünyaya karşı koruyan bir mimari tasarım yaklaşımıdır. Onion mimarisi, katmanları birbirinden bağımsız ve değişime kapalı bir şekilde organize ederek, uygulamanın sürdürülebilirliğini, test edilebilirliğini ve genişletilebilirliğini artırmayı amaçlar. İsmi Şekil 3'te ki gibi katmanların birbirlerine olan bağlantısının soğana benzemesinden almaktadır.



Şekil 3. Onion Architecture

Domain Katmanı: Domain katmanı, uygulamanın temel iş mantığını ve domain modellerini içerir. Bu katman genellikle bağımsız bir proje olarak ayrılır ve diğer katmanlarla doğrudan iletişim kurmaz. Domain katmanı, uygulamanın ana varlıklarını ve iş kurallarını temsil eden sınıfları içerir. Örneğin, müşteri, sipariş, ürün gibi domain modelleri ve bu modellerin iş kurallarını içeren sınıflar bu katmanda yer alır. Ayrıca, bu katman genellikle veri tabanı bağımsızdır ve iş mantığına odaklanır. Domain

katmanının temel amacı, uygulamanın iş kurallarını ve domain mantığını merkezi bir yerde toplamak ve dış dünyadan izole etmektir. Bu katman, uygulamanın çekirdek işlevselliğini temsil eder ve genellikle servis, repository veya diğer altyapı detaylarına bağımlı değildir.

Application Katmanı: Application katmanı, uygulamanın işlevselliğini gerçekleştiren servisler, uygulama servisleri (application services) ve DTO'ları içerir. Bu katman, domain katmanındaki domain modelleriyle dış dünya arasındaki arabirimi sağlar. Kullanıcı isteklerini alır, işlemleri yürütür ve sonuçları dış dünyaya döndürür. Örneğin, müşteri hizmetleri, ürün hizmetleri gibi servisler bu katmanda yer alır. Application katmanı, domain katmanındaki iş mantığını kullanarak belirli iş süreçlerini yönetir ve koordine eder. Bu katman, domain nesneleri ile çalışarak iş kurallarını uygular ve veriyi gerekli şekilde işleminden geçirir. Ayrıca, veri transfer nesneleri (DTO) kullanarak veri alışverişini düzenler ve domain nesnelerinin dış dünya ile paylaşılmasını sınırlar.

Persistence ve Infrastructure Katmanı: Persistence ve Infrastructure katmanı, veri tabanı erişimini, dış servis entegrasyonunu ve diğer altyapı işlemlerini yönetir. Persistence alt katmanı, veri tabanı ile etkileşim sağlar ve veri tabanına erişim işlemlerini gerçekleştirir. Bu alt katman, genellikle repository desenini kullanarak veri tabanı işlemlerini soyutlar ve veri tabanı ile ilgili detayları saklar. Infrastructure alt katmanı ise, dış servislerle iletişim kurar, dosya işlemleri gibi altyapı işlemlerini gerçekleştirir ve dış kaynaklara erişim sağlar. Bu katmanlar, genellikle Entity Framework, Dapper gibi ORM araçlarını ve harici API'leri kullanır. Persistence katmanı, domain katmanının veri tabanı bağımlılıklarından soyutlanmasına yardımcı olurken, infrastructure katmanı ise dış dünya ile olan tüm diğer teknik bağımlılıkları yönetir. Bu katmanlar, uygulamanın altyapı gereksinimlerini karşılar ve iş mantığının altyapı detaylarından bağımsız kalmasını sağlar.

Web API Katmanı: Web API katmanı, uygulamanın dış dünyaya açılan yüzüdür. HTTP isteklerini alır, iş mantığını yürütür ve sonuçları HTTP yanıtlarına dönüştürerek istemcilere iletilmesini sağlar. Bu katman genellikle ASP.NET Core gibi bir web frameworkü kullanır ve RESTful API'lerin oluşturulmasını sağlar. Web API katmanı, kullanıcı arayüzleri (UI) veya diğer istemci uygulamaları ile iletişim kurar. Bu katman,

genellikle kontrolörler (controllers) aracılığıyla HTTP isteklerini karşılar, application katmanındaki servisleri kullanarak işlemleri gerçekleştirir ve sonuçları istemcilere döner. Ayrıca, güvenlik, hata yönetimi ve isteklere yanıt verme gibi işlemleri de yönetir. Web API katmanı, istemcilerin uygulamanın işlevselliğine erişmesini sağlar ve iş mantığını dış dünyaya sunar.

Bu katmanlar arasındaki ilişkiler genellikle Onion Architecture prensiplerine göre organize edilir. Core katmanı, diğer katmanlara bağımlı olmayan, uygulamanın çekirdek işlevselliğini sağlayan katmandır. Application katmanı, Core katmanına bağımlıdır ve iş mantığını uygular. Persistence ve Infrastructure katmanları ise, iş mantığının veri tabanı ve dış dünya ile iletişimini sağlar. Web API katmanı ise, kullanıcı isteklerini alır, işlemleri yönetir ve sonuçları istemcilere iletilmesini sağlar. Onion Architecture yapısı uygulamanın sürdürülebilirliğini, test edilebilirliğini ve genişletilebilirliğini artırmaya yardımcı olur.

3.1.4. MSSQL

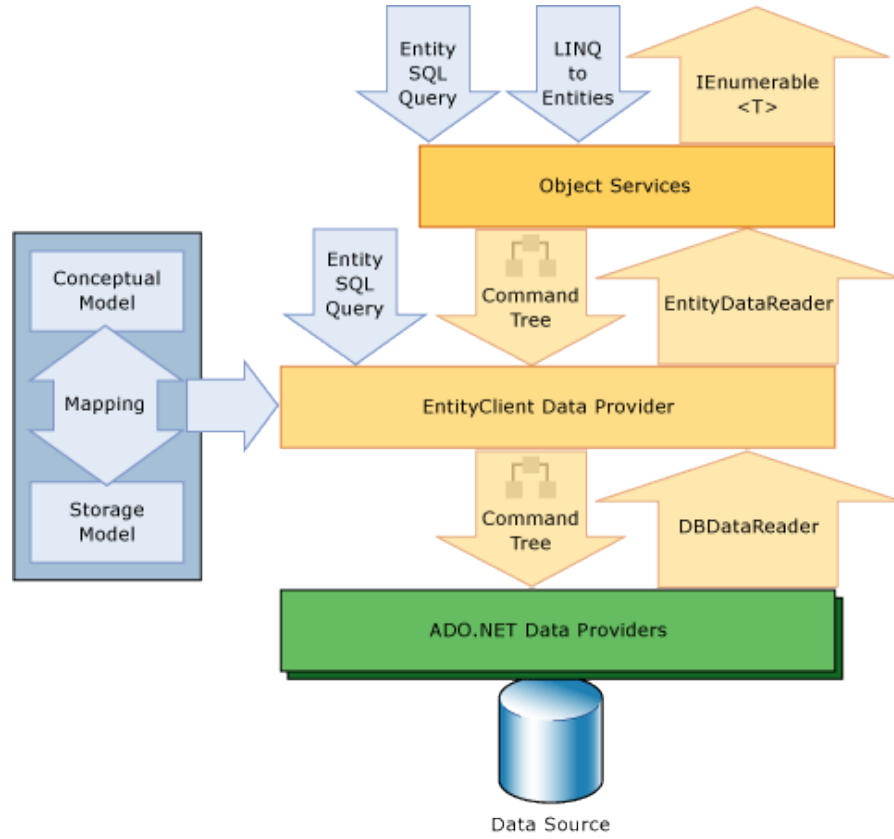
Microsoft SQL Server (MSSQL), Microsoft tarafından geliştirilen ve işletilen bir ilişkisel veri tabanı yönetim sistemidir (RDBMS). MSSQL, veri depolama, yönetim ve erişim işlemlerini etkin bir şekilde gerçekleştirmek için tasarlanmış bir yazılım sistemidir ve çeşitli veri tabanlarını oluşturarak, bu veriler üzerinde sorgulama, analiz ve raporlama işlemlerini gerçekleştirmek için kullanılır. MSSQL'in temel işlevlerinden biri, verilerin güvenli ve tutarlı bir şekilde saklanmasını sağlamaktır. Bu amaçla, veri bütünlüğünü koruyan ve veri erişimini kontrol eden çeşitli özellikler sunar. Örneğin, veri bütünlüğünü sağlamak için transaction management (işlem yönetimi), concurrency control (eşzamanlılık kontrolü) ve ACID (Atomicity, Consistency, Isolation, Durability) ilkelerini destekler. Bu özellikler, veri tabanı işlemlerinin güvenli ve hatasız bir şekilde gerçekleştirilmesini sağlar.

MSSQL, T-SQL (Transact-SQL) adı verilen ve SQL (Structured Query Language) standartlarına dayanan özel bir sorgulama dilini kullanır. T-SQL, veri sorgulama, veri manipülasyonu (INSERT, UPDATE, DELETE), veri tanımlama (CREATE, ALTER, DROP) ve veri kontrol (GRANT, REVOKE) işlemlerini gerçekleştirmek için ek fonksiyonlar ve prosedürler sunar. Bu dil, kullanıcıların veri tabanları ile etkileşim

kurmasını ve karmaşık sorgular ve işlemler gerçekleştirmesini sağlar. MSSQL, yüksek performanslı veri erişimi ve yönetimi için çeşitli optimizasyon teknikleri ve araçlar sunar. Bu optimizasyonlar, indeksleme, sorgu optimizasyonu, veri sıkıştırma ve bellek yönetimi gibi teknikleri içerir. Ayrıca, büyük veri kümeleri üzerinde hızlı ve verimli sorgulama yapmak için partitioning (bölümlendirme) ve clustering (kümeleme) gibi özellikler sunar. Bu özellikler büyük ve karmaşık veri tabanları ile çalışırken performansı artırır ve veri tabanı yönetimini kolaylaştırır. Güvenlik, MSSQL'in önemli özelliklerinden biridir. MSSQL veri güvenliğini sağlamak için çeşitli kimlik doğrulama yöntemleri, rol tabanlı erişim kontrolü ve veri şifreleme teknikleri sunar. SQL Server Audit ve Transparent Data Encryption (TDE) gibi özellikler veri tabanı üzerinde gerçekleşen işlemleri izleme ve veri hırsızlığına karşı koruma sağlama amacı taşır. MSSQL yedekleme ve kurtarma özellikleri ile de dikkat çeker. Veri tabanı yedeklemeleri, veri kaybı durumunda verilerin geri yüklenebilmesini sağlar. SQL Server tam yedekleme, fark yedekleme ve transaction log yedekleme gibi çeşitli yedekleme stratejileri sunar. Always On Availability Groups ve SQL Server Replication gibi özellikler, yüksek erişilebilirlik ve veri bütünlüğünü sağlamak için kullanılır. MSSQL kullanıcı dostu arayüzler ve yönetim araçları ile de bilinir. SQL Server Management Studio (SSMS) veri tabanı yöneticilerinin ve geliştiricilerin veri tabanlarını kolayca yönetmelerini, sorgular oluşturup çalıştırmalarını ve performans analizleri yapmalarını sağlar. Ayrıca SQL Server Data Tools (SSDT) gibi araçlar, veri tabanı geliştirme ve yönetimini destekler.

3.1.5. Entity Framework Core

Entity Framework Core (EF Core), Microsoft tarafından geliştirilen ve .NET platformu için sunulan modern bir nesne-ilişkisel haritalama (ORM) aracıdır. EF Core, geliştiricilerin ilişkisel veri tabanları ile çalışırken veri tabanı işlemlerini daha kolay ve daha az kod yazarak gerçekleştirmelerini sağlar. Bu araç veri tabanı nesneleri ile .NET sınıfları arasında bir köprü oluşturarak, veri erişim işlemlerini nesne yönelimli bir şekilde yapmayı mümkün kılar. EF Core, geliştiricilerin veri tabanı ile doğrudan SQL sorguları yazmak yerine, .NET sınıfları ve LINQ kullanarak veri tabanı işlemlerini gerçekleştirmelerini sağlar. Bu sayede veri erişim katmanı daha okunabilir, bakım yapılabilir ve hatalardan arındırılmış hale gelir. Şekil 4'te Entity Framework'ün çalışma yapısı gösterilmiştir.



Şekil 4. Entity Framework (Microsoft Learn, 2021)

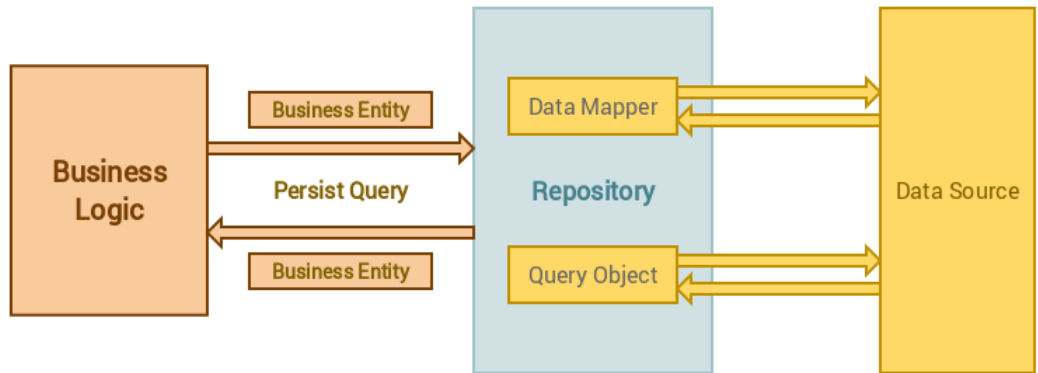
EF Core'un temel özelliklerinden biri Code First ve Database First yaklaşımlarını desteklemesidir. **Code First** yaklaşımında, geliştiriciler veri tabanı şemalarını .NET sınıfları üzerinden tanımlar ve bu sınıflardan yola çıkarak veri tabanı tabloları oluşturulur. Bu yaklaşım, hızlı prototip geliştirme ve test süreçleri için oldukça kullanışlıdır. **Database First** yaklaşımında ise, var olan bir veri tabanı şemasından yola çıkarak .NET sınıfları oluşturulur ve bu sınıflar üzerinden veri tabanı işlemleri gerçekleştirilir. Bu yöntem, mevcut veri tabanı ile entegrasyon gerektiren projeler için idealdir. EF Core, veri modeli değişikliklerini yönetmek için güçlü bir migration sistemi sunar. Migration'lar, veri modeli üzerinde yapılan değişiklikleri veri tabanına yansıtan adımlardır. Bu sistem, veri modeli değişikliklerini izlemeyi, versiyonlamayı ve gerektiğinde geri almayı kolaylaştırır. Migration'lar sayesinde, geliştiriciler veri tabanı şemasını kod üzerinden yönetebilir ve değişikliklerin veri tabanına güvenli bir şekilde uygulanmasını sağlayabilir.

Performans optimizasyonu, EF Core'un önemli özelliklerinden biridir. Lazy Loading (tembel yükleme), Eager Loading (hevesli yükleme) ve Explicit Loading (açık yükleme) gibi veri yükleme stratejileri, geliştiricilerin veri tabanı işlemlerini optimize etmelerine

olarak tanır. Ayrıca, sorgu performansını artırmak için çeşitli araçlar ve teknikler sunar, örneğin, sorgu derleme, nakil (caching) ve indeksleme gibi özellikler büyük veri kümeleri ile çalışırken performansı artırmak için kritik öneme sahiptir. EF Core, ilişkisel veri modellerini tanımlamak ve yönetmek için çeşitli ilişki türlerini destekler. Birçok-bire (one-to-many), bire-bir (one-to-one) ve birçok-birçok (many-to-many) ilişkiler, veri modellerinde tanımlanabilir ve EF Core, bu ilişkilerin doğru bir şekilde haritalanmasını ve yönetilmesini sağlar. İlişkisel modelleme, karmaşık veri yapılarının kolayca temsil edilmesini ve işlenmesini mümkün kılar. Ayrıca EF Core, platformlar arası uyumluluk ve esneklik sağlar. Hem masaüstü hem de web uygulamaları için uygun olan EF Core, .NET Core, .NET 5/6 ve .NET Framework ile uyumlu çalışır. Bu geniş uyumluluk, EF Core'un farklı türde uygulamalarda kullanılabilmesini ve çeşitli çalışma ortamlarında verimli bir şekilde çalışmasını sağlar.

3.1.6. Generic Repository

Generic Repository yazılım geliştirmede veri erişim katmanını daha modüler, yönetilebilir ve yeniden kullanılabilir hale getirmek için kullanılan bir tasarım desendir. Veri erişim işlemlerini soyutlamak ve her bir varlık türü için tekrar eden kod yazımını önlemek amacıyla kullanılır. Özellikle büyük ve karmaşık projelerde, veri erişim katmanını düzenlemek ve sürdürmek için oldukça yararlıdır. Temel amaç, veri tabanı işlemlerini gerçekleştiren genel bir yapı oluşturarak, belirli bir varlık türüyle ilgili tüm veri erişim kodlarını merkezi bir yerde toplamak ve bu kodları yeniden kullanılabilir hale getirmektir. Şekil 5'te Generic Repository'nin yapısı gösterilmiştir.



Şekil 5. Generic Repository (Yıldız, 2019)

Yapı iki aşamada kurulur birinci aşama veri erişim işlemlerini tanımlayan bir genel arayüz içerir. Arayüzde tipik olarak CRUD işlemlerini gerçekleştiren yöntemleri tanımlar. Arayüzdeki yöntemler, belirli bir varlık türüne (T) yönelik genel veri erişim işlemlerini kapsar. Arayüzün jenerik (generic) olması, farklı varlık türleriyle çalışmak için aynı arayüzün kullanılabilmesini sağlar. İkinci aşamada Generic arayüzü uygulayan genel bir sınıf, veri erişim işlemlerinin gerçek uygulamalarını sağlar. Bu sınıf, belirli bir veri bağlamı (context) ile çalışır ve arayüzde tanımlanan yöntemleri hayata geçirir. Genel sınıf, veri tabanı işlemlerini gerçekleştirmek için genellikle bir ORM aracı kullanır.

Generic Repository aşağıdaki gibi avantajlar sağlar;

Kod Tekrarının Azaltılması: Generic Repository deseni, veri erişim katmanındaki tekrar eden kodları ortadan kaldırarak, her varlık türü için ayrı ayrı veri erişim sınıfları yazma gereksinimini ortadan kaldırır. Bu, kod tabanını daha temiz ve yönetilebilir hale getirir.

Tip Güvenliği: Bu desen, veri erişim işlemlerinde tip güvenliğini sağlar. Hataların derleme zamanında yakalanmasına yardımcı olarak, çalışma zamanında karşılaşılan hataların azaltılmasına katkıda bulunur.

Bakım Kolaylığı: Veri erişim katmanındaki değişiklikler, tek bir genel sınıfta yapılabilir ve bu değişiklikler tüm varlık türleri için geçerli olur. Bu, bakım ve güncelleme süreçlerini basitleştirir.

Yeniden Kullanılabilirlik: Generic Repository, veri erişim işlemlerini merkezi bir yerde toplar ve bu işlemleri tüm varlık türleri için kullanılabilir hale getirir. Bu, kodun yeniden kullanılabilirliğini artırır.

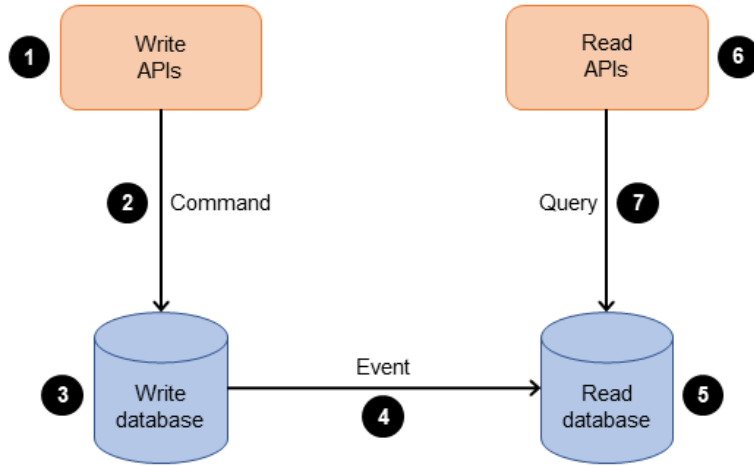
Test Edilebilirlik: Veri erişim işlemlerinin soyutlanması, bağımlılıkların enjeksiyonu ile birlikte kullanıldığında, birim testlerin daha kolay yapılabilmesini sağlar. Test sürecinde, gerçek veri tabanı yerine sahte veri erişim nesneleri kullanılabilir.

3.1.7. SQRS

CQRS yazılım mimarisinde kullanılan bir desendir ve sistemlerin veri işlemlerini daha etkili ve verimli hale getirmeyi amaçlar. CQRS bir sistemi iki ana bileşene ayırarak çalışır: komutlar (commands) ve sorgular (queries). Bu desen, komutlar ve sorguların farklı amaçlara hizmet ettiği ve farklı gereksinimlere sahip olduğu varsayımına dayanır. Komutlar, sistemdeki bir durumu değiştiren işlemler olarak tanımlanır. Veri tabanında veri eklemek, güncellemek veya silmek gibi eylemleri içerir. Komutlar, yazma işlemleri olarak da adlandırılabilir. CQRS, komutların uygulamanın iş mantığını içerdiği ve bu nedenle karmaşık doğrulama ve işleme kurallarına tabi olabileceği gerçeğini kabul eder.

Bir komut işlendiğinde, sistemin mevcut durumunda bir değişiklik meydana gelir. Sorgular ise sistemden veri okuma işlemlerini temsil eder bu durum veri tabanındaki bilgileri getirmek veya belirli kriterlere göre veri aramak anlamına gelir. Sorgular sadece veri okuma amaçlı olduğu için yan etkisizdir ve sistemin durumunu değiştirmez.

CQRS, sorguların yüksek performans gerektirdiğini ve genellikle hızlı ve verimli bir şekilde işlenmesi gerektiğini öngörür. Bu nedenle, sorguların iş mantığı içermemesi ve doğrudan veri okuma işlemlerine odaklanması sağlanır. CQRS deseni, sistemlerin bu iki bileşenini birbirinden ayırarak birçok avantaj sunar. Bu ayırım, her bir bileşenin kendi gereksinimlerine ve performans hedeflerine uygun şekilde optimize edilmesine olanak tanır. Şekil 6’da CQRS deseninin çalışma mantığı gösterilmiştir.



Şekil 6. SQRS (AWS, n.d.)

Komutlar ve sorgular farklı veri modelleri kullanabilir; bu, veri yazma ve okuma işlemlerinin birbirinden bağımsız olarak ölçeklenmesini ve optimize edilmesini sağlar.

Örneğin, yazma modeli normalize edilmiş veri tabanı tablolarını kullanırken, okuma modeli denormalize edilmiş veya önceden hesaplanmış görünüm (views) kullanabilir. CQRS, sistemlerin daha esnek ve modüler olmasını teşvik eder. Komutların ve sorguların ayrı bileşenler olarak ele alınması, her birinin bağımsız olarak geliştirilmesi, test edilmesi ve dağıtılmasına olanak tanır. Bu, özellikle büyük ve karmaşık sistemlerde, geliştirme sürecini hızlandırabilir ve sistemin bakımını kolaylaştırabilir. CQRS, aynı zamanda, Event Sourcing gibi diğer ileri seviye tasarım desenleri ile birlikte kullanılabilir. Event Sourcing, sistemin durumunu olaylar (events) dizisi olarak saklar ve her bir olay, sistemde meydana gelen bir durumu temsil eder. Bu, sistemin tüm geçmişini izlemenize ve gerektiğinde durumu yeniden oluşturmanıza olanak tanır. CQRS deseni, veri tutarlılığı ve bütünlüğü konularında da çeşitli stratejiler sunar. Geleneksel veri tutarlılığı modeline kıyasla, eventual consistency (nihai tutarlılık) gibi yaklaşımları destekleyerek, yüksek kullanılabilirlik ve performans gereksinimlerini karşılayabilir. Bu, özellikle dağıtık sistemlerde, kullanıcıların hızlı ve kesintisiz bir deneyim yaşamasını sağlar.

3.1.8. Microsoft Identity

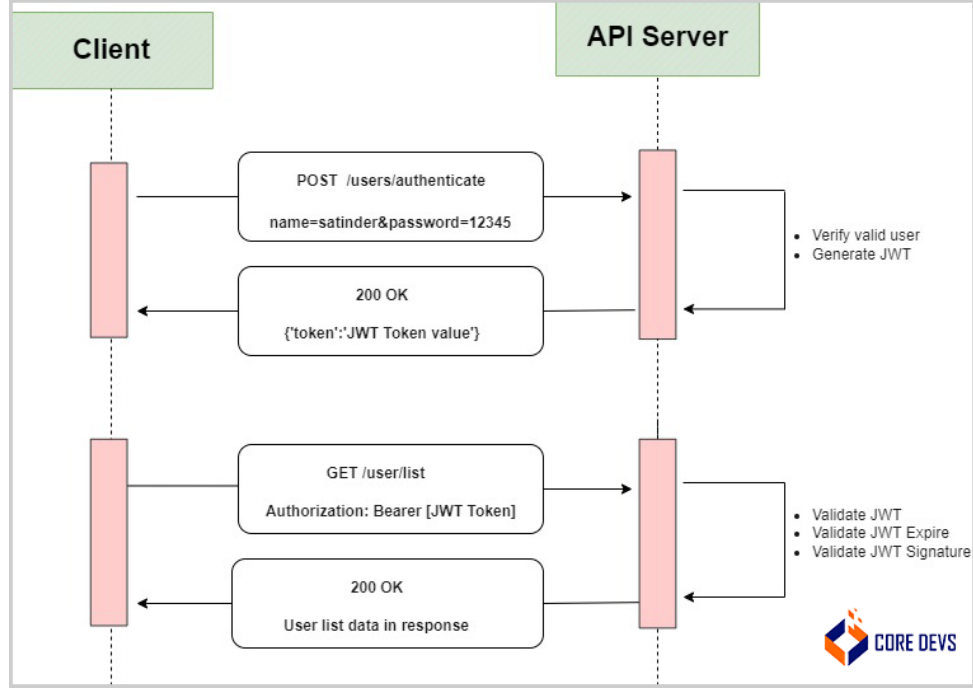
Microsoft Identity, ASP.NET Core uygulamaları için kimlik doğrulama ve yetkilendirme hizmetleri sunan bir kütüphanedir. Bu kütüphane, kullanıcı yönetimini, rol tabanlı erişim kontrolünü ve güvenli kimlik doğrulama işlemlerini kolaylaştırarak, geliştiricilerin kullanıcı hesapları ve güvenlik yönetimi ile ilgili zorlukları daha az kodla çözmelerini sağlar. Microsoft Identity kullanıcı kayıt ve giriş işlemlerini, parola yönetimini, e-posta doğrulamasını ve iki faktörlü kimlik doğrulamayı destekler ve kullanıcı kimlik bilgilerini ve ilgili verileri yönetmek için esnek ve genişletilebilir bir model sunar. Kullanıcılar ve roller sistemde kimlik doğrulama ve yetkilendirme işlemlerini gerçekleştirmek için kullanılan temel varlıklardır. Kullanıcılar benzersiz bir kimlik ile tanımlanır ve kullanıcı adları, parolalar, e-posta adresleri gibi bilgiler içerir. Roller ise, kullanıcıların belirli yetkilere sahip olmasını sağlayan grup ya da kategori yapılarıdır. Kullanıcı yönetimi, Microsoft Identity'nin temel özelliklerinden biridir. Geliştiriciler, kullanıcı kayıt işlemlerini kolayca ekleyebilir, kullanıcı hesaplarını etkinleştirebilir veya devre dışı bırakabilir ve kullanıcı bilgilerini güncelleyebilir. Parola yönetimi de bu kütüphanenin önemli bir parçasıdır. Microsoft Identity, güçlü parola politikalarını destekler ve kullanıcıların parolalarını güvenli bir şekilde değiştirmelerine olanak tanır. Ayrıca,

unutulan parolaların kurtarılması için gerekli mekanizmaları sağlar. Kimlik doğrulama işlemleri, Microsoft Identity ile güvenli ve esnek bir şekilde yönetilir. Kütüphane, hem geleneksel kullanıcı adı ve parola ile kimlik doğrulamayı hem de sosyal medya hesapları (örneğin, Google, Facebook) ve dış kimlik sağlayıcıları (örneğin, Microsoft, Azure AD) ile kimlik doğrulamayı destekler. Bu sayede, kullanıcılar çeşitli kimlik doğrulama yöntemleri ile sisteme giriş yapabilirler. Yetkilendirme işlemleri, kullanıcıların sistemde hangi kaynaklara erişebileceğini ve hangi işlemleri gerçekleştirebileceğini belirler. Microsoft Identity, rol tabanlı yetkilendirme ve politika tabanlı yetkilendirme modellerini destekler. Rol tabanlı yetkilendirme, kullanıcıları belirli rollerle ilişkilendirir ve bu rollere göre erişim yetkisi tanımlar. Politika tabanlı yetkilendirme ise, daha ince ayar yapılmış ve belirli koşullara dayanan erişim denetimleri sağlar. Bu, geliştiricilerin esnek ve güçlü erişim kontrolü uygulamalarına olanak tanır. Microsoft Identity, iki faktörlü kimlik doğrulama (2FA) ve hesap kitleme gibi gelişmiş güvenlik özelliklerini de içerir. İki faktörlü kimlik doğrulama, kullanıcının kimliğini doğrulamak için ek bir güvenlik katmanı sağlar. Bu, kullanıcıların bir parola dışında ek bir doğrulama yöntemi (örneğin, SMS kodu, e-posta kodu) kullanmalarını gerektirir. Hesap kitleme, belirli sayıda başarısız giriş denemesinden sonra kullanıcının hesabını kitleyerek, brute force saldırılarına karşı koruma sağlar. Microsoft Identity ayrıca, kullanıcıların kimlik bilgilerini ve diğer hassas verilerini güvenli bir şekilde saklamak için çeşitli veri koruma mekanizmaları kullanır. Veriler şifrelenmiş olarak saklanır ve yalnızca yetkili kişiler tarafından erişilebilir hale getirilir bu durum kullanıcı verilerinin güvenliğini ve gizliliğini korumak için önemlidir.

3.1.9. JWT Bearer

JWT (JSON Web Token) Bearer, modern web uygulamalarında güvenli ve stateless kimlik doğrulama sağlamak için yaygın olarak kullanılan bir mekanizmadır. JWT, kullanıcı kimlik bilgilerini ve diğer ilgili bilgileri JSON formatında kodlayan ve dijital olarak imzalanan bir token türüdür. Bu token, istemci ve sunucu arasında taşınarak, kullanıcının kimliğini doğrulamak ve yetkilendirme işlemlerini gerçekleştirmek için kullanılır. JWT Bearer kimlik doğrulama, uygulamaların stateless olmasını, yani sunucunun kullanıcı durumu hakkında bilgi saklamasını gerektirmeyen bir yapı sunar. Bu durum özellikle dağıtık sistemlerde ve mikro hizmet mimarilerinde, ölçeklenebilirlik ve

performans açısından büyük avantaj sağlar. JWT Bearer ile kimlik doğrulama, istemcinin her istekle birlikte taşıdığı bir token kullanılarak gerçekleştirilir. Bu token, sunucu tarafından doğrulanarak, kullanıcının kimliği ve yetkileri hakkında bilgi sağlar. Şekil 7’de JWT’nin çalışma sistemi gösterilmiştir.



Şekil 7. JWT (Fariha, 2023)

JWT Bearer ile kimlik doğrulama süreci, birkaç temel adımdan oluşur. İlk olarak, kullanıcı kimlik doğrulama bilgileri (genellikle kullanıcı adı ve parola) sunucuya gönderilir. Sunucu, bu bilgileri doğrular ve kullanıcının kimliğini onaylarsa, bir JWT oluşturur. Bu JWT, kullanıcının kimliğini, yetkilerini ve diğer gerekli bilgileri içeren yük (payload) kısmı ile birleştirilir ve bir imza ile dijital olarak imzalanır. İmza, token'ın değiştirilmediğini ve güvenilir bir kaynaktan geldiğini doğrulamak için kullanılır. JWT, genellikle üç parçadan oluşur: başlık (header), yük (payload) ve imza (signature). Başlık kısmı, token'ın türünü ve kullanılan imzalama algoritmasını belirtir. Yük kısmı, kullanıcının kimliği, yetkileri ve token'ın geçerlilik süresi gibi bilgileri içerir. İmza kısmı ise, başlık ve yük kısımlarının bir araya getirilip, belirli bir algoritma kullanılarak imzalanmasıyla oluşturulur. Bu yapı, token'ın güvenli ve değişmez olmasını sağlar. JWT oluşturulduktan sonra, istemciye gönderilir ve istemci bu token'ı her istekte Authorization başlığı altında "Bearer" ön eki ile birlikte sunucuya gönderir. Örneğin, "Authorization:

Bearer <token>" şeklinde. Sunucu, gelen her istekte bu token'ı alır ve doğrular. Doğrulama işlemi, token'ın imzasını kontrol etmeyi ve yük kısmındaki bilgileri incelemeyi içerir. İmza doğrulanırsa ve token geçerliyse, sunucu kullanıcının kimliğini ve yetkilerini onaylar. JWT Bearer kimlik doğrulama, çeşitli avantajlar sunar. Öncelikle, token tabanlı kimlik doğrulama mekanizması, sunucunun kullanıcı durumu hakkında bilgi saklamasını gerektirmediği için statelessdir ve bu, ölçeklenebilirlik açısından büyük bir avantaj sağlar. Ayrıca, JWT'ler taşınabilir ve platform bağımsızdır; bu da farklı platformlar arasında kolay entegrasyon sağlar. Token'ların JSON formatında olması, kullanıcı bilgilerini kolayca taşıyabilmeyi ve işlemeyi sağlar. JWT Bearer ayrıca, güvenlik açısından da önemli faydalar sağlar. Token'lar dijital olarak imzalandığı için, güvenilir bir kaynaktan geldiklerini ve değiştirilmediklerini doğrulamak mümkündür. Ek olarak, token'ların belirli bir geçerlilik süresi vardır ve bu süre dolduğunda token geçersiz hale gelir, bu da güvenliği artırır. Token'ların belirli bir süre için geçerli olması, uzun süreli oturumların güvenli bir şekilde yönetilmesini sağlar.

3.1.10. Refresh Token

Refresh Token, kimlik doğrulama sistemlerinde, özellikle JWT (JSON Web Token) tabanlı kimlik doğrulama senaryolarında kullanılan bir mekanizmadır. Refresh Token, erişim tokenlarının (access tokens) güvenliğini ve uzun ömürlülüğünü sağlamak amacıyla kullanılır. JWT gibi erişim tokenları belirli bir süre geçerli olduğundan bu süre dolduğunda kullanıcının tekrar kimlik doğrulaması yapması gerekir. Refresh Token kullanıcıya yeniden kimlik doğrulama yapmadan yeni bir erişim tokenı alabilme imkânı tanır bu durum kullanıcı deneyimini iyileştirir ve güvenliği artırır. Refresh Token genellikle daha uzun ömürlü ve daha az sıklıkla kullanılan bir token türüdür. Kullanıcı ilk kez kimlik doğrulaması yaptığında, sunucu hem bir erişim tokenı hem de bir refresh token oluşturur. Erişim tokenı kısa süreliğine geçerlidir ve kullanıcı isteklerinde sunucuya gönderilir. Refresh token ise, istemcide güvenli bir şekilde saklanır ve erişim tokenının süresi dolduğunda kullanılır.

Refresh Token kullanım süreci şöyle işler: Kullanıcı kimlik doğrulama bilgileriyle (örneğin kullanıcı adı ve parola) giriş yapar ve sunucu kullanıcıyı doğrular. Sunucu bu doğrulamanın ardından kullanıcıya bir erişim tokenı ve bir refresh token gönderir. Erişim

tokenı genellikle kısa bir süre (örneğin 15 dakika) geçerli olacak şekilde ayarlanır ve bu süre boyunca kullanıcı erişim tokenını kullanarak güvenli API istekleri yapabilir. Erişim tokenının süresi dolduğunda, kullanıcı yeniden kimlik doğrulama bilgilerini girmek zorunda kalmaz. Bunun yerine, istemci, sunucuya bir yenileme isteği (refresh request) gönderir ve bu istekte refresh tokenı kullanır. Sunucu refresh tokenını doğrular ve geçerli olduğunu onaylarsa kullanıcıya yeni bir erişim tokenı ve gerekirse yeni bir refresh token gönderir ve bu süre zarfında kullanıcının kesintisiz bir şekilde oturumunu sürdürmesini sağlayarak kullanıcı deneyimini iyileştirir. Şekil 8’de Refresh Token’ın çalışma mantığı gösterilmiştir.



Şekil 8. Refresh Token (Shukla, 2020)

Refresh Token kullanımı güvenlik açısından çeşitli avantajlar sunar. Öncelikle erişim tokenlarının kısa ömürlü olması, token çalındığında veya kötüye kullanıldığında potansiyel zararın minimize edilmesine yardımcı olur. Token çalınsa bile kısa sürede geçersiz hale geleceği için saldırganın kullanım süresi sınırlıdır. İkincisi refresh tokenları, daha uzun süre geçerli olmasına rağmen genellikle daha güvenli bir şekilde saklanır ve daha az sıklıkla kullanılır bu da saldırı yüzeyini azaltır. Refresh Token aynı zamanda, sistemin stateless doğasını koruyarak kullanıcının oturumunu yönetmeyi sağlar. Refresh tokenlarının kullanımı, sunucunun kullanıcı durumu hakkında bilgi saklamasını gerektirmez. Bu durum dağıtık sistemler ve mikro hizmet mimarilerinde büyük bir avantaj sağlar çünkü sunucular arasında durum senkronizasyonu yapmadan kullanıcı oturumları yönetilebilir. Refresh Token ile ilgili önemli bir güvenlik önlemi, refresh tokenlarının güvenli bir şekilde saklanması ve iletilmesidir. İstemci tarafında refresh

tokenları güvenli depolama yöntemleri kullanılarak saklanmalıdır. Ayrıca refresh tokenları yalnızca güvenli (HTTPS) bağlantılar üzerinden iletilmelidir. Sunucu tarafında ise refresh tokenlarının geçerliliği ve güvenliği düzenli olarak kontrol edilmeli ve gerektiğinde yenilenmelidir.

3.1.11. Serilog

Serilog, .NET uygulamaları için esnek ve genişletilebilir bir logging (kayıt tutma) kütüphanesidir. Geliştiricilerin loglama ihtiyaçlarını karşılamak üzere tasarlanmış olan Serilog, yapılandırılabilirliği ve genişletilebilirliği ile öne çıkar. Log verileri, JSON, XML, düz metin gibi çeşitli formatlarda kaydedilebilir ve dosyalar, veri tabanları, konsol çıkışları, bulut tabanlı log servisleri (örneğin, Azure Application Insights, Elasticsearch) gibi farklı hedeflere yönlendirilebilir. Bu esneklik, geliştiricilerin log verilerini toplamak ve analiz etmek için ihtiyaç duydukları araçları seçmelerine olanak tanır. Serilog, yapılandırma işlemlerini basit ve anlaşılır hale getirir. Geliştiriciler Serilog'u uygulamalarına kolayca entegre edebilir ve loglama davranışını kodla veya yapılandırma dosyalarıyla tanımlayabilirler.

Serilog, zengin log mesajları oluşturmayı ve log seviyelerini yönetmeyi destekler. Geliştiriciler log mesajlarına ek bilgi ekleyerek (kullanıcı kimliği, işlem süresi, hata kodları gibi) daha anlamlı loglar oluşturabilirler. Log seviyeleri, Verbose, Debug, Information, Warning, Error ve Fatal gibi çeşitli kategorilere ayrılır ve uygulamanın loglama gereksinimlerine göre ayarlanabilir bu durum gereksiz loglardan kaçınılarak performansın artırılmasını sağlar. Ayrıca, Serilog'un genişletilebilir yapısı, özel sink'ler ve formatlayıcılar oluşturarak loglama işlevselliğini genişletmeyi kolaylaştırır. Asenkron loglama desteği ile log yazma işlemleri arka planda gerçekleştirilir, böylece uygulamanın yanıt verme süresi iyileştirilir. Serilog, .NET uygulamalarının hata ayıklama, izleme ve bakım süreçlerini kolaylaştırarak yazılım geliştirme sürecini daha verimli hale getirir.

3.1.12. iTextSharp

iTextSharp .NET platformu için güçlü ve esnek bir PDF oluşturma ve manipülasyon kütüphanesidir. Geliştiricilere dinamik ve statik içerikleri içeren PDF belgeleri oluşturma, düzenleme ve okuma imkânı tanır. iTextSharp PDF belgelerini programatik olarak

oluşturmak ve yönetmek için geniş bir API seti sunar. Bu, raporlar, faturalar, etiketler ve diğer birçok türde belgeyi dinamik olarak oluşturmayı kolaylaştırır. Kütüphane metin, grafikler, tablolar, listeler ve daha fazlasını içeren zengin içerikleri destekler ve bu içeriklerin stil, boyut ve yerleşimini ayrıntılı bir şekilde kontrol etme imkânı verir.

iTextSharp PDF belgelerinin güvenliğini ve bütünlüğünü sağlamak için dijital imzalar, şifreleme ve izinler gibi özellikler sunar. Ayrıca, mevcut PDF belgelerini okuma, veri çıkarma ve düzenleme işlemlerini de destekler. Geliştiriciler PDF belgelerindeki belirli sayfaları, alanları veya öğeleri programatik olarak değiştirebilir, belgeye yeni içerik ekleyebilir veya mevcut içeriği güncelleyebilir. iTextSharp geniş dokümantasyon ve topluluk desteği ile birlikte gelir bu da geliştiricilerin ihtiyaç duydukları bilgiye ve çözümlere hızlıca ulaşmalarını sağlar. Bu kütüphane kurumsal düzeyde PDF oluşturma ve manipülasyon ihtiyaçlarını karşılamak için ideal bir çözümdür ve .NET uygulamalarına kolayca entegre edilebilir.

3.1.13. Angular

Angular, Google tarafından geliştirilmiş açık kaynaklı bir web uygulama çerçevesidir. Modern web uygulamaları geliştirmek için kullanılan Angular, özellikle tek sayfa uygulamalar (SPA) için güçlü bir yapı sunar. İlk olarak 2010 yılında AngularJS olarak tanıtılan çerçeve, 2016 yılında tamamen yeniden yazılarak Angular adıyla piyasaya sürüldü. Bu geçiş, çerçevenin mimarisinde köklü değişiklikler getirdi ve modern web standartlarıyla uyumlu hale getirdi. Angular, Typescript dilinde yazılmış olup, bu dilin sunduğu statik tip kontrolü ve gelişmiş IDE desteği gibi avantajları kullanır. Angular'ın modüler yapısı, geliştiricilere uygulamalarını daha iyi organize etme imkânı verir. Modüller, uygulamanın farklı bileşenlerini bir araya getirir ve yeniden kullanılabilir kod parçacıkları oluşturur. Angular bileşen tabanlı bir mimariyi benimser, bu da kullanıcı arayüzünün küçük, bağımsız ve yeniden kullanılabilir bileşenler halinde geliştirilmesine olanak tanır. Bileşenler HTML şablonları ve TypeScript kodu ile tanımlanır ve birbirleriyle veri bağlama (data binding) mekanizmaları aracılığıyla etkileşime girer.

Angular yönlendirme (routing) sistemi ile tek sayfa uygulamalarının (SPA) farklı görünümüler arasında gezinmesini sağlar. Bu, sayfa yenilemesi olmadan farklı bileşenlerin dinamik olarak yüklenmesine ve kullanıcı deneyiminin geliştirilmesine yardımcı olur.

Angular'ın forms modülü, hem şablon tabanlı (template-driven) hem de reaktif (reactive) formlar için güçlü destek sunar. Bu modül, form doğrulama, form verilerinin yönetimi ve kullanıcı girdilerinin işlenmesi gibi işlemleri kolaylaştırır. Angular'ın HTTPClient modülü, RESTful API'lere ve diğer HTTP tabanlı servislerle etkileşim kurmayı basitleştirir. Bu modül, HTTP isteklerini yapmayı, yanıtları almayı ve bu veriler üzerinde işlemler yapmayı kolaylaştırır. Ayrıca, Angular'ın dependency injection (bağımlılık enjeksiyonu) mekanizması, bileşenler ve servisler arasındaki bağımlılıkları yönetmeyi ve kodun test edilebilirliğini artırmayı sağlar.

Angular CLI Angular projelerini hızlı bir şekilde oluşturmak ve yönetmek için kullanılan güçlü bir araçtır. Bu CLI, yeni bileşenler, servisler ve modüller oluşturmayı, projeyi derlemeyi, test etmeyi ve dağıtmayı kolaylaştırır. Angular geniş bir ekosisteme sahiptir ve Angular Material gibi hazır bileşen kütüphaneleri, uygulama geliştirmeyi daha hızlı ve etkili hale getirir. Ayrıca Angular'ın sağladığı gelişmiş hata ayıklama araçları ve performans izleme mekanizmaları, uygulamaların kalitesini ve performansını artırmaya yardımcı olur. Angular güçlü tip kontrolü, genişletilebilir yapısı ve geniş ekosistemi ile modern web uygulamaları geliştirmek için kapsamlı bir çerçeve sunar. Geliştiricilere, karmaşık uygulamaları yönetilebilir ve sürdürülebilir bir şekilde oluşturma imkânı verirken, kullanıcılar için zengin ve etkileşimli deneyimler sağlar. Google ve geniş bir topluluk tarafından sürekli olarak güncellenen ve desteklenen Angular, web geliştirme alanında önemli bir yer tutar ve modern web uygulamalarının gereksinimlerini karşılamaya devam eder.

3.1.14. Bootstrap

Bootstrap Twitter tarafından geliştirilen ve şu anda açık kaynak topluluğu tarafından sürdürülen, en popüler HTML, CSS ve JavaScript çerçevelerinden biridir. 2011 yılında Mark Otto ve Jacob Thornton tarafından başlatılan Bootstrap başlangıçta Twitter'ın iç kullanımına yönelik olarak geliştirildi, ancak kısa sürede büyük bir kitleye ulaşarak web tasarım ve geliştirme alanında standart bir araç haline geldi. Bootstrap geliştiricilere ve tasarımcılara duyarlı (responsive) ve mobil uyumlu web siteleri oluşturma sürecini hızlandıran, kullanımı kolay ve geniş bir bileşen kütüphanesi sunar. Duyarlı tasarım, web sitelerinin ve uygulamaların farklı cihaz ekranlarında uyumlu bir şekilde

görüntülenmesini sağlar ve Bootstrap'ın ızgara sistemi (grid system) bu uyumu kolaylaştırır. Bu ızgara sistemi, sayfayı satır ve sütunlar halinde bölerek düzenlemeyi basit ve esnek hale getirir.

Bootstrap tipografi, düğmeler, formlar, navigasyon menüleri, uyarılar ve daha birçok bileşen için hazır CSS ve JavaScript çözümleri sunar. Bu bileşenler, tutarlı ve estetik açıdan hoş kullanıcı arayüzleri oluşturmayı kolaylaştırır. Örneğin düğmeler ve formlar gibi yaygın kullanılan UI bileşenleri, Bootstrap sayesinde standartlaştırılmış ve optimize edilmiştir, böylece geliştiriciler zaman kaybetmeden hızlıca çalışmaya başlayabilirler. Ayrıca, Bootstrap'ın bileşenleri, genişletilebilir ve özelleştirilebilir niteliktedir, bu da geliştiricilere kendi projelerinin ihtiyaçlarına göre bu bileşenleri uyarlama esnekliği verir. Bootstrap'ın JavaScript bileşenleri, jQuery tabanlıdır ve modallar, açılır menüler (dropdowns), açılır pencereler (popovers) ve kaydırıcılar (carousel) gibi dinamik ve etkileşimli UI öğelerini içerir. Bu bileşenler, web sayfalarına interaktivite eklemek için kullanılır ve kullanımı oldukça basittir.

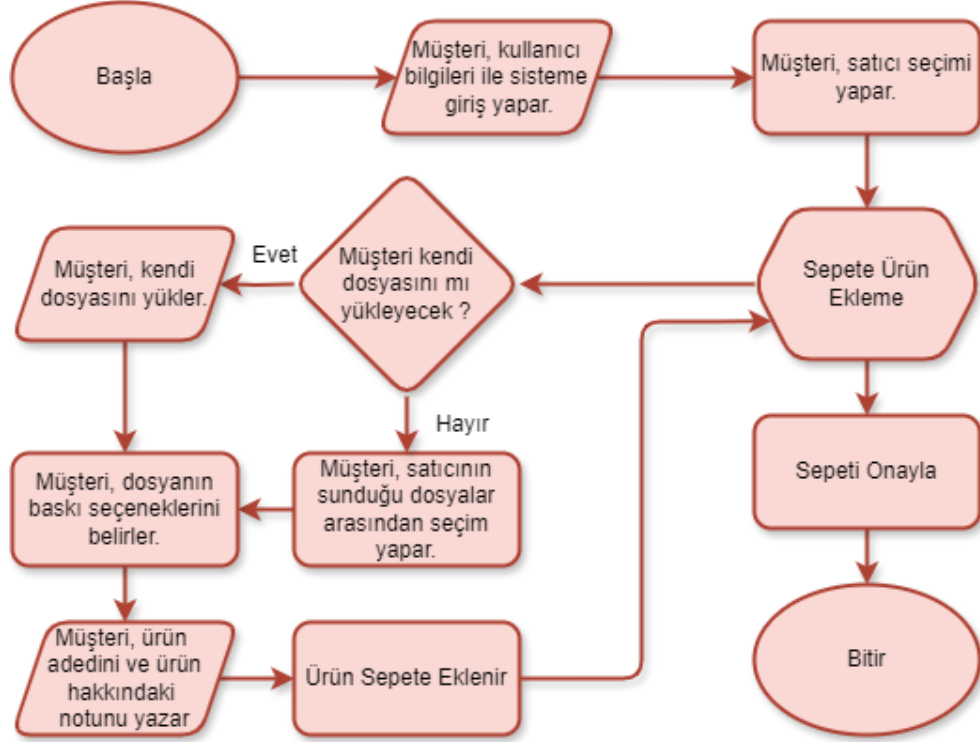
Bootstrap geliştiricilere kolayca özelleştirilebilen bir yapı sunar. Bootstrap'ın SASS (Syntactically Awesome Stylesheets) değişkenleri, geliştiricilerin renkler, arka planlar, kenar boşlukları ve diğer stil özelliklerini projeye özgü gereksinimlere göre hızlıca değiştirmesine olanak tanır. Bu özelleştirilebilir yapı Bootstrap'ın temel bileşenlerinden yararlanırken aynı zamanda marka kimliğine uygun tasarımlar oluşturmayı mümkün kılar. Bootstrap kapsamlı dokümantasyonu ve geniş topluluk desteği ile de dikkat çeker. Bootstrap'ın resmi dokümantasyonu tüm bileşenlerin, sınıfların ve JavaScript işlevlerinin nasıl kullanılacağını ayrıntılı bir şekilde açıklar.



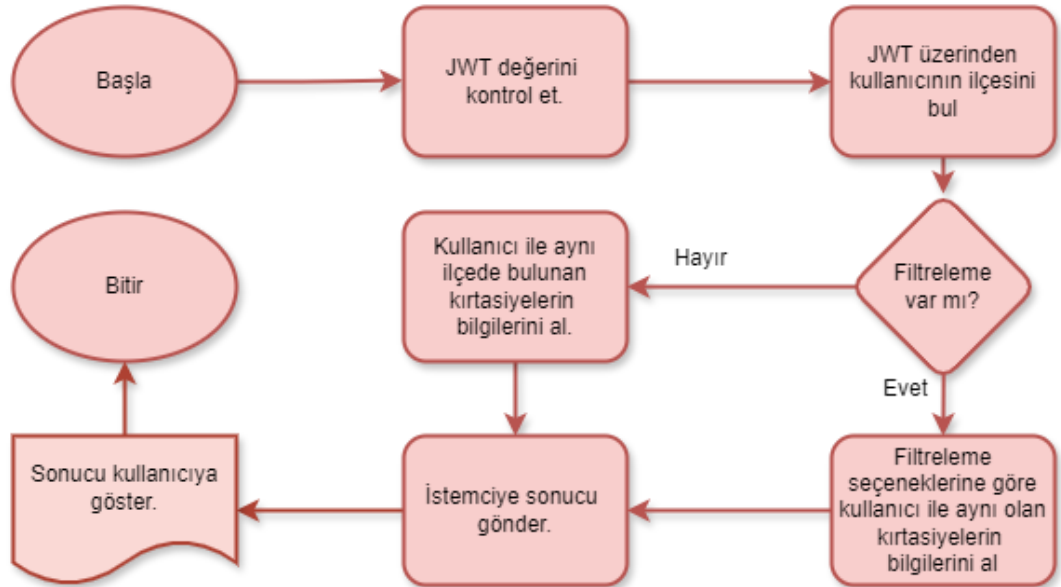
Şekil 9. Bootstrap (Bootstrap, 2024)

3.2. Yöntem

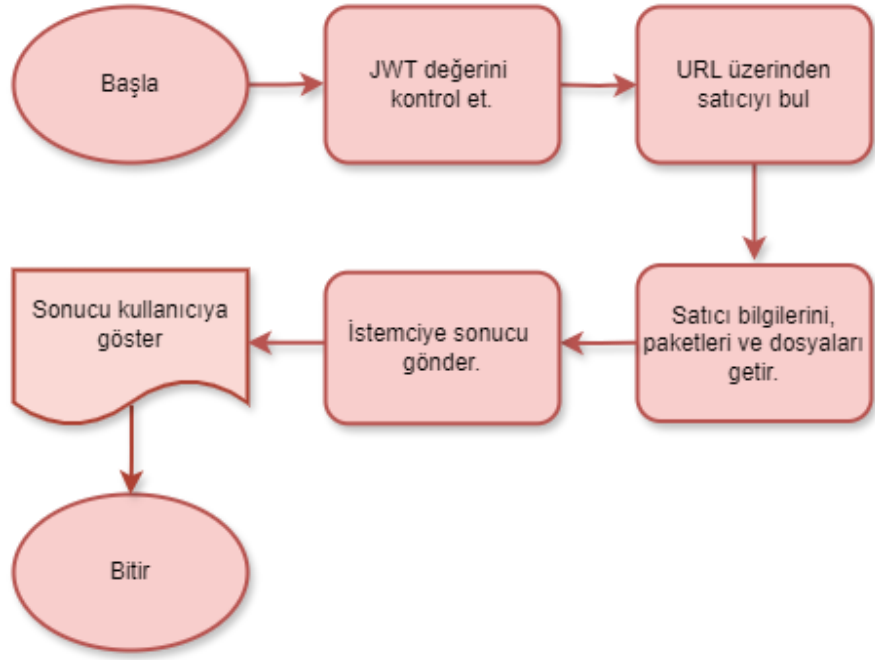
3.2.1. Sistem Tasarımı



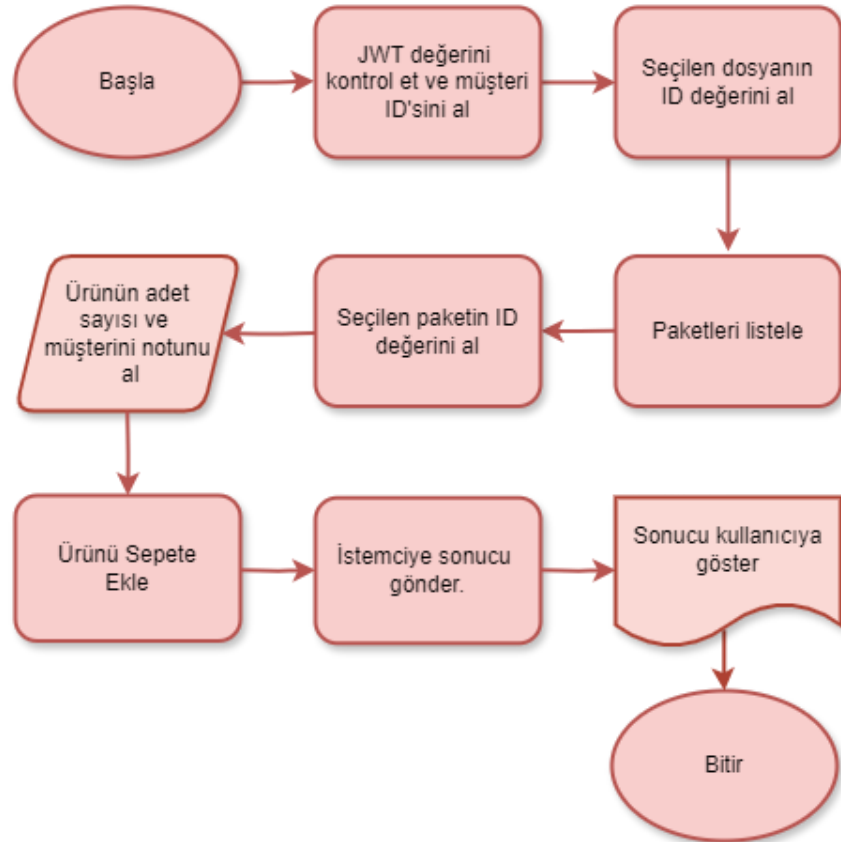
Şekil 10. Sipariş Oluşturma - Akış Şeması



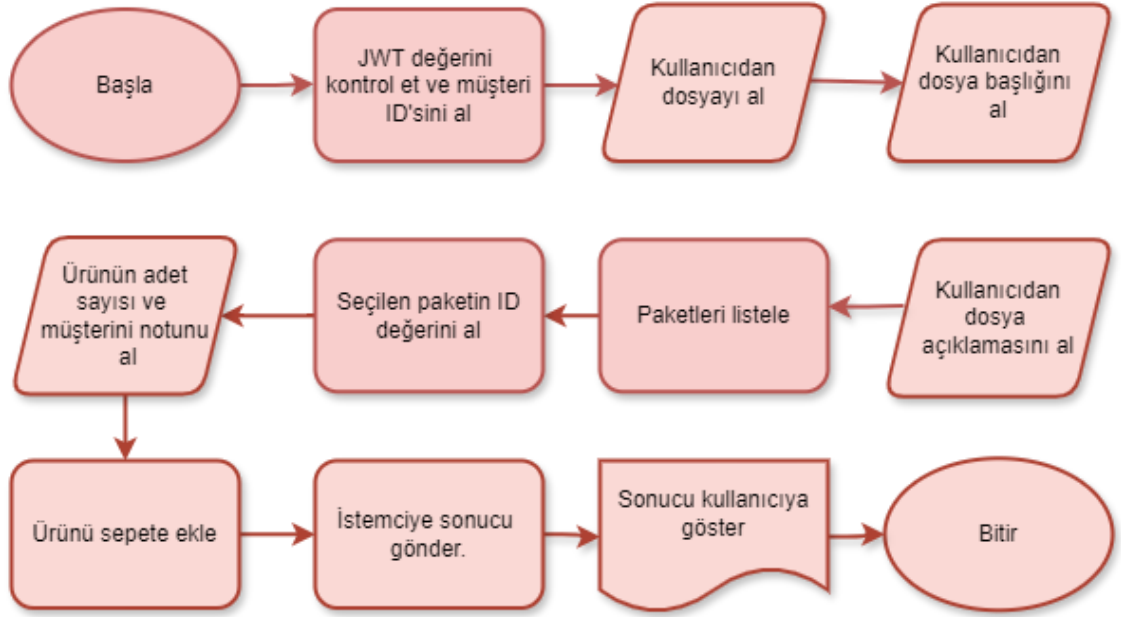
Şekil 11. Satıcıları Listeleme - Akış Şeması



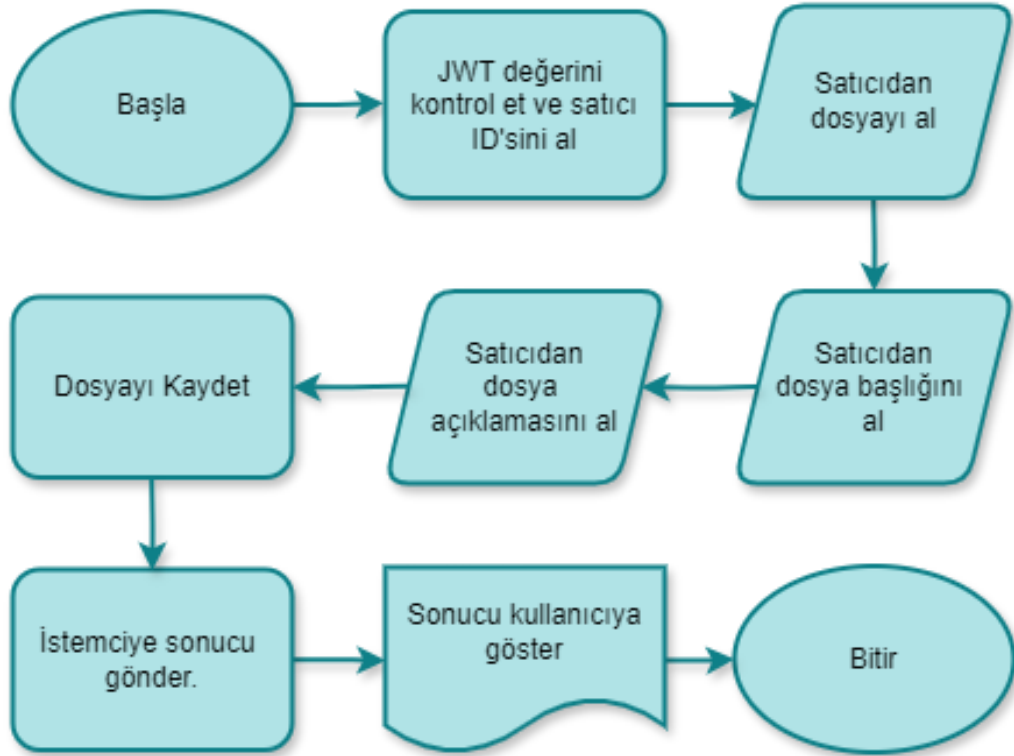
Şekil 12. Satıcı Bilgilerini Getirme - Akış Şeması



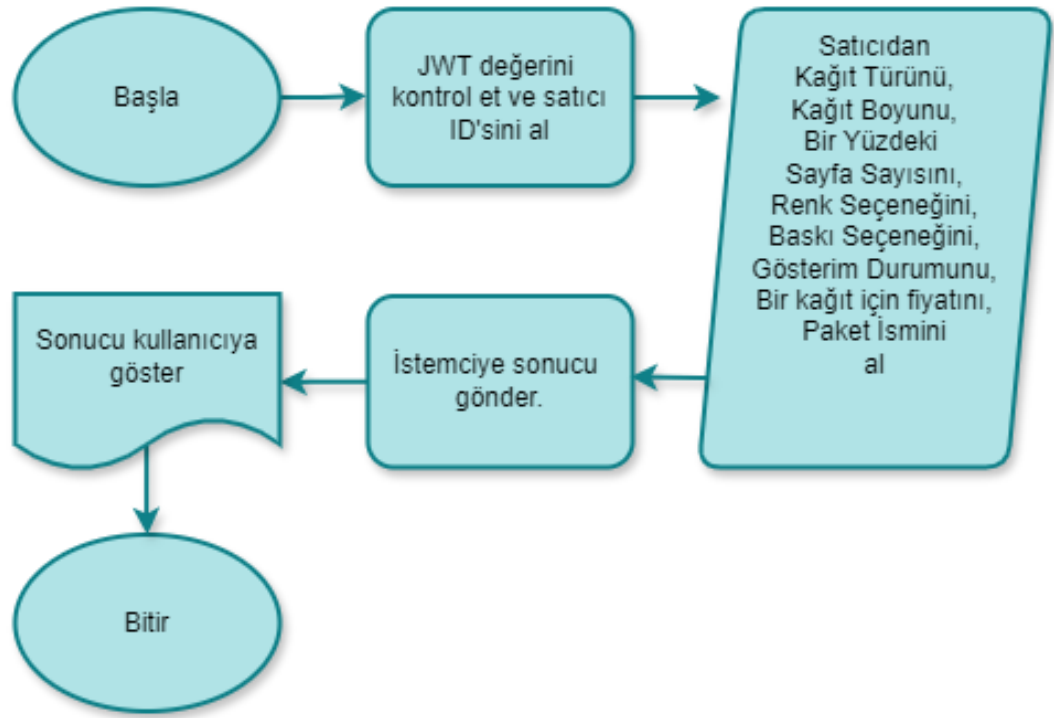
Şekil 13. Sepete Ürün Ekleme - Akış Şeması



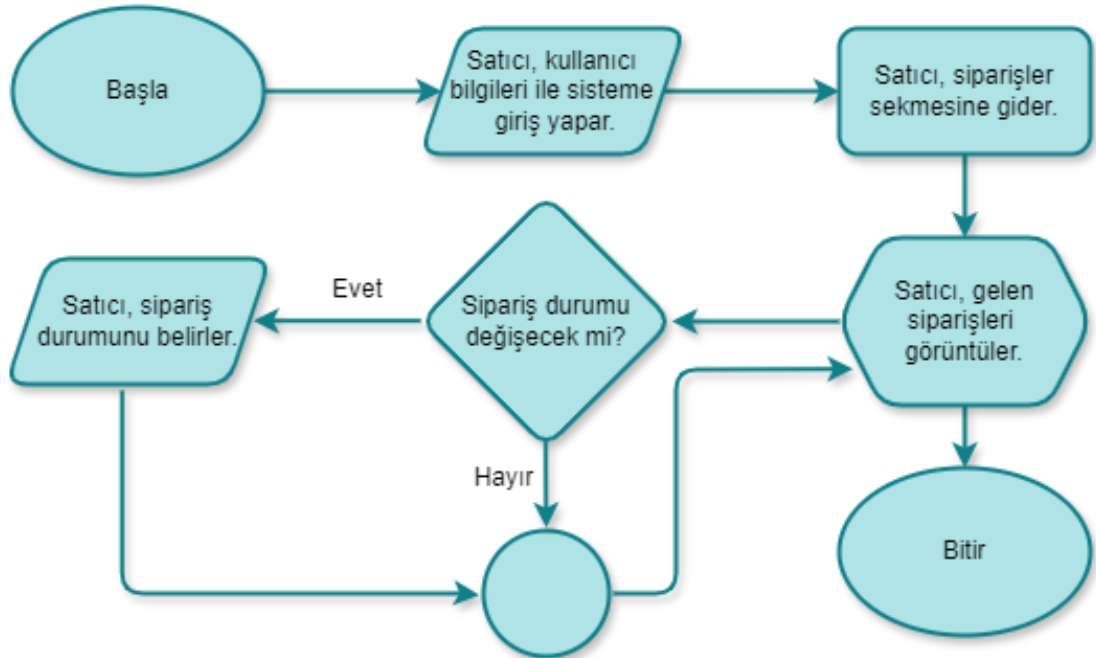
Şekil 14. Sepete Özel Ürün Ekleme - Akış Şeması



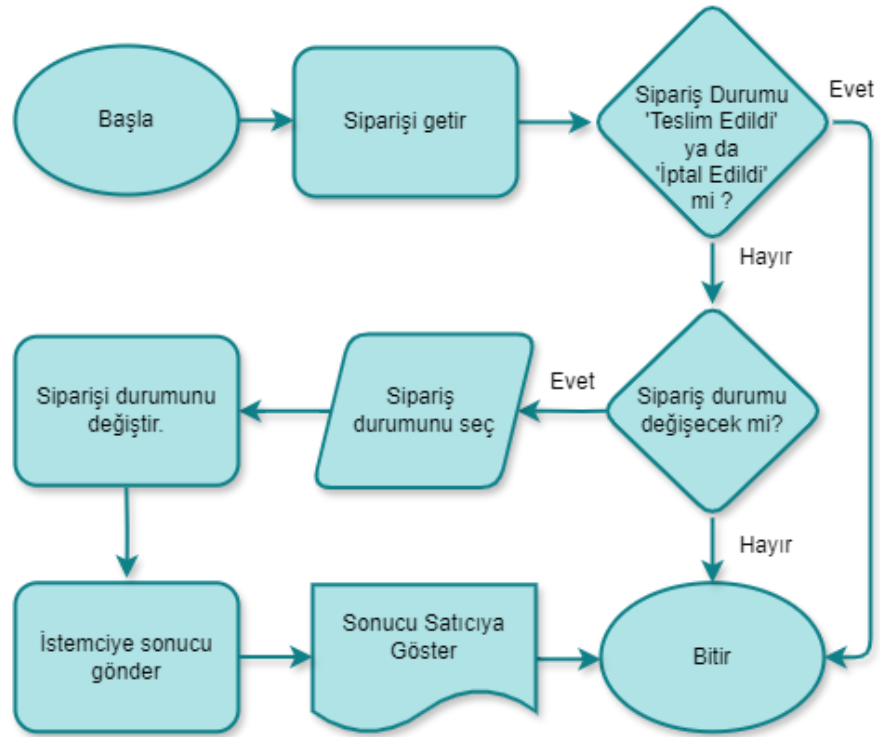
Şekil 15. Satıcı Dosya Ekleme - Akış Şeması



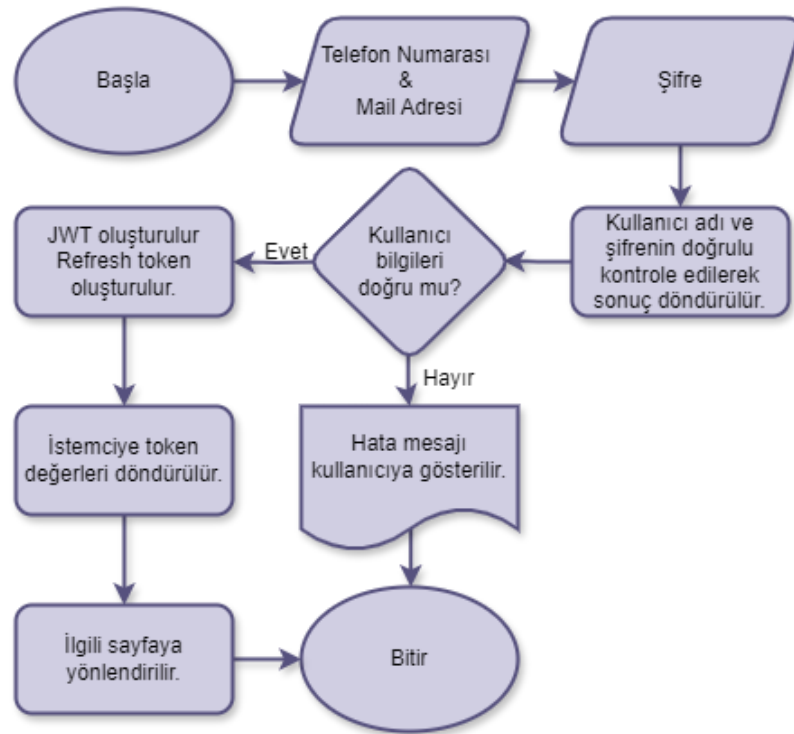
Şekil 16. Satıcı Paket Ekleme – Akış Şeması



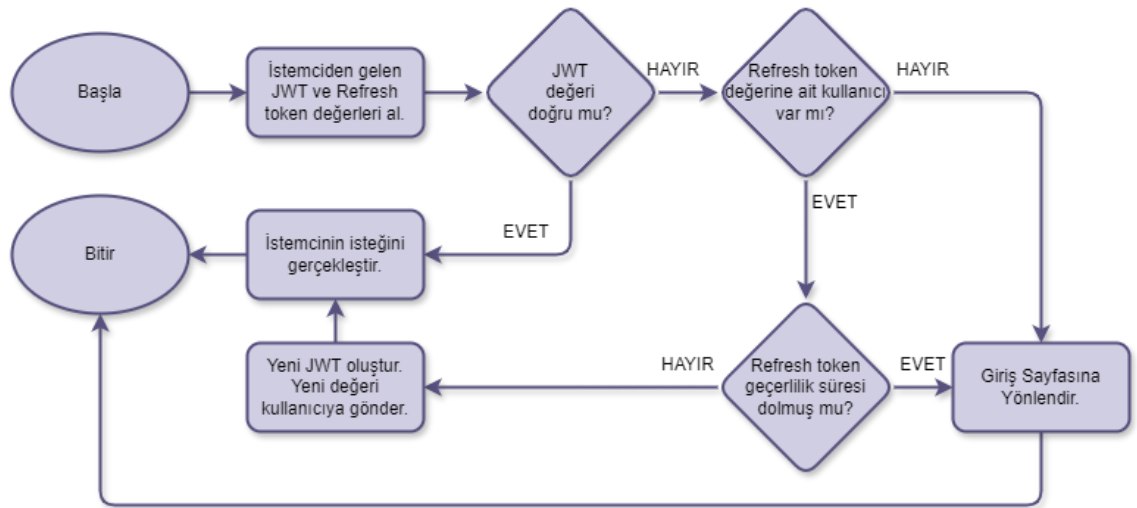
Şekil 17. Satıcının Siparişleri Görüntülemesi - Akış Şeması



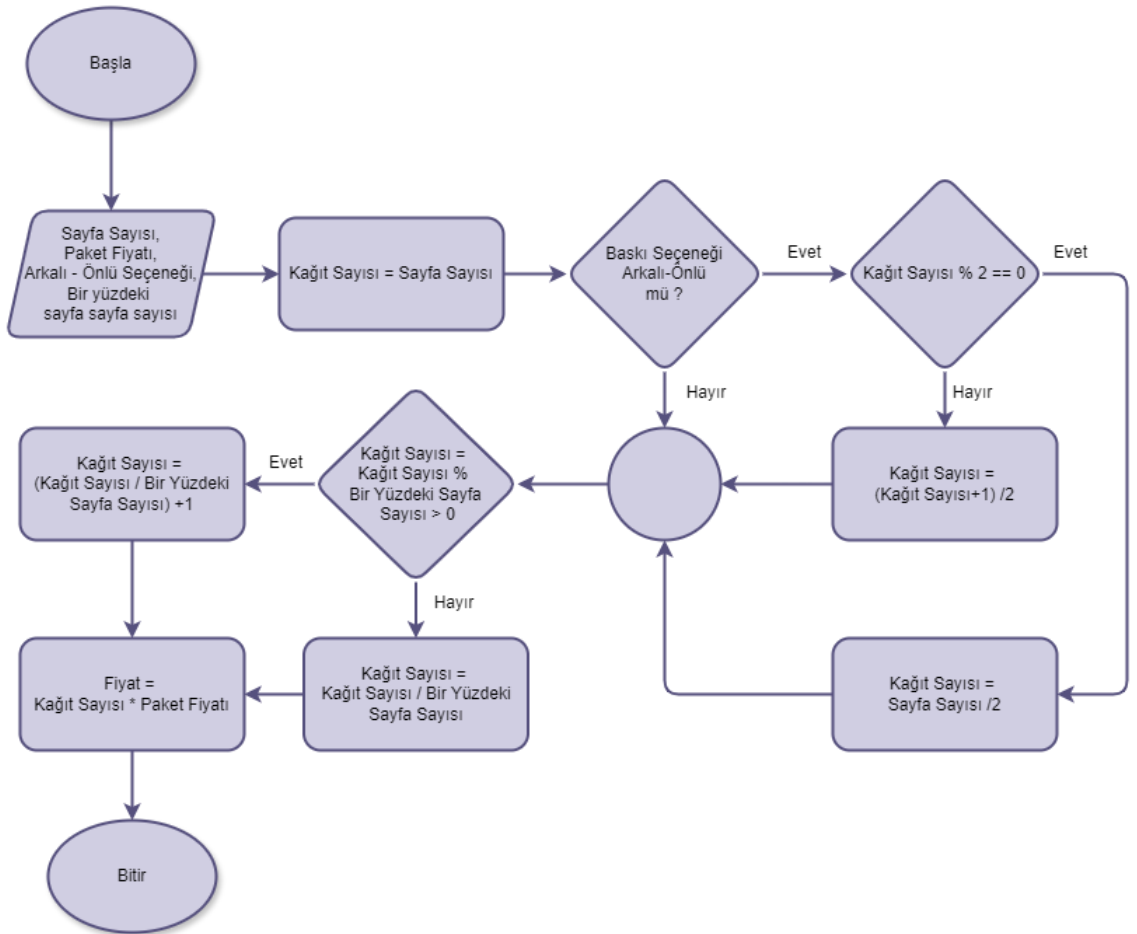
Şekil 18. Satıcı Sipariş Durumunu Değiştirme - Akış Şeması



Şekil 19. Kullanıcı Girişi - Akış Şeması



Şekil 20. Kullanıcı Doğrulama - Akış Şeması

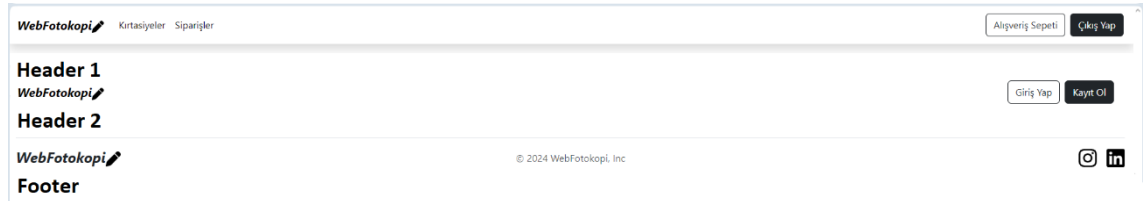


Şekil 21. Fiyat Belirleme Algoritması – Akış Şeması

3.2.2. Kullanıcı ve Sistem Arayüzü Tasarımları

3.2.2.1. Müşteri Arayüz Tasarımı

Şekil 22'de uygulamada kullanılan header ve footer tasarımları gösterilmektedir. Header1, kullanıcı giriş yaptığı durumlarda gösterilen menü bölümünü temsil etmektedir; buna karşın Header2, kullanıcı giriş yapmadan önce görüntülenen menü bölümüdür. Footer tasarımı ise her iki durumda da ortak olarak kullanılmaktadır.



Şekil 22. Müşteri Sistemi - Header ve Footer Tasarımı

Şekil 23'de müşteri sistemine girildiğinde kullanıcıya gösterilecek olan anasayfanın ilk kısmı gösterilmiştir. WebFotokopi logosunu kalem yazıyormuş gibi gösteren bir animasyon tasarımı yapılmıştır.



Şekil 23. Müşteri Sistemi - Anasayfa Tasarımı 1.Kısım

Şekil 24’de müşteri sistemine girildiğinde kullanıcıya gösterilecek olan anasayfanın ikinci kısmı gösterilmiştir.



Şekil 24. Müşteri Sistemi - Anasayfa Tasarımı 2.Kısım

Şekil 25’de müşterilerin kullanıcı girişi yapmaları için oluşturulan sayfa tasarımı verilmiştir.

WebFotokopi

Tekrardan Hoşgeldiniz

Mail Adresi
byrmgng@gmail.com

Şifre

☒ Beni Hatırla

[Giriş Yap](#)

[Üye Ol](#) [Şifremi Unuttum](#)

Diğer Giriş Seçenekleri

[G](#) [f](#) [t](#) [Apple](#)

Şekil 25. Müşteri Sistemi - Kullanıcı Girişi Tasarımı

Şekil 26'da müşterilerin kullanıcı kaydı yapabilmeleri için oluşturulan sayfanın tasarımı sunulmaktadır. Bu sayfada, kayıt işlemi esnasında verilerin doğru formatta girilmesini sağlamak amacıyla reactive form kullanılarak veri doğrulaması yapılmaktadır. Form alanlarından herhangi birinin boş bırakılması veya yanlış formatta veri girilmesi durumunda, Şekil 27'deki gibi kullanıcıya small etiketi ile uyarı mesajı gösterilmektedir. Veri format doğrulaması şu kriterlere dayanmaktadır:

- Ad: Boş bırakılamaz, en az 3 karakterden oluşmalıdır, en fazla 50 karakterden oluşmalıdır.
- Soyad: Boş bırakılamaz, en az 3 karakterden oluşmalıdır, en fazla 50 karakterden oluşmalıdır.
- İlçe: Boş bırakılamaz.
- Adres: Boş bırakılamaz, en az 20 karakterden oluşmalıdır, en fazla 200 karakterden oluşmalıdır.
- Telefon Numarası: Boş bırakılamaz, 10 karakterden oluşmalıdır. Sadece rakam girilebilir.
- Email: Boş bırakılamaz, email formatında olmalıdır, en az 5 karakterden oluşmalıdır, en fazla 50 karakterden oluşmalıdır.
- Şifre: Boş bırakılamaz, en az 8 karakterli olmalıdır, en fazla 30 karakter olmalıdır, en az bir özel karakter içermelidir, en az bir büyük-küçük harf olmalıdır.
- Şifre Tekrar: Şifre ile aynı olmalıdır.
- Kabul Ediyorum: Checkbox onaylanmalıdır.

Bu doğrulama kuralları, öncelikle frontend tarafında reactive formlar aracılığıyla kontrol edilmekte olup, kullanıcıların girdikleri verilerin uygunluğunu anında değerlendirmektedir. Ancak, sistemin güvenilirliğini ve veri bütünlüğünü sağlamak amacıyla, bu doğrulama işlemi backend tarafında da tekrarlanmaktadır. Backend doğrulaması, kullanıcıdan gelen verilerin sunucuya ulaşmadan önce yine aynı kurallara göre denetlenmesini ve yanlış formatlarda giriş yapılması durumunda, kullanıcının bilgilendirilmesini sağlamaktadır. Bu süreçte, kullanıcıya olası hataları ve eksiklikleri bildirmek için toastr mesajları kullanılmaktadır. Bu çift aşamalı doğrulama yaklaşımı hem istemci tarafında hem de sunucu tarafında veri bütünlüğünü ve güvenliğini en üst düzeye çıkararak, sistemin sağlıklı bir şekilde çalışmasını temin eder.

Kayıt Ol

Üyelğin var mı? [Giriş Yap](#)

Ad

Soyad

İl Seçiniz

İlçe Seçiniz

Adres

Telefon

Mail Adresi

Şifre

Şifre Tekrar

☐ **Kullanım koşullarını** kabul ediyorum.

Kayıt Ol

ya da

Google ile Üye Ol

Facebook ile Üye Ol

Apple ile Üye Ol

Şekil 26. Müşteri Sistemi - Kullanıcı Kayıt Formu Tasarımı

Kayıt Ol

Üyelğin var mı? [Giriş Yap](#)

Ad

Ad alanı boş bırakılamaz.

Soyad

Soyad alanı boş bırakılamaz.

İl Seçiniz

İlçe Seçiniz

Lütfen ilçenizi seçiniz.

Adres

Soyad alanı boş bırakılamaz.

Telefon

Telefon numarası boş bırakılamaz.

Mail Adresi

Email adresi boş bırakılamaz.

Şifre

Şifre boş bırakılamaz.

Şifre Tekrar

Şifre boş bırakılamaz.

☐ **Kullanım koşullarını** kabul ediyorum.
Lütfen kullanıcı sözleşmesini onaylayınız!

Kayıt Ol

ya da

Google ile Üye Ol

Facebook ile Üye Ol

Apple ile Üye Ol

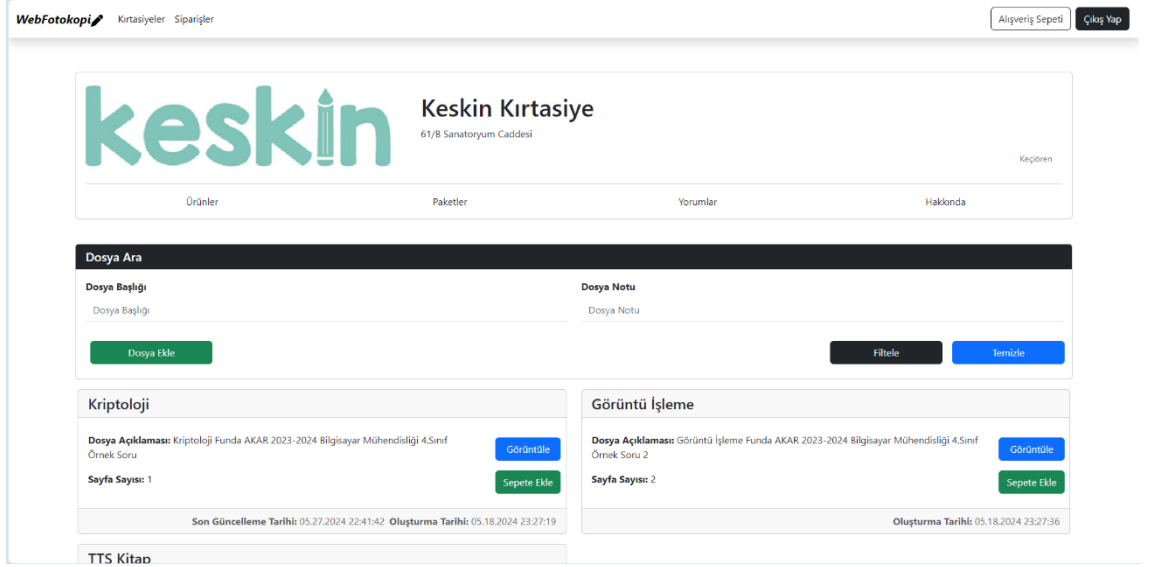
Şekil 27. Müşteri Sistemi - Kullanıcı Kayıt Formu Doğrulama Tasarımı

40

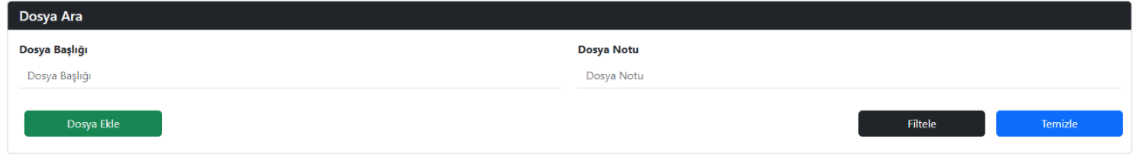
Şekil 28’de müşterilerin satıcıları seçebilmesi için oluşturulan sayfanın tasarımı verilmiştir. Sisteme kayıtlı olan kırtasiyeler dinamik olarak listelenmektedir. Müşteriler sisteme kayıt oldukları ilçelerde ki kırtasiyeleri görebilmektedir. Sistemdeki kırtasiyeleri isimlerine arayabilmek için bir filtreleme bölümü eklenmiştir. Filtreleme bölümündeki “Baskı Seçeneklerine Göre Ara” butonuna tıklayarak istedikleri baskı türünü sunan kırtasiyeleri de filtrelenebilmektedir.

Şekil 28. Müşteri Sistemi - Satıcı Seçimi Tasarımı

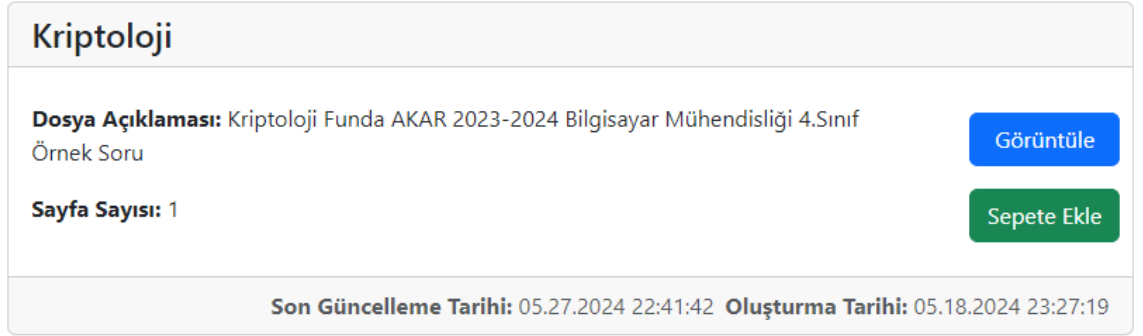
Şekil 29’da satıcıyı seçtikten sonra satıcıya ait sayfanın tasarımı bulunmaktadır. Satıcının logosu, ismi, adres bilgileri kart olarak gösterilmektedir. Alt tarafında ise bir menü bulunmaktadır. Bu menüde ise ürünler (Kırtasiyecinin sisteme eklemiş olduğu dosyalar), paketler (Kırtasiyenin baskı seçeneği olarak sunduğu özellikler ve fiyatları), yorumlar ve hakkında sekmeleri bulunmaktadır. Bu kartın altında Şekil 30’daki gibi dosya isimlerine ya da açıklamalarına göre filtreleme yapılabilecek bir bölüm bulunmaktadır. Bu sayede bir derse ya da bir hocaya ait not araması yapılmak istendiğinde bu dosyaya erişim sağlanabilecektir. Ek olarak bu bölümde “Dosya Ekle” seçeneği oluşturulmuştur. Bu sayede kullanıcı kendi dosyalarını sisteme yükleyebilecektir. Şekil 31’de Satıcıların sisteme yükledikleri dosyaların listelenmesi için dinamik bir container oluşturulmuştur. “Görüntüle” butonuna tıklanılarak dosyalar görüntülenebilir, “Sepete Ekle” butonuna basılarak ise sepete eklenebilir durumdadır. Container’ın alt bölümünde ise dosyanın sisteme eklenme tarihi ve varsa güncelleme tarihi görüntülenmektedir.



Şekil 29. Müşteri Sistemi - Satıcı Sayfası Tasarımı



Şekil 30. Müşteri Sistemi - Dosya Filtreleme Menü Tasarımı



Şekil 31. Müşteri Sistemi – Satıcı Dosyalarının Görüntülenmesi İçin Tasarım

Şekil 32’de paketler butonuna tıklanıldığında satıya ait olan paketlerin listelenmesi görüntülenmektedir. İstenilen özelliklere ait paketin olup olmadığını kontrol edebilmek için filtreleme bölümü bulunmaktadır. Bu sayede satıcının müşterilere sunduğu baskı seçenekleri ve fiyatları sepete ürün eklenmeden önce görülebilmektedir.

keskin

Keskin Kırtasiye

61/B Sanatoryum Caddesi

Keçören

Ürünler

Paketler

Yorumlar

Hakkında

Paket seçenekleri

Paket Adı	Kağıt Boyu	Kağıt Türü	Bir yüzdeki sayfa sayısı	Renk Seçeneği	Baskı Seçeneği
Paket Adı	Tümü	Tümü	Tümü	Tümü	Tümü

Filtre Temizle

Paket Adı : Renkli A5 Paketi

Kağıt Boyu	Kağıt Türü	Bir yüzdeki sayfa sayısı	Renk Seçeneği	Baskı Seçeneği	Fiyat
A5	120GR(Kalın Kağıt)	1	Renkli	Arkalı-Önlü	10 TL

Paket Adı : Standart Paket

Kağıt Boyu	Kağıt Türü	Bir yüzdeki sayfa sayısı	Renk Seçeneği	Baskı Seçeneği	Fiyat
A4 (Standart)	80GR (Standart)	1	Siyah-Beyaz	Arkalı-Önlü	1.1 TL

Paket Adı : Bir Yüzde 4 Tane (Arkalı Önlü 8 tane) Siyah Beyaz Arkalı Önlü

Şekil 32. Müşteri Sistemi - Satıcı Paketlerinin Görüntülenmesi İçin Tasarım

Şekil 33'te Kullanıcıların dosyaları satıcıya iletebilmesi için oluşturulan modal'ın tasarımı bulunmaktadır. Bu modal da dosya başlığı, dosya için düşülen not girilmelidir ve dosya seçilmelidir. “İleri” butonuna tıklanıldığında paket seçimi modalına yönlendirilmekte ve “İptal Et” butonuna tıklanıldığında modal temizlenerek kapatılmaktadır.

keskin

Ürünler

Paketler

Yorumlar

Hakkında

Dosya Ekle

Dosya Başlığı:

Bulut Bilgişim Ödevi

✓

Dosya Notu:

191104018 Bayram GÜNGÖR Bulut Bilgişim Vize Ödevi

✓

Dosya:

Dosya Seç

bayramgungor_19110018_odev1.pdf

İptal Et

İleri

Dosya Ara

Dosya Başlığı:

Dosya Başlığı

Dosya Ekle

Dosya Notu:

Dosya Notu

Filtre

Temizle

Kriptoloji

Dosya Açıklaması: Kriptoloji Funda AKAR 2023-2024 Bilgisayar Mühendisliği 4.Sınıf Örnek Soru

Görüntüle

Sayfa Sayısı: 1

Sepete Ekle

Son Güncelleme Tarihi: 05.27.2024 22:41:42 Oluşturma Tarihi: 05.18.2024 23:27:19

TTS Kitap

Dosya Açıklaması: Text-to-Speech (TTS) Teknolojisi: Gelişimi, Uygulamaları ve Geleceği Dr. Öğr. Üyesi Funda AKAR, Bayram GÜNGÖR

Görüntüle

Sayfa Sayısı: 2

Sepete Ekle

Oluşturma Tarihi: 05.18.2024 23:27:36

Şekil 33. Müşteri Sistemi - Müşteri Dosya Yükleme İşlemi Modal Tasarımı

Şekil 31’deki satıcı dosyalarının listelendiği bölümde “Görüntüle” butonuna tıklanıldığında Şekil 34’deki modal açılarak dosya içeriği görüntülenmektedir. Modal’ın sol tarafında PDF dosyasının içeriği filigranlı olarak görüntülenmektedir. Bu sayede satıcı dosyaları başka bir satıcı tarafından kullanımı engellenir. Modal’ın sağ tarafında ise dosya başlığı ve dosya açıklaması bulunmaktadır. Modal’ın alt tarafında ise “Geri Dön” ve “Sepete Ekle” butonları bulunmaktadır.

Şekil 34. Müşteri Sistemi - Satıcı Dosyalarını Görüntüleme Modal Tasarımı

Şekil 34'te ve Şekil 31'de gösterilen "Sepete Ekle" butonuna tıklanıldığında ve Şekil 33'teki dosya gönderme işlemi sırasında yer alan "İleri" butonuna tıklanıldığında, kullanıcılar, Şekil 35'te gösterilen ürüne ait paket seçim modül ekranını açmaktadır. Bu modal ekran, kullanıcıların istedikleri baskı seçeneklerine göre paketleri filtreleyebilmeleri için tasarlanmış bir bölüm içermektedir. Filtreleme işleminin ardından, kullanıcıların uygun paketi seçebilmeleri için "İleri" butonu bulunmaktadır. "İleri" butonuna tıklanıldığında, kullanıcılar, Şekil 36'da bulunan modal ekranına yönlendirilir ve burada ürünün adeti ile ürüne ilişkin satıcıya iletilmesi istenen not eklenebilmektedir.

Bu aşamada, "Sepete Ekle" butonuna tıklanıldığında, seçilen ürün kullanıcı alışveriş sepetine eklenir, bu da kullanıcının alışveriş sürecine devam etmesini sağlar. Alternatif olarak, "İptal Et" butonuna tıklanıldığında, yapılan tüm işlemler iptal edilir. Bu iptal işlemi, dosya gönderme süreçlerini de kapsayacak şekilde, kullanıcının o anki işlemlerinin tümünü geri alır ve kullanıcıyı başlangıç durumuna geri döndürür.

Paket Seç

Paket seçenekleri

Paket Adı	Kağıt Boyu	Kağıt Türü	Bir yüzdeki sayfa sayısı	Renk Seçeneği	Baskı Seçeneği	
Paket Adı	Tümü	Tümü	Tümü	Tümü	Tümü	<div>Filtre</div> <div>Temizle</div>

Paket Adı : Renkli A5 Paketi

Kağıt Boyu	Kağıt Türü	Bir yüzdeki sayfa sayısı	Renk Seçeneği	Baskı Seçeneği	Kağıt Başına Fiyat	İleri
A5	120GR(Kalın Kağıt)	1	Renkli	Arkalı-Önlü	10 TL	

Paket Adı : Standart Paket

Kağıt Boyu	Kağıt Türü	Bir yüzdeki sayfa sayısı	Renk Seçeneği	Baskı Seçeneği	Kağıt Başına Fiyat	İleri
A4 (Standart)	80GR (Standart)	1	Siyah-Beyaz	Arkalı-Önlü	1.1 TL	

Paket Adı : Bir Yüzde 4 Tane (Arkalı Önlü 8 tane) Siyah Beyaz Arkalı Önlü

Kağıt Boyu	Kağıt Türü	Bir yüzdeki sayfa sayısı	Renk Seçeneği	Baskı Seçeneği	Kağıt Başına Fiyat	İleri
A4 (Standart)	80GR (Standart)	4	Siyah-Beyaz	Arkalı-Önlü	1.5 TL	

Paket Adı : Renkli - Tek Yüz Fotokopi (Standart Kağıt)

Kağıt Boyu	Kağıt Türü	Bir yüzdeki sayfa sayısı	Renk Seçeneği	Baskı Seçeneği	Kağıt Başına Fiyat	İleri

Şekil 35. Müşteri Sistemi Ürün Paket Seçimi Modal Tasarımı

Dosya Ara

Dosya Başlığı

Dosya Başlığı

Dosya Ekle

Ürün Ekle

Kopya Sayısı:

3

Ürün Notu:

Lütfen dosyalar silik çıkmasın

İptal Et

Sepete Ekle

Kriptoloji

Dosya Açıklaması: Kriptoloji Funda AKAR 2023-2024 Bilgisayar Mühendisliği 4.Sınıf Örnek Soru

Görüntüle

Sayfa Sayısı: 1

Sepete Ekle

Son Güncelleme Tarihi: 05.27.2024 22:41:42 Oluşturma Tarihi: 05.18.2024 23:27:19

Görüntü İşleme

Dosya Açıklaması: Görüntü İşleme Funda AKAR 2023-2024 Bilgisayar Mühendisliği 4.Sınıf Örnek Soru 2

Görüntüle

Sayfa Sayısı: 2

Sepete Ekle

Oluşturma Tarihi: 05.18.2024 23:27:36

TTS Kitap

Dosya Açıklaması: Text-to-Speech (TTS) Teknolojisi: Gelişimi, Uygulamaları ve Geleceği Dr. Öğr. Üyesi Funda AKAR, Bayram GÖNGÖR

Görüntüle

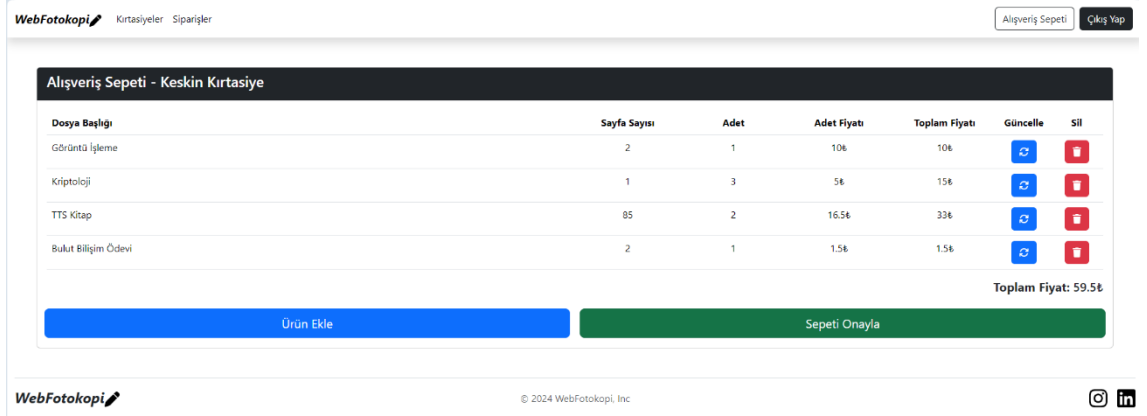
Sayfa Sayısı: 85

Sepete Ekle

Son Güncelleme Tarihi: 05.18.2024 23:28:35 Oluşturma Tarihi: 05.18.2024 23:27:54

Şekil 36. Müşteri Sistemi - Ürünün Sepete Eklenmesi Modal Tasarımı

Satıcıda bulunan dosyalar ya da eklenmek istenen dosyalar ürüne dönüştürüldükten sonra header üzerinde bulunan “Alışveriş Sepeti” butonuna tıklanılarak eklenen ürünler ve fiyatları görüntülenmektedir. Fiyatlar dosyaların sayfa sayısı ve seçilen pakete göre Şekil 21’ de bulunan algoritma ile hesaplanmaktadır. Şekil 37’de Alışveriş sepeti tasarımı bulunmaktadır. Alışveriş sepetinde ürünlerin dosya başlıkları, sayfa sayıları, adetleri ve fiyatları görüntülenmektedir. Her ürün için “Güncelle” ve “Sil” butonları oluşturularak Alışveriş Sepeti düzenlenebilmektedir. “Sepeti Onayla” butonuna tıklanıldığında sanal pos olmadığı için direkt olarak sipariş oluşacaktır.



Şekil 37. Müşteri Sistemi - Alışveriş Sepeti Tasarımı

Sepet onaylandıktan sonra, kullanıcılar, header'daki "Siparişler" sekmesi aracılığıyla verdikleri siparişlerin durumlarını takip edebilmektedir. Şekil 38'de gösterilen siparişler sayfasının tasarımında, siparişlerin teslimat durumları ayrıntılı bir şekilde izlenebilmektedir. Başlangıçta "Onay Bekleniyor" olarak belirlenen teslimat durumu, satıcı tarafından "Hazırlanıyor", "Hazır", "Teslim Edildi" ve "İptal Edildi" gibi durumlara güncellenebilmektedir. Her bir siparişin içerdiği ürünler ayrı ayrı görüntülenebilmekte ve kullanıcıya detaylı bilgi sağlanmaktadır. Teslimat durumu "Teslim Edildi" olduğunda, ilgili siparişin rengi yeşile dönmekte ve bu durumda teslimat tarihi de görüntülenebilmektedir. Benzer şekilde, teslimat durumu "İptal Edildi" olarak güncellendiğinde, siparişin rengi kırmızıya dönmektedir. Bu sistem, kullanıcıların sipariş süreçlerini daha etkili bir şekilde yönetmelerine ve takip etmelerine olanak tanımaktadır.

Kırtasiye: Uğur Kırtasiye		Teslimat Durumu: İptal Edildi
TTS Kitap		▼
Toplam Ücret: 1€		Sipariş Tarihi: 05.28.2024 00:56:11
Kırtasiye: Keskin Kırtasiye		Teslimat Durumu: Onay Bekliyor
Görüntü İşleme		^
Dosya Özellikleri Dosya Başlığı: Görüntü İşleme Dosya Notu: Görüntü İşleme Funda AKAR 2023-2024 Bilgisayar Mühendisliği 4.Sınıf Örnek Soru 2 Sayfa Sayısı: 2 Adet: 1		Yazdırma Özellikleri Renk Seçeneği: Renkli Baskı Seçeneği: Arkalı-Önlü Kağıt Boyu: A5 Kağıt Türü: 120GR(Kalın Kağıt) Bir Yüzdeki Sayfa Sayısı: 1
		Ürün Fiyatı: 10€
Kriptoloji		▼
TTS Kitap		▼
Bulut Bilgi Ödevi		▼
Toplam Ücret: 59.5€		Sipariş Tarihi: 05.28.2024 00:55:04
Kırtasiye: Keskin Kırtasiye		Teslimat Durumu: Teslim Edildi
Kriptoloji		▼
Kriptoloji		▼

Şekil 38. Müşteri Sistemi - Sipariş Sayfasının Tasarımı

3.2.2.2. Satıcı Arayüz Tasarımı

Şekil 39’da satıcılarının kullanıcı girişi yapabilmesi için oluşturulan sayfanın tasarımı bulunmaktadır. Aynı zamanda giriş yapılmadan önceki header ve footer tasarımları da görülmektedir.

Şekil 39. Satıcı Sistemi - Kullanıcı Girişi Tasarımı

Şekil 40’da bayilik başvurusu için doldurulması gereken formun tasarımı gösterilmiştir. Form bilgileri müşteri sistemindekine benzer şekilde reactive form ile kontrolü yapılmaktadır. Veri formatında hata olması durumunda small etiketi çalışmaktadır.

Şekil 40. Satıcı Sistemi - Kullanıcı Başvurusu Tasarımı

Şekil 41’de satıcıların dosya işlemlerini yönetebilmeleri için oluşturulan sayfanın tasarımı gösterilmektedir. Dosyaları filtreleyebilmek için bir bölüm oluşturulmuştur. Dosya ekleyebilmek için “Dosya Ekle” butonu eklenmiştir.

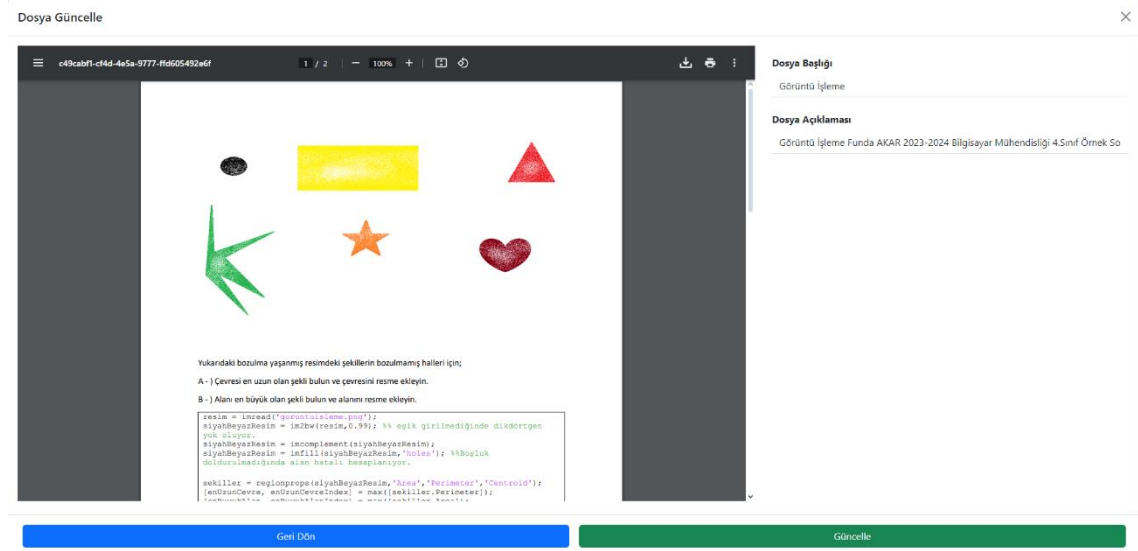
Şekil 41. Satıcı Sistemi - Satıcı Dosya Yönetim Sayfası Tasarımı

Şekil 42’de dosyanın özelliklerini gösteren bir bölümün tasarımı bulunmaktadır. “Görüntüle” butonuna tıklanıldığında dosya içeriği, başlığı ve açıklaması görüntülenmektedir ve güncellenebilmektedir. Sil butonuna tıklanıldığında Şekil 43’teki gösterilen modal ile onaya tabii tutulur.

Şekil 42. Satıcı Sistemi – Satıcı Dosyalarının Görüntülenmesi İçin Tasarım

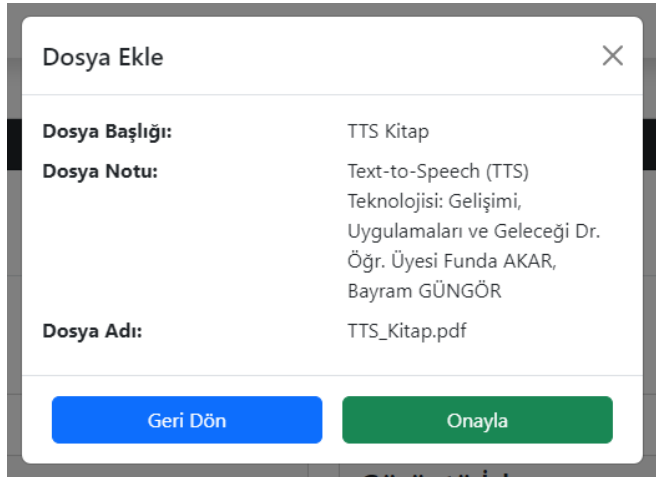
Şekil 43. Satıcı Sistemi - Satıcı Dosya Silmeyi Onaylama Modalının Tasarımı

Şekil 42’deki “Görüntüle” butonuna tıklanıldığında ise Şekil 44’deki modal açılarak dosya özellikleri modal aracılığıyla görüntülenmektedir. Eğer özellikler değiştirilmek isteniyorsa modalın sağ tarafında bulunan inputlar değiştirildikten sonra “Güncelle” butonuna tıklanılarak güncelleme işlemi yapılmaktadır.



Şekil 44. Satıcı Sistemi - Satıcı Dosya Güncelleme İşlemleri İçin Modal Tasarımı

Satıcı dosya yükleyebilmek için Şekil 33’teki modalın aynısı kullanılmaktadır. Modaldaki “İleri” butonuna tıklanıldığında Şekil 45’teki modal ile kullanıcıya girdiği bilgiler gösterilerek onay istenmektedir. “Onayla” butonuna tıklanıldığında dosya görüntülenebilir hale gelmektedir. “Geri Dön” butonuna tıklanıldığında ise önceki modala dönlür.



Şekil 45. Satıcı Sistemi - Satıcı Dosya Ekleme Onay Modalının Tasarımı

Şekil 42’de satıcıların paket işlemlerini yönetebilmeleri için oluşturulan sayfanın tasarımı gösterilmektedir. Paketleri filtreleyebilmek için bir bölüm oluşturulmuştur. Paket ekleyebilmek için “Paket Ekle” butonu eklenmiştir. Paketlerin kâğıt boyu, kâğıt türü, bir yüzdeki sayfa sayısı, renk seçeneği, baskı seçeneği, gösterim durumu ve fiyat özellikleri görüntülenmektedir.

Paket Adı	Kağıt Boyu	Kağıt Türü	Bir yüzdeki sayfa sayısı	Renk Seçeneği	Baskı Seçeneği	Gösterim Durumu
Paket Adı	Tümü	Tümü	Tümü	Tümü	Tümü	Tümü
Paket Ekle						Filtre Temizle

Paket Adı: Renkli A5 Paketi						
Kağıt Boyu	A5	Kağıt Türü	120GR(Kalın Kağıt)	Bir Yüzdeki Sayfa Sayısı	1	Gösterim Durumu
Renk Seçeneği	Renkli	Baskı Seçeneği	Arkalı-Önlu	Fiyat	10 TL	
						Güncelle Sil
Son Güncelleme Tarihi: 05.27.2024 23:00:15 Oluşturma Tarihi: 05.20.2024 23:04:22						

Paket Adı: Standart Paket						
Kağıt Boyu	A4 (Standart)	Kağıt Türü	80GR (Standart)	Bir Yüzdeki Sayfa Sayısı	1	Gösterim Durumu
Renk Seçeneği	Siyah-Beyaz	Baskı Seçeneği	Arkalı-Önlu	Fiyat	1.1 TL	
						Güncelle Sil
Son Güncelleme Tarihi: 05.19.2024 03:28:56 Oluşturma Tarihi: 05.18.2024 23:28:52						

Paket Adı: Bir Yüzde 4 Tane (Arkalı Önlu 8 tane) Siyah Beyaz Arkalı Önlu						
Kağıt Boyu		Kağıt Türü		Bir Yüzdeki Sayfa Sayısı		Gösterim Durumu
						Güncelle

Şekil 46. Satıcı Sistemi – Satıcı Paket Yönetim Sayfasının Tasarımı

Paketlerdeki “Güncelle” butonuna tıklanıldığında Şekil 47’deki modal açılarak paket özellikleri görüntülenebilmekte istenilen şekilde düzenlenebilmektedir.

Paket Güncelleme

Paket Adı:

Renkli A5 Paketi

Kağıt Boyu:

A5

Kağıt Türü

120GR(Kalın Kağıt)

Bir Yüzdeki Sayfa Sayısı

1

Renk Seçeneği

☒ Renkli
 ☐ Siyah-Beyaz

Baskı Seçeneği

☐ Tek Yüz
 ☒ Arkalı-Önlu

Paket Fiyatı (Kağıt Başına):

TL

İptal Et

Geri

Şekil 47. Satıcı Sistemi - Satıcı Paket Güncelleme Modalının Tasarımı

Şekil 47’deki modalda istenilen değişiklikler yapıldıktan sonra “İleri” butonuna tıklanıldığında Şekil 48’deki onay modalı açılacaktır. Modaldaki “Aktiflik durumu” seçeneği istenilen paket özelliğindeki stok sayısının bitmesi durumunda kullanıcıya gözükmemesi için kullanılmaktadır. Paketin müşterilere gözükmemesi için kapalı halde bırakılması gerekmektedir.

Paket Adı:	Renkli A5 Paketi
Kağıt Boyutu:	A5
Kağıt Türü:	120GR(Kalın Kağıt)
Bir Yüzdeki Sayfa Sayısı:	1
Renk Seçeneği:	Renkli
Baskı Seçeneği:	Arkalı-Önlü
Fiyat:	10
Aktiflik Durumu:	<input checked="" type="checkbox"/> (Bu seçenek aktifken paket müşteriler tarafından kullanılabilir.)

Geri Dön Onayla

Şekil 48. Satıcı Sistemi – Satıcı Paket Güncelleme Onaylama Modalının Tasarımı

Şekil 46’deki “Sil” butonuna tıklanıldığında ise Şekil 49’daki modal açılarak kullanıcıdan onay istenmektedir.

Paket Adı:	Renkli A5 Paketi
Kağıt Boyu:	A5
Kağıt Türü:	120GR(Kalın Kağıt)
Bir Yüzdeki Sayfa Sayısı:	Renkli A5 Paketi
Renk Seçeneği:	Arkalı-Önlü
Baskı Seçeneği:	Renkli
Gösterim Durumu:	Aktif
Fiyat:	10 TL

Paketin aktiflik durumunu pasif hale getirerek müşterilerin kullanmasını engelleyebilirsiniz.

Vazgeç Sil

Şekil 49. Satıcı Sistemi - Satıcı Paket Silmeyi Onaylama Modalının Tasarımı

Şekil 42’de sipariş işlemlerini yönetebilmeleri için oluşturulan sayfanın tasarımı gösterilmektedir. Sipariş filtreleyebilmek için bir bölüm oluşturulmuştur. Bu bölümde siparişler sıralanır. Sipariş veren müşterilerin bilgileri, sipariş ücreti, sipariş tarihi ve eğer sipariş durumu “Teslim Edildi” ya da “İptal Edildi” değil ise sipariş durumu değiştir butonu bulunmaktadır.

Sipariş Ara

Müşteri Adı	Müşteri Numarası	Sipariş Durumu
Müşteri Adı	Müşteri Numarası	Tümü

Filtrele

Temizle

Müşteri Ad Soyad: Bayram Güngör

Teslimat Durumu: Onay Bekliyor

Müşteri Adı: Bayram Güngör

Müşteri Adresi: Kalaba mah. Halliç Cad. 28/8 KEÇİÖREN ANKARA

Müşteri Numarası: 5070262571

Görüntü İşleme

Kriptoloji

TTS Kitap

Bulut Bilgişim Ödevi

Sipariş Durumunu Değiştir

Toplam Ücret: 59.54

Sipariş Tarihi: 05.28.2024 00:55:04

Müşteri Ad Soyad: Bayram Güngör

Teslimat Durumu: Teslim Edildi

Müşteri Adı: Bayram Güngör

Müşteri Adresi: Kalaba mah. Halliç Cad. 28/8 KEÇİÖREN ANKARA

Müşteri Numarası: 5070262571

Şekil 50. Satıcı Sistemi - Satıcı Sipariş Yönetim Sayfasının Tasarımı

Şekil 50’deki siparişlerden ürünün içeriğine tıklanıldığında Şekil 51’deki bölme açılır. Bu bölmede dosya özellikleri, yazdırma özellikleri ve müşteri notu görüntülenmektedir.

Müşteri Ad Soyad: Bayram Güngör

Teslimat Durumu: Onay Bekliyor

Müşteri Adı: Bayram Güngör

Müşteri Adresi: Kalaba mah. Halliç Cad. 28/8 KEÇİÖREN ANKARA

Müşteri Numarası: 5070262571

Görüntü İşleme

7d67c56d-7072-44fa-afc3-f932a51b19b9

1 / 2

100%

Yukarıdaki bozulma yağanmış resimdeki şekillerin bozulmamış halleri için;

A-) Çevresi en uzun olan şekli bulun ve çevresini resme ekleyin.

B-) Alanı en büyük olan şekli bulun ve alanını resme ekleyin.

```

resim = imread('goruntuisleme.png');
siyahBeyazResim = im2bw(resim,0.99); %% eşik girilmediğinde dikdörtgen
yok oluyor.
siyahBeyazResim = imcomplement(siyahBeyazResim);
siyahBeyazResim = imfill(siyahBeyazResim,'holes'); %%Boşluk
doldurulmadığında alan hatalı hesaplanıyor.

```

Dosya Özellikleri

Dosya Başlığı: Görüntü İşleme

Dosya Notu: Görüntü İşleme Funda AKAR 2023-2024 Bilgisayar Mühendisliği 4.Sınıf Örnek Soru

Sayfa Sayısı: 2

Yazdırma Özellikleri

Renk Seçeneği: Renkli

Baskı Seçeneği: Arkalı-Önlu

Kağıt Boyu: A5

Kağıt Türü: 120GR(Kalın Kağıt)

Bir Yüzdeki Sayfa Sayısı: 1

Adet: 1

Müşteri Notu

Renkler canlı çıksın ve silik olmasın.

Ürün Fiyatı: 10₺

Şekil 51. Satıcı Sistemi - Satıcı Siparişteki Ürünleri Görüntülemesi İçin Tasarım

Satıcıların bilgilerini ve görsellerini değiştirebilmeleri için, kullanıcı arayüzündeki header bölümünde yer alan "Hesap" butonuna tıklamaları gerekmektedir. Bu eylem sonucunda açılan sayfada, Şekil 52, Şekil 53 ve Şekil 54'te tasarımları sunulan bölümler görüntülenecektir. Bu bölümler, satıcının mevcut ve güncel bilgileriyle doldurulmuş olarak gelmektedir. Satıcıların bu bilgilerde değişiklik yapmak istemeleri durumunda, ilgili input alanlarını güncellemeleri ve ardından "Kaydet" butonuna tıklamaları gerekmektedir. Bu işlem, yapılan değişikliklerin sistemde kaydedilmesini ve güncellenmesini sağlayarak, kullanıcı bilgilerinin sürekli olarak doğru ve güncel tutulmasını mümkün kılar.

Adres Bilgileri	
Şehir Ankara	İlçe Keçiören
Adres 61/B Sanatoryum Caddesi	
<button>Kaydet</button>	

Şekil 52. Satıcı Sistemi - Satıcı Hesap Bilgileri Adres Değişikliği Tasarımı

Şirket Bilgileri	
Şirket Adı Keskin Kırtasiye	
Telefon Numarası 5070262571	
Şirket Hakkında fasd	
<button>Kaydet</button>	

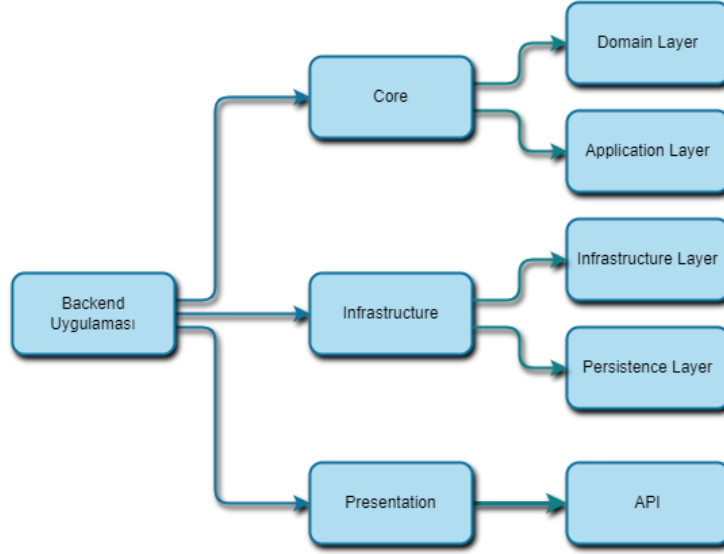
Şekil 53. Satıcı Sistemi - Satıcı Hesap Bilgileri Şirket Değişikliği Tasarımı

Görseller	
Logo Dosya Seç Dosya seçilmedi	
<p>Görsel Boyutu 200x300 piksel olmalıdır.</p>	
Kapak Fotoğrafı Dosya Seç Dosya seçilmedi	
<p>Görsel Boyutu 500x120 piksel olmalıdır.</p>	
<button>Kaydet</button>	

Şekil 54. Satıcı Sistemi - Satıcı Hesap Bilgileri Görsel Değişikliği Tasarımı

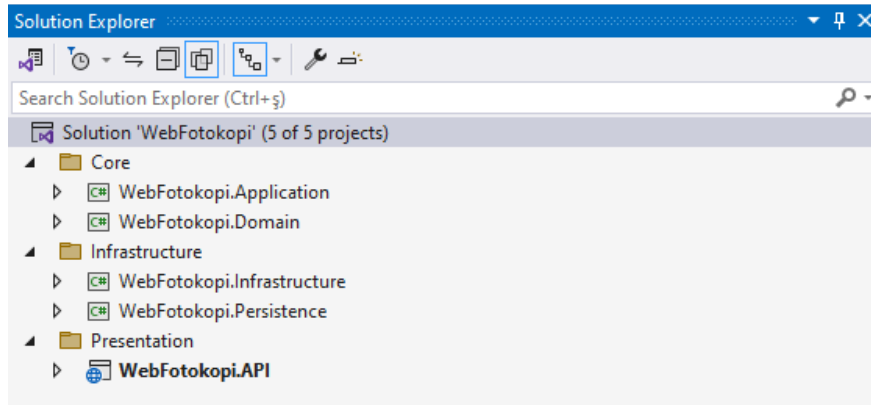
3.2.3. Yazılım Tasarımı

3.2.3.1. Backend Tasarımı



Şekil 55. Onion Mimaride Proje Yapısı

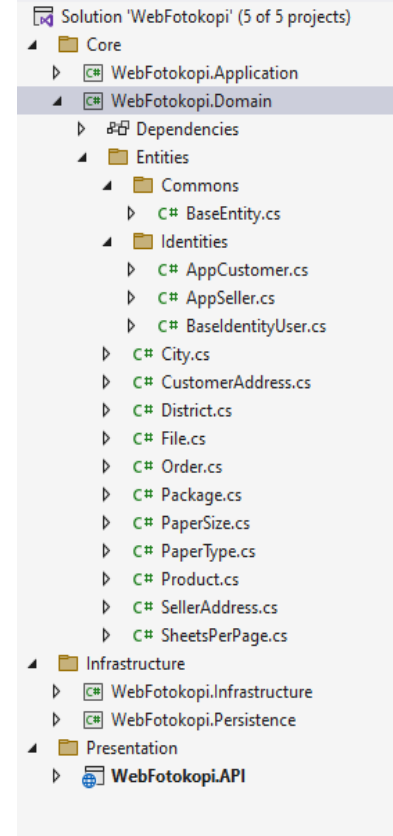
Backend uygulaması oluşturulurken, onion mimarisini benimsemek amacıyla başlangıçta boş bir proje oluşturulmuş ve Core, Infrastructure ve Presentation isimli klasörler yapılandırılmıştır. Core klasörü altında WebFotokopi.Domain ve WebFotokopi.Application isimli class library'ler (.NET 8.0 Framework) oluşturulmuştur. Infrastructure klasöründe ise WebFotokopi.Infrastructure ve WebFotokopi.Persistence isimli class library'ler yer almaktadır. Presentation klasöründe ise WebFotokopi.API isimli bir ASP.NET Core Web API projesi yapılandırılmıştır. Şekil 56'da Backend projesinin yapısı gösterilmiştir.



Şekil 56. Backend Proje Yapısı

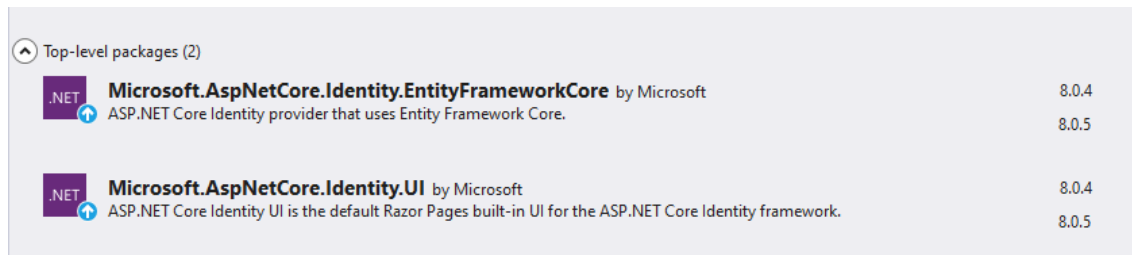
- Application katmanı, Domain katmanını referans göstermiştir.
- Infrastructure katmanı, Application ve Persistence katmanını referans göstermiştir.
- Persistence katmanı, Application ve Infrastructure katmanını referans göstermiştir.
- API katmanı, Persistence ve Infrastructure katmanını referans göstermiştir.

Domain katmanı içerisinde, uygulamanın iş mantığını ve veri modellerini organize etmek amacıyla Entities isimli bir klasör yapılandırılmıştır. Bu klasörde, entity'lerin ortak atalarını ve genel özelliklerini barındıran Commons isimli bir alt klasör oluşturulmuştur. Commons klasörü, kod tekrarını önlemek ve sürdürülebilirliği artırmak için tüm entity'lerin kalıtım yoluyla paylaşacakları genel özellikleri ve metodları içerir. Ayrıca, kullanıcı kimlik doğrulama ve yetkilendirme süreçlerinde kullanılan entity'leri yönetmek amacıyla Identities isimli bir alt klasör oluşturulmuştur. Bu yapı, uygulamanın güvenlik katmanını sağlamlaştırarak kimlik yönetimi ile ilgili işlemleri merkezi bir noktada toplar. Uygulamanın işlevselliğini sağlamak için gerekli olan diğer tüm entityler ilgili oldukları bağlama göre bu klasör yapısı içerisinde tanımlanmıştır.



Şekil 57. Domain Katmanı

Şekil 57’de domain katmanındaki dosyalar ve sınıflar gösterilmiştir. Şekil 58’de domain katmanında kullanılan nuget paketlerinin isimleri ve versiyonları bulunmaktadır. (Proje yazımı sırasında var olan son sürüm olan 8.0.4 kullanılmıştır.)

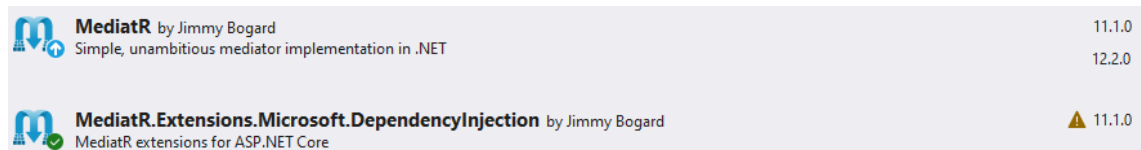


Şekil 58. Domain Katmanının da Kullanılan Nuget Paketleri

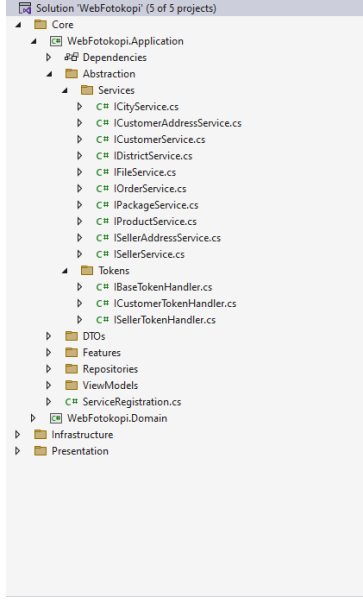
Application katmanı içerisinde;

- Abstraction isimli bir klasör, soyut sınıfların tanımlanması amacıyla yapılandırılmıştır. Bu klasör içerisinde, servislerin soyut sınıflarını tanımlamak için Services, token'ların soyut sınıflarını tanımlamak için ise Tokens isimli alt klasörler oluşturulmuştur. (Şekil 60)
- Uygulamada kullanılacak olan DTO dosyaları için DTOs isimli bir klasör yapılandırılmıştır. Bu klasör içerisinde ilgili sonuçların gruplandırıldığı alt klasörler bulunmakta ve her bir DTO bu klasörlerde tanımlanmaktadır. (Şekil 61)
- CQRS mimarisine uygun dosyalar için Features isimli bir klasör oluşturulmuştur. Bu klasör içerisinde Commands ve Queries isimli alt klasörler yer almaktadır. Commands klasöründe Create, Delete ve Update işlemlerine ilişkin alt klasörler; Queries klasöründe ise Read işlemleri için alt klasörler bulunmaktadır. Her bir alt klasör içerisinde Handler, Request ve Response sınıfları tanımlanmıştır. (Şekil 62)
- Generic Repository deseninin uygulanabilmesi ve gerekli soyut dosyaların eklenmesi amacıyla Repositories isimli bir klasör oluşturulmuştur. Bu klasör içerisinde, entity'lerin isimlerine uygun olarak her entity için (identity entity'leri hariç) alt klasörler tanımlanmıştır. Her bir entity için Read ve Write işlemlerini ayrı ayrı ele alan repository sınıfları oluşturulmuştur. (Şekil 63)
- ViewModel'lerin eklenmesi için ViewModels isimli bir klasör yapılandırılmıştır. Bu klasör, işlem yapılacak tabloların isimlerine uygun alt klasörler içermekte olup, gerekli sınıflar bu alt klasörler içerisinde tanımlanmaktadır. (Şekil 64)
- Son olarak, servis kayıtlarının eklenmesi amacıyla ServiceRegistration isimli bir sınıf oluşturulmuştur. Bu sınıf, bağımlılıkların kaydedilmesi ve uygulama bileşenlerinin merkezi bir noktadan yönetilmesi işlevini görmektedir.

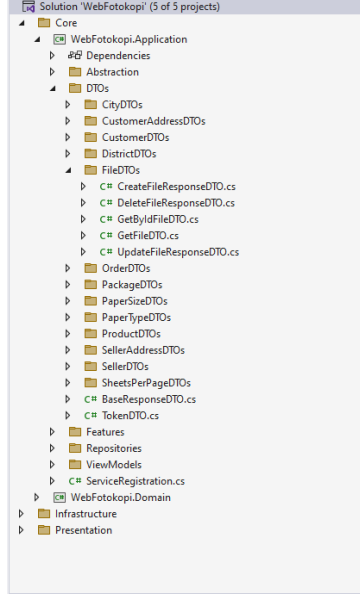
Şekil 59'de domain katmanındaki dosyalar ve sınıflar gösterilmiştir. Şekil 58'de domain katmanında kullanılan nuget paketlerinin isimleri ve versiyonları bulunmaktadır.



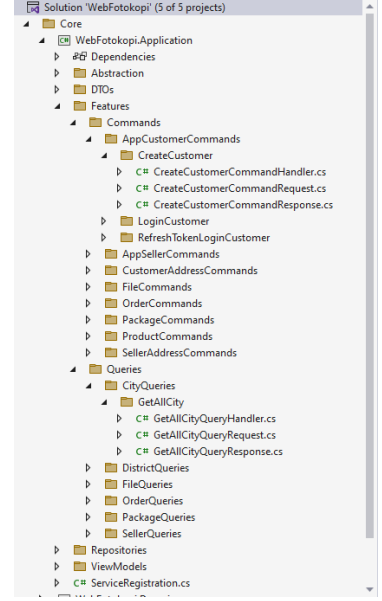
Şekil 59. Application Katmanının da Kullanılan Nuget Paketleri



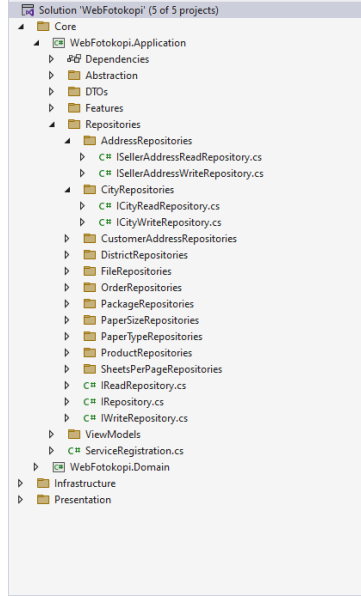
Şekil 60. Application Katmanı
Abstraction Klasörü



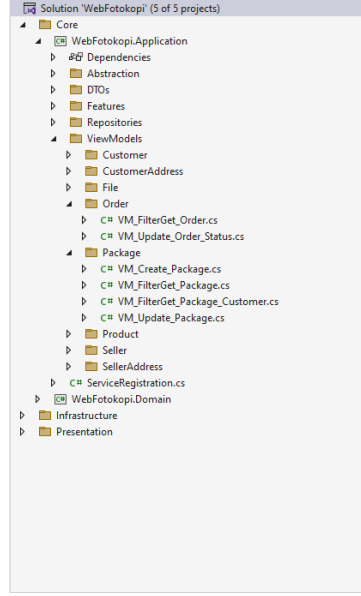
Şekil 61. Application Katmanı
DTOs Klasörü



Şekil 62. Application Katmanı
Features Klasörü

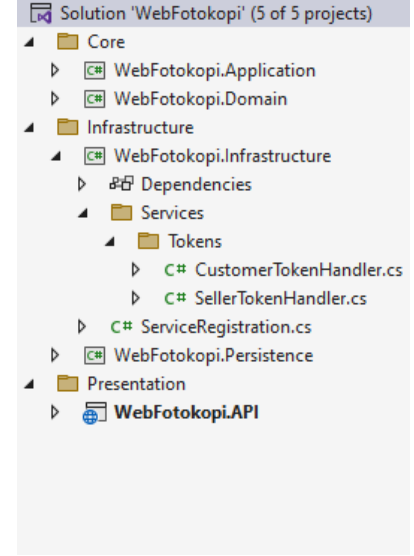


Şekil 63. Application Katmanı
Repositories Klasörü



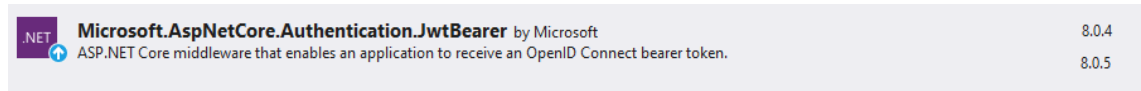
Şekil 64. Application Katmanı
ViewModels Klasörü

Infrastructure katmanı içerisinde, servis ile ilgili dosyaları organize etmek amacıyla Services isimli bir klasör ve servis kayıtlarını eklemek için ServiceRegistration isimli bir sınıf oluşturulmuştur. Services klasörü içerisinde, token ile ilgili dosyaları depolamak amacıyla Tokens isimli bir alt klasör yapılandırılmıştır. Bu klasör içerisinde ise, CustomerTokenHandler ve SellerTokenHandler adında iki sınıf tanımlanmıştır. Şekil 65'te, Infrastructure katmanındaki dosyalar ve sınıflar detaylı bir şekilde gösterilmiştir.



Şekil 65. Infrastructure Katmanı

Şekil 66'da infrastructure katmanında kullanılan nuget paketi ve sürümü gösterilmiştir.



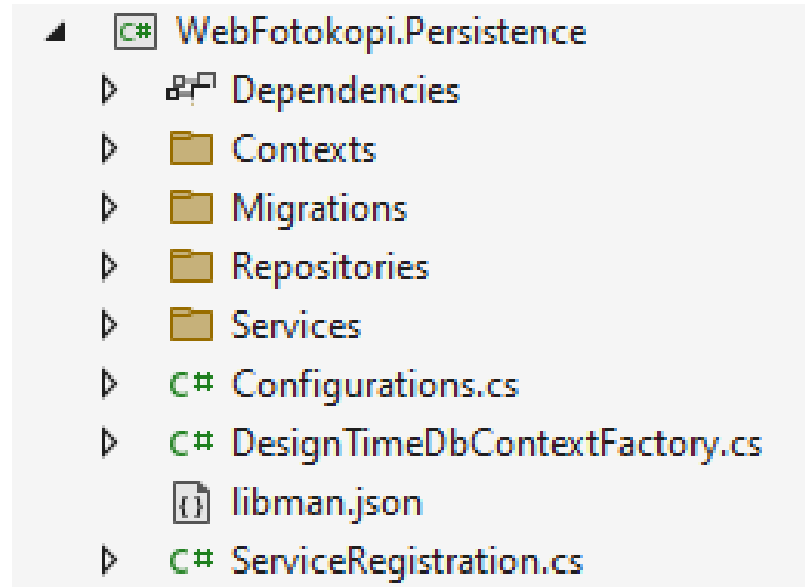
Şekil 66. Infrastructure Katmanında Kullanılan Nuget Paketi

Persistence katmanı içerisinde çeşitli görevleri yerine getirmek amacıyla spesifik klasörler ve sınıflar yapılandırılmıştır.

- Veri tabanı context işlemlerini yönetmek için Contexts isimli bir klasör oluşturulmuş ve bu klasör içerisinde WebFotokopiDbContexts isimli bir sınıf tanımlanmıştır.
- Migration işlemlerini kolaylaştırmak amacıyla otomatik olarak Migrations klasörü oluşturulmuş olup bu klasörde migration sınıfları bulunmaktadır.
- Veri erişim katmanını soyutlamak ve genel CRUD operasyonlarını gerçekleştirmek için Repositories klasörü yapılandırılmış, bu klasör içerisinde soyut sınıfları oluşturulan repository'lerin içerikleri doldurulmuştur.
- Servis işlemlerini yönetmek için ise soyut sınıfları oluşturulan servislerin içerikleri tanımlanmıştır.

- Bu yapılandırmaya ek olarak Configurations, DesignTimeDbContextFactory ve ServiceRegistration sınıfları da eklenmiştir. Bu ek sınıflar, veri tabanı yapılandırmaları, tasarım zamanı veri tabanı context fabrikası ve servis kayıt işlemlerini yönetmek amacıyla kullanılmaktadır.

Şekil 67'de, Persistence katmanının içerisindeki klasörler ve sınıflar detaylı bir şekilde gösterilmiştir. Şekil 68'de ise, Persistence katmanında kullanılan NuGet paketlerinin isimleri ve versiyonları belirtilmiştir.



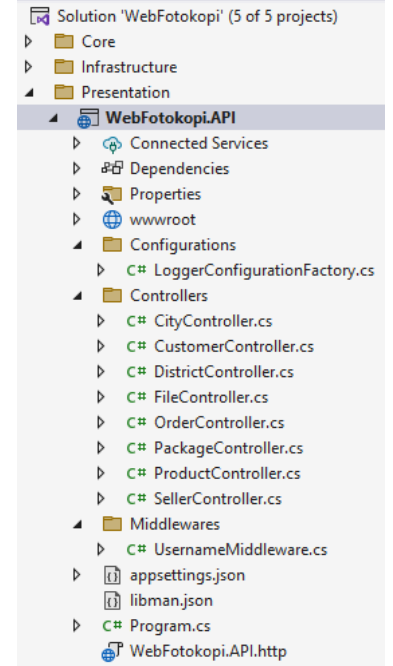
Şekil 67. Persistence Katmanı

	iTextSharp by iText Software iTextSharp is a DEPRECATED library for PDF generation written entirely in C# for the .NET platform. Please use iText 7 instead. iText 7 Community: https://www.nuget.org/packages/itext7/	5.5.13.3
	Microsoft.EntityFrameworkCore by Microsoft Entity Framework Core is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations. EF Core works with SQL Server, Azure SQL Database, SQLite, Azure Cosmos DB, MySQL, PostgreSQL, and other databases through a provider plugin API.	8.0.4 8.0.5
	Microsoft.EntityFrameworkCore.Design by Microsoft Shared design-time components for Entity Framework Core tools.	8.0.4 8.0.5
	Microsoft.EntityFrameworkCore.SqlServer by Microsoft Microsoft SQL Server database provider for Entity Framework Core.	8.0.4 8.0.5
	Microsoft.Extensions.Configuration by Microsoft Implementation of key-value pair based configuration for Microsoft.Extensions.Configuration. Includes the memory configuration provider.	8.0.0
	Microsoft.Extensions.Configuration.Json by Microsoft JSON configuration provider implementation for Microsoft.Extensions.Configuration. This package enables you to read your application's settings from a JSON file. You can use <code>JsonConfigurationExtensions.AddJsonFile</code> extension method on <code>IConfigurationBuilder</code> to add the JSON configuration...	8.0.0

Şekil 68. Persistence Katmanında Kullanılan Nuget Paketleri







API katmanının içerisinde;

- Konfigürasyon dosyalarının tutulması için Configurations isimli bir klasör oluşturulmuştur. Bu klasör içerisinde log kayıtları için LoggerConfigurationFactory isimli bir sınıf tanımlanmıştır.
- API için istekler ilgi başlık altında sınıflandırılarak ayrı ayrı controller'ler oluşturulmuştur.
- Middleware'lerin eklenmesi için Middlewares isimli klasör oluşturulmuştur. Bu klasör içerisinde işlem yapan kullanıcının username değerini alabilmek için kullanılan UsernameMiddleware isimli bir sınıf tanımlanmıştır.
- Appsettings.json dosyası içerisinde gerekli yapılandırmalar yapılmıştır.



Şekil 69. API Katmanı

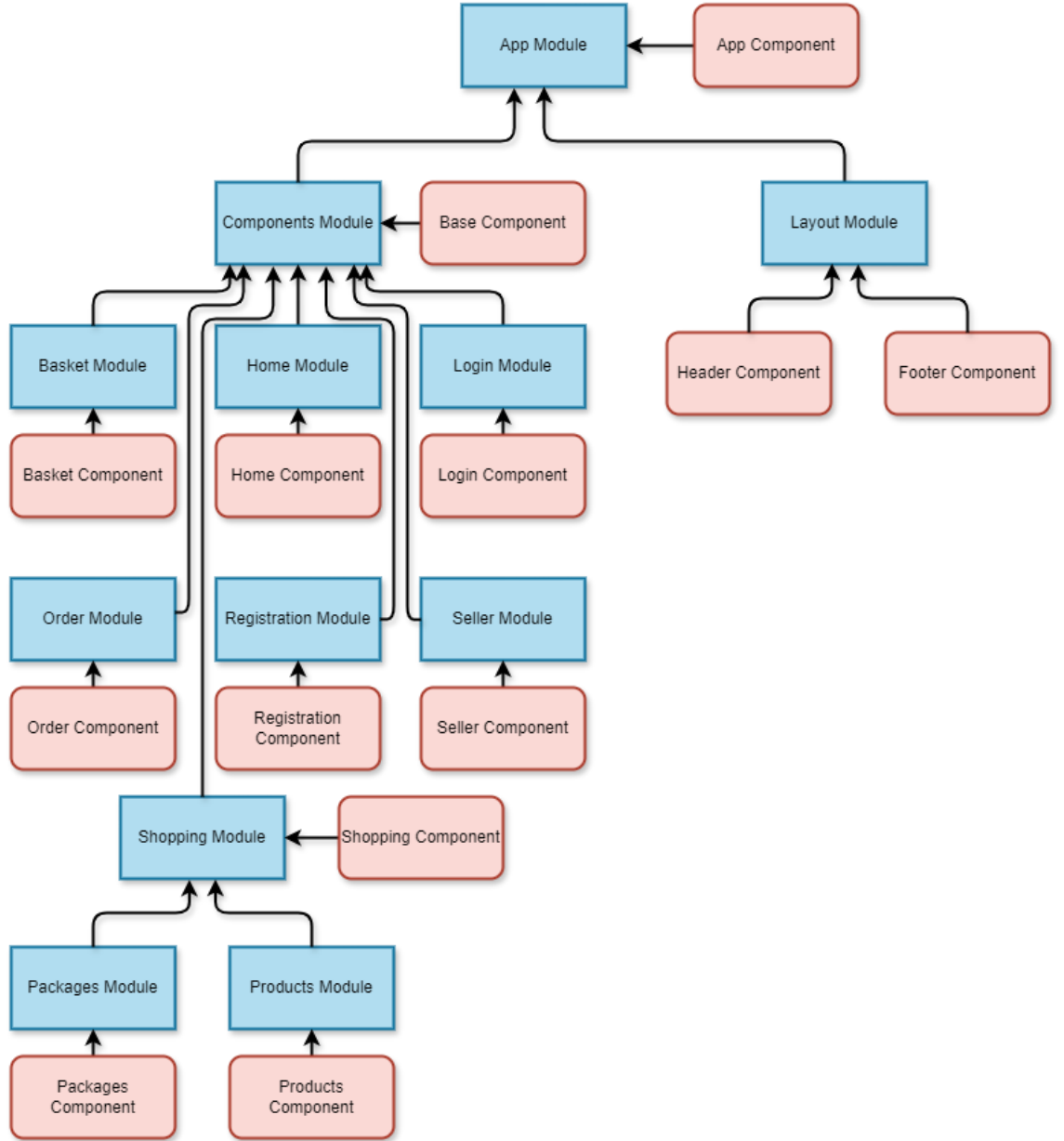
Şekil 70'de Persistence katmanında kullanılan NuGet paketlerinin isimleri ve versiyonları belirtilmiştir.

	Microsoft.AspNetCore.Authentication.JwtBearer by Microsoft ASP.NET Core middleware that enables an application to receive an OpenID Connect bearer token.	8.0.4 8.0.5
	Microsoft.EntityFrameworkCore.Tools by Microsoft Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.	8.0.4 8.0.5
	Serilog by Serilog Contributors Simple .NET logging with fully-structured events	3.1.1
	Serilog.AspNetCore by Microsoft, Serilog Contributors Serilog support for ASP.NET Core logging	8.0.1
	Serilog.Sinks.MSSqlServer by Michiel van Oudheusden, Christian Kadluba, Serilog Contributors A Serilog sink that writes events to Microsoft SQL Server and Azure SQL	6.6.0
	Swashbuckle.AspNetCore by Swashbuckle.AspNetCore Swagger tools for documenting APIs built on ASP.NET Core	6.5.0 6.6.2

Şekil 70. API Katmanında Kullanılan Nuget Paketleri

3.2.3.2. Müşteri Frontend Uygulamasının Tasarımı

Müşteriler için oluşturulan frontend uygulaması, Angular'ın modül yapısı kullanılarak Şekil 71'de gösterilen şekilde tasarlanmıştır.



Şekil 71. Müşteri Frontend Uygulamasının Module - Component İlişkisi

Şekil 72'de, müşteriler için tasarlanmış olan frontend uygulamasının route ve guard yapısı sunulmuştur. Bu görsele ek olarak, alışveriş (shopping) route'unun altında boş ("") path için product component ve "packages" route'unun altında ise packages component yer

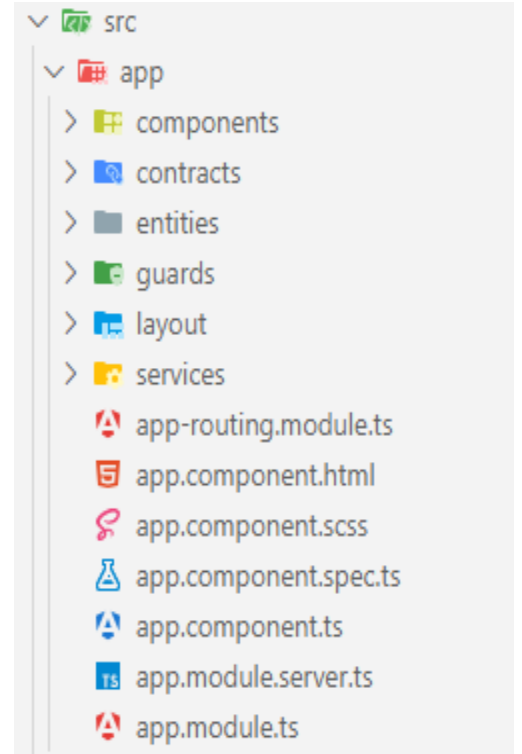
almaktadır. AuthGuard eklenen path'lere sadece oturum açılarak girilebilirken, LogoutAuthGuard eklenen path'lere sadece oturum kapalıyken girilebilmektedir.

```
const routes: Routes = [
  {path: "", component: HomeComponent, loadChildren: () =>
    import("./components/home/home.module").then(module=>module.HomeModule), canActivate: [logoutAuthGuard]},
  {path: "seller", component: SellerComponent, loadChildren: () =>
    import("./components/seller/seller.module").then(module=>module.SellerModule), canActivate: [authGuard]},
  {path: "basket", component: BasketComponent, loadChildren: () =>
    import("./components/basket/basket.module").then(module=>module.BasketModule), canActivate: [authGuard]},
  {path: "order", component: OrderComponent, loadChildren: () =>
    import("./components/order/order.module").then(module=>module.OrderModule), canActivate: [authGuard]},
  {path: "shopping", component: ShoppingComponent, loadChildren: () =>
    import("./components/shopping/shopping.module").then(module=>module.ShoppingModule), canActivate: [authGuard]},
  {path: "login", component: LoginComponent, loadChildren: () =>
    import("./components/login/login.module").then(module=>module.LoginModule), canActivate: [logoutAuthGuard]},
  {path: "registration", component: RegistrationComponent, loadChildren: () =>
    import("./components/registration/registration.module").then(module=>module.RegistrationModule), canActivate: [logoutAuthGuard]},
  { path: "**", redirectTo: "" }
];
```

Şekil 72. Müşteri Frontend Uygulamasının Route Yapısı

Angular projesinin app klasörünün içerisinde aşağıdaki dosyalar oluşturulmuştur.

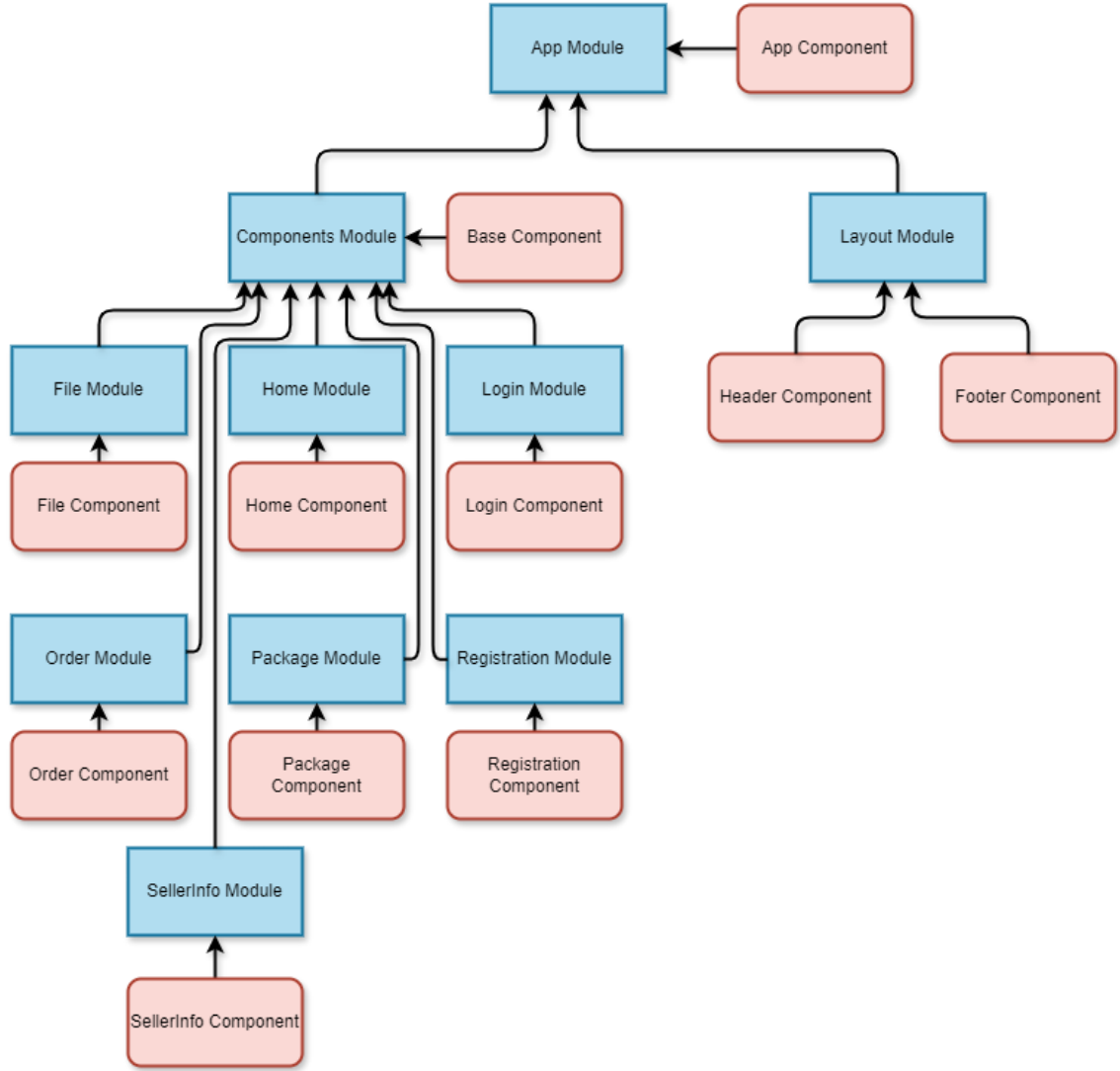
- components: Components klasörü, Şekil 71'de gösterilen component modül yapısını içermektedir.
- contracts: Contract dosyalarının bulunduğu klasör.
- entities: Entity dosyalarının bulunduğu klasör.
- guards: Guard dosyalarının bulunduğu klasör.
- layout: Layout klasörü, Şekil 71'deki layout modül yapısının tanımlandığı yerdir.
- services: Servis dosyaları bu klasör altında tanımlanmıştır.



Şekil 73. Müşteri Frontend Uygulamasının Klasör Yapısı

3.2.3.2. Satıcı Frontend Uygulamasının Tasarımı

Satıcılar için oluşturulan frontend uygulaması, Angular'ın modül yapısı kullanılarak Şekil 74'de gösterilen şekilde tasarlanmıştır.



Şekil 74. Satıcı Frontend Uygulamasının Module - Component İlişkisi

Şekil 75'de, müşteriler için tasarlanmış olan frontend uygulamasının route ve guard yapısı sunulmuştur. AuthGuard eklenen path'lere sadece oturum açılarak girilebilirken, LogoutAuthGuard eklenen path'lere sadece oturum kapalıyken girilebilmektedir.

```

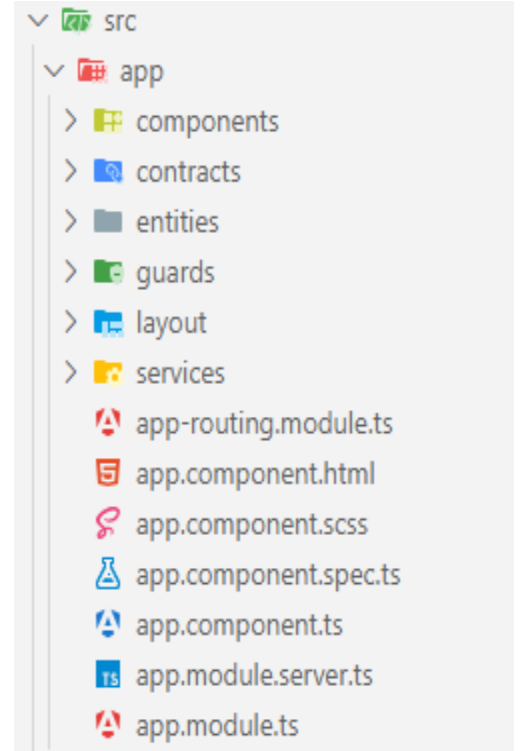
const routes: Routes = [
  {path:"",component:HomeComponent,loadChildren:() =>
    import("../components/home/home.module").then(module=>module.HomeModule)},
  {path:"login",component:LoginComponent,loadChildren:() =>
    import("../components/login/login.module").then(module=>module.LoginModule),canActivate:[logoutAuthGuard]},
  {path:"registration",component:RegistrationComponent,loadChildren:() =>
    import("../components/registration/registration.module").then(module=>module.RegistrationModule),canActivate:[logoutAuthGuard]},
  {path:"orders",component:OrdersComponent,loadChildren:() =>
    import("../components/orders/orders.module").then(module=>module.OrdersModule),canActivate:[AuthGuard]},
  {path:"packages",component:PackagesComponent,loadChildren:() =>
    import("../components/packages/packages.module").then(module=>module.PackagesModule),canActivate:[AuthGuard]},
  {path:"sellerInfo",component:SellerInfoComponent,loadChildren:() =>
    import("../components/seller-info/seller-info.module").then(module=>module.SellerInfoModule),canActivate:[AuthGuard]},
  {path:"files",component:FilesComponent,loadChildren:() =>
    import("../components/files/files.module").then(module=>module.FilesModule),canActivate:[AuthGuard]},
  { path: "****", redirectTo: "" }
];

```

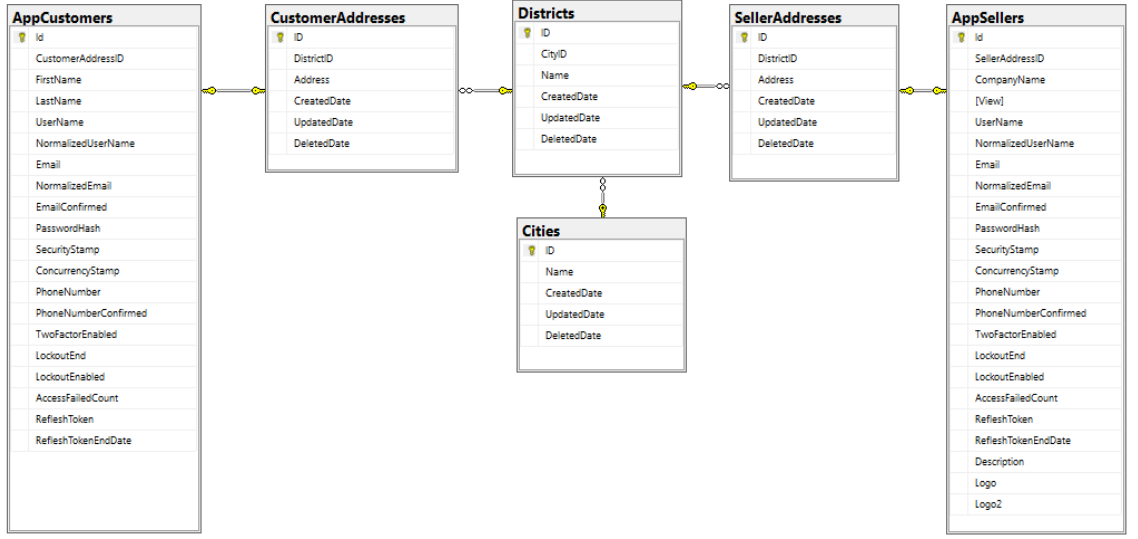
Şekil 75. Satıcı Frontend Uygulamasının Route Yapısı

Angular projesinin app klasörünün içerisinde aşağıdaki dosyalar oluşturulmuştur.

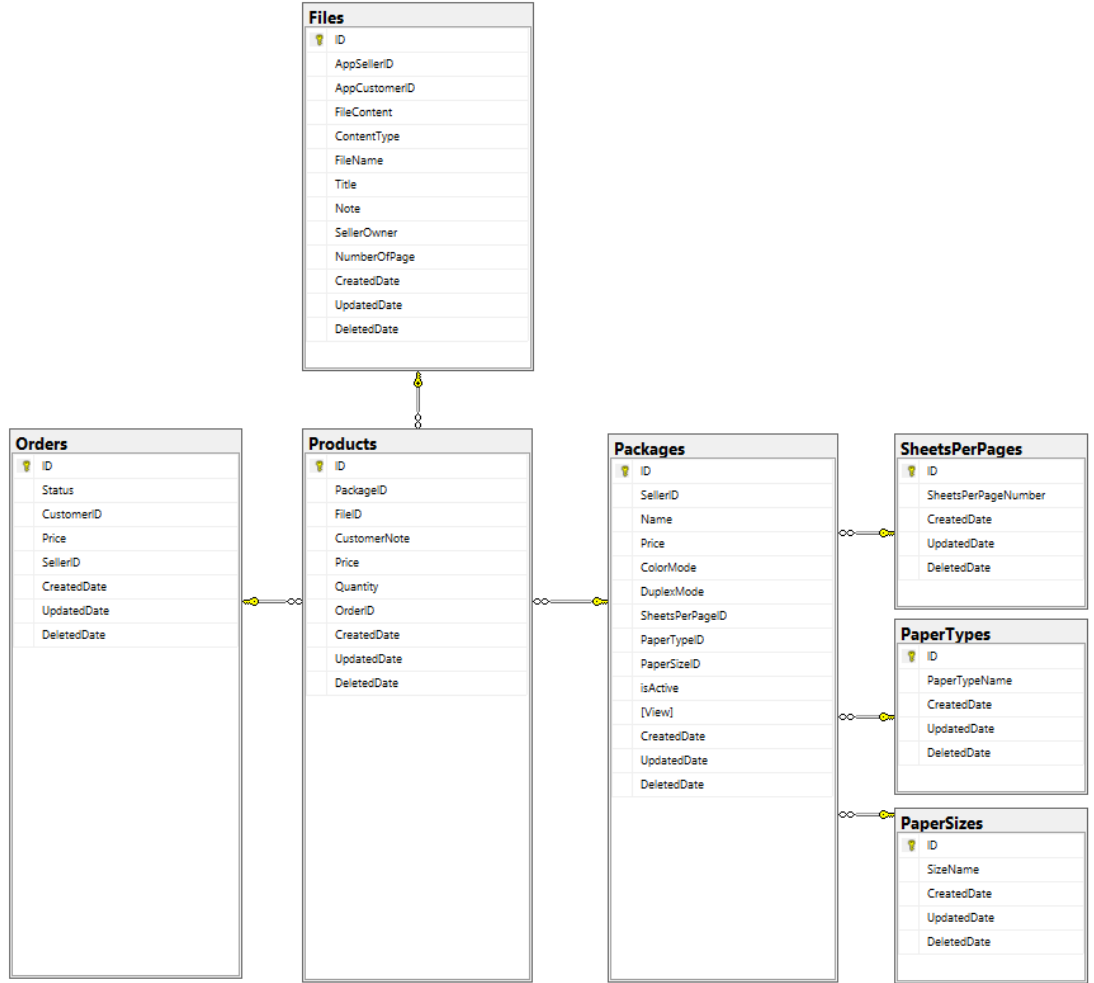
- components: Components klasörü, Şekil 71'de gösterilen component modül yapısını içermektedir.
- contracts: Contract dosyalarının bulunduğu klasör.
- entities: Entity dosyalarının bulunduğu klasör.
- guards: Guard dosyalarının bulunduğu klasör.
- layout: Layout klasörü, Şekil 71'deki layout modül yapısının tanımlandığı yerdir.
- services: Servis dosyaları bu klasör altında tanımlanmıştır.



Şekil 76. Satıcı Frontend Uygulamasının Klasör Yapısı



Şekil 79. Veri Tabanı Tablo İçerikleri Kısım 2



Şekil 80. Veri Tabanı Tablo İçerikleri Kısım 3

4. SONUÇ ve ÖNERİLER

Bu çalışma kapsamında, satıcılar ve müşteriler için Angular ile iki farklı web sitesi tasarımı ve .NET Core ile bir backend uygulaması geliştirilmiştir. Uygulamada Onion Architecture, EF Core, Generic Repository, CQRS, Microsoft Identity, JwtBearer, Refresh Token, Serilog, iTextSharp, Bootstrap, Reactive Form gibi çeşitli teknolojiler kullanılmıştır. Bu teknolojiler aracılığıyla aşağıdaki özelliklere sahip bir web uygulaması geliştirilmiştir:

Satıcılar için

- **Bayilik Başvuru Sayfası:** Satıcılar, bu sayfada WebFotokopi sistemine bayilik başvurusunda bulunabilirler.
- **Giriş Sayfası:** Satıcı bayiler, bu sayfadan WebFotokopi sistemine giriş yapabilirler.
- **Paketler Sayfası:** Satıcılar, baskı seçeneklerine göre (örn. Renkli - A4 – 150gr – 5 TL) paketler oluşturabilirler. Ayrıca, oluşturulan paketleri filtreleyerek görüntüleyebilir, güncelleyebilir, silebilir ve aktiflik durumlarını değiştirebilirler.
- **Dosyalar Sayfası:** Satıcılar, müşterilerine filigranlı şekilde sunmak istedikleri dosyaları sisteme yükleyebilir, yüklenen dosyaları filtreleyerek görüntüleyebilir, güncelleyebilir ve silebilirler.
- **Hesap Sayfası:** Satıcılar, bu sayfada şirket bilgilerini, adreslerini ve logolarını güncelleyebilirler.
- **Siparişler Sayfası:** Satıcılar, bu sayfada gelen siparişleri görüntüleyebilir ve sipariş durumlarını değiştirebilirler.

Müşteriler için

- **Kullanıcı Kayıt Sayfası:** Müşteriler, bu sayfada WebFotokopi sistemine kayıt olabilirler.
- **Giriş Sayfası:** Müşteriler, bu sayfadan WebFotokopi sistemine giriş yapabilirler.
- **Kırtasiyelerin Listelendiği Sayfa:** Müşteriler, bu sayfada satıcıları konumlarına ve çeşitli filtreleme seçeneklerine göre listeleterek görüntüleyebilirler.

- **Satıcının Sayfası:** Bu sayfada, satıcıların müşterilerine sunmak istedikleri dosyalar, aktif olan paketler ve şirket bilgileri görüntülenir. Müşteriler, sipariş oluşturabilmek için dosya seçtikten sonra paket seçimi yapabilirler. Paket seçiminden sonra, ürün için dosya notu, adet gibi bilgilerin girilmesi gereken bir modal daha görüntülenir. Bu bilgilerin girilmesiyle ürün alışveriş sepetine eklenir.
- **Alışveriş Sepeti:** Bu sayfada, oluşturulan ürünler ve fiyatları listelenir. Alışveriş sepetindeki ürünler güncellenebilir veya silinebilir.
- **Siparişler Sayfası:** Müşteriler, bu sayfada daha önce oluşturulan siparişlerin içeriklerini, fiyatlarını ve sipariş durumlarını görüntüleyebilirler.

Yukarıda belirtilen sayfalar aracılığıyla, giriş bölümünde tanımlanan kırtasiyelerin ve öğrencilerin yaşadıkları problemler çözüme kavuşturulmuştur.

Öneriler

Benzer bir uygulama geliştirilmek istendiğinde, aşağıdaki öneriler göz önünde bulundurulmalıdır:

- **Dosya Yönetimi:** Dosyalar şu anda BASE64 formatında veri tabanında saklanmaktadır. Bu yöntem, dosyaların yavaş yüklenmesine sebep olmaktadır. Bu sorunun üstesinden gelmek için, dosyalar Microsoft Azure gibi bir bulut hizmetine kaydedilmeli ve yalnızca dosya yolları veri tabanında saklanmalıdır.
- **Doğrulama (Validation):** Uygulamanın doğrulama işlemlerinde eksiklikler bulunmaktadır. Bazı doğrulamalar frontend tarafında gerçekleştirilse de, güvenlik amacıyla backend tarafında da doğrulama yapılmalıdır.
- **Responsive Tasarım:** Bazı sayfalar Bootstrap ile responsive olarak tasarlanmış olmasına rağmen, mobil cihazlarda tasarımlar bozulmaktadır. Bu sorunu gidermek için, tüm tasarımlar tamamen responsive hale getirilmelidir.
- **Kullanıcı İşlemleri:** Kullanıcı işlemleri için bir e-posta servisi eklenebilir. Bu, kullanıcı kayıt ve şifre sıfırlama gibi işlemler için gereklidir.
- **Veri Tabanı Tasarımı:** Veri tabanındaki tablolar sadeleştirilebilir. Bu, veri tabanının daha verimli ve yönetilebilir olmasını sağlar.

- **Tür Dönüşümleri:** Tür dönüşümleri için Automapper kullanılabilir. Bu, veri transfer nesneleri (DTO) ve model sınıfları arasında dönüşümleri kolaylaştırır ve kodun daha temiz olmasını sağlar.
- **İşlem Yönetimi:** İşlemlerin yarım kalması durumunda tüm işlemi iptal edebilmek için UnitOfWork yapısı kullanılabilir. Bu, veri tutarlılığını sağlar ve işlemler arasında bütünlüğü korur.

Bu öneriler dikkate alındığında, geliştirilecek olan uygulamanın performansı, güvenliği ve kullanıcı deneyimi önemli ölçüde iyileştirilebilir.

KAYNAKLAR

- Ateş, Ü. Ç. (2024). *Pandemi Sürecinde E-ticaret Uygulamasının Evrildiği Yeni Ortaklıklar.. Dördüncü Taraf Lojistik.*
- AWS. (n.d.). *CQRS pattern.* Retrieved May 27, 2024, from <https://docs.aws.amazon.com/prescriptive-guidance/latest/modernization-data-persistence/cQRS-pattern.html#:~:text=The%20command%20query%20responsibility%20segregation,throughput%2C%20latency%2C%20or%20consistency.>
- Bootstrap. (2024). *Bootstrap 5.3.3.* <https://blog.getbootstrap.com>
- Özer, B. (2008). *YENİ BİR İŞ MODELİ OLARAK İNTERNET TABANLI DİJİTAL BASKI SİSTEMLERİ.*
- Fariha, F. (2023). *JWT Token Authentication Web API.* <https://coredevsltd.com/articles/jwt-token-authentication-web-api/>
- Microsoft Developer Support. (2017). *.NET Core Overview.* <https://devblogs.microsoft.com/premier-developer/net-core-overview/>
- Microsoft Learn. (2021). *Entity Framework overview.* <https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/ef/overview>
- Microsoft Learn. (2024). *Create a web API with ASP.NET Core.* <https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-8.0&tabs=visual-studio>
- Mutlu, B., Özcan, A., & Hayta, P. (2018). *Matbaacılıkta E-Ticaret ve Web2Print MATBAACILIKTA E-TİCARET VE WEB2PRINT.*
- Nanehkaran, Y. A. (2013). An Introduction To Electronic Commerce. *INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH*, 2. www.ijstr.org

Shukla, H. (2020). *What is the difference between access and refresh token?*
<https://medium.com/@greekykhs/springsecurity-what-is-the-difference-between-access-and-refresh-token-65296bcb13fc>

T.C. Ticaret Bakanlığı. (2019). *Dijital Ticaret Tanım ve Kavramlar*.
<https://ticaret.gov.tr/hizmet-ticareti/elektronik-ticaret/dijital-ticaret-tanim-ve-kavramlar>

Yıldız, G. (2019). *C# Repository Design Pattern*. <https://www.gencayyildiz.com/blog/c-repository-design-patternrepository-tasarim-deseni/>

EKLER

Ek-1. Backend Uygulaması GitHub Adresi

<https://github.com/byrmgng/WebFotokopi-API>

Ek-2. Müşteri Frontend Uygulaması GitHub Adresi

<https://github.com/byrmgng/WebFotokopi-Customer-Client>

Ek-2. Satıcı Frontend Uygulaması GitHub Adresi

<https://github.com/byrmgng/WebFotokopi-Seller-Client>