



ONDOKUZ MAYIS ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

VERİTABANI YÖNETİM SİSTEMLERİ

DÖNEM SONU ÖDEVİ

Dönem Sonu Ödev Raporu

ZEYNEP ALTUN

BERİL TÖMAN

ALPER ÖZKUL

ONUR YAVUZ

Danışman

Dr. Öğr. Üyesi İSMAİL İŞERİ

OCAK, 2023

İÇİNDEKİLER

I GİRİŞ

| | |
|--------|---|
| 1 Amaç | 2 |
|--------|---|

II MATERİYAL

| | |
|---|---|
| 2 Nesne-İlişkisel Haritalama | 4 |
| 2.1 C#: Entity Framework | 4 |
| 3 Microsoft SQL Server | 6 |
| 3.1 SQL Server Management Studio (SSMS) | 6 |
| 4 Git | 7 |
| 4.1 GitHub | 7 |

III Yöntem

| | |
|--|----|
| 5 YÖNTEM | 9 |
| 5.1 ER Diyagramları | 9 |
| 5.1.1 UML Notasyonu | 9 |
| 5.1.2 Chen Notasyonu | 10 |
| 5.1.3 İlişki Şeması | 11 |
| 5.2 Proje Arayüzü ve İstenirleri | 12 |
| 5.3 ORM Kullanımı | 29 |
| 5.4 Github Kullanımı | 36 |

| | |
|---|----|
| 5.5 Youtube video ve Github Repository Linkleri | 45 |
|---|----|

IV SONUÇ

| | |
|---------|----|
| 6 Sonuç | 47 |
|---------|----|

Ş E K İ L L E R L İ S T E S İ

| | | |
|----|---|----|
| 1 | Entity Framework çalışma durumları | 5 |
| 2 | UML notasyonu | 10 |
| 3 | Chen notasyonu | 11 |
| 4 | Kullanılan ilişki şeması | 11 |
| 5 | Menü görünümü | 12 |
| 6 | Ürünler formu | 13 |
| 7 | İrsaliye dosyaları | 13 |
| 8 | Müşteri ekleme | 14 |
| 9 | Satış ekranına giriş için kullanıcı adı arayüzü | 15 |
| 10 | Satış ekranına giriş için şifre arayüzü | 15 |
| 11 | Satış form arayüzü | 16 |
| 12 | Veresiye satış | 17 |
| 13 | Müşterinin borç ödeme bilgileri | 18 |
| 14 | Ürün stok ekleme | 18 |
| 15 | Tedarikçi borç bilgileri | 19 |
| 16 | Tedarikçiye yapılan ödeme bilgileri | 20 |
| 17 | Sepete ürün ekleme | 21 |
| 18 | Sepetten ürün silme | 21 |
| 19 | Müşteri borç bilgileri | 22 |

| | | |
|----|--|----|
| 20 | Müşteriye yapılan satışın silinmesi | 23 |
| 21 | Ürün bazlı kar/zarar durum raporu | 24 |
| 22 | Toplu müşteri borç durum raporu | 25 |
| 23 | Müşteri bazlı rapor | 26 |
| 24 | Marketin kar zarar durum raporu | 27 |
| 25 | En çok satandan en az satana ürün listesi | 28 |
| 26 | Stok miktarı 20'nin altında ise uyarı verilmektedir. | 28 |
| 27 | ORM kullanılmadan yazılması gereken örnek kod | 30 |
| 28 | ORM kullanıldığındaysa yazılacak örnek kod | 31 |
| 29 | Projede ORM kullanımı | 32 |
| 30 | Projede ORM kullanımı 2 | 33 |
| 31 | Projede ORM kullanımı 3 | 34 |
| 32 | Projede ORM kullanımı 4 | 34 |
| 33 | Projede ORM kullanımı 5 | 35 |
| 34 | Projeye database tanıtımı | 35 |
| 35 | Veritabanı bağlantısı | 36 |
| 36 | Sistem için giriş ekranı | 36 |
| 37 | Menü, raporlama ve satış form arayüz tasarımları | 36 |
| 38 | Tedarikçiler ve müşteriler form oluşturulması | 37 |
| 39 | Müşteriler form arayüz tasarımı | 37 |
| 40 | Menü formu içerişine diğer formların getirilmesi | 37 |
| 41 | Müşteriler ve menü form arayüz düzenlemesi | 37 |
| 42 | Ürün formu arayüz tasarımı | 38 |
| 43 | Formların arayüz tasarımı | 38 |

| | | |
|----|--|----|
| 44 | Satış ve ürünler form tasarım değişikliği ve fonksiyon tanımlanması | 38 |
| 45 | Tedarikçi form tasarım değişikliği ve fonksiyon tanımlanması | 38 |
| 46 | Tedarikçi formuna borç ödeme fonksiyonu tanımlanması | 38 |
| 47 | Tedarikçi formuna ekle, güncelle, sil ve diğer fonksiyonların tanımlanması | 39 |
| 48 | Form tasarım değişikliği ve müşteri formuna ekle, güncelle, sil fonksiyon tanımlanması | 39 |
| 49 | Satış işlemlerinin gerçekleştirilmesi | 39 |
| 50 | Veritabanı güncellemesi ve satış formunun tamamlanması | 39 |
| 51 | Dosyadan okuma yöntemi ile ürün ekleme fonksiyonlarının tamamlanması | 40 |
| 52 | Borç ödeme form oluşturulması ve müşteri form tasarım değişikliği | 40 |
| 53 | Müşteri satın alımları listeleme ve borç ödeme fonksiyonları | 40 |
| 54 | İrsaliye üzerinden yeni ürün eklenmesi | 40 |
| 55 | Satış formuna giriş için kullanıcı adı ve şifre fonksiyonları eklenmesi | 41 |
| 56 | Satış sonrası stok azalma ve stok uyarı verme işlemleri | 41 |
| 57 | Raporlama formları oluşturulması | 41 |
| 58 | Yapılan satışın silinmesi | 41 |
| 59 | Yapılan satışın silinebilmesi için şifre istenmesi | 42 |
| 60 | Borç ödeme fonksiyonlarının taşınması | 42 |
| 61 | Sepetteki bütün ürünlerin silinmesi | 42 |
| 62 | Müşteri borç ödeme bilgisi geçmişi | 42 |
| 63 | Tedarikçiye ödenen borç bilgisi geçmişi | 43 |
| 64 | Müşteri bazlı rapor fonksiyonları ve Excel'e aktarım | 43 |

| | | |
|----|---|----|
| 65 | Ürün bazlı kar zarar rapor fonksiyonları | 43 |
| 66 | Çok satandan az satana ürün listesi fonksiyonları | 43 |
| 67 | Market kar zarar rapor fonksiyonları | 44 |
| 68 | Müşteri bazlı rapor fonksiyonları | 44 |
| 69 | Ekip üyeleri arasında iş dağılımı | 44 |
| 70 | Ekip üyeleri arasında iş dağılımı | 45 |

ÇİZELGELER LİSTESİ

BÖLÜM: I

GİRİŞ

I

A M A Ç

Dünya üzerinde her geçen gün manuel sistemler yerini bilgisayar bilimlerine ve yazılima bırakmaktadır. Bu sayede manuel sistemlerde var olabilecek hatalar ve eksiklikler yazılım sayesinde azaltılmaktadır. Yazılım ile oluşturulan sistemler bizlere zaman tasarrufu sağlamaktadır. Projede gerçekleştirilen Market Sistemi, el ile tutulan verilerin kaybolması, yazılmlarının unutulması ve hatalı yazımlardan kurtulabilmek amacı ile oluşturulmuştur. Amaçlanan sistem; bir marketi, ihtiyacı olan tüm verilerin tek bir sistemde var olması ve bu veriler kullanılarak yapılacak hesaplamaların her gün düzenli olarak kayıt altında tutulması için tasarlanmıştır. Amaçlanan sistem fonksiyonları kendi içerisinde işlevsel olarak çalışmaktadır. Sistem temelde bir markette satış yapılması, müşterilerin bilgilerinin tutulması, stok bilgilerinin saklanması ve market bilgilerinin raporlanması amacı ile tasarlanmıştır. Gerçeklenmesi amaçlanan sistem olası manuel hataları en aza indirecek, zaman tasarrufu sağlayacak ve el ile yazılan market bilgilerinde harcanacak kağıt israfını engelleyecektir. Böylece marketin süreç yönetimi daha anlaşılır bir sistem kullanarak planlanabilecektir.

BÖLÜM: II

MATERIAL

2

NESNE - İLİŞKİSEL HARITALAMA

Nesne-İlişkisel Haritalama (Object-Relational Mapping: ORM), nesne odaklı olarak ve ritabanından sorgu yapılmasına izin veren bir yöntemdir. Bu yöntemi kullanan yazılım geliştiricileri, yönteme uyan bir kütüphaneye ihtiyaç duyarlar. ORM kütüphaneleri, verileri nesneye dönüştürerek sorgu yapmadan veriler ile doğrudan iletişime geçilmesini sağlar. Programlama dilleri kendine ait çeşitli ORM kütüphanelerine sahiptir. ORM kullanımının pek çok avantajı vardır. Bunlar;

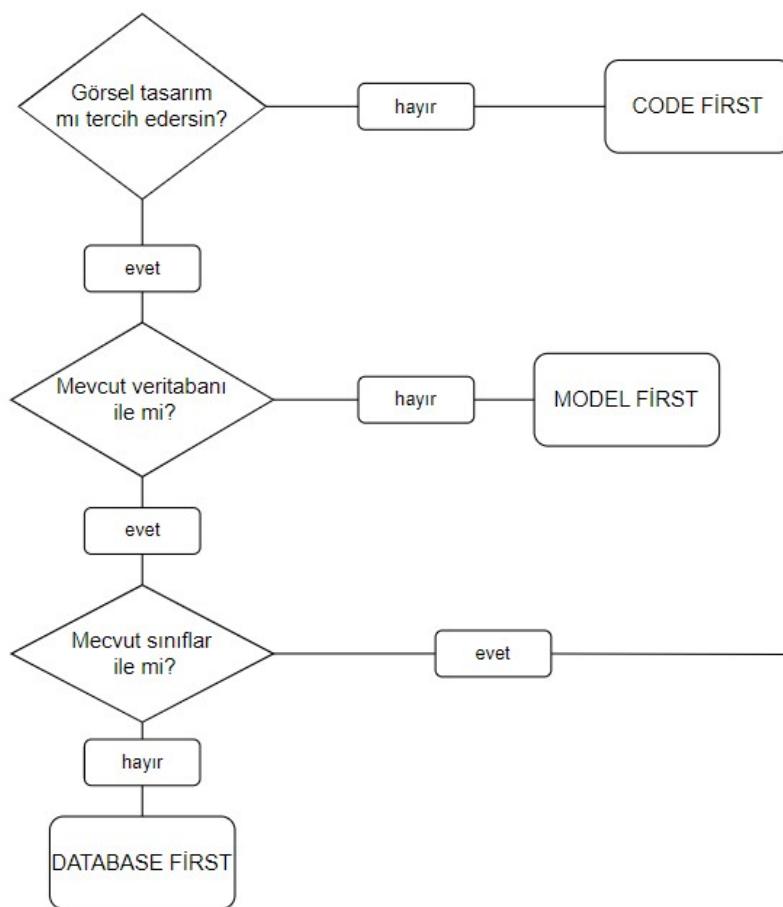
- Nesne yönelimli programlama standartları mevcuttur.
- Hazır sınıflar kullanılabilir, metodlar çağrılabılır.
- Veritabanı platform bağımlılığı yoktur.
- Kod yazma süresi kısalır ve kodun okunabilirliği artar.

2.1 C#: Entity Framework

Entity Framework, 2008 yılından itibaren Microsoft tarafından geliştirilen ORM aracıdır. Yazılım geliştirmek için 4 farklı yöntemi içinde barındırır. Bunlar;

- **Model First:** İlk olarak model sayfası oluşturulup veritabanları bu model üzerinden tasarlanır.
- **Database First:** Önceden oluşturulmuş veritabanlarına model bağlanarak uygulanan bir yöntemdir.
- **Code First (Kod Yazarak):** Sınıflar oluşturularak veritabanları bu sınıflardan türetilmektedir.
- **Code First (Var Olan Veritabanını Kullanma):** Mevcut veritabanının üstüne sınıflar oluşturularak kullanılır.

Çalışma durumlarının açıklamaları ile ilgili örnek [Şekil 1](#)'de gösterilmiştir.



Şekil 1: Entity Framework çalışma durumları

3

MICROSOFT SQL SERVER

Microsoft tarafından 1989 yılında ilk versiyonu piyasaya sunulmuştur. Yazılım geliştirme uygulamalarında kaydedilen tüm verilerin saklandığı veritabanları bulunur. Microsoft tarafından geliştirilen Microsoft SQL Server (MSSQL) veritabanı pazarında oldukça sık bir şekilde kullanılır. Windows işletim sistemlerinde .NET programlama dili kullanılarak oluşturulan web sitesi ve yazılımlarında veritabanı görevi görür.

3.1 *SQL Server Management Studio (SSMS)*

Microsoft tarafından geliştirilen MSSQL, veritabanı için bir editördür. Editör kullanılarak bir çok SQL sorguları yapılabilir. Veri okuma, veri silme, veri analizi, veri ekleme ve raporlama gibi genel işlemler gerçekleştirilebilir. Ayrıca SSMS yardımı ile yeni veritabanları oluşturulabilir, yeni veri tabloları tasarılanabilir ve gereken tüm düzenlemeler yapılabilir.

4

G I T

Proje geliştirilirken yapılan değişiklikleri takip etmek, projeyi internet üzerinde bir depoda saklamak gibi işlevlerin yanında, bir ekip olarak proje geliştirmeye ve değişikliklerin birleştilmesine olanak sağlar. Git, versiyon kontrol sistemidir. Uzak bir depoyu yerel bilgisayara çekip, üzerinde işlemler yapılabilmesini ve yapılan değişikliklerin depoya işlenmesini sağlar. Versiyonlama işlerinde oldukça kullanışlı ve popülerdir.

4.1 GitHub

İçerisinde versiyon kontrol sistemi (Version Control System: VCS) bulunduran, bulut tabanlı bir hizmettir. Yazılım geliştiricilerinin git değişikliklerini takip edebilmelerini oldukça kolaylaştırır. Geliştiriciler için sosyal bir ağ konumundadır.

BÖLÜM: III

YÖNTEM

5

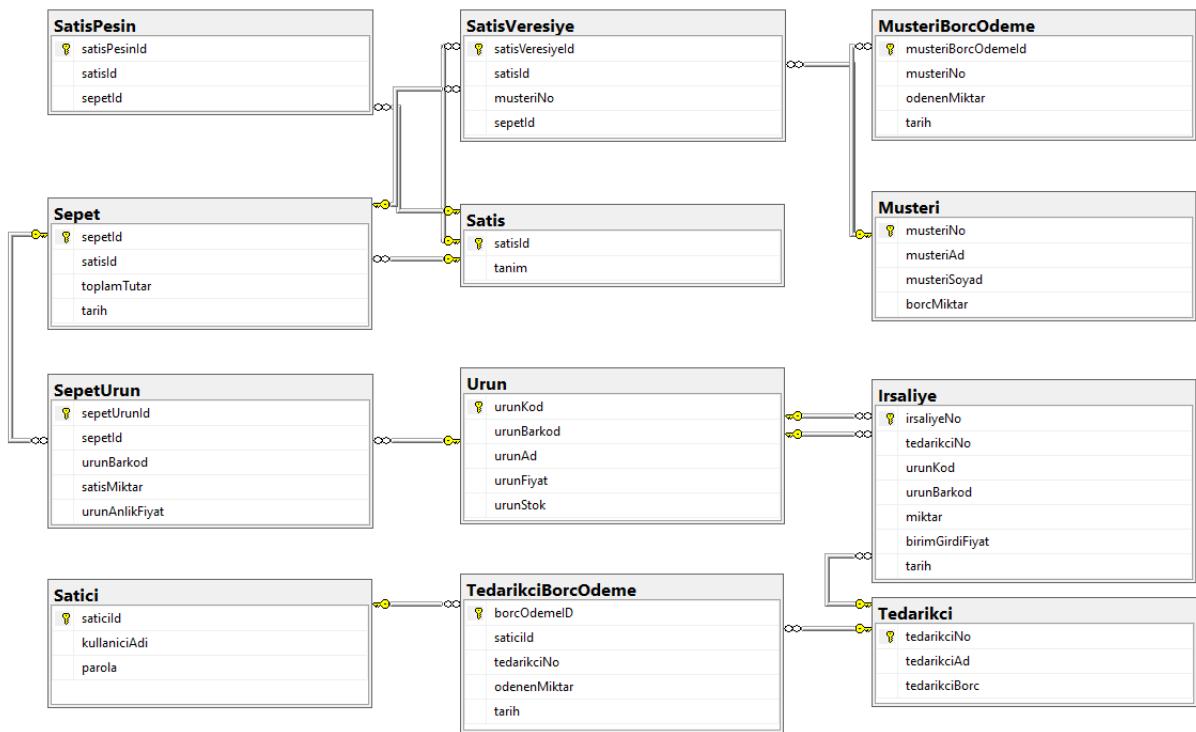
YÖNTEM

5.1 *ER Diyagramları*

Veritabanını oluşturan tabloların ilişkilerini gösteren diyagamlardır. Tasarım amacıyla, ilişkisel model ile oluşturulur. Veritabanından bağımsız veri çözümlemede ve semantik veri modellemede sıkılıkla kullanılırlar.

5.1.1 *UML Notasyonu*

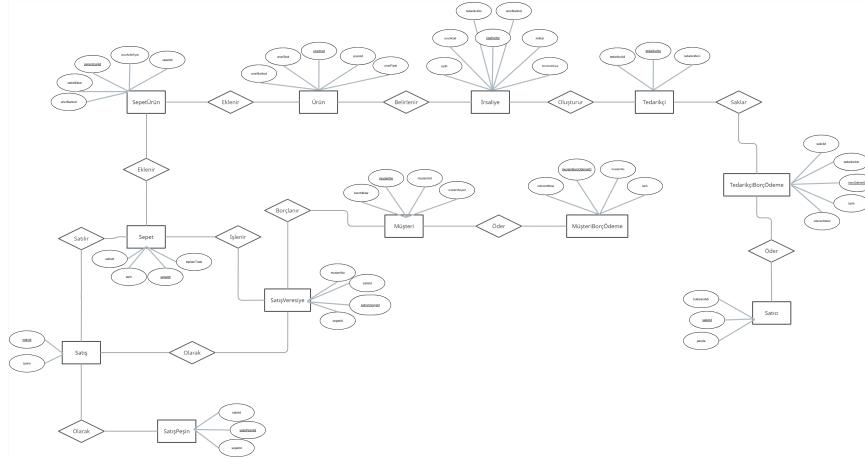
Yazılım süreçlerinde analizden sonra gelir. Bir yazılım sisteminin tanımlanmasında ve yazılımın akışını modellemede kullanılan notasyondur. Genel olarak nesne tabanlı programlama kullanılarak tasarlanır. Bir sınıfı (class) tablolar şeklinde ifade eder. Projede kullanılan UML notasyonu Şekil 2'de gösterilmiştir.



Şekil 2: UML notasyonu

5.1.2 Chen Notasyonu

Veritabanını oluşturan varlıklar ve bu varlıkların birbirleriyle bağlantılarını en iyi açıklayan notasyonlardan biridir. Varlık (entity), ilişki (relationship) ve nitelik (attribute) olmak üzere üç temel bileşeni vardır. Bu bileşenler bazı koşullarda özelleştirilebilir. Sadece kendi özellikleri ile benzersiz şekilde tanımlanamayan, dolayısıyla bir yabancı anahtara ihtiyaç duyan zayıf varlık (weak entity), özelleşmiş bir bileşendir. Projede kullanılan Chen notasyonu Şekil 3'te gösterilmiştir.



Şekil 3: Chen notasyonu

5.1.3 İlişki Şeması

Bir ilişkinin ve niteliklerinin isimleriyle oluşturan bir tanımlamadır. Verilerin mantıklı bir şekilde düzenlenmesini sağlar. Bu tanımlamada ilişkinin, anahtar niteliğinin altı çizilir. Eğer birden fazla anahtar nitelik varsa aralarından en anlamlısı seçilir. Proje gerçekleştirilirken kullanılan ilişki şeması Şekil 4'te gösterilmiştir.

```

Urun(urunKod, urunBarkod, urunAd, urunFiyat, urunStok)
Irsaliye(irsaliyeNo, tedarikciNo, urunKod, urunBarkod, birimGirdiFiyat, miktar, tarih)
Tedarikci(tedarikciNo, tedarikciAd, tedarikciBorc)
TedarikciBorcOdeme(borcOdemeID, saticild, tedarikciNo, odenenMiktar, tarih)
Satıcı(saticild, kullaniciAdi, parola)
Musteri(musteriNo, musteriAd, musteriSoyad, borcMiktar)
MusteriBorcOdeme(musteriBorcOdemeID, musteriNo, odenenMiktar, tarih)
Satis(satisId, tanim)
SatisVeresiye(satisVeresiyeId, satisId, musteriNo, sepetId)
SatisPesin(satisPesinId, satisId, sepetId)
Sepet(sepetId, satisId, toplamTutar, tarih)
SepetUrun(sepetUrunId, sepetId, urunBarkod, satisMiktar, urunAnlikFiyat)

```

Şekil 4: Kullanılan ilişki şeması

5.2 Proje Arayüzü ve İstenirleri

Bu bölümde projede gerçekleştirilmesi istenen maddelerin kodları ve arayüzleri anlatılmaktadır. İstenen 20 maddededen, 18 numaralı madde hariç hepsi gerçekleştirılmıştır. Konu bütünlüğü açısından maddeler alanlarına göre grupperlərlərə incelemiştir. Projenin ana menüsü Şekil 5'te gösterilmektedir.



Şekil 5: Menü görünümü

Madde 1: Her ürün sisteme benzersiz bir ürün kodu ile ürün kodu kullanılarak üretilmiş bir barkod numarası ile tanımlanmaktadır.

Şekil 6'da ürünler formu gösterilmektedir. Ürünler, ürün kodunun sonuna "0" yazılarak üretilmiş barkod numarası ile sisteme tanımlanmaktadır. DataGridView'a ürünlerin listesi ve ritabanından çekilmektedir.

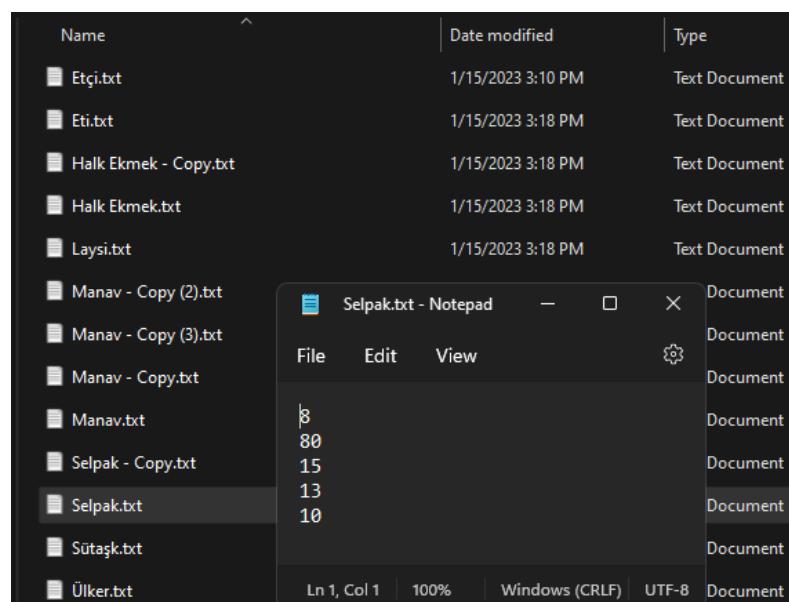
| | Ürün Kod | Ürün Barkod | Ad | Fiyat | Stok |
|---|----------|-------------|----------------|-------|------|
| ▶ | 1 | 10 | Süt 1L | 9.89 | 106 |
| | 2 | 20 | Yumurta | 2 | 291 |
| | 3 | 30 | Ekmek | 4 | 12 |
| | 4 | 40 | Kahve 200gr | 29.9 | 34 |
| | 5 | 50 | Şeker 1KG | 35 | 45 |
| | 6 | 60 | Süt 500ml | 5 | 189 |
| | 7 | 70 | Şeker 2KG | 69.9 | 27 |
| | 8 | 80 | Peynir Blı | 20 | 16 |
| | 9 | 90 | Çikolata | 7.9 | 153 |
| | 10 | 100 | Yoğurt 1KG | 25.9 | 43 |
| | 11 | 110 | Tavuk Göğüs | 75.5 | 225 |
| | 12 | 120 | Tuvalet Kağıdı | 79.9 | 97 |
| | 13 | 130 | Simit | 3 | 95 |
| | 14 | 140 | Çakmak | 5 | 42 |
| | 15 | 150 | Oda Kokusu | 25 | 47 |
| | 16 | 160 | Deterjan 1L | 50.9 | 29 |

Şekil 6: Ürünler formu

Madde 2: Ürünler stoğa farklı tarihlerde farklı ırsaliye numaraları ile eklenmektedir.

Her ırsaliyede ürünün birim girdi fiyatı ve miktarı (kg, adet vs.) değişmektedir. Buna bağlı olarak her ürünün karlılığı satış fiyatı sabit tutulduğu veya zaman içerisinde belli aralıklarla arttırıldığı düşünüldüğünde girdi fiyatına göre değişmektedir.

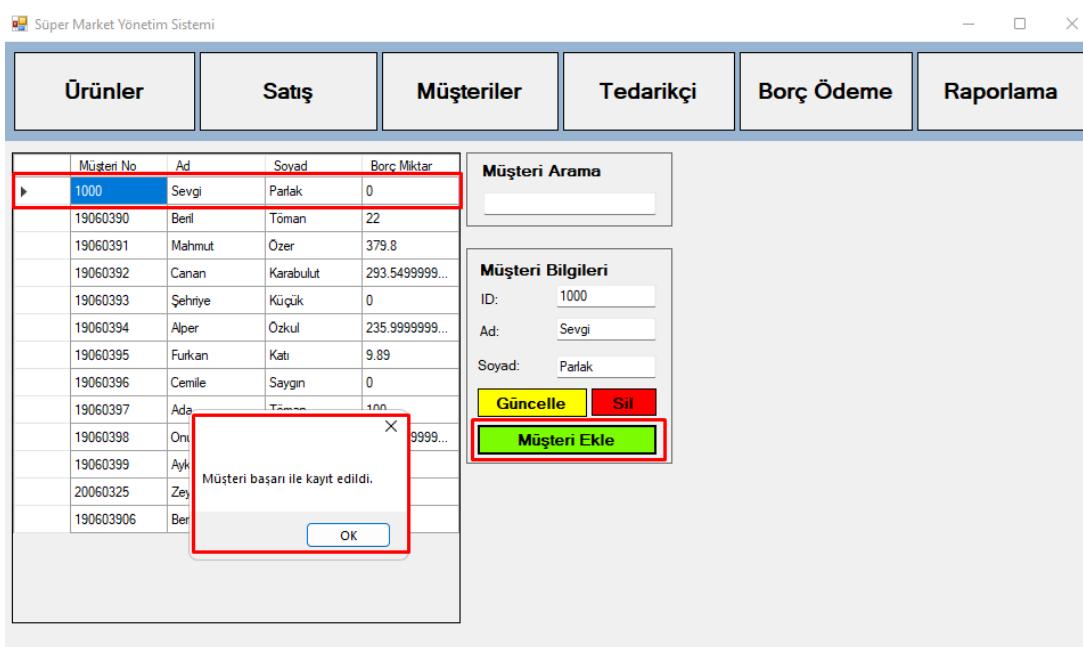
Şekil 7'de ırsaliye dosyaları gösterilmiştir. İrsaliyeden alınan birim girdi fiyatı ile ürünler tablosundaki satış fiyatı farklı olduğu için, zaman içerisinde kar zarar durumu değişmektedir.



Şekil 7: ırsaliye dosyaları

Madde 3: Müşteriler sisteme bir kez tanımlanmaktadır.

Müşteri formu [Şekil 8](#)'de gösterilmiştir. Müşteriler sisteme ancak bir kişinin sahip olabileceği "ID" değeri ile tanımlanmaktadır. Bu sayede müşterilerin birden fazla tanımlanmasının önüne geçilmiştir. Bilgiler doldurulduktan sonra "Müşteri Ekle" butonuna tıklayarak işlem tamamlanır. Veritabanında bu numaraya sahip bir müşteri tanımlıysa hata mesajı vermektedir. [Şekil 8](#)'de gösterilen örnekte "1000 ID" numaralı, Sevgi Parlak isimli bir müşteri tanımlanmıştır ve işlem başarılı mesajı ile karşılaşılmıştır.



Şekil 8: Müşteri ekleme

Madde 4: Yazılımda perakende satış arayüzü bulunmaktadır ve bu arayüze her bir satıcı belirli bir kullanıcı adı ve şifre ile girmektedir.

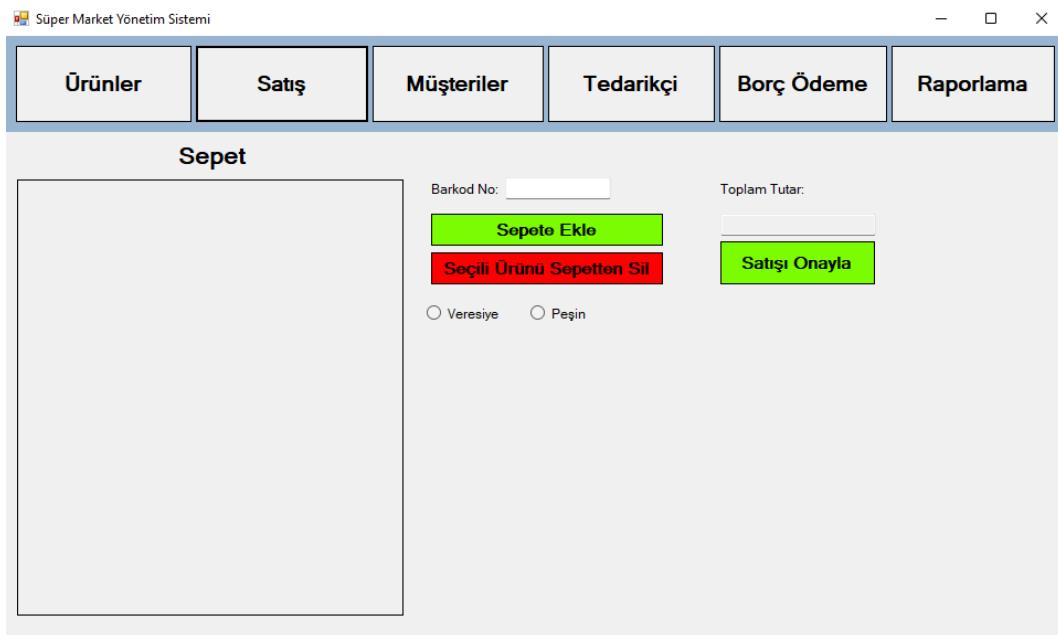
Menü formunda "Satış" butonuna tıklandığı zaman karşımıza [Şekil 9](#)'da gösterilen arayüz gelmektedir. Kutucuğa "admin" kullanıcı adı girildikten sonra karşımıza [Şekil 10](#) gelmektedir. Buradaki kutucuğa da "admin" verisi girildikten sonra yazılım, veritabanında bulunan satıcı tablosundaki verileri kontrol etmektedir. Eşleşme varsa [Şekil 11](#)'de gösterilen satış formuna yönlendirmektedir. Eğer eşleşme yoksa hata mesajı ile kullanıcı bilgilendirilmektedir.



Şekil 9: Satış ekranına giriş için kullanıcı adı arayüzü



Şekil 10: Satış ekranına giriş için şifre arayüzü



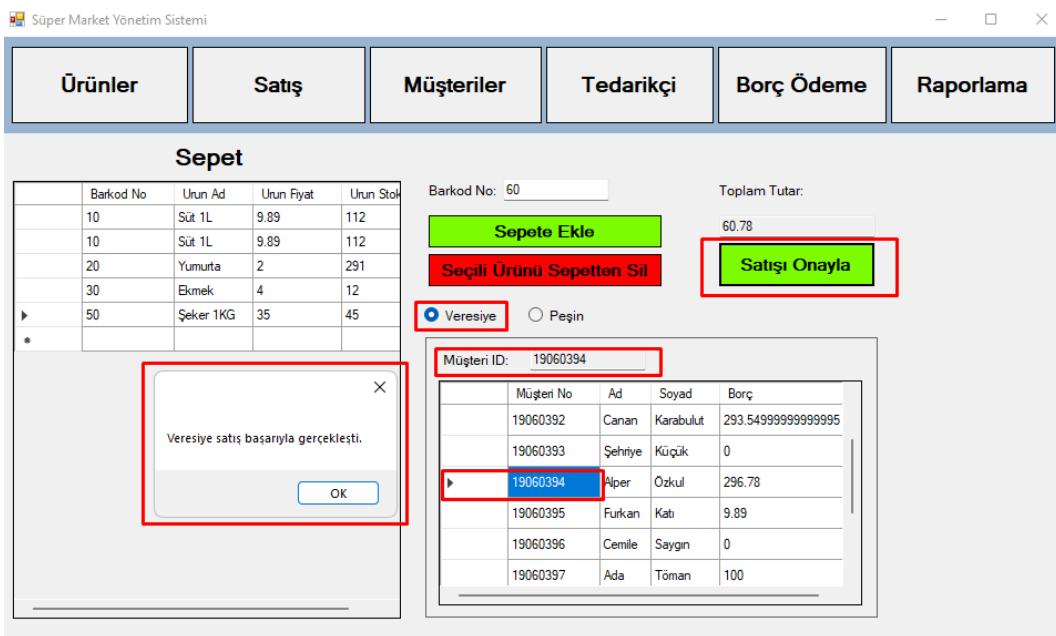
Şekil 11: Satış form arayüzü

Madde 5: Perakende satış arayüzünde cari satış veya peşin satış gerçekleştirmektedir.

Satış formunda karşımıza iki adet radyo butonu(radio button) gelmektedir. Veresiye ya da peşin satış butonları seçilerek satış gerçekleştirilebilmektedir. Veresiye satış butonuna tıklanıldığında müşteri seçilmeli, peşin butonuna tıklanıldığında müşteri seçilmeden "satışı onayla" butonuna tıklanılarak satış gerçekleştirilmektedir. Şekil 11'de satış arayüzünde bulunan butonlar görülmektedir.

Madde 6: Veresiye satışta müşterinin seçilmesi gerekmektedir. Bu sayede müşterinin geri tarihli olarak hangi tarihte hangi ürünü aldığı ve ne kadar borcunun oluştuğu bilgisi elde edilebilmektedir.

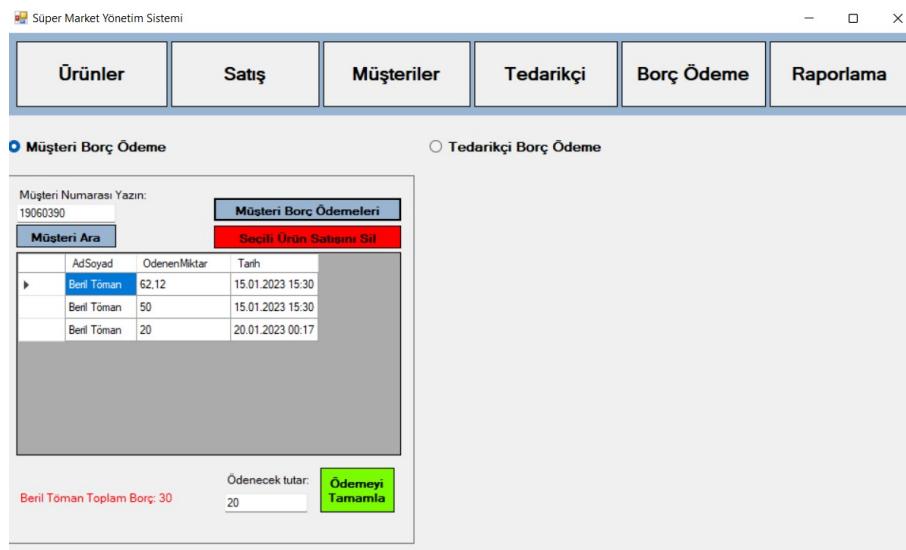
Şekil 12'de görüldüğü gibi veresiye "RadioButton" seçildiyse panel açılmaktadır. Açılan bu panel üzerinden müşteri numarası seçilerek "Satışı Onayla" butonuna tıklanmaktadır. Bu sayede satış işlemi tamamlanmaktadır.



Şekil 12: Veresiye satış

Madde 7: Her müşteriye bir veya birden fazla ürün satılarak farklı zamanlarda satılarak zaman bilgisi ile kaydedilmekte ve borç bakiyesi oluşmaktadır. Müşteri farklı zamanlarda ödeme yaparak borcunu kapatmaktadır. Müşterinin ödemeleri tarih bazlı olarak tutulmaktadır. Herhangi bir anda borç durumu görülebilmektedir.

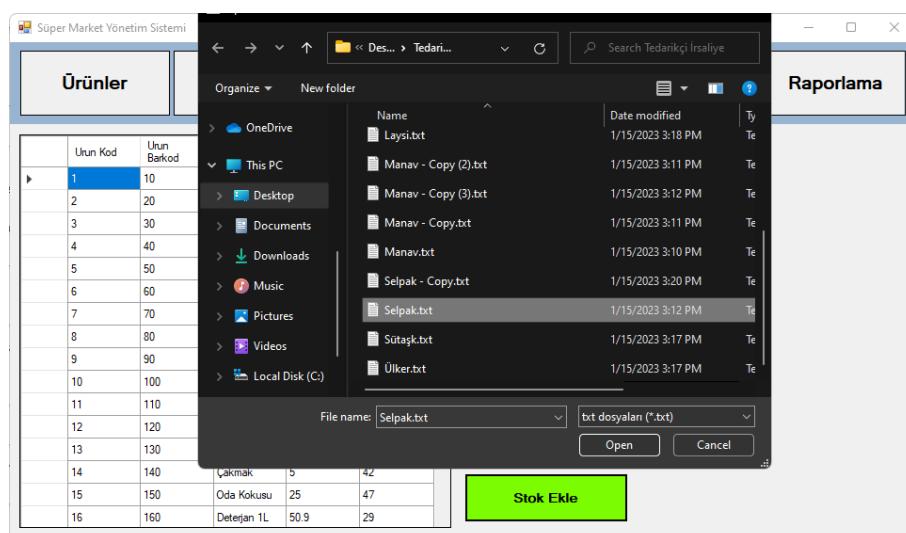
Şekil 13'de görüldüğü üzere anasayfada yer alan "Borç Ödeme" butonuna tıklanıldığında ekrana iki adet seçenek gelmektedir. Müşteri borç ödemelerini görüntüleyebilmek için "Müşteri Borç Ödeme" yazısının yanında bulunan kutucuk işaretlenmelidir. Kutucuk işaretlendiği durumda ekrana bir form açılmaktadır. Bu formda müşteri numarası gerekli kutucuğa yazıldığı ve "Müşteri Borç Ödemeleri" butonuna basıldığında ekranın yer alan DataGridView'a müşteri numarasına ait şu ana kadarki borç ödeme bilgileri bastırılmaktadır. DataGridView sütunları; müşteri adı ve soyadı, ödenen miktar ve tarih bilgilerinden oluşmaktadır. Bu sayede müşteri numarası girilen müşterinin hangi tarihte ne kadar ödeme yaptığı görüntülenebilmektedir.



Şekil 13: Müşterinin borç ödeme bilgileri

Madde 8: Herhangi bir anda markete farklı tedarikçilerden farklı çeşit ve adetlerde ürünler gelmekte ve market sahibi tarafından stoğa işlenmektedir.

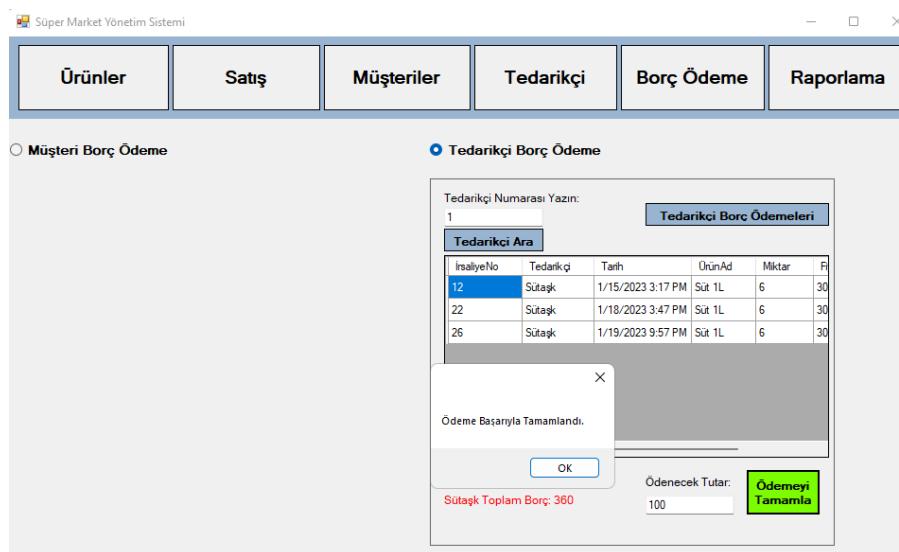
Şekil 7'de bulunan formda "Stok Ekle" butonuna tıklandiktan sonra Şekil 14'te gösterilen ekrandan dosya seçilerek işlem tamamlanır. Dosya doğruysa "Ürünler stoşa işlendi." mesajı ile karşılaşılırken ürün hatalı ise "Lütfen doğru txt dosyasını seçtiğinizden emin olunuz." mesajı ile karşılaşılmaktadır.



Şekil 14: Ürün stok ekleme

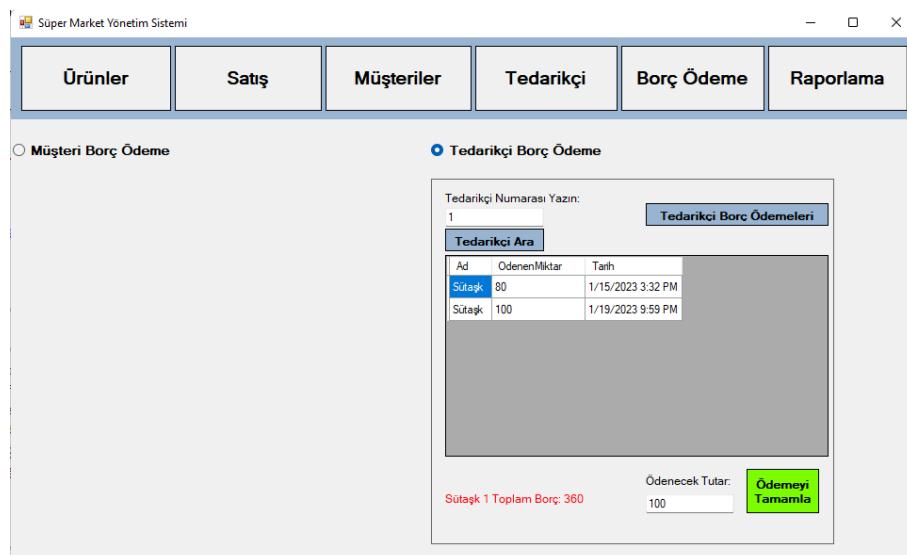
Madde 9: Ürünler stoğa bir txt dosya içerisinde yer alan ırsaliye no, ürün kodu, barkod numarası, girdi birim fiyatı, miktar bilgileri okunarak dosyadan okuma yöntemi ile eklenmektedir. Bu dosyalar tedarikçi firmalar tarafından sağlanmaktadır. Market sahibi tedarikçilere olan borç ve ödeme bilgilerini de sistemde görmek ve takip etmektedir. Ürün girdisi olduğunda bir tedarikçiye borç oluşmakta ve zaman içerisinde tedarikçiye ödeme yapmakta ve bu ödemeyi sisteme islemektedir.

Şekil 7'de ırsaliye dosyaları ve içerikleri gösterilmiştir. Dosyalarda sırasıyla ürün kod, ürün barkod, ürün stok, ürün fiyat ve tedarikçi numarası bilgileri elde edilmektedir. ırsaliye numarası veritabanında birincil anahtar olarak belirlenmiştir. Bu sebeple her dosya için özel ırsaliye numarası ile veritabanına kayıt sağlanmaktadır. Tedarikçi numarası ile tedarikçiye borç bilgisi aktarılmaktadır. Şekil 15'de "Tedarikçi Ara" butonu ile tedarikçiden gelen ırsaliye bilgilerinin gösterimi sağlanmaktadır. "Ödemeyi Tamamla" butonu ile kutucuğa yazılan tutar kadar ödeme yapılmaktadır. Yazılan tutar doğruya "Ödeme Başarıyla Tamamlandı." mesajı ekrana bastırılmaktadır.



Şekil 15: Tedarikçi borç bilgileri

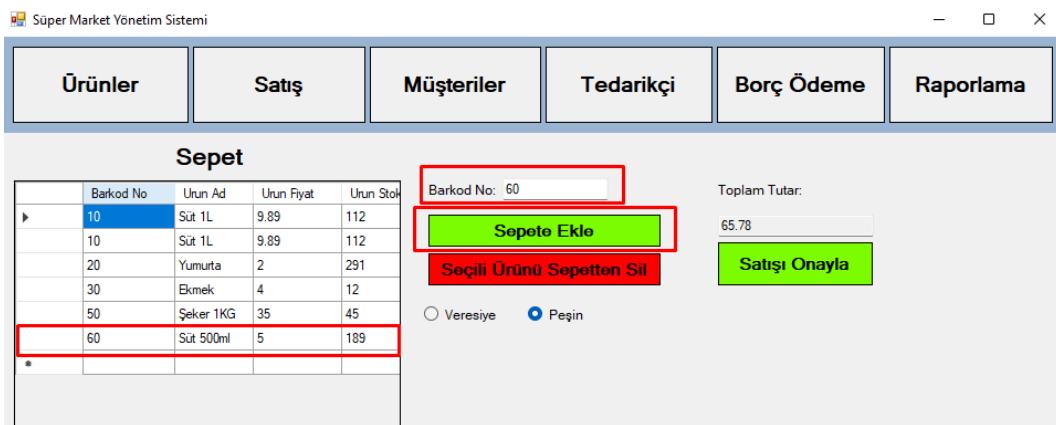
Şekil 16'da bulunan "Tedarikçi Borç Ödemeleri" butonuna tıklandığında, tedarikçiye zaman içerisinde yapılan ödeme bilgileri DataGridView'da gösterilmektedir.



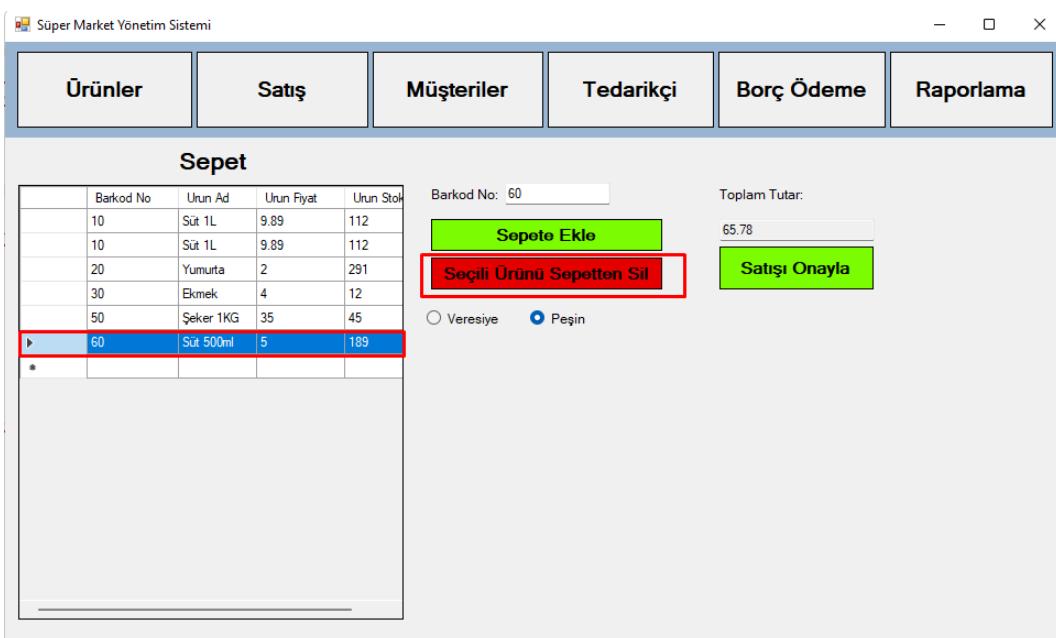
Şekil 16: Tedarikçiye yapılan ödeme bilgileri

Madde 10: Perakende satış ekranında bir müşterinin alışveriş anında aldığı birden fazla ürün barkod okuyucu ile okutulmakta ve alt alta listelenmektedir. Müşteri bir üründen vazgeçerse listeden çıkarılabilir. Buna bağlı olarak müşterinin ödeme yöntemi seçilerek ödeme gerçekleşti ise satış sisteme işlenmelidir. Aynı durum veresiye satış için de mümkün olmalıdır.

Satış formunda bulunan kutucuğa ürün barkod numarası yazarak "Sepete Ekle" butonuna tıklanmaktadır. Barkod numarası veritabanında kontrol edilerek sepete eklenmektedir. Yanlış barkod değeri girildiyse hata mesajı ile karşılaşılmaktadır. Örnek Şekil 17'de gösterilmiştir. Aynı formda bulunan "Seçili Ürünü Sepetten Sil" butonuna tıklayarak ürün silme işlemi gerçekleştirilmektedir. Örnek Şekil 18'de gösterilmiştir.



Şekil 17: Sepete ürün ekleme

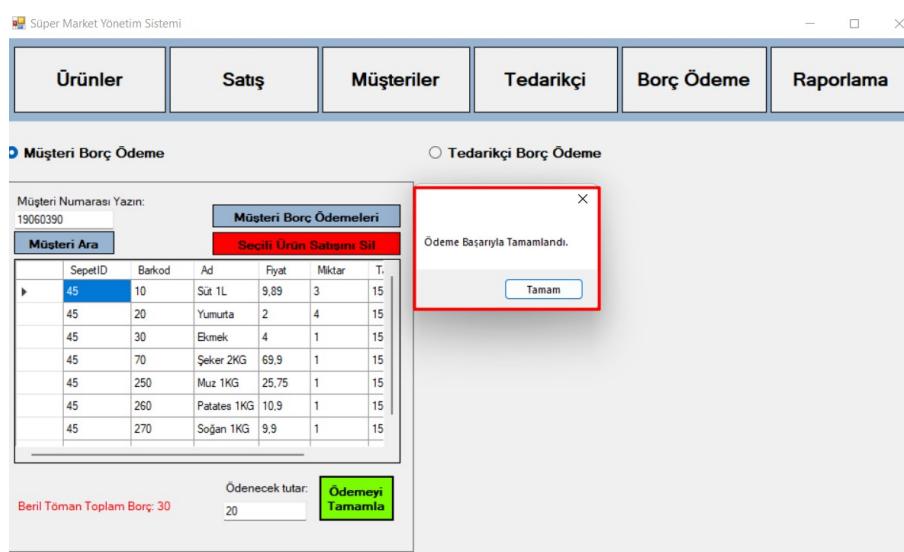


Şekil 18: Sepetten ürün silme

Madde 11: Borç ödeme ekranında müşteri araması yapılarak bulunur ve müşterinin yaptığı satın almalar listelenerek toplam borcu görüntülenir. Ödeme yapılan tutar bir kutucuğa girilecek ödeme işlemi gerçekleştirilir. Bu sayede müşterinin ödemeleri ile borçları kullanılarak müşterinin anlık borç bilgisi ekranada gösterilebilir.

Şekil 19'da görüldüğü üzere anasayfada yer alan "Borç Ödeme" butonuna basıldığında ekranada iki adet seçenek görüntülenmektedir. "Müşteri Borç Ödeme" yazısının yanında yer alan kutucuk işaretlendiğinde bir panel açılmaktadır. Bu panelde yer alan müşteri araması

kutucüğuna müşteri numarası yazılarak arama yapılır. "Müşteri Ara" butonuna tıklanlığında DataGridView'da yazılan müşteri numarasının yaptığı satın alımlar listelenir. Ekranda yer alan DataGridView'ın sütunları Sepet numaraları, Ürün barkodu, Ürün adı, Fiyatı, alım miktarı ve alım tarihinden oluşmaktadır. DataGridView'ın altında yer alan textbox ödemelere bağlı olarak değişmekte ve müşterinin anlık borç bilgisini içermektedir. Panelin altında yer alan "Ödenecek Tutar" isimli metin kutusuna ödenecek tutar girilir "Ödemeyi tamamla" butonuna tıklanılarak ödeme gerçekleştirilir. Ödeme gerçekleştirildiği durumda sayfanın altında yer alan "xxx xxx toplam borç: xxx" metni de düzenlenmektedir.



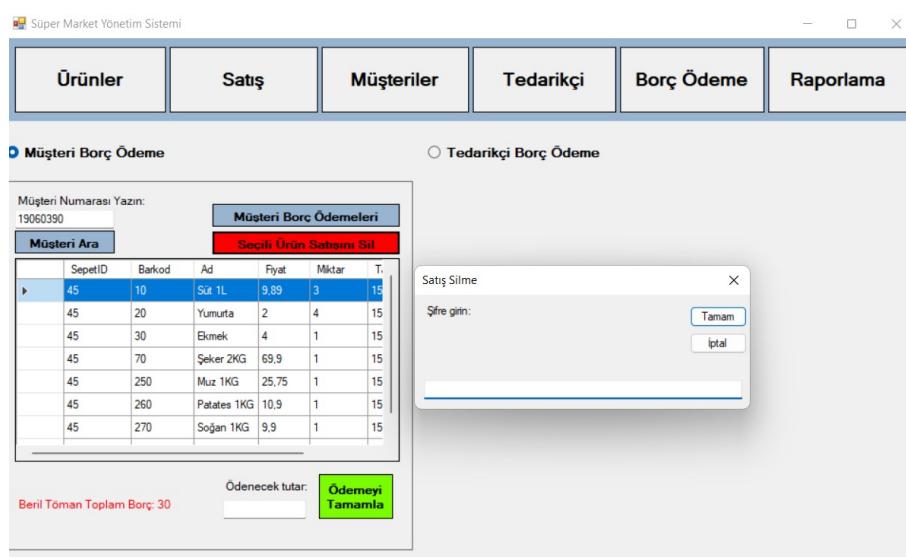
Şekil 19: Müşteri borç bilgileri

Madde 12: Stok giriş ekranında txt dosya seçilerek stoğa işleme butonu ile stoğa işleme gerçekleştirilecektir.

Şekil 7'de bulunan formda "Stok Ekle" butonuna tıklandıktan sonra Şekil 14'te gösterilen ekranın dosya seçilerek işlem tamamlanır. Dosya doğruysa "Ürünler stoğa işlendi." mesajı ile karşılaşılırken ürün hatalı ise "Lütfen doğru txt dosyasını seçtiğinizden emin olunuz." mesajı ile karşılaşılmaktadır.

Madde 13: Sistemden bir müşteriye yapılan satışın silinebilmesi için bir şifre gerekmektedir. Bu şifre bir satış silinmek istendiğinde istenmektedir sadece şifre girildiğinde satış silinebilmektedir.

Şekil 20'de görülebileceği gibi anasayfada yer alan Borç Ödeme ekranında "Müşteri Borç Ödeme" yazısının yanındaki kutucuk işaretlenerek formun açılması sağlanır. Müşteri numarası gerekli kutucuğa yazılıp "Müşteri Ara" butonuna tıklanarak müşteri araması yapılır. Ekranda bulunan DataGridView'a müşteri satışıları listelenmektedir. Sistemden bir müşteriye yapılan satışın silinmesi için DataGridView'daki gerekli satır soldaki boş alandan seçilerek "Seçili Ürün Satışını Sil" butonuna tıklanılır. Bu butona tıklandığına ekrana bir arayüz açılmakta ve şifre girilmesi beklenmektedir. Şifrenin yanlış ya da eksik girilmesi halinde satış silinememektedir. Şifre girilip "Tamam" butonuna tıklanıldığında satış silinecektir.



Şekil 20: Müşteriye yapılan satışın silinmesi

Madde 14: Raporlama arayüzünde ürün bazlı kar zarar durum raporları ve grafikleri verilebilmelidir.

Şekil 21'de görülebileceği üzere anasayfada yer alan "Raporlama" butonuna tıklanıldığında ekrana 6 adet raporlama butonu gelmektedir. "Ürün Bazlı Kar Zarar Durum Raporu"

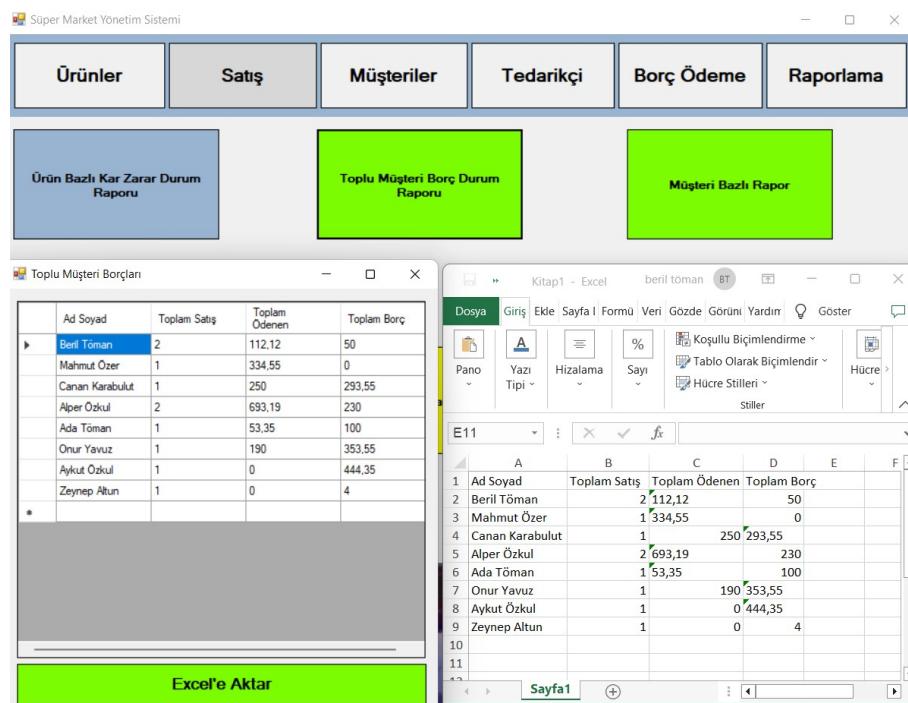
isimli butona tıklanlığında ekranda yeni bir form açılmakta ve bu formun içerisinde bir DataGridView gösterilmektedir. DataGridView sütunlarında ürün barkodu, ürün adı, bu ürün-den toplam kaç adet satış yapıldığını gösteren toplam satış, tedarikçilerden kaç adet alım yapıldığını gösteren toplam alış ve toplam ürün başına kar/zar durumunu görüntüleyebil-mek için toplam kar/zarar sütunu bulunmaktadır. Market sisteminde kayıtlı bütün ürünler bu DataGridView'da bulunmakta ve listelenmektedir.

| Ürün Barkod | Ürün Ad | Toplam satış | Toplam alış | Toplam Kar Zarar |
|-------------|--------------------|--------------|-------------|------------------|
| 10 | Süt 1L | 39,56 | 180 | -140,44 |
| 20 | Yumurta | 18 | 20 | -2 |
| 30 | Ekmek | 56 | 60 | -4 |
| 40 | Kahve 200gr | 119,6 | 90 | 29,6 |
| 50 | Şeker 1KG | 210 | 0 | 210 |
| 60 | Süt 500ml | 50 | 0 | 50 |
| 70 | Şeker 2KG | 489,3 | 0 | 489,3 |
| 80 | Pepete 8'l | 100 | 195 | -95 |
| 90 | Çikolata | 86,9 | 300 | -213,1 |
| 100 | Yoğurt 1KG | 233,1 | 0 | 233,1 |
| 110 | Tavuk Göğüs 1KG | 679,5 | 1000 | -320,5 |
| 120 | Tuvalet Kağıdı 8'l | 0 | 2250 | -2250 |
| 130 | Simit | 21 | 40 | -19 |
| 140 | Çeşmeli | 25 | 40 | -15 |
| 150 | Oda Kokusu | 75 | 0 | 75 |
| 160 | Deterjan 1L | 50,9 | 0 | 50,9 |
| 170 | Yumuşatıcı 1L | 204,5 | 0 | 204,5 |

Şekil 21: Ürün bazlı kar/zarar durum raporu

Madde 15: Toplu müşteri borç durum raporu (pdf, word, excel) alınabilmelidir. (Müşteri Adı Soyadı, Toplam Satış, Toplam Ödeme, Toplam Kalan)

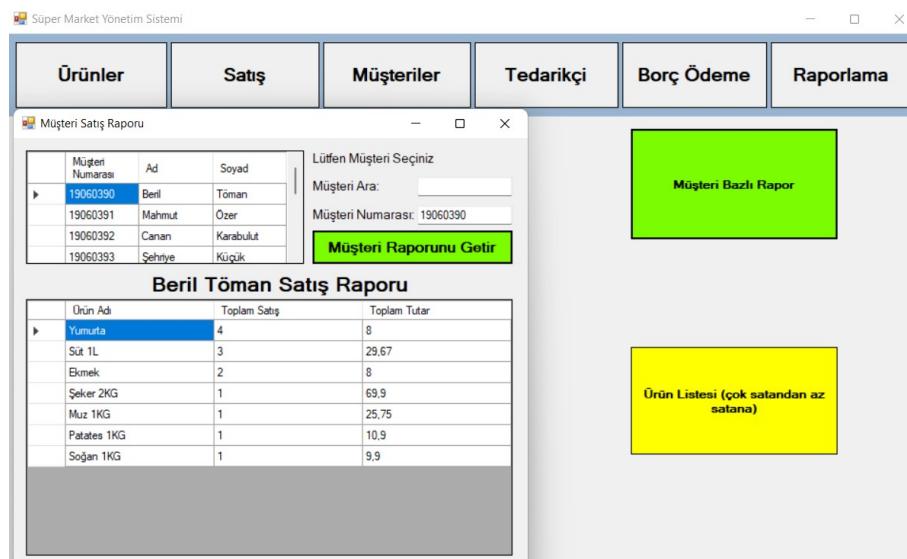
Şekil 22 'de görülebileceği gibi raporlama arayüzü butonuna tıklanıldığı zaman karşımıza 6 adet raporlama butonu çıkmaktadır. "Toplu Müşteri Borç Durum Raporu" butonuna tıklanıldığında karşımıza müşterilerin adı soyadı, toplam alım sayıları (toplam satış), toplam ödeme ve toplam kalan sütunlarından oluşmuş bir DataGridView gelmektedir. Bu DataGridView, Excel olarak alınmak istenilirse DataGridView'ın altında yer alan "Excel'e Aktar" butonu ile bilgisayarımızda bulunan Excel uygulaması açılmakta ve DataGridView'da bulunan veriler bu Excel dosyasında görüntülenebilmektedir.



Şekil 22: Toplu müşteri borç durum raporu

Madde 16: Müşteri Bazlı Rapor -> Satılan Ürün bazlı gruplama yapılacaktır. Raporda bir müşterinin satın almalarına bağlı olarak Ürün Adı, Toplam Satış Miktarı , Toplam Tutar

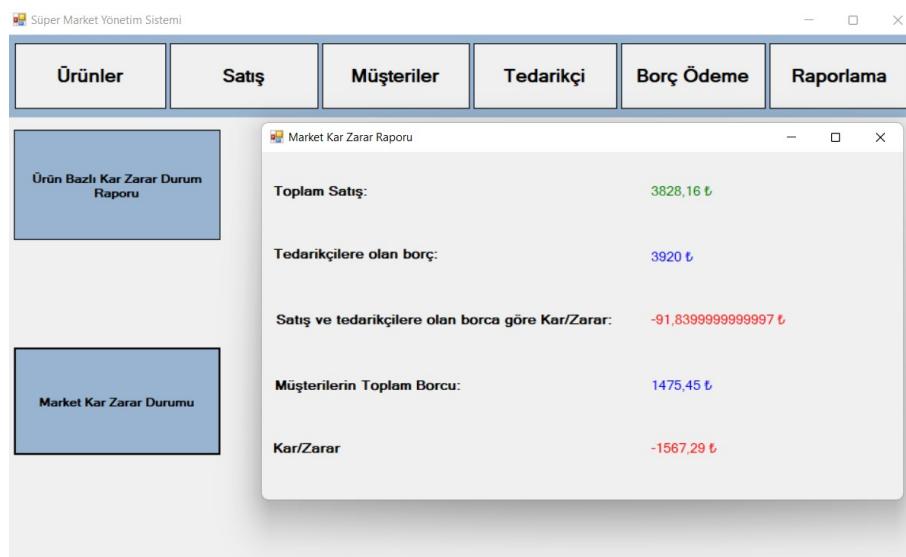
Şekil 23'te görüldüğü gibi raporlama arayüzünde bulunan "Müşteri Bazlı Rapor" butonuna tıkladığımızda "Satış Raporu" kısmının boş olduğu bir form açılmaktadır. Bu formun üst kısmında yer alan müşteri listesinden müşteri seçilerek ya da müşteri numarası kutucuğa girilerek müşteri araması yapılmaktadır. "Müşteri Raporunu Getir" butonuna tıklanıldığında numarası yazılan müşteriye ait, yapılan her satış listelenmektedir. Ürün adı, toplam bu ürününden kaç adet satın aldığı ve toplam ne kadar para ödediği DataGridView'da görüntülenebilmektedir.



Şekil 23: Müşteri bazlı rapor

Madde 17: Marketin ilk satış yaptığı günden anlık bir duruma kadar olan süreçte kar zarar durumu raporlanabilmelidir. Tedarikçilere olan borç, yapılan toplam satış bilgisi anlık olarak görülebilmelidir.

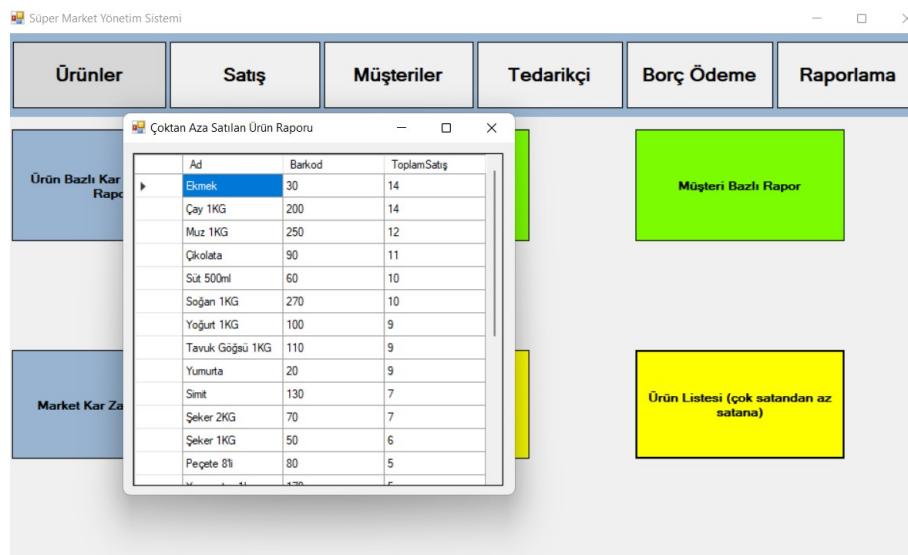
Şekil 24 'de anasayfada bulunan raporlama arayüzü sekmesine tıklanıldığında raporlama başlığı altında incelenen 6 adet buton bulunmaktadır. "Market Kar Zarar Durumu" butonuna tıklanıldığında marketin ilk satış yaptığı günden şu anki duruma kadar olan süreç içerisinde yaptığı toplam satış fiyatı, tedarikçilere olan borcu ve bu bilgiler kullanılarak oluşturulan kar zarar bilgisi görüntülenmektedir. Kar durumunda sayılar yeşil ile gösterilirken, zarar durumunda sayılar başında "-" işaretini ve kırmızı renkle gösterilmektedir. Bulunan borç bilgisi müşterilerin markete olan borç bilgilerini içermemişti için müşterilerin markete olan toplam borçları da dahil edilerek oluşturulmuş kar/zarar bilgileri de formun son satırında görüntülenebilmektedir.



Şekil 24: Marketin kar zarar durum raporu

Madde 19: En çok satandan en az satana doğru ürünler listelenebilmelidir.

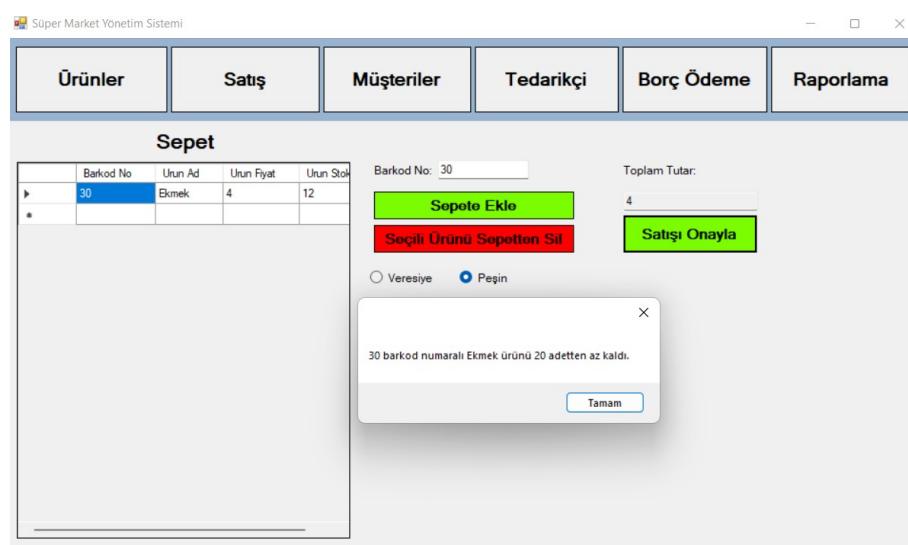
Şekil 25'te görüldüğü üzere anasayfamızda yer alan raporlama arayüzü sekmesine tıklanıldığında karşımıza 6 adet ana başlık gelmektedir. Bunlardan bir tanesi "Ürün Listesi(Çok satandan az satana)" olarak Şekil 25'te görülmektedir. Bu butona tıklandığında, sistemde kayıtlı ürünlerin satış sayıları en çok satandan en az satana doğru ürünler listelenmiştir. Listedede ürünlerin adı, barkod numaraları ve toplam satış miktarları yer almaktadır.



Şekil 25: En çok satandan en az satana ürün listesi

Madde 20: Stok miktarı belli bir eşigin altına indiğinde perakende satış ekranında uyarı verilebilmelidir.

Satışı yapılan bir ürünün stok miktarı 20'nin altına düşince ekrana barkod numarası ve isim yazılarak bilgilendirme yapılır. Şekil 26'da ekrana "30 barkod numaralı Ekmek ürünü 20 adetten az kaldı" mesajı gösterilerek kullanıcı bilgilendirilmiştir.



Şekil 26: Stok miktarı 20'nin altında ise uyarı verilmektedir.

5.3 *ORM Kullanımı*

Nesne-İlişkisel Haritalama (Object-Relational Mapping: ORM), yazılımcının SQL sorguları ile yaptığı işlemleri SQL sorgularına ihtiyaç duymadan gerçekleştirebilmesine olanak sağlar. ORM kullanım amacı yazılımdaki objeler ile veritabanı arasında kurulacak bir bağlantı sayesinde veritabanı ve yazılımımızın eşleşmesini sağlamaktadır. Nesne yönelik programlama dilleri (Object Oriented Programming Languages) ile ilişkisel veri tabanı sistemlerini (Relational Database Systems) birbirine bağlayan (Mapping) kavramın adıdır. Her programlama dili için ORM çerçeveleri (framework) değişiklik göstermektedir.

- C#: Entity Framework, Dapper, ECO, XPO, Norm
- Java: Hibernate, Ebean, Torque, JPA, MyBatis
- Php: CakePHP, CodeIgniter, RedBean, Doctrine, Propel, PdoMap
- Python: Django, South, Storm

ORM kullanmanın avantajları kısaca şunlardır;

- ORM sayesinde program veritabanı uygulamasından bağımsız çalışabilmektedir.
- Veri tabanı sorgularını öğrenmeye gerek kalmadan işlem yapılabilir.
- Yarat, oku, güncelle, sil (create, read, update, delete: CRUD) işlemleri kolaylıkla yapılır.
- Çoğu ORM aracı açık kaynaklı yazılımlar sınıfında olduğu için maliyeti düşüktür.
- Bir yazılımcının kod yazma süresini azaltır.
- Okunabilirlik SQL sorgularına göre daha rahattır.

SQL sorguları kullanılarak yazılan örnek bir C# kodu Şekil 27 'de gösterilmiştir. Aynı kodu ORM kullanarak yazdığımızda ise Şekil 28 'de görüldüğü üzere daha okunabilir ve kısa bir şekilde istediğimiz işlemleri gerçekleştirebiliyoruz.

```
3  public class Customer
4  {
5      public string Name { get; set; }
6      public int Age { get; set; }
7      public int row_id { get; set; }
8  }
9  public static class DBHelper
10 {
11     public static List<Customer> RetrieveAllCustomers()
12     {
13         DataTable data = new DataTable();
14         List<Customer> customers = new List<Customer>();
15
16         using (SqlConnection conn = new SqlConnection(GetConnectionString()))
17         {
18             conn.Open();
19             if (conn.State == ConnectionState.Open)
20             {
21                 using (SqlCommand cmd = conn.CreateCommand())
22                 {
23                     cmd.CommandText = "SELECT name, age, row_id FROM customer";
24
25                     DataReader reader = cmd.ExecuteReader();
26                     data.Load(reader);
27
28                     foreach (DataRow row in data.Rows)
29                     {
30                         customers.Add(new Customer { Name = row.Field<string>("NAME"),
31                             | Age = row.Field<int>("AGE"), row_id = row.Field<int>("ROW_ID") });
32                     }
33                 }
34             }
35         }
36         return customers;
37     }
38 }
```

Şekil 27: ORM kullanılmadan yazılmazı gereken örnek kod

```
3 [Table(Name="Customer")]
4 public class Customer
5 {
6     [Column(IsPrimaryKey = true)]
7     public int row_id { get; set; }
8
9     [Column]
10    public string Name { get; set; }
11
12    [Column]
13    public int Age { get; set; }
14
15 }
16 public class MyDatabase : DataContext
17 {
18     public Table<Customer> customerTable;
19     public MyDatabase(string connection) : base(connection) { }
20 }
21
22 public static class DBHelper
23 {
24     public static List<Customer> RetrieveAllCustomers()
25     {
26         MyDatabase db = new MyDatabase(GetConnectionString());
27
28         var customers = db.customerTable.Select(row => row);
29
30         return customers.ToList();
31     }
32 }
```

Şekil 28: ORM kullanıldığında yazılacak örnek kod

ORM modelleme yaklaşımları 3 ana başlık altında incelenmektedir.

1. Veritabanı Öncelikli Yaklaşım (Database First: DB first): Bu yaklaşımda oluşturulacak olan modeller hazırda var olan veritabanı üzerinde tanımlanır.
2. Kod Öncelikli Yaklaşım (Code First): yazılımcının veritabanı ile olan ilişkisinin minimalize edildiği yaklaşımıdır. Yazılımcı veritabanında oluşturulması beklenen sınıfları, kodları ile oluşturur. Kod öncelikli yaklaşımında sınıflar, veritabanındaki tablo ve kolonlar anlamına gelmektedir.

3. Model Öncelikli Yaklaşım (Model First): Kod yazmadan ya da veritabanı ile fiziksel bir bağlantı kurmadan modeller üzerinden veritabanı oluşturulur. Hem veritabanı hem de kodlar otomatik olarak oluşturulur.

Projemizde veritabanı öncelikli yaklaşım kullanarak, veritabanı ve veritabanındaki tabloları oluşturduk. C# programlama dilinde kullanılabilen ORM çerçevelerinden Entity Framework kullandık. Proje ilerleme aşamalarında SQL sorguları kullanmadığımız için projenin birçok alanında ORM örnekleri görebilmek mümkündür. Bunlardan bazıları Şekil 29, 30, 31, 32 ve 33'de görüntülenebilmektedir.

Şekil 29'da görülen kod parçasığında "sepet ürün" listesi veritabanında oluşturulan tablo dan çekilerek; ürün isimleri, barkod numaraları veri ızgarası görünümüne (DataGridView) yazıdırılmıştır. Ardından toplam satışı belirlemek için veritabanı işlemleri yerine tek satırda toplama metodu (SUM) kullanılarak toplam satış miktarı da bulunup DataGridView üzerine yazıdırılmıştır.

```
List<SatisVeresiye> sorgu = db.SatisVeresiyes.OrderBy(p => p.musteriNo).ToList();
dataGridView1.DataSource = sorgu;

List<MusteriBorcOdeme> musterisorgu = db.MusteriBorcOdemeler.OrderBy(p => p.musteriNo).ToList()

foreach (var veresiye in sorgu)
{
    if (veresiye.musteriNo == mNo)
    {
        toplamSatis += 1;
        var query = db.Musteris.Find(mNo);
        string isim = query.musteriAd + " " + query.musteriSoyad;

        foreach (DataGridViewRow row in dataGridView1.Rows)
        {
            if (row.Cells[0].Value.ToString().Equals(isim))
            {
                row.Cells[1].Value = toplamSatis;
                break;
            }
        }
    }
    else
    {
        toplamSatis = 0;
        mNo = veresiye.musteriNo;
        toplamSatis += 1;
    }
}
```

Şekil 29: Projede ORM kullanımı

Şekil 30'da kullanılan "list" sınıfı diziler ve veri yapıları için kullanılmaktadır. Yine bu örnekte de veritabanındaki tablo verilerini ORM kullanımı sayesinde SQL sorguları ile uğraşmadan DataGridView üzerinde gösterimini sağlamış bulunmaktayız. Veritabanımızda bulunan Müşteri tablosunu "db.Musteris" kod parçasıyla ile sisteme tanıtmış ve ardından "Find" metodu ile müşteri numarası kutucuğa girilmiş olan müşteriyi yazılımın bulması sağlanmıştır.

```
private void UrunListCoktanAza_Load(object sender, EventArgs e)
{
    try
    {
        var sorgu = from sepetUrun in db.SepetUruns
                    group sepetUrun by sepetUrun.urunBarkod into s
                    select new
                    {
                        Ad = db.Uruns.Where(x => x.urunBarkod == s.Key).
                            Select(x => x.urunAd).FirstOrDefault(),
                        Barkod = s.Key,
                        ToplamSatış = s.Sum(x => x.satisMiktar),
                    };
        DGVListelemeUrun.DataSource = sorgu.OrderByDescending(x => x.ToplamSatış).ToList();
    }
    catch
    {
        MessageBox.Show("Lütfen kontrol edin");
    }
}
```

Şekil 30: Projede ORM kullanımı 2

Şekil 31 'de veritabanında bulunan tablolardan sepeturun, veresiye ve sepet; var anahtarı içerisine eklenmiştir. Metin kutucuğuna girilmiş olan müşteri numarası, veresiye tablosu içerisinde yer alan müşteri numarasına eşit olan değerleri bulabilmek adına "where vere-siye.musteriNo== mNo" metodu kullanılmıştır.

```

int mNo = Convert.ToInt32(TB_MusteriNo.Text);
var sorgu = from sepeturun in db.SepetUruns
            join veresiye in db.SatisVeresiyes
            on sepeturun.sepetId equals veresiye.sepetId
            join sepet in db.Sepets
            on veresiye.sepetId equals sepet.sepetId
            join urun in db.Uruns
            on sepeturun.urunBarkod equals urun.urunBarkod
            where veresiye.musteriNo == mNo
            select new
            {
                SepetID = sepet.sepetId,
                Barkod = urun.urunBarkod,
                Ad = urun.urunAd,
                Fiyat = sepeturun.urunAnlikFiyat,
                Miktar = sepeturun.satisMiktar,
                Tarih = sepet.tarih,
            };
DGV_Musteri_Borc.DataSource = sorgu.ToList();

var musteri = db.Musteris.Find(mNo);
if(musteri == null)
{
    MessageBox.Show("Böyle bir müşteri bulunmamaktadır.");
}
else
{
    musteri_bilgi.Text = musteri.musteriAd + " " + musteri.musteriSoyad +
        " Toplam Borç: " + musteri.borcMiktar.ToString();
    musteri_bilgi.Visible = true; musteri_bilgi.ForeColor = Color.Red;
}

```

Şekil 31: Projede ORM kullanımı 3

Şekil 32'de DataGridView içerisinde veritabanında bulunan tedarikçi tablosundan veriler listenecektir. "db.Tedarikcis" databasedeki tedarikçi tablosunun sayfamızda okunmasına olanak sağlar.

```

private void Tedarikciler_Load_1(object sender, EventArgs e)
{
    DGV_Tedarikciler.DataSource = db.Tedarikcis.ToList();
    DGV_Tedarikciler.Columns[3].Visible = false; DGV_Tedarikciler.Columns[4].Visible = false;
    DGV_Tedarikciler.Columns[0].HeaderText = "Tedarikci No";
    DGV_Tedarikciler.Columns[1].HeaderText = "Tedarikci Ad";
}

```

Şekil 32: Projede ORM kullanımı 4

Şekil 33'de metin kutucuklarına girilen değerler "musteri" isimli bir değişkende tutulmaktadır. Ardından gelen "db.Musteris.Add(musteri);;" kod parçaslığında musteri değişkeninde

tutulan bu değerleri veritabanındaki Müsteri tablosuna ekleme işlemi yapılmıştır. Yine bu bölümde bulunan "db.SaveChanges();" database üzerinde yapılan değişikliklerin kaydedilmesini sağlamaktadır. Veritabanı tablomuzdaki verileri DataGridView'da listelemek için de "DGVMusteriler.DataSource = db.Musteris.ToList();" kod satırı kullanılmıştır. Müsteri formunda bulunan DataGridView'ın ismi "DGVMusteriler" olarak tanımlanmıştır. Her formda DataGridView ismi değişiklik göstermektedir.

```
private void BtnYeniMusteri_Click(object sender, EventArgs e)
{
    Musteri musteri = new Musteri();
    musteri.musteriAd = TBMusteriAd.Text;
    musteri.musteriSoyad = TBMusteriSoyad.Text;
    musteri.borcMiktar = 0;
    db.Musteris.Add(musteri);
    db.SaveChanges();
    MessageBox.Show("Müşteri başarı ile kayıt edildi.");
    DGVMusteriler.DataSource = db.Musteris.ToList();
    TextBoxTemizle();
}
```

Şekil 33: Projede ORM kullanımı 5

Projede yer alan tabloları kod parçacığımıza ekleyebilmek için sayfalarda database tanıtımı yapmamız gerekmektedir. Sayfamızda veritabanı ile işlemler yapabilmek amacı ile şekil 34 'te görüldüğü üzere veritabanını sayfamıza tanıttık.

```
static DataTable dt = new DataTable();
MarketManagementSystemEntities1 db = new MarketManagementSystemEntities1();
1 başvuru
```

Şekil 34: Projeye database tanıtımı

Proje Entity Framework kullanılarak oluşturulmuştur. Veritabanı öncelikli yaklaşım ile oluşturulan veritabanı ve tabloların ardından, tablolar kullanılarak yapılacak olan işlemlerin her birinde ORM kullanım ile karşılaşılmaktadır. Bu sebeple ORM kullanımını bu örnekler

ile anlatılmıştır. Projenin diğer sınıflarına bölüm 5.5’de bulunan GitHub linkinden ulaşılabilir mekte ve ORM kullanımlarına daha detaylı bir şekilde erişilebilmektedir.

5.4 Github Kullanımı

Şekil 35’té gösterilen git işleminde (commit) veritabanı bağlantısı ve ürün listeleme işlemi gerçekleştirildi.



Şekil 35: Veritabanı bağlantısı

Şekil 36’da gösterilen git commitinde, market yönetim sistemi için giriş ekranı tasarımı ve fonksiyonları gerçekleştirildi.



Şekil 36: Sistem için giriş ekranı

Şekil 37’de gösterilen git commitinde; Menü, Raporlama ve Satış formları oluşturuldu. Formların arayüz tasarımları gerçekleştirildi.



Şekil 37: Menü, raporlama ve satış form arayüz tasarımı

Şekil 38’de gösterilen git commitinde, Tedarikçiler ve Müşteriler formları oluşturuldu.



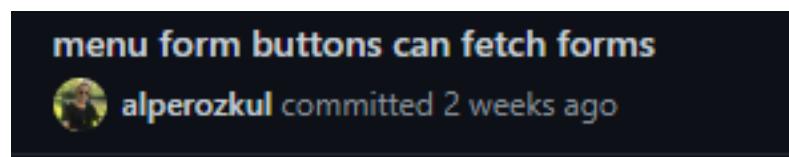
Şekil 38: Tedarikçiler ve müşteriler form oluşturulması

Şekil 39'da gösterilen git commitinde, Müşteriler formunun arayüz tasarımlı gerçekleştirildi.



Şekil 39: Müşteriler form arayüz tasarımı

Şekil 40'ta gösterilen git commitinde, butonlar aracılığıyla formların ekrana getirilme işlemi gerçekleştirildi.



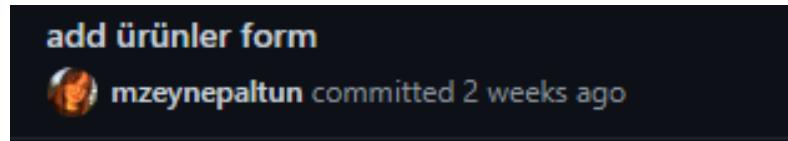
Şekil 40: Menü formu içerisinde diğer formların getirilmesi

Şekil 41'de gösterilen git commitinde, Müşteriler ve Menü formları düzenlendi.



Şekil 41: Müşteriler ve menü form arayüz düzenlemesi

Şekil 42'de gösterilen git commitinde, Ürün formu oluşturuldu. Formun arayüz tasarımları gerçekleştirildi.



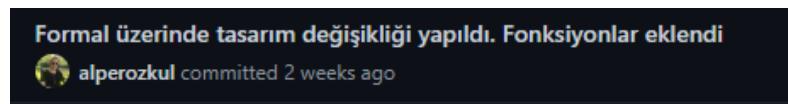
Şekil 42: Ürün formu arayüz tasarımlı

Şekil 43'te gösterilen git commitinde, formların arayüz tasarımları güncellendi.



Şekil 43: Formların arayüz tasarımlı

Şekil 44'te gösterilen git commitinde, Satış ve Ürünler form tasarımları güncellendi ve fonksiyonlar eklendi.



Şekil 44: Satış ve ürünler form tasarım değişikliği ve fonksiyon tanımlanması

Şekil 45'de gösterilen git commitinde, Tedarikçi form tasarımları güncellendi ve fonksiyonlar eklendi.



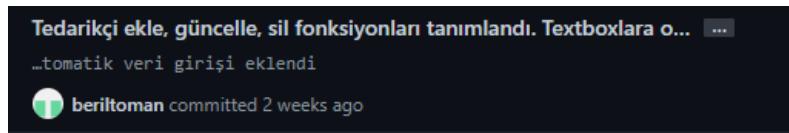
Şekil 45: Tedarikçi form tasarım değişikliği ve fonksiyon tanımlanması

Şekil 46'da gösterilen git commitinde, Tedarikçi formuna borç ödeme fonksiyonu eklendi.



Şekil 46: Tedarikçi formuna borç ödeme fonksiyonu tanımlanması

Şekil 47'de gösterilen git commitinde; Tedarikçi ekleme, güncelleme ve silme fonksiyonları tanımlandı. Metin kutularına otomatik veri girişi sağlandı.



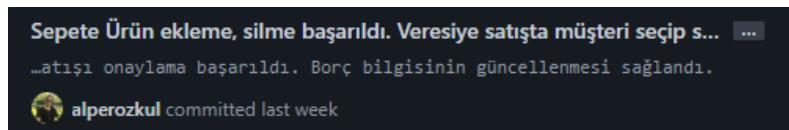
Şekil 47: Tedarikçi formuna ekle, güncelle, sil ve diğer fonksiyonların tanımlanması

Şekil 48'de gösterilen git commitinde; Müşteriler, Ürünler ve Tedarikçiler formlarının tasarımları güncellendi. Müşteri formunda ekle, güncelle, sil fonksiyonları aktifleştirildi.



Şekil 48: Form tasarım değişikliği ve müşteri formuna ekle, güncelle, sil fonksiyon tanımlanması

Şekil 49'da gösterilen git commitinde, Sepet listesine ürün ekleme ve sepetten ürün silme fonksiyonları eklendi. Veresiye satışlarda müşteri seçip satışın onaylanma işlemi gerçekleştirildi. Borç bilgisinin güncellenmesi sağlandı.



Şekil 49: Satış işlemlerinin gerçekleştirilmesi

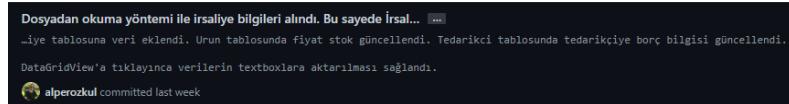
Şekil 50'de gösterilen git commitinde, veritabanı güncellendi ve Satış formu tamamlandı.



Şekil 50: Veritabanı güncellemesi ve satış formunun tamamlanması

Şekil 51'de gösterilen git commitinde, dosyadan okuma yöntemi ile ırsaliye bilgileri alındı.

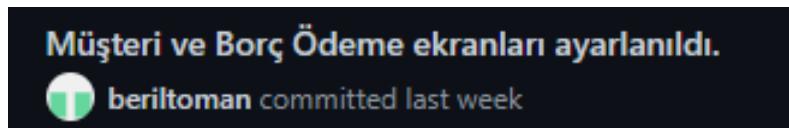
Bu sayede İrsaliye tablosuna veri eklendi. Ürün tablosunda fiyat stok güncellenme işlemi sağlandı. Tedarikçi tablosunda tedarikçiye borç ekleme işlemi tamamlandı.



Şekil 51: Dosyadan okuma yöntemi ile ürün ekleme fonksiyonlarının tamamlanması

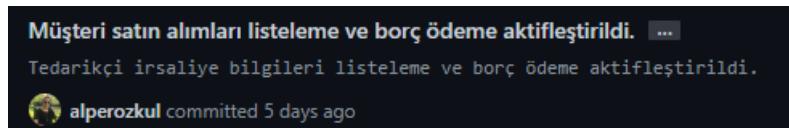
Şekil 52'de gösterilen git commitinde, Borç Ödeme formu oluşturuldu ve tasarıtı yapıldı.

Müşteri form tasarımı değiştirildi.



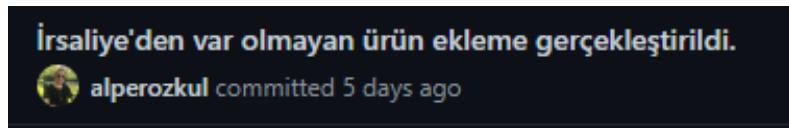
Şekil 52: Borç ödeme form oluşturulması ve müşteri form tasarım değişikliği

Şekil 53'te gösterilen git commitinde, Müşteri formunda satın alımları listeleme ve borç ödeme fonksiyonları aktifleştirildi.



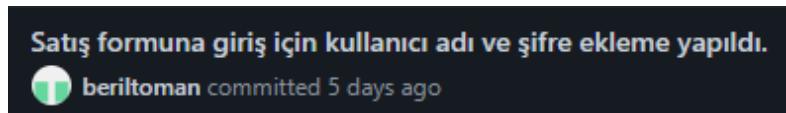
Şekil 53: Müşteri satın alımları listeleme ve borç ödeme fonksiyonları

Şekil 54'te gösterilen git commitinde; Ürün tablosunda bulunmayan yeni ürünü, ırsaliye üzerinden ekleme işlemi gerçekleştirildi.



Şekil 54: İrsaliye üzerinden yeni ürün eklenmesi

Şekil 55'te gösterilen git commitinde, Satış formuna erişebilmek için kullanıcı adı ve şifre ile giriş fonksiyonları eklendi.



Şekil 55: Satış formuna giriş için kullanıcı adı ve şifre fonksiyonları eklenmesi

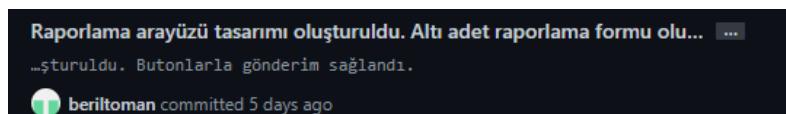
Şekil 56'da gösterilen git commitinde, satış yaptığı durumlarda stok azalma sağlandı ve stok miktarı az olan ürünler için satış sonrası uyarı verme işlemi gerçekleştirildi.



Şekil 56: Satış sonrası stok azalma ve stok uyarı verme işlemleri

Şekil 57'de gösterilen git commitinde, Raporlama formunun arayüz tasarımı oluşturuldu.

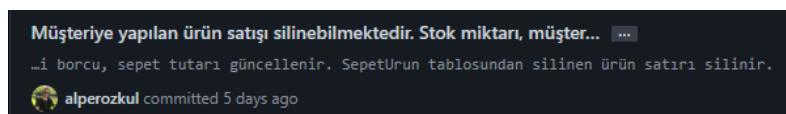
Her raporlama için ayrı form oluşturuldu ve butonlar ile formlara erişim sağlandı.



Şekil 57: Raporlama formları oluşturulması

Şekil 58'de gösterilen git commitinde, Müşteriye yapılan satışın silinmesi sağlandı.

Silinen satış sebebiyle stok miktarı, müşteri borcu ve sepet tutarı güncellendi. Sepet Ürün tablosundan silinen ürün kaldırılması sağlandı.



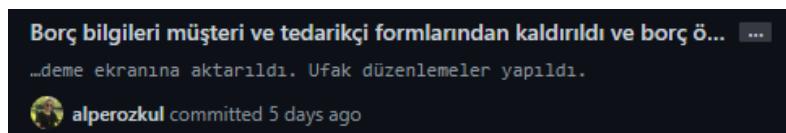
Şekil 58: Yapılan satışın silinmesi

Şekil 59'da gösterilen git commitinde, Müşteriye yapılan satışın silinebilmesi için şifre istemesi gerçekleştirildi.



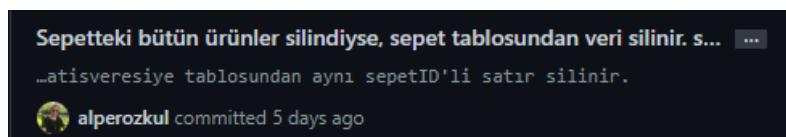
Şekil 59: Yapılan satışın silinebilmesi için şifre istenmesi

Şekil 60'da gösterilen git commitinde, Müşteri ve Tedarikçi formunda bulunan borç ödeme fonksiyonları, Borç Ödeme formuna aktarıldı.



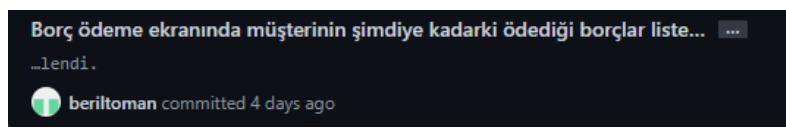
Şekil 60: Borç ödeme fonksiyonlarının taşınması

Şekil 61'de gösterilen git commitinde, sepetteki bütün ürünlerin silinmesi durumunda Sepet tablosundan da verilerin silinme işlemi gerçekleştirildi. Aynı şekilde SatisVeresiye tablosundan da verilerin silinme işlemi gerçekleştirildi.



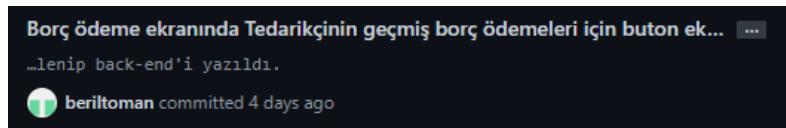
Şekil 61: Sepetteki bütün ürünlerin silinmesi

Şekil 62'de gösterilen git commitinde, Borç Ödeme formunda borç sorgu işlemi gerçekleştirildi.



Şekil 62: Müşteri borç ödeme bilgisi geçmişi

Şekil 63'te gösterilen git commitinde, Borç ödeme formunda Tedarikçinin geçmiş borç ödemeleri için buton eklendi ve fonksiyonları tanımlandı.



Şekil 63: Tedarikçiye ödenen borç bilgisi geçmişi

Şekil 64'de gösterilen git commitinde, müşteriye yapılan satışların toplamı hesaplandı. Müşterinin ödediği toplam borçlar hesaplandı ve Excel'e aktarım sağlandı.



Şekil 64: Müşteri bazlı rapor fonksiyonları ve Excel'e aktarım

Şekil 65'te gösterilen git commitinde, ürün bazlı kar-zarar raporlama işlemi gerçekleştirildi.



Şekil 65: Ürün bazlı kar zarar rapor fonksiyonları

Şekil 66'da gösterilen git commitinde, satış trendi için ürünlerin çok satandan az satana sıralaması yapıldı.



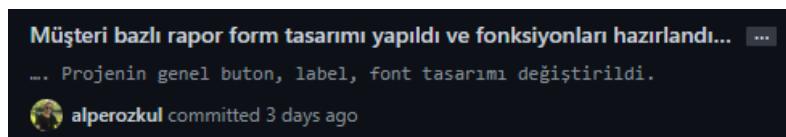
Şekil 66: Çok satandan az satana ürün listesi fonksiyonları

Şekil 67'de gösterilen git commitinde, marketin satışları sonucunda oluşan kar zarar raporlaması gerçekleştirildi.



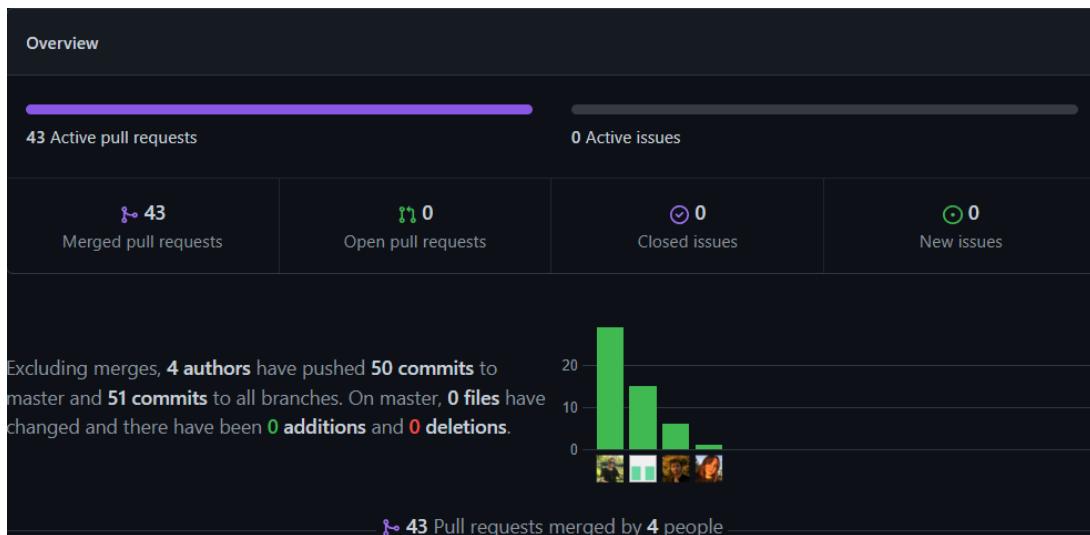
Şekil 67: Market kar zarar rapor fonksiyonları

Şekil 68'de gösterilen git commitinde, müşteri bazlı rapor formunun arayüz tasarımlı yapıldı ve fonksiyonları eklendi. Yazılımın genel arayüz tasarımlı düzenlendi.



Şekil 68: Müşteri bazlı rapor fonksiyonları

Şekil 70 ve 69'te gösterilen analiz doğrultusunda teslim edilen git işlemlerinin ekip üyeleri arasındaki dağılımı verilmiştir.



Şekil 69: Ekip üyeleri arasında iş dağılımı



Şekil 70: Ekip üyeleri arasında iş dağılımı

5.5 Youtube video ve Github Repository Linkleri

Yapılan projenin detayları, kullanılabılır bağlantılar ile aşağıda verilmiştir.

Youtube Video Linki: [Youtube video linki](#).

Github Repository Linki: [Github repository linki](#).

BÖLÜM: IV

S O N U Ç

6

S O N U Ç

Veritabanı Yönetim Sistemleri dersi için gerçekleştirilen proje sonucunda 20 istenirin 19 tanesi başarılı bir şekilde tamamlanmıştır. Proje, bir market sisteminin ihtiyacı olacak verileri saklamak ve gerekli işlemleri yapabilmek amacı ile oluşturulmuştur. Projenin gerçek hayatı kullanılması durumunda; market sahibi ürün alımlarını yaptığı tedarikçilerin sağladığı txt dosyaları sayesinde ekstra herhangi bir işlem yapmadan bu txt dosyasını sisteme yükleyerek stok bilgisini artırabilmektedir. Ayrıca tedarikçi borçlarını da, herhangi başka bir uygulama ya da el yazısı kullanmadan sistem otomatik olarak oluşturmaktadır. Bu sayede market sahibinin kullanacağı tek bir masaüstü uygulaması sayesinde bütün bilgiler saklanabilmektedir. Aynı zamanda satışlar ve müşteriler de uygulama içerisinde ve veritabanında saklandığı için gerekli durumlarda raporlar alınabilmekte ya da marketin gidişatı ile ilgili bilgiler yine tek uygulamadan elde edilebilmektedir. Bu uygulama sayesinde bir marketin ihtiyaçları karşılanması ve gerekli verilerin veritabanında saklanması yapılarak uzun ömürlülük sağlanmıştır.