ANKARA ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLM 4537 PROJE RAPORU

IOS İle Uygulama Geliştirme I

Stock Tracking – KOBİ'ler İçin Stok Takip Sistemi

Zeynep Hacısalihoğlu

Github: https://github.com/zeyneph61/stok_takip_mobil

Video Linki: https://www.youtube.com/watch?v=ZOwL1O10CQU

22290449

Enver Bağcı

15.01.2026

ÖZET

Bu projede küçük ve orta boyutlu işletmelerin (KOBİ) ürün ve stok süreçlerini daha düzenli ve kontrollü şekilde yönetebilmelerini sağlamak amacıyla IOS odaklı bir stok takip sistemi geliştirilmiştir. Geliştirilen sistem; ürün yönetimi, stok giriş ve çıkışlarının takibi, düşük stok ve son kullanma tarihi uyarıları ile satış raporlarının oluşturulması gibi temel işlevleri tek bir yapı altında sunmaktadır.

Uygulama, ASP.NET Core Web API ile geliştirilen bir backend ve Flutter kullanılarak geliştirilen bir iOS mobil uygulamasından oluşmaktadır. Backend tarafında tutulan veriler, iOS uygulaması tarafından API üzerinden alınarak kullanıcıya gösterilmektedir.

İÇİNDEKİLER

ANKARA ÜNİVERSİTESİ 1
ÖZET ii
İÇİNDEKİLER iii
1. GİRİŞ 5
1.1 Projenin Amacı 5
1.2 Proje Kapsamı 5
2. PROJEYE GENEL BAKIŞ 6
2.1 Sistem Mimarisi 6
2.2 Temel İşlevler 7
2.3 Proje Çıktıları 7
3. KULLANILAN TEKNOJİLER 8
3.1 Backend Teknolojileri 8
3.2 Frontend Teknojileri 9
3.3 Geliştirme Araçları 10
4. VERİTABANI YAPISI 12
4.1 Genel Bakış 12
4.2 Products Tablosu 12
4.3 StockMovements Tablosu 13
4.4 Categories Tablosu 13
4.5 PriceHistories Tablosu 14
4.6 LowStockAlerts Tablosu 15
4.7 ExpiryAlert Tablosu 15
4.8 SalesReport Tablosu 16
4.9 Veritabanı İlişkileri 16
5. API YAPISI 17
5.1 Genel Bakış 17
5.2 API Controller'ları 17

5.3 API Dokümantasyonu 19
6. FRONTEND YAPISI 20
6.1 Genel Bakış 20
6.2 Sayfa Yapısı 20
6.2.1 Dashboard (dashboard_tab.dart) 20
6.2.2 Inventory (inventory_tab.dart) 21
6.2.3 Stock Movements (stock_movements_screen.dart) 22
6.3 Tasarım ve Stil Yapısı 23
6.4 Dart İşlevleri 24
7. PROJE ÇALIŞMA MANTIĞI 25
7.1 Veri Akışı Örneği: Ürün Listeleme (Flutter) 25
7.2 Veri Akışı Örneği: Ürün Güncelleme (Flutter) 26
7.3 Otomatik Kayıt Oluşturma Mekanizması 26
8. YAPAY ZEKA KULLANIMI 28
8.1 Kullanılan Yapay Zeka Araçları 28
9. PROJE ÖZELLİKLERİ VE YETENEKLERİ 29
9.1 Temel Özellikler 29
9.2 Teknik Yetenekler 31
10. KURULUM VE ÇALIŞTIRMA 32
10.1 Gereksinimler 32
10.2 Backend Kurulumu 32
10.3 Flutter Kurulumu (Android Emulator için) 33
11. SONUÇ 35
11.1 Proje Kazanımları 35
12. EKLER 36

1. GİRİŞ

1.1 Projenin Amacı

Bu proje küçük ve orta büyüklükteki işletmelerin (KOBİ) stok yönetimi ihtiyaçlarını karşılamak üzere geliştirilmiş IOS tabanlı bir stok takip sistemidir. Projenin amacı, işletmelerin ürün envanterlerini dijital ortamda yönetmelerini, stok hareketlerini takip etmelerini ve stok düzeylerini kontrol etmelerini sağlamaktır

1.2 Proje Kapsamı

Kapsam İçinde:

- Mobil (iOS) tabanlı ürün yönetimi (ekleme, silme, güncelleme, listeleme)
- Stok giriş/çıkış hareketlerinin kayıt altına alınması
- Kategori bazlı ürün organizasyonu
- Düşük stok ve son kullanma tarihi uyarı sistemleri
- Dashboard üzerinde özet istatistikler ve raporlar
- RESTful API mimarisi ile backend geliştirme

Kapsam Dışında:

- Kullanıcı kimlik doğrulama ve yetkilendirme sistemi
- Barkod okuma özelliği
- Muhasebe ve fatura yönetimi

2. PROJEYE GENEL BAKIŞ

2.1 Sistem Mimarisi

Proje, backend ve frontend olmak üzere iki ana bileşenden oluşmaktadır:

Backend:

- ASP.NET Core 8.0 Web API ile geliştirilmiştir
- SQL Server veritabanı kullanılarak verilerin saklanması sağlanmıştır
- Entity Framework Core ile veritabanı işlemleri yapılmaktadır
- RESTful API prensiplerine uygun endpoint'ler tasarlanmıştır
- Swagger ile API dokümantasyonu oluşturulmuştur

Frontend (iOS)

- Flutter framework'ü kullanılarak iOS için geliştirilmiştir
- Uygulama üç ana ekran içermektedir: Dashboard, Inventory ve Stock Movements; ürün ekleme ve düzenleme işlemleri için ayrıca bir Product Management ekranı bulunmaktadır
- Backend ile iletişim RESTful API üzerinden HTTP istekleri ile sağlanmaktadır

Çalışma Prensibi

- Kullanıcı iOS cihazı üzerinden mobil uygulamayı açar
- Flutter uygulaması backend API'ye HTTP istekleri gönderir
- Backend API istekleri işler ve veritabanından gerekli verileri alır
- Veriler JSON formatında mobil uygulamaya gönderilir
- Flutter uygulaması gelen verileri işleyerek ekranlarda kullanıcıya sunar

2.2 Temel İşlevler

Ürün Yönetimi:

- Yeni ürün ekleme ve mevcut ürünleri düzenleme
- Ürün silme
- Kategori bazlı ürün organizasyonu
- Alış ve satış fiyatı takibi

Stok Takibi:

- Stok giriş (In) ve çıkış (Out) işlemleri
- Tüm stok hareketlerinin tarihsel kaydı
- Anlık stok durumu görüntüleme
- Otomatik stok hesaplaması

Uyarı ve Bildirim Sistemi:

- Düşük stok seviyesi sayısı (threshold değerine göre)
- Son kullanma tarihi yaklaşan ürün sayısı
- Stokta olmayan ürün sayısı

Raporlama ve Analiz:

- Dashboard üzerinde özet istatistikler
- Kategori bazlı satış analizleri
- Aylık gelir ve kar hesaplamaları
- En çok satan ürün listeleri

2.3 Proje Çıktıları

- Tamamen işlevsel iOS mobil uygulaması (Flutter) tabanlı stok yönetim sistemi
- RESTful API servisleri
- Swagger dokümantasyonu ile API referansı
- SQL Server veritabanı şeması
- Teknik dokümantasyon

3. KULLANILAN TEKNOJİLER

3.1 Backend Teknolojileri

ASP.NET Core 8.0 - Web API Framework:

- Stok takip sisteminin tüm backend işlemlerini yönetmek için kullanılmıştır.
- Product, StockMovement, Category gibi controller'lar ile modüler yapı oluşturulmuştur.
- HTTP metodları (GET, POST, PUT, DELETE) ile CRUD işlemleri gerçekleştirilmiştir.
- CORS politikası sayesinde frontend'in API'ye güvenli erişimi sağlanmıştır.
- Program.cs dosyasında servisler yapılandırılmıştır.

Entity Framework Core:

- SQL Server veritabanı ile C# kodları arasında köprü görevi görmüştür.

SQL Server - Veritabanı Yönetim Sistemi:

- 7 adet tablo (Products, StockMovement, Categories, PriceHistories, ExpiryAlerts, SalesReports, LowStockAlerts) oluşturulmuştur.
- İlişkisel veritabanı yapısı ile ürün-kategori, ürün-stok hareketi gibi bağlantılar kurulmuştur
- stok_takip_DB adında veritabanı oluşturulmuştur
- Connection string ile API'den veritabanına bağlantı sağlanmıştır
- Decimal, int, nvarchar gibi veri tipleri ile uygun veri saklama yapılmıştır

C# - Programlama Dili

- Tüm backend kodları C# ile yazılmıştır
- Controller sınıflarında (ProductController, StockMovementController vb.) API metodları geliştirilmiştir
- Model sınıflarında (Products, StockMovement vb.) veri yapıları tanımlanmıştır

3.2 Frontend Teknojileri

Flutter – Mobil Uygulama Geliştirme:

- Flutter kullanılarak iOS öncelikli, Android destekli mobil uygulama geliştirilmiştir.
- Uygulama; Dashboard, Inventory, Stock Movements 3 ana ekrandan ve Inventory içinde çalışan ek bir Product Management ekranından oluşmaktadır
- Listeleme işlemleri için ListView ve Card widget'ları kullanılmıştır
- Ürün ekleme ve düzenleme işlemleri için form yapıları (TextField, DropdownButton) oluşturulmuştur

UI Tasarım ve Stil:

- Arayüz tasarımları Figma üzerinde hazırlanmıştır
- Tasarımlar Flutter widget yapısına uygun şekilde uygulanmıştır

Dart – Dinamik İşlevler:

- Backend ile iletişim için Dart http paketi kullanılmıştır
- RESTful API üzerinden GET, POST, PUT ve DELETE istekleri gönderilmiştir
- API'den alınan veriler JSON formatında işlenmiştir
- Sayfalama ve filtreleme işlemleri uygulama tarafında gerçekleştirilmiştir

3.3 Geliştirme Araçları

Visual Studio 2022 – Entegre Geliştirme Ortamı (IDE):

- ASP.NET Core Web API projesi oluşturulmuş ve düzenlenmiştir

Visual Studio Code – Kod Editörü:

- Flutter tabanlı iOS mobil uygulaması Visual Studio Code üzerinde geliştirilmiştir
- Dart dili için gerekli eklentiler (Flutter, Dart) kullanılmıştır
- Mobil uygulama kodlarının düzenlemesi yapılmıştır

Android Studio – Mobil Geliştirme ve Emulator Aracı:

- Android Emulator kullanılarak uygulamanın mobil cihaz üzerinde testleri yapılmıştır
- Flutter projelerinin çalıştırılması gerçekleştirilmiştir

Git & GitHub – Versiyon Kontrol Sistemi:

- Proje dosyaları GitHub deposunda saklanmıştır
- Her önemli değişiklik commit mesajları ile kaydedilmiştir
- Repository: zeyneph61/Stok_Takip

Figma – UI/UX Tasarım Aracı:

- Uygulama arayüzleri Figma üzerinde tasarlanmıştır
- Dashboard, Inventory ve Stock Movements ekranları için tasarımlar hazırlanmıştır
- Renk paleti ve font seçimleri belirlenmiştir

Swagger – API Dokümantasyon ve Test Aracı:

- http://localhost:5000/swagger adresinden API dokümantasyonuna erişilmiştir
- POST ve PUT isteklerinde JSON formatı Swagger üzerinden test edilmiştir

GitHub Copilot – Yapay Zeka Destekli Kod Asistanı:

- Flutter ve Dart kodlarının yazımında destek sağlamıştır
- Tekrarlayan kod blokları için öneriler sunmuştur
- Yorum satırları ve kod yapısının iyileştirilmesinde kullanılmıştır

4. VERİTABANI YAPISI

4.1 Genel Bakış

Proje SQL Server üzerinde stok_takip_DB adında bir veritabanı kullanmaktadır. Veritabanı, normalize edilmiş ilişkisel tablo yapısına sahiptir ve toplamda 7 ana tablodan oluşmaktadır. Tablolar arasında Foreign Key ilişkileri kurularak veri bütünlüğü sağlanmıştır. Veritabanı tasarımında, stok takibi için gerekli tüm bilgilerin (ürün bilgileri, stok hareketleri, fiyat değişiklikleri, uyarılar ve raporlar) ayrı tablolarda saklanması prensibi benimsenmiştir.

4.2 Products Tablosu

Ürün bilgilerini saklar ve sistemin ana tablosudur. Diğer birçok tablo bu tabloya referans vermektedir.

Alan Adı	Veri Tipi	Açıklama
Id	int (PK)	Benzersiz ürün ID'si
Name	nvarchar	Ürün adı
Category	nvarchar	Kategori adı (text olarak)
CategoryId	int (FK)	Kategori tablosuna referans
BuyingPrice	decimal(10,2)	Alış fiyatı
SellPrice	decimal(10,2)	Satış fiyatı
Quantity	int	Mevcut stok adedi
ThresholdValue	int	Minimum stok eşik değeri
ExpiryDate	datetime	Son kullanma tarihi
Availability	nvarchar	Stok durumu (In Stock, Low Stock, Out of Stock)
SoldLastMonth	int	Geçen ay satılan adet

Kullanım Amacı: Sistemdeki tüm ürünlerin temel bilgilerini tutmak, stok durumunu takip etmek ve satış performansını izlemek.

4.3 StockMovements Tablosu

Stok giriş ve çıkış hareketlerini kaydeder. Her stok değişikliği bu tabloda loglanır.

Alan Adı	Veri Tipi	Açıklama
Id	int (PK)	Hareket kimliği
ProductId	int (FK)	İlgili ürün
MovementType	nvarchar	Hareket tipi (In/Out)
Quantity	int	Hareket miktarı
TotalPrice	decimal(10,2)	Toplam tutar
Date	datetime	İşlem tarihi

Kullanım Amacı: Ürünlerin stok giriş ve çıkışlarını kaydetmek, stok hareketlerinin geçmişini tutmak ve raporlama yapmak.

4.4 Categories Tablosu

Ürün kategorilerini tutar ve ürünlerin gruplandırılmasını sağlar.

Alan Adı	Veri Tipi	Açıklama
Id	int (PK)	Kategori ID
Name	nvarchar	Kategori adı
Description	nvarchar	Kategori açıklaması
CreatedAt	datetime	Oluşturulma tarihi

Kullanım Amacı: Ürünleri kategorilere ayırmak, kategori bazlı raporlama yapmak ve ürün yönetimini kolaylaştırmak.

4.5 PriceHistories Tablosu

Fiyat değişikliklerini takip eder ve fiyat geçmişini saklar.

Alan Adı	Veri Tipi	Açıklama
Id	int (PK)	Kayıt ID
ProductId	int (FK)	İlgili ürün
OldBuyingPrice	decimal(10,2)	Eski alış fiyatı
NewBuyingPrice	decimal(10,2)	Yeni alış fiyatı
OldSellPrice	decimal(10,2)	Eski satış fiyatı
NewSellPrice	decimal(10,2)	Yeni satış fiyatı
ChangeDate	datetime	Değişiklik tarihi
Reason	nvarchar	Değişiklik nedeni

Kullanım Amacı: Ürün fiyatlarındaki değişiklikleri takip etmek, fiyat analizi yapmak ve geçmiş fiyatları görmek.

4.6 LowStockAlerts Tablosu

Düşük stok uyarılarını saklar ve eşik değerinin altına düşen ürünleri takip eder.

Alan Adı	Veri Tipi	Açıklama
Id	int (PK)	Uyarı kimliği
ProductId	int (FK)	İlgili ürün
CurrentQuantity	int	Mevcut miktar
ThresholdValue	int	Eşik değeri
AlertDate	datetime	Uyarı tarihi
IsResolved	bit	Çözüldü mü?

Kullanım Amacı: Stok seviyesi kritik değerin altına düşen ürünleri tespit etmek ve uyarı vermek.

4.7 ExpiryAlert Tablosu

Son kullanma tarihi yaklaşan ürünleri takip eder.

Alan Adı	Veri Tipi	Açıklama
Id	int (PK)	Uyarı ID
ProductId	int (FK)	İlgili ürün
ExpiryDate	datetime	Son kullanma tarihi
DaysRemaining	int	Kalan gün sayısı
AlertDate	datetime	Uyarı tarihi
IsResolved	bit	Çözüldü mü?

Kullanım Amacı: Son kullanma tarihi yaklaşan ürünleri tespit etmek ve zamanında müdahale edilmesini sağlamak.

4.8 SalesReport Tablosu

Satış raporlarını tutar ve aylık bazda performans takibi yapar.

Alan Adı	Veri Tipi	Açıklama
Id	int (PK)	Rapor ID
ProductId	int (FK)	İlgili ürün
Month	int	Ay
Year	int	Yıl
QuantitySold	int	Satılan adet
TotalRevenue	decimal(10,2)	Toplam gelir
TotalProfit	decimal(10,2)	Toplam kar
GeneratedAt	datetime	Rapor tarihi

Kullanım Amacı: Aylık satış performansını takip etmek, gelir ve kar analizleri yapmak.

4.9 Veritabanı İlişkileri

Tablolar arası ilişkiler Foreign Key kullanılarak kurulmuştur:

İlişkilerin Önemi:

- Veri bütünlüğü sağlanmıştır (bir ürün silindiğinde ilgili kayıtlar da yönetilir)
- Gereksiz veri tekrarı önlenmiştir
- Entity Framework Core bu ilişkileri otomatik yönetmektedir

5. API YAPISI

5.1 Genel Bakış

Proje RESTful API mimarisi kullanmaktadır. Tüm API endpoint'leri /api/[controller] formatında yapılandırılmıştır ve http://localhost:5000 adresinde çalışmaktadır. API JSON formatında veri alışverişi yapmakta ve CORS politikası ile frontend'e güvenli erişim sağlamaktadır.

Bu API yapısı hem web hem de mobil uygulama tarafından ortak olarak kullanılmaktadır.

5.2 API Controller'ları

Controller'lar altında tanımlanan bazı endpoint'ler, frontend sayfalarında farklı amaçlarla birlikte kullanılmaktadır. Projede 7 ana controller bulunmaktadır:

1. ProductController (/api/product)

- GET - Tüm ürünleri listeler
- GET /{id} - Belirli bir ürünü getirir
- POST - Yeni ürün ekler
- PUT /{id} - Ürün günceller (stok değişikliğinde otomatik StockMovement, fiyat değişikliğinde PriceHistory kaydı oluşturur)
- DELETE /{id} - Ürün siler

2. CategoryController (/api/category)

- GET - Tüm kategorileri listeler
- GET /{id} - Belirli bir kategoriyi getirir
- POST - Yeni kategori ekler
- PUT /{id} - Kategori günceller
- DELETE /{id} - Kategori siler

3. StockMovementController (/api/stockmovement)

- GET - Tüm stok hareketlerini listeler
- POST - Yeni stok hareketi ekler

4. PriceHistoryController (/api/pricehistory)

- GET - Tüm fiyat geçmişini listeler
- GET /product/{productId} - Belirli ürünün fiyat geçmişini getirir
- GET /recent?count=10 - Son fiyat değişikliklerini getirir
- POST - Yeni fiyat kaydı ekler
- DELETE /{id} - Fiyat geçmişi kaydını siler
- GET /date-range - Tarih aralığına göre fiyat geçmişini getirir

5. LowStockAlertController (/api/lowstockalert)

- GET - Çözülmemiş düşük stok uyarılarını listeler
- POST /check - Düşük stok kontrolü yapar ve gerekirse uyarı oluşturur
- PUT /{id}/resolve - Uyarıyı çözüldü olarak işaretler

6. ExpiryAlertController (/api/expiryalert)

- GET - Çözülmemiş son kullanma tarihi uyarılarını listeler
- POST /check - Son kullanma tarihi kontrolü yapar ve gerekirse uyarı oluşturur
- PUT /{id}/resolve - Uyarıyı çözüldü olarak işaretler
- GET /product/{productId} - Belirli ürüne ait uyarıları getirir
- DELETE /{id} - Uyarı kaydını siler

7. SalesReportController (/api/salesreport)

- GET - Tüm satış raporlarını listeler
- GET /{id} - Belirli bir raporu getirir
- GET /product/{productId} - Belirli ürünün satış raporlarını getirir
- GET /monthly - Aylık satış raporlarını getirir
- GET /yearly/{year} - Belirtilen yıla ait yıllık özet raporu getirir
- GET /top-selling - En çok satan ürünleri getirir
- POST - Yeni satış raporu ekler
- PUT /{id} - Rapor günceller
- DELETE /{id} - Rapor siler

5.3 API Dokümantasyonu

Swagger kullanılarak API dokümantasyonu oluşturulmuştur. http://localhost:5000/swagger adresinden tüm endpoint'ler görüntülenebilmekte ve test edilebilmektedir. Swagger UI sayesinde her endpoint için örnek request/response formatları, parametre tipleri ve HTTP durum kodları detaylı olarak sunulmaktadır.

6. FRONTEND YAPISI

6.1 Genel Bakış

Frontend, Flutter framework'ü ve Dart programlama dili kullanılarak geliştirilmiştir. Proje, iOS ve Android platformlarında çalışan cross-platform bir mobil uygulama sunmaktadır. Uygulama üç ana kullanıcı ekranı içermektedir: Dashboard (Ana Sayfa), Inventory (Ürün Yönetimi) ve Stock Movements (Stok Hareketleri).

6.2 Sayfa Yapısı

6.2.1 Dashboard (dashboard_tab.dart)

Ana kontrol paneli sayfasıdır. Kullanıcının uygulamayı açtığında gördüğü ilk ekrandır.

Özellikler:

- En çok kâr getiren kategorileri gösterir
- Toplam stok miktarını gösterir
- Yakında son kullanma tarihi geçecek ürünleri listeler
- Düşük stoklu ürün sayısını gösterir
- Stokta olmayan ürün sayısını gösterir
- En çok satan ürünleri listeler (ilk 3 ürün)

API Bağlantıları:

Dashboard sayfası 5 farklı servis kullanmaktadır:

- /api/Product - Tüm ürünleri yükler ve envanter özetini hesaplar
- /api/Category - Kategorileri yükler
- /api/ExpiryAlert - Son kullanma tarihi uyarılarını getirir
- /api/LowStockAlert - Düşük stok uyarılarını getirir
- /api/SalesReport/top-selling?count=5 - En çok satan 5 ürünü getirir (ilk 3'ü gösterilir)

İstatistik Hesaplamaları:

Dashboard, API'den gelen verilerle şu hesaplamaları yapar:

- Toplam stok miktarı
- Toplam ürün sayısı
- Düşük stok ve stok dışı ürün sayıları
- En çok kâr getiren kategori, SalesReport tablosundaki TotalProfit alanı kullanılarak hesaplanmıştır.
- En çok satan ürünler (ilk 3 tanesi)

Veri Modelleme:

Dashboard, verileri DashboardStats custom class ile modeller ve FutureBuilder pattern ile asenkron veri yönetimi yapar.

6.2.2 Inventory (inventory_tab.dart)

Ürün listesi ve envanter yönetim sayfasıdır.

Özellikler:

- Tüm ürünleri liste halinde gösterir
- Ürün arama özelliği (isim ve kategoriye göre filtreleme)
- Sayfalama (pagination) - Sayfa başı 5 ürün gösterilir
- Toplam kategori sayısı
- Toplam ürün sayısı ve stok miktarı
- Düşük stok ve stok dışı ürün sayıları
- Bottom sheet ile ürün düzenleme

API Bağlantıları:

Inventory sayfası 2 farklı API endpoint'i kullanmaktadır:

- /api/Product - Tüm ürünleri yükler ve listede gösterir
- /api/Category - Kategorileri yükler ve kategori sayısını hesaplar

Arama Özelliği:

Kullanıcı arama kutusuna yazdığı metne göre ürünler anlık olarak filtrelenir. Hem ürün adı hem de kategori adına göre arama yapılabilir.

Sayfalama Mantığı:

Her sayfada 5 ürün gösterilir (_itemsPerPage = 5). Kullanıcı sayfa numaralarına tıklayarak diğer ürünleri görüntüleyebilir.

6.2.3 Stock Movements (stock_movements_screen.dart)

Stok giriş ve çıkış hareketlerini görüntüleyen sayfadır.

Özellikler:

- Tüm stok hareketlerini tarih sırasına göre listeler
- Hareket tipi (In/Out) gösterir
- Her hareket için ürün bilgisi, miktar ve tarih bilgisi görüntülenir
- Pull-to-refresh özelliği

Filtreleme Özellikleri:

Stock Movements sayfasında 5 farklı filtreleme özelliği bulunmaktadır:

- Start Date - Başlangıç tarihi filtresi
- End Date - Bitiş tarihi filtresi
- Movement Type - Hareket tipi (In/Out)
- Category - Kategori filtresi
- Product Search - Ürün adına göre arama
- Clear Button - Tüm filtreleri temizleme

API Bağlantıları:

Stock Movements sayfası 2 farklı API endpoint'i kullanmaktadır:

- /api/Product - Tüm ürünleri yükler (filtreleme için gerekli)
- /api/StockMovement - Tüm stok hareketlerini yükler ve listeler,

Filtreleme Mantığı:

Kullanıcı filtreleri seçtiğinde, tamamen client-side filtreleme yapılır. Seçilen tarih aralığı, hareket tipi, kategori ve ürün adına göre stok hareketleri dinamik olarak filtrelenir.

6.3 Tasarım ve Stil Yapısı

Kullanılan Flutter Widget'ları:

- Scaffold - Sayfa yapısı
- AppBar - Üst navigasyon
- ListView - Ürün ve hareket listeleri
- GridView - Dashboard kart yapısı
- Card - Kart tasarımları
- TextField - Arama kutuları
- BottomSheet - Modal ekranlar (product_edit_bottom_sheet.dart)
- RefreshIndicator - Pull-to-refresh için
- FutureBuilder - Asenkron veri yükleme
- CircularProgressIndicator - Loading state
- FloatingActionButton - Refresh butonu
- BottomNavigationBar - Alt navigasyon
- Drawer - Yan menü

Tasarım Özellikleri:

- Material Design
- Responsive layout
- Pull-to-refresh
- Loading ve error state gösterimleri

6.4 Dart İşlevleri

HTTP Package Kullanımı:

Tüm sayfalarda backend ile iletişim için http package kullanılmıştır. Sayfalar temel olarak GET istekleri ile veri çekmektedir. PUT isteği product_service.dart servisinde tanımlanmış olup, product_edit_bottom_sheet.dart widget'ında kullanılmıştır.

State Management:

StatefulWidget ile local state yönetimi yapılmıştır. API'den gelen veriler setState() ile UI'a yansıtılır. FutureBuilder pattern ile veri yükleme yapılmıştır. Dashboard verileri DashboardStats custom class ile modellenir.

Veri İşleme:

Dart list metodları kullanılarak veri işleme yapılmıştır:

- where() - Arama ve filtreleme işlemlerinde
- map() - Veri dönüşümlerinde
- reduce() - Toplam hesaplamalarında
- sort() - Tarihe göre sıralama işlemlerinde
- take() - Belirli sayıda ürün almak için (örn: topProducts.take(3))

JSON formatında veri alışverişi yapılmış, API'den gelen veriler dart:convert ile Dart nesnelerine dönüştürülerek kullanılmıştır.

7. PROJE ÇALIŞMA MANTIĞI

7.1 Veri Akışı Örneği: Ürün Listeleme (Flutter)

Kullanıcı Inventory ekranını açtığında ürünlerin nasıl yüklendiğini adım adım inceleyelim:

- Adım 1: Kullanıcı uygulamayı başlatır ve Inventory sekmesine tıklar
- Adım 2: inventory_tab.dart widget'ı yüklenir ve initState() metodu çalışır
- Adım 3: ProductService.getProducts() metodu çağrılır ve HTTP GET isteği gönderilir
- Adım 4: Backend'deki ProductController isteği karşılar
- Adım 5: Entity Framework Core, SQL sorgusu üretir ve veritabanından verileri çeker
- Adım 6: SQL Server veritabanından Products tablosundaki tüm kayıtlar getirilir
- Adım 7: Entity Framework Core, verileri C# Product nesnelerine dönüştürür
- Adım 8: API, verileri JSON formatında mobile uygulamaya gönderir
- Adım 9: http paketi response'u alır ve status code'u kontrol eder
- Adım 10: Dart'ta json.decode() ile JSON verisi parse edilir
- Adım 11: Product.fromJson() metodları ile Dart nesnelerine dönüştürülür
- Adım 12: setState() ile UI güncellenir ve veriler state'e atanır
- Adım 13: ListView.builder widget'ı ile ürünler ekrana render edilir
- Adım 14: Kullanıcı ekranda ürün listesini görür

7.2 Veri Akışı Örneği: Ürün Güncelleme (Flutter)

Kullanıcı bir ürünü düzenlediğinde:

- Adım 1: Kullanıcı ürün kartına tıklar
- Adım 2: product_edit_bottom_sheet.dart modal açılır
- Adım 3: Kullanıcı değişiklik yapar ve "Kaydet" butonuna tıklar
- Adım 4: Form validasyonu yapılır (boş alan kontrolü)
- Adım 5: ProductService.updateProduct() metodu çağrılır
- Adım 6: HTTP PUT isteği gönderilir
- Adım 7: Backend ProductController.UpdateProduct() metodu çalışır
- Adım 8: Stok değişikliği varsa otomatik olarak StockMovement kaydı oluşturulur
- Adım 9: Fiyat değişikliği varsa otomatik olarak PriceHistory kaydı oluşturulur
- Adım 10: Database güncellenir (SaveChangesAsync())
- Adım 11: Backend başarı mesajı döner
- Adım 12: Flutter'da setState() ile liste yenilenir
- Adım 13: SnackBar ile kullanıcıya başarı mesajı gösterilir
- Adım 14: Modal kapanır ve güncel liste görünür

7.3 Otomatik Kayıt Oluşturma Mekanizması

Sistemde bazı işlemler otomatik olarak ilgili tablolara kayıt oluşturur:

Stok Değişikliğinde:

- Ürün güncellenirken stok miktarı değişirse, PUT işlemi sırasında otomatik olarak StockMovements tablosuna kayıt eklenir
- Stok artışı "In" (giriş), azalış "Out" (çıkış) olarak kaydedilir
- MovementDate otomatik olarak işlem zamanına ayarlanır
- TotalPrice hesaplanarak kaydedilir

Fiyat Değişikliğinde:

- Ürün güncellenirken alış veya satış fiyatı değişirse, otomatik olarak PriceHistories tablosuna eski ve yeni fiyatlar kaydedilir
- Bu sayede fiyat değişikliklerinin geçmişi tutulur
- ChangeDate otomatik olarak işlem zamanına ayarlanır

7.4 Flutter State Management Akışı

Flutter uygulamasında state yönetimi şu şekilde çalışır:

- Initial Load (İlk Yükleme):
- initState() çağrılır
- Future<List<Product>> _loadProducts() başlatılır
- FutureBuilder widget'ı loading state gösterir
- API'den veri gelince setState() çağrılır
- UI otomatik olarak yeniden render edilir

Pull-to-Refresh (Yenileme):

- Kullanıcı ekranı aşağı çeker
- RefreshIndicator onRefresh callback'i tetiklenir
- API'ye yeni istek gönderilir
- Gelen veriler state'e atanır (setState())
- Liste güncellenir ve indicator kaybolur

Search/Filter (Arama/Filtreleme):

- Kullanıcı arama kutusuna yazar
- TextField'ın onChanged eventi tetiklenir
- _filteredProducts listesi güncellenir
- setState() çağrılır
- ListView sadece filtrelenmiş ürünleri gösterir

8. YAPAY ZEKA KULLANIMI

Proje geliştirme sürecinde yapay zeka destekli araçlar aktif olarak kullanılmıştır.

8.1 Kullanılan Yapay Zeka Araçları

GitHub Copilot

Dart Kodlarında Kullanım:

- Model sınıflarında fromJson() ve toJson() metodlarının otomatik yazımı
- Service sınıflarında HTTP request kodlarının tamamlanması
- Flutter widget'larının kod tamamlama önerileri
- ListView.builder ve FutureBuilder pattern'lerinin hızlı yazımı
- Yorum satırları yazılarak Copilot'tan kod önerileri alınması
- Tekrarlayan kod bloklarının (örn: card widget'ları) hızlı yazımı

ChatGPT

Flutter Geliştirmede Kullanım:

- Flutter widget yapıları ve lifecycle metodları hakkında bilgi alınması
- State management pattern'leri (StatefulWidget vs StatelessWidget) konusunda danışma
- API entegrasyonu için Dart async/await kullanımı örnekleri
- JSON serialization/deserialization konusunda yardım
- Flutter navigation ve routing yapısının kurulumu
- Error handling best practices
- Dart list metodları (where, map, fold, sort) kullanım örnekleri

Claude (GitHub Copilot Chat)

Kullanım Alanları:

- Karmaşık widget tree'lerinin yapılandırılması
- Hata mesajlarını anlama ve çözüm bulma

9. PROJE ÖZELLİKLERİ VE YETENEKLERİ

9.1 Temel Özellikler

Ürün Yönetimi:

- Ürün ekleme, güncelleme, silme ve listeleme (CRUD işlemleri)
- Ürün bilgileri: Ad, kategori, alış fiyatı, satış fiyatı, miktar, eşik değeri, son kullanma tarihi
- Kategori bazlı ürün sınıflandırma
- Stok miktarı otomatik güncelleme
- Ürün arama (isim ve kategoriye göre)
- Sayfalama sistemi (5 ürün/sayfa)
- Bottom sheet modal ile hızlı düzenleme

Stok Takibi:

- Stok giriş/çıkış hareketlerinin kaydı
- Otomatik stok güncelleme (ürün değişikliklerinde)
- Stok hareketleri geçmişi
- Tarih aralığına göre filtreleme
- Kategori bazlı filtreleme
- Hareket tipi (In/Out) filtreleme
- Ürün adına göre arama

Uyarı Sistemleri:

- Düşük Stok Uyarıları: Ürün miktarı eşik değerinin altına düştüğünde otomatik uyarı
- Son Kullanma Tarihi Uyarıları: 30 gün içinde süresi dolacak ürünler için uyarı
- Dashboard'da uyarı sayılarının görüntülenmesi
- Renk kodlu uyarı gösterimleri (kırmızı, sarı, yeşil)

Dashboard Özellikleri:

- Özet Kartlar: Toplam stok, ürün sayısı, kategori sayısı
- En Çok Kâr Getiren Kategoriler: Profit hesaplaması ile sıralama
- En Çok Satan Ürünler: İlk 3 ürünün gösterimi
- Uyarı Özeti: Düşük stok, son kullanma tarihi ve stokta yok sayıları
- Kategori Listesi: Tüm kategorilerin görüntülenmesi

Satış Raporları:

- Aylık satış verilerinin görüntülenmesi
- En çok satan ürünler listesi (API'den top 5)
- Ürün bazlı satış analizleri
- Kategori bazlı kar hesaplamaları
- soldLastMonth verisi ile performans takibi

Kategori Yönetimi:

- Kategorilerin listelenmesi
- Dashboard'da toplam kategori sayısı
- Kategori bazlı ürün filtreleme
- Stock Movements'ta kategori filtresi

Fiyat Geçmişi:

- Ürün fiyat değişikliklerinin otomatik kaydı
- Alış ve satış fiyatı geçmişi
- Fiyat değişim tarihlerinin kaydedilmesi
- Tarih aralığına göre fiyat geçmişi sorgulama

9.2 Teknik Yetenekler

Backend Yetenekleri:

- RESTful API Mimarisi: Standart HTTP metodları (GET, POST, PUT, DELETE)
- Entity Framework Core: ile veritabanı işlemleri
- SQL Server Entegrasyonu: LocalDB
- CORS Desteği: Cross-origin isteklere izin verme (Mobile App için)
- Swagger UI: API dokümantasyonu ve test arayüzü

Frontend (Flutter) Yetenekleri:

- Client-Side Filtering: Dart ile anlık filtreleme
- Pagination: Sayfa numaraları ile gezinme
- Modal Sheets: Bottom sheet ile hızlı düzenleme
- Navigation: Named routes ile sayfa geçişleri
- Drawer Menu: Yan menü ile kolay erişim

API İletişim Yetenekleri:

- HTTP Package: Dart'ta HTTP istekleri
- JSON Parsing: dart:convert ile veri dönüşümü
- Model Mapping: fromJson() ve toJson() metodları

Veri İşleme Yetenekleri:

- List Operations: where(), map(), fold(), sort(), take()
- Date Formatting: intl paketi ile tarih formatları
- Search & Filter: Real-time arama ve çoklu filtreleme
- Calculations: Kar hesaplama, toplam stok hesaplama
- Sorting: Tarih ve değer bazlı sıralama

10. KURULUM VE ÇALIŞTIRMA

10.1 Gereksinimler

Backend Gereksinimleri:

Yazılım:

- .NET 8 SDK
- Visual Studio 2022 veya Visual Studio Code
- SQL Server

Sistem:

- Windows 10/11
- Minimum 2GB boş disk alanı

Flutter (Mobile App) Gereksinimleri:

Yazılım:

- Flutter SDK 3.9.2 veya üzeri
- Android Studio

Sistem:

- Windows 10/11
- Minimum 5GB boş disk alanı
- Minimum 8GB RAM

10.2 Backend Kurulumu

Adım 1: Projeyi İndirin

- Projeyi GitHub'dan indirin veya ZIP olarak açın
- Backend klasörüne gidin

Adım 2: Veritabanı Ayarları

- appsettings.json dosyasını açın
- Connection string'i kontrol edin:
- Server=SIZIN_SUNUCU_ADINIZ\DB;Database=stok_takip_DB;Trusted_Connection=True;

Adım 3: Veritabanını Oluşturun

- Visual Studio'da Package Manager Console açın
- Bu komutları çalıştırın:
    - Add-Migration InitialCreate
    - Update-Database

Adım 4: Paketleri Yükleyin

- Solution Explorer'da projeye sağ tık
- "Restore NuGet Packages" seçin

Adım 5: Çalıştırın

- F5'e basın veya "Start" butonuna tıklayın
- Backend http://localhost:5000 adresinde başlayacak

Adım 6: Test Edin

- Tarayıcıda http://localhost:5000/swagger açın
- API endpoint'lerini test edin

10.3 Flutter Kurulumu (Android Emulator için)

Adım 1: Flutter SDK Kurun

- https://flutter.dev/docs/get-started/install adresine gidin
- Windows için Flutter SDK indirin
- C:\flutter\ klasörüne çıkarın
- Sistem Path'e C:\flutter\bin ekleyin

Adım 2: Android Studio Kurun

- Android Studio indirin ve kurun
- Kurulum sırasında Android SDK'yı da yükleyin
- Android Emulator yükleyin

Adım 3: Emulator Oluşturun

- Android Studio'yu açın
- Tools > Device Manager
- "Create Device" tıklayın
- Pixel 5 veya benzeri seçin
- System Image indirin (örn: Android 13)

Adım 4: Flutter Projesini İndirin

- Flutter proje klasörüne gidin
- Komut satırında flutter pub get çalıştırın

Adım 5: API Ayarlarını Yapın

Adım 6: Emulator'u Başlatın

- Komut satırında flutter emulators yazın
- Emulator ID'sini kopyalayın
- flutter emulators --launch <emulator_id> yazın

Adım 7: Uygulamayı Çalıştırın

- Backend'in çalıştığından emin olun
- Komut satırında flutter run yazın
- Uygulama emulator'de açılacak

11. SONUÇ

Bu proje, küçük ve orta büyüklükteki işletmeler (KOBİ) için kapsamlı bir stok takip sistemi geliştirmeyi hedeflemiştir. Proje sürecinde mobil ve backend teknolojileri kullanılarak bir envanter yönetim sistemi oluşturulmuştur.

11.1 Proje Kazanımları

Teknik Beceriler:

Proje geliştirme sürecinde aşağıdaki teknik konularda deneyim kazanılmıştır:

Backend Geliştirme:

- ASP.NET Core Web API ile RESTful servis mimarisi tasarımı ve implementasyonu
- Veritabanı Yönetimi: SQL Server kullanımı, tablo tasarımı, ilişkisel veri modelleme ve normalizasyon
- ORM Kullanımı: Entity Framework Core ile Code-First yaklaşımı, migration yönetimi ve LINQ sorguları
- API Dokümantasyonu: Swagger entegrasyonu ve API test süreçleri

Mobile Frontend Geliştirme:

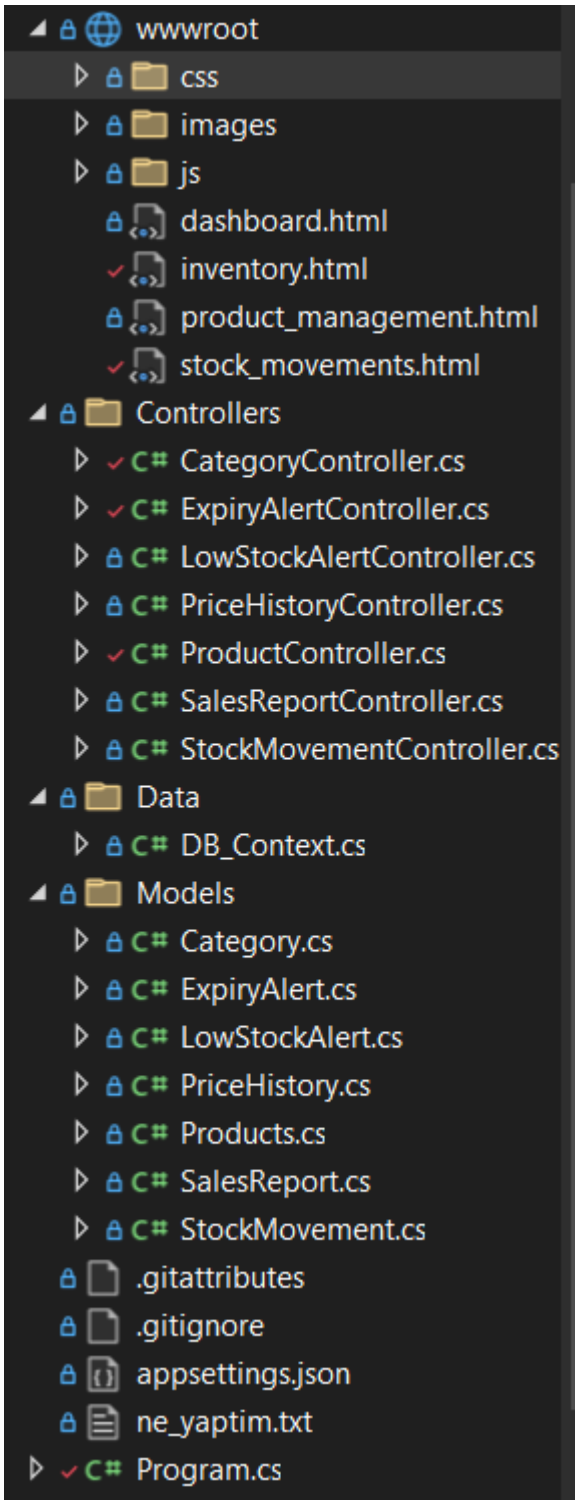
- Flutter framework ile cross-platform mobil uygulama geliştirme (iOS/Android)
- Dart programlama dili kullanımı
- Material Design prensiplerine uygun UI/UX tasarımı
- StatefulWidget ile state management

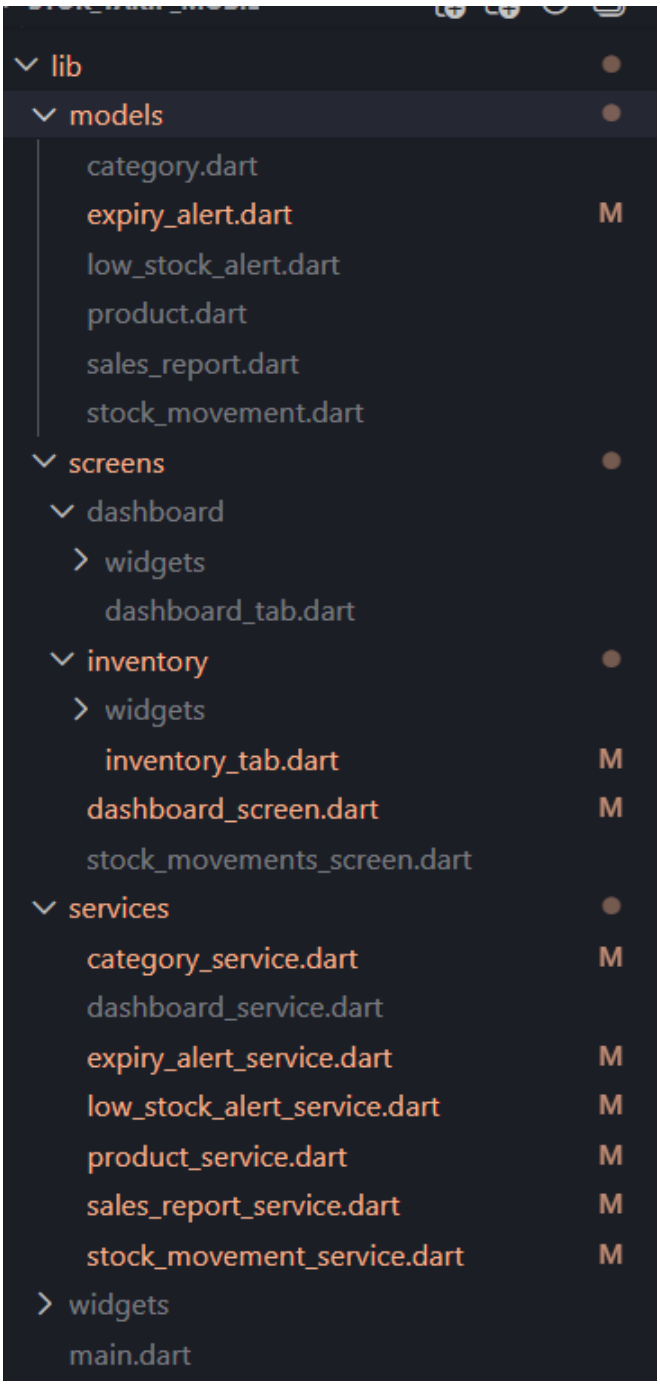
API İletişimi:

- HTTP package kullanımı, asenkron programlama (async/await) ve JSON veri formatı ile çalışma
- RESTful API entegrasyonu ve error handling
- Versiyon Kontrol:
- Git ve GitHub kullanımı, commit yönetimi ve proje dokümantasyonu

12. EKLER

EK 1: Proje Dosya Yapısı





EK 2: Veritabanı Diyagramı