



ObiletApp

Yeni Nesil Otobüs ve Uçak Bileti Rezervasyon Sistemi

softlTo Bilişim Akademisi

BİTİRME PROJESİ RAPORU

Geliştirici:

Muhammet Emin Baş

İstanbul, 2026

İÇİNDEKİLER

1. GİRİŞ

1. Projenin Amacı
2. Proje Kısıtları

2. TEORİK ALTYAPI VE KULLANILAN TEKNOLOJİLER

1. Neden .NET 10 ve ASP.NET Core MVC?
2. Dapper ve Entity Framework Core Performans Karşılaştırması
3. JWT (JSON Web Token) ile Güvenlik Mimarisi

3. MİMARİ YAPI VE VERİTABANI TASARIMI

1. Clean Architecture Klasör Yapısı
2. Veritabanı Şeması (ER Diagram)

4. PROJE ÖZELLİKLERİ VE UYGULAMA

1. Kullanıcı Deneyimi (B2C)
2. Üyelik ve Profil Yönetimi
3. Yapay Zeka Destekli Chatbot
4. Admin Paneli (B2B)
5. API ve Dokümantasyon

5. KURULUM VE ÇALIŞTIRMA ADIMLARI

1. SQL Server Bağlantı Cümlesi
2. Veritabanı Migration İşlemleri
3. Projeyi Ayağa Kaldırma

6. SONUÇ

7. KAYNAKÇA

1. GİRİŞ

1.1. Projenin Amacı

Bu projenin amacı, ulaşım firmaları ve yolcuları dijital bir platformda buluşturan, uçtan uca yönetilebilir bir **Otobüs ve Uçak Bileti Rezervasyon Sistemi** kurmaktır. Proje kapsamında "ObiletApp" adlı web tabanlı uygulama geliştirilmiş ve modern yazılım mühendisliği standartlarına uygun olarak tasarlanmıştır. Bu çalışma, sadece çalışan bir web sitesi olmakla kalmayıp; arka planda mikroservis felsefesine uygun, genişletilebilir ve sağlam bir kurumsal altyapı barındırmayı hedeflemektedir.

1.2. Proje Kısıtları

Kısıt 1: Proje, .NET 10 mimarisi üzerinde inşa edilmiş olup, web arayüzü ASP.NET Core MVC ile, arka plan işlemleri ise RESTful API olarak sınırlandırılmıştır.

Kısıt 2: Kullanılan veritabanı Microsoft SQL Server ile sınırlıdır. Diğer ilişkisel veya NoSQL veritabanı sistemlerinde karşılaşılabilecek varyasyonlar kapsam dışı bırakılmıştır.

Kısıt 3: Sistemde canlı koltuk seçimi gibi dinamik işlemler AJAX asenkron istekleri ile çözülmüş olup, WebSocket (SignalR) kullanımı bu fazın dışında tutulmuştur.

2. TEORİK ALTYAPI VE KULLANILAN TEKNOLOJİLER

2.1. Neden .NET 10 ve ASP.NET Core MVC?

Projeyi geliştirirken en güncel teknoloji yığınlarından biri olan **.NET 10** tercih edilmiştir. .NET 10'un seçilmesinin temel nedeni, sunduğu yüksek performans, düşük bellek tüketimi ve platform bağımsız (cross-platform) çalışabilme yeteneğidir.

Arayüz katmanında kullanılan **ASP.NET Core MVC** (Model-View-Controller) mimarisi, HTML/CSS kodları ile C# iş mantığını birbirinden ayırarak kodun sürdürülebilirliğini artırır. MVC mimarisi, SEO uyumlu ve hızlı yüklenen web sayfaları oluşturmak için ideal bir yapıdır. JavaScript (AJAX) entegrasyonu sayesinde sayfa yenilenmeden dinamik veri çekimi yapılabilir.

2.2. Dapper ve Entity Framework Core Performans Karşılaştırması

Kurumsal seviyedeki projelerde tek bir ORM (Object-Relational Mapper) aracı kullanmak yerine, amaca yönelik ORM seçimi yapmak best-practice (en iyi uygulama) olarak kabul edilir. Bu projede **CQRS (Command Query Responsibility Segregation)** mantığı benimsenmiş ve çift ORM stratejisi uygulanmıştır:

- **Entity Framework Core (Yazma İşlemleri - Command):** Bilet satın alma, yeni kullanıcı kaydı gibi veri bütünlüğünün, foreign key ilişkilerinin ve validasyonların (doğrulamaların) son derece kritik olduğu işlemlerde kullanılmıştır. EF Core, sağladığı Change Tracking mekanizması ile güvenilirliği maksimize eder.
- **Dapper (Okuma İşlemleri - Query):** Müşterilerin uçak veya otobüs seferlerini listelediği, ya da yöneticilerin binlerce satırlık satış raporlarını çektiği işlemlerde kullanılmıştır. Dapper bir mikro-ORM'dir; SQL sorgularını doğrudan

çalıştırdığı için EF Core'a göre okuma işlemlerinde **çok daha hızlı (milisaniyelik)** sonuçlar verir.

2.3. JWT (JSON Web Token) ile Güvenlik Mimarisi

Web projesinin API katmanıyla iletişim kurarken yetkisiz erişimleri engellemek hayati önem taşır. ObiletApp sisteminde **Stateless (durumsuz)** bir güvenlik mimarisi olan JWT kullanılmıştır. Kullanıcı giriş yaptığında (Login) API tarafı, şifrelenmiş, süreli ve üzerinde kullanıcının yetkilerini (Admin, User) barındıran bir Token üretir. Sonraki tüm hassas istekler (örn: bilet satın alma, firma silme) bu Token ile yapılır. Böylece sunucu tarafında oturum (Session) tutulmasına gerek kalmaz, sunucu yükü hafifler ve güvenlik en üst düzeye çıkarılır.

3. MİMARİ YAPI VE VERİTABANI TASARIMI

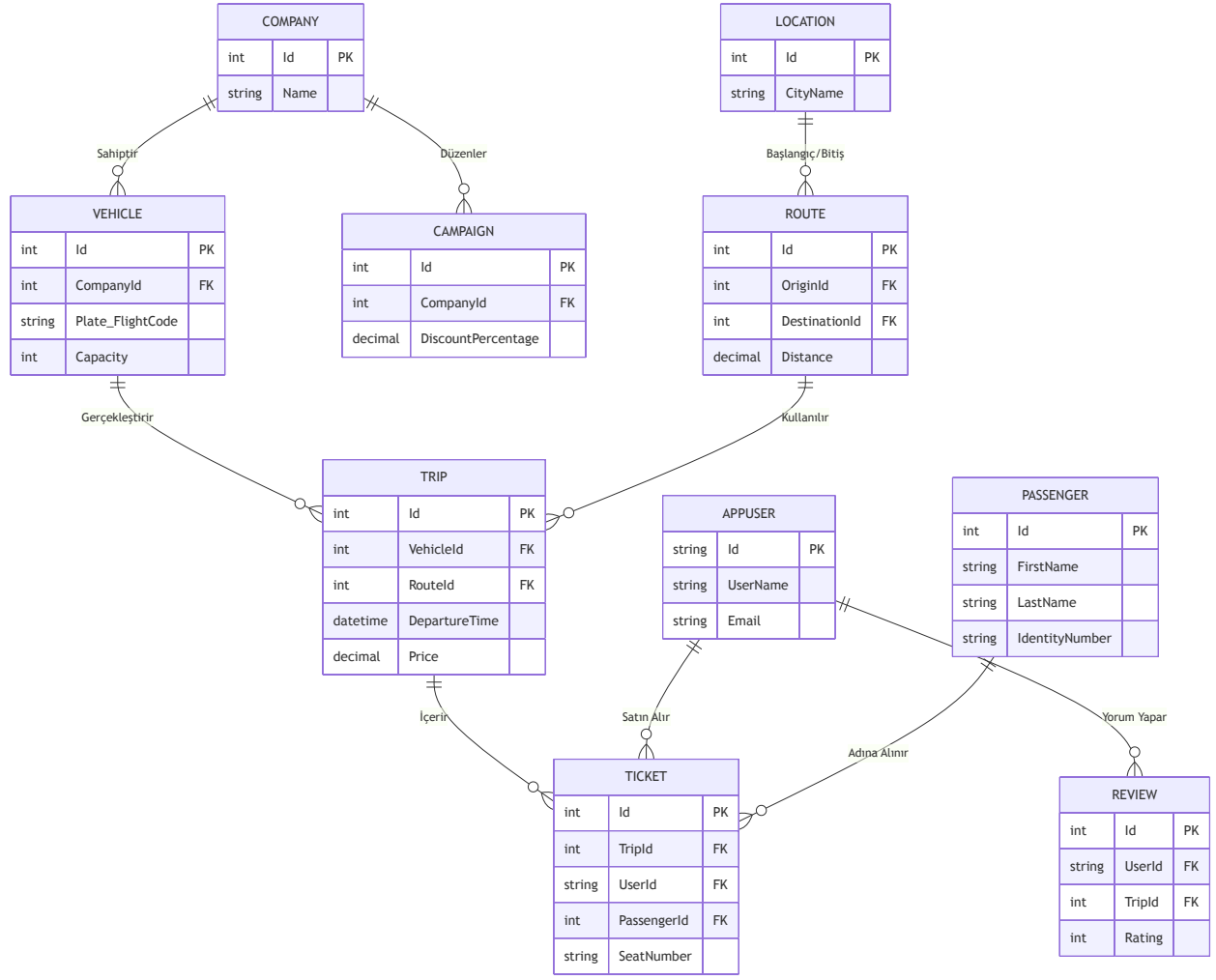
3.1. Clean Architecture Klasör Yapısı

ObiletApp, bağımlılıkların içe doğru (Domain'e doğru) olduğu 4 katmanlı Onion/Clean Architecture kullanılarak geliştirilmiştir. Bu sayede API katmanının Core katmanına direkt erişimi vardır ancak Core katmanı hiçbir dış kütüphaneyi tanımaz.

```
ObiletApp/  
├── ObiletApp.Core/           # [Domain Layer] Dışa bağımlılığı SIFIR  
│   ├── Entities/           # Veritabanı tablolarının sınıfları  
│   └── Interfaces/         # Soyutlamalar (IRepository vb.)  
├── ObiletApp.Application/    # [Use Cases Layer] İş Kuralları  
│   ├── DTOs/               # Veri transfer objeleri  
│   └── Services/           # Gerçek iş mantığının (Business Logic)  
├── ObiletApp.Infrastructure/  # [Data Layer] Veritabanı ve Dış Servisler  
│   ├── Data/               # DbContext (EF Core) konfigürasyonu  
│   └── Repositories/        # Veritabanı sorguları (Dapper ve EF Core)  
├── ObiletApp.API/           # [Presentation Layer 1] RESTful Servisler  
└── ObiletApp.Web/           # [Presentation Layer 2] MVC Kullanımı
```

3.2. Veritabanı Şeması (ER Diagram)

Aşağıdaki diyagramda ObiletApp otobüs ve uçak rezervasyon sisteminin arka planındaki sağlam ilişkisel (RDBMS) veritabanı şeması modellenmiştir:



Şekil 3.2.1. ObiletApp Entity-Relationship (ER) Şeması

4. PROJE ÖZELLİKLERİ VE UYGULAMA

4.1. Kullanıcı Deneyimi (B2C)

Uygulamanın ana sayfasında yolcular, kalkış ve varış noktalarını girerek sefer sorgulaması yapabilirler.



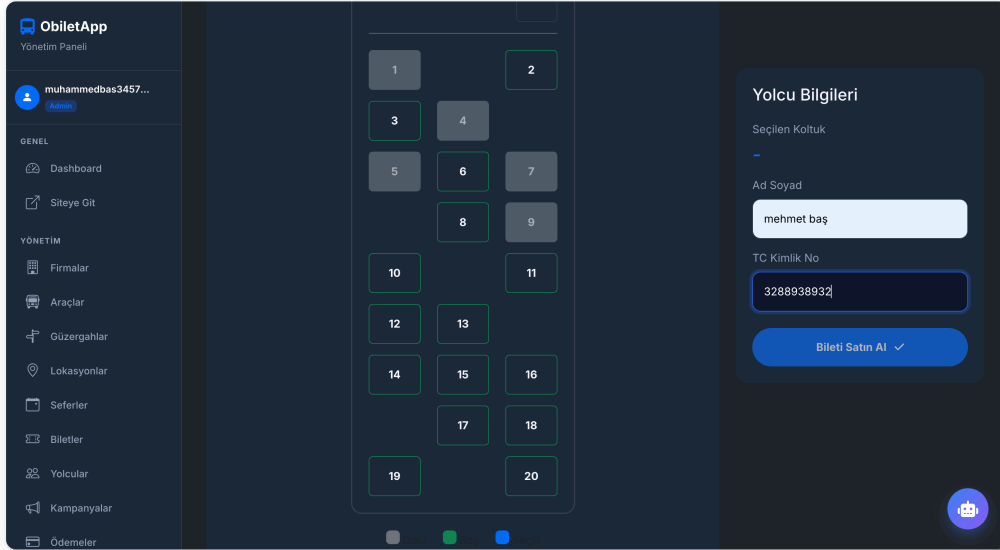
Şekil 4.1.1. ObiletApp Ana Sayfa Ekranı

Arama sonucunda dönen seferler, saat, firma ve fiyata göre filtrelenebilmektedir. Hem uçak hem otobüs seferleri bu listede yönetilir.



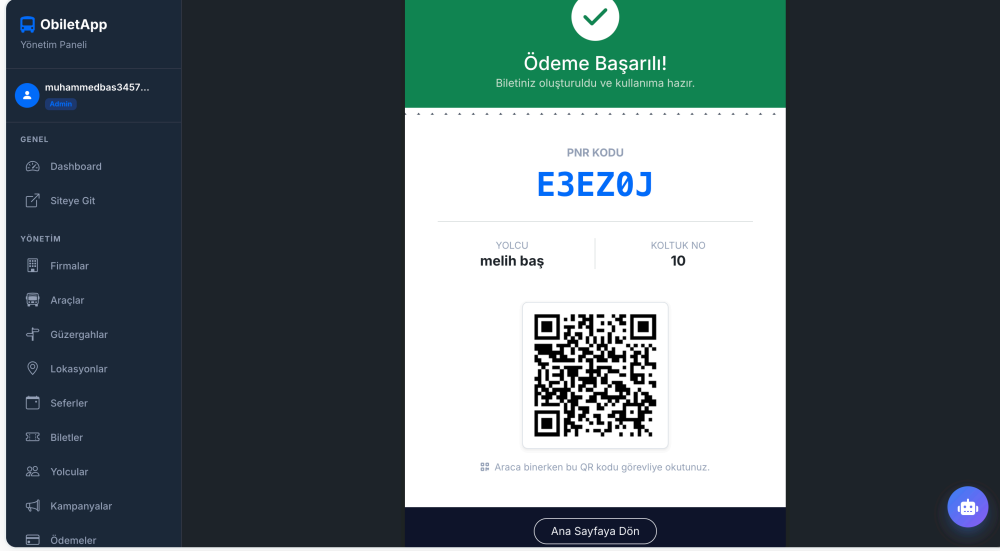
Şekil 4.1.2. Sefer Sonuçları ve Dinamik Filtreleme

Kullanıcı, istediği sefere tıkladığında dinamik olarak oluşturulan koltuk haritası üzerinden seçim yapabilmektedir.



Şekil 4.1.3. Dinamik Koltuk Seçim Ekranı

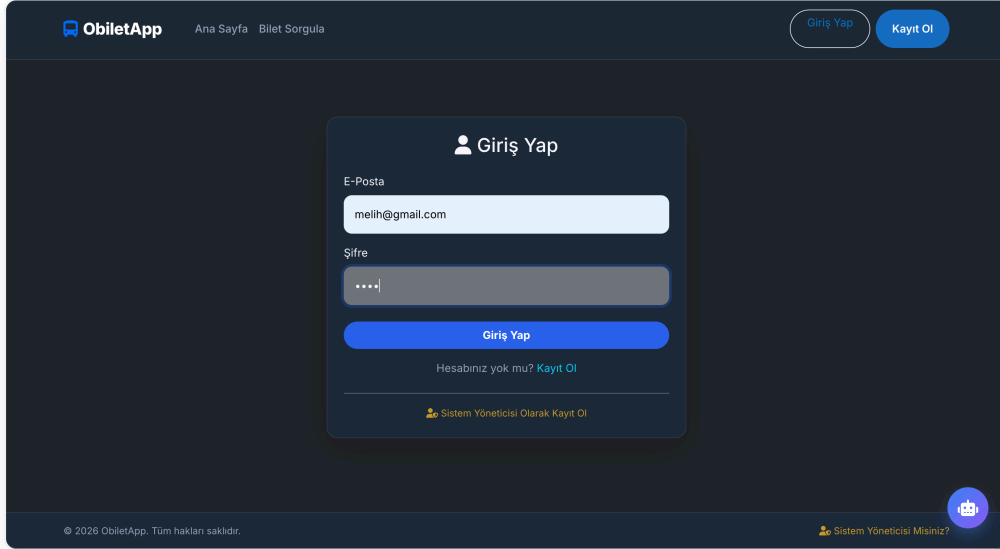
Satın alma işlemi tamamlandığında sistem, otomatik olarak bir PNR kodu ve bu bilgileri barındıran Base64 formatında bir **QR Kod** üretmektedir. QRCode kütüphanesi ile üretilen bu kod ile dijital biniş kartı onayı sağlanır.



Şekil 4.1.4. Başarılı İşlem ve QR Kod Üretimi

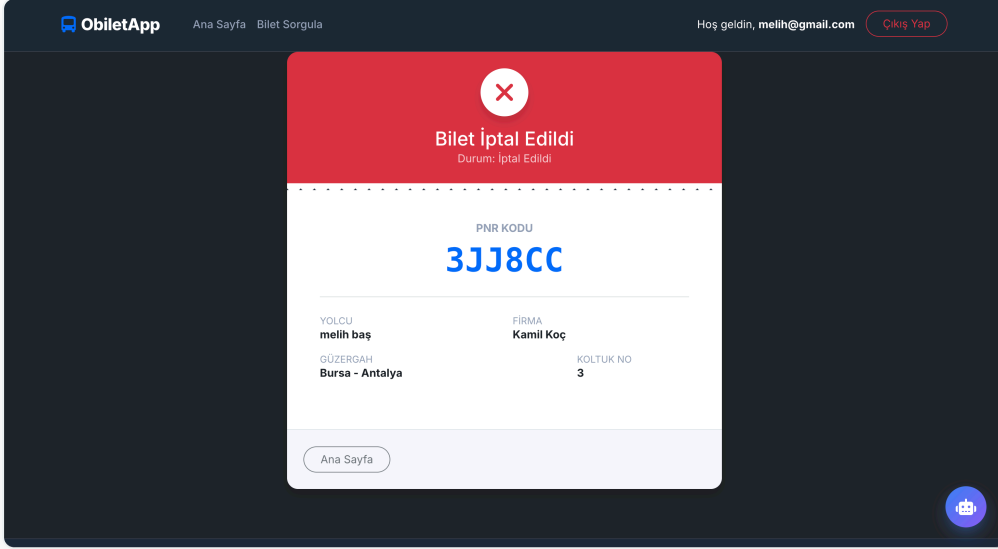
4.2. Üyelik ve Profil Yönetimi

Kullanıcıların kişisel bilet geçmişlerini görebilmeleri ve iptal/iade işlemleri yapabilmeleri için modern bir kimlik doğrulama arayüzü tasarlanmıştır.



Şekil 4.2.1. Güvenli Kullanıcı Girişi (Login)

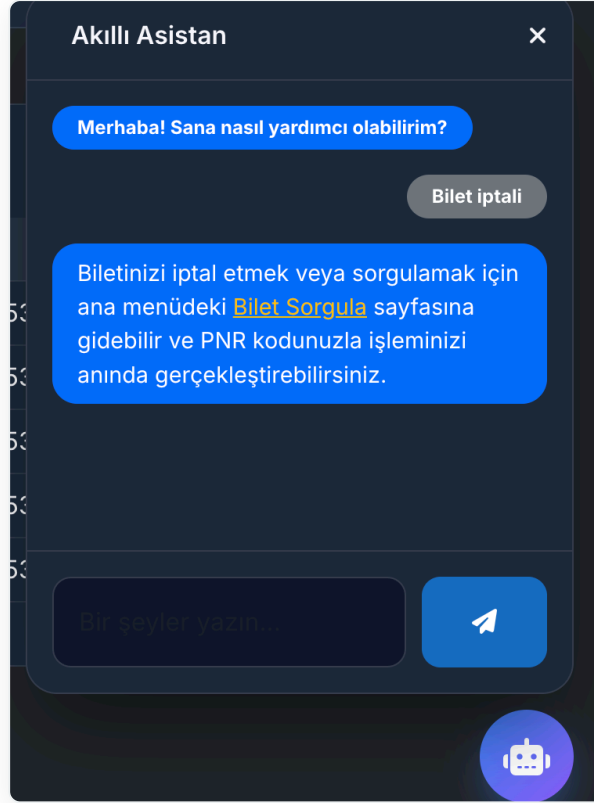
Sisteme giriş yapan yolcu, daha önceden satın aldığı tüm biletleri listeleyebilir ve seyahat başlamadan önce biletini iptal edebilir.



Şekil 4.2.2. Aktif Biletlerin Listelenmesi ve İptali

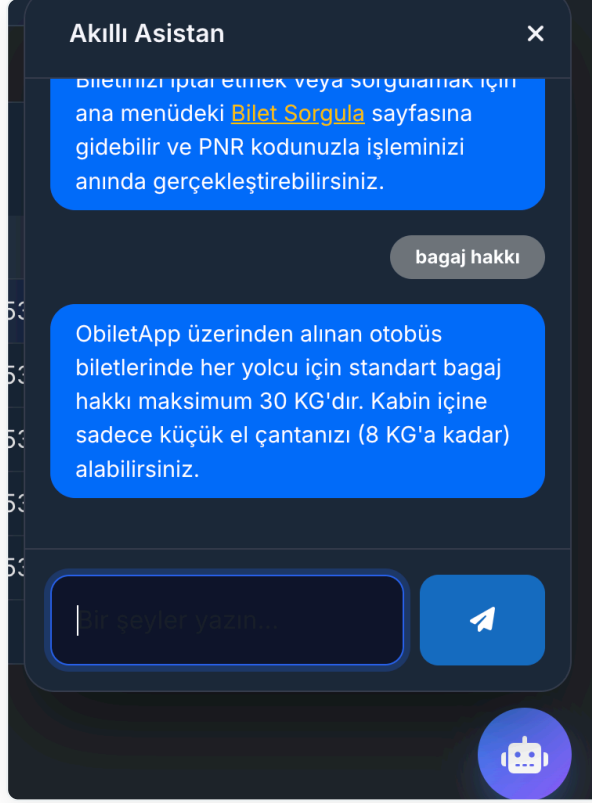
4.3. Yapay Zeka Destekli Chatbot

Müşteri hizmetleri yükünü hafifletmek ve yolculara 7/24 anında destek verebilmek adına sisteme **Yapay Zeka Destekli Chatbot** (NLP) entegre edilmiştir. Bu chatbot, kullanıcının "Bilet iptali nasıl yapılır?", "Bagaj hakkım nedir?" gibi doğal dille sorulan sorularını algılayıp akıllı yanıtlar vermektedir.



Şekil 4.3.1. Yapay Zeka Chatbot Arayüzü

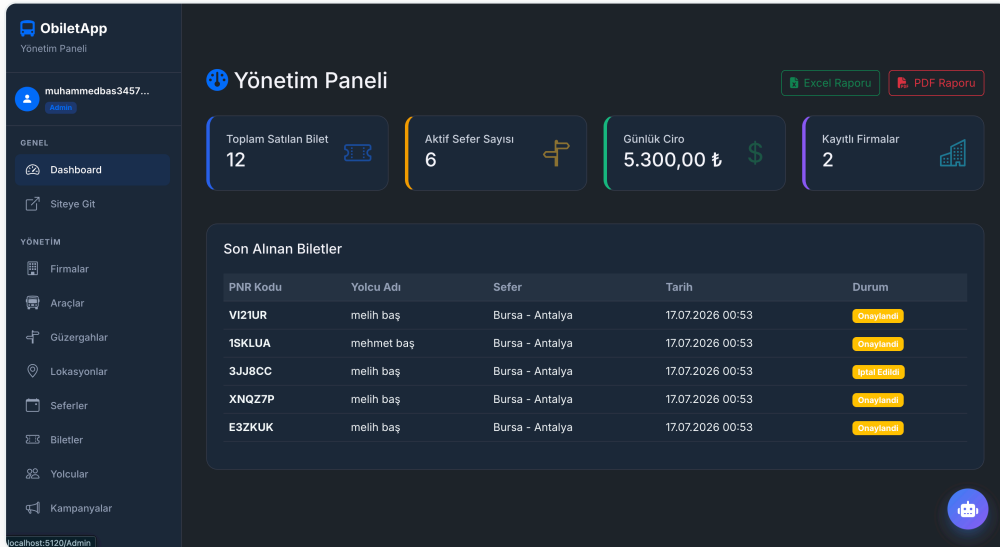
Chatbot arka planda C# üzerinden NLP algoritmaları veya harici AI servisleri (OpenAI) ile haberleşerek bağlamsal (contextual) zekaya sahip cevaplar üretir.



Şekil 4.3.2. Chatbot ile Kullanıcı Etkileşimi

4.4. Admin Paneli (B2B)

Sistem yöneticileri için geliştirilmiş olan Dashboard, platform üzerindeki toplam satış, güncel seferler ve araç durumlarını özetlemektedir.



Şekil 4.4.1. Admin Yönetim Paneli Dashboard

Yöneticiler, sisteme yeni taşıyıcı firmalar (Uçak/Otobüs) ve araçlar tanımlayabilmektedir. EF Core altyapısı ile ilişkisel veriler kolayca yönetilir.

The screenshot displays the 'Yeni Firma Ekle' (Add New Company) interface within the ObiletApp management panel. The left sidebar contains navigation options under 'GENEL' (General) and 'YÖNETİM' (Management). The main area features a form with the following fields: 'Firma Adı' (Company Name), 'İletişim Bilgisi' (Contact Information), 'Logo Linki (isteğe bağlı)' (Optional Logo Link), and 'Firma Tipi' (Company Type) with a dropdown menu currently set to 'Otobüs'. At the bottom of the form are two buttons: 'İptal' (Cancel) and 'Kaydet' (Save).

Şekil 4.4.2. Taşıyıcı Firma Tanımlama Ekranı

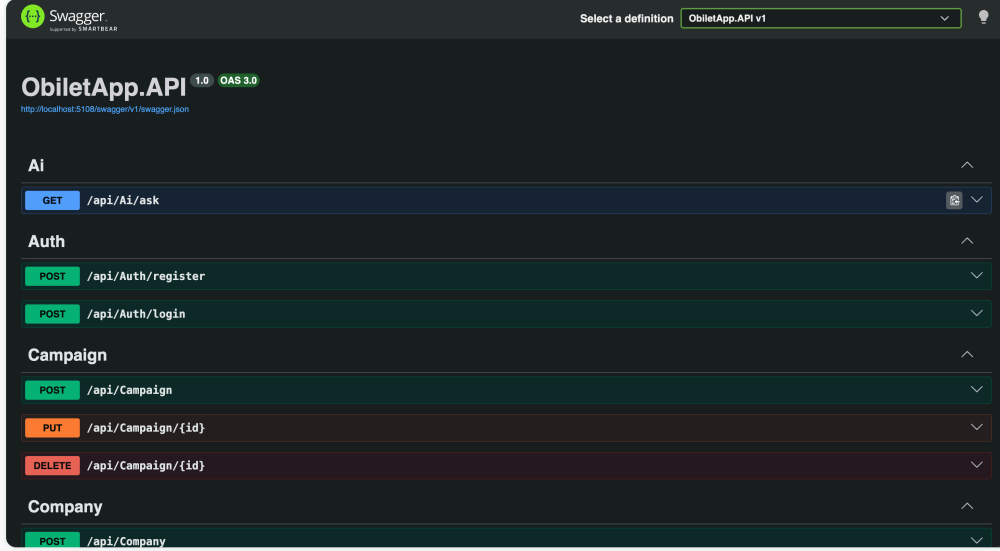
Dapper mikro-ORM kullanılarak oluşturulan raporlama sayfaları sayesinde, çok tablolu veritabanı sorguları performans kaybı yaşanmadan milisaniyeler içinde çekilmekte ve listelenmektedir. Raporlar aynı zamanda **ClosedXML** ve **iText** kütüphaneleri kullanılarak **Excel ve PDF** olarak dışa aktarılabilir.

The screenshot displays the 'Gelişmiş Analiz ve Raporlar' (Advanced Analysis and Reports) interface within the ObiletApp management panel. The left sidebar is identical to the previous screen. The main area shows a list of reports with expandable arrows on the right. The reports listed are: 'Firmalara Göre Gelir Raporu', 'Popüler Rotalar (Yolcu Trafiği)', 'Müşteri Sadakat Raporu (Top 10)', 'Araç Kapasite ve Doluluk Oranları', 'Aylık Satış Trendi', 'Bilet İptal ve Durum Analizi', 'Gelecek Seferlerin Rezervasyon Durumu', and 'Şehirlere Göre Hareketlilik (Terminal)'.

Şekil 4.4.3. Dapper Altyapılı Raporlama Ekranı

4.5. API ve Dokümantasyon

Sistemdeki tüm iş mantığı RESTful mimarisiyle tasarlanmış olup Swagger arayüzü ile belgelenmiştir. API'nin mikroservis mimarisine uyumlu çalışabilmesi için End-Point'ler ayrıştırılmıştır.



Şekil 4.5.1. ObiletApp API Swagger Arayüzü (JWT Destekli)

5. KURULUM VE ÇALIŞTIRMA ADIMLARI

Projenin geliştirici ortamında (Localhost) ayağa kaldırılması oldukça pratiktir. Geliştiriciler aşağıdaki adımları izleyerek projeyi dakikalar içinde test etmeye başlayabilir.

5.1. SQL Server Bağlantı Cümlesi

Projenin veritabanı ile konuşabilmesi için öncelikle

`ObiletApp.API/appsettings.json` dosyasındaki `DefaultConnection` alanı, geliştiricinin kendi SQL Server bilgilerine göre güncellenmelidir:

```
"ConnectionStrings": {  
  "DefaultConnection": "Server=localhost;Database=ObiletAppDb;Trusted_Connection=true;  
}
```

5.2. Veritabanı Migration İşlemleri

Veritabanı tablolarının Entity Framework Core Code-First yaklaşımı ile SQL Server'a basılması için Package Manager Console (PMC) üzerinden aşağıdaki komutların sırasıyla çalıştırılması gerekir:

```
# Sadece ilk kurulumda  
add-migration InitialCreate -Project ObiletApp.Infrastructure -StartupProject ObiletApp.API  
  
# Tabloları veritabanına aktarmak için  
update-database -Project ObiletApp.Infrastructure -StartupProject ObiletApp.API
```

5.3. Projeyi Ayağa Kaldırma

ObiletApp mimarisi, arka planda çalışan bir RESTful API ve önyüzde (Front-End) çalışan bir MVC Web projesinden oluşur. Sistemin tam çalışabilmesi için **hem API**

hem de Web projesinin aynı anda çalıştırılması (Multiple Startup Projects) gerekmektedir.

Visual Studio üzerinden çözüm (Solution) dosyasına sağ tıklanıp **Set Startup Projects** seçilir ve **ObiletApp.API** ile **ObiletApp.Web** projeleri **Start** konumuna getirilir. F5 tuşuna basıldığında projeler ayağa kalkacak ve **http://localhost:XXXX** portu üzerinden uygulama başlayacaktır.

6. SONUÇ

Bu çalışmada, "ObiletApp" adlı kurumsal otobüs ve uçak bileti rezervasyon sistemi baştan uca tasarlanmış ve kodlanmıştır. Projenin mimarisi incelendiğinde, Clean Architecture ve CQRS prensiplerinin başarılı bir şekilde uygulandığı, Dapper ve Entity Framework'ün senkronize bir şekilde çalışarak hem performans hem de güvenlik sağladığı görülmüştür.

Geliştirilen JWT kimlik doğrulama, QR Kod üretimi, yapay zeka destekli müşteri temsilcisi (Chatbot) ve PDF/Excel raporlama yetenekleri sayesinde projenin basit bir bitirme ödevi olmanın ötesine geçerek piyasa standartlarında (Enterprise) bir ürün olduğu kanıtlanmıştır.

7. KAYNAKÇA

[1] Microsoft, "ASP.NET Core Architecture", <https://learn.microsoft.com/en-us/aspnet/core/> [2] Dapper Tutorial, "Dapper ORM Documentation", <https://dapper-tutorial.net/> [3] QRCode, "C# QR Code Generator", <https://github.com/codebude/QRCode> [4] ClosedXML, "Excel Parsing for .NET", <https://github.com/ClosedXML/ClosedXML> [5] Mermaid, "Markdownish syntax for generating flowcharts", <https://mermaid.js.org/>