# YOLDANÇEK - OTO KURTARMA PLATFORMU

**Final Proje Raporu**

**Proje Adı:** Yoldançek - Oto Kurtarma Platformu

**Öğrenci:** Muhammed Emin KEÇİK

**Öğrenci Numarası:** 22290749

**Ders:** BLM4531

**GitHub Repository:** https://github.com/meminkecik/Blm4531

**Demo Video Linki:** https://drive.google.com/file/d/1NrLwVli5ZON4AOaheYPQK0_gynAzNC05/view?usp=sharing

📋 İÇİNDEKİLER

# 1. PROJE ÖZETİ

## 1.1 Projenin Amacı

Yoldançek, kullanıcıların GPS koordinatlarını kullanarak en yakın oto kurtarma firmalarını ve çekicileri bulmasını sağlayan bir web platformudur. Platform iki ana kullanıcı grubuna hizmet vermektedir:

1. **Araç Sahipleri (Kullanıcılar):** Yolda kaldıklarında konum bazlı arama yaparak en yakın çekiciyi bulabilir, firma bilgilerini görebilir, yorum/puan bırakabilir ve şikayet bildirebilirler.

2. **Oto Kurtarma Firmaları:** Sisteme kayıt olarak çekici araçlarını, şoför bilgilerini ve çalışma bölgelerini yönetebilirler.

## 1.2 Temel Özellikler

| Özellik | Açıklama |

|---------|----------|

| Konum Bazlı Arama | Haversine formülü ile en yakın çekicileri bulma |

| Firma Yönetimi | Kayıt, giriş, profil güncelleme |

| Çekici Yönetimi | Ekleme, düzenleme, silme, aktif/pasif durumu |

| Yorum & Puan Sistemi | Kullanıcıların çekicilere puan ve yorum bırakması |

| Şikayet Sistemi | Kötüye kullanım raporlama |

| Admin Paneli | Firma/çekici yönetimi, şikayet takibi |

| KVKK Uyumu | Kişisel veri koruma mevzuatına uygunluk |

## 2. KULLANILAN TEKNOLOJİLER

### 2.1 Backend Teknolojileri

| Teknoloji | Versiyon | Kullanım Amacı |
|---|---|---|
| .NET | 8.0 | Web API framework |
| Entity Framework Core | 8.0 | ORM (Object-Relational Mapping) |
| PostgreSQL | 15 | İlişkisel veritabanı |
| JWT | - | Kimlik doğrulama ve yetkilendirme |
| AutoMapper | 12.0 | Nesne dönüşüm kütüphanesi |
| Swagger/OpenAPI | - | API dokümantasyonu |

### 2.2 Frontend Teknolojileri

| Teknoloji | Versiyon | Kullanım Amacı |
|---|---|---|
| Next.js | 16.1.1 | React framework (App Router) |
| React | 19.2.3 | UI kütüphanesi |
| TypeScript | 5 | Tip güvenli JavaScript |
| CSS Modules | - | Bileşen bazlı stil yönetimi |

### 2.3 DevOps & Deployment

| Teknoloji | Kullanım Amacı |

|---|---|

| Docker | Containerization |

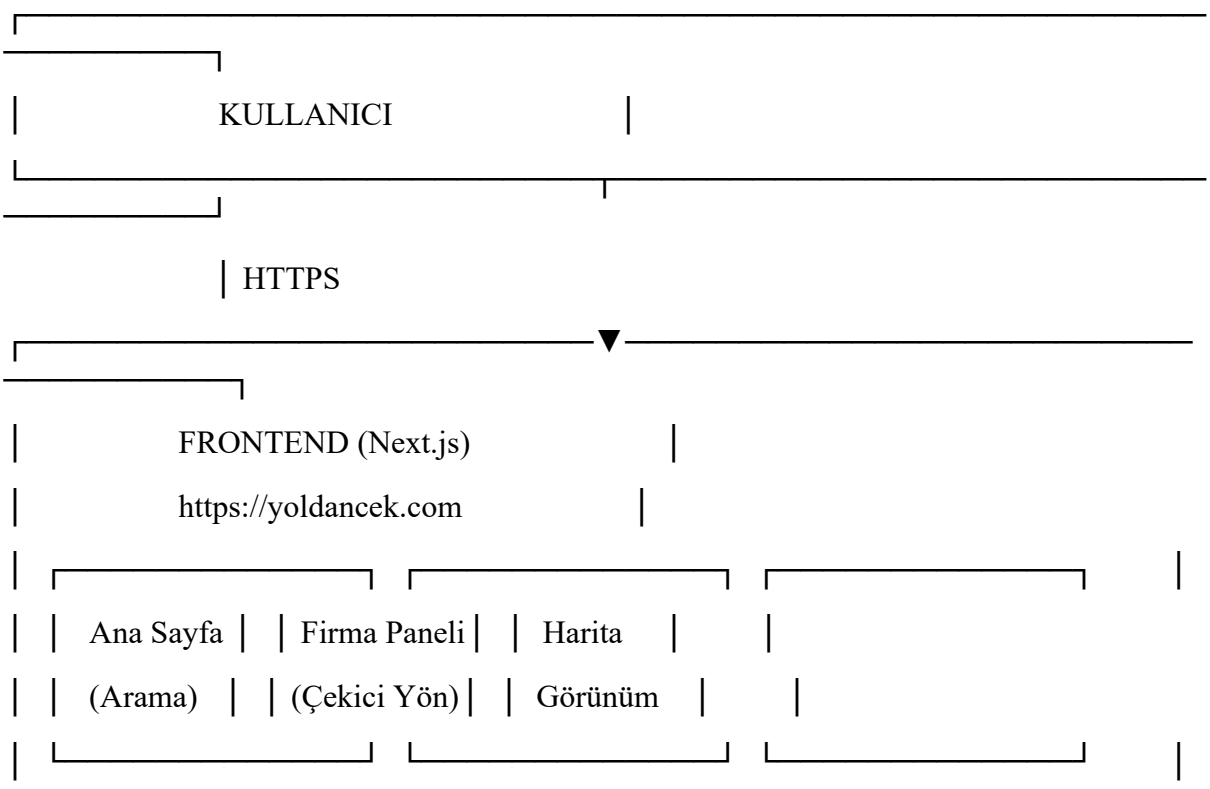
| Docker Compose | Multi-container orchestration |

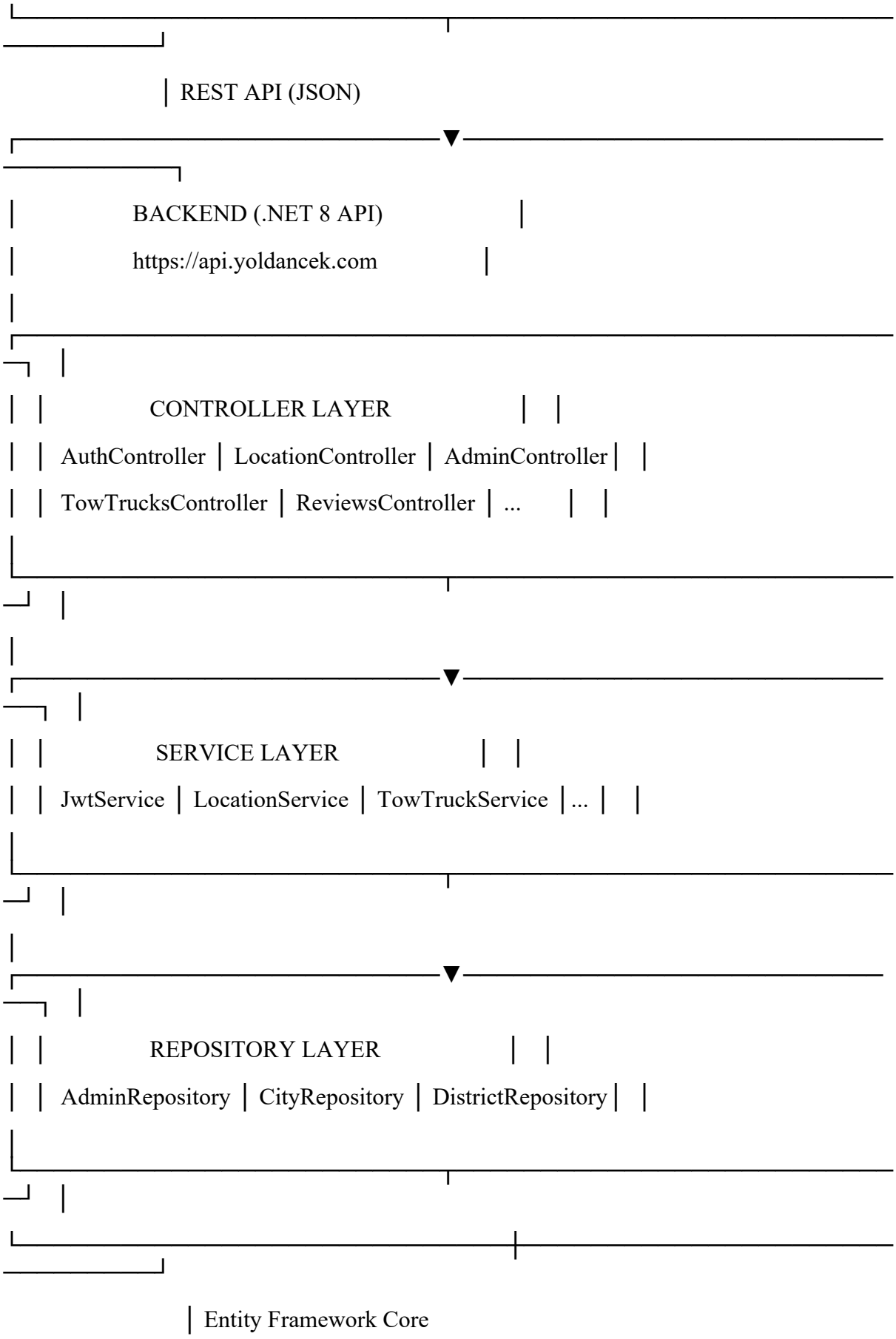
| GitHub Actions | CI/CD pipeline |

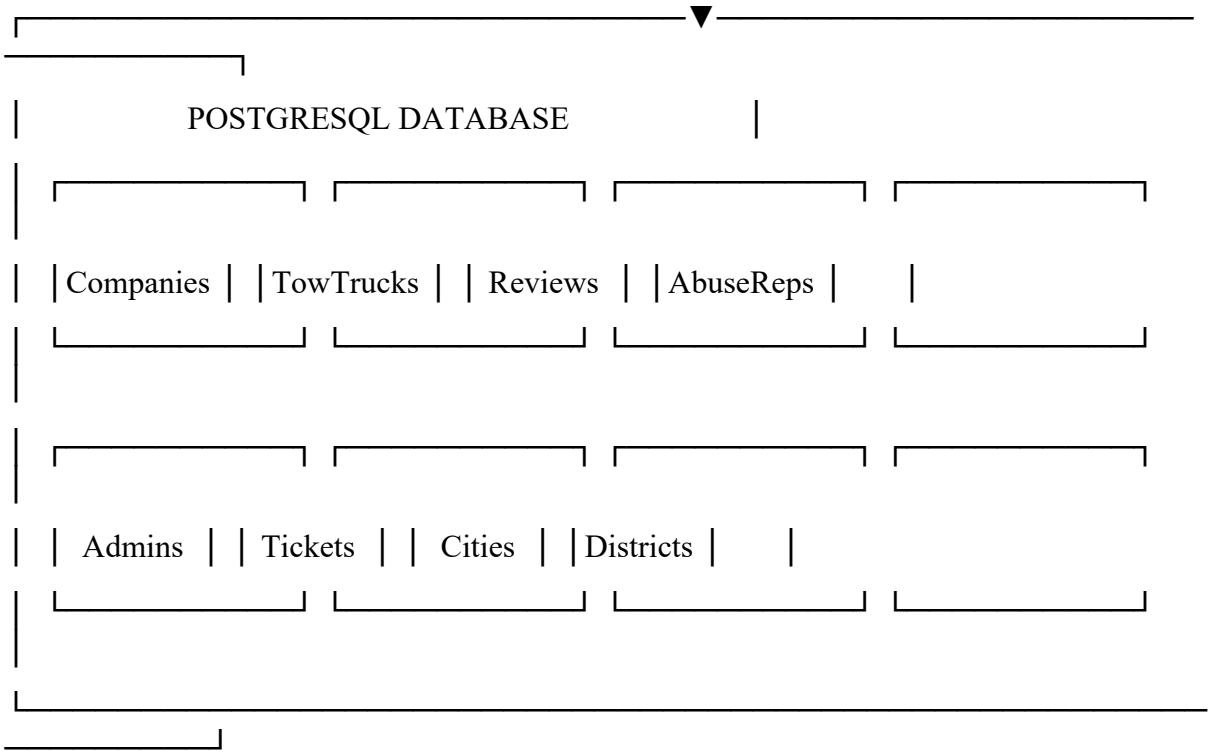
| Nginx | Reverse proxy |

## 3. SİSTEM MİMARİSİ

### 3.1 Genel Mimari




REST API (JSON)
BACKEND (.NET 8 API)
https://api.yoldancek.com
CONTROLLER LAYER
AuthController | LocationController | AdminController
TowTrucksController | ReviewsController | ...
SERVICE LAYER
JwtService | LocationService | TowTruckService | ...
REPOSITORY LAYER
AdminRepository | CityRepository | DistrictRepository
Entity Framework Core




### 3.2 Katmanlı Mimari

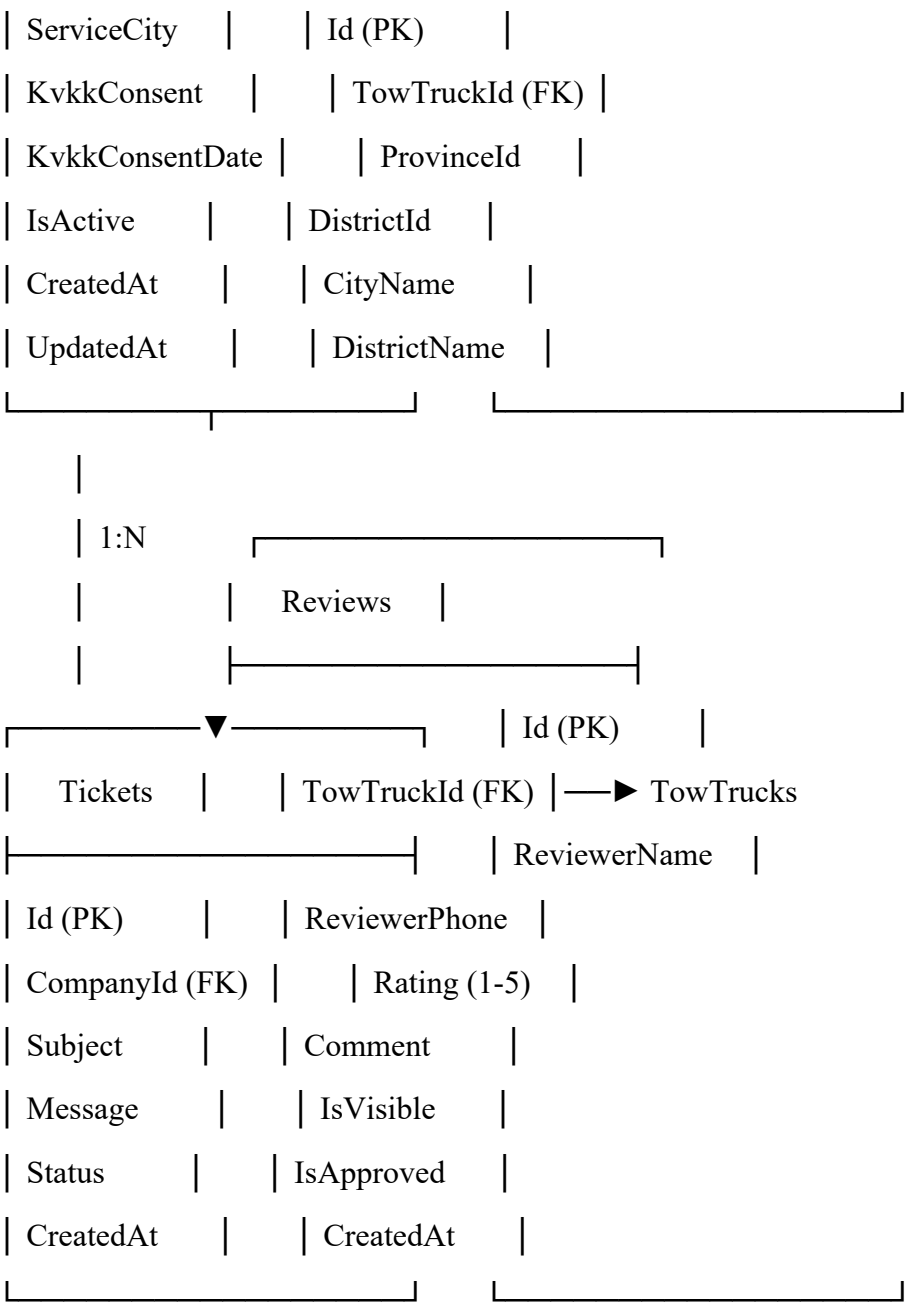
Proje **N-Tier (Katmanlı) Mimari** kullanmaktadır:

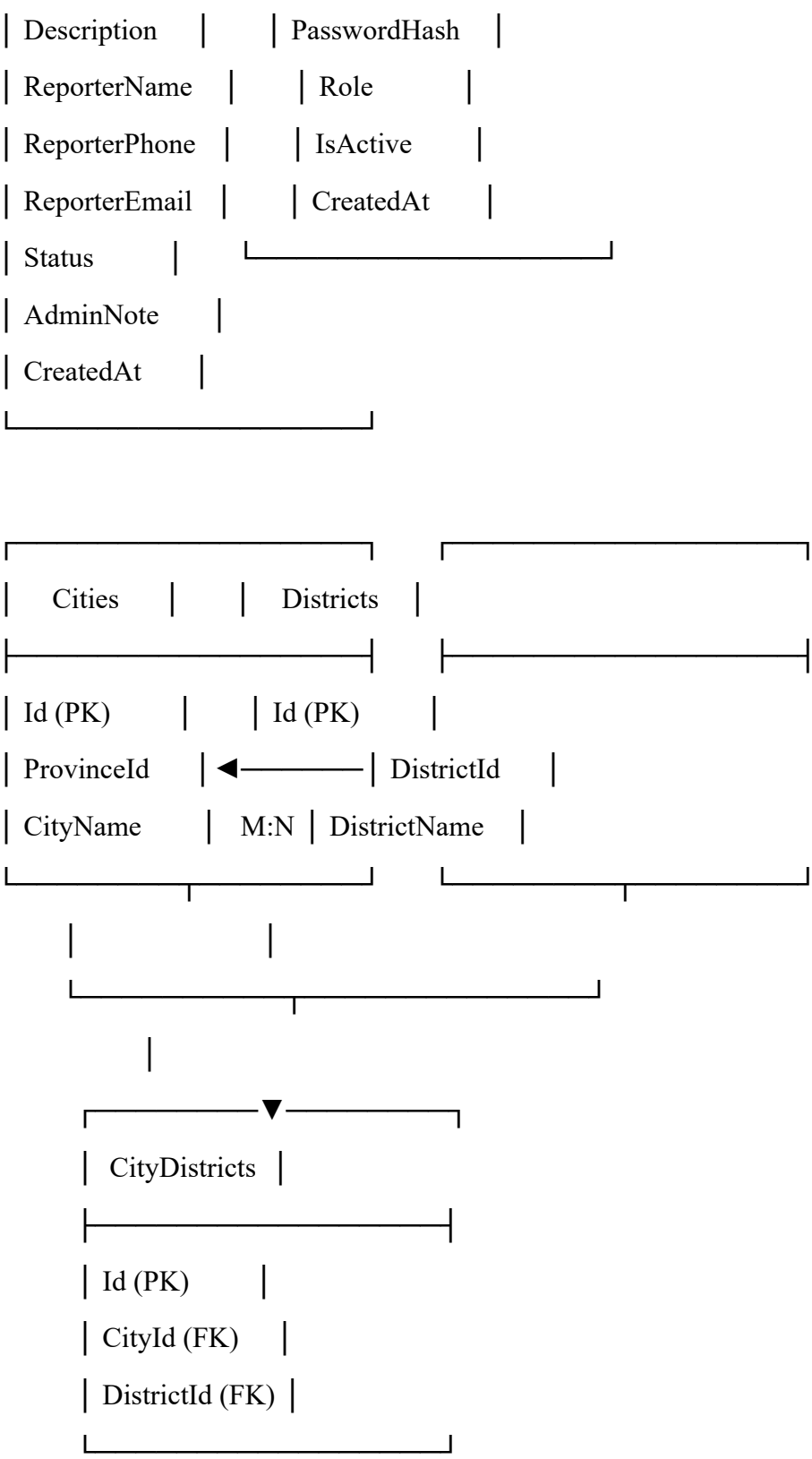
1. **Presentation Layer (Controller):** HTTP isteklerini alır, yanıtları döndürür
2. **Business Layer (Service):** İş mantığını içerir
3. **Data Access Layer (Repository):** Veritabanı erişimini soyutlar
4. **Data Layer (Entity Framework):** ORM ile veritabanı işlemleri

## 4. VERİTABANI TASARIMI

### 4.1 Entity-Relationship Diyagramı

```
┌──────────────────────────┐      ┌──────────────────────────┐
│        Companies         │      │        TowTrucks         │
├──────────────────────────┤      ├──────────────────────────┤
│ Id (PK)                  │◄─────│ CompanyId (FK)           │
│ FirstName                │ 1:N  │ Id (PK)                  │
│ LastName                 │      │ LicensePlate             │
│ CompanyName              │      │ DriverName               │
│ Email (Unique)           │      │ DriverPhotoUrl           │
│ PhoneNumber              │      │ IsActive                 │
│ PasswordHash             │      │ CreatedAt                │
│ ProvinceId               │      │ UpdatedAt                │
│ DistrictId               │      └──────────┬───────────────┘
│ City                     │                 │
│ District                 │                 │ 1:N
│ FullAddress              │      ┌──────────▼───────────────┐
│ Latitude                 │      │      TowTruckAreas       │
│ Longitude                │      ├──────────────────────────┤
```


ServiceCity
Id (PK)
KvkkConsent
TowTruckId (FK)
KvkkConsentDate
ProvinceId
IsActive
DistrictId
CreatedAt
CityName
UpdatedAt
DistrictName
1:N
Reviews
Tickets
Id (PK)
TowTruckId (FK)
TowTrucks
ReviewerName
Id (PK)
ReviewerPhone
CompanyId (FK)
Rating (1-5)
Subject
Comment
Message
IsVisible
Status
IsApproved
CreatedAt
CreatedAt
AbuseReports
Admins
Id (PK)
Id (PK)
TowTruckId (FK)
Email (Unique)
CompanyId (FK)
FirstName
ReportType
LastName


Description
ReporterName
ReporterPhone
ReporterEmail
Status
AdminNote
CreatedAt
PasswordHash
Role
IsActive
CreatedAt
Cities
Id (PK)
ProvinceId
CityName
Districts
Id (PK)
DistrictId
DistrictName
M:N
CityDistricts
Id (PK)
CityId (FK)
DistrictId (FK)

### 4.2 Tablo Açıklamaları

| Tablo | Açıklama | Kayıt Sayısı (Örnek) |
|---|---|---|
| Companies | Oto kurtarma firmaları | ~50 |
| TowTrucks | Çekici araçlar | ~150 |
| TowTruckAreas | Çekici çalışma bölgeleri | ~300 |
| Reviews | Kullanıcı yorumları | ~500 |
| AbuseReports | Şikayet raporları | ~20 |
| Admins | Sistem yöneticileri | 1 |
| Tickets | İletişim talepleri | ~100 |
| Cities | Türkiye illeri | 81 |
| Districts | Türkiye ilçeleri | ~973 |
| CityDdistricts | İl-ilçe ilişkileri | ~973 |

## 5. BACKEND (API) YAPISI

### 5.1 Proje Klasör Yapısı

```
Yoldançek/
├── Controllers/           API endpoint'leri
│   ├── AuthController.cs          Kayıt, giriş işlemleri
│   ├── LocationController.cs      Konum bazlı arama
│   ├── CompaniesController.cs     Firma işlemleri
│   ├── TowTrucksController.cs     Çekici yönetimi
│   ├── ReviewsController.cs       Yorum sistemi
│   ├── AbuseReportsController.cs  Şikayet sistemi
```

```
│   ├── AdminController.cs          Admin işlemleri
│   ├── AddressController.cs        İl/ilçe verileri
│   ├── ProfileController.cs        Profil yönetimi
│   └── TicketsController.cs        İletişim talepleri
├── Services/               İş mantığı katmanı
│   ├── JwtService.cs               Token üretimi/doğrulama
│   ├── LocationService.cs          Mesafe hesaplama
│   ├── TowTruckService.cs          Çekici işlemleri
│   ├── ReviewService.cs            Yorum işlemleri
│   ├── AbuseReportService.cs       Şikayet işlemleri
│   ├── AdminService.cs             Admin işlemleri
│   ├── AddressService.cs           Adres işlemleri
│   ├── AddressHelperService.cs     Harici API entegrasyonu
│   ├── EmailService.cs             E-posta gönderimi
│   └── TicketService.cs            Ticket işlemleri
├── Repositories/           Veri erişim katmanı
│   ├── AdminRepository.cs
│   ├── CityRepository.cs
│   ├── DistrictRepository.cs
│   └── CityDistrictRepository.cs
├── Models/                 Veritabanı entity'leri
│   ├── Company.cs
│   ├── TowTruck.cs
│   ├── TowTruckArea.cs
│   ├── Review.cs
│   ├── AbuseReport.cs
│   ├── Admin.cs
│   ├── Ticket.cs
│   └── Address/
```

```
│       ├── City.cs
│       ├── District.cs
│       └── CityDistrict.cs
├── DTOs/                  Data Transfer Objects
│   ├── CompanyDto.cs
│   ├── CompanyRegistrationDto.cs
│   ├── AuthResponseDto.cs
│   ├── ReviewDto.cs
│   ├── AbuseReportDto.cs
│   └── TowTruck/
│       ├── TowTruckDto.cs
│       ├── TowTruckCreateDto.cs
│       └── UpdateTowTruckDto.cs
├── Mappings/              AutoMapper profilleri
│   ├── CompanyMappingProfile.cs
│   ├── TowTruckMappingProfile.cs
│   └── AdminMappingProfile.cs
├── Data/                  Veritabanı context
│   └── ApplicationDbContext.cs
├── Migrations/            EF Core migration'ları
├── Program.cs             Uygulama başlangıç noktası
├── Dockerfile             Docker konfigürasyonu
└── docker-compose.yml     Multi-container yapılandırma
```

## 5.2 Önemli Servis Açıklamaları

### 5.2.1 LocationService - Konum Bazlı Arama

**Haversine Formülü ile Mesafe Hesaplama:**

```csharp
// Services/LocationService.cs - Satır 54-68
private double CalculateDistance(double lat1, double lon1, double lat2, double lon2)
{
    const double R = 6371; // Dünya yarıçapı (km)

    var dLat = ToRadians(lat2 - lat1);
    var dLon = ToRadians(lon2 - lon1);

    var a = Math.Sin(dLat / 2)  Math.Sin(dLat / 2) +
            Math.Cos(ToRadians(lat1))  Math.Cos(ToRadians(lat2))
            Math.Sin(dLon / 2)  Math.Sin(dLon / 2);

    var c = 2  Math.Atan2(Math.Sqrt(a), Math.Sqrt(1 - a));

    return R  c; // Kilometre cinsinden mesafe
}
```

**Nasıl Çalışır:**

1. İki GPS koordinatı arasındaki açısal farklar hesaplanır

2. Haversine formülü küresel yüzey üzerindeki mesafeyi hesaplar

3. Dünya yarıçapı (6371 km) ile çarpılarak gerçek mesafe bulunur

**Kademeli Arama Algoritması:**

```csharp
// Services/LocationService.cs - Satır 19-100
public async Task<List<CompanyDto>> GetYoldançekCompaniesAsync(...)
{
    // 1. Önce kullanıcının ilçesindeki firmalar aranır
    if (districtId.HasValue)
    {
        companies = await baseQuery
            .Where(c => c.DistrictId == districtId.Value)
            .ToListAsync();
    }
    // 2. İlçede yoksa, aynı ildeki firmalar aranır
    if (!companies.Any() && provinceId.HasValue)
    {
        companies = await baseQuery
            .Where(c => c.ProvinceId == provinceId.Value)
            .ToListAsync();
    }
    // 3. İlde de yoksa, tüm aktif firmalar listelenir
    if (!companies.Any())
    {
        companies = await baseQuery.ToListAsync();
    }
    // 4. Mesafeye göre sıralama yapılır
```

```
    return companies.OrderBy(c => c.Distance).Take(limit).ToList();
}
```

### 5.2.2 JwtService - Token Yönetimi

```csharp
// Services/JwtService.cs - Satır 18-39
public string GenerateToken(Company company)
{
    var claims = new[]
    {
        new Claim("CompanyId", company.Id.ToString()),
        new Claim("Role", "Company"),
        new Claim(ClaimTypes.Email, company.Email),
        new Claim(ClaimTypes.Name, company.CompanyName)
    };

    var key = new SymmetricSecurityKey(
        Encoding.UTF8.GetBytes(_configuration["Jwt:Key"]!));

    var credentials = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

    var token = new JwtSecurityToken(
        issuer: _configuration["Jwt:Issuer"],
        audience: _configuration["Jwt:Audience"],
        claims: claims,
        expires: DateTime.UtcNow.AddDays(7),
        signingCredentials: credentials
    );
```

```
    return new JwtSecurityTokenHandler().WriteToken(token);
}
```

**Token Yapısı:**

- **Header:** Algoritma bilgisi (HS256)
- **Payload:** CompanyId, Role, Email, Name, Expiration
- **Signature:** HMAC SHA256 imzası

#### 5.2.3 ReviewService - Yorum Sistemi

```csharp
// Services/ReviewService.cs
public async Task<TowTruckReviewsResponseDto> GetTowTruckReviewsAsync(
    int towTruckId, int page, int pageSize)
{
    var towTruck = await _context.TowTrucks
        .FirstOrDefaultAsync(t => t.Id == towTruckId);

    // Onaylanmış ve görünür yorumları getir
    var query = _context.Reviews
        .Where(r => r.TowTruckId == towTruckId && r.IsVisible && r.IsApproved)
        .OrderByDescending(r => r.CreatedAt);

    var totalCount = await query.CountAsync();
    var reviews = await query
        .Skip((page - 1)  pageSize)
        .Take(pageSize)
        .ToListAsync();
```

```
    // Ortalama puan hesapla
    var averageRating = reviews.Any()
        ? Math.Round(reviews.Average(r => r.Rating), 1)
        : 0;

    return new TowTruckReviewsResponseDto
    {
        TowTruckId = towTruckId,
        DriverName = towTruck.DriverName,
        RatingSummary = new RatingSummaryDto
        {
            AverageRating = averageRating,
            ReviewCount = totalCount
        },
        Reviews = _mapper.Map<List<ReviewDto>>(reviews),
        TotalCount = totalCount,
        Page = page,
        PageSize = pageSize
    };
}
```

## 6. FRONTEND YAPISI

### 6.1 Proje Klasör Yapısı

yoldancek-main/

```
├── app/                    Next.js App Router
│   ├── page.tsx            Ana sayfa (3752 satır)
│   ├── layout.tsx          Root layout
│   ├── globals.css         Global stiller
│   ├── page.module.css     Ana sayfa stilleri
│   ├── loading.tsx         Loading component
│   ├── not-found.tsx       404 sayfası
│   ├── robots.ts           SEO - robots.txt
│   ├── sitemap.ts          SEO - sitemap.xml
│   ├── kvkk/               KVKK bilgilendirme sayfası
│   ├── gizlilik-politikasi/    Gizlilik politikası
│   ├── cerez-politikasi/       Çerez politikası
│   └── kullanim-kosullari/     Kullanım koşulları
├── lib/                    Yardımcı kütüphaneler
│   └── api.ts              API fonksiyonları (426 satır)
├── public/                 Statik dosyalar
├── package.json            Bağımlılıklar
├── next.config.ts          Next.js yapılandırma
├── tsconfig.json           TypeScript yapılandırma
└── Dockerfile              Docker konfigürasyonu
```

### 6.2 Ana Sayfa Bileşenleri (page.tsx)

Ana sayfa tek bir dosyada tüm işlevselliği barındırmaktadır:

| Bileşen/Özellik | Satır Aralığı | Açıklama |
|---|---|---|
| State Tanımlamaları | 40-180 | React useState hook'ları |
| useEffect Hook'ları | 200-400 | Sayfa yüklenme işlemleri |
| Konum Tespiti | 450-520 | Geolocation API kullanımı |
| Firma Arama | 600-700 | En yakın çekici arama |
| Kayıt/Giriş Modalları | 1000-1500 | Firma kayıt ve giriş formları |
| Çekici Yönetimi | 1600-2200 | CRUD işlemleri |
| Yorum Sistemi | 2300-2800 | Yorum görüntüleme/ekleme |
| Şikayet Sistemi | 2900-3200 | Şikayet formu |
| Harita Görünümü | 3300-3500 | Google Maps iframe |
| Footer | 3600-3752 | Alt bilgi ve linkler |

### 6.3 API Entegrasyon Katmanı (lib/api.ts)

```typescript
// lib/api.ts - Temel API fonksiyonu
async function api<T>(path: string, options: ApiOptions = {}) {
  const { token, headers, ...rest } = options;

  const res = await fetch(`${API_BASE}${path}`, {
    ...rest,
    headers: {
      "Content-Type": "application/json",
      ...(headers || {}),
      ...(token ? { Authorization: `Bearer ${token}` } : {}),
```

```
    },
    cache: "no-store",
  });

  const text = await res.text();
  const data = text ? (JSON.parse(text) as T) : (undefined as T);

  if (!res.ok) {
    throw new Error(data?.message || res.statusText);
  }

  return data;
}
```

**Önemli API Fonksiyonları:**

| Fonksiyon | HTTP Method | Endpoint | Açıklama |
|---|---|---|---|
| `getCities()` | GET | /api/Address/cities | İl listesi |
| `getDistricts(id)` | GET | /api/Address/districts/{id} | İlçe listesi |
| `registerCompany()` | POST | /api/Auth/register | Firma kayıt |
| `loginCompany()` | POST | /api/Auth/login | Firma giriş |
| `findYoldançekTowTrucks()` | GET | /api/location/Yoldançek | En yakın çekiciler |
| `addTowTruck()` | POST | /api/TowTrucks | Çekici ekleme |
| `updateTowTruck()` | PUT | /api/TowTrucks/{id} | Çekici güncelleme |
| `deleteTowTruck()` | DELETE | /api/TowTrucks/{id} | Çekici silme |
| `getTowTruckReviews()` | GET | /api/reviews/towtruck/{id} | Yorumları getir |
| `createReview()` | POST | /api/reviews | Yorum ekle |
| `createAbuseReport()` | POST | /api/abusereports | Şikayet bildir |

### 6.4 Önemli React Hook'ları

```typescript
// Konum tespiti
useEffect(() => {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(
      (position) => {
        setGeoLocation({
          lat: position.coords.latitude,
          lng: position.coords.longitude,
        });
      },
      (error) => {
        console.error("Konum alınamadı:", error);
      }
    );
  }
}, []);

// En yakın çekicileri getir
const fetchYoldançekTowTrucks = useCallback(async () => {
  setCompaniesLoading(true);
  try {
    const params: any = { limit: 50 };

    if (geoLocation) {
      params.latitude = geoLocation.lat;
      params.longitude = geoLocation.lng;
```

```
    }

    if (provinceId) params.provinceId = Number(provinceId);
    if (districtId) params.districtId = Number(districtId);

    const data = await findYoldançekTowTrucks(params);
    setCompanies(data);
  } catch (err) {
    setToast("Çekiciler yüklenirken hata oluştu");
  } finally {
    setCompaniesLoading(false);
  }
}, [geoLocation, provinceId, districtId]);
```

## 7. API ENDPOINT'LERİ

### 7.1 Public Endpoint'ler (Token Gerektirmez)

| Method | Endpoint | Açıklama |
|---|---|---|
| POST | /api/Auth/register | Firma kayıt |
| POST | /api/Auth/login | Firma giriş |
| POST | /api/Admin/login | Admin giriş |
| GET | /api/Address/cities | İl listesi |
| GET | /api/Address/districts/{provinceId} | İlçe listesi |
| GET | /api/location | Tüm aktif çekiciler |
| GET | /api/location/Yoldançek | En yakın çekiciler |
| GET | /api/Companies | Tüm firmalar |
| GET | /api/Companies/Yoldançek | En yakın firmalar |

| POST | /api/Tickets | İletişim talebi |

| GET | /api/reviews/towtruck/{id} | Çekici yorumları |

| POST | /api/reviews | Yorum ekle |

| POST | /api/abusereports | Şikayet bildir |

### 7.2 Protected Endpoint'ler (Token Gerekir)

| Method | Endpoint | Rol | Açıklama |

|--------|----------|-----|----------|

| GET | /api/Profile | Company/Admin | Profil bilgisi |

| PUT | /api/Companies/me | Company | Profil güncelle |

| POST | /api/TowTrucks | Company | Çekici ekle |

| GET | /api/TowTrucks/my | Company | Çekicilerimi listele |

| PUT | /api/TowTrucks/{id} | Company | Çekici güncelle |

| DELETE | /api/TowTrucks/{id} | Company | Çekici sil |

| PUT | /api/TowTrucks/{id}/deactivate | Company | Çekici pasif yap |

| GET | /api/Tickets | Company/Admin | Ticket listesi |

| PUT | /api/Tickets/{id}/status | Company/Admin | Ticket durumu güncelle |

| PUT | /api/Admin/address | Admin | Adres verilerini güncelle |

| GET | /api/Admin/tickets | Admin | Tüm ticket'lar |

| GET | /api/Admin/companies | Admin | Tüm firmalar |

| PUT | /api/Admin/companies/{id} | Admin | Firma güncelle |

| DELETE | /api/Admin/companies/{id} | Admin | Firma sil |

| GET | /api/Admin/towtrucks | Admin | Tüm çekiciler |

| PUT | /api/Admin/towtrucks/{id} | Admin | Çekici güncelle |

| DELETE | /api/Admin/towtrucks/{id} | Admin | Çekici sil |

| GET | /api/abusereports | Admin | Şikayet listesi |

| PUT | /api/abusereports/{id} | Admin | Şikayet güncelle |

### 7.3 Örnek API İstekleri

**Firma Kayıt:**

```http
POST /api/Auth/register
Content-Type: application/json

{
  "firstName": "Ahmet",
  "lastName": "Yılmaz",
  "companyName": "Yılmaz Oto Kurtarma",
  "phoneNumber": "05551234567",
  "provinceId": 34,
  "districtId": 1,
  "fullAddress": "Kadıköy Mah. No:15",
  "latitude": 41.0082,
  "longitude": 29.0094,
  "serviceCity": "İstanbul",
  "email": "ahmet@yilmaz.com",
  "password": "Sifre123!",
  "kvkkConsent": true,
  "kvkkConsentVersion": "1.0"
}
```

**En Yakın Çekiciler:**

```http
GET /api/location/Yoldançek?latitude=41.0082&longitude=29.0094&limit=10&provinceId=34
```

**Yorum Ekleme:**

```http
POST /api/reviews
Content-Type: application/json

{
  "towTruckId": 15,
  "reviewerName": "Mehmet K.",
  "reviewerPhone": "05559876543",
  "rating": 5,
  "comment": "Çok hızlı ve profesyonel hizmet aldım."
}
```

## 8. GÜVENLİK ÖNLEMLERİ

### 8.1 Kimlik Doğrulama (Authentication)

- **JWT Token:** 7 günlük geçerlilik süresi
- **HMAC SHA256:** Token imzalama algoritması
- **Token Yenileme:** Otomatik yenileme mekanizması

### 8.2 Şifre Güvenliği

```csharp
// SHA256 ile şifre hashleme
private string HashPassword(string password)
{
    using var sha256 = SHA256.Create();
```

```
    var hashedBytes = sha256.ComputeHash(Encoding.UTF8.GetBytes(password));
    return Convert.ToBase64String(hashedBytes);
}
```

### 8.3 Yetkilendirme (Authorization)

- **Role-Based Access Control:** Admin ve Company rolleri
- **Claim-Based Authorization:** Token içindeki bilgilere göre erişim kontrolü
- **[Authorize] Attribute:** Endpoint bazlı koruma

### 8.4 CORS Politikası

```csharp
builder.Services.AddCors(options =>
{
    options.AddPolicy("AllowFrontend", policy =>
    {
        policy.WithOrigins(
            "https://yoldancek.com",
            "http://localhost:3000"
        )
        .AllowAnyMethod()
        .AllowAnyHeader();
    });
});
```

### 8.5 KVKK Uyumu

- Firma kaydında KVKK onayı zorunlu
- Onay tarihi ve versiyonu kaydedilir
- IP adresi loglanır
- KVKK, Gizlilik Politikası, Çerez Politikası sayfaları mevcut

## 9. DEPLOYMENT

### 9.1 Docker Yapılandırması

**Backend Dockerfile:**

```dockerfile
# Build stage
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build
WORKDIR /src
COPY *.csproj ./
RUN dotnet restore
COPY . ./
RUN dotnet build -c Release -o /app/build

# Publish stage
FROM build AS publish
RUN dotnet publish -c Release -o /app/publish

# Runtime stage
FROM mcr.microsoft.com/dotnet/aspnet:8.0 AS runtime
WORKDIR /app
```

```
RUN mkdir -p /app/wwwroot/uploads/drivers
COPY --from=publish /app/publish .
EXPOSE 80
ENTRYPOINT ["dotnet", "Yoldançek.dll"]
```

**Docker Compose:**

```yaml
version: '3.8'
services:
  postgres:
    image: postgres:15-alpine
    environment:
      POSTGRES_USER: Yoldançek
      POSTGRES_PASSWORD: ${DB_PASSWORD}
      POSTGRES_DB: Yoldançekdb
    volumes:
      - postgres_data:/var/lib/postgresql/data

  Yoldançek-api:
    build: .
    ports:
      - "5000:80"
    depends_on:
      - postgres
    environment:
      - ConnectionStrings__DefaultConnection=Host=postgres;Database=Yoldançekdb;Username=Yoldançek;Password=${DB_PASSWORD}
      - Jwt__Key=${JWT_KEY}
```

```
volumes:
  postgres_data:
```

### 9.2 Deployment Akışı

1. Geliştirici kod değişikliği yapar
   ↓
2. Git commit & push (main branch)
   ↓
3. Production branch'e merge
   ↓
4. Sunucuda deploy script çalışır:
   - git pull
   - docker compose build
   - docker compose up -d
   ↓
5. Migration'lar otomatik çalışır
   ↓
6. Uygulama yayında!

### 9.3 Canlı URL'ler

- **Frontend:** https://yoldancek.com
- **Backend API:** https://api.yoldancek.com
- **Swagger Docs:** https://api.yoldancek.com/swagger

## 10. YAPAY ZEKA KULLANIMI

## 10.1 Yapay Zeka Kullanılan Bölümler

Bu projede aşağıdaki bölümlerde yapay zeka (GitHub Copilot) desteği alınmıştır:

| Bölüm | Dosya | Açıklama |
|---|---|---|
| Haversine Formülü | LocationService.cs | Mesafe hesaplama algoritması |
| JWT Token Üretimi | JwtService.cs | Token yapısı ve claim'ler |
| AutoMapper Profilleri | Mappings/.cs | Entity-DTO dönüşümleri |
| Docker Yapılandırması | Dockerfile, docker-compose.yml | Multi-stage build |
| KVKK Entegrasyonu | Company.cs, CompanyRegistrationDto.cs | Onay alanları |
| Review Sistemi | ReviewService.cs, ReviewsController.cs | Yorum CRUD işlemleri |
| Abuse Report Sistemi | AbuseReportService.cs | Şikayet mekanizması |

## 10.2 Öğrenilen Konular

**1. Haversine Formülü:**

- Küresel koordinatlar arasında mesafe hesaplama
- Trigonometrik fonksiyonların kullanımı
- Dünya yarıçapı sabiti

**2. JWT Authentication:**

- Token yapısı (Header, Payload, Signature)
- Claim-based identity
- Token geçerlilik süresi yönetimi

**3. Entity Framework Core:**

- Code-First yaklaşımı
- Migration yönetimi
- İlişki tanımlamaları (1:N, M:N)

**4. Next.js App Router:**

- Server/Client components
- Route handlers
- SEO optimizasyonu

## 11. EKRAN GÖRÜNTÜLERİ

### 11.1 Ana Sayfa

[Ana sayfa ekran görüntüsü]

- Konum bazlı çekici arama
- İl/ilçe filtreleme
- Harita görünümü

### 11.2 Firma Paneli

[Firma paneli ekran görüntüsü]

- Çekici ekleme/düzenleme
- Şoför fotoğrafı yükleme
- Çalışma bölgesi seçimi

### 11.3 Yorum Sistemi

[Yorum sistemi ekran görüntüsü]

- Puan verme (1-5 yıldız)
- Yorum yazma
- Ortalama puan gösterimi

### 11.4 Şikayet Formu

[Şikayet formu ekran görüntüsü]

- Şikayet türü seçimi
- Açıklama girişi
- İletişim bilgileri

### 11.5 Swagger API Dokümantasyonu

[Swagger ekran görüntüsü]

- Endpoint listesi
- Request/Response örnekleri
- Authorization desteği

## 12. SONUÇ VE DEĞERLENDİRME

### 12.1 Tamamlanan Özellikler

✅ JWT tabanlı kimlik doğrulama sistemi

✅ Kullanıcı kayıt/giriş ve token yenileme

✅ [Authorize] attribute ile endpoint güvenliği

✅ Firma CRUD işlemleri

✅ Çekici CRUD işlemleri

✅ Konum bazlı arama (Haversine formülü)

✅ İl/ilçe bazlı filtreleme

✅ Yorum ve puan sistemi

✅ Kötüye kullanım raporlama

✅ Admin paneli (firma/çekici yönetimi)

✅ KVKK uyumu

✅ Docker ile deployment

✅ Swagger API dokümantasyonu

✅ React + Next.js frontend

## 12.2 Gelecek Geliştirmeler

- Push bildirim sistemi
- Gerçek zamanlı konum takibi
- Mobil uygulama (React Native)
- Ödeme entegrasyonu
- SMS doğrulama

## 12.3 Karşılaşılan Zorluklar ve Çözümleri

| Zorluk | Çözüm |
|---|---|
| CORS hataları | Frontend ve backend URL'leri için policy tanımlandı |
| Migration çakışmaları | Otomatik migration ile çözüldü |
| Docker volume sorunları | Named volume ve path mapping düzenlendi |
| Konum izni reddi | Fallback olarak il/ilçe seçimi eklendi |

## 12.4 Proje İstatistikleri

| Metrik | Değer |
|---|---|
| Toplam Kod Satırı (Backend) | ~5000 |
| Toplam Kod Satırı (Frontend) | ~4500 |
| API Endpoint Sayısı | 25+ |
| Veritabanı Tablosu | 10 |
| Migration Sayısı | 6 |
| Docker Container | 2 |

## KAYNAKLAR


1. Microsoft .NET Documentation - https://docs.microsoft.com/dotnet
2. Entity Framework Core - https://docs.microsoft.com/ef/core
3. Next.js Documentation - https://nextjs.org/docs
4. PostgreSQL Documentation - https://www.postgresql.org/docs
5. JWT.io - https://jwt.io
6. Docker Documentation - https://docs.docker.com


**Proje GitHub:** https://github.com/meminkecik/Blm4531

**Demo Video:**
**https://drive.google.com/file/d/1NrLwVli5ZON4AOaheYPQK0_gynAzNC05/view?usp=sharing**

**Canlı Site:** https://yoldancek.com

**API Swagger:** https://api.yoldancek.com/swagger