SUNUCU TABANLI PROGRAMLAMA PROJE GÖREVİ

Araç Filo Yönetim ve Karar Destek Sistemi

Proje Raporu

Proje Adı: Araç Filo Yönetim ve Karar Destek Sistemi
Öğrenci Adı Soyadı: Ahmet EFE
Öğrenci No: 2022469142
Tarih: Ocak 2026

İÇİNDEKİLER

---

1. PROJENİN AMACI

Bu proje, sunucu taraflı yazılım geliştirme becerilerinin kazanılması amacıyla geliştirilmiştir. Projenin temel hedefleri:

- **Sunucu taraflı yazılım geliştirme:** Node.js ve Express.js kullanarak sunucu uygulaması geliştirme
- **MVC mimarisini doğru ve tutarlı biçimde uygulama:** Model-View-Controller katmanlı mimari
- **REST prensiplerine uygun API tasarlama:** HTTP metotları ve kaynak tabanlı URL yapısı
- **Veri modeli, iş mantığı ve uç noktalarını ayrıştırma:** Sorumlulukların net ayrımı
- **Yazılım projelerinde okunabilirlik, sürdürülebilirlik ve ölçeklenebilirlik sağlama**

Uygulama **katı biçimde MVC mimarisine** uygun tasarlanmıştır.

---

2. SENARYO TANIMI

2.1 İş Problemi

Türkiye'de faaliyet gösteren bir tur şirketi, 4 farklı güzergahta düzenli turlar düzenlemektedir. Şirket, aşağıdaki sorunlarla karşı karşıyadır:

- Hangi dönemlerde araç kapasitesi yetersiz kalıyor?
- Hangi dönemlerde araçlar boşta kalıyor?
- Dış araç kiralama maliyeti mi yoksa turdan vazgeçme kaybı mı daha yüksek?

2.2 Güzergahlar

#	Güzergah	Süre
1	Muğla	2 gün
2	İzmir – Efes Antik Kenti	2 gün
3	Kapadokya	3 gün
4	İstanbul	2 gün

2.3 Sistem Özellikleri

Sistem şu analizleri üretmektedir:

- Tur yoğunluğu analizi (günlük, haftalık, aylık, yıllık)
- Araç yeterlilik/yetersizlik durumu
- Filo eşzamanlılık hesaplaması (peak concurrent)
- Maliyet senaryosu karşılaştırması

---

3. MVC MİMARİSİ UYUMU

3.1 MVC Nedir?

MVC (Model-View-Controller), yazılım geliştirmede kullanılan bir mimari desendir. Bu desen, uygulamayı üç ana bileşene ayırarak kodun organize edilmesini sağlar.

3.2 Projede MVC Katmanları

Katman	Klasör	Sorumluluk
Model	`server/models/`	Veritabanı entity sınıfları ve iş mantığı
View	`server/views/`	EJS şablon dosyaları (kullanıcı arayüzü)
Controller	`server/controllers/`	İstek işleme ve yanıt oluşturma
Routes	`server/routes/`	URL eşleştirme ve yönlendirme

3.3 Model Katmanı

Model katmanı, veritabanı erişimini ve iş mantığını içerir. Her entity için ayrı bir model sınıfı oluşturulmuştur.

Model Dosyaları:

Dosya	Tablo	Açıklama
`Arac.model.js`	araclar	Araç CRUD işlemleri, müsaitlik kontrolü
`Tur.model.js`	turlar	Tur CRUD işlemleri, iş kuralları
`Guzergah.model.js`	guzergahlar	Güzergah bilgileri
`AdminKullanici.model.js`	admin_kullanicilar	Kimlik doğrulama

Model katmanını kullanma nedenleri:

- Veritabanı erişimini merkezi bir noktada toplamak
- İş kurallarını veri katmanında uygulamak
- Controller'ları veritabanı detaylarından soyutlamak
- Kod tekrarını önlemek

3.4 View Katmanı

View katmanı, kullanıcıya gösterilen arayüzleri içerir. **EJS (Embedded JavaScript)** şablon motoru kullanılmıştır.

View Dosyaları:

Dosya	Açıklama
`login.ejs`	Kullanıcı giriş formu
`dashboard.ejs`	Ana gösterge paneli (grafikler, tablolar)
`error.ejs`	Hata sayfası

EJS tercih nedenleri:

- JavaScript syntax'ına yakın
- Express.js ile mükemmel entegrasyon
- Sunucu tarafında render yapabilme

3.5 Controller Katmanı

Controller'lar HTTP isteklerini işler ve uygun yanıtı oluşturur.

Controller Dosyaları:

Dosya	Sorumluluk
`auth.controller.js`	Giriş/çıkış, JWT token yönetimi
`analytics.controller.js`	Dashboard verileri, raporlama
`admin.controller.js`	Veritabanı yönetimi, seed işlemleri
`health.controller.js`	Sistem sağlık kontrolü

3.6 Routes Katmanı

Routes, URL'leri controller fonksiyonlarına eşleştirir.

```
// Örnek route tanımları
router.get('/summary', analyticsController.getDashboardSummary);
router.post('/login', authController.login);
router.post('/seed', adminController.seedDatabase);
```

---

4. CRUD İŞLEMLERİ

Projede CRUD (Create, Read, Update, Delete) işlemleri **Stored Procedures** aracılığıyla gerçekleştirilmektedir.

4.1 CREATE (Oluşturma)

```
-- Tur oluşturma (iş kuralları dahil)
CALL sp_tur_olustur(guzergah_id, arac_id, baslangic, bitis, yolcu_sayisi, fiyat);

-- Temel veri yükleme
CALL sp_temel_veri_yukle();
```

4.2 READ (Okuma)

```
-- Tur yoğunluğu raporu
CALL sp_rapor_tur_yogunluk(baslangic_tarih, bitis_tarih, gruplama);

-- Güzergah bazlı hacim
CALL sp_rapor_guzergah_hacim(baslangic_tarih, bitis_tarih, gruplama);

-- Filo eşzamanlılık
CALL sp_rapor_filo_eszamanlilik(baslangic_tarih, bitis_tarih);
```

4.3 UPDATE (Güncelleme)

```
// Admin şifre güncelleme
await query('UPDATE admin_kullanicilar SET sifre_hash = ? WHERE id = ?', [hash, id]);

// Son giriş tarihi güncelleme
await query('UPDATE admin_kullanicilar SET son_giris_tarihi = NOW() WHERE id = ?', [id]);
```

4.4 DELETE (Silme)

```
-- Seed öncesi veri temizleme
TRUNCATE TABLE turlar;
TRUNCATE TABLE araclar;
```

---

5. İŞ KURALLARI (ÖZEL SENARYOLAR)

Projede **2 özel iş kuralı** uygulanmıştır:

5.1 İş Kuralı 1: Araç Tarih Çakışma Kontrolü

Kural: Aynı araç, aynı tarih aralığında birden fazla tura atanamaz.

Uygulama Yeri: `sp_tur_olustur` stored procedure

```
-- Çakışma kontrolü
SELECT COUNT(*) INTO v_cakisma_sayisi
FROM turlar
WHERE arac_id = p_arac_id
```

```
  AND (p_baslangic < bitis_tarihi AND p_bitis > baslangic_tarihi);

IF v_cakisma_sayisi > 0 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Bu araç seçilen tarih aralığında başka bir turda kullanılmaktadır.';
END IF;
```

Neden Gerekli?

- Bir araç fiziksel olarak aynı anda iki farklı yerde olamaz
- Müşterilere güvenilir hizmet sunulması sağlanır
- Operasyonel karışıklıklar önlenir

Test Senaryosu:

1. Araç #1 için 15-17 Ocak tarihlerinde tur oluştur → Başarılı
2. Araç #1 için 16-18 Ocak tarihlerinde tur oluştur → **HATA: Çakışma tespit edildi**

5.2 İş Kuralı 2: Yolcu Sayısı Validasyonu

Kural: Her turdaki yolcu sayısı 1 ile 50 arasında olmalıdır.

Uygulama Yeri: Hem stored procedure hem de veritabanı constraint

```
-- Stored procedure kontrolü
IF p_yolcu_sayisi < 1 OR p_yolcu_sayisi > 50 THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Yolcu sayısı 1-50 arasında olmalıdır.';
END IF;

-- Veritabanı constraint
CONSTRAINT chk_yolcu_araligi
    CHECK (yolcu_sayisi BETWEEN 1 AND 50)
```

Neden Gerekli?

- Araç kapasitesi sınırlarını korur
- Mantıksız veri girişini önler (0 veya negatif yolcu)
- Güvenlik ve konfor standartlarını sağlar

Test Senaryosu:

1. Yolcu sayısı: 10 → Başarılı
2. Yolcu sayısı: 0 → **HATA: Yolcu sayısı 1-50 arasında olmalıdır**
3. Yolcu sayısı: 100 → **HATA: Yolcu sayısı 1-50 arasında olmalıdır**

---

6. API TASARIMI VE REST UYUMU

6.1 REST Prensipleri

Projede REST (Representational State Transfer) prensipleri uygulanmıştır:

Prensip	Uygulama

Kaynak tabanlı URL'ler	`/api/analytics/tour-volume`
HTTP metotlarının doğru kullanımı	GET okuma, POST yazma için
Stateless iletişim	Her istek bağımsız, JWT ile kimlik doğrulama
JSON formatında yanıtlar	Tüm API yanıtları JSON

6.2 API Endpoint Listesi

Authentication Endpoints

Metot	Endpoint	Açıklama
POST	`/auth/login`	Kullanıcı girişi
POST	`/auth/logout`	Oturum sonlandırma
GET	`/auth/me`	Mevcut kullanıcı bilgisi

Analytics Endpoints (JWT Korumalı)

Metot	Endpoint	Açıklama
GET	`/api/analytics/summary`	Dashboard özet verileri
GET	`/api/analytics/tour-volume?from=&to=&group=`	Tur yoğunluğu raporu
GET	`/api/analytics/route-volume?from=&to=&group=`	Güzergah bazlı hacim
GET	`/api/analytics/fleet-concurrency?from=&to=`	Filo eşzamanlılık
GET	`/api/analytics/monthly-fleet-balance?year=`	Aylık filo dengesi
GET	`/api/analytics/recommendations?year=`	Yönetici önerileri
GET	`/api/analytics/daily-tours?from=&to=`	Günlük tur verileri

Admin Endpoints

Metot	Endpoint	Açıklama
POST	`/api/admin/seed`	Veritabanı seed
GET	`/api/admin/db-status`	Veritabanı durumu
POST	`/api/admin/test-overlap`	Çakışma testi
GET	`/api/admin/tour-distribution`	Tur dağılımı

6.3 Örnek API İsteği ve Yanıtı

İstek:

```
GET /api/analytics/summary?from=2024-01-01&to=2024-12-31
Authorization: Bearer <JWT_TOKEN>
```

Yanıt:


```
{
  "success": true,
  "data": {
    "summary": {
      "toplamTur": 2100,
      "toplamGelir": 126000000,
      "ortalamaYolcu": 7
    },
    "routeStats": [
      { "guzergah_adi": "Kapadokya", "tur_sayisi": 580 },
      { "guzergah_adi": "İstanbul", "tur_sayisi": 540 }
    ]
  },
  "meta": {
    "from": "2024-01-01",
    "to": "2024-12-31"
  }
}
```

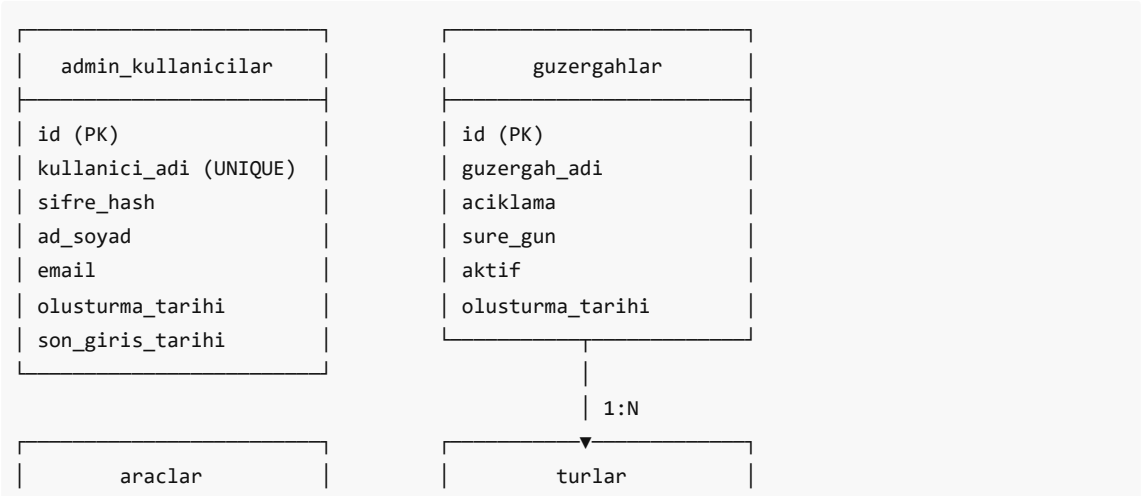

---

7. VERİTABANI TASARIMI VE ER DİYAGRAMI

7.1 Veritabanı Tabloları

Tablo	Açıklama	Kayıt Sayısı
admin_kullanicilar	Sistem yöneticisi	1
guzergahlar	Tur güzergahları	4
araclar	Araç filosu	70
turlar	Tur kayıtları	binlerce

7.2 ER Diyagramı



```
┌──────────────────────────┐          ┌──────────────────────────┐
│ id (PK)                  │◄─────────┤ id (PK)                  │
│ plaka (UNIQUE)           │   N:1    │ guzergah_id (FK)         │
│ kapasite                 │          │ arac_id (FK)             │
│ marka                    │          │ baslangic_tarihi         │
│ model                    │          │ bitis_tarihi             │
│ satin_alma_yili          │          │ yolcu_sayisi             │
│ durum                    │          │ fiyat_tl                 │
│ olusturma_tarihi         │          │ notlar                   │
└──────────────────────────┘          │ olusturma_tarihi         │
                                      └──────────────────────────┘
```

7.3 Foreign Key İlişkileri

```sql
-- Tur -> Güzergah ilişkisi
CONSTRAINT fk_turlar_guzergah
    FOREIGN KEY (guzergah_id) REFERENCES guzergahlar(id)
    ON DELETE RESTRICT ON UPDATE CASCADE

-- Tur -> Araç ilişkisi
CONSTRAINT fk_turlar_arac
    FOREIGN KEY (arac_id) REFERENCES araclar(id)
    ON DELETE SET NULL ON UPDATE CASCADE
```

7.4 Veritabanı Kısıtlamaları

```sql
-- Tarih sırası kontrolü
CONSTRAINT chk_tarih_sirasi
    CHECK (bitis_tarihi > baslangic_tarihi)

-- Yolcu sayısı kontrolü
CONSTRAINT chk_yolcu_araligi
    CHECK (yolcu_sayisi BETWEEN 1 AND 50)
```

---

8. PROJE KLASÖR YAPISI

Proje klasör yapısı derste anlatılan standart MVC yapısına uygundur:

```
arac-yonetim-sistemi/
├── server/
│   ├── app.js                    # Ana Express uygulaması
│   │
│   ├── config/                   # Konfigürasyon
│   │   ├── db.js                 # Veritabanı bağlantısı
│   │   └── env.example           # Örnek environment dosyası
│   │
│   ├── models/                   # MODEL KATMANI
│   │   ├── index.js              # Model export
│   │   ├── Arac.model.js
```

```
│   │   ├── Tur.model.js
│   │   ├── Guzergah.model.js
│   │   └── AdminKullanici.model.js
│   │
│   ├── controllers/                # CONTROLLER KATMANI
│   │   ├── auth.controller.js
│   │   ├── analytics.controller.js
│   │   ├── admin.controller.js
│   │   └── health.controller.js
│   │
│   ├── routes/                     # ROUTE TANIMLARI
│   │   ├── auth.routes.js
│   │   ├── analytics.routes.js
│   │   ├── admin.routes.js
│   │   └── health.routes.js
│   │
│   ├── middlewares/                # ARA KATMAN
│   │   └── auth.middleware.js      # JWT doğrulama
│   │
│   ├── views/                      # VIEW KATMANI
│   │   ├── login.ejs
│   │   ├── dashboard.ejs
│   │   └── error.ejs
│   │
│   ├── public/                     # STATİK DOSYALAR
│   │   ├── css/style.css
│   │   └── js/dashboard.js
│   │
│   └── sql/                        # VERİTABANI SCRIPTLERİ
│       ├── 01_schema.sql
│       ├── 02_procedures.sql
│       └── 03_seed.sql
│
├── .env                            # Environment değişkenleri
├── .env.example                    # Örnek environment
├── .gitignore
├── package.json
├── README.md
└── PROJE_RAPORU.md
```

---

9. ENVIRONMENT KONFİGÜRASYONU (.env)

9.1 .env.example Dosyası

```
# Server Konfigürasyonu
PORT=3000
NODE_ENV=development

# MySQL Veritabanı (XAMPP)
DB_HOST=localhost
```

```
DB_PORT=3306
DB_USER=root
DB_PASSWORD=
DB_NAME=kds_arac_yonetim

# JWT Ayarları
JWT_SECRET=kds-arac-yonetim-gizli-anahtar-2024
JWT_EXPIRES_IN=24h

# Admin Seed Token (Güvenlik için değiştirin)
ADMIN_SEED_TOKEN=kds-seed-token-2024

# Filo Kapasitesi
FLEET_CAPACITY=70
TOUR_PRICE_TL=5000
```

9.2 Environment Değişkenlerinin Kullanımı

```
// Veritabanı bağlantısında
const pool = mysql.createPool({
    host: process.env.DB_HOST,
    user: process.env.DB_USER,
    password: process.env.DB_PASSWORD,
    database: process.env.DB_NAME
});

// JWT token oluşturmada
const token = jwt.sign(payload, process.env.JWT_SECRET, {
    expiresIn: process.env.JWT_EXPIRES_IN
});
```

---

10. KURULUM ADIMLARI

10.1 Gereksinimler

- Node.js (v16 veya üstü)
- XAMPP veya MySQL 8.0
- Modern web tarayıcı

10.2 Adım Adım Kurulum

1. Projeyi klonlayın:

```
git clone <repo-url>
cd arac-yonetim-sistemi
```

2. Bağımlılıkları yükleyin:

```
npm install
```

3. MySQL veritabanını oluşturun:

```
CREATE DATABASE kds_arac_yonetim
    CHARACTER SET utf8mb4
    COLLATE utf8mb4_turkish_ci;
```

4. SQL dosyalarını sırasıyla çalıştırın:

```
mysql -u root -p kds_arac_yonetim < server/sql/01_schema.sql
mysql -u root -p kds_arac_yonetim < server/sql/02_procedures.sql
mysql -u root -p kds_arac_yonetim < server/sql/03_seed.sql
```

5. Environment dosyasını yapılandırın:

```
cp .env.example .env
```

6. Sunucuyu başlatın:

```
npm run dev
```

7. Tarayıcıda açın:

```
http://localhost:3000
```

10.3 Varsayılan Giriş Bilgileri

Alan	Değer
Kullanıcı Adı	admin
Şifre	Admin123!

---

11. KOD KALİTESİ VE YAPI

11.1 Kullanılan Teknolojiler

Teknoloji	Sürüm	Amaç
Node.js	v16+	Runtime environment
Express.js	4.18.2	Web framework
MySQL	8.0	Veritabanı
EJS	3.1.9	Template engine
JWT	9.0.2	Kimlik doğrulama
Bcrypt	2.4.3	Şifre hashleme

11.2 Güvenlik Önlemleri

Önlem	Açıklama
JWT Authentication	Tüm korumalı endpoint'ler token gerektirir
Bcrypt Hashing	Şifreler düz metin olarak saklanmaz
Environment Variables	Hassas bilgiler .env dosyasında
Prepared Statements	SQL injection koruması
CORS Configuration	Cross-origin istekler kontrollü

11.3 Stored Procedures

Prosedür	Açıklama
sp_tur_olustur	Çakışma kontrolü ile tur ekleme
sp_temel_veri_yukle	Temel veri seed
sp_turlari_olustur	Toplu tur üretimi
sp_rapor_tur_yogunluk	Tur hacmi raporu
sp_rapor_guzergah_hacim	Güzergah bazlı analiz
sp_rapor_filo_eszamanlilik	Peak concurrent hesaplama
sp_rapor_aylik_filo_denge	Aylık shortage/surplus
sp_yonetici_oneri	Yönetici öneri paneli

---

12. SONUÇ

Bu proje kapsamında aşağıdaki gereksinimler başarıyla karşılanmıştır:

Gereksinim	Durum	Açıklama
MVC Mimarisi	✅	Model-View-Controller-Routes ayrımı
CRUD İşlemleri	✅	Stored procedures ile
İş Kuralı 1	✅	Araç tarih çakışma kontrolü
İş Kuralı 2	✅	Yolcu sayısı validasyonu (1-50)
REST API	✅	HTTP metotları, JSON yanıtlar
.env.example	✅	Kök dizinde mevcut
README.md	✅	Kurulum ve API bilgileri
ER Diyagramı	✅	Bu raporda mevcut

Proje, sunucu taraflı yazılım geliştirme, MVC mimarisi ve REST API tasarımı konularında gerçekçi bir iş problemi üzerinden pratik deneyim sağlamıştır.

Hazırlayan: Ahmet
Tarih: Ocak 2026