T.C. HARRAN ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

JAVA İLE PROGRAMLAMA DERSİ YAZILIM PROJE RAPORU

PROJE ADI: OTOBÜS OTOMASYONU (ÇAĞDAŞ GÜVEN TURİZM)

Hazırlayan: Mahmut Esat Ali ÖZKAN [230504041]

Teslim Tarihi: [29.12.2025]

Programcının Adı-Soyadı: Mahmut Esat Ali ÖZKAN

Geliştirilme Ortamları: JetBrains IntelliJ IDEA, MySQL Workbench

Programlama Dili: Java (JDK 18)

Kullanılan Kütüphaneler: Java Swing, AWT, FlatLaf (UI Teması)

Programın Adı: Otobüs Otomasyonu (Çağdaş Güven Turizm)

Bilgisayar Sistemi: Intel Core i5/i7 İşlemci, 16GB RAM, SSD Depolama

İşletim Sistemi: Windows 10/11

Gereksinimler: Java Runtime Environment (JRE 18+), MySQL Server

Kaynak Kod Satır Sayısı: 4500+ Satır

1- Problemin Tanımlanması ve Programın Amacı

Günümüzde şehirlerarası yolcu taşımacılığı yapan firmaların en büyük operasyonel sorunu; bilet satışlarının, sefer planlamalarının ve personel (kaptan) yönetiminin manuel defterler veya entegre olmayan eski sistemlerle yürütülmesidir. Bu durum veri kayıplarına, aynı koltuğun birden fazla kişiye satılmasına (çakışma), raporlama eksikliklerine ve müşteri memnuniyetsizliğine yol açmaktadır.

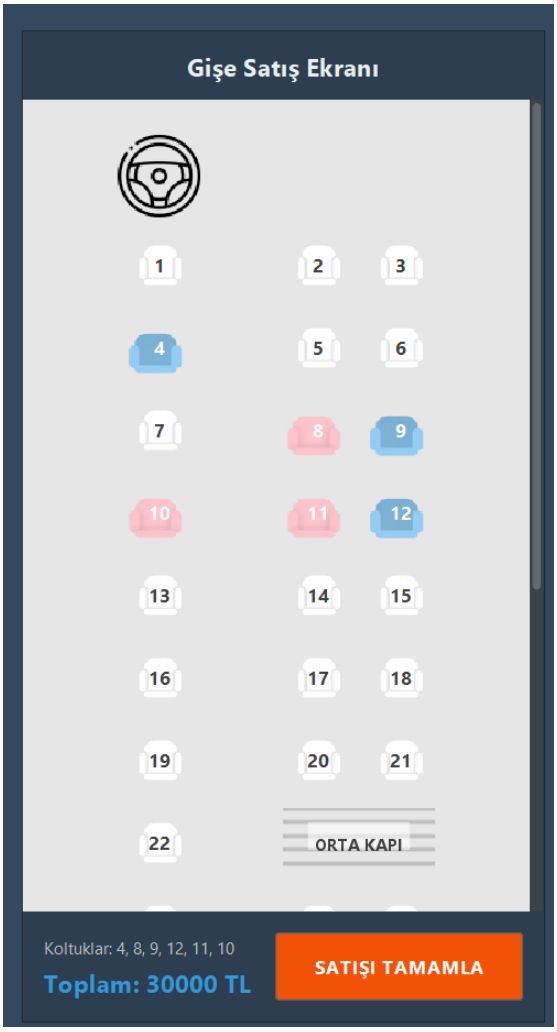
Bu projenin temel amacı; "Çağdaş Güven Turizm" firması örneği üzerinden, bir otobüs firmasının tüm operasyonel süreçlerini **dijitalleştirmek**, verileri **ilişkisel bir veritabanı (RDBMS)** üzerinde güvenle saklamak ve hem yönetici (Admin) hem de son kullanıcı (Müşteri) için **kullanıcı dostu (User-Friendly)** bir masaüstü yazılımı geliştirmektir. Proje, manuel hataları sıfıra indirmeyi ve yönetimsel kararlar için anlık veri akışı sağlamayı hedeflemektedir.

2- Problemin Çözümü ve Çözüm Tasarımı

2.1- Son Kullanıcıya (Arabirime) Yönelik Tasarım

Yazılımın arayüzü tasarlanırken modern tasarım prensipleri (Flat Design) benimsenmiş ve Java Swing'in standart görünümü yerine **FlatLaf** kütüphanesi entegre edilerek profesyonel bir görünüm elde edilmiştir.

- **Müşteri Paneli:** Kullanıcıdan kalkış-varış noktası ve tarih bilgisi istenerek filtreleme yapılır. Bilet alım ekranında, otobüsün iç yapısını birebir yansıtan **Görsel Koltuk Seçim Modülü** tasarlanmıştır.
  Bu modülde;
  - Boş koltuklar beyaz,
  - Dolu koltuklar (Cinsiyete göre) pembe veya mavidir.



- **Admin Paneli:** Yönetici girişleri için veri yoğunluklu formlar yerine, grafiksel öğelerle zenginleştirilmiş bir **Dashboard** tasarlanmıştır. Sol menüde kullanıcının hangi sayfada olduğunu belirten "Active State" (Aktif Durum) renklendirmesi yapılmıştır.

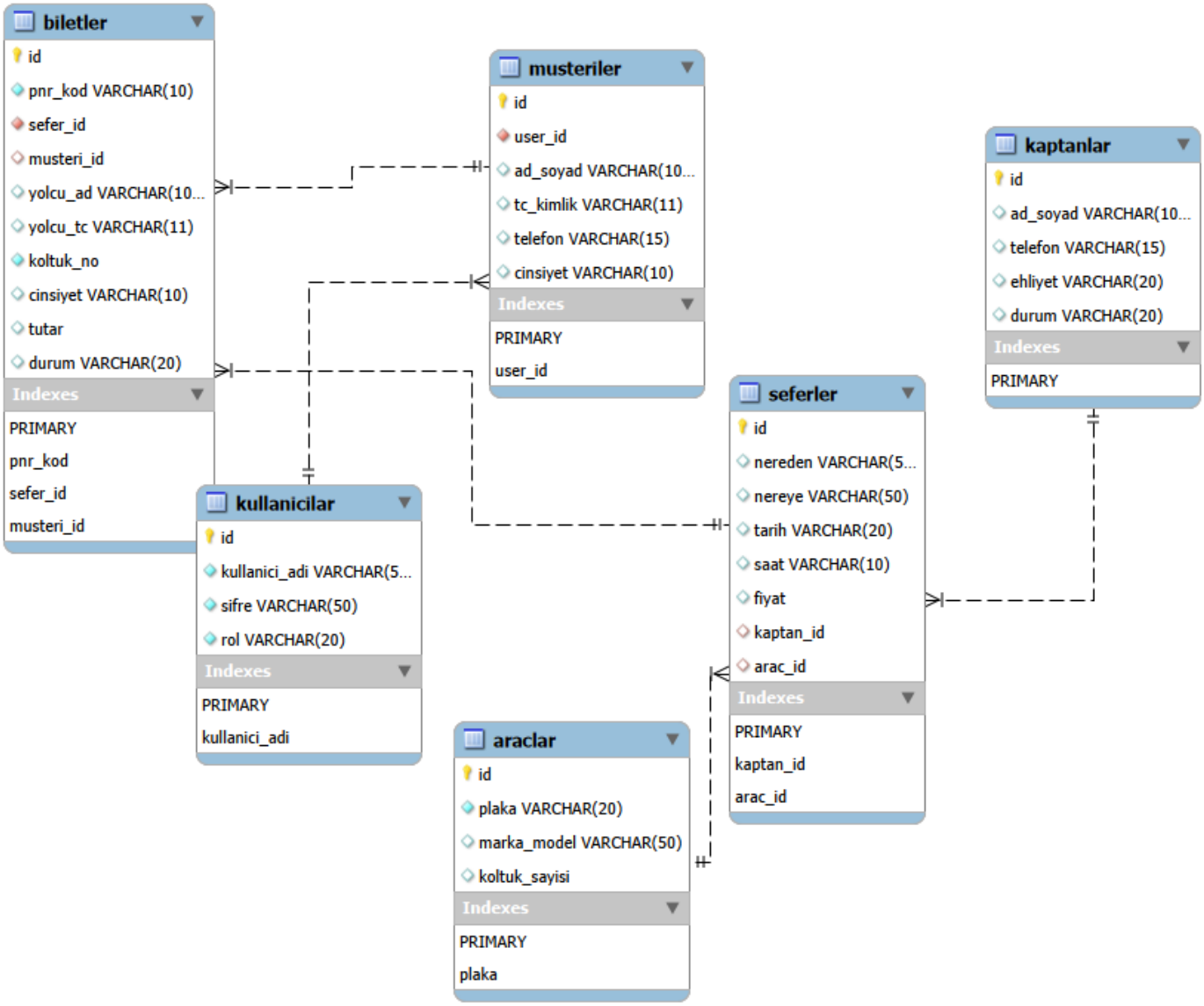
2.2- Programlamaya Yönelik Tasarım

2.2.1- Sistem Çizelgesi (Veri Akışı)

Program, **Nesne Yönelimli Programlama (OOP)** ve **Katmanlı Mimari** prensiplerine sadık kalınarak geliştirilmiştir. Veri akışı şu şekildedir:

1. **Girdi Katmanı:** Kullanıcı arayüzdeki (View) formları doldurur veya butonlara tıklar.
2. **İşlem Katmanı:** Java sınıfları veriyi doğrular (Validasyon) ve iş mantığını çalıştırır.
3. **Veri Erişim Katmanı (DAO):** `DB_Baglanti` sınıfı üzerinden JDBC kullanılarak MySQL veritabanına sorgu gönderilir.
4. **Çıktı Katmanı:** Veritabanından dönen `ResultSet`, nesnelere dönüştürülerek tablolara (`JTable`) veya grafiklere yansıtılır.

İlgili konuya ilişkin görsel aşağıdadır.



2.2.2- Algoritmik Yapı (Kritik Algoritma: Çakışma Kontrolü)

Projenin en kritik algoritmalarından biri **Kaptan Müsaitlik Kontrolü**'dür. Başlangıçta kaptan tablosunda sabit bir "Durum" sütunu tutulması planlanmış, ancak bu yöntemin sürdürülebilir olmadığı (kaptan seferden dönse bile meşgul görünmesi) tespit edilmiştir.

Bunun yerine **Tarih Bazlı Dinamik Sorgu Algoritması** geliştirilmiştir:

1. Admin bir kaptanı yeni bir sefere atamak istediğinde sistem tetiklenir.
2. Seçilen kaptanın ID'si ve Sefer Tarihi parametre olarak alınır.
3. Veritabanına şu SQL sorgusu gönderilir: `SELECT COUNT(*) FROM seferler WHERE kaptan_id = ? AND tarih = ?`
4. Eğer sorgu sonucu `0`'dan büyükse, sistem "Bu kaptan o tarihte başka bir seferde görevli!" uyarısı verir ve atamayı engeller. Bu sayede veri bütünlüğü %100 sağlanır.

2.2.3- Kullanıcı Tanımlı Alt Yordamlar ve Fonksiyonlar

Kod tekrarını önlemek ve modülerliği sağlamak için geliştirilen temel metotlar:

1. **`updateDashboard()`:** Admin paneli her görüntülendiğinde tetiklenir. Veritabanından anlık *Toplam Ciro*, *Satılan Bilet* ve *Müşteri Sayısı* verilerini çeker; paneli temizleyip (`removeAll`) grafikleri yeniden çizdirir (`repaint`).
2. **`seferleriFiltrele(String nereden, nereye, tarih)`:** Kullanıcının arama kriterlerine göre dinamik bir `WHERE` koşulu oluşturur. Sadece gelecekteki ve kriterlere uyan seferleri `JTable` üzerinde listeler.
3. **`createStyledButton(String text, Color color)`:** `JButton` sınıfının `paintComponent` metodu override edilerek oluşturulmuştur. Standart dikdörtgen butonlar yerine; kenarları yuvarlatılmış (Rounded Border), üzerine gelince renk değiştiren (Hover Effect) ve FlatLaf temasına uygun özel butonlar üretir.

```
// 1. ADIM: Kaptan Çakışma Kontrolü (Conflict Check)
String sqlKaptan = "SELECT COUNT(*) FROM seferler WHERE kaptan_id = ? AND
tarih = ?";
PreparedStatement psKaptan = conn.prepareStatement(sqlKaptan);
psKaptan.setInt(1, secilenKaptan.getValue());
psKaptan.setString(2, tarihStr);
ResultSet rsKaptan = psKaptan.executeQuery();

if (rsKaptan.next() && rsKaptan.getInt(1) > 0) {
    JOptionPane.showMessageDialog(this, "HATA: Bu kaptan o tarihte başka bir
seferde görevli!");
    return; // İşlemi durdur
}

// 2. ADIM: Araç Çakışma Kontrolü
String sqlArac = "SELECT COUNT(*) FROM seferler WHERE arac_id = ? AND tarih =
?";
PreparedStatement psArac = conn.prepareStatement(sqlArac);
psArac.setInt(1, secilenArac.getValue());
psArac.setString(2, tarihStr);
ResultSet rsArac = psArac.executeQuery();

if (rsArac.next() && rsArac.getInt(1) > 0) {
    JOptionPane.showMessageDialog(this, "HATA: Bu araç o tarihte zaten
seferde!");
    return; // İşlemi durdur
}
```

3- Sonuçların Test Edilmesi

3.1- Hatalar ve Eksiklikler

Geliştirme sürecinin başında, kaptanların sefer durumu "statik veri" olarak tutulduğunda, ertesi gün olduğunda bile kaptanın sistemde "Seferde" olarak kaldığı görülmüştür. Bu durum, yukarıda açıklanan **Dinamik SQL Sorgusu** mantığına geçilerek tamamen çözülmüştür. Şu an sistem, tarihe göre kaptanın uygunluğunu otomatik hesaplamaktadır.

3.2- Amaca Ne Kadar Hizmet Edebildiği

Geliştirilen "Çağdaş Güven Otomasyonu";

- 4500 satırı aşan kapsamlı kod yapısı,
- İlişkisel veritabanı mimarisi,
- Özelleştirilmiş Swing bileşenleri ile proje hedeflerini tam olarak karşılamaktadır. Manuel bilet kesimindeki hatalar engellenmiş ve yönetimsel raporlama dijitalleşmiştir.

3.3- Sonuç ve Öneriler

Bu proje ile **Bilgisayar Mühendisliği** eğitiminde teorik olarak görülen OOP, Veritabanı Yönetimi ve Algoritma Tasarımı konuları, gerçek dünya senaryosuna sahip ticari bir uygulamaya dönüştürülmüştür. Özellikle **FlatLaf** kullanımı ve **Custom Component** tasarımı ile modern masaüstü uygulama geliştirme yetkinliği kazanılmıştır.