Ömer Kerem Çataklı

Burak Kaan Karaçay

ARAÇ KİRALAMA OTOMASYONU (CARENTO)

1. GİRİŞ VE PROJE KAPSAMI

Bu proje, **BLM2001 - Bilgisayar Programlama II** dersi kapsamında, nesneye dayalı programlama (OOP) prensipleri kullanılarak geliştirilmiş bir masaüstü uygulamasıdır. Projenin temel amacı, bir araç kiralama şirketinin envanter takibini, müşteri yönetimini ve kiralama/iade süreçlerini dijitalleştirmektir.

Proje, kullanıcı dostu bir grafik arayüz (GUI) ile güçlü bir veritabanı altyapısını birleştirerek veri bütünlüğünü ve kullanım kolaylığını hedeflemiştir. İstenen dosya tabanlı (JSON/CSV) yapı yerine, endüstri standartlarına daha uygun olan **İlişkisel Veritabanı (SQLite)** mimarisi tercih edilerek projenin ölçeklenebilirliği artırılmıştır.

2. SİSTEM MİMARİSİ VE TEKNOLOJİ YIĞINI

Proje, **Model-View-Controller (MVC)** tasarım desenine benzer, katmanlı bir mimari ile geliştirilmiştir. Bu yapı, arayüz kodları ile iş mantığını birbirinden ayırarak kodun okunabilirliğini ve bakımını kolaylaştırır.

2.1. Kullanılan Teknolojiler

- **Programlama Dili:** Python 3.13 ve üstü
- **Arayüz Kütüphanesi:** PyQt5 (Qt Designer ile tasarım, Python entegrasyonu)
- **Veritabanı:** SQLite3 (Yerel, sunucu gerektirmeyen SQL motoru)
- **Tasarım Yaklaşımı:** OOP (Kapsülleme, Kalıtım, Çok Biçimlilik)

2.2. Katmanlı Yapı

1. **Veri Katmanı (Data Layer):** `db/` klasörü altındaki modüllerdir. Doğrudan SQL sorgularını çalıştırır ve veritabanı bağlantısını yönetir (Örn: `sql_utils.py`, `arac.py`).
2. **İş Mantığı Katmanı (Business Logic Layer):** Kullanıcı girdilerini doğrulayan, ücret hesaplamalarını yapan ve veri katmanını çağıran sınıflardır (Örn: `Arac` sınıfındaki `kirala` metodu).
3. **Sunum Katmanı (Presentation Layer):** Kullanıcının etkileşime girdiği `.py` dosyalarıdır. `QMainWindow` sınıfından türetilmiştir (Örn: `Admin.py`, `User.py`, `Login.py`).

3. Kullanıcı Arayüzü Akışı

3.1 Giriş Ekranı

- Program açıldığında kullanıcıya **“User”** veya **“Admin”** seçenekleri sunulur.
- Seçilen role göre ilgili panel açılır.

3.2 Araç Listesi

- Her iki panelde de araçlar bir tablo ile listelenir.
- Sol üstteki **Filtreleme Paneli** butonuna tıklayarak araçlar marka, model, uygunluk gibi kriterlere göre filtrelenebilir.

3.3 Admin Paneli

- Araç listesinde herhangi bir araca **sağ tıklayarak**:
    - **Düzenle:** Araç bilgilerini güncelleme.
    - **Sil:** Aracı listeden kaldırma.
- **“Ekle”** butonu ile tabloya yeni araç ekleme imkânı vardır.

3.4 Kullanıcı Paneli

- Araç listesinde her araca karşılık gelen:
    - **Kirala:** Seçilen araç için tarih ve kullanıcı adı girilerek kiralama işlemi yapılır. Araç müsait değilse kiralama gerçekleşmez.
    - **İade Et:** Daha önce kiralanmış araç, kullanıcı adı girilerek iade edilir.
- **Erken İade:** Araç planlanan süreden önce iade edilirse, kalan günlerin ücreti kullanıcıya iade edilir.

3.5 İş Kuralları ve Özellikler

- Kullanıcı sadece uygun araçları kiralayabilir.
- Kiralama sırasında tarih seçimi zorunludur.

- Erken iade durumunda sistem otomatik olarak kalan günlerin ücretini hesaplar ve kullanıcıya iade eder.
- Admin panelinde yapılan ekleme, silme ve düzenleme işlemleri anlık olarak tabloya yansır.

3.6 Teknik Notlar

- Araç bilgileri veri tabanında saklanır.
- Kiralama ve iade işlemleri backend üzerinden kontrol edilir:
    - **Kirala:** Kullanıcı adı ve tarih kontrolü yapılır, araç müsaitse kiralama kaydı oluşturulur.
    - **İade Et:** Kiralama kaydı doğrulanır ve araç müsait duruma getirilir.
    - **Erken İade:** Backend kalan gün sayısını hesaplar ve iade edilecek ücreti belirler.

4. VERİTABANI TASARIMI (SQLite)

Veritabanı, veri tekrarını önlemek ve tutarlılığı sağlamak amacıyla **Normalizasyon** kurallarına uygun tasarlanmıştır. `db/sql_utils.py` ve veritabanı şeması incelendiğinde aşağıdaki yapı ortaya çıkmaktadır:

4.1. Tablolar

- **`arac_tip`:** Marka ve Model bilgilerini tutar. Her marka-model ikilisi tekilleştirilmiştir (`UNIQUE(marka, model)`). Bu sayede veri tekrarı engellenir.
- **`arac`:** Araçların fiziksel bilgilerini tutar.

    - `plaka`: Benzersiz (Unique) alan.
    - `ucret`: Günlük kiralama bedeli.
    - `tip_id`: `arac_tip` tablosuna referans (Foreign Key).
- **`kullanici`:** Sistemi kullanan müşterilerin bilgilerini tutar.
- **`kiralamalar`:** Kiralama işlemlerinin tarihçesini tutar. Hangi aracın, kim tarafından, ne zaman kiralandığı ve iade durumu burada saklanır.

4.2. Görünümler (Views)

- **`arac_view`:** Karmaşık `JOIN` işlemlerini basitleştirmek için oluşturulmuştur. `arac`, `arac_tip` ve `kiralamalar` tablolarını birleştirerek aracın o anki müsaitlik durumunu (`musait_mi`) ve marka/model bilgilerini tek bir sanal tabloda sunar.

5. MODÜLLER VE İŞLEYİŞ DETAYLARI

5.1. Yönetici (Admin) Modülü

Yöneticilerin envanter üzerindeki tam yetkili olduğu modüldür.

- **Dinamik Listeleme ve Filtreleme:** `Admin.py` içerisinde `listele()` fonksiyonu, kullanıcının girdiği marka, model, fiyat aralığı ve durum filtrelerine göre dinamik bir SQL sorgusu oluşturur (`Arac.listele` metodu aracılığıyla).
- **Sağ Tık Menüsü (Context Menu):** Tablo üzerinde sağ tıklanarak "Düzenle" ve "Sil" seçeneklerine hızlı erişim sağlanmıştır (`table_context_menu` fonksiyonu).

- **CRUD İşlemleri:** Araç ekleme, silme ve güncelleme işlemleri anlık olarak veritabanına yansıtılır ve arayüz `PyQt5 Signal` yapısı kullanılarak yenilenir.

5.2. Müşteri (User) Modülü

Müşterilerin araç kiraladığı modüldür.

- **Kiralama Mantığı (`Rent.py`):**
    - Kullanıcı tarih seçtiğinde, `QDateEdit` bileşenindeki `dateChanged` sinyali tetiklenir ve `ucret_degistir` fonksiyonu anlık olarak toplam tutarı hesaplar.
    - **Algoritma:** `(Bitiş Tarihi - Başlangıç Tarihi) * Günlük Ücret` formülü ile hesaplama yapılır.
- **İade Mantığı (`Return.py`):**
    - İade ekranında, aracın kiralandığı tarihler ve sistem tarihi (bugün) karşılaştırılır.
    - Kullanıcı adı doğrulaması yapılarak, aracı kiralayan kişi ile iade eden kişinin aynı olması zorunluluğu getirilmiştir (`hatalar.YanlisKullanici`).

5.3. Nesne Yönelimli Yaklaşım (OOP)

`db/arac.py` dosyasındaki `Arac` sınıfı, projenin OOP gücünü gösterir:

- **Property Decorators:** `@property` ve `@setter` kullanılarak, sınıfın özelliklerine (örneğin `plaka` veya `ucret`) atama yapıldığı anda veritabanı güncellemesi otomatik tetiklenir. Bu, veri tutarlılığını kod seviyesinde garanti altına alır.

- **Statik Metotlar:** `Arac.bul(plaka)` ve `Arac.listele()` gibi metotlar, nesne oluşturulmadan veritabanından veri çekilmesini sağlar (Factory Pattern benzeri kullanım).

6. HATA YÖNETİMİ VE GÜVENLİK

Projede kullanıcı deneyimini iyileştirmek için özel hata sınıfları (`Custom Exceptions`) tanımlanmıştır:

- **Veri Doğrulama:** Boş alan bırakılması (`BosStringHatasi`), negatif ücret girilmesi (`GecersizUcretHatasi`) engellenir.
- **Mantıksal Kontroller:** Bitiş tarihinin başlangıç tarihinden önce seçilmesi (`GecersizTarihHatasi`) gibi durumlarda kullanıcı `QMessageBox` ile uyarılır.

7. SONUÇ

Bu proje, Python ve PyQt5 kullanılarak geliştirilmiş, sağlam bir veritabanı altyapısına sahip, genişletilebilir bir araç kiralama otomasyonudur. Modüler yapısı sayesinde yeni özelliklerin eklenmesine açıktır ve dersin kazanımları olan veritabanı yönetimi, GUI tasarımı ve algoritma geliştirme yetkinliklerini başarıyla sergilemektedir.