T.C. KARADENİZ TEKNİK ÜNİVERSİTESİ

OF/TEKNOLOJİ FAKÜLTESİ

YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ

CATLARY KÜTÜPHANE YÖNETİM SİSTEMİ Proje Raporu

Ders Adı**:** Veri Tabanı-Hakan Aydın

Öğrenci Ad-soyad: Melisa Taşkara(436581)

Github hesabı: https://github.com/melisa-6

Teslim Tarihi: 29 / 12 / 2025

PROJENİN AMACI VE KAPSAMI

Catlary Kütüphane Yönetim Sistemi projesinin temel amacı, bir kütüphanede bulunan kitapların, kullanıcıların ve ödünç alma işlemlerinin dijital ortamda yönetilebileceği bütünleşik bir sistem geliştirmektir.

Projenin Amacı

*Kütüphane yönetim süreçlerini otomatikleştirmek
*Kitap, kullanıcı, yazar ve kategori yönetimini sağlamak
*Ödünç alma ve iade işlemlerini kontrol altına almak
*Geç iade durumlarında otomatik ceza hesaplamak
*Veritabanı ile entgere çalışan REST API'ler geliştirmek

Projenin Kapsamı

Proje aşağıdaki işlevleri kapsamaktadır:

Kullanıcı girişi (JWT tabanlı kimlik doğrulama)
Kitap arama ve listeleme
Kitap ödünç alma ve iade işlemleri
Geç iade durumunda ceza hesaplama ve kayıt altına alma
Kullanıcılar için CRUD işlemleri
API testleri ve temel kullanıcı arayüzü

KULLANILAN TEKNOLOJİLER

Bu projede kullanılan teknolojiler aşağıda listelenmiştir:

Backend: Flask (Python)
Veritabanı: MySQL
Frontend: HTML, CSS, JavaScript
Kimlik Doğrulama: JWT (JSON Web Token)
API Testi: Postman

SİSTEM MİMARİSİ

Projede katmanlı mimari yapısı kullanılmıştır. Bu mimari sayesinde kodun okunabilirliği, sürdürülebilirliği ve yönetilebilirliği artırılmıştır.

Kullanılan katmanlar şunlardır:

Entity Katmanı: Veritabanı tablolarını temsil eden sınıflar
Repository Katmanı: Veritabanı işlemlerinin gerçekleştirildiği katman
Service Katmanı: İş kurallarının tanımlandığı katman
Controller Katmanı: API uç noktalarının tanımlandığı katman

VERİTABANI TASARIMI

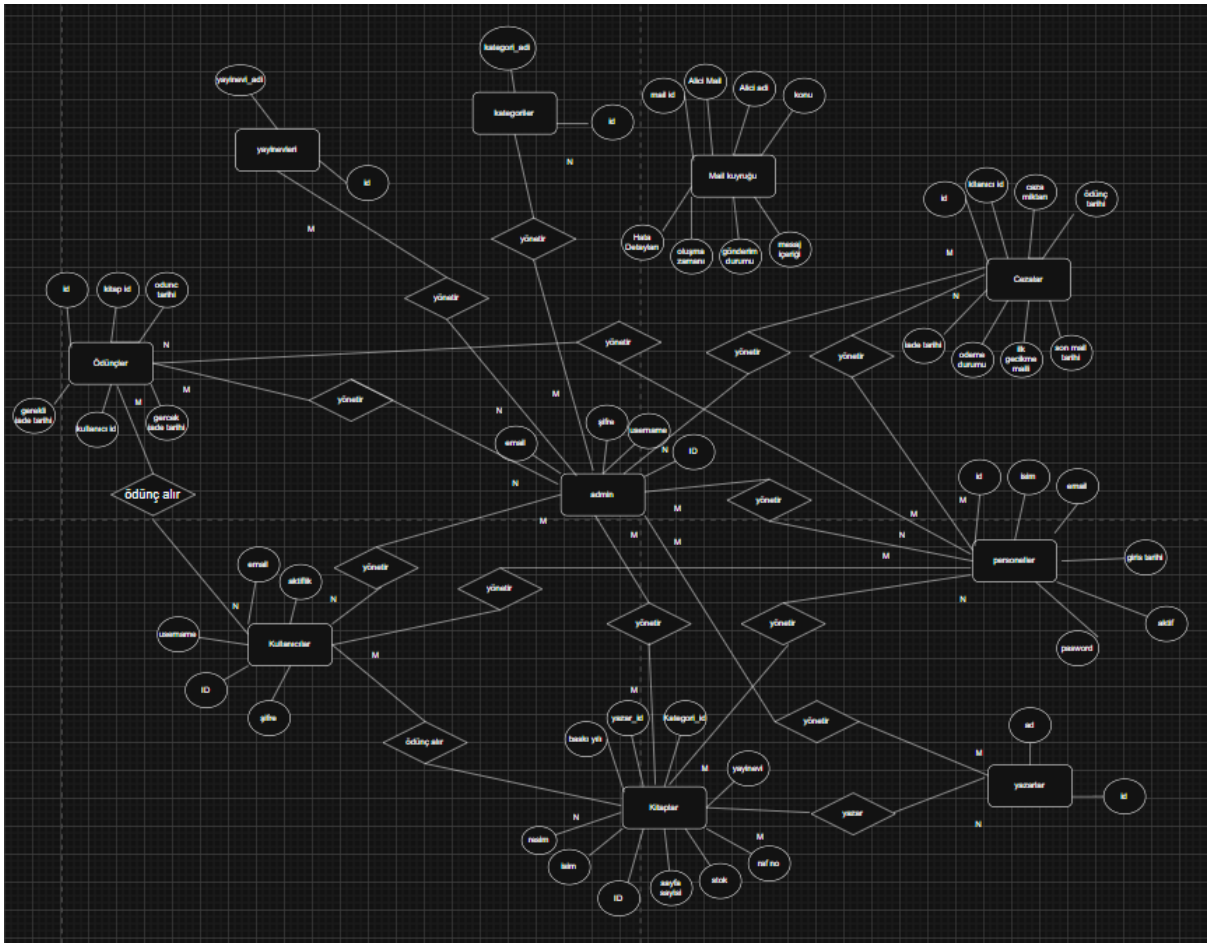
Veritabanı tasarımı, sistemin temelini oluşturmaktadır. Projede ilişkisel veritabanı modeli kullanılmıştır.

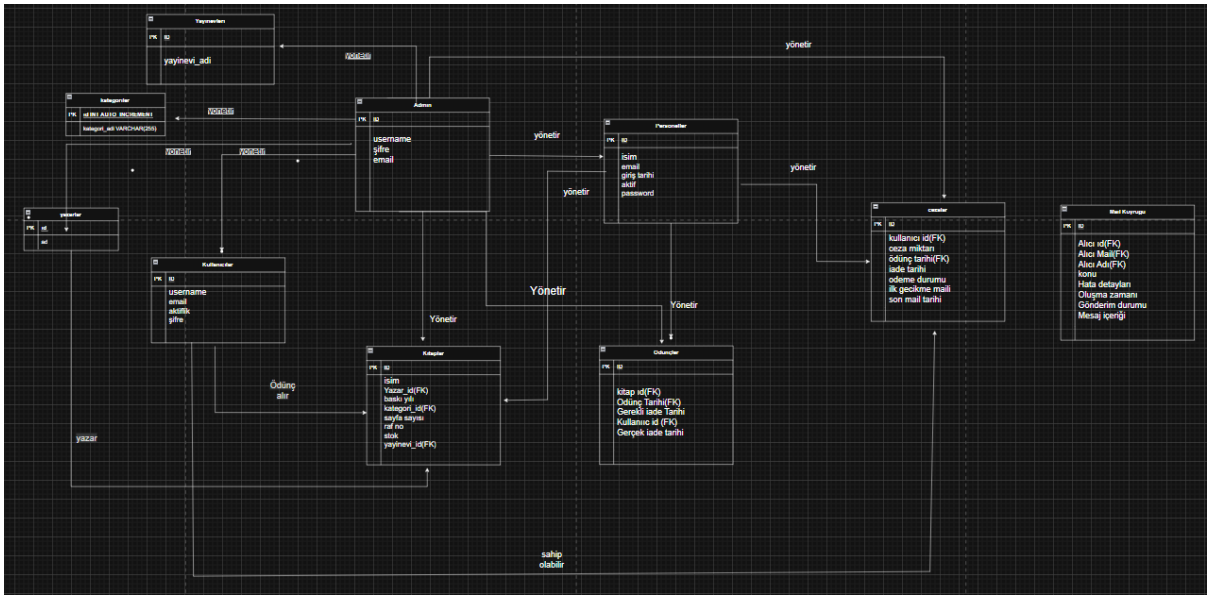
ER Diyagramı

Sistemde yer alan temel varlıklar şunlardır:

- Admin
- Personel
- Kullanıcı
- Kitap
- Yazar
- Kategori
- Ödünç
- Ceza
- Yayınevi

Bu varlıklar arasındaki ilişkiler ER diyagramı ile gösterilmiştir.





Veritabanı Mimarisi: Trigger, Stored Procedure ve Event Kullanımı

Bu projede veritabanı mimarisi, iş kurallarının uygulama katmanından bağımsız olarak kontrol edilebilmesi, veri tutarlılığının korunması ve sistem performansının artırılması amacıyla Stored Procedure, Trigger ve Event yapıları kullanılarak tasarlanmıştır.

Ayrıca, özellikle test ve sunum süreçlerinde sistem davranışlarını anlık olarak gözlemleyebilmek için veritabanı; “Gerçek Zamanlı Dakikalık Simülasyon” mantığında çalışacak şekilde yapılandırılmaya çalışılmıştır.

1) Stored Procedures

Sistemdeki karmaşık ve tekrar eden işlemler, performans ve yönetilebilirlik sağlamak amacıyla saklı yordamlar içinde toplanmıştır:

*ceza : Klasik sistemlerdeki günlük faiz hesaplamalarından farklı olarak bu prosedür, dakikalık hassasiyetle çalışacak şekilde tasarlanmıştır.
Kitap ödünç verildiği anda süre işlemeye başlar ve teslim süresi dolduğunda gecikme dakikası bazlı ceza üretir. Kitap iade edilmediği sürece kullanıcının cezaları dakika bazında güncellenir.

*iadebugunhatirlatmasi : Kitabın teslim süresi dolduğu anda otomatik olarak çalışır ve kullanıcıya “Süreniz doldu” içeriğinde e-posta bildirimi oluşturur.

*AnaKontrol: Sistemin ana kontrol mekanizmasıdır. Çalıştığında Ceza ve Hatırlatma prosedürlerini sırasıyla tetikleyerek tüm süreçleri senkronize şekilde yönetir.

*AdminRaporMailiOlustur : Son 5 dakikadaki kitap ödünç alma, iade ve ceza hareketlerini raporlayarak yöneticiler için otomatik bir bilgilendirme e-postası oluşturur.

2) Events (Zamanlanmış Olaylar)

Sistemin tamamen otonom çalışabilmesi için veritabanına iki adet zamanlanmış olay yerleştirilmiştir:

*DakikalikGenelKontrol (Her 1 Dakikada): Sistemin temel kontrol noktasıdır. Her dakika çalışarak AnaKontrol prosedürünü tetikler ve gecikme durumlarını anlık olarak işler.

*DakikalikRaporlama (Her 5 Dakikada): Yönetici raporları için kullanılır. Kullanıcı aktivitelerinden bağımsız olarak sistem verilerini analiz eder ve yönetime göndermek üzere e-posta üretir.

3)Triggers (Tetikleyiciler)

Veri güvenliğini ve bütünlüğünü korumak için çeşitli tetikleyiciler kullanılmıştır:

*Stok Yönetimi Tetikleyicileri (stok_azaltma, stok_arttir):

- Ödünç verildiğinde stok otomatik olarak azalır.
- Kitap iade edildiğinde stok artırılır.
  Bu işlemler doğrudan veritabanı seviyesinde garanti altına alınmıştır.

*Bildirim Tetikleyicileri (odunc_mail, iademaili): Ödünç alma ve iade sırasında tetiklenir. Dakika hassasiyetinde zaman damgası oluşturarak kullanıcı için bilgilendirme e-postası üretir.

*Ceza Ödeme Güvenlik Tetikleyicisi (trg_ceza_odeme_kontrol): Sistemde “Kitap iade edilmeden ceza ödenemez.” kuralını uygular.
Eğer kitap kullanıcıdayken ceza ödeme girişimi yapılırsa işlem durdurulur ve hata mesajı üretilir.

Özetle bu yapı sayesinde:

Bir kullanıcı kitabı 18:00’da ödünç aldığında, sistem:

- 18:01’de gecikmeyi tespit eder,
- cezayı hesaplar,
- kullanıcıya hatırlatma maili oluşturur,
- stok verisini günceller.

Tüm bu işlemler, uygulama kodundan bağımsız olarak doğrudan veritabanı motoru üzerinde otomatik şekilde gerçekleşir.

BACKEND (API) GELİŞTİRME

Backend tarafında REST mimarisi kullanılarak API uç noktaları geliştirilmiştir.
Örnek API uç noktaları:

*Kimlik Doğrulama & Oturum

HTTP Metodu	Endpoint	Açıklama
GET	/	Uygulama ana sayfası
POST	/kayitol	Yeni kullanıcı kayıt işlemi
POST	/girisyap	Kullanıcı / Admin / Personel giriş
GET	/cikis	Kullanıcı çıkışı

*Kullanıcı İşlemleri

HTTP	Endpoint	Açıklama
GET	/kullanici_sayfasi	Kullanıcı ana sayfası
POST	/kullaniciekle	Admin/Personel tarafından kullanıcı ekleme
POST	/kullanici_durum_degistir	Kullanıcı aktif/pasif durumu değiştirme
GET	/kullanicilar	Tüm kullanıcıları listeleme
POST	/kullanici_sifre_degistir	Kullanıcı kendi şifresini değiştirme
POST	/kullanici_sifre_sifirla	Admin/Personel kullanıcı şifre sıfırlama

*Admin İşlemleri

HTTP	Endpoint	Açıklama
GET	/admin_anasayfasi	Admin paneli
POST	/adminekle	Yeni admin ekleme
POST	/adminsil	Admin silme
GET	/adminler	Admin listesini görüntüleme
POST	/admin_sifre_degistir	Admin şifre değiştirme

* Personel İşlemleri

HTTP	Endpoint	Açıklama
GET	/personel_sayfasi	Personel paneli
POST	/personel_ekle	Personel ekleme
GET	/personel_liste	Personel listeleme
POST	/personel/aktiflik_degistir	Personel aktif/pasif
POST	/personel_sifre_degistir	Personel şifre değiştirme
POST	/personel_sifre_sifirla	Admin tarafından personel şifre sıfırlama

* Kitap İşlemleri

HTTP	Endpoint	Açıklama
GET	/kitaplari_goruntule	Kitap listeleme / arama
POST	/kitapekle	Kitap ekleme
POST	/kitapsil	Kitap silme

*Ödünç Alma – İade İşlemleri

HTTP	Endpoint	Açıklama
GETPOST	/kitapoduncver	Kitap ödünç verme
POST	/kitapiadeal	Kitap iade alma
GET	/oduncalmagecmisim/<username>	Kullanıcı ödünç geçmişi
GET	/tumkullanicioduncalmagecmisigoster	Tüm kullanıcıların ödünç geçmişi

*Ceza ve Ödeme İşlemleri

HTTP	Endpoint	Açıklama
GET	/cezatablosunugoster	Tüm cezaları listeleme
GET	/cezalarimigoster/<username>	Kullanıcının kendi cezaları
POST	/ceza_sorgula	Ceza ID ile borç sorgulama
POST	/cezaode	Ceza ödeme
GET/POST	/kendiborc_ode/<username>	Kullanıcının kendi borcunu ödeme

*Yayınevleri

HTTP	Endpoint	Açıklama
GET	/yayinevleri	Yayinevlerini listeleme
POST	/yayineviekle	Yayınevi ekleme
POST	/yayinevisil	Yayınevi silme

*Kategoriler

HTTP	Endpoint	Açıklama
GET	/kategoriler	Kategori listeleme
POST	/kategoriekle	Kategori ekleme
POST	/kategorisil/<id>	Kategori silme

*Yazarlar

HTTP	Endpoint	Açıklama
GET	/yazarlar	Yazar listeleme
POST	/yazarekle	Yazar ekleme
POST	/yazarsil/<id>	Yazar silme

FRONTEND TASARIMI

Frontend tarafında kullanıcıların sistemle etkileşime geçebileceği temel arayüzler geliştirilmiştir.

***Anasayfa ve Giriş Sayfaları**

Sayfa Adı	Dosya	Açıklama
Anasayfa	anasayfa.html	Sistemin genel tanıtımının yapıldığı ve kullanıcıların giriş yönlendirmelerinin bulunduğu ana ekran
Kullanıcı Paneli	kullanici.html	Giriş yapan kullanıcıların temel işlemleri gerçekleştirdiği ana panel
Personel Paneli	personel.html	Giriş yapan personellerin rolune uygun işlemleri gerçekleştitdiği panel
Admin Paneli	admin.html	Giriş yapan adminlerin rolune uygun işlemleri gerçekleştirdiği panel

***Kitap, Yazar ve Kategori Yönetimi**

Sayfa Adı	Dosya	Açıklama
Kitap Listesi	kitaplar.html	Kitap arama, listeleme,silme ,düzenleme ve stok görüntüleme ekranı
Yazar Yönetimi	yazarlar.html	Yazar ekleme, silme ve listeleme işlemleri
Kategori Yönetimi	kategoriler.html	Kategori ekleme, silme ve listeleme işlemleri
Yayinevi Yönetimi	yayinevleri.html	Yayınevi ekleme,silme be listeleme işlemleri

***Ödünç Alma ve İade İşlemleri**

Sayfa Adı	Dosya	Açıklama
Ödünç Verme	oduncver.html	Kitap ödünç verme işlemlerinin gerçekleştirildiği sayfa
Ödünç Alma	odunc.html	Kullanıcının kitap ödünç alma işlemleri
Ödünç Geçmişim	odunc_gecmisim.html	Kullanıcının kendi ödünç geçmişini görüntülediği ekran
Tüm Ödünç Geçmişi	tum_odunc_gecmisi.html	Admin ve personelin tüm ödünç kayıtlarını görüntülediği ekran

***Ceza ve Borç İşlemleri**

Sayfa Adı	Dosya	Açıklama
Borç Ödeme	borcode.html	Kullanıcının ceza borçlarını ödeyebildiği sayfa
Cezalarım	cezalarim.html	Kullanıcıya ait gecikme cezalarının listelendiği ekran
Tüm Cezalar	tum_cezalar.html	Admin ve personelin tüm ceza kayıtlarını görüntülediği ekran

***Admin Arayüzü**

Sayfa Adı	Dosya	Açıklama
Admin Ana Sayfası	admin.html	Admin yetkisine sahip kullanıcıların yönetim işlemlerine eriştiği panel
Admin Listesi	adminler.html	Sistemde tanımlı admin kullanıcıların listelendiği ekran
Kullanıcı Yönetimi	kullanicilar.html	Kullanıcı ekleme, silme ve durum yönetimi işlemleri

Personel Yönetimi	personel.html	Personel ekleme ve yönetim işlemleri
Personel Listesi	personel_listesi.html	Kayıtlı personellerin listelendiği sayfa

TEST SÜRECİ

Geliştirilen API'ler Postman aracılığıyla test edilmiştir. Başarılı ve hatalı senaryolar değerlendirilmiştir.

Örnek senaryo(GET isteği);

Postmanden gelen istek önce görselde gösterilen Endpoint'e gelir ve yönlendirilir.

```python
@yazar_bp.route("/yazarlar", methods=["GET"])
@login_required # Giriş zorunluluğu.
def yazarlar_getir_route():
    # İstemcinin JSON yanıtı isteyip istemediğini kontrol eder
    is_api_request = request.accept_mimetypes.best == "application/json" or request.is_json

    # Controller'dan yazarları çeker
    yazarlar = yazarlar_controller.tum_yazarlari_getir_controller()

    basarili_mi = True
    mesaj = "Yazarlar listelendi"

    # API isteği için JSON dönüşü.
    if is_api_request:
        return jsonify({
            "success": basarili_mi,
            "message": mesaj,
            "yazarlar": yazarlar
        }), 200

    # Web tarayıcısı için HTML dönüşü.
    return render_template(
        "yazarlar.html",
        yazarlar=yazarlar,
        username=getattr(g, "username", "")
    )
```

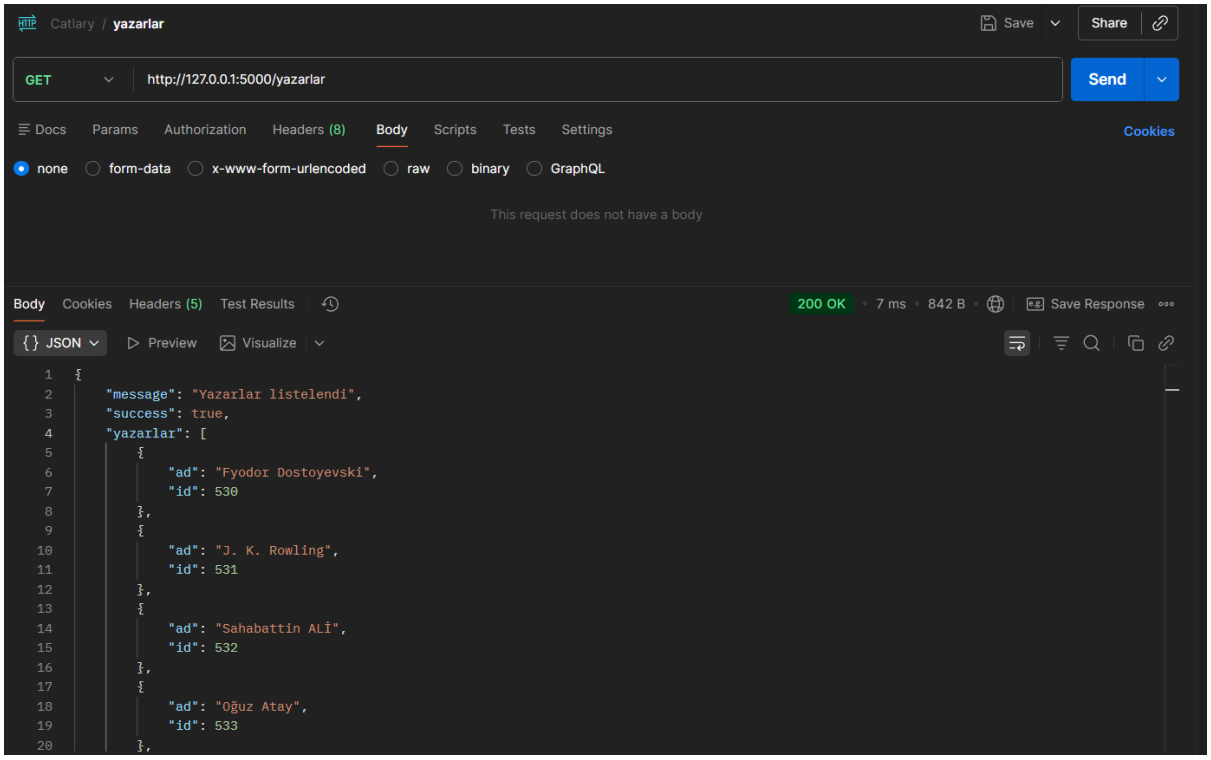
```python
def tum_yazarlari_getir_controller(self):
    #uygun service e yonlendirir
    try:
        return self.service.tum_yazarlar()
    except Exception as e:
        print(f"Controller HATA: Yazarlar çekilemedi: {e}")
        return []
```

```python
def tum_yazarlar(self):
    #uygun repoya yonlendirir
    return self.repo.tum_yazarlar()
```

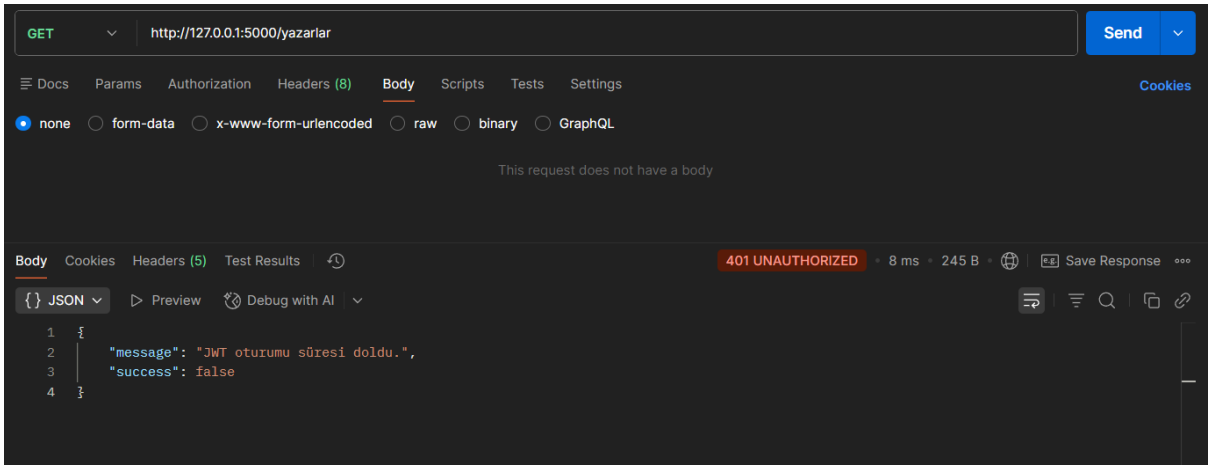
```python
def tum_yazarlar(self):
    #yazarlar tablosundan tum yazarları ceker
    try:
        cursor = self.connection.cursor(dictionary=True)
        #sonucları dict formatında dondurmesi icin
        cursor.execute("SELECT * FROM yazarlar")
        sonuc = cursor.fetchall()
        cursor.close()
        return sonuc
    except Exception as e:
        print("Repository HATA:", e)
        return []
```

Postman’den isteğin gönderilmesi:

*Başarılı olma durumu



*Hata verme durumu



Örnek senaryo (Post isteği);

Postmanden gelen istek önce görselde gösterilen Endpoint'e gelir ve yönlendirilir.

```python
@yazar_bp.route("/yazarekle", methods=["POST"])
def yazarekle():
    # Yazar adını alır
    ad = request.form.get("yazar_adi") or (request.json.get("yazar_adi") if request.is_json else None)

    # Controller ile yazarı ekler
    success, msg = yazarlar_controller.yazarekle_controller(ad)

    # Eğer istek JSON formatındaysa JSON dön.
    if request.is_json:
        return jsonify({
            "success": success,
            "message": msg
        }), (200 if success else 400)

    # Değilse Flash mesajı oluştur.
    flash(msg, "success" if success else "danger")

    yazarlar = yazarlar_controller.tum_yazarlari_getir_controller()
    return render_template("yazarlar.html", yazarlar=yazarlar)
```

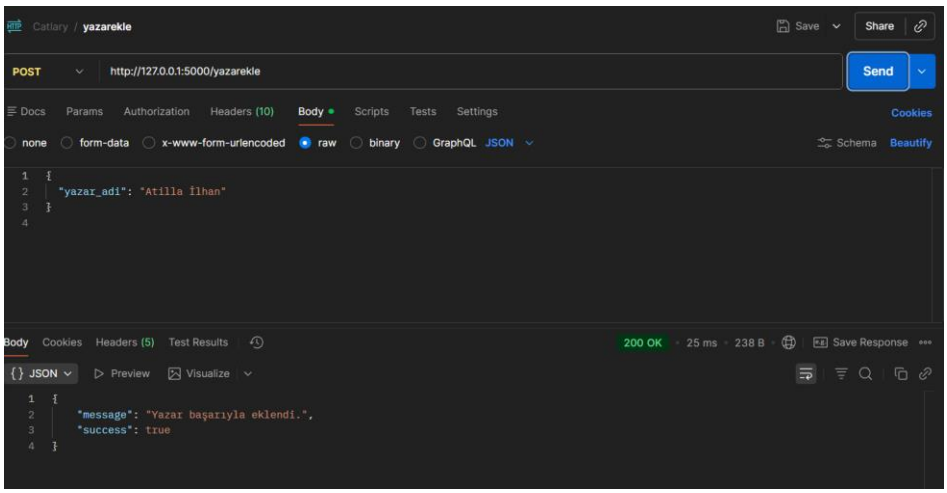
```python
def yazarekle_controller(self, ad):
    mevcut = self.service.yazar_bul(ad)
#yazar mevcutsa hata verir
    if mevcut:
        return False, "Bu yazar zaten kayıtlı!"
#mevcut deilse eklemek icin ilgili service yonlendirir
    self.service.yazar_ekle(ad)
    return True, "Yazar başarıyla eklendi."
```

```python
#uygun repoya yonlendirir
def yazar_bul(self, ad):
    return self.repo.yazar_bul(ad)
#uygun repoya yonlendirir
def yazar_ekle(self, ad):
    return self.repo.yazar_ekle(ad)
```

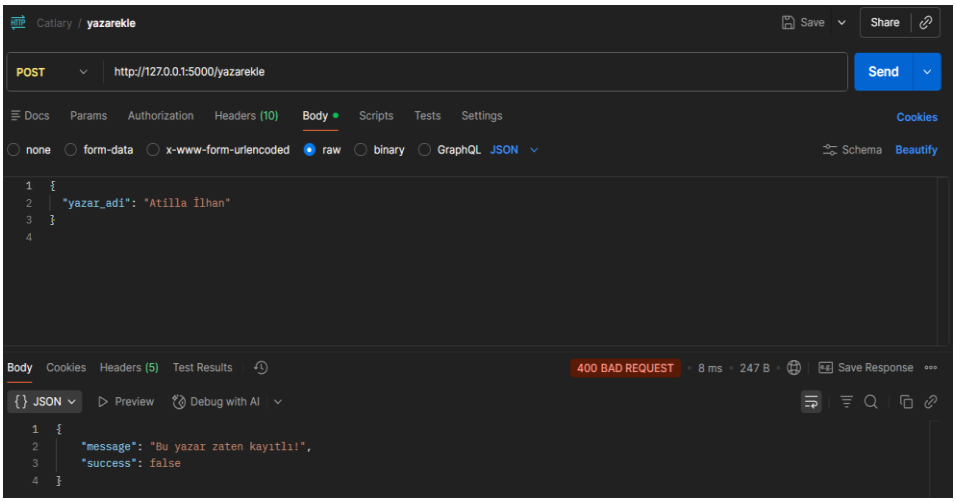
```python
def yazar_bul(self, ad):
    try:
        cursor = self.connection.cursor(dictionary=True)
        sql = "SELECT * FROM yazarlar WHERE ad = %s"
        cursor.execute(sql, (ad,))
        sonuc = cursor.fetchone()
        cursor.close()
        return sonuc
    except Exception as e:
        print("Yazar Bul Hatası:", e)
        return None

def yazar_ekle(self, yazar_adi):
    try:
        cursor = self.connection.cursor()
        query = "INSERT INTO yazarlar (ad) VALUES (%s)"
        cursor.execute(query, (yazar_adi,))
        self.connection.commit()
        cursor.close()
        return True
    except Exception as e:
        print("Ekleme Hatası:", e)
        return False
```

*Postman’den isteğin gönderilmesi:

*Başarılı olma durumu:

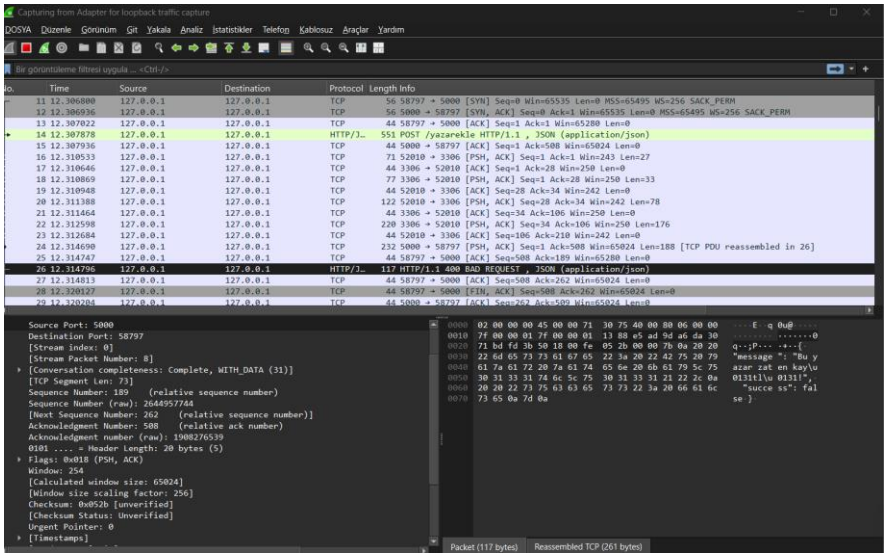


*Hata durumu:

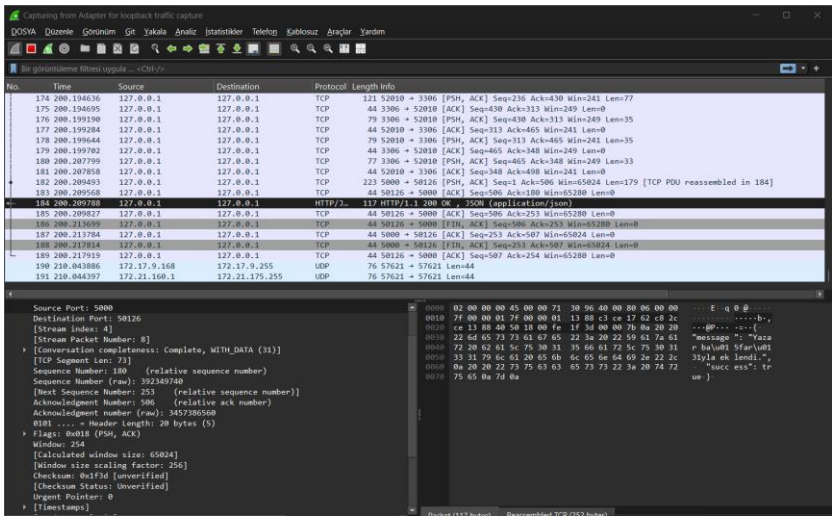


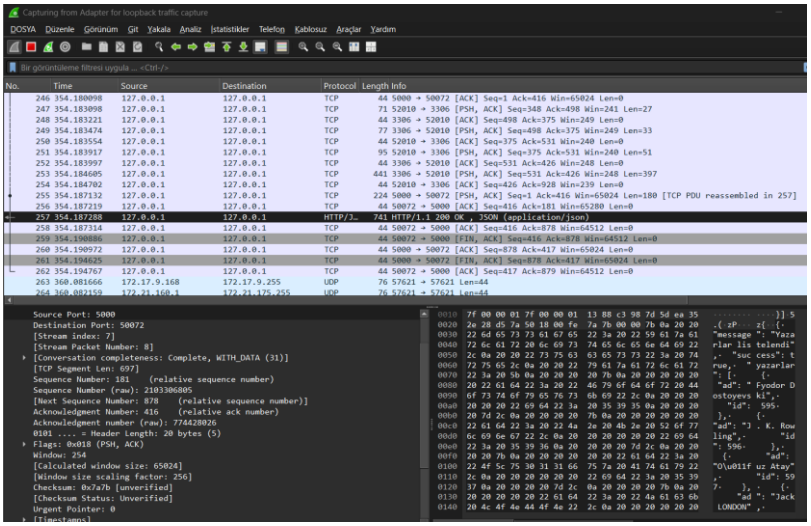
İsteklerin WireShark'ta görüntülenmesi:

*Hata durumu



*Başarılı olma durumu:





VİDEO TANITIMI

Projenin tanıtımı amacıyla kısa bir demo videosu hazırlanmış ve YouTube platformuna yüklenmiştir.

Youtube Catlary Kanal linki: www.youtube.com/@Catlaryy