KARADENİZ TEKNİK ÜNİVERSİTESİ

OF TEKNOLOJİ FAKÜLTESİ

YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ

VERİ TABANI YÖNETİMİ

2025-2026 GÜZ DÖNEMİ

AKILLI KÜTÜPHANE OTOMASYONU PROJE RAPORU

HAZIRLAYAN: 445849

ESRA KARACA

İÇİNDEKİLER

1. ÖZET

Bu proje kapsamında, geleneksel kütüphane süreçlerinin dijitalleştirilmesini sağlayan "Akıllı Kütüphane Otomasyon Sistemi" geliştirilmiştir. Sistem; kitap envanterinin yönetilmesi, üyelerin takibi, ödünç alma süreçlerinin kayıt altına alınması ve iade edilmeyen kitaplar için otomatik ceza hesaplanması işlevlerini yerine getirmektedir.

Proje, Java programlama dili ve Spring Boot çerçevesi kullanılarak geliştirilmiş, veri tabanı yönetim sistemi olarak Microsoft SQL Server tercih edilmiştir. Arayüz tasarımında Thymeleaf şablon motoru ve Bootstrap 5 kütüphanesi kullanılarak, tüm cihazlarla uyumlu (responsive) modern bir görünüm elde edilmiştir. Projenin en özgün yanı, sunum ve test süreçlerini kolaylaştırmak adına entegre edilen "Hızlandırılmış Zaman Modu"dur; bu mod sayesinde ödünç alınan kitapların iade süresi 1 dakika olarak simüle edilmekte ve dakika bazlı ceza yansıtılmaktadır.

Github linki:

https://github.com/esrakaracaa/veri-taban-proje

2.GİRİŞ

2.1. Projenin Amacı

Günümüzde bilgiye erişimin hızı, kurumların verimliliğini doğrudan etkilemektedir. Manuel olarak yürütülen kütüphane işlemleri (kart sistemi, defter kaydı vb.) insan hatasına açık, yavaş ve istatistiksel veri sunmaktan uzaktır. Bu projenin temel amacı:

- Kitap stok durumunun anlık olarak izlenmesi.
- Hangi kitabın kimde olduğunun ve ne zaman döneceğinin saniyeler içinde tespit edilmesi.
- Gecikme cezalarının manuel hesaplama hatalarından arındırılıp sistem tarafından otomatik hesaplanması.
- Kullanıcı (Öğrenci) ve Personel (Yönetici) rollerinin ayrılarak güvenli bir yönetim ortamı sağlanmasıdır.

2.2. Projenin Kapsamı

Sistem üç ana modülden oluşmaktadır:

1.Yönetici Modülü: Kitap ekleme/silme/güncelleme, üye yönetimi, iade alma ve istatistik görüntüleme.

2.Üye Modülü: Kitap kataloğunu tarama, mevcut ödünçlerini görüntüleme, güncel borç durumunu takip etme ve iade talebi oluşturma.

3.Sistem Arka Planı: Otomatik ceza hesaplama, stok düşme/arttırma trigger mantıkları (servis seviyesinde).

2.3. Hedef Kitle

Sistem, üniversite kütüphaneleri, halk kütüphaneleri ve okul kütüphaneleri gibi hem envanterin hem de kullanıcı sirkülasyonunun yoğun olduğu kurumlar için tasarlanmıştır.

3. MATERYAL VE YÖNTEM

3.1. Kullanılan Yazılım Teknolojileri

Projenin geliştirilmesinde endüstri standardı olan sağlam ve ölçeklenebilir teknolojiler tercih edilmiştir:

- **Backend (Sunucu Tarafı):**
    - **Java 17:** Uzun süreli destek (LTS) sürümü olması ve performans avantajları nedeniyle seçilmiştir.
    - **Spring Boot 3.4.12:** Uygulamanın konfigürasyon yükünü azaltmak ve hızlı geliştirme (Rapid Application Development) sağlamak için kullanılmıştır.
    - **Spring Data JPA (Hibernate):** SQL sorgularını nesne tabanlı (ORM) yapıya dönüştürerek veri tabanı bağımsızlığı ve geliştirme hızı sağlamıştır.
- **Veri Tabanı:**
    - **Microsoft SQL Server:** İlişkisel veri tabanı yönetim sistemi (RDBMS) olarak kullanılmıştır. Bağlantı application.properties dosyasında com.microsoft.sqlserver.jdbc.SQLServerDriver ile yapılandırılmıştır.
- **Frontend (Ön Yüz):**
    - **Thymeleaf:** Sunucu taraflı Java verilerinin HTML içerisine dinamik olarak gömülmesi için kullanılmıştır.
    - **Bootstrap 5 & FontAwesome**: Kullanıcı dostu arayüz elementleri ve ikonlar için kullanılmıştır.

3.2. Geliştirme Ortamı ve Araçlar

- **IDE:** Visual Studio Code (Proje dosyalarındaki .vscode klasöründen anlaşılmaktadır).
- **Build Tool:** Apache Maven (Bağımlılık yönetimi ve proje derleme).

4. SİSTEM ANALİZİ VE VERİ TABANI TASARIMI

Veri tabanı tasarımı, verilerin tutarlılığını sağlamak amacıyla "Normalization" kurallarına uygun olarak yapılmıştır.

4.1. Varlık-İlişki (E-R) Analizi ve Tablolar

Sistemde kullanılan ana tablolar ve görevleri şöyledir:

1. **Kullanicilar (Users):**
    - **Sistemi kullanan herkesin (Yönetici, Personel, Üye) tutulduğu tablodur.**
    - ***Alanlar:*** **KullaniciId (PK), AdSoyad, Email (Unique), Sifre, Rol (Enum: ADMIN, UYE), Borc (Ceza bakiyesi).**
2. **Kitaplar (Books):**
    - **Kütüphane envanteridir.**
    - ***Alanlar:*** **KitapId (PK), KitapAdi, ISBN, BasimYili, Stok (Adet), AktifMi.**
    - ***İlişkiler:*** **YazarId (FK -> Yazarlar), KategoriId (FK -> Kategoriler).**
3. **OduncIslemleri (Loans):**
    - **Sistemin işlem hareketlerini tutan en kritik tablodur.**
    - ***Alanlar:*** **OduncId (PK), OduncTarihi, BeklenenIadeTarihi, TeslimTarihi, CezaMiktari, Durum.**
    - ***İlişkiler:*** **KullaniciId (FK -> Kullanicilar), KitapId (FK -> Kitaplar).**
4. **Yazarlar & Kategoriler:**
    - **Kitapları normalize etmek ve veri tekrarını önlemek için kullanılan Lookup (Referans) tablolarıdır.**

4.2. Tablo İlişkileri

- **One-to-Many (1:N): Bir Yazarın birden çok Kitabı olabilir.**
- **One-to-Many (1:N): Bir Kategoride birden çok Kitap olabilir.**
- **One-to-Many (1:N): Bir Kullanıcı birden çok kez ödünç alma işlemi yapabilir (OduncIslemleri tablosuna bağlantı).**

SQL DE TABLO OLUŞTURMA

SQLQuery20.sql - bağlı değil* SQLQuery2.sql - bağlı değil*

```
USE LibraryDB;
GO
CREATE TABLE Kullanicilar (
    KullaniciId INT IDENTITY(1,1) PRIMARY KEY,
    AdSoyad NVARCHAR(100) NOT NULL,
    Email NVARCHAR(100) UNIQUE NOT NULL,
    Sifre NVARCHAR(255) NOT NULL,
    Rol NVARCHAR(20) NOT NULL, -- ADMIN / OGRENCI
    AktifMi BIT DEFAULT 1,
    Borc DECIMAL(10,2) DEFAULT 0,
    OlusturmaTarihi DATETIME DEFAULT GETDATE()
);
CREATE TABLE Yazarlar (
    YazarId INT IDENTITY(1,1) PRIMARY KEY,
    AdSoyad NVARCHAR(100) NOT NULL,
    AktifMi BIT DEFAULT 1
);
CREATE TABLE Kategoriler (
    KategoriId INT IDENTITY(1,1) PRIMARY KEY,
    Ad NVARCHAR(100) NOT NULL,
    AktifMi BIT DEFAULT 1
);
CREATE TABLE Kitaplar (
    KitapId INT IDENTITY(1,1) PRIMARY KEY,
    KitapAdi NVARCHAR(200) NOT NULL,
    ISBN NVARCHAR(20),
    YazarId INT NOT NULL,
    KategoriId INT NOT NULL,
    BasimYili INT,
    AktifMi BIT DEFAULT 1,
    FOREIGN KEY (YazarId) REFERENCES Yazarlar(YazarId),
    FOREIGN KEY (KategoriId) REFERENCES Kategoriler(KategoriId)
);
CREATE TABLE KitapKopyalari (
    KopyaId INT IDENTITY(1,1) PRIMARY KEY,
    KitapId INT NOT NULL,
    Barkod NVARCHAR(50) UNIQUE NOT NULL,
    RafBilgisi NVARCHAR(50),
    Durum NVARCHAR(20) DEFAULT 'MUSAIT', -- MUSAIT / ODUNC_VERILDI
    FOREIGN KEY (KitapId) REFERENCES Kitaplar(KitapId)
);
CREATE TABLE OduncIslemleri (
    OduncId INT IDENTITY(1,1) PRIMARY KEY,
    KullaniciId INT NOT NULL,
    KopyaId INT NOT NULL,
    OduncTarihi DATE NOT NULL,
    TeslimTarihi DATE NOT NULL,
    IadeTarihi DATE NULL,
    Durum NVARCHAR(20) DEFAULT 'AKTIF',
    FOREIGN KEY (KullaniciId) REFERENCES Kullanicilar(KullaniciId),
    FOREIGN KEY (KopyaId) REFERENCES KitapKopyalari(KopyaId)
);
CREATE TABLE Cezalar (
    CezaId INT IDENTITY(1,1) PRIMARY KEY,
    KullaniciId INT NOT NULL,
    OduncId INT NOT NULL,
    GecikmeGun INT NOT NULL,
    Tutar DECIMAL(10,2) NOT NULL,
    OlusturmaTarihi DATETIME DEFAULT GETDATE(),
    FOREIGN KEY (KullaniciId) REFERENCES Kullanicilar(KullaniciId),
    FOREIGN KEY (OduncId) REFERENCES OduncIslemleri(OduncId)
);
```

TRİGGER VE SP OLUŞTURMA

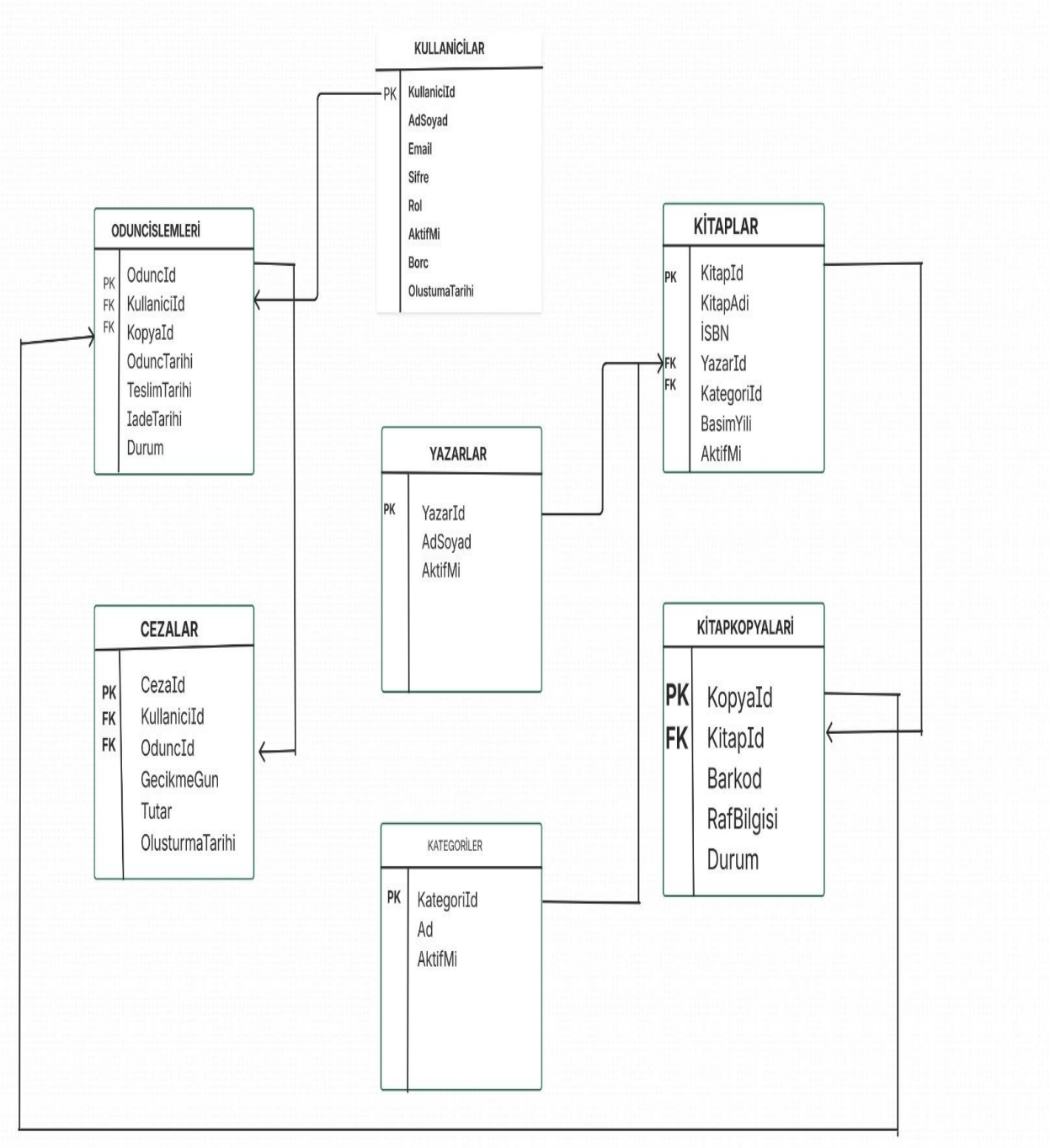
SQLQuery20.sql - bağlı değil* | SQLQuery2.sql - bağlı değil* | SQLQuery1.sql - bağlı değil* | **SQLQuery3.sql - bağlı değil*** | SQLQuery4.sql - bağlı değil*

```sql
CREATE PROCEDURE sp_GecikenKitaplariGetir
    @KullaniciId INT
AS
BEGIN
    SELECT
        o.OduncId,
        k.KitapAdi,
        o.OduncTarihi,
        o.TeslimTarihi
    FROM OduncIslemleri o
    JOIN KitapKopyalari kk ON kk.KopyaId = o.KopyaId
    JOIN Kitaplar k ON k.KitapId = kk.KitapId
    WHERE o.KullaniciId = @KullaniciId
      AND o.IadeTarihi IS NULL
      AND o.TeslimTarihi < GETDATE();
END;
```

SQLQuery20.sql - bağlı değil* | **SQLQuery2.sql - bağlı değil*** | SQLQuery1.sql - bağlı değil

```sql
CREATE TRIGGER trg_IadeCezaHesapla
ON OduncIslemleri
AFTER UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE
        @OduncId INT,
        @KullaniciId INT,
        @TeslimTarihi DATE,
        @IadeTarihi DATE,
        @GecikmeGun INT;

    SELECT
        @OduncId = i.OduncId,
        @KullaniciId = i.KullaniciId,
        @TeslimTarihi = i.TeslimTarihi,
        @IadeTarihi = i.IadeTarihi
    FROM inserted i;

    IF @IadeTarihi IS NOT NULL
    BEGIN
        SET @GecikmeGun = DATEDIFF(DAY, @TeslimTarihi, @IadeTarihi);

        IF @GecikmeGun > 0
        BEGIN
            INSERT INTO Cezalar (KullaniciId, OduncId, GecikmeGun, Tutar)
            VALUES (@KullaniciId, @OduncId, @GecikmeGun, @GecikmeGun * 2);

            UPDATE Kullanicilar
            SET Borc = Borc + (@GecikmeGun * 2)
            WHERE KullaniciId = @KullaniciId;
        END;

        UPDATE OduncIslemleri
        SET Durum = 'IADE_EDILDI'
        WHERE OduncId = @OduncId;

        UPDATE KitapKopyalari
        SET Durum = 'MUSAIT'
        WHERE KopyaId = (SELECT KopyaId FROM inserted);
    END
END;
```

ER-DİYAGRAMI



5. YAZILIM MİMARİSİ

Proje, modern yazılım geliştirmede kabul görmüş **MVC (Model-View-Controller)** mimari desenine sadık kalınarak tasarlanmıştır. Bu yapı, kodun okunabilirliğini arttırmış ve bakımını kolaylaştırmıştır.

5.1. Model Katmanı (Entity & Repository)

Veri tabanı nesnelerinin Java sınıfları olarak temsil edildiği (com.library.library.entity paketi) ve veri erişim işlerinin yapıldığı katmandır. JpaRepository arayüzü sayesinde standart CRUD (Create, Read, Update, Delete) işlemleri için kod yazılmasına gerek kalmamış, sadece findByAktifMiTrue gibi özel sorgular yazılmıştır.

5.2. View Katmanı (Arayüz)

Kullanıcının etkileşime girdiği HTML sayfalarıdır (src/main/resources/templates).

Thymeleaf motoru sayesinde, Java tarafından gönderilen Model verileri (örneğin kitap listesi, kullanıcı adı) HTML etiketleri içinde işlenmiştir.

5.3. Controller Katmanı

İstemciden gelen HTTP isteklerini (GET, POST) karşılayan ve ilgili servislere yönlendiren katmandır.

1.WebController: Sayfa yönlendirmeleri ve oturum kontrollerini yapar.

2.AuthController: Login/Register işlemlerini yönetir.

3.OduncController: API seviyesinde ödünç alma işlemlerini yönetir.

6. UYGULAMA DETAYLARI VE KOD İNCELEMESİ

Bu bölümde, projenin en karmaşık ve önemli iş mantıkları detaylandırılmıştır.

6.1. Ödünç Alma ve "Hızlandırılmış Test Modu"

Akademik sunumlarda bir kitabın iadesini günlerce beklemek mümkün olmadığından, sistemde özel bir test algoritması geliştirilmiştir. OduncServiceImpl.java sınıfındaki oduncAl metodu incelendiğinde kod bloğu, kitabın alındığı andan itibaren 1 dakika içinde iade edilmesini bekler. Bu, sistemin ceza mekanizmasının canlı olarak test edilebilmesini sağlar.

6.2. Dinamik Ceza Hesaplama Mantığı

Ceza hesaplaması iki aşamalı çalışır:

1. **Kesinleşmiş Ceza (Veri Tabanı Kaydı):** Kitap fiziksel olarak iade edildiğinde teslimEt metodu çalışır. Teslim tarihi ile beklenen tarih arasındaki fark dakika cinsinden hesaplanır (ChronoUnit.MINUTES). Dakika başına **5 TL** ceza kesilir.
2. **Canlı Ceza (Arayüz Gösterimi):** Kitap henüz iade edilmemiş olsa bile, kullanıcı panelinde (WebController -> anaSayfa) o anki zamana göre biriken potansiyel borç hesaplanır ve kullanıcıya "Güncel Borç" olarak gösterilir. Bu, kullanıcının borcunun arttığını anlık görmesini sağlar.

```
    // 1 dakikalık test süresi kurulumu
    odunc.setOduncTarihi(LocalDateTime.now());
    odunc.setBeklenenIadeTarihi(LocalDateTime.now().plusMinutes(1));
    return oduncRepository.save(odunc);

    @Transactional
    public Odunc teslimEt(Integer oduncId) {
    Odunc odunc = oduncRepository.findById(oduncId)
            .orElseThrow(() -> new RuntimeException("Kayıt bulunamadı"));

    LocalDateTime simdi = LocalDateTime.now();

    odunc.setTeslimTarihi(simdi);
    odunc.setIadeTarihi(simdi);
    odunc.setDurum("TESLIM_EDILDI");

    if (simdi.isAfter(odunc.getBeklenenIadeTarihi())) {
        long gecikenDakika = ChronoUnit.MINUTES.between(odunc.getBeklenenIadeTarihi(), simdi);

        odunc.setCezaMiktari((double) gecikenDakika * 5.0);
    } else {
        odunc.setCezaMiktari(0.0);
    }

    return oduncRepository.save(odunc);
}
```

```
    // ✅ GÜNCEL BORÇ TOPLAMA MANTIĞI: Teslim edilmeyen her şeyi kapsar
    for (Odunc o : tumOduncler) {
        // 1. Veritabanındaki hali hazırda kayıtlı cezayı ekle
        toplamBorc += (o.getCezaMiktari() != null ? o.getCezaMiktari() : 0.0);

        // 2. CANLI HESAPLAMA: Kitap henüz gerçekten teslim edilmediyse (teslimTarihi null ise)
        if (o.getTeslimTarihi() == null && o.getBeklenenIadeTarihi() != null) {
            if (simdi.isAfter(o.getBeklenenIadeTarihi())) {
                long dk = ChronoUnit.MINUTES.between(o.getBeklenenIadeTarihi(), simdi);
                toplamBorc += (dk * 5.0);
            }
        }
    }

    long uzerimdekilerCount = tumOduncler.stream()
        .filter(o -> o.getTeslimTarihi() == null)
        .count();

    model.addAttribute("uzerimdekiler", uzerimdekilerCount);
    model.addAttribute("okunanlar", tumOduncler.size());
    model.addAttribute("toplamBorc", toplamBorc);
    return "uye-paneli";
}
```

6.3. Kullanıcı Yetkilendirme ve Oturum Yönetimi

Spring Security'nin karmaşıklığından kaçınmak ve temel oturum mantığını göstermek amacıyla HttpSession kullanılmıştır.

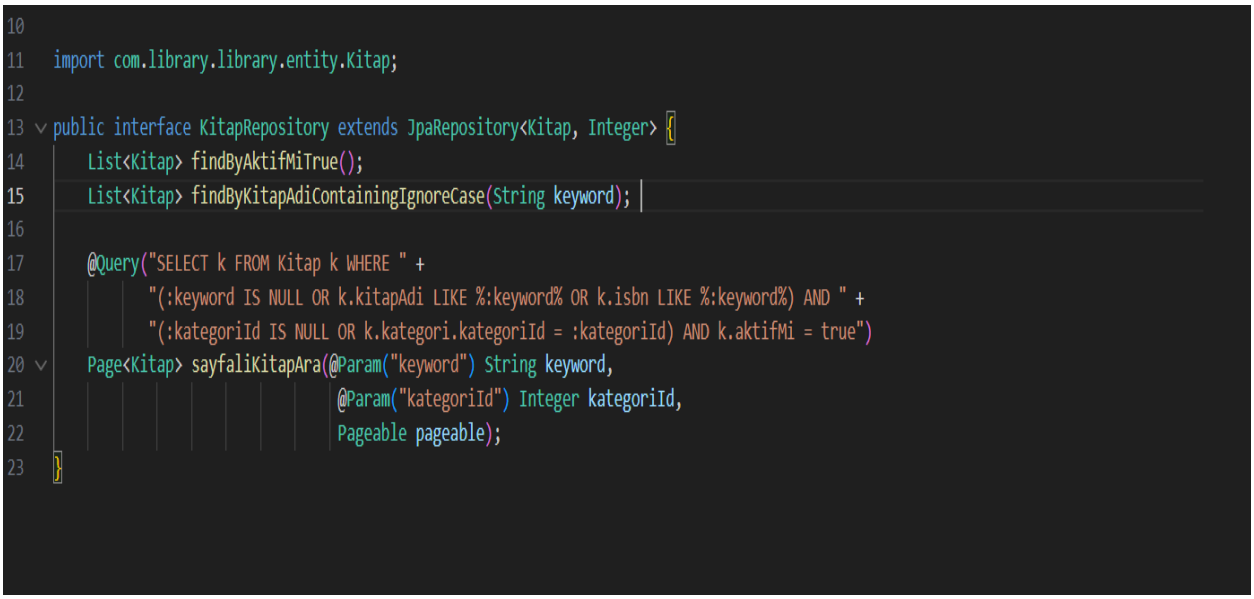
Kullanıcı giriş yaptığında: session.setAttribute("oturum", kullanici) ile nesne oturuma atılır.

Sayfa geçişlerinde: yetkiliMi(session) metodu ile kullanıcının ADMIN veya PERSONEL olup olmadığı kontrol edilir. Yetkisiz erişimler Login sayfasına yönlendirilir.

```
@PostMapping("/login")
public String girisYap(@RequestParam String username,
                       @RequestParam String password,
                       jakarta.servlet.http.HttpSession session,
                       Model model) {

    Kullanici kullanici = kullaniciService.girisKontrol(username, password);

    if (kullanici != null) {
        session.setAttribute("oturum", kullanici);
        return "redirect:/";
    } else {
        model.addAttribute("hata", "E-posta veya şifre hatalı!");
        return "login";
    }
}
```

6.4. Arama ve Filtreleme

Kitap arama modülü (KitapRepository), kullanıcı deneyimini artırmak için "ContainingIgnoreCase" (Büyük/küçük harf duyarsız, içerir) mantığıyla çalışır. Hem kitap isminde hem de ISBN numarasında aynı anda arama yapabilir.

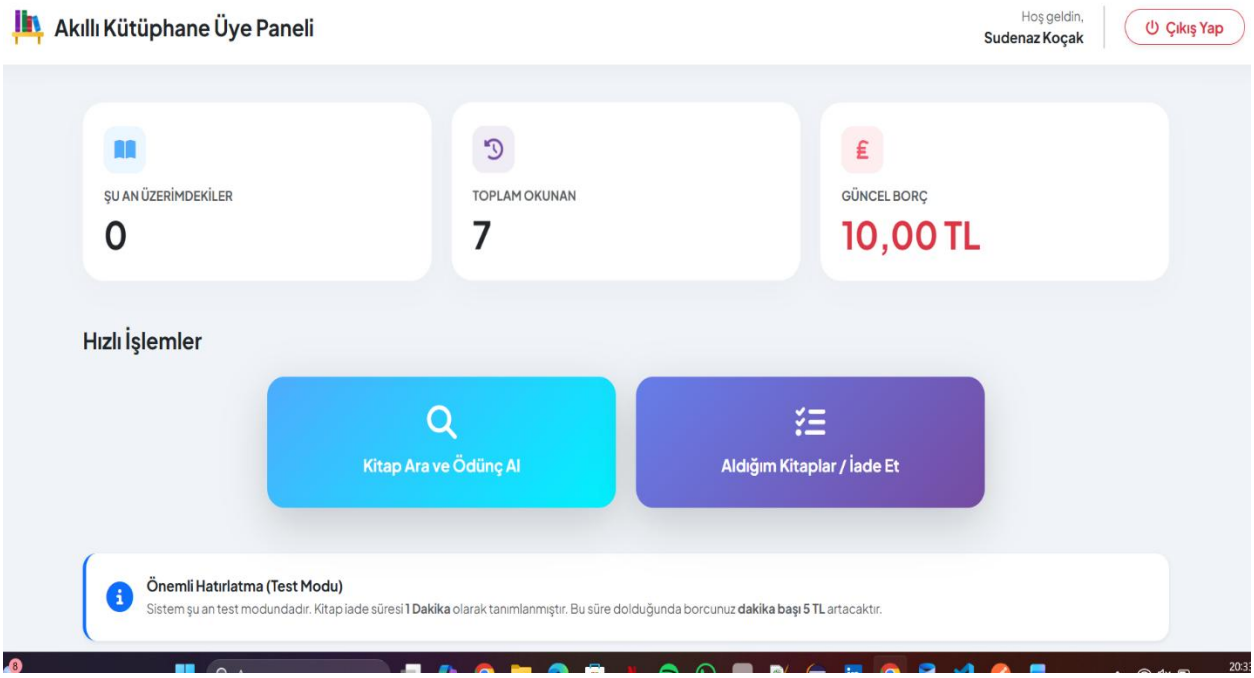


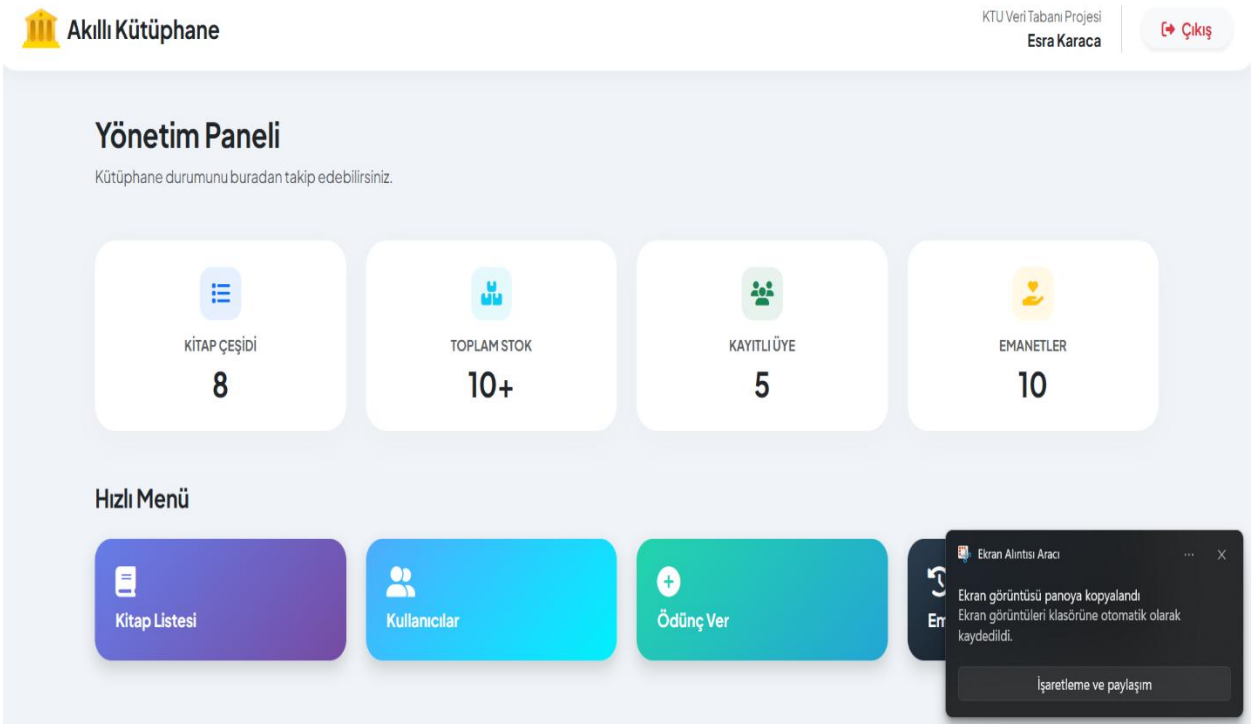
```
10
11  import com.library.library.entity.Kitap;
12
13  public interface KitapRepository extends JpaRepository<Kitap, Integer> {
14      List<Kitap> findByAktifMiTrue();
15      List<Kitap> findByKitapAdiContainingIgnoreCase(String keyword);
16
17      @Query("SELECT k FROM Kitap k WHERE " +
18              "(:keyword IS NULL OR k.kitapAdi LIKE %:keyword% OR k.isbn LIKE %:keyword%) AND " +
19              "(:kategoriId IS NULL OR k.kategori.kategoriId = :kategoriId) AND k.aktifMi = true")
20      Page<Kitap> sayfaliKitapAra(@Param("keyword") String keyword,
21                                  @Param("kategoriId") Integer kategoriId,
22                                  Pageable pageable);
23  }
```

7. KULLANICI ARAYÜZÜ VE DENEYİMİ (UI/UX)

Kullanıcı arayüzü tasarlanırken "Temiz Tasarım" prensibi benimsenmiştir.

- Dashboard (Ana Sayfa): Kullanıcı rolüne göre şekillenir.
    - *Yönetici:* Toplam Kitap, Toplam Üye, Aktif Emanet sayılarını içeren istatistik kartları görür.
    - *Üye:* Sadece kendi üzerindeki kitap sayısını ve borcunu görür.
- Renk Kodlamaları:
    - emanetler.html sayfasında, süresi geçen kitapların satırları otomatik olarak kırmızı (row-delayed CSS sınıfı) ile, zamanı gelmemiş olanlar ise standart renkte gösterilir. Bu, personelin problemli kayıtları bir bakışta görmesini sağlar.
- Geri Bildirimler: İşlem başarıyla gerçekleştiğinde veya hata alındığında (Örn: "E-posta zaten kullanımda"), kullanıcıya Bootstrap Alert bileşenleri ile görsel geri bildirim verilir.


Akıllı Kütüphane Üye Paneli
Hoş geldin,
Sudenaz Koçak
Çıkış Yap
ŞU AN ÜZERİMDEKİLER
0
TOPLAM OKUNAN
7
GÜNCEL BORÇ
10,00 TL
Hızlı İşlemler
Kitap Ara ve Ödünç Al
Aldığım Kitaplar / İade Et
Önemli Hatırlatma (Test Modu)
Sistem şu an test modundadır. Kitap iade süresi 1 Dakika olarak tanımlanmıştır. Bu süre dolduğunda borcunuz dakika başı 5 TL artacaktır.


Akıllı Kütüphane
KTÜ Veri Tabanı Projesi
Esra Karaca
Çıkış
Yönetim Paneli
Kütüphane durumunu buradan takip edebilirsiniz.
KİTAP ÇEŞİDİ
8
TOPLAM STOK
10+
KAYITLI ÜYE
5
EMANETLER
10
Hızlı Menü
Kitap Listesi
Kullanıcılar
Ödünç Ver
Ekran Alıntısı Aracı
Ekran görüntüsü panoya kopyalandı
Ekran görüntüleri klasörüne otomatik olarak kaydedildi.
İşaretleme ve paylaşım

8. SENARYOLAR VE TEST SONUÇLARI

Sistemin başarısını ölçmek için aşağıdaki senaryolar test edilmiş ve doğrulanmıştır:

Senaryo 1: Yeni Üye Kaydı

- İşlem: Kayıt ol formuna Ad, Soyad, E-posta ve Şifre girilir.
- Beklenen: Veri tabanına Rol: UYE, Aktif: True olarak kaydedilmesi.
- Sonuç: Başarılı. Varsayılan değerler KullaniciServiceImpl tarafından doğru atandı.

Senaryo 2: Kitap Ödünç Alma ve Ceza

- İşlem: Yönetici, "Sefiller" kitabını "Ahmet Yılmaz" kullanıcısına verir.
- Zaman Akışı: 1 dakika beklenir (Sistem saati ilerler).
- Gözlem: Emanetler sayfasında ilgili satır kırmızıya döner.
- İade: Yönetici "İade Al" butonuna basar.
- Sonuç: Sistem, geçen süre (örneğin 2 dakika) * 5 TL = 10 TL cezayı hesaplar ve Odunc tablosuna yazar. Kitap durumu "TESLIM_EDILDI" olur.

Senaryo 3: Yetkisiz Erişim

- İşlem: Normal bir üye, tarayıcı adres çubuğuna /kullanici-sil/1 yazar.
- Beklenen: Sistemin işlemi reddetmesi ve ana sayfaya atması.
- Sonuç: Başarılı. WebController içindeki yetki kontrolü (Interceptor mantığı) devreye girdi.

9. SONUÇ VE ÖNERİLER

Bu proje çalışması ile; veri tabanı tasarımı, backend servis mimarisi ve frontend entegrasyonu konularında uçtan uca bir çözüm üretilmiştir. MS SQL Server ile Spring Boot'un entegrasyonu başarılı bir şekilde sağlanmış, JPA teknolojisi etkin kullanılmıştır.

Özellikle kurgulanan **"1 Dakikalık Test Modu"**, projenin sunumu sırasında ceza mantığının çalıştığının kanıtlanması açısından kritik bir yeniliktir. Sistem, temel fonksiyonları eksiksiz yerine getirmektedir.

Gelecek Çalışmalar İçin Öneriler:

1. **Spring Security:** Mevcut manuel oturum yönetimi yerine Spring Security entegre edilerek daha kapsamlı (Remember Me, OAuth2 vb.) güvenlik sağlanabilir.
2. **Raporlama:** PDF veya Excel formatında aylık kitap hareket dökümü özelliği eklenebilir.
3. **E-Posta Bildirimi:** İade süresi yaklaşan kullanıcılara otomatik e-posta gönderen bir "Scheduler" (Zamanlayıcı) eklenebilir.

10. KAYNAKÇA

1. Spring Boot Documentation. (2025). *Spring.io*.
2. Thymeleaf Documentation. (2025). *Thymeleaf.org*.
3. Bootstrap 5.3 Documentation. *Getbootstrap.com*.
4. Proje Kaynak Kodları (Esra Karaca GitHub Deposu).