AKILLI KÜTÜPHANE YÖNETİM SİSTEMİ PROJESİ RAPORU

Ders: Veri Tabanı Ve Yönetimi , **Dönem:** 2025-2026 Güz Dönemi

Öğrenci Ad,Soyad: Furkan Aloğlu / **Öğrenci No:** 445857

1. YÖNETİCİ ÖZETİ (ABSTRACT)

Bu proje kapsamında, geleneksel kütüphane süreçlerini dijitalleştiren, verimliliği artıran ve insan hatasını minimize eden web tabanlı bir "Akıllı Kütüphane Yönetim Sistemi" geliştirilmiştir. Proje, modern yazılım geliştirme prensipleri olan **Katmanlı Mimari (Layered Architecture)**, **RESTful Servisler** ve **İlişkisel Veri Tabanı Yönetimi (RDBMS)** temellerine dayanmaktadır. Sistem; kitap takibi, dinamik stok yönetimi, kullanıcı yetkilendirme (RBAC) ve otomatik ceza hesaplama mekanizmalarını içermektedir. Geliştirilen çözümde backend için Java Spring Boot, veri kalıcılığı için PostgreSQL ve kullanıcı arayüzü için platform bağımsız JavaScript teknolojileri tercih edilmiştir.

2. GİRİŞ VE PROJENİN AMACI

Günümüzde bilgiye erişim hızı, kütüphanelerin fiziksel yönetiminden dijital yönetimine geçişi zorunlu kılmıştır. Bu projenin temel amacı; kitap ödünç alma/iade süreçlerindeki manuel takibi ortadan kaldırmak ve veri tutarlılığını (Data Consistency) garanti altına almaktır.

Projenin teknik hedefleri şunlardır:

- **Veri Bütünlüğü:** ACID (Atomicity, Consistency, Isolation, Durability) prensiplerine uygun veritabanı tasarımı.
- **Güvenlik:** Yetkisiz erişimlerin engellenmesi ve hassas verilerin korunması.
- **Otomasyon:** Gecikmiş kitapların tespiti ve ceza süreçlerinin veritabanı seviyesinde (Trigger/Procedure) otomatikleştirilmesi.
- **Ölçeklenebilirlik:** İleride mobil uygulama veya farklı istemcilerin de sisteme entegre olabilmesi için API tabanlı yapı.

3. SİSTEM MİMARİSİ VE KULLANILAN TEKNOLOJİLER

Proje, **Monolitik** bir yapıda kurgulanmış olsa da, **Servis Yönelimli Mimari (SOA)** prensiplerine sadık kalınarak Backend ve Frontend tamamen birbirinden izole edilmiştir (Decoupled Architecture).

3.1. Backend Teknolojileri (Sunucu Tarafı)

- **Dil ve Çerçeve:** Java 17 ve Spring Boot 3.x kullanılmıştır. Spring Boot'un sağladığı "Dependency Injection" (Bağımlılık Enjeksiyonu) sayesinde bileşenler arası sıkı bağlılık (Tight Coupling) önlenmiştir.
- **ORM (Object Relational Mapping):** Hibernate ve JPA kullanılarak, SQL sorgularının nesne yönelimli programlama mantığıyla yönetilmesi sağlanmıştır. Bu sayede veritabanı bağımsızlığı (Database Agnostic) elde edilmiştir.
- **Güvenlik Katmanı:** Spring Security ile entegre **JWT (JSON Web Token)** yapısı kurulmuştur. Sistem "Stateless" (Durumsuz) çalışmaktadır; yani sunucu, kullanıcı oturumunu bellekte tutmaz, her istekte token doğrulaması yapar.

3.2. Veritabanı Teknolojileri

- **DBMS:** PostgreSQL tercih edilmiştir. Tercih sebebi; gelişmiş prosedürel dil desteği (PL/pgSQL) ve yüksek işlem hacmindeki kararlılığıdır.
- **Migration Yönetimi:** Veritabanı şema değişikliklerinin versiyonlanması ve takibi için **Flyway** (veya proje yapına göre manuel script yönetimi) prensipleri esas alınmıştır.

3.3. Frontend Teknolojileri (İstemci Tarafı)

- Projede herhangi bir hazır şablon motoru (Thymeleaf vb.) yerine, modern web standartlarına uygun **Saf (Vanilla) JavaScript, HTML5 ve CSS3** kullanılmıştır.
- Backend ile iletişim **Fetch API** üzerinden asenkron (AJAX) olarak sağlanmıştır. Bu tercih, sayfa yenilenmeden dinamik içerik yüklenmesine olanak tanımış ve kullanıcı deneyimini (UX) artırmıştır.

4. VERİTABANI TASARIMI VE İŞ MANTIĞI

Veritabanı tasarımı 3. Normal Form (3NF) kurallarına uygun olarak gerçekleştirilmiştir. Veri tekrarı önlenmiş ve ilişkisel bütünlük sağlanmıştır.

4.1. Varlık İlişki Yapısı (ER Diagram Özeti)

Sistemin omurgasını oluşturan temel tablolar ve ilişkileri şöyledir:

1. **Users (Kullanıcılar):** Sistemdeki aktörleri tutar. `Roles` tablosu ile `Many-to-Many` veya Enum yapısı üzerinden yetkilendirme yapılır.
2. **Books (Kitaplar):** Kitap detaylarını ve anlık stok bilgisini tutar.
3. **Borrowings (Ödünç İşlemleri):** Hangi kullanıcının hangi kitabı ne zaman aldığı bilgisini tutar. Bu tablo `User` ve `Book` tabloları arasında bir köprü görevi görür.
4. **Penalties (Cezalar):** Gecikme cezalarını saklar. Doğrudan kullanıcıya değil, `Borrowing` işlemine bağlıdır (`1-to-1`). Bu sayede cezanın hangi işlemden kaynaklandığı geriye dönük izlenebilir.

4.2. İleri Seviye Veritabanı İşlemleri (Stored Procedure & Triggers)

Projenin en kritik teknik özelliklerinden biri, iş mantığının bir kısmının veritabanına devredilmesidir:

- **Otomatik Ceza Mekanizması (Trigger):** `trg_calculate_penalty` isimli tetikleyici, `borrowings` tablosunda bir güncelleme (iade işlemi) olduğunda devreye girer. Eğer iade tarihi, son teslim tarihini geçmişse, sistem arka planda gecikme gün sayısını hesaplar, belirlenen günlük ceza tutarı ile çarpar ve `penalties` tablosuna otomatik olarak kayıt ekler. Bu işlem, uygulama katmanındaki hatalardan etkilenmez.
- **Manuel Müdahale ve Yönetim (Stored Procedure):** Otomatik sistemlerin yetersiz kaldığı durumlar veya idari af/değişiklik gerektiren senaryolar için `calculate_penalty` isimli Saklı Yordam (Stored Procedure) geliştirilmiştir. Bu prosedür, yönetici (Admin) onayı ile manuel olarak tetiklenebilir ve veritabanı seviyesinde güvenli bir şekilde hesaplama/güncelleme yapar.

5. API UÇ NOKTALARI VE İŞLEVLERİ (REST API)

Sistem, HTTP protokolü üzerinden JSON formatında veri alışverişi yapar. Temel Controller yapıları şöyledir:

- **AuthController:** `/api/auth/register` ve `/api/auth/login` uç noktaları ile kullanıcı kaydı ve güvenli token üretimi sağlar.
- **BookController:** Kitapların CRUD (Ekleme, Okuma, Güncelleme, Silme) işlemlerini yönetir. Kullanıcılar için sadece `GET` (listeleme) izni varken, Adminler `POST`, `PUT`, `DELETE` işlemlerini yapabilir.

- **BorrowingController:**
    - `POST /borrow`: Stok kontrolü yapar, kullanıcı limitini (örn: aynı anda en fazla 3 kitap) denetler ve işlemi onaylar. İşlem anında kitap stoğu "Transactional" (Atomik) olarak düşülür.
    - `PUT /return/{id}`: Kitap iadesini alır ve stoğu tekrar artırır.

6. KARŞILAŞILAN ZORLUKLAR VE ÇÖZÜMLER

Proje geliştirme sürecinde karşılaşılan teknik zorluklar ve üretilen çözümler:

1. **N+1 Sorgu Sorunu:** Kitapları listelerken yazarları çekmek için oluşan gereksiz sorgu yükü, JPA'nın `FetchType.LAZY` ve `EntityGraph` özellikleri kullanılarak optimize edilmiştir.
2. **Tarih/Saat Uyumluluğu:** Java `LocalDateTime` ile PostgreSQL `TIMESTAMP` veri tipleri arasındaki uyumsuzluklar, doğru konfigürasyonlar ve `TimeZone` ayarları ile giderilmiştir.
3. **CORS (Cross-Origin Resource Sharing):** Frontend ve Backend'in farklı portlarda çalışması sonucu oluşan güvenlik engeli, Spring Security konfigürasyonunda `CorsConfigurationSource` özelleştirilerek aşılmıştır.

7. SONUÇ VE GELECEK ÇALIŞMALAR

Bu proje ile, temel kütüphane fonksiyonlarını eksiksiz yerine getiren, güvenli ve genişletilebilir bir sistem ortaya konulmuştur. Öğrenci olarak; **Backend-Frontend entegrasyonu, veritabanı normalizasyonu ve API güvenliği** konularında derinlemesine yetkinlik kazanılmıştır.

Gelecek sürümler için planlanan özellikler:

- Mikroservis mimarisine geçiş (Bildirim servisi ve Kullanıcı servisinin ayrılması).
- Redis kullanılarak sık sorgulanan kitapların önbelleğe (Cache) alınması.
- Google Books API entegrasyonunun genişletilerek ISBN üzerinden otomatik kitap verisi çekilmesi.