Library App Full Stack

Full Stack Kütüphane Uygulaması — Teknik Plan

> Hedef: Üyelerin kitap arayıp rezerve/ödünç alabildiği, admin'in envanteri ve kullanıcıları yönettiği modern bir kütüphane sistemi. Üretime hazır, testli, güvenli.

▼ Draft Plan (click)

1) Mimari & Teknolojiler

Varsayılan Stack

- **Frontend:** React + Vite, TypeScript, React Router, TanStack Query, Zustand (veya Redux Toolkit), TailwindCSS
- **Backend:** Spring Boot 3 (Java 21), Spring Web, Spring Data, Spring Security (JWT), Validation, Lombok, MapStruct
- **Veritabanı:** MongoDB Atlas (M10+), alternatif: PostgreSQL 16
- **İletişim:** REST + JSON (opsiyonel: WebSocket ile canlı rezervasyon/ödünç akışı)
- **Diğer:** Docker Compose, Testcontainers, JUnit 5, WireMock, GitHub Actions CI, Flyway (eğer PostgreSQL), OpenAPI/Swagger

Alternatif Stack (Node.js)

- **Backend:** Node.js + Express / NestJS, TypeScript, Zod, Prisma/Mongoose, JWT Auth

🧱 1) Mimari & Teknolojiler — Revize

Frontend

✅ **Kalacak:** React + Vite + React Router + TanStack Query + Zustand + TailwindCSS

💬 **TypeScript hakkında:**

Evet, "sadece veri tipleri ve arayüzleri (interface) tanımlıyoruz" düzeyinde bir farktan ibaret. React'te props ve state tanımlarında hataları önlediği için uzun vadede çok faydalı.

İstersen TypeScript dosyalarıyla başlayalım ama yazarken sana "JS eşdeğeri" karşılığını da açıklayayım. Böylece zorlanmadan öğrenmiş olursun.

> Özet: TypeScript'le devam edelim ama açıklamalı gidelim.

Backend

✅ **Kalacak:** Spring Boot 3 (Java 21), Spring Web, Spring Data, Spring Security (JWT), Validation, Lombok

🧩 **MapStruct nedir?**

MapStruct, entity ↔ DTO dönüşümlerini otomatik yapan bir "mapper" kütüphanesi.

Yani:

```
Book book = mapper.toEntity(bookDto);
BookDto dto = mapper.toDto(book);
```

şeklinde kullanılır, setter'ları elle yazmana gerek kalmaz.

Alternatif olarak `ModelMapper` de kullanılabilir ama MapStruct compile-time'da çalıştığı için çok daha hızlıdır.

Veritabanı

📌 Kesin olarak: **MongoDB Atlas** (örneğin M10 cluster)

Yerel geliştirme için Docker üzerinde `mongo` container, deploy'da Atlas bağlantısı kullanacağız.

> Atlas URI .env dosyasında olacak:
> `MONGODB_URI=mongodb+srv://user:pass@cluster.mongodb.net/library`

API iletişimi

Sadece **REST + JSON** (kesin, başka protokol yok).

- `axios` ile frontend istekleri
- Backend: `@RestController` endpoint'ler
- JSON parse işlemleri otomatik (`@RequestBody`, `@ResponseBody`)

DevOps / Çalışma Ortamı

🚀 Docker Compose kullanılacak

- `mongo` (DB)
- `server` (Spring Boot)
- `client` (React)

  hepsi tek `docker-compose.yml` içinde olacak.

---

2) Özellikler (MVP → Genişleme)

MVP

- Üye kayıt / giriş (JWT)
- Kitap arama (başlık, yazar, ISBN, etiket)
- Kitap detay sayfası
- Ödünç alma (loan) ve iade
- Rezervasyon (hold) — sırayla çağırma mantığı
- Admin paneli: kitap/örnek kopya (copy) ekleme/silme/düzenleme, kullanıcı rolleri

Genişleme

- Ceza/ücret hesaplama (gecikme)
- Bildirimler (e-posta/Telegram) — rezervasyon sıra geldiğinde
- Raporlama (en çok okunanlar, aktif kullanıcılar, gecikmeler)
- Barkod/QR ile hızlı teslim-iade
- Çoklu şube ve envanter transferi

---

3) Veri Modeli

MongoDB (Önerilen)

- `users` `{ _id, email, passwordHash, role: 'ADMIN'|'MEMBER', fullName, createdAt }`
- `authors` `{ _id, name, bio?, birthYear?, deathYear? }`
- `books` `{ _id, isbn, title, authorIds: ObjectId[], tags: string[], description?, coverUrl?, createdAt }`
- `copies` `{ _id, bookId, barcode, status: 'AVAILABLE'|'LOANED'|'HOLD', location: 'MAIN'|string }`
- `loans` `{ _id, copyId, userId, loanedAt, dueAt, returnedAt? }`
- `holds` `{ _id, bookId, userId, queuedAt, expiresAt?, status: 'QUEUED'|'NOTIFIED'|'EXPIRED'|'FULFILLED' }`
- **Not:** Kitap/Author normalizasyonu + `copies` ayrı koleksiyon ile stok yönetimi netleşir. Raporlar için `loanEvents` event-sourcing de eklenebilir.

PostgreSQL (Alternatif ER)

- `users(id, email, password_hash, role, full_name, created_at)`
- `authors(id, name, bio, birth_year, death_year)`
- `books(id, isbn, title, description, cover_url, created_at)`
- `book_authors(book_id, author_id)` (N:N)
- `copies(id, book_id, barcode, status, location)`
- `loans(id, copy_id, user_id, loaned_at, due_at, returned_at)`
- `holds(id, book_id, user_id, queued_at, expires_at, status)`

İndeksler

- `books: { isbn: 1 } (unique), { title: 'text', tags: 1 }, { authorIds: 1 }`
- `copies: { bookId: 1, status: 1 }`
- `loans: { userId: 1, dueAt: 1 }, { copyId: 1, returnedAt: 1 }`
- `holds: { bookId: 1, status: 1, queuedAt: 1 }`

---

4) İş Kuralları

- **Ödünç:** Sadece `AVAILABLE` copy ödünç verilir → `status=LOANED` + `dueAt=loanedAt + policyDays` (örn. 14 gün)
- **İade:** `returnedAt=now`, copy `AVAILABLE`
- **Rezervasyon:** Eğer tüm kopyalar LOANED ise kullanıcı `holds` kuyruğuna girer. Bir kopya iade olur olmaz `holds` en üstteki kullanıcı `NOTIFIED`, 48 saat içinde almazsa `EXPIRED` → sıradaki çağrılır.
- **Ceza:** `returnedAt > dueAt` gün farkı * cezaBirimi; admin manuel affedebilir.
- **Roller:** `ADMIN` her şeyi yönetir; `MEMBER` kendi loans/holds + arama/ödünç.

---

5) REST API Tasarımı (Özet)

Auth

- `POST /api/auth/register` → {email, password, fullName}
- `POST /api/auth/login` → {email, password} → {accessToken}

Books

- `GET /api/books?query=&tags=&author=` → arama (paginated)
- `GET /api/books/{id}` → detay (+ mevcut kopya sayıları)
- `POST /api/books` (ADMIN)
- `PUT /api/books/{id}` (ADMIN)
- `DELETE /api/books/{id}` (ADMIN)

Copies

- `POST /api/books/{id}/copies` (ADMIN) — adetli ekleme destekli

- `PATCH /api/copies/{id}` status/location (ADMIN)

Loans

- `POST /api/loans` {copyId} (MEMBER) — uygun değilse 409
- `POST /api/loans/{id}/return` (MEMBER/ADMIN)
- `GET /api/loans/me` (MEMBER) — aktif/geçmiş

Holds

- `POST /api/holds` {bookId} (MEMBER)
- `GET /api/holds/me` (MEMBER)
- `POST /api/holds/{id}/cancel` (MEMBER)
- `POST /api/holds/{id}/fulfill` (ADMIN) — bildirim sonrası teslimat anında

Admin

- `GET /api/admin/dashboard` — metrikler
- `GET /api/admin/users` — listele/ara

Hata Kodları

- 400 (validation), 401 (auth), 403 (authorization), 404, 409 (business conflict), 422 (domain)

---

6) UI Akışı (React)

- **Public:** Login/Register, Arama/List, Kitap Detay
- **Member:** Benim Ödünçlerim, Rezervasyonlarım, Bildirimler
- **Admin:** Kitap Yönetimi, Kopya Yönetimi, Kullanıcı Yönetimi, Raporlar

Bileşenler

- `BookCard`, `BookFilters`, `BookList`, `BookDetail`
- `LoanButton` (duruma göre: available/hold/return)
- Admin formları: `BookForm`, `CopyBatchForm`

---

7) Proje Yapısı

Frontend

```
client/
  src/
    api/ (fetch clients, axios + interceptors)
    pages/ (Books, BookDetail, Loans, Holds, Admin/*)
    components/
    store/ (Zustand or Redux)
    hooks/ (useAuth, useBooksQuery, etc.)
    routing/
  index.tsx
  vite.config.ts
```

Backend (Spring Boot)

```
server/
  src/main/java/com/acme/library/
    config/
    auth/ (JwtService, filters, SecurityConfig)
    books/ (Book, BookRepository, BookService, BookController)
    copies/
    loans/
    holds/
    users/
    common/ (exceptions, dto, mappers)
  src/main/resources/
    application.yml
```

---

8) Örnek DTO & Endpoint (Spring Boot, kısaltılmış)

```
// Book.java (Mongo)
@Document("books")
@Data @Builder
```

```
public class Book { @Id String id; String isbn; String title; List<String> authorIds; List<String> tags; String description; String coverUrl; Instant createdAt; }
```

```
// BookController.java (özet)
@RestController @RequestMapping("/api/books")
@RequiredArgsConstructor
public class BookController {
  private final BookService service;
  @GetMapping
  public Page<BookDto> search(@RequestParam Optional<String> query, Pageable pageable){
    return service.search(query.orElse(""), pageable);
  }
  @GetMapping("/{id}")
  public BookDetailDto get(@PathVariable String id){ return service.get(id); }
  @PostMapping @PreAuthorize("hasRole('ADMIN')")
  public BookDto create(@Valid @RequestBody UpsertBookDto dto){ return service.create(dto); }
}
```

9) Test Stratejisi

- **Backend:** JUnit + Spring Boot Test, Testcontainers (MongoDB), WireMock (harici servis varsa)
- **Frontend:** Vitest + React Testing Library; kritik akışlar (login, search, loan)
- **E2E:** Playwright (opsiyonel)

10) Güvenlik

- JWT Access Token (15–30 dk) + Refresh Token (7–14 gün)
- Parola: BCrypt(12)
- Rate limit (reverse proxy/Nginx veya Spring Bucket4j)

- CORS: `client` origin
- Giriş denemesi limiti (Redis/DB sayacı)

---

11) DevOps

- Dockerfile (client & server), `docker-compose.yml` (server + mongo + mongo-express)
- GitHub Actions: build, test, docker push
- Deployment: Fly.io/Render/VM; ortam değişkenleri (DB_URI, JWT_SECRET ..)

---

12) Sprint Planı (7 Günlük Hızlı Çıkış)

Gün 1: Repo'lar, temel iskelet (client/server), Auth register/login (backend), Mongo bağlantısı.

Gün 2: Books CRUD (ADMIN), arama endpoint'i, temel indeksler.

Gün 3: Copies modeli + kopya ekleme/istatüler, frontend list & detay sayfaları.

Gün 4: Loans akışı (ödünç/iade) + dueAt politikası, "Benim Ödünçlerim".

Gün 5: Holds kuyruğu + notify süreci (sistem job taslağı), "Benim Rezervasyonlarım".

Gün 6: Admin paneli formları, doğrulamalar, hata yönetimi, yetkiler.

Gün 7: Testler, Docker Compose, küçük raporlar, seed verisi, hazır demo.

---

13) Tohum (Seed) Verileri

- 100 yazar, 300 kitap, her kitap için 1–3 kopya
- 1 admin, 5 demo üye

---

14) Açık İşler / Notlar

- Rezervasyon bildirimi için cron/scheduler (Spring Scheduling) ve e-posta sağlayıcısı

- Event loglama ve min. audit trail
- Çoklu şube desteği için `location` genişletme

---

15) Komutlar (Hızlı Başlangıç)

Backend

```
sdk use java 21 # ya da Temurin 21
spring init --dependencies=web,data-mongodb,validation,security,lombok --b
uild=gradle library-api
```

Frontend

```
npm create vite@latest library-client -- --template react-ts
cd library-client && npm i @tanstack/react-query axios zustand react-router-d
om tailwindcss postcss autoprefixer && npx tailwindcss init -p
```

> Sonraki adım: Auth + Books modüllerini birlikte ayağa kaldıralım (Gün 1–2).
> İstersen Node/Nest alternatifi için eşdeğer plan da hazır.

silinebilir area:

🧩 Yöntem 1 (en kolay ve garantili): Manuel indirme

1. Tarayıcıdan şu adrese git:

   👉 https://start.spring.io

2. Ayarları şöyle yap:
   - **Project:** Gradle - Groovy
   - **Language:** Java
   - **Spring Boot:** 3.3.x
   - **Group:** `com.library`

- **Artifact:** `library-api`
- **Name:** `library-api`
- **Java:** 21
- **Dependencies:**
  - Spring Web
  - Spring Data MongoDB
  - Spring Security
  - Validation
  - Lombok

3. “Generate” butonuna tıkla → `.zip` dosyası inecek.
4. İnen `library-api.zip` dosyasını `C:\Users\banbu\Desktop\FullStackPlayground\library-app\server` klasörüne taşı.
5. Üzerine **sağ tık → “Buraya ayıkla”** (veya 7zip/WinRAR ile çıkar).