Nama : Muh. Nur Afrizal
NIM : 222212738
Kelas / No : 3SI1 / 22

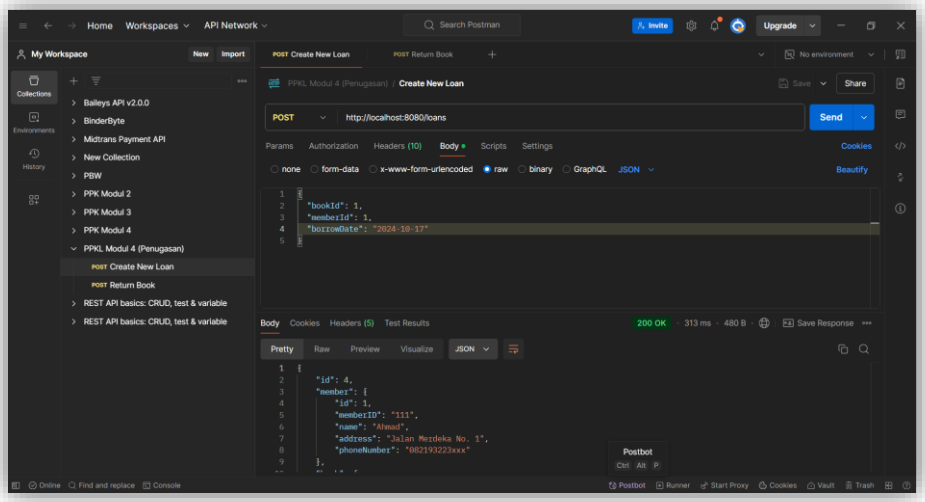
# LAPORAN PRAKTIKUM PEMROGRAMAN PLATFORM KHUSUS
# MODUL 4
# RESTful API

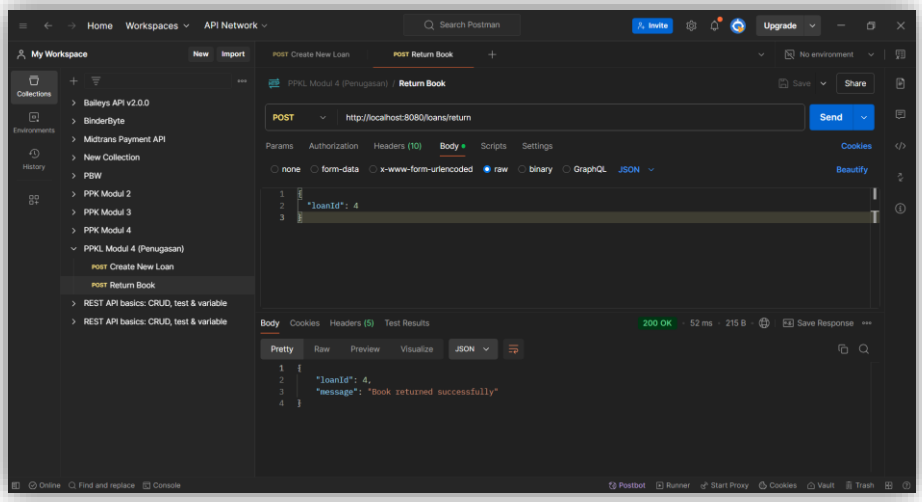
---

Modifikasi aplikasi perpustakaan dengan menambahkan RESTful API untuk layanan peminjaman dan pengembalian buku oleh member. Atribut peminjaman buku antara lain: id member yang meminjam, id buku yang dipinjam, tanggal pinjam, tanggal kembali, status peminjaman (sedang dipinjam, sudah dikembalikan), dan jumlah hari telat dikembalikan.

Link repository git: https://github.com/afrzl/Pemrograman-Platform-Khusus/tree/master/Pertemuan%204/perpustakaan

## A. Output

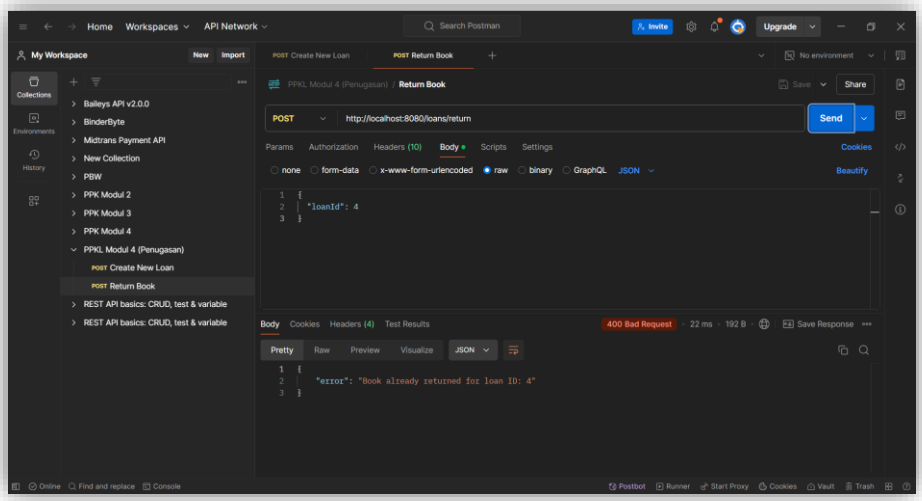


Nama : Muh. Nur Afrizal

NIM : 222212738

Kelas / No : 3SI1 / 22




## B. BookLoanController

```
package com.polstat.perpustakaan.controller;

import ...

@RestController
@RequestMapping("/loans")
public class BookLoanController {

    private static final Logger logger =
LoggerFactory.getLogger(BookLoanController.class);

    @Autowired
    private BookLoanRepository bookLoanRepository;

    @Autowired
    private BookRepository bookRepository;

    @Autowired
    private MemberRepository memberRepository;
    @Autowired
    private BookLoanService bookLoanService;

    @PostMapping
    public ResponseEntity<?> createLoan(@RequestBody BookLoanRequest
request) {
        Book book = bookRepository.findById(request.getBookId())
                .orElseThrow(() -> new RuntimeException("Book not found"));

        Member member = memberRepository.findById(request.getMemberId())
                .orElseThrow(() -> new RuntimeException("Member not
found"));

        BookLoan loan = new BookLoan();
        loan.setBook(book);
        loan.setMember(member);
        loan.setBorrowDate(request.getBorrowDate());
        loan.setStatus(BookLoan.LoanStatus.BORROWED);

        BookLoan savedLoan = bookLoanRepository.save(loan);
        return ResponseEntity.ok(savedLoan);
```

Nama : Muh. Nur Afrizal
NIM : 222212738
Kelas / No : 3SI1 / 22

```
    }

    @PostMapping("/return")
    public ResponseEntity<?> returnBook(@RequestBody BookReturnRequest
request) {
        try {
            BookLoan returnedLoan =
bookLoanService.returnBook(request.getLoanId());
            return ResponseEntity.ok(new
BookReturnResponse(returnedLoan.getId(), "Book returned successfully"));
        } catch (IllegalStateException e) {
            logger.warn("Attempt to return an already returned book: {}",
e.getMessage());
            return ResponseEntity.badRequest().body(new
ErrorResponse(e.getMessage()));
        } catch (Exception e) {
            logger.error("Error returning book", e);
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                    .body(new ErrorResponse("An unexpected error occurred
while returning the book"));
        }
    }


    @Setter
    @Getter
    private static class BookLoanRequest {
        private Long bookId;
        private Long memberId;
        private LocalDate borrowDate;
    }

    @Setter
    @Getter
    private static class BookReturnRequest {
        private Long loanId;
    }

    @Getter
    @AllArgsConstructor
    private static class BookReturnResponse {
        private Long loanId;
        private String message;
    }

    @Getter
    @AllArgsConstructor
    private static class ErrorResponse {
        private String error;
    }

}
```

BookLoanController mengelola proses peminjaman dan pengembalian buku di perpustakaan. Dalam fungsi createLoan, anggota perpustakaan bisa meminjam buku dengan mengirimkan detail peminjaman seperti ID buku, ID anggota, dan tanggal peminjaman. Jika buku dan anggota ditemukan di database, peminjaman disimpan dan statusnya diatur menjadi "BORROWED". Fungsi returnBook menangani pengembalian buku, di mana ID peminjaman

dikirim dan statusnya diperbarui menjadi "RETURNED" jika buku belum dikembalikan. Jika buku sudah dikembalikan sebelumnya, maka akan muncul peringatan.

## C. BookDTO

```
package com.polstat.perpustakaan.dto;

import ...

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class BookDto {
    private Long id;

    @NotEmpty(message = "Judul buku wajib diisi.")
    private String title;
    @NotNull(message = "Penulis buku wajib diisi.")
    private String author;
    private String description;
}
```

BookDto adalah kelas Data Transfer Object (DTO) yang digunakan untuk memindahkan data buku dalam aplikasi perpustakaan. Kelas ini memiliki atribut id, title, author, dan description. Anotasi @NotEmpty pada title memastikan judul buku harus diisi, sementara @NotNull pada author memastikan penulis buku tidak boleh null. Anotasi Lombok seperti @Data, @Builder, @NoArgsConstructor, dan @AllArgsConstructor membantu menggenerate getter, setter, dan metode bantu lainnya secara otomatis.

## D. Book

```
package com.polstat.perpustakaan.entity;

import ...
@Setter
@Getter
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Entity
@Table(name = "books")
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false)
    private String title;

    @Column(nullable = false)
    private String author;

    @Column(nullable = true)
```

```
    private String description;
}
```

Kelas BookEntity ini adalah entitas JPA yang merepresentasikan tabel 'books' dalam database. Memiliki empat atribut: id (kunci utama yang di-generate secara otomatis), title, author, dan description. Anotasi @Entity menandai kelas sebagai entitas JPA, sementara @Table menentukan nama tabel. Kelas ini menyediakan getter dan setter untuk semua atribut, memungkinkan akses dan modifikasi data buku dalam aplikasi.

## E. BookLoan

```
package com.polstat.perpustakaan.entity;

import ...

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Entity
@Table(name = "book_loans")
public class BookLoan {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "member_id", nullable = false)
    @JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
    private Member member;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "book_id", nullable = false)
    @JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
    private Book book;

    @Column(nullable = false)
    private LocalDate borrowDate;

    @Column(nullable = true)
    private LocalDate returnDate;

    @Enumerated(EnumType.STRING)
    @Column(nullable = false)
    private LoanStatus status;

    @Column(nullable = true)
    private Integer daysOverdue;

    public enum LoanStatus {
        BORROWED, RETURNED
    }
}
```

BookLoan adalah entitas yang merepresentasikan peminjaman buku dalam aplikasi perpustakaan. Entitas ini memiliki atribut seperti id, member, book, borrowDate, returnDate,

status, dan daysOverdue. Relasi @ManyToOne menghubungkan BookLoan dengan entitas Member dan Book, menggunakan lazy fetching untuk efisiensi. Anotasi @JsonIgnoreProperties digunakan untuk menghindari masalah serialisasi Hibernate. Status peminjaman dikelola menggunakan enumerasi LoanStatus, dengan nilai BORROWED atau RETURNED. Atribut daysOverdue mencatat jumlah hari keterlambatan pengembalian.

## F. Member

```
package com.polstat.perpustakaan.entity;

import ...

@Setter
@Getter
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Entity
@Table(name = "members")
public class Member {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(nullable = false)
    private String memberID;
    @Column(nullable = false)
    private String name;
    @Column(nullable = false)
    private String address;
    @Column(nullable = true)
    private String phoneNumber;
}
```

Member adalah entitas yang merepresentasikan anggota perpustakaan. Entitas ini memiliki atribut seperti id, memberID, name, address, dan phoneNumber. Atribut id dihasilkan secara otomatis dengan strategi IDENTITY, dan beberapa kolom seperti memberID, name, dan address diberi batasan nullable = false untuk memastikan data wajib diisi. Atribut phoneNumber bersifat opsional (nullable = true). Anotasi @Entity menunjukkan bahwa ini adalah entitas JPA, sementara anotasi Lombok seperti @Getter, @Setter, @AllArgsConstructor, @NoArgsConstructor, dan @Builder mempermudah pembuatan getter/setter dan konstruktor.

## G. BookMapper

```
package com.polstat.perpustakaan.mapper;

import ...

public class BookMapper {
    public static Book mapToBook(BookDto bookDto){
        return Book.builder()
                .id(bookDto.getId())
```

Nama : Muh. Nur Afrizal
NIM : 222212738
Kelas / No : 3SI1 / 22

```
                .title(bookDto.getTitle())
                .description(bookDto.getDescription())
                .author(bookDto.getAuthor())
                .build();
    }

    public static BookDto mapToBookDto(Book book){
        return BookDto.builder()
                .id(book.getId())
                .title(book.getTitle())
                .description(book.getDescription())
                .author(book.getAuthor())
                .build();
    }
}
```

BookMapper adalah kelas utilitas yang bertugas untuk memetakan antara objek BookDto dan Book. Kelas ini memiliki dua metode statis: mapToBook, yang mengonversi objek BookDto menjadi objek Book, dan mapToBookDto, yang mengonversi objek Book kembali menjadi objek BookDto. Kedua metode ini menggunakan pola builder untuk menyusun objek baru dengan mengisi atribut yang relevan dari objek sumber, sehingga memudahkan konversi data antara lapisan DTO (Data Transfer Object) dan entitas JPA.

## H. BookLoanRepository

```
package com.polstat.perpustakaan.repository;

import ...

@RepositoryRestResource(collectionResourceRel = "loans", path = "loans")
public interface BookLoanRepository extends JpaRepository<BookLoan, Long> {

    @RestResource(path = "byMember")
    List<BookLoan> findByMemberId(@Param("memberId") Long memberId);

    @RestResource(path = "byBook")
    List<BookLoan> findByBookId(@Param("bookId") Long bookId);

    @RestResource(path = "byStatus")
    List<BookLoan> findByStatus(@Param("status") BookLoan.LoanStatus
status);

    @RestResource(path = "currentLoans")
    List<BookLoan> findByStatusAndReturnDateIsNull(@Param("status")
BookLoan.LoanStatus status);

    @RestResource(path = "overdue")
    List<BookLoan> findByStatusAndReturnDateIsNullAndBorrowDateBefore(
            @Param("status") BookLoan.LoanStatus status,
            @Param("date") LocalDate date);

    @RestResource(exported = false)
    default BookLoan createLoan(@Param("memberId") Long memberId,
                                @Param("bookId") Long bookId,
                                @Param("borrowDate") LocalDate borrowDate,
                                MemberRepository memberRepository,
                                BookRepository bookRepository) {
```

Nama : Muh. Nur Afrizal
NIM : 222212738
Kelas / No : 3SI1 / 22

```
        Member member = memberRepository.findById(memberId)
                .orElseThrow(() -> new RuntimeException("Member not
found"));
        Book book = bookRepository.findById(bookId)
                .orElseThrow(() -> new RuntimeException("Book not found"));

        BookLoan bookLoan = BookLoan.builder()
                .member(member)
                .book(book)
                .borrowDate(borrowDate)
                .status(BookLoan.LoanStatus.BORROWED)
                .build();

        return save(bookLoan);
    }
}
```

BookLoanRepository adalah antarmuka yang mengelola operasi database untuk entitas BookLoan, mewarisi metode dasar dari JpaRepository. Dengan menggunakan anotasi @RepositoryRestResource, kelas ini menyediakan akses RESTful ke koleksi peminjaman buku melalui URL /loans. Terdapat beberapa metode khusus yang ditandai dengan anotasi @RestResource, yang memungkinkan pencarian peminjaman berdasarkan ID anggota, ID buku, status peminjaman, dan untuk mendapatkan peminjaman yang saat ini aktif atau sudah jatuh tempo. Selain itu, terdapat metode createLoan yang tidak diekspor secara langsung, berfungsi untuk membuat peminjaman baru dengan memeriksa keberadaan anggota dan buku sebelum menyimpannya, memastikan bahwa data yang diperlukan ada dan valid.

## I. BookRepository

```
package com.polstat.perpustakaan.repository;

import com.polstat.perpustakaan.entity.Book;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import java.util.List;

public interface BookRepository extends JpaRepository<Book, Long> {
    @Query("SELECT b FROM Book b WHERE LOWER(b.title) LIKE
LOWER(CONCAT('%', :searchTerm, '%')) OR LOWER(b.author) LIKE
LOWER(CONCAT('%', :searchTerm, '%'))")
    List<Book> searchBooks(@Param("searchTerm") String searchTerm);

    @Query("SELECT b FROM Book b WHERE LOWER(b.author) LIKE
LOWER(CONCAT('%', :author, '%')) AND LOWER(b.title) LIKE LOWER(CONCAT('%',
:title, '%'))")
    List<Book> searchBooksByAuthorAndTitle(@Param("author") String author,
@Param("title") String title);
}
```

Interface BookRepository ini memperluas JpaRepository untuk entitas BookEntity, menyediakan operasi CRUD standar dan fungsionalitas pencarian khusus:

Nama : Muh. Nur Afrizal
NIM : 222212738
Kelas / No : 3SI1 / 22

1. Mewarisi metode CRUD dasar dari JpaRepository untuk BookEntity.
2. searchBooks: Mencari buku berdasarkan kata kunci dalam judul atau penulis, menggunakan query JPQL kustom.
3. searchBooksByAuthorAndTitle: Mencari buku berdasarkan penulis dan judul secara bersamaan.

Kedua metode pencarian menggunakan anotasi @Query dengan JPQL untuk mendefinisikan logika pencarian yang fleksibel, memungkinkan pencarian case-insensitive dengan operator LIKE.

## J. MemberRepository

```
 package com.polstat.perpustakaan.repository;

import ...

import java.util.List;

@RepositoryRestResource(collectionResourceRel = "member", path = "member")
public interface MemberRepository extends
PagingAndSortingRepository<Member, Long>, CrudRepository<Member, Long> {
    List<Member> findByName(@Param("name") String name);
    List<Member> findByMemberID(@Param("member_id") String memberID);
}
```

MemberRepository adalah antarmuka yang digunakan untuk mengelola entitas Member, mewarisi metode dari PagingAndSortingRepository dan CrudRepository untuk mendukung operasi CRUD dan pagination. Dengan anotasi @RepositoryRestResource, kelas ini menyediakan akses RESTful ke data anggota melalui URL /member. Terdapat dua metode khusus, findByName dan findByMemberID, yang memungkinkan pencarian anggota berdasarkan nama atau ID anggota yang diberikan. Ini memudahkan pengguna untuk menemukan informasi anggota dengan cepat melalui endpoint RESTful yang disediakan.

## K. BookLoanService

```
package com.polstat.perpustakaan.service;

import ...
@Service
public class BookLoanService {

    private static final Logger logger =
LoggerFactory.getLogger(BookLoanService.class);

    @Autowired
    private BookLoanRepository bookLoanRepository;

    @Autowired
    private BookRepository bookRepository;
```

Nama : Muh. Nur Afrizal

NIM : 222212738

Kelas / No : 3SI1 / 22

```
    @Autowired
    private MemberRepository memberRepository;

    public BookLoan createLoan(Long memberId, Long bookId, LocalDate
borrowDate) {
        return bookLoanRepository.createLoan(memberId, bookId, borrowDate,
memberRepository, bookRepository);
    }

    public BookLoan borrowBook(Long memberId, Long bookId) {
        Member member = memberRepository.findById(memberId)
                .orElseThrow(() -> new RuntimeException("Member not
found"));
        Book book = bookRepository.findById(bookId)
                .orElseThrow(() -> new RuntimeException("Book not found"));

        BookLoan bookLoan = BookLoan.builder()
                .member(member)
                .book(book)
                .borrowDate(LocalDate.now())
                .status(BookLoan.LoanStatus.BORROWED)
                .build();

        return bookLoanRepository.save(bookLoan);
    }

    @Transactional
    public BookLoan returnBook(Long loanId) {
        BookLoan bookLoan = bookLoanRepository.findById(loanId)
                .orElseThrow(() -> new RuntimeException("Loan not found
with ID: " + loanId));

        Hibernate.initialize(bookLoan.getMember());
        Hibernate.initialize(bookLoan.getBook());

        if (bookLoan.getStatus() == BookLoan.LoanStatus.RETURNED) {
            throw new IllegalStateException("Book already returned for loan
ID: " + loanId);
        }

        LocalDate returnDate = LocalDate.now();
        bookLoan.setReturnDate(returnDate);
        bookLoan.setStatus(BookLoan.LoanStatus.RETURNED);

        long daysOverdue =
ChronoUnit.DAYS.between(bookLoan.getBorrowDate(), returnDate) - 7;
        bookLoan.setDaysOverdue(daysOverdue > 0 ? (int) daysOverdue : 0);

        return bookLoanRepository.save(bookLoan);
    }

    public List<BookLoan> getLoansByMember(Long memberId) {
        return bookLoanRepository.findByMemberId(memberId);
    }

    public List<BookLoan> getLoansByBook(Long bookId) {
        return bookLoanRepository.findByBookId(bookId);
    }

    public List<BookLoan> getCurrentLoans() {
        return
```

Nama : Muh. Nur Afrizal
NIM : 222212738
Kelas / No : 3SI1 / 22

```
bookLoanRepository.findByStatus(BookLoan.LoanStatus.BORROWED);
    }
}
```

BookLoanService adalah kelas layanan yang bertanggung jawab untuk mengelola proses peminjaman dan pengembalian buku dalam sistem perpustakaan. Kelas ini menggunakan repositori BookLoanRepository, BookRepository, dan MemberRepository untuk melakukan operasi database. Metode createLoan dan borrowBook digunakan untuk membuat peminjaman baru, sementara metode returnBook menangani pengembalian buku, termasuk perhitungan denda jika ada keterlambatan. Metode lain seperti getLoansByMember, getLoansByBook, dan getCurrentLoans menyediakan akses untuk mendapatkan daftar peminjaman berdasarkan anggota, buku, atau peminjaman yang sedang aktif. Kelas ini juga mencatat aktivitas menggunakan logger untuk membantu dalam debugging dan pemantauan.

## L. BookService

```
package com.polstat.perpustakaan.service;

import ...
public interface BookService {

    void createBook(BookDto bookDto);

    public List<BookDto> getBooks();

    public List<BookDto> searchBooks(String searchTerm);

    public List<BookDto> searchBooks(String title, String author);
}
```

## M. BookServiceImpl

```
package com.polstat.perpustakaan.service;

import com.polstat.perpustakaan.dto.BookDto;
import com.polstat.perpustakaan.entity.Book;
import com.polstat.perpustakaan.mapper.BookMapper;
import com.polstat.perpustakaan.repository.BookRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.stream.Collectors;

@Service
public class BookServiceImpl implements  BookService {
    @Autowired
    private BookRepository bookRepository;

    @Override
    public void createBook(BookDto bookDto) {
        bookRepository.save(BookMapper.mapToBook(bookDto));
    }
    @Override
```

Nama : Muh. Nur Afrizal
NIM : 222212738
Kelas / No : 3SI1 / 22

```
    public List<BookDto> getBooks() {
        List<Book> books = bookRepository.findAll();
        List<BookDto> bookDtos = books.stream()
                .map((product) -> (BookMapper.mapToBookDto(product)))
                .collect(Collectors.toList());
        return bookDtos;
    }

    public List<BookDto> searchBooks(String searchTerm) {
        return bookRepository.searchBooks(searchTerm).stream()
                .map(BookMapper::mapToBookDto)
                .collect(Collectors.toList());
    }

    @Override
    public List<BookDto> searchBooks(String author, String title) {
        return bookRepository.searchBooksByAuthorAndTitle(author,
title).stream()
                .map(BookMapper::mapToBookDto)
                .collect(Collectors.toList());
    }
}
```

Kelas BookServiceImpl ini mengimplementasikan BookService, menyediakan logika bisnis untuk operasi buku:

1. Menggunakan BookRepository untuk akses data.
2. createBook: Menyimpan buku baru ke database.
3. getBooks: Mengambil semua buku dari database.
4. searchBooks(String): Mencari buku berdasarkan kata kunci.
5. searchBooks(String, String): Mencari buku berdasarkan penulis dan judul.

Semua metode menggunakan BookMapper untuk konversi antara entitas database (BookEntity) dan objek SOAP (Book). Kelas ini menggunakan Stream API untuk transformasi data.