Entity Layer:

```
package com.book.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Table(
        name="books",
        uniqueConstraints = {@UniqueConstraint(columnNames =
{"title"})}
)
public class Book {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    @Column(name="title", nullable=false)
    private String title;

    @Column(name="description", nullable=false)
    private String description;

    @Column(name="content", nullable=false)
    private String content;

}
```

Repository Layer:

```
package com.book.repositories;

import com.book.entities.Book;
import org.springframework.data.jpa.repository.JpaRepository;

public interface BookRepository extends JpaRepository<Book,Long>
{
}
```

DTO Layer:

```
package com.book.payload;

import lombok.Data;

@Data
public class BookDto {
    private long id;
    private String title;
    private String description;
    private String content;
}
```

Controller Layer:

```
package com.book.controller;

import com.book.payload.BookDto;
import com.book.repositories.BookRepository;
import com.book.service.BookService;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;
```

```java
@RestController
@RequestMapping("/api/books")
public class BookController {

    private BookService bookService;

    public BookController(BookService bookService) {
        this.bookService = bookService;
    }


    //Create:
    //http://localhost:8080/api/books
    @PostMapping
    public ResponseEntity<BookDto> createBook(@RequestBody
BookDto bookDto){
        BookDto dto = bookService.createBook(bookDto);
        return new ResponseEntity<>(dto, HttpStatus.CREATED);
    }

    //Read or Fetching all the data from the db:

    @GetMapping
    public List<BookDto> getAllBooks(){
        List<BookDto> bookDtos = bookService.getAllBooks();
        return bookDtos;
    }


    //Read Books based on id:
    //http://localhost:8080/api/books/1
    @GetMapping("/{id}")
    public
ResponseEntity<BookDto>getBookById(@PathVariable("id") long id){
        BookDto dto = bookService.getBookById(id);
        return new ResponseEntity<>(dto, HttpStatus.OK);
    }



    //Update the Book:
    //http://localhost:8080/api/books/1
    @PutMapping("/{id}")
    public ResponseEntity<BookDto> updateBook(@RequestBody
BookDto bookDto,
```

```
                                                @PathVariable("id") long
id){
        BookDto dto=bookService.updateBook(bookDto,id);
         return new ResponseEntity<>(dto,HttpStatus.OK);
    }


    //Delete the Book:

    @DeleteMapping("/{id}")
    public ResponseEntity<String>
deleteBook(@PathVariable("id")long id){
        bookService.deleteBook(id);
        return new ResponseEntity<>("Book is deleted",
HttpStatus.OK);
    }
}
```

Service Layer:

```
package com.book.service;

import com.book.payload.BookDto;

import java.util.List;

public interface BookService {
    BookDto createBook(BookDto bookDto);
    List<BookDto> getAllBooks();
    BookDto getBookById(long id);
    BookDto updateBook(BookDto bookDto, long id);
    void deleteBook(long id);
}
```

ServiceImpl Layer:

```
package com.book.service.impl;

import com.book.entities.Book;
import com.book.exception.ResourceNotFoundException;
import com.book.payload.BookDto;
import com.book.repositories.BookRepository;
import com.book.service.BookService;
import org.springframework.stereotype.Service;

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

@Service
public class BookServiceImpl implements BookService {

    private BookRepository bookRepository;

    public BookServiceImpl(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    @Override
    public BookDto createBook(BookDto bookDto) {
        //Lets save the bookDto to DB, before that convert it
into entity
        Book book = mapToEntity(bookDto);
        Book savedBook = bookRepository.save(book);

        //Now convert that back into dto
        BookDto dto = mapToDto(savedBook);
        return dto;

    }
    @Override
    public List<BookDto> getAllBooks() {
        List<Book> books = bookRepository.findAll();

        //but we will get all the book records un Entity, so
convert each into Dto first
        List<BookDto> bookDtos = books.stream().map(book ->
mapToDto(book)).collect(Collectors.toList());
        return bookDtos;
    }

    @Override
```

```
    public BookDto getBookById(long id) {
        //firstly check if the id is present or not?
        Book book = bookRepository.findById(id).orElseThrow(
                () -> new ResourceNotFoundException("Post Not
Found with the Id: " + id)
        );
        BookDto dto = mapToDto(book);
        return dto;
    }

    @Override
    public BookDto updateBook(BookDto bookDto, long id) {
        //before updating, check if the Book Id is available in
the db or not?
        Book book = bookRepository.findById(id).orElseThrow(
                () -> new ResourceNotFoundException("Post not
Found with id " + id)
        );

        book.setTitle(bookDto.getTitle());
        book.setContent(bookDto.getContent());
        book.setDescription(bookDto.getDescription());

        Book updatedBook = bookRepository.save(book);
        return mapToDto(updatedBook);
    }

    @Override
    public void deleteBook(long id) {
        //before deleting, check if the book exist in the db or
not?
        bookRepository.findById(id).orElseThrow(
                ()-> new ResourceNotFoundException("Book not
found with id "+id)
        );

        bookRepository.deleteById(id);
    }

    Book mapToEntity(BookDto bookDto){
        Book book=new Book();

        book.setTitle(bookDto.getTitle());
        book.setContent(bookDto.getContent());
        book.setDescription(bookDto.getDescription());

        return book;
```

```
    }
    BookDto mapToDto(Book book){
        BookDto dto=new BookDto();

        dto.setId(book.getId());
        dto.setTitle(book.getTitle());
        dto.setContent(book.getContent());
        dto.setDescription(book.getDescription());

        return dto;
    }

}
```