```java
import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.web.bind.annotation.*;

import org.springframework.http.ResponseEntity;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.data.jpa.repository.JpaRepository;

import javax.persistence.*;

import java.util.List;

import java.util.Optional;
```

```java
/**

* Library API Application

*

* This API provides CRUD operations for managing books in a library system.

*/

@SpringBootApplication

public class LibraryApplication {

  public static void main(String[] args) {

      SpringApplication.run(LibraryApplication.class, args);

  }

}
```

```
/**

* Book Entity

*/

@Entity

class Book {

  @Id

  @GeneratedValue(strategy = GenerationType.IDENTITY)

  private Long id;

  private String title;

  private String author;
  private String isbn;

  private int available = 1;



  // Getters and Setters

}



/**

* Repository for Book Entity

*/

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

```
/**
* Service for Book operations
*/
@Service
class BookService {
  @Autowired
  private BookRepository repository;

  /**
   * Create a new book
   */
```

```
public Book createBook(Book book) {

    return repository.save(book);

}



/**

 * Get a book by ID

 */

public Optional<Book> getBook(Long id) {

    return repository.findById(id);

}
```

```
/**

 * Update a book's details

 */

public Book updateBook(Long id, Book updatedBook) {

    return repository.findById(id).map(book -> {

        book.setTitle(updatedBook.getTitle());

        book.setAuthor(updatedBook.getAuthor());

        return repository.save(book);

    }).orElseThrow(() -> new RuntimeException("Book not found"));

}
```

```
  /**
   * Delete a book by ID
   */
  public void deleteBook(Long id) {
      repository.deleteById(id);
  }
}


/**
* Controller for Book endpoints
*/
```

```
@RestController
@RequestMapping("/books")
class BookController {
  @Autowired
  private BookService service;


  /**
   * Add a new book
   */
  @PostMapping
  public ResponseEntity<Book> createBook(@RequestBody Book book) {
      return ResponseEntity.ok(service.createBook(book));
  }


  /**
   * Retrieve a book by ID
   */
  @GetMapping("/{id}")
  public ResponseEntity<Book> getBook(@PathVariable Long id) {
      return service.getBook(id).map(ResponseEntity::ok)
              .orElse(ResponseEntity.notFound().build());
  }
```

```
/**
 * Update a book's information
 */
@PutMapping("/{id}")
public ResponseEntity<Book> updateBook(@PathVariable Long id, @RequestBody Book book) {
    return ResponseEntity.ok(service.updateBook(id, book));
}


/**
 * Delete a book by ID
```

```
     */

    @DeleteMapping("/{id}")

    public ResponseEntity<Void> deleteBook(@PathVariable Long id) {

        service.deleteBook(id);

        return ResponseEntity.noContent().build();

    }

}
```