```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.hateoas.EntityModel;
import org.springframework.hateoas.Link;
import org.springframework.hateoas.server.mvc.WebMvcLinkBuilder;
import org.springframework.web.bind.annotation.*;
import java.util.*;

@SpringBootApplication
public class BookstoreApplication {
    public static void main(String[] args) {
        SpringApplication.run(BookstoreApplication.class, args);
    }
}

// Book model
class Book {
    private Long id;
    private String title;
    private String author;

    // Constructors
    public Book() {}
    public Book(Long id, String title, String author) {
        this.id = id;
        this.title = title;
        this.author = author;
    }

    // Getters and Setters
```

```
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }
    public String getAuthor() { return author; }
    public void setAuthor(String author) { this.author = author; }
}

// Repository for managing books
@RestController
@RequestMapping("/books")
class BookController {
    private final Map<Long, Book> bookStore = new HashMap<>();
    private Long bookIdCounter = 1L;

    // Create a book (POST)
    @PostMapping
    public EntityModel<Book> addBook(@RequestBody Book book) {
        book.setId(bookIdCounter++);
        bookStore.put(book.getId(), book);
        return addHateoasLinks(book);
    }

    // Get a book by ID (GET)
    @GetMapping("/{id}")
    public EntityModel<Book> getBook(@PathVariable Long id) {
        Book book = bookStore.get(id);
        if (book == null) {
            throw new RuntimeException("Book Not Found");
        }
```

```
        return addHateoasLinks(book);
    }

    // Update a book (PUT)
    @PutMapping("/{id}")
    public EntityModel<Book> updateBook(@PathVariable Long id, @RequestBody Book
bookDetails) {
        if (!bookStore.containsKey(id)) {
            throw new RuntimeException("Book Not Found");
        }
        bookDetails.setId(id);
        bookStore.put(id, bookDetails);
        return addHateoasLinks(bookDetails);
    }

    // Delete a book (DELETE)
    @DeleteMapping("/{id}")
    public String deleteBook(@PathVariable Long id) {
        if (!bookStore.containsKey(id)) {
            return "Book Not Found";
        }
        bookStore.remove(id);
        return "Book Deleted Successfully";
    }

    // Add HATEOAS links to the book entity
    private EntityModel<Book> addHateoasLinks(Book book) {
        EntityModel<Book> model = EntityModel.of(book);

model.add(WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(BookController.clas
s).getBook(book.getId())).withSelfRel());
```

```
model.add(WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(BookController.clas
s).updateBook(book.getId(), book)).withRel("update"));


model.add(WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(BookController.clas
s).deleteBook(book.getId())).withRel("delete"));
        return model;
    }
}
```