Государственный Университет Молдовы

Факультет Математики и Информатики

Департамент Информатики

**Лабораторная работа №1**

по курсу “Программирование на SPRING Framework”

Выполнил студент группы I2302:

Socolov

Daniil.,

Проверил преподаватель:

O.Ciornei

Кишинэу, 2025

**Условия работы**

Разработай Spring Boot приложение для управления библиотекой, где будут 3 контроллера, 3 сервиса и 5 взаимосвязанных сущностей. Приложение должно поддерживать CRUD-операции через REST API и использовать JPA для работы с базой данных.  
API должен работать с DTO а не с entity

Author:

Один автор может написать много книг (One-to-Many).

Publisher:

Один издатель может издать много книг (One-to-Many).

Book:

Каждая книга принадлежит одному автору (Many-to-One).  
Каждая книга имеет одного издателя (Many-to-One).  
Каждая книга может принадлежать к нескольким категориям (Many-to-Many).

Category:

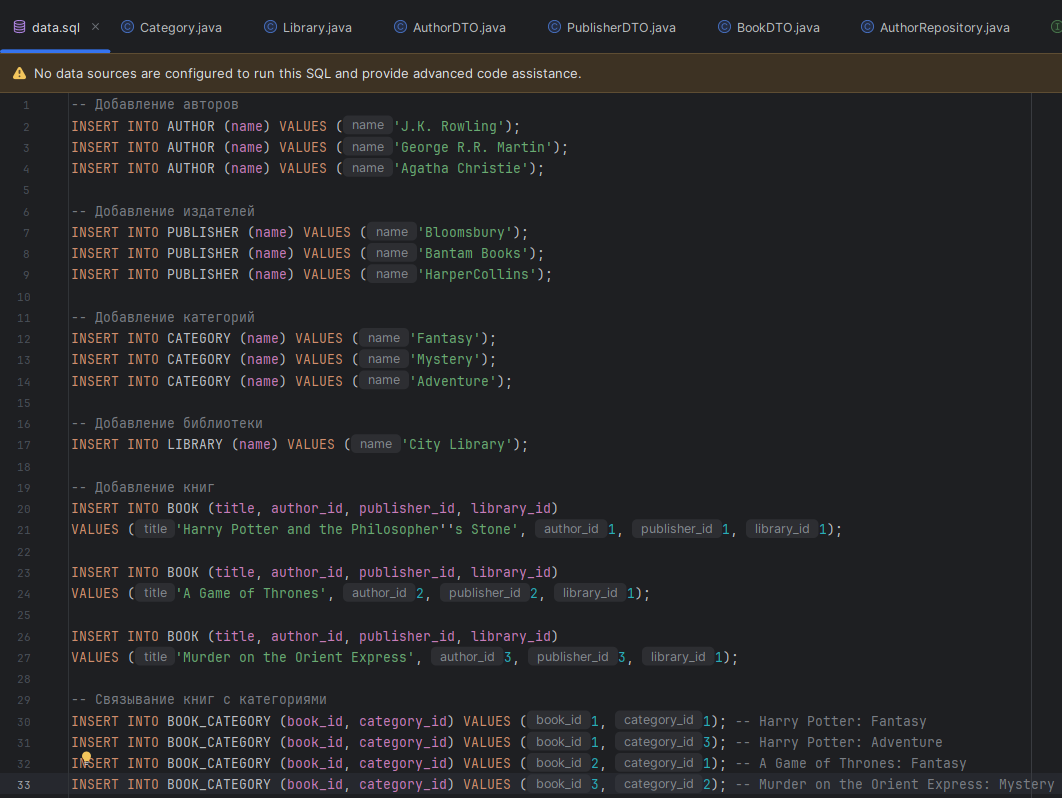
Одна категория может быть связана с несколькими книгами (Many-to-Many).

Library:

Библиотека содержит коллекцию книг в виде списка (ElementCollection).

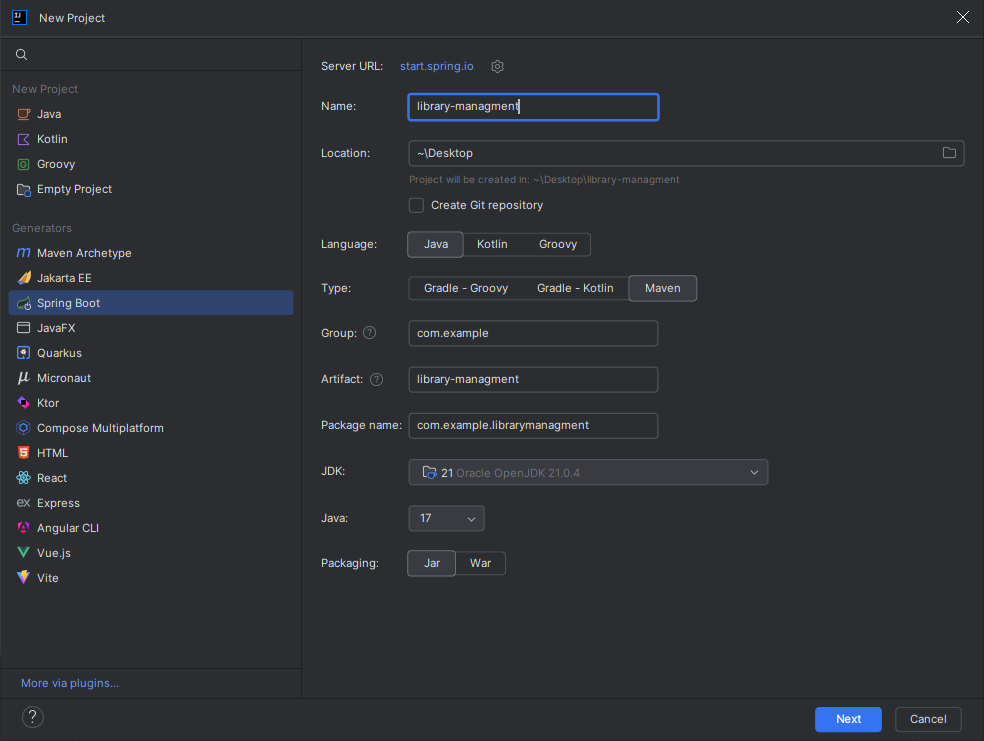
**База данных**

Создал фалй data.sql в папке resourses в котором прописал следующий код, для автоматического заполнения таблиц.

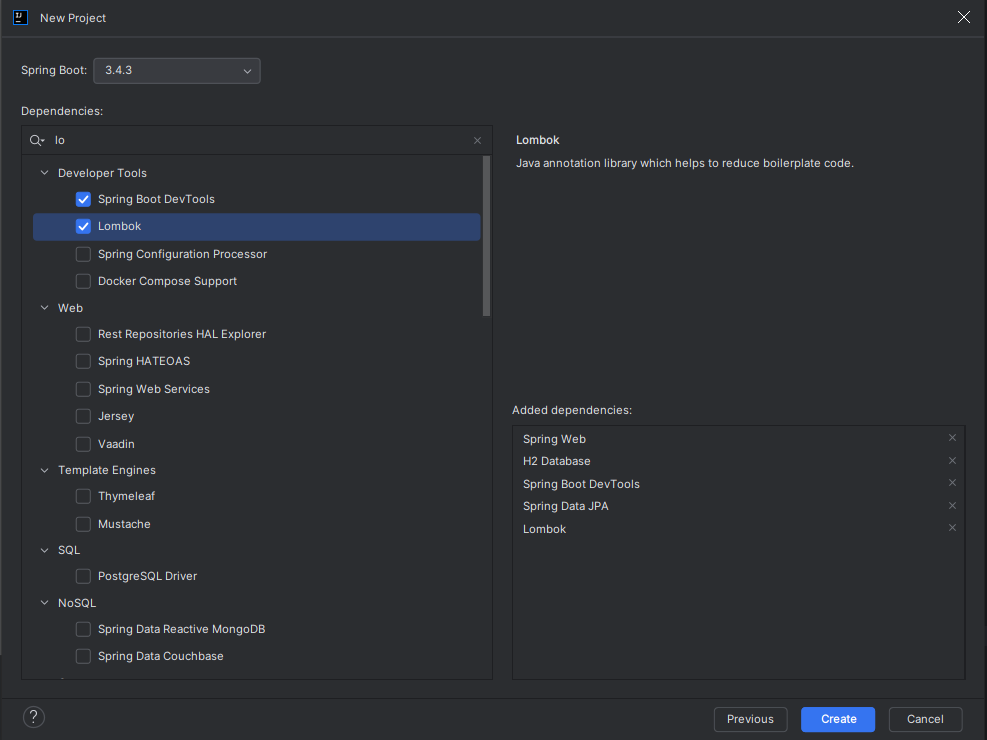


**Создание SPRING приложение**

### Создаю проект приложения в IntelliJ IDEA использую Spring Boot



Добавляю следующие зависимости: Spring Web, H2 Database, Spring Boot DevTools, Spring Data JPA, Lombok

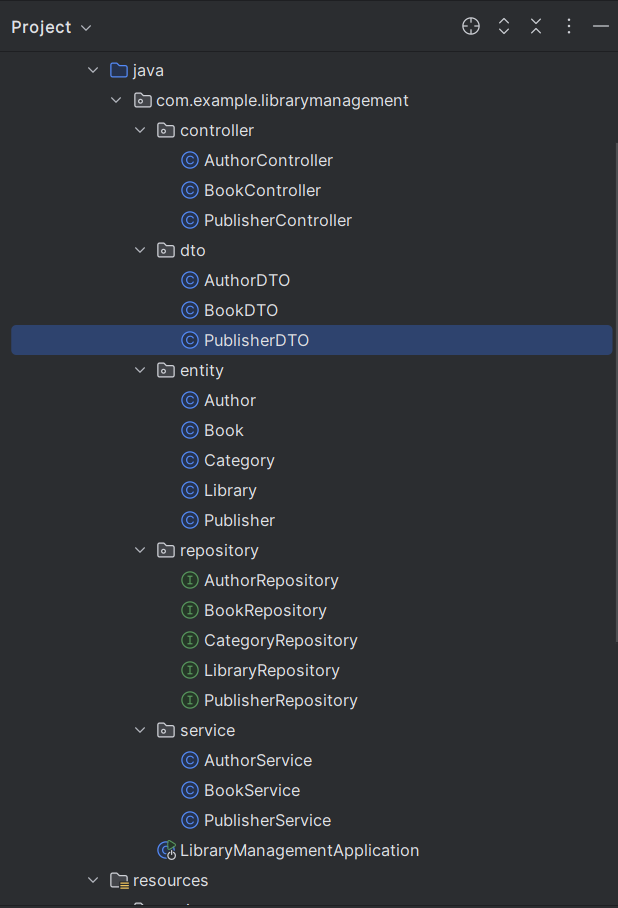


Вот краткое описание каждой зависимости:

Spring Web нужен для создания веб-приложений и REST API, предоставляя встроенный сервер и инструменты для обработки HTTP-запросов. H2 Database — это лёгкая встраиваемая база данных, идеальная для разработки и тестов, с удобной консолью для просмотра данных.

Spring Boot DevTools ускоряет разработку благодаря автоматическому перезапуску приложения при изменении кода и другим удобствам для отладки. Spring Data JPA упрощает работу с базой через JPA, предлагая готовые репозитории для операций с данными и интеграцию с Hibernate. Lombok же сокращает шаблонный код, автоматически генерируя геттеры, сеттеры и конструкторы с помощью аннотаций.

Создал структуру приложения, и создал файлы: 3 контроллера, 3 сервиса и 5 взаимосвязанных сущностей, а также 5 репозиториев и 3 dto. Сначала я продумал структуру, разделив код на пакеты вроде controller, service, entity, repository и dto, чтобы держать всё в порядке.



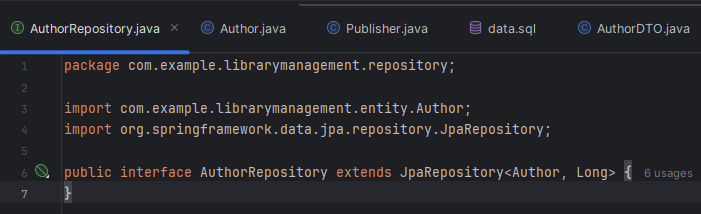
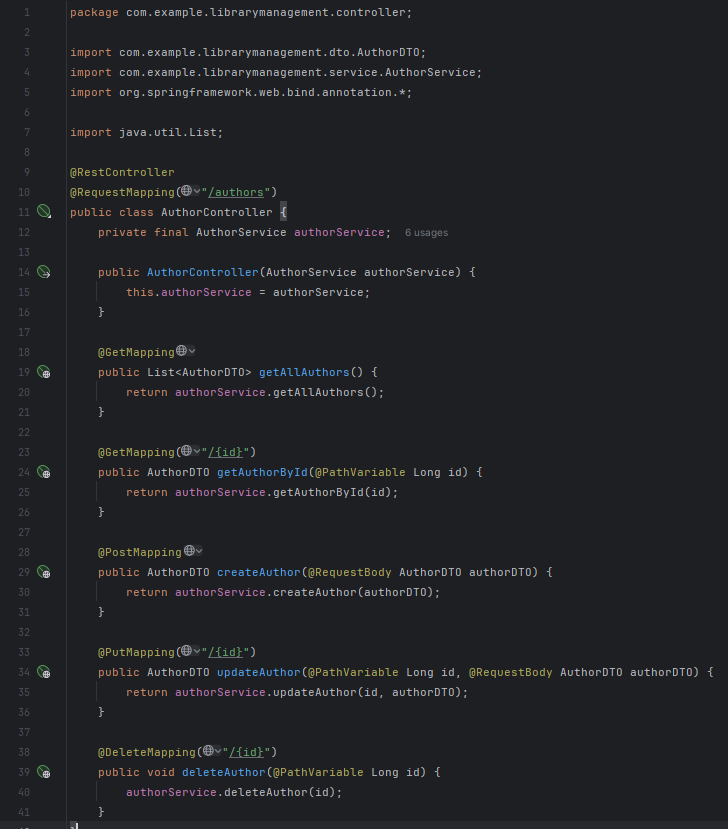
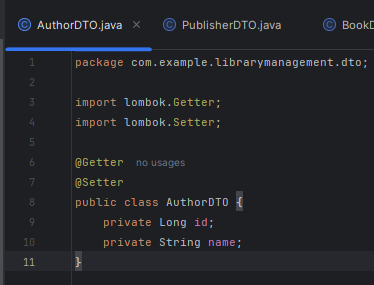
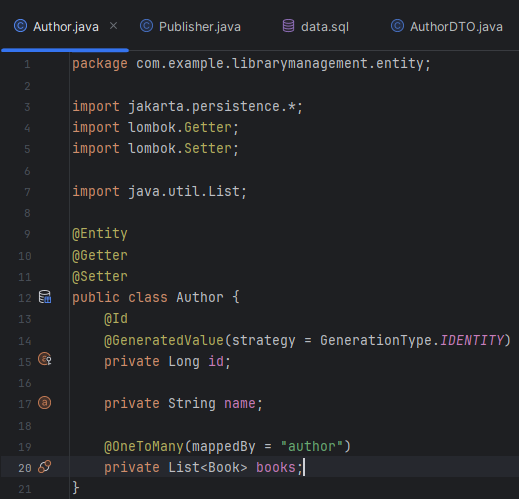
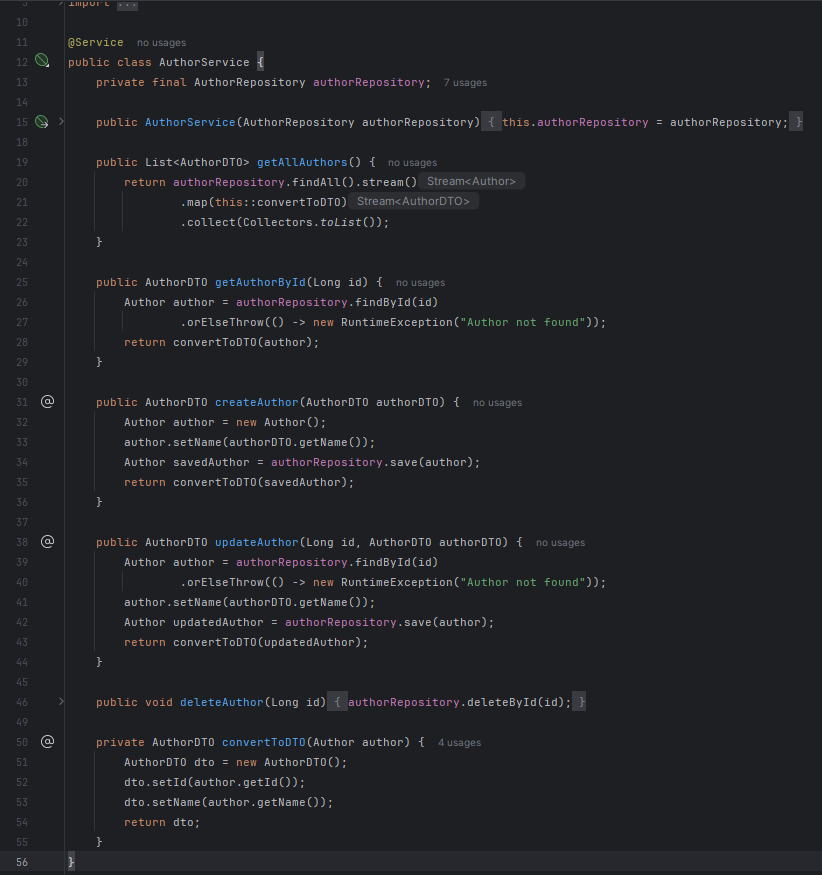
**Пример заполнения каждого типа файла на примере Author:**  
Потом я написал три контроллера — AuthorController, PublisherController и BookController. Они у меня принимают HTTP-запросы, например, чтобы добавить книгу или обновить автора через эндпоинты вроде POST /books или PUT /authors/{id}, и отправляют ответы в JSON.

Дальше я сделал три сервиса — AuthorService, PublisherService и BookService. Это основа моей бизнес-логики: они берут данные от контроллеров, проверяют их (например, есть ли автор в базе), преобразуют между DTO и сущностями и работают с базой через репозитории. Благодаря им приложение не просто сохраняет данные, а умно их обрабатывает.

Я добавил пять взаимосвязанных сущностей — Author, Publisher, Book, Category и Library. Это мои модели данных, которые описывают библиотеку. Я пометил их @Entity, чтобы Hibernate создал таблицы, и настроил связи: например, у книги (Book) есть @ManyToOne с автором и издателем, а с категориями — @ManyToMany. Теперь у меня есть полноценная структура данных с отношениями, как список книг у автора.

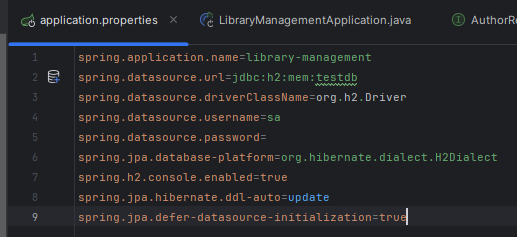
Для работы с базой я создал пять репозиториев — AuthorRepository, PublisherRepository, BookRepository, CategoryRepository и LibraryRepository. Они унаследованы от JpaRepository, так что я получил готовые методы для сохранения, поиска или удаления данных без лишнего кода.

Ещё я сделал три DTO — AuthorDTO, PublisherDTO и BookDTO. Это классы, которые я использую для передачи данных через API. Они содержат только нужные поля, например, id и title для книги, и помогают мне скрыть внутренности сущностей от внешнего мира, упрощая работу с JSON.



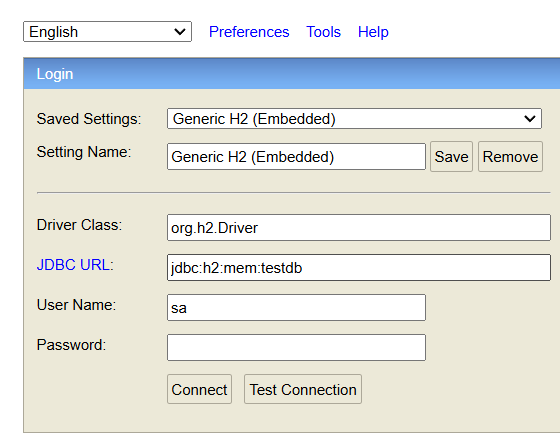
### application.properties:

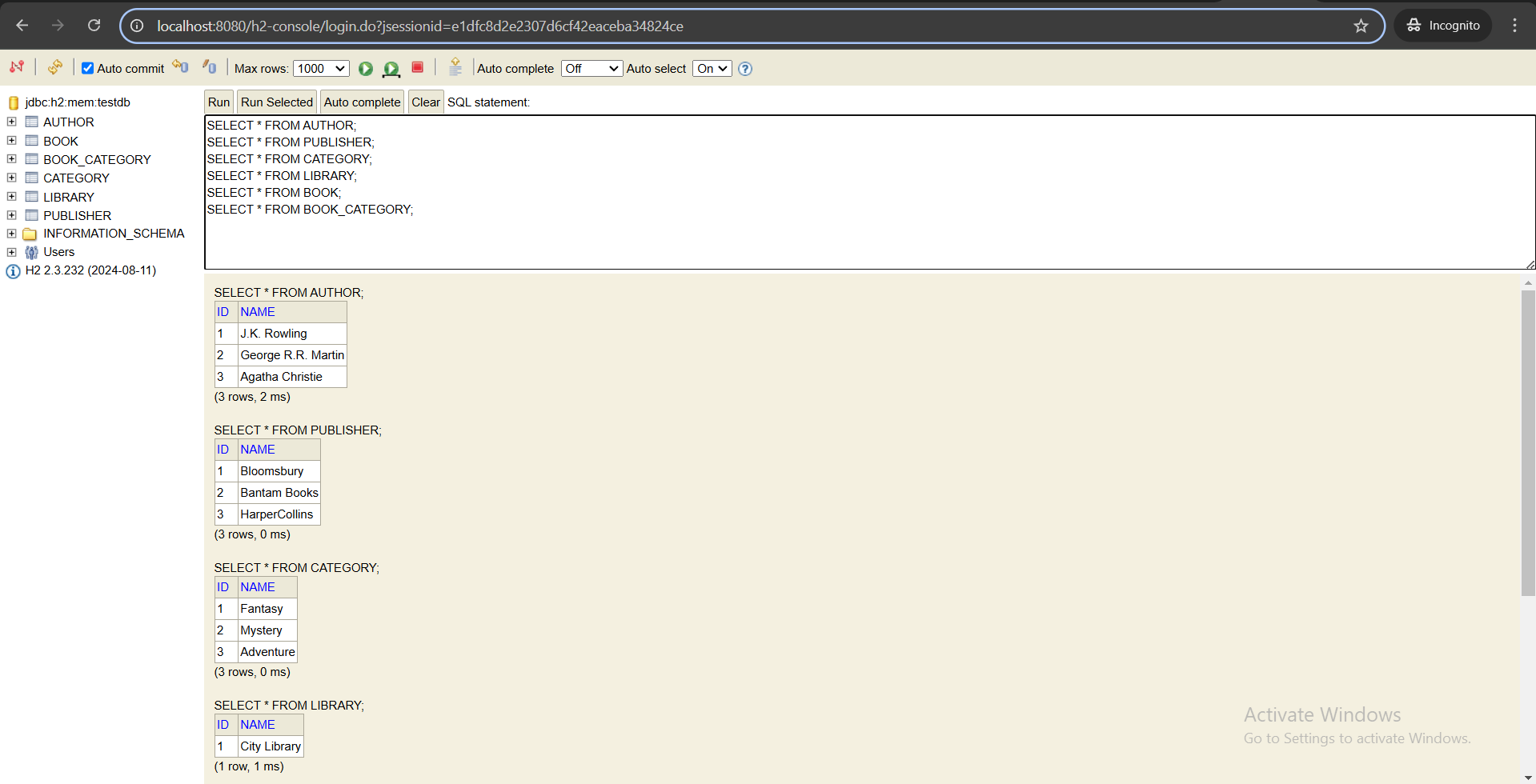
В application.properties я добавил настройки для подключения к H2 базе данных: указал URL (jdbc:h2:mem:testdb), драйвер, логин и пароль. Ещё я настроил Hibernate через spring.jpa.hibernate.ddl-auto=update для автоматического создания таблиц и добавил spring.jpa.defer-datasource-initialization=true, чтобы данные из data.sql загружались после создания таблиц. Плюс включил H2-консоль и указал диалект для работы с H2.



**Запуск SPRING приложение**

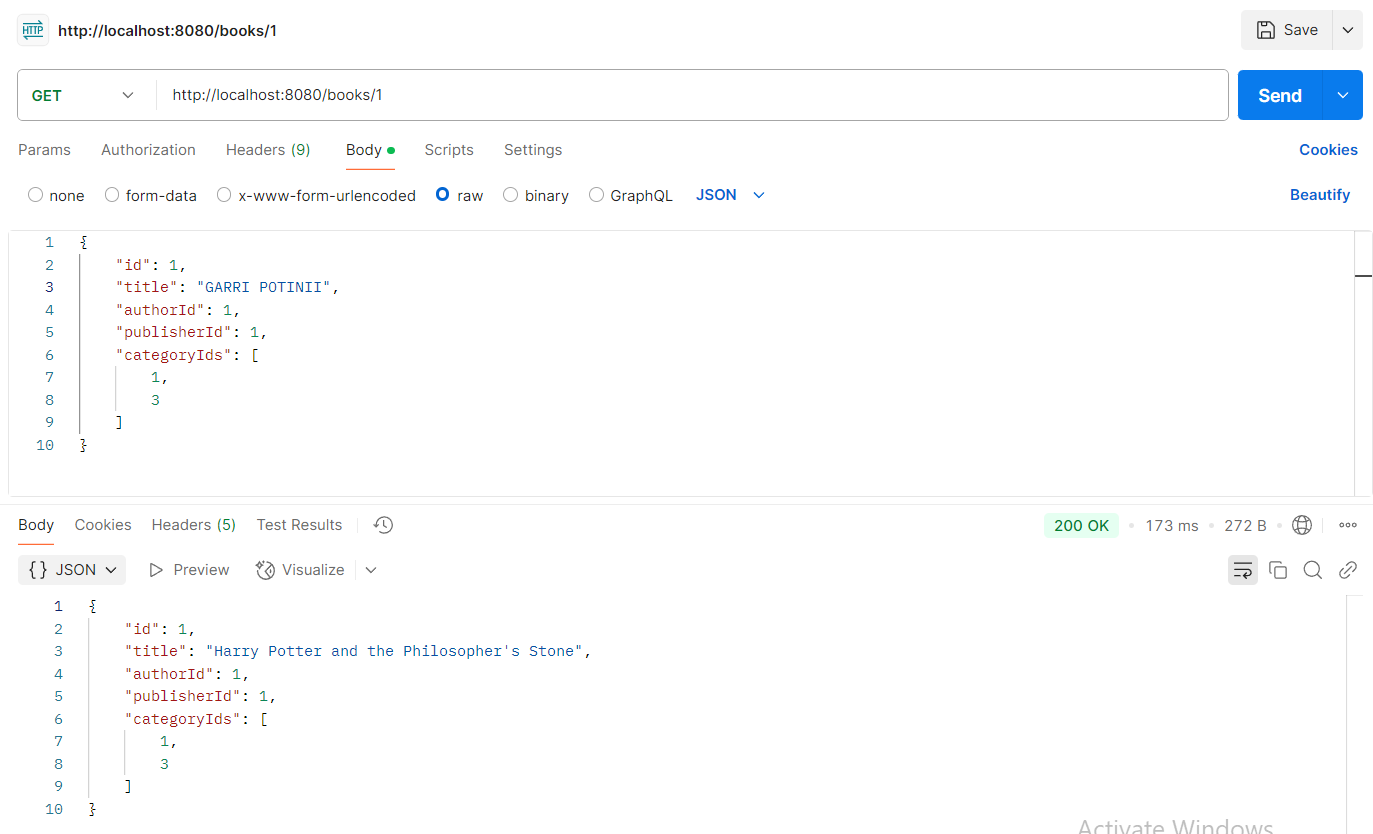
1. **Проверка базы данных**

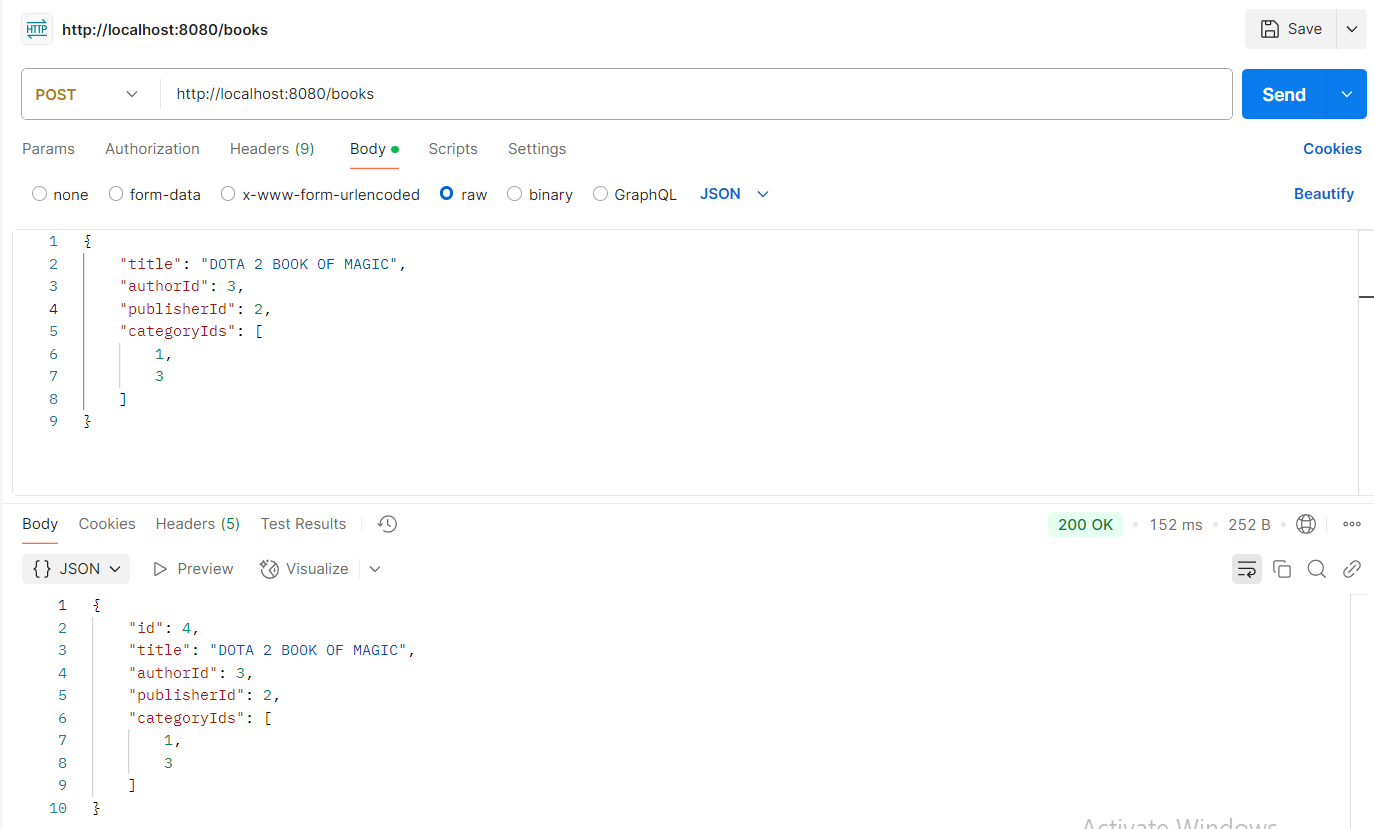
****

****

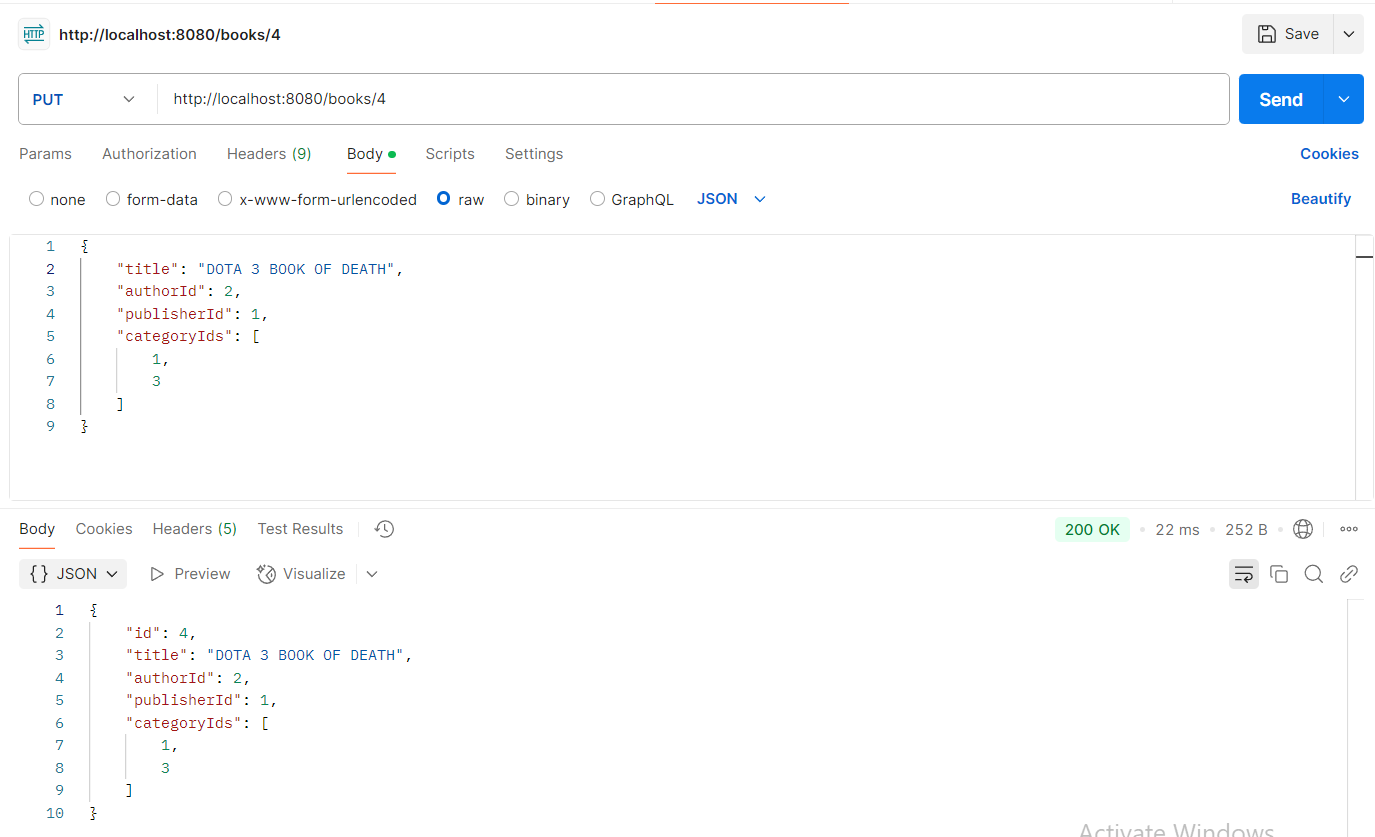
1. **Проврека CRUD запросов**

**GET**

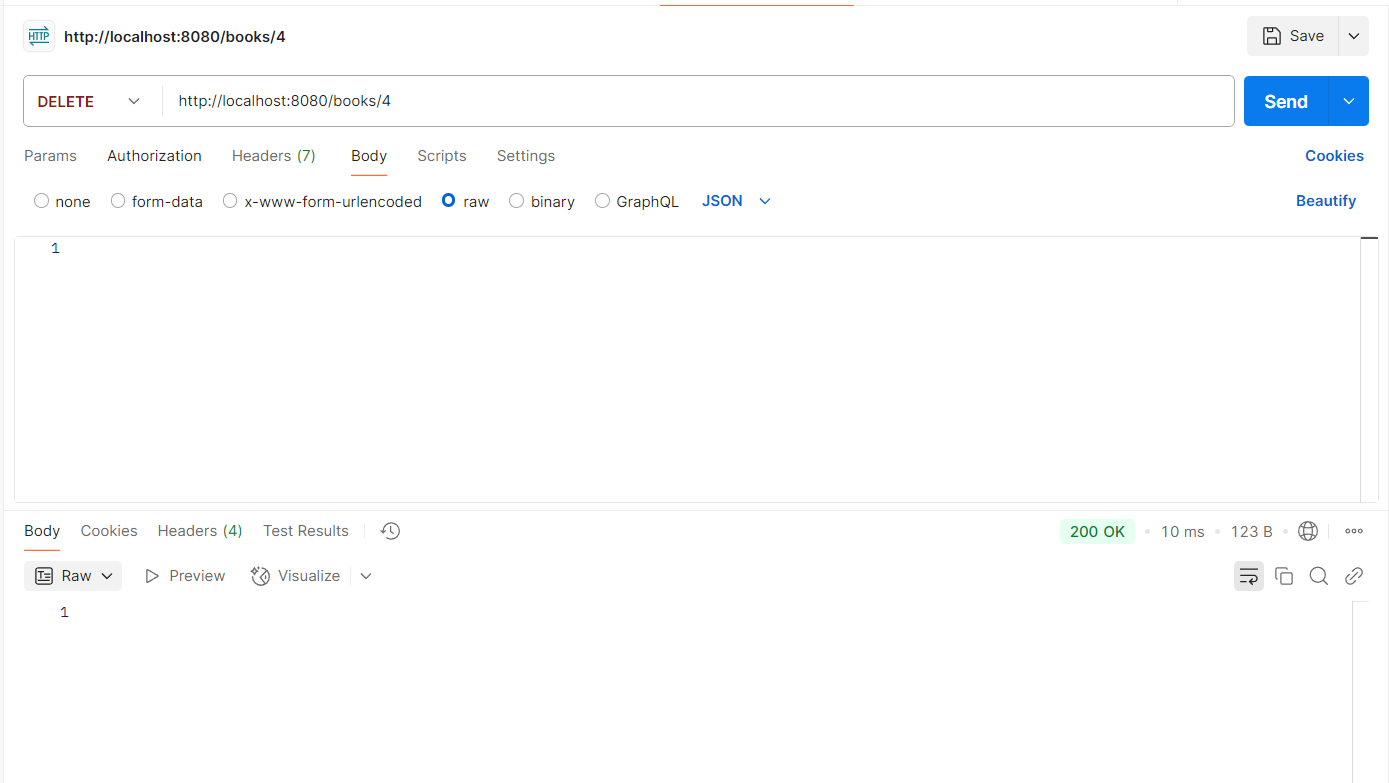
****

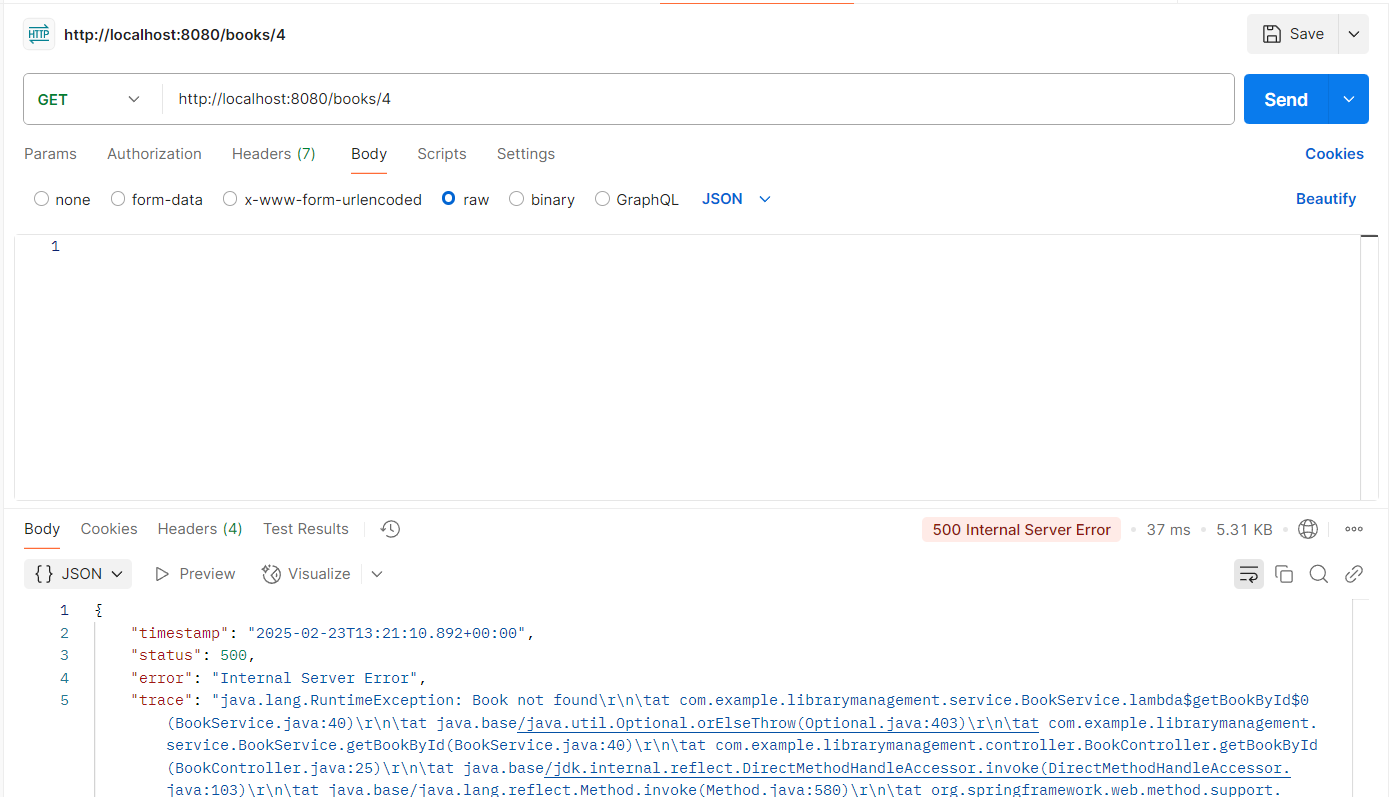
**POST  
**

**PUT**

****

**DELETE**

****

****

**Вывод:**

Я создал Spring Boot приложение для управления библиотекой с REST API, где три контроллера (Author, Publisher, Book) обрабатывают запросы, три сервиса управляют логикой, пять сущностей (Author, Publisher, Book, Category, Library) описывают данные с их связями, пять репозиториев упрощают работу с базой H2, а три DTO передают данные наружу. В application.properties я настроил подключение к H2, автоматическое создание таблиц и загрузку тестовых данных, чтобы всё работало гладко.

**Библиография:**

https://grok.com/