Министерство образования и науки РФ

ФГБОУ ВПО «Омский государственный технический университет»

# Кафедра «Прикладная математика и фундаментальная информатика»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

База данных книг

Выполнил студент гр. ФИТ-221

Шохин Егор Павлович

Проверил

ст. преподаватель Федотова И.В.

Омск 2023

Теоретическая часть по фреймворку Oat++

Фреймворк Oat++ предназначен для написания Backend приложения на языке C++. предоставляет разработчикам широкий набор инструментов для реализации различных функций веб-приложений: маршрутизацию запросов, обработку HTTP-запросов и ответов, работу с базами данных и многое другое. Кроме того, фреймворк имеет интегрированный набор инструментов для разработки и отладки приложений, что значительно упрощает процесс разработки и сокращает время на развертывание приложения в продакшене.

Одной из особенностей oat++ является его модульная архитектура, позволяющая использовать только те компоненты, которые необходимы для конкретного проекта. Благодаря этому, разработчики могут создавать масштабируемые и гибкие приложения, которые легко адаптируются к изменяющимся потребностям пользователей.

Теоретическая часть по базе данных SQLite

SQLite является самым простым вариантом реализации базы данных. Он поддерживает большинство функций и команд, которые доступны в других реляционных базах данных, таких как MySQL и PostgreSQL. Он позволяет хранить данные в таблицах, которые могут быть связаны друг с другом посредством внешних ключей. SQLite также поддерживает индексирование данных, что ускоряет процесс поиска и сортировки.

SQLite может быть использован для различных целей, включая хранение данных приложения, кэширование данных, хранение данных сенсоров и многое другое. Он также может быть использован в сочетании с другими технологиями, такими как JSON, для упрощения обмена данными между приложениями.

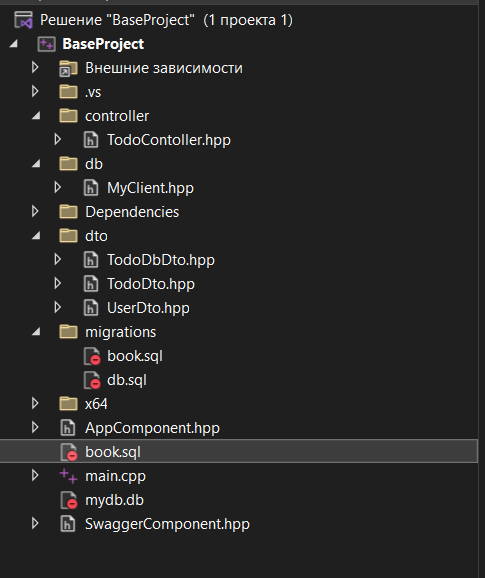
Теоретическая часть по Postman

Postman - это инструмент для тестирования API и разработки программного обеспечения, который облегчает создание, отправку, тестирование и отладку HTTP-запросов. Postman позволяет разработчикам быстро и удобно проверять работоспособность API и взаимодействовать с ними без необходимости написания собственных скриптов или использования других инструментов.

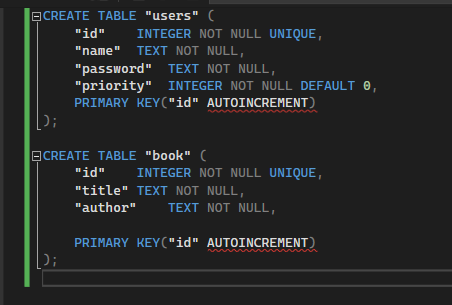
Postman имеет простой и интуитивно понятный интерфейс, который позволяет создавать и редактировать запросы, устанавливать параметры запросов и просматривать ответы от серверов. Он также поддерживает автоматическую генерацию кода для разных языков программирования, что упрощает интеграцию с различными приложениями.

Postman также имеет множество функций, которые помогают улучшить процесс разработки и тестирования, такие как сохранение истории запросов, создание коллекций запросов и переменных окружения, использование коллаборации и многое другое. Он также позволяет тестировать авторизацию и безопасность API, включая тестирование JWT-токенов и проверку SSL- сертификатов.

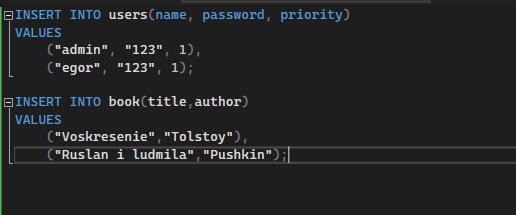
Общая структура проекта:



Создание таблиц users(пользователи) и books(книги):



Значения в таблицах:

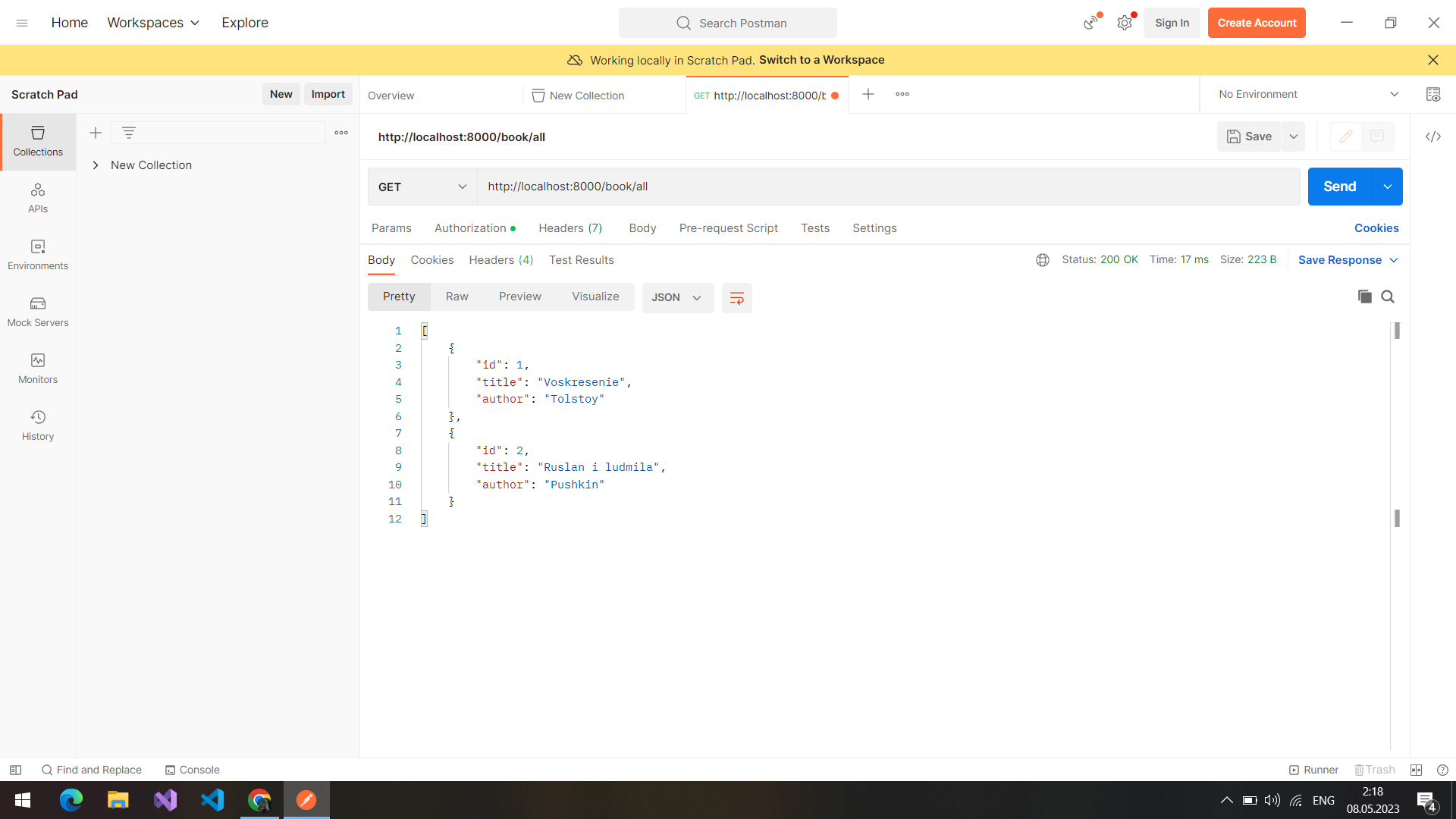


В качестве пользователей выступает admin,egor.

Рассмотрим такие методы как:

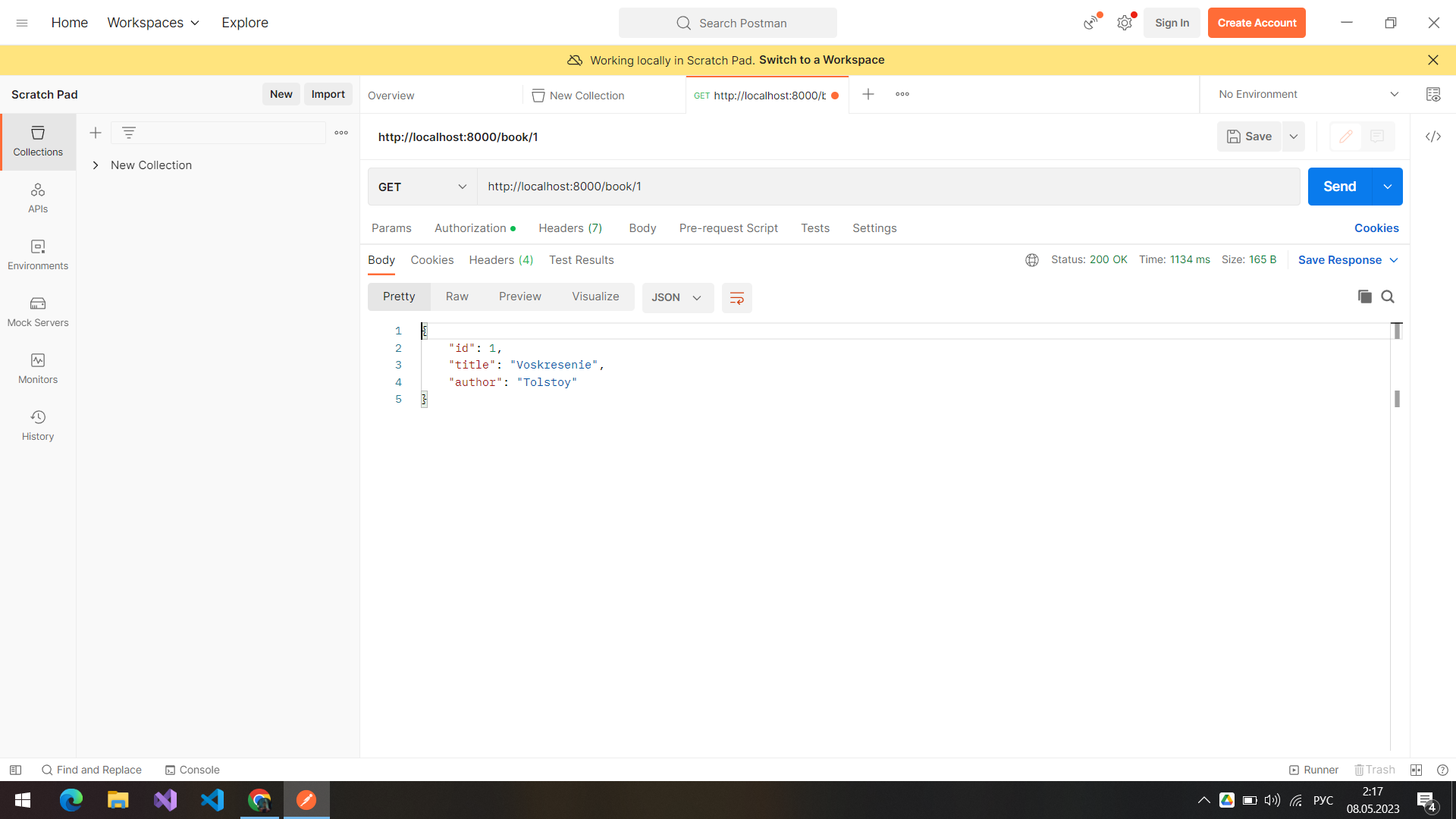
1. Вывод книг
2. Вывод книги по id
3. Обновление данных библиотеки
4. Удаление книги по id
5. Добавление книги

**Вывод городов:**



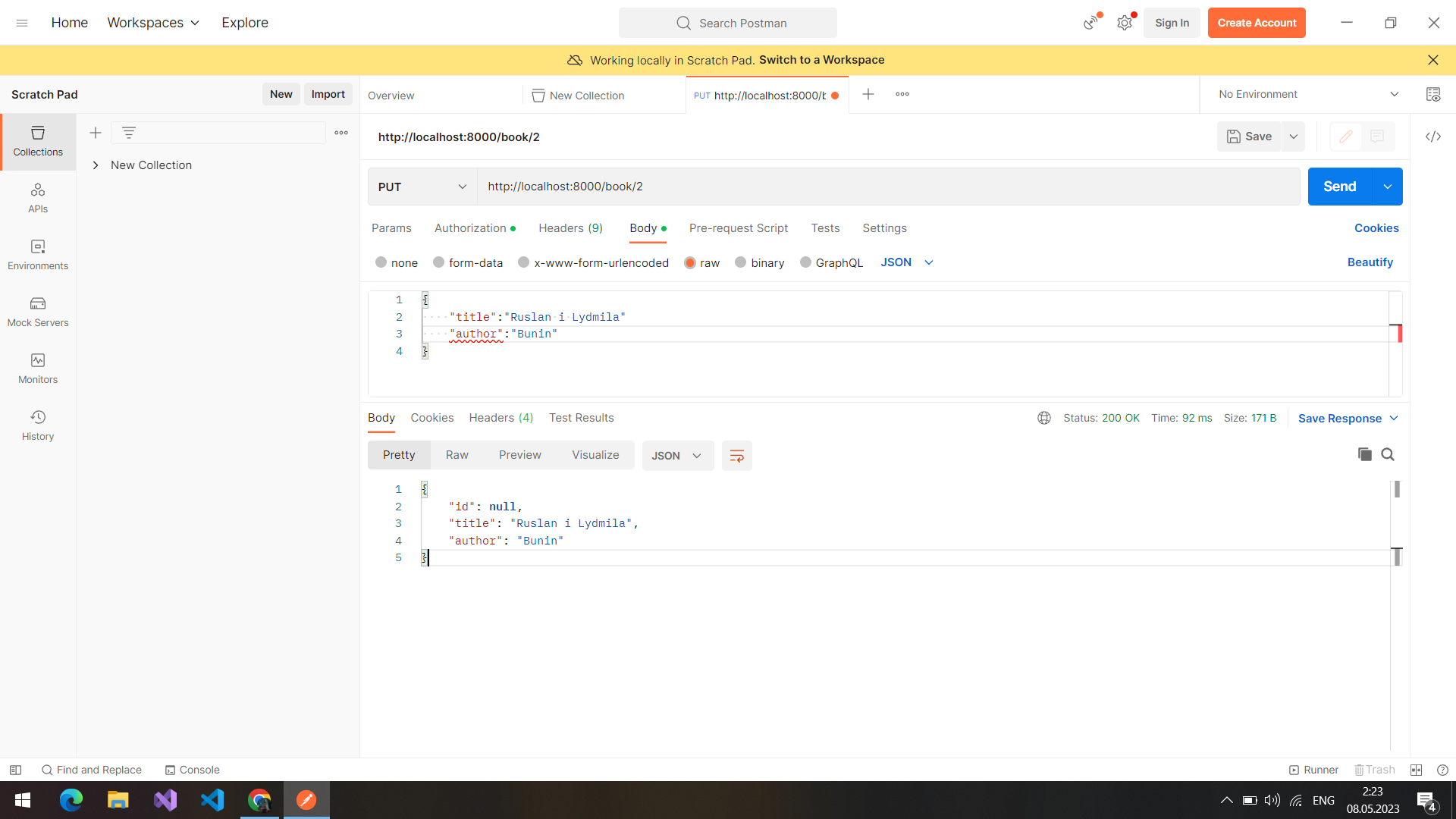
Можно увидеть, что вывелись все книги

**Вывод города по id:**



После запроса вывело книгу “Воскресение” Толстого с id = 1

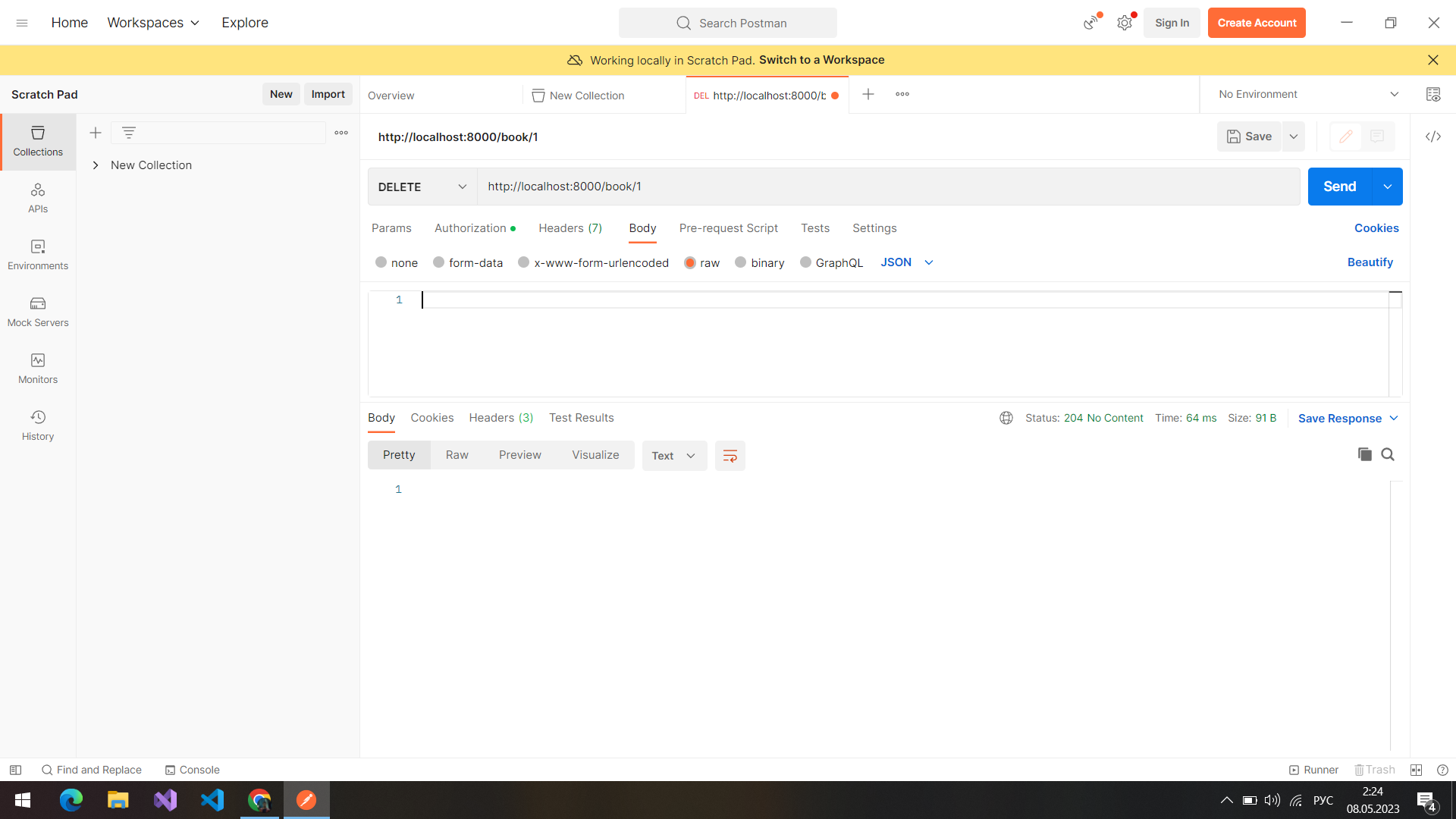
**Обновление данных библиотеки:**



Запрос меняет информацию о городе с id = 2

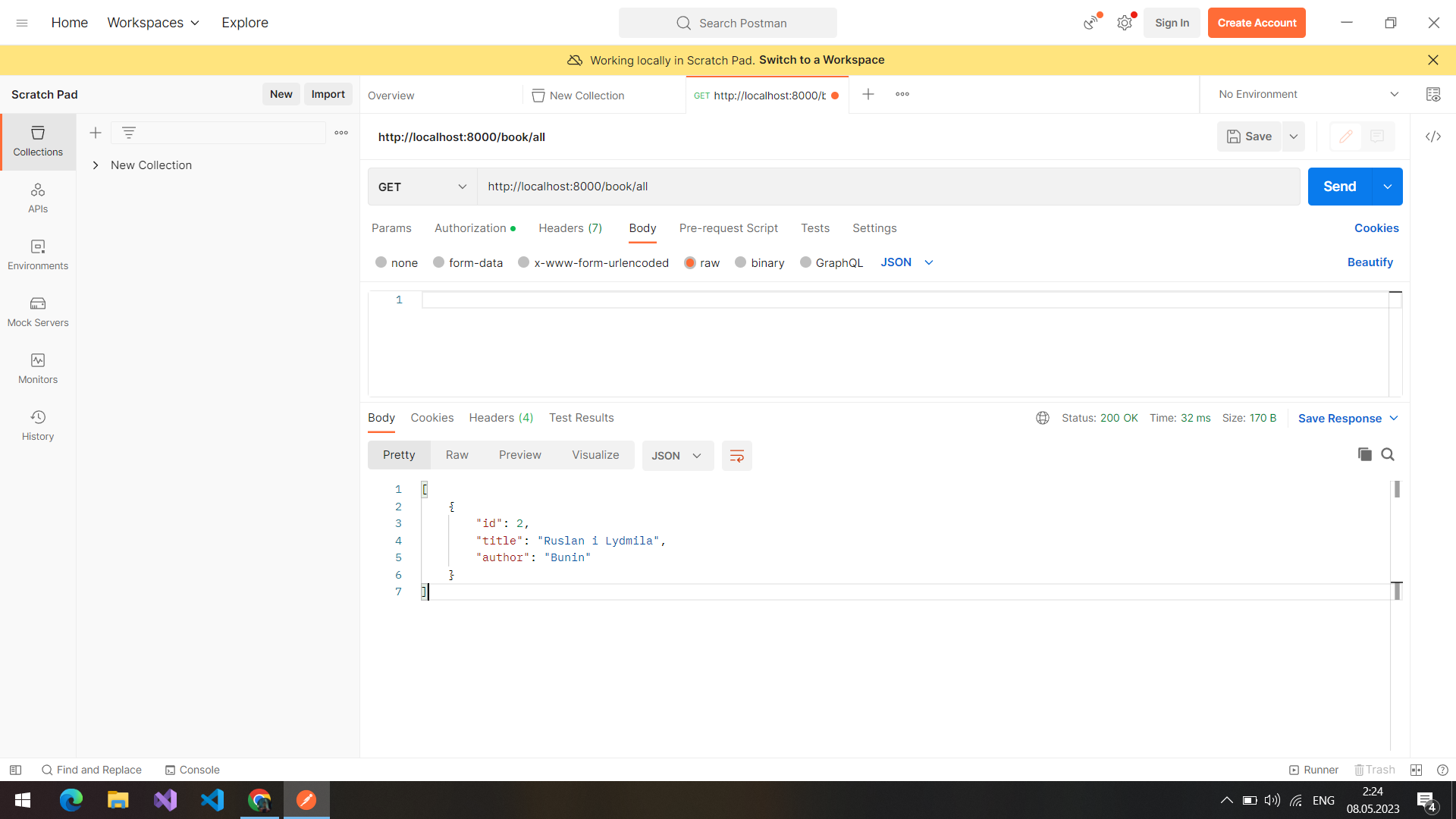
После выполнения вся информация обновилась на введённую

**Удаление книги по id:**

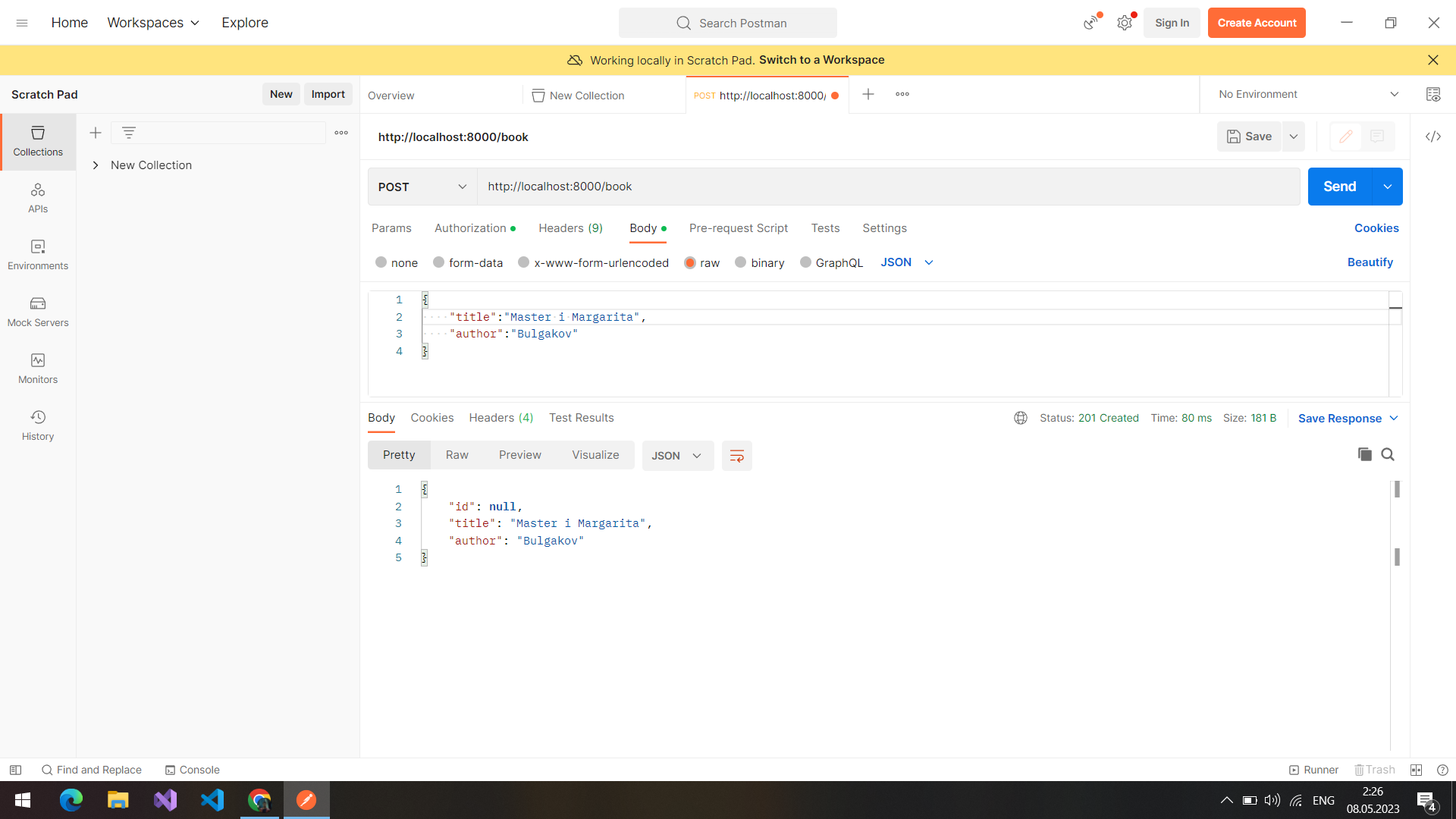


Удалили книгу с id = 1

**После удаления список книг стал таким:**



**Добавление города:**



Была добавлена книга “Мастер и Маргарита”

**Пример кода в файле main:**

#include <iostream>

#include <oatpp/network/Server.hpp>

#include "oatpp/web/server/HttpConnectionHandler.hpp"

#include "oatpp/network/tcp/server/ConnectionProvider.hpp"

#include "AppComponent.hpp"

#include "controller/TodoContoller.hpp"

#include "oatpp-swagger/Controller.hpp"

void runServer()

{

AppComponent components;

OATPP\_COMPONENT(std::shared\_ptr<oatpp::web::server::HttpRouter>, httpRouter);

//подключаем контроллеры к роутеру

oatpp::web::server::api::Endpoints docEndpoints;

docEndpoints.append(httpRouter->addController(std::make\_shared<BookController>())->getEndpoints());

//добавляем контролленры к swagger UI

httpRouter->addController(oatpp::swagger::Controller::createShared(docEndpoints));

//получаем ConnectionHandler

OATPP\_COMPONENT(std::shared\_ptr<oatpp::network::ConnectionHandler>, serverConnectionHandler);

//получаем ConnectionProvider

OATPP\_COMPONENT(std::shared\_ptr<oatpp::network::ServerConnectionProvider>, ServerConnectionProvider);

//создаём сервре

oatpp::network::Server server(ServerConnectionProvider, serverConnectionHandler);

//запускаем сервер

server.run();

}

int main() {

setlocale(LC\_CTYPE, "Rus");

oatpp::base::Environment::init();

runServer();

//system("pause");

return 0;

}