|  |
| --- |
|  |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** |
| Институт информационных технологий (ИТ) |
| Кафедра ИиППО |

|  |  |
| --- | --- |
| **ОТЧЕТ**  **ПО ПРАКТИЧЕСКОЙ РАБОТЕ №6** | |
| **по дисциплине** |  |
| **«Архитектура клиент-серверных приложений»**  **Тема: GraphQL** | |
| Выполнил студент группы ИКБО-24-21 | Новиков А.А. |
| Принял преподаватель | Волков М.Ю. |

|  |  |  |
| --- | --- | --- |
| Практические работы выполнены | « » 2023г. |  |
| «Зачтено» | « » 2023 г. |  |

Москва 2023

**Теоретическое введение**

В предыдущих работах, нами было рассмотрен один из самых распространённых, на сегодняшний момент, подходов к проектированию и разработке программных интерфейсов в вебе, который основан на архитектуре REST. Используемый ресурсно-ориентированный подход предполагает организацию данных поверх веб-ресурсов, определяемых посредством URI, а для реализации базовых операций, таких как чтение/запись, удаление, обновления использовались базовые глаголы протокола HTTP. Однако, несмотря на все преимущества данного подхода, у REST подхода имеется ряд существенных недостатков, которые могут влиять, при определённых сценариях использования, в том числе, на производительность программного интерфейса.

Самый типичный сценарий взаимодействия с API, демонстрирующий ограничения REST по сравнению с архитектурой на основе графовой модели, является сложная выборка данных. Например, имеется веб-приложение, реализующую функционал социальной сети. Клиентская часть этого приложения (например, веб-клиент) должна отображать множество данных, таких как количество просмотров страницы, количество друзей, подписчиков и т.д. В случае использования REST API, для извлечения всех требуемых данных, обычно, необходимо обратиться к нескольким конечным точкам. Таким образом, количество запросов для отображения полноценной страницы пользователя социальной сети может потребовать несколько десятков, а то и сотен запросов к API. В случае большого количества числа единовременных пользователей такой системы, количество реальных запросов к API может достигать нескольких миллионов запросов в минуту, снижая общую производительность системы в целом. Для решение этой задачи, в рамках REST архитектуры, необходимо обеспечить, либо оптимизацию соответствующей структуры конечных точек, либо оптимизировать количество запросов на сервер API.

**Постановка задачи**

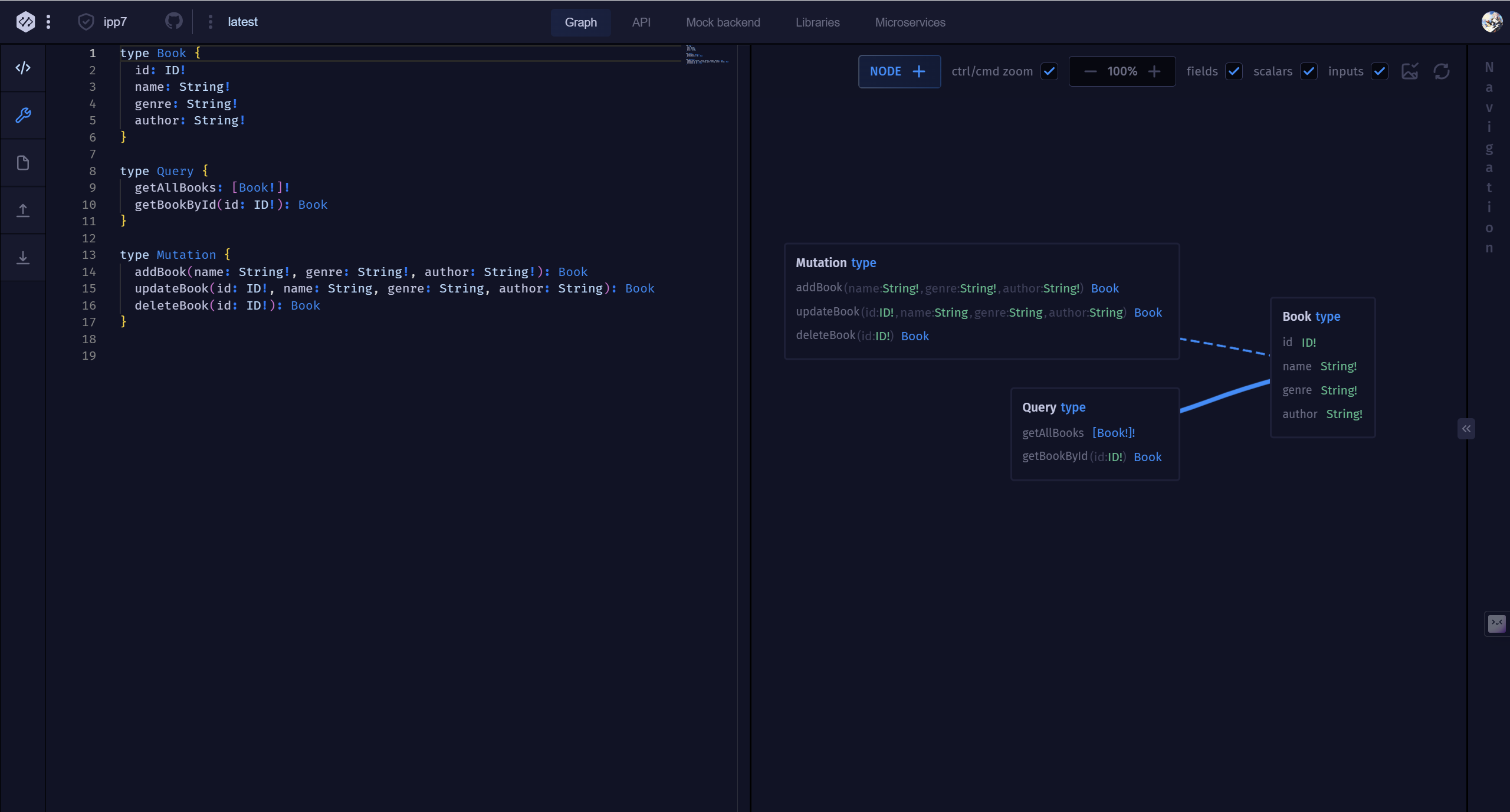
Целью данной практической работы является знакомство обучающихся с набирающим популярность современным подходом к проектированию и реализации API на основе Графовых моделей и с реализующей данный подход технологией на основе спецификации GraphQL.

Используя теоретические сведения из данной практической работы, открытые интернет-источники, официальную документацию по GraphQL необходимо, с использованием SDL создать схему, реализовать сервер и клиента GraphQL для следующих бизнес-задач (по выбору):

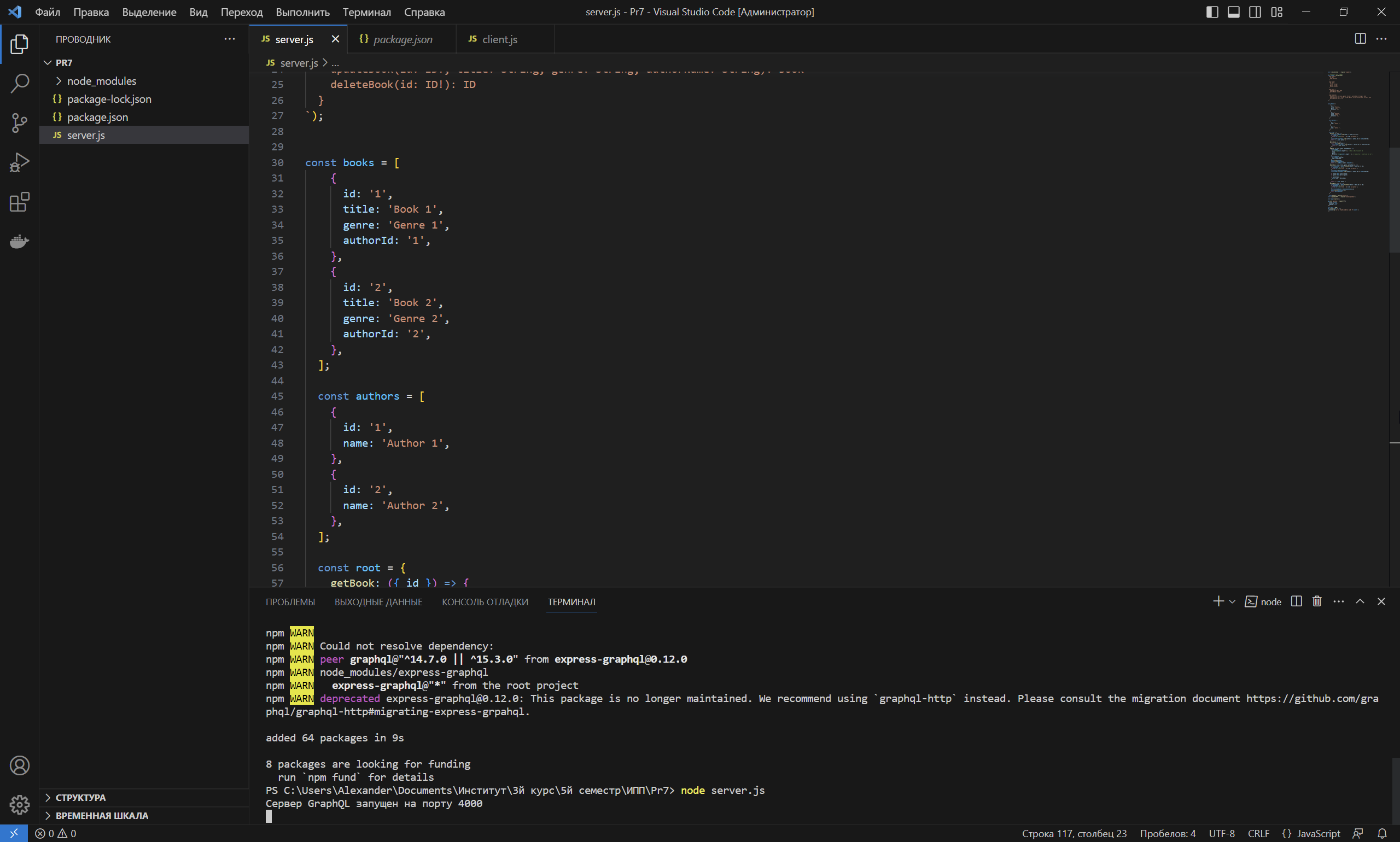
1. Создание приложения для хранения списка книг в библиотеке. Схема должна определять требуемые типы данных с полями, возвращающими определённые данные. Должны быть определены поля: name (название книги), genre(жанр книги), id (уникальный идентификатор книги), name (ФИО автора книги). Дополнительные поля и соответствующие типы, если они будут нужны для решения данной задачи, определяются самостоятельно.

Клиентом API может быть любая сущность, например, веб-клиент (приложение для браузера), мобильный клиент (приложение для android), десктоп приложение для операционной системы Windows и т. д. Используемый стек технологий не ограничен, но, желательно, использовать обозначенную выше связку.

**Программный код**

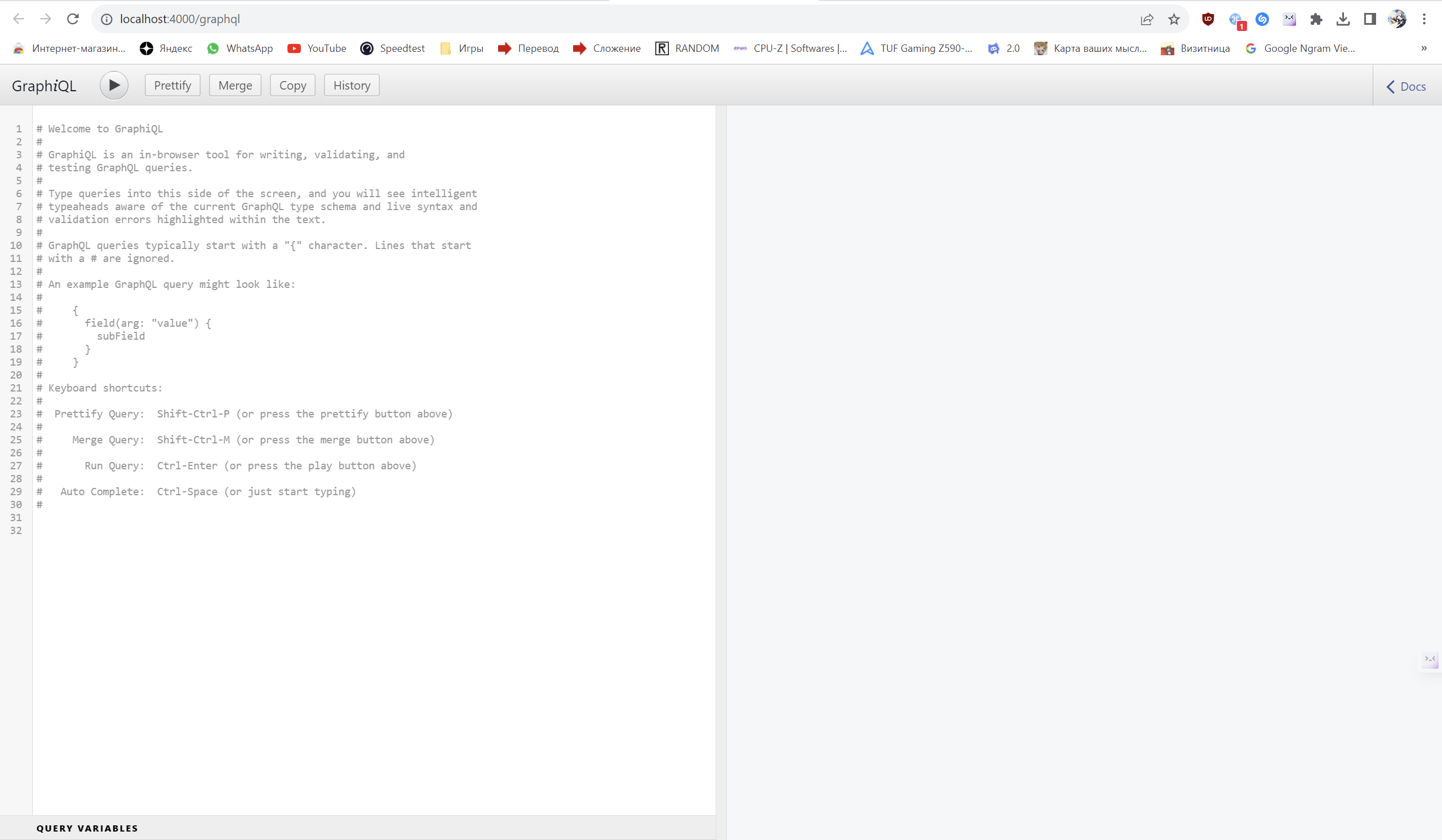


**Рисунок 1 – Создание схемы**

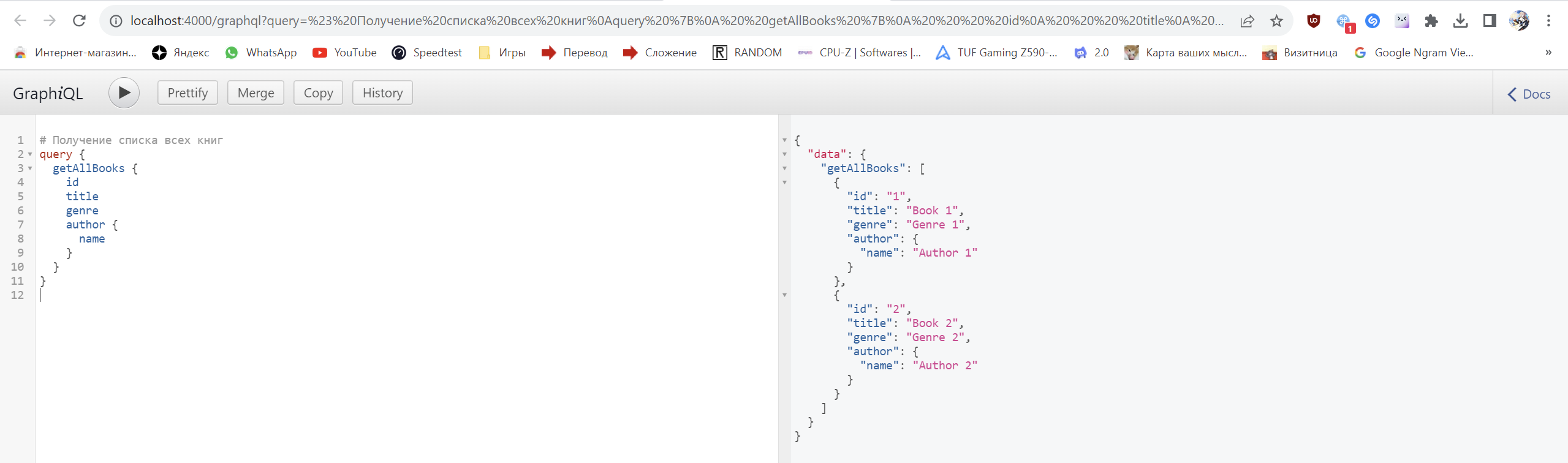


**Рисунок 2 – Server.js**

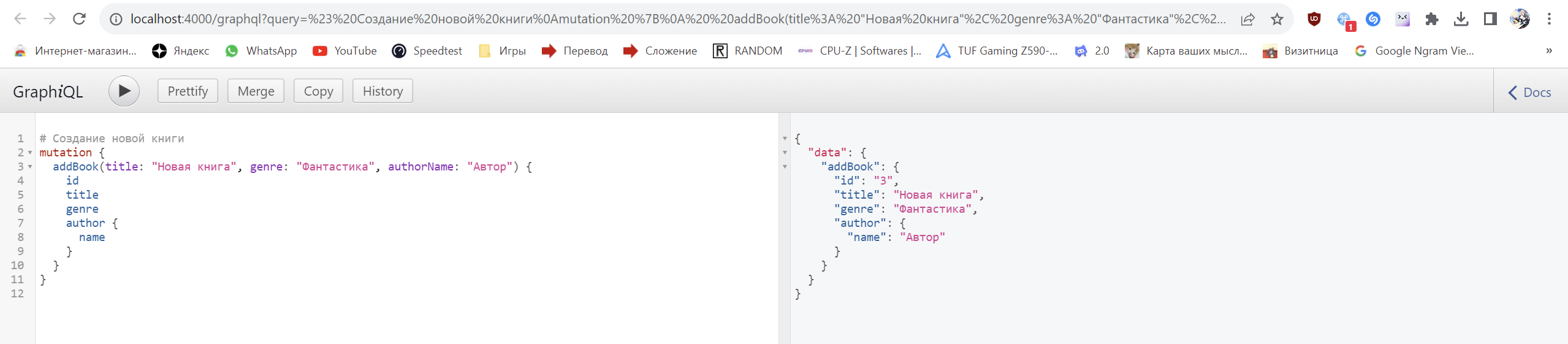
**Вывод программы**



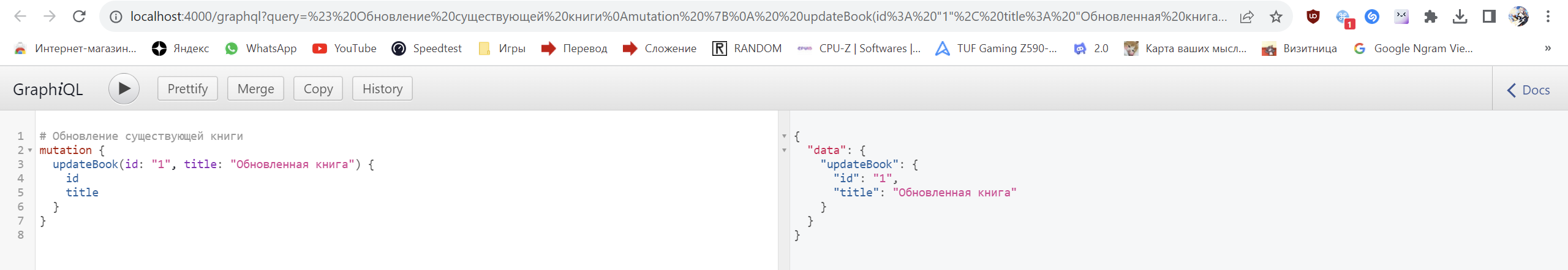
**Рисунок 3 – Запущенный сервер**



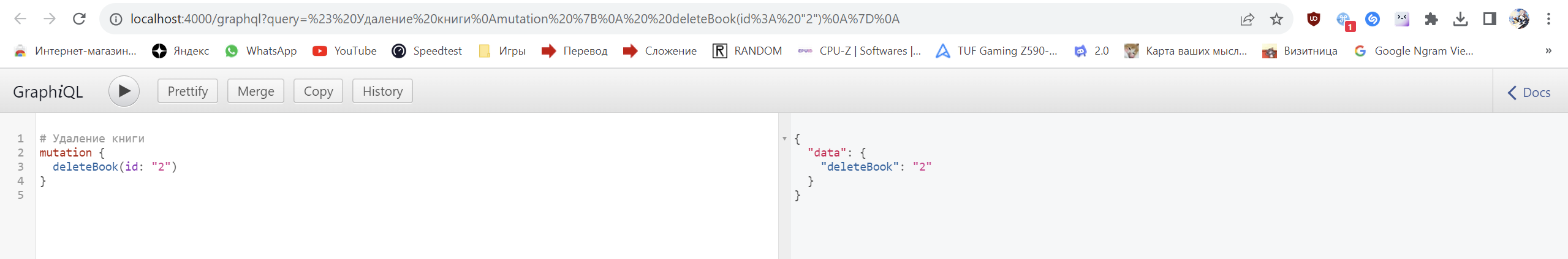
**Рисунок 4 – Список всех книг**



**Рисунок 5 – Создание новой книги**



**Рисунок 6 – Обновление существующей книги**



**Рисунок 7 – Удаление книги**

**Вывод**

В ходе выполнения данной работы было выполнено создание Графовых моделей и с реализующей данный подход технологией на основе спецификации GraphQL.

**Список информационных источников**

1. Конспект лекций по дисциплине «Архитектура клиент-серверных приложений», РТУ МИРЭА (дата обращения 14.10.2023)
2. Документация Java – URL: https://docs.oracle.com/en/java/ (дата обращения 14.10.2023)
3. Курс практик по дисциплине «Архитектура клиент-серверных приложений», РТУ МИРЭА (дата обращения 14.10.2023)