Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский технологический университет «МИСИС»

Направление подготовки 09.03.01
«Информатика и вычислительная техника»

КУРСОВАЯ РАБОТА

По дисциплине:

«Разработка клиент-серверных приложений»

на тему:

«Читательский дневник»

Выполнил:
студент группы БИВТ-21-10
Мальцев Н.А.

Проверил:
ассистент кафедры АСУ
Рзазаде У. А.

Москва
2023

Содержание

1. Введение

1.1. Краткий обзор области

В современных реалиях интернет является интегральной технологией. Именно с помощью неё возможно обеспечить беспрепятственный обмен информацией между людьми. Именно на этом принципе и строятся клиент-серверные приложения: обмен информацией между клиентом и сервером. Благодаря ему возможно обеспечить, например, читателей возможностью хранить свои рецензии на прочитанные произведения, имея возможность вернуться к своим заметкам в любой необходимый момент.

1.2. Актуальность

Актуальность курсовой работы обусловлена тем, что клиент-серверное приложение является крайне удобным инструментом для создания небольших приложений-функций, что открывает возможность для создания удобных инструментов для повышения качества жизни пользователей современных технологий.

1.3. Цель работы

Целью работы является работка серверной и клиентской части клиент-серверного приложения реализующего функции читательского дневника.

2. Постановка задачи

Разработать клиент-серверное приложение с использованием базы данных в соответствии с модифицированным вариантом 20: Книжный склад, жанры, авторы. При этом должны обеспечиваться следующие требования:

- Возможность изменения, добавления и удаления записей для сущностей.
- Функция регистрации новых пользователей с соблюдением уникальности имени пользователя.
- Функция авторизации уже зарегистрированных пользователей.
- Возможность отображать на страницах сайта только ту информацию, к которой данный пользователь имеет доступ.
- Авторизация запросов. Например, простой пользователь не должен иметь возможности как-либо модифицировать записи в сущностях других пользователей.
- Возможность получить информацию о пользователях, зарегистрированных на платформе.

3. Описание архитектуры

3.1. Клиент серверные приложения

Клиент-серверное приложение – это веб-приложение, в котором клиентом выступает браузер, а сервером – веб-сервер (в широком смысле).

Основная часть приложения, как правило, находится на стороне веб-сервера, который обрабатывает полученные запросы в соответствии с бизнес- логикой продукта и формирует ответ, отправляемый пользователю. На этом этапе в работу включается браузер, именно он преобразовывает полученный ответ от сервера в графический интерфейс, понятный пользователю.

Архитектура «клиент-сервер» определяет общие принципы организации взаимодействия в сети, где имеются серверы, узлы-поставщики некоторых специфичных функций (сервисов) и клиенты (потребители этих функций).

Практические реализации такой архитектуры называются клиент-серверными технологиями.

Например, одной из них является двухзвенная архитектура – распределение трех базовых компонентов между двумя узлами (клиентом и сервером). Двухзвенная архитектура, схема которой показана на рисунке 1 используется в клиент-серверных системах, где сервер отвечает на клиентские запросы напрямую и в полном объеме.

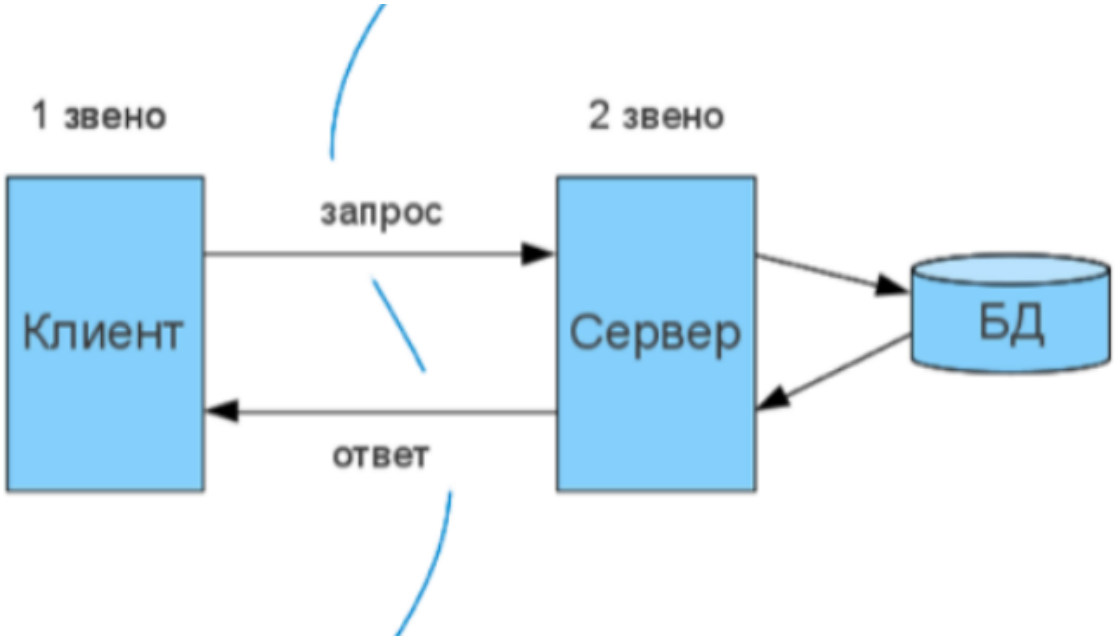

Рис. 1: Схема двузвенной архитектуры.

При этом стоит сразу заметить, что база данных фактически не является частью веб-сервера, но большинство приложений просто не могут выполнять все возложенные на них функции без нее, так как именно в базе данных хранится вся динамическая информация приложения.

База данных – это информационная модель, позволяющая упорядоченно хранить данные об объекте или группе объектов, обладающих набором свойств, которые можно категоризировать. Базы данных функционируют под управлением так называемых систем управления базами данных. Самыми популярными СУБД являются PostgreSQL, Oracle, MongoDB, Redis.

3.2. Выбор технических средств

На основании задач, поставленных передо мной, были использованы следующие технические средства:

- язык программирования - Python.
- веб-фреймворки - Django Framework и Django Rest Framework.
- мененджер зависимостей и сред разработки - Poetry.
- база данных - SQLite.
- сервис Postman и ряд утилит оперционной системы linux (httpie, curl) для тестирования http запросов и имитации работы клиентской части.

3.3. Общее описание архитектуры программной реализации

Серверная часть реализована в виде api(application programming interface), к которому обращается клиент с помощь Http запросов и получает в ответ информацию в формате JSON строк с информацией, регламентированной в документации. Следовательно, вся бизнес логика, т.е. логика работы с данными, инкапсулирована на сервере, а клиентская часть в свою очередь полностью решает проблемы репрезентации данных, полученных с помощью api серверной части.

Для построения архитектуры серверной части использована архитектура MVC (Model View Controller), в роли View на серверной части выступают наборы url, обращаясь к которым клиент будет выстраивать логику реального представления в инкапсулированном модуле. Роль клиента будет играть ряд запросов, сгенерированных сторонним сервисов. Данный паттерн часто возникает в контексте разработки SPA-приложений(Single Page Application). Данный вид приложений часто содежит нагруженную логику представления, работая только с одной корневой страницей, состояние которой изменяется за счёт применения Javascript функций, создающих динамику и реализующих взаимодействие с сервером для получения информации.

4. Описание структуры базы данных

4.1. Вербальная модель

Данная база данных должна обеспечивать возможность свободного обращения как с заметкой о книге, так и составных информационных частях такой заметки. Т.е. должна существует возможность взаимодействовать со всеми элементами заметки по отдельности, чтобы база данных могла обслуживать расширенный функционал связанный с менеджментом информации об авторах, издательствах, жанрах, книгах.

4.2. Реляционная модель

С этой целью были созданы следующие сущности 2:

- Core_Note
  Сущность с информацией о заметке о книге.
- Core_Book
  Сущность с ниформацией о книге.
- Core_Author
  Сущность и информацией об авторе.
- Core_SubGenre
  Сущность с информацией о субжанре. Информация об основном жанре ограничена в виде выбора из вариантов «fiction», т.е. художественная литература и «nonfiction» т.е. документальная или техническая литература.
- Core_PubInfo
  Сущность с информацией об издателе книги.
- Auth_User и связные с ней с префиксом auth
  Сущности, сгенерированные для реализации аутентификации и авторизации пользователей, а также администрирования 3.
- Часть базы данных, которая связанная с сопровождением работы фреймворка и дополнительными возможностями для менеджмента пользователей, функционал которых не был задействован 4.

Сущности связаны следующими соостношениями:

- OneToOne
  Core_Note - Core_Book.
- OneToMany
  Auth_User - Core_Note, Core_SubGenre - Core_Book, Core_PubInfo - Core_Book.
- ManyToMany
  Core_Book - Core_Author.

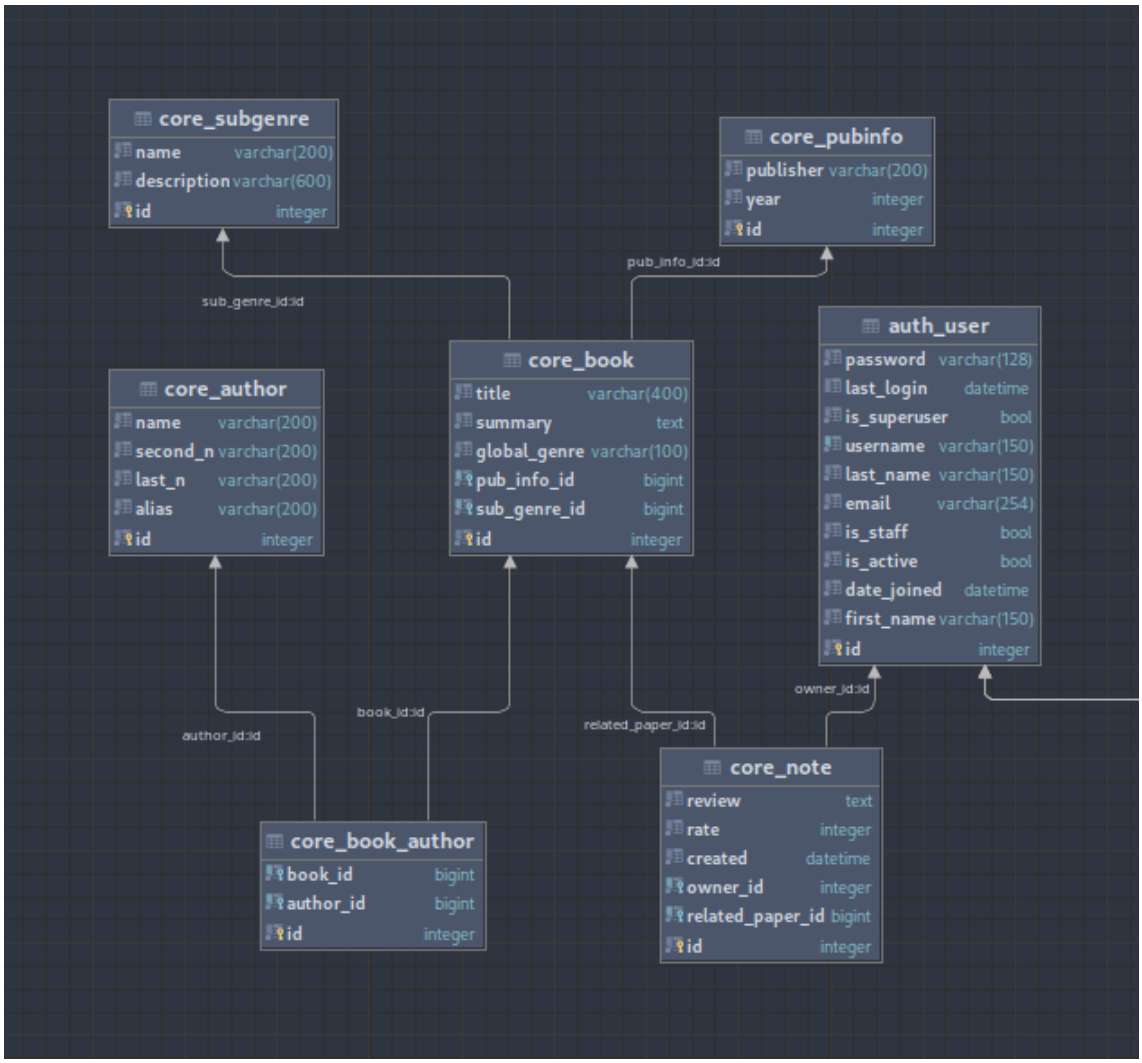

Рис. 2: Часть базы данных, связанная с работой основного функционала приложения.

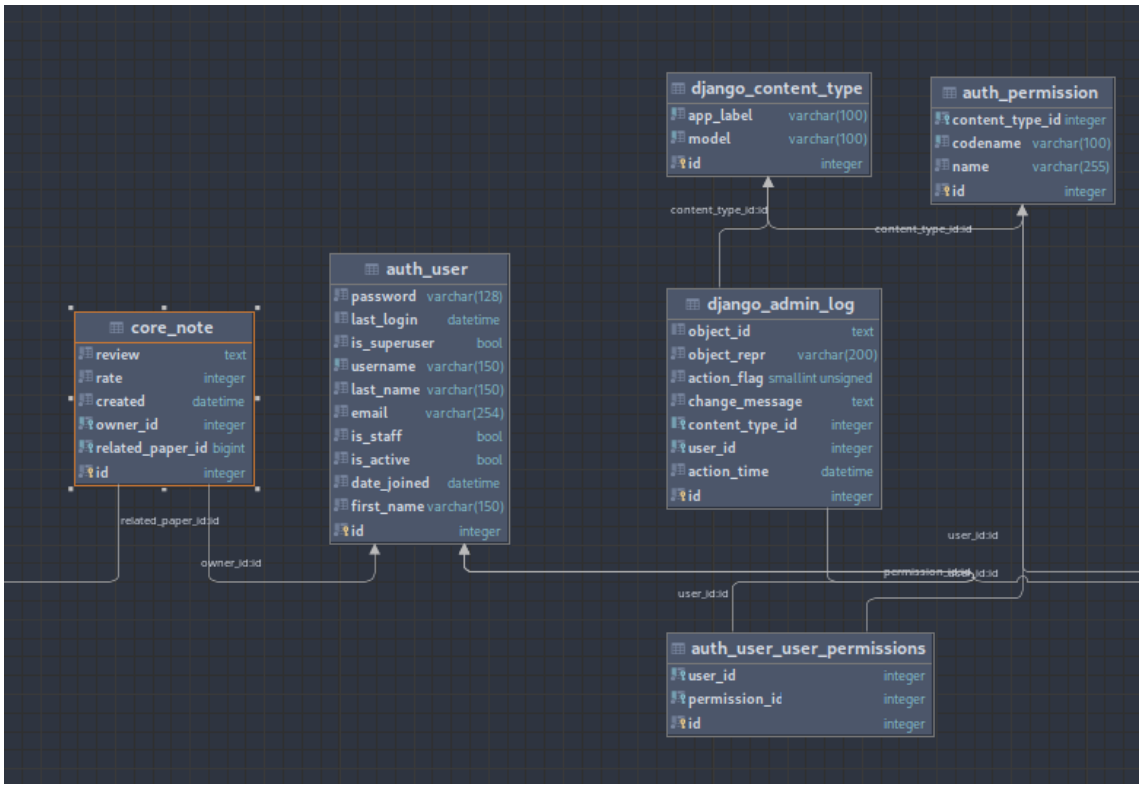

Рис. 3: Часть базы данных, связанная с аутентификацией и авторизацией пользователей.

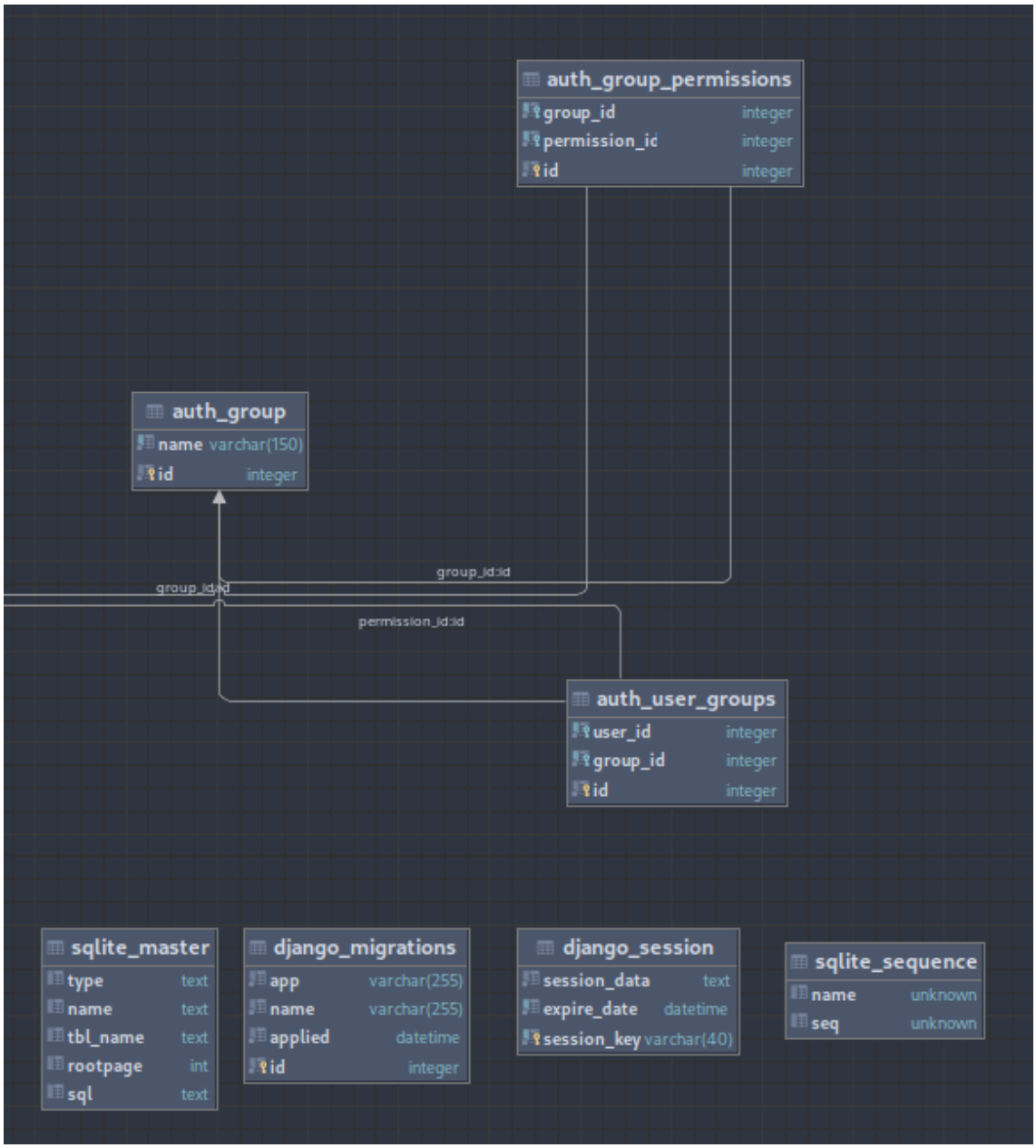

Рис. 4: Часть базы данных, связанная с функциями фреймворка и дополнительным функционалом, который не был задействован в работе.

5. Описание серверной части

Серверная часть была реализована на основе паттерна MVC. Необходимо упомянуть, что трактовка паттерна является свободной, поэтому его реализация может и будет отличаться, в зависимости от используемого языка и фреймворка. В данной работе определение становится ещё более размытым, так как явно заданное View отсутствует и представляет собой api для работы с бизнес логикой.

5.1. Model

В данной части реализовано взаимодействие с базой данных, которое состоит из двух частей:

- Первичное создание сущностей в базе данных и работа с миграциями в случае их модификации.
- Работа с базой данных с помощью ORM
  Object Relation Mapping - абстракция, реализующая взаимодействие с бд без использования сырого SQL

5.1.1. Создание сущностей и миграция

Реализации логики, связанной с созданием сущностей реализовано в файле core/models.py в классах:

- Note
- Book
- Author
- SubGenre
- PubInfo
- User
  генерируется автоматически, в качестве одного из базовых классов для аутентификации и авторизации (Basic Authorization - т.е. используется только имя пользоватея и пароль), механизмы для которых уже реализованы внутри фреймворка. Регистрация же пользователя также реализована с использованием дополнительной зависимости rest_registration.

О логике представления в базе данных:
Сущности Author, SubGenre, PubInfo могут быть свободно модифицированны. В частности, важно, что удаление данных из этих сущностей не повлечёт удаление заметок о книгах (Note) и самих книг (Book), а лишь повлечёт за собой замену в необходимых строках на базовое значение ("no info").

Удаление книги (строка в сущности Book), в свою очередь повлечёт полное удаление заметок об этой книге.

5.1.2. ORM

Взаимодействие с базой данных (CRUD операции) реализована в файле core/view.py (как было упомянуто выше, названия файлов и директорий может варьироваться в зависимости от фреймворка) в классах:

- UserViewSet
- SubGenreViewSet
- PubInfoViewSet
- AuthorViewSet
- BookViewSet
- NoteViewSet

О логике работы ORM: Все классы кроме UserViewSet предоставляют полный набор CRUD операций. Класс UserViewSet даёт возможность для использования только безопасных операций (т.е. операции получения информации, но не изменения). Возможность добавления администраторов реализуется через утилиты CLI django проекта, а добавления обычных пользователей - через процедуру аутентификации.

5.2. Controller и View

В данной части реализован роутинг запросов, т.е. сопоставление запроса и обработчика запроса. Т.е. для модели api - это и контроллер и представление.

Роутинг реализован в файле core/urls.py.

Кроме url для обработки запросов по работе с основными сущностями для работы с заметками о книгах, реализованы end-point'ы дл работы с аккаунтами пользователей(не все возможности для работы с аккаунтом из представленных url(путь ../accounts) доступны, так как они требуют дополнительной конфигурации и коллаборации с внешними сервисами. Например, есть возможность расширить возможности для работы с паролями, например реализовать возможности смены пароля пользователя с помощью почты, но в таком случае необходима реализация взаимодействия сервера с почтовым сервисом. Такая задача в данной курсовой поставлена не была). Также реализовано более удобное представление структуры api, т.е. интерактивная документация для создания запросов к серверу через сервис swagger-like сервис.

6. Описание клиентской части

Клиентская часть реализована в виде обращений к api через эквивалентные варианты - swagger-документацию 5 и сервис postman 6.

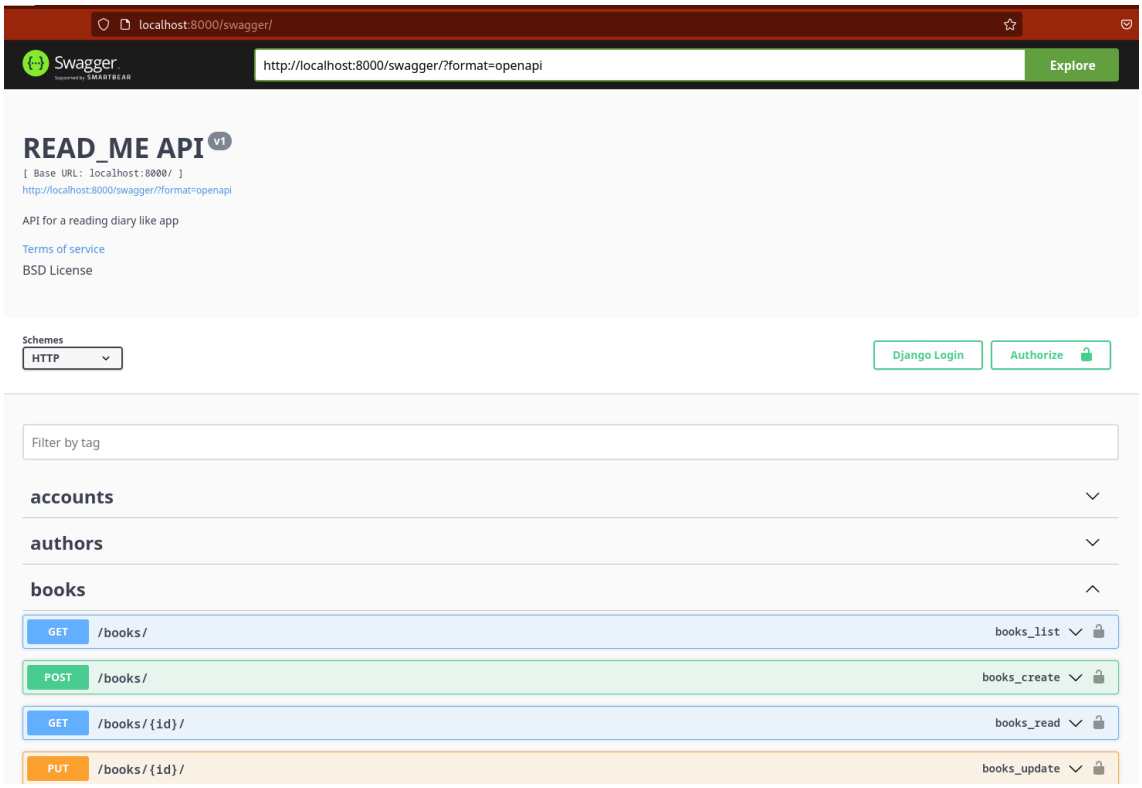

Рис. 5: Работа с запросами через swagger.

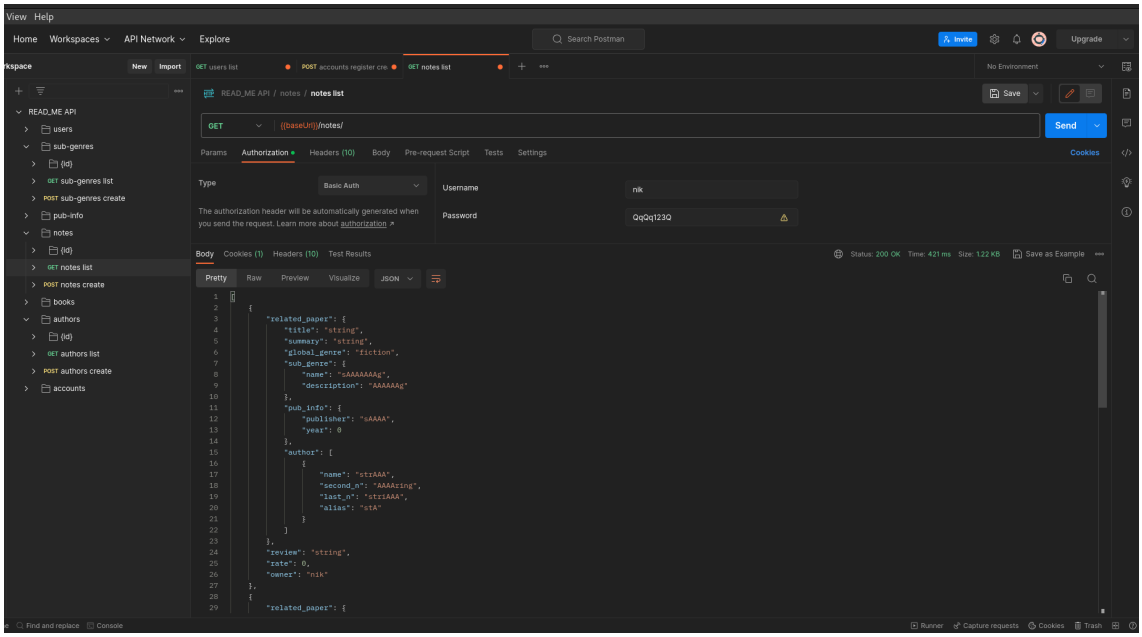

Рис. 6: Работа с запросами через postman.

7. Приложения

Кодовая база располагается по ссылке `https://github.com/mmmhdp/read_me`. Ссылка в формате QR кода 7.

Рис. 7: QR-code

Список литературы

[1] Django Software Foundation. Django documentation. https://django.readthedocs.io/_/downloads/en/3.2.x/pdf/, 2023.

[2] Agiliq. Building apis with django and django rest framework. https://buildmedia.readthedocs.org/media/pdf/djangoapibook/latest/djangoapibook.pdf, 2023.

[3] Meta Open Source. React documentation. https://react.dev/learn, 2023.