

Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,
МЕХАНИКИ И ОПТИКИ”

Факультет Информационных технологий

Образовательная программа 09.03.02

Направление подготовки (специальность) Информационные системы и технологии

О Т Ч Е Т

по курсовой работе

Тема задания: РЕАЛИЗАЦИЯ WEB-СЕРВИСОВ СРЕДСТВАМИ Django REST framework

Обучающийся Бакирова Шоола Даулетбековна, К3440

Руководитель: Говоров А. И.

Практика пройдена с оценкой ____

Подписи членов комиссии:

(подпись)

(подпись)

(подпись)

Дата ____

Санкт-Петербург
2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ	4
1.1. Описание предметной области	4
1.2. Описание функциональных требований	4
2. ОПИСАНИЕ СЕРВЕРНОЙ ЧАСТИ	6
2.1. Технологии создания серверной части	6
2.2. Создание базы данных и моделей	6
2.3. Создание отображений	8
2.4. Реализованные интерфейсы в панели Django REST	8
3. ОПИСАНИЕ КЛИЕНТСКОЙ ЧАСТИ	15
3.1. Технологии создания клиентской части	15
3.2. Реализованные интерфейсы	15
4. КОНТЕЙНЕРИЗАЦИЯ	23
ЗАКЛЮЧЕНИЕ	26
СПИСОК ЛИТЕРАТУРЫ	27
Приложение 1. Содержание файла models.py	28
Приложение 2. Содержимое файла serializers.py	30
Приложение 3. Содержание файла views.py	32

ВВЕДЕНИЕ

В качестве варианта курсовой работы было выбрано задание, по которому создать программную систему, предназначенную для работников библиотеки. Такая система должна обеспечивать хранение сведений об имеющихся в библиотеке книгах, о читателях библиотеки и читальных залах.

Для каждой книги в БД должны храниться следующие сведения: название книги, автор(ы), издательство, год издания, раздел, число экземпляров этой книги в каждом зале библиотеки, а также шифр книги и дата закрепления книги за читателем. Сведения о читателях библиотеки должны включать номер читательского билета, ФИО читателя, номер паспорта, дату рождения, адрес, номер телефона, образование, наличие ученой степени.

Читатели закрепляются за определенным залом и могут записываться и выписываться из библиотеки. Библиотека имеет несколько читальных залов, которые характеризуются номером, названием и вместимостью, то есть количеством людей, которые могут одновременно работать в зале. Библиотека может получать новые книги и списывать старые. Шифр книги может измениться в результате переклассификации, а номер читательского билета в результате перерегистрации.

Библиотекаря могут потребоваться следующие сведения о текущем состоянии библиотеки:

- Какие книги закреплены за определенным читателем?
- Кто из читателей взял книгу более месяца тому назад?
- За кем из читателей закреплены книги, количество экземпляров которых в библиотеке не превышает 2?
- Сколько в библиотеке читателей младше 20 лет?
- Сколько читателей в процентном отношении имеют начальное образование, среднее, высшее, ученую степень?

Библиотекарь может выполнять следующие операции:

- Записать в библиотеку нового читателя.
- Исключить из списка читателей людей, записавшихся в библиотеку более года назад и не прошедших перерегистрацию.
- Списывать старую или потерянную книгу.

1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

1.1. Описание предметной области

Предметной областью для данной курсовой работы является администрирование библиотеки. Основным пользователем проектируемой системы - работник библиотеки.

В его задачи входит:

- Регистрация новых читателей
 - Исключение читателя из библиотеки
- Принятие книг в фонд библиотеки
- Списание старых и потерянных книг из фонда библиотеки
- Изменение данных читателя, в том числе читательского билета, в случае перерегистрации
- Изменение данных книги, а также шифра книги в случае переклассификации
- Учет книг за читателями
- Добавление читателей в зал библиотеки

Кроме того, работнику библиотеки может понадобиться следующая информация:

- Какие книги закреплены за определенным читателем?
- Кто из читателей взял книгу более месяца тому назад?
- Сколько читателей в процентном отношении имеют начальное образование, среднее, высшее, ученую степень?
- За кем из читателей закреплены книги, количество экземпляров которых в библиотеке не превышает 2?
- Сколько в библиотеке читателей младше 20 лет?

В библиотеке существует несколько читальных залов, каждый из которых имеет название и вмещает определенное количество человек. Читатель может, как взять книги с собой, так и читать их в одном из залов библиотеки.

1.2. Описание функциональных требований

Исходя из описания предметной области, разрабатываемое web-приложение должно отвечать всем, приведенным выше, требованиям и запросам. Все функциональные требования могут быть разделены на несколько групп по виду взаимодействия с базой данных, согласно CRUD (Create-Read-Update-Delete) операциям, а именно: функции добавления информации, функции просмотра информации, функции модификации информации, функции удаления информации. Отдельно следует выделить такие функциональные требования, как авторизация пользователя, регистрация нового пользователя, выполнение запросов.

Следовательно, были определены функциональные требования:

- Функции добавления (Create):
 - Принятие в фонд библиотеки новой книги
 - Регистрация новых читателей
 - Запись определенной книги за определенным читателем
 - Запись читателя в зал библиотеки
- Функции просмотра (Read)
 - Список всех читателей, зарегистрированных в библиотеке
 - Список всех, имеющих в библиотеке, книг
 - Список читальных залов с информацией о них
 - Список читателей, находящихся в конкретном зале
 - Подробная информация об определенном читателе
 - Подробная информация об определенной книге
 - Учет всех книг
- Функции модификации (Update)
 - Изменение данных о читателе
 - Изменение данных книги
 - Изменение закрепление книги
- Функции удаления (Delete)
 - Исключение читателя
 - Списание книги
- Функция авторизации работника библиотеки
- Функции выполнения запросов

2. ОПИСАНИЕ СЕРВЕРНОЙ ЧАСТИ

2.1. Технологии создания серверной части

Для разработки серверной части web-приложения был использован следующий стек технологий:

PostgreSQL 13 – свободная объектно-реляционная система управления базами данных. Благодаря свободной лицензии и открытому коду, PostgreSQL разрешается использовать бесплатно, изменять и распространять всем и для любых целей: личных, коммерческих или учебных. В силу того, что хранение базы данных происходит на сервере, а не локально, переход web-приложения со стадии разработки на стадию продакшена происходит быстрее и проще.

Django 2 – свободный фреймворк для web-приложений на языке программирования Python, использующий шаблон проектирования MVC (Model-View-Controller). Контроллер классической модели MVC примерно соответствует уровню, который в Django называется Представление (англ. View), а презентационная логика Представления реализуется в Django уровнем Шаблонов (англ. Template). Из-за этого уровневую архитектуру Django часто называют «Модель-Шаблон-Представление» (MTV). Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других, например, Ruby on Rails.

Django REST framework – удобный инструмент для работы с REST основанный на идеологии фреймворка Django. Rest (сокр. англ. Representational State Transfer, «передача состояния представления») — стиль построения архитектуры распределенного приложения. Данные в REST должны передаваться в виде небольшого количества стандартных форматов (например HTML, XML, JSON). Сетевой протокол, как и HTTP, должен поддерживать кэширование, не должен зависеть от сетевого слоя, не должен сохранять информацию о состоянии между парами «запрос-ответ». Утверждается, что такой подход обеспечивает масштабируемость системы и позволяет ей эволюционировать с новыми требованиями.

2.2. Создание базы данных и моделей

Согласно описанию предметной области и выявленным функциональным требованиям, была разработана следующая схема базы данных, представленная на рисунке 1.

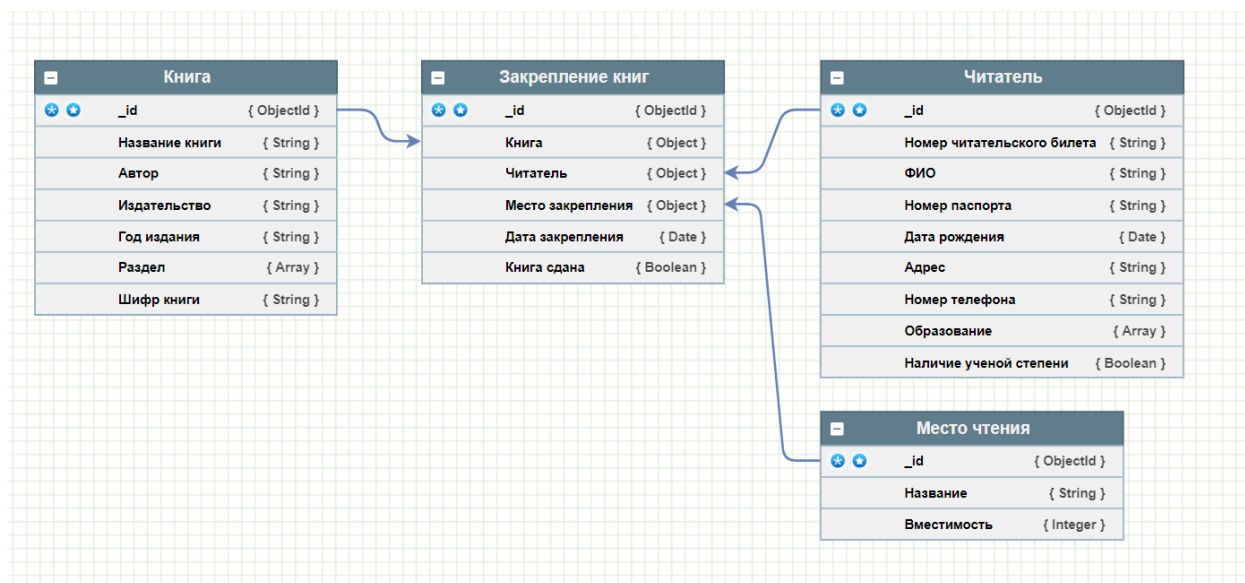


Рисунок 1 – схема базы данных

Полученная схема данных содержит 3 стержневые сущности – Книга, Читатель, Место чтения и 1 ассоциативную сущность – Закрепление книг. Данные сущности были реализованы в качестве классов моделей Django в файле `models.py`, листинг кода которого приведен в приложении 1.

По итогу были разработаны следующие классы моделей:

- Book (Книга) со следующими атрибутами:
 - название книги,
 - автор,
 - издательство,
 - год издания,
 - раздел с выбором параметра: художественная, учебная, психология, детские;
- Readers (Читатель) со следующими атрибутами:
 - номер читательского билета,
 - ФИО,
 - номер паспорта,
 - дата рождения,
 - адрес,
 - номер телефона,
 - образование с выбором параметра: дошкольное, начальное общее, основное общее, среднее общее, неполное общее, высшее,
 - наличие ученой степени,
- Place (Место чтения) со следующими атрибутами:
 - название,
 - вместимость,
- Fix (Закрепление книг) со следующими атрибутами:
 - книга,
 - читатель,
 - место закрепления,

- дата закрепления,
- книга сдана;

2.3. Создание отображений

Для того, чтобы обеспечить обмен данными между сервером, написанным на Django, и клиентской частью, написанной на Vue.js, необходимо провести сериализацию. Сериализация – процесс преобразования структур данных или объекта состояния в формат, который может быть сохранен или передан и реконструирован позже. Среда сериализации в Django предоставляет механизм для «перевода» моделей Django в другие форматы, например, в JSON, который принимает Vue.js.

Все сериализаторы для созданных моделей описаны в файле `serializers.py`, приведенном в приложении 2. Далее были созданы отображения, описанные в файле `views.py`, приведенном в приложении 3.

2.4. Реализованные интерфейсы в панели Django REST

Далее приведены некоторые из реализованных интерфейсов в панели Django REST:

- Для модели Читатель:
 - Отображение всех читателей. Скриншот приведен на рисунке 2.

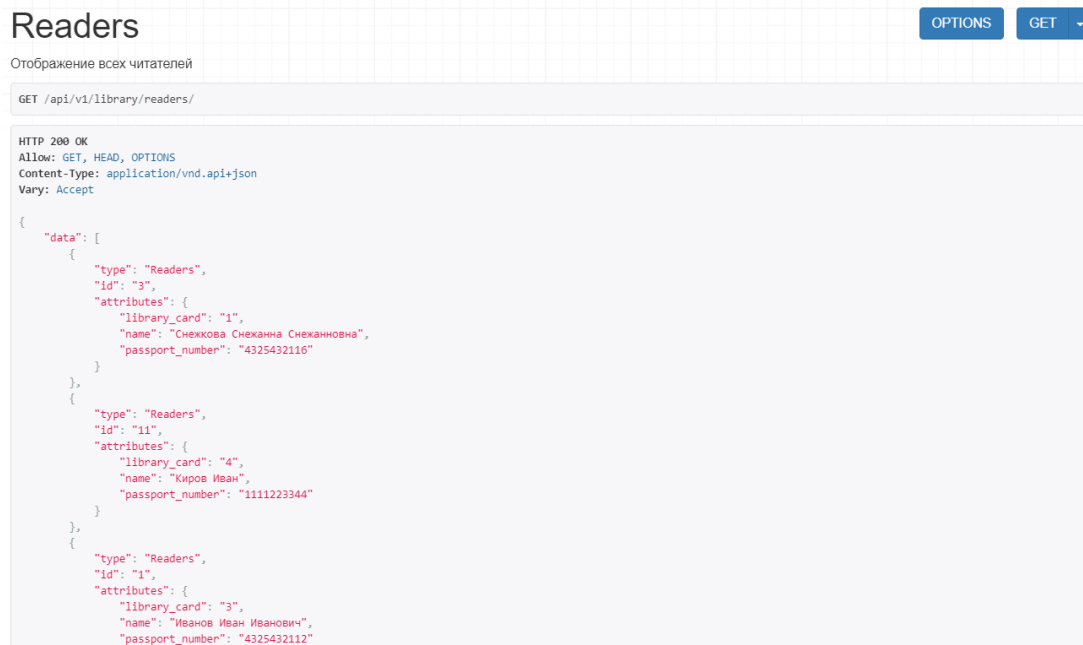


Рисунок 2

- Отображение одного читателя и все взятые им книги на данный момент. Скриншот приведен на рисунке 3.

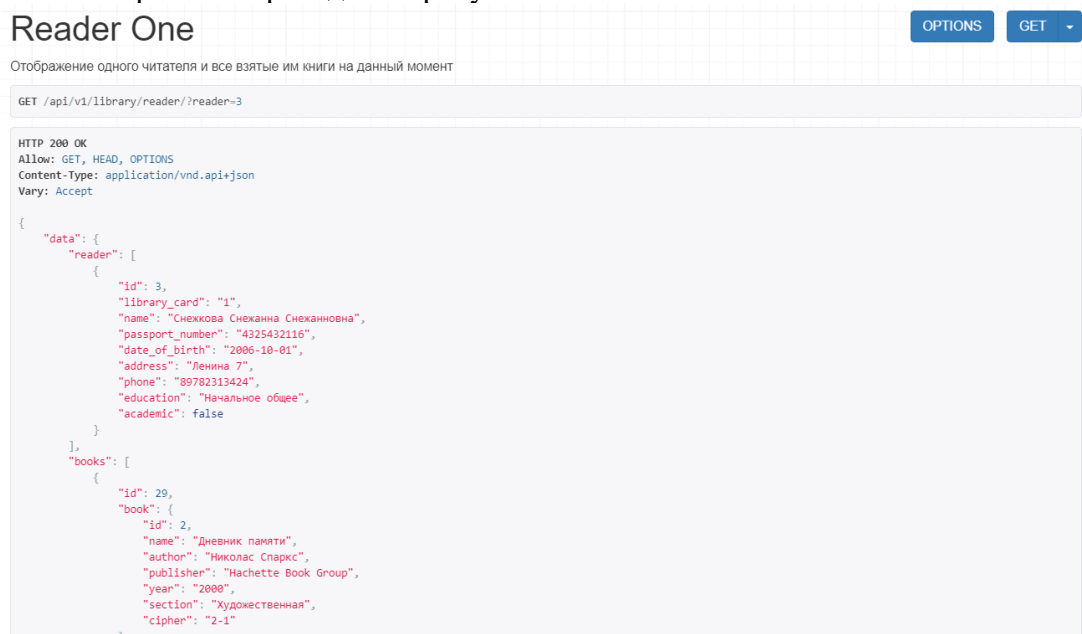


Рисунок 3.

- Редактирование модели Читатель. Скриншот приведен на рисунке 4.

Reader Del Upd OPTIONS

Отображение для модели Взятие книги

GET /api/v1/library/reader_update/1/

```

HTTP 405 Method Not Allowed
Allow: POST, OPTIONS
Content-Type: application/vnd.api+json
Vary: Accept

{
  "errors": [
    {
      "detail": "Метод \"GET\" не разрешен.",
      "source": {
        "pointer": "/data"
      },
      "status": "405"
    }
  ]
}

```

Raw data HTML form

Номер читательского билета

ФИО

Номер паспорта

Дата рождения

Адрес

Номер телефона

Образование

Наличие ученой степени ☐

POST

Рисунок 4

- Для модели Книга:
 - Отображение всех читателей. Скриншот приведен на рисунке 5.

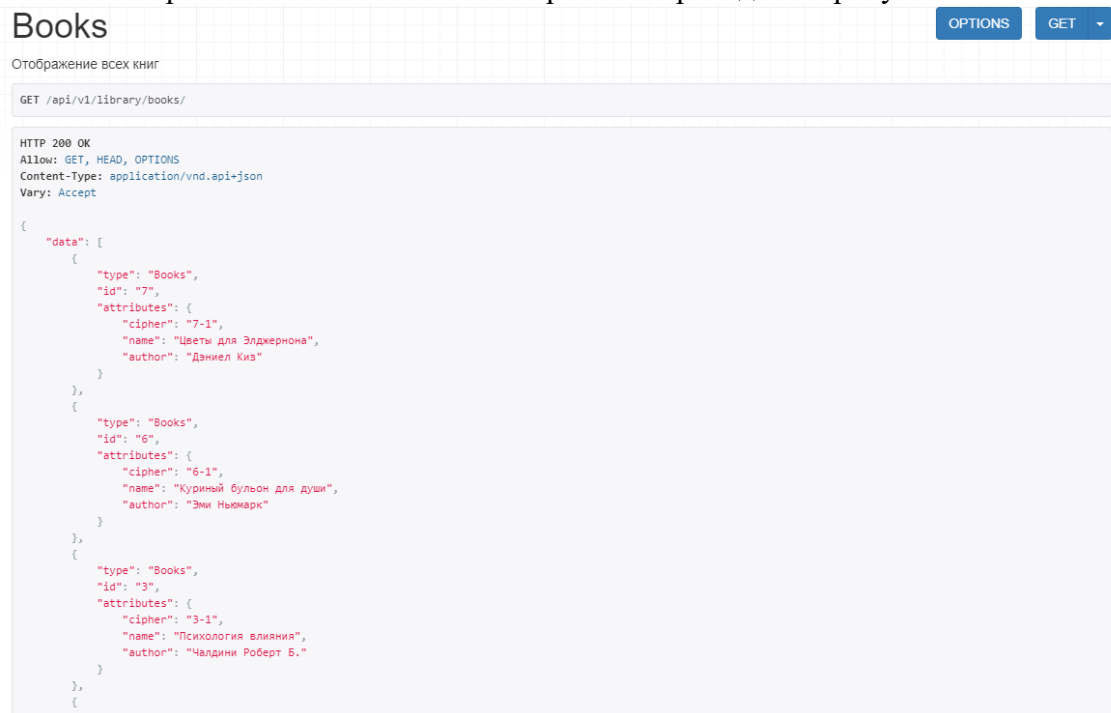


Рисунок 5.

- Отображение одной книги. Скриншот приведен на рисунке 6.

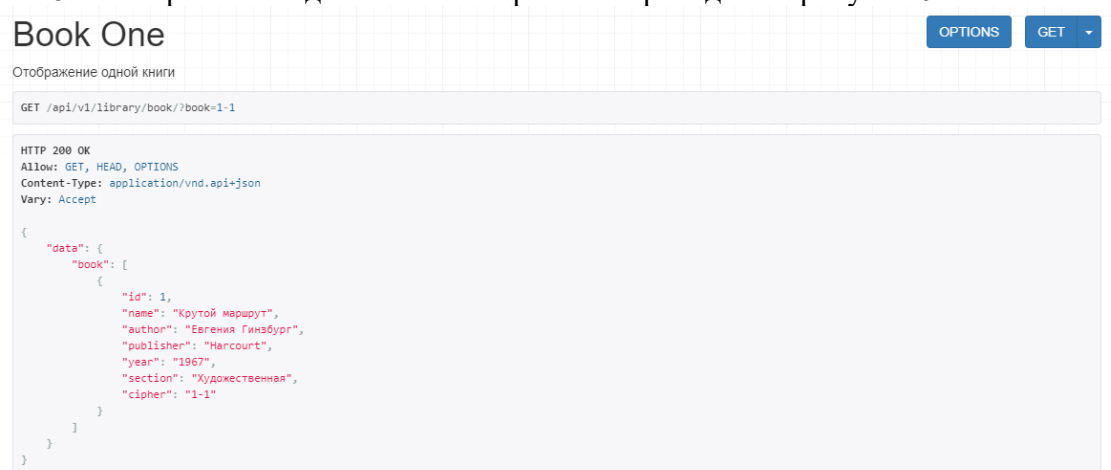


Рисунок 6.

- Редактирование модели Книга. Скриншот приведен на рисунке 7.

Book Del Upd

Отображение для модели Взятие книги

GET /api/v1/library/book_update/1/

HTTP 405 Method Not Allowed

Allow: POST, OPTIONS

Content-Type: application/vnd.api+json

Vary: Accept

```
{
  "errors": [
    {
      "detail": "Метод \"GET\" не разрешен.",
      "source": {
        "pointer": "/data"
      },
      "status": "405"
    }
  ]
}
```

Raw data

HTML form

Название книги

Автор

Издательство

Год издания

Раздел

Шифр книги

Художественная литература

POST

Рисунок 7.

- Для модели Место чтения
 - Отображение всех мест для чтения. Скриншот приведен на рисунке 8.

Places

Места

GET /api/v1/library/places/

HTTP 200 OK

Allow: GET, HEAD, OPTIONS

Content-Type: application/vnd.api+json

Vary: Accept

```
{
  "data": {
    "place": [
      {
        "id": 2,
        "name": "С собой",
        "capacity": null
      },
      {
        "id": 5,
        "name": "Взрослый зал",
        "capacity": 30
      },
      {
        "id": 3,
        "name": "Детский зал",
        "capacity": 10
      },
      {
        "id": 4,
        "name": "Школьный зал",
        "capacity": 20
      }
    ]
  }
}
```

OPTIONS

GET

Рисунок 8.

- Отображение одного зала библиотеки со всеми зафиксированными в нем читателями и книгами на данный момент. Скриншот приведен на рисунке 9.

Place One

OPTIONS GET ▾

Отображение одного зала и все закрепления в нем

GET /api/v1/library/place/?place=3

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/vnd.api+json
Vary: Accept

{
  "data": {
    "place": [
      {
        "id": 3,
        "name": "Детский зал",
        "capacity": 10
      }
    ],
    "fixes": [
      {
        "id": 43,
        "book": {
          "id": 10,
          "cipher": "1-2",
          "name": "Крутой маршут",
          "author": "Евгения Гинабург"
        },
        "reader": {
          "id": 2,
          "library_card": "2",
          "name": "Снежкова Снежана Олеговна",
          "passport_number": "4325431113"
        },
        "date_fix": "2020-10-28",
        "place": 3
      },
      {
        "id": 44,
        "book": {
          "id": 11,
          "cipher": "1-3",
          "name": "Крутой маршут",
          "author": "Евгения Гинабург"
        },
        "reader": {
          "id": 11,
          "library_card": "4",
          "name": "Киров Иван",
          "passport_number": "1111223344"
        },
        "date_fix": "2020-10-28",
        "place": 3
      }
    ]
  }
}
```

Рисунок 9.

- Для модели Закрепления
 - Отображение всех закреплений. Скриншот приведен на рисунке 10.

Fixes

OPTIONS GET ▾

Закрепления

GET /api/v1/library/fix/

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/vnd.api+json
Vary: Accept

{
  "data": {
    "data": [
      {
        "id": 14,
        "book": {
          "id": 1,
          "cipher": "1-1",
          "name": "Крутой маршут",
          "author": "Евгения Гинабург"
        },
        "reader": {
          "id": 2,
          "library_card": "2",
          "name": "Снежкова Снежана Олеговна",
          "passport_number": "4325431113"
        },
        "date_fix": "2020-09-09",
        "place": 2
      },
      {
        "id": 7,
        "book": {
          "id": 6,
          "cipher": "6-1",
          "name": "Хрустальный буллон для души",
          "author": "Эми Ньюмарк"
        },
        "reader": {
          "id": 1,
          "library_card": "3",
          "name": "Иванов Иван Иванович",
          "passport_number": "4325431112"
        },
        "date_fix": "2020-10-11",
        "place": 2
      },
      {
        "id": 42,
        "book": {
          "id": 3,
          "cipher": "3-1",
          "name": "Психология влюбленности",
          "author": "Мадлен Роберт Б."
        },
        "reader": {
          "id": 1,
          "library_card": "3",
          "name": "Иванов Иван Иванович",
          "passport_number": "4325431112"
        },
        "date_fix": "2020-10-11",
        "place": 2
      }
    ]
  }
}
```

Рисунок 10.

- Редактирование модели Закрепление. Фиксирование открепления книги от читателя. Скриншот приведен на рисунке 11.

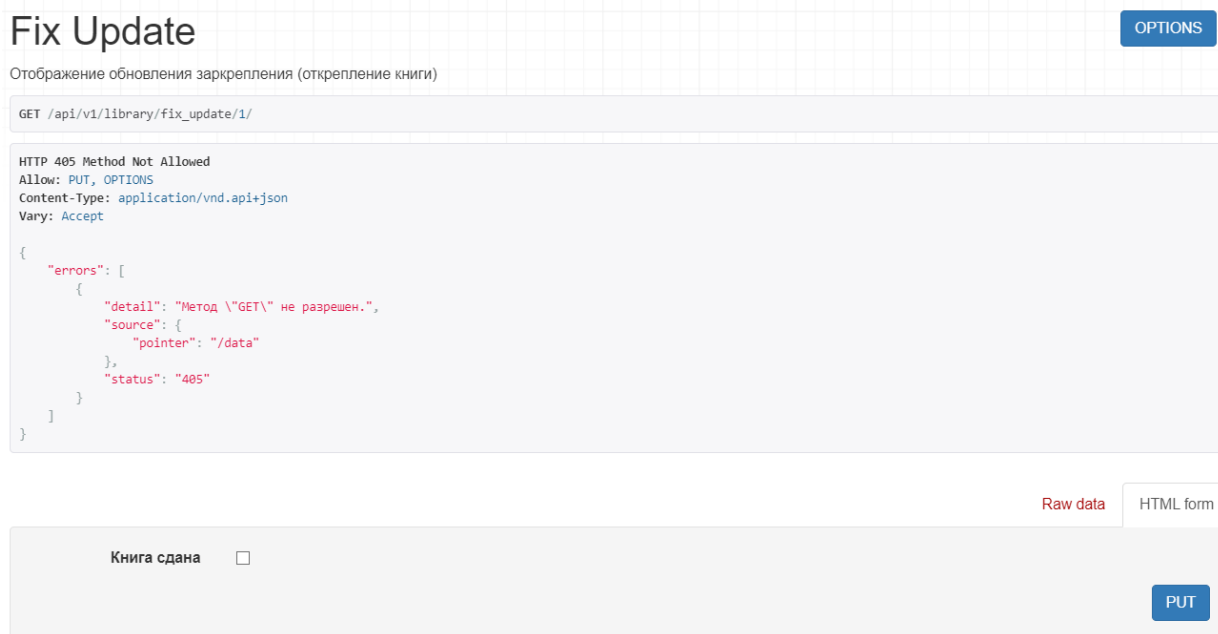


Рисунок 11.

- Отображение для запросов, которые могут понадобиться администратору библиотеки. Скриншоты запросов приведены на рисунках 12, 13 и 14.

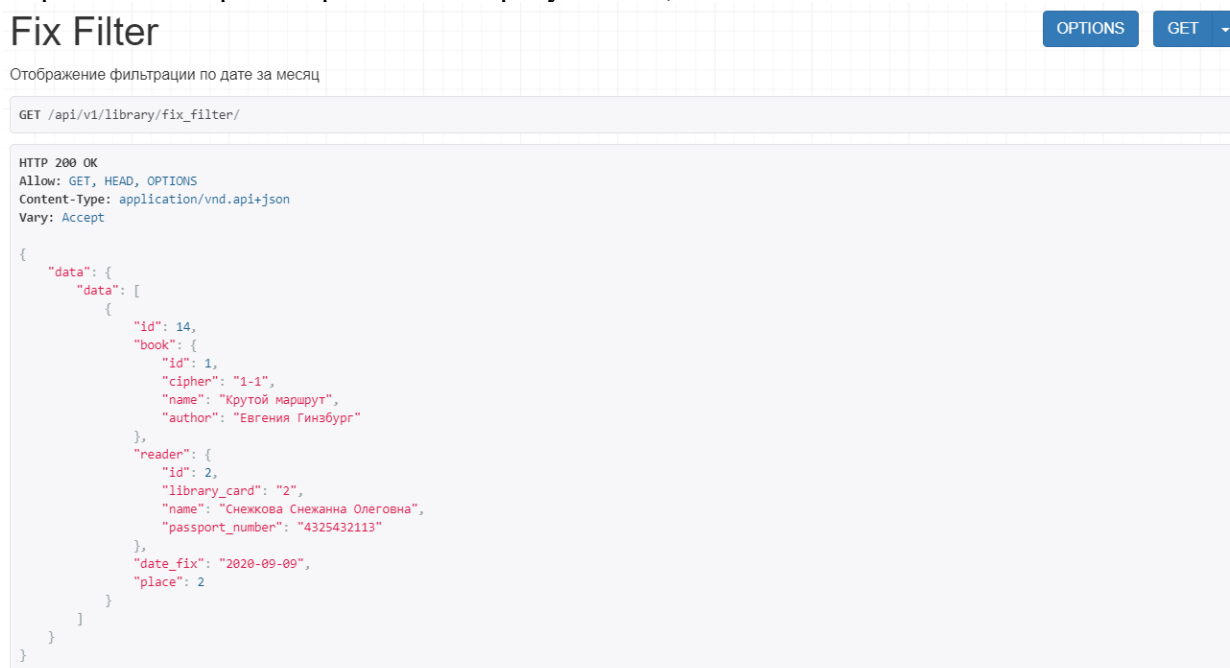


Рисунок 12 – отображение фильтрации по дате за месяц

Readers Filter

[OPTIONS](#)[GET](#) ▾

Отображение фильтрации по дате рождения (меньше 20 лет)

GET /api/v1/library/readers_filter/

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/vnd.api+json
Vary: Accept

{
  "data": {
    "data": [
      {
        "id": 3,
        "library_card": "1",
        "name": "Снежкова Снежанна Снежанновна",
        "date_of_birth": "2006-10-01"
      },
      {
        "id": 1,
        "library_card": "3",
        "name": "Иванов Иван Иванович",
        "date_of_birth": "2020-10-06"
      },
      {
        "id": 2,
        "library_card": "2",
        "name": "Снежкова Снежанна Олеговна",
        "date_of_birth": "2020-10-01"
      }
    ]
  }
}
```

Рисунок 13 – отображение фильтрации по дате рождения (меньше 20 лет).

Education Filer

[OPTIONS](#)[GET](#) ▾

Сколько читателей в процентном отношении имеют начальное образование, среднее, высшее, ученую степень?

GET /api/v1/library/education/

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/vnd.api+json
Vary: Accept

{
  "data": {
    "primary": 25.0,
    "secondary": 0.0,
    "high": 75.0,
    "academic": 25.0
  }
}
```

Рисунок 14 – процентное отношение читателей, которые имеют определенное образование, от всего количества читателей

3. ОПИСАНИЕ КЛИЕНТСКОЙ ЧАСТИ

3.1. Технологии создания клиентской части

Для создания клиентской части web-приложения были использованы следующие технологии:

- Vue.js – это прогрессивный фреймворк для создания пользовательских интерфейсов. В отличие от фреймворков-монолитов Vue создан пригодным для постепенного внедрения. Его ядро в первую очередь решает задачи уровня представления (view), что упрощает интеграцию с другими библиотеками и существующими проектами. С другой стороны, Vue полностью подходит и для создания сложных одностраничных приложений (SPA, Single-Page Applications), если использовать его совместно с современными инструментами и дополнительными библиотеками.
- Muse-UI – библиотека Vue.js, основанная на библиотеке компонентов Material Design

3.2. Реализованные интерфейсы

Согласно функциональным требованиям, были созданы пользовательские интерфейсы. Далее приведены все имеющиеся интерфейсы с учетом выявленных выше функциональных требований:

- Интерфейс стартовой страницы, содержащая задание варианта работы и вход для неавторизованного (Рисунок 15) и авторизованного (Рисунок 16) пользователя. В самом задании имеются вопросы, ссылки которых введут на ответы в виде всплывающего диалога, либо на страницы, где запрос реализован.

Библиотека ВХОД

Задание 2

Создать программную систему, предназначенную для работников библиотеки. Такая система должна обеспечивать хранение сведений об имеющихся в библиотеке книгах, о читателях библиотеки и читальных залах.

Для каждой книги в БД должны храниться следующие сведения: название книги, автор (ы), издательство, год издания, раздел, число экземпляров этой книги в каждом зале библиотеки, а также шифр книги и дата закрепления книги за читателем. Сведения о читателях библиотеки должны включать номер читательского билета, ФИО читателя, номер паспорта, дату рождения, адрес, номер телефона, образование, наличие ученой степени.

Читатели закрепляются за определенным залом и могут записываться и выписываться из библиотеки. Библиотека имеет несколько читальных залов, которые характеризуются номером, названием и вместимостью, то есть количеством людей, которые могут одновременно работать в зале. Библиотека может получать новые книги и списывать старые. Шифр книги может измениться в результате переклассификации, а номер читательского билета в результате перерегистрации.

Библиотекарию могут потребоваться следующие сведения о текущем состоянии библиотеки:

- Какие книги закреплены за определенным читателем?
- Кто из читателей взял книгу более месяца тому назад?
- За кем из читателей закреплены книги, количество экземпляров которых в библиотеке не превышает 2?
- Сколько в библиотеке читателей младше 20 лет?
- Сколько читателей в процентном отношении имеют начальное образование, среднее, высшее, ученую степень?

Рисунок 15 – стартовая страница для неавторизованного пользователя

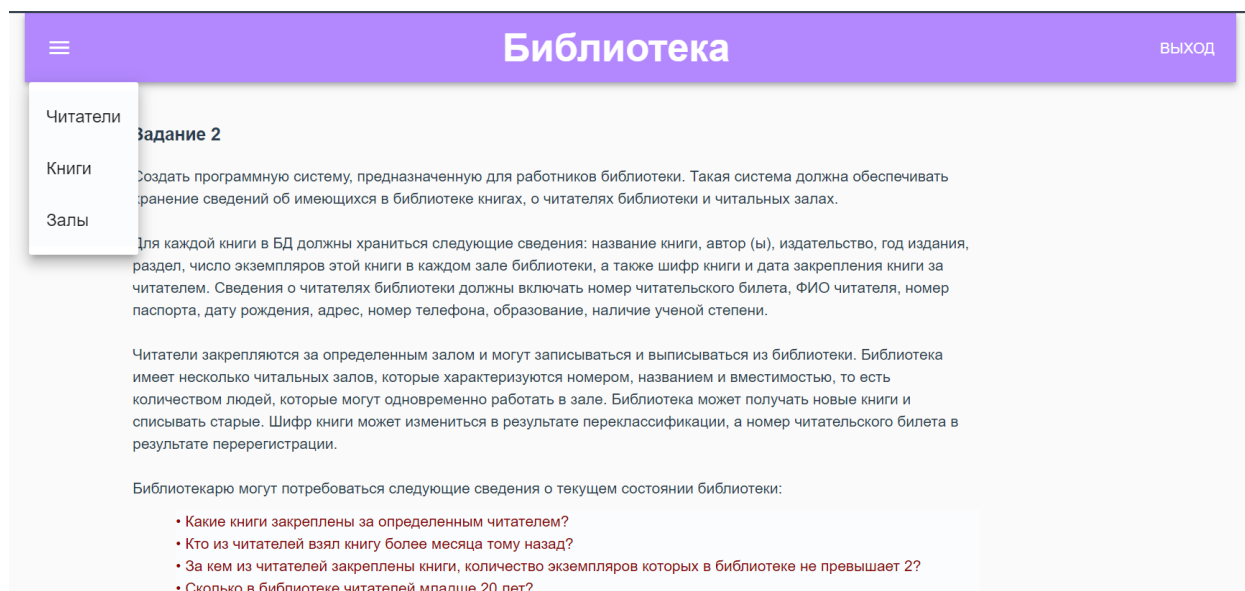


Рисунок 16 - стартовая страница для неавторизованного пользователя

- Интерфейс для вывода всех читателей, зарегистрированных в библиотеке. Кликая по выбранному читателю или вводя его читательский билет/паспортные данные в поиск, выводится полная информация о нем и закрепленные за ним книги. Скриншот представлен на рисунке 17. Также имеется возможность редактирования данных – кнопка добавление, при клике на которую открывается окно с формой добавления читателя (Рисунок 18); кнопка изменить - открывается окно с формой изменения данных читателя (Рисунок 19); кнопка удалить – удаляет читателя из библиотеки; кнопка открепить, при нажатии на которую в модели Закрепление значение атрибута Книга сдана из False меняется на True; кнопка закрепить книгу - открывается окно с формой закрепления книги за читателем (Рисунок 20).

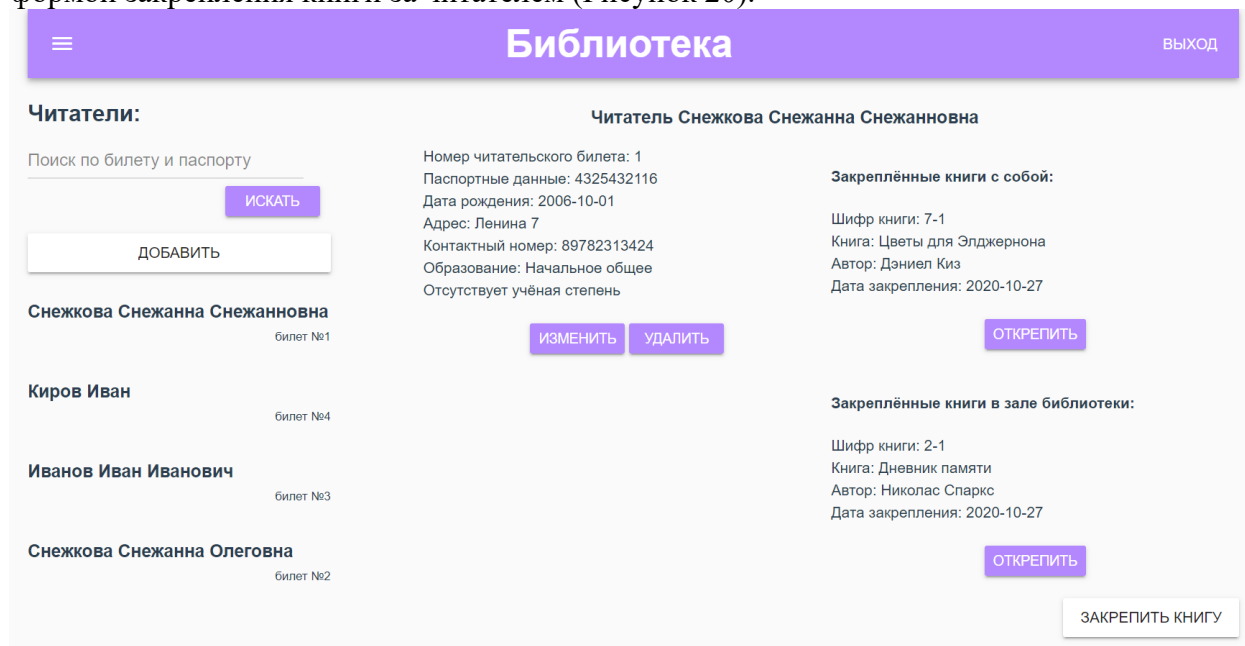


Рисунок 17 – интерфейс с выводом всех читателей и полной информации по выбранному читателю

The screenshot shows a web application interface with a purple header and a sidebar on the left. The sidebar contains a menu icon, the title 'Читатели', a search bar, and a list of names: 'Снежкова С', 'Киров Иван', 'Иванов Ива', and 'Снежкова С'. The main content area is a modal window titled 'Добавление читателя'. It contains two columns of form fields: 'Номер читательского билета', 'Адрес', 'ФИО', 'Контактный номер', 'Паспортные данные', 'Образование', 'Дата рождения', and 'Наличие учёной степени'. The 'Образование' and 'Наличие учёной степени' fields are dropdown menus. At the bottom right of the modal are two buttons: 'ДОБАВИТЬ' (blue) and 'ЗАКРЫТЬ' (purple). The background shows a 'ВЫХОД' button in the top right and a 'ПОИСК КНИГУ' button in the bottom right.

Добавление читателя

Номер читательского билета

Адрес

ФИО

Контактный номер

Паспортные данные

Образование

Дата рождения

Наличие учёной степени

ДД-ММ-ГГГГ

ДОБАВИТЬ

ЗАКРЫТЬ

Рисунок 18 – добавления нового читателя

The screenshot shows the same web application interface as Figure 18. The modal window is titled 'Изменение данных'. It contains the same form fields as Figure 18, but with the instruction 'Введите те данные, которые хотите изменить, в соответствующих полях' at the top. The 'ДОБАВИТЬ' button is replaced by an 'ИЗМЕНИТЬ' (blue) button. The 'ЗАКРЫТЬ' (purple) button remains. The background elements are the same as in Figure 18.

Изменение данных

Введите те данные, которые хотите изменить, в соответствующих полях

Номер читательского билета

Адрес

ФИО

Контактный номер

Паспортные данные

Образование

Дата рождения

Наличие учёной степени

ДД-ММ-ГГГГ

ИЗМЕНИТЬ

ЗАКРЫТЬ

Рисунок 19 – изменение данных читателя

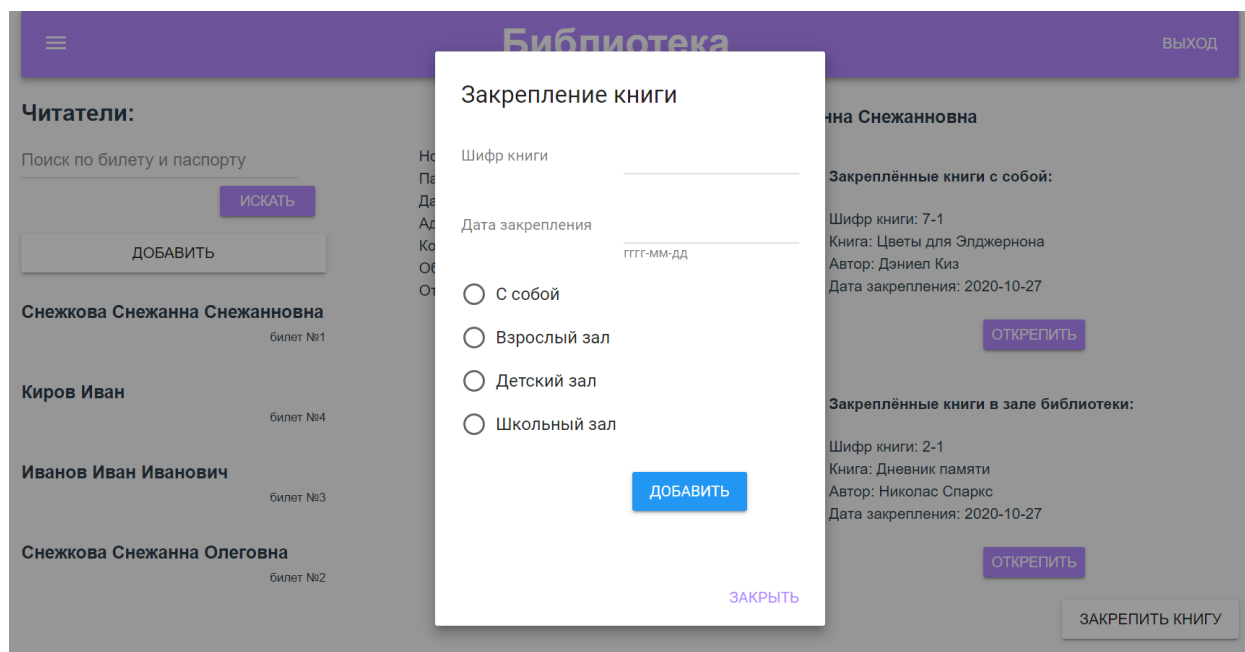


Рисунок 20 – закрепление книги за читателем

- Интерфейс для вывода всех книг в фонде библиотеки. Кликая по выбранной книге или вводя шифр в поиск, выводится полная информация о ней. Скриншот представлен на рисунке 21. Также имеется возможность редактирования данных – кнопка добавление, при клике на которую открывается окно с формой добавления книги (Рисунок 22); кнопка изменить - открывается окно с формой изменения данных книги (Рисунок 23); кнопка удалить – удаляет книгу из библиотеки.

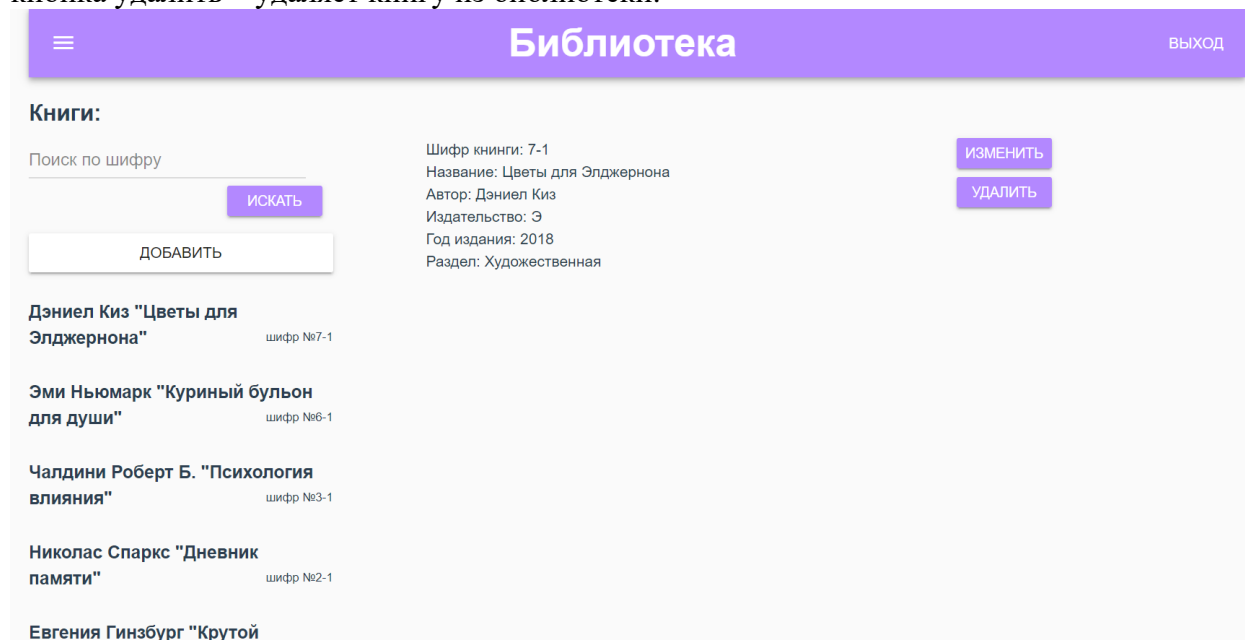


Рисунок 21 – интерфейс с выводом всех книг и полной информации по выбранному книге

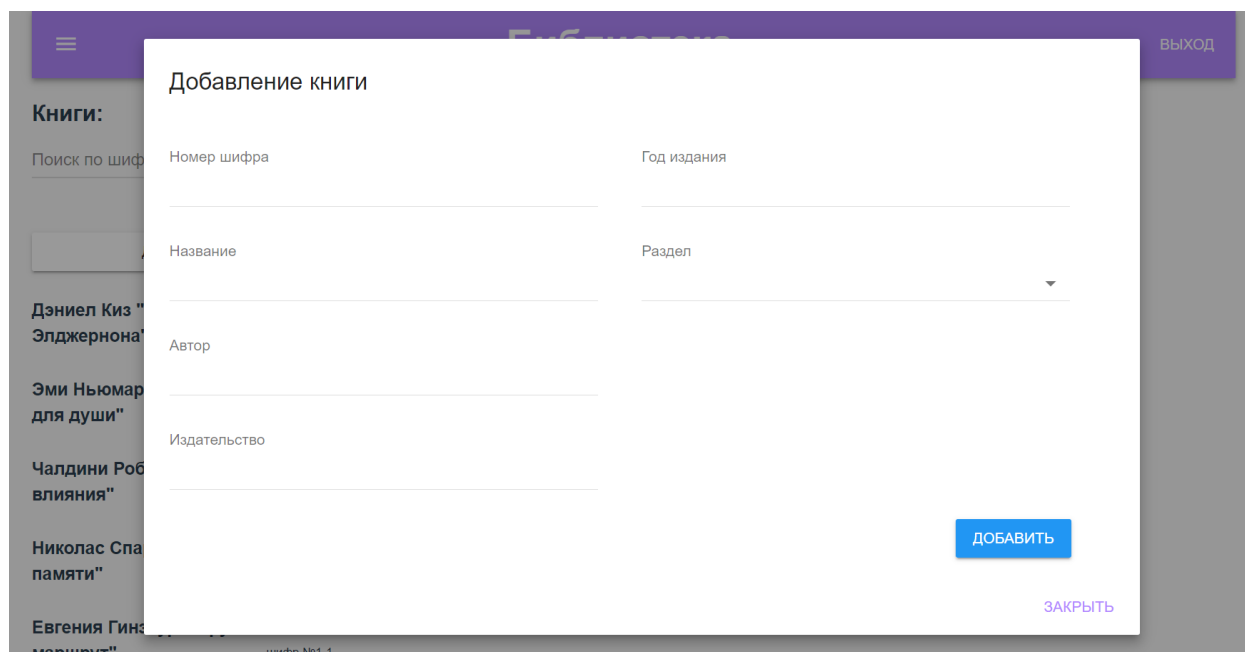


Рисунок 22 – добавления новой книги

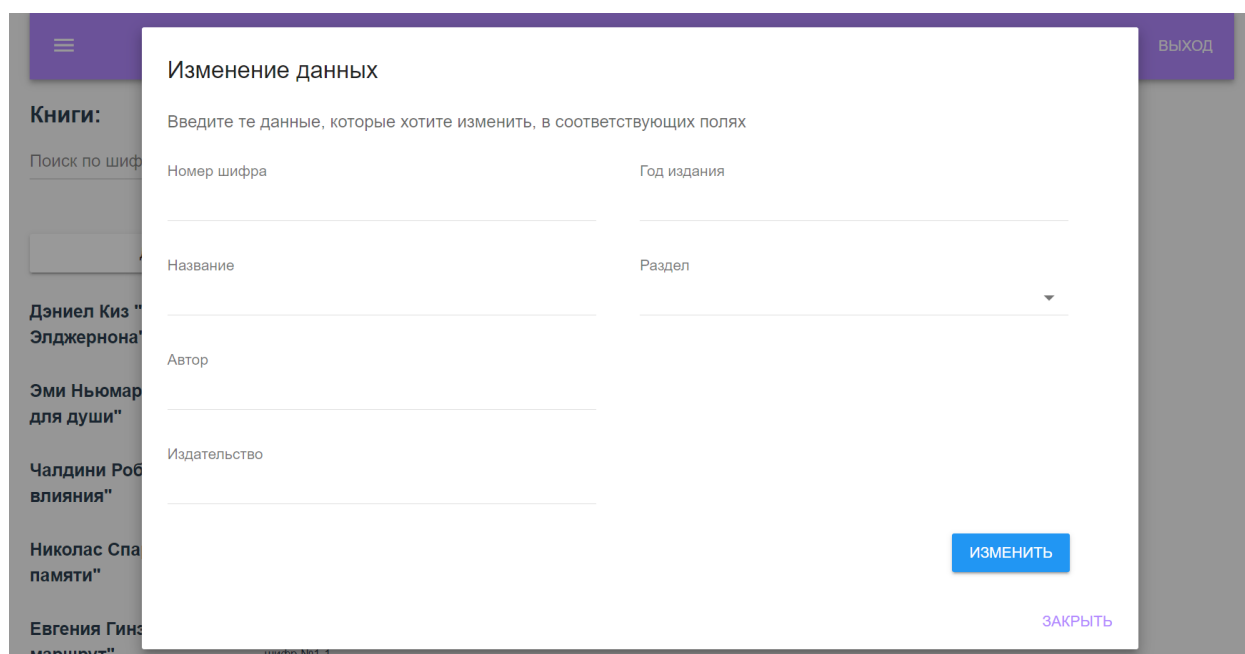


Рисунок 23 – изменение данных книги

- Интерфейс для отображения всех залов библиотеки. При нажатии на выбранный зал открывается информация, о находящихся на данных момент читателей и книг, которые они взяли. Скриншот приведен на рисунке 24.

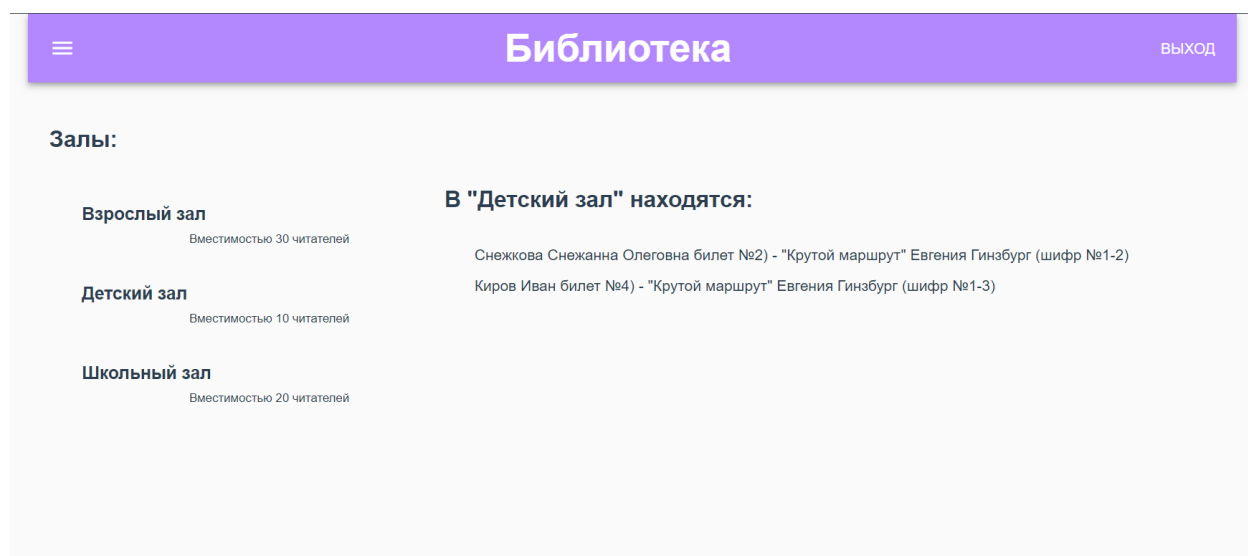


Рисунок 24 - интерфейс с выводом всех залов и полной информации по выбранному залу

- Интерфейсы запросов, открывающиеся с главной страницы сайта при клике на соответствующий запрос, для администратора библиотеки:
 1. Кто из читателей взял книгу более месяца тому назад? (Рисунок 25)
 2. За кем из читателей закреплены книги, количество экземпляров которых в библиотеке не превышает 2? (Рисунок 26)
 3. Сколько в библиотеке читателей младше 20 лет? (Рисунок 27)
 4. Сколько читателей в процентном отношении имеют начальное образование, среднее, высшее, ученую степень? (Рисунок 28)

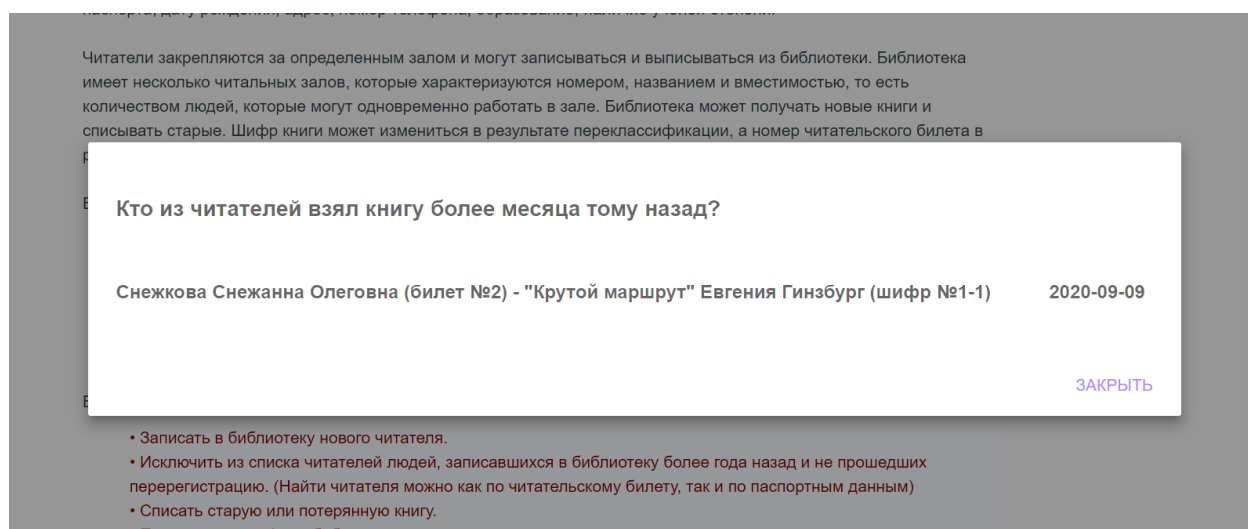


Рисунок 25 – запрос 1

читателем. Сведения о читателях библиотеки должны включать номер читательского билета, ФИО читателя, номер паспорта, дату рождения, адрес, номер телефона, образование, наличие ученой степени.

За кем из читателей закреплены книги, количество экземпляров которых в библиотеке не превышает 2?

Снежкова Снежанна Снежанновна (билет №1):

- "Дневник памяти" (шифр №2-1)
- "Цветы для Эдгера" (шифр №7-1)

Иванов Иван Иванович (билет №3):

- "Куриный бульон для души" (шифр №6-1)
- "Психология влияния" (шифр №3-1)

ЗАКРЫТЬ

- Принять книгу в фонд библиотеки.

Рисунок 25 – запрос 2

Читатели закрепляются за определенным залом и могут записываться и выписываться из библиотеки. Библиотека имеет несколько читальных залов, которые характеризуются номером, названием и вместимостью, то есть

Сколько в библиотеке читателей младше 20 лет?

Всего 3:

Снежкова Снежанна Снежанновна (билет №1) - дата рождения 2006-10-01

Иванов Иван Иванович (билет №3) - дата рождения 2020-10-06

Снежкова Снежанна Олеговна (билет №2) - дата рождения 2020-10-01

ЗАКРЫТЬ

перерегистрацию. (Найти читателя можно как по читательскому билету, так и по паспортным данным)

- Списать старую или потерянную книгу.

Рисунок 25 – запрос 3

Читатели закрепляются за определенным залом и могут записываться и выписываться из библиотеки. Библиотека имеет несколько читальных залов, которые характеризуются номером, названием и вместимостью, то есть

Сколько читателей в процентном отношении имеют начальное образование, среднее, высшее, ученую степень?

Начальное образование: 25%

Среднее образование: 0%

Высшее образование: 75%

Ученую степень: 25%

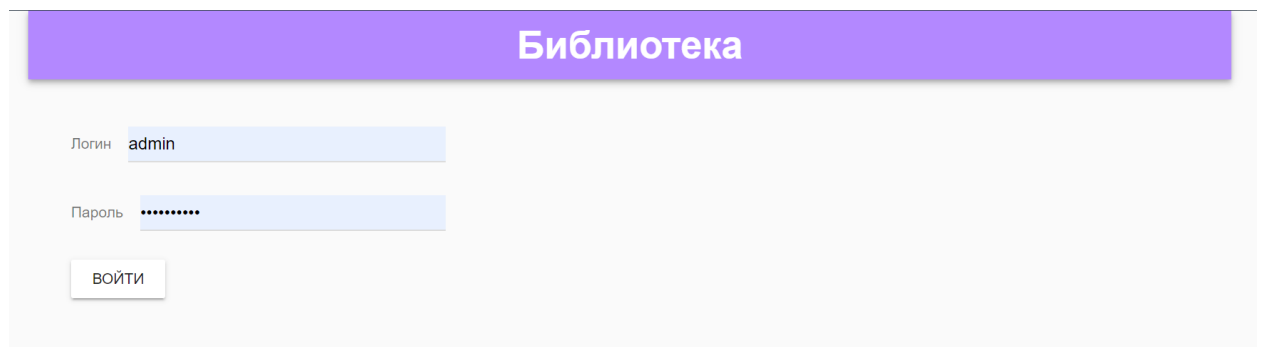
ЗАКРЫТЬ

перерегистрацию. (Найти читателя можно как по читательскому билету, так и по паспортным данным)

- Списать старую или потерянную книгу.
- Принять книгу в фонд библиотеки.

Рисунок 25 – запрос 4

- Интерфейс для авторизации администратора библиотеки (Рисунок 26).



The image shows a web interface for library administration. At the top, there is a purple header bar with the word "Библиотека" (Library) in white. Below the header, on a light gray background, are two input fields. The first field is labeled "Логин" (Login) and contains the text "admin". The second field is labeled "Пароль" (Password) and contains a series of dots. Below these fields is a button labeled "ВОЙТИ" (Log In).

Рисунок 26 – авторизация пользователя

4. КОНТЕЙНЕРИЗАЦИЯ

Docker — это открытая платформа для разработки, доставки и эксплуатации приложений. С помощью технологии Docker (контейнеризации) можно разделить исходное приложение на несколько компонентов, взаимодействие между которыми возможно реализовать. Подобный подход может привести к разным методам реализации компонентов, тем самым предоставляя разработчику широкий спектр возможностей. Благодаря этому разработчику предоставляется возможность создания микро-сервисных архитектур.

Dockerfile — это сценарий, который состоит из последовательности команд и аргументов, необходимых для создания образа. Такие сценарии упрощают развёртывание и процесс подготовки приложения к запуску. Создается виртуальное окружение, внутри которого будет собираться/исполняться программа. Dockerfile для контейнеризации клиентской части разработанного web-приложения представлен на рисунке 27, для серверной части на рисунке 28.

```
FROM node:12.16.3

WORKDIR /library-vue

ENV PATH /library-vue/node_modules/.bin:$PATH

CMD npm install

COPY . /library-vue
```

Рисунок 27 – Dockerfile для клиентской части

```
FROM python:3.8.2

ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

WORKDIR /project_library

RUN pip install --upgrade pip

COPY ./requirements.txt /project_library/requirements.txt
RUN pip install -r /project_library/requirements.txt

COPY . /project_library

EXPOSE 8000
```

Рисунок 28 – Dockerfile для серверной части

Docker Compose - инструмент для создания и запуска многоконтейнерных Docker приложений. В Compose используется специальный файл для конфигурирования сервисов приложения. Затем используется простая команда для создания и запуска всех сервисов из конфигурационного файла. Это самая тривиальная реализация микросервисной архитектуры. Для реализованного web-приложения собираются и запускаются контейнеры для базы данных PostgreSQL, для серверной части Django и для клиентской Vue.js. Данные хранятся в docker-compose.yml, который представлен на рисунке 29. Работа docker-compose представлена на рисунке 30.

```
version: '3.7'
services:
  backend:
    build: ./project_library
    command: bash -c "
      sleep 3 &&
      python3 manage.py makemigrations && python3 manage.py migrate &&
      python3 manage.py runserver --insecure 0.0.0.0:8000";
    volumes:
      - ./project_library:/project_library
    ports:
      - 8000:8000
    depends_on:
      - db
  db:
    image: postgres:13
    volumes:
      - post_data:/var/lib/postgresql/data/
    environment:
      - POSTGRES_USER=admin
      - POSTGRES_PASSWORD=admin
      - POSTGRES_DB=dockerlibrary
  frontend:
    build: ./library-vue
    command: npm start --start;
    volumes:
      - ./library-vue:/library-vue
      - /library-vue/node_modules
    ports:
      - 8080:8080
    depends_on:
      - backend
volumes:
  post_data:
```

Рисунок 29 – содержание docker-compose.yml


```

Starting docker_library_db_1 ... done
Starting docker_library_backend_1 ... done
Starting docker_library_frontend_1 ... done
Attaching to docker_library_db_1, docker_library_backend_1, docker_library_frontend_1
db_1      | PostgreSQL Database directory appears to contain a database; Skipping initialization
db_1      |
db_1      | 2020-11-04 08:34:20.392 UTC [1] LOG:  starting PostgreSQL 13.0 (Debian 13.0-1.pgdg100+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 8.3.0-6) 8.3.0, 64-bit
db_1      |
db_1      | 2020-11-04 08:34:20.392 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
db_1      | 2020-11-04 08:34:20.392 UTC [1] LOG:  listening on IPv6 address "::", port 5432
db_1      | 2020-11-04 08:34:20.409 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
db_1      | 2020-11-04 08:34:20.433 UTC [25] LOG:  database system was shut down at 2020-11-04 08:33:33 UTC
db_1      | 2020-11-04 08:34:20.459 UTC [1] LOG:  database system is ready to accept connections
frontend_1|
frontend_1| > library@1.0.0 start /library-vue
frontend_1| > npm run dev
frontend_1|
frontend_1| > library@1.0.0 dev /library-vue
frontend_1| > webpack-dev-server --inline --progress --config build/webpack.dev.conf.js
frontend_1|
backend_1 | No changes detected
frontend_1 | 13% building modules 32/38 modules 6 active ...late&index=0!/library-vue/src/App.vue{ parser: "babylon" } is deprecated; we now treat it as { parser: "babel"
frontend_1 | }.
backend_1 | Operations to perform:
backend_1 |   Apply all migrations: admin, auth, authtoken, contenttypes, django_summernote, library, sessions
backend_1 | Running migrations:
backend_1 |   No migrations to apply.
backend_1 | Performing system checks...
backend_1 |
backend_1 | System check identified no issues (0 silenced).
backend_1 | November 04, 2020 - 08:34:33
backend_1 | Django version 2.0.6, using settings 'project_library.settings'
backend_1 | Starting development server at http://0.0.0.0:8080/
backend_1 | Quit the server with CONTROL-C.
95% emitting DONE Compiled successfully in 1170ms8:34:37 AM
frontend_1 |
frontend_1 | I Your application is running here: http://localhost:8080
backend_1 | [04/Nov/2020 08:34:58] "GET /admin/library/reader/ HTTP/1.1" 200 5598

```

Рисунок 30 – запущенный docker-compose

ЗАКЛЮЧЕНИЕ

Данная курсовая работа по дисциплине «Основы web-программирования» показывает полученные навыки разработки и создания web-приложений с помощью стека технологий:

- PostgreSQL – свободная объектно-реляционная система управления базами данных
- Django и Django REST Framework – web-фреймворк языка программирования Python для создания web-приложений
- Vue.js – web-фреймворк языка программирования JavaScript для создания пользовательских интерфейсов
- MUSE-UI – библиотека Vue.js для дизайна пользовательского интерфейса, основанная на Material Design
- Docker – программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации.

Реализованное в рамках курсовой работы web-приложение полностью отвечает потребностям предметной области и выполняет все выявленные функциональные требования, а именно:

- CRUD для моделей «Читатель», «Книга» и «Место чтения», «Закрепление»
- Просмотр информации обо всех созданных моделях
- Авторизация в web-приложении
- Предоставление результатов выполнения запросов

СПИСОК ЛИТЕРАТУРЫ

1. Введение — Vue.js [Электронный ресурс]. – URL: <https://ru.vuejs.org/v2/guide/> (дата обращения 26.09.2020)
2. Django REST framework [Электронный ресурс]. – URL: <https://www.django-restframework.org/> (дата обращения 26.09.2020)
3. Современный учебник JavaScript [Электронный ресурс]. – URL: <https://learn.javascript.ru/> (дата обращения 03.10.2020)
4. Документация PostgreSQL – <https://postgrespro.ru/docs/postgresql/9.6/intro-what-is> (дата обращения: 03.10.2020)
5. Serializing Django objects – <https://docs.djangoproject.com/en/3.0/topics/serialization/> (дата обращения: 03.10.2020)
6. ViewSets – <https://www.django-rest-framework.org/api-guide/viewsets/> (дата обращения: 03.10.2020)
7. Class-based Views - <https://www.django-rest-framework.org/api-guide/views/> (дата посещения: 03.10.2020)
8. Muse-UI - <https://muse-ui.org/#/en-US/layout> (дата посещения: 10.10.2020)
9. Docker - <https://www.docker.com> (дата посещения: 28.10.2020)

Приложение 1. Содержание файла models.py

```
from django.db import models

class Book(models.Model):
    section_type = [
        ('Художественная', 'Художественная литература'),
        ('Учебная', 'Учебная литература'),
        ('Психология', 'Психология'),
        ('Детские', 'Детская литература'),
    ]

    name = models.CharField('Название книги', max_length=200)
    author = models.CharField('Автор', max_length=200)
    publisher = models.CharField('Издательство', max_length=100)
    year = models.CharField('Год издания', max_length=4)
    section = models.CharField('Раздел', choices=section_type, max_length=50)
    cipher = models.CharField('Шифр книги', max_length=30)

    class Meta:
        verbose_name = "Книга"
        verbose_name_plural = "Книги"

    def __str__(self):
        return self.name


class Reader(models.Model):
    education_type = (
        ('Дошкольное', 'Дошкольное'),
        ('Начальное общее', 'Начальное общее'),
        ('Основное общее', 'Основное общее'),
        ('Среднее общее', 'Среднее общее'),
        ('Неполное высшее', 'Неполное высшее'),
        ('Высшее', 'Высшее'))

    library_card = models.CharField('Номер читательского билета', max_length=16)
    name = models.CharField('ФИО', max_length=200)
    passport_number = models.CharField('Номер паспорта', max_length=12)
    date_of_birth = models.DateField('Дата рождения')
    address = models.CharField('Адрес', max_length=200)
    phone = models.CharField('Номер телефона', max_length=13)
    education = models.CharField('Образование', choices=education_type, max_length=50)
    academic = models.BooleanField('Наличие ученой степени', default=False)

    class Meta:
        verbose_name = "Читатель"
        verbose_name_plural = "Читатели"

    def __str__(self):
        return self.name
```

```
class Place(models.Model):
    name = models.CharField('Название места чтения', max_length=50)
    capacity = models.IntegerField('Вместимость', blank=True, null=True)

    class Meta:
        verbose_name = "Место чтения"
        verbose_name_plural = "Места чтения"

    def __str__(self):
        return self.name

class Fix(models.Model):
    book = models.ForeignKey(Book, on_delete=models.CASCADE, verbose_name='Книга')
    reader = models.ForeignKey(Reader, on_delete=models.CASCADE, verbose_name='Читатель')
    date_fix = models.DateField('Дата закревления', auto_now_add=True)
    handed = models.BooleanField('Книга сдана', default=False)
    place = models.ForeignKey(Place, on_delete=models.CASCADE, verbose_name='Место чтения', related_name='place_fix')

    class Meta:
        verbose_name = "Закрепление книги"
        verbose_name_plural = "Закрепления книг"

    def __str__(self):
        return ""+str(self.reader.library_card)+" (" +self.book.name+)"
```

Приложение 2. Содержимое файла serialisers.py

```
from rest_framework import serializers
from rest_framework.relations import SlugRelatedField

from .models import *


class ReadersSerializer(serializers.ModelSerializer):
    """Сериализация читателей"""
    class Meta:
        model = Reader
        fields = ('id', 'library_card', 'name', 'passport_number')


class ReaderSerializer(serializers.ModelSerializer):
    """Сериализация читателя"""
    class Meta:
        model = Reader
        fields = ('id', 'library_card', 'name', 'passport_number', 'date_of_birth',
                  'address', 'phone', 'education', 'academic')


class BooksSerializer(serializers.ModelSerializer):
    """Сериализация книг"""
    class Meta:
        model = Book
        fields = ('id', 'cipher', 'name', 'author')


class BookSerializer(serializers.ModelSerializer):
    """Сериализация книги"""
    class Meta:
        model = Book
        fields = ('id', 'name', 'author', 'publisher', 'year', 'section', 'cipher')


class PlacesSerializer(serializers.ModelSerializer):
    """Сериализация мест"""
    class Meta:
        model = Place
        fields = "__all__"
```

```

class FixUpdSerializer(serializers.ModelSerializer):
    """Сериализатор для открепления книги """
    class Meta:
        model = Fix
        fields = ('id', 'handed')

class FixesSerializer(serializers.ModelSerializer):
    """Сериализация закрепления"""
    book = BooksSerializer(read_only=True)
    reader = ReadersSerializer(read_only=True)

    class Meta:
        model = Fix
        fields = ('id', 'book', 'reader', 'date_fix', 'place')

class FixAddSerializer(serializers.ModelSerializer):
    """Сериализация для добавления закрепления"""

    class Meta:
        model = Fix
        fields = ('book', 'reader', 'place')

```

```

class FixReaderSerializer(serializers.ModelSerializer):
    """Сериализация вывода закрепленных книг у читателя"""
    book = BookSerializer()

    class Meta:
        model = Fix
        fields = ('id', 'book', 'date_fix', 'place', 'handed')

class PlaceUpdSerializer(serializers.ModelSerializer):
    """Сериализатор для открепления книги """
    class Meta:
        model = Fix
        fields = ('id', 'place')

class ReadersDateSerializer(serializers.ModelSerializer):
    """Сериализация читателей"""
    class Meta:
        model = Reader
        fields = ('id', 'library_card', 'name', 'date_of_birth')

```

Приложение 3. Содержание файла views.py

```
from datetime import datetime, timedelta

from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import permissions
from rest_framework import viewsets

from .models import *
from .serializers import (ReaderSerializer, BooksSerializer, BookSerializer, ReadersSerializer,
                          FixUpdSerializer, FixesSerializer, FixReaderSerializer, FixAddSerializer,
                          PlacesSerializer, PlaceUpdSerializer, ReadersDateSerializer)

class Readers(APIView):
    """Отображение всех читателей"""

    def get(self, request):
        readers = Reader.objects.all()
        serializers = ReadersSerializer(readers, many=True)
        return Response(serializers.data)

class ReaderOne(APIView):
    """Отображение одного читателя и все взятые им книги на данный момент"""

    def get(self, request):
        id_reader = request.GET.get("reader")
        reader = Reader.objects.filter(id=id_reader)
        reader_serializer = ReaderSerializer(reader, many=True)
        fixes = Fix.objects.filter(reader=id_reader, handed=False)
        book_serializer = FixReaderSerializer(fixes, many=True)
        return Response({"reader": reader_serializer.data, "books": book_serializer.data})

class ReaderAdd(APIView):
    """Добавление читателя"""
    permission_classes = [permissions.IsAuthenticated, ]

    def post(self, request):
        reader = ReaderSerializer(data=request.data)
        if reader.is_valid():
            reader.save()
            return Response({"status": 201})
        else:
            return Response({"status": 400})
```



```

class ReaderDelUpd(viewsets.ModelViewSet):
    """Отображение для модели Взятые книги"""
    queryset = Reader.objects.all()

    def get_serializer_class(self):
        if self.action == 'destroy':
            return ReaderSerializer
        elif self.action == 'update':
            return ReaderSerializer

class Books(APIView):
    """Отображение всех книг"""

    def get(self, request):
        books = Book.objects.all()
        serializers = BooksSerializer(books, many=True)

        return Response(serializers.data)

```

```

class BookOne(APIView):
    """Отображение одной книги"""

    def get(self, request):
        id_book = request.GET.get("book")
        book = Book.objects.filter(cipher=id_book)
        serializer = BookSerializer(book, many=True)
        return Response({"book": serializer.data})

class BookAdd(APIView):
    """Добавление книги"""
    permission_classes = [permissions.IsAuthenticated, ]

    def post(self, request):
        book = BookSerializer(data=request.data)
        if book.is_valid():
            book.save()
            return Response({"status": 201})
        else:
            return Response({"status": 400})

```

```

class BookDelUpd(viewsets.ModelViewSet):
    """Отображение для модели Взятые книги"""
    queryset = Book.objects.all()

    def get_serializer_class(self):
        if self.action == 'destroy':
            return BookSerializer
        elif self.action == 'update':
            return BookSerializer

class Fixes(APIView):
    """Закрепления"""
    permission_classes = [permissions.IsAuthenticated, ]

    def get(self, request):
        fixes = Fix.objects.all()
        serializers = FixesSerializer(fixes, many=True)
        return Response({"data": serializers.data})

```

```

class FixAdd(APIView):
    """Добавление закреплений"""
    permission_classes = [permissions.IsAuthenticated, ]

    def post(self, request):
        fix = FixAddSerializer(data=request.data)
        if fix.is_valid():
            fix.save()
            return Response({"status": 201})
        else:
            return Response({"status": 400})

class FixUpdate(viewsets.ModelViewSet):
    """Отображение обновления закреплений (открепление книги)"""
    queryset = Fix.objects.all()

    def get_serializer_class(self):
        if self.action == 'update':
            return FixUpdSerializer

```

```

class Places(APIView):
    """Места"""

    def get(self, request):
        places = Place.objects.all()
        serializers = PlacesSerializer(places, many=True)
        return Response({"place": serializers.data})

class PlaceOne(APIView):
    """Отображение одного зала и все закреплений в нем"""
    permission_classes = [permissions.IsAuthenticated, ]

    def get(self, request):
        id_place = request.GET.get("place")
        place_single = Place.objects.filter(id=id_place)
        serializer = PlacesSerializer(place_single, many=True)
        fixes = Fix.objects.filter(place=id_place, handed=False)
        fixes_serializer = FixesSerializer(fixes, many=True)
        return Response({"place": serializer.data, "fixes": fixes_serializer.data})

```

```

class PlaceUpd(viewsets.ModelViewSet):
    """Отображение обновления закреплений (открепление книги)"""
    queryset = Fix.objects.all()

    def get_serializer_class(self):
        if self.action == 'update':
            return PlaceUpdSerializer
        elif self.action == 'list':
            return PlaceUpdSerializer

class FixFilter(APIView):
    """Отображение фильтрации по дате за месяц"""

    def get(self, request):
        now = datetime.now() - timedelta(days=30)
        fixes = Fix.objects.filter(date_fix__lte=now)
        serializer = FixesSerializer(fixes, many=True)
        return Response({"data": serializer.data})

```

```

class ReadersFilter(APIView):
    """Отображение фильтрации по дате рождения (меньше 20 лет)"""
    def get(self, request):
        now = datetime.now() - timedelta(days=365*20)
        readers = Reader.objects.filter(date_of_birth__gte=now)
        serializer = ReadersDateSerializer(readers, many=True)
        return Response({"data": serializer.data})

class EducationFiler(APIView):
    """Сколько читателей в процентном отношении имеют начальное образование, среднее, высшее, ученую степень?"""
    def get(self, request):
        readers = Reader.objects.count()
        primary = Reader.objects.filter(education='Начальное общее').count()
        secondary = Reader.objects.filter(education='Среднее общее').count()
        high = Reader.objects.filter(education='Высшее').count()
        primary_percent = (primary/readers) * 100
        secondary_percent = (secondary/readers) * 100
        high_percent = (high/readers) * 100
        academic = Reader.objects.filter(academic='True').count()
        academic_percent = (academic/readers) * 100
        return Response({'primary': primary_percent, 'secondary': secondary_percent, 'high': high_percent,
                        'academic': academic_percent})

```

5.