

Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИТМО”

Факультет Инфокоммуникационных технологий

Образовательная программа Интеллектуальные системы в гуманитарной сфере
(Академический бакалавр, Очная ф/о)

Направление подготовки (специальность) 45.03.04 - Интеллектуальные системы в гуманитарной сфере

О Т Ч Е Т

по курсовой работе по дисциплине «Основы web-программирования»

Тема задания: Разработка web-приложения для сотрудников библиотеки

Обучающийся Таиров Назим Кахрамонович, К3442

Руководитель курсовой работы: Говоров Антон Игоревич, ассистент

Оценка по курсовой работе ____

Подписи членов комиссии:

____ Говоров А. И.
(подпись)

____ Чунаев А. В.
(подпись)

____ Антонов М. Б.
(подпись)

Дата ____

СОДЕРЖАНИЕ

Оглавление

| | |
|--|----|
| ВВЕДЕНИЕ..... | 4 |
| 1. Предметная область..... | 5 |
| 1.1. Общее описание предметной области..... | 5 |
| 1.2. Функциональные требования к web-сервису в рамках предметной области..... | 5 |
| 1.3. Выводы..... | 6 |
| 2. Модели данных..... | 7 |
| 2.1. Теория..... | 7 |
| 2.2. Разработка моделей данных для создаваемого web-приложения..... | 7 |
| 2.2.1. Модель читального зала..... | 8 |
| 2.2.2. Модель читателя..... | 8 |
| 2.2.3. Модель книги..... | 9 |
| 2.2.4. Модель закрепления..... | 10 |
| 2.3. Выводы..... | 11 |
| 3. Серверная часть..... | 12 |
| 3.1. Теория..... | 12 |
| 3.2. Админ-панель..... | 12 |
| 3.2.1. Теория..... | 12 |
| 3.2.2. Код..... | 13 |
| 3.2.3. Внешний вид..... | 14 |
| 3.3. Django REST Framework..... | 15 |
| 3.3.1. Общий принцип работы..... | 15 |
| 3.3.2. URL-пути серверной части web-приложения..... | 16 |
| 3.3.3. Сериализаторы..... | 17 |
| 3.3.4. Представления..... | 18 |
| 3.4. Выводы..... | 27 |
| 4. Клиентская часть..... | 28 |
| 4.1. Теория..... | 28 |
| 4.2. Компоненты..... | 29 |
| 4.2.1. Файл index.js..... | 29 |
| 4.3. Компонент Home..... | 30 |
| 4.3.1. Код..... | 30 |
| 4.3.2. Отображение на сайте..... | 33 |
| 4.4. Компонент Login..... | 34 |
| 4.4.1. Код..... | 34 |
| 4.4.2. Отображение на сайте..... | 36 |
| 4.5. Компонент Reader..... | 37 |
| 4.5.1. Код..... | 37 |
| 4.5.2. Отображение на сайте..... | 38 |

| | | |
|--------|---|-----|
| 4.6. | Компонент Reader_books..... | 39 |
| 4.6.1. | Код..... | 39 |
| 4.6.2. | Отображение на сайте..... | 41 |
| 4.7. | Компонент Books..... | 42 |
| 4.7.1. | Код..... | 42 |
| 4.7.2. | Отображение на сайте..... | 43 |
| 4.8. | Компоненты ReaderChange, BookChange и ReaderForm..... | 47 |
| 4.9. | Muse-UI..... | 50 |
| 4.10. | Выводы..... | 50 |
| 5. | ЗАКЛЮЧЕНИЕ..... | 52 |
| | СПИСОК ЛИТЕРАТУРЫ..... | 53 |
| | Приложение А. Сериализаторы..... | 54 |
| | Приложение Б. Представления..... | 58 |
| | Приложение В. Страницы Django REST Framework'а..... | 61 |
| | Приложение Г. Код компонентов..... | 65 |
| | Приложение Д. Код компонента Books..... | 74 |
| | Приложение Е. Код компонента ReaderChange..... | 95 |
| | Приложение Ж. Код компонента BookChange..... | 103 |
| | Приложение З. Код компонента ReaderForm..... | 111 |

ВВЕДЕНИЕ

Курсовая работа заключается в разработке web-приложения, предназначенного для сотрудников библиотеки.

Цель выполнения данной курсовой работы в рамках прохождения обучения по дисциплине «основы web-программирования» заключается в овладении навыками и умениями реализации web-сервисов с использованием таких программ библиотек и компонентов, как Django, Django REST Framework, Vue.js, Muse-UI

Основные задачи курсовой работы:

1. Анализ предметной области;
2. Разработка функциональных требований к сайту в рамках данной предметной области;
3. Создание модели данных;
4. Реализация серверной части web-приложения средствами Django REST Framework;
5. Реализация клиентской части web-приложения средствами Vue.js и организация связи между серверной и клиентской частями.

В первой главе проведён анализ содержания работ и требований к их выполнению. Во второй главе рассмотрена разработка модели данных. В третьей главе разобрано создание серверной части реализуемого web-сервиса. В четвёртой главе представлена разработка клиентской части сервиса. В приложении А приведены сериализаторы, в приложении Б – представления, в приложении В – отображение страниц Django REST Framework'а, в приложениях Г, Д, Е, Ж и З – код компонентов Vue.

1. Предметная область

1.1. Общее описание предметной области

Общее описание предметной области было взято из текста индивидуального задания к курсовой работе.

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

Книги в библиотеке в качестве главной характеристики имеют свой шифр, и они закреплены за определённым залом. Книги могут храниться на полках, а могут быть выданы конкретному читателю.

Читатели, в свою очередь, закрепляются за определённым залом и могут записываться и выписываться из библиотеки. Библиотека имеет несколько читальных залов, которые характеризуются номером и названием. Библиотека может получать новые книги и списывать старые. Шифр книги может измениться в результате переклассификации, а номер читательского билета в результате перерегистрации.

Информация по книгам, добавленным для примера, взята с сайта электронного каталога РНБ [1], а по формированию шифра книги взята с сайта ГПИБ России (Историческая библиотека) [2].

1.2. Функциональные требования к web-сервису в рамках предметной области

Для каждой книги в БД должны храниться следующие сведения: название книги, автор(ы), издательство, год издания, раздел, 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 учёной степени.

Библиотекарь могут потребоваться следующие сведения о текущем состоянии библиотеки:

- Какие книги закреплены за определённым читателем?
- Какие экземпляры книги закреплены за читателями, а какие – хранятся в библиотеке?

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.
- Списать старую или потерянную книгу.
- Принять книгу в фонд библиотеки.
- Изменять информацию в базе данных о конкретных читателях в соответствии с реальными изменениями.

1.3. Выводы

В результате проведённого анализа предметной области были сформулированы функциональные требования к реализуемому в рамках курсовой работы web-приложению. Разработка модели данных, а также серверной и клиентской части приложения будут описаны в последующих главах.

2. Модели данных

2.1. Теория

Модель данных в Django – это одиночный определённый источник информации о данных разработчика. Она содержит необходимые поля и поведение данных, которые он хранит. Обычно, каждая модель относится к одной единственной таблице базы данных. Через инструмент моделей Django позволяет разработчику вносить любые изменения в базу данных, не взаимодействуя с ней напрямую.

2.2. Разработка моделей данных для создаваемого web-приложения

Модели данных для создаваемого web-приложения были разработаны на основе функциональных требований к приложению. Было решено создать 4 модели, представляющие 4 таблицы в базе данных:

1. Зал;
2. Книга;
3. Читатель;
4. Закрепление.

На рис. 1 показана схема полученной базы данных.

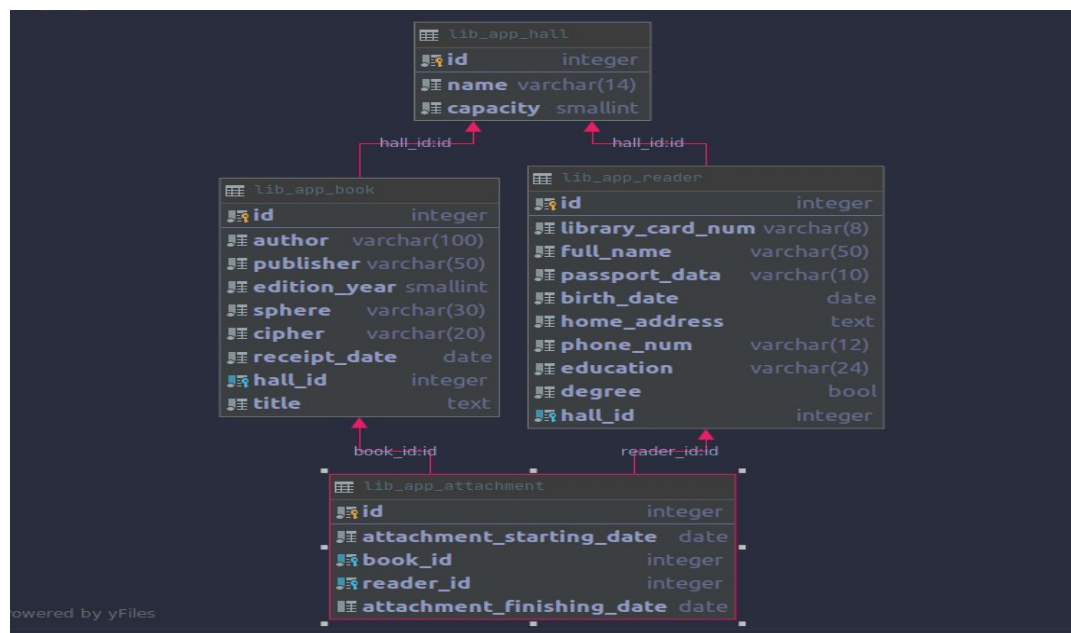


Рисунок 1 – Схема базы данных разрабатываемого web-приложения

Как было сказано ранее, каждая таблица базы данных была задана, как модель Django. Эти модели прописываются в файле `models.py`.

2.2.1. Модель читального зала

Модель Django для читальных залов (рис. 2) состоит из следующих полей:

1. *name* – название зала, текстовое поле длиной до 14 символов;
2. *capacity* – вместимость, числовое поле (тип данных: small integer);
3. *id* – автоматически создаваемое поле-счётчик, не прописываемое при задании модели.

```
1 from django.db import models
2 # from django.contrib.auth.models import User
3
4
5 class Hall(models.Model):
6     name = models.CharField(max_length=14, verbose_name="название")
7     capacity = models.SmallIntegerField(verbose_name="вместимость (человек)")
8
9     class Meta:
10         verbose_name = "Зал"
11         verbose_name_plural = "Залы"
12
13     def __str__(self):
14         return "Зал " + str(self.id) + " (" + self.name + ")"
15
16
17 class Book(models.Model):
18     title = models.TextField(verbose_name="название")
19     author = models.CharField(max_length=100, verbose_name="автор(ы)")
20     publisher = models.CharField(max_length=50, verbose_name="издатель")
21     edition_year = models.SmallIntegerField(verbose_name="год издания")
22     sphere = models.CharField(max_length=30, verbose_name="раздел")
23     cipher = models.CharField(max_length=20, verbose_name="шифр книги")
24     receipt_date = models.DateField(verbose_name="дата поступления")
25     hall = models.ForeignKey(Hall, on_delete=models.CASCADE, verbose_name="зал")
26
27     class Meta:
28         verbose_name = "Книга"
29         verbose_name_plural = "Книги"
```

Рисунок 2 – Модель читального зала в Django

Как видно на рис. 2, для каждого поля и для самой модели также прописаны названия, которые будут отображаться в admin-панели (*verbose_name* и *verbose_name_plural*). Помимо этого, с помощью метода `__str__` прописано, как там же будут записаны экземпляры данной модели. Данные переменные и метод прописаны для всех моделей, поэтому далее, для последующих моделей, это не будет описываться.

2.2.2. Модель читателя

Модель Django для читателя (рис. 3) состоит из следующих полей:

1. *library_card_name* – номер читательского билета, текстовое поле длиной до 8 СИМВОЛОВ;
2. *full_name* – ФИО, текстовое поле длиной до 50 символов;

3. *passport_data* – серия и номер паспорта, текстовое поле длиной до 10 символов;
4. *birth_date* – дата рождения, поле формата date;
5. *home_address* – адрес, текстовое поле с неограниченной длиной;
6. *phone_num* – контактный номер, текстовое поле длиной до 12 символов;
7. *education* – образование, текстовое поле длиной до 24 символов, для которого в переменной *EdType* прописаны возможные варианты;
8. *degree* – наличие учёной степени, логическое поле;
9. *hall* – зал, поле-внешний ключ, осуществляющее связь данной модели с моделью Hall;
10. *books* – книги, поле, осуществляющее связь типа многое-ко-многому между данной моделью и моделью Book посредством модели Attachment. У данного поля прописан параметр *related_name*, который задаёт название, с помощью которого будет производиться обращение к объекту данной модели от объекта связанной с ней модели;
11. *id* – автоматически создаваемое поле-счётчик, не прописываемое при задании модели.

```
34
35 class Reader(models.Model):
36     EdType = (
37         ('среднее общее', 'среднее общее'),
38         ('среднее профессиональное', 'среднее профессиональное'),
39         ('неполное высшее', 'неполное высшее'),
40         ('бакалавр', 'бакалавр'),
41         ('специалист', 'специалист'),
42         ('магистр', 'магистр'),
43         ('аспирантура', 'аспирантура'))
44     library_card_num = models.CharField(max_length=8, verbose_name="номер читательского билета")
45     full_name = models.CharField(max_length=50, verbose_name="ФИО")
46     passport_data = models.CharField(max_length=10, verbose_name="серия, номер паспорта")
47     birth_date = models.DateField(verbose_name="дата рождения")
48     home_address = models.TextField(verbose_name="адрес")
49     phone_num = models.CharField(max_length=12, verbose_name="контактный номер")
50     education = models.CharField(choices=EdType, max_length=24, verbose_name="образование")
51     degree = models.BooleanField(verbose_name="наличие учёной степени")
52     hall = models.ForeignKey(Hall, on_delete=models.CASCADE, verbose_name="зал")
53     books = models.ManyToManyField(Book, through='Attachment', verbose_name="книги", related_name='attached_book')
54
55     class Meta:
56         verbose_name = "Читатель"
57         verbose_name_plural = "Читатели"
58
59     def str(self):
```

Рисунок 3 – Модель читателя в Django

2.2.3. Модель книги

Модель Django для книги (рис. 4) состоит из следующих полей:

1. *title* – название, текстовое поле с неограниченной длиной;
2. *author* – автор(ы), текстовое поле длиной до 100 символов;
3. *publisher* – издатель, текстовое поле длиной до 50 символов;
4. *edition_year* – год издания, числовое поле (тип данных: small integer);
5. *sphere* – раздел, текстовое поле длиной до 30 символов;
6. *cipher* – шифр, текстовое поле длиной до 20 символов;
7. *receipt_date* – дата поступления, поле формата date;
8. *hall* – зал, поле-внешний ключ, осуществляющее связь данной модели с

моделью Hall.

```
class Book(models.Model):
    title = models.TextField(verbose_name="название")
    author = models.CharField(max_length=100, verbose_name="автор(ы)")
    publisher = models.CharField(max_length=50, verbose_name="издатель")
    edition_year = models.SmallIntegerField(verbose_name="год издания")
    sphere = models.CharField(max_length=30, verbose_name="раздел")
    cipher = models.CharField(max_length=20, verbose_name="шифр книги")
    receipt_date = models.DateField(verbose_name="дата поступления")
    hall = models.ForeignKey(Hall, on_delete=models.CASCADE, verbose_name="зал")

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

    def __str__(self):
        return 'единица №'+str(self.id)+' экземпляр книги '+self.title+' '+self.cipher[-1]
```

Рисунок 4 – Модель книги в Django

2.2.4. Модель закрепления

Модель Django для закрепления – ассоциативная модель, осуществляющая связь многое-ко-многому между моделями книги и читателя. Она состоит из следующих полей (рис. 5):

1. *reader* – читатель, поле-внешний ключ, осуществляющее связь данной модели с моделью Reader;
2. *book* – книга, поле-внешний ключ, осуществляющее связь данной модели с моделью Book;
3. *attachment_starting_date* – дата закрепления, поле формата date;

4. *attachment_finishing_date* – дата открепления, поле формата date.

```
class Attachment(models.Model):
    reader = models.ForeignKey(Reader, on_delete=models.CASCADE, verbose_name="читатель")
    book = models.ForeignKey(Book, on_delete=models.CASCADE, verbose_name="книга")
    attachment_starting_date = models.DateField(verbose_name="время закрепления")
    attachment_finishing_date = models.DateField(null=True, blank=True, verbose_name="время открепления")

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

    def __str__(self):
        return 'закрепление №'+str(self.id)+'', '+str(self.book)+'', '+str(self.reader)'
```

Рисунок 5 – Модель закрепления в Django

2.3. Выводы

В результате разработки структуры базы данных web-приложения были созданы схема базы данных и модели для каждой таблицы базы данных из этой схемы, прописаны все поля в созданных моделях. Также были заданы связи между различными моделями. Для каждой модели и для каждого поля в моделях определены параметры *verbose_name* и *verbose_name_plural*, и для каждой модели описан метод *__str__*.

3. Серверная часть

Серверная часть разрабатываемого web-приложения была реализована с помощью средств Django и Django REST Framework.

3.1. Теория

Некоторая информация про Django была представлена в предыдущей главе в контексте моделей Django. Теперь это будет рассмотрено несколько подробнее. Django – это свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC.

MVC – Model-View-Controller («Модель-Представление-Контроллер», «Модель-Вид-Контроллер») – схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер – таким образом, что модификация каждого компонента может осуществляться независимо.

- Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.
- Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.
- Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

В отличие от других фреймворков, обработчики URL в Django конфигурируются явно при помощи регулярных выражений.

Для работы с базой данных Django использует собственный ORM, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных [3].

Django REST framework – это удобный и гибкий фреймворк, содержащий набор инструментов для конструирования API web-приложений [4].

3.2. Админ-панель

3.2.1. Теория

Админ-панель в Django – это автоматический интерфейс админа. Он считывает метаданные с записанных в файле `models.py` моделей и создаёт быстрый интерфейс, центрированный на моделях, с помощью которого пользователи с достаточным уровнем доступа могут управлять контентом сайта. Использование админ-панели рекомендовано

ограничить внутренним управлением в организации. Она не создавалась, чтобы представлять собой весь фронтэнд сайта. Админ-панель наиболее важна на начальных этапах разработки web-сервиса.

3.2.2. Код

Прописывается она в файле `admin.py` образом, представленным на рис. 6 и рис. 7.

```
1  from django.contrib import admin
2
3  from lib_app.models import Hall, Book, Reader, Attachment
4
5
6  class HallAdmin(admin.ModelAdmin):
7
8      list_display = ("name", "capacity")
9
10
11  class BookAdmin(admin.ModelAdmin):
12
13      list_display = ("title", "author", "publisher", "edition_year",
14                     "sphere", "cipher", "receipt_date", "hall")
15
16
17  class ReaderAdmin(admin.ModelAdmin):
18
19      list_display = ("library_card_num", "full_name", "passport_data",
20                     "birth_date", "home_address", "phone_num",
21                     "education", "degree", "hall"
22                     )
23
24
25  class AttachmentAdmin(admin.ModelAdmin):
26
27      list_display = ("reader", "book", "attachment_starting_date",
28                     "attachment_finishing_date")
```

Рисунок 6 – Содержимое файла `admin.py` (первая часть)

```
29
30  admin.site.register(Hall, HallAdmin)
31  admin.site.register(Book, BookAdmin)
32  admin.site.register(Reader, ReaderAdmin)
33  admin.site.register(Attachment, AttachmentAdmin)
34
```

Рисунок 7 – Содержимое файла `admin.py` (вторая часть)

Сначала импортируются модели приложения, затем для них прописываются специальные классы, которые далее регистрируются в админ-панели.

3.2.3. Внешний вид

Зайти на админ-панель можно по адресу, прописанному в файле `urls.py` всего сайта (рис. 8).

```
16 import ...|
18
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21     path('api/lib/', include('lib_app.urls')),
22     path('auth/', include('djoser.urls')),
23     path('auth/', include('djoser.urls.authtoken')),
24     path('auth/', include('djoser.urls.jwt')),
25 ]
26
```

Рисунок 8 – Содержимое файла `urls.py` всего сайта

На рис. 9, 10 и 11 представлено, как выглядит админ-панель.

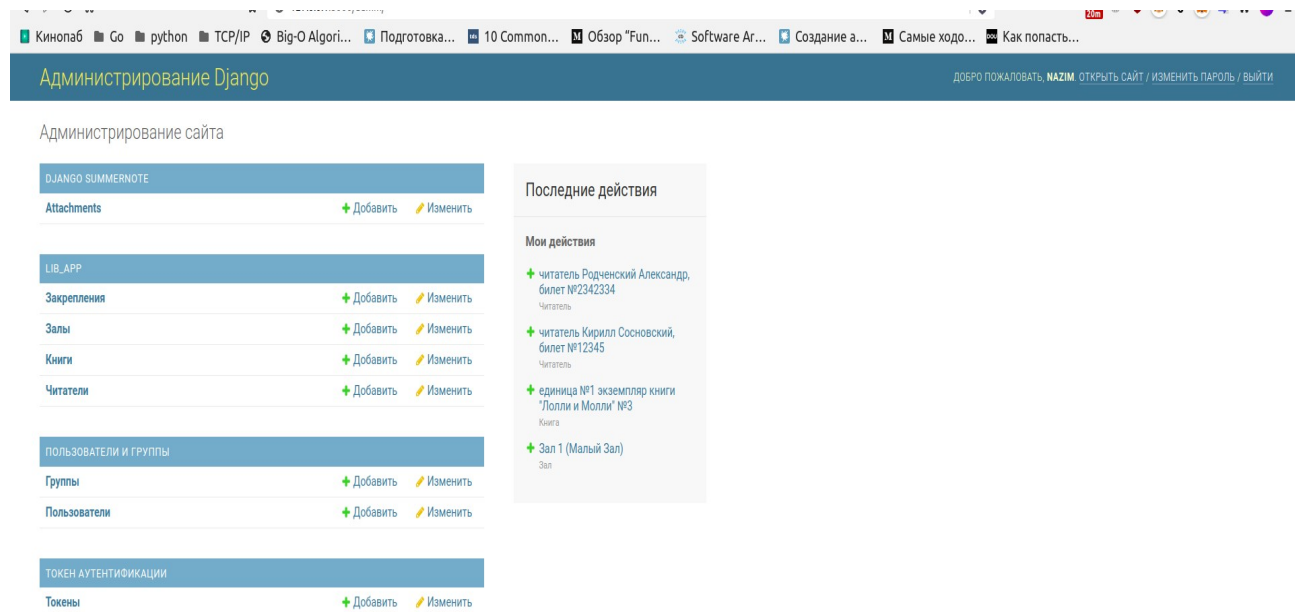


Рисунок 9 – Общий вид админ-панели

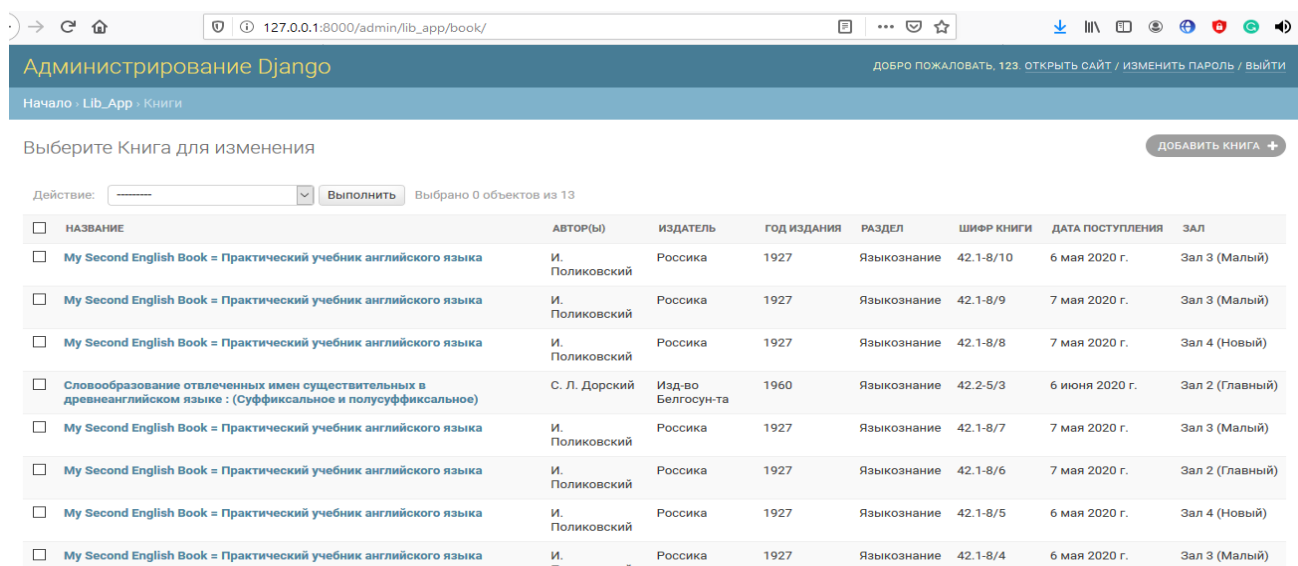


Рисунок 10 – Модель в админ-панели

Изменить Книга ИСТОРИЯ

Название:

Автор(ы):

Издатель:

Год издания:

Раздел:

Шифр книги:

Дата поступления: Сегодня | 📅

Зал: ✏️ +

Удалить Сохранить и добавить другой объект Сохранить и продолжить редактирование СОХРАНИТЬ

Рисунок 11 – Изменение экземпляра модели в админ-панели

3.3. Django REST Framework

3.3.1. Общий принцип работы

Django REST framework — это API фреймворк для Django в стиле REST. У создания фреймворков в таком стиле есть два основных принципа:

1. Сервер не должен ничего знать о текущем состоянии Клиента. В запросе от Клиента должна быть вся необходимая информация для обработки этого запроса Сервером.

2. Каждый ресурс на Сервере должен иметь определённый Id, а также уникальный URL, по которому осуществляется доступ к этому ресурсу.

На данный момент мы можем найти фреймворк для создания приложений в стиле REST практически для каждого языка программирования, используемого в веб-разработке. Логика построения Web API на Сервере в этих фреймворках реализована одинаково.

Действия для управления данными привязаны к определенным HTTP-методам. Существует несколько стандартных действий для работы с данными – это Create, Read, Update, Delete. Часто их обобщают как CRUD.

Для создания объекта используется http-метод POST.

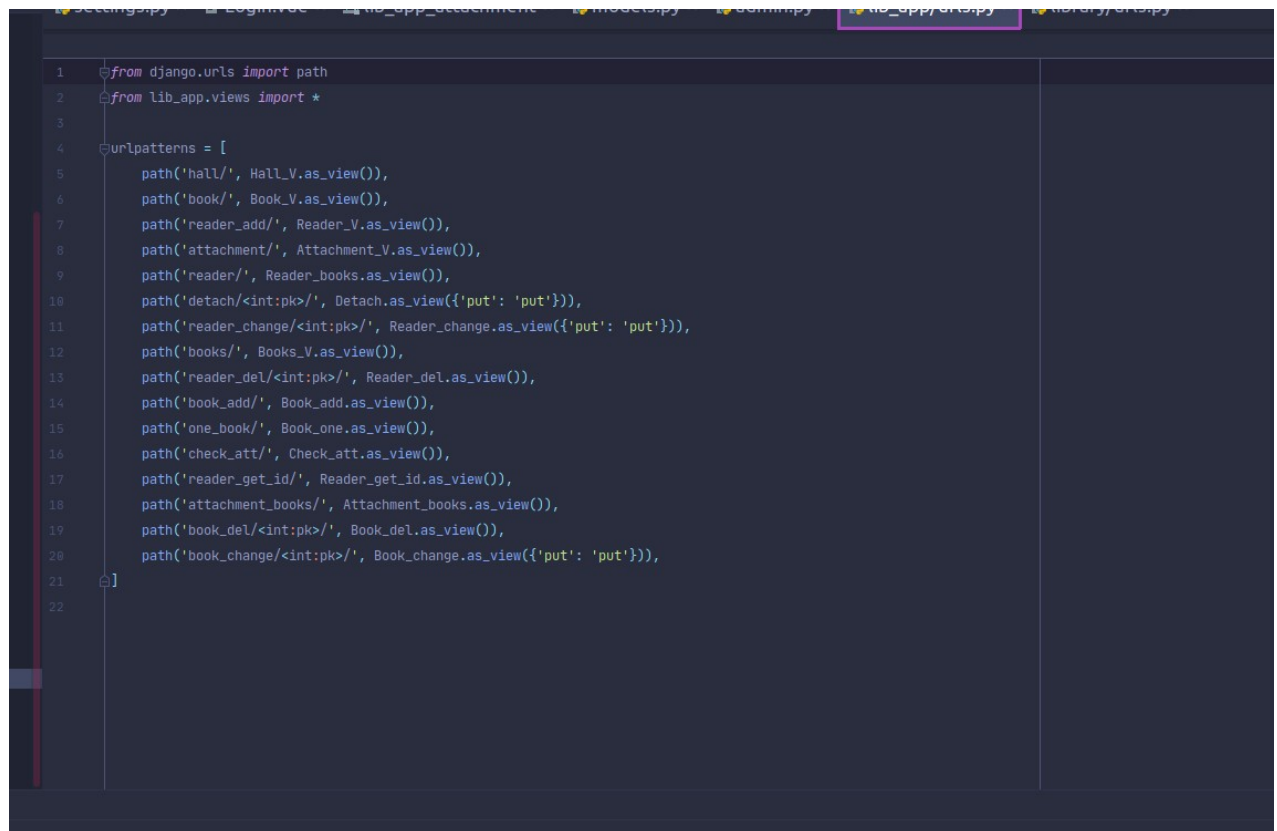
Для чтения — http-метод GET.

Для изменения — http-метод PUT.

Для удаления — http-метод DELETE [5].

3.3.2. URL-пути серверной части web-приложения

На рис. 8 видно, что прописан путь ‘api/lib/’. По этому пути подключаются все пути, прописанные в файле urls.py разрабатываемого приложения. Содержимое этого файла представлено на рис. 12.



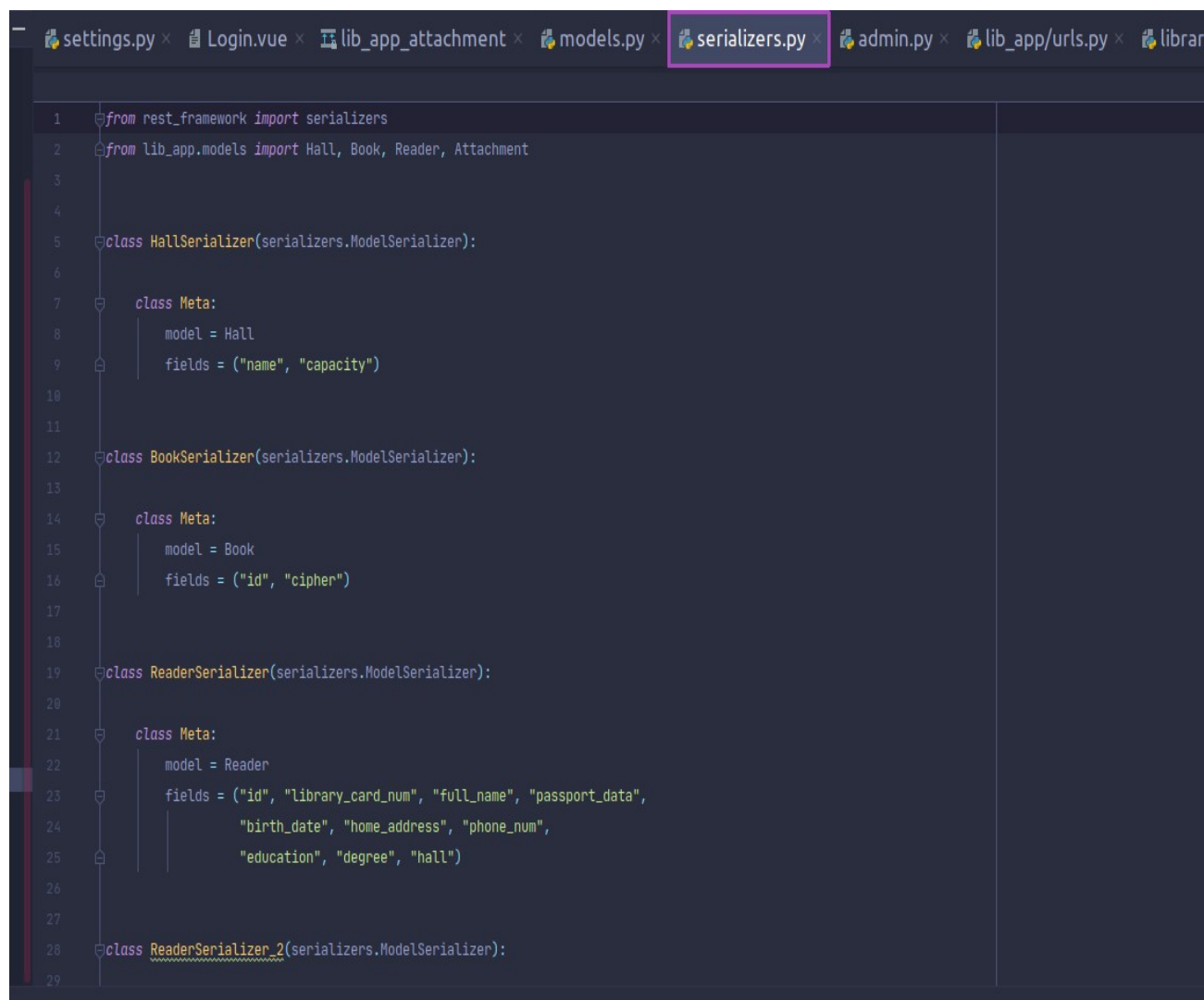
```
1 from django.urls import path
2 from lib_app.views import *
3
4 urlpatterns = [
5     path('hall/', Hall_V.as_view()),
6     path('book/', Book_V.as_view()),
7     path('reader_add/', Reader_V.as_view()),
8     path('attachment/', Attachment_V.as_view()),
9     path('reader/', Reader_books.as_view()),
10    path('detach/<int:pk>', Detach.as_view({'put': 'put'})),
11    path('reader_change/<int:pk>', Reader_change.as_view({'put': 'put'})),
12    path('books/', Books_V.as_view()),
13    path('reader_del/<int:pk>', Reader_del.as_view()),
14    path('book_add/', Book_add.as_view()),
15    path('one_book/', Book_one.as_view()),
16    path('check_att/', Check_att.as_view()),
17    path('reader_get_id/', Reader_get_id.as_view()),
18    path('attachment_books/', Attachment_books.as_view()),
19    path('book_del/<int:pk>', Book_del.as_view()),
20    path('book_change/<int:pk>', Book_change.as_view({'put': 'put'})),
21 ]
```

Рисунок 12 – Содержимое файла urls.py приложения

Здесь прописаны все пути, который вызывают определённые представления, находящиеся в файле views.py.

3.3.3. Сериализаторы

Прежде чем переходить к рассматриванию представлений необходимо поговорить про сериализаторы. Сериализация — это преобразование объекта или дерева объектов в какой-либо формат с тем, чтобы потом эти объекты можно было восстановить из этого формата. Сериализаторы, соответственно, — это то, что осуществляет сериализацию. В Django сериализаторы переводят данные из базы данных в формат json, и расположены в файле serializers.py (рис. 13).



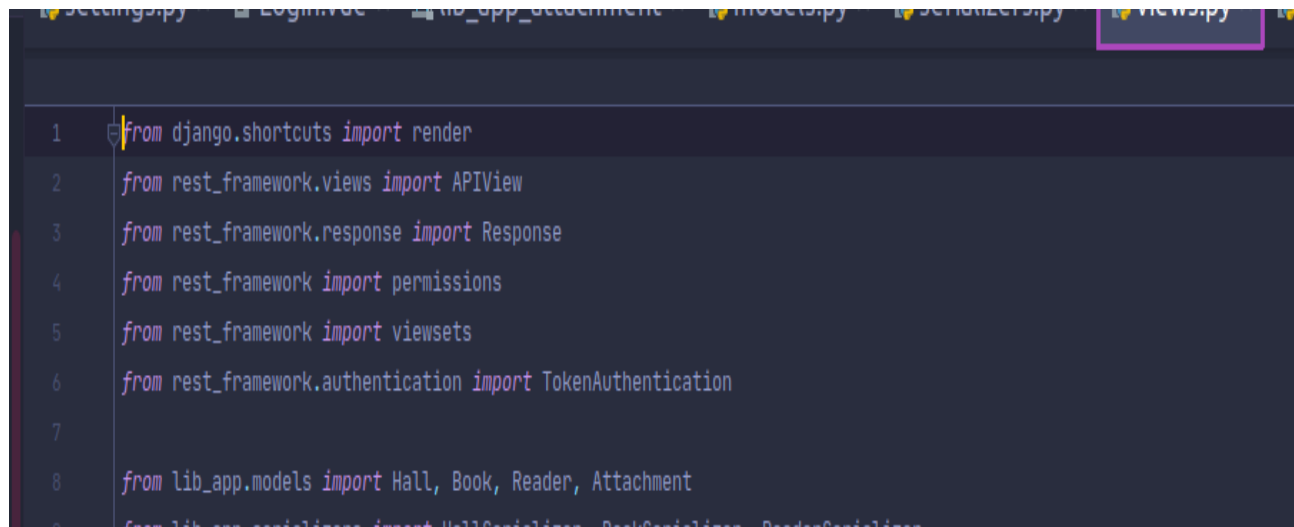
```
1 from rest_framework import serializers
2 from lib_app.models import Hall, Book, Reader, Attachment
3
4
5 class HallSerializer(serializers.ModelSerializer):
6
7     class Meta:
8         model = Hall
9         fields = ("name", "capacity")
10
11
12 class BookSerializer(serializers.ModelSerializer):
13
14     class Meta:
15         model = Book
16         fields = ("id", "cipher")
17
18
19 class ReaderSerializer(serializers.ModelSerializer):
20
21     class Meta:
22         model = Reader
23         fields = ("id", "library_card_num", "full_name", "passport_data",
24                 "birth_date", "home_address", "phone_num",
25                 "education", "degree", "hall")
26
27
28 class ReadersSerializer_2(serializers.ModelSerializer):
29
```

Рисунок 13 – Некоторые сериализаторы и шапка файла serializers.py

Сериализаторы создаются достаточно просто: объявляется модель и задаётся список необходимых полей.

3.3.4. Представления

В Django представления задаются в файле `views.py`. На рис. 14 представлены первые строки данного файла.



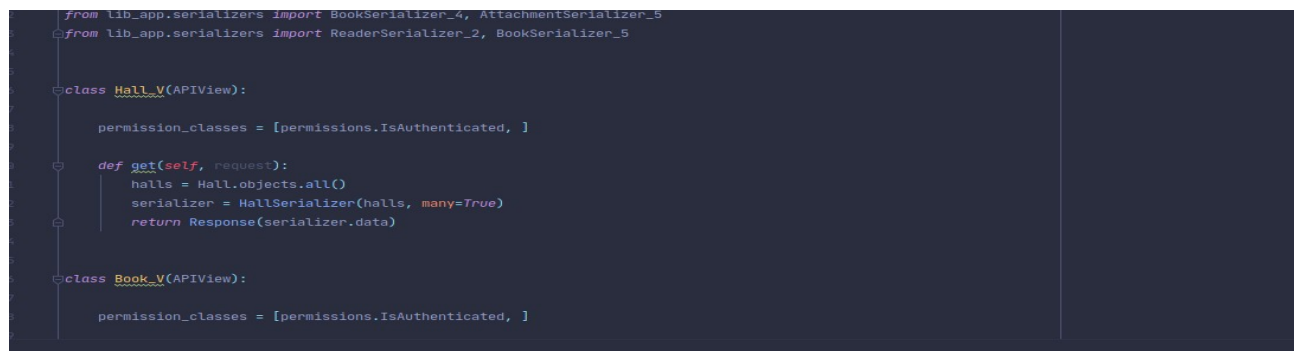
```
1 from django.shortcuts import render
2 from rest_framework.views import APIView
3 from rest_framework.response import Response
4 from rest_framework import permissions
5 from rest_framework import viewsets
6 from rest_framework.authentication import TokenAuthentication
7
8 from lib_app.models import Hall, Book, Reader, Attachment
9 from lib_app.serializers import HallSerializer, BookSerializer, ReaderSerializer
```

Рисунок 14 – Шапка файла `views.py`

Во `views.py` импортируются необходимые для создания представлений компоненты, а также модели и сериализаторы разрабатываемого web-приложения.

Далее рассмотрены представления, вызываемые конкретными путями (path).

1. Путь `'hall/'` вызывает представление `Hall_V`. Его строение представлено на рис. 15.



```
from lib_app.serializers import BookSerializer_4, AttachmentSerializer_5
from lib_app.serializers import ReaderSerializer_2, BookSerializer_5

class Hall_V(APIView):
    permission_classes = [permissions.IsAuthenticated, ]

    def get(self, request):
        halls = Hall.objects.all()
        serializer = HallSerializer(halls, many=True)
        return Response(serializer.data)

class Book_V(APIView):
    permission_classes = [permissions.IsAuthenticated, ]
```

Рисунок 15 – Представление `Hall_V`

В начале задаётся класс пользователей, которые могут вызывать данное представление. Представление `Hall_V` доступно для всех авторизованных пользователей. Далее описываются методы http-запросов. У представления `Hall_V` только один метод: `get`. Он получает все объекты модели `Hall`, сериализует их, используя `HallSerializer` (Приложение А, рис. 1), и выдаёт полученные сериализатором данные в качестве ответа. Посмотреть на эти данные можно по указанному адресу (рис. 16).

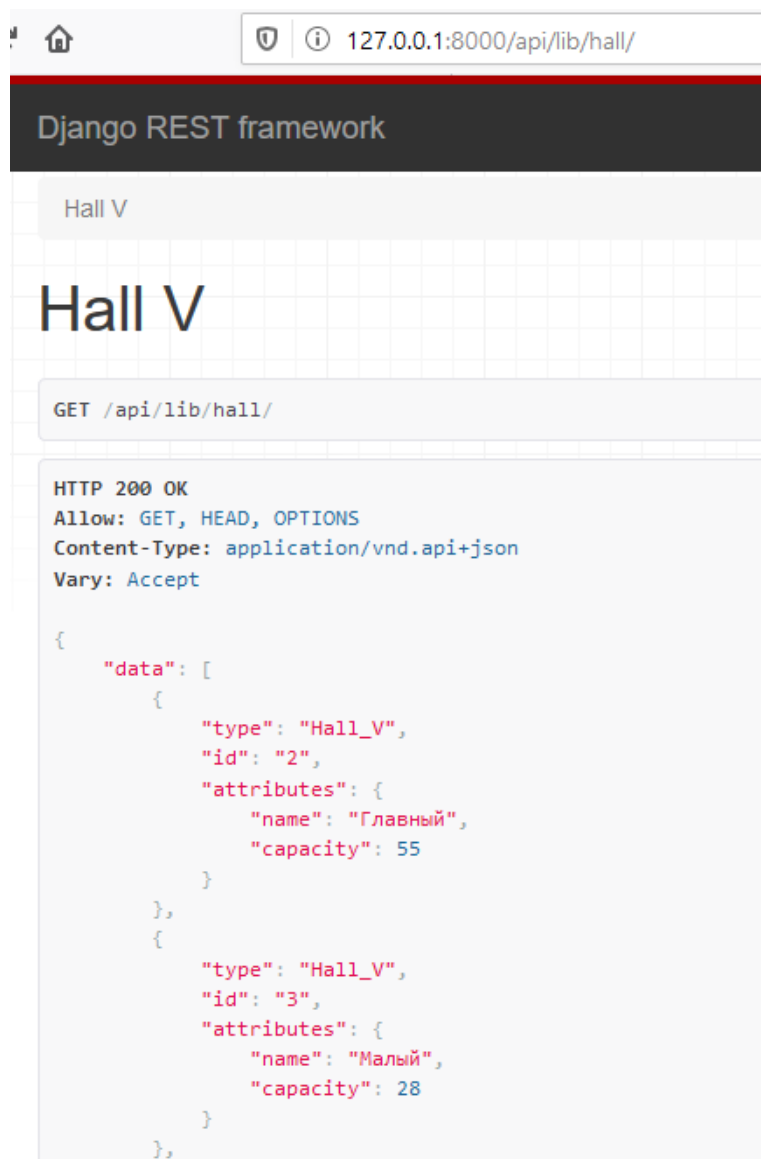


Рисунок 16 – Страница по адресу '127.0.0.1:8000/api/lib/hall/'

2. Путь 'book/' вызывает представление Book_V. Его строение представлено на рис. 17.

```
25
26 class Book_V(APIView):
27
28     permission_classes = [permissions.IsAuthenticated, ]
29
30     def get(self, request):
31         book = request.GET.get("book")
32         books = Book.objects.filter(cipher=book)
33         serializer = BookSerializer(books, many=True)
34         return Response(serializer.data)
```

Рисунок 17 – Представление Book_V

Метод `get` данного представления не просто получает данные из модели, но также оно содержит в запросе переменную, с помощью которых осуществляется фильтрация выдаваемых значений. И в ответ выдаётся сериализация (сериализатор `BookSerializer`, приложение А, рис. 1) уже не всех, а только отфильтрованных (в данном случае, по шифру) объекты модели (рис. 18).

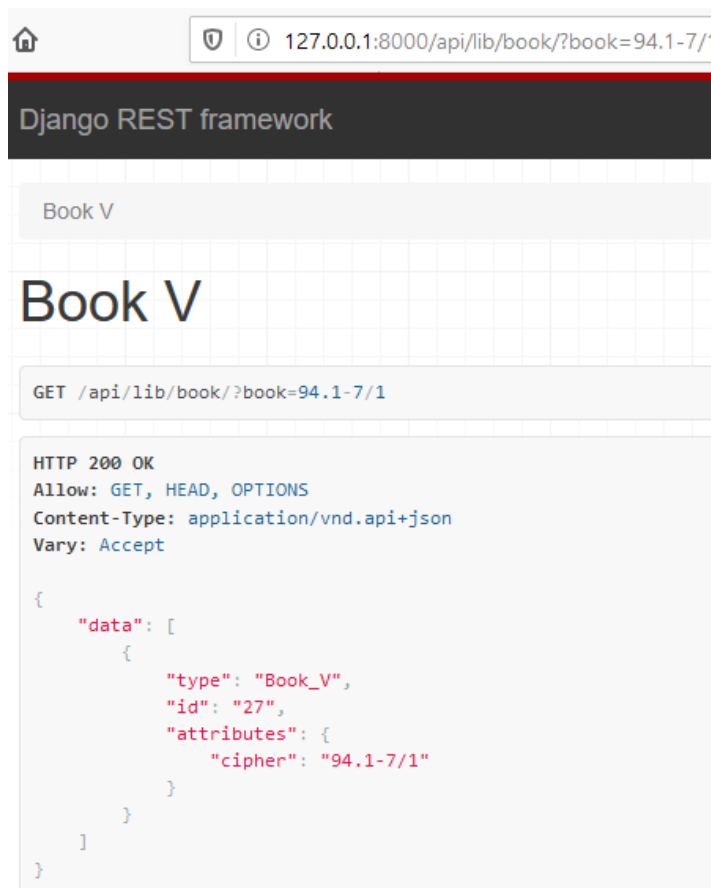


Рисунок 18 – страница по адресу `'127.0.0.1:8000/api/lib/book/?book=94.1-7/1'`

3. Путь `'reader_add/'` вызывает представление `Reader_V`. Его строение представлено на рис. 19.

```

46
47 class Reader_V(APIView):
48
49     permission_classes = [permissions.IsAuthenticated, ]
50
51     def get(self, request):
52         readers = Reader.objects.all()
53         serializer = ReaderSerializer(readers, many=True)
54         return Response(serializer.data)
55
56     def post(self, request):
57         readers = ReaderSerializer(data=request.data)
58         if readers.is_valid():
59             readers.save()
60             return Response({"status": 201})
61         else:
62             return Response({"status": 400})
63
64

```

Рисунок 19 – Представление Reader_V

Данное представление имеет не только метод `get`, но и метод `post`. Первый просто выдаёт сериализованные (сериализатор `ReaderSerializer`, приложение А, рис. 1) объекты модели (рис. 20). Второй же заключается в том, что он получает из запроса данные, которые используются для создания нового объекта модели `Reader`. Затем эти данные проверяются на валидность, и сохраняются.

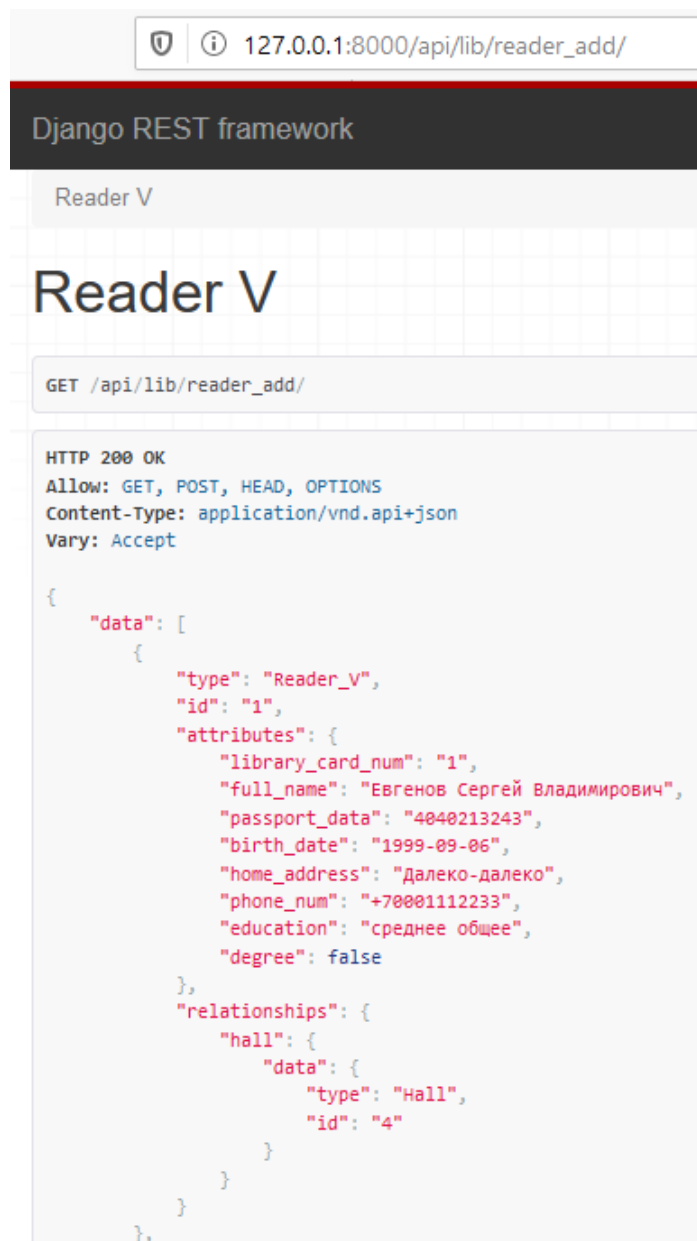


Рисунок 20 – страница по адресу 'http://127.0.0.1:8000/api/lib/reader_add/'

4. Путь 'attachment/' вызывает представление Attachment_V. Его строение представлено на рис. 21.

```

63
64
65 class Attachment_V(APIView):
66
67     permission_classes = [permissions.IsAuthenticated, ]
68
69     def get(self, request):
70         attachments = Attachment.objects.all()
71         serializer = AttachmentSerializer(attachments, many=True)
72         return Response({"data": serializer.data})
73

```

Рисунок 21 – Представление Attachment_V

У представления Attachment_V только один метод: get. Он получает все объекты модели Attachment, сериализует их, используя AttachmentSerializer (Приложение А, рис. 3), и выдаёт полученные сериализатором данные в качестве ответа. Посмотреть на эти данные можно по указанному адресу (рис. 22).

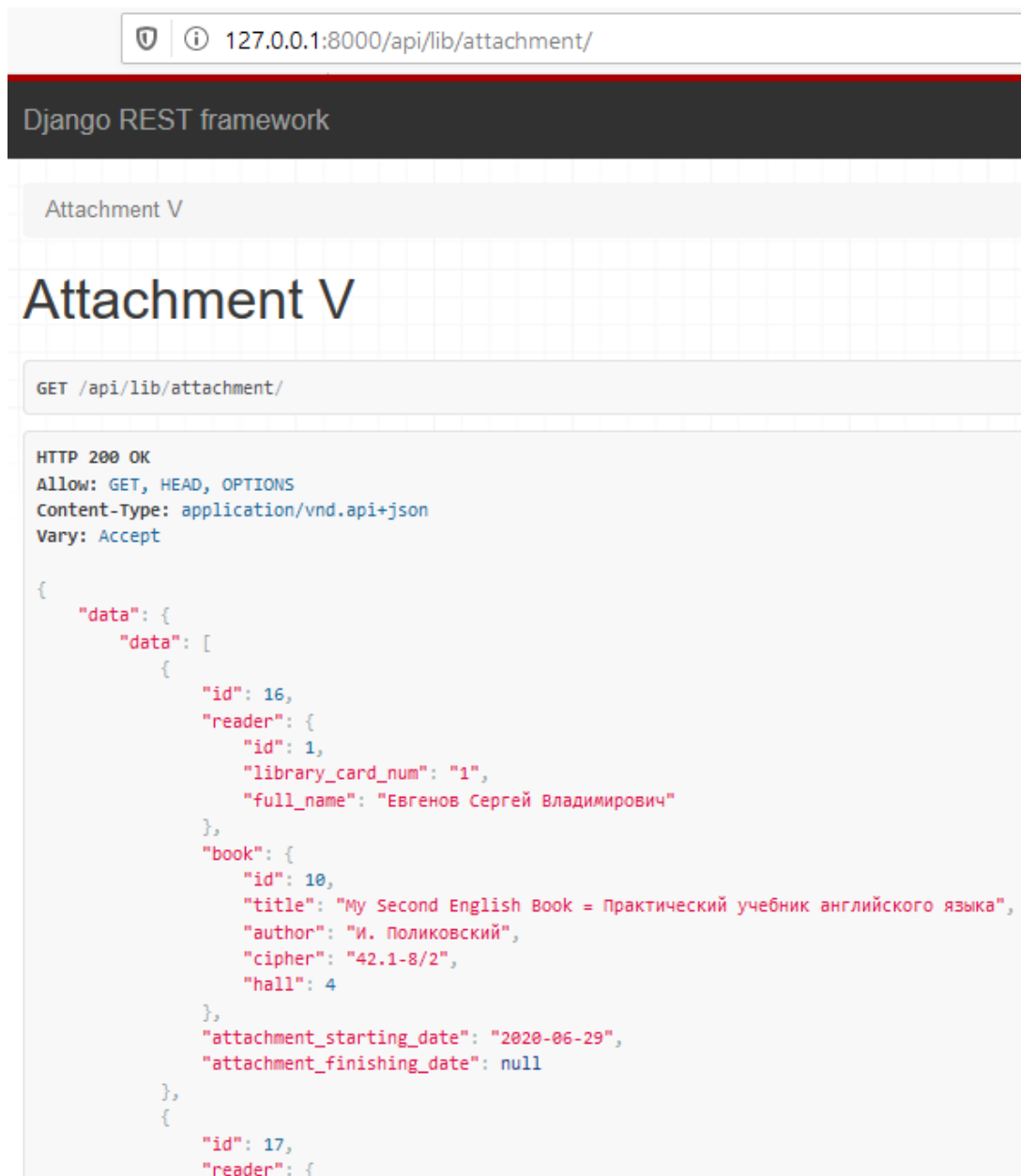


Рисунок 22 – Страница по адресу '127.0.0.1:8000/api/lib/attachment/'

5. Путь 'reader/' вызывает представление Reader_books. Его строение представлено на рис. 23.

```

class Reader_books(APIView):
    permission_classes = [permissions.IsAuthenticated, ]

    def get(self, request):
        person = request.GET.get("reader")
        reader_full = Reader.objects.filter(id=person)
        reader = ReaderSerializer(reader_full, many=True)
        attachments = Attachment.objects.filter(reader=person, attachment_finishing_date=None)
        serializer = AttachmentSerializer_2(attachments, many=True)
        return Response({"reader": reader.data, "books": serializer.data})

    def post(self, request):
        attachments = AttachmentSerializer_3(data=request.data)
        if attachments.is_valid():
            attachments.save()
            return Response({"status": 201})
        else:
            return Response({"status": 400})

class Detach(viewsets.ModelViewSet):

```

Рисунок 23 – Представление Reader_books

Данное представление имеет не только метод `get`, но и метод `post`. Первый для начала находит всю информацию по читателю, чей `id` был передан с запросом, затем сериализует её (сериализатор `ReaderSerializer`, приложение А, рис. 1). После этого находит все закрепления, принадлежащие данному читателю, которые ещё не были закрыты, сериализует их (сериализатор `AttachmentSerializer_2`, приложение А, рис. 3) и в результате возвращает данные по читателю и по его книгам (рис. 24). Метод `post` же заключается в том, что он получает из запроса данные, которые используются для создания нового объекта модели `Attachment`. Затем эти данные проверяются на валидность, и сохраняются.

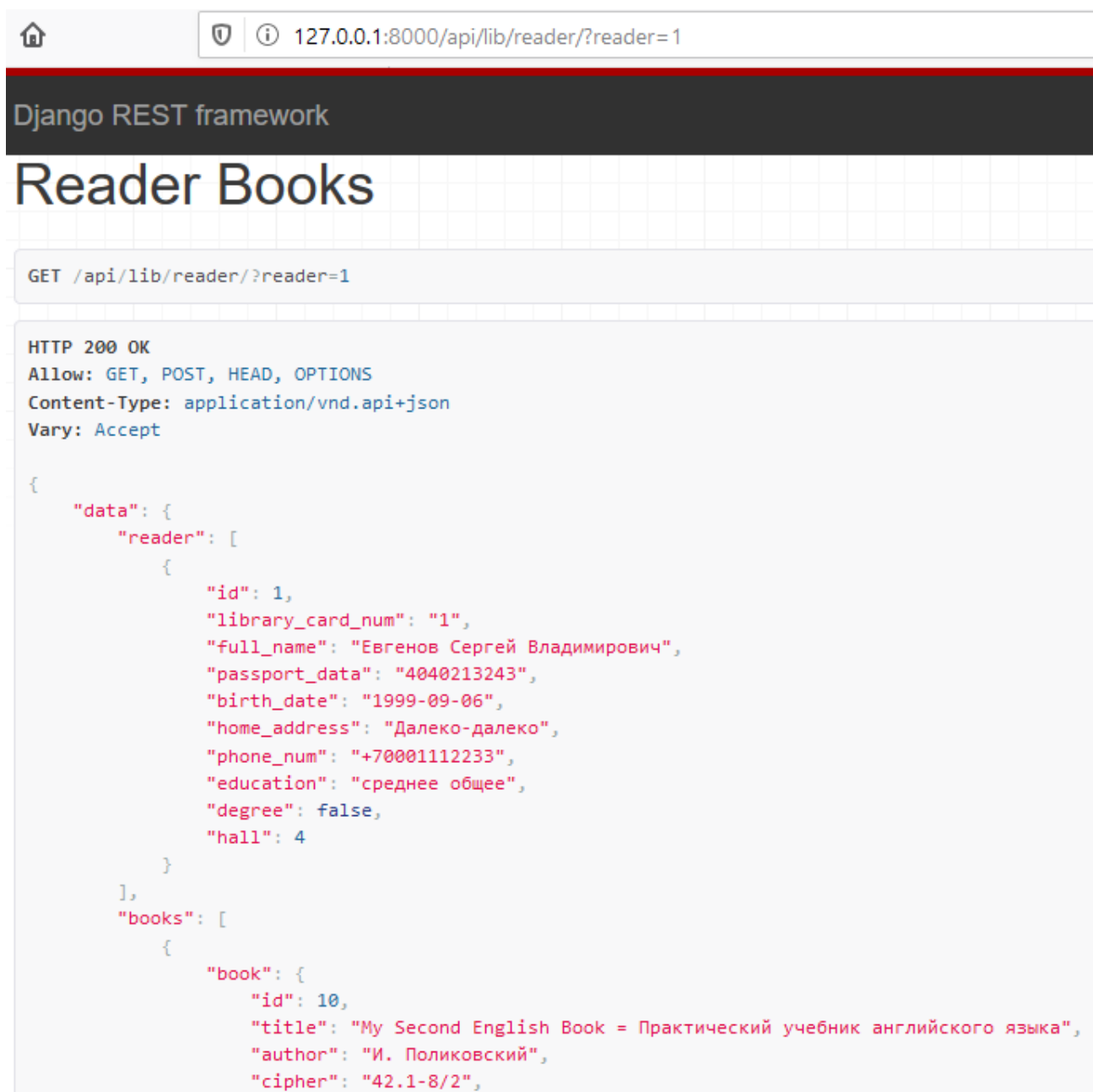


Рисунок 24 – Страница по адресу `'127.0.0.1:8000/api/lib/reader/?reader=1`

6. Путь `'detach/<int:pk>/'` вызывает представление `Detach` со значением `{'put': 'put'}`. Строение данного представления показано на рис. 25.

```

95
96 class Detach(viewsets.ModelViewSet):
97
98     queryset = Attachment.objects.all()
99     serializer_class = AttachmentSerializer_4
100     authentication_classes = [TokenAuthentication, ]
101     permission_classes = [permissions.IsAuthenticated, ]
102
103     def put(self, request, *args, **kwargs):
104         return self.partial_update(request, *args, **kwargs)
105
106

```

Рисунок 25 – Представление Detach

Данное представление имеет исключительно метод `put`. Также, в адресной строке передаётся переменная, `id` закрепления, которое изменяет данный метод. Для начала определяется *queryset*, в который входят все объекты модели `Attachment`. Далее определяется используемый сериализатор (сериализатор `AttachmentSerializer_4`, приложение А, рис. 4). А после уже в методе `put` к *queryset* у применяются частичные изменения.

7. Путь `'reader_del/<int:pk>/'` вызывает представление `Reader_del`. Строение данного представления показано на рис. 26.

```

118 class Reader_del(APIView):
119
120     permission_classes = [permissions.IsAuthenticated, ]
121
122     def delete(self, request, pk):
123         reader = Reader.objects.get(pk=pk)
124         reader.delete()
125         return Response({"status": "Delete"})
126
127

```

Рисунок 26 – Представление Reader_del

Данное представление имеет исключительно метод `delete`. Также, в адресной строке передаётся переменная, `id` читателя, которого удаляет данный метод. Сначала в этом методе берётся нужный объект модели `Reader`. А затем он удаляется, и возвращается статус, который сообщает, что объект был удалён.

Последующие представления похожи на описанные ранее и отдельно описываться не будут. С ними можно ознакомиться в приложении Б, а со страницами, которые появляются при переходе по пути этих представлений, – в приложении В.

3.4. Выводы

В результате разработки серверной части web-приложения были прописаны сериализаторы в файл `serializers.py`, пути – в файл `urls.py` и представления – в файл `views.py`. С помощью изменения данных трёх файлов были успешно решены почти все задачи, которые были поставлены в рамках выполнения функциональных требований, что будет продемонстрировано позднее, в описании клиентской части разрабатываемого web-приложения.

4. Клиентская часть

Клиентская часть разрабатываемого web-приложения была реализована с помощью средств JavaScript-фреймворка Vue.js.

4.1. Теория

Для начала стоит сказать, что такое JavaScript. JavaScript – это мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Он обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java. Языком JavaScript не владеет какая-либо компания или организация, что отличает его от ряда языков программирования, используемых в веб-разработке.

Название «JavaScript» является зарегистрированным товарным знаком корпорации Oracle в США.

Также, необходимо определить, чем является Node.js. Node.js – это программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи NW.js, AppJS или Electron для Linux, Windows и macOS) и даже программировать микроконтроллеры (например, tessel, low.js и espruino). В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

И теперь можно сказать, что Vue.js – это JavaScript-фреймворк на основе Node.js с открытым исходным кодом для создания пользовательских интерфейсов. Легко интегрируется в проекты с использованием других JavaScript-библиотек. Может

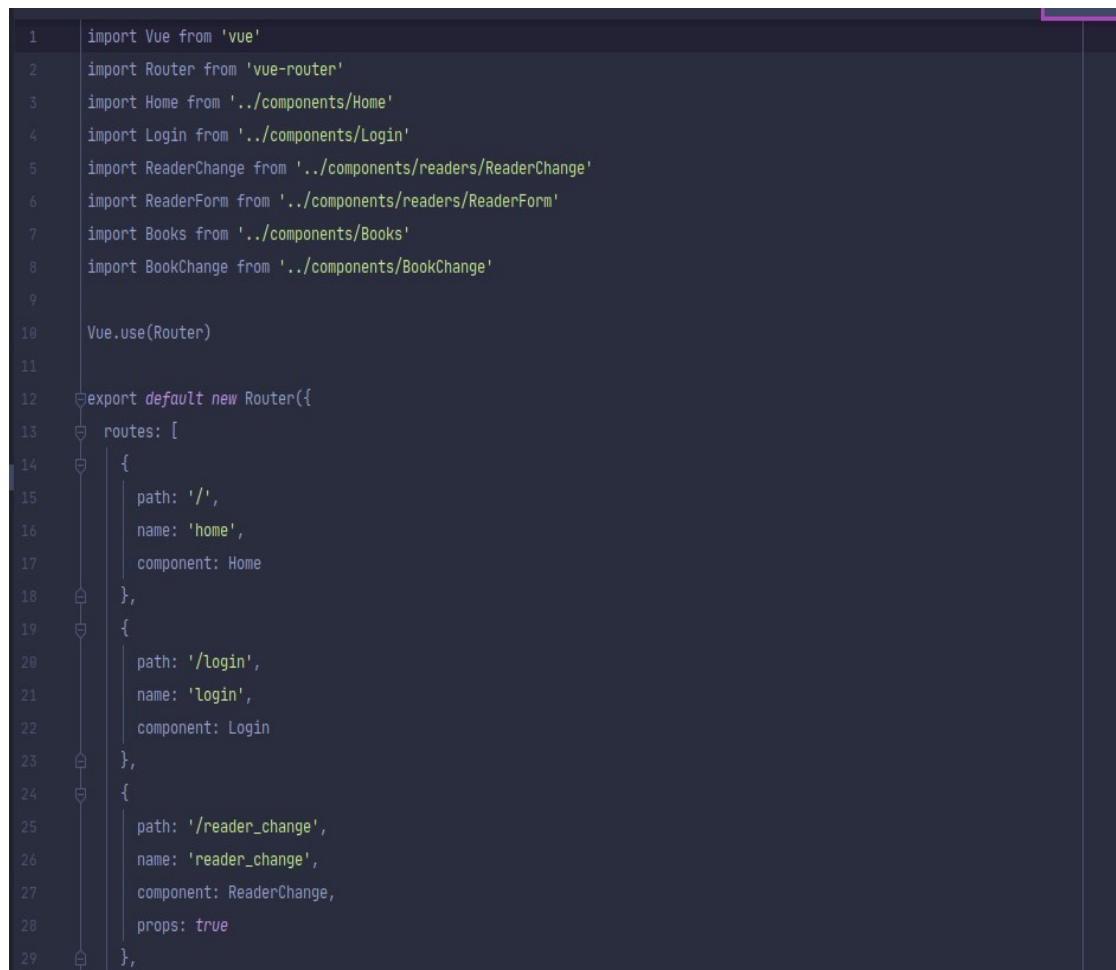
функционировать как веб-фреймворк для разработки одностраничных приложений в реактивном стиле [6].

4.2. Компоненты

Компоненты – это переиспользуемые экземпляры Vue со своим именем, файлы .vue. Их можно использовать как пользовательский тег внутри корневого экземпляра Vue, созданного с помощью `new Vue`. Так как компоненты это переиспользуемые экземпляры Vue, то они принимают те же опции что и `new Vue`, такие как `data`, `computed`, `watch`, `methods`, хуки жизненного цикла [6].

4.2.1. Файл index.js

Файл `index.js` играет примерно ту же роль, что `urls.py` в серверной части web-приложения. В нём прописывается, какие компоненты будут загружаться по какому адресу. На рис. 27 и рис. 28 приведено содержание данного файла.



```
1  import Vue from 'vue'
2  import Router from 'vue-router'
3  import Home from '../components/Home'
4  import Login from '../components/Login'
5  import ReaderChange from '../components/readers/ReaderChange'
6  import ReaderForm from '../components/readers/ReaderForm'
7  import Books from '../components/Books'
8  import BookChange from '../components/BookChange'
9
10  Vue.use(Router)
11
12  export default new Router({
13    routes: [
14      {
15        path: '/',
16        name: 'home',
17        component: Home
18      },
19      {
20        path: '/login',
21        name: 'login',
22        component: Login
23      },
24      {
25        path: '/reader_change',
26        name: 'reader_change',
27        component: ReaderChange,
28        props: true
29      },
30      {
31        path: '/book_change',
32        name: 'book_change',
33        component: BookChange,
34        props: true
35      }
36    ]
37  })
```

Рисунок 27 – Содержимое файла `index.js` (часть 1)

```

{
  path: '/login',
  name: 'login',
  component: Login
},
{
  path: '/reader_change',
  name: 'reader_change',
  component: ReaderChange,
  props: true
},
{
  path: '/reader_form',
  name: 'reader_form',
  component: ReaderForm
},
{
  path: '/books',
  name: 'books',
  component: Books
},
{
  path: '/book_change',
  name: 'book_change',
  component: BookChange,
  props: true
}
]
}))

```

Рисунок 28 – Содержимое файла index.js (часть 2)

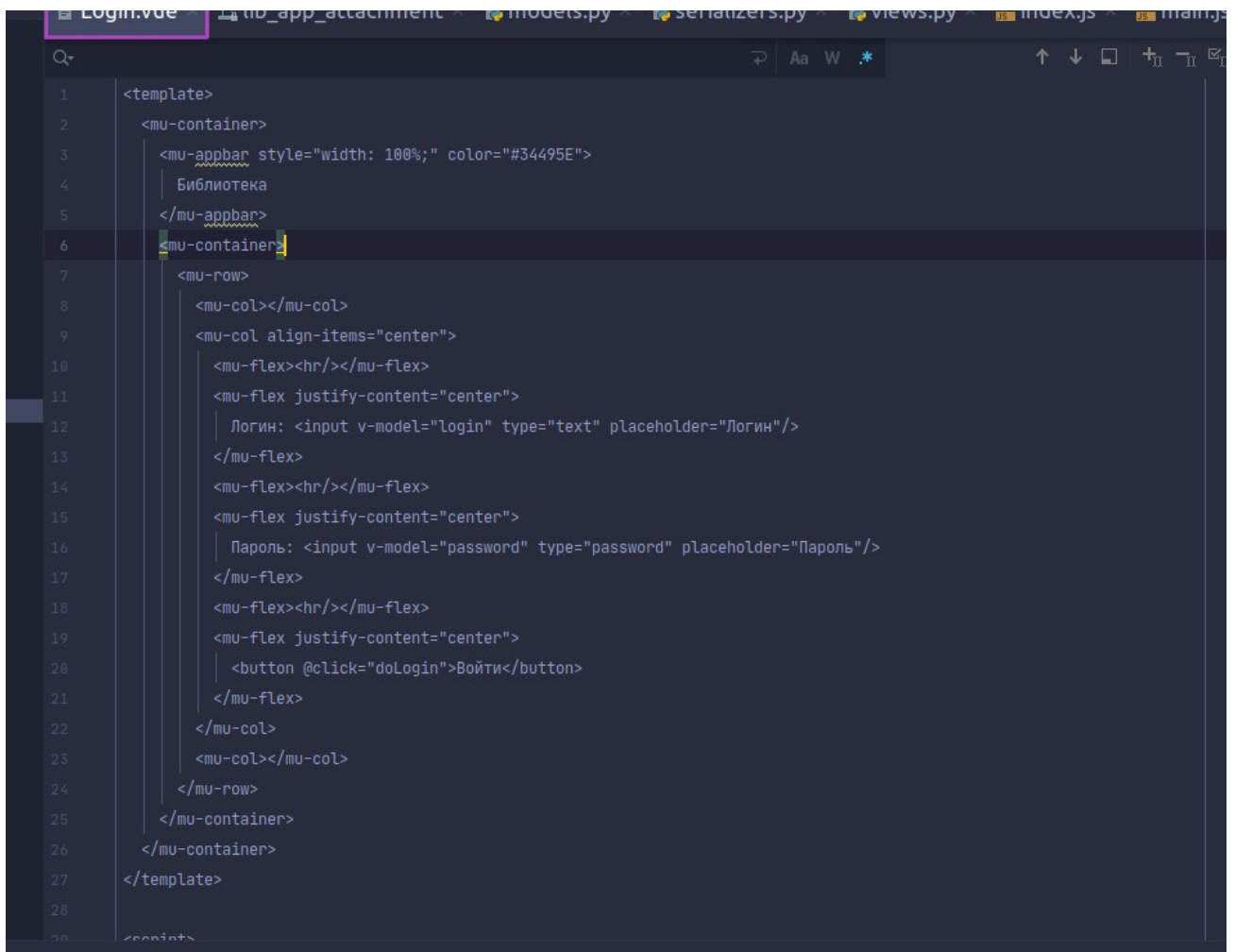
Как мы видим, в шапке файла импортируются необходимые компоненты, далее запускается роутер, и ниже приведён список объектов, в каждом из которых прописан путь, название данного пути/объекта и загружаемый компонент. Также, в некоторых объектах прописан атрибут *props*, который при значении *true* позволяет при переходе по данному пути передавать значения переменных.

4.3. Компонент Home

Home – основной компонент разрабатываемого web-приложения.

4.3.1. Код

В данном приложении в компонент



```
1 <template>
2   <mu-container>
3     <mu-appbar style="width: 100%;" color="#34495E">
4       Библиотека
5     </mu-appbar>
6     <mu-container>
7       <mu-row>
8         <mu-col></mu-col>
9         <mu-col align-items="center">
10           <mu-flex><hr/></mu-flex>
11           <mu-flex justify-content="center">
12             |  Login: <input v-model="login" type="text" placeholder="Login"/>
13           </mu-flex>
14           <mu-flex><hr/></mu-flex>
15           <mu-flex justify-content="center">
16             |  Пароль: <input v-model="password" type="password" placeholder="Пароль"/>
17           </mu-flex>
18           <mu-flex><hr/></mu-flex>
19           <mu-flex justify-content="center">
20             |  <button @click="doLogin">Войти</button>
21           </mu-flex>
22         </mu-col>
23       </mu-col></mu-col>
24     </mu-row>
25   </mu-container>
26 </mu-container>
27 </template>
28
29 <script>
```

енте Home прописано меню приложения, а также в него загружается компонент Reader (рис. 29).

```

1 <template>
2   <mu-container>
3     <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
4       rel="stylesheet">
5     <mu-appbar style="width: 100%;" color="#34495E">
6       <mu-menu slot="left" v-if="auth">
7         <mu-button flat icon>
8           <mu-icon value="menu"></mu-icon>
9         </mu-button>
10        <mu-list slot="content">
11          <mu-list-item button>
12            <mu-list-item-content>
13              <mu-list-item-title>Читатели</mu-list-item-title>
14            </mu-list-item-content>
15          </mu-list-item>
16          <mu-list-item button @click="goBooks()">
17            <mu-list-item-content>
18              <mu-list-item-title>Книги</mu-list-item-title>
19            </mu-list-item-content>
20          </mu-list-item>
21        </mu-list>
22      </mu-menu>
23      Библиотека
24      <mu-button v-if="!auth" @click="goLogin" flat slot="right">Вход</mu-button>
25      <mu-button v-else @click="logout" flat slot="right">Выход</mu-button>
26    </mu-appbar>
27    <reader v-if="auth" class="reader-class"></reader>
28  </mu-container>
29 </template>

```

Рисунок 29 – Тэг <template ...> в файле Home.vue

Тэг <template> – это один из трёх тегов файла .vue. В нём прописывается html-код, описывающий внешний вид компонента. Также здесь загружается набор иконок, одна из которых используется на странице. Два других тэга файла .vue – это <script> (рис. 30), в котором прописан код на языке JavaScript, который выполняется в данном компоненте, и <style> – необязательный тег, в котором прописываются стили CSS, используемые в данном файле (рис. 31).

Как видно по рис. 30, в теге <script ...> импортируются необходимые компоненты с помощью команд(ы) import и экспортируются все данные, методы и т.д. с помощью команды export default {}. Первым экспортируется имя компонента, затем импортированные сюда другие компоненты, дальше вычисляемые значения и в конце методы. В случае большинства компонентов также экспортируются инструкции, которые выполняются при открытии страницы (created () {...}), и множество переменных с различными значениями (data () {return {...}}). Это будет присутствовать в других компонентах.

4.3.2. Отображение на сайте

На рис. 32 представлен вид данного компонента в браузере.

The screenshot shows a web interface for a library. At the top, there is a dark blue header with a hamburger menu icon on the left, the word "Библиотека" in the center, and "ВЫХОД" on the right. Below the header, the page is divided into two main sections. The left section is titled "Читатели:" and contains a search form with a label "Поиск читателя по номеру билета:" and a text input field. Below the input field is a blue button labeled "ОТКРЫТЬ". To the right of the search form is a green button labeled "ДОБАВИТЬ НОВОГО ЧИТАТЕЛЯ". The right section is titled "Читатель Родченский Александр" and displays his details: "Номер читательского билета: 2342334", "Зал: 1", "Адрес: Кировка", "Паспортные данные: 23233223", "Дата рождения: 1994-10-14", "Контактный номер: 234234342234", "Образование: бакалавр", and "Отсутствует учёная степень". Below these details is a purple button labeled "ИЗМЕНИТЬ ДАННЫЕ О ЧИТАТЕЛЕ". In the center, there is a section titled "Закреплённые за читателем в данный момент книги:" which lists two books: "Кирилл Сосновский" (билет №12345) and "Родченский Александр" (билет №2342334). Below this list is a form to "Добавить новое закрепление книги за данным читателем:" with fields for "Шифр книги" and "Дата закрепления" (with a note "В формате: год(4 цифры)-месяц-день"). A blue button labeled "ДОБАВИТЬ" is at the bottom of this form.

Рисунок 32 – Вид компонента Home в браузере (вместе с компонентами Reader и Reader_books)

Меню приложения одинаково по всем адресам сайта, кроме адреса '/login'. Справа выпадающий список из двух основных страниц сайта со ссылками на них (рис. 33). Посередине название сайта, а слева расположена кнопка, при нажатии на которую пользователь разлогинивается и переходит на страницу по адресу '/login'.

This screenshot is identical to the previous one, but it highlights the dropdown menu that appears when the user profile is clicked. The menu is located on the left side of the page, below the header. It contains two items: "Читатели" and "Книги". The "Читатели" item is currently selected, and the "Книги" item is visible below it. The rest of the page content remains the same.

Рисунок 33 – Выпадающее меню аппарата

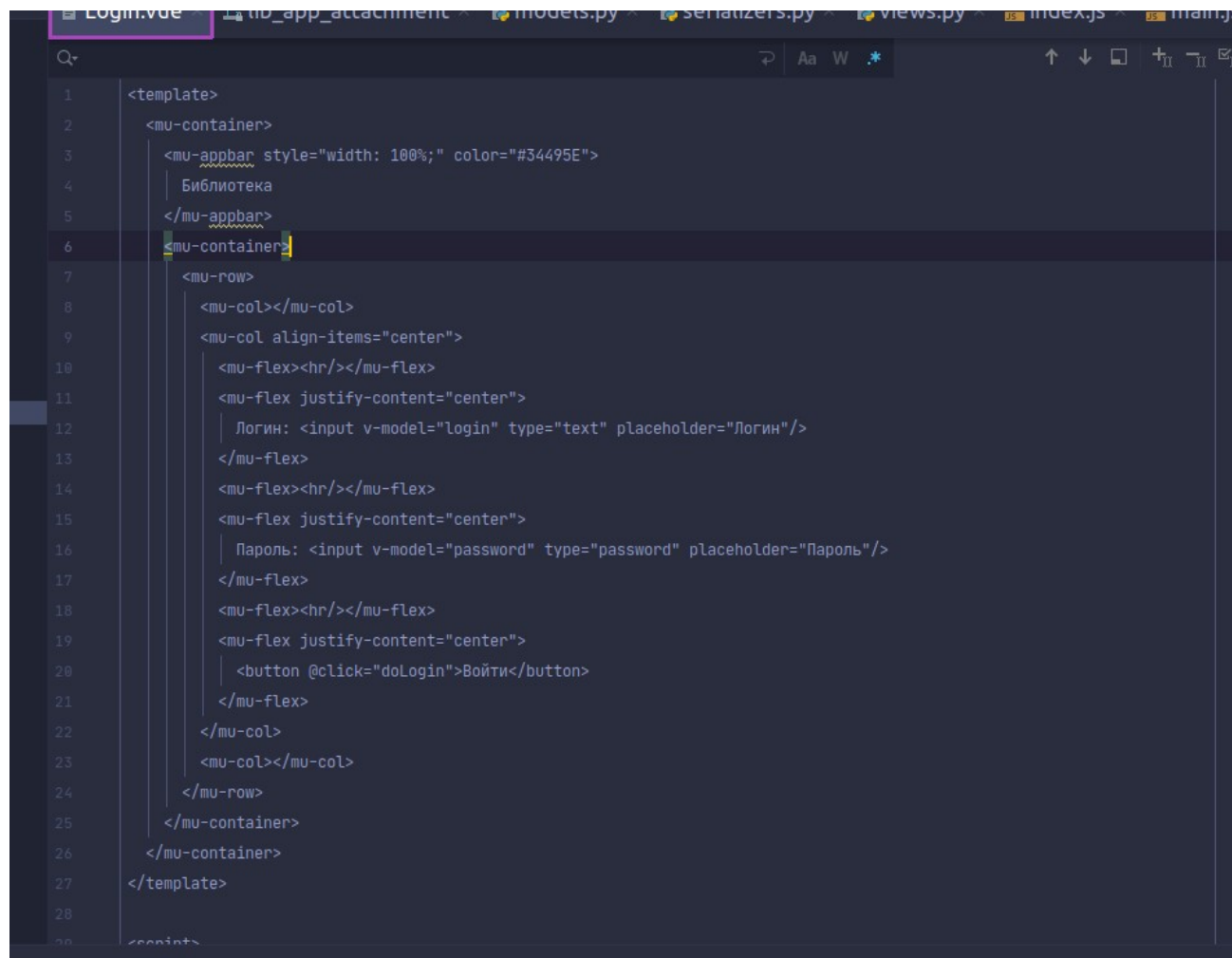
Остальное содержимое данной страницы сайта лежит в других компонентах, которые будут рассмотрены далее.

4.4. Компонент Login

Login – компонент, отвечающий за авторизацию на сайте, открывается по пути '/login'.

4.4.1. Код

На рис. 34, рис. 35 и рис. 36 представлен весь код компонента Login.



```
1 <template>
2   <mu-container>
3     <mu-appbar style="width: 100%;" color="#34495E">
4       Библиотека
5     </mu-appbar>
6   </mu-container>
7   <mu-row>
8     <mu-col></mu-col>
9     <mu-col align-items="center">
10      <mu-flex><hr/></mu-flex>
11      <mu-flex justify-content="center">
12        |  Login: <input v-model="login" type="text" placeholder="Логин"/>
13      </mu-flex>
14      <mu-flex><hr/></mu-flex>
15      <mu-flex justify-content="center">
16        |  Пароль: <input v-model="password" type="password" placeholder="Пароль"/>
17      </mu-flex>
18      <mu-flex><hr/></mu-flex>
19      <mu-flex justify-content="center">
20        |  <button @click="doLogin">Войти</button>
21      </mu-flex>
22    </mu-col>
23  </mu-col></mu-col>
24  </mu-row>
25  </mu-container>
26</mu-container>
27</template>
28
29<script>
```

Рисунок 34 – Содержимое файла Login.vue (часть 1)

Внутри кода <template ...> (рис. 34) присутствует тэг <input ...>, который позволяет делать простейшие формы.

```
1 <template>
2   <mu-container>
3     <mu-appbar style="width: 100%;" color="#34495E">
4       Библиотека
5     </mu-appbar>
6     <mu-container>
7       <mu-row>
8         <mu-col></mu-col>
9         <mu-col align-items="center">
10          <mu-flex><hr/></mu-flex>
11          <mu-flex justify-content="center">
12            Логин: <input v-model="login" type="text" placeholder="Логин"/>
13          </mu-flex>
14          <mu-flex><hr/></mu-flex>
15          <mu-flex justify-content="center">
16            Пароль: <input v-model="password" type="password" placeholder="Пароль"/>
17          </mu-flex>
18          <mu-flex><hr/></mu-flex>
19          <mu-flex justify-content="center">
20            <button @click="doLogin">Войти</button>
21          </mu-flex>
22        </mu-col>
23      </mu-col></mu-col>
24    </mu-row>
25  </mu-container>
26 </mu-container>
27 </template>
28
29 <script>
```

Рисунок 35 – Содержимое файла Login.vue (часть 2)

```
36
37   },
38   methods: {
39     doLogin () {
40       // eslint-disable-next-line
41       $.ajax({
42         url: 'http://127.0.0.1:8000/auth/token/login/',
43         type: 'POST',
44         data: {
45           username: this.login,
46           password: this.password
47         },
48         success: (response) => {
49           alert('Вы успешно вошли')
50           sessionStorage.setItem('auth_token', response.data.attributes.auth_token)
51           this.$router.push({name: 'home'})
52         },
53         error: (response) => {
54           if (response.status === 400) {
55             alert('Логин или пароль не верен')
56           }
57         }
58       })
59     }
60   }
61 }
```

Рисунок 36 – Содержимое файла Login.vue (часть 3)

В тэге `<script ...>` в теле операции `export default {}` присутствует ранее упомянутый элемент `data () {return {}}` (рис. 35). Внутри `return`'а данного подобия функции задаются все переменные, которые могут использоваться далее в методах, при обращении к ним `'this.имяПеременной'`. Далее в той же операции прописан метод `doLogin () {...}`, который авторизует пользователя с использованием ајах-запроса и, в случае успеха, переносит его на главную страницу.

4.4.2. Отображение на сайте

На рис. 37 представлен вид данного компонента в браузере.

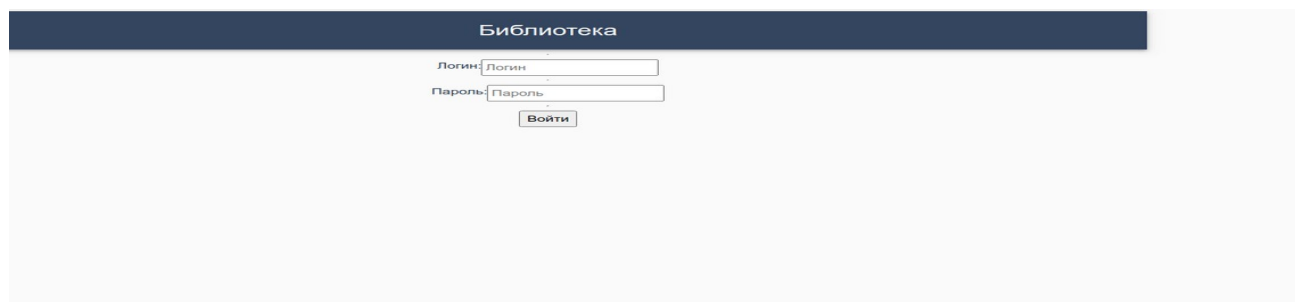


Рисунок 37 – Вид компонента Login в браузере

При правильном вводе логина и пароля и нажатии кнопки «Войти» появляется окошко с уведомлением «Вы успешно вошли» (рис. 38) и открывается основная страница сайта.

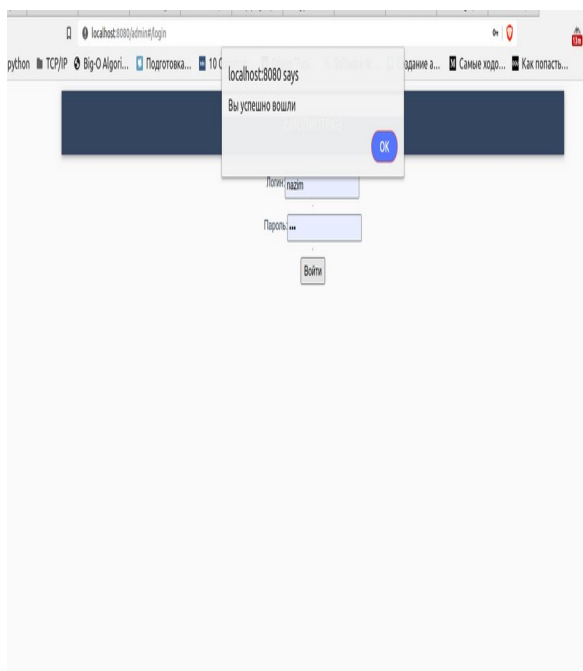


Рисунок 38 – Окошко с уведомлением «Вы успешно вошли» на странице login

4.5. Компонент Reader

Данный компонент отвечает за формирование списка читателей библиотеки, поиск читателя по номеру его билета и отображение карточки читателя (вызов компонента Reader_books).

4.5.1. Код

На рис. 39 представлен код тэга <template ...> компонента Reader.

```
1 <template>
2   <mu-row>
3     <mu-col span="2" xl="2">
4       <h2>Читатели:</h2>
5       Поиск читателя по номеру билета:
6       <mu-flex>
7         <mu-form :model="form" class="reader-card-form" :label-position="labelPosition" label-width="50">
8           <mu-form-item prop="readerId" label="Номер">
9             <mu-text-field v-model="form.reader"></mu-text-field>
10          </mu-form-item>
11        </mu-form>
12        <mu-button color="primary" @click="openByCard()" small class="form-button">
13          Открыть
14        </mu-button>
15      </mu-flex>
16      <mu-button color="success" @click="openReaderForm()" class="new-reader-button">
17        Добавить нового читателя
18      </mu-button>
19    <div v-for="reader in readers" v-bind:key="reader.id" class="readers">
20      <h3 @click="openFull(reader.id)">{{reader.attributes.full_name}}</h3>
21      <small>Читательский билет №{{reader.attributes.library_card_num}}</small><br>
22    </div>
23  </mu-col>
24  <reader_books v-if="reader_full.show" :id="reader_full.id"></reader_books>
25 </mu-row>
26 </template>
27
28 <script>
29   // eslint-disable-next-line
30   import Reader_books from './Reader_books'
```

Рисунок 39 – Тэг <template ...> компонента Reader

Как видно в тэге <template ...>, в данном компоненте прописаны функции openByCard () {} и openReaderForm () {}. Первая отвечает за отображение карточки читателя, чей номер билета был введён в форму, а вторая – за открытие страницы добавления нового пользователя (рис. 40).

```

74     openByCard () {
75       this.readers.forEach(function (value) {
76         if (value.attributes.library_card_num === this.form.reader) {
77           let reader = value.id
78           this.openFull(reader)
79         }
80       }).bind(this))
81     },
82     openReaderForm () {
83       this.$router.push({ 'name': 'reader_form' })
84     },
85   },
86   created () {
87     // eslint-disable-next-line
88     $.ajaxSetup({
89       headers: { 'Authorization': 'Token ' + sessionStorage.getItem('auth_token') })
90     })
91     this.loadReader()
92   }
93 }
94 </script>
95
96 <style scoped>
97 h3 {
98   cursor: pointer
99 }
100 .readers {
101   box-shadow: 1px 2px 3px #888888
102 }

```

Рисунок 40 – Функции openByCard () {} и openReaderForm () {} компонента Reader

Далее в тэге <template ...> есть кнопка вызова функции openByCard () {}. Данная функция включает отображение компонента Reader_books и передаёт в него определённое id читателя (рис. 41).

```

60 },
61 openFull (id) {
62   if (this.reader_full.show === true) {
63     let id2 = id
64     this.reader_full.show = false
65     setTimeout(function (id) {
66       this.reader_full.id = id
67       this.reader_full.show = true
68     }).bind(this), 10, id = id2)
69   } else {
70     this.reader_full.id = id
71     this.reader_full.show = true
72   }
73 },

```

Рисунок 41 – Функция openFull () {} компонента Reader

Целиком код тэгов <script ...> и <style ...> данного компонента приведён на рисунках 1-4 в Приложении Г.

4.5.2. Отображение на сайте

Вид данного компонента в браузере представлен на рис. 32 (занимает левую часть страницы). На рис. 42 изображён он же, но отдельно от других компонентов.

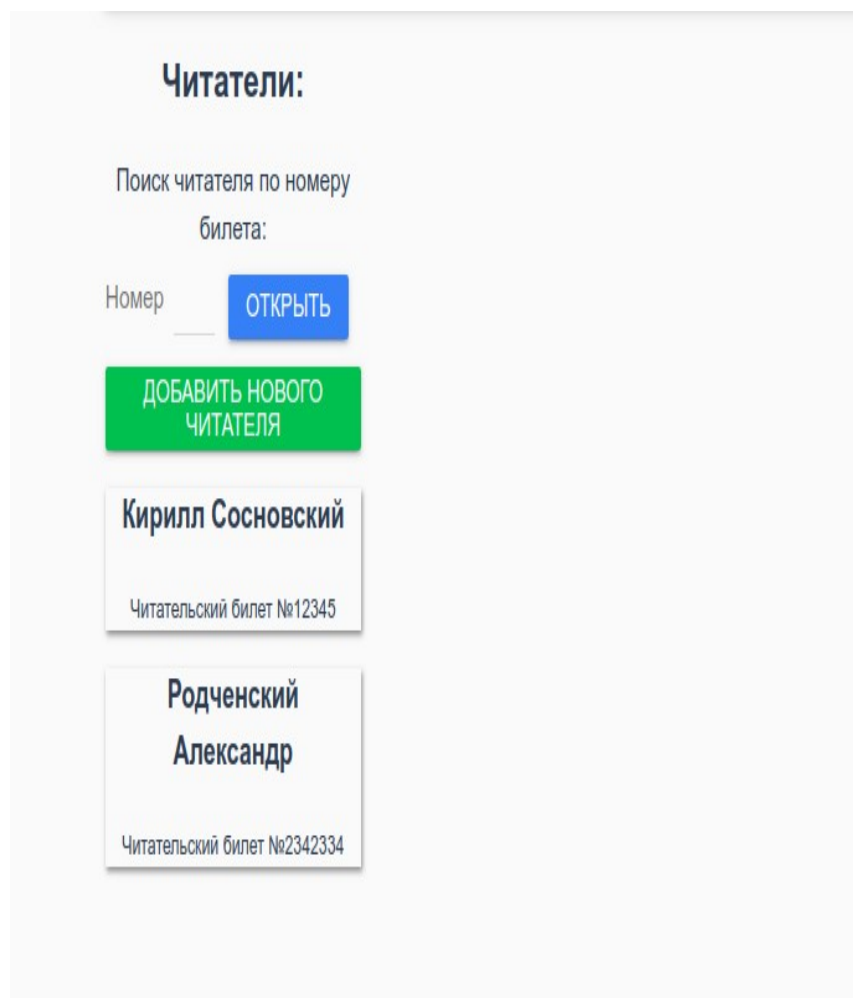


Рисунок 42 – Отображение компонента Reader на главной странице сайта

Функции, вызываемые при нажатии на все представленные кнопки, были описаны в предыдущем пункте.

4.6. Компонент Reader_books

Данный компонент отвечает за отображение карточки читателя и книг, которые за ним закреплены.

4.6.1. Код

На рис. 43 представлен код тэга `<div ...>`, расположенного внутри тэга `<template ...>` компонента Reader_books и отвечающего за отображение карточки читателя.

```

1 <template>
2 <mu-col span="9" xl="9">
3 <mu-container><div v-for="person in person_books.reader" v-bind:key="person.id">
4 <h3>Читатель {{person.full_name}}</h3>
5 <mu-row>
6 <mu-col span="1"></mu-col>
7 <mu-col justify-content="end">
8 <mu-flex>Номер читательского билета: {{person.library_card_num}}</mu-flex>
9 <mu-flex>Зал: {{person.hall}}</mu-flex>
10 <mu-flex>Адрес: {{person.home_address}}</mu-flex>
11 <mu-flex>Паспортные данные: {{person.passport_data}}</mu-flex>
12 </mu-col>
13 <mu-col justify-content="end">
14 <mu-flex>Дата рождения: {{person.birth_date}}</mu-flex>
15 <mu-flex>Контактный номер: {{person.phone_num}}</mu-flex>
16 <mu-flex>Образование: {{person.education}}</mu-flex>
17 <mu-flex v-if="person.degree">Имеется учёная степень</mu-flex>
18 <mu-flex v-else>Отсутствует учёная степень</mu-flex>
19 </mu-col>
20 </mu-row>
21 <mu-flex><hr/></mu-flex>
22 <mu-flex justify-content="end">
23 <mu-button color="indigo400" @click="changeReader(person_books.reader[0])" small>
24 Изменить данные о читателе
25 </mu-button>
26 </mu-flex>
27 <mu-flex><h4>Закреплённые за читателем в данный момент книги:</h4></mu-flex>
28 </div>
29 <div v-for="book in person_books.books" v-bind:key="book.book_cipher">
30 <mu-row>

```

Рисунок 43 – Тэга <div ...>, расположенный внутри тэга <template ...> компонента

Reader_books

В нижней части рис. 43 видно начало описания другого тэга <div ...>, который отображает книги, закреплённые за данным читателем.

Данный компонент имеет в операции export default тэга <script ...> такой элемент, как props (рис. 44). Этот элемент описывает переменные, которые передаются в данный компонент извне (из другого компонента, в приведённом случае, из компонента Reader).

Главный метод компонента Reader_books – это метод по выгрузке книг и данных пользователя, loadBooks () {} (рис. 45).


```

95 |   this.loadBooks()
96 | },
97 | methods: {
98 |   loadBooks () {
99 |     // eslint-disable-next-line
100 |     $.ajax({
101 |       url: 'http://127.0.0.1:8000/api/lib/reader/',
102 |       type: 'GET',
103 |       data: {
104 |         reader: this.id
105 |       },
106 |       success: (response) => {
107 |         this.person_books = response.data
108 |         let idList = []
109 |         for (let j = 0; j < this.person_books.books.length; j++) {
110 |           idList.push(this.person_books.books[j].book.id)
111 |         }
112 |         let maxNum = Math.max(...idList)
113 |         for (let i = 0; i < maxNum + 1; i++) {
114 |           this.detachement_form.push({attachment: '', date: ''})
115 |         }
116 |       }
117 |     })
118 |   },
119 |   addAtt () {
120 |     // eslint-disable-next-line
121 |     $.ajax({
122 |       url: 'http://127.0.0.1:8000/api/lib/book/',

```

Рисунок 45 – Метод loadBooks компонента Reader_books

Целиком код данного компонента приведён на рисунках 5-12 в Приложении Г.

4.6.2. Отображение на сайте

Вид данного компонента в браузере представлен на рис. 32 (занимает правую часть страницы). На рис. 46 изображён он же, но отдельно от других компонентов.

Библиотека выход

Читатели: Читатель Кирилл Сосновский

Поиск читателя по номеру билета:

Номер ОТКРЫТЬ

ДОБАВИТЬ НОВОГО ЧИТАТЕЛЯ

Кирилл Сосновский

Читательский билет №12345

Родченский Александр

Читательский билет №2342334

Номер читательского билета: 12345
Зал: 1
Адрес: Вяземский пер
Паспортные данные: 123452

Дата рождения: 2020-10-14
Контактный номер: 79854547
Образование: бакалавр
Отсутствует учёная степень

ИЗМЕНИТЬ ДАННЫЕ О ЧИТАТЕЛЕ

Закреплённые за читателем в данный момент книги:

Добавить новое закрепление книги за данным читателем:

Шифр книги

Дата закрепления

В формате: год(4 цифры)-месяц-день

ДОБАВИТЬ

Рисунок 46 – Отображение компонента Reader_books на главной странице сайта

По кнопке «Изменить данные о читателе» осуществляется переход на страницу reader_change, при вводе в форму «дата открепления» валидной даты и нажатии кнопки «Открепить» изменяется объект закрепления данной книги за данным читателем (добавляется дата открепления). При заполнении формы внизу и нажатии кнопки «Добавить» за данным читателем закрепляется новая книга.

4.7. Компонент Books

Данный компонент загружается на странице по пути '/books', второй из двух основных страниц сайта. Он отвечает за отображение всех книг библиотеки, поиск среди этих книг, и их добавление.

4.7.1. Код

Тэг <template ...> данного компонента состоит из аппбара и трёх замещающих друг друга вкладок (рис. 47).

```

1 <template>
2 <mu-container>
3   <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
4     rel="stylesheet">
5   <mu-appbar style="width: 100%;" color="#34495E">
6     <mu-menu slot="left">
7       <mu-button flat icon>
8         <mu-icon value="menu"></mu-icon>
9       </mu-button>
10      <mu-list slot="content">
11        <mu-list-item button @click="goHome()">
12          <mu-list-item-content>
13            <mu-list-item-title>Читатели</mu-list-item-title>
14          </mu-list-item-content>
15        </mu-list-item>
16        <mu-list-item button>
17          <mu-list-item-content>
18            <mu-list-item-title>Книги</mu-list-item-title>
19          </mu-list-item-content>
20        </mu-list-item>
21      </mu-list>
22    </mu-menu>
23    Библиотека
24    <mu-button @click="logout()" flat slot="right">Выход</mu-button>
25  </mu-appbar>
26  <mu-tabs :value.sync="active" color="#C08E67">
27    <mu-tab>Список книг</mu-tab>
28    <mu-tab>Поиск и изменение книг</mu-tab>
29    <mu-tab>Добавить новую книгу</mu-tab>
30  </mu-tabs>

```

Рисунок 47 – Сокращённая форма тэга <template ...> компонента Books
Целиком код данного компонента приведён в Приложении Д.

4.7.2. Отображение на сайте

Вид данного компонента в браузере представлен на рис. 48-52.

Показывать только незакреплённые ни за кем в данный момент книги ☐

Показаны все книги

Для внесения изменений нажмите на шифр книги

| Шифр | Название книги | Автор | Издатель | Год издания | Раздел | Дата поступления | Зал |
|--------------|----------------|----------|----------|-------------|--------|------------------|-----|
| 2131232 3 | Лолли и Молли | Гончаров | Екг | 1997 | наука | 2020-10-14 | 1 |

Рисунок 48 – Отображение вкладки «Список книг» компонента Books на сайте

На рис. 48 книги отсортированы по шифру в порядке уменьшения, также виден переключатель, который убирает из отображения книги, которые закреплены за кем-то в данный момент. Как выглядит список книг при включении данного переключателя видно на рис. 49.

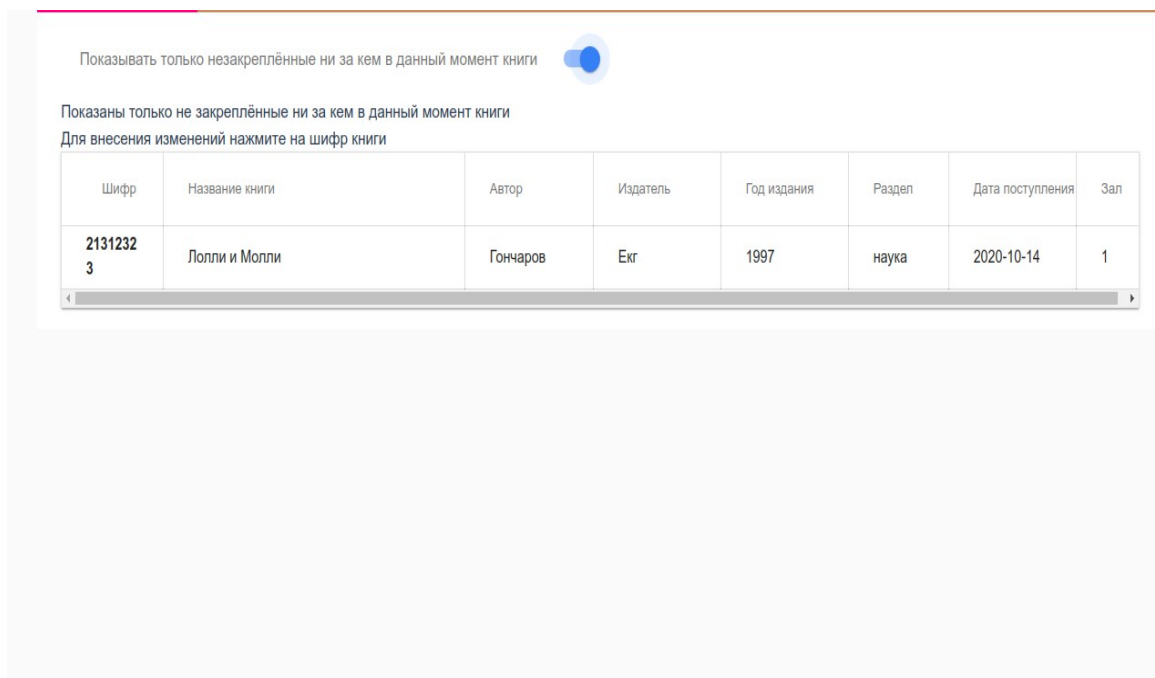


Рисунок 49 – Отображение вкладки «Список книг» компонента Books на сайте при включённом переключателе

На второй вкладке данной страницы осуществляется поиск, изменение, закрепление за читателем и добавление экземпляров книг (рис. 50 и рис. 51). Поиск может осуществляться как по названию, так и по шифру книги. Закрепление книги за читателем доступно только в случае, когда отображены не закрепленные ни за кем книги.

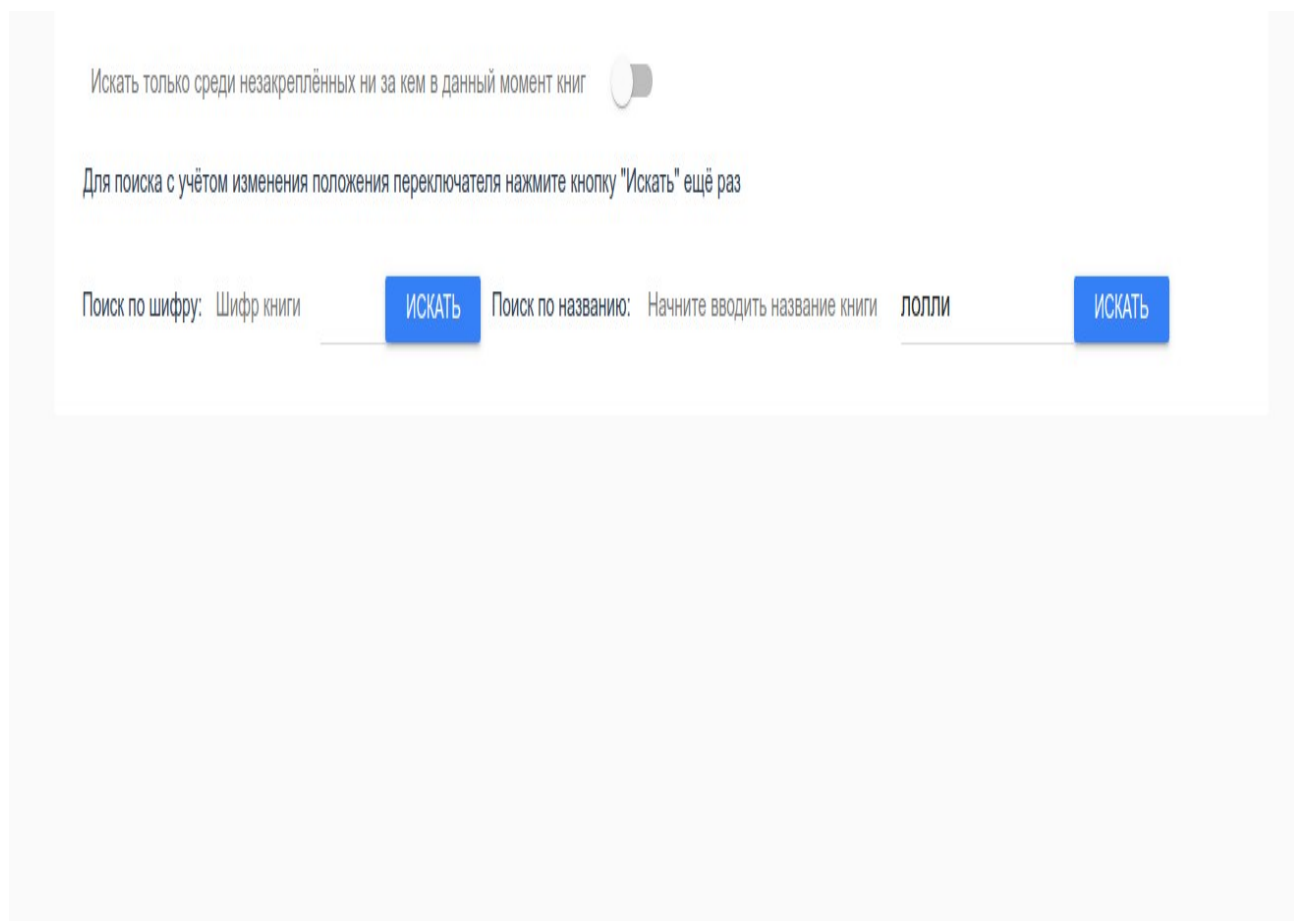


Рисунок 51 – Отображение вкладки «Поиск и изменение» компонента Books на сайте при включённом переключателе

Список книг Поиск и изменение книг Добавить новую книгу

Шифр

Название

Автор

Издатель

Год издания

Раздел

Дата поступления

В формате: год(4 цифры)-месяц-день

Зал

ДОБАВИТЬ КНИГУ

Рисунок 52 – Отображение вкладки «Добавить новую книгу» компонента Books на сайте

4.8. Компоненты ReaderChange, BookChange и ReaderForm

Данные служебные компоненты открываются на отдельных страницах, путь к которым прописан в index.js. Первые два из них позволяют менять соответствующие объекты, а последний добавлять новых читателей. Код данных трёх компонентов приведён в Приложениях Е, Ж и З соответственно. Их отображение представлено на рис. 53-55.

≡

Библиотека

ВЫХОД

Читатель Родченский Александр

УДАЛИТЬ
ЧИТАТЕЛЯ

Нынешние данные:

Номер читательского билета: 2342334

Зал: 1

Адрес: Кировка

Паспортные данные: 23233223

Дата рождения: 1994-10-14

Контактный номер: 234234342234

Образование: бакалавр

Отсутствует учёная степень

Новые данные:

(Заполняйте только те поля, значения которых собираетесь изменить)

Номер читательского билета

Полное имя (ФИО)

Зал

Адрес

Паспортные данные

Дата рождения

В формате: год(4 цифры)-месяц-день

Контактный номер

Образование

Рисунок 53 – Отображение компонента ReaderChange на сайте

| | |
|-----------------------------|--------------------------------|
| Зал: 1 | Контактный номер: 234234342234 |
| Адрес: Кировка | Образование: бакалавр |
| Паспортные данные: 23233223 | Отсутствует учёная степень |

Новые данные:

(Заполняйте только те поля, значения которых собираетесь изменить)

| | |
|----------------------------|------------------------------------|
| Номер читательского билета | <input type="text"/> |
| Полное имя (ФИО) | <input type="text"/> |
| Зал | <input type="text"/> |
| Адрес | <input type="text"/> |
| Паспортные данные | <input type="text"/> |
| Дата рождения | <input type="text"/> |
| | В формате: год(4 цифры)-месяц-день |
| Контактный номер | <input type="text"/> |
| Образование | <input type="text"/> |
| Наличие учёной степени | <input type="text"/> |

Рисунок 54 – Отображение компонента BookChange на сайте

Библиотека

ВЫХОД

Книга 21312323

СПИСАТЬ КНИГУ

Нынешние данные:

Название: Лолли и Молли
Автор: Гончаров
Издатель: Екп
Год издания: 1997
Сфера: наука
Дата поступления: 2020-10-14
Зал: 1

Новые данные:

(Заполняйте только те поля, значения которых собираетесь изменить)

Шифр

Название

Автор

Издатель

Год издания

Раздел

Дата поступления

Рисунок 55 – Отображение компонента ReaderForm на сайте

4.9. Muse-UI

Во всех компонентах разработанного web-приложения в тэгах `<template ...>` были использованы тэги Muse-UI. Muse-Ui – это библиотека дизайна компонентов Vue.js [7].

4.10. Выводы

В результате разработки клиентской части web-приложения были созданы следующие компоненты Vue.js:

1. Home,
2. Reader,
3. Reader_books,
4. ReaderChange,
5. ReaderForm,
6. Books,
7. BookChange,
8. Login.

Данные компоненты позволяют отобразить всё необходимое для web-приложения в заданной предметно области исходя из установленных в первой главе отчёта по данной курсовой работе функциональных требований. Дизайн всех данных компонентов сделан с помощью библиотеки Muse_UI.

5. ЗАКЛЮЧЕНИЕ

За время выполнения курсовой работы были выполнены все поставленные функциональные задачи.

В рамках реализации задачи по отображению карточки читателя и книг, закреплённых за определённым читателем, была использована выгрузка множества компонентов на одну страницу сайта, что позволило максимально эффективно построить интерфейс.

Для реализации страницы сайта, связанной с книгами была использована конструкция с несколькими вкладками, что позволило уместить много функционала на одной странице.

Также были созданы служебные страницы для авторизации на сайте, изменения книг и читателей, а также добавления новых читателей.

Полученное web-приложение уже готово к использованию, однако может быть развито в направлении увеличения удобства отображения книг – группировки экземпляров одной книги вместе. Также, возможно создание различных отчётов, и контроль за посещаемостью библиотеки читателями.

СПИСОК ЛИТЕРАТУРЫ

1. Каталог Российской Национальной Библиотеки [Электронный ресурс] URL: https://primo.nlr.ru/primo-explore/search?vid=07NLR_VU1 (дата обращения: 30.06.2020)
2. ГПИБ России (Историческая библиотека): Практические советы по созданию личной библиотеки // Life Journal [Электронный ресурс] URL: <https://gpib.livejournal.com/29519.html> (дата обращения: 28.06.2020)
3. Django documentation // The Web framework for perfectionists with deadlines [Электронный ресурс] URL: <https://www.djangoproject.com/> (дата обращения: 20.06.2020)
4. Home // Django REST framework [Электронный ресурс] URL: <https://www.django-rest-framework.org/> (дата обращения: 20.06.2020)
5. Курылев А. Что такое Django REST Framework? // MkDev Personal [Электронный ресурс] URL: <https://mkdev.me/posts/chto-takoe-django-rest-framework> (дата обращения: 20.06.2020)
6. Introduction // Vue.js [Электронный ресурс] URL: <https://vuejs.org/v2/guide/> (дата обращения: 25.06.2020)
7. Muse UI [Электронный ресурс] URL: <https://muse-ui.org/#/en-US> (дата обращения: 25.06.2020)

Приложение А. Сериализаторы

```
1  from rest_framework import serializers
2  from lib_app.models import Hall, Book, Reader, Attachment
3
4
5  class HallSerializer(serializers.ModelSerializer):
6
7      class Meta:
8          model = Hall
9          fields = ("name", "capacity")
10
11
12  class BookSerializer(serializers.ModelSerializer):
13
14      class Meta:
15          model = Book
16          fields = ("id", "cipher")
17
18
19  class ReaderSerializer(serializers.ModelSerializer):
20
21      class Meta:
22          model = Reader
23          fields = ("id", "library_card_num", "full_name", "passport_data",
24                  "birth_date", "home_address", "phone_num",
25                  "education", "degree", "hall")
26
27
28  class ReaderSerializer_2(serializers.ModelSerializer):
29
```

Рисунок 1 – Содержимое файла serializers.py (часть 1)

```

47
48
49 class AttachmentSerializer(serializers.ModelSerializer):
50
51     reader = ReaderSerializer_2()
52     book = BookSerializer_2()
53     class Meta:
54         model = Attachment
55         fields = ("id", "reader", "book", "attachment_starting_date",
56                 "attachment_finishing_date")
57
58
59 class AttachmentSerializer_2(serializers.ModelSerializer):
60
61     book = BookSerializer_2()
62     class Meta:
63         model = Attachment
64         fields = ("book", "attachment_starting_date", "id")
65
66
67 class AttachmentSerializer_3(serializers.ModelSerializer):
68
69     class Meta:
70         model = Attachment
71         fields = ("reader", "book", "attachment_starting_date")

```

Рисунок 2 – Содержимое файла serializers.py (часть 2)

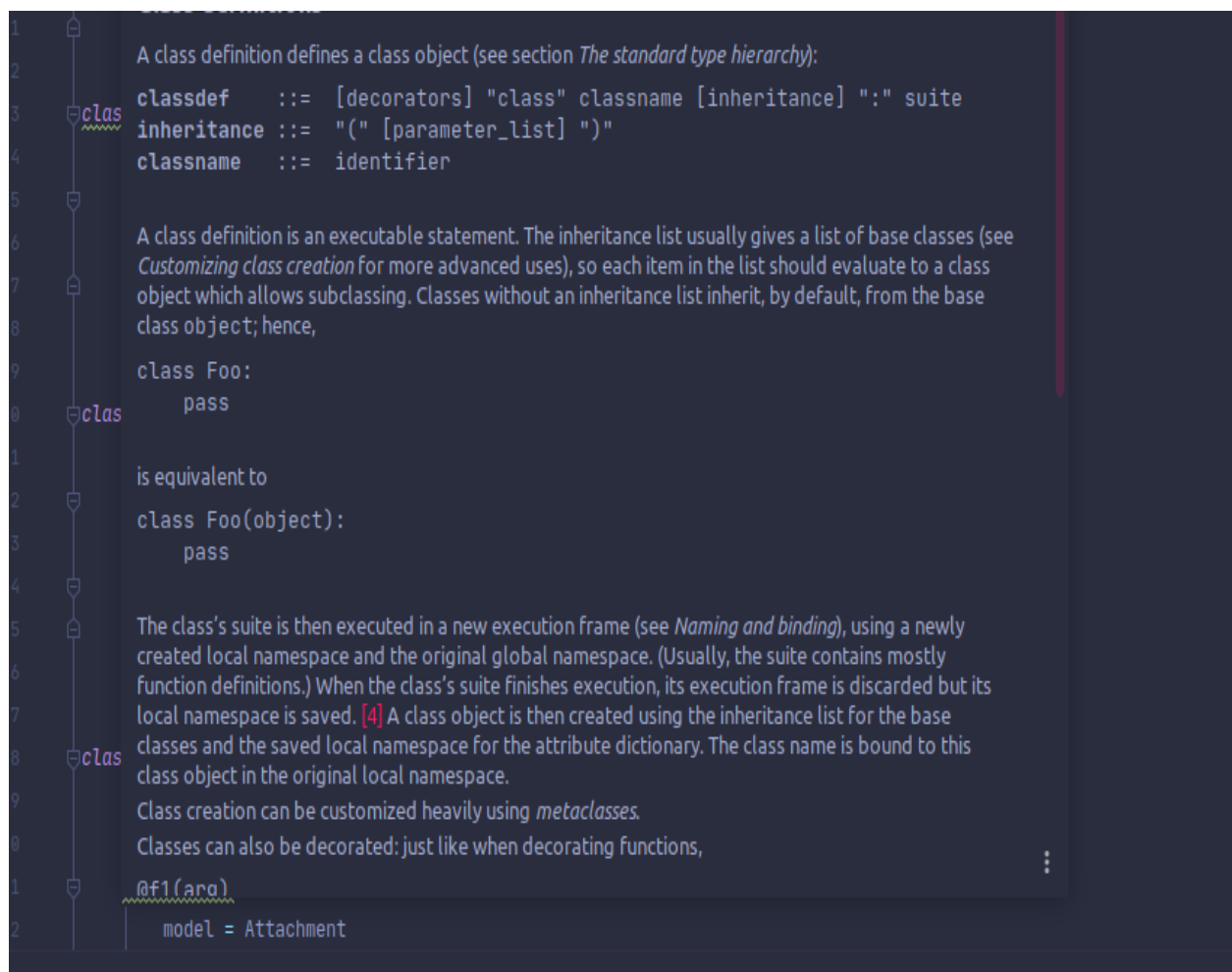


Рисунок 3 – Содержимое файла serializers.py (часть 3)



Рисунок 4 – Содержимое файла serializers.py (часть 4)

Приложение Б. Представления

```
class Detach(viewsets.ModelViewSet):
    queryset = Attachment.objects.all()
    serializer_class = AttachmentSerializer_4
    authentication_classes = [TokenAuthentication, ]
    permission_classes = [permissions.IsAuthenticated, ]

    def put(self, request, *args, **kwargs):
        return self.partial_update(request, *args, **kwargs)

class Reader_change(viewsets.ModelViewSet):
    queryset = Reader.objects.all()
    serializer_class = ReaderSerializer
    authentication_classes = [TokenAuthentication, ]
    permission_classes = [permissions.IsAuthenticated, ]

    def put(self, request, *args, **kwargs):
        return self.partial_update(request, *args, **kwargs)
```

Рисунок 1 – Представление Reader_change

```
class BookViewset(viewsets.ModelViewSet):
    """
    return_stmt ::= "return" [expression_list]
    return may only occur syntactically nested in a function definition, not within a nested class definition.
    If an expression list is present, it is evaluated, else None is substituted.
    return leaves the current function call with the expression list (or None) as return value.
    When return passes control out of a try statement with a finally clause, that finally clause is
    executed before really leaving the function.
    In a generator function, the return statement indicates that the generator is done and will cause
    StopIteration to be raised. The returned value (if any) is used as an argument to construct
    StopIteration and becomes the StopIteration.value attribute.
    """
    return Response(serializer.data)

class Books_V(APIView):
    permission_classes = [permissions.IsAuthenticated, ]

    def get(self, request):
        books = Book.objects.all()
        serializer = BookSerializer_4(books, many=True)
        return Response(serializer.data)
```

Рисунок 2 – Представления Book_add и Book_one


```

74
75 class Reader_books(APIView):
76
77     permission_classes = [permissions.IsAuthenticated, ]
78
79     def get(self, request):
80         person = request.GET.get("reader")
81         reader_full = Reader.objects.filter(id=person)
82         reader = ReaderSerializer(reader_full, many=True)
83         attachments = Attachment.objects.filter(reader=person, attachment_finishing_date=None)
84         serializer = AttachmentSerializer_2(attachments, many=True)
85         return Response({"reader": reader.data, "books": serializer.data})
86
87     def post(self, request):
88         attachments = AttachmentSerializer_3(data=request.data)
89         if attachments.is_valid():
90             attachments.save()
91             return Response({"status": 201})
92         else:
93             return Response({"status": 400})
94
95

```

Рисунок 3 – Представления Check_att и Reader_get_id

```

116
117
118 class Reader_del(APIView):
119
120     permission_classes = [permissions.IsAuthenticated, ]
121
122     def delete(self, request, pk):
123         reader = Reader.objects.get(pk=pk)
124         reader.delete()
125         return Response({"status": "Delete"})
126
127
128 class Book_add(APIView):
129
130     permission_classes = [permissions.IsAuthenticated, ]
131
132     def post(self, request):
133         book = BookSerializer_4(data=request.data)
134         if book.is_valid():
135             book.save()
136             return Response({"status": 201})
137         else:
138             return Response({"status": 400})
139
140
141 class Book_one(APIView):
142
143     permission_classes = [permissions.IsAuthenticated, ]

```

Рисунок 4 – Представления Attachment_books и Book_del

```

permission_classes = [permissions.IsAuthenticated, ]

def get(self, request):
    book_id = request.GET.get("book")
    attachments = Attachment.objects.filter(book=book_id).filter(attachment_f
    serializer = AttachmentSerializer_5(attachments, many=True)
    return Response({"data": serializer.data})

class Book_del(APIView):
    permission_classes = [permissions.IsAuthenticated, ]

```

Рисунок 5 – Представление Book_change

Приложение В. Страницы Django REST Framework'a

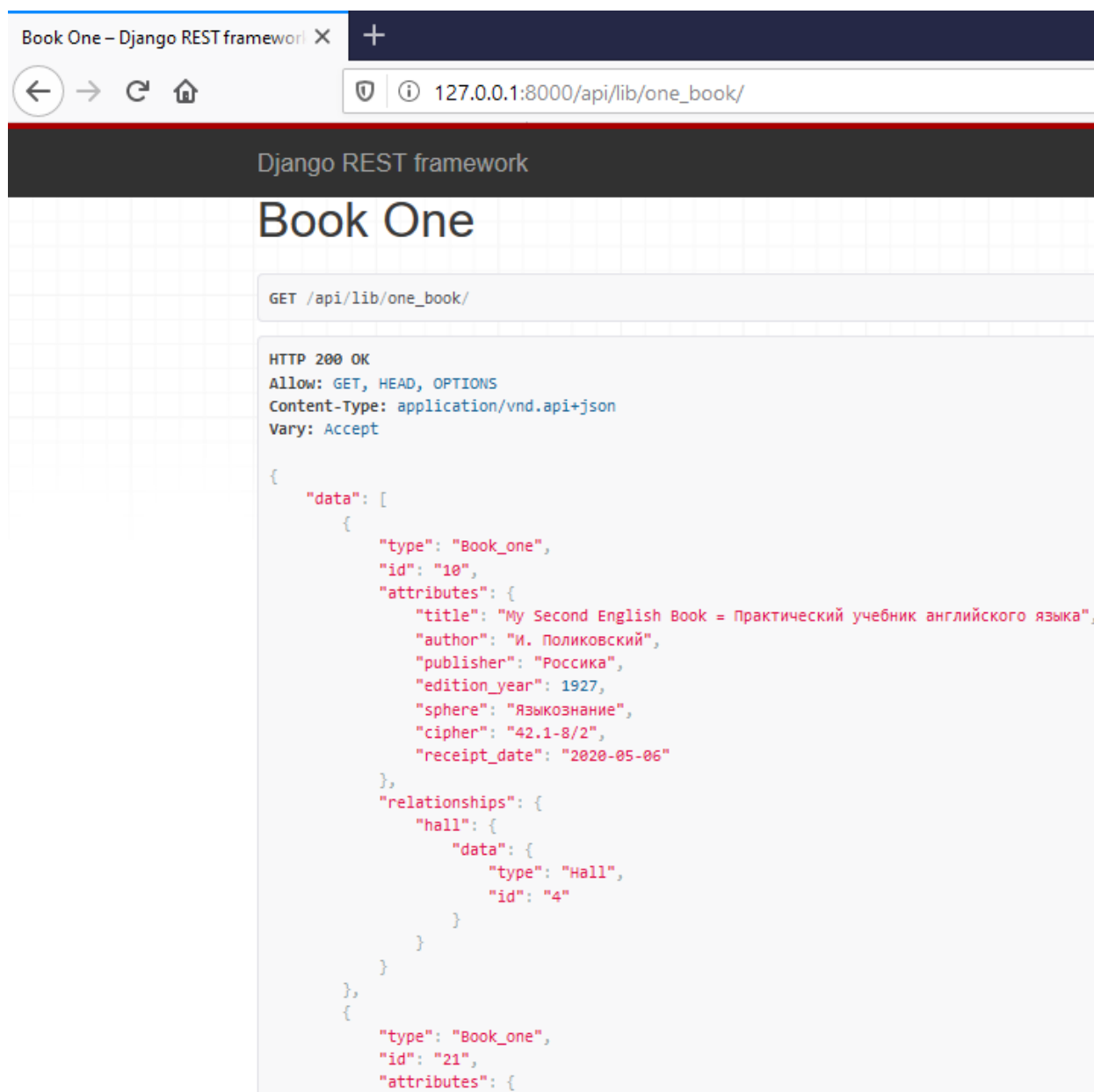


Рисунок 1 – Страница по адресу `'127.0.0.1:8000/api/lib/one_book/'`

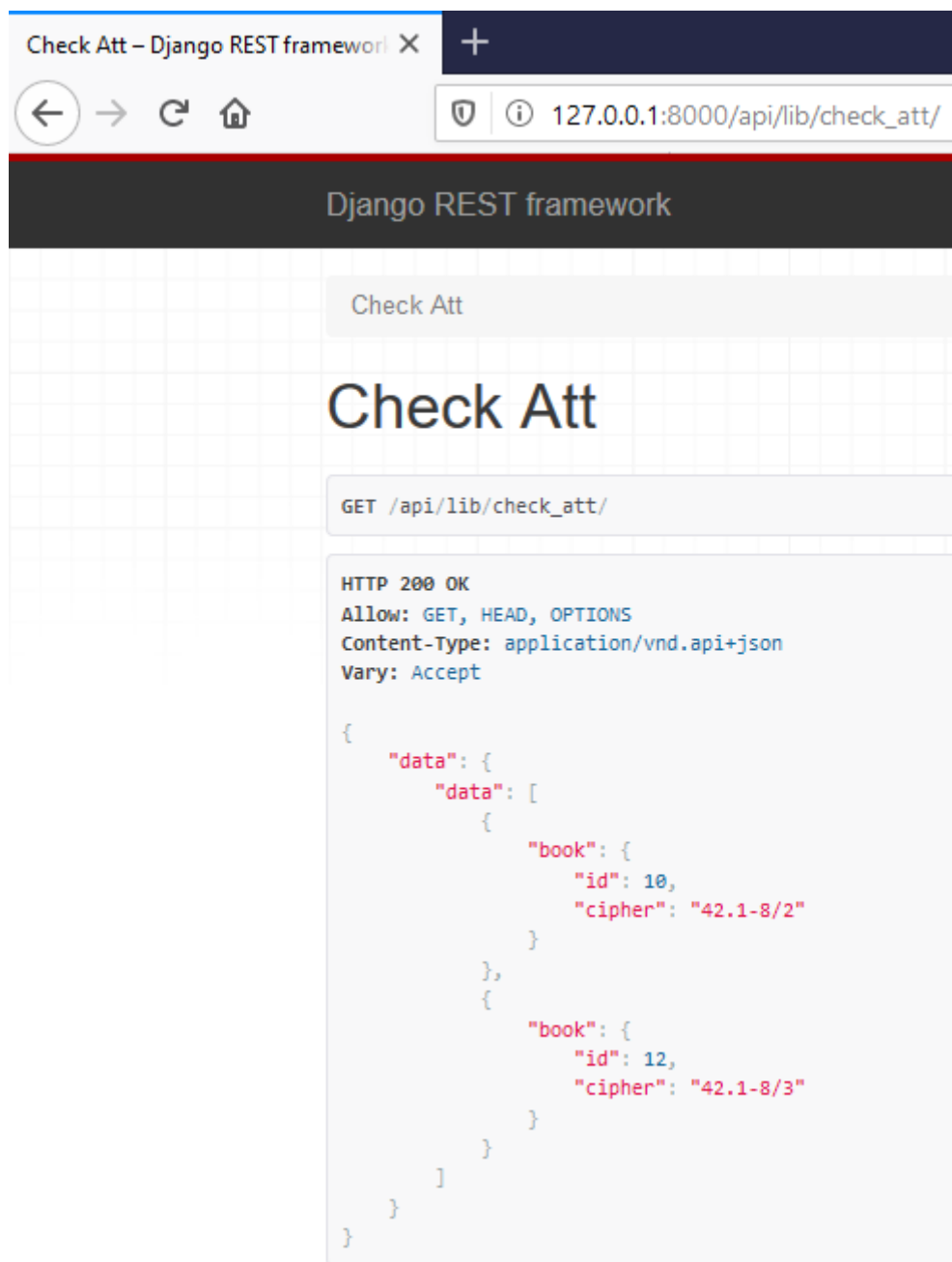


Рисунок 2 – Страница по адресу '127.0.0.1:8000/api/lib/check_att/'

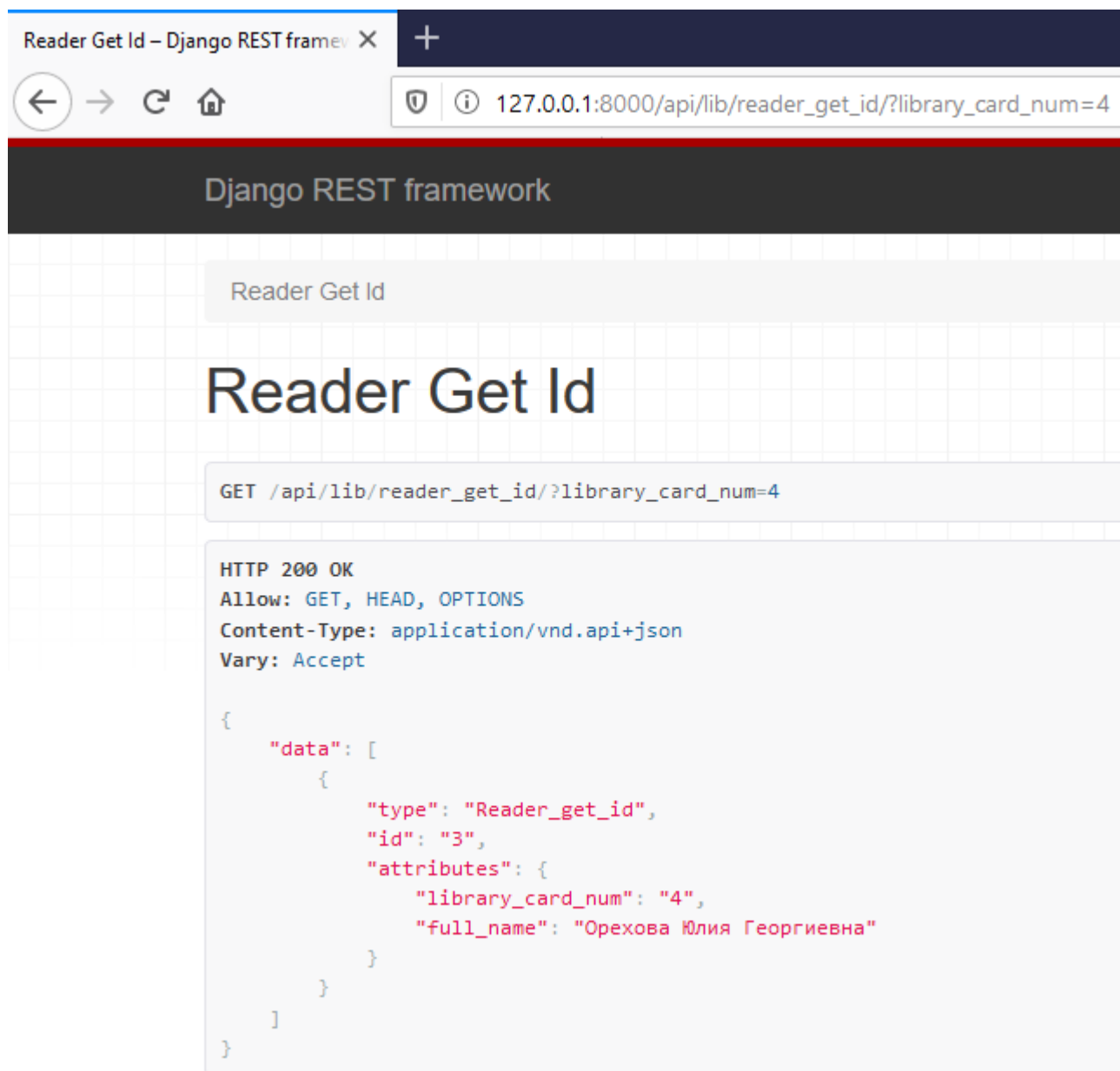


Рисунок 3 – Страница по адресу '127.0.0.1:8000/api/lib/reader_get_id/?library_card_num=4'

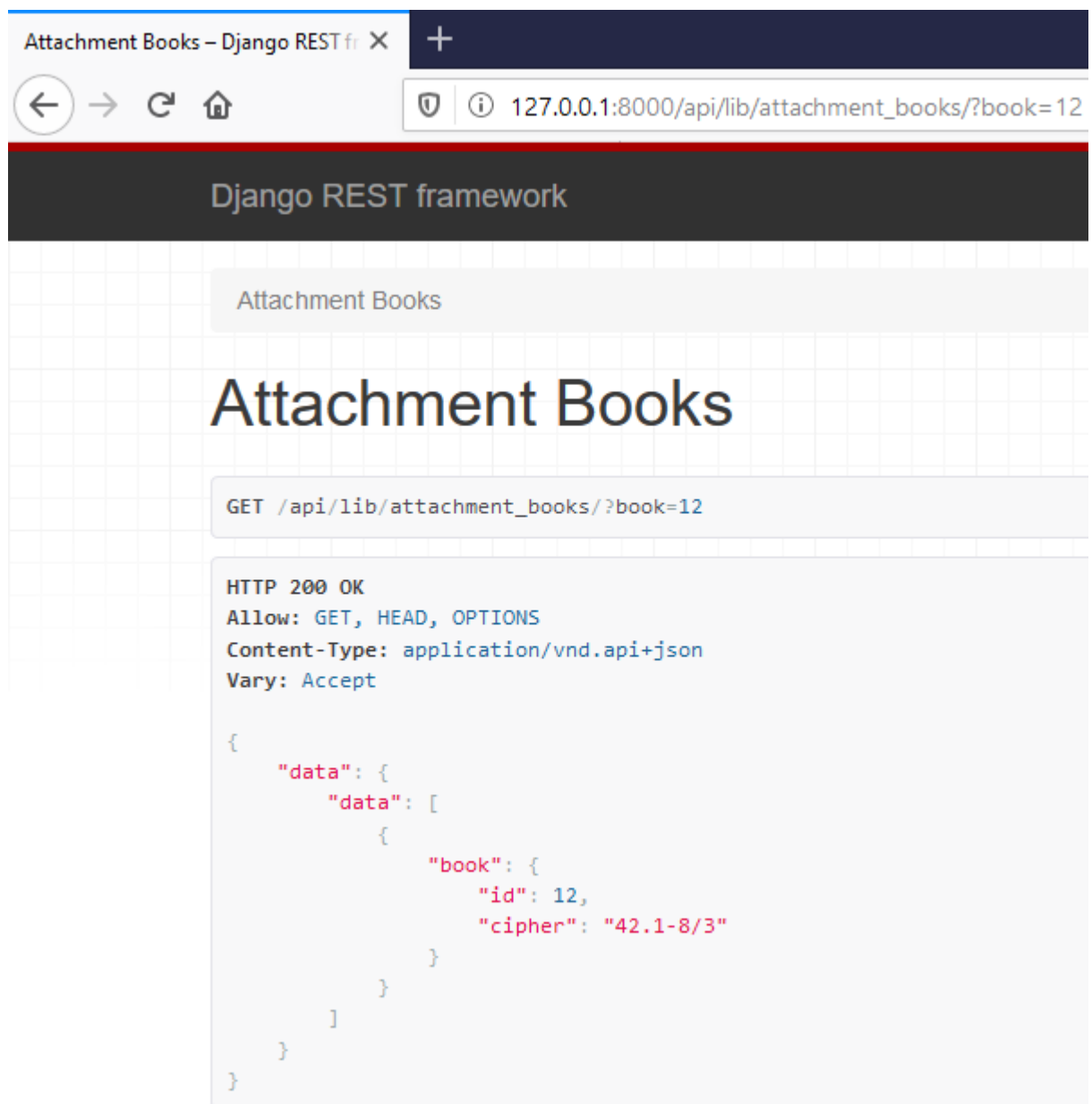


Рисунок 4 – Страница по адресу '127.0.0.1:8000/api/lib/attachment_books/?book=12'

Приложение Г. Код компонентов

```
93 | headers: { Authorization: 'Token ' + sessionStorage.getItem('auth_token') }
94 | })
95 | this.loadBooks()
96 | },
97 | methods: {
98 |   loadBooks () {
99 |     // eslint-disable-next-line
100 |     $.ajax({
101 |       url: 'http://127.0.0.1:8000/api/lib/reader/',
102 |       type: 'GET',
103 |       data: {
104 |         reader: this.id
105 |       },
106 |       success: (response) => {
107 |         this.person_books = response.data
108 |         let idList = []
109 |         for (let j = 0; j < this.person_books.books.length; j++) {
110 |           idList.push(this.person_books.books[j].book.id)
111 |         }
112 |         let maxNum = Math.max(...idList)
113 |         for (let i = 0; i < maxNum + 1; i++) {
114 |           this.detachement_form.push({attachment: '', date: ''})
115 |         }
116 |       }
117 |     })
118 |   },
119 |   addAtt () {
120 |     // eslint-disable-next-line
121 |     $.ajax({
122 |       url: 'http://127.0.0.1:8000/api/lib/book/',
```

Рисунок 1 – Код тэга <script ...> компонента Reader (часть 1)

```

6         book: this.book_form,
7         attachment_starting_date: this.form.date
8     },
9     success: (response) => {
10         alert('Закрепление добавлено')
11         this.form.input = ''
12         this.form.date = ''
13         this.loadBooks()
14     },
15     error: (response) => {
16         alert('Error')
17     }
18 })
19 } else {
20     alert('Книга с введённым шифром отсутствует в библиотеке')
21 }
22 },
23 error: (response) => {
24     alert('Error')
25 }
26 })
27 },
28 detach (idAtt, idForm) {
29     this.detachment_form[idForm].attachment = idAtt
30     let data = {
31         data: {
32             type: 'Attachment',

```

Рисунок 2 – Код тэга <script ...> компонента Reader (часть 2)

Приложение Д. Код компонента Books

```
<template>
  <mu-container>
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
      rel="stylesheet">
    <mu-appbar style="width: 100%;" color="#8B4513">
      <mu-menu slot="left">
        <mu-button flat icon>
          <mu-icon value="menu"></mu-icon>
        </mu-button>
        <mu-list slot="content">
          <mu-list-item button @click="goHome()">
            <mu-list-item-content>
              <mu-list-item-title>Читатели</mu-list-item-title>
            </mu-list-item-content>
          </mu-list-item>
          <mu-list-item button>
            <mu-list-item-content>
              <mu-list-item-title>Книги</mu-list-item-title>
            </mu-list-item-content>
          </mu-list-item>
        </mu-list>
      </mu-menu>
      Сайт библиотеки на Vue.js
      <mu-button @click="logout()" flat slot="right">Выход</mu-button>
    </mu-appbar>
    <mu-tabs :value.sync="active" color="#C08E67">
      <mu-tab>Список книг</mu-tab>
      <mu-tab>Поиск и изменение книг</mu-tab>
      <mu-tab>Добавить новую книгу</mu-tab>
    </mu-tabs>
    <div class="demo-text" v-if="active === 0">
```

```

<mu-container>
  <mu-flex>
    <mu-form :model="micro_form" class="cipher-search-form"
      :label-position="labelPosition" label-width="490">
      <mu-form-item prop="unattached" label="Показывать только незакреплённые ни
за кем в данный момент книги">
        <mu-switch v-model="micro_form.unattached"></mu-switch>
      </mu-form-item>
    </mu-form>
  </mu-flex>
  <mu-flex>
    {{unattachedBooks}}
  </mu-flex>
  <mu-flex>Для внесения изменений нажмите на шифр книги</mu-flex>
  <mu-paper :z-depth="1">
    <mu-data-table border stripe :columns="columns" :sort.sync="sort"
      @sort-change="handleSortChange" :data="finalBooksList">
      <template slot-scope="scope">
        <td class="is-left"><strong>
          <div @click="goToBook(scope.row.cipher)" class="cipher-link">
            {{scope.row.cipher}}
          </div>
        </strong></td>
        <td class="is-left">{{scope.row.title}}</td>
        <td class="is-left">{{scope.row.author}}</td>
        <td class="is-left">{{scope.row.publisher}}</td>
        <td class="is-left">{{scope.row.edition_year}}</td>
        <td class="is-left">{{scope.row.sphere}}</td>
        <td class="is-left">{{scope.row.receipt_date}}</td>
        <td class="is-left">{{scope.row.hall}}</td>
      </template>
    </mu-data-table>
  </mu-paper>

```

```

</mu-container>
</div>
<div class="demo-text" v-if="active === 1">
  <mu-row>
    <mu-flex>
      <mu-form :model="micro_form" class="cipher-search-form"
        :label-position="labelPosition" label-width="490">
        <mu-form-item prop="unattached" label="Искать только среди незакреплённых
ни за кем в данный момент книг">
          <mu-switch v-model="micro_form.unattached"></mu-switch>
        </mu-form-item>
      </mu-form>
    </mu-flex>
  </mu-row>
  <mu-row>
    <mu-col span="8">
      <mu-flex style="margin-left: 11px">
        Для поиска с учётом изменения положения переключателя нажмите кнопку
        "Искать" ещё раз
      </mu-flex>
    </mu-col>
    <mu-col span="5" style="color: #ffffff;">
      {{unattachedBooks}}
    </mu-col>
  </mu-row>
  <mu-row>
    <mu-flex>
      <div class="near-form-text">Поиск по шифру:</div>
      <mu-form :model="search_form" class="cipher-search-form"
        :label-position="labelPosition" label-width="100">
        <mu-form-item prop="bookCipher" label="Шифр книги">
          <mu-text-field v-model="search_form.bookCipher"></mu-text-field>
        </mu-form-item>

```

```

</mu-form>
    <mu-button color="primary" @click="openByCipher()" small class="search-form-
button">

        Искать
    </mu-button>
</mu-flex>
<mu-flex>
    <div class="near-form-text">Поиск по названию:</div>
    <mu-form :model="search_form" class="title-search-form"
        :label-position="labelPosition" label-width="240">
        <mu-form-item prop="bookTitle" label="Начните вводить название книги">
            <mu-text-field v-model="search_form.bookTitle"></mu-text-field>
        </mu-form-item>
    </mu-form>
    <mu-button color="primary" @click="openByTitle()" small class="search-form-
button">

        Искать
    </mu-button>
</mu-flex>
</mu-row>
<div v-if="foundBooksShow">
    <div v-for="book in foundBooksList" v-bind:key="book.cipher"><mu-row>
        <mu-col span="1"><strong>{{book.cipher}}</strong></mu-col>
        <mu-col>
            <mu-flex class="title">Книга: {{book.title}}</mu-flex>
            <mu-flex>Автор: {{book.author}}</mu-flex>
            <mu-flex>Издатель: {{book.publisher}}</mu-flex>
            <mu-flex>Год издания: {{book.edition_year}}</mu-flex>
            <mu-flex>Сфера: {{book.sphere}}</mu-flex>
            <mu-flex>Дата поступления: {{book.receipt_date}}</mu-flex>
            <mu-flex>Зал: {{book.hall}}</mu-flex>
            <mu-flex>
                <mu-button color="indigo400" @click="changeBook(book)"

```

```

        small class="search-form-button">
        Изменить/Списать книгу
    </mu-button>
</mu-flex>
</mu-col>
<mu-col justify-content="end">
    <mu-flex>
        <mu-form :model="add_form[book.id]" class="add-form"
            :label-position="labelPosition" label-width="50">
            <mu-form-item prop="bookTitle" label="Загл:">
                <mu-select v-model="add_form[book.id].hall">
                    <mu-option v-for="option in h_options" :key="option[0]"
                        :label="option[1]" :value="option[0]"></mu-option>
                </mu-select>
            </mu-form-item>
        </mu-form>
    </mu-flex>
    <mu-flex>
        <mu-form :model="add_form[book.id]" class="add-form"
            :label-position="labelPosition" label-width="150">
            <mu-form-item prop="receiptDate" label="Дата поступления">
                <mu-text-field v-model="add_form[book.id].receipt_date"></mu-text-field>
            </mu-form-item>
        </mu-form>
    </mu-flex>
    <mu-flex>
        <mu-button color="success" @click="addCopy(book.id)" small class="copy-form-
button">

        Добавить экземпляр книги
    </mu-button>
</mu-flex>
</mu-col>
<div v-if="micro_form.unattached">

```

```

    <mu-row justify-content="start">
        <mu-flex class="near-att-form-text">Закрепить данную книгу за
читателем:</mu-flex>
    </mu-row>
    <mu-row>
        <mu-flex justify-content="start">
            <mu-form :model="add_form[book.id]" class="attachment-form" :label-
position="labelPosition" label-width="250">
                <mu-form-item prop="date" label="Дата закрепления" help-text="В формате:
год(4 цифры)-месяц-день">
                    <mu-text-field v-model="add_form[book.id].date"></mu-text-field>
                </mu-form-item>
                <mu-form-item prop="library_card_num" label="Номер читательского
билета">
                    <mu-text-field v-model="add_form[book.id].library_card_num"></mu-text-
field>
                </mu-form-item>
            </mu-form>
        </mu-flex>
    </mu-row>
    <mu-flex justify-content="start" class="attachment-button">
        <mu-button color="success" @click="addAtt(book.id)">
            Закрепить
        </mu-button>
    </mu-flex>
</div>
</mu-row></div>
</div>
<div v-if="notFound">
    По вашему запросу книг не найдено
</div>
</div>
<div class="demo-text" v-if="active === 2">

```

```

    <mu-form :model="form" class="book-form" :label-position="labelPosition" label-
width="180">
    <mu-form-item prop="cipher" label="Шифр">
        <mu-text-field v-model="form.cipher"></mu-text-field>
    </mu-form-item>
    <mu-form-item prop="title" label="Название">
        <mu-text-field multi-line :rows="3" :rows-max="4" v-model="form.title"></mu-
text-field>
    </mu-form-item>
    <mu-form-item prop="author" label="Автор">
        <mu-text-field v-model="form.author"></mu-text-field>
    </mu-form-item>
    <mu-form-item prop="publisher" label="Издатель">
        <mu-text-field v-model="form.publisher"></mu-text-field>
    </mu-form-item>
    <mu-form-item prop="edition_year" label="Год издания">
        <mu-text-field v-model="form.edition_year"></mu-text-field>
    </mu-form-item>
    <mu-form-item prop="sphere" label="Раздел">
        <mu-text-field v-model="form.sphere"></mu-text-field>
    </mu-form-item>
    <mu-form-item prop="receipt_date" label="Дата поступления" help-text="В
формате: год(4 цифры)-месяц-день">
        <mu-text-field v-model="form.receipt_date"></mu-text-field>
    </mu-form-item>
    <mu-form-item prop="hall" label="Зал">
        <mu-select v-model="form.hall">
            <mu-option v-for="option in h_options" :key="option[0]"
:label="option[1]" :value="option[0]"></mu-option>
        </mu-select>
    </mu-form-item>
</mu-form>
<mu-flex>

```

```
<mu-button color="success" @click="addBook()">
  Добавить книгу
</mu-button>
</mu-flex>
</div>
</mu-container>
</template>
```

```
<script>
export default {
  name: 'Books',
  data () {
    return {
      sort: {
        name: "",
        order: 'asc'
      },
      columns: [
        {
          title: 'Шифр',
          width: 100,
          name: 'cipher',
          sortable: true
        },
        {
          title: 'Название книги',
          width: 300,
          // align: 'left',
          name: 'title'
        },
        {
          title: 'Автор',
          // width: 120,
```



```
// align: 'left',
name: 'author'
},
{
  title: 'Издатель',
  // width: 120,
  // align: 'left'
  name: 'publisher'
},
{
  title: 'Год издания',
  // width: 100,
  // align: 'left'
  name: 'edition_year'
},
{title: 'Раздел',
  width: 100,
  // align: 'left'
  name: 'sphere'
},
{title: 'Дата поступления',
  // width: 120,
  // align: 'left'
  name: 'receipt_date'
},
{title: 'Зал',
  width: 70,
  // align: 'left'
  name: 'hall'
}
],
bookList: [],
i: "
```

```
j: ",
ii: ",
active: 0,
form: {
  cipher: ",
  title: ",
  author: ",
  publisher: ",
  edition_year: ",
  sphere: ",
  receipt_date: ",
  hall: "
},
labelPosition: 'left',
h_options: [
  [2, 'Зал №2, Главный'],
  [3, 'Зал №3, Малый'],
  [4, 'Зал №4, Новый']
],
search_form: {
  bookCipher: ",
  bookTitle: "
},
rawBooksList: ",
foundBooksList: [],
foundBooksShow: false,
notFound: false,
micro_form: {
  unattached: false
},
unattachedBooksList: [],
finalBooksList: [],
add_form: [],
```

```

    goalBook: "
  }
},
created () {
  // eslint-disable-next-line
  $.ajaxSetup({
    headers: {'Authorization': 'Token ' + sessionStorage.getItem('auth_token')}
  })
  this.loadBooks()
  this.checkAttachments()
  this.finalBooksList = this.bookList
},
computed: {
  unattachedBooks: function () {
    if (this.micro_form.unattached) {
      // eslint-disable-next-line
      this.finalBooksList = this.unattachedBooksList
      return 'Показаны только не закреплённые ни за кем в данный момент книги'
    } else {
      // eslint-disable-next-line
      this.finalBooksList = this.bookList
      return 'Показаны все книги'
    }
  }
},
methods: {
  logout () {
    sessionStorage.removeItem('auth_token')
    this.$router.push({'name': 'home'})
  },
  goHome () {
    this.$router.push({'name': 'home'})
  },

```

```

handleSortChange ({name, order}) {
  this.finalBooksList = this.finalBooksList.sort(function (a, b) {
    if (order === 'asc') {
      return parseFloat(a[name]) + 0.0001 * parseInt(a[name].split('/')[1]) -
parseFloat(b[name]) - 0.0001 * parseInt(b[name].split('/')[1])
    } else {
      return parseFloat(b[name]) + 0.0001 * parseInt(b[name].split('/')[1]) -
parseFloat(a[name]) - 0.0001 * parseInt(a[name].split('/')[1])
    }
  })
},
loadBooks () {
  // eslint-disable-next-line
$.ajax({
  url: 'http://127.0.0.1:8000/api/lib/books/',
  type: 'GET',
  success: (response) => {
    this.bookList = response.data
    this.i = 0
    let idList = []
    for (let p = 0; p < this.bookList.length; p++) {
      idList.push(this.bookList[p].id)
    }
    let maxNum = Math.max(...idList)
    for (let m = 0; m < maxNum + 1; m++) {
      this.add_form.push({
        id: m,
        hall: "",
        receipt_date: "",
        library_card_num: "",
        date: ""
      })
    }
  }
}

```

```

this.bookList.forEach(function () {
  let newVal = {}
  for (let key in this.bookList[this.i].attributes) {
    newVal[key] = this.bookList[this.i].attributes[key]
  }
  newVal['hall'] = this.bookList[this.i].relationships.hall.data.id
  newVal['id'] = this.bookList[this.i].id
  this.bookList[this.i] = newVal
  this.add_form[newVal.id].receipt_date = newVal.receipt_date
  this.i++
}).bind(this))
})
},
addBook () {
  // eslint-disable-next-line
$.ajax({
  url: 'http://127.0.0.1:8000/api/lib/book_add/',
  type: 'POST',
  data: this.form,
  success: (response) => {
    alert('Новая книга успешно добавлена')
    for (let key in this.form) {
      this.form[key] = ""
    }
    this.loadBooks()
    this.checkAttachments()
    this.finalBooksList = this.bookList
    this.active = 0
  },
  error: (response) => {
    alert('Error')
  }
})

```

```

    })
  },
  openByCipher () {
    this.notFound = false
    if (this.foundBooksShow) {
      this.foundBooksShow = false
      this.foundBooksList = []
      this.search_form.bookTitle = "
    }
    this.j = 0
    let neededLength = this.search_form.bookCipher.length
    this.finalBooksList.forEach(function () {
      if (this.finalBooksList[this.j].cipher.substring(0, neededLength) ===
this.search_form.bookCipher) {
        this.foundBooksList.push(this.finalBooksList[this.j])
        this.j++
      } else {
        this.j++
      }
    }).bind(this))
    if (this.foundBooksList.length) {
      this.foundBooksShow = true
    } else {
      this.foundBooksShow = true
      this.notFound = true
    }
  },
  openByTitle () {
    this.notFound = false
    if (this.foundBooksShow) {
      this.foundBooksShow = false
      this.foundBooksList = []
      this.search_form.bookCipher = "

```

```

    }
    this.j = 0
    let neededLength = this.search_form.bookTitle.length
    this.finalBooksList.forEach(function () {
        if (this.finalBooksList[this.j].title.substring(0, neededLength) ===
this.search_form.bookTitle) {
            this.foundBooksList.push(this.finalBooksList[this.j])
            this.j++
        } else {
            this.j++
        }
    }).bind(this))
    if (this.foundBooksList.length) {
        this.foundBooksShow = true
    } else {
        this.foundBooksShow = true
        this.notFound = true
    }
},
checkAttachments () {
    // eslint-disable-next-line
    $.ajax({
        url: 'http://127.0.0.1:8000/api/lib/check_att',
        type: 'GET',
        success: (response) => {
            let attachedBooksList = response.data.data
            let k = 0
            attachedBooksList.forEach(function () {
                attachedBooksList[k] = attachedBooksList[k].book.cipher
                k++
            })
            let l = 0
            this.bookList.forEach(function () {

```

```

        if (attachedBooksList.indexOf(this.bookList[l].cipher) === -1) {
            this.unattachedBooksList.push(this.bookList[l])
        }
        l++
    }.bind(this))
}
})
},
changeBook (book) {
    this.$router.push({name: 'book_change', params: {book}})
},
addCopy (bookId) {
    // Получаю данные о книге, экземпляр который хочу добавить (по id)
    this.ii = 0
    this.bookList.forEach(function () {
        if (this.bookList[this.ii].id === bookId) {
            this.goalBook = this.bookList[this.ii]
        }
        this.ii++
    }.bind(this))
    // Ищу последний экземпляр данной книги
    let numList = []
    let numBookList = []
    for (let n = 0; n < this.bookList.length; n++) {
        if (this.bookList[n].title === this.goalBook.title) {
            let cipherTempList = this.bookList[n].cipher.split('/')
            let numTemp = parseInt(cipherTempList[1])
            numList.push(numTemp)
            numBookList.push(this.bookList[n])
        }
    }
    let maxNum = Math.max(...numList)
    let maxNumId = numList.indexOf(maxNum)

```



```
this.goalBook = numBookList[maxNumId]
maxNum += 1
// Формирую шифр
let cipherList = this.goalBook.cipher.split('/')
let newCipher = cipherList[0] + '/' + maxNum
// Формирую данные для отправки POST-запросом
let data = {
  cipher: newCipher,
  title: this.goalBook.title,
  author: this.goalBook.author,
  publisher: this.goalBook.publisher,
  edition_year: this.goalBook.edition_year,
  sphere: this.goalBook.sphere,
  receipt_date: this.add_form[bookId].receipt_date,
  hall: this.add_form[bookId].hall
}
// eslint-disable-next-line
$.ajax({
  url: 'http://127.0.0.1:8000/api/lib/book_add/',
  type: 'POST',
  data: data,
  success: (response) => {
    alert('Новая книга успешно добавлена')
    this.loadBooks()
    this.active = 0
    this.finalBooksList = this.bookList
    this.goalBook = "
    this.search_form = {
      bookCipher: "",
      bookTitle: "
    }
    this.foundBooksList = []
    this.foundBooksShow = false
```

```

    },
    error: (response) => {
        alert('Error')
    }
})
},
goToBook (bookCipher) {
    this.notFound = false
    if (this.foundBooksShow) {
        this.foundBooksShow = false
        this.foundBooksList = []
        this.search_form.bookTitle = "
    }
    this.j = 0
    this.search_form.bookCipher = bookCipher
    this.finalBooksList.forEach(function () {
        if (this.finalBooksList[this.j].cipher === this.search_form.bookCipher) {
            this.foundBooksList.push(this.finalBooksList[this.j])
            this.j++
        } else {
            this.j++
        }
    }).bind(this))
    this.active = 1
    this.foundBooksShow = true
},
addAtt (bookId) {
    // eslint-disable-next-line
    $.ajax({
        url: 'http://127.0.0.1:8000/api/lib/reader_get_id/',
        type: 'GET',
        data: {
            library_card_num: this.add_form[bookId].library_card_num

```

```

    },
    success: (response) => {
        if (response.data.length !== 0) {
            let data = {
                reader: response.data[0].id,
                book: bookId,
                attachment_starting_date: this.add_form[bookId].date
            }
            // eslint-disable-next-line
            $.ajax({
                url: 'http://127.0.0.1:8000/api/lib/reader/',
                type: 'POST',
                data: data,
                success: (response) => {
                    alert('Закрепление добавлено')
                    this.loadBooks()
                    this.unattachedBooksList = []
                    this.checkAttachments()
                    this.finalBooksList = this.bookList
                    this.active = 0
                },
                error: (response) => {
                    alert('Error')
                }
            })
        } else {
            alert('Книга с введённым шифром отсутствует в библиотеке')
        }
    },
    error: (response) => {
        alert('Error')
    }
})

```

```
    }  
  }  
}  
</script>
```

```
<style scoped>  
  .demo-text {  
    padding: 16px;  
    background: #ffffff;  
  }  
  .book-form {  
    width: 800px;  
    height: 560px;  
  }  
  .cipher-search-form {  
    width: 160px;  
    height: 50px;  
    margin-left: 10px;  
  }  
  .search-form-button {  
    margin-top: 8px;  
    margin-bottom: 10px;  
  }  
  .near-form-text {  
    margin-top: 9px;  
    margin-left: 10px;  
  }  
  .title-search-form {  
    width: 400px;  
    height: 50px;  
    margin-left: 10px;  
  }  
  .add-form {
```

```
width: 240px;
height: 50px;
margin-left: 10px;
}
.copy-form-button {
margin-left: 12px;
}
.cipher-link {
cursor: pointer
}
.attachment-form {
width: 500px;
height: 120px;
margin-right: 300px;
margin-left: 80px;
}
.attachment-button {
margin-left: 90px;
margin-bottom: 20px;
}
.near-att-form-text {
margin-top: 9px;
margin-left: 90px;
}
.title {
text-align: justify;
}
</style>
```

Приложение Е. Код компонента ReaderChange

```
<template>
  <mu-container>
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
      rel="stylesheet">
    <mu-appbar style="width: 100%;" color="#8B4513">
      <mu-menu slot="left">
        <mu-button flat icon>
          <mu-icon value="menu"></mu-icon>
        </mu-button>
        <mu-list slot="content">
          <mu-list-item button @click="goBack()">
            <mu-list-item-content>
              <mu-list-item-title>Читатели</mu-list-item-title>
            </mu-list-item-content>
          </mu-list-item>
          <mu-list-item button @click="goBooks()">
            <mu-list-item-content>
              <mu-list-item-title>Книги</mu-list-item-title>
            </mu-list-item-content>
          </mu-list-item>
        </mu-list>
      </mu-menu>
      Сайт библиотеки на Vue.js
      <mu-button @click="logout()" flat slot="right">Выход</mu-button>
    </mu-appbar>
  </mu-container>
  <mu-row>
    <mu-col span="10">
      <h3>Читатель {{person.full_name}}</h3>
      <h4>Нынешние данные:</h4>
    </mu-col>
```

```

</mu-col></mu-col>
<mu-col justify-content="end">
  <mu-button color="error" class="button-del" @click="personDel">
    Удалить читателя
  </mu-button>
</mu-col>
</mu-row>
<mu-row>
  <mu-col span="1"></mu-col>
  <mu-col justify-content="end">
    <mu-flex>Номер читательского билета: {{person.library_card_num}}</mu-flex>
    <mu-flex>Зал: {{person.hall}}</mu-flex>
    <mu-flex>Адрес: {{person.home_address}}</mu-flex>
    <mu-flex>Паспортные данные: {{person.passport_data}}</mu-flex>
  </mu-col>
  <mu-col justify-content="end">
    <mu-flex>Дата рождения: {{person.birth_date}}</mu-flex>
    <mu-flex>Контактный номер: {{person.phone_num}}</mu-flex>
    <mu-flex>Образование: {{person.education}}</mu-flex>
    <mu-flex v-if="person.degree">Имеется учёная степень</mu-flex>
    <mu-flex v-else>Отсутствует учёная степень</mu-flex>
  </mu-col>
</mu-row>
<mu-row>
  <mu-col span="10">
    <h4>Новые данные:</h4>
    <h5>(Заполняйте только те поля, значения которых собираетесь изменить)</h5>
  </mu-col>
  <mu-col></mu-col>
</mu-row>
<mu-row class="near-form">
  <mu-col span="1"></mu-col>
  <mu-col justify-content="start">

```

```

<mu-form :model="form" class="reader-form" :label-position="labelPosition" label-
width="300">
  <mu-form-item prop="library_card_num" label="Номер читательского билета">
    <mu-text-field v-model="form.library_card_num"></mu-text-field>
  </mu-form-item>
  <mu-form-item prop="full_name" label="Полное имя (ФИО)">
    <mu-text-field v-model="form.full_name"></mu-text-field>
  </mu-form-item>
  <mu-form-item prop="hall" label="Зал">
    <mu-select v-model="form.hall">
      <mu-option v-for="option in h_options" :key="option[0]"
        :label="option[1]" :value="option[0]"></mu-option>
    </mu-select>
  </mu-form-item>
  <mu-form-item prop="home_address" label="Адрес">
    <mu-text-field multi-line :rows="2" :rows-max="3" v-
model="form.home_address"></mu-text-field>
  </mu-form-item>
  <mu-form-item prop="passport_data" label="Паспортные данные">
    <mu-text-field v-model="form.passport_data"></mu-text-field>
  </mu-form-item>
  <mu-form-item prop="birth_date" label="Дата рождения" help-text="В формате:
год(4 цифры)-месяц-день">
    <mu-text-field v-model="form.birth_date"></mu-text-field>
  </mu-form-item>
  <mu-form-item prop="phone_num" label="Контактный номер">
    <mu-text-field v-model="form.phone_num"></mu-text-field>
  </mu-form-item>
  <mu-form-item prop="education" label="Образование">
    <mu-select v-model="form.education">
      <mu-option v-for="option in e_options" :key="option"
        :label="option" :value="option"></mu-option>
    </mu-select>

```



```

    </mu-form-item>
    <mu-form-item prop="degree" label="Наличие учёной степени">
      <mu-select v-model="form.degree">
        <mu-option v-for="option in d_options" :key="option"
          :label="option" :value="option"></mu-option>
      </mu-select>
    </mu-form-item>
  </mu-form>
</mu-col>
</mu-row>
<mu-flex><hr/></mu-flex>
<mu-row>
  <mu-col span="4"></mu-col>
  <mu-col justify-content="end">
    <mu-button color="success" @click="applyChanges()">
      Применить изменения
    </mu-button>
    <mu-button color="error" @click="goBack()">
      Назад
    </mu-button>
  </mu-col>
</mu-row>
<mu-flex><hr/></mu-flex>
</mu-container>
</mu-container>
</template>

```

```

<script>
export default {
  name: 'ReaderChange',
  props: ['person'],
  data () {
    return {

```

```
form: {
  id: "",
  full_name: "",
  library_card_num: "",
  hall: "",
  home_address: "",
  passport_data: "",
  birth_date: "",
  phone_num: "",
  education: "",
  degree: ""
},
labelPosition: 'left',
e_options: [
  'среднее общее', 'среднее профессиональное', 'неполное высшее',
  'бакалавр', 'специалист', 'магистр', 'аспирантура'
],
h_options: [[2, 'Зал №2, Главный'],
  [3, 'Зал №3, Малый'],
  [4, 'Зал №4, Новый']],
d_options: ['есть', 'отсутствует']
}
},
methods: {
  logout () {
    sessionStorage.removeItem('auth_token')
    this.$router.push({name: 'home'})
  },
  applyChanges () {
    let idPers = this.person.id
    let attr = {}
    for (let key in this.form) {
      if (this.form[key]) {
```

```

        attr[key] = this.form[key]
    }
}
if (attr.degree) {
    if (attr.degree === 'есть') {
        attr.degree = true
    } else {
        attr.degree = false
    }
}
let data = {
    data: {
        type: 'Reader',
        id: idPers,
        attributes: attr
    }
}
fetch(`http://127.0.0.1:8000/api/lib/reader_change/${this.person.id}/`,
{
    method: 'PUT',
    headers: {
        'Authorization': 'Token ' + sessionStorage.getItem('auth_token'),
        'Content-Type': 'application/vnd.api+json'
    },
    body: JSON.stringify(data)
}
).then(response => {
    alert('Данные читателя успешно изменены')
    this.goBack()
}))
},
goBack () {
    this.$router.push({'name': 'home'})
}

```

```

},
goBooks () {
  this.$router.push({'name': 'books'})
},
personDel () {
  // eslint-disable-next-line
  $.ajaxSetup({
    headers: {'Authorization': 'Token ' + sessionStorage.getItem('auth_token')}
  })
  // eslint-disable-next-line
  $.ajax({
    url: 'http://127.0.0.1:8000/api/lib/reader/',
    type: 'GET',
    data: {
      reader: this.person.id
    },
    success: (response) => {
      if (response.data.books.length === 0) {
        // eslint-disable-next-line
        $.ajax({
          url: 'http://127.0.0.1:8000/api/lib/reader_del/' + this.person.id + '/',
          type: 'DELETE',
          success: (response) => {
            alert('Читатель успешно удалён')
            this.goBack()
          }
        })
      } else {
        alert('Невозможно удалить читателя, он вернул не все взятые книги')
      }
    }
  })
}

```

```
}  
}  
</script>
```

```
<style scoped>  
  .reader-form {  
    width: 800px;  
    height: 50px;  
  }  
  .near-form {  
    height: 580px;  
  }  
  .button-del {  
    margin-top: 10px;  
    font: 10pt sans-serif;  
  }  
</style>
```

Приложение Ж. Код компонента BookChange

```
<template>
  <mu-container>
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
      rel="stylesheet">
    <mu-appbar style="width: 100%;" color="#8B4513">
      <mu-menu slot="left">
        <mu-button flat icon>
          <mu-icon value="menu"></mu-icon>
        </mu-button>
        <mu-list slot="content">
          <mu-list-item button @click="goHome()">
            <mu-list-item-content>
              <mu-list-item-title>Читатели</mu-list-item-title>
            </mu-list-item-content>
          </mu-list-item>
          <mu-list-item button @click="goBooks()">
            <mu-list-item-content>
              <mu-list-item-title>Книги</mu-list-item-title>
            </mu-list-item-content>
          </mu-list-item>
        </mu-list>
      </mu-menu>
      Сайт библиотеки на Vue.js
      <mu-button @click="logout()" flat slot="right">Выход</mu-button>
    </mu-appbar>
  </mu-container>
  <mu-row>
    <mu-col span="10" align-items="start">
      <h3>Книга {{book.cipher}}</h3>
      <h4>Нынешние данные:</h4>
    </mu-col>
```

```

</mu-col></mu-col>
<mu-col justify-content="end">
  <mu-button color="error" class="button-del" @click="bookDel()">
    Списать книгу
  </mu-button>
</mu-col>
</mu-row>
<mu-row>
  <mu-col>
    <mu-row>
      <strong><mu-col span="3" class="attr">Название:</mu-col></strong>
      <mu-col class="title">{{book.title}}</mu-col>
    </mu-row>
    <mu-row>
      <strong><mu-col span="3" class="attr">Автор:</mu-col></strong>
      <mu-col class="title">{{book.author}}</mu-col>
    </mu-row>
    <mu-row>
      <strong><mu-col span="3" class="attr">Издатель:</mu-col></strong>
      <mu-col class="title">{{book.publisher}}</mu-col>
    </mu-row>
    <mu-row>
      <strong><mu-col span="3" class="attr">Год&nbsp;издания:</mu-col></strong>
      <mu-col class="title">{{book.edition_year}}</mu-col>
    </mu-row>
    <mu-row>
      <strong><mu-col span="3" class="attr">Сфера:</mu-col></strong>
      <mu-col class="title">{{book.sphere}}</mu-col>
    </mu-row>
    <mu-row>
      <strong><mu-col span="3" class="attr">Дата&nbsp;поступления:</mu-col></strong>
      <mu-col class="title">{{book.receipt_date}}</mu-col>
    </mu-row>
  </mu-col>

```

```

</mu-row>
<mu-row>
  <strong><mu-col span="3" class="attr">Зал:</mu-col></strong>
  <mu-col class="title">{{book.hall}}</mu-col>
</mu-row>
</mu-col>
</mu-row>
<mu-row>
  <mu-col span="10">
    <h4>Новые данные:</h4>
    <h5>(Заполняйте только те поля, значения которых собираетесь изменить)</h5>
  </mu-col>
  <mu-col></mu-col>
</mu-row>
  <mu-form :model="change_form" class="book-change-form" :label-
position="labelPosition" label-width="180">
    <mu-form-item prop="cipher" label="Шифр">
      <mu-text-field v-model="change_form.cipher"></mu-text-field>
    </mu-form-item>
    <mu-form-item prop="title" label="Название">
      <mu-text-field multi-line :rows="3" :rows-max="4" v-model="change_form.title"></
mu-text-field>
    </mu-form-item>
    <mu-form-item prop="author" label="Автор">
      <mu-text-field v-model="change_form.author"></mu-text-field>
    </mu-form-item>
    <mu-form-item prop="publisher" label="Издатель">
      <mu-text-field v-model="change_form.publisher"></mu-text-field>
    </mu-form-item>
    <mu-form-item prop="edition_year" label="Год издания">
      <mu-text-field v-model="change_form.edition_year"></mu-text-field>
    </mu-form-item>
    <mu-form-item prop="sphere" label="Раздел">

```



```

    <mu-text-field v-model="change_form.sphere"></mu-text-field>
  </mu-form-item>
  <mu-form-item prop="receipt_date" label="Дата поступления" help-text="В формате:
год(4 цифры)-месяц-день">
    <mu-text-field v-model="change_form.receipt_date"></mu-text-field>
  </mu-form-item>
  <mu-form-item prop="hall" label="Зал">
    <mu-select v-model="change_form.hall">
      <mu-option v-for="option in h_options" :key="option[0]"
        :label="option[1]" :value="option[0]"></mu-option>
    </mu-select>
  </mu-form-item>
</mu-form>
<mu-flex><hr/></mu-flex>
<mu-row>
  <mu-col span="2"></mu-col>
  <mu-col justify-content="end">
    <mu-button color="success" @click="applyChanges()">
      Применить изменения
    </mu-button>
    <mu-button color="error" @click="goBooks()">
      Назад
    </mu-button>
  </mu-col>
</mu-row>
<mu-flex><hr/></mu-flex>
</mu-container>
</mu-container>
</template>

<script>
export default {
  name: 'BookChange',

```

```
props: ['book'],
data () {
  return {
    change_form: {
      cipher: "",
      title: "",
      author: "",
      publisher: "",
      edition_year: "",
      sphere: "",
      receipt_date: "",
      hall: ""
    },
    labelPosition: 'left',
    h_options: [[2, 'Зал №2, Главный'],
      [3, 'Зал №3, Малый'],
      [4, 'Зал №4, Новый']]
  }
},
created () {
  // eslint-disable-next-line
  $.ajaxSetup({
    headers: {'Authorization': 'Token ' + sessionStorage.getItem('auth_token')}
  })
},
methods: {
  logout () {
    sessionStorage.removeItem('auth_token')
    this.$router.push({name: 'home'})
  },
  goHome () {
    this.$router.push({name: 'home'})
  },
}
```

```

goBooks () {
  this.$router.push({name: 'books'})
},
bookDel () {
  // eslint-disable-next-line
  $.ajax({
    url: 'http://127.0.0.1:8000/api/lib/attachment_books/',
    type: 'GET',
    data: {
      book: this.book.id
    },
    success: (response) => {
      if (response.data.data.length === 0) {
        // eslint-disable-next-line
        $.ajax({
          url: 'http://127.0.0.1:8000/api/lib/book_del/' + this.book.id + '/',
          type: 'DELETE',
          success: (response) => {
            alert('Книга успешно списана')
            this.goBooks()
          }
        })
      } else {
        alert('Невозможно списать книгу, так как она закреплена за читателем')
      }
    }
  })
},
applyChanges () {
  let idBook = this.book.id
  let attr = {}
  for (let key in this.change_form) {
    if (this.change_form[key]) {

```

```

        attr[key] = this.change_form[key]
    }
}
let data = {
    data: {
        type: 'Book',
        id: idBook,
        attributes: attr
    }
}
fetch(`http://127.0.0.1:8000/api/lib/book_change/${this.book.id}/`,
{
    method: 'PUT',
    headers: {
        'Authorization': 'Token ' + sessionStorage.getItem('auth_token'),
        'Content-Type': 'application/vnd.api+json'
    },
    body: JSON.stringify(data)
}
).then(response => {
    alert('Данные книги успешно изменены')
    this.goBooks()
})
}
}
}
</script>

```

```

<style scoped>
.book-change-form {
    width: 800px;
    height: 560px;
    text-align: justify;

```

```
}  
.button-del {  
  margin-top: 10px;  
  font: 10pt sans-serif;  
}  
.title {  
  text-align: justify;  
}  
.attr {  
  text-align: justify;  
  width: 180px;  
}  
</style>
```

Приложение 3. Код компонента ReaderForm

```
<template>
  <mu-container>
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
      rel="stylesheet">
    <mu-appbar style="width: 100%;" color="#8B4513">
      <mu-menu slot="left">
        <mu-button flat icon>
          <mu-icon value="menu"></mu-icon>
        </mu-button>
        <mu-list slot="content">
          <mu-list-item button @click="goBack()">
            <mu-list-item-content>
              <mu-list-item-title>Читатели</mu-list-item-title>
            </mu-list-item-content>
          </mu-list-item>
          <mu-list-item button @click="goBooks()">
            <mu-list-item-content>
              <mu-list-item-title>Книги</mu-list-item-title>
            </mu-list-item-content>
          </mu-list-item>
        </mu-list>
      </mu-menu>
      Сайт библиотеки на Vue.js
      <mu-button @click="logout" flat slot="right">Выход</mu-button>
    </mu-appbar>
  </mu-container>
  <mu-row>
    <mu-col span="1"></mu-col>
    <mu-col span="10">
      <h3>Новый читатель</h3>
    </mu-col>
```

```

    <mu-col></mu-col>
  </mu-row>
  <mu-row class="near-form">
    <mu-col span="1"></mu-col>
    <mu-col justify-content="start">
      <mu-form :model="form" class="reader-form" :label-position="labelPosition" label-
width="300">
        <mu-form-item prop="library_card_num" label="Номер читательского билета">
          <mu-text-field v-model="form.library_card_num"></mu-text-field>
        </mu-form-item>
        <mu-form-item prop="full_name" label="Полное имя (ФИО)">
          <mu-text-field v-model="form.full_name"></mu-text-field>
        </mu-form-item>
        <mu-form-item prop="hall" label="Зал">
          <mu-select v-model="form.hall">
            <mu-option v-for="option in h_options" :key="option[0]"
              :label="option[1]" :value="option[0]"></mu-option>
          </mu-select>
        </mu-form-item>
        <mu-form-item prop="home_address" label="Адрес">
          <mu-text-field multi-line :rows="2" :rows-max="3" v-
model="form.home_address"></mu-text-field>
        </mu-form-item>
        <mu-form-item prop="passport_data" label="Паспортные данные">
          <mu-text-field v-model="form.passport_data"></mu-text-field>
        </mu-form-item>
        <mu-form-item prop="birth_date" label="Дата рождения" help-text="В формате:
год(4 цифры)-месяц-день">
          <mu-text-field v-model="form.birth_date"></mu-text-field>
        </mu-form-item>
        <mu-form-item prop="phone_num" label="Контактный номер">
          <mu-text-field v-model="form.phone_num"></mu-text-field>
        </mu-form-item>
      </mu-form>
    </mu-col>
  </mu-row>

```

```

    <mu-form-item prop="education" label="Образование">
      <mu-select v-model="form.education">
        <mu-option v-for="option in e_options" :key="option"
          :label="option" :value="option"></mu-option>
      </mu-select>
    </mu-form-item>
    <mu-form-item prop="degree" label="Наличие учёной степени">
      <mu-select v-model="form.degree">
        <mu-option v-for="option in d_options" :key="option"
          :label="option" :value="option"></mu-option>
      </mu-select>
    </mu-form-item>
  </mu-form>
</mu-col>
</mu-row>
<mu-flex><hr/></mu-flex>
<mu-row>
  <mu-col span="4"></mu-col>
  <mu-col justify-content="end">
    <mu-button color="success" @click="addReader()">
      Добавить читателя
    </mu-button>
    <mu-button color="error" @click="goBack()">
      Назад
    </mu-button>
  </mu-col>
</mu-row>
<mu-flex><hr/></mu-flex>
</mu-container>
</mu-container>
</template>

<script>

```



```
export default {
  name: 'ReaderCh',
  props: ['person'],
  data () {
    return {
      form: {
        ful_name: "",
        library_card_num: "",
        hall: "",
        home_address: "",
        passport_data: "",
        birth_date: "",
        phone_num: "",
        education: "",
        degree: ""
      },
      labelPosition: 'left',
      e_options: [
        'среднее общее', 'среднее профессиональное', 'неполное высшее',
        'бакалавр', 'специалист', 'магистр', 'аспирантура'
      ],
      h_options: [
        [2, 'Зал №2, Главный'],
        [3, 'Зал №3, Малый'],
        [4, 'Зал №4, Новый']
      ],
      d_options: ['есть', 'отсутствует']
    }
  },
  methods: {
    logout () {
      sessionStorage.removeItem('auth_token')
      this.$router.push({'name': 'home'})
    }
  }
}
```

```

    },
    addReader () {
      if (this.form.degree === 'есть') {
        this.form.degree = true
      } else {
        this.form.degree = false
      }
      // eslint-disable-next-line
      $.ajax({
        url: 'http://127.0.0.1:8000/api/lib/reader_add/',
        type: 'POST',
        data: this.form,
        success: (response) => {
          alert('Новый читатель успешно добавлен')
          this.$router.push({'name': 'home'})
        },
        error: (response) => {
          alert('Error')
        }
      })
    },
    goBack () {
      this.$router.push({'name': 'home'})
    },
    goBooks () {
      this.$router.push({'name': 'books'})
    }
  }
}
</script>

```

```

<style scoped>
.reader-form {

```

```
width: 800px;
height: 50px;
}
.near-form {
height: 580px;
}
</style>
```