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

Лабораторная работа №2-3

по основам web-программирования

на тему «РЕАЛИЗАЦИЯ WEB-СЕРВИСОВ СРЕДСТВАМИ Django REST framework, Vue.js, Muse-UI»

Выполнила: Бакирова Ш. Д.

Группа: К3340

Преподаватель: Говоров А. И.

Санкт-Петербург

2020 г.

Цель: овладеть практическими навыками и умениями реализации web-сервисов средствами Django REST framework, Vue.js, Muse-UI.

Программное обеспечение: Python 3.6, Django REST framework, Vue.js, Muse-UI (или аналогичная библиотека), PostgreSQL *.

Практическое задание:

Реализовать сайт используя вышеуказанные технолгии, в соответствии с практическим заданием.

Вариант 2.

Создать программную систему, предназначенную для работников библиотеки.

Такая система должна обеспечивать хранение сведений об имеющихся в библиотеке книгах, о читателях библиотеки и читальных залах.

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

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

Библиотекарю могут потребоваться следующие сведения о текущем состоянии библиотеки:

• Какие книги закреплены за определенным читателем?

• Кто из читателей взял книгу более месяца тому назад?

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

• Сколько в библиотеке читателей младше 20 лет?

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

Библиотекарь может выполнять следующие операции:

• Записать в библиотеку нового читателя.

• Исключить из списка читателей людей, записавшихся в библиотеку более

года назад и не прошедших перерегистрацию.

• Списать старую или потерянную книгу.

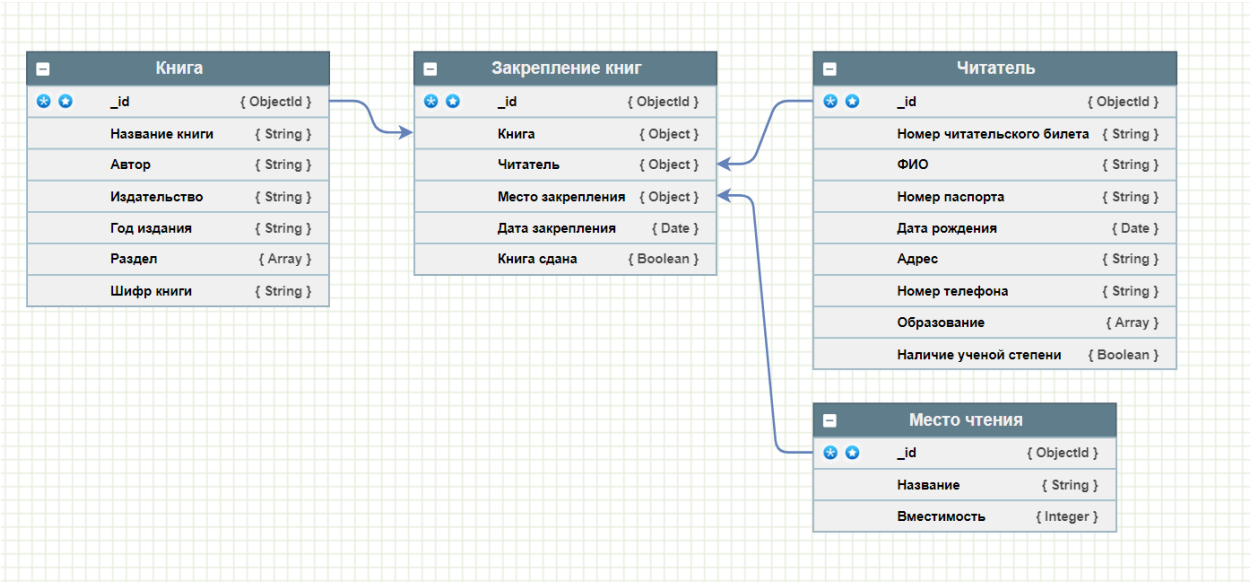
• Принять книгу в фонд библиотеки.

Необходимо предусмотреть возможность выдачи отчета о работе библиотеки в течение месяца. Отчет должен включать в себя следующую информацию: количество книг и читателей на каждый день в каждом из залов и в библиотеке в целом, количество читателей, записавшихся в библиотеку в каждый зал и в библиотеку за отчетный месяц.

Выполнение работы:

1. Модель базы данных

В соответствии с вариантом 2 была разработана модель БД:



Основные сущности и их атрибуты:

Книга: название книги, автор, издательство, год издания, раздел (художественная, учебная, психология, детские)

Читатель: номер читательского билета, ФИО, номер паспорта, дата рождения, адрес, номер телефона, образование (дошкольное, начальное общее, основное общее, среднее общее, неполное общее, высшее), наличие ученой степени

Место чтения: название, вместимость

Закрепление книг: книга (FK), читатель (FK), место закрепление (FK) , дата закрепления, книги сдана

2. Создание моделей в Django

Все модели хранятся в файле models.py

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

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

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

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

```python
class Reader(models.Model):
    education_type = (
        ('Дошкольное', 'Дошкольное'),
        ('Начальное общее', 'Начальное общее'),
        ('Основное общее', 'Основное общее'),
        ('Среднее общее', 'Среднее общее'),
        ('Неполное высшее', 'Неполное высшее'),
        ('Высшее', 'Высшее'))
    library_card = models.CharField('Номер читательского билета', max_length=16)
    name = models.CharField('ФИО', max_length=200)
    passport_number = models.CharField('Номер паспорта', max_length=12)
    date_of_birth = models.DateField('Дата рождения')
    address = models.CharField('Адрес', max_length=200)
    phone = models.CharField('Номер телефона', max_length=13)
    education = models.CharField('Образование', choices=education_type, max_length=50)
    academic = models.BooleanField('Наличие ученой степени', default=False)

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

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

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

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

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

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

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

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

3. Создание админ панели

Содержимое файла admin.py:

```python
class ReaderAdmin(admin.ModelAdmin):
    list_display = ('name', 'library_card', 'education')


class BookAdmin(admin.ModelAdmin):
    list_display = ('id', 'name', 'author', 'section', 'cipher')


class PlaceAdmin(admin.ModelAdmin):
    list_display = ('id', 'name', 'capacity')


class FixAdmin(admin.ModelAdmin):
    list_display = ('reader', 'book', 'date_fix', 'handed', 'place')


admin.site.register(Book, BookAdmin)
admin.site.register(Reader, ReaderAdmin)
admin.site.register(Place, PlaceAdmin)
admin.site.register(Fix, FixAdmin)
```

4. Создание контролеров для обработки данных

Представления размещены в файле view.py. Они были созданы с помощью serialisers.ModelSerializer и для обновления/удаления данных viewsets.ModelViewSet.

Вывод и редактирование читателя:

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

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


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

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

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

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


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

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

Вывод и редактирование книг:

```python
class Books(APIView):
    """Книги"""

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

        return Response(serializers.data)


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

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

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

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


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

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

Вывод и редактирование места чтения:

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

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


class PlaceOne(APIView):
    """Отображение одного зала"""
    permission_classes = [permissions.IsAuthenticated, ]

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

```python
class PlaceUpd(viewsets.ModelViewSet):
    """Отображение обновления заркрепления (откреплние книги)"""
    queryset = Fix.objects.all()

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

Отображение закреплений и их редактирования:

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

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


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

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

```python
class FixUpdate(viewsets.ModelViewSet):
    """Отображение обновления заркрепления (откреплние книги)"""
    queryset = Fix.objects.all()

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

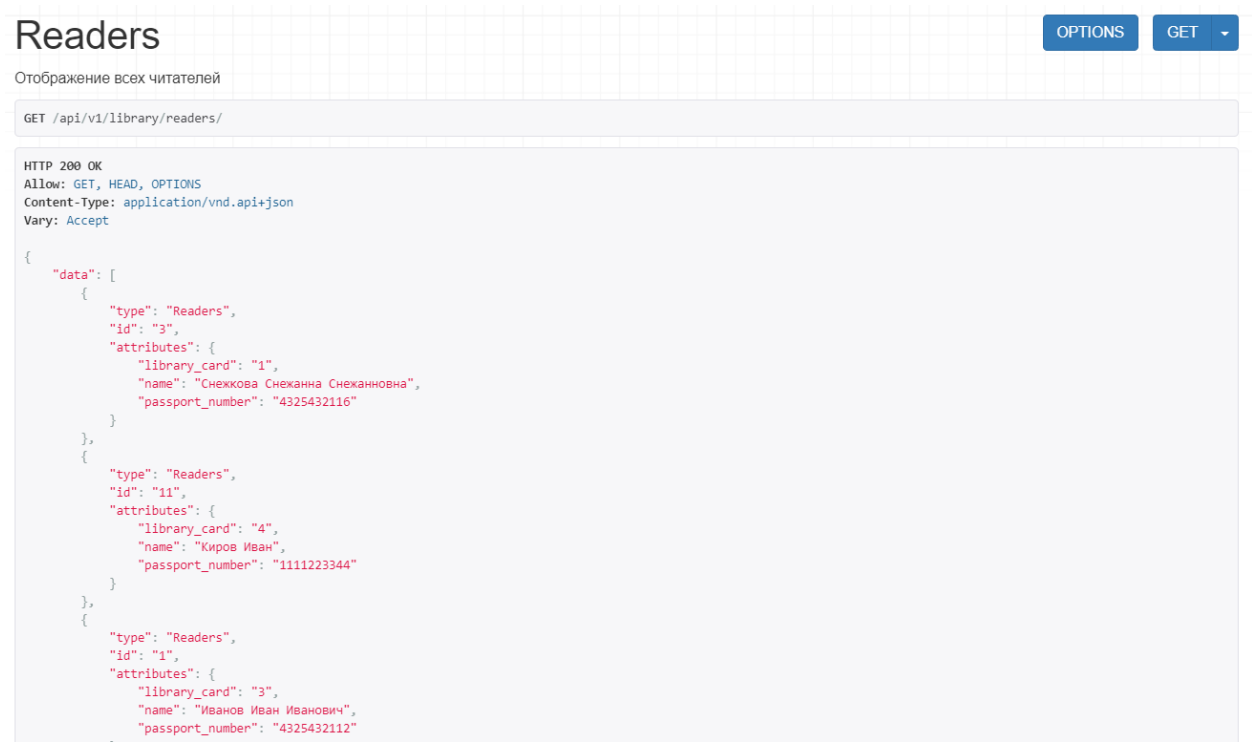
Также были добавлены отображения исходя из запросов по заданию

```python
class FixFilter(APIView):
    """Отображение фильтрации по дате за месяц"""
    def get(self, request):
        now = datetime.now() - timedelta(days=30)
        fixes = Fix.objects.filter(date_fix__lte=now)
        serializer = FixesSerializer(fixes, many=True)
        return Response({"data": serializer.data})


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

```python
class EducationFiler(APIView):
    """Сколько читателей в процентном отношении имеют начальное образование, среднее, высшее, ученую степень?

    def get(self, request):
        readers = Reader.objects.count()
        primary = Reader.objects.filter(education='Начальное общее').count()
        secondary = Reader.objects.filter(education='Среднее общее').count()
        high = Reader.objects.filter(education='Высшее').count()
        primary_percent = (primary/readers) * 100
        secondary_percent = (secondary/readers) * 100
        high_percent = (high/readers) * 100
        academic = Reader.objects.filter(academic='True').count()
        academic_percent = (academic/readers) * 100
        return Response({'primary': primary_percent, 'secondary': secondary_percent, 'high': high_percent,
                         'academic': academic_percent})
```

5. Полученные интерфейсы в Django Rest

Для модели Читатель:



Reader One

OPTIONS GET

Отображение одного читателя и все взятые им книги на данный момент

```
GET /api/v1/library/reader/?reader=3
```

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

{
    "data": {
        "reader": [
            {
                "id": 3,
                "library_card": "1",
                "name": "Снежкова Снеханна Снеханновна",
                "passport_number": "4325432116",
                "date_of_birth": "2006-10-01",
                "address": "Ленина 7",
                "phone": "89782313424",
                "education": "Начальное общее",
                "academic": false
            }
        ],
        "books": [
            {
                "id": 29,
                "book": {
                    "id": 2,
                    "name": "Дневник памяти",
                    "author": "Николас Спаркс",
                    "publisher": "Hachette Book Group",
                    "year": "2000",
                    "section": "Художественная",
                    "cipher": "2-1"
                },
```

Reader Del Upd

OPTIONS

Отображение для модели Взятие книги

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

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

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

Raw data | HTML form

Номер читательского билета

ФИО

Номер паспорта

Дата рождения дд.мм.гггг

Адрес

Номер телефона

Образование Дошкольное

Наличие ученой степени ☐

POST

Для модели Книга

Books

OPTIONS GET

Отображение всех книг

```
GET /api/v1/library/books/
```

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

{
    "data": [
        {
            "type": "Books",
            "id": "7",
            "attributes": {
                "cipher": "7-1",
                "name": "Цветы для Элджернона",
                "author": "Дэниел Киз"
            }
        },
        {
            "type": "Books",
            "id": "6",
            "attributes": {
                "cipher": "6-1",
                "name": "Куриный бульон для души",
                "author": "Эми Ньюмарк"
            }
        },
        {
            "type": "Books",
            "id": "3",
            "attributes": {
                "cipher": "3-1",
                "name": "Психология влияния",
                "author": "Чалдини Роберт Б."
            }
        },
        {
            "type": "Books"
```

Book One

OPTIONS GET

Отображение одной книги

```
GET /api/v1/library/book/?book=1-1
```

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

{
    "data": {
        "book": [
            {
                "id": 1,
                "name": "Крутой маршрут",
                "author": "Евгения Гинзбург",
                "publisher": "Harcourt",
                "year": "1967",
                "section": "Художественная",
                "cipher": "1-1"
            }
        ]
    }
}
```

Book Del Upd

OPTIONS

Отображение для модели Взятие книги

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

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

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

Raw data | HTML form

Название книги

Автор

Издательство

Год издания

Раздел: Художественная литература

Шифр книги

POST

Отображение для модели Место чтения

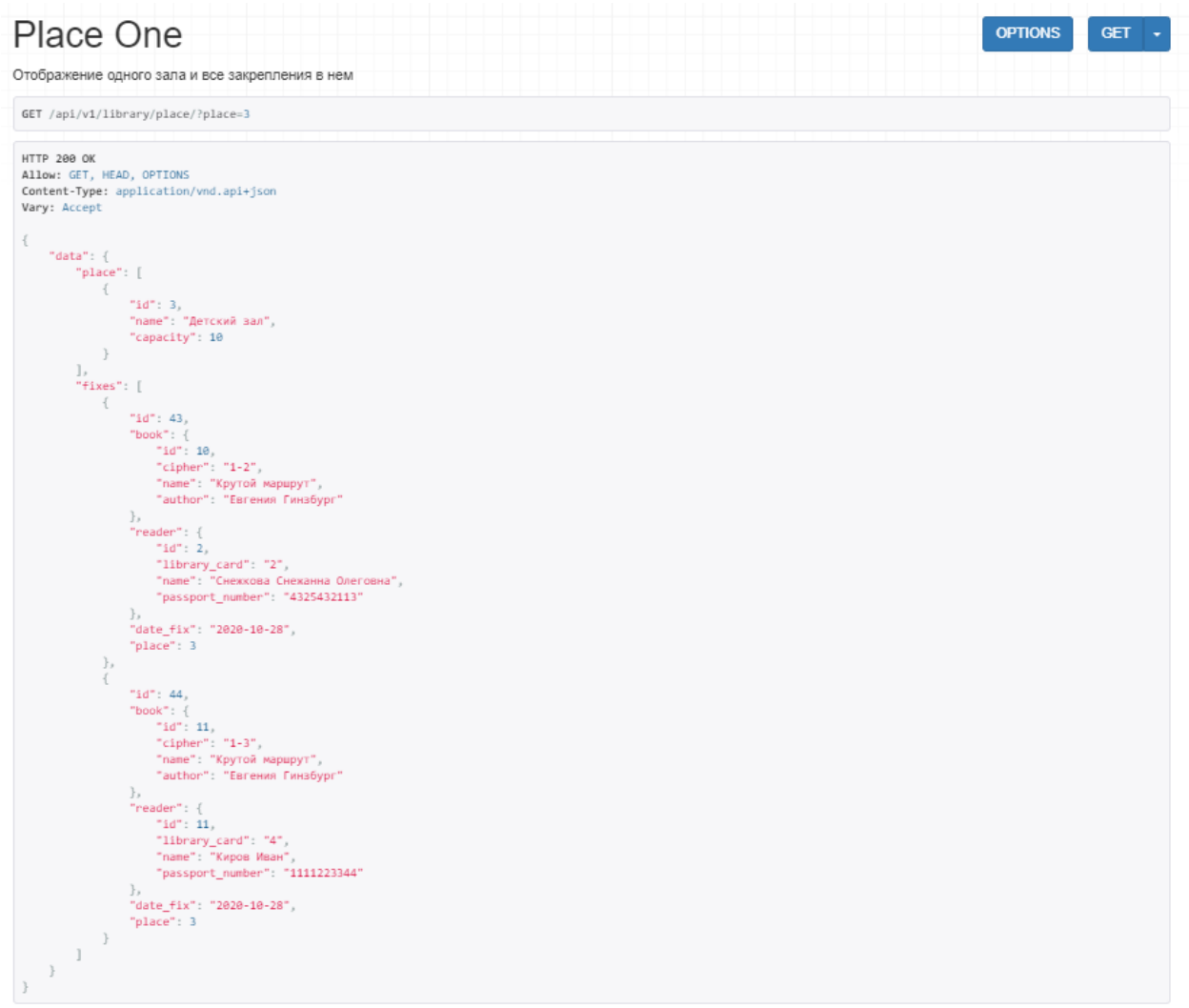
Places

OPTIONS | GET

Места

```
GET /api/v1/library/places/
```

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

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



Отображения для модели Закрепления

Fixes

OPTIONS GET

Закрепления

```
GET /api/v1/library/fix/
```


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

{
    "data": {
        "data": [
            {
                "id": 14,
                "book": {
                    "id": 1,
                    "cipher": "1-1",
                    "name": "Крутой маршрут",
                    "author": "Евгения Гинзбург"
                },
                "reader": {
                    "id": 2,
                    "library_card": "2",
                    "name": "Снежкова Снежанна Олеговна",
                    "passport_number": "4325432113"
                },
                "date_fix": "2020-09-09",
                "place": 2
            },
            {
                "id": 7,
                "book": {
                    "id": 6,
                    "cipher": "6-1",
                    "name": "Куриный бульон для души",
                    "author": "Эми Ньюмарк"
                },
                "reader": {
                    "id": 1,
                    "library_card": "3",
                    "name": "Иванов Иван Иванович",
                    "passport_number": "4325432112"
                },
                "date_fix": "2020-10-11",
                "place": 2
            },
            {
                "id": 42,
                "book": {
                    "id": 3,
                    "cipher": "3-1",
                    "name": "Психология влияния",
                    "author": "Чалдини Роберт Б."
                },
                "reader": {
                    "id": 1,
                    "library_card": "3",
                    "name": "Иванов Иван Иванович",
                    "passport_number": "4325432112"
                },
```


Fix Update

OPTIONS

Отображение обновления заркрепления (откреление книги)

```
GET /api/v1/library/fix_update/1/
```


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

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


Raw data | HTML form

Книга сдана ☐

PUT

Отображение для запросов:

Fix Filter

OPTIONS GET

Отображение фильтрации по дате за месяц

```
GET /api/v1/library/fix_filter/
```

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

{
    "data": {
        "data": [
            {
                "id": 14,
                "book": {
                    "id": 1,
                    "cipher": "1-1",
                    "name": "Крутой маршрут",
                    "author": "Евгения Гинзбург"
                },
                "reader": {
                    "id": 2,
                    "library_card": "2",
                    "name": "Снежкова Снежанна Олеговна",
                    "passport_number": "4325432113"
                },
                "date_fix": "2020-09-09",
                "place": 2
            }
        ]
    }
}
```

Readers Filter

OPTIONS GET

Отображение фильтрации по дате рождения (меньше 20 лет)

```
GET /api/v1/library/readers_filter/
```

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

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

Education Filer

OPTIONS GET

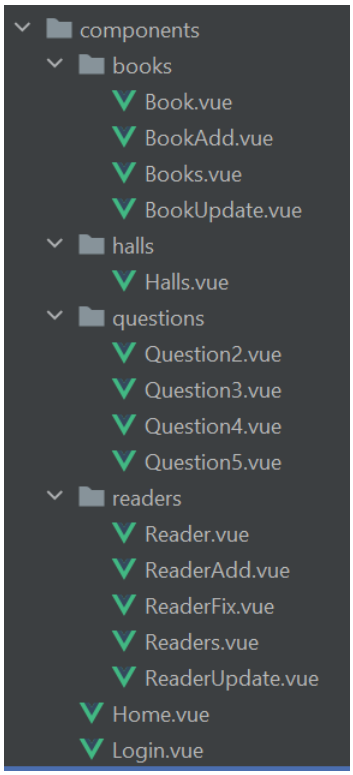
Сколько читателей в процентном отношении имеют начальное образование, среднее, высшее, ученую степень?

```
GET /api/v1/library/education/
```

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

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

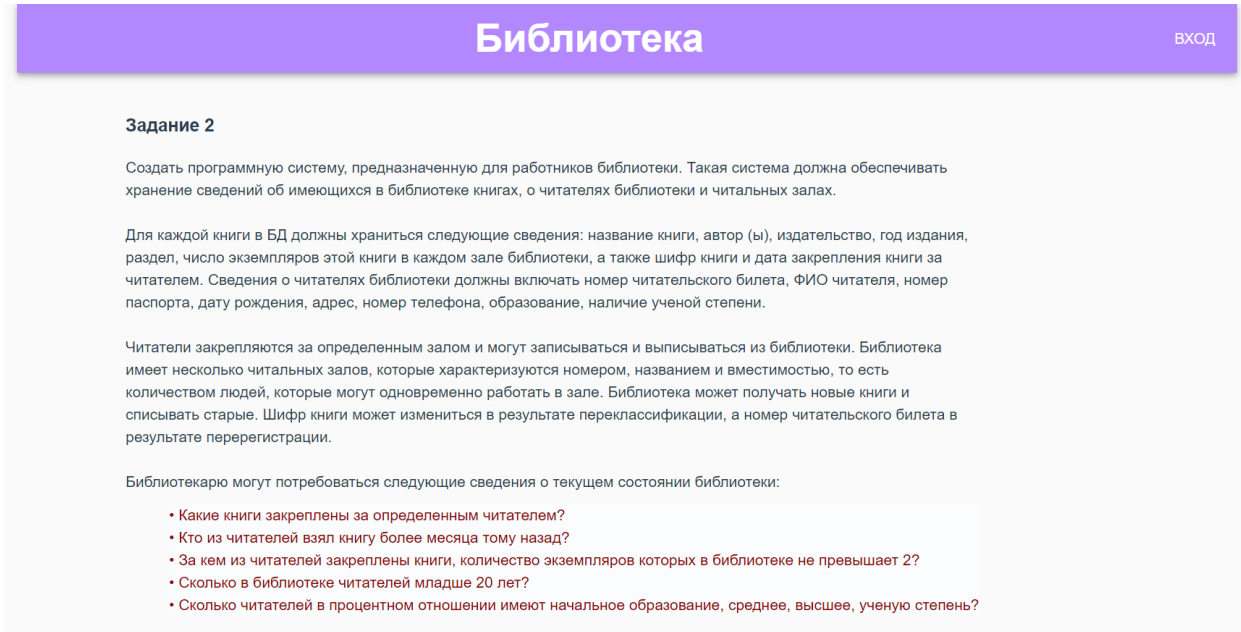
6. Внешний вид приложения на Vue.js

Для реализации внешнего вида приложения были созданы следующие компоненты:



Компонент Home – стартовая страница, содержащее задание варианта работы и вход для неавторизованного пользователя.

Также в самом задании имеются вопросы, ссылки которых введут на ответы в виде всплывающего диалога, либо на страницы, где запрос реализован.

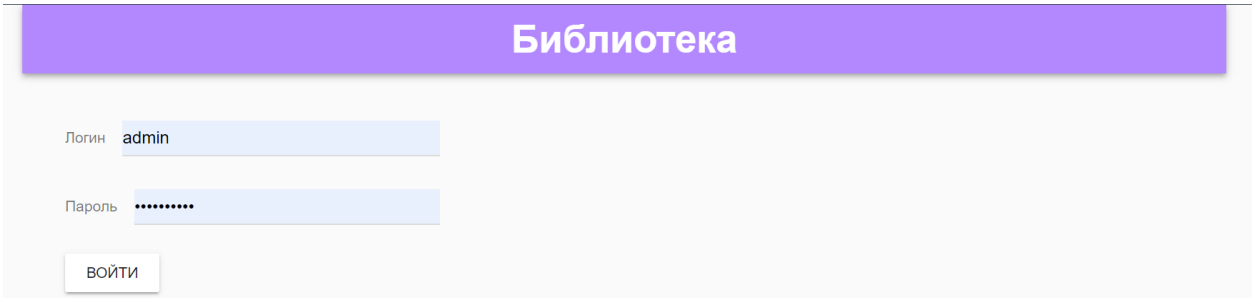


Библиотекарь может выполнять следующие операции:

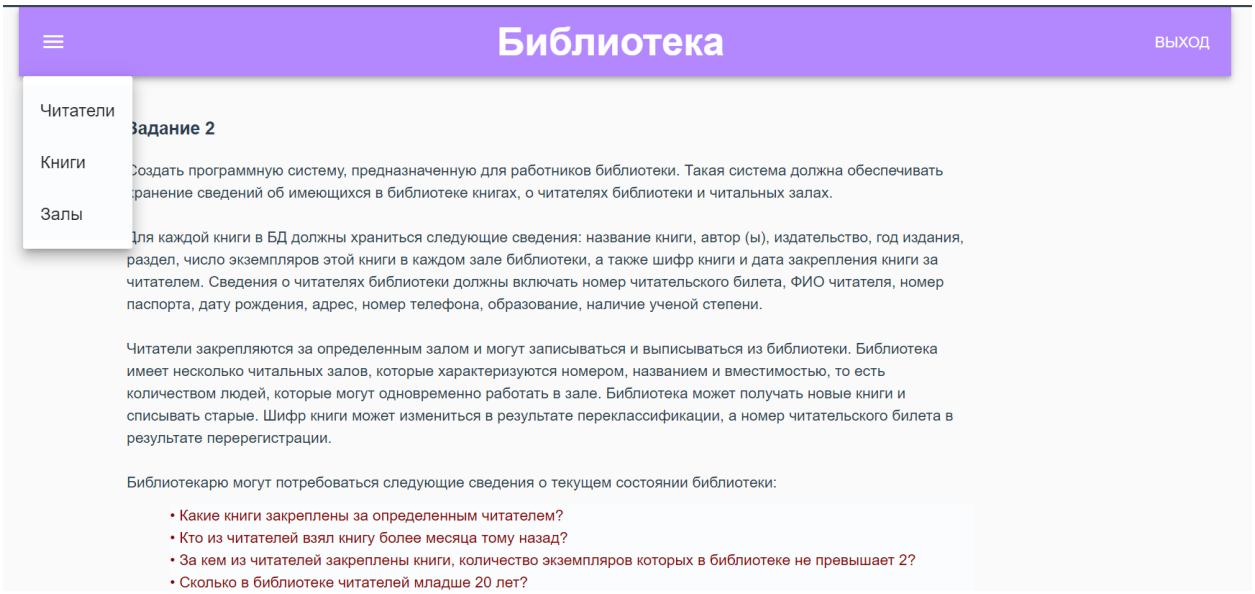
- Записать в библиотеку нового читателя.
- Исключить из списка читателей людей, записавшихся в библиотеку более года назад и не прошедших перерегистрацию. (Найти читателя можно как по читательскому билету, так и по паспортным данным)
- Списать старую или потерянную книгу.
- Принять книгу в фонд библиотеки.

Необходимо предусмотреть возможность выдачи отчета о работе библиотеки в течение месяца. Отчет должен включать в себя следующую информацию: количество книг и читателей на каждый день в каждом из залов и в библиотеке в целом, количество читателей, записавшихся в библиотеку в каждый зал и в библиотеку за отчетный месяц.

Компонент Login – страница с авторизацией пользователя

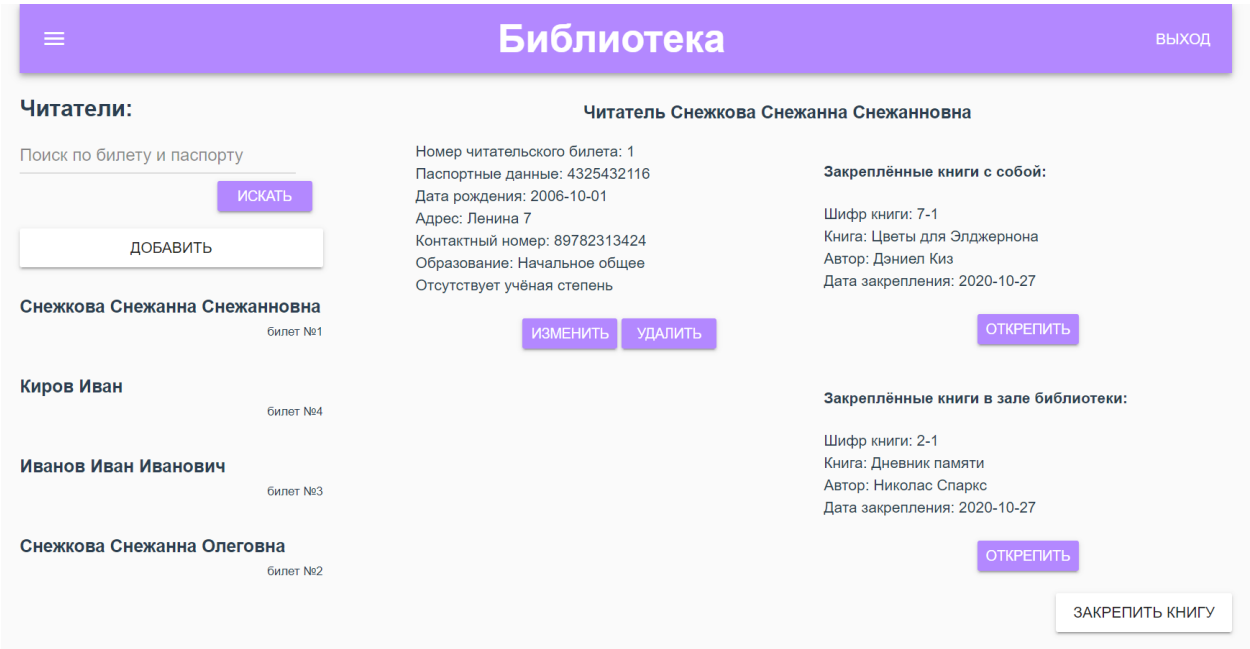


После авторизации стартовая страница видоизменяется и предоставляет отображение данных библиотеки

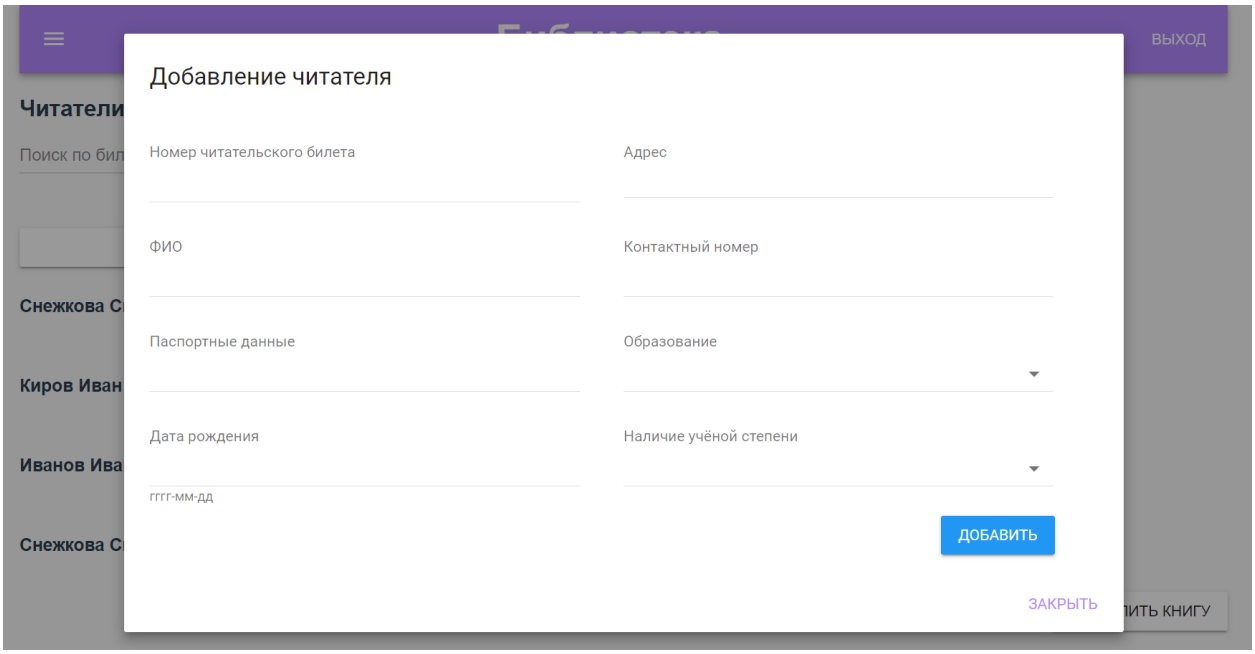


Компонент Readers – основной интерфейс для вывода всех читателей, зарегистрированных в библиотеке. Кликая по выбранному читателю или вводя его читательский билет/паспортные данные в поиск, открывается компонент Reader, где выводится полная информация о нем и закрепленные к нему книги. Также имеется возможность редактирования данных – кнопки при клике на которых открывается

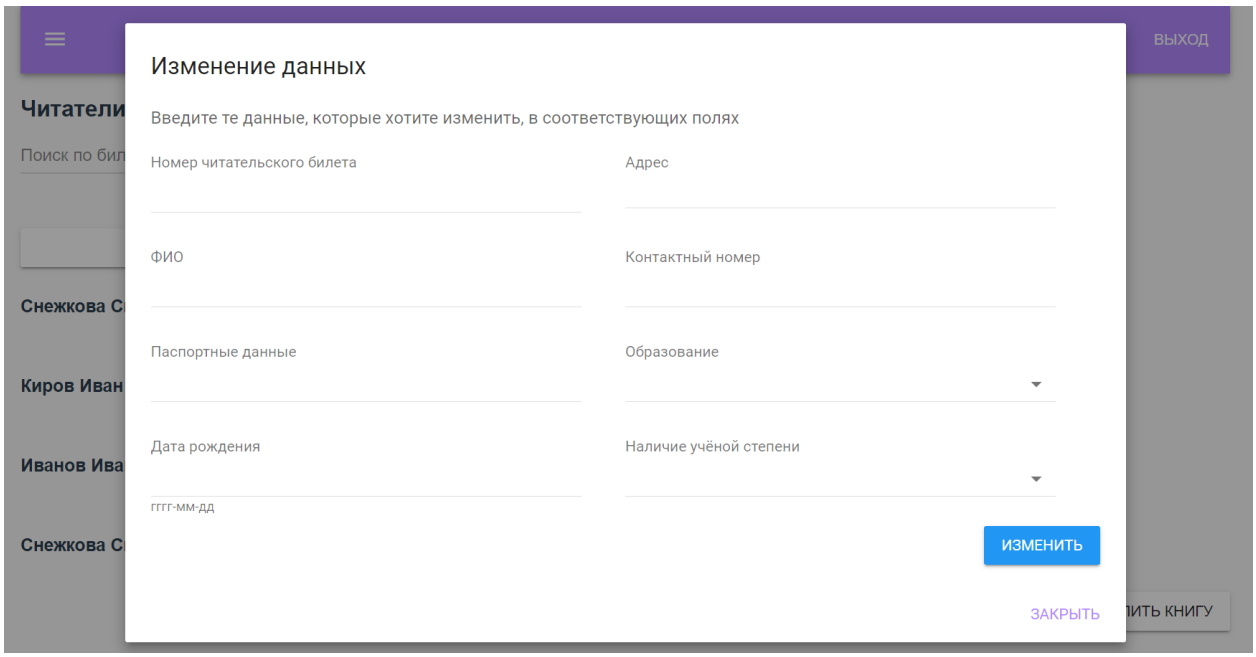
соответствующее окно или выполняется соответствующее действие. При нажатии на «открепить» в модели *Закрепление* значение атрибута *Книга сдана* из False меняется на True.



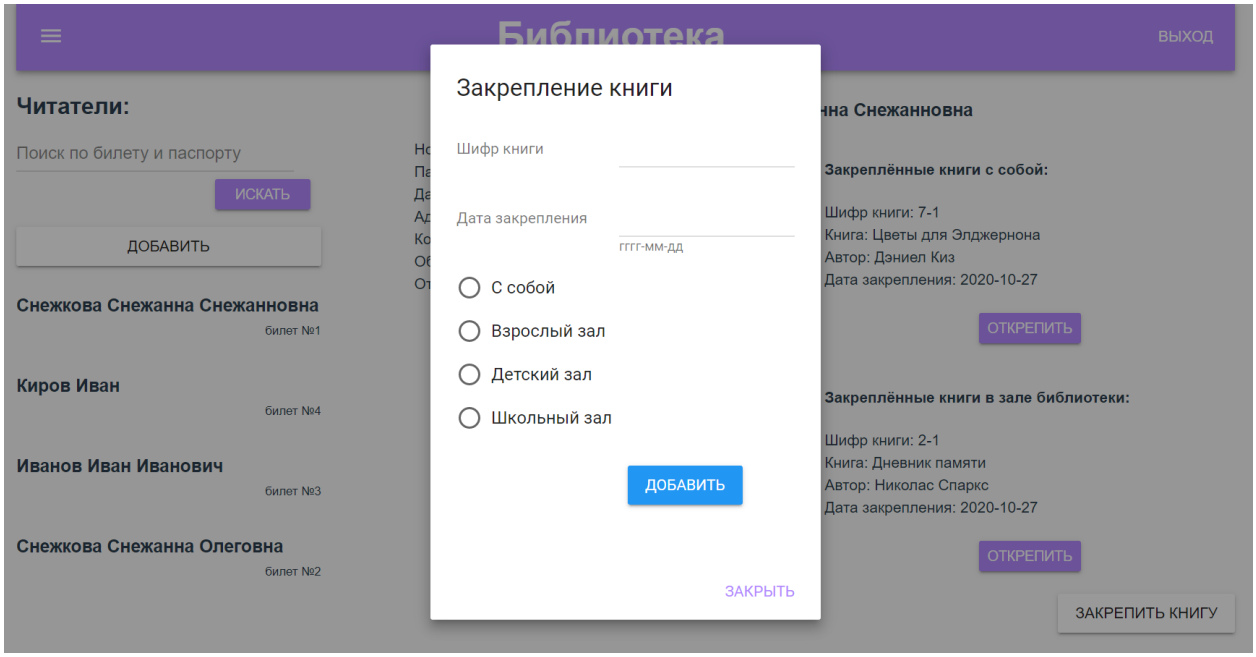
Компонент ReaderAdd – форма для добавления читателя



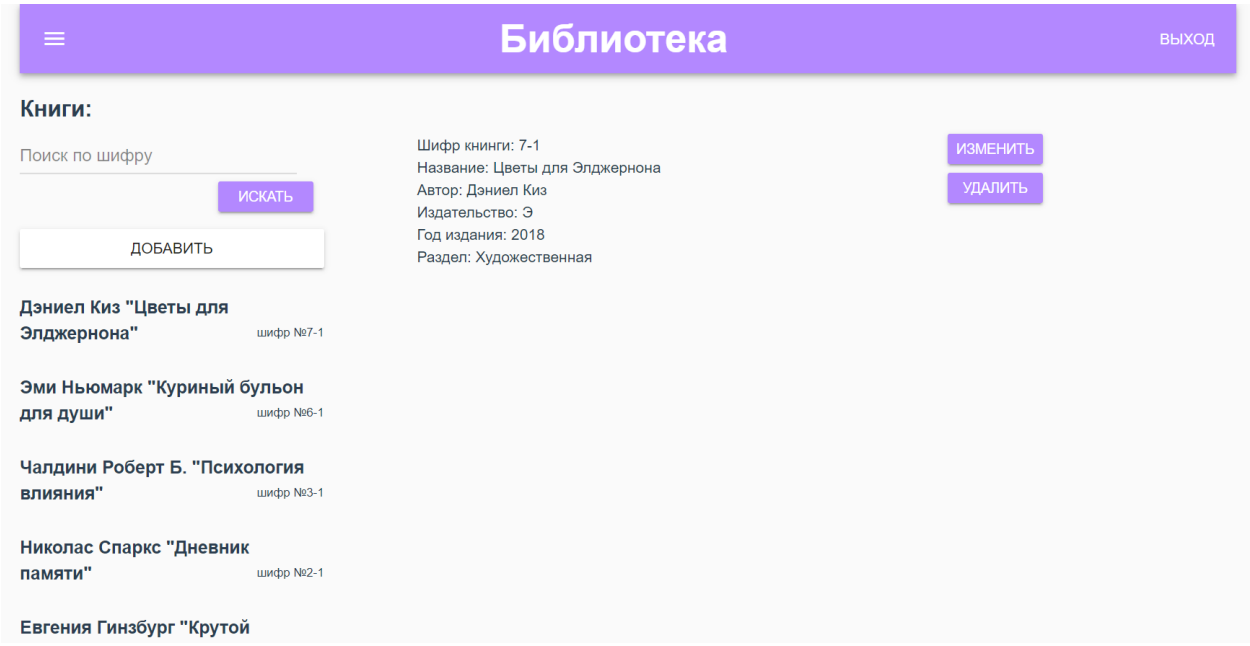
Компонент ReaderUpdate – форма для изменения данных читателя



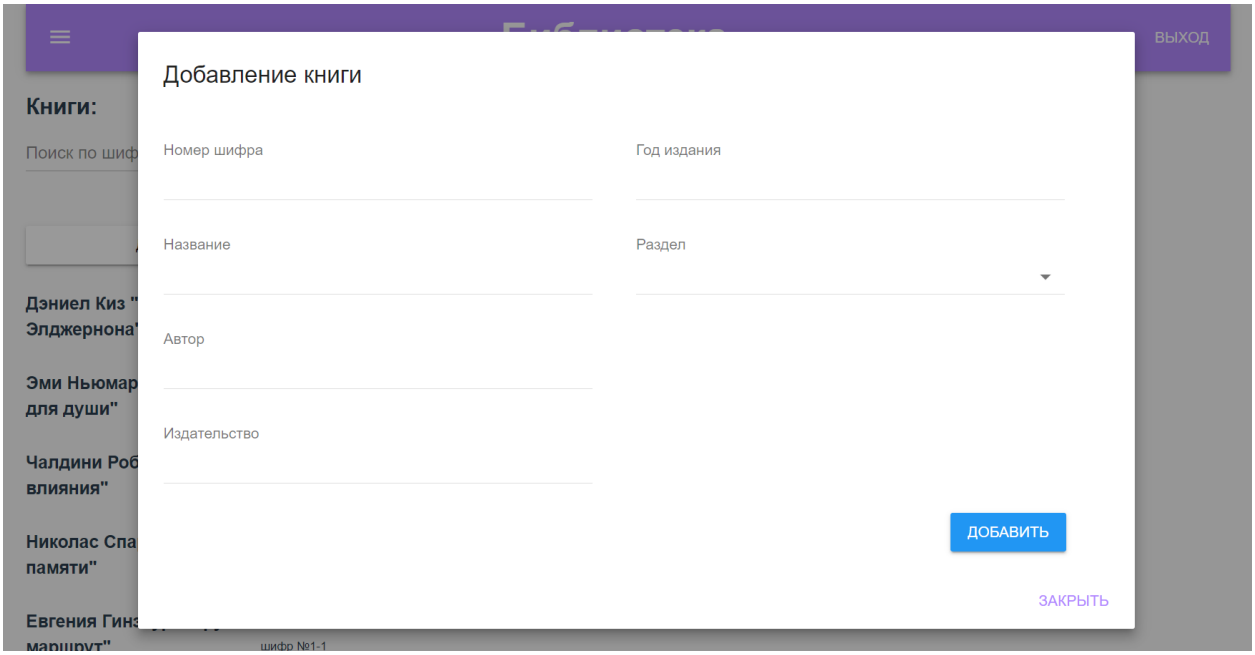
Компонент ReaderFix – форма для закрепления книги к читателю



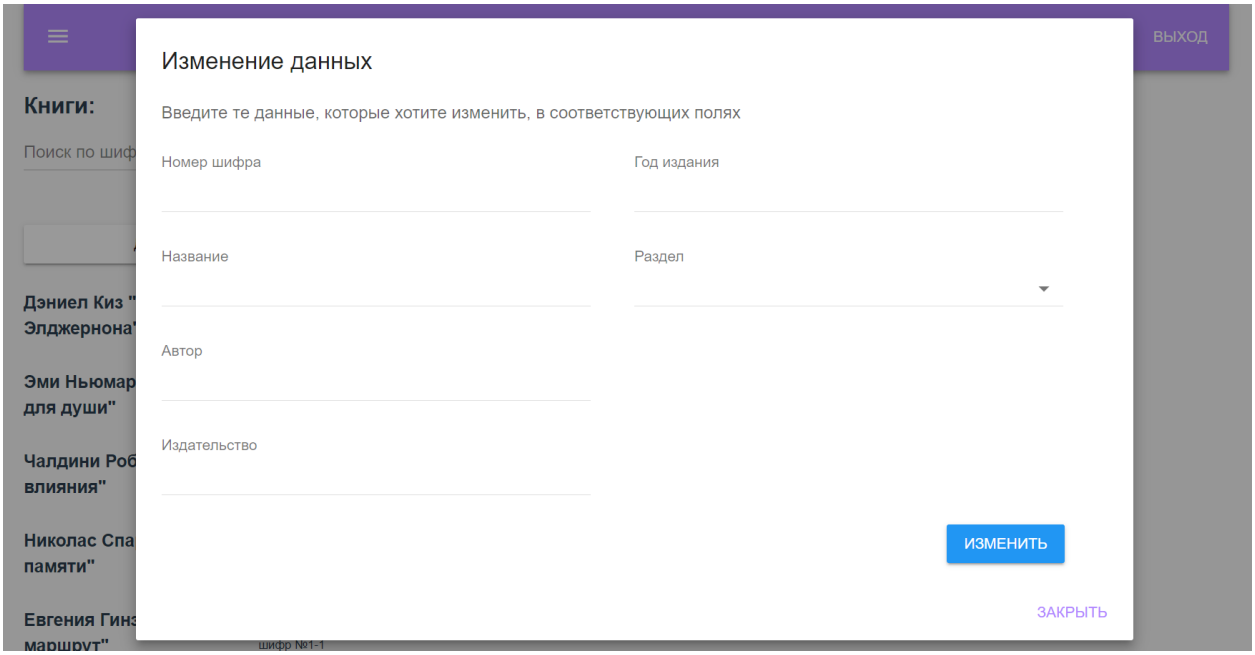
Компонент Books - основной интерфейс для вывода всех книг в библиотеке. Кликая по выбранной книге или вводя ее шифр в поиск, открывается компонент Book, где выводится полная информация о ней. Также имеется возможность редактирования данных – кнопки при клике, на которых открывается соответствующее окно или выполняется соответствующее действие.



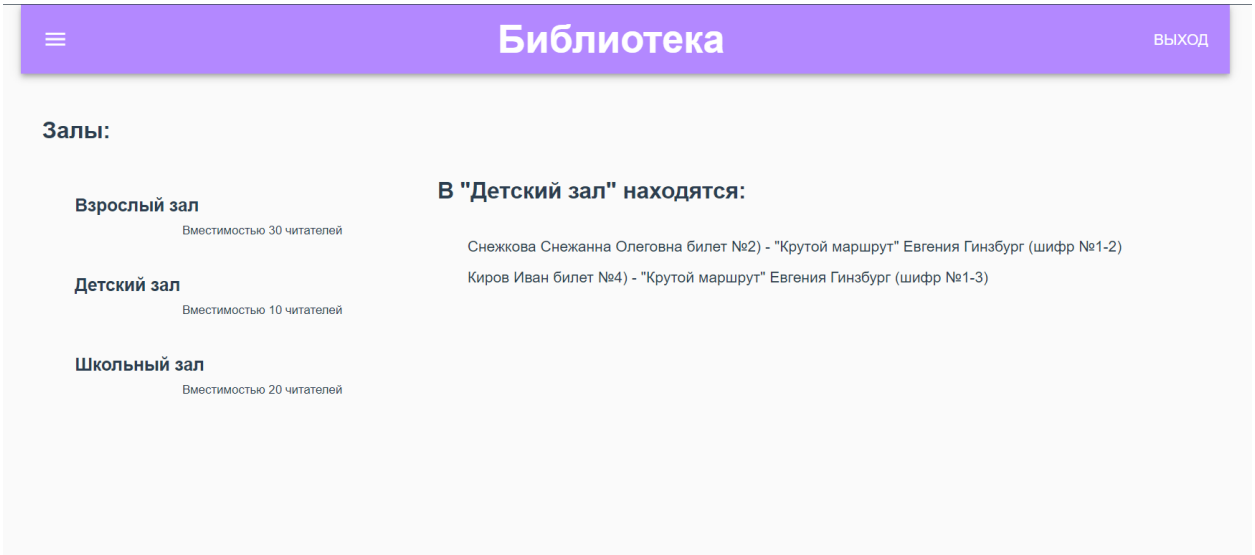
Компонент BookAdd – форма для добавления книги



Компонент ReaderUpdate – форма для изменения данных книги



Компонент Halls – страница, где отображены все залы в библиотеке. При нажатии на выбранный зал открывается информация, о находящихся на данных момент читателей и книг, которые они взяли.



В компонентах Question2, Question3, Question4, Question5 содержится информация по соответствующему запросу

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

Кто из читателей взял книгу более месяца тому назад?

Снежкова Снежанна Олеговна (билет №2) - "Крутой маршрут" Евгения Гинзбург (шифр №1-1) 2020-09-09

ЗАКРЫТЬ

• Записать в библиотеку нового читателя.
• Исключить из списка читателей людей, записавшихся в библиотеку более года назад и не прошедших перерегистрацию. (Найти читателя можно как по читательскому билету, так и по паспортным данным)
• Списать старую или потерянную книгу.

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

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

Снежкова Снежанна Снежанновна (билет №1):

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

Иванов Иван Иванович (билет №3):

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

ЗАКРЫТЬ

• Принять книгу в фонд библиотеки.

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

Сколько в библиотеке читателей младше 20 лет?

Всего 3:

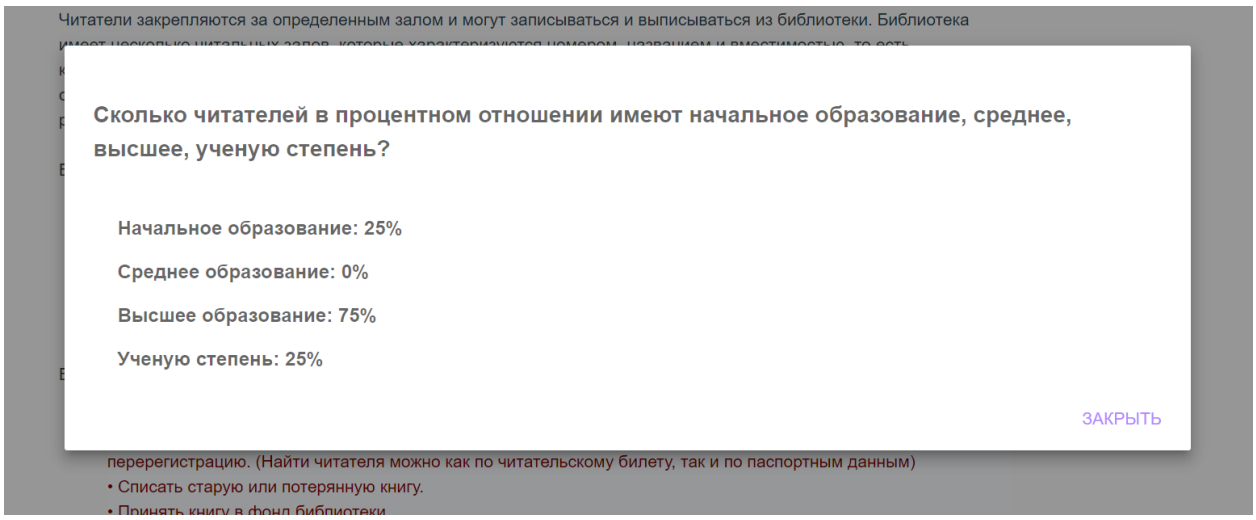
Снежкова Снежанна Снежанновна (билет №1) - дата рождения 2006-10-01

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

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

ЗАКРЫТЬ

перерегистрацию. (Найти читателя можно как по читательскому билету, так и по паспортным данным)
• Списать старую или потерянную книгу.



Выводы:

В ходе лабораторной работы были получены практические навыки и умения реализации web-сервисов средствами Django REST framework, Vue.js, Muse-UI. Был реализован сайт для варианта 2 («Библиотека»), по средствам использования фреймворка Django REST для серверной части, javascript-фреймворк Vue.js и Muse-UI для создания пользовательского интерфейса практического задания.