

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

Факультет ИКТ

Образовательная программа Интеллектуальные системы в гуманитарной сфере

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

О Т Ч Е Т

по курсовой работе

Тема задания: РЕАЛИЗАЦИЯ WEB-СЕРВИСОВ СРЕДСТВАМИ Django REST framework.

Обучающийся: Шупак Валентина К3343

Руководитель: Говоров А. И.

Подписи членов комиссии:

_____ Ф.И.О.
(подпись)

_____ Ф.И.О.
(подпись)

_____ Ф.И.О.
(подпись)

Дата _____

Санкт-Петербург
2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ.....	4
1.1. Описание предметной области	4
1.2. Описание функциональных требований.....	4
2. ОПИСАНИЕ СЕРВЕРНОЙ ЧАСТИ.....	6
2.1. Технологии создания серверной части	6
2.2. Создание базы данных и моделей	7
2.3. Создание отображений.....	8
2.4. Реализованные интерфейсы в панели Django REST	9
3. ОПИСАНИЕ КЛИЕНТСКОЙ ЧАСТИ.....	12
3.1. Технологии создания клиентской части.....	12
3.2. Реализованные интерфейсы	12
4. КОНТЕЙНЕРИЗАЦИЯ WEB-ПРИЛОЖЕНИЯ.....	18
ЗАКЛЮЧЕНИЕ.....	19
СПИСОК ЛИТЕРАТУРЫ	20
ПРИЛОЖЕНИЯ	21

ВВЕДЕНИЕ

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

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

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

Библиотекаря могут потребоваться следующие сведения о текущем состоянии библиотеки:

- Какие книги закреплены за определенным читателем?
- Кто из читателей взял книгу более месяца тому назад?
- Сколько читателей в процентном отношении имеют начальное образование, среднее, высшее, ученую степень?

Библиотекарь может выполнять следующие операции:

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

1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

1.1. Описание предметной области

Предметной областью для данной курсовой работы является администрирование библиотеки. Основным пользователем проектируемой системы является работник библиотеки. В его задачи входит:

- Регистрация новых читателей
- Принятие книг в фонд библиотеки
- Списание старых и потерянных книг
- Исключение читателя из библиотеки
- Изменение номера читательского билета в случае перерегистрации
- Изменение шифра книги в случае переклассификации
- Учет книг за читателями

Кроме того, работнику библиотеки может понадобиться следующая информация:

- Какие книги закреплены за определенным читателем?
- • Кто из читателей взял книгу более месяца тому назад?
- • Сколько читателей в процентном отношении имеют начальное образование, среднее, высшее, ученую степень?

В библиотеке существует несколько читальных залов, каждый из которых имеет название и вмещает определенное количество человек. В каждом из этих залов может находиться разное количество экземпляров одной книги.

1.2. Описание функциональных требований

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

Следовательно, были определены функциональные требования:

- Функции добавления (Create):

- Принятие в фонд библиотеки новой книги
 - Регистрация новых читателей
 - Запись определенной книги за определенным читателем
- Функции просмотра (Read)
 - Список всех читателей, зарегистрированных в библиотеке
 - Список всех, имеющихся в библиотеке, книг
 - Количество экземпляров каждой книги в каждом читальном зале
 - Список читальных залов с информацией о них
 - Подробная информация об определенном читателе
 - Подробная информация об определенной книге
 - Учет всех книг
- Функции модификации (Update)
 - Изменение данных о читателе
 - Изменение шифра книги
 - Изменение количества экземпляров уже имеющейся книги
- Функции удаления (Delete)
 - Исключение читателя
 - Списание книги
- Функции авторизации и регистрации
- Функции выполнения запросов

2. ОПИСАНИЕ СЕРВЕРНОЙ ЧАСТИ

2.1. Технологии создания серверной части

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

- PostgreSQL 12 – свободная объектно-реляционная система управления базами данных. Благодаря свободной лицензии и открытому коду, PostgreSQL разрешается использовать бесплатно, изменять и распространять всем и для любых целей: личных, коммерческих или учебных. В силу того, что хранение базы данных происходит на сервере, а не локально, переход web-приложения со стадии разработки на стадию продакшена происходит быстрее и проще.
- Django 3 – свободный фреймворк для web-приложений на языке программирования Python, использующий шаблон проектирования MVC (Model-View-Controller). Контроллер классической модели MVC примерно соответствует уровню, который в Django называется Представление (англ. View), а презентационная логика Представления реализуется в Django уровнем Шаблонов (англ. Template). Из-за этого уровневую архитектуру Django часто называют «Модель-Шаблон-Представление» (MTV). Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других, например, Ruby on Rails.
- Django REST framework – удобный инструмент для работы с REST основанный на идеологии фреймворка Django. Rest (сокр. англ. Representational State Transfer, «передача состояния представления») — стиль построения архитектуры распределенного приложения. Данные в REST должны передаваться в виде небольшого количества стандартных форматов (например HTML, XML, JSON). Сетевой протокол, как и HTTP, должен поддерживать кэширование, не должен зависеть от сетевого слоя, не должен сохранять информацию о состоянии между парами «запрос-ответ». Утверждается, что такой подход обеспечивает масштабируемость системы и позволяет ей эволюционировать с новыми требованиями.

2.2.Создание базы данных и моделей

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

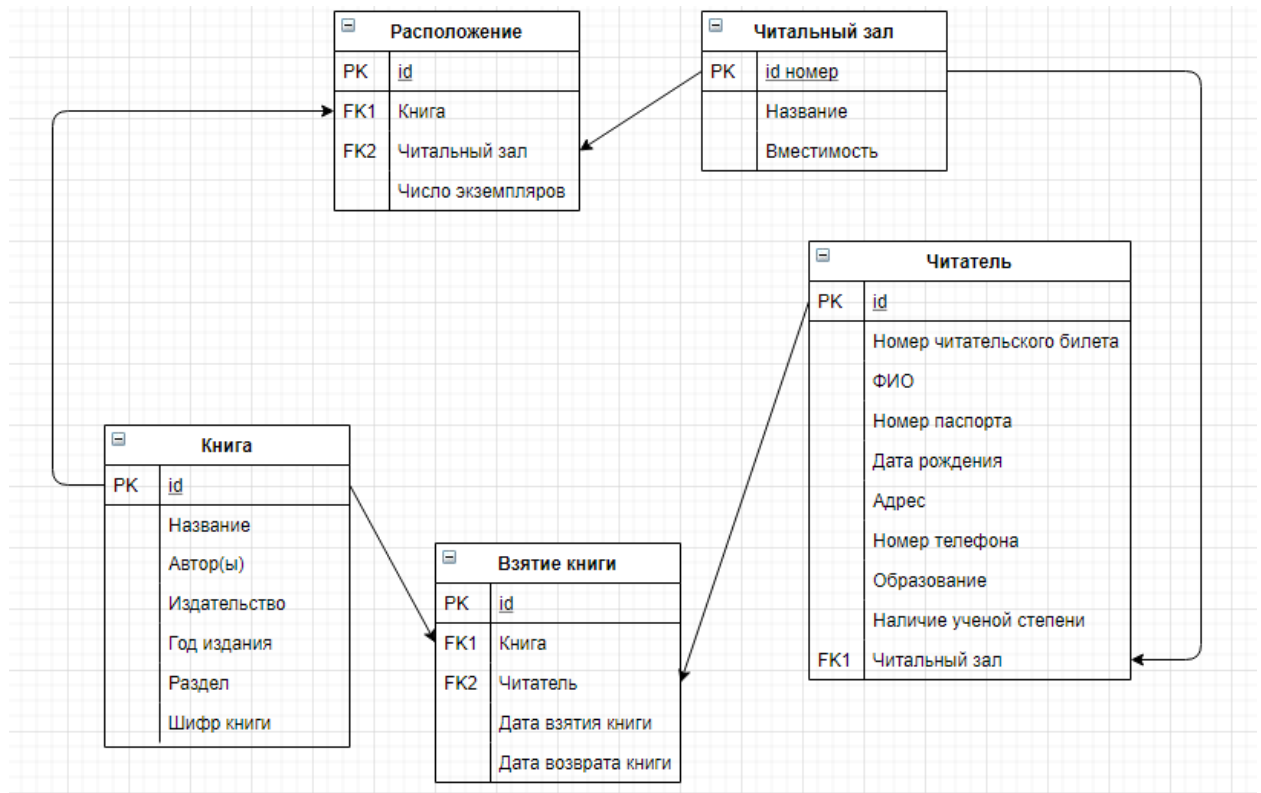


Рисунок 1 – схема данных

Полученная схема данных содержит 2 ассоциативные сущности – Расположение и Взятие книги, 3 стержневые сущности – Книга, Читальный зал, Читатель. Данные сущности были реализованы в качестве классов моделей Django в файле models.py, листинг кода которого приведен в приложении 1. В итоге, были разработаны следующие классы моделей:

- Class Book – Книга – данный класс моделей содержит информацию о названии книги, авторе (или авторах), издательстве, годе издания, секции: Художественная литература, Учебная литература, Научная литература, Документальная литература; шифре книги.
- Class ReadingRoom – Читальный зал – данный класс моделей содержит информацию о названии читального зала и его вместимости.
- Class BookPlace – Расположение книги – данный класс моделей содержит информацию о количестве экземпляров каждой книги в каждом зале.

- Class Reader – Читатель – данный класс моделей содержит информацию о номере читательского билета, ФИО читателя, номере паспорта, дате рождения, адресе, номере телефона, образовании: Начальное, Среднее неоконченное, Среднее, Высшее неоконченное, Высшее; о наличии ученой степени, о том, к какому читальному залу закреплён данный читатель.
- Class TakingBook – Взятие книги – данный класс моделей содержит информацию о том какой читатель когда взял какую книгу и когда он должен ее вернуть.

2.3. Создание отображений

Для того, чтобы обеспечить обмен данными между сервером, написанным на Django, и клиентской частью, написанной на Vue.js, необходимо провести сериализацию. Сериализация – процесс преобразования структур данных или объекта состояния в формат, который может быть сохранен или передан и реконструирован позже. Среда сериализации в Django предоставляет механизм для «перевода» моделей Django в другие форматы, например, в JSON, который принимает Vue.js. Все сериализаторы для созданных моделей описаны в файле `serializers.py`, приведенном в приложении 2. Далее были созданы отображения, описанные в файле `views.py`, приведенном в приложении 3.

2.4. Реализованные интерфейсы в панели Django REST

Далее приведены некоторые из реализованных интерфейсов в панели Django REST:

1. Вывод книг с описанием количества экземпляров в каждом читальном зале. Скриншот приведен на рисунке 2.

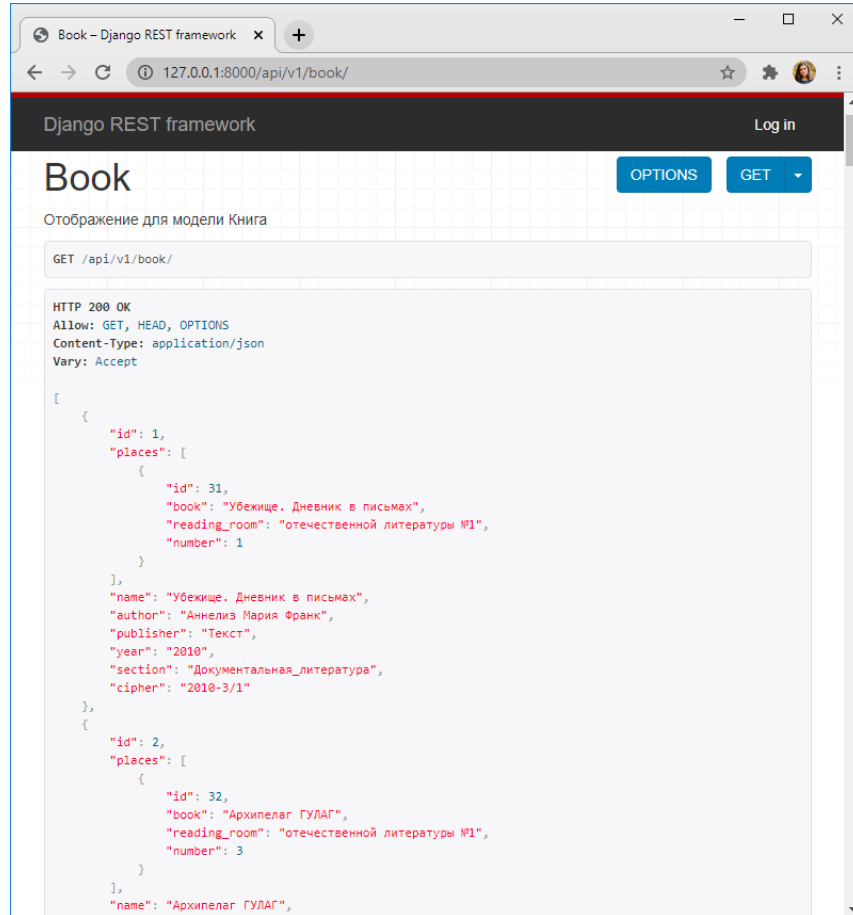


Рисунок 2 – Книги

2. Добавление нового читателя. Скриншот приведен на рисунке 3.

Reader OPTIONS

Отображение для модели Читатель

GET /api/v1/reader/create

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

{
 "detail": "Метод \"GET\" не разрешен."
}

Raw data HTML form

Номер читательского билета

ФИО

Номер паспорта

Дата рождения

Адрес

Номер телефона

Образование

Наличие ученой степени ☐

Reading room

POST

Рисунок 3 – Добавление нового читателя

3. Изменение количества экземпляров книги. Скриншот приведен на рисунке 4.

Book Place OPTIONS

Отображение для модели Расположение книги

GET /api/v1/book_place/1/update

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

{
 "detail": "Метод \"GET\" не разрешен."
}

Raw data HTML form

Количество экземпляров

POST

Рисунок 4 – Изменение количества экземпляров книги

4. Удаление записи из учета книг. Скриншот приведен на рисунке 5.

Taking Book DELETE OPTIONS

Отображение для модели Взятие книги

GET /api/v1/taking_book/4/delete

HTTP 405 Method Not Allowed
Allow: DELETE, OPTIONS
Content-Type: application/json
Vary: Accept

{
 "detail": "Метод \"GET\" не разрешен."
}

Рисунок 5 –удаление одной из записей

5. Вывод информации о читальных залах. Скриншот приведен на рисунке 6.

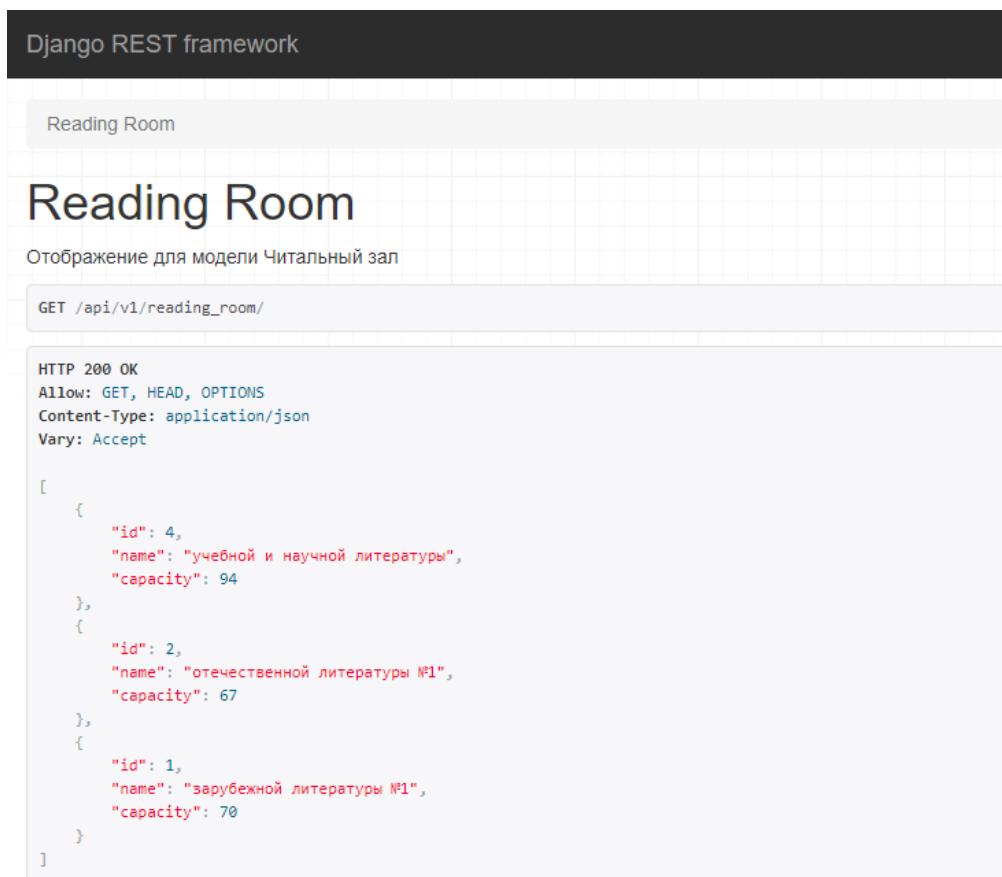


Рисунок 6 –Читальные залы

3. ОПИСАНИЕ КЛИЕНТСКОЙ ЧАСТИ

3.1. Технологии создания клиентской части

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

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

3.2. Реализованные интерфейсы

Согласно функциональным требованиям, были созданы пользовательские интерфейсы. Далее приведены примеры интерфейсов на каждую группу выявленных функциональных требований:

- Интерфейс с функцией добавления

Данный интерфейс содержит функцию добавления новой записи в модель «Книга». При нажатии на кнопку «Принять книгу в фонд библиотеки» появляется форма с полями ввода и выбора. После нажатия на кнопку «Подтвердить», в случае корректного заполнения формы, данные о новой книге вносятся в базу и страница автоматически обновляется, в противном случае, появляется уведомление о том, что данные введены неверно. Скриншот представлен на рисунке 7.

Web-сервис для работника библиотеки МЕНЮ ВОЙТИ

Список книг, которые имеются в библиотеке

Название

Автор(ы)

Издательство

Год издания

Секция

Шифр

ПОДТВЕРДИТЬ

- Убежище. Дневник в письмах
- Архипелаг ГУЛАГ
- Повесть о настоящем человеке
- Чернобыльская молитва. Хроника будущего
- Хладнокровное убийство
- Пианист. Варшавские дневники 1939-1945
- Бабий Яр
- Цинковые мальчики
- Крутой маршрут
- Четвертая высота
- Теория литературы. Поэтика. Учебное пособие
- Школа альпинизма. Учебное пособие
- История русской литературы XX века. Поэзия Серебряного века. Учебное пособие
- Философия: учебное пособие для вузов
- Практическая психология образования. Учебное пособие
- Словарь по обществознанию. Учебное пособие для абитуриентов вузов
- Русский авангард. Изобразительное искусство. Литература. Театр. Учебное пособие
- Компьютерные сети. Учебный курс.
- Основы теории литературы. Учебное пособие для вузов. В 2 частях. Частях 2.
- Стихосложение и литературный процесс
- Философская интерпретация художественных текстов на примере русской литературы XIX века. Учебное пособие
- Краткая история почти всего на свете

Рисунок 7 – добавление новой книги

- Интерфейс с функцией просмотра

Данный интерфейс содержит информацию о выбранном читателе. При нажатии на кнопку «Показать взятые книги» появляется таблица с информацией о книгах, которые закреплены за данным читателем. Скриншот представлен на рисунке 8.

Web-сервис для работника библиотеки

МЕНЮ Выйти

Читательский билет №40164750

ФИО	Хюппенен А.Д.
Номер паспорта	4246555377
Дата рождения	1997-01-16
Адрес	Невский пр-т, д. 57, Россия, Санкт-Петербург
Номер телефона	+79995240890
Образование	Высшее
Наличие ученой степени	нет
Читальный зал	учебной и научной литературы

Книга	Саргента. Краткая история человечества
Дата взятия	2020-05-09
Дата возврата	2020-05-12
Книга	Повесть о настоящем человеке
Дата взятия	2020-03-20
Дата возврата	2020-03-26

ДОБАВИТЬ КНИГУ

Книга

Дата взятия

Дата возврата

ДОБАВИТЬ

ИЗМЕНИТЬ ДАННЫЕ ПОКАЗАТЬ ВЗЯТЫЕ КНИГИ УДАЛИТЬ ЧИТАТЕЛЯ

Рисунок 8 – информация о читателе и взятых им книгах

- Интерфейс с функцией модификации

Данный интерфейс содержит функцию модификации записи в модели «Расположение книги». При нажатии на кнопку «Изменить количество элементов» рядом с каждой записью в таблице с информацией о количестве книг в каждом читальном зале появляется опция выбора, под самой таблицей появляется поле для ввода количества книг. После нажатия на кнопку «Изменить» данные в базе модифицируются и страница обновляется. Скриншот представлен на рисунке 9.

Web-сервис для работника библиотеки

МЕНЮ Войти

Информация о книге

Название	Школа альпинизма. Учебное пособие
Автор	Аленцев И., Брык Р.
Издательство	Москва
Год	2017
Секция	Учебная литература
Шифр	2017-2/12

Читальный зал	Количество экземпляров
<input checked="" type="radio"/> учебной и научной литературы	3

ИЗМЕНИТЬ КОЛИЧЕСТВО ЭКЗЕМПЛЯРОВ

32

ИЗМЕНИТЬ

ПЕРЕКЛАССИФИКАЦИЯ КОЛИЧЕСТВО ЭКЗЕМПЛЯРОВ УДАЛИТЬ КНИГУ

Рисунок 9 –изменение количества экземпляров книги в выбранном читальном зале

- Интерфейс с функцией удаления

Данный интерфейс содержит функцию удаления записи в модели «Читатель». При нажатии на кнопку «Удалить читателя» в таблице со всеми читателями рядом с ФИО каждого появляется опция выбора. После нажатия на кнопку «Подтвердить удаление» выбранный читатель удаляется из базы данных и страница автоматически обновляется. Скриншот представлен на рисунке 10.



Рисунок 10 – удаление читателя

- Интерфейс с авторизацией

Данный интерфейс включает в себя функцию авторизации пользователя. В случае правильного ввода имени пользователя и пароля пользователь получит уведомление, после чего будет перенаправлен на главную страницу web-сервиса. Скриншот представлен на рисунке 11.

Web-сервис для работы Добро пожаловать!

ОК

valueat1998

Пароль

.....

Покалуйста, введите пароль

ВОЙТИ

Регистрация

Рисунок 11 – Вход

- Интерфейс с запросом

Данный интерфейс представляет собой результат выполнения запроса. Скриншот представлен на рисунке 12.

Запрос №3	
Сколько читателей в процентном отношении имеют начальное образование, среднее, высшее, ученую степень?	
Образование	Процент читателей
Начальное	0
Среднее неоконченное	0
Среднее	0
Высшее неоконченное	0.25
Высшее	0.75
Есть ученая степень	0.5
Ученой степени нет	0.5

Рисунок 12 –результат выполнения запроса 3

- Интерфейс с фильтрацией

Функция фильтрации является расширением интерфейса «Книги». При нажатии на кнопку «Показать фильтры» появляются поля для выбора способа фильтрации: по названию, по автору и по шифру. Скриншот работы фильтров представлен на рисунках 13, 14 и 15.

Фильтр по автору
Дубровина И.В. ▼

Фильтр по названию
----- ▼

Фильтр по шифру
----- ▼

ОТФИЛЬТРОВАТЬ

Автор	Название	Шифр
Дубровина И.В.	Практическая психология образования. Учебное пособие	2004-2/15

Рисунок 13 –фильтрация по названию

Фильтр по автору
----- ▼

Фильтр по названию
Школа альпинизма. Учебное пособие ▼

Фильтр по шифру
----- ▼

ОТФИЛЬТРОВАТЬ

Автор	Название	Шифр
Аленцев И., Брызг Р.	Школа альпинизма. Учебное пособие	2017-2/12

Рисунок 14 –фильтрация по автору

Фильтр по автору
----- ▼

Фильтр по названию
----- ▼

Фильтр по шифру
2017-1/26 ▼

ОТФИЛЬТРОВАТЬ

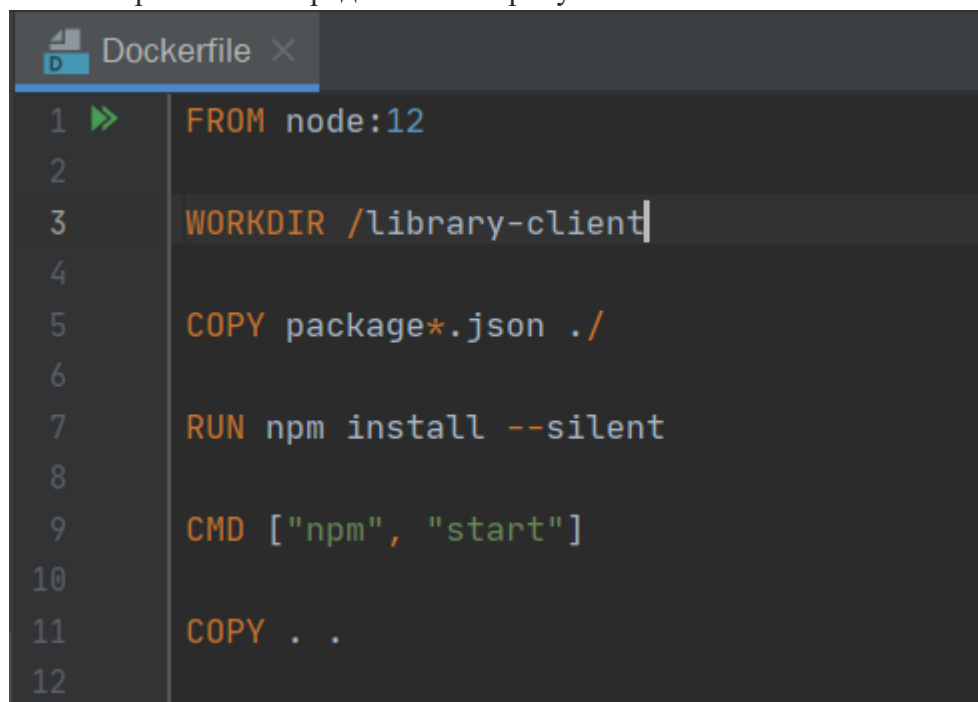
Автор	Название	Шифр
Харари Юваль Ной	Sapiens. Краткая история человечества	2017-1/26

Рисунок 15 –фильтрация по шифру

4. КОНТЕЙНЕРИЗАЦИЯ WEB-ПРИЛОЖЕНИЯ

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

Dockerfile – это сценарий, который состоит из последовательности команд и аргументов, необходимых для создания образа. Такие сценарии упрощают развёртывание и процесс подготовки приложения к запуску. Создается виртуальное окружение, внутри которого будет собираться/исполняться программа. Dockerfile для контейнеризации клиентской части разработанного web-приложения представлен на рисунке 16.



```
1  >> FROM node:12
2
3  WORKDIR /library-client
4
5  COPY package*.json ./
6
7  RUN npm install --silent
8
9  CMD ["npm", "start"]
10
11 COPY . .
12
```

Рисунок 16 –Dockerfile

Docker Compose - инструмент для создания и запуска многоконтейнерных Docker приложений. В Compose используется специальный файл для конфигурирования сервисов приложения. Затем используется простая команда для создания и запуска всех сервисов из конфигурационного файла. Это самая тривиальная реализация микросервисной архитектуры.

ЗАКЛЮЧЕНИЕ

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

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

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

- CRUD для моделей «Читатель», «Книга» и «Расположение книги»
- Просмотр информации обо всех созданных моделях
- Авторизация и регистрация в web-приложении
- Создание новой записи в модели «Взятие книги»
- Предоставление результатов выполнения запросов

СПИСОК ЛИТЕРАТУРЫ

1. Введение — Vue.js [Электронный ресурс]. – URL: <https://ru.vuejs.org/v2/guide/> (дата обращения 03.06.2020)
2. Django REST framework [Электронный ресурс]. – URL: <https://www.django-rest-framework.org/> (дата обращения 03.06.2020)
3. Современный учебник JavaScript [Электронный ресурс]. – URL: <https://learn.javascript.ru/> (дата обращения 03.06.2020)
4. Документация PostgreSQL – <https://postgrespro.ru/docs/postgresql/9.6/intro-what-is> (дата обращения: 27.06.2020)
5. Serializing Django objects – <https://docs.djangoproject.com/en/3.0/topics/serialization/> (дата обращения: 27.06.2020)
6. ViewSets – <https://www.django-rest-framework.org/api-guide/viewsets/> (дата обращения: 27.06.2020)
7. Class-based Views - <https://www.django-rest-framework.org/api-guide/views/> (дата посещения 28.06.2020)

ПРИЛОЖЕНИЯ

```

from django.db import models

class Book(models.Model):
    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_type = models.TextChoices('section_type', 'Художественная литература Учебная литература Научная литература Документальная литература')
    section = models.CharField('Секция', blank=True, choices=section_type.choices, max_length=50)
    cipher = models.CharField('Шифр книги', max_length=30)

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

    def __str__(self):
        return self.name

class ReadingRoom(models.Model):
    name = models.CharField('Название читального зала', max_length=50)
    capacity = models.IntegerField('Вместимость')

    class Meta:
        verbose_name = "Читальный зал"
        verbose_name_plural = "Читальные залы"

    def __str__(self):
        return "Читальный зал " + self.name

class BookPlace(models.Model):
    book = models.ForeignKey(Book, on_delete=models.CASCADE, related_name='places')
    reading_room = models.ForeignKey(ReadingRoom, on_delete=models.CASCADE)
    number = models.IntegerField('Количество экземпляров')

    class Meta:
        verbose_name = "Расположение книги"
        verbose_name_plural = "Расположение книги"

class Reader(models.Model):
    library_card = models.CharField('Номер читательского билета', max_length=16)
    fio = 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_type = models.TextChoices('section_type', 'Начальное Среднее неоконченное Среднее Высшее неоконченное Высшее')
    education = models.CharField('Образование', blank=True, choices=education_type.choices, max_length=50)
    academic = models.BooleanField('Наличие ученой степени')
    reading_room = models.ForeignKey(ReadingRoom, on_delete=models.CASCADE)

```

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

def __str__(self):
    return self.fio

class TakingBook(models.Model):
    book = models.ForeignKey(Book, on_delete=models.CASCADE)
    reader = models.ForeignKey(Reader, on_delete=models.CASCADE, related_name='books')
    date_take = models.DateField('Дата взятия книги')
    date_back = models.DateField('Дата возврата книги')

    class Meta:
        verbose_name = "Взятие книги"
        verbose_name_plural = "Взятие книги"
# Create your models here.
```

Приложение 2. Файл serializers.py

```
from rest_framework import serializers
from .models import Book, ReadingRoom, Reader, BookPlace, TakingBook

class ReadingRoomSerializer(serializers.ModelSerializer):
    """Сериализатор для модели Читальный зал"""

    class Meta:
        model = ReadingRoom
        fields = "__all__"

class ReaderCreateSerializer(serializers.ModelSerializer):
    """Сериализатор для добавления/обновления Читателя"""

    class Meta:
        model = Reader
        fields = "__all__"

class BookPlaceSerializer(serializers.ModelSerializer):
    """Сериализатор для модели Расположение Книги"""
    book = serializers.SlugRelatedField(slug_field="name", read_only=True)
    reading_room = serializers.SlugRelatedField(slug_field="name", read_only=True)

    class Meta:
        model = BookPlace
        fields = "__all__"

class BookPlaceUpdateSerializer(serializers.ModelSerializer):
    """Сериализатор для обновления Расположения Книги"""

    class Meta:
        model = BookPlace
        fields = ['id', 'number']

class BookPlaceCreateSerializer(serializers.ModelSerializer):
    """Сериализатор для добавления Расположения Книги"""

    class Meta:
        model = BookPlace
        fields = "__all__"

class BookSerializer(serializers.ModelSerializer):
    """Сериализатор для модели Книга"""
    places = BookPlaceSerializer(many=True)

    class Meta:
        model = Book
        fields = "__all__"

class BookUpdateSerializer(serializers.ModelSerializer):
```



```

"""Сериализатор для добавления/обновления Книги """

class Meta:
    model = Book
    fields = ['id', 'section', 'cipher']

class BookCreateSerializer(serializers.ModelSerializer):
    """Сериализатор для добавления/обновления Книги """

    class Meta:
        model = Book
        fields = "__all__"

class TakingBookSerializer(serializers.ModelSerializer):
    """Сериализатор для модели Взятие книги"""
    book = serializers.SlugRelatedField(slug_field="name", read_only=True)
    reader = serializers.SlugRelatedField(slug_field="fio", read_only=True)

    class Meta:
        model = TakingBook
        fields = "__all__"

class TakingBookCreateSerializer(serializers.ModelSerializer):
    """Сериализатор для добавления/обновления Взятия книги """

    class Meta:
        model = TakingBook
        fields = "__all__"

class ReaderSerializer(serializers.ModelSerializer):
    """Сериализатор для модели Читатель"""
    reading_room = serializers.SlugRelatedField(slug_field="name", read_only=True)
    books = TakingBookSerializer(many=True)
    class Meta:
        model = Reader
        fields = "__all__"

```

```

from django.shortcuts import render
from rest_framework import generics, permissions, viewsets, renderers
from rest_framework.views import APIView
from rest_framework.response import Response
from collections import Counter
from django.db.models import Count, Avg
from datetime import timedelta
from .models import Book, ReadingRoom, Reader, BookPlace, TakingBook
from .serializers import BookSerializer, ReadingRoomSerializer, ReaderSerializer, BookPlaceSerializer, \
    TakingBookSerializer, BookCreateSerializer, ReaderCreateSerializer, BookPlaceCreateSerializer, \
    TakingBookCreateSerializer, BookUpdateSerializer, BookPlaceUpdateSerializer

class BookViewSet(viewsets.ModelViewSet):
    """Отображение для модели Книга"""
    queryset = Book.objects.all()

    def get_serializer_class(self):
        if self.action == 'create':
            return BookCreateSerializer
        elif self.action == 'update':
            return BookUpdateSerializer
        elif self.action == 'destroy':
            return BookSerializer
        elif self.action == 'list':
            return BookSerializer
        elif self.action == 'retrieve':
            return BookSerializer

class ReadingRoomViewSet(viewsets.ModelViewSet):
    """Отображение для модели Читальный зал"""
    queryset = ReadingRoom.objects.all()
    serializer_class = ReadingRoomSerializer

class ReaderViewSet(viewsets.ModelViewSet):
    """Отображение для модели Читатель"""
    queryset = Reader.objects.all()

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

class BookPlaceViewSet(viewsets.ModelViewSet):
    """Отображение для модели Расположение книги"""
    queryset = BookPlace.objects.all()

```

```

def get_serializer_class(self):
    if self.action == 'create':
        return BookPlaceCreateSerializer
    elif self.action == 'update':
        return BookPlaceUpdateSerializer
    elif self.action == 'destroy':
        return BookPlaceSerializer
    elif self.action == 'list':
        return BookPlaceSerializer
    elif self.action == 'retrieve':
        return BookPlaceSerializer

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

    def get_serializer_class(self):
        if self.action == 'create':
            return TakingBookCreateSerializer
        elif self.action == 'update':
            return TakingBookCreateSerializer
        elif self.action == 'destroy':
            return TakingBookSerializer
        elif self.action == 'list':
            return TakingBookSerializer
        elif self.action == 'retrieve':
            return TakingBookSerializer

"""Запросы к курсовой работе"""

class Query1(APIView):
    """Какие книги закреплены за определенным читателем?"""

    def get(self, request):
        reader = request.GET.get('reader')
        books = Reader.objects.filter(id=reader)
        book = ReaderSerializer(books, many=True)
        return Response({'result': book.data})

class Query5(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').count()
        secondary_unfinished = Reader.objects.filter(education='Среднее_неоконченное').count()
        secondary = Reader.objects.filter(education='Среднее').count()
        high_unfinished = Reader.objects.filter(education='Высшее_неоконченное').count()
        high = Reader.objects.filter(education='Высшее').count()
        primary_percent = primary/readers
        secondary_unfinished_percent = secondary_unfinished/readers

```

```
secondary_percent = secondary/readers
high_unfinished_percent = high_unfinished/readers
high_percent = high/readers
academic = Reader.objects.filter(academic='True').count()
non_academic = Reader.objects.filter(academic='False').count()
academic_percent = academic/readers
non_academic_percent = non_academic/readers
return Response({'primary': primary_percent, 'secondary_unfinished': secondary_unfinished_percent,
                  'secondary': secondary_percent, 'high_unfinished': high_unfinished_percent, 'high': high_percent,
                  'academic': academic_percent, 'non_academic': non_academic_percent})
```

```
# Create your views here.
```