Министерство образования и науки Российской Федерации

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)

ОТДЕЛЕНИЕ КОМПЬЮТЕРНЫХ НАУК ИПМКН
Кафедра теоретических основ информатики (ТОИ)

ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК
Руководитель ООП
канд. техн. наук, доцент
_____ А.Л. Фукс
«_8_» _06_____2018 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ ХРАНЕНИЯ И ЧТЕНИЯ ЭЛЕКТРОННЫХ КНИГ В ФОРМАТЕ FB2

по основной образовательной программе подготовки бакалавров
направление подготовки
02.03.02 Фундаментальная информатика и информационные технологии

Вечканов Игорь Сергеевич

Руководитель ВКР
канд. техн. наук, доцент
_____А.В. Приступа
подпись
«_30_» _мая_____2018 г.

Автор работы
студент группы № 1441
_____И. С. Вечканов
подпись

Томск-2018

Реферат

Выпускная квалификационная бакалаврская работа, 38 с., 14 рисунков, 13 источников, 1 приложение.

ВЕБ-ПРИЛОЖЕНИЕ, ВЕБ-РАЗРАБОТКА, JAVASCRIPT, MONGODB, EXPRESSJS, BOOTSTRAP, NODEJS, JSON.

Объект исследования – технология разработки веб-приложения на языке JavaScript.

Цель работы – спроектировать и разработать веб-приложение для хранения и просмотра электронных книг в формате fb2.

Метод исследования – практическая работа на ЭВМ.

Основные результаты – спроектирована архитектура приложения, на основе которой создано веб-приложение, в котором реализованы инструменты организации структурированного хранения и просмотра электронных книг в формате fb2.

Оглавление

Реферат 2
ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ 4
Введение 5
1 Анализ 6
1.1 Определение требований 6
1.2 Обзор существующих решений 6
1.3 Используемые технологии 8
1.4 Формат FictionBook 11
2 Проектирование 15
2.1 MVC 15
2.2 База данных 16
2.2.1 MongoDB 16
2.2.2 Схема документа 17
3 Реализация 19
3.1 Создание проекта 19
3.2 Главная страница приложения 21
3.2.1 Загрузка книги на сервер 22
3.2.2 Система тегов 23
3.3 Страница с информацией о книге 25
3.3.1 Редактирование тегов 27
3.4 Приложение для чтения книг 28
3.4.1 Алгоритм разбивки контента на страницы 31
3.4.2 Синхронизация позиции чтения 32
Заключение 33
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ 34
Приложение А. Руководство пользователя 36

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ

AJAX – Asynchronous Javascript and XML (асинхронный JavaScript и XML), подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером.

API – набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) для использования во внешних программных продуктах.

DOM – Document Object Model (объектная модель документа), независящий от платформы и языка программный интерфейс, позволяющий программам и скриптам получить доступ к содержимому HTML-, XHTML- и XML-документов, а также изменять содержимое, структуру и оформление таких документов.

JSON – JavaScript Object Notation, текстовый формат обмена данными, основанный на JavaScript.

NoSQL – not only SQL (не только SQL), термин, обозначающий ряд подходов, направленных на реализацию хранилищ баз данных, имеющих существенные отличия от моделей, используемых в традиционных реляционных СУБД с доступом к данным средствами языка SQL.

LocalStorage – свойство глобального объекта браузера (window). Используется для хранения данных.

Введение

В современном мире, несмотря на бурное развитие интернета большая часть доступной человечеству информации по-прежнему сосредоточена в книгах. С развитием технологий и появлением новых электронных устройств книги все чаще сразу издаются в том числе и в электронном виде, а также происходит оцифровывание уже изданных.

Ежегодно продажи электронных книг растут высокими темпами. По данным «Яндекс.Деньги» с января по июль 2017 года число покупателей электронных книг на 13% превысило количество купивших печатные издания. А объем выручки крупнейшего сервиса электронных книг, компании ЛитРес, за 2017 год вырос на 50% по сравнению с 2016 годом. Это связано в первую очередь с тем, что стоимость электронных книг на сегодняшний день существенно меньше бумажных. Также отметим, что данная статистика не учитывает количество книг, получаемых из неофициальных источников.

С каждым годом у читателей, предпочитающих электронные книги бумажным, их накапливается все больше, в связи с чем появляется потребность не только в удобном хранении, но и в доступе к ним с любого своего устройства.

Все вышеперечисленное привело к идее создать приложение, которое позволяло бы пользователю хранить и читать книги на различных электронных устройствах.

1 Анализ

1.1 Определение требований

В ходе общения с заинтересованными сторонами были выявлены следующие требования к разрабатываемому приложению:

- Приложение должно быть развернуто на собственном сервере.
- Приложение должно позволять удобно организовывать хранение файлов электронных книг.
- Приложение должно иметь возможность загружать и скачивать книги.
- Приложение должно позволять просматривать (читать) книги на устройствах пользователя.
- Приложение должно уметь синхронизировать последнюю просматриваемую позицию между различными устройствами пользователя.
- Пользовательский интерфейс приложения должен быть сверстан адаптивно, т.е. одинаково хорошо отображаться на устройствах с различными размерами экрана.
- Приложение должно поддерживать формат fb2.

1.2 Обзор существующих решений

В ходе подготовки к решению поставленной задачи были проанализированы следующие существующие решения:

1. Google Play Книги [9]. Сервис в первую очередь направлен на продажу книг из собственного магазина, но также позволяет пользователю загружать собственные книги в форматах epub и pdf. Данный сервис позволяет читать книги напрямую с веб-страницы. Помимо этого, для каждой книги существует возможность продолжить

чтение с того места, на котором пользователь остановился в предыдущий раз.

К недостаткам данного сервиса можно отнести следующее:

- ограничение количества загружаемых книг – пользователь может загрузить только 1000 книг;
- отсутствие поддержки формата fb2 (данное ограничение можно обойти, сконвертировав пользовательские файлы в формат epub);
- отсутствие возможности каким-либо образом структурировать хранимые книги для облегчения дальнейшего поиска;
- отсутствие возможности установить данный сервис на собственном сервере пользователя.

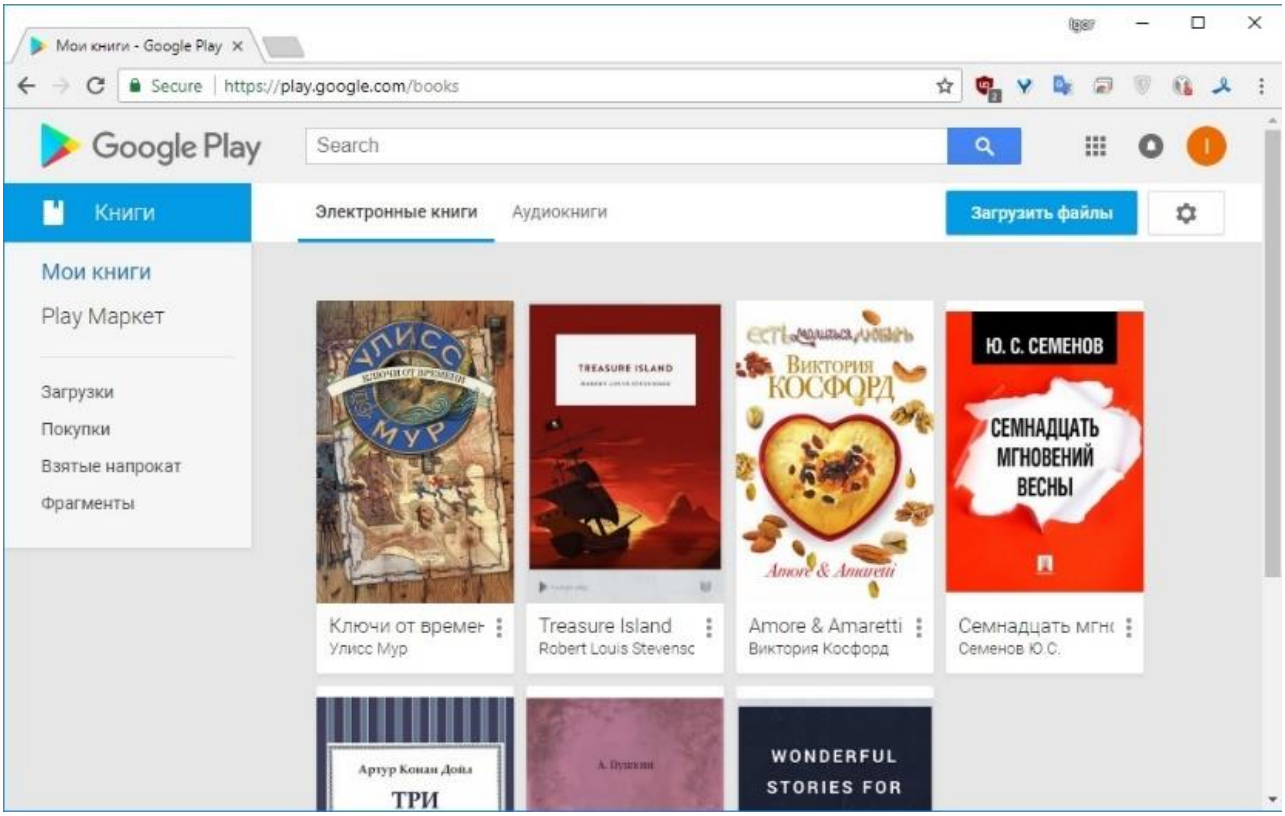


Рисунок 1 – Главная страница Google Play Книги

2. Readium-js-viewer [13]. Web-приложение, предназначенное для чтения книг в формате epub. Данное приложение возможно

установить на свой сервер. В Readium-js-viewer реализована возможность продолжить чтение с предыдущей позиции, но только в пределах одного браузера. Для хранения последней позиции используется LocalStorage.

Имеет следующие недостатки:

- поддержка единственного формата epub;
- отсутствие возможности упорядочить хранимые книги;
- отсутствие синхронизации последней позиции между различными браузерами, устройствами.

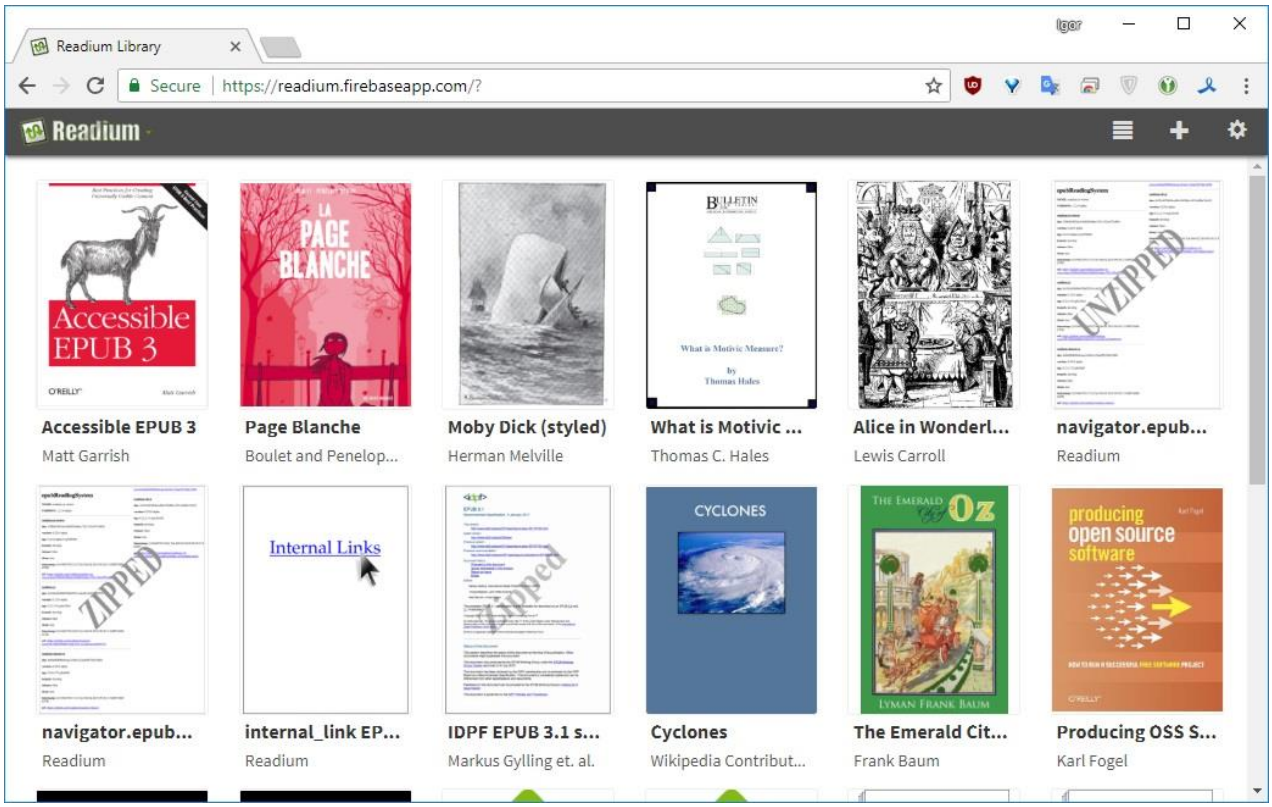


Рисунок 2 – Главная страница Readium

1.3 Используемые технологии

Поскольку целью настоящей работы является разработка web-приложения, то необходимо определиться со стеком используемых

технологий. При выборе стека технологий определяющими явились следующие основные требования:

- высокая скорость разработки;
- наличие исчерпывающей документации;
- простота создания пользовательского интерфейса.

Проанализировав данные требования, в частности требование к скорости разработки, принято решение использовать единый язык программирования для разработки серверной и клиентской части. Так как для написания клиентской части альтернатив языку JavaScript нет, то для разработки серверной части использована платформа Node.js. Таким образом, для серверной части выбран следующий технологический стек:

1. Node.js или Node – 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 существует 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 и espruino). В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.
2. Express.js (или Express) – быстрый, гибкий, минималистичный веб-фреймворк, который является де-факто стандартным фреймворком для Node.js.

3. Handlebars – шаблонизатор позволяющий строить эффективные семантические шаблоны. Несмотря на то, что стандартным шаблонизатором для Express.js является Jade, в настоящей работе выбор сделан в пользу Handlebars с целью обеспечить выполнение требования простоты создания пользовательского интерфейса. В отличии от Jade где, применяется свой собственный язык разметки, в Handlebars шаблоны выглядят как обычный HTML, со встроенными handlebars-выражениями.

4. MongoDB – документоориентированная система управления базами данных (СУБД) с открытым исходным кодом, не требующая описания схемы таблиц. Классифицирована как NoSQL, использует JSON-подобные документы и схему базы данных. Т.к. в MongoDB используются JSON-подобные документы, с которыми очень удобно работать в JavaScript, было принято решение использовать данную СУБД.

При выборе стека технологий, используемого для разработки клиентской части, был добавлен дополнительный критерий: выбранная технология должна позволять достаточно легко разрабатывать интерфейс, который адаптируется для наилучшего отображения информации на устройстве пользователя, в зависимости от физических размеров экрана, то есть адаптивный интерфейс. С учетом данного критерия выбраны следующие технологии:

1. Bootstrap – популярный HTML, CSS, и JS фреймворк для разработки адаптивных и мобильных web-проектов.

2. jQuery – библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с

AJAX. Помимо этого, выбор jQuery обусловлен тем, что он входит в зависимости фреймворка Bootstrap.

1.4 Формат FictionBook

FictionBook – формат представления электронных версий книг в виде XML-документов, где каждый элемент книги описывается своими тегами. Стандарт призван обеспечить совместимость с любыми устройствами и форматами. XML позволяет легко создавать документы, готовые к непосредственному использованию и программной обработке (преобразованию, хранению, управлению) в любой среде. Документы, обычно имеющие расширение .fb2, могут содержать структурную разметку основных элементов текста, некоторое количество информации о книге, а также вложения с двоичными файлами, в которых могут храниться иллюстрации, например обложка. Стандарт был разработан группой разработчиков во главе с Дмитрием Грибовым и Михаилом Мацневым.

Документ FictionBook состоит из двух частей - заголовка (<description>) и тела (<body>). За ними могут следовать включенные в документ двоичные файлы.

Заголовок состоит из трех частей:

1. <title-info> - информация о содержимом книги.
2. <document-info> - информация о XML-документе.
3. <publish-info> - информация о бумажном издании, на котором основывается данный документ XML.

В заголовке обязательными являются только <title-info> и <document-info>. <title-info> описывает название произведения, автора, жанр, язык и может содержать дополнительную информацию (например, аннотацию, ключевые слова, язык оригинала и т.п.) <publish-info> описывает бумажное издание, на котором основан документ и является необязательным.

Тело документа FictionBook (<body>) состоит из одной или более части (<section>). <section> может быть вложена внутрь другой <section>, при этом одновременное вложение текстов и <section> не допускается.

Всегда допустимые теги внутри <section>:

1. <title> - аналог атрибута title, допускающий использование параграфов (<p>) и пустых строк (<empty-line>) для сложных заголовков.
2. <image> - рисунок.
3. <epigraph> - эпиграф, допустим в начале, может быть несколько.
4. <annotation> - краткое описание главы.

Теги, допустимые в любом порядке и количестве внутри <section>, если в нее не вложена другая <section>:

1. <p> - параграф. Допускает использование стилей <style>, выделения и сносок <note>.
2. <image> - рисунок.
3. <empty-line> - пустая строка.
4. <poem> - стихи.
5. <subtitle> - смысловой разделитель, имеющий внутри себя текст.
6. <cite> - цитата.

Ниже представлен пример электронной книги в формате FictionBook (fb2).

```
<?xml version="1.0" encoding="UTF-8"?>
<FictionBook xmlns="http://www.gribuser.ru/xml/fictionbook/2.0" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:l="http://www.w3.org/1999/xlink">
    <description>
```

```
        <title-info>
            <genre>sf_cyberpunk</genre>
            <author>
                <first-name>FirstName</first-name>
                <last-name>LastName</last-name>
            </author>
            <src-lang>en</src-lang>
            <book-title>Book Title</book-title>
            <annotation>...</annotation>
            <coverpage>
                <image l:href="#cover.jpg"/>
            </coverpage>
            <lang>ru</lang>
            <sequence name="Альтернатива. Фантастика"/>
        </title-info>
        <document-info>
            <author>
                <first-name>Name</first-name>
            </author>
        </document-info>
    </description>
    <body>
        <title>
            <p>Author Name</p>
            <empty-line />
            <p>Book Title</p>
        </title>
        <section>
        ...
        </section>
```

```
    </body>
    <binary content-type="image/jpeg" id="cover.jpg">...</binary>
</FictionBook>
```

2 Проектирование

2.1 MVC

Перед тем как приступить к разработке, необходимо определиться с архитектурой web-приложения. Наиболее популярным архитектурным паттерном для построения web-приложений на сегодняшний день является паттерн “модель-представление-контроллер (model-view-controller)” [3].

В рамках данного паттерна подразумевается выделение трех отдельных ролей:

1. Модель – это объект, содержащий данные предметной области, логику создания и управления данными предметной области, а также предоставляет API для получения доступа к данным и операциями над ними.

2. Представление отображает содержимое модели средствами графического интерфейса.

3. Контроллер содержит логику изменения модели на основании действий пользователя.

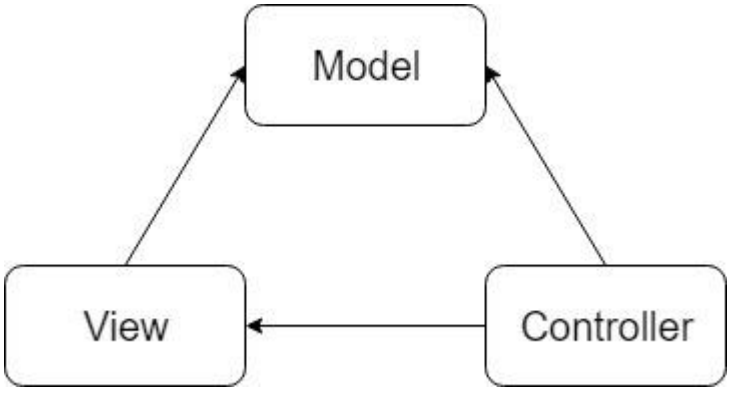


Рисунок 3 – Model-View-Controller

В случае web-приложения это выглядит так: браузер пользователя отправляет запрос на сервер, контроллер обрабатывает запрос, получает необходимые данные из модели и отправляет их в представление (view). Представление получает данные из контроллера и превращает их в HTML страничку, которую контроллер в итоге отправляет пользователю.

2.2 База данных

2.2.1 MongoDB

MongoDB реализует подход к построению баз данных, где нет таблиц, схем, запросов SQL, внешних ключей и многих других вещей, которые присущи объектно-реляционным базам данных. В отличие от реляционных баз данных MongoDB предлагает документо-ориентированную модель данных. Для хранения в MongoDB применяется формат, который называется BSON (сокращение от binary JSON). BSON позволяет работать с данными быстрее: быстрее выполняется поиск и обработка. Хотя надо отметить, что BSON в отличие от хранения данных в формате JSON имеет небольшой недостаток: в целом данные в JSON-формате занимают меньше места, чем в формате BSON, с другой стороны, данный недостаток с лихвой окупается скоростью. Если реляционные базы данных хранят строки, то MongoDB хранит документы. В отличие от строк документы могут хранить сложную по структуре информацию. Документ можно представить как хранилище ключей и значений. Ключ представляет простую метку, с которым ассоциировано определенный кусок данных. Однако при всех различиях есть одна особенность, которая сближает MongoDB и реляционные базы данных. В реляционных СУБД встречается такое понятие как первичный ключ. Это понятие описывает некий столбец, который имеет уникальные значения. В MongoDB для каждого документа имеется уникальный идентификатор, который называется _id. И если явным образом не указать его значение, то MongoDB автоматически сгенерирует для него значение. Каждому ключу сопоставляется определенное значение. Но здесь также надо учитывать одну особенность: в реляционных базах есть четко очерченная структура, где есть поля. Если какое-то поле не имеет значение, ему (в зависимости от настроек конкретной бд) можно присвоить значение NULL. В MongoDB все иначе. Если какому-то ключу не сопоставлено значение, то этот ключ просто опускается в документе и не употребляется. Если в реляционных базах

данных содержимое составляют таблицы, то в MongoDB база данных состоит из коллекций. В реляционных базах данных таблицы хранят однотипные жестко структурированные объекты, коллекции же могут содержать самые разные объекты, имеющие различную структуру и различный набор свойств. Система хранения данных в MongoDB представляет набор реплик. В этом наборе есть основной узел, а также может присутствовать набор вторичных узлов. Все вторичные узлы сохраняют целостность и автоматически обновляются вместе с обновлением главного узла. И если основной узел по каким-то причинам выходит из строя, то один из вторичных узлов становится главным. Отсутствие жесткой схемы базы данных и в связи с этим потребности при малейшем изменении концепции хранения данных пересоздавать эту схему значительно облегчают работу с базами данных MongoDB и дальнейшим их масштабированием.

2.2.2 Схема документа

Для хранения в документе базы данных, были выбраны следующие атрибуты:

- fileName – имя файла электронной книги.
- author – информация об авторе книги.
  - first – имя.
  - middle – отчество.
  - last – фамилия.
- title – название книги.
- annotation – аннотация (краткое описание книги).
- cover – обложка книги.
  - type – формат файла (png/jpeg).
  - content – закодированное в base64 изображение.
- tags – массив тегов.

- progress – информация о прогрессе чтения книги пользователем.
    - section – последняя читаемая глава.
    - position – количество прочитанных в главе слов.

Ниже представлена получившаяся структура документа в виде JSON-объекта:

```
{
    fileName: "",
    autor: {
        first: "",
        middle: "",
        last: ""
    },
    title: "",
    annotation: "",
    cover: {
        type: "",
        content: ""
    },
    tags: [
        0: "Все"
    ],
    progress: {
        section: 0,
        position: 0
    }
}
```

3 Реализация

3.1 Создание проекта

Предпочтительным вариантом создания каркаса приложения является express-generator - инструмент для генерации приложений Express.js.

На первом шаге необходимо установить express-generator с помощью следующей команды:

npm install express-generator –g.

Далее необходимо сгенерировать каркас приложения с помощью команды:

express --view=hbs library.

Здесь “library” является именем приложения, а параметр “--view=hbs” указывает на то, что будет использоваться шаблонизатор “Handlebars”.

Последним шагом нужно установить необходимые зависимости:

cd library && npm install.

В результате получается каркас приложения. Express не диктует строгой структуры для файлов приложения. Таким образом получается следующая структура:

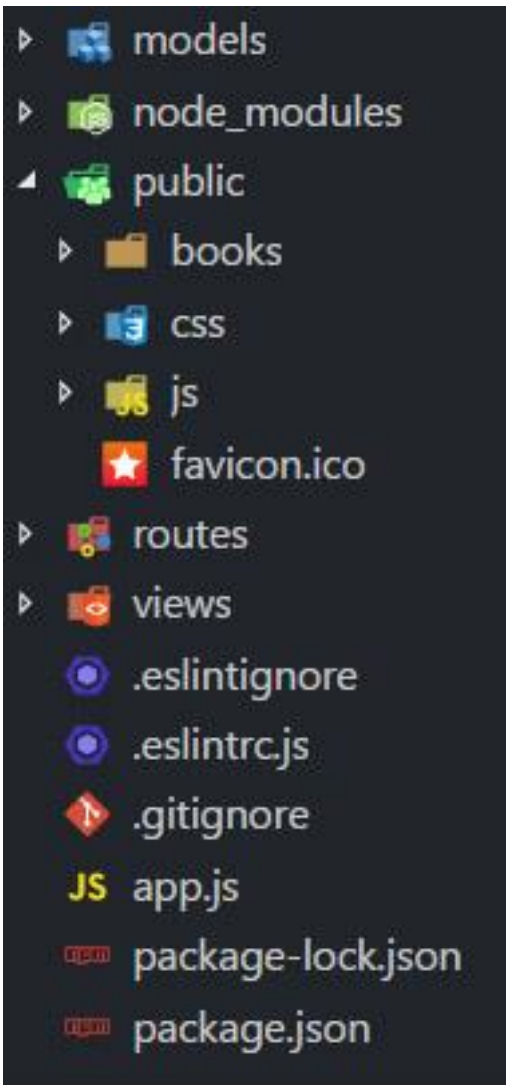


Рисунок 4 – Структура проекта

Основные папки и файлы проекта:

1. models – содержит классы моделей в соответствии с паттерном MVC.
2. node_modules – хранит зависимости проекта, автоматически загруженные npm.
3. В public/books будут помещаться загруженные пользователем файлы книг.
4. public/css и public/js – содержат клиентские стили и скрипты соответственно.
5. routes – хранит классы контроллеров.

6. view – шаблоны Handlebars, они же являются представлениями в MVC.
7. app.js – загрузчик приложения.
8. package.json – конфигурационный файл проекта.

3.2 Главная страница приложения

Основной проблемой при реализации главной страницы оказалась необходимость обеспечить пользователю возможность группировать хранимые книги. Для облегчения поиска и навигации среди списка всех книг, загруженных в приложение, принято решение предоставить пользователю возможность самостоятельно группировать книги так, как ему хочется, а не привязываться к существующим стандартам библиотечной каталогизации. Для реализации этой цели решено использовать систему тегов, то есть для каждой книги реализована возможность задания нескольких ключевых слов (тегов). При этом на главной странице пользователь сможет сам выбрать из списка тегов интересующий его на данный момент, и приложение отобразит все книги, которым присвоен выбранный тег.

Таким образом на главной странице приложения пользователю должны быть доступны следующие действия:

1. Загрузить книгу в приложение.
2. Посмотреть список всех загруженных книг.
3. Выбрать интересующую пользователя категорию книг.
4. Перейти к просмотру детальной информации о конкретной книге.

На рисунке ниже представлена диаграмма вариантов использования.

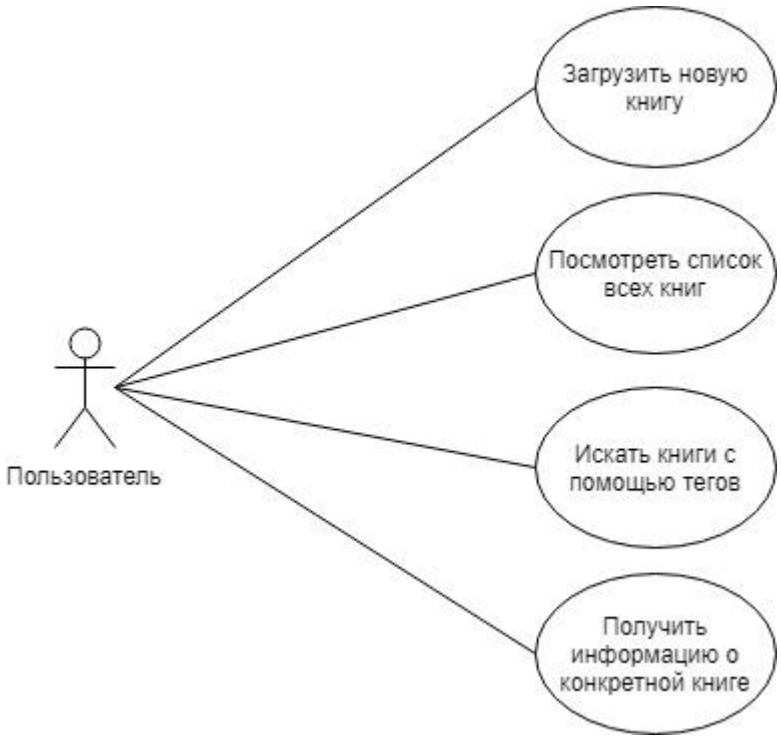


Рисунок 5 – Диаграмма вариантов использования главной страницы

3.2.1 Загрузка книги на сервер

Загрузка файла электронной книги на сервер осуществляется посредством отправки следующей формы:

<form ref='uploadForm'

id='uploadForm'

action='{{url}}'

method='post'

encType="multipart/form-data">

<input type="file" name="uploadFile" id="uploadFile"/>

</form>,

Где {{url}} – handlebars-выражение, на место которого будет вставлен url-адрес при генерации страницы. По данному адресу необходимо отправить содержимое формы.

После загрузки файла приложение, используя класс FB2Parser.js, парсит файл и создает новый документ в базе данных, добавляя туда информацию о полученной книге:

- имя файла;
- имя автора;
- название книги;
- аннотацию;
- обложку книги (например, обложка книги в формате fb2, хранится в виде изображения, закодированного в base64).

Из перечисленных полей, только имя файла является обязательным, так как остальные поля могут быть попросту незаполненные.

3.2.2 Система тегов

При проектировании базы данных принято решение использовать стратегию “все встроено”. Она заключается в том, что все связные данные встраиваются в один документ, что позволяет совершать большинство запросов к данным намного проще. То есть в каждом документе хранится список тегов, присвоенных конкретной книге. Недостатком данной стратегии является тот факт, что для получения списка всех тегов, нужно пройтись по всем документам и получить список тегов для каждого документа, а затем отфильтровать повторяющиеся теги.

Для отображения главной страницы необходимы два списка: список всех тегов и список книг, принадлежащий конкретному тегу. Для того чтобы упростить выбор всех книг, каждой загружаемой книге автоматически присваивается тег “Все”, который пользователь не может редактировать.

Для получения списка тегов и списка книг была реализована функция, которая возвращает объект вида:

```
{ 'books':books, 'tags':tags },
```

где books является массивом книг для конкретного тега, а tags – массив, содержащий все теги. Таким образом для получения всех книг, достаточно вызвать эту функцию, передав ей в качестве параметра тег “Все”. На рисунках 6 и 7 приведены примеры получившейся главной страницы приложения, открытой на ПК и мобильном устройстве соответственно.

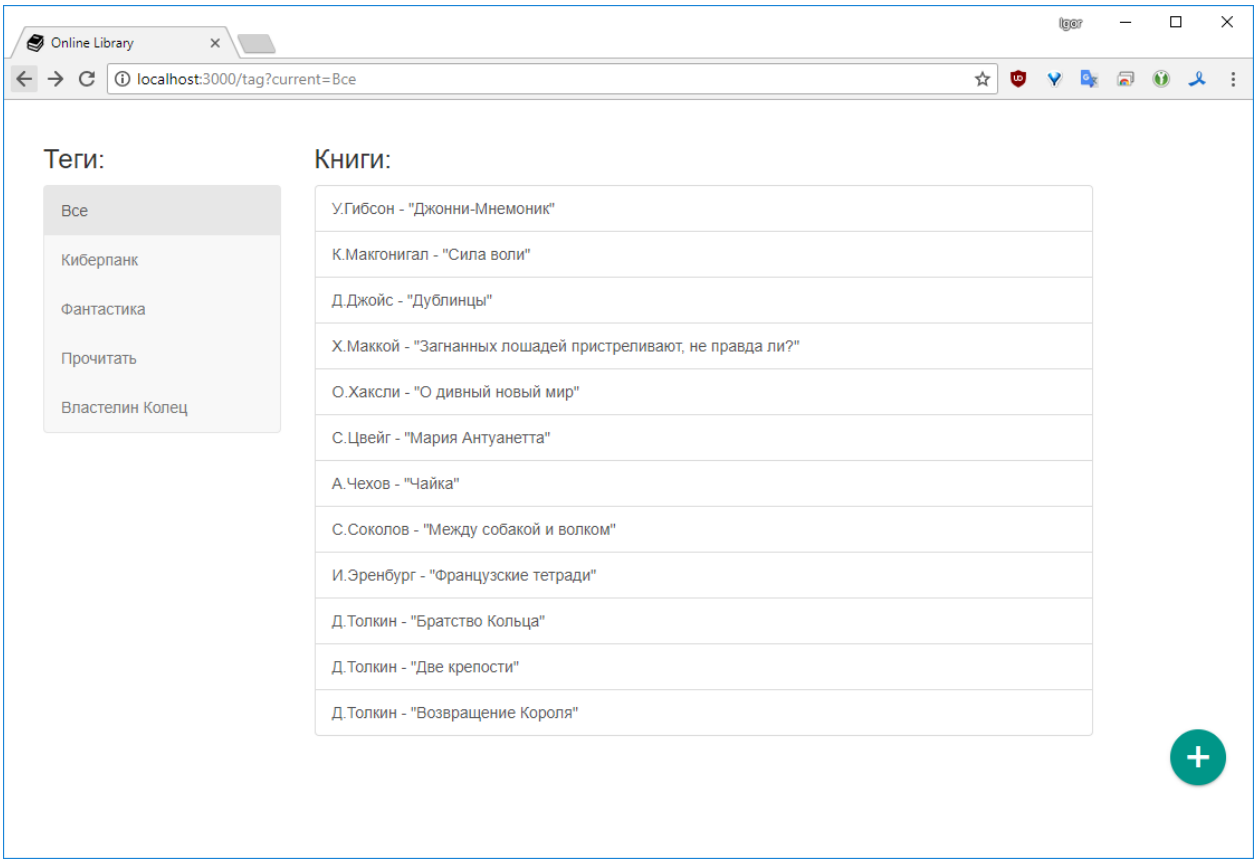


Рисунок 6 – Пример главной страницы, открытой на ПК

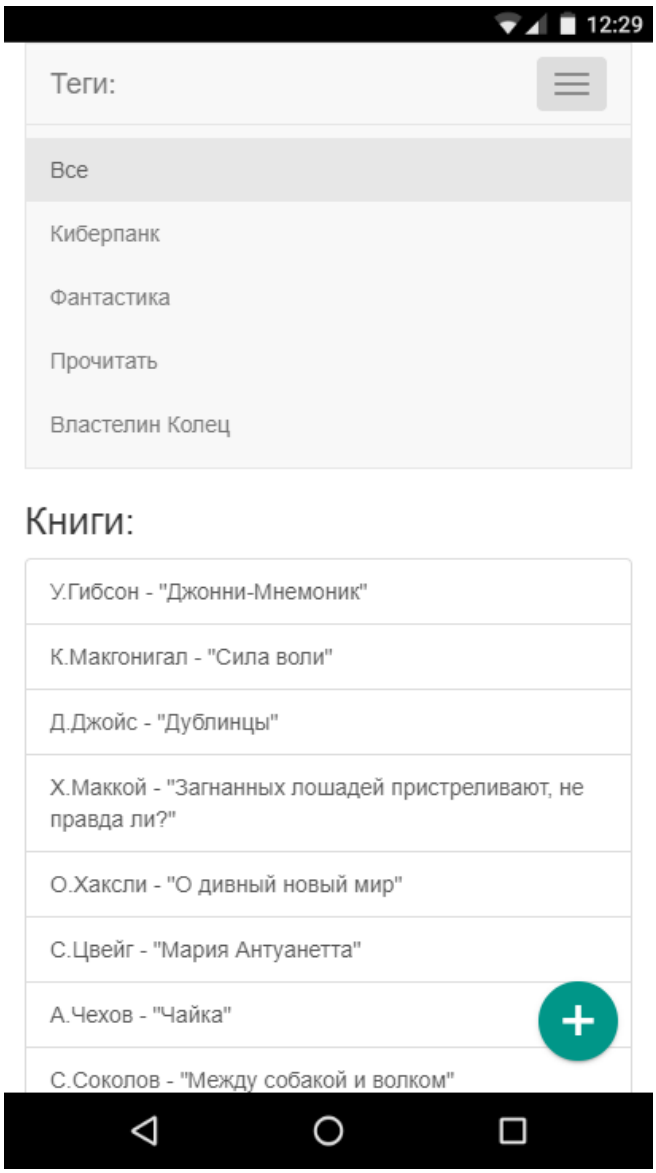


Рисунок 7 – Пример главной страницы, открытой на мобильном устройстве

3.3 Страница с информацией о книге

При нажатии на конкретную книгу в списке книг осуществляется переход на страницу, содержащую информацию об этой книге. При запросе страницы происходят следующие действия:

1. На сервер передается запрос, содержащий имя файла.
2. Класс DBHelper извлекает данные из базы данных, создает класс BookInfo и заполняет его извлеченными данными.

Класс DBHelper осуществляет все манипуляции с базой данных.

Класс BookInfo манипулирует информацией о книге, полученной из базы данных.

Оба этих класса являются моделями в паттерне MVC.

3. Контроллер страницы заполняет представление данными, полученными из класса BookInfo и возвращает готовую страницу.

Диаграмма последовательности, соответствующая исполнению запроса изображена на рисунке 8.

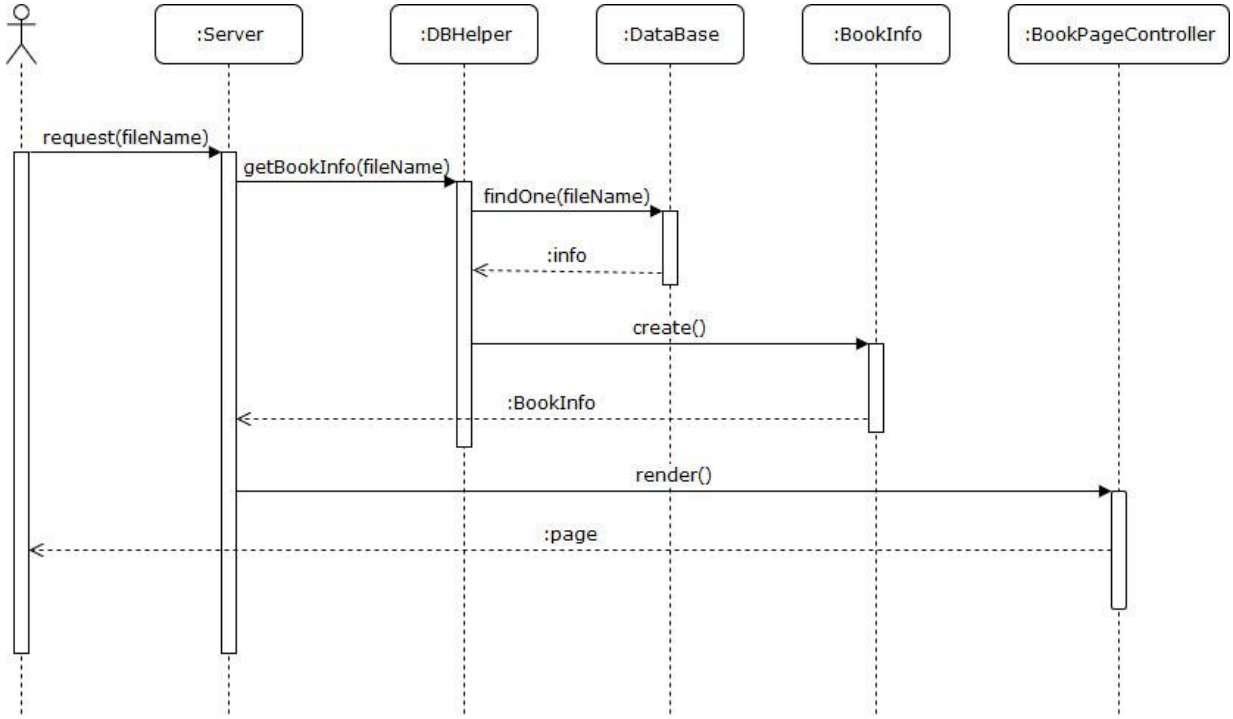


Рисунок 8 – Диаграмма последовательности выполнения запроса

На странице с информацией о книге пользователь может:

- скачать книгу;
- удалить книгу с сервера;
- редактировать теги;
- открыть книгу для чтения;

На рисунке 9 представлена диаграмма вариантов использования страницы с информацией о книге.

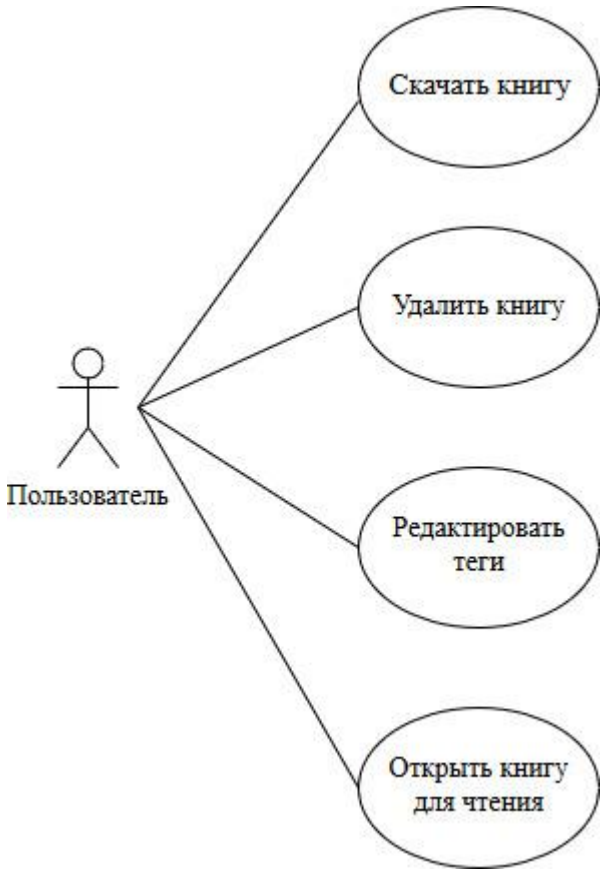


Рисунок.9 – Диаграмма вариантов использования страницы с информацией о книге

3.3.1 Редактирование тегов

Теги, заданные для книги, за исключением специального тега “Все”, отображаются в html-элементе input:

<input type="text" id="tags-input" value="{{tags}}" data-role="tagsinput" />

После загрузки страницы происходят следующие действия:

1. инициализируется компонент tagsInput.
2. tagsInput осуществляет AJAX запрос на сервер.
3. Сервер после выполнения запроса возвращает список всех тегов, ранее используемых в приложении, в виде JSON, без тега “Все”, этот список используется для автодополнения вводимых пользователем новых тегов.

Добавление новых тегов осуществляется посредством AJAX запроса с параметром “&addTag=” и именем добавляемого тега. Удаление происходит аналогичным образом, только используется параметр “&removeTag=” и имя удаляемого тега.

На рисунке 10 представлен пример страницы с информацией о книге, открытой на ПК.

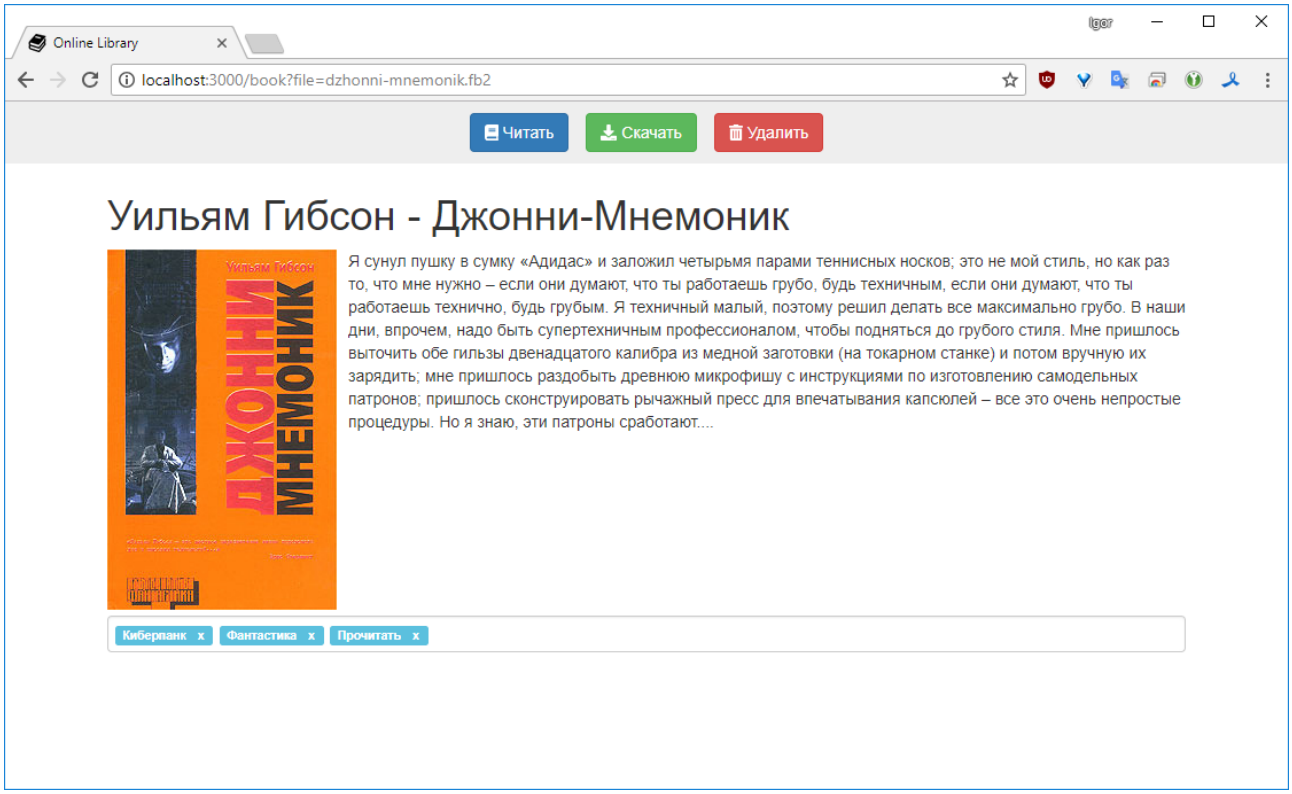


Рисунок 10 – Пример страницы с информацией о книге, открытой на ПК

3.4 Приложение для чтения книг

При нажатии на кнопку «читать» пользователь перенаправляется на страницу, содержащую приложение для чтения книг. Ниже приведена последовательность действий после загрузки страницы:

1. Инициализация приложения для чтения книг.
2. Приложение с помощью AJAX делает запрос контента на сервер.

3. На сервере приложение используя методы класса FB2Parser открывает файл электронной книги, затем, с помощью методов класса DBHelper, происходит извлечение информации о последней главе книги, которую читал пользователь. Если пользователь еще не читал данную книгу, то возвращается первая глава.

4. Приложение с помощью методов класса FB2Parser, извлекает контент из файла книги нужной главы.

5. FB2Parser конвертирует контент из формата книги в формат html.

6. Сервер возвращает JSON-объект:

```
{
        content: sectionContent,
        position: position,
        section: section
 },
```

здесь sectionContent – контент нужной главы в формате html; position – последняя позиция чтения; section – номер главы.

7. Приложение для чтения книг формирует страницы из полученного контента.

8. По завершению приложение отображает пользователю свой интерфейс, представленный на рисунке 11.

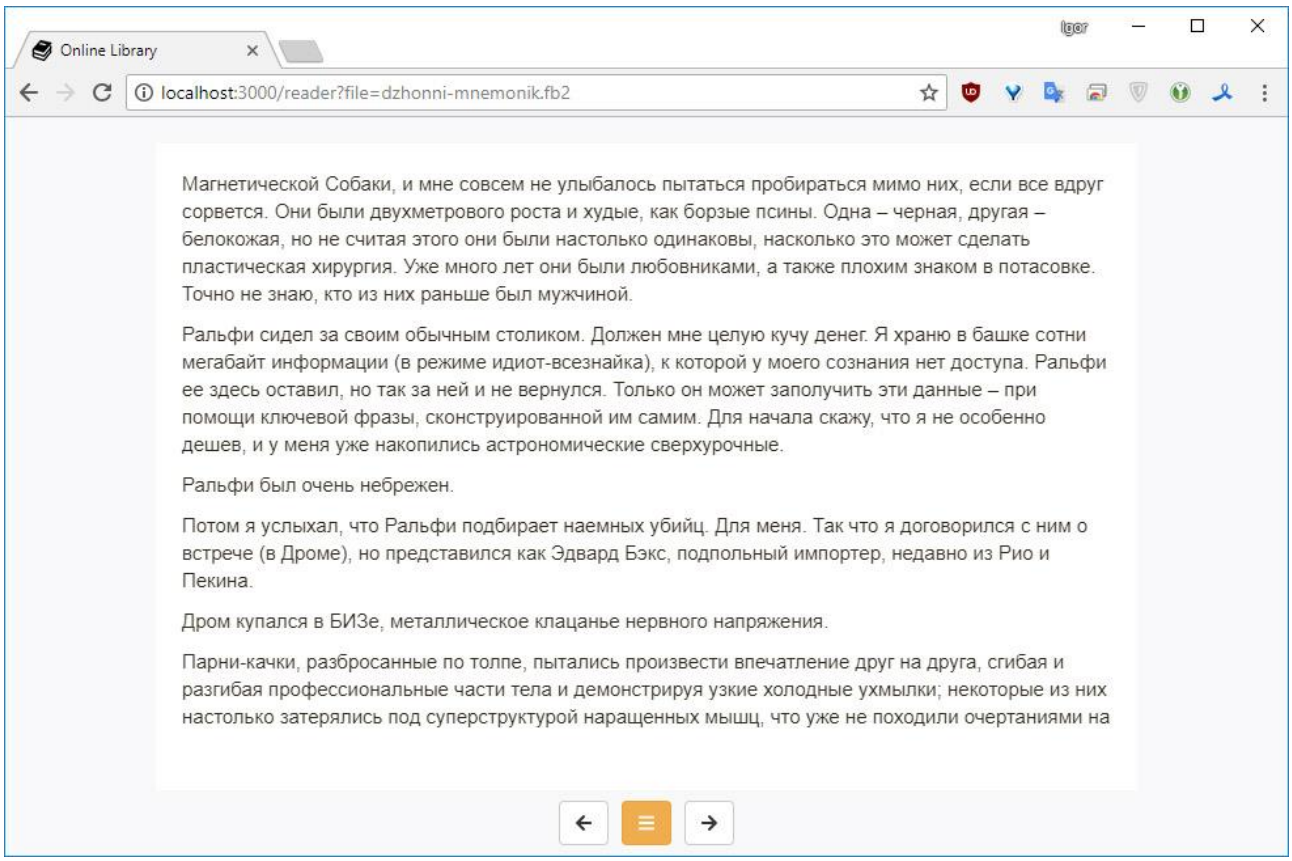


Рисунок 11 – Пример страницы приложения для чтения книги, открытой на ПК

Во время выполнения шагов 1-7 пользователь видит загрузочный экран, представленный на рисунке 12.

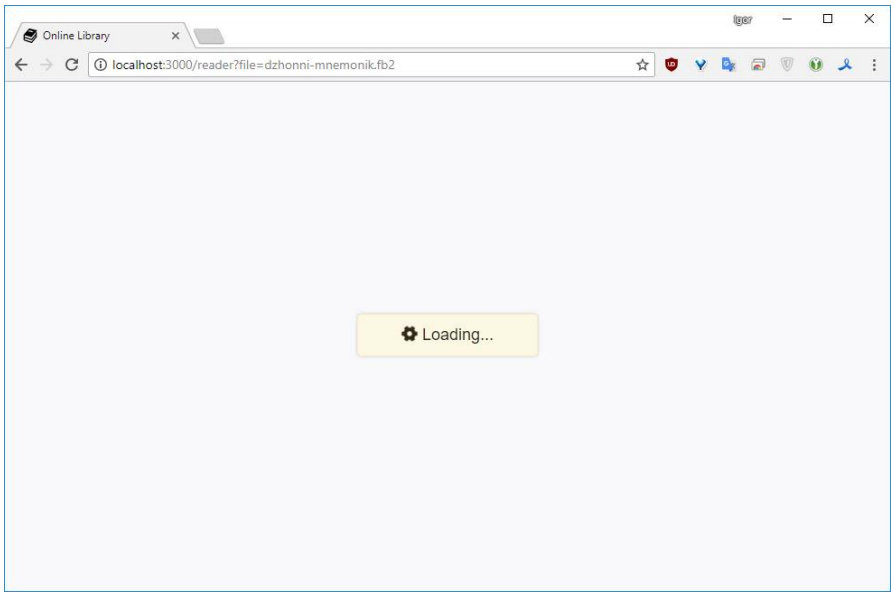


Рисунок 12 – Пример страницы загрузки приложения для чтения книги, открытой на ПК

3.4.1 Алгоритм разбивки контента на страницы

Для того чтобы отображать контент по страницам реализован следующий алгоритм:

1. Контент разбивается на слова и помещается в массив.
2. Из этого массива извлекается некоторое количество слов, в зависимости от высоты страницы, либо 500, либо 200 (500 – для устройств с большим экраном, 200 для мобильных устройств).
3. Полученные на предыдущем шаге слова помещаются в html-элемент div с установленным стилем visibility: hidden, тем самым контент внутри div’a не виден пользователю, но браузер будет его отрисовывать, что позволяет получить высоту контента.
4. Если высота контента больше высоты, заданной для области чтения, то удаляется одно слово.
5. Шаги 3-4 повторяется до тех пор, пока высота не станет равной или меньше высоты области чтения. Как только это произошло, происходит переход на шаг 8.
6. Если высота контента меньше высоты области чтения, то добавляется одно слово.
7. Шаги 3 и 6 повторяются до тех пор, пока высота контента не станет равной высоте области чтения. После этого происходит переход на шаг 8.
8. На данном шаге получается одна, полностью сформированная страница, которая помещается в массив страниц.
9. Возврат на шаг 2 до тех пор, пока массив слов не опустеет.

По завершению работы алгоритма получается массив, содержащий сформированные страницы.

3.4.2 Синхронизация позиции чтения

Синхронизация последней прочитанной позиции достигается следующим образом: в базе данных для каждой книги хранится информация о последней главе, которую читал пользователь, а также о количестве прочитанных слов в этой главе.

Эта информация передается на сервер в JSON-объекте с помощью AJAX, каждый раз, когда пользователь перелистывает страницу:

```
{
        position: position,
        section: section
 },
```

где position – количество слов; section – номер главы.

Информация из переданного JSON-объекта сохраняется в базу данных.

После того как все страницы сформированы, приложение вычисляет страницу, на которой находится последнее прочитанное пользователем слово, и устанавливает эту страницу в качестве текущей читаемой.

Заключение

Целью данной работы являлась разработка веб-приложения, которое позволило бы осуществлять хранение и чтение электронных книг в формате fb2 на различных устройствах.

В ходе данной работы выполнен анализ существующих решений, выявлены требования к разрабатываемому приложения, создана база данных, спроектирована архитектура приложения, проведена работа по визуальному оформлению.

Результатом проделанной работы стало веб-приложение, в котором реализованы необходимые инструменты для просмотра и организации хранения электронных книг в формате fb2.

В настоящее время заканчивается доработка приложения, производится тестирование. Планируется публикация приложения в сети Интернет на open-source основе.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Аквино К., Ганди Т. Front-end. Клиентская разработка для профессионалов. Node.js, ES6, REST : пер. с англ./ Аквино К., Ганди Т. — СПб.: Питер, 2017. — 512ст.
2. Браун И. Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript : пер. с англ./ Браун И. — СПб : Питер, 2017. — 336ст.
3. Гамма Э. Приемы объектно-ориентированного программирования. Паттерны проектирования: пер. с англ./ Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес. – СПБ : Питер, 2010. – 368ст.
4. Моделирование данных в MongoDB [Электронный ресурс] // URL: https://habr.com/post/144798/ (дата последнего обращения: 5.04.2018).
5. Описание Fictionbook [Электронный ресурс] // URL: http://fictionbook.org:80/index.php/Описание_Fictionbook (дата последнего обращения: 6.03.2018)
6. Пауэрс Ш. Изучаем Node : пер. с англ./ Ш. Пауэрс – 2-е изд. – СПБ. : Питер, 2017. – 304ст.
7. Справочник по API Express [Электронный ресурс] // URL: http://expressjs.com/ru/4x/api.html (дата последнего обращения: 2.05.2018).
8. Bootstrap components [Электронный ресурс] // URL: https://getbootstrap.com/docs/3.3/components/ (дата последнего обращения: 15.04.2018).
9. Google Play Книги [Электронный ресурс] // URL: https://play.google.com/books (дата последнего обращения: 22.04.2018).
10. MongoDB Docs [Электронный ресурс] // URL: https://docs.mongodb.com (дата последнего обращения: 7.02.2018).

11. Node.js Documentation [Электронный ресурс] // URL: https://nodejs.org/dist/latest-v8.x/docs/api/ (дата последнего обращения: 17.02.2018).
12. Node.js MongoDB Driver API [Электронный ресурс] // URL: https://mongodb.github.io/node-mongodb-native/2.2/api/index.html (дата последнего обращения: 10.05.2018).
13. Readium-js-viewer [Электронный ресурс] // URL: https://github.com/readium/readium-js-viewer (дата последнего обращения: 23.04.2018).

Приложение А. Руководство пользователя

1. Системные требования

Для работы серверной части приложения на сервер должны быть установлено следующее программное обеспечение:

1. MongoDB версии не ниже 3.6.
2. Node.js версии не ниже 8.

Для использования веб-приложения необходима рабочая станция либо мобильное устройство под управление ОС Android или IOS с наличием Интернет-соединения со скоростью не менее 256 килобит в секунду и установленным одним из веб-браузеров:

- Google Chrome 56 и выше.
- Microsoft Edge 13 и выше.
- Apple Safari 10 и выше.
- Mozilla Firefox 52 и выше.
- Opera 48 и выше.

Для корректной работы сайта не требуется установка дополнительных приложений или плагинов.

2. Поиск и добавление книги

Поиск книг осуществляется на главной странице приложения по средствам выбора интересующего тега, после чего произойдет переход страницу, содержащую книги, которым присвоен данный тег.

Добавление новой книги происходит с помощью нажатия на кнопку “+” расположенную на главной странице, затем пользователю откроется диалоговое окно с предложением выбрать файл электронной книги в формате fb2. Пример данного диалога представлен на рисунке А.1.

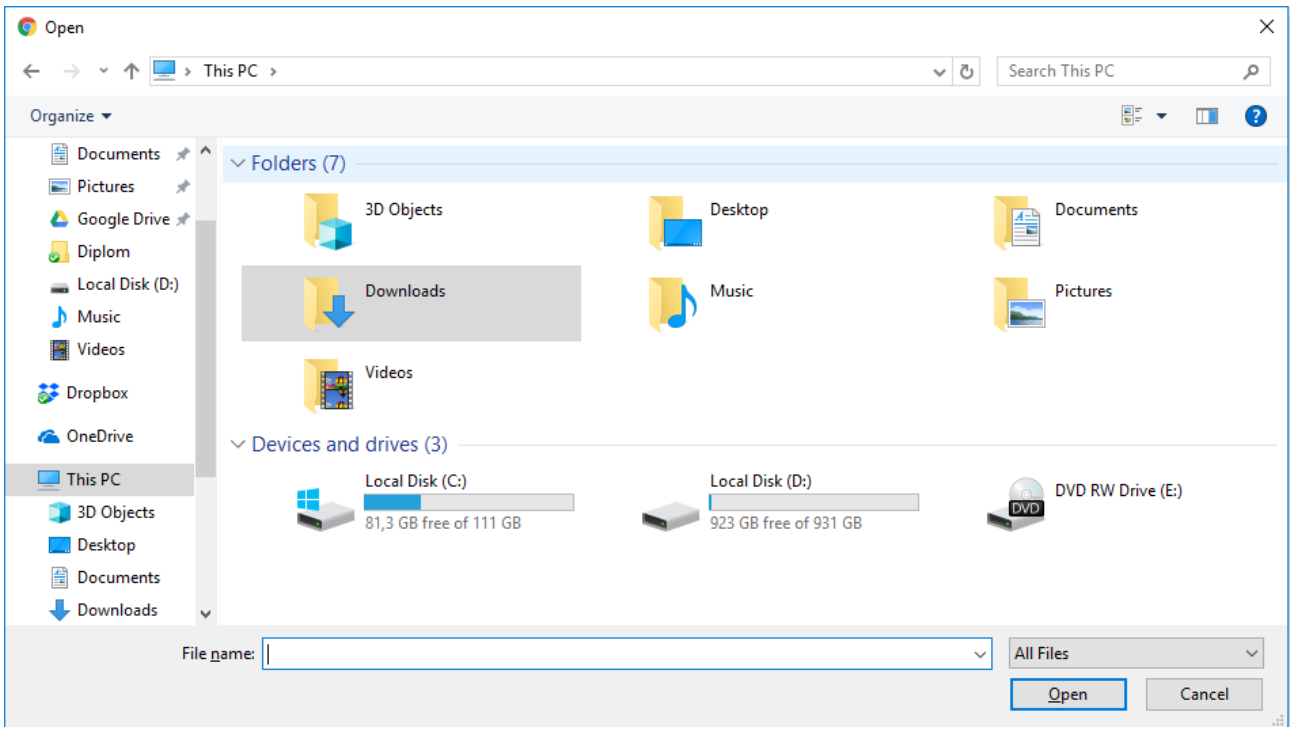


Рисунок А.1 – Пример диалога выбора файла, открытого на ПК под управлением ОС Windows 10

После загрузки книги, произойдет переход на страницу с информацией о книге, на которой пользователь может:

- Добавить теги для данной книги.
- Удалить книгу.
- Скачать на свое устройство.
- Читать книгу.

3. Чтение книги

При нажатии на кнопку “читать” произойдет переход на страницу, содержащую приложение для чтения книг. Если пользователь ранее уже открывал выбранную книгу, то он увидит предыдущую позицию, на которой закончил чтение.

Навигация по страницам осуществляется с помощью кнопок “вперед” и “назад”. При нажатии на среднюю кнопку откроется оглавление. На рисунке А.2 представлена панель навигации приложения для чтения.



Рисунок А.2 – Панель навигации приложения для чтения книг.

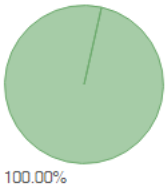
Обработан файл:
VechkanovDiplom.docx.

Год публикации: 2018.

Оценка оригинальности документа - 100.0%

Процент условно корректных заимствований - 0.0%

Процент некорректных заимствований - 0.0%



Просмотр заимствований в документе

Время выполнения: 18 с.

Заимствования отсутствуют