МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра физики открытых систем
наименование кафедры

Разработка базы данных публикационной активности
наименование темы выпускной квалификационной работы полужирным шрифтом
сотрудников СГУ и интерфейса к ней

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 4041 группы

направления 09.03.02 «Информационные системы и технологии»
код и наименование направления
Института физики
наименование факультета
Фальяна Георгия Альбертовича
фамилия, имя, отчество

Научный руководитель

профессор, д.ф.-м.н, доцент		Москаленко О.И.
должность, ученая степень, уч. звание	подпись, дата	Инициалы Фамилия

Зав. кафедрой физики открытых систем
полное наименование кафедры

д.ф.-м.н., профессор		Короновский А. А.
должность, ученая степень, уч. звание	подпись, дата	Инициалы Фамилия

Саратов 2022 г.

Введение

Базы данных являются важной частью практически любых программных продуктов. От правильности их реализации зависит то, как быстро, эффективно и безопасно происходит обмен информацией с приложением. Надежное хранилище данных крайне необходимо, особенно, когда речь идет о хранении банковских или пользовательских сведений. В связи с этим очевидно, что специалисты по разработке баз данных имеют высокий спрос на рынке труда. В рамках настоящей бакалаврской работы пройден полный процесс разработки такой базы данных.

В работе рассмотрены все этапы создания веб-приложения, содержащего информацию о публикационной активности научного сообщества Саратовского государственного университета имени Н. Г. Чернышевского за последние несколько лет. Базу данных журналов, статей и научных сотрудников, участвовавших в их написании, предоставляют ресурсы Web of Science и Scientific Journal Rankings (SJR). В качестве системы управления базами данных (в дальнейшем СУБД) использовался MongoDB – один из классических примеров NoSQL-систем, который использует JSON-подобные документы и схему базы данных. Помимо всего, в работе созданы несколько скриптов, которые считывают и делают выборку необходимых сведений из файлов, предоставляемых указанными выше ресурсами. Также в ходе выполнения работы написан сценарий автозаполнения базы данных. Для написания серверной части приложения, а также построения REST API применялась программная платформа Node.js совместно с фреймворком Express, позволяющая использовать язык программирования JavaScript в качестве серверного. Построение клиентской части производилось посредством JavaScript-фреймворка Vue.js. Все упомянутые технологии описаны в теоретической части работы.

Для ускорения и упрощения написания приложения использовалось еще несколько инструментов, наиболее часто применяемых в веб-разработке, которые также подробно рассмотрены в работе.

1. Применяемые программы и технологии

В данном проекте применялся так называемый MEVN стек технологий (MongoDB, Express, Vue.js, Node.js), используемый для полного цикла создания веб-приложения (full-stack). Также стоит отметить, что в разработке применялось лицензионное и предоставленное для бесплатного пользования программное обеспечение.

JavaScript (JS) – мультипарадигменный язык программирования, поддерживающий объектно-ориентированный, императивный и функциональный стили. JS является реализацией спецификации ECMAScript. Язык «безопасен» и не предоставляет низкоуровневый доступ к памяти или процессору, поскольку изначально был создан для браузеров, которые не требовали такого функционала.

Node или **Node.js –** программная платформа, превращающая JavaScript из узкоспециализированного языка в язык общего назначения. В основе этой технологии лежит движок V8, применяемый в браузере Google Chrome. V8 компилирует исходный код программы, написанной на JavaScript, непосредственно в машинный код, минуя стадию промежуточного байт-кода. Важнейшая особенность Node.js, которая сделала его незаменимым в веб-разработке, – это его менеджер пакетов NPM (Node Package Manager).

NPM (Node Package Manager) существенно облегчил современную веб-разработку за счет возможности установки огромного количества пакетов, предоставляющих различный инструментарий. К ним можно отнести фреймворки, библиотеки, сборщики проектов и многие другие инструменты, активно применяемые программистами. Кроме того, имеется возможность написания своих собственных модулей, которыми можно также делиться с другими разработчиками.

Также в процессе работы использовалась интегрированная среда разработки (IDE) **Visual Studio Code** – один из самых популярных редакторов кода. Он бесплатный и предоставляет множество различных плагинов, ускоряющих процесс написания кода и позволяющих настраивать окружение под любые нужды. В выборе редактора нет особой закономерности. Здесь дело касается только личного удобства.

MongoDB – документоориентированная система управления базами данных, не требующая описания схемы таблиц. Применяется в веб-разработке, в частности, в рамках JavaScript-ориентированного стека MEVN. MongoDB имеет структуру, состоящую из коллекций и документов. Каждая коллекция в такой базе данных – это сущность, представленная в виде некоторого числа документов. Документ же, в свою очередь, являться JSON-подобным объектом. Такой тип организации информации очень удобен и, поэтому, популярен не только в WEB'е, но и в других сферах разработки программных продуктов

Express – это простой и гибкий веб-фреймворк для приложений на Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений. Фреймворк предоставляет множество методов, позволяющих просто и быстро «поднять» локальный веб-сервер, настраивать роутинг (навигацию) сайта, а также обрабатывать HTTP запросы.

Vue.js – JavaScript-фреймворк с открытым исходным кодом для создания пользовательских веб-интерфейсов. Он легко интегрируется в проекты с использованием других JavaScript-библиотек, а также может функционировать как веб-фреймворк для разработки одностраничных приложений в реактивном стиле. Vue.js использует компонентный подход к разработке интерфейсов. Фреймворк имеет большую экосистему и дополнительный инструментарий, разработанный создателями Vue.js.

Написание стилей на классическом CSS является несколько рутинным занятием. По этой причине разработчики прибегают к использованию различных препроцессоров. В данной работе был применен препроцессор **SASS/SCSS**.

Bootstrap Vue – фреймворк уже готовых компонентов для Vue.js. Здесь все просто. После подключения, в проект можно добавлять различные компоненты из библиотеки. По умолчанию они уже будут иметь определенные стили, которые по необходимости можно переписывать под стилистику своего проекта.

2. Основы взаимодействия клиент-сервер

В основе любого современного веб-ресурса лежит сложная архитектура клиент-серверного взаимодействия. Он может состоять из различных сервисов и прочих составляющих, которые имеют непростую схему взаимодействия. Для того, чтобы понять базовый механизм в полной мере, необходимо немного окунуться в историю развития всемирной сети. Еще 10-20 лет назад типичный веб-сайт представлял из себя несколько HTML страниц, у которых есть один или несколько файлов стилей и JavaScript-сценариев, добавляющих несложный функционал или анимацию. Каждый раз, когда пользователь вводит в адресную строку браузера URL сайта, сервер в случае успешного выполнения запроса возвращает готовый HTML файл, стили, скрипты, изображения и т.д. После этого, браузер отрисовывает страницу, и пользователь, наконец, может увидеть результат. Как можно заметить, передавать все эти файлы клиенту при каждом переходе пользователя между страницами сайта слишком ресурсоемкая задача, хотя такая схема применяется уже долгое время. Более того, если речь идет о больших проектах с тысячами пользователей, такой подход будет слишком сильно нагружать сервер, что однозначно, учитывая трафик, приведет к «падению» сервера.

На смену устаревшему и низкоэффективному подходу к взаимодействию клиент-сервер пришел REST API. Этот архитектурный стиль подразумевает, что пользователь получает HTML файл только один раз при первом заходе на веб-ресурс, а при всех последующих HTTP запросах сервер возвращает уже не готовые файлы, а данные (например, JSON или XML). Ориентируясь по ним, клиентская часть приложения динамически с помощью JavaScript отрисовывает интерфейс. Это решение позволяет существенно повысить скорость загрузки сайтов и производительность. По большому счету, **REST** (от англ. Representational State Transfer — «передача репрезентативного состояния») – это набор правил того, как программисту необходимо организовать написание кода серверного приложения,

чтобы все системы легко обменивались данными и приложение можно было масштабировать.

Коротко остановимся на том, как функционирует протокол HTTP. **HTTP** (англ. HyperText Transfer Protocol — «протокол передачи гипертекста») – это протокол прикладного уровня передачи данных (сетевая модель TCP/IP), изначально – в виде гипертекстовых документов в формате HTML, сейчас же используется для передачи произвольных данных. В настоящий момент большинство веб-ресурсов взаимодействуют по его расширению HTTPS, добавляющему поддержку шифрования и повышающему безопасность взаимодействия клиент-сервер. Протокол имеет несколько методов. Среди них самыми часто используемыми являются GET, POST, DELETE, PUT для получения, отправки, удаления и обновления содержимого на сервере, соответственно. Каждый запрос имеет тело отправки **request** и ответа **response.** Вместе с ответом сервер также возвращает код состояния, указывающий на то, успешно ли был произведен запрос или же что-то пошло не так.

3. Архитектура приложения

На рисунке 1 представлена простая схема устройства приложения. Как уже говорилось, интерфейс написан на JavaScript фреймворке Vue с использованием готовых компонентов Bootstrap Vue. Сервер же разработан с применением фреймворка Express и предоставляет REST API, с помощью которого клиент может получить доступ к данным. Клиентская и серверная части в процессе разработки одновременно работают на локальном сервере с разными портами.

При обращении пользователя к серверу последний обрабатывает соответствующий маршрут (url адрес) и, по необходимости, выполняет запрос в базу данных MongoDB с определенными параметрами (если они были указаны в теле запроса). В случае, если произошло исключение, в теле ответа придет текст ошибки. Если же запрос прошел успешно, сервер обрабатывает информацию, полученную из БД, и отправляет пользователю. В любом случае вместе с телом ответа в браузер приходит код состояния запроса.

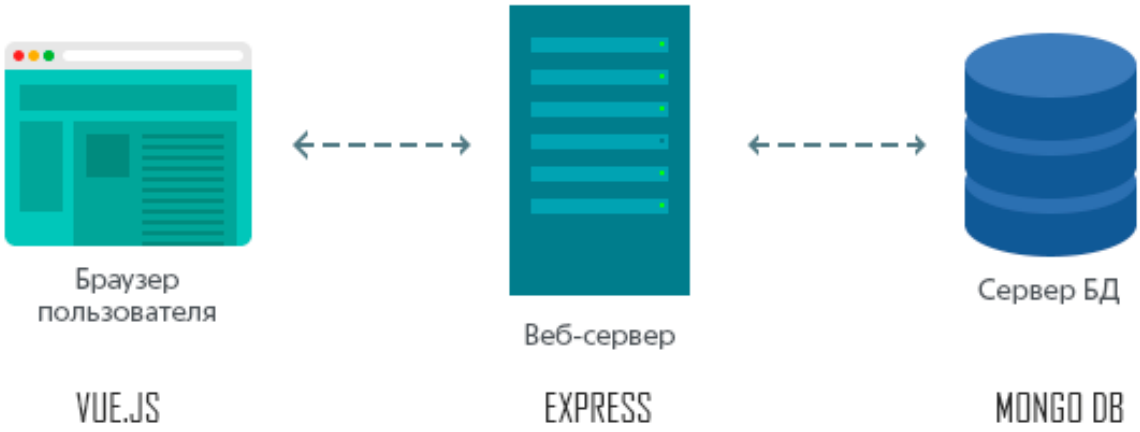

Рисунок 1 – Схема приложения

4. Реализация парсера данных

Работа начинается с реализации скриптов, которые будут считывать, анализировать и делать выборку данных. С БД Web of Science был импортирован файл в формате HTML, который хранит в себе все сведения о научных публикациях сотрудников Саратовского государственного университета. Для того чтобы считать информацию из него, необходимо проанализировать DOM дерево документа. Так как реализация скрипта производилась с использованием Node.js (классический JavaScript лишен возможности оперировать с файловой системой, поскольку работает только в браузере) и у него отсутствуют стандартные методы работы с DOM, через NPM была установлена библиотека **jsdom**, добавляющая методы для операций с деревом HTML элементов. Каждая отдельная статья в документе представляла собой таблицу (тег <table>). Поэтому вся задача заключалась, в том, чтобы обойти все таблицы в файле и считать с них данные в отдельные JS объекты. После того как этот этап был выполнен с ресурса Scientific Journal Rankings был импортирован файл Excel, содержащий дополнительные сведения о журналах, в которых были опубликованы научные работы. Конкретно в HTML документе отсутствовала информация об импакт-факторах журналов,

которые нужны для расчета рейтинга статей. Чтобы можно было считывать данные с файла в формате .xls, была установлена библиотека **csv-parser**.

Для дополнения считанных с HTML данных информацией о журналах был применен следующий алгоритм: сначала перебиралась каждая строка Excel-файла с журналами и, если название журнала, ISSN или eISSN совпадали с соответствующими полями в объекте, содержащим отдельную статью, то последнему дописывались сведения из этой строки. Финальный результат был преобразован и сохранен в виде JSON файла.

5. Сценарий автозаполнения базы данных

Последним этапом подготовки данных к разработке приложения было написание скрипта автозаполнения базы данных MongoDB. Но перед этим необходимо сначала зарегистрироваться на официальном сайте СУБД и создать непосредственно саму БД. В процессе создания, нужно придумать название, пароль и выбрать ближайший регион с облачным хранилищем. После этого пользователь может подключиться к БД через URL, сгенерированный при создании.

Далее был создан новый JS файл. Для начала, внутри него был открыт в режиме чтения сгенерированный после выполнения работы парсера JSON файл с информацией о статьях. Структура БД представляет собой 3 коллекции: для статей (Articles), авторов (Authors) и журналов (Journals). Чтобы создать эти сущности, импортировалась библиотека Mongoose. Mogoose – довольно мощная библиотека, имеющая множество инструментов для работы с MongoDB. Затем, с помощью предоставленных библиотекой методов и сгенерированного URL было реализовано подключение к БД.

Алгоритм заполнения базы данных прост. Было создано три функции, которые перебирают объекты в массиве данных, сформированных при парсинге, параллельно делая выборку и формируя документ (JSON объект) для последующего сохранения в БД.

Вместе с выборкой данных для отправки в БД функции также рассчитывали рейтинг. Для коллекции статей (Articles) он рассчитывался по следующей формуле.

$$Q = \frac{A(1+m)}{n} \quad (1)$$

где Q – суммарный балл за статью, A – балловая оценка статьи (для журнала, входящего в Web of Science A=30), m – импакт-фактор журнала (взят из базы данных SJR), n – число соавторов. Что касается рейтинга отдельного автора, то он представляет собой сумму рейтингов статей, в которых данный научный сотрудник значится соавтором.

Этот этап является финальным в подготовке данных для приложения. Если понадобится дополнить БД свежей информацией, то в директории проекта достаточно заменить HTML файл с данными и заново запустить описанные выше скрипты.

6. Реализация серверной части приложения

Как уже говорилось ранее, сервер реализован с помощью фреймворка Express. Вся разработка сервера сводится к тому, чтобы написать интерфейс, благодаря которому клиент сможет запрашивать сведения с БД и, в случае успеха, получать обработанную и, по необходимости, отсортированную информацию. Внутри фреймворка уже имеются методы обработки HTTP запросов.

В главном файле осуществляется подключение к базе данных и создается экземпляр приложения Express. Далее в этом файле, перед кодом подключения к БД, указываются все промежуточные обработчики. Функции промежуточной обработки (middleware) – это функции, имеющие доступ к объекту запроса (req), объекту ответа (res) и к следующей функции промежуточной обработки в цикле “запрос-ответ”. Они необходимы, чтобы при каждом HTTP запросе проводить с этими объектами некоторые операции или получать определенные параметры. Таких промежуточных обработчиков может быть сколь угодно много. Разумеется, программист может писать свои обработчики. Написанный самостоятельно обработчик способен проводить операции с определенным маршрутом, на который идет HTTP-запрос. Так, в данном приложении имеется три базовых пути (routes),

приписываемые к корневому адресу (в нашем случае локальный сервер http://localhost:5000):

1. /api/articles
2. /api/journals
3. /api/authors

Функции-обработчики могут отслеживать один и тот же маршрут, но разные методы запроса. Например, одна функция работает с методом DELETE, другая – с POST. Далее, внутри функции берутся входные данные из объекта req и запрашиваются данные с БД.

В целом, вся логика состоит в том, чтобы просто отслеживать определенный маршрут, и, если метод и путь запроса соответствуют обработчику, вызывать установленную функцию, которая будет извлекать данные из объекта req, фиксировать ошибки при их наличии, запрашивать и обрабатывать данные сервера и, наконец, возвращать их обратно клиенту в объекте response вместе со статус-кодом.

7. Реализация клиентской части веб-приложения

Клиент веб-приложения, как и любое другое современное ПО, реализуется с использованием большого количества различных фреймворков и библиотек. В частности, они могу быть предназначены для быстрой развертки локального development сервера, оптимизации JavaScript кода, преобразования стилей, написанных с помощью препроцессоров, в CSS, сжатия изображений, трансформации форматов шрифтов и т.д. Чтобы все эти инструменты исполняли свои функции автоматически, используются сборщики проектов. **Webpack** – это один из самых популярных сборщиков модулей JavaScript с открытым исходным кодом. Он создан, в первую очередь, для JavaScript, но может преобразовывать внешние ресурсы, такие как HTML, CSS и изображения, если включены соответствующие загрузчики. Webpack принимает модули с зависимостями и генерирует статические ресурсы, представляющие эти модули.

Webpack требует настройки, но есть уже готовые популярные реализации. Так, экосистема Vue предоставляет разработчикам продукт Vue-CLI. Это уже подготовленный к началу работы проект Vue.js с некоторыми предустановленными библиотеками и настроенным сборщиком, который может быть подкорректирован под определенные нужды.

HTTP запросы реализованы с помощью библиотеки **Axios**, предоставляющей более удобные функции для взаимодействия с api. Передавая необходимые параметры в запрос, сервер сможет корректно выполнить обработку соответствующего пути и, в случае успешного завершения операции, вернуть необходимые данные клиенту.

Теперь речь пойдет про навигацию по сайту. Как упоминалось ранее, ресурс предоставляет сведения в виде отдельных таблиц статей, авторов и журналов. К этому еще добавляется главная страница сайта, а также адреса для каждого отдельного автора и статьи. Например, на странице автора можно узнать динамику изменения его рейтинга в течение последних нескольких лет, а также получить ссылки на публикации с его участием. Внешний вид веб-сайта можно посмотреть в разделе «Результаты». Для того чтобы настроить навигацию, использовался еще один инструмент из экосистемы Vue – **Vue-router**. Данная библиотека предоставляет функционал для настройки переключения между vue-компонентами визуализации.

На текущем этапе разработку клиента можно считать завершенной. Имеется несколько страниц визуализации интерфейса, маршрутизация и подключение к серверу. Данный проект, разумеется, можно масштабировать, добавляя различные страницы и функционал, а также оптимизировать написанный код. Тем не менее, была решена основная задача, а именно реализация удобного интерфейса для представления базы данных статей. Далее последует финальный раздел с представлением результатов работы.

Заключение

В рамках бакалаврской работы разработано полноценное веб-приложение, позволяющее проанализировать публикационную активность сотрудников Саратовского государственного университета за несколько последних лет работы. Оно представляет собой базу данных научных публикаций сотрудников университета, а в качестве основных инструментов для реализации проекта используется популярный в среде веб-разработки стек технологий MEVN (MongoDM, Express, Vue.js, Node.js). В процессе выполнения работы реализованы сценарии считывания и обработки исходного источника данных, импортированного с интернет ресурса Web Of Science, а также скрипт автозаполнения БД MongoDB. В процессе затрагивалась тема архитектуры клиент-серверного взаимодействия и основы протокола HTTP, знание которого необходимо для понимания работы веб-приложений.

Функционал разработанного приложения позволяет производить поиск по любому конкретному автору, журналу или статье. Полученные результаты можно отсортировать по нескольким различным критериям. Важно отметить, что каждый автор или статья имеют собственную страницу с более подробной информацией. В частности, благодаря используемой рейтинговой системе оценки качества публикаций можно узнать, насколько часто фамилия сотрудника фигурирует в списке авторов публикаций.

Проект может быть размещен на хостинге для того, чтобы любой пользователь мог иметь к нему доступ через поисковые системы.