Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра физики открытых систем

Разработка веб-приложения по сетевой модели клиент-сервер

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студента 4 курса 431 группы
направления 09.03.02 «Информационные системы и технологии»
факультета нелинейных процессов
Отпущенникова Константина Сергеевича

Научный руководитель

д.ф.-м.н., доцент		С.А. Куркин
должность, уч. степень, уч. звание	дата, подпись	инициалы, фамилия

Заведующий кафедрой

д.ф.-м.н., профессов		А.А. Короновский
должность, уч. степень, уч. звание	дата, подпись	инициалы, фамилия

Саратов 2018г.

ВВЕДЕНИЕ

Целью данной работы является разработка веб-приложения по сетевой модели клиент-сервер. Приложение будет представлять из себя социальную сеть с широким функционалом.

На сегодняшний день сетевые технологии являются одной из самых актуальных и перспективных направлений в нашем мире. Но, к сожалению, в данной сфере существует множество нерешенных актуальных проблем. В данной работе будут решены некоторые из них, а именно: низкий пользовательский опыт UX (user experience), нерациональное использование сетевого трафика.

Для решения проблемы с UX будет разработан совершенно новый подход маршрутизации (routing), который будет производиться на клиентской стороне, что в корне отличается от традиционного подхода, когда маршрутизация осуществлялась на серверной стороне, что требовало дополнительных нагрузок на сервер для рендеринга HTML, а также существенно снижало пользовательский опыт UX.

Для решения проблемы с нерациональным использованием сетевого трафика будет разработана система кеширования и проксирования веб-запросов и статических ресурсов. В разрабатываемую систему будут внедрены новые технологии, например, фоновые потоки ServiceWorkers.

В качестве результата будет получено приложение с решенными вышеперечисленными актуальными проблемами со следующим функционалом: создание новых пользователей, создание и редактирование текстовых постов и изображений, подписка на других пользователей, новостная лента, лайки и комментарии, редактирование профиля пользователя.

Работа структурирована и содержит последовательно изложенную информацию, заключенную в главы. Основных заголовков два. Они

описывают клиентскую и серверную часть веб-приложения. Каждый заголовок содержит подзаголовки, которые раскрывают тему полностью.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Основная часть имеет два основных заголовка, отвечающих за описание «Клиентской стороны» и «Серверной стороны» разрабатываемого веб-приложения.

1 Клиентская сторона

1.1 Компоненты и их отрисовка

Строительными блоками веб-приложения будут являются компоненты. Компонент – это функция, либо класс, у которого есть метод render. Метод render или функция должны вернуть разметку в формате JSX (JavaScript Syntax Extension) – это особый формат, для написания HTML-подобного кода прямо в JavaScript, который в конечном итоге рендерит HTML-разметку. Компонент получает через свойство класса props из родительского компонента или Store данные, которые требуются для отображения компонента. Также компонент содержит в себе (как правило, если это класс) функционал, который отвечает за пользовательские сценарии, присущему соответствующему компоненту.

1.2 Хранилище сотояния приложения

Store, или по-другому хранилище – это, наряду с компонентами, одна из ключевых концепций React – библиотека пользовательских интерфейсов. Хранилище предназначено для трех главных вещей: хранение данных, распределение (фильтрация) данных по компонентам, обновление данных путем обработки пользовательских сценариев. Современные веб-приложения разрабатываются по принципу state-base (базируются на состоянии). Состояние программируется и отвечает за рендеринг как отдельно взятых компонентов, так и всего приложения в целом. Также хранилище напрямую общается с серверной частью (backend) нашего приложения посредством Ajax. Также важно заметить, что хранилище является неким состоянием нашего

приложения и «единственным источником правды» с точки зрения клиентской стороны.

1.3 Маршрутизация и веб-запросы

Маршрутизация на клиенте (роутинг) – это концептуально новый подход к маршрутизации в веб-сайтах и приложениях. Роутинг приложения на клиентской стороне позволяет улучшить пользовательский опыт (UX – user experience). Еще несколько лет назад, традиционно, мы использовали серверный роутинг. Такой подход работает, но он плох тем, что при отправке серверного запроса, сервер генерировал готовую HTML страницу и отправлял ее клиенту, в то время как браузер, получая эту самую HTML страницу, полностью обновлял свое окно (перезагружал страницу). Все изменилось с появлением Ajax (Asynchronous JavaScript and XML). Ajax подход отличается от традиционного тем, что мы делаем запросы на данные, а не на готовую, отрендереную на сервере, страницу. В результате получаем данные, которые, применяя на клиенте, позволяют динамически изменить страницу без эффекта перезагрузки. Такой подход создает для конечного пользователя эффект плавности и непрерывности, свойственные нативным приложениям.

1.4 Технология Service Workers

В данном разделе описывается технология минимизации сетевого трафика (одна из актуальных проблем). В данном разделе рассматривается кеширование статического контента приложения, а также проксирование веб-запросов.

1.5 Структура проекта клиентской стороны

Пятый подраздел описывает базовую структуру проекта клиентской стороны. В проекте задана следующая структура файлов и каталогов:

- public – файлы, которые доступны браузеру через URL стандартным GET запросом
- src – файлы с исходным кодом (суммарно около ~5000 строк кода)
- actions – обработка пользовательских сценариев и хранилища (~2000 строк кода)
- components – компоненты интерфейса (~300 строк кода)
- reducers – обработчики, предназначенные непосредственно для обновления Store (~800 строк кода)
- sass – дизайнерское оформление (~500 строк кода)
- services – служебные функции и классы (~600 строк кода)
- store ~ хранилище
- app.jsx – корневой файл приложения
- router.jsx – маршрутизатор

Вид структуры проекта можно посмотреть на рисунке 1.

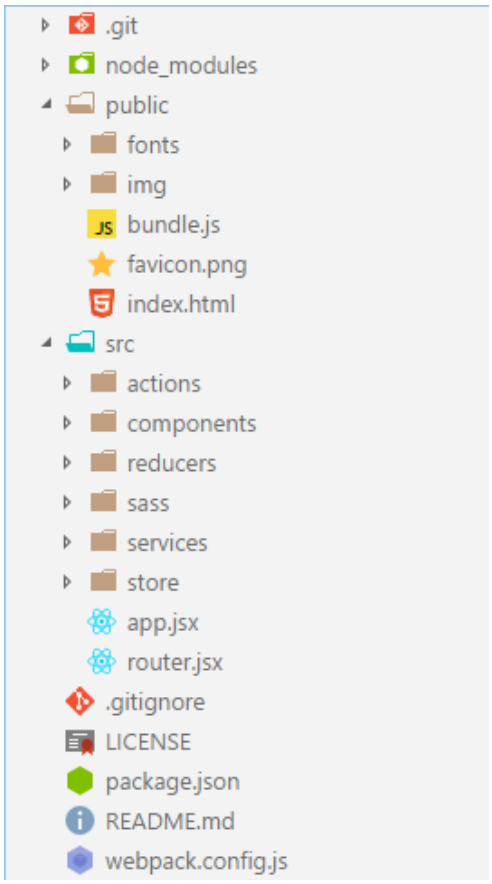

Рисунок 1 – структура проекта серверной стороны

2 Серверная сторона

2.1 Язык программирования NodeJS

На серверной стороне используется NodeJS. JavaScript используется в различных программных средах, одна из которых является NodeJS. NodeJS – это интерпретируемый язык программирования, который может использоваться где угодно, главное чтобы был установлен интерпретатор. От веб-сервера, до холодильников и микроконтроллеров Arduino.

NodeJS (часто просто Node) – это программная платформа, основанная на движке V8 (транслируещем JavaScript в машинный код, разработан командой google), превращающая JavaScript из узкоспециализированного языка сценариев в язык общего назначения. Node добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызов к ним из JavaScript кода. Node применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node и десктопные оконные приложения, например, с помощью фреймворка Electron, и даже программировать микроконтроллеры.

В основе Node лежит событийно-ориентированное и асинхронное программирование с неблокирующими операциями ввода-вывода (I\O). Это означает что, если мы ждем какой-либо ввод от клиента на сервер, мы можем продолжать слушать запросы и обрабатывать их. Эта идея отличает Node от других языков программирования, делая его на быстрее конкурентов.

2.2 База данных MongoDB

Для того, чтобы хранить данные нам также потребуется база данных, мы будем использовать MongoDB.

MongoDB – это документно-ориентированная база данных, с открытым исходным кодом. Классифицирована как NoSQL (Not only SQL), использует специальный JSON подобный формат хранения данных, который называется BSON (Binary JSON). Mongo написана на языке C++, C и JavaScript.

В отличие от хорошо знакомых многим SQL баз данных, Mongo отличается от последних тем, что хранить данные не в виде таблиц, а в виде документов. Популярные виды связи (один к одному, один ко многим и многие ко многим) реализуются посредством связывания коллекций документов. Коллекция – это набор однотипных документов, например, пользователей или постов пользователей. MongoDB имеет следующие достоинства по сравнению с тем же, например, SQL:

- Имеет распределенный доступ к данным, расположенных на нескольких серверах;
- Возможно параллельное извлечение данных;
- Более быстрое извлечение простых структур данных;
- Может хранить неструктурную информацию;
- Гибкость.

2.3 Программирование веб-ответов. REST API

Для обработки клиентских запросов реализуется REST API – это архитектура построения веб-сервера, которая четко регламентирует http-глаголы и http-коды для той или иной задачи обработки запроса [10]. Давайте посмотрим какие правила нам задает этот стиль в нашем приложении, при использовании http-глаголов:

- PUT – полное обновление
- PATCH – частичное обновление
- GET – получение
- POST – добавление
- DELETE – удаление

В плане http кодов в целом всё достаточно просто, главное следовать здравому смыслу. Успешная запись – 200, ошибка сервера – 500, перенаправление – 300 и т.п. Полный список http кодов и за что они отвечают смотрите в разделе «Ссылки».

2.4 Контроллеры

Четвертый раздел описывает функции обратного вызова, ассоциированные с муршрутами (endpoints) REST API.

2.5 Структура проекта серверной стороны

Пятый подраздел описывает базовую структуру проекта серверной стороны. В проекте задана следующая структура файлов и каталогов:

- server.js – веб-сервер
- routes.js – маршруты, обрабатывающие клиентские запросы
- middlewares – некоторое промежуточное программное обеспечение
- services – служебные функции и классы
- uploads – файлы загружаемые клиентом (например изображение профиля)
- models – модели для представления в базе данных, которые также содержат различные собственные методы
- controllsers – обработчики REST API

Вид структуры проекта можно посмотреть на рисунке 2.

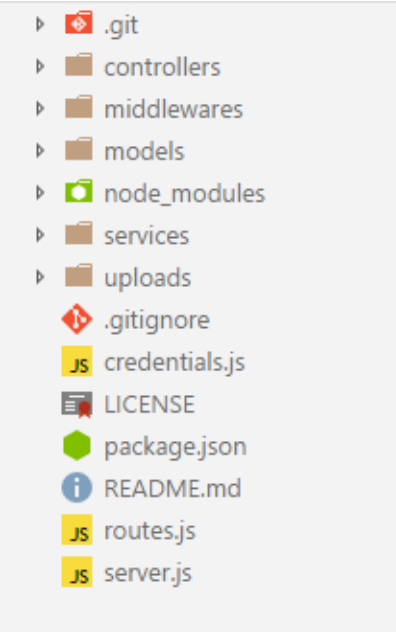

Рисунок 2 – структура серверной часты

ЗАКЛЮЧЕНИЕ

В результате выполнения данной работы была разработана социальная сеть с широким функционалом, которую можно увидеть в приложении 3 и 4, а также решены все поставленные задачи и актуальные проблемы.

В ходе работы был разработан совершенно новый подход к маршрутизации (routing). Теперь маршрутизация производится на клиентской стороне, что лучше сказывается на пользовательском опыте UX. Теперь навигация веб-приложения производится без перезагрузки страницы.

Также была разработана система кеширования и проксирования веб-запросов и статических ресурсов. Данная система позволяет существенно экономить сетевой трафик, что благополучно сказывается на нагрузке сервера, а также на затратах на его содержание.