

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе № 3
по дисциплине «WEB-технологии»
Тема: Модуль администрирования приложения «Социальная
сеть»

Студент гр. 2382

Преподаватель

Муравин Е. Е.

Беляев С. А.

Санкт-Петербург

2024

Цель работы

Изучение возможностей применения компилятора Babel, библиотеки jQuery, препроцессора LESS, препроцессора SASS/SCSS, инструмента выполнения повторяющихся задач GULP, освоение инструмента сборки Webpack, регистрация разработанных модулей, формирование навыков построения структурированных web-приложений, освоение особенностей стандартных библиотек.

Для достижения поставленной цели требуется решить следующие задачи:

1. Разработка интерфейса web-приложения с использованием Figma.
2. Создание web-сервера на основе Express, настройка маршрутов, подготовка и обработка REST-запросов (серверная часть).
3. Создание шаблонов web-страниц с использованием Pug или EJS, указание путей подключения JS-файлов.
4. Разработка стилей web-приложения с использованием LESS или SASS/SCSS.
5. Разработка клиентских JS-файлов с использованием библиотеки jQuery и новейших возможностей в соответствии с последним стандартом ECMAScript.
6. Конфигурирование GULP для решения задач преобразования Pug файлов в формат HTML, LESS-файлов и SASS-файлов в CSS-файлы, обработка JS-файлов с использованием Babel.

Задание

Необходимо создать web-приложение, обеспечивающее администрирование социальной сети: можно управлять участниками, их ролями, сообществами. Основные требования следующие:

1. Перечень участников, их друзей, сообщений и т.п. хранится в JSON файлах на сервере.
2. В качестве сервера используется Node.JS с модулем express.
3. Разработка ведется с использованием стандарта не ниже ECMAScript2015, используются ES6 модули.
4. Стили описываются с использованием LESS или SASS, при этом используются ключевые методы LESS/SASS (переменные, вложенные блоки, миксины, операторы и т. п.).
5. Клиентская часть разрабатывается с использованием jQuery (работа с DOM, AJAX-запросы), используются компоненты jQuery UI или Bootstrap.
6. Предусмотрена HTML-страница для списка пользователей (ФИО, дата рождения, email, фотография, роль, статус). Предусмотрена возможность редактировать данные пользователя, изменять роль (администратор, пользователь), изменять статус (не подтверждённый пользователь, активный, заблокированный).
7. Предусмотрены:
 - HTML-страница для списка друзей пользователя;
 - HTML-страница для списка новостей друзей пользователей.
8. Взаимодействие браузера с сервером осуществляется по протоколу HTTPS, все изменения сохраняются в соответствующие JSON-файлы на сервере.
9. Сборка клиентской части (преобразования less или sass, pug или ejs, babel, минификация) осуществляется с использованием двух

инструментов: GULP и Webpack. Это должны быть две отдельные сборки в разные папки.

10.Регистрация и удаление разработанных модулей в прм.

11.Для всех страниц web-приложения разработан макет интерфейса с использованием Figma (<https://www.figma.com/>).

Основные теоретические сведения

LESS и *SASS/SCSS* – это динамические языки стилей, обеспечивающие следующие расширения *CSS*: переменные, вложенные блоки, миксины, операторы и функции. *LESS* и *SASS/SCSS* могут работать на стороне клиента или на стороне сервера под управлением *Node.js*.

jQuery – библиотека JavaScript, предназначенная для упрощения взаимодействия JavaScript и HTML. Библиотека *jQuery* помогает получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими и предоставляет простой API для работы с AJAX.

Babel – компилятор JavaScript, который позволяет разработчику использовать в своих проектах самые последние стандарты ECMAScript с поддержкой во всех браузерах.

Gulp – это менеджер задач для автоматического выполнения часто используемых задач, написанный на JavaScript. Программное обеспечение поддерживает командную строку для запуска задач, определенных в конфигурационном файле.

Webpack (<https://webpack.js.org/>) – модуль JavaScript, обеспечивающий сборку статических пакетов («bundle»). На вход он получает «точки входа» (js-файлы), в которых он находит все зависимости и формирует соответствующие пакеты (по одному пакету на одну «точку входа»). Пакет представляет собой специально оформленный js-файл, в него входят не только связанные js-файлы, но и ресурсы, например, css-файлы.

Выполнение работы

Реализуем шаблон проекта на сайте Figma.

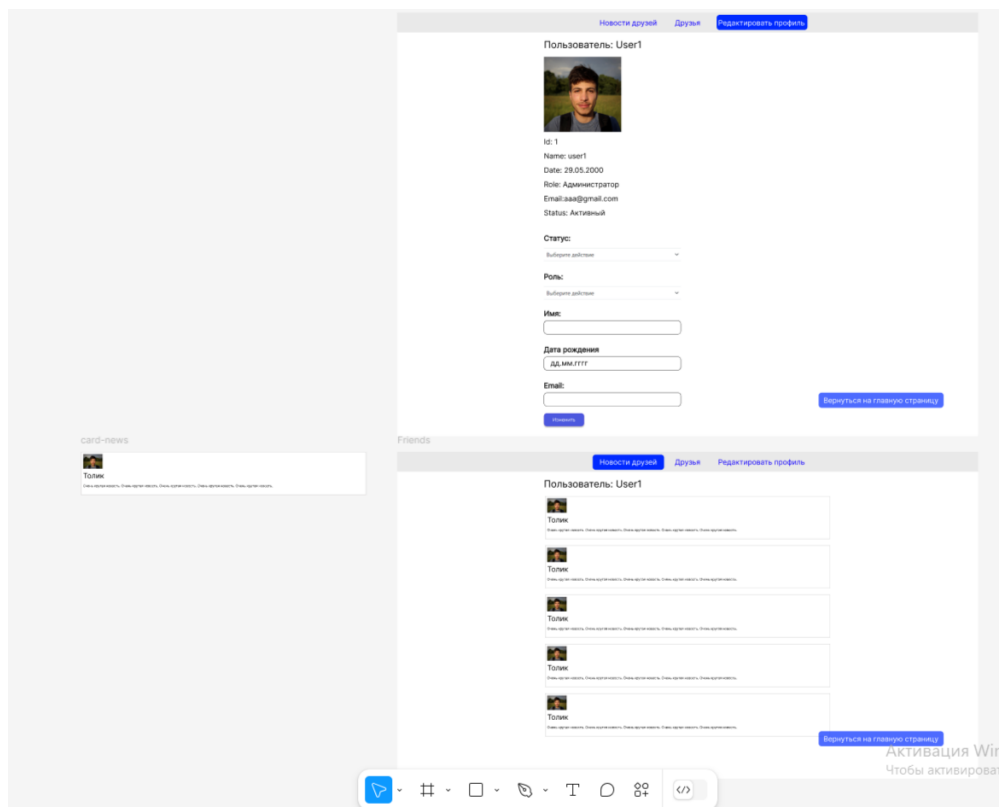


Рисунок 1 – шаблон проекта 1

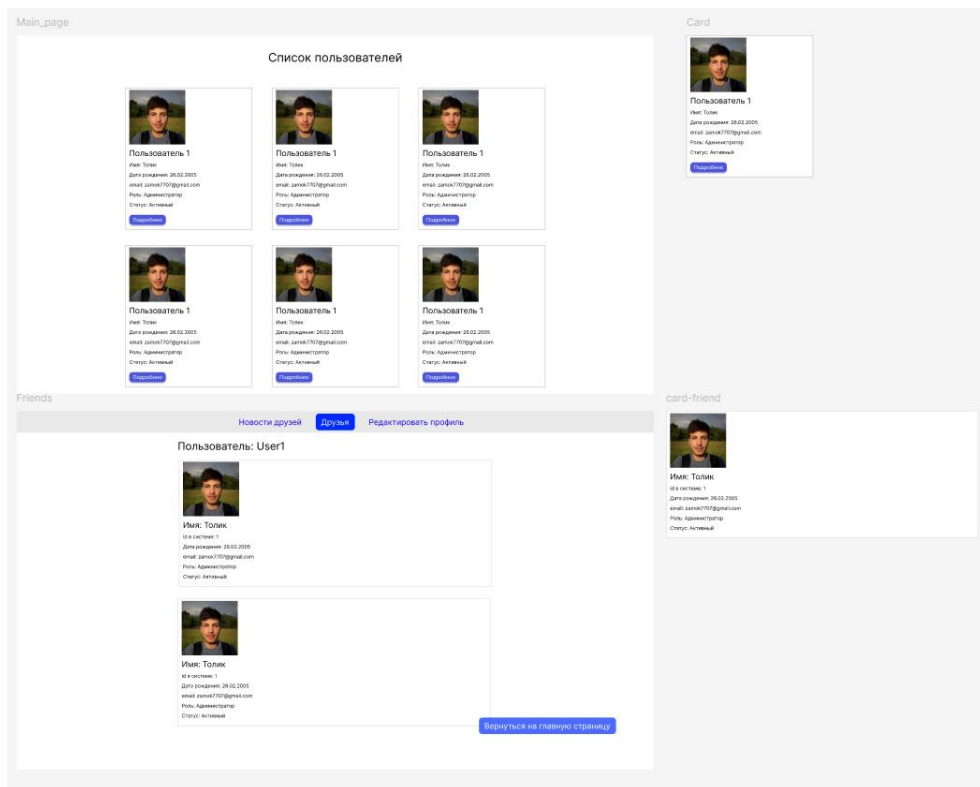


Рисунок 2 – шаблон проекта 2

Для взаимодействия по защищённому протоколу *HTTPS* сгенерируем самоподписанные *SSL*-сертификаты. Затем, с помощью функции *createServer* из модуля *https*, создадим сервер, который будет работать по этому протоколу.

В папке *views* создадим отдельные шаблоны **.pug* для каждой страницы. В каждом шаблоне подключим *Bootstrap* и *jQuery*, чтобы стилизовать элементы и упростить работу с DOM-структурой.

В папке *style* создадим отдельные **.less* файлы для каждой страницы, что позволит разделить стили для разных шаблонов. Это улучшит модульность и упростит управление стилями.

Клиентские *JavaScript* файлы будут храниться в папке *static*. В основном серверном файле *app.js* отметим, что папка *static* является статической, чтобы сервер мог напрямую обслуживать файлы из этой директории.

Для преобразования файлов *.pug* в *HTML* и *.less* в *CSS* использовались менеджеры задач *Gulp* и *Webpack*.

В *Gulp* каждое действие задается отдельной функцией (*task*). Для выполнения задания указывается путь к исходным файлам с помощью *gulp.src*, после чего через метод *.pipe* задаются необходимые действия, такие как преобразование шаблонов, минимизация *CSS* и *HTML*, конкатенация файлов в один. В завершение задачи указывается путь для сохранения преобразованных файлов через *gulp.dest()*, а затем вызывается функция *callback* для перехода к следующему заданию.

Для задания последовательности выполнения задач в *Gulp* используются функции *series* и *parallel*. Функция *series* задает последовательное выполнение задач, выполняя их одну за другой, а *parallel* позволяет выполнять несколько задач одновременно.

Для отслеживания изменений в файлах программы применяется метод *watch*, который следит за указанными файлами и при их изменении автоматически вызывает заданную функцию для обработки этих изменений.

Webpack выполняет почти такие же преобразования, как и *Gulp*, но организует их несколько иначе. Для работы *Webpack* требуется конфигурационный файл *webpack.config.js*, в котором настраиваются задачи для сборки.

Основные настройки прописываются в *module.exports*, где задаются следующие параметры:

- *mode* — режим работы проекта, указывающий на состояние разработки или готовности к публикации;
- *entry* — пути к JavaScript-файлам, которые *Webpack* будет обрабатывать в ходе сборки;
- *output* — место для выхода обработанных файлов;
- *module* — с использованием параметра *rule* устанавливаются правила обработки различных файлов, определяемых через параметр *test*;
- *plugins* — задаются дополнительные задачи, такие как минификация CSS-файлов (в данном варианте вывод в */style/main.css*), преобразование *Pug*-файлов из папки *views* в *HTML* и их перенос в папку *html*.

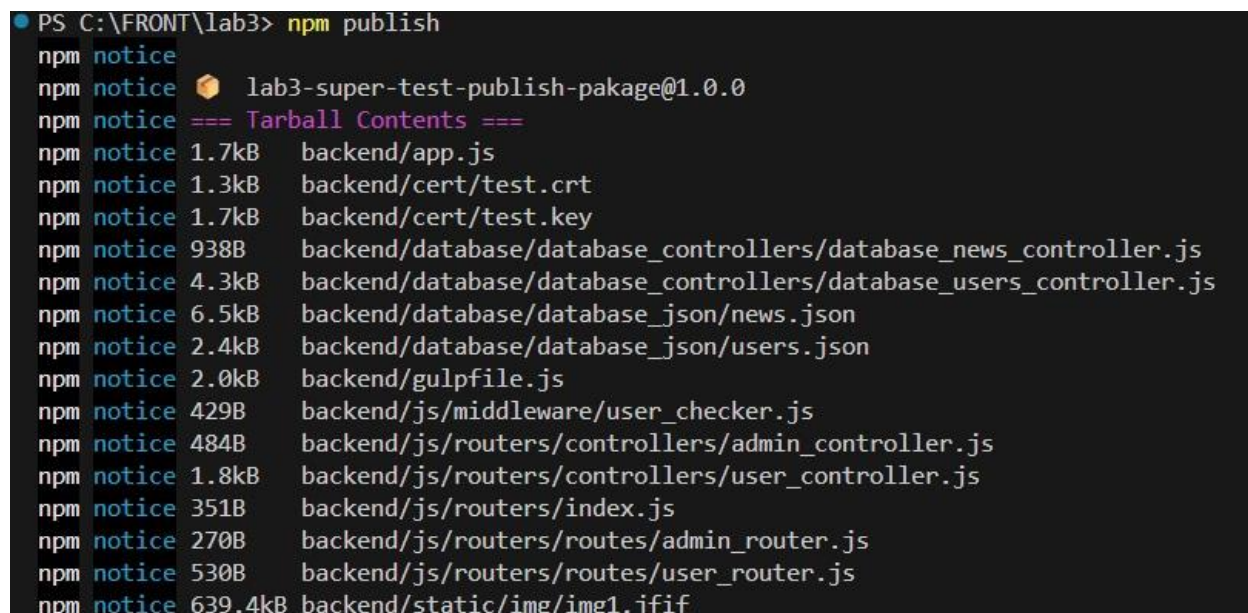
Все данные пользователей сохраняются в JSON-файлах на сервере. Для управления ими реализованы контроллеры *database_user_controller.js* и *database_news_controller.js*. Эти классы выполняют функции, похожие на ORM (Object-Relational Mapping), обеспечивая взаимодействие с "базой данных" через методы классов.

Для маршрутизации по страницам сайта на сервере настроены маршруты с использованием *Router*. Для обработки запросов по маршрутам созданы два контроллера, которые принимают и обрабатывают запросы от клиента.

Работа сервера начинается с отправки *HTML*-файла по *GET*-запросу, при этом клиенту отправляется практически пустой *HTML*-файл. Затем клиент посылает *AJAX*-запрос для получения данных конкретного пользователя, указанных в *URL*. После получения данных на стороне клиента карточки пользователя добавляются с помощью метода *jQuery append*. Большая часть стилизации страниц выполняется с использованием *Bootstrap 5.0*.

На странице редактирования пользователя изменение данных осуществляется по нажатию кнопки "Изменить". Клиент отправляет на сервер объект с данными, и сервер обрабатывает его, проверяя параметры объекта. Если параметры присутствуют, данные в базе обновляются. Чтобы избежать перезагрузки страницы при каждом изменении, текущие данные пользователя обновляются на клиенте с помощью функции *set_new_info*.

После завершения написания кода проект был загружен как модуль на *npt* с использованием команды *npm publish*, предварительно выполнив авторизацию. После загрузки модуль был удален.



```
PS C:\FRONT\lab3> npm publish
npm notice
npm notice 📦 lab3-super-test-publish-pakage@1.0.0
npm notice === Tarball Contents ===
npm notice 1.7kB backend/app.js
npm notice 1.3kB backend/cert/test.crt
npm notice 1.7kB backend/cert/test.key
npm notice 938B backend/database/database_controllers/database_news_controller.js
npm notice 4.3kB backend/database/database_controllers/database_users_controller.js
npm notice 6.5kB backend/database/database_json/news.json
npm notice 2.4kB backend/database/database_json/users.json
npm notice 2.0kB backend/gulpfile.js
npm notice 429B backend/js/middleware/user_checker.js
npm notice 484B backend/js/routers/controllers/admin_controller.js
npm notice 1.8kB backend/js/routers/controllers/user_controller.js
npm notice 351B backend/js/routers/index.js
npm notice 270B backend/js/routers/routes/admin_router.js
npm notice 530B backend/js/routers/routes/user_router.js
npm notice 639.4kB backend/static/img/img1.jfif
```

Рисунок 3 – Загрузка модуля

```
PS C:\FRONT\lab3> npm ls
lab3-super-test-publish-pakage@1.0.0 C:\FRONT\lab3
├─┬ @babel/core@7.26.0
│   └─┬ @babel/polyfill@7.12.1
│       ├── @babel/preset-env@7.26.0
│       ├── @babel/register@7.25.9
│       ├── @webpack-cli/generators@3.0.7
│       ├── babel-core@4.7.16
│       ├── babel-loader@9.2.1
│       ├── babel-preset-env@1.7.0
│       ├── clean-webpack-plugin@4.0.0
│       ├── css-loader@6.11.0
│       ├── express-fileupload@1.5.1
│       ├── express@4.21.1
│       ├── fs-extra@11.2.0
│       ├── gulp-babel@8.0.0
│       ├── gulp-clean-css@4.3.0
│       ├── gulp-cli@2.3.0
│       ├── gulp-concat@2.6.1
│       ├── gulp-htmlmin@5.0.1
│       ├── gulp-less@5.0.0
│       ├── gulp-nodemon@2.5.0
│       ├── gulp-pug@5.0.0
│       ├── gulp-uglify@3.0.2
│       ├── gulp@4.0.2
│       ├── gulp4@4.0.2
│       ├── html-webpack-plugin@5.6.3
│       ├── less-loader@11.1.4
│       ├── less@4.2.0
│       ├── mini-css-extract-plugin@2.9.1
│       ├── nodemon@3.1.7
│       ├── pug-loader@2.4.0
│       ├── pug@3.0.3
│       ├── sass-loader@13.3.3
│       ├── sass@1.80.4
│       ├── style-loader@3.3.4
│       └── webpack-cli@5.1.4
```

Рисунок 4 – Загруженный модуль

```
PS C:\FRONT\lab3> npm unpublish lab3-super-test-publish-pakage --force
npm WARN using --force Recommended protections disabled.
- lab3-super-test-publish-pakage
```

Рисунок 5 – Удаление модуля

Список пользователей







 <p>Пользователь 1 Имя: Egor Дата рождения: 26.02.2005 email: zamok7707@gmail.com Роль: Администратор Статус: Активный Подробнее</p>	 <p>Пользователь 2 Имя: Егорчик Дата рождения: 16.05.2004 email: slavik2004@gmail.com Роль: Пользователь Статус: Не подтвержденный Подробнее</p>	 <p>Пользователь 3 Имя: Valeros Дата рождения: 09.05.1996 email: valeros@gmail.com Роль: Пользователь Статус: Заблокированный Подробнее</p>
 <p>Пользователь 4 Имя: Petrusok Дата рождения: 01.04.2009 Подробнее</p>	 <p>Пользователь 5 Имя: Alexandr Дата рождения: 31.12.2006 Подробнее</p>	 <p>Пользователь 6 Имя: Olega Дата рождения: 15.11.2003 Подробнее</p>


Рисунок 6 – Главная страница

Новости друзей

Друзья

Редактировать профиль

Пользователь: Egor



Имя: Егорчик


Дата рождения: 16.05.2004

Почтовый адрес: slavik2004@gmail.com

Роль: Пользователь

Статус: Не подтвержденный

id в системе: 2



Имя: Valeros

Дата рождения: 09.05.1996

Почтовый адрес: valeros@gmail.com

Роль: Пользователь

Активация

Чтобы активировать профиль, нажмите на кнопку "Вернуться на главную страницу".


Рисунок 7 – Страница друзей

Новости друзей

Друзья


Редактировать профиль

Пользователь: Egor




Егорчик

Однажды я решил устроить друзьям сюрприз и провести весь день, говоря только стихами. Сначала это было весело — каждый ответ был как маленький стих, и все смеялись. Но к середине дня друзья начали устало смотреть на меня, просили «говорить нормально», но я не сдавался. К вечеру я уже сам устал придумывать рифмы, но, как только они признали меня чемпионом стихов, понял, что всё было не зря!



Valeros

Я всегда мечтал о собственном велосипеде, и когда мне наконец подарили красный байк, я не мог сдержать радости. Каждый день после школы я катался по парку, ощущая ветер в волосах и свободу. Однажды я даже принял участие в небольшом соревновании, и это было незабываемо!



Valeros

Когда я устроился на свою первую работу в кафе, я был полон волнения и страха. Мне нужно было учиться быстро, и иногда я путал заказы, но коллеги всегда помогали и поддерживали меня. Со временем я стал увереннее в себе и даже завел новых друзей.

Активация


Чтобы активировать профиль, нажмите на кнопку "Вернуться на главную страницу".

Рисунок 8 – Новости друзей

11

Новости друзейДрузьяРедактировать профиль

Пользователь: Egor



id: 1
Name: Egor
Date: 26.02.2005
Email: damok7707@gmail.com
Role: Администратор
Status: Активный

Статус:
Выберите действие

Роль:
Выберите действие

Имя:

Дата рождения:
ДД.ММ.ГГГГ

Email:

Изменить

Активация Windows
Чтобы активировать Windows, [вернитесь на главную страницу](#)

Рисунок 9 – Страница редактирования

Выводы

В ходе работы над проектом были изучены новые технологии веб-разработки, такие как Babel, jQuery и LESS. Серверная часть, реализованная на Node.js с использованием Express, обеспечила удобную маршрутизацию и обработку запросов. Данные пользователей хранились в формате JSON с помощью контроллеров, функционирующих по принципу ORM. Для автоматизации процессов преобразования файлов и минимизации кода использовались Gulp и Webpack. Взаимодействие с сервером осуществлялось через AJAX-запросы. Также была произведена загрузка и удаление модуля на сервис npm.