МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Веб-технологии»

Тема: Модуль администрирования приложения «Социальная сеть»

|  |  |  |
| --- | --- | --- |
| Студент гр. 0304 |  | Асташёнок М.С. |
| Преподаватель |  | Беляев С.А. |

Санкт-Петербург

2022

Цель работы:

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

Для достижения поставленной цели требуется решить следующие задачи:

– разработка интерфейса web-приложения с использованием Figma (<https://www.figma.com/>);

– создание web-сервера на основе express, настройка маршрутов, подготовка и обработка REST-запросов (серверная часть);

– создание шаблонов web-страниц с использованием pug или ejs, указание путей подключения js-файлов;

– разработка стилей web-приложения с использованием LESS или SASS/SCSS;

– разработка клиентских js-файлов с использованием библиотеки jQuery и с использованием новейших возможностей в соответствии последним стандартом ECMAScript;

– 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.

Общая формулировка задачи

Необходимо создать web-приложение, обеспечивающее администрирование социальной сети: можно управлять участниками, их ролями, сообщениями. 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 npm.

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

Преимуществом будет, если будет предусмотрена работа с:

– фотографиями пользователя: список фотографий и возможность блокирования и активации фотографий администратором;

– новостями: возможность блокирования и активации новостей.

Выполнение работы

Рисование, верстка и стиллизация.

Первым делом был подготовлен макет Figma. В данном макете представлено 4 представления:

* Список пользователей сайта
* Профиль пользователя: новости
* Профиль пользователя: друзья
* Редактирование профиля.

Далее по данным представлениям были сверстаны 5 страниц: index.pug (начальная страница), not\_found.pug (страница ошибки клиента), control\_panel.pug (список пользователей для администратора), edit\_profile.pug (страница редактирования профиля) и profile.pug (страница профиля).

Для написанных шаблонов были составлены одноименные таблицы стилей, сперва в виде CSS, а затем после подключения к каждому шаблону скрипта для компиляции LESS кода, стили были переписаны под LESS.

Написание сервера.

Далее был написан сервер. В файле server.js был инициализирован сервер на express-js, а также настройки для подключения самоподписанных ключей для запуска сервера по протоколу HTTPS. Ключ и сертификат хранятся в директории public/certificate, для запуска сервера по протоколу HTTPS был импортирован модуль “https”.

В качестве базы данных выступает файл database.json, который хранит в себе два массива: пользователей (users) и новостей (posts).

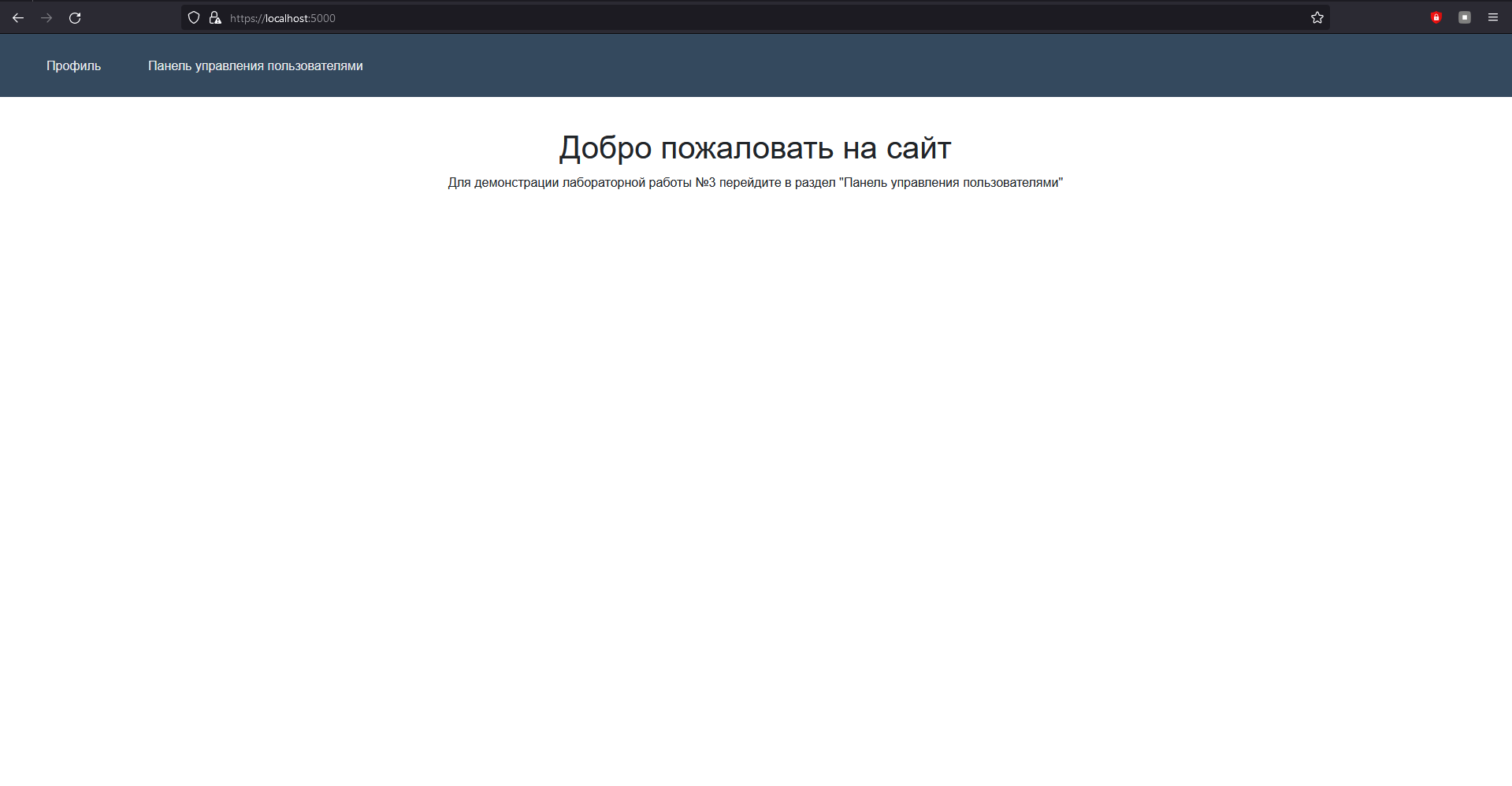
Следующим шагом было написание путей запросов для работы с API сервера. В папке routes было создано три JS-файла:

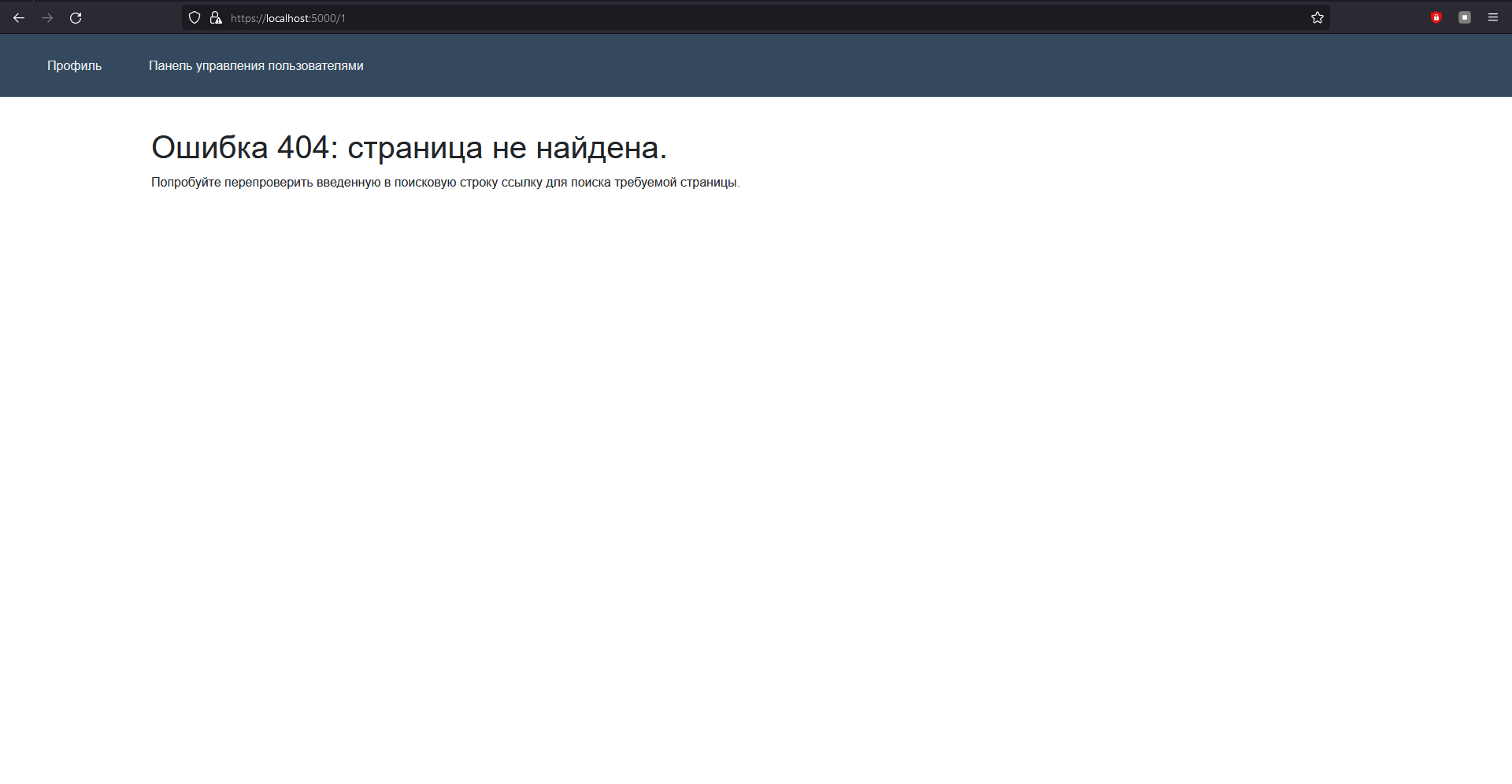
* Render – содержит в себе пути для отрисовки шаблонов .pug страницы
* Users – содержит в себе пути для оформления запросов, связанных с пользователем
* Posts – содержит в себе пути для оформления запросов, связанных с постами (новостями).

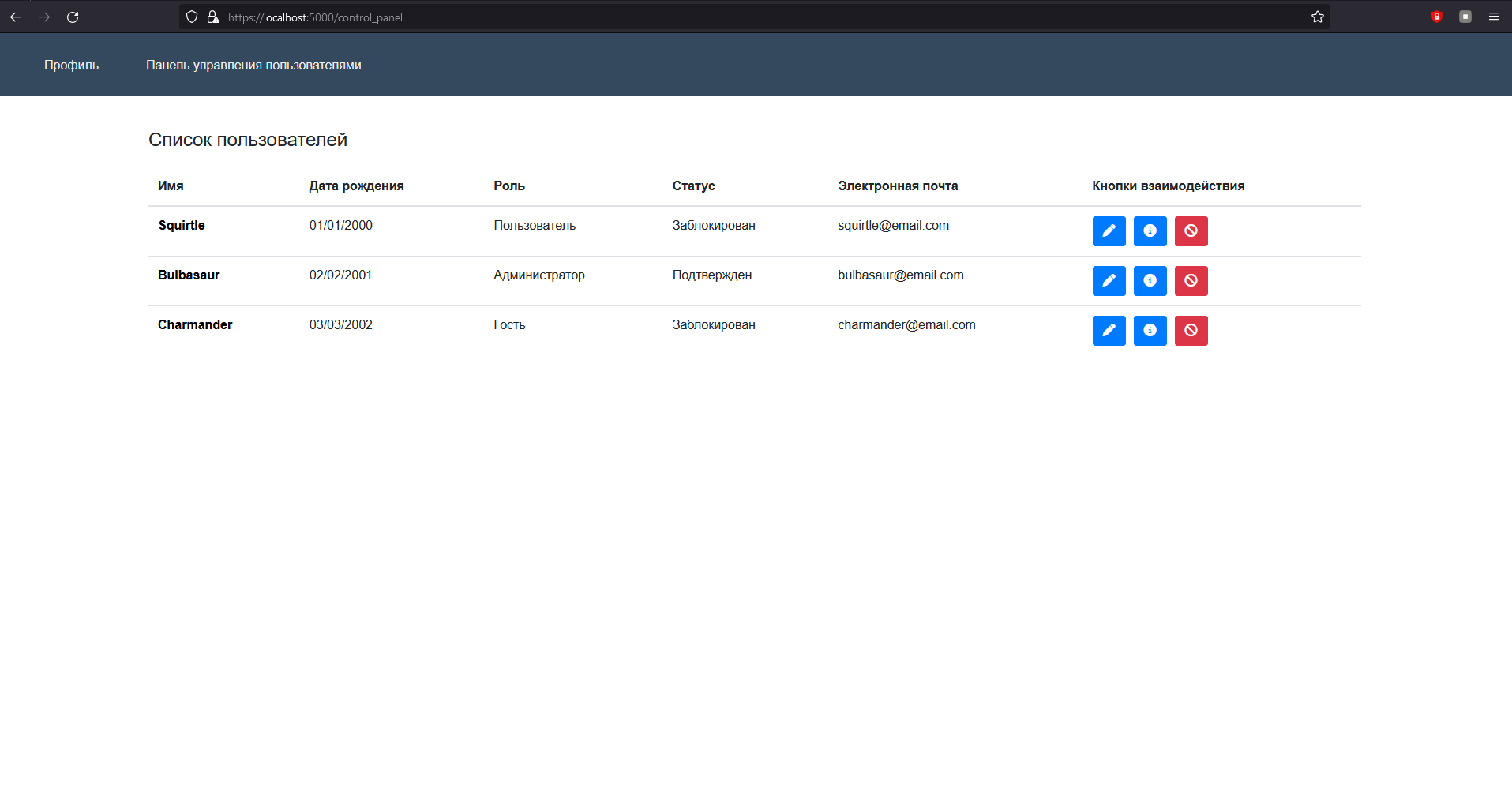
Когда все запросы были описаны, далее необходимо было настроить клиентскую часть проекта. Сперва был написан файл public/scripts/client.js, который содержит в себе все необходимые AJAX-запросы на сервер. Затем для каждого шаблона был разработан свой js-скрипт для взаимодействия с DOM-элементами и добавления функционала к кнопкам и ссылкам на странице. В качестве основного инструмента при написании клиентской части сайта выступает JQuery.

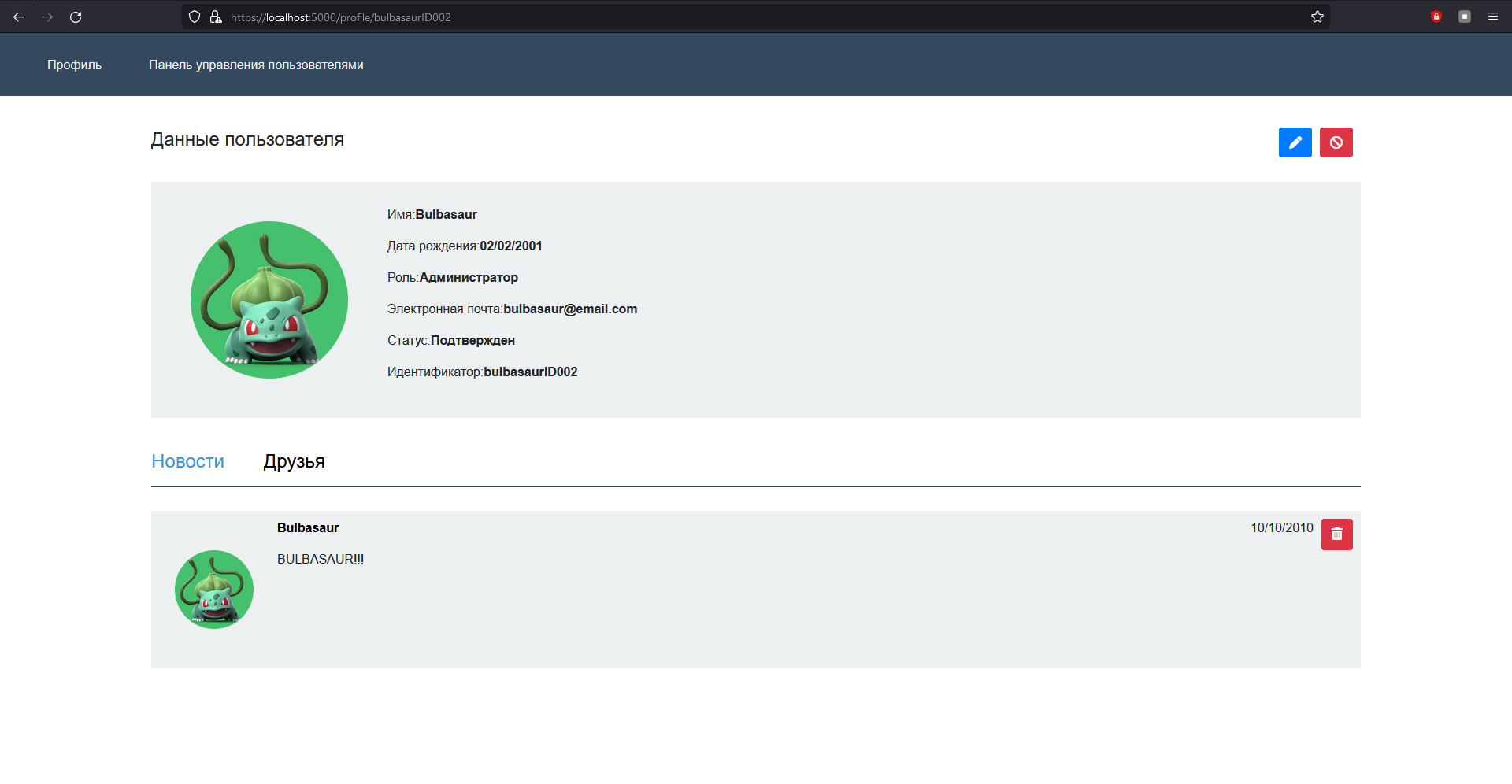
Вид сайта.

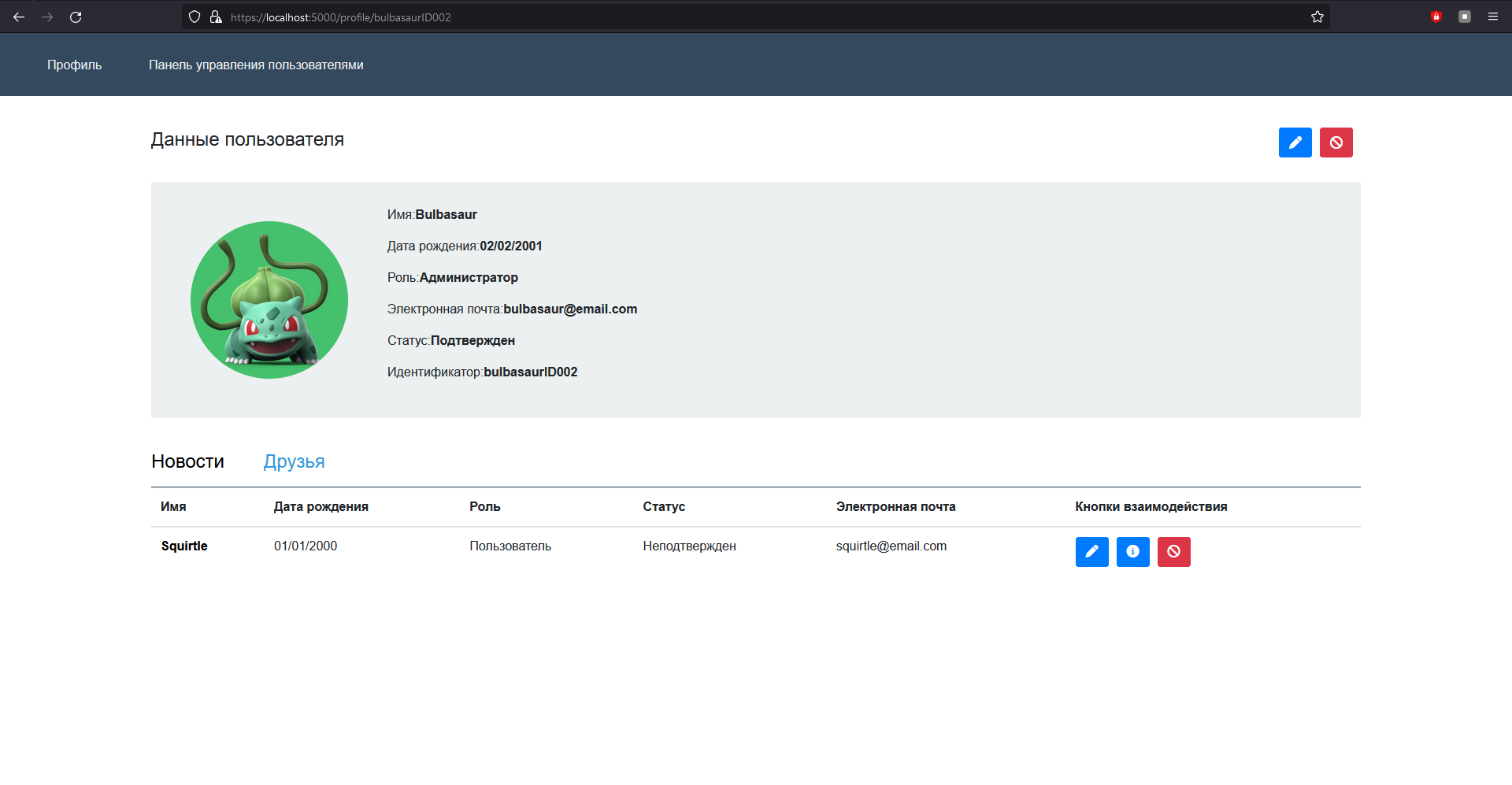
Ниже представлен вид сайта при запуске приложения:

Рисунок 1. Стартовая страница сайта.

Рисунок 2. Страница ошибки клиента.

Рисунок 3. Панель управления пользователями.

Рисунок 4. Страница профиля: новости.

Рисунок 5. Страница профиля: друзья.

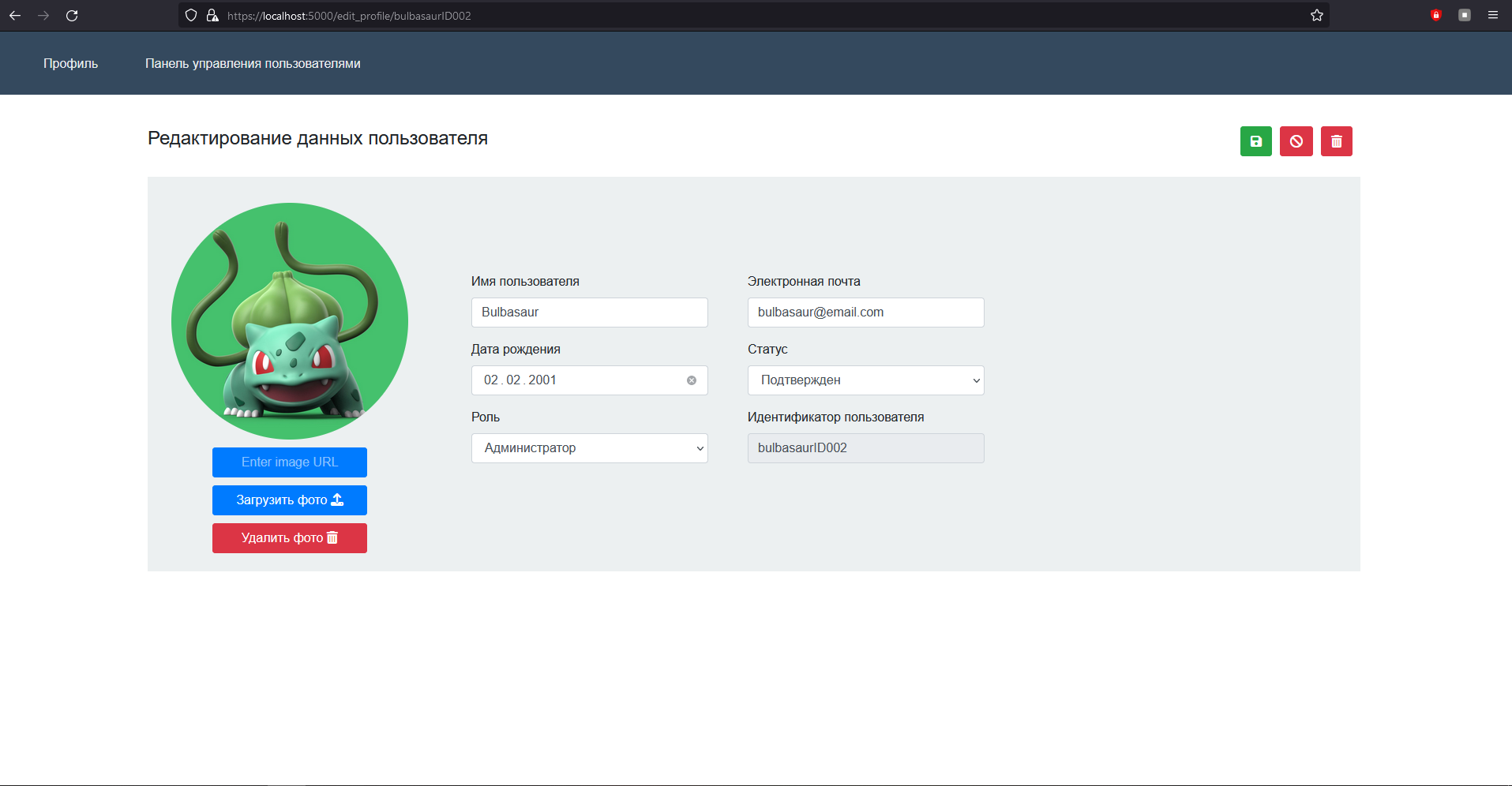


Рисунок 6. Страница редактирования профиля.

Настройка сборщиков.

Первым делом были созданы специальные версии шаблонов и скриптов для сборки приложения с помощью gulp или webpack, т.к. исходные файлы были бы несовместимы при работе с собранной сборщиками версией приложения.

В папке public/webpack\_scripts были указаны скрипты, которые подгружают необходимые стили и клиентские скрипты. Скрипты, указанные в данной папке, являются входной точкой при сборке приложения через webpack

В папке public/build хранятся другие ресурсы, необходимые для сборки, однако они уже более универсальны и подходят для обоих сборщиков. Gulp\_views и webpack\_views содержат в себе шаблоны для сборки соответственно с помощью gulp и webpack, а routes и scripts содержат в себе пути для запросов и клиентские скрипты соответственно для собранных версий приложения с помощью разных сборщиков.

Далее, в корне приложения было создано два конфигурационных сборочных файла:

* Gulpfile.js для Gulp
* Webpack.config.js для Webpack.

Для того, чтобы собрать проект с помощью Gulp, необходимо в терминале корневой папки ввести следующую строку:

npm run gulp\_build

Для Webpack соответственно:

npm run webpack\_build

Собранные приложения попадут в директорию dist\_gulp или dist\_webpack, в зависимости от способа сборки.

Запуск приложения.

Для того, чтобы запустить «ванильную» версию приложения, в корневом терминале достаточно прописать следующую команду:

npm start

Для запуска собранного с помощью Gulp приложения:

node dist\_gulp/server.js

Для запуска собранного с помощью Webpack приложения:

node dist\_webpack/server.js

Вывод.

В ходе выполнения данной лабораторной работы были изучены принципы работы с JQuery, LESS, компилятором каскадных таблиц стилей, а также применены на практике знание по настройке сборщиков, таких как Gulp и Webpack.

В ходе выполнения лабораторной работы был написан модуль администрирования приложения «Социальная сеть»