**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 |  | Муравин Е. Е. |
| Преподаватель |  | 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:

## Разработка интерфейса web-приложения с использованием Figma.

## 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 новейших возможностей в соответствии с 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.

## Задание

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

## Перечень участников, их друзей, сообщений и т.п. хранится в JSON файлах на сервере.

## В качестве сервера используется Node.JS с модулем express.

## Разработка ведется с использованием стандарта не ниже ECMAScript2015, используются ES6 модули.

## Стили описываются с использованием LESS или SASS, при этом используются ключевые методы LESS/SASS (переменные, вложенные блоки, миксины, операторы и т. п.).

## Клиентская часть разрабатывается с использованием jQuery (работа с DOM, AJAX-запросы), используются компоненты jQuery UI или Bootstrap.

## 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).

## Предусмотрены:

## - HTML-страница для списка друзей пользователя;

## - HTML-страница для списка новостей друзей пользователей.

## Взаимодействие браузера с сервером осуществляется по протоколу HTTPS, все изменения сохраняются в соответствующие JSON-файлы на сервере.

## 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.

## Регистрация и удаление разработанных модулей в npm.

## 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 *npm* с использованием команды *npm publish*, предварительно выполнив авторизацию. После загрузки модуль был удален.

## https://sun1-95.userapi.com/impg/P1XXkh3ztwcqYd4XXy71UxgOWfodjWLawOWoGw/Aw6LdvItqsk.jpg?size=811x396&quality=96&sign=63461e5a40bf3743540fea3feb792202&type=album

## Рисунок 3 – Загрузка модуля

## https://sun1-98.userapi.com/impg/Mo52wO8CIAX8pHaSf-QS4-xvrCjw7vcmLNmiNw/eCWkcuCVl4U.jpg?size=467x779&quality=96&sign=8f4513103b2404ee7e30e60400e355d7&type=album

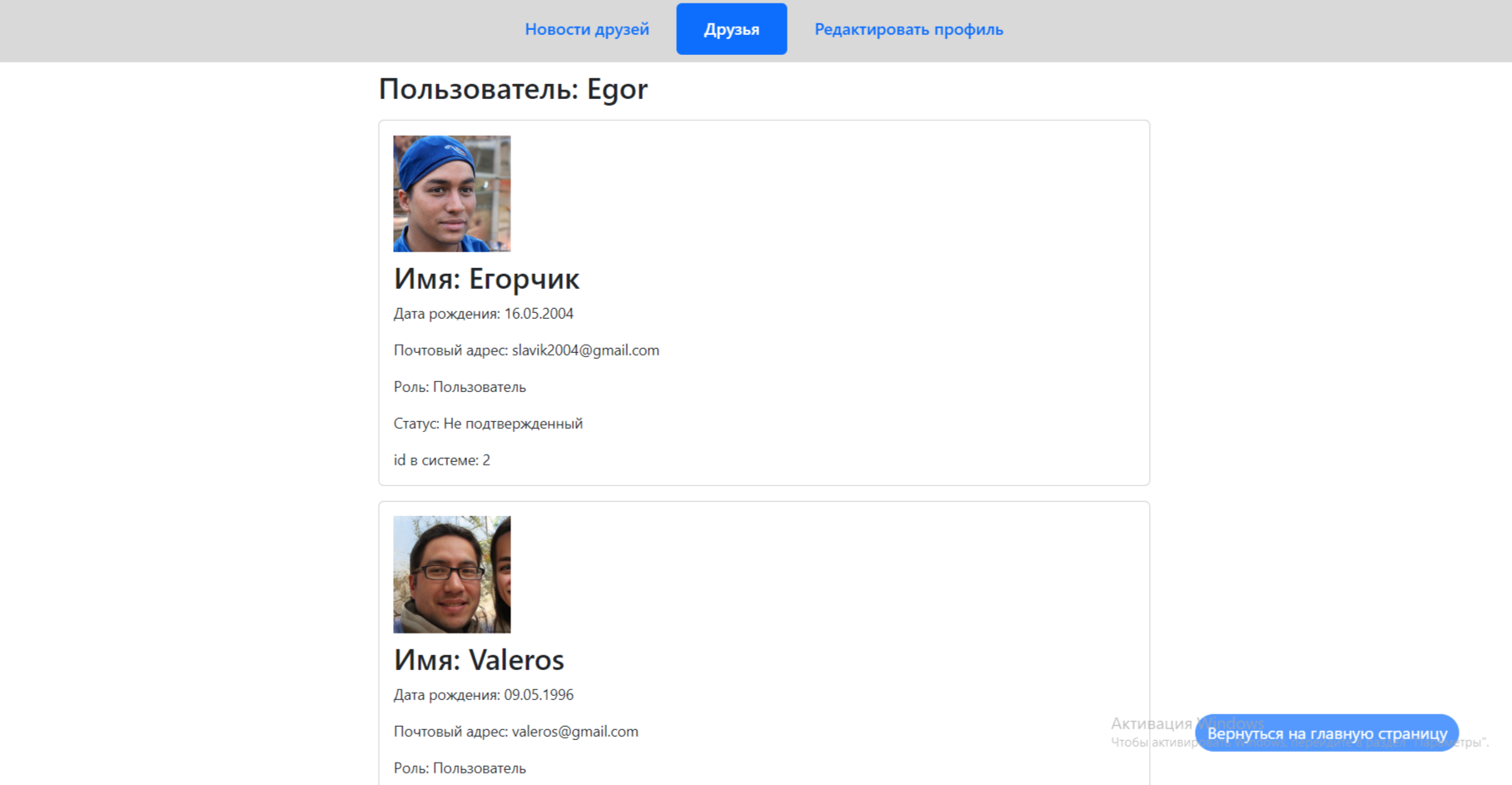
## Рисунок 4 – Загруженный модуль

## https://sun9-67.userapi.com/impg/zkXWSG7TWNhnbS2EHYANq-b9FUfQt_CkZWRQEg/WO-ffHmuSjw.jpg?size=683x67&quality=96&sign=77d4de12cd5967eb23435f4baad232f3&type=album

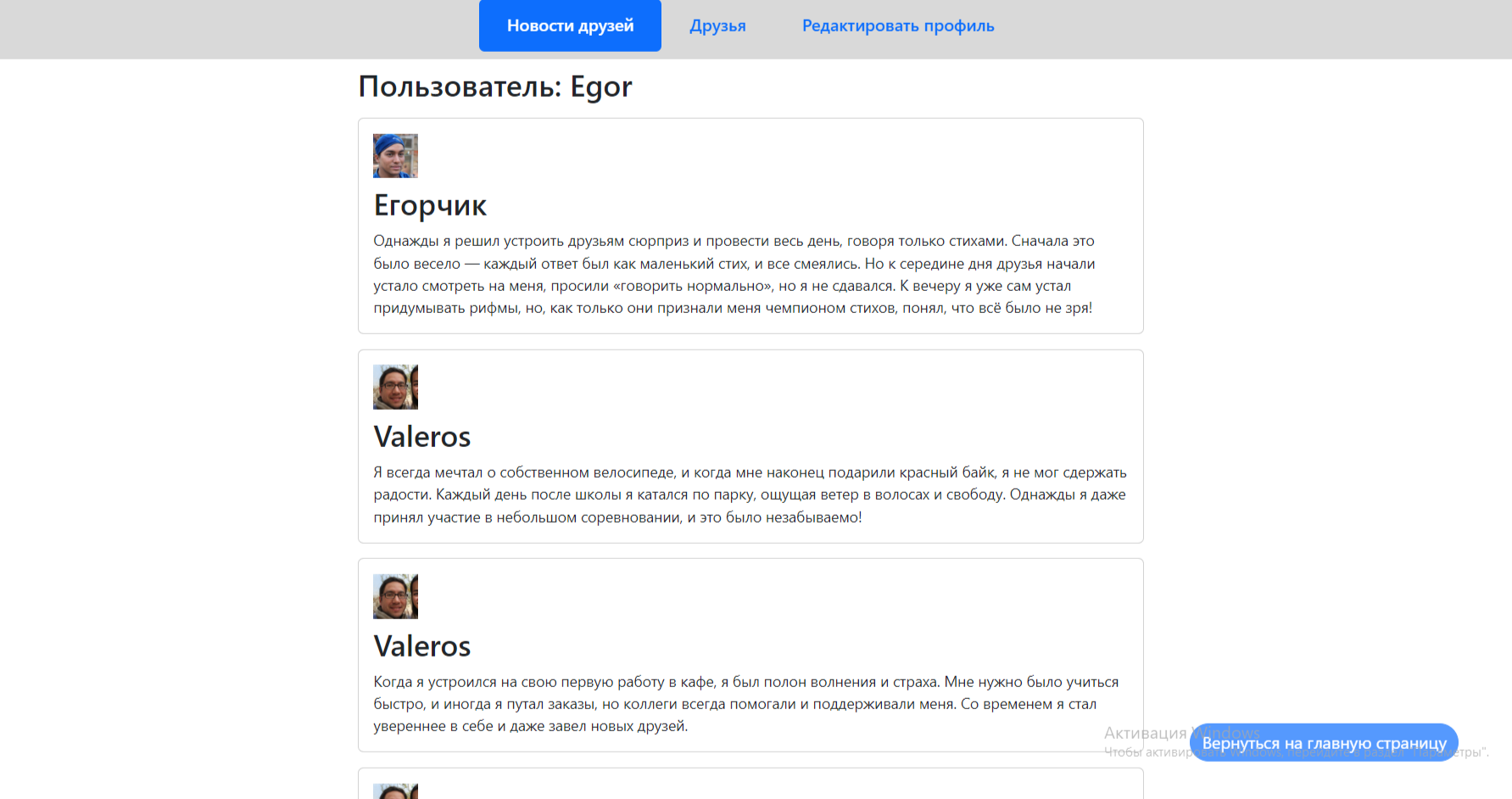
## Рисунок 5 – Удаление модуля

## 

## Рисунок 6 – Главная страница



## Рисунок 7 – Страница друзей

Рисунок 8 – Новости друзей

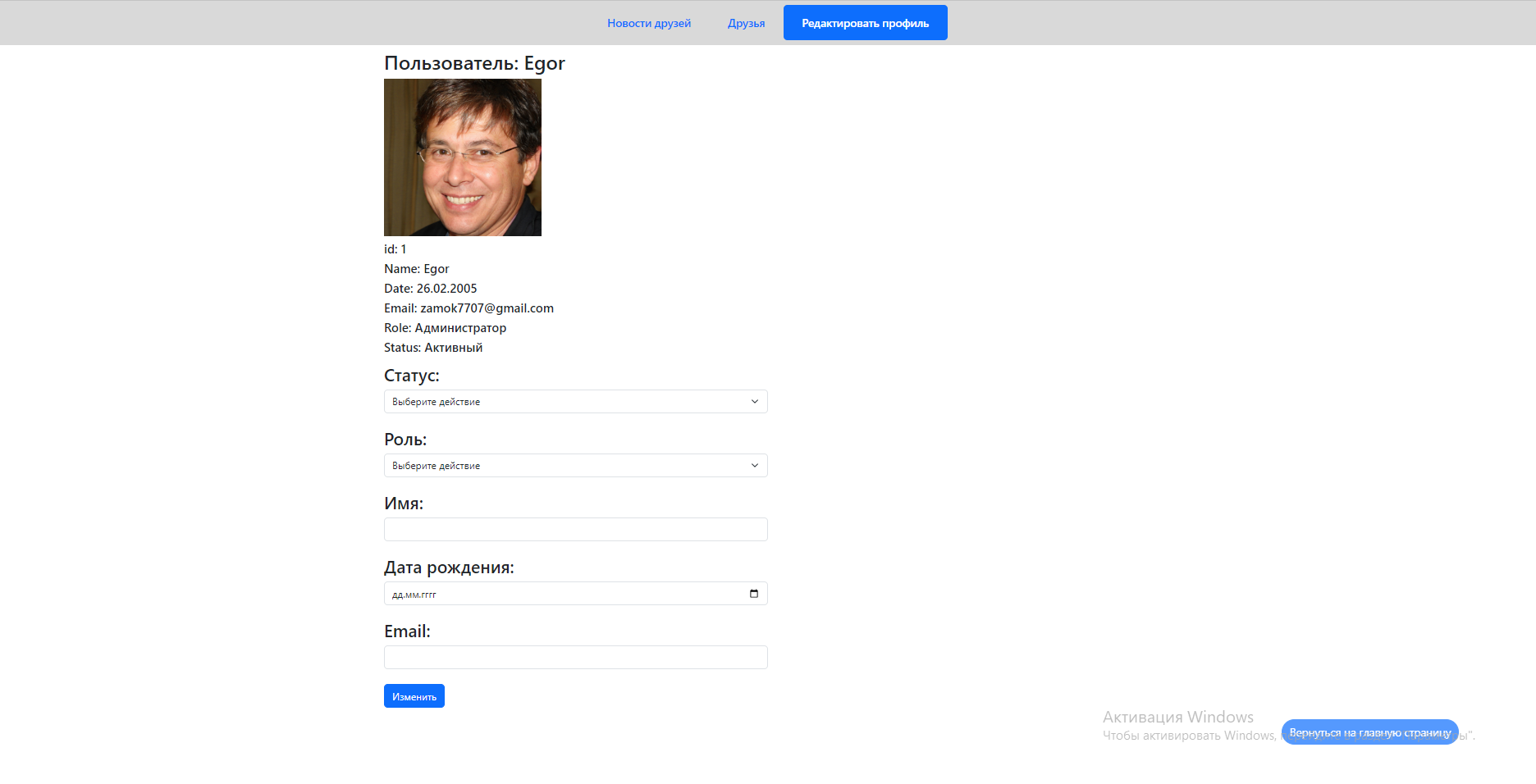


Рисунок 9 – Страница редактирования

**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.