Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное   
образовательное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ ВЕБСАЙТА НА REACT.JS ДЛЯ КОМПАНИИ "АКТИВ"

Курсовой проект по учебной дисциплине «Системное программирование»

по специальности 09.03.01 Информатика и вычислительная техника  
ПТИ.КП 3091.004.42ПЗ

|  |  |
| --- | --- |
|  | Руководитель  / И. Ю. Кулаков  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2025 г. |
|  | Студент группы 3091  / Р. А. Михайлов  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2025 г. |

Содержание

[Введение 4](#_Toc21633)

[1 Постановка задачи 6](#_Toc13334)

[1.1 Обоснование необходимости разработки 6](#_Toc11839)

[1.2 Технико-математическое описание задачи 6](#_Toc24122)

[1.3 Анализ предметной области 7](#_Toc30549)

[1.4 Требования к программе 7](#_Toc25484)

[1.5 Обоснование проектных решений 7](#_Toc8252)

[1.6 Обзор и анализ существующих программных систем 8](#_Toc11872)

[1.7 Выводы предпроектного исследования 9](#_Toc7884)

[2 Теоретическая часть 10](#_Toc1716)

[2.1 История научных исследований по выбранной теме 10](#_Toc18024)

[2.2 Определение и анализ ключевых терминов 10](#_Toc30775)

[2.3 Актуальные взгляды на выбранную проблему 11](#_Toc19062)

[3 Практическая часть 13](#_Toc11621)

[3.1 Анализ задачи 13](#_Toc1416)

[3.2 Описание логической структуры 13](#_Toc4224)

[3.3 Разработка программы для работы 15](#_Toc13029)

[3.4 Сравнительная характеристика работы 16](#_Toc18811)

[Заключение 17](#_Toc30042)

[Используемые источники 18](#_Toc10253)

[Приложение А 19](#_Toc7535)

[Приложение Б 20](#_Toc17286)

[Приложение В 21](#_Toc29307)

[Приложение Г 22](#_Toc9756)

Введение

Современная медицина нуждается в специализированных цифровых платформах, объединяющих врачей-реабилитологов и пациентов с нарушениями опорно-двигательного аппарата. Несмотря на обилие медицинских ресурсов, существующие решения часто предлагают разрозненную информацию без удобных инструментов для профессионального общения. Особенно проблематичным остаётся отсутствие адаптивных интерфейсов, одинаково удобных как для специалистов, так и для родителей пациентов.

Разработка платформы "socbook" направлена на создание универсального пространства для обмена медицинскими знаниями и практическим опытом. Её ключевое отличие – двунаправленный подход: врачи получат доступ к актуальным исследованиям и инструментам для публикации кейсов, а родители – к проверенным методикам реабилитации и системам контроля прогресса. Это позволит преодолеть существующий информационный разрыв между профессиональным сообществом и пациентами.

Социальная значимость проекта обусловлена его потенциалом для повышения качества реабилитационной помощи. Удобный интерфейс с персонализированными функциями сделает современные методики доступнее, особенно для жителей удалённых регионов. Интеграция профессиональных знаний и практических инструментов в единую платформу создаст новую цифровую экосистему для всех участников реабилитационного процесса.

Пользовательская аудитория данного сайта представляет из себя врачей-реабилитологов, в частности специалисты по опорно-двигательной системе, заинтересованных в предоставлении таким же врачам достоверной актуальной информации в сфере реабилитации. Так же пользователями могут быть родители с детьми, которые имеют проблемы с опорно-двигательной системой

Цель: разработка пользовательского интерфейса для веб-платформы «socbook»

Задачи:

- проведение анализа существующих решений и технологий для реализации;

- разработка удобного пользовательского интерфейса, дизайна;

- создание макета на основе разработанного дизайна;

- написание программного кода, опираясь на созданный макет;

- тестирование разработанного интерфейса;

- подготовка документации.

1. [Постановка задачи](#_Toc452730688)
   1. Обоснование необходимости разработки

В соответствии с заданием на курсовую работу, требуется разработать клиентскую часть веб-платформы «socbook» для компании «Актив»

Сейчас врачам и пациентам не хватает удобных онлайн-платформ для реабилитации. Хотя медицинских сайтов много, у них есть несколько проблем:

- информация разбросана по разным источникам;

- нет удобных инструментов для общения врачей между собой и с пациентами;

- сайты неудобные - плохо работают на телефонах и компьютерах.

Особенно сложно родителям, которые помогают детям с проблемами опорно-двигательной системы. Им нужен простой и понятный сервис, где:

- можно быстро найти нужные упражнения;

- посоветоваться с врачами;

- следить за прогрессом.

* 1. Технико-математическое описание задачи

Для разработки клиентской части сервиса будут использованы такие языки как HTML, CSS, JavaScript.

Для создания дизайна и макета сайта будет использована Figma, онлайн векторный редактор.

Для ускорения разработки страниц сайта будет использована библиотека React.JS, основные преимущества это поддержка компонентов и виртуального DOM’а, который позволяет локально изменять содержимое страницы, оптимизируя работу сервиса.

* 1. Анализ предметной области

Целевой аудиторией данного проекта является компания «Актив». Эта компания заинтересована в создании соцсети для врачей реабилитологов, производителей технических средств реабилитации, организаций и общественных организаций. Их целью является создание единого цифрового пространства для вышеперечисленных групп, а так же их пациентов с проблемами опорно-двигательного аппарата, жалеющий получать качественную информации о современных способах реабилитации.

* 1. Требования к программе

Любой браузер, поддерживающий JavaScript, доступ в Интернет, Microsoft Visual Studio Code с установленным Vite (для запуска сайта на локальной машине)

* 1. Обоснование проектных решений
     1. Обоснование выбора языков программирования

HTML — единственный язык разметки, который все браузеры понимают без дополнительных преобразований.

CSS — позволяет реализовать любой визуальный стиль (анимации, сетки, тени, градиенты).

JavaScript — позволяет создавать динамические веб-страницы, поддерживает современных фреймворки, например, React, для сложных интерфейсов.

Bash — командная оболочка для написания скриптов, автоматизирующих рутинные процессы, например, создание директорий с необходимыми файлами.

* + 1. Инструментальные средства

React.JS — библиотека для JavaScript, удобный компонентный подход, использование виртуального DOM’а.

React-Bootstrap —  это библиотека React-компонентов, подходящих для повторного использования, которая реализует возможности популярного шаблона Bootstrap от Twitter

Vite — локальный сервер разработки для JavaScript-программистов. Используется для настройки среды разработки и сборки проекта.

* + 1. Обоснование выбора среды программирования

Microsoft Visual Studio Code — текстовый редактор от Microsoft, поддерживающий установку расширений, трансформирующих его в полноценную среду разработки web-приложений. Удобство работы со встроенной консолью, поддерживающую powershell и ubuntu.

* + 1. Информационное обеспечение

GitHub — сервис от Microsoft для облачного хранения файлов исходного кода, контроля версий и деплоя, в моём случае для вебсайта.  
 WPS Office — офисный пакет, созданный китайским разработчиком программного обеспечения Kingsoft Corporation как бесплатная альтернатива Microsoft Office.

Draw.io — это бесплатное кросс-платформенное приложение для создания диаграмм (блок-схемы, UML, сетевые схемы, архитектурные диаграммы и др.). Доступно в онлайн-версии и как оффлайн-приложение.

* 1. Обзор и анализ существующих программных систем

Аналогом проектируемого пользовательского интерфейса является интерфейс такого сайта, как Sermo.

Sermo — это социальная сеть и профессиональная платформа для врачей, где они могут общаться, обсуждать клинические случаи, проводить опросы и получать экспертные мнения коллег. Доступ к сайту ограничен только для сертифицированных медицинских специалистов.

Сайт обеспечивает такие функциональные возможности, как: лента новостей — врачи публикуют вопросы, делятся опытом, обсуждают сложные случаи, доступны комментарии и реакции — можно комментировать посты, ставить "лайки" и участвовать в дискуссиях, реализованы хэштеги и категории — удобная фильтрация по темам (кардиология, онкология, педиатрия и др.). Ключевой механикой сайта является case consultations — возможность получить совет по сложному диагнозу от специалистов со всего мира, кроме того существуют анонимные обсуждения — можно публиковать случаи без указания пациентских данных.

Важной функциональностью выступают актуальные статьи — подборки новостей медицины, клинические рекомендации, проводятся вебинары и CME (Continuing Medical Education) — образовательные мероприятия для врачей. Функционируют платные опросы — участие в маркетинговых исследованиях фармкомпаний.

* 1. Выводы предпроектного исследования

**Целью разработки клиентской части является создание удобного адаптивного интерфейса для вебсайта «socbook». Который позволит врачам-реабилитологам, их пациентам и организациям, заинтересованных в реабилитации, взаимодействовать друг с другом, делиться актуальной информацией, делиться опытом. В свою очередь подразумевается, что различные организации будет сотрудничать с врачами, снабжать необходимыми техническими средствами реабилитации.**

1. **Теоретическая часть**
   1. **История научных исследований по выбранной теме**

**Разработка цифровых платформ для медицинской реабилитации и взаимодействия врачей с пациентами имеет глубокие корни в исследованиях по телемедицине, человеко-компьютерному взаимодействию (HCI) и веб-разработке. Можно выделить несколько ключевых этапов:**

**Ранние исследования в телемедицине (1990–2000-е). Первые работы по дистанционному мониторингу пациентов показали потенциал телекоммуникаций в медицине. Появились специализированные медицинские форумы и базы знаний, но они имели ограниченную интерактивность.**

**Развитие веб-приложений для медицины (2010–2015). С распространением AJAX и Single Page Applications (SPA) стали появляться более динамичные медицинские платформы. Исследования подтвердили эффективность веб-инструментов для реабилитации, но отметили недостаток персонализации.**

**Эра мобильных и адаптивных интерфейсов (2015–2020). С ростом популярности React.js и других SPA-фреймворков возросла важность UX/UI в медицинских сервисах. Появились первые исследования о роли соцсетей в обмене медицинскими знаниями.**

**Современный этап (2020–н.в.). Акцент на безопасность (GDPR, HIPAA) и доступность (WCAG 2.1). Гибридные решения, сочетающие веб, мобильные приложения и IoT для реабилитации.**

* 1. **Определение и анализ ключевых терминов**

**Деплой (deploy) — это развёртывание и запуск веб-приложения или сайта в его рабочей среде, то есть на сервере или хостинге. Разработчик загружает приложение, написанное на локальном компьютере, в специальное пространство, из которого оно доступно в интернете.**

**React — JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов. React разрабатывается и поддерживается Facebook, Instagram и сообществом отдельных разработчиков и корпораций. React может использоваться для разработки одностраничных и мобильных приложений.**

**Хуки — это функции, с помощью которых вы можете «подцепиться» к состоянию и методам жизненного цикла React из функциональных компонентов. Хуки не работают внутри классов — они дают вам возможность использовать React без классов.**

**React-элементы — это составляющие блоки React-приложений. Их можно перепутать с более известной концепцией «компонентов», но в отличие от компонента, элемент описывает то, что вы хотите увидеть на экране.**

* 1. **Актуальные взгляды на выбранную проблему**

Современные цифровые решения в медицине требуют специализированных платформ, объединяющих врачей и пациентов. В области реабилитации опорно-двигательного аппарата ключевыми трендами являются:

- развитие телемедицины – онлайн-реабилитация становится альтернативой очным консультациям, но большинство сервисов не обеспечивают полноценного взаимодействия между специалистами и пациентами;

- персонализация интерфейсов – врачам нужны аналитические инструменты, а пациентам – простые способы контроля прогресса. Современные технологии (например, React.js) позволяют создавать адаптивные и удобные интерфейсы;

- доступность и безопасность – соответствие стандартам инклюзивности (WCAG) и защита данных критически важны для медицинских платформ.

Разработка клиентской части на React.js отвечает этим требованиям, предлагая гибкое, производительное и безопасное решение для компании «Актив».

1. Практическая часть
   1. Анализ задачи

Клиентская часть представляет собой динамический сайт, соцсеть для врачей-реабилитологов и производителей ТСР, написанный на JavaScript, с использованием React. В шапке сайта должны быть ссылка для перехода на страницы: «Специалисты», «Производители», «Организации», «Общественные организации». Необходимо разработать боковое меню с возможностью перемещения на соответствующие разделы: «Главная», «Мой профиль», «Лента новостей», «Мессенджер», «Друзья», «Новая запись», «Настройки». Сверстать содержимое и подготовить к подключению к API вышеперечисленных разделов сайта.

* 1. Описание логической структуры
     1. Алгоритм программы

Переход между страницами происходит посредством нажатия на соответствующие кнопки в текущем разделе.

При использовании этого фреймворка сайт является не простым набором веб-страниц (множеством HTML страниц), а одним HTML файлом, внутри которого средствами JavaScript реализуется иллюзия механики перехода на разные страницы динамическим изменением отображаемого содержимого, данный функционал возможен благодаря ReactRouter.

Реализована минимальная обработка ошибок при вводе данных на страницах: «Регистрация», «Вход», «Новая запись», «Комментарии». Остальные страницы не имею полей для ввода данных, за исключением «Мессенджера», в котором должна быть реализована передача любых, безопасных, сообщений, любых символов и знаков.

Код компонентов на данном этапе, для демонстрации функционала, использует в качестве базы данных Session Storage, встроенный в браузер. Передача данных из сессионного хранилища реализована таким образом, чтобы в дальнейшем, при деплое и продакшене, подключение к API, разрабатываемому другой частью команды, было минимально затратным по времени. Данные передаются в виде массива объектов, с соответствующими полями, что в дальнейшем будет получатся от API в виде JSON, объекта из объектов, такой формат легко преобразовывается в целевой вид, а именно массив объектов.

На данном этапе разработки любые запросы, в частности поиск по сайту, регистрация пользователей, написание сообщений, реализовано только визуально, сообщения об успешном выполнении каких-либо действий выводится непосредственно на экран.

Блок-схемы некоторые алгоритмов, реализованных в ходе выполнения проекта, представлены в Приложении В. Листинги файлов исходного кода представлены в Приложении Г.

* + 1. Составные части программы и связи между ними

Программа, сайт, состоит из так называемых React-компонентов, они в свою очередь представляют из себя JavaScript функции, в которых описана функциональная логика, используемые данные и визуальное представление (верстка) элемента. Компоненты взаимодействую между собой в следующей форме.

Реализовано три группы компонентов: PhotoComponents, VideoComponents, PostComponents. Эти группы являются основой для интерфейса сайта, каждая из них состоит из следующих обязательных компонентов: Element – базовый компонент представляющий фото, видео или публикацию с соответствующим содержимым, ElementRow – ряд из Element и ElementSlider – слайдер из Element. Эти компоненты используются повсеместно по сайту, на личных станицах пользователей, в ленте новостей, на страница организаций.

Сверстаны такие разделы сайта, как: Profile, Messenger, Friends, Register, Login, Feed, Main. Элементы интерфейса: Header, Sidebar, Footer.

Все вышеперечисленные компоненты используются в корневом компоненте по название App. В этом компоненте использован ReactRouter для маршрутизации по страницам сайта без перезагрузки страницы. Такие элементы, как: Header, Sidebar и Footer, находятся вне ReactRouter, то есть они отображаются на любой открытой странице, как должно происходить по техническому заданию.

* 1. Разработка программы для работы
     1. Входные и выходные данные

На данном этапе разработки сайта входные и выходные данные определены следующим образом: на вход сайта подаются массивы объектов, представляющих из себя пары «ключ - значение», например, «path: photoURL», это пример объекта для отображения фотографии на страницах сайта.

В профилях пользователей предусмотрена возможность публикации статей, в этом случае входные данные имеют вид: «labelPath: название публиции, textPath: текст публикации, photoPath: фотография к публикации». На текущий момент фотографии получаются по ссылке к стоковым сервисам, а именно Unsplash, для получения различных фотографий, имитирующих реальные публикации и статьи, и XSGames, для получения фотографий профилей.

В программе для демонстрации функционала заранее определены пользователи, данные о них также хранятся в массивах объектов.

На выход из алгоритмов получем элементы пользовательского интерфейса, отображающие переданные данные, такие как фотографии и текст, в предусмотренном виде. Из этих элементов составляется клиентская часть соцсети «socbook».

* + 1. Используемые методы

Во время работы над проектом было использовано множество методов, позволяющих достаточно быстро и качественно реализовать задуманную идею.

Для создания клиентской части был выбран следующий стек технологий: HTML, CSS, JavaScript, React, React-Bootstrap, Vite.

Первые три технологии являются базовыми для создания веб-страниц, HTML представляет из себя язык гипертекствой разметки, создающий «скелет» будущего сайта, после, для добавления приятной визуальной составляющей присоединяется CSS, это каскадная таблица стилей для элементов размеченной страницы, позволяет добавлять красивое оформление и анимации.

JavaScript представляет из себя язык программирования, «оживляющий» свёрстанную веб-страницу, позволяя добавлять динамически изменяемое содержимое. React – фреймворк, написанный на JavaScript, очень удобен для создания крупных приложений, таких как данный проект, поддерживает компонентный подход, что очень актуально для работы в команде, где каждый занимается разработкой отдельных частей интерфейса.

Для упрощённой разработки пользовательских интерфейсов была использована библиотека React-Bootstrap. Эта библиотека представляет из себя набор готовых базовых компонентов для написания вёрстки страницы. Является адаптацией популярной библиоткеи «Bootstrap» под фреймворй React, оптимизированной за счёт возможности отдельного подключения требуемых компонентов, а не всей библиотеки, как реализовано в оригинальной версии. Vite служит системой сборки и сервером для React-приложений, удобен для разработки и локальной отладки разрабатываемых сайтов.

* + 1. Тестирование

Выполнено 5 тест-кейсов и 28 модульных тестов, результаты представлены в Приложении Б. Для написания модульных тестов была использована библиотека Vitest.

* + 1. Эксплуатационные документы – руководство пользователя
       1. Назначение программы

Разработка клиентской части производилась для компании «Актив». Программа представляет собой полноценный интерфейс для соцсети «socbook». Целевой аудиторией являются врачи-реабилитологи, производители технических средств реабилитации, пациентов с нарушениями опорно-двигательного аппарата, желающих получить актуальную достоверную информацию по интересующему вопросу, прочитав публичные статьи зарегистрированных врачей или организаций, как коммерческих, так и общественных.

* + - 1. Условия выполнения

Для выполнения разработанного frontend’а необходим браузер с поддержкой JavaScript. Доступ к итоговому интерфейсу можно получить по ссылке, по которой уже развернут готовый сайт, или запуск проекта локально, для этого необходим любой дистрибутив операционной системы Linux, поддерживающий установку Node.JS.

* + - 1. Выполнение программы

Для доступа по ссылке необходимо чтобы кто-либо заранее развернул сайт на хостинге. Для локального запуска необходимо установить Node.JS, при помощи команды «sudo apt-get install -y nodejs», после чего можно произвести запуск локального сервера Vite при помощи команды «npm run dev».

Для сборки проекта в привычный вид, состоящий из группы трёх файлов расширений: html, css, js, необходимо ввести команду «npm run build». После чего проект соберётся в папку «dist», и появится возможность запуска проекта, просто открывая файл index.html.

* 1. Сравнительная характеристика работы

Аналогом разрабатываемого пользовательского интерфейса является сайт Sermo. Sermo представляет собой социальную сеть и профессиональную платформу для врачей, где они могут обмениваться мнениями, обсуждать клинические случаи, проводить опросы и получать экспертные советы от коллег. Доступ к платформе ограничен только для сертифицированных медицинских специалистов.

Сайт предлагает такие функции, как лента новостей, где врачи могут задавать вопросы, делиться опытом и обсуждать сложные случаи. Пользователи могут комментировать посты, ставить "лайки" и участвовать в дискуссиях. Реализованы хэштеги и категории, что позволяет удобно фильтровать контент по темам. Ключевой особенностью платформы являются консультации по клиническим случаям, которые позволяют получить советы по сложным диагнозам от специалистов со всего мира. Также предусмотрены анонимные обсуждения, где можно делиться случаями без указания данных пациентов.

Кроме того, важной частью функционала являются актуальные статьи, включающие подборки новостей медицины и клинические рекомендации, а также проводимые вебинары и мероприятия по непрерывному медицинскому образованию (CME). Также доступны платные опросы, в которых врачи могут участвовать в маркетинговых исследованиях фармацевтических компаний.

Скриншоты реализованного интерфейса представлены в Приложении А.

Заключение

В рамках данного курсового проекта была разработана клиентская часть вебсайта **«socbook»** для компании **«Актив»**, направленная на создание удобной цифровой платформы для взаимодействия врачей-реабилитологов, пациентов с нарушениями опорно-двигательного аппарата и заинтересованных организаций.

Проект решает актуальную проблему отсутствия специализированных сервисов, объединяющих медицинских специалистов и пациентов в едином цифровом пространстве.

Разработанный сайт успешно решает поставленные задачи и может быть использован в качестве основы для дальнейшего развития платформы, включая расширение функционала (например, добавление видеоконсультаций, интеграцию с медицинскими устройствами).

Проект демонстрирует применение современных веб-технологий в социально значимой сфере – медицине и реабилитации, что делает его перспективным как с практической, так и с научной точки зрения.

Были освоены базовые навыки разработки web-прилоежний при помощи библиотеки React со сборщиком Vite, и основы создания модульных тестов для написанного кода с использованием библиотеки Vitest.

Используемые источники

1. Bashshur R. L., Shannon G. W. History of telemedicine: Evolution, context, and transformation. — New York: Mary Ann Liebert, Inc., 1997. — 300 p.
2. Bin Uzayr S., Cloud N., Ambler T. JavaScript Frameworks for Modern Web Development. – Apress, 2019.
3. Kvedar J., Coye M. J., Everett W. Connected health: A review of technologies and strategies to improve patient care with telemedicine and telehealth // Health Affairs. — 2014. — Vol. 33, № 2. — P. 194–199.
4. Zhang Y., Liu H. User experience design in medical applications: A literature review // Journal of Medical Systems. — 2018. — Vol. 42, № 6. — P. 1–10.
5. Lupton D. Digital health: Critical and cross-disciplinary perspectives. — London: Routledge, 2022. — 210 p.
6. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. — 4-е изд. — СПб.: Питер, 2016. — 768 с. — (Серия «Бестселлеры O’Reilly»).
7. React.JS: документация [Электронный ресурс]. — URL: https://react.dev/ (дата обращения: 15.04.2025).
8. React-Bootstrap: документация [Электронный ресурс]. — URL: https://react-bootstrap.netlify.app/ (дата обращения: 17.04.2025).
9. Vercel: документация [Электронный ресурс]. — URL: https://vercel.com/docs/getting-started-with-vercel/projects-deployments (дата обращения: 26.04.2025).
10. Библиотеки для React: обзор [Электронный ресурс] // Habr. — 2023. — URL: https://habr.com/ru/companies/ruvds/articles/346090/ (дата обращения: 16.04.2025).

Приложение А  
(обязательное)

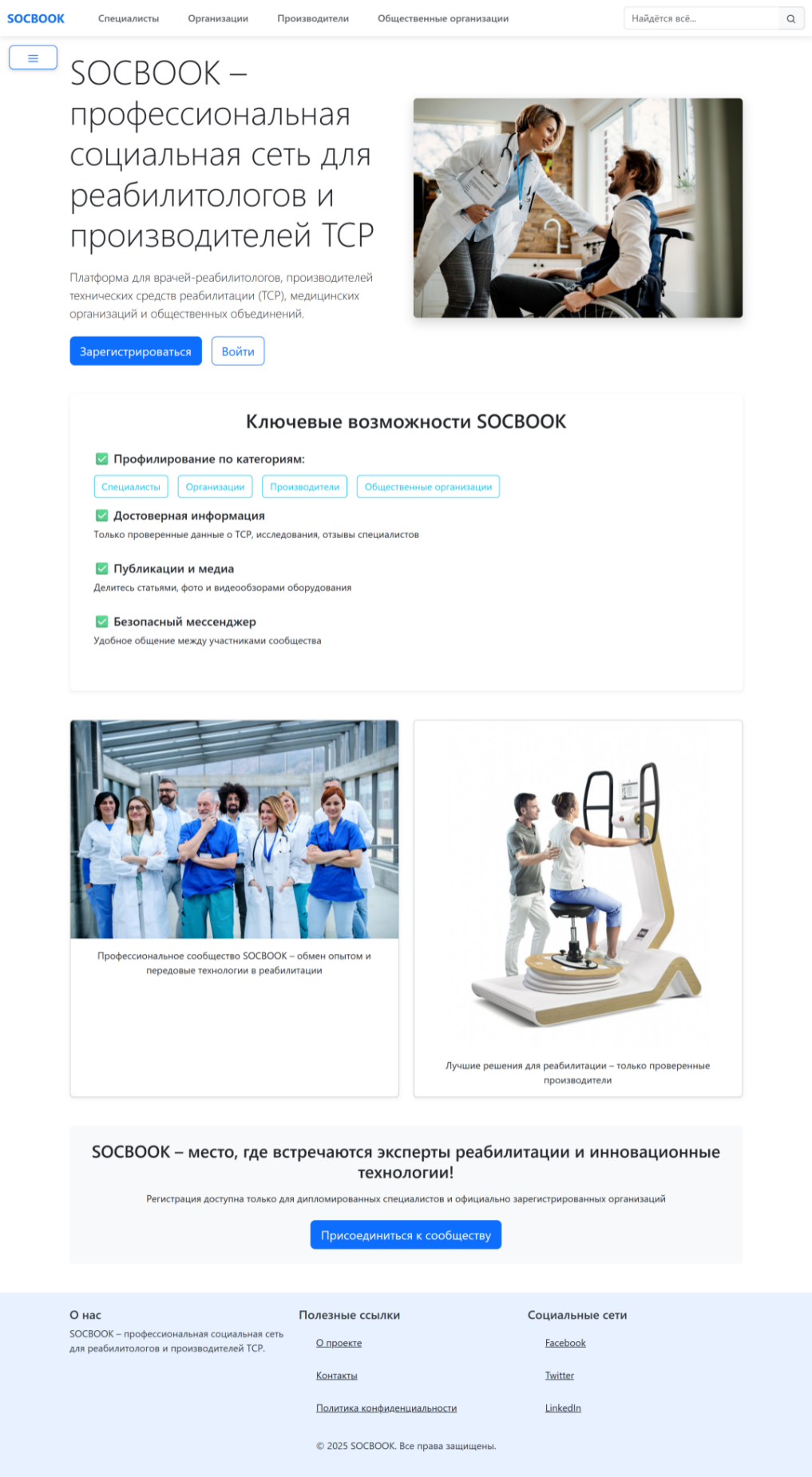


Рисунок А.1 — Главная страница

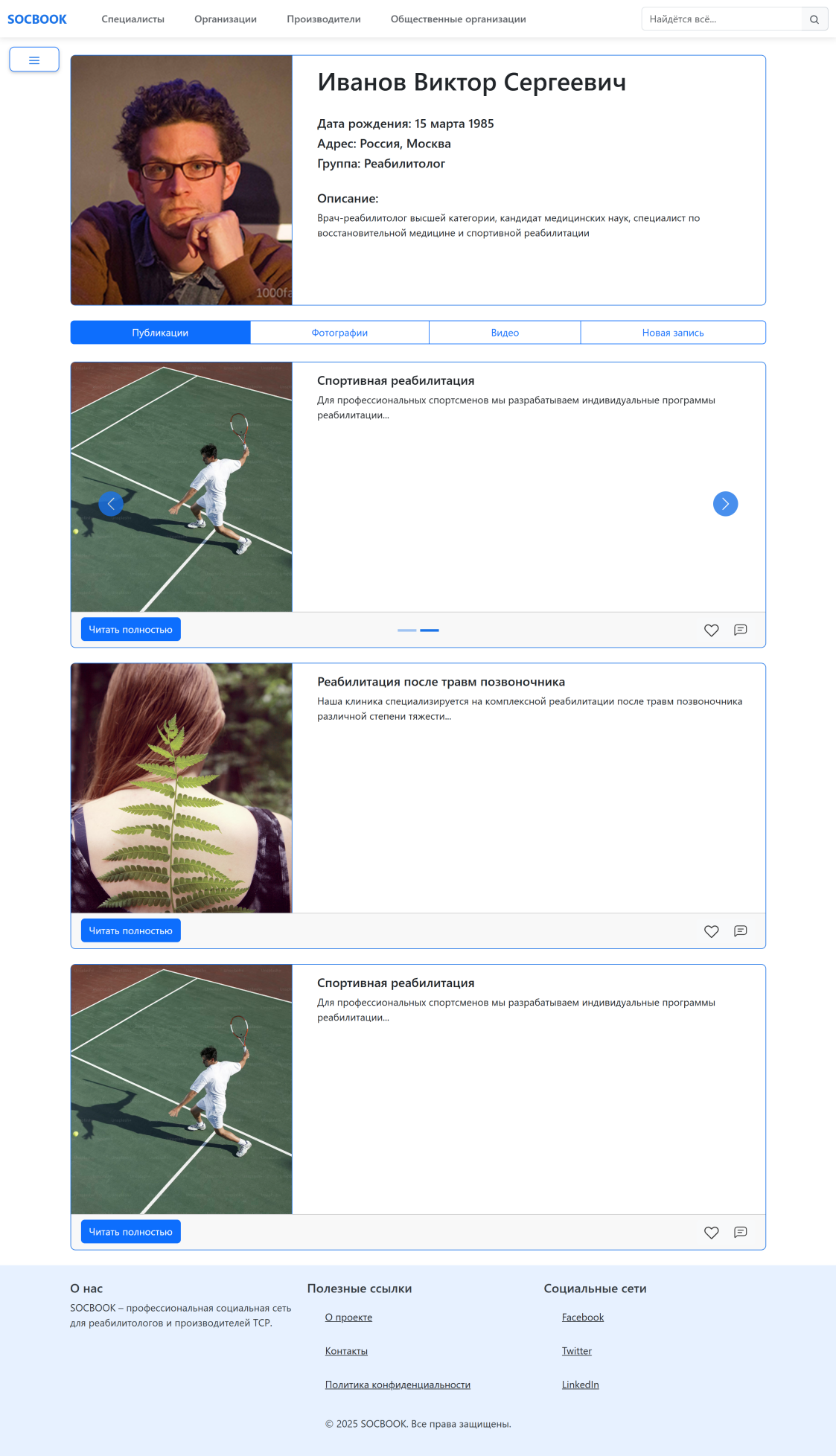


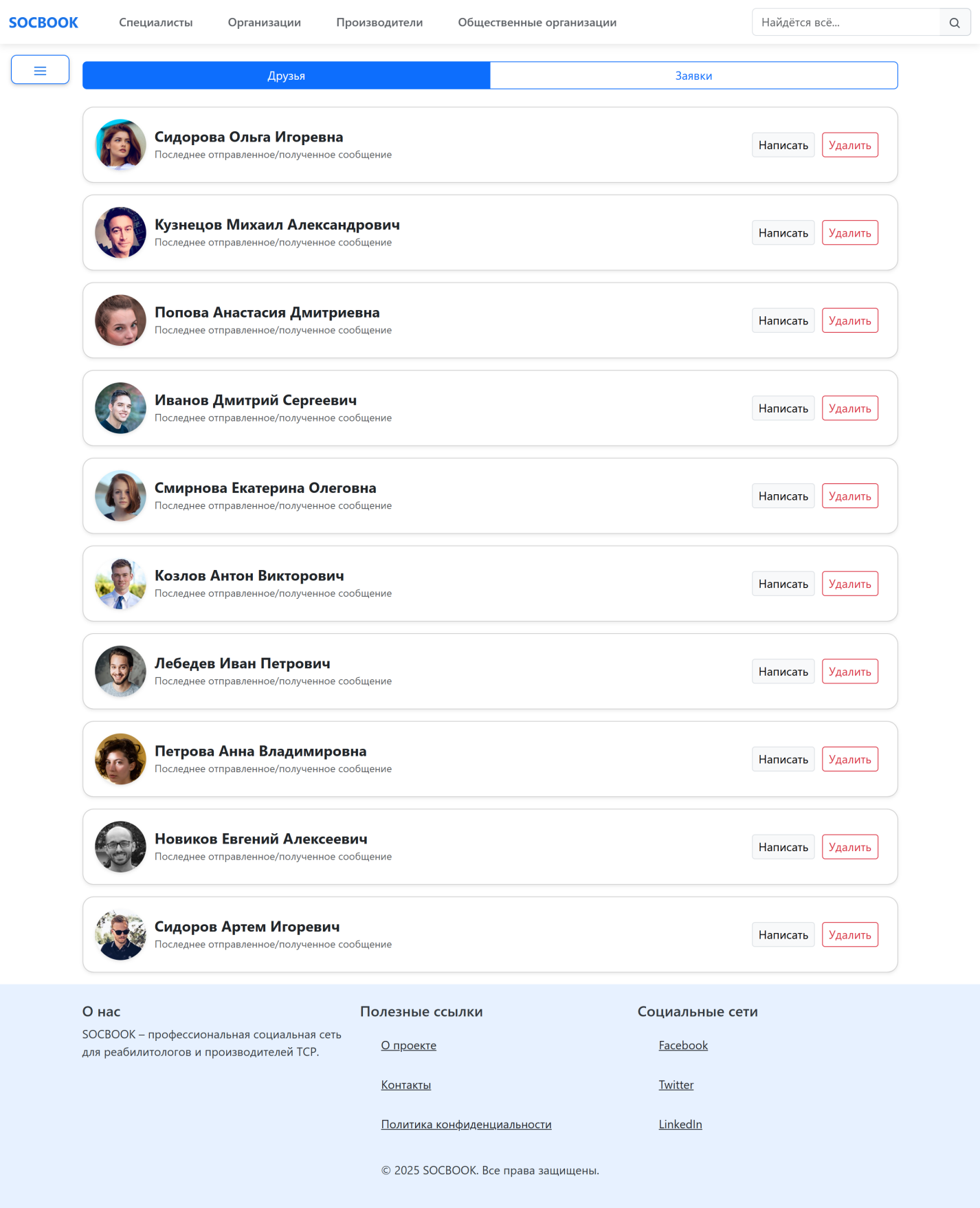
Рисунок А.2 — Мой профиль

Рисунок А.3 — Друзья

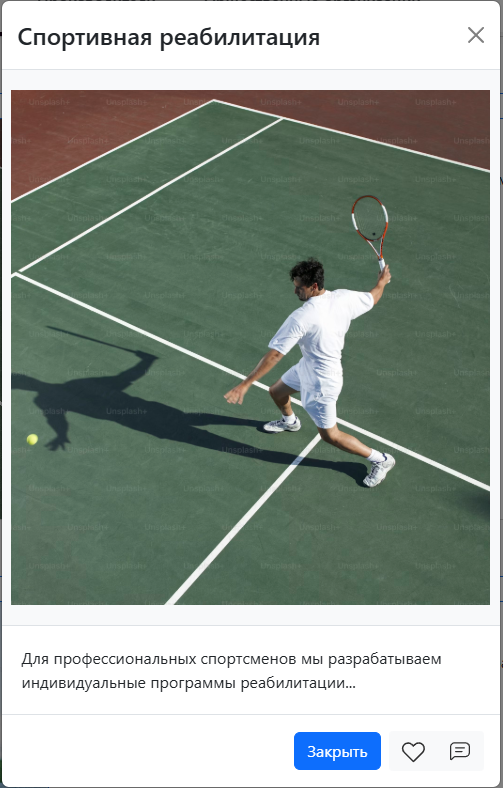


Рисунок А.4 — Модальное окно

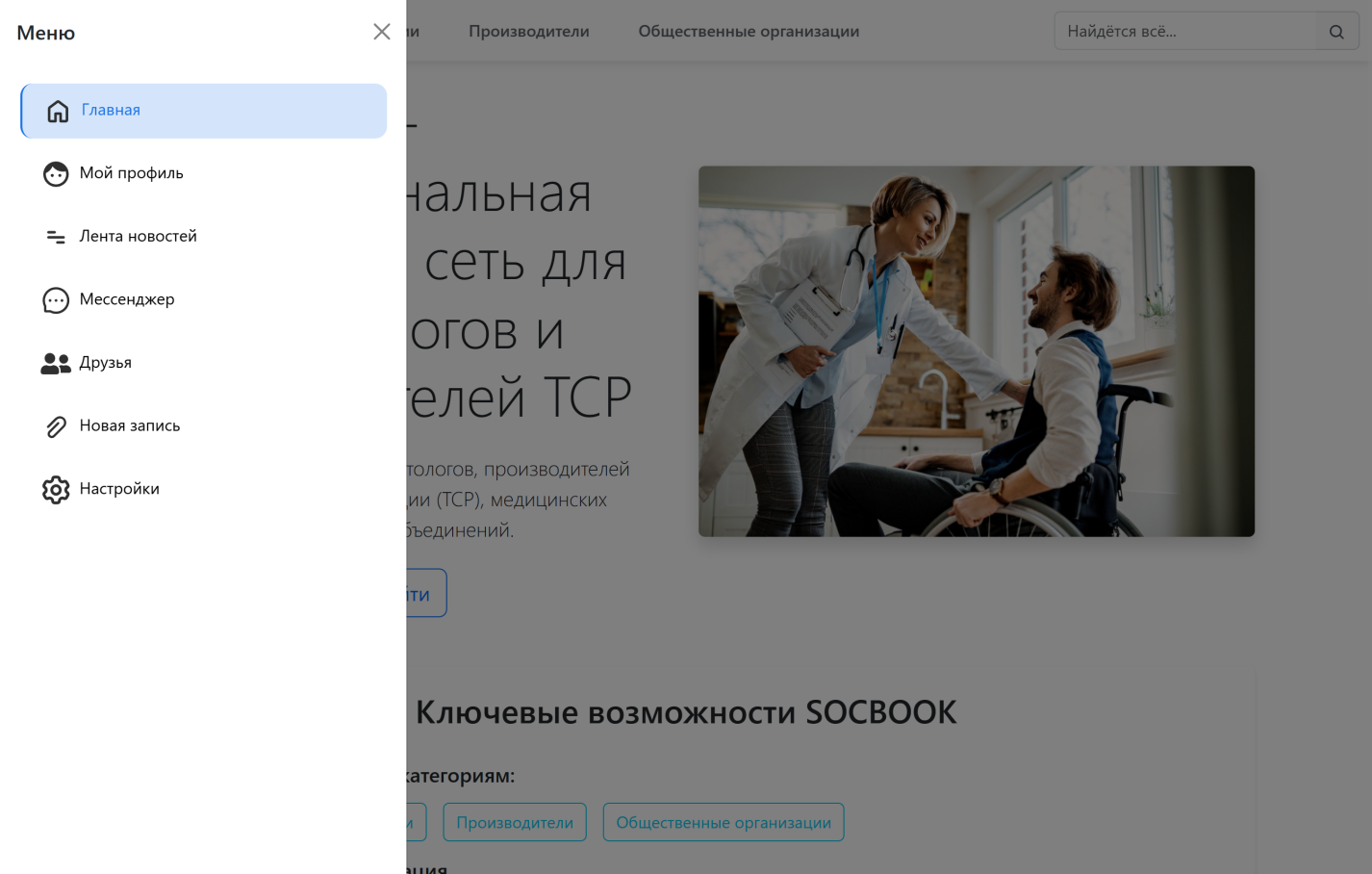


Рисунок А.5 — Боковое меню

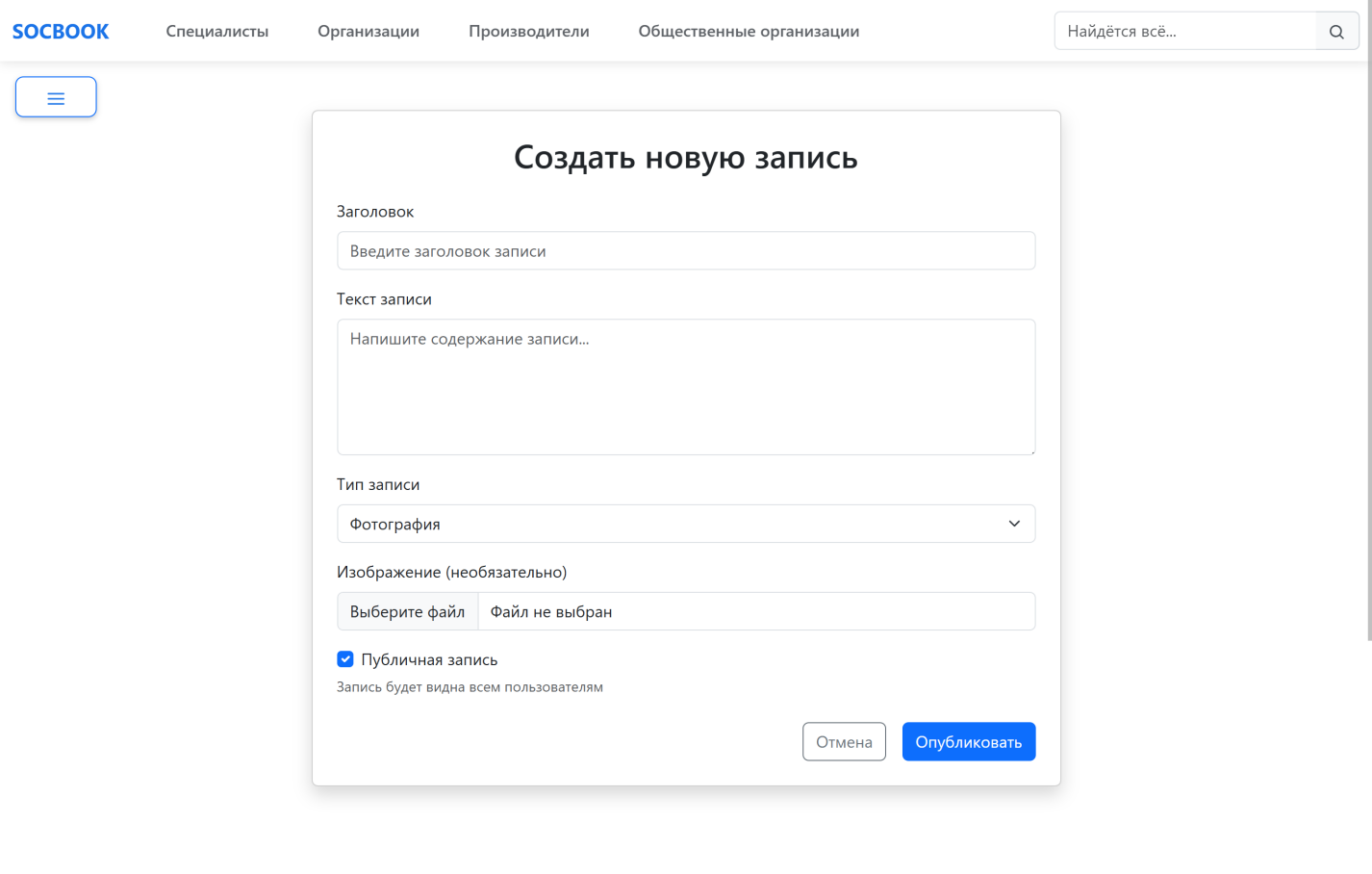


Рисунок А.6 — Новая запись

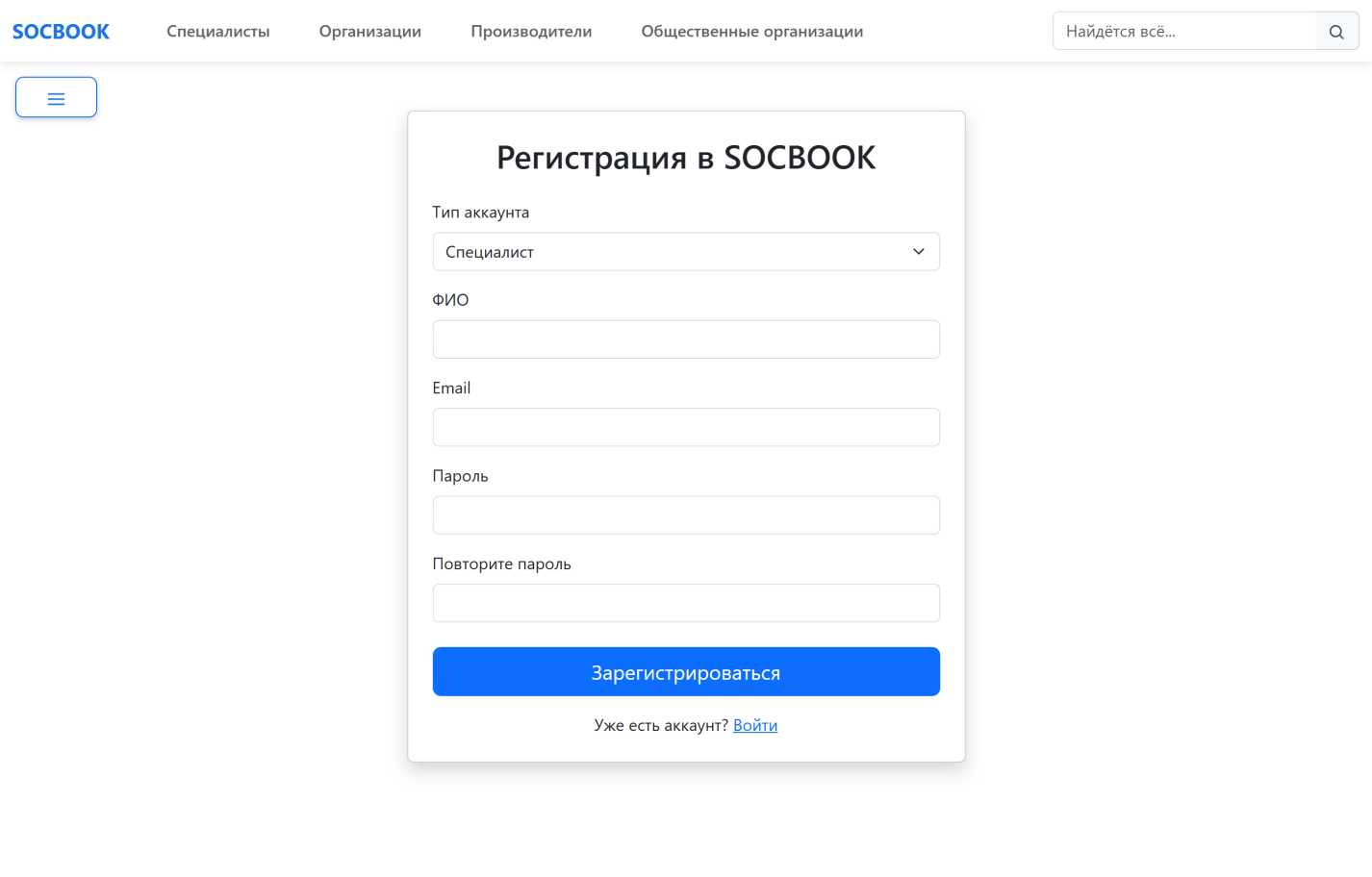


Рисунок А.7 — Регистрация

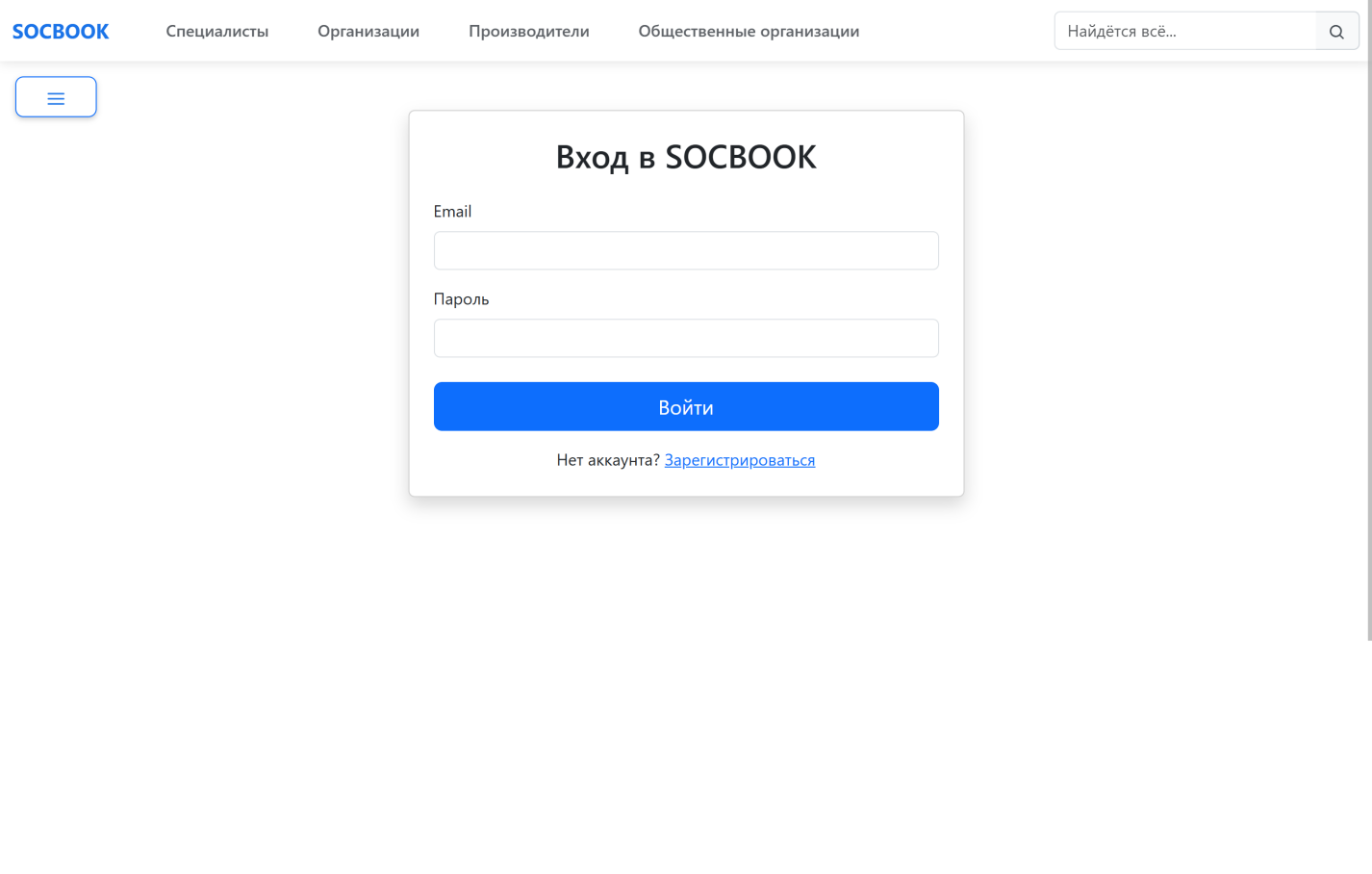


Рисунок А.8 — Вход

Приложение Б  
(обязательное)

Таблица Б.1 — Тест-кейс, кросс-браузерное и кросс-платформенное тестирование интерфейса

|  |  |  |
| --- | --- | --- |
| ID тест-кейса/Приоритет | TU01 | 1 |
| Идея: Кросс-браузерное и кросс-платформенное тестирование интерфейса  Входные параметры: — | | |
| История редактирования | | |
| Создан 27.04.2025 | Новый тест кейс | |
| Предусловие | | |
| Установлены браузеры: Google Chrome, Mozilla Firefox, Yandex Browser | | |
| Выполняемая часть | | |
| Список команд | Ожидаемый результат | Результат теста |
| 1. Открытие сайта в Google Chrome на Windows 2. Открытие сайта в Google Chrome на Ubuntu 3. Открытие сайта в Mozilla Firefox на Windows 4. Открытие сайта в Mozillf Firefox на Ubuntu 5. Открытие сайта в Yandex Browser | Интерфейс отображается корректно в разных браузерах на разных платформах | Pass |

Таблица Б.2 — Тест-кейс, проверка переходов между страницами

|  |  |  |
| --- | --- | --- |
| ID тест-кейса/Приоритет | TU02 | 2 |
| Идея: Проверка переходов между страницами  Входные параметры: — | | |
| История редактирования | | |
| Создан 27.04.2025 | Новый тест кейс | |
| Предусловие | | |
| Открытая главная страница сайта | | |
| Выполнимая часть | | |
| Список команд | Ожидаемый результат | Результат теста |
| 1. Нажать на копку «Меню» 2. Выбрать пункт «Мой профиль» 3. Нажать кнопку «Фотографии» 4. Выбрать пункт «Новая запись» 5. Нажать на логотип сайта на шапке сайте | Были совершены переходы на верные страницы | Pass |

Таблица Б.3 — Тест-кейс, проверка устойчивости к XSS-уязвимости

|  |  |  |
| --- | --- | --- |
| ID тест-кейса/Приоритет | TU03 | 3 |
| Идея: Проверка устойчивости к XSS-уязвимости  Входные параметры: <script>alert(1)</script> | | |
| История редактирования | | |
| Создан 27.04.2025 | Новый тест кейс | |
| Предусловие | | |
| Открытая главная страница сайта | | |
| Выполнимая часть | | |
| Список команд | Ожидаемый результат | Результат теста |
| 1. Нажать на копку «Меню» 2. Выбрать пункт «Мессенджер» 3. Нажать на имя любого пользователя 4. Ввести тестовые данные и отправить | Уязвимость не обнаружена | Pass |

Таблица Б.4 — Тест-кейс, проверка корректной работы модальных окон

|  |  |  |
| --- | --- | --- |
| ID тест-кейса/Приоритет | TU04 | 4 |
| Идея: Проверка корректной работы модальных окон  Входные параметры: — | | |
| История редактирования | | |
| Создан 27.04.2025 | Новый тест кейс | |
| Предусловие | | |
| Открыта главная страница сайта | | |
| Выполнимая часть | | |
| Список команд | Ожидаемый результат | Результат теста |
| 1. Нажать на копку «Меню» 2. Выбрать пункт «Друзья» 3. Нажать на имя любого пользователя 4. Нажать на кнопку «Читать подробнее» на любой публикации 5. Нажать на копку «Закрыть модальное окно» вверху открытого модального окна | Модальное окно успешно закрыто | Pass |

Таблица Б.5 — Тест-кейс, проверка обработки ошибок при вводе данных при регистрации

|  |  |  |
| --- | --- | --- |
| ID тест-кейса/Приоритет | TU05 | 5 |
| Идея: Проверка обработки ошибок при вводе данных при регистрации  Входные параметры: | | |
| История редактирования | | |
| Создан 27.04.2025 | Новый тест кейс | |
| Предусловие | | |
| Открыта главная страница сайта | | |
| Выполнимая часть | | |
| Список команд | Ожидаемый результат | Результат теста |
| 1. Нажать на кнопку «Зарегистрироваться» 2. Ввод в поле для ввода электронной почты email из тестовых данны 3. Ввод в поле для ввода пароля password из тестовых данных | Корректная обработка неверных данных | Pass |

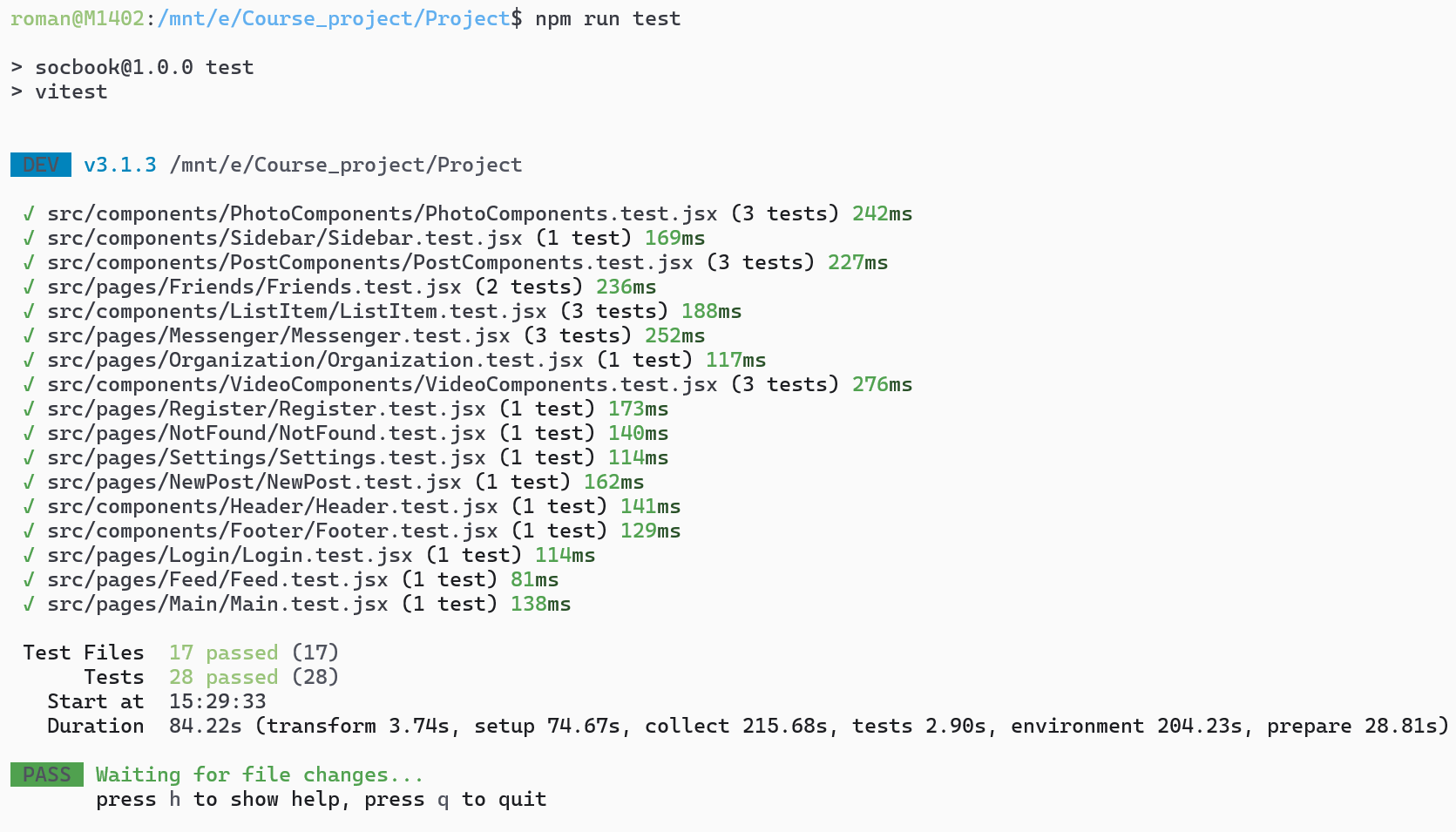


Рисунок Б.1 — Скриншот результатов модульных тестов  
(все тесты пройдены)

Приложение В  
(обязательное)



Рисунок В.1 — Блок-схема перемещения объекта внутри сессионного хранилища



Рисунок В.2 — Блок-схема удаления объекта из списка запросов в друзья

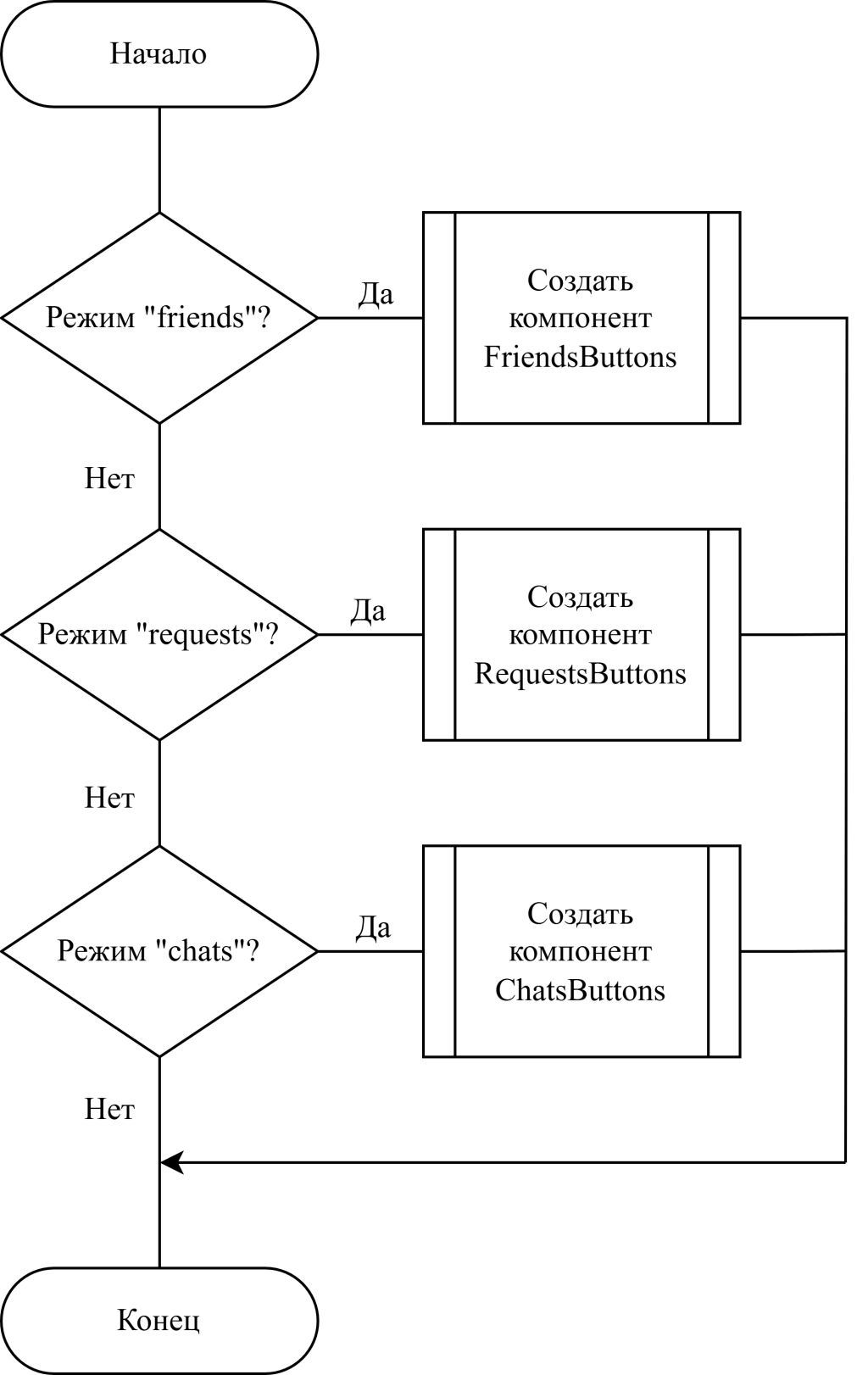


Рисунок В.3 — Блок-схема выбора подходящего компонента в зависимости от режима

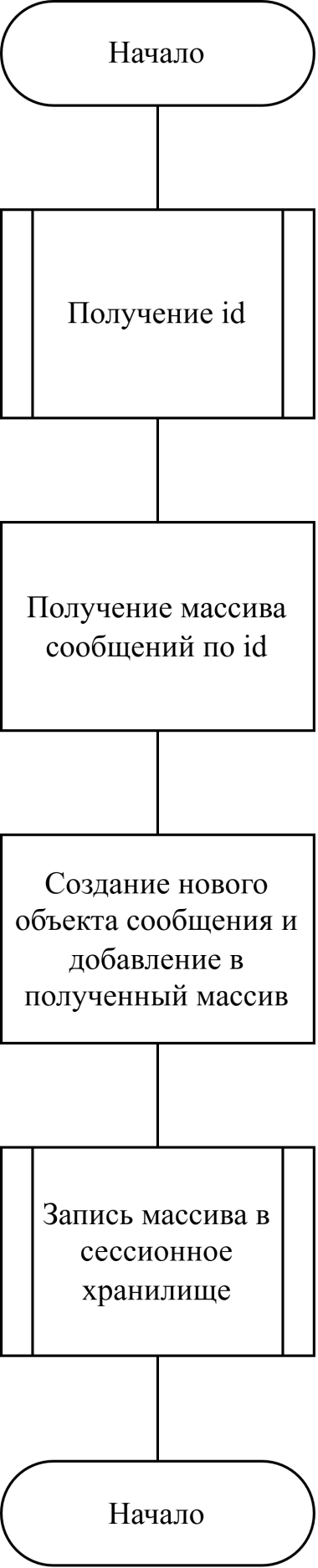


Рисунок В.4 — Блок-схема визуализации отправки сообщения в мессенджере

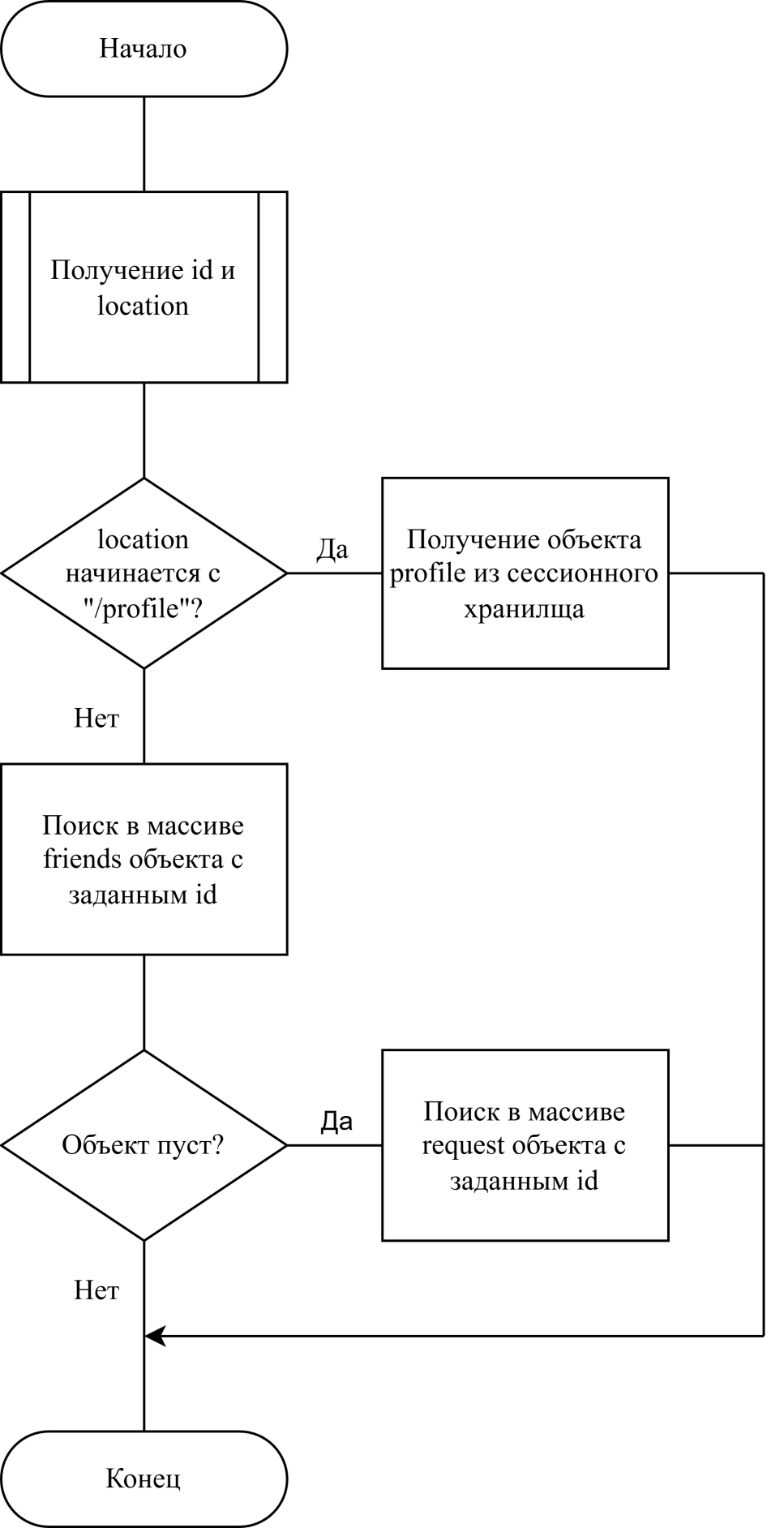


Рисунок В.5 — Блок-схема генерации отображаемых элементов в компоненте Profile



Рисунок В.6 — Блок-схема создания слайдера по переданному массиву объектов



Рисунок В.7 — Блок-схема установки значений в сессионное хранилище

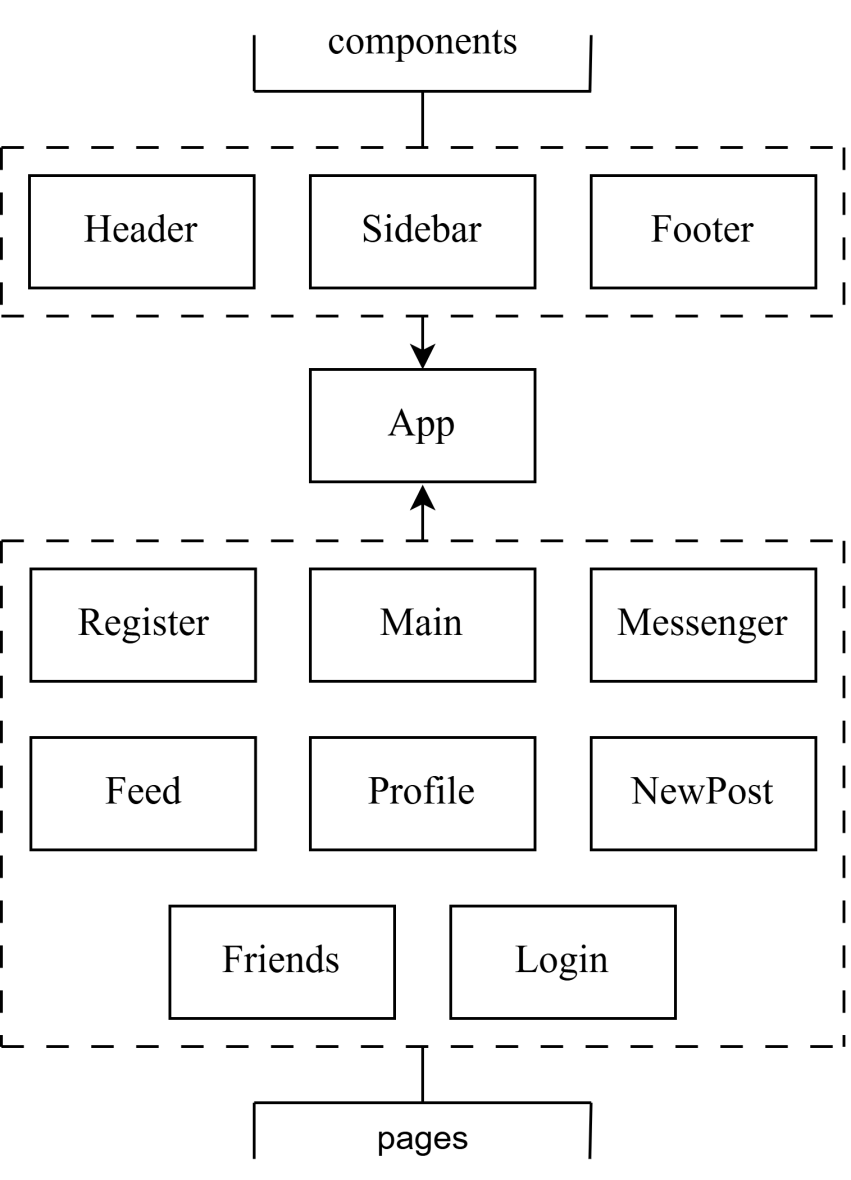


Рисунок В.8 — Структура приложения

Приложение Г  
(обязательное)

Листинг Г.1 — код файла index.html

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<link rel="icon" type="image/svg+xml" href="/logo.svg" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>SOCBOOK</title>

</head>

<body>

<div id="root"></div>

<script type="module" src="/src/App.jsx"></script>

</body>

</html>

Листинг Г.2 — код файла App.jsx

import { StrictMode } from 'react'

import { createRoot } from 'react-dom/client'

import 'bootstrap/dist/css/bootstrap.min.css';

import styled from 'styled-components';

import { BrowserRouter, Routes, Route } from 'react-router-dom';

import SetData from './SetData';

import Sidebar from './Sidebar';

import Header from './Header'

import Footer from './Footer';

import Settings from './Settings';

import Main from './Main'

import Profile from './Profile'

import Feed from './Feed'

import Messenger from './Messenger'

import Friends from './Friends'

import Organization from './Organization'

import NotFound from "./NotFound";

export const ContentWrapper = styled.div`

margin-top: 60px;

min-height: calc(100vh - 60px);`;

function App() {

return (

<BrowserRouter>

<Header />

<Sidebar />

<SetData />

<ContentWrapper>

<Routes>

<Route path="/" element={<Main />} />

<Route path="/profile/photos" element={<Profile />} />

<Route path="/profile/videos" element={<Profile />} />

<Route path="/profile/posts" element={<Profile />} />

<Route path="/settings" element={<Settings />} />

<Route path="/:id/photos" element={<Profile />} />

<Route path="/:id/videos" element={<Profile />} />

<Route path="/:id/posts" element={<Profile />} />

<Route path="/feed" element={<Feed />} />

<Route path="/messenger" element={<Messenger />} />

<Route path="/messenger/chat/:id" element={<Messenger />} />

<Route path="/friends" element={<Friends />} />

<Route path="/friends/requests" element={<Friends />} />

<Route path="/manufacturers" element={<Organization />} />

<Route path="/public\_organizations" element={<Organization />} />

<Route path="/organizations" element={<Organization />} />

<Route path="/specialists" element={<Organization />} />

<Route path="\*" element={<NotFound />} />

</Routes>

</ContentWrapper>

<Footer />

</BrowserRouter>

);

}

createRoot(document.getElementById('root')).render(

<StrictMode>

<App />

</StrictMode>

)

Листинг Г.3 — код файла Header.jsx

import { Navbar, Nav, Form, Button, Container, InputGroup } from 'react-bootstrap';

import { BiSearch } from 'react-icons/bi';

import { Link, useLocation } from 'react-router-dom';

import styled from 'styled-components';

const StyledHeader = styled(Navbar)`

position: fixed;

top: 0;

left: 0;

right: 0;

height: 60px;

background: #fff !important;

box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);

z-index: 1000;

padding: 0;

.navbar-brand {

font-weight: 700;

font-size: 1.25rem;

color: #1a73e8;

margin-right: 2rem;

}

.search-form {

width: 300px;

margin-left: auto;

}

`;

const NavItem = styled(Nav.Link)`

position: relative;

margin: 0 0.25rem;

padding: 0.75rem 1.25rem;

color: #5f6368 !important;

font-weight: 500;

text-decoration: none;

border-radius: 8px;

transition: all 1s ease;

&:hover, &.active {

color: #1a73e8 !important;

background: rgba(26, 115, 232, 0.08);

border-left: 2px solid #1a73e8;

}

`;

const SearchButton = styled(Button)`

border-color: #ced4da;

border-left: none;

background-color: #f8f9fa;

color: #495057;

transition: all 0.2s;

&:hover {

background-color: #e9ecef;

border-color: #ced4da;

color: #495057;

}

`;

export default function Header() {

const location = useLocation();

const navItems = [

{ path: "/specialists", description: "Специалисты" },

{ path: "/organizations", description: "Организации" },

{ path: "/manufacturers", description: "Производители" },

{ path: "/public\_organizations", description: "Общественные организации" }

];

return (

<StyledHeader bg="light" expand="lg" sticky="top">

<Container fluid="xxl">

<Navbar.Brand as={Link} to="/">SOCBOOK</Navbar.Brand>

<Navbar.Toggle aria-controls="navbar" />

<Navbar.Collapse id="navbar">

<Nav className="me-auto">

{navItems.map((item, index) => (

<NavItem

key={index}

as={Link}

to={item.path}

className={location.pathname === item.path ? 'active' : ''}

>

{item.description}

</NavItem>

))}

</Nav>

<Form className="search-form">

<InputGroup>

<Form.Control

type="search"

placeholder="Найдётся всё..."

className="border-end-0"

/>

<SearchButton variant="outline-secondary">

<BiSearch size={18} />

</SearchButton>

</InputGroup>

</Form>

</Navbar.Collapse>

</Container>

</StyledHeader>

);

}

Листинг Г.4 — код файла Footer.jsx

import { Container, Row, Col, ListGroup, Button } from 'react-bootstrap';

import { Link } from 'react-router-dom';

export default function Footer() {

return ( <footer style={{ backgroundColor: '#e7f1ff', color: '#343a40' }} className="py-4">

<Container> <Row>

<Col md={4}>

<h5>О нас</h5>

<p>SOCBOOK – профессиональная социальная сеть для реабилитологов и производителей ТСР.</p>

</Col>

<Col md={4}>

<h5>Полезные ссылки</h5>

<ListGroup variant="flush">

<ListGroup.Item style={{ backgroundColor: '#e7f1ff', border: 'none' }}>

<Button as={Link} to="/about" variant="link" className="text-dark">О проекте</Button>

</ListGroup.Item>

<ListGroup.Item style={{ backgroundColor: '#e7f1ff', border: 'none' }}>

<Button as={Link} to="/contact" variant="link" className="text-dark">Контакты</Button>

</ListGroup.Item>

<ListGroup.Item style={{ backgroundColor: '#e7f1ff', border: 'none' }}>

<Button as={Link} to="/privacy" variant="link" className="text-dark">Политика конфиденциальности</Button>

</ListGroup.Item>

</ListGroup>

</Col>

<Col md={4}>

<h5>Социальные сети</h5>

<ListGroup variant="flush">

<ListGroup.Item style={{ backgroundColor: '#e7f1ff', border: 'none'}}>

<Button as="a" href="https://facebook.com" target="\_blank" variant="link" className="text-dark">Facebook</Button>

</ListGroup.Item>

<ListGroup.Item style={{ backgroundColor: '#e7f1ff', border: 'none' }}>

<Button as="a" href="https://twitter.com" target="\_blank" variant="link" className="text-dark">Twitter</Button>

</ListGroup.Item>

<ListGroup.Item style={{ backgroundColor: '#e7f1ff', border: 'none' }}>

<Button as="a" href="https://linkedin.com" target="\_blank" variant="link" className="text-dark">LinkedIn</Button>

</ListGroup.Item>

</ListGroup>

</Col>

</Row>

<Row className="mt-4">

<Col className="text-center">

<p>&copy; {new Date().getFullYear()} SOCBOOK. Все права защищены.</p>

</Col>

</Row></Container> </footer> );}

Листинг Г.5 — код файла Sidebar.jsx

import { useState } from "react";

import { Link, useLocation } from 'react-router-dom';

import { Nav, Offcanvas, Button, Image} from 'react-bootstrap';

import styled from 'styled-components';

const SidebarContainer = styled.div`

position: fixed;

z-index: 100;

`

const SidebarItem = styled(Nav.Link)`

margin: 0.25rem 0 0.25rem 0.25rem;

padding: 0.75rem 1.25rem;

text-decoration: none;

color: black;

border-radius: 0.75rem;

transition: 1s ease;

width: 360px;

&:hover, &.active {

color: #1a73e8;

background-color: #1a73e830;

border-left: 2px solid #1a73e8;

}

img {

height: 30px;

width: 30px;

margin-right: 0.5rem;

}

`

const MenuButton = styled(Button)`

position: fixed;

top: 75px;

left: 15px;

z-index: 99;

background-color: white;

color: #1a73e8;

border-radius: 8px;

width: 80px;

height: 40px;

box-shadow: 0 2px 5px rgba(0,0,0,0.2);

&:hover {

border: 2px solid #1a73e8;

}

@media (max-width: 800px) {

top: 15px;

}

`

export default function Sidebar() {

const [show, setShow] = useState(false);

const handleShow = () => setShow(true);

const handleClose = () => setShow(false);

const location = useLocation();

const items = [

{ path: "/", description: "Главная", icon: "/main.svg" },

{ path: "/profile/posts", description: "Мой профиль", icon: "/logo.svg" },

{ path: "/feed", description: "Лента новостей", icon: "/feed.svg" },

{ path: "/messenger", description: "Мессенджер", icon: "/messenger.svg" },

{ path: "/friends", description: "Друзья", icon: "/friends.svg" },

{ path: "/new\_post", description: "Новая публикация", icon: "/new\_post.svg" },

{ path: "/settings", description: "Настройки", icon: "/settings.svg" }

];

return (

<>

<MenuButton onClick={handleShow} aria-label="Открыть меню">

<svg width="20" height="20" viewBox="0 0 20 20" fill="none" xmlns="http://www.w3.org/2000/svg">

<path d="M2.5 5H17.5" stroke="#1a73e8" strokeWidth="1.5" strokeLinecap="round"/>

<path d="M2.5 10H17.5" stroke="#1a73e8" strokeWidth="1.5" strokeLinecap="round"/>

<path d="M2.5 15H17.5" stroke="#1a73e8" strokeWidth="1.5" strokeLinecap="round"/>

</svg>

</MenuButton>

<Offcanvas show={show} onHide={handleClose}>

<Offcanvas.Header closeButton>

<Offcanvas.Title>Меню</Offcanvas.Title>

</Offcanvas.Header>

<Offcanvas.Body>

<SidebarContainer>

<Nav className="flex-column">

{items.map((item, index) => (

<SidebarItem

key={index}

as={Link}

to={item.path}

className={location.pathname === item.path ? 'active' : ''}

onClick={handleClose}

>

<img className="sidebar-icon" src={item.icon} alt={item.description} />

<span className="sidebar-text">{item.description}</span>

</SidebarItem>

))}

</Nav>

</SidebarContainer>

</Offcanvas.Body>

</Offcanvas>

</>

);

}

Листинг Г.6 — код файла Main.jsx

import { Container, Row, Col, Card, Button, Image, ListGroup } from 'react-bootstrap';

import { Link } from 'react-router-dom';

export default function Main() {

return (

<Container className="pt-4">

<Row className="mb-5 align-items-center">

<Col md={6}>

<h1 className="display-4 mb-4">SOCBOOK – профессиональная социальная сеть для реабилитологов и производителей ТСР</h1>

<p className="lead">

Платформа для врачей-реабилитологов, производителей технических средств реабилитации (ТСР),

медицинских организаций и общественных объединений.

</p>

<div className="d-flex gap-3 mt-4">

<Button as={Link} to="/register" variant="primary" size="lg">Зарегистрироваться</Button>

<Button as={Link} to="/profile/posts" variant="outline-primary" size="lg">Войти</Button>

</div>

</Col>

<Col md={6}>

<Image src="https://care-place.ru/image/catalog/article/5818.jpg" fluid rounded className="shadow" alt="Врачи-реабилитологи" />

</Col>

</Row>

<Row className="mb-5">

<Col>

<Card className="border-0 shadow-sm">

<Card.Body className="p-4">

<h2 className="mb-4 text-center">Ключевые возможности SOCBOOK</h2>

<ListGroup variant="flush" className="mb-4">

<ListGroup.Item className="border-0">

<h5>✅ Профилирование по категориям:</h5>

<div className="d-flex flex-wrap gap-3 mt-3">

<Button as={Link} to="/specialists" variant="outline-info">Специалисты</Button>

<Button as={Link} to="/organizations" variant="outline-info">Организации</Button>

<Button as={Link} to="/manufacturers" variant="outline-info">Производители</Button>

<Button as={Link} to="/public\_organizations" variant="outline-info">Общественные организации</Button>

</div>

</ListGroup.Item>

<ListGroup.Item className="border-0">

<h5>✅ Достоверная информация</h5>

<p>Только проверенные данные о ТСР, исследования, отзывы специалистов</p>

</ListGroup.Item>

<ListGroup.Item className="border-0">

<h5>✅ Публикации и медиа</h5>

<p>Делитесь статьями, фото и видеообзорами оборудования</p>

</ListGroup.Item>

<ListGroup.Item className="border-0">

<h5>✅ Безопасный мессенджер</h5>

<p>Удобное общение между участниками сообщества</p>

</ListGroup.Item>

</ListGroup>

</Card.Body>

</Card>

</Col>

</Row>

<Row className="mb-5">

<Col md={6} className="mb-4 mb-md-0">

<Card className="h-100 shadow-sm">

<Card.Img variant="top" src="doctors.avif" />

<Card.Body>

<Card.Text className="text-center">

Профессиональное сообщество SOCBOOK – обмен опытом и передовые технологии в реабилитации

</Card.Text>

</Card.Body>

</Card>

</Col>

<Col md={6}>

<Card className="h-100 shadow-sm">

<Card.Img variant="top" src="https://med-ob.ru/image/cache/catalog/easyphoto/8597/sistema-neyromyshechnoy-reabilitatsii-lpg-huber-360-md-10-1200x1200.jpg" />

<Card.Body>

<Card.Text className="text-center">

Лучшие решения для реабилитации – только проверенные производители

</Card.Text>

</Card.Body>

</Card>

</Col>

</Row>

<Row className="text-center mb-5">

<Col>

<div className="p-4 bg-light rounded">

<h3 className="mb-3">SOCBOOK – место, где встречаются эксперты реабилитации и инновационные технологии!</h3>

<p className="mb-4">Регистрация доступна только для дипломированных специалистов и официально зарегистрированных организаций</p>

<Button as={Link} to="/register" variant="primary" size="lg">Присоединиться к сообществу</Button>

</div>

</Col>

</Row>

</Container>

);

}

Листинг Г.7 — код файла NotFound.jsx

import { Container, Row, Col, Card, Button, Image } from 'react-bootstrap';

import { Link } from 'react-router-dom';

export default function NotFound() {

return (

<Container className="py-5">

<Row className="justify-content-center text-center mb-5">

<Col md={8}>

<h1 className="display-1 text-primary mb-4">404</h1>

<h2 className="display-4 mb-4">Страница не найдена</h2>

<p className="lead mb-4">

К сожалению, запрашиваемая вами страница не существует или была перемещена.

Возможно, вы ошиблись при вводе адреса или перешли по неработающей ссылке.

</p>

<div className="d-flex justify-content-center gap-3 mt-4">

<Button as={Link} to="/" variant="primary" size="lg">На главную</Button>

<Button as={Link} to="/profile/posts" variant="outline-primary" size="lg">Войти в аккаунт</Button>

</div>

</Col>

</Row>

<Row className="mb-5">

<Col>

<Card className="border-0 shadow-sm">

<Card.Body className="p-4 text-center">

<h2 className="mb-4">Попробуйте следующие разделы SOCBOOK</h2>

<div className="d-flex flex-wrap justify-content-center gap-3 mt-3">

<Button as={Link} to="/specialists" variant="outline-info">Специалисты</Button>

<Button as={Link} to="/organizations" variant="outline-info">Организации</Button>

<Button as={Link} to="/manufacturers" variant="outline-info">Производители</Button>

<Button as={Link} to="/public\_organizations" variant="outline-info">Общественные организации</Button>

</div>

</Card.Body>

</Card>

</Col>

</Row>

</Container>

);

}

Листинг Г.8 — код файла ListItem.jsx

import { useState } from "react";

import { Button, Col, Image, Modal, Row, Card } from "react-bootstrap";

import { PhotoButtons } from "./PhotoComponents.jsx";

import { Link } from "react-router-dom";

import styled from "styled-components";

const StyledButton = styled(Button)`

line-height: 1.5;

font-size: 1rem;

margin-right: 10px;

`

function moveTo(from, to, id) {

const object = JSON.parse(sessionStorage.getItem(from)).find(p => p.id === id)

const fromResponse = JSON.parse(sessionStorage.getItem(from)) || []

const updatedFrom = fromResponse.filter(item => item.id !== object.id)

sessionStorage.setItem(from, JSON.stringify(updatedFrom))

const toResponse = JSON.parse(sessionStorage.getItem(to)) || []

toResponse.push(object)

sessionStorage.setItem(to, JSON.stringify(toResponse))

// console.log('from: ' + from + ' to: ' + to + ' id: ' + id)

}

function remove({id}) {

const requests = JSON.parse(sessionStorage.getItem('requests'))

requests.splice(id, 1);

sessionStorage.setItem('requests', JSON.stringify(requests))

}

function FriendsButtons({id}) {

return (

<>

<Button

variant="light"

size="sm"

style={{ border: '1px solid #dee2e6', lineHeight: "1.5", fontSize: "1rem", marginRight: "10px" }}

as={Link}

to={`/messenger/chat/${encodeURIComponent(id)}}`}

>

Написать

</Button>

<StyledButton

variant="outline-danger"

size="sm"

onClick={() => moveTo('friends', 'requests', id)}

>

Удалить

</StyledButton>

</>

)

}

function RequestsButtons({id}) {

return (

<>

<StyledButton

variant="outline-success"

size="sm"

onClick={() => moveTo('requests', 'friends', id)}

>

Принять

</StyledButton>

<StyledButton

variant="outline-danger"

size="sm"

onClick={() => remove(id)}

>

Отклонить

</StyledButton>

</>

)

}

function ChatsButtons({id}) {

return (

<>

<Button

variant="light"

as={Link}

to={`/messenger/chat/${encodeURIComponent(id)}`}

size="sm"

style={{ border: '1px solid #dee2e6', lineHeight: "1.5", fontSize: "1rem", marginRight: "10px" }}

>

Написать

</Button>

</>

)

}

function Buttons({mode, id}) {

return (

(mode === 'friends' ? (

<FriendsButtons id={id} />

) : mode === 'requests' ? (

<RequestsButtons id={id} />

) : mode === 'chats' ? (

<ChatsButtons id={id} />

) : null)

)

}

export default function ListItem({ name, photoPath, id, mode = '' }) {

const [show, setShow] = useState(false);

const handleClose = () => setShow(false);

const handleShow = () => setShow(true);

return (

<Card className="mb-3 shadow-sm" style={{ borderRadius: '15px'}}>

<Card.Body className="p-3">

<Row className="align-items-center">

<Col xs="auto">

<Image

onClick={handleShow}

src={photoPath}

roundedCircle

width={70}

height={70}

style={{

cursor: 'pointer',

objectFit: 'cover',

boxShadow: '0 2px 4px rgba(0,0,0,0.1)'

}}

className="hover-zoom"

/>

</Col>

<Col className="ps-0">

<Link

to={mode === 'chats' ? `/messenger/chat/${encodeURIComponent(id)}` :`/${encodeURIComponent(id)}/posts` }

className="text-decoration-none"

style={{ color: '#212529' }}

>

<h5 className="mb-0 fw-bold">{name}</h5>

<small className="text-muted">Последнее отправленное/полученное сообщение</small>

</Link>

</Col>

<Col xs="auto">

<Buttons mode={mode} id={id}/>

</Col>

</Row>

</Card.Body>

<Modal show={show} onHide={handleClose} centered>

<Modal.Header closeButton className="border-0 pb-0">

<Modal.Title className="fw-bold">{name}</Modal.Title>

</Modal.Header>

<Modal.Body className="d-flex justify-content-center align-items-center p-0" style={{ minHeight: '50vh' }}>

<Image

src={photoPath}

style={{

maxHeight: '70vh',

maxWidth: '100%',

width: 'auto',

height: 'auto',

objectFit: 'contain',

}}

fluid

/>

</Modal.Body>

<Modal.Footer className="border-0 pt-0">

<PhotoButtons />

<Button

variant="primary"

onClick={handleClose}

>

Закрыть

</Button>

</Modal.Footer>

</Modal>

</Card>

);

}

Листинг Г.9 — код файла Profile.jsx

import {Container, Row, Card, Col, Button, Image, ButtonGroup, Modal} from "react-bootstrap";

import {PhotoSlider, PhotoRows, StyledPhotoCard, PhotoButtons} from "./PhotoComponents.jsx";

import {VideoSlider, VideoRows} from "./VideoComponents.jsx";

import {PostsSlider, Posts} from "./PostComponents.jsx";

import {useLocation, Link, useParams} from "react-router-dom";

import {useState} from "react";

function ProfileInfo(props) {

const [show, setShow] = useState(false);

const handleClose = () => {setShow(false)};

const handleShow = () => {setShow(true)};

const { photoPath, name, birthDate, address, group, description } = props.profile

function ModalPhoto({path}) {

return (

<Modal show={show} onHide={handleClose}>

<Modal.Header closeButton>

<Modal.Title>Фотография</Modal.Title>

</Modal.Header>

<Modal.Body className="d-flex justify-content-center align-items-center"

style={{

minHeight: '50vh',

padding: 0,

}}>

<Image src={path}

style={{

maxHeight: '80vh',

maxWidth: '100%',

width: 'auto',

height: 'auto',

objectFit: 'contain'

}} fluid/>

</Modal.Body>

<Modal.Footer>

<PhotoButtons />

<Button variant="primary" onClick={handleClose}>

Закрыть

</Button>

</Modal.Footer>

</Modal>

)

}

return (

<StyledPhotoCard className="mb-4 mt-4">

<Row>

<Col xs={4}>

<Card.Img onClick={handleShow} src={photoPath} style={{

width: '100%',

height: '400px',

objectFit: 'cover',

cursor: 'pointer',

borderRight: '1px solid #1a73e8',

borderStartEndRadius: '0',

borderEndEndRadius: '0'

}}></Card.Img>

</Col>

<Col xs={8}>

<Card.Body>

<h1>{name}</h1><br/>

<h5>Дата рождения: {birthDate}</h5>

<h5>Адрес: {address}</h5>

<h5>Группа: {group}</h5><br/>

<h5>Описание: </h5>

<p>{description}</p>

</Card.Body>

</Col>

<ModalPhoto path={photoPath}/>

</Row>

</StyledPhotoCard>

)

}

function ProfileNavigation({id}) {

let path

if(!id)

path = 'profile'

else

path = id.toString()

return (

<ButtonGroup className="mb-4 w-100">

<Button as={Link} to={`/${path}/posts`} variant={location.pathname === `/${path}/posts` ? 'primary' : 'outline-primary'}>Публикации</Button>

<Button as={Link} to={`/${path}/photos`} variant={location.pathname === `/${path}/photos` ? 'primary' : 'outline-primary'}>Фотографии</Button>

<Button as={Link} to={`/${path}/videos`} variant={location.pathname === `/${path}/videos` ? 'primary' : 'outline-primary'}>Видео</Button>

{(path === 'profile' ? (

<Button as={Link} to={`/new\_post`} variant={'outline-primary'}>Новая запись</Button>

) : (

<Button variant={'outline-primary'}>{(id < 21 ? ('Удалить из друзей') : ('В друзья'))}</Button>

))}

</ButtonGroup>

)

}

function myProfile() {

const location = useLocation();

const photos = JSON.parse(sessionStorage.getItem('photos'))

const videos = JSON.parse(sessionStorage.getItem('videos'))

const posts = JSON.parse(sessionStorage.getItem('posts'))

{switch(location.pathname) {

case '/profile/photos' : return (<>

<PhotoSlider photos={photos}/>

<PhotoRows photos={photos} />

</>)

case '/profile/videos' : return (<>

<VideoSlider videos={videos} />

<VideoRows videos={videos} />

</>)

case '/profile/posts' : return (<>

<PostsSlider posts={posts} />

<Posts posts={posts} />

</>)

}}

}

function userProfile() {

const loc = location.pathname.split('/')

let photos = [], videos = [], posts = []

const paths = ['https://images.unsplash.com/photo-1464822759023-fed622ff2c3b?ixlib=rb-4.0.3&q=85&fm=jpg&crop=entropy&w=1920', 'https://images.unsplash.com/photo-1480714378408-67cf0d13bc1b?ixlib=rb-4.0.3&q=85&fm=jpg&crop=entropy&w=1280', 'https://images.unsplash.com/photo-1546069901-ba9599a7e63c?ixlib=rb-4.0.3&q=85&fm=jpg&crop=entropy&w=800', 'https://images.unsplash.com/photo-1518770660439-4636190af475?ixlib=rb-4.0.3&q=85&fm=jpg&crop=entropy&w=1024', 'https://images.unsplash.com/photo-1514888286974-6c03e2ca1dba?ixlib=rb-4.0.3&q=85&fm=jpg&crop=entropy&w=500', 'https://images.unsplash.com/photo-1558591710-4b4a1ae0f04d?ixlib=rb-4.0.3&q=85&fm=jpg&crop=entropy&w=1600']

for(let i = 0; i < 6; i++) {

photos.push({path: paths[i]})

videos.push({path: paths[i]})

posts.push({labelPath: 'Название публикации', photoPath: paths[i], textPath: 'Текст публикации'})

}

{switch (loc[2]) {

case 'photos' : return (<>

<PhotoSlider photos={photos}/>

<PhotoRows photos={photos} />

</>)

case 'videos' : return (<>

<VideoSlider videos={videos} />

<VideoRows videos={videos} />

</>)

case 'posts' : return (<>

<PostsSlider posts={posts} />

<Posts posts={posts} />

</>)

}}

}

function renderContent() {

const location = useLocation();

return (location.pathname.split('/')[1] === 'profile' ? myProfile() : userProfile())

}

export default function Profile() {

const location = useLocation();

const {id} = useParams()

let profile = null

if(location.pathname.startsWith('/profile')) {

profile = JSON.parse(sessionStorage.getItem('profile'))

}

else {

profile = JSON.parse(sessionStorage.getItem('friends')).find((f) => f.id.toString() === id)

if(profile == null)

profile = JSON.parse(sessionStorage.getItem('requests')).find((f) => f.id.toString() === id)

const details = JSON.parse(sessionStorage.getItem('users')).find((f) => f.id.toString() === id)

Object.assign(profile, details)

}

return (

<Container>

<div className={"pt-1"}>

<ProfileInfo profile={profile}/>

<ProfileNavigation id={id} />

{renderContent()}

</div>

</Container>

);

}

Листинг Г.10 — код файла VideoComponents.jsx

import styled from "styled-components";

import {Button, ButtonGroup, Card, Carousel, Col, Image, Modal, Row} from "react-bootstrap";

import {useState} from "react";

import classNames from 'classnames'

import {PhotoItem, PhotoRows, StyledCarousel} from "./PhotoComponents.jsx";

export const StyledVideoCard = styled(Card)`

border: 1px solid #1a73e8;

border-radius: 8px;

`

const LikeButton = styled(Button)`

&:hover {

background-color: #e6032a;

}

`

const CommentButton = styled(Button)`

&:hover {

background-color: #1a73e8;

}

`

export function VideoButtons() {

return (

<ButtonGroup size="sm" className={'float-end'}>

<LikeButton variant="light" size="sm">

<Image src="/like.svg" width={30} height={30} />

</LikeButton>

<CommentButton variant="light" size="sm">

<Image src="/comment.svg" width={30} height={30} />

</CommentButton>

</ButtonGroup>

)

}

export function VideoItem({path}) {

const [show, setShow] = useState(false);

const handleClose = () => {setShow(false)};

const handleShow = () => {setShow(true)};

function ModalVideo({path}) {

return (

<Modal show={show} onHide={handleClose}>

<Modal.Header closeButton>

<Modal.Title>Фотография</Modal.Title>

</Modal.Header>

<Modal.Body className="d-flex justify-content-center align-items-center"

style={{

minHeight: '50vh',

padding: 0,

}}>

<Image src={path}

style={{

maxHeight: '80vh',

maxWidth: '100%',

width: 'auto',

height: 'auto',

objectFit: 'contain',

}} fluid/>

</Modal.Body>

<Modal.Footer>

<VideoButtons />

<Button variant="primary" onClick={handleClose}>

Закрыть

</Button>

</Modal.Footer>

</Modal>

)

}

return (

<Col className="d-flex">

<StyledVideoCard className="w-100">

<div style={{

maxHeight: '350px',

overflow: 'hidden',

display: 'flex',

alignItems: 'center',

justifyContent: 'center'

}}>

<Card.Img

variant="top"

src={path}

style={{

width: '100%',

height: 'auto',

objectFit: 'cover',

cursor: 'pointer'

}}

onClick={handleShow}

/>

</div>

<Card.Body className="d-flex flex-column">

<Card.Title className="mb-2">Видео</Card.Title>

<div className="mt-auto text-end">

<VideoButtons />

</div>

</Card.Body>

</StyledVideoCard>

<ModalVideo path={path}/>

</Col>

)

}

export function VideoRows({videos = [], m = 'mb-4'}) {

return (

<Row xs={3} md={3} className={classNames('g-4', m)}>

{videos.map((video, index) => (

<VideoItem key={index} path={video.path}/>

))}

</Row>

)

}

export function VideoSlider({ videos = [] }) {

if (!videos.length)

return null;

const videoGroups = [];

for (let i = 0; i < videos.length; i += 3) {

videoGroups.push(videos.slice(i, i + 3));

}

return (

<StyledCarousel indicators controls className="mb-4">

{videoGroups.map((group) => (

<Carousel.Item>

<VideoRows videos={group} m={'mb-0'} />

</Carousel.Item>

))}

</StyledCarousel>

);

}

Листинг Г.11 — код файла PhotoComponents.jsx

import styled from "styled-components";

import {Button, ButtonGroup, Card, Carousel, Col, Image, Modal, Row} from "react-bootstrap";

import {useState} from "react";

import classNames from 'classnames'

export const StyledPhotoCard = styled(Card)`

border: 1px solid #1a73e8;

border-radius: 8px;

`

export const StyledCarousel = styled(Carousel)`

.carousel-control-prev,

.carousel-control-next {

width: 40px;

height: 40px;

top: 50%;

transform: translateY(-50%);

opacity: 0.8;

transition: opacity 0.2s;

background: none; /\* Убираем стандартный фон \*/

&:hover {

opacity: 1;

}

}

.carousel-control-prev-icon,

.carousel-control-next-icon {

background-color: #1a73e8;

border-radius: 50%;

width: 100%;

height: 100%;

background-size: 60%;

display: flex;

align-items: center;

justify-content: center;

}

.carousel-control-prev {

left: 45px;

}

.carousel-control-next {

right: 45px;

}

.carousel-indicators {

button {

background-color: #1a73e8;

opacity: 0.4;

width: 30px;

height: 4px;

border-radius: 2px;

margin: 0 3px;

transition: opacity 0.3s ease;

&.active {

opacity: 1;

}

}

}

`;

const LikeButton = styled(Button)`

&:hover {

background-color: #e6032a;

}

`

const CommentButton = styled(Button)`

&:hover {

background-color: #1a73e8;

}

`

export function PhotoButtons() {

return (

<ButtonGroup size="sm" className={'float-end'}>

<LikeButton variant="light" size="sm">

<Image src="/like.svg" width={30} height={30} />

</LikeButton>

<CommentButton variant="light" size="sm">

<Image src="/comment.svg" width={30} height={30} />

</CommentButton>

</ButtonGroup>

)

}

export function PhotoItem({path}) {

const [show, setShow] = useState(false);

const handleClose = () => {setShow(false)};

const handleShow = () => {setShow(true)};

function ModalPhoto({path}) {

return (

<Modal show={show} onHide={handleClose}>

<Modal.Header closeButton>

<Modal.Title>Фотография</Modal.Title>

</Modal.Header>

<Modal.Body className="d-flex justify-content-center align-items-center"

style={{

minHeight: '50vh',

padding: 0,

}}>

<Image src={path}

style={{

maxHeight: '80vh',

maxWidth: '100%',

width: 'auto',

height: 'auto',

objectFit: 'contain',

}} fluid/>

</Modal.Body>

<Modal.Footer>

<PhotoButtons />

<Button variant="primary" onClick={handleClose}>

Закрыть

</Button>

</Modal.Footer>

</Modal>

)

}

return (

<Col className="d-flex">

<StyledPhotoCard className="w-100">

<div style={{

maxHeight: '350px',

overflow: 'hidden',

display: 'flex',

alignItems: 'center',

justifyContent: 'center'

}}>

<Card.Img

variant="top"

src={path}

style={{

width: '100%',

height: 'auto',

objectFit: 'cover',

cursor: 'pointer'

}}

onClick={handleShow}

/>

</div>

<Card.Body className="d-flex flex-column">

<div className="mt-auto text-end">

<PhotoButtons />

</div>

</Card.Body>

</StyledPhotoCard>

<ModalPhoto path={path}/>

</Col>

)

}

export function PhotoRows({photos = [], m = 'mb-4'}) {

return (

<Row xs={3} md={3} className={classNames('g-4', m)}>

{photos.map((photo, index) => (

<PhotoItem key={index} path={photo.path}/>

))}

</Row>

)

}

export function PhotoSlider({ photos = [] }) {

if (!photos.length)

return null;

const photoGroups = [];

for (let i = 0; i < photos.length; i += 3) {

photoGroups.push(photos.slice(i, i + 3));

}

return (

<StyledCarousel indicators controls className="mb-4">

{photoGroups.map((group) => (

<Carousel.Item>

<PhotoRows photos={group} m={'mb-0'} />

</Carousel.Item>

))}</StyledCarousel>);}

Листинг Г.12 — код файла PostComponents.jsx

import {useState} from "react";

import {Button, Card, Carousel, Col, Image, Modal, Row} from "react-bootstrap";

import {PhotoButtons, StyledCarousel, StyledPhotoCard} from "../done/PhotoComponents.jsx";

import classNames from "classnames";

export function Post({m='mb-4', labelPath, photoPath, textPath}) {

const [show, setShow] = useState(false);

const handleClose = () => {setShow(false)};

const handleShow = () => {setShow(true)};

function ModalPost({ labelPath, photoPath, textPath }) {

return (

<Modal show={show} onHide={handleClose}>

<Modal.Header closeButton>

<Modal.Title>{labelPath}</Modal.Title>

</Modal.Header>

<Modal.Body style={{ padding: 0 }}>

{/\* Блок с изображением \*/}

<div className="d-flex justify-content-center align-items-center"

style={{

minHeight: '50vh',

backgroundColor: '#f8f9fa', // Фон на случай если изображение маленькое

padding: '20px 0'

}}>

<Image

src={photoPath}

style={{

maxHeight: '60vh',

maxWidth: '100%',

width: 'auto',

height: 'auto',

objectFit: 'contain',

}}

fluid

/>

</div>

{/\* Блок с текстом \*/}

<div style={{

padding: '20px',

borderTop: '1px solid #dee2e6' // Граница между изображением и текстом

}}>

<p style={{ margin: 0 }}>{textPath}</p>

</div>

</Modal.Body>

<Modal.Footer>

<Button variant="primary" onClick={handleClose}>

Закрыть

</Button>

<PhotoButtons />

</Modal.Footer>

</Modal>

)

}

return (

<StyledPhotoCard className={classNames(" mt-1", m)}>

<Row>

<Col xs={4}>

<Card.Img src={photoPath} style={{

width: '100%',

height: '400px',

objectFit: 'cover',

borderRight: '1px solid #1a73e8',

borderEndStartRadius: '0',

borderStartEndRadius: '0',

borderEndEndRadius: '0'

}}></Card.Img>

</Col>

<Col xs={8}>

<Card.Body>

<Card.Title>{labelPath}</Card.Title>

<Card.Text>{textPath}</Card.Text>

</Card.Body>

</Col>

<ModalPost labelPath={labelPath} textPath={textPath} photoPath={photoPath}/>

</Row>

<Card.Footer>

<Button onClick={handleShow}>Читать полностью</Button>

<PhotoButtons />

</Card.Footer>

</StyledPhotoCard>

)

}

export function Posts({posts = []}) {

return (

<>

{posts.map((post, index) => {

return <Post key={index} photoPath={post.photoPath} labelPath={post.labelPath} textPath={post.textPath}/>

})}

</>

)

}

export function PostsSlider({posts = []}) {

return (

<StyledCarousel className="mb-4">

{posts.map((post, index) => {

return (

<Carousel.Item key={index}>

<Post m={'mb-0'} photoPath={post.photoPath} labelPath={post.labelPath} textPath={post.textPath}/>

</Carousel.Item>

)

})}

</StyledCarousel>

)

}

Листинг Г.13 — код файла Messenger.jsx

import {useState} from "react";

import {Container, Button, Image, Form, InputGroup, Card, ListGroup, Row, Col} from 'react-bootstrap';

import ListItem from "../done/ListItem.jsx";

import {useLocation, useParams, Link} from "react-router-dom";

function Chat() {

const {id} = useParams();

const profile = JSON.parse(sessionStorage.getItem('friends')).find((f) => f.id.toString() === id);

const [messages, setMessages] = useState([

{id: 1, text: 'Привет! Как дела?', sender: 'them', time: '10:00'},

{id: 2, text: 'Отлично, спасибо! А у тебя?', sender: 'me', time: '10:02'},

{id: 3, text: 'Тоже всё хорошо. Давно не виделись!', sender: 'them', time: '10:05'},

]);

const [newMessage, setNewMessage] = useState('');

const handleSendMessage = async (e) => {

e.preventDefault();

if (!newMessage.trim()) return;

// Добавляем сообщение пользователя

const userMessage = {

id: messages.length + 1,

text: newMessage,

sender: 'me',

time: new Date().toLocaleTimeString([], { hour: '2-digit', minute: '2-digit' })

};

setMessages(prev => [...prev, userMessage]);

setNewMessage('');

};

return (

<Container fluid className="vh-100 d-flex flex-column p-0" style={{maxHeight: '85vh'}}>

<Card className="border-bottom shadow-sm" style={{flex: '0 0 auto'}}>

<Card.Body className="py-2">

<Row className="align-items-center">

<Col xs="auto">

<Button

as={Link}

to="/messenger"

variant="outline-primary"

className="me-2"

>

Назад

</Button>

</Col>

<Col xs="auto">

<Image

src={profile.photoPath}

roundedCircle

width={40}

height={40}

className="border"

/>

</Col>

<Col>

<h5 className="mb-0">{profile.name}</h5>

<small className="text-muted">В сети</small>

</Col>

</Row>

</Card.Body>

</Card>

<div

className="flex-grow-1 p-3 overflow-auto"

style={{

backgroundColor: '#f8f9fa',

minHeight: 0 // Важно для корректного скролла

}}

>

<ListGroup variant="flush">

{messages.map((message) => (

<ListGroup.Item

key={message.id}

className="border-0 bg-transparent p-1"

>

<div

className={`d-flex mb-2 ${message.sender === 'me' ? 'justify-content-end' : 'justify-content-start'}`}

>

<Card

className={`${message.sender === 'me' ? 'bg-primary text-white' : 'bg-light'}`}

style={{maxWidth: '75%'}}

>

<Card.Body className="p-2">

<p className="mb-1">{message.text}</p>

<small

className={`d-block text-end ${message.sender === 'me' ? 'text-white-50' : 'text-muted'}`}>

{message.time}

</small>

</Card.Body>

</Card>

</div>

</ListGroup.Item>

))}

</ListGroup>

</div>

<Card className="border-top" style={{flex: '0 0 auto'}}>

<Card.Body className="py-2">

<Form onSubmit={handleSendMessage}>

<InputGroup>

<Form.Control

type="text"

placeholder="Напишите сообщение..."

value={newMessage}

onChange={(e) => setNewMessage(e.target.value)}

className="border-0"

/>

<Button

variant="primary"

type="submit"

disabled={!newMessage.trim()}

>

Отправить

</Button>

</InputGroup>

</Form>

</Card.Body>

</Card>

</Container>

);

}

function ChatsList({chats = []}) {

return (

(chats.length === 0 ? (

<h1>Список чатов пуст</h1>

) : (

chats.map((chat, index) => (<>

<ListGroup.Item key={index}>

<ListItem name={chat.name} photoPath={chat.photoPath} id={chat.id} mode={'chats'}/>

</ListGroup.Item>

</>))))

)

}

export default function Messenger() {

const location = useLocation()

return (

<Container className="pt-4">

{location.pathname === '/messenger' ? (<ChatsList chats={JSON.parse(sessionStorage.getItem('friends'))} />) : location.pathname.startsWith('/messenger/chat/') ? <Chat /> : null}

</Container>

);}

Листинг Г.13 — код файла Friends.jsx

import {Link, useLocation} from "react-router-dom";

import {Container, ButtonGroup, Button} from "react-bootstrap";

import ListItem from "../done/ListItem.jsx";

function FriendsList({friends = []}) {

return (

(friends.length === 0 ? (<h1>Список друзей пуст</h1>) : (friends.map((friend) => (

<>

<ListItem key={friend.id} name={friend.name} photoPath={friend.photoPath} id={friend.id} mode={'friends'} />

</>

))))

)

}

function RequestsList({requests = []}) {

return (

(requests.length === 0 ? (<h1>Список запросов пуст</h1>) : (requests.map((request) => (

<>

<ListItem key={request.id} name={request.name} photoPath={request.photoPath} id={request.id} mode={'requests'} />

</>

))))

)

}

export default function Friends() {

const location = useLocation();

function renderContent() {

switch(location.pathname){

case '/friends': {return (

<FriendsList key={'friendsList'} friends={JSON.parse(sessionStorage.getItem('friends'))} />

)}

case '/friends/requests': {return (

<RequestsList key={'requestsList'} requests={JSON.parse(sessionStorage.getItem('requests'))} />

)}

}

}

return (

<Container className="pt-4">

<ButtonGroup className="mb-4 w-100">

<Button as={Link} to={'/friends'} variant={location.pathname === '/friends' ? 'primary' : 'outline-primary'}>Друзья</Button>

<Button as={Link} to={'/friends/requests'} variant={location.pathname === '/friends/requests' ? 'primary' : 'outline-primary'}>Заявки</Button>

</ButtonGroup>

{renderContent()}

</Container>

);}

Листинг Г.14 — код файла Register.jsx

import {useState} from 'react'

import {useNavigate, Link} from 'react-router-dom';

import { Container, Row, Col, Card, Button, Form } from 'react-bootstrap';

export default function RegisterPage() {

const [formData, setFormData] = useState({

name: '',

email: '',

password: '',

userType: 'specialist',

organization: ''

});

const [errors, setErrors] = useState({});

const [success, setSuccess] = useState(false);

const navigate = useNavigate();

const handleChange = (e) => {

const { name, value } = e.target;

setFormData(prev => ({

...prev,

[name]: value

}));

};

const validate = () => {

const newErrors = {};

if (!formData.name) newErrors.name = 'Введите имя';

if (!formData.email) newErrors.email = 'Введите email';

if (!formData.password || formData.password.length < 6) newErrors.password = 'Пароль должен быть не менее 6 символов';

if (formData.userType === 'organization' && !formData.organization) newErrors.organization = 'Введите название организации';

setErrors(newErrors);

return Object.keys(newErrors).length === 0;

};

const handleSubmit = (e) => {

e.preventDefault();

if (validate()) {

// Здесь должна быть логика отправки данных на сервер

console.log('Регистрация:', formData);

setSuccess(true);

setTimeout(() => navigate('/progile/posts'), 2000);

}

};

return (

<Container className="py-5">

<Row className="justify-content-center">

<Col md={8} lg={6}>

<Card className="shadow">

<Card.Body className="p-4">

<h2 className="text-center mb-4">Регистрация в SOCBOOK</h2>

{success && (

<Alert variant="success" className="text-center">

Регистрация прошла успешно! Перенаправляем на страницу входа...

</Alert>

)}

<Form onSubmit={handleSubmit}>

<Form.Group className="mb-3">

<Form.Label>Тип аккаунта</Form.Label>

<Form.Select

name="userType"

value={formData.userType}

onChange={handleChange}

>

<option value="specialist">Специалист</option>

<option value="organization">Организация</option>

<option value="manufacturer">Производитель ТСР</option>

<option value="public">Общественная организация</option>

</Form.Select>

</Form.Group>

{formData.userType === 'organization' && (

<Form.Group className="mb-3">

<Form.Label>Название организации</Form.Label>

<Form.Control

type="text"

name="organization"

value={formData.organization}

onChange={handleChange}

isInvalid={!!errors.organization}

/>

<Form.Control.Feedback type="invalid">

{errors.organization}

</Form.Control.Feedback>

</Form.Group>

)}

<Form.Group className="mb-3">

<Form.Label>{formData.userType === 'specialist' ? 'ФИО' : 'Контактное лицо'}</Form.Label>

<Form.Control

type="text"

name="name"

value={formData.name}

onChange={handleChange}

isInvalid={!!errors.name}

/>

<Form.Control.Feedback type="invalid">

{errors.name}

</Form.Control.Feedback>

</Form.Group>

<Form.Group className="mb-3">

<Form.Label>Email</Form.Label>

<Form.Control

type="email"

name="email"

value={formData.email}

onChange={handleChange}

isInvalid={!!errors.email}

/>

<Form.Control.Feedback type="invalid">

{errors.email}

</Form.Control.Feedback>

</Form.Group>

<Form.Group className="mb-3">

<Form.Label>Пароль</Form.Label>

<Form.Control

type="password"

name="password"

value={formData.password}

onChange={handleChange}

isInvalid={!!errors.password}

/>

<Form.Control.Feedback type="invalid">

{errors.password}

</Form.Control.Feedback>

</Form.Group>

<div className="d-grid gap-2 mt-4">

<Button variant="primary" type="submit" size="lg">

Зарегистрироваться

</Button>

</div>

</Form>

<div className="text-center mt-3">

<span>Уже есть аккаунт? </span>

<Link to="/login">Войти</Link>

</div>

</Card.Body>

</Card>

</Col>

</Row>

</Container>

);

}

Листинг Г.15 — код файла Auth.jsx

import {useState} from 'react'

import {useNavigate, Link} from 'react-router-dom'

import { Container, Row, Col, Card, Button, Form } from 'react-bootstrap';

export default function LoginPage() {

const [formData, setFormData] = useState({

email: '',

password: ''

});

const [error, setError] = useState('');

const navigate = useNavigate();

const handleChange = (e) => {

const { name, value } = e.target;

setFormData(prev => ({

...prev,

[name]: value

}));

};

const handleSubmit = (e) => {

e.preventDefault();

// Здесь должна быть проверка данных на сервере

if (formData.email && formData.password) {

console.log('Авторизация:', formData);

navigate('/profile/posts');

} else {

setError('Введите email и пароль');

}

};

return (

<Container className="py-5">

<Row className="justify-content-center">

<Col md={8} lg={6}>

<Card className="shadow">

<Card.Body className="p-4">

<h2 className="text-center mb-4">Вход в SOCBOOK</h2>

{error && <Alert variant="danger">{error}</Alert>}

<Form onSubmit={handleSubmit}>

<Form.Group className="mb-3">

<Form.Label>Email</Form.Label>

<Form.Control

type="email"

name="email"

value={formData.email}

onChange={handleChange}

/>

</Form.Group>

<Form.Group className="mb-3">

<Form.Label>Пароль</Form.Label>

<Form.Control

type="password"

name="password"

value={formData.password}

onChange={handleChange}

/>

</Form.Group>

<div className="d-grid gap-2 mt-4">

<Button variant="primary" type="submit" size="lg">

Войти

</Button>

</div>

</Form>

<div className="text-center mt-3">

<span>Нет аккаунта? </span>

<Link to="/register">Зарегистрироваться</Link>

</div>

</Card.Body>

</Card>

</Col>

</Row>

</Container>

);

}

Листинг Г.16 — код файла NewPost.jsx

import {useState} from "react";

import {Link} from "react-router-dom";

import { Container, Row, Col, Card, Button, Image, Form } from 'react-bootstrap';

export default function CreatePostPage() {

const [formData, setFormData] = useState({

title: '',

content: '',

isPublic: true,

image: null

});

const [preview, setPreview] = useState('');

const [success, setSuccess] = useState(false);

const handleChange = (e) => {

const { name, value, type, checked } = e.target;

setFormData(prev => ({

...prev,

[name]: type === 'checkbox' ? checked : value

}));

};

const handleImageChange = (e) => {

const file = e.target.files[0];

if (file) {

setFormData(prev => ({ ...prev, image: file }));

const reader = new FileReader();

reader.onloadend = () => {

setPreview(reader.result);

};

reader.readAsDataURL(file);

}

};

const handleSubmit = (e) => {

e.preventDefault();

// Здесь должна быть отправка данных на сервер

console.log('Создание публикации:', formData);

setSuccess(true);

setTimeout(() => setSuccess(false), 3000);

};

return (

<Container className="py-5">

<Row className="justify-content-center">

<Col md={10} lg={8}>

<Card className="shadow">

<Card.Body className="p-4">

<h2 className="text-center mb-4">Создать новую публикацию</h2>

{success && (

<Alert variant="success" className="text-center">

Публикация успешно создана!

</Alert>

)}

<Form onSubmit={handleSubmit}>

<Form.Group className="mb-3">

<Form.Label>Заголовок</Form.Label>

<Form.Control

type="text"

name="title"

value={formData.title}

onChange={handleChange}

placeholder="Введите заголовок публикации"

required

/>

</Form.Group>

<Form.Group className="mb-3">

<Form.Label>Текст публикации</Form.Label>

<Form.Control

as="textarea"

rows={5}

name="content"

value={formData.content}

onChange={handleChange}

placeholder="Напишите содержание публикации..."

required

/>

</Form.Group>

<Form.Group className="mb-3">

<Form.Label>Изображение (необязательно)</Form.Label>

<Form.Control

type="file"

accept="image/\*"

onChange={handleImageChange}

/>

{preview && (

<div className="mt-3">

<Image src={preview} thumbnail style={{ maxHeight: '200px' }} />

</div>

)}

</Form.Group>

<Form.Group className="mb-4">

<Form.Check

type="checkbox"

label="Публичная запись"

name="isPublic"

checked={formData.isPublic}

onChange={handleChange}

/>

<Form.Text className="text-muted">

{formData.isPublic

? 'Публикация будет видна всем пользователям'

: 'Публикация будет видна только вашим подписчикам'}

</Form.Text>

</Form.Group>

<div className="d-flex justify-content-end gap-3">

<Button variant="outline-secondary" as={Link} to="/profile/posts">

Отмена

</Button>

<Button variant="primary" type="submit">

Опубликовать

</Button>

</div>

</Form>

</Card.Body>

</Card>

</Col>

</Row>

</Container>

);

}

Листинг Г.17 — код файла Feed.jsx

import { Container } from "react-bootstrap";

import { PhotoSlider } from "./PhotoComponents.jsx";

import { Posts } from "./PostComponents.jsx";

import { VideoSlider } from "./VideoComponents.jsx";

export default function Feed() {

const posts = JSON.parse(sessionStorage.getItem("posts")) || [];

const photos = JSON.parse(sessionStorage.getItem("photos")) || [];

const videos = JSON.parse(sessionStorage.getItem("videos")) || [];

return (

<Container>

<div className="pt-4">

<Posts posts={posts.slice(0, 1)} />

<PhotoSlider photos={photos} />

<Posts posts={posts.slice(1, 2)} />

<VideoSlider videos={videos} />

<Posts posts={posts.slice(2, 3)} />

</div>

</Container>

);

}

Листинг Г.18 — код файла PhotoComponents.test.jsx

import '@testing-library/jest-dom';

import { render, screen} from '@testing-library/react';

import { PhotoItem, PhotoButtons, PhotoSlider } from './PhotoComponents.jsx';

describe('PhotoItem', () => {

const photoPath = 'photo1.jpg';

test('renders PhotoItem', () => {

render(<PhotoItem path={photoPath} />);

expect(screen.getByText(/фотография/i)).toBeInTheDocument();

});

});

describe('PhotoButtons', () => {

test('renders PhotoButtons', () => {

render(<PhotoButtons />);

expect(screen.getByRole('button', { name: /like/i })).toBeInTheDocument();

expect(screen.getByRole('button', { name: /comment/i })).toBeInTheDocument();

});

});

describe('PhotoSlider', () => {

const photos = [

{ path: 'photo1.jpg' }

];

test('renders PhotoSlider', () => {

render(<PhotoSlider photos={photos} />);

expect(screen.getByText(/фотография/i)).toBeInTheDocument();

});

});

Листинг Г.19 — код файла VideoComponents.test.jsx

import '@testing-library/jest-dom';

import { render, screen} from '@testing-library/react';

import { VideoItem, VideoButtons, VideoSlider } from './VideoComponents.jsx';

describe('VideoItem', () => {

const photoPath = 'photo1.jpg';

test('renders VideoItem', () => {

render(<VideoItem path={photoPath} />);

expect(screen.getByText(/Видео/i)).toBeInTheDocument();

});

});

describe('VideoButtons', () => {

test('renders VideoButtons', () => {

render(<VideoButtons />);

expect(screen.getByRole('button', { name: /like/i })).toBeInTheDocument();

expect(screen.getByRole('button', { name: /comment/i })).toBeInTheDocument();

});

});

describe('VideoSlider', () => {

const videos = [

{ path: 'video.jpg' }

];

test('renders VideoSlider', () => {

render(<VideoSlider videos={videos} />);

expect(screen.getByText(/Видео/i)).toBeInTheDocument();

});

});

Листинг Г.20 — код файла VideoComponents.test.jsx

import '@testing-library/jest-dom';

import { render, screen} from '@testing-library/react';

import { Post, Posts, PostsSlider } from './PostComponents.jsx';

describe('Post', () => {

const photoPath = 'photo';

const textPath = 'text';

const labelPath = 'Публикация'

test('renders Post', () => {

render(<Post photoPath={photoPath} textPath={textPath} labelPath={labelPath}/>);

expect(screen.getByText(/Публикация/i)).toBeInTheDocument();

});

});

describe('Posts', () => {

const testPosts = [

{ photoPath: 'photo.jpg', textPath: 'text', labelPath: 'Публикация' }

];

test('renders Posts', () => {

render(<Posts posts={testPosts} />);

expect(screen.getByRole('posts')).toBeInTheDocument();

});

});

describe('PostsSlider', () => {

const Posts = [

{ photoPath: 'photo.jpg', textPath: 'text', labelPath: 'Публикация' }

];

test('renders PostSlider', () => {

render(<PostsSlider Posts={Posts} />);

expect(screen.getByRole('slider')).toBeInTheDocument();

});

});

Листинг Г.21 — код файла create\_components.sh

TEST\_CODE="import '@testing-library/jest-dom';

import { MemoryRouter } from 'react-router-dom';

import { render, screen } from '@testing-library/react';

import $2 from './$2.jsx';

describe('$2', () => {

test('test $2', () => {

render(

<MemoryRouter>

<$2 />

</MemoryRouter>

);

expect(screen.getByRole('$2')).toBeInTheDocument();

});

});"

if [ $# -ne 2 ]; then

echo "Ошибка: требуется 2 аргумента"

echo "Использование: $0 <категория> <название\_компонента>"

exit 1

fi

if [ ! -e "src/$1" ]; then

mkdir "src/$1"

echo "Созадна директория /src/$1"

fi

if [ -e "src/$1/$2" ]; then

echo "src/$1/$2 уже существует"

if [ ! -e "src/$1/$2/$2.jsx" ]; then

touch "src/$1/$2/$2.jsx"

echo "создан src/$1/$2/$2.jsx"

else

echo "src/$1/$2/$2.jsx уже существует"

fi

if [ ! -e "src/$1/$2/$2.test.jsx" ]; then

touch "src/$1/$2/$2.test.jsx"

echo "$TEST\_CODE" > "src/$1/$2/$2.test.jsx"

echo "создан src/$1/$2/$2.test.jsx"

else

echo "src/$1/$2/$2.test.jsx уже существует"

fi

exit 0

fi

if [ -e "src/$1" ]; then

mkdir "src/$1/$2"

touch "src/$1/$2/$2.jsx"

touch "src/$1/$2/$2.test.jsx"

echo "$TEST\_CODE" > "src/$1/$2/$2.test.jsx"

fi