Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Кафедра информационных систем и

программной инженерии

**ОТЧЕТ**

**О ПРОХОЖДЕНИИ ПРОИЗВОДСТВЕННОЙ ПРАКТИКИ**

**ООО «АИСТСОФТ»**

**по направлению обучения**

**09.03.04 – Программная инженерия**

**с 16.10.2023 по 10.12.2023**

Выполнил:

студент группы ПРИ-120

Грачев Д.А.

Руководитель практики от ВлГУ:

Озерова М.И.

Владимир, 2023

СОДЕРЖАНИЕ

[ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ 3](#_Toc135397329)

[КАЛЕНДАРНЫЙ ГРАФИК ПРОХОЖДЕНИЯ ПРАКТИКИ 4](#_Toc135397330)

[ВВЕДЕНИЕ 5](#_Toc135397331)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 6](#_Toc135397332)

[1. Изучение HTML 6](#_Toc135397333)

[2. Изучение CSS 7](#_Toc135397334)

[3. Изучение JavaScript 8](#_Toc135397335)

[2 ВЕРСТКА МАКЕТА 10](#_Toc135397336)

[1. Заголовок 10](#_Toc135397337)

[2. Заголовок 10](#_Toc135397338)

[ЗАКЛЮЧЕНИЕ 11](#_Toc135397339)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 12](#_Toc135397340)

[ПРИЛОЖЕНИЕ А 13](#_Toc135397341)

# ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

|  |  |
| --- | --- |
| Определения, обозначения, сокращения | Расшифровка |
| Селектор | Указатель в системе стилей CSS |
| Тег | Часть языка разметки HTML, обозначающая отдельный блок |

# КАЛЕНДАРНЫЙ ГРАФИК ПРОХОЖДЕНИЯ ПРАКТИКИ

|  |  |
| --- | --- |
| **Вид работы** | **Время выполнения** |
| Инструктаж по технике безопасности. | 16.10.2023 - 17.10.2023 |
| Изучение разрабатываемой ИС | 18.10.2023 – 24.10.2023 |
| Создание макетов в Figma | 25.10.2023 – 31.10.2023 |
| Настройка окружения | 01.11.2023 – 05.11.2023 |
| Разработка лендинга | 06.11.2023 – 19.11.2023 |
| Разработка веб-приложения | 20.11.2023 – 03.12.2023 |
| Оформление отчета | 04.12.2023 – 10.12.2023 |

# ВВЕДЕНИЕ

В рамках производственной практики был получен опыт работы в компании, специализирующейся на разработке высоконагруженного программного обеспечения для сбора, обработки и анализа данных в реальном времени. Компания предоставляет решения, которые позволяют клиентам преобразовывать огромные объемы данных в ценную информацию для принятия важных решений. Цель состояла в получении опыта работы с современными технологиями.

В ходе практики в различных проектах, был получен практический опыт по разработке высоконагруженных пользовательских интерфейсов, изучен теоретический курс минимально необходимый для начала разработки качественных пользовательских интерфейсов. В данном отчете мною будет описан процесс изучения теоретических тем, верстки страницы по макету и создания React-приложения.

# 1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

В ходе этой части практики я уделял основное внимание изучению теоретических аспектов и практическому освоению технологий создания пользовательских интерфейсов.

## Изучение HTML

HTML (HyperText Markup Language): HTML является основным языком разметки веб-страниц и его усвоение важно для освоения веб-разработки. В процессе изучения HTML я тщательно ознакомился с основными тегами и их применением, включая заголовки, параграфы, списки, таблицы, ссылки и изображения. Однако, стоит отметить, что одним из наиболее значимых аспектов изучения HTML было освоение семантических тегов, таких как <header>, <nav>, <section> и <footer>. Их использование способствует улучшению структуры содержимого веб-страницы и повышению ее доступности для поисковых систем и пользователей с ограниченными возможностями. Использование семантических тегов в HTML может помочь в SEO (поисковой оптимизации) следующим образом:

* + Улучшение структуры сайта: семантические теги позволяют лучше организовать содержимое страницы, что делает ее более удобной для восприятия поисковыми роботами, и как следствие, повышает ее ранжирование в результатах поиска.
  + Усиление смысла: использование семантических тегов позволяет определить, какой контент является наиболее значимым для страницы. Это позволяет поисковым роботам лучше понимать, о чем идет речь на странице и наложить на нее наиболее релевантный поисковой запрос.
  + Улучшение пользовательского опыта: использование семантических тегов помогает пользователям лучше понять, какой контент относится к какой части страницы, что улучшает их опыт и повышает удовлетворенность от посещения веб-сайта.

## Изучение CSS

CSS (Cascading Style Sheets): CSS используется для оформления веб-страниц и задания их внешнего вида. В процессе изучения CSS я уделил особое внимание различным методам применения стилей к элементам HTML, включая встроенные, внутренние и внешние таблицы стилей. Я также углубленно изучил селекторы CSS, принципы каскадирования стилей и наследования. Помимо этого, одним из интересных моментов в ходе работы с CSS было использование псевдоэлементов и псевдоклассов, которые позволяют создавать специальные эффекты и анимации, такие как наведение и фокусировка. В CSS есть несколько видов селекторов:

* + Селектор типа: выбирает элементы по их названию тега. Например, `h1` выберет все заголовки первого уровня на странице.
  + Селектор идентификатора: выбирает элемент по уникальному идентификатору, заданному в атрибуте `id`. Например, `#header` выберет элемент с id="header".
  + Селектор класса: выбирает элементы по наличию класса, заданного в атрибуте `class`. Например, `.content` выберет все элементы с классом "content".
  + Селектор атрибута: выбирает элементы по наличию атрибута с определенным значением. Например, `[type="text"]` выберет все элементы с атрибутом "type" и значением "text".
  + Селектор потомка: выбирает элементы, которые являются потомками указанного родительского элемента. Например, `ul li` выберет все элементы `li`, которые являются потомками элемента `ul`.
  + Селектор дочернего элемента: выбирает элементы, которые являются непосредственными потомками указанного родительского элемента. Например, `ul > li` выберет все элементы `li`, которые являются непосредственными потомками элемента `ul`.
  + Селектор соседнего элемента: выбирает элементы, которые следуют непосредственно за указанным элементом и имеют тот же уровень вложенности. Например, `h1 + p` выберет первый абзац после заголовка первого уровня на странице.
  + Селектор псевдо-класса: выбирает элементы в зависимости от их состояния или положения на странице. Например, `:hover` выберет элементы, на которые наведен курсор мыши.
  + Селектор псевдо-элемента: позволяет выбирать определенную часть элемента (например, первый символ или первую строку), и применять к ней стили. Например, `::after` создает псевдо-элемент "после" заданного элемента.

## Изучение JavaScript

JavaScript (JS): JavaScript - это язык программирования, который придает динамичность и интерактивность веб-страницам. В процессе изучения JS я углубленно освоил его основы, включая работу с переменными, операторами, условными конструкциями и циклами. Я также приобрел практические навыки работы с функциями, массивами и объектами. Одним из наиболее интересных аспектов, которыми я занимался в ходе изучения JS, было использование событий и обработчиков событий для реагирования на действия пользователей, такие как щелчки мыши и отправка форм. JavaScript отличается от других языков программирования по нескольким основным причинам:

* + Динамическая типизация: в JavaScript переменные не имеют фиксированного типа данных, и их тип может меняться в процессе выполнения программы.
  + Функциональная природа: JavaScript имеет поддержку функций первого класса, что позволяет использовать функции в качестве значений и передавать их в качестве параметров другим функциям.
  + Асинхронная модель программирования: JavaScript может выполнить несколько задач одновременно, используя механизмы асинхронного программирования, такие как событийная модель или промисы.
  + Изначально разработан для браузеров: JavaScript был создан для добавления интерактивности на веб-страницы, и поэтому наличие окружения браузера сильно влияет на то, как с ним работать.
  + Язык со слабой типизацией: опечатки в именах переменных или функций могут привести к неожиданным результатам и ошибкам при выполнении кода.
  + Нестрогая обработка ошибок: в языке JavaScript отсутствует строгая обработка ошибок и исключений, что может привести к трудноуловимым ошибкам и багам.
  + Прототипное наследование: в JavaScript используется прототипное наследование вместо классического наследования, что требует особого подхода при создании и наследовании объектов.

В итоге, изучение HTML/CSS и JS представило собой увлекательный и познавательный опыт. Я усвоил основные концепции и развил практические навыки, которые окажутся неоценимыми в моей дальнейшей работе в области веб-разработки. Учитывая важность данных технологий в современной веб-разработке, я с нетерпением ожидаю возможности применить полученные знания на практике.

# 2 ВЕРСТКА МАКЕТА

## Макет для верстки

В процессе производственной практики я получал макет для верстки от своего куратора. Для этого мне отправлялся макет в виде проекта в Figma. Я получал задание на верстку страницы или сайта и самостоятельно работал над переносом дизайна на код. В случаях, когда возникали вопросы или непонимание, я мог обратиться к своему куратору за помощью. После завершения работы я отправлял ее на проверку и далее на исправление или утверждение. В процессе работы я улучшал свои навыки верстки и дизайна, а также научился работать в команде и координировать свою работу с работой других специалистов.

Отдельные страницы макета приведены в приложении А.

## Верстка макета

В процессе верстки макета были использованы ранее изученные технологии, используемым в разработке пользовательских интерфейсов. Ниже будут представлены интересные части этапа верстки макета:

Создание вкладок используя только HTML и CSS, без использования JavaScript



Рисунок 1 - Код вкладок на HTML

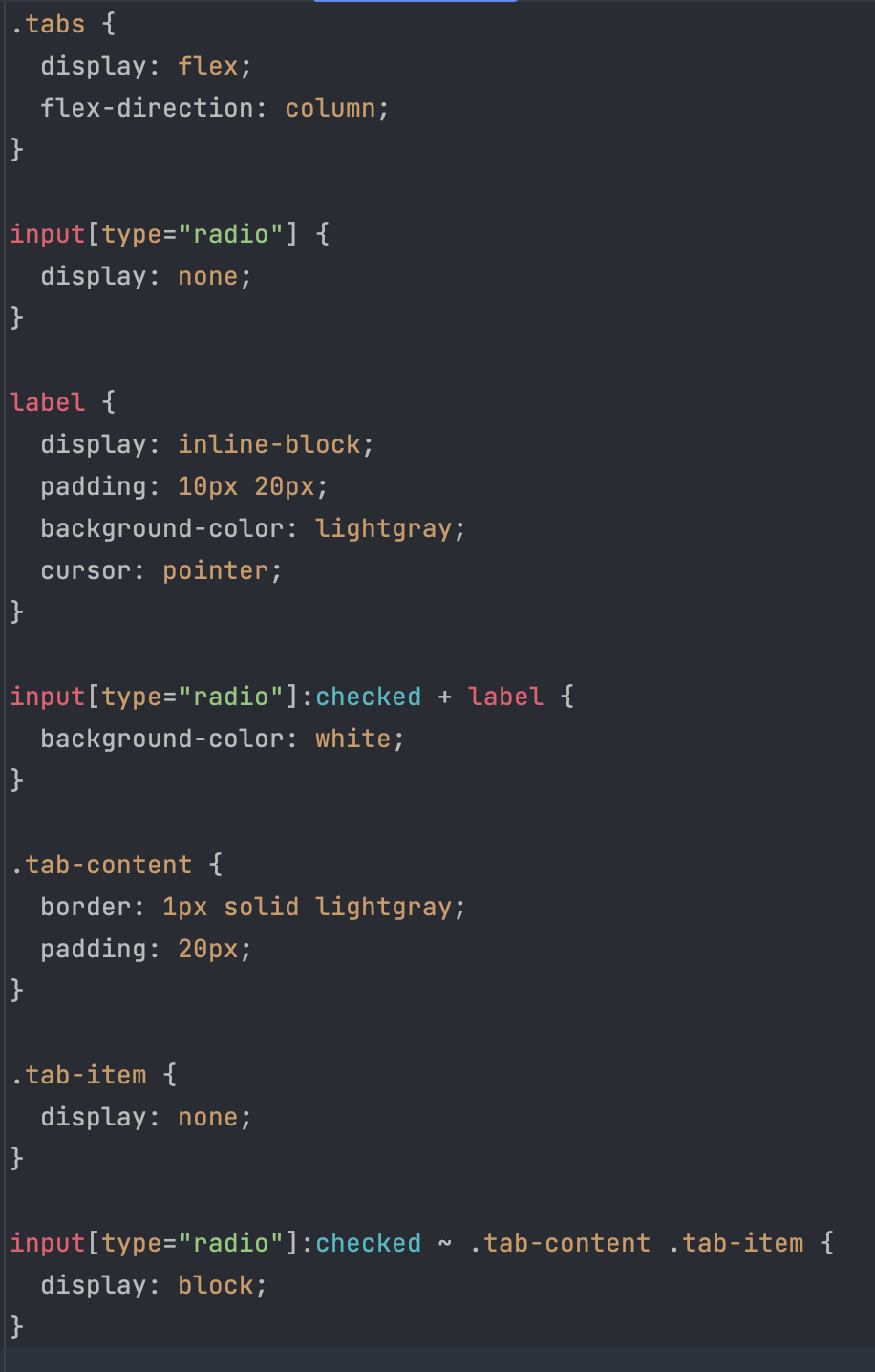


Рисунок 2 - Код вкладок на CSS

В данном примере используются входные элементы радиокнопок *(<input type="radio">)*, метки *(<label>)*, а также контейнеры для содержимого вкладок *(<div class="tab-content">)* и сами вкладки *(<div class="tab-item">).*

CSS стили задают внешний вид вкладок и их состояний: активной вкладки и скрытого содержимого.

При клике на метку вкладки, связанная с ней радиокнопка становится выбранной *(:checked)*, что позволяет применять соответствующие стили и отображать соответствующее содержимое вкладки.

Это простой пример реализации вкладок, используя только HTML и CSS. Однако, обратите внимание, что для более сложного функционала, например, обработки событий или анимации, может потребоваться использование JavaScript.

JavaScript может быть использован в данном примере для добавления дополнительного функционала и улучшения пользовательского опыта. Вот несколько примеров, для чего можно было бы использовать JavaScript в данном примере с вкладками:

* Переключение вкладок по клику: вместо использования радиокнопок, можно добавить обработчики событий JavaScript для переключения вкладок при клике на метки. Это позволит пользователю переключаться между вкладками без необходимости использования радиокнопок.
* Анимация переключения вкладок: С помощью JavaScript можно добавить анимацию при переключении между вкладками, чтобы сделать переход более плавным и привлекательным для пользователя. Например, вы можете использовать CSS-анимацию или библиотеки анимации, такие как jQuery или GSAP, для создания эффектов переключения.
* Загрузка содержимого вкладок по требованию: вместо загрузки всех содержимых вкладок сразу можно использовать JavaScript для загрузки содержимого вкладок по требованию пользователя. Это позволит ускорить начальную загрузку страницы и улучшить производительность. При переключении на конкретную вкладку можно загрузить ее содержимое динамически через AJAX-запросы или другие техники.
* Динамическое создание вкладок: JavaScript позволяет динамически создавать и управлять вкладками на основе данных или пользовательских действий. Например, вы можете использовать данные из базы данных или API для создания вкладок динамически, а затем добавить соответствующие обработчики событий для переключения и загрузки содержимого.

# 3 НАПИСАНИЕ REACT-ПРИЛОЖЕНИЯ

Для реализации пользовательского интерфейса была выбрана библиотека React.js – javascript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов, которая разрабатывается и поддерживается запрещенной в России организацией. Данная библиотека позволяет разрабатывать одностраничные веб-приложения без перезагрузок, с помощью подмены данных на странице

1. Реализация маршрутизации

С помощью подключения дополнительной библиотеки “react-router-dom” можно настроить маршрутизацию в приложении (рис. 3)

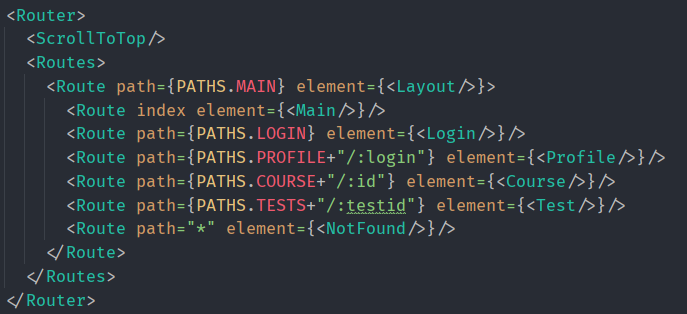


Рисунок 3 - Маршруты приложения

Также для удобства разработки пути были вынесены в перечисляемый тип enum(рис. 4)

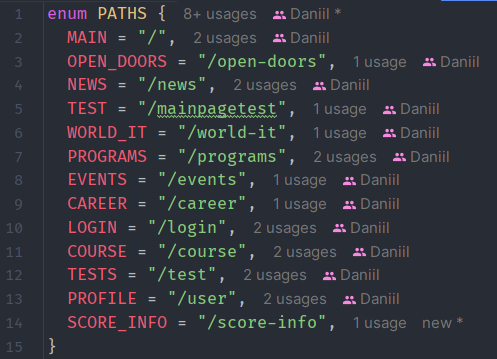


Рисунок 4. Перечисление маршрутов приложения

1. Главная страница приложения

Вверху главной страницы приложения мы можем увидеть три составные части(рис. 5). В самом верху располагается навигационная панель приложения, при нажатии на элементы которой будет происходить переход на другие страницы(рис. 5-а). На востоке от него находится метка пользователя, на данном рисунке видно, что пользователь вошел в свой аккаунт, так как отображается его имя и соответствующий роли цвет, если пользователь еще не вошел в свой аккаунт или не зарегистрировался, в этом месте будет отображаться надпись «Войти», по нажатию на которую произойдет переход на страницу регистрации и входа(рис. 5-б). Внизу рисунка отображается информация об университете, в данной области можно увидеть количество программ, преподавателей и выпускников в университете(рис. 5-в). Немного ниже этого так же можно увидеть информацию о ближайшем дне открытых дверей.

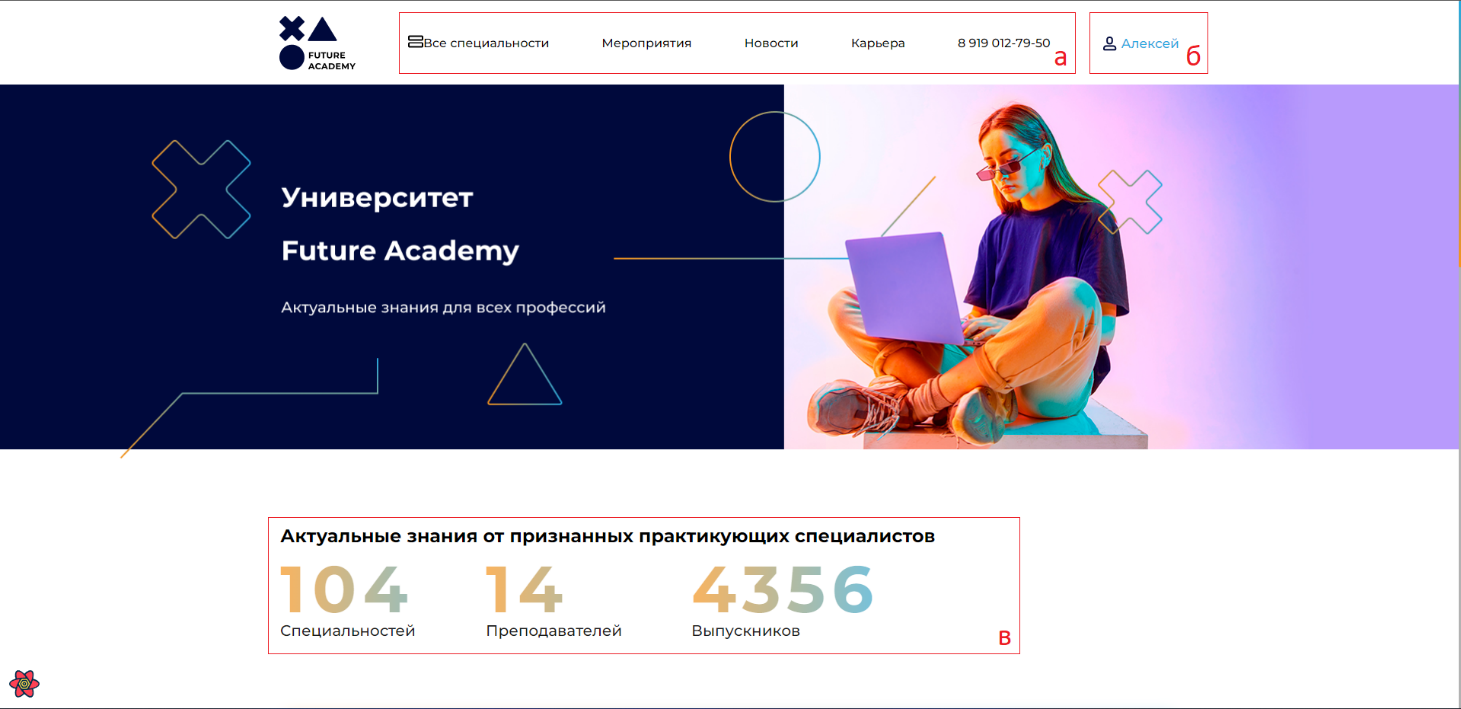


Рисунок 5. Главная страница приложения(а – навигация, б – метка пользователя, в – информация о университете)

1. Профиль пользователя

При открытии профиля пользователя мы увидим разделение на два блока: левый и правый(рис. 6). В левом блоке можно увидеть аватар пользователя, краткую информацию о его имени, почте и очках, также переключатель вкладок, который будет отображаться по-разному в зависимости от роли пользователя, а также от того находится он на странице своего аккаунта или нет(рис. 6-а). В правой части профиля можно заметить открытую вкладку «Общие сведения», в которой будет указана более подробная информация об аккаунте пользователя, данная вкладка открывается по стандарту и будет единственной доступной при открытии чужого профиля(рис. 6-б).

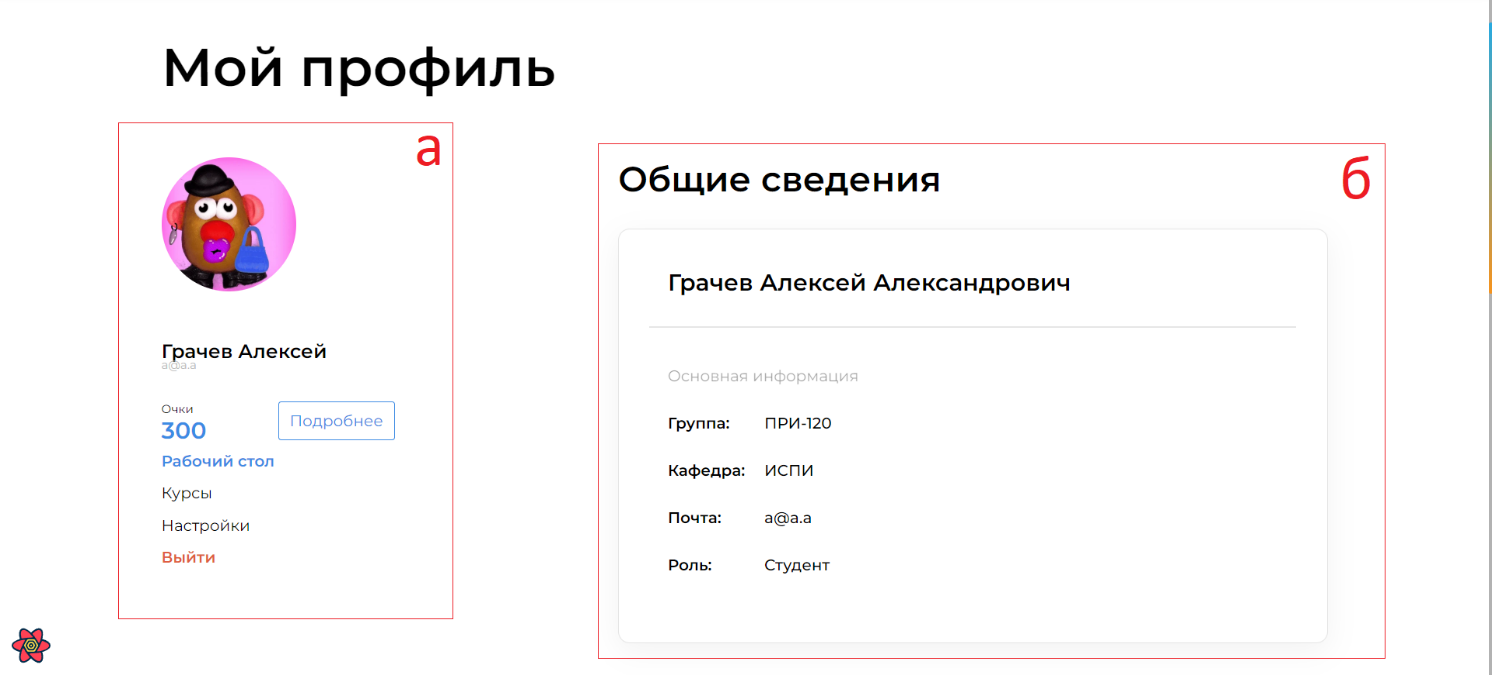


Рисунок 6. Профиль пользователя(а – краткая информация, б – вкладки)

Так же на данной странице можно перейти к курсам студента и настройкам аккаунта.

Информация о пользователе в приложении является “реактивной”, что обозначает ее обновлении на странице при изменении данных на сервере, например, если пользователь обновит данные о себе, с помощью вкладки настройки, то он увидит изменения в информации без перезагрузки страницы. Данный эффект достигается с помощью библиотеки “react-query”, которая раз в указанный период сверяет данные в браузере с данными на сервере и, в случае их отличия, подгружает новые данные, которые мгновенно показываются пользователю.

1. Регистрация, авторизация и выход из аккаунта

Перейдя на страницу входа и регистрации, если уже не вошли в аккаунт, мы можем увидеть две вкладки с формами для заполнения: вход и регистрацию(рис. 7).

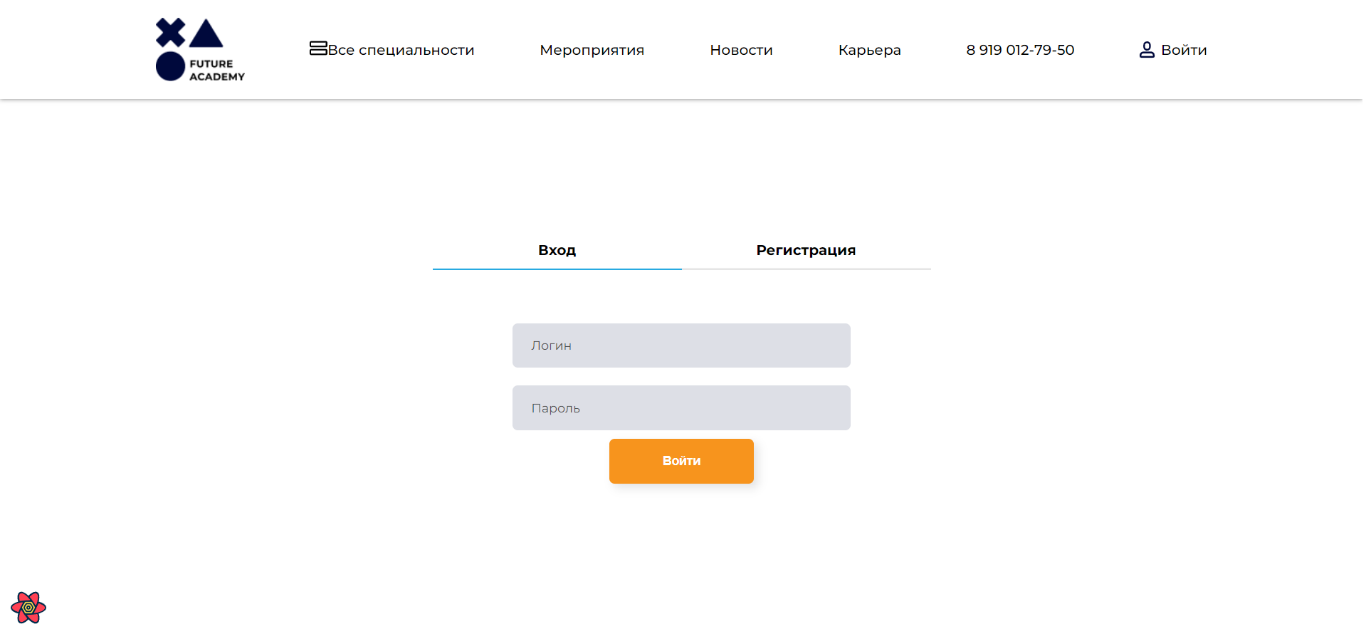


Рисунок 7. Страница входа/регистрации

Данные формы были реализованы с помощью библиотеки “react-hook-form”, которая позволяет быстро и удобно делать формы и отправлять данные с них. Когда пользователь заполнил форму, происходит валидация данных в ней. В целях безопасности данных пользователя при сохранении паролей происходит их хеширование по алгоритму SHA1.

1. Тесты и их сериализация

Зайдя под аккаунтом студента, мы можем пройти доступные нам тесты(тесты могут быть не доступны по нескольким причинам: преподаватель явно установил у теста статус «Закрыт», у теста истек срок сдачи, вы уже прошли данный тест), они будут находиться в курсе, которых находится во вкладке «Курсы» в профиле(рис. 9).

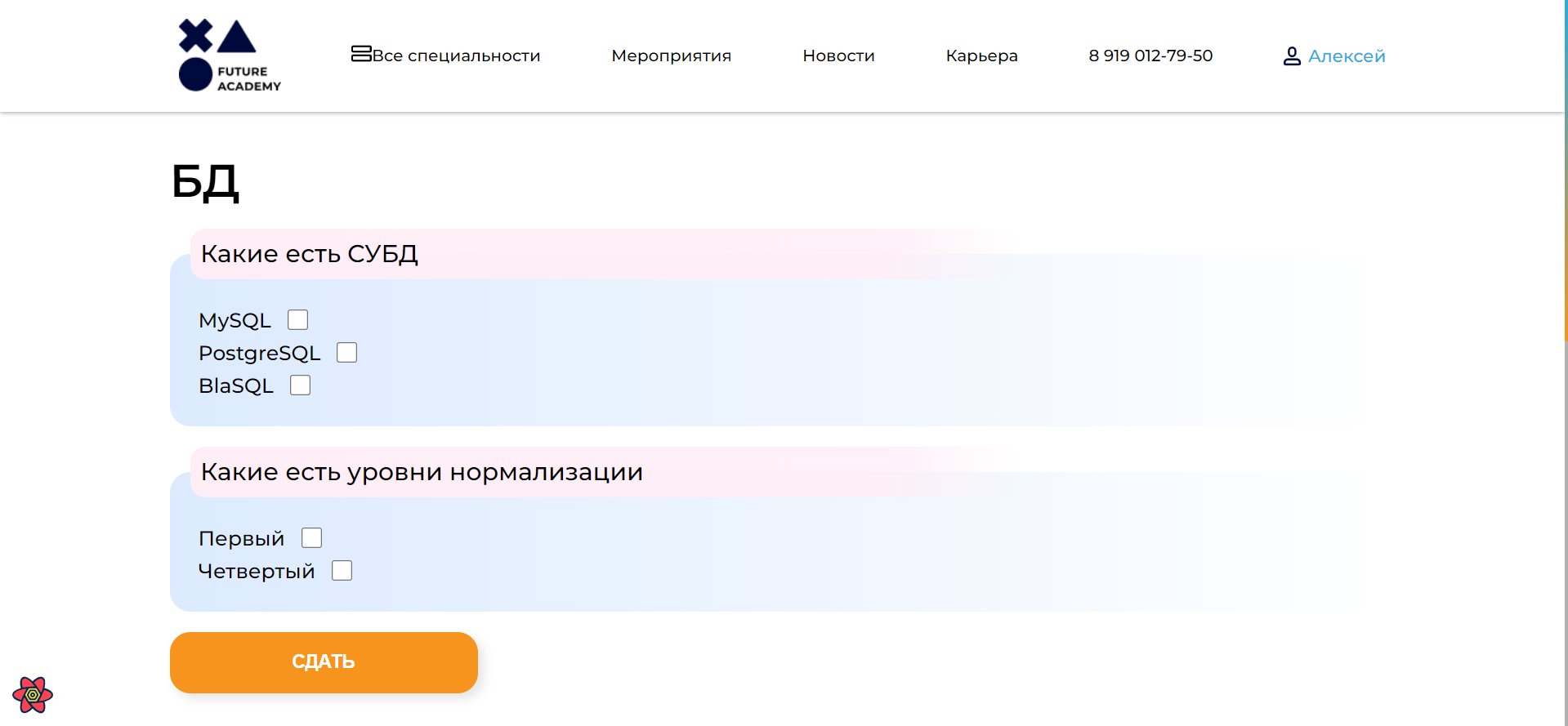


Рисунок 9. Страница теста

На странице теста будут представлены все вопросы по этому тесту, а вверху страницы будет записана тема теста. После того, как студент выберет все ответы, которые считает нужным выбрать, он может нажать кнопку сдать. После этого происходит обработка данных формы на стороне клиента: данные из формы преобразуются в читабельный для сервера вид с помощью сериалайзера(рис. 10)



Рисунок 10. Сериалайзер формы

Про оценивание тестов будет описано в пункте *2.4.3. Оценка теста*

1. Использование API

Так как приложение использует шаблон WEB-API – интерфейс общается с сервером посредством HTTP-запросов по адресу, указанному в контроллере. На уровне клиента для отправки запросов, не требующих обновления данных, как для информации о пользователях, используется библиотека axios, которая позволяет легко обратиться к API и получить ответ(рис. 11). Метод get, отвечающий за отправку HTTP-GET запроса, принимает в себя обязательный параметр “url”, который является путем до конечной точки в контроллере, который и обрабатывает этот запрос. В данном примере, после получения данных, они записываются в “стейт” компонента.

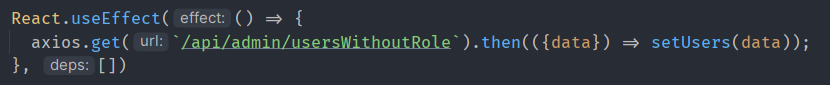


Рисунок 11. GET-запрос с помощью axios

1. Панель администратора

В панели администратора присутствует единственный интерактивный элемент – это таблица со списком новых пользователей, в которой имеется информация о логине и дате регистрации, также имеется выпадающее меню с возможностью выбрать роль для нового пользователя и кнопкой «Присвоить»(рис. 12).

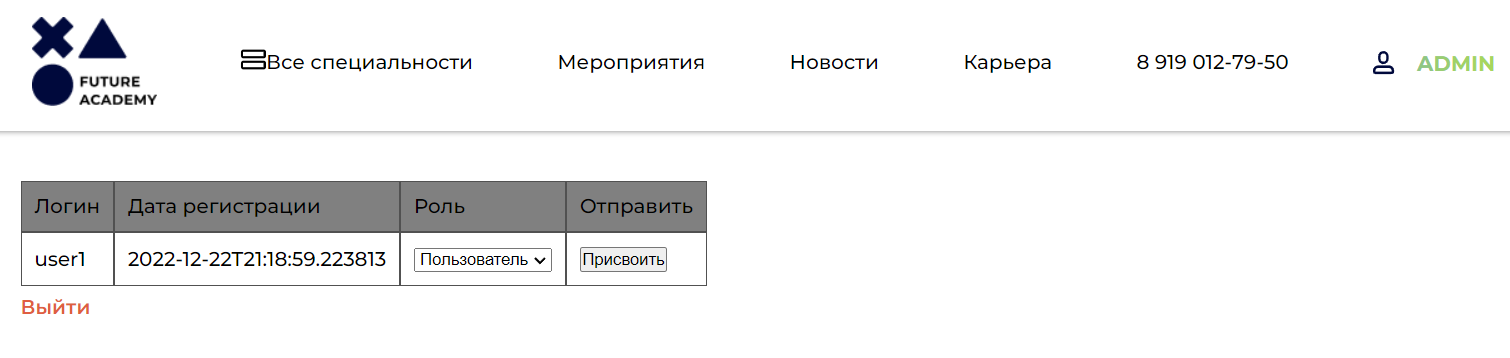


Рисунок 12. Панель администратора

# ЗАКЛЮЧЕНИЕ

В ходе производственной практики в компании, специализирующейся на разработке высоконагруженного программного обеспечения для сбора, обработки и анализа данных в реальном времени, я получил ценный опыт работы с современными технологиями. Основной целью практики было приобретение практических навыков в разработке высоконагруженных пользовательских интерфейсов, а также изучение теоретического курса, необходимого для создания качественных пользовательских интерфейсов.

В рамках практики я успешно освоил основы верстки страницы по предоставленному макету, приобрел понимание структуры HTML и стилей CSS. Я изучил принципы создания гибких и адаптивных макетов, применение различных элементов дизайна, а также методы работы с разными типами контента.

Кроме того, я ознакомился с основами разработки React-приложений. Работа с React позволила мне создавать компоненты, управлять состоянием приложения, обрабатывать события и взаимодействовать с API. Я также получил опыт работы с современными инструментами и пакетами, используемыми в разработке на базе React.

В процессе практики я столкнулся с различными задачами и вызовами, которые требовали аналитического мышления и творческого подхода к решению проблем. Я научился эффективно коммуницировать и сотрудничать в команде, а также применять лучшие практики разработки, такие как версионирование кода и использование системы контроля версий.

В результате прохождения практики я значительно расширил свои знания и практические навыки в области разработки высоконагруженных пользовательских интерфейсов. Я готов применить полученные знания и опыт на практике и продолжить свой профессиональный рост в области веб-разработки.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. MDN Web Docs [Электронный ресурс]. 2005. URL: [developer.mozilla.org](https://developer.mozilla.org/) (Дата обращения 16.03.2023)
2. React Documentation [Электронный ресурс]. 2022. URL: https://react.dev/ (Дата обращения 24.04.2023)

# ПРИЛОЖЕНИЕ А

