Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ А.С. Кузнецов
подпись инициалы, фамилия

«___ » _____ 2021 г

БАКАЛАВРСКАЯ РАБОТА

09.03.04 Программная инженерия
код и наименование специальности

Разработка Front-end части модуля анализа успеваемости студентов на основе СЭО СФУ
тема

Руководитель ВКР _____ доцент, канд. техн. наук А. В. Хныкин
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ А. А. Соколов
подпись, дата инициалы, фамилия

Красноярск 2021

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ А.С. Кузнецов
подпись инициалы, фамилия

«___ » _____2021 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Соколову Александру Александровичу
фамилия, имя, отчество

Группа КИ17-16Б Направление (специальность) 09.03.04
номер код

Программная инженерия
наименование

Тема выпускной квалификационной работы Разработка front-end части модуля анализа успеваемости студентов на основе СЭО СФУ

Утверждена приказом по университету № 5308/с от 19.04.2021 г.

Руководитель ВКР А.В. Хныкин, доцент, канд. техн. наук, ИКИТ СФУ
инициалы, фамилия, должность, ученое звание и место работы

Исходные данные для ВКР материалы, справочная и научная литература, ресурсы Интернет

Перечень разделов ВКР введение, анализ предметной области, проектирование, разработка и тестирование, описание результатов разработки, заключение, список использованных источников

Перечень графического материала презентация

Руководитель ВКР _____ А.В. Хныкин
подпись инициалы и фамилия

Задание принял к исполнению _____ А.А. Соколов
подпись инициалы и фамилия студента

«___ » _____ 20__ г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Разработка front-end части модуля анализа успеваемости студентов на основе СЭО СФУ» содержит 42 страницы текстового документа, 21 использованных источников, 27 иллюстраций.

ВЕБ-ПРИЛОЖЕНИЕ, FRONT-END, ВЕБ-ПЛАТФОРМА, АНАЛИЗ УСПЕВАЕМОСТИ.

Целью данной выпускной квалификационной работы является разработка front-end части модуля анализа успеваемости студентов на основе системы электронного образования СФУ. Разработанное веб-приложение позволяет родителям просматривать учебную информацию в удобном и лаконичном виде, а преподавателям просматривать и анализировать статистику по своим предметам, что должно помочь с определением слабых мест в образовательном процессе.

Для достижения цели были решены следующие задачи:

– проанализированы существующие решения;

– рассмотрены технические средства разработки и определён технологический стек;

– спроектированы макеты экранов приложения;

– спроектирована клиентская часть приложения.

В результате исследования предметной области были проанализированы существующие решения внутри института и был сделан вывод, что в данный момент информация показывается в виде таблиц с множественным уровнем вложенности. Проанализировать аналоги не было возможности, так как вся информация конфиденциальна, и для её просмотра необходимо являться работником, либо студентом конкретного учебного заведения. Были составлены функциональные требования к приложению.

В итоге была разработана front-end часть приложения со следующими возможностями:

– авторизация в системе;

– просмотр родителями учебной информации своего ребёнка;

– просмотр преподавателями агрегированного отчёта по выбранным группам;

– просмотр преподавателями отчёта по каждой выбранной группе отдельно.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Во время пандемии COVID-19 все учебные заведения перешли на дистанционное обучение. В связи с этим возник ряд сложностей с контролем образовательного процесса. В частности, у преподавателей возросла необходимость в просмотре посещаемости занятий и выполнении заданий, а также появилась необходимость в сравнении показателей качества образования до и во время дистанционного обучения с целью определения слабых мест образовательного процесса. Решить возникшие задачи позволила бы модернизация существующего программного обеспечения, использующегося в образовательном процессе университета.

Целью данной выпускной квалификационной работы является разработка front-end части веб-приложения, предоставляющей подробную статистику по успеваемости студентов ИКИТ СФУ.

Новая функциональность веб-приложения направлена на родителей и преподавателей. Она позволит родителям просматривать учебную информацию своего ребенка, такую как его подробную успеваемость и посещаемость в лаконичном и упорядоченном виде, а преподавателям просматривать и сравнивать (в виде графиков, гистограмм и диаграмм) успеваемость студентов по конкретным дисциплинам в разное время в агрегированном виде или по учебным группам в отдельности.

Для достижения цели были решены следующие задачи:

– проведен анализ существующих решений;

– исследована предметная область;

– создана модель предметной области;

– спроектирован интерфейс front-end части приложения;

– реализована front-end часть приложения;

– проведено тестирование.

1 Анализ предметной области

В настоящее время большинство возможностей для отслеживания успеваемости студентов, предоставляемых университетом преподавателям и родителям, представлены в виде однообразных таблиц с множественным уровнем вложенности, в которых легко запутаться.

Предлагаемое веб-приложение позволит родителям просматривать основную информацию об их ребенке, такую как его подробную успеваемость и посещаемость в лаконичном и упорядоченном виде, а преподавателям позволит просматривать и сравнивать (в виде графиков, гистограмм и диаграмм) успеваемость студентов по конкретным дисциплинам в разное время в агрегированном виде или по группам в отдельности.

Помимо анализа предметной области, для реализации проекта необходимо было выбрать технологический стек для ведения разработки.

1.1 Анализ существующих решений

В век облачных технологий большинство высших учебных заведений имеют свои версии отслеживания успеваемости студентов.

В свободном доступе были найдены электронные журналы посещаемости и успеваемости студентов Томского Государственного университета Систем управления и радиоэлектроники (ТУСУР), Казанского федерального университета и Южного института менеджмента (ЮИМ), но на каждом сайте необходимо проходить процедуру авторизации, то есть информацию могут просматривать только учащиеся и преподаватели данного учебного заведения.

Так как я не имею никакого отношения к данным университетам, провести полный анализ их приложений невозможно.

1.2 Определение требований к системе

Так как не было возможности проанализировать аналоги, ключевые функциональные требования к системе были составлены из пожеланий родителей и преподавателей:

– авторизация в системе посредством ввода логина и пароля;

– страница родителя должна содержать шапку и блок с основной информацией. Шапка содержит: выпадающий список с семестрами студента и строку прогресса текущего семестра. Основной блок содержит: ФИО студента, его процент посещаемости, общий процент успеваемости, позицию в группе и таблицу, содержащую название предмета, процент выполнения курса, его форму контроля, дату проведения аттестации и оценку;

– страница преподавателя должна содержать блок фильтров и блок отчёта. Блок фильтров содержит: выбор предмета, множественный выбор года набора, множественный выбор групп, соответствующих тем годам, которые были выбраны выше, radiobutton для выбора формы отчета: агрегированный или отдельно по каждой группе и кнопку «Показать». Блок агрегированного отчета содержит: название всех групп, суммарное количество студентов, суммарное количество студентов, которые сдали предмет (получили оценку за экзамен\зачет), график со средней посещаемостью лекций каждой группы, график со средней посещаемостью практик каждой группы, гистограмму с общим прогрессом на электронных курсах, круговую диаграмму с количеством долгов (берутся данные о количестве просроченных заданий) и круговую диаграмму с информацией о том, на какую оценку студенты сдали экзамен. Блок отчета по группам в отдельности содержит: меню для выбора группы и аналогичную информацию, что и в агрегированном отчете, только для конкретной группы.

1.3 Выбор средств front-end разработки

Для разработки front-end части выбор был между тремя фреймворками\библиотеками JavaScript [11, 12], такими как Vue.js, React и Angular. Каждая из них является инструментом для создания SPA (Single Page Application), плюсом которого является создание приложения на основе компонентов, с помо-

щью которых можно работать только с одним html файлом, просто меняя структуру DOM-дерева. Рассмотрим подробнее каждый из них.

Vue.js – это прогрессивный JavaScript-фреймворк, выпущенный в 2014 году Эваном Ю, используется для создания одностраничных пользовательских веб-приложений любой сложности. Плюсы Vue.js:

– Vue.js схож по характеристикам с Angular, что помогает оптимизации блоков из-за использования набора компонентов;

– Vue.js имеет подробную, проработанную документацию, ускоряющую процесс обучения. Используя базовые знания HTML и JavaScript можно начинать разрабатывать приложения в данном фреймворке;

– Vue.js – «лёгкий» фреймворк, занимающий около 20КБ, что позволяет ему быть производительнее других фреймворков;

– Vue.js позволяет создавать шаблоны различной сложности за небольшой промежуток времени.

Минусы:

– Vue.js не такой популярный, как React или Angular, что означает, что найти решение какой-либо проблемы будет труднее [18].

React – это JavaScript библиотека, разработанная в 2013 году компанией Facebook [3], как и Vue.js, используется для разработки одностраничных веб-приложений [1]. Плюсы данной библиотеки:

– JSX, благодаря которому можно создавать шаблоны, используя схожий с HTML синтаксис;

– имеет подробную документацию на 17 языках (на данный момент разрабатывается ещё для 9 языков);

– эта библиотека довольно быстрая, так как использует React Virtual DOM при рендере компонентов;

– поддерживает Server Side Rendering [5], что уменьшает нагрузку на клиентскую сторону;

– можно писать код, используя TypeScript [11] или Flow, которые имеют встроенную поддержку JSX;

– простая миграция между версиями;

– кроме разработки веб-приложений React можно использовать в мобильной разработке, установив библиотеку React Native [14];

– использование «хуков», которые позволяют писать только функциональные компоненты, не используя классовые.

Минусы:

– использование JSX и логики приложения в одном файле может сбить с толку новых разработчиков.

Angular – это JavaScript-фреймворк от Google, использующий паттерн MVVM, выпущенный в 2010 году, отлично подходит для создания интерактивных одностраничных веб-приложений [20]. Плюсы Angular:

– Angular использует при разработке TypeScript;

– двусторонняя привязка данных, которая сводит к минимуму риск возможных ошибок;

– использование паттерна MVVM (Model-View-ViewModel).

Минусы:

– из-за строгой типизации, Angular использует множество различных структур, что увеличивает порог вхождения для новых разработчиков, по сравнению с React и Vue.js;

– работа с основным DOM-деревом негативно сказывается на производительности фреймворка.

Проанализировав достоинства и недостатки библиотек и фреймворков, был осуществлен выбор конкретного инструмента для создания проекта. Front-end часть было решено разрабатывать на библиотеке React, так как благодаря функциональным компонентам код, написанный на React проще писать и редактировать, не используя лишние конструкторы и методы, как это было раньше. Теперь в React приложениях можно управлять жизненными циклами и со-

стоянием компонента с помощью React Hooks [4], что увеличивает производительность всего приложения. Также React имеет большое количество вспомогательных библиотек [16].

В качестве графической библиотеки была выбрана Semantic UI React [9], которая содержит набор готовых различных дизайнерских компонентов и подробную документацию.

Так как приложение содержит векторные графики, то необходимо выбрать библиотеку для создания SVG графиков. Выбор был из двух библиотек Rechart и Nivo. Обе эти библиотеки написаны на основе популярной графической JavaScript библиотеки D3.js.

Rechart прост в изучении и реализации, но не поддерживает некоторые виды графиков, необходимых для выполнения работы, поэтому была выбрана библиотека Nivo [8], которая имеет интерактивную документацию и позволяет создавать и модернизировать различные графики.

В качестве редактора кода выбран Visual Studio Code.

Visual Studio Code – бесплатный редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. В отличии от других IDE он достаточно «лёгкий» и «гибкий» для кроссплатформенной разработки, что позволяет использовать его для разработки любых приложений и на любых языках. Включает в себя такие инструменты, как:

– отладчик;

– инструменты для работы с Git;

– подсветку синтаксиса;

– средства для рефакторинга.

Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации.

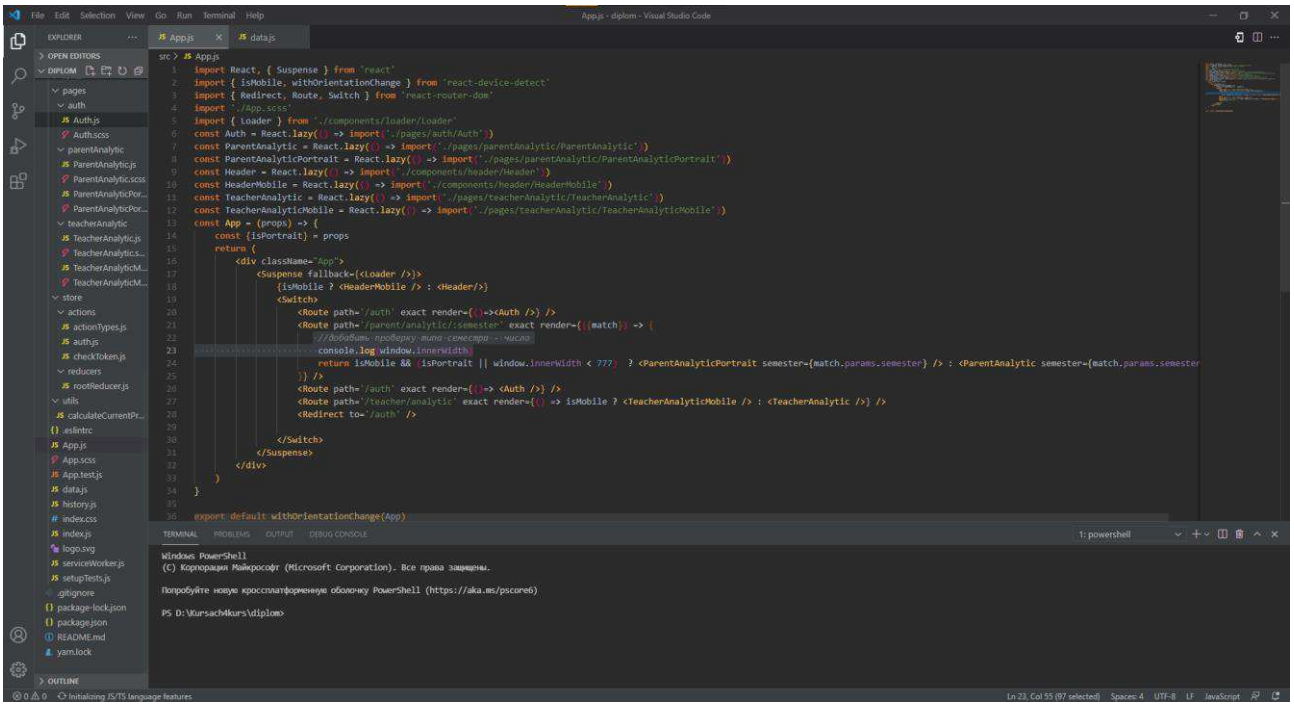

Рисунок 1 – Visual Studio Code

В качестве программы для разработки макета выбран Balsamiq Mockups.

Balsamiq Mockups – графический редактор для построения макетов пользовательских интерфейсов, с целью упрощения разработки веб-сайтов, мобильных и десктопных приложений. Позволяет создавать и настраивать различные окна и компоненты отдельно, а затем сохранять их в качестве шаблона. Также позволяет взаимодействовать между различными командами, так что член команды может сотрудничать с различными совместно расположенными или удаленно расположенными командами. Его способность экспортировать каркасы в форматы PDF или PNG обеспечивает гибкость при совместном использовании каркасов с командами.

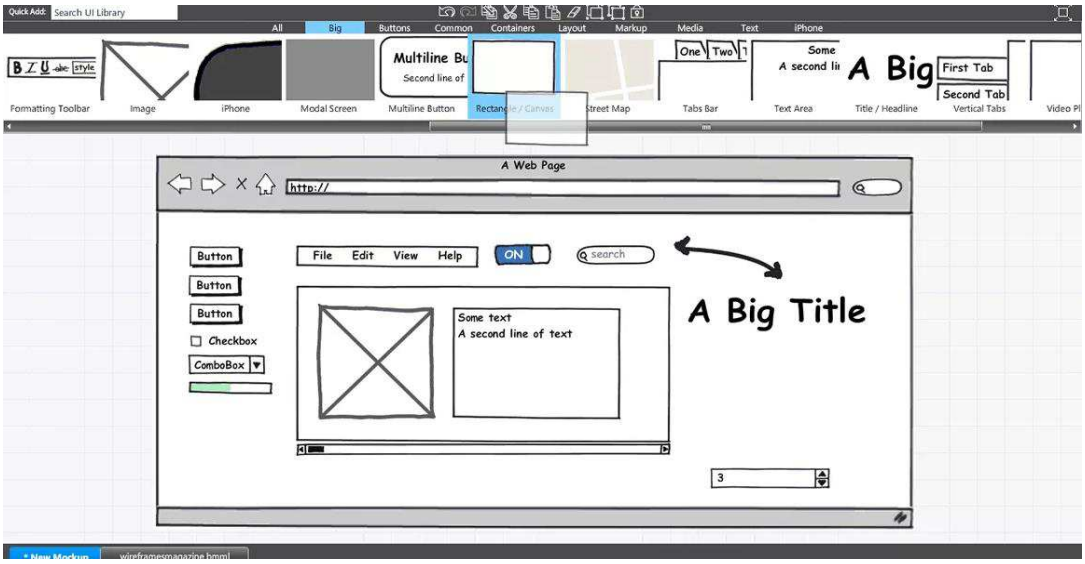

Рисунок 2 – Интерфейс Balsamiq Mockups

При разработке веб-приложения использовалась система контроля версий Git, репозиторий размещался на хостинге Github. Git является распределенной системой, что означает, что в один момент времени главная копия проекта может находиться на нескольких машинах. Git позволяет как легко просматривать изменения между версиями, так и быстро перемещаться между ними, совершать откаты, слияния и прочее.

1.4 Методология разработки

Гибкая методология разработки является одним из самых популярных подходов к разработке программного обеспечения. Одним из способов реализации данного подходя является Scrum [10]. Данных подход использовался при разработке данного мобильного приложения.

Гибкая модель разработки позволяет вносить изменения в систему на любой из этапов разработки. Данная модель является приоритетной при разработке данного приложения, так как на каждом этапе проектирования появляются новые идеи различных функций системы.

Основные принципы Scrum:

– работа в небольших командах, что подходит под рамки дипломной работы;

– собрания для отслеживания кто и что делает на каждом из этапов разработки проекта;

– работа производится по итерациям, что позволяет следить за развитием продукта.

Визуализация процесса работы над проектом проводилось в web-сервисе Trello.

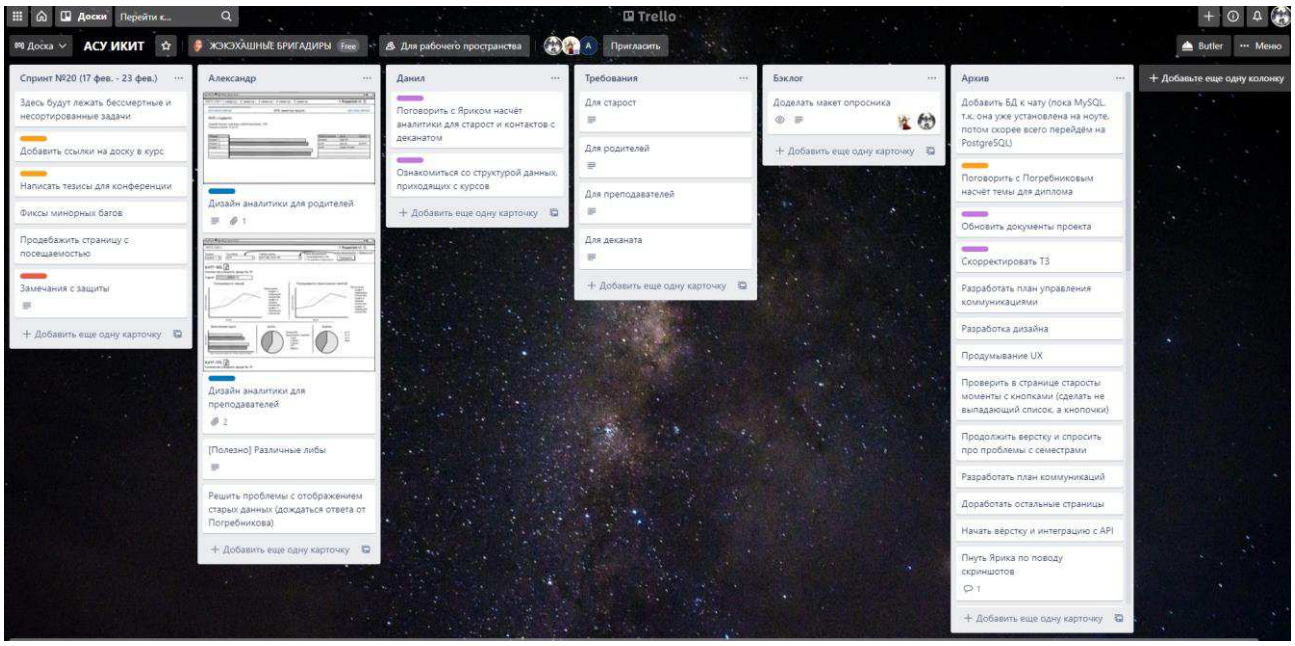

Рисунок 3 – Доска на Trello

1.5 Выводы по разделу

Целью раздела являлось изучение предметной области, в результате которого были собраны основные требования к разрабатываемому веб-приложению.

Были выбраны средства разработки и полностью определен технологический стек: React – библиотека построения Front-end, Semantic UI React – библиотека дизайн-компонентов, Nivo – графическая библиотека для создания графиков, VS Code – среда разработки, Balsamiq Mockups – сервис для создания макетов. В качестве системы контроля версий было решено использовать Git и разместить репозиторий на платформе GitHub.

2 Проектирование

2.1 Архитектура информационной системы

Архитектура веб-приложения представляет из себя отношения и взаимодействия между пользовательскими интерфейсами, сервисами обработки данных, базой данных и так далее. Основная цель – удостовериться, что все элементы корректно отрабатывают друг с другом.

Все веб-приложения состоят из двух частей – клиентской (front-end) и серверной (back-end), которая также включает в себя базу данных [13, 15]. Клиентская часть отображает пользователю графический интерфейс и реагирует на запросы, серверная часть обрабатывает и отвечает на поступающие запросы.

Основные функции веб-приложений делятся на уровни. Что позволяет разрабатывать или обновлять каждый слой независимо. Существует четыре основных уровня веб-приложения [2]:

– уровень представления (PL);

– уровень обслуживания (DSL);

– уровень бизнес-логики (BLL);

– уровень доступа к данным (DAL).

Уровень представления представляет из себя пользовательский интерфейс, взаимодействие с пользователем. Данный уровень предоставляет необходимую информацию клиентской стороне: отображает данные для пользователя, различные графические компоненты. Также существуют компоненты, которые задают взаимодействие с пользователем. Основная цель уровня – получить входные данные, обработать пользовательские запросы, отправить их на другой уровень и отобразить результаты.

Уровень бизнес-логики отвечает за обмен данными. Он определяет логику бизнес-операций и правил.

Уровень службы является связующим звеном между представлением и бизнес-логикой. Он передает данные, обработанные уровнем бизнес-логики, на

уровень представления, гарантируя безопасность данных и изолируя бизнес-логику от стороны клиента.

Уровень доступа к данным предоставляет доступ к вызываемым данным, находящимся в хранилищах, к примеру текстовые файлы, XML-файлы или двоичные файлы. Также этот уровень управляет CRUD-операциями (create, read, update, delete).

2.2 Типы архитектуры веб-приложений

Существует несколько типов архитектуры веб-приложений, в зависимости от того, как распределяется логика приложения между клиентской и серверной частями [15]. Наиболее распространенные и часто используемые архитектуры [2]:

– одностраничные приложения (SPA);

– многостраничные приложения (MPA);

– прогрессивные веб-приложения (PWA);

– архитектура микросервисов.

Разберем каждый вид в деталях.

2.2.1 Одностраничные приложения

Single-Page Application – это веб-приложение, которое загружает всю необходимую информацию при первом входе в приложение. Одностраничные приложения имеют преимущество – они обеспечивают удобный пользовательский интерфейс и пользовательский опыт, так как компоненты на странице заменяются без полной перезагрузки страницы. Одностраничные веб-приложения разрабатываются с использованием различных JavaScript библиотек и фреймворков, например: Angular, React, Vue.js.

Наиболее популярные SPA: Gmail (глобальный почтовый сервис Google), Facebook (мировая социальная сеть), Twitter (мировая социальная сеть), Slack (корпоративный мессенджер), Вконтакте (социальная сеть, наиболее популярная в странах СНГ).

2.2.2 Многостраничные приложения

Multiple-Page Application в прошлом являлось популярным типом архитектуры, так как все веб-сайты были MPA. Сейчас компании реже выбирают MPA, только в том случае, если их веб-сайт довольно большой и содержит много отличающихся от себя страниц (например, eBay или Amazon). Такие приложения каждый раз перезагружают веб-страницу при загрузке или отправке информации на пользователей.

Наиболее популярные MPA: eBay (популярный интернет-аукцион и интернет-магазин), Amazon (популярный интернет-магазин).

2.2.3 Прогрессивные веб-приложения

В последнее время прогрессивные веб-приложения набирают популярность. Данные веб-приложения работают как самостоятельные мобильные приложения. PWA используют push-уведомления, offline доступ, возможность установки приложения на домашний экран.

Также такие приложения могут использовать дополнительные функции устройства – NFC, геолокации, Bluetooth и другие.

Наиболее популярные PWA: Uber (приложение для вызова такси), Starbucks (всемирная сеть кофеен), Pinterest (веб-сервис, позволяющий пользователям добавлять и делиться рисунками, фотографиями с другими пользователями).

2.2.4 Архитектура микросервисов

Микросервисная архитектура позволяет разработчикам создавать веб-приложение из различных сервисов. Каждый компонент создаётся и разворачивается отдельно и независимо.

Данная архитектура выгодна для больших и сложных проектов, поскольку каждый компонент можно «вытащить» и изменить, и это никак не повлияет на работу системы в целом.

Наиболее популярные сервисы: Netflix (сервис для просмотра фильмов и сериалов), Spotify (сервис потокового аудио, позволяющий легально прослушивать музыкальные композиции, аудиокниги и подкасты, не скачивая их на устройство), PayPal (крупнейшая дебетовая электронная платёжная система).

2.3 Интерфейс веб-приложения

В настоящее время успех веб-сайта во многом зависит от его интерфейса [19]. Грамотно спроектированный интерфейс прост в освоении, его основные функции вынесены на передний план, а лишние спрятаны. Расположение элементов управления влияет на удобство работы с сайтом, а его внешний вид на первое впечатление и удовольствие от использования.

Так как проблемы с определением целевой аудитории отсутствуют (родители и преподаватели), то можно выделить основные требования к интерфейсу:

– лаконичность и простота;

– использование цветовой гаммы согласно стандартам СФУ;

– использование шрифтов согласно стандартам СФУ;

– отсутствие рекламы;

– адаптация интерфейсов под различные разрешения экрана;

– наличие мобильной версии.

В ходе разработки интерфейса веб-приложения были созданы макеты основных страниц приложения. Макеты страниц представлены на рисунках 4-7.

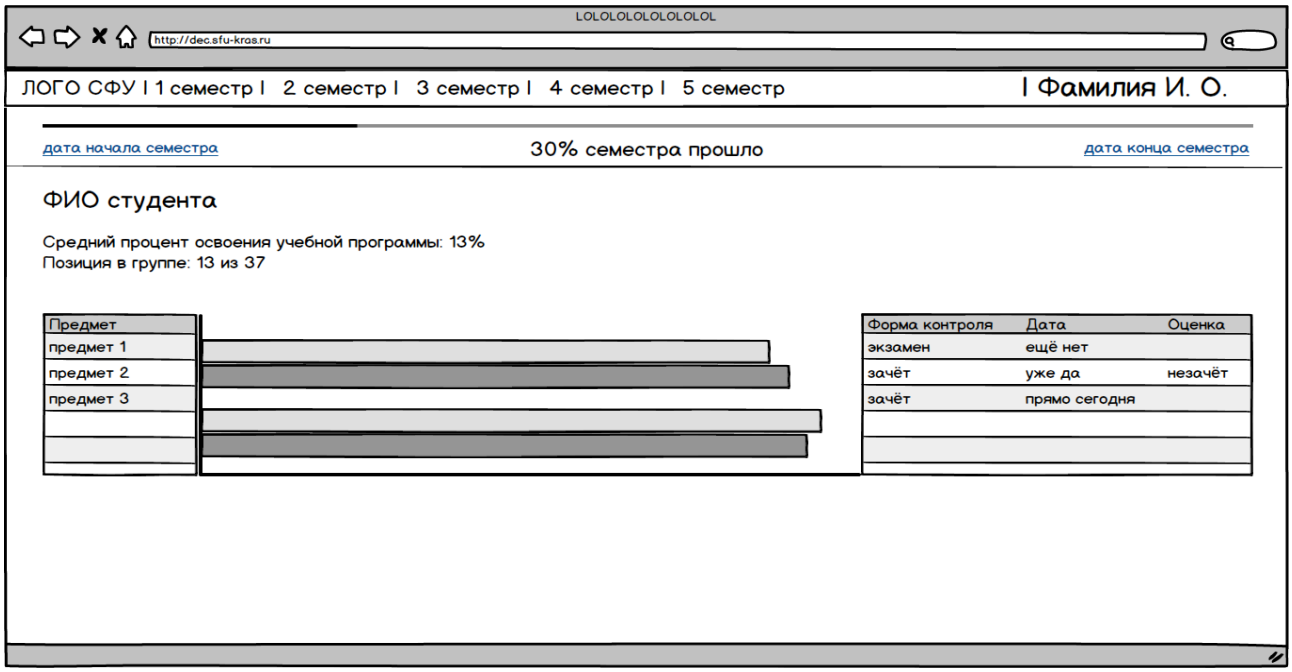

Рисунок 4 – Макет страницы родителя (Desktop)

Рисунок 5 – Макет страницы родителя (Mobile)

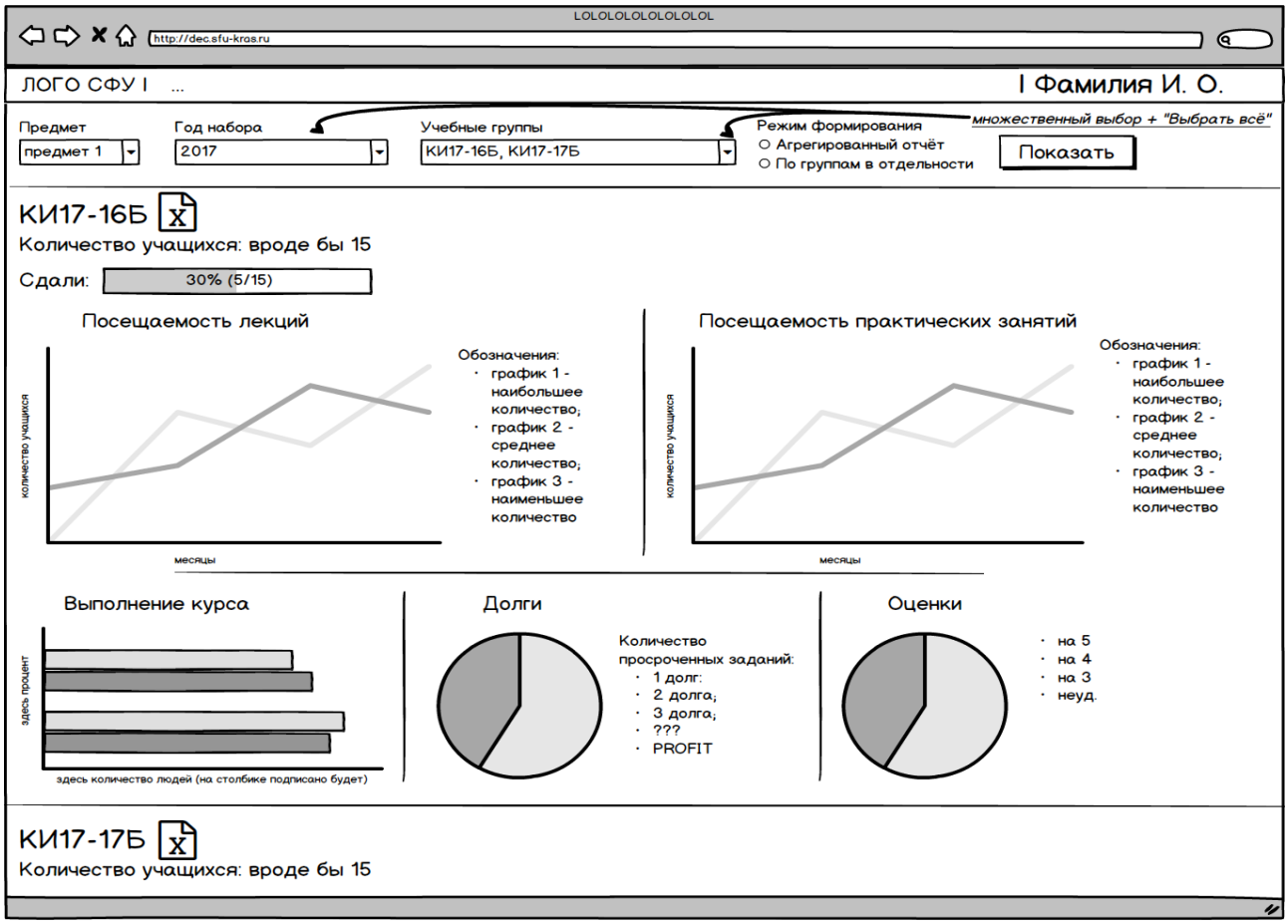

Рисунок 6 – Макет страницы преподавателя (Desktop)

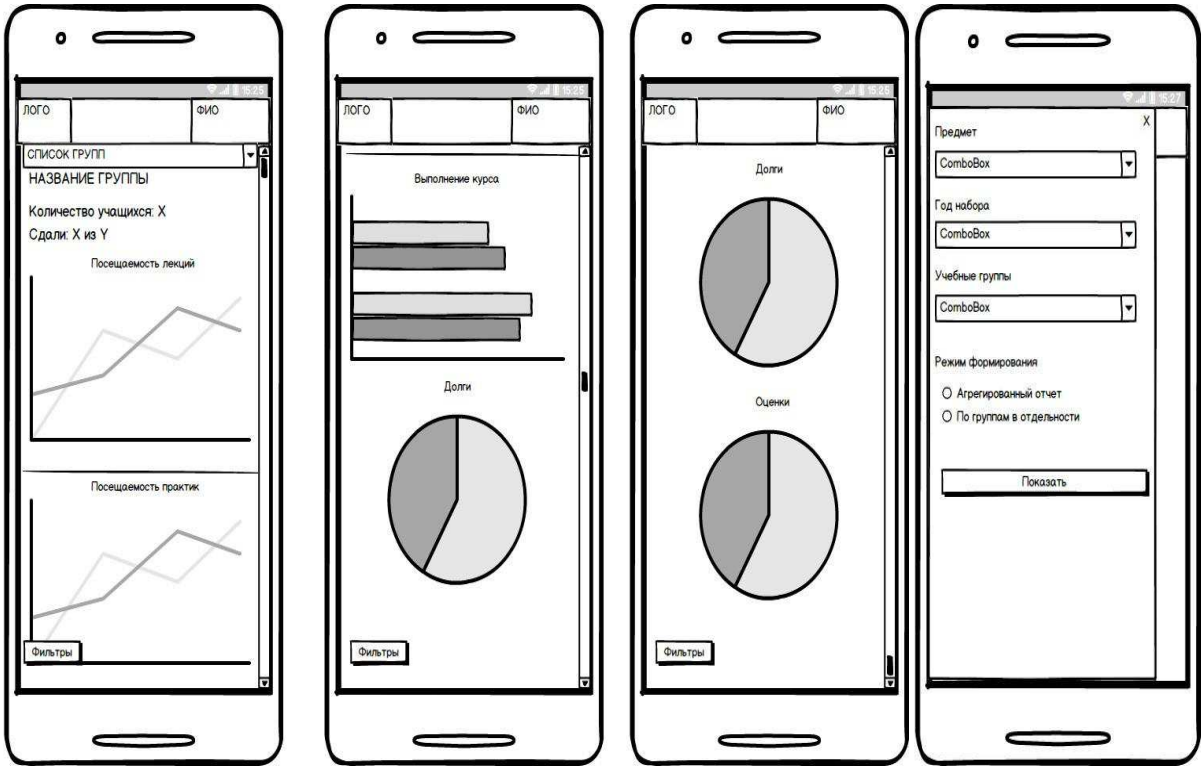

Рисунок 7 – Макет страницы преподавателя (Mobile)

2.4 Модуль API

Взаимодействие с сервером является одной из основных частей приложения, так как от нее зависит получение пользователем каких-либо данных, а также их отправление на сервер. Хорошо спроектированное взаимодействие с сервером позволит сэкономить время и память на ненужных запросах, а также предупреждать пользователя о возникновении ошибок при запросе. Также, стоит учитывать, что выбранная архитектура приложения – это клиент-сервер, что означает равную значимость как взаимодействие клиента и сервера, так и взаимодействие пользователя и интерфейса.

Модуль API будет реализован при помощи библиотеки Axios [7], которая позволит в простой форме выполнять запросы типа GET/POST/DELETE/PUT, настраивать тела запроса, обрабатывать ошибки и т.д.

Для авторизации пользователя приложение будет посылать запрос с логином и паролем, в ответ на который, при верно введенных данных и наличии пользователя в системе, придет токен, с помощью которого можно будет в дальнейшем идентифицировать пользователя и посылать следующие запросы.

Благодаря библиотеке Axios, можно в простой форме посылать запросы на сервер. Для этого достаточно вызвать функцию create с URL-адресом сервера и необходимыми заголовками, затем вызывать функцию в соответствии с типом запроса, передавая в параметрах необходимые поля. Ответ от сервера приходит в формате JSON.

2.5 Диаграммы вариантов использования

Диаграмма вариантов использования (англ. use case diagram) – это диаграмма, на которой изображаются актёры и варианты использования (также называются прецедентами) выполненная с использованием унифицированного языка моделирования UML (англ. Unified Modeling Language) [6]. Диаграммы используются для определения взаимодействия пользователя с системой, описывая какие действия может осуществлять актер [21].

Разработанные диаграммы представлены на рисунках 8-9.

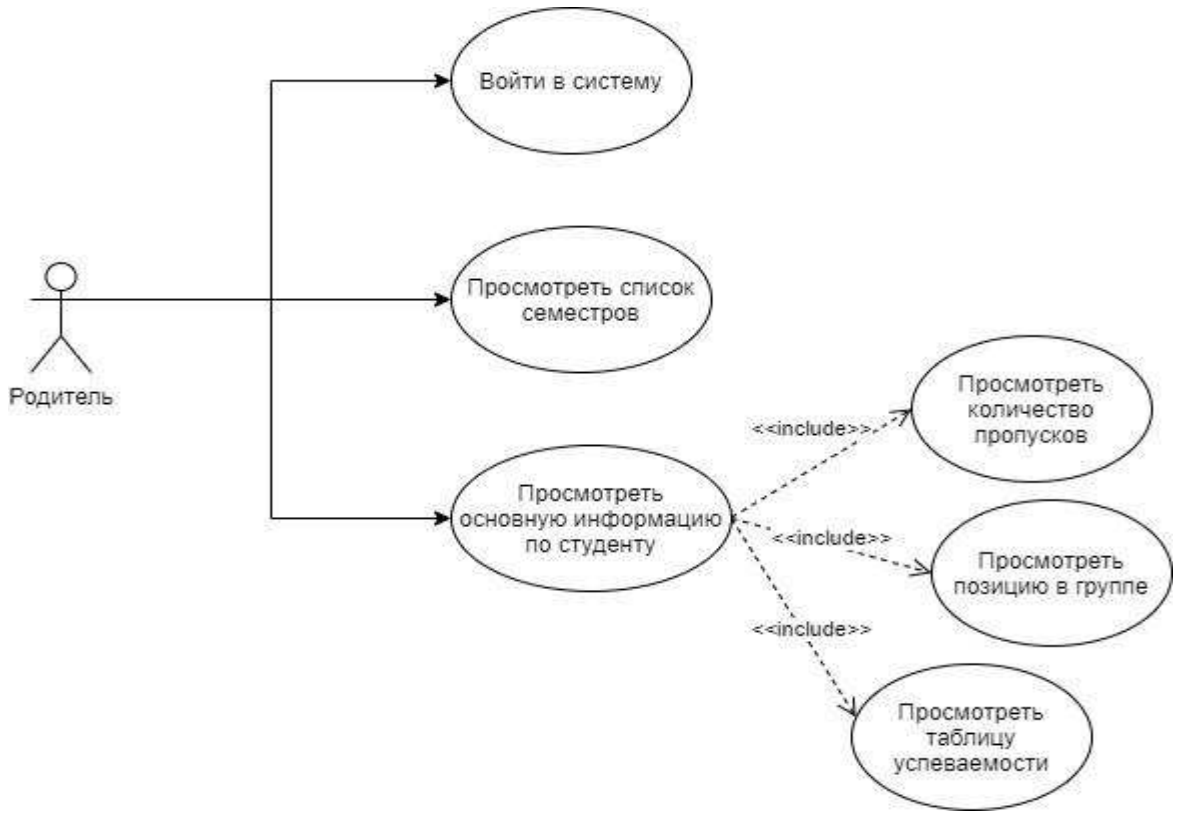

Рисунок 8 – Диаграмма вариантов использования для родителей

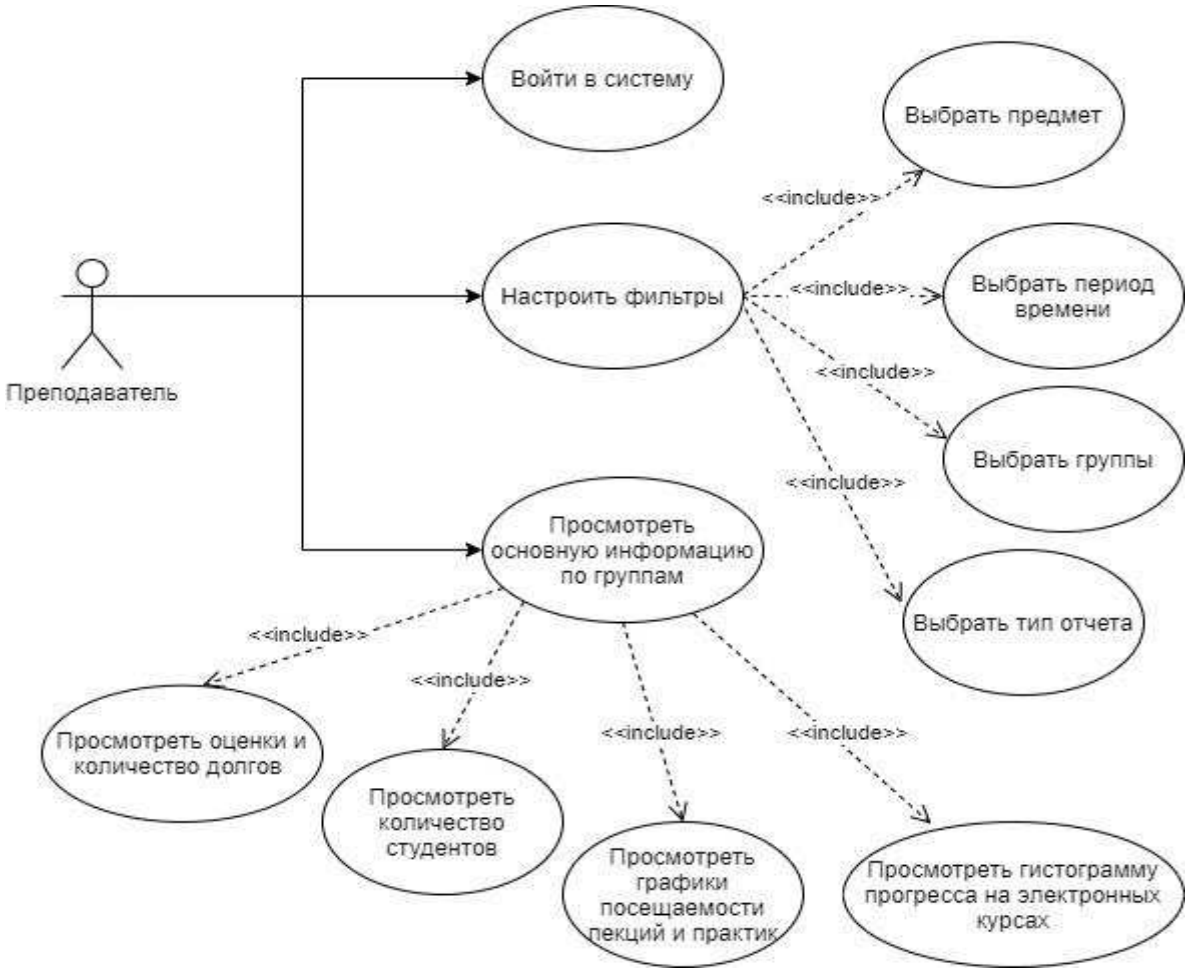

Рисунок 9 – Диаграмма вариантов использования для преподавателей

2.6 Выводы по разделу

Целью раздела являлось описание проектирования разрабатываемой веб-платформы. Были описаны архитектуры веб-приложений, представлены их особенности. Также был разработан макет интерфейса приложения, построены диаграммы вариантов использования и представлены требования к интерфейсу веб-приложения.

3 Разработка и тестирование

Началом этапа разработки является определение файловой структуры проекта, представленной на рисунке 10.

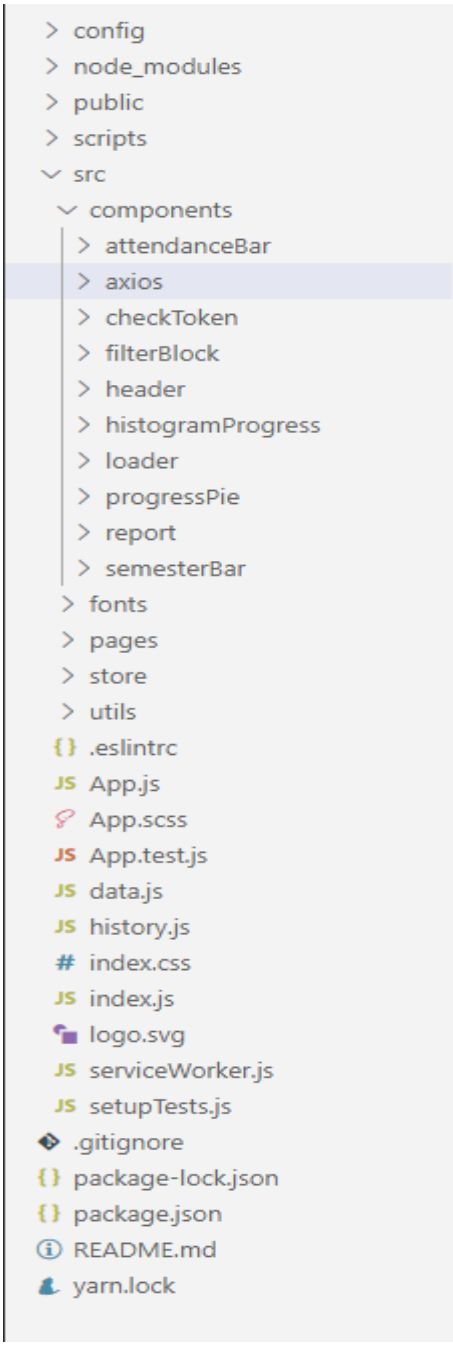

Рисунок 10 – Файловая структура проекта

Файлы с настройками проекта находятся в корневом каталоге.

Корневой каталог включает в себя:

– config – содержит настройки Webpack;

– node_modules – папка, для хранения всех npm пакетов;

– public – содержит html-файл приложения;

– scripts – содержит скрипты для тестирования, постройки и запуска приложения;

– src – содержит все React компоненты и файлы со стилями.

3.1 Front-end

Начальным этапом разработки front-end части проекта являлось создание корневых React компонентов для каждого раздела [17]. В директории src была создана директорая pages, содержащая главные компоненты всех разделов (рисунок 11): Auth (раздел авторизации), ParentAnalytic (раздел для родителей) и TeacherAnalytic (раздел для преподавателей).

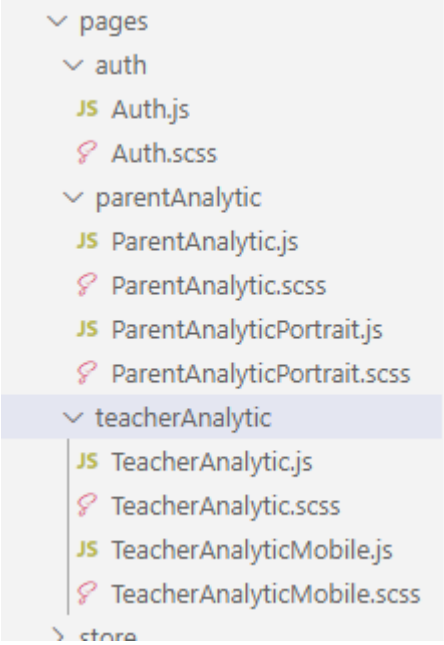

Рисунок 11 – Директория pages, с главными компонентами разделов

Промежуточные компоненты, используемые в главных, хранятся в директории components.

Пример React компонента представлен на рисунке 12. Компонент Header является панелью с логотипом СФУ, списком семестров студента (для родителей) и информацией об авторизированном пользователе.

```
const Header = ({user, getUser}) => {
    const [isVisible, setVisible] = useState(false)

    useEffect(() => {
        if(!user) {
            const token = localStorage.getItem('HEADER_TOKEN')
            getUser(token)
        }
        return () => {

        }
    // eslint-disable-next-line react-hooks/exhaustive-deps
    }, [])
    const logout = () => {
        localStorage.removeItem('HEADER_TOKEN')
        history.push('/auth')
    }
    return (
        <header className="header">
            {user ?
                <>
                    <Link to='/profile' className="header__logo">
                        <img src={logo} alt="logo" className="header__logo-img" />
                        <div className="header__logo-text">АСУ ИКИТ</div>
                    </Link>
                    <div className="header__navbar">
                        {user.type === 'parent' ? menuList.map((menu, index) => {
                            return(
                                <React.Fragment key={`menuItem__${index}`}>
                                    <div className="dropmenu">
                                        <div className="text dropmenu__text">
                                            {menu.course} курс 
                                            <Icon name="dropdown" size='small'/>
                                        </div>
                                        <div className="dropmenu__content">
                                            <div className="dropmenu__content-elem"><Link className='dropmenu__content-elem-link' to={`/parent/analytic/${menu.sem1}`}>{menu.sem1} семестр</Link></div>
                                            <div className="dropmenu__content-elem"><Link className='dropmenu__content-elem-link' to={`/parent/analytic/${menu.sem2}`}>{menu.sem2} семестр</Link></div>
                                        </div>
                                    </div>
                                </React.Fragment>
                            )
                        }) : null}
                    </div>
                    <div className="header__FIO">
                        <div  onClick={()=>setVisible(!isVisible)} className="header__FIO-name">
                            {`${user.surname} ${user.name[0]}.${user.patronymic[0]}.`} 
                            <Icon name="dropdown" size='small'/>
                        </div>
                        <div onBlur={()=>setVisible(false)} onClick={()=>setVisible(!isVisible)} className={`header__FIO-dropmenu ${isVisible ? 'dropvisible' : 'dropunvisible'}`}>
                            <div className="header__FIO-dropmenu-elem" onClick={logout}>Выйти</div>
                        </div>
                    </div>
                </>: null}

        </header>
```

Рисунок 12 – Компонент Header

При разработке компонентов использовались React Hooks [4], которые появились в обновлении библиотеки 16.8. Разработка с помощью «хуков» позволяет создавать проекты, состоящие только из функциональных компонентов, в отличии от прошлых версий React, где приходилось создавать классовые и функциональные компоненты, что делало код загруженным и менее лаконичным. В проекте использовались такие «хуки», как: useState (для управления состоянием компонента) и useEffect (для управления побочными эффектами, к примеру обращение к API или обновление DOM-дерева). Пример данных «хуков» показан на рисунке 13.

```
const [isVisible, setVisible] = useState(false)

useEffect(() => {
    if(!user) {
        const token = localStorage.getItem('HEADER_TOKEN')
        getUser(token)
    }
    return () => {

    }
// eslint-disable-next-line react-hooks/exhaustive-deps
}, [])
const logout = () => {
```

Рисунок 13 – Пример «хуков» useState и useEffect

Здесь useState используется для изменения состояния переменной isVisible, отвечающей за отображение выпадающего меню, а useEffect при рендере компонента проверяет, если пользователь не найден, то запросить данные из API по уникальному токену.

Для упрощения отслеживания ошибок использовалась библиотека PropTypes, которая выполняет валидацию переменных компонента. Если происходит несовпадение типов данных, то библиотека посылает предупреждение в консоль браузера. Пример использования библиотеки представлен на рисунке 14.

```
Auth.propTypes = {
    username: PropTypes.string.isRequired,
    password: PropTypes.string.isRequired,
    error: PropTypes.string,
}
```

Рисунок 14 – Пример использования PropTypes в компоненте Auth

Для работы с состоянием всего приложения использовалась библиотека Redux, которая хранит необходимые данные в глобальном объекте store. Для изменения глобального хранилища используется чистая функция reducer. Для вызова reducer используется функция dispatch, передающая объект action, с новыми значениями и событием, по которому reducer будет редактировать состояние.

Для хранения всех событий и «редьюсеров» была создана директория store, содержащая директории actions и reducers. Далее в файл index.js (файл, с которого приложение запускается) были импортированы необходимые функции, объекты и компоненты из библиотек redux, react-redux и redux-thunk, был создан объект store, хранящий глобальное состояние приложения и передан в компонент Provider. Создание глобального хранилища показано на рисунке 15.

```
import {applyMiddleware, createStore} from 'redux'
import {Provider} from 'react-redux'
import rootReducer from './store/reducers/rootReducer'
import thunk from 'redux-thunk'

const store = createStore(rootReducer, applyMiddleware(thunk))
ReactDOM.render(
    <React.StrictMode>
        <Provider store={store}>
            <Router history={history}>
                <App />
            </Router>
        </Provider>
    </React.StrictMode>,
    document.getElementById('root')
)
```

Рисунок 15 – Файл index.js

Для отправки запросов к серверу использовалась библиотека axios [7]. Все фукнции для работы с запросами были созданы в директории actions, так как после ответа от сервера необходимо менять глобальное состояние (рисунок 16).

```
function loginStart() {
    return {
        type: LOGIN_START
    }
}

function loginEnd() {
    return {
        type: LOGIN_SUCCESS
    }
}

function loginError(e) {
    return {
        type: LOGIN_ERROR,
        error: e
    }
}
export const login = ({username, password}) => {
    return async dispatch => {
        try {
            dispatch(loginStart())
            const response = await axios().post('/auth/login', {
                username,
                password,
                appToken: APP_TOKEN
            })
            const {data} = response
            if(data.status === 'error') {
                dispatch(loginError('Неверный логин или пароль'))
            } else {
                const {usertoken} = data
                localStorage.setItem('HEADER_TOKEN', usertoken)
                dispatch(loginEnd())
                history.push('/profile')
            }
        } catch (e) {
            dispatch(loginError(e))
        }
    }
}
```

Рисунок 16 – Авторизация пользователя

На рисунке 16 продемонстрирована функция login для отправки данных пользователя на сервер. Так как используется redux, то вначале посылается событие, указывающее на начало запроса (обновится store, который «сообщит»

подписанным компонентам, что необходимо отобразить loader), далее идёт сам запрос, затем, в зависимости от ответа, посылается событие на отображение ошибки или на сохранение полученного токена.

Для разработки векторных диаграмм и графиков использовалась библиотека Nivo, содержащая набор различных React-компонентов.

На рисунке 17 показан компонент, созданный на основе существующих компонентов Nivo [8].

```
import {ResponsiveBar} from '@nivo/bar'

export const HistogramProgress = ({data, className}) => {
    const tooltip = bsp => {
        if(bsp.value === 1) return <div>{`${bsp.value} студенто выполнил курс на ${bsp.indexValue}`}</div>
        if(bsp.value < 5) return <div>{`${bsp.value} студента выполнили курс на ${bsp.indexValue}`}</div>
        return <div>{`${bsp.value} студентов выполнили курс на ${bsp.indexValue}`}</div>
    }
    const viewSettings = {
        margin: className=='mobile' ? { top: 50, right: 20, bottom: 80, left: 60 } : { top: 10, right: 130, bottom: 50, left: 100 },
    }
    return (
        <ResponsiveBar
            data={data}
            keys={[ 'students' ]}
            indexBy="progress"
            margin={viewSettings.margin}
            padding={0.1}
            layout="horizontal"
            valueScale={{ type: 'linear' }}
            indexScale={{ type: 'band', round: true }}
            colors={'#D13C16'}
            minValue={0}
            borderColor={{ from: 'color', modifiers: [ [ 'darker', 1.6 ] ] }}
            axisTop={null}
            axisRight={null}
            axisBottom={{
                tickSize: 5,
                tickPadding: 5,
                tickRotation: 0,
                legend: 'Количество студентов',
                legendPosition: 'middle',
                legendOffset: 32
            }}
            axisLeft={{
                tickSize: 5,
                tickPadding: 5,
                tickRotation: 0,
                legend: 'Процент выполнения',
                legendPosition: 'middle',
                legendOffset:-70
            }}
            label={d => `${d.value}`}
            labelSkipWidth={12}
            labelSkipHeight={12}
            labelTextColor={'#fafafa'}
            animate={true}
            motionStiffness={90}
            motionDamping={15}
            tooltip={bsp => tooltip(bsp)}
            borderWidth={1}
        />
    )
}
```

Рисунок 17 – Компонент HistogramProgress

Данный компонент принимает необходимый набор значений и возвращает гистограмму успеваемости студентов в электронных курсах. Компонент содержит набор настроек, таких как: цвет шкалы, кастомный вид легенды и осей X и Y, различные параметры анимации и отступы, в зависимости от типа устройства (смартфон или компьютер).

3.2 Тестирование

Тестирование компонентов производилось с использованием библиотеки React Test Library. Данная библиотека даёт простые инструменты, построенные на базе библиотеки react-dom. Все тесты взаимодействуют с реальными узлами DOM-дерева, что напоминает реальный сеанс работы с приложением. Функции, созданные в библиотеке, обращаются к DOM-дереву эмулируя поведения реального пользователя. К примеру поиск ссылок, кнопок и различных элементов выполняется по тексту.

На рисунке 18 показан файл с тестированием компонента авторизации Auth, в котором проверяется корректность сохранения информации при вводе логина и пароля и отображение окон ошибки или успеха. При помощи функции describe можно разделять тесты на логические блоки, функции beforeEach и afterEach позволяют работать с фикстурами, функция beforeEach вызывается перед каждым тестом, функция afterEach – после.

```
let nameInput, passInput
describe('UI testing with out fake fetch', () => {
    beforeEach(() => {
        root = render(<Auth />)
        nameInput = root.getByTestId('name')
        passInput = root.getByTestId('pass')
    })

    test('change input name', async() => {
        fireEvent.change(nameInput, { target: {value: 'ASokolov-KI17'}})
        expect(nameInput.value).toBe('ASokolov-KI17')
    })
    test('should error window hide', () => {
        expect(root.queryByTestId('error')).not.toBeTruthy()
    })
    test('should select change', () => {
        expect(passInput.value).toBe('12345')
        fireEvent.change(passInput, { target: {value: '12345'}})
        expect(passInput.value).toBe('12345')
    })
    test('should error window show and 1 input empty', () => {
        fireEvent.change(nameInput, { target: {value: 'ASokolov-KI17'}})
        fireEvent.change(passInput, { target: {value: ''}})
        const btn = root.getByTestId('btn_submit')
        fireEvent.click(btn)
        expect(root.queryByTestId('error')).toBeTruthy()
    })
    test('should error window show', () => {
        const btn = root.getByTestId('btn_submit')
        fireEvent.click(btn)
        expect(root.queryByTestId('error')).toBeTruthy()
    })


    afterEach(cleanup)
})
const mockRequest = jest.fn()
describe('UI testing with fake fetch', () => {
    beforeEach(() => {
        root = render(<Auth request={mockRequest}/>)
        nameInput = root.getByTestId('name')
        passInput = root.getByTestId('pass')

        fireEvent.change(nameInput, { target: {value: 'ASokolov-KI17'}})
        fireEvent.change(passInput, { target: {value: '12245'}})
    })
    afterEach(cleanup)
    test('should all inputs not empty and call request function', () => {
        const btn = root.getByTestId('btn_submit')
        fireEvent.click(btn)
        expect(mockRequest).toHaveBeenCalledTimes(1)
        expect(root.queryByTestId('error')).not.toBeTruthy()
        expect(root.queryByTestId('success')).toBeTruthy()
    })
})
```

Рисунок 18 – Файл с тестами компонента Auth

Результаты тестирования предоставлены на рисунке 19.

```
> react-scripts test
 PASS  src/App.test.js
  UI testing with out fake fetch
    √ change input name (59 ms)
    √ should error window hide (16 ms)
    √ should select change (25 ms)
    √ should error window show and 1 input empty (35 ms)
    √ should error window show (14 ms)
  UI testing with fake fetch
    √ should all inputs not empty and call request function (38 ms)

Test Suites: 1 passed, 1 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        3.614 s
Ran all test suites.
```

Рисунок 19 – Результаты тестирования компонента Auth

3.3 Выводы по разделу

Целью раздела являлось описание ключевых моментов разработки и тестирования клиентской части.

В разделе 3.1 была описана работа с библиотеками React, Redux, Nivo и Axios для разработки веб-приложения, учитывая их особенности и нововведения. В разделе 3.2 была описана работа с библиотекой React Test Library для тестирования приложения.

4 Описание результатов разработки

4.1 Front-end

Результатом разработки front-end части веб-приложения являются страницы:

– страница авторизации;

– страница родителей, для просмотра информации о студенте;

– страница преподавателей с агрегированным отчётом;

– страница преподавателей с подробным отчётом по каждой группе.

Все страницы и компоненты веб-приложения адаптированы под различные экраны десктопных устройств. Также существует мобильная версия, в которой отображение некоторых компонентов отличается от десктопной. Вышеприведённые страницы представлены на рисунках 20-27. В целях конфиденциальности за основу контента взяты тестовые данные.

Страница авторизации (рисунок 20) содержит поля для ввода логина и пароля и кнопку «Войти», после ввода корректных данных происходит отображение одной из страниц (для родителей или для преподавателей).

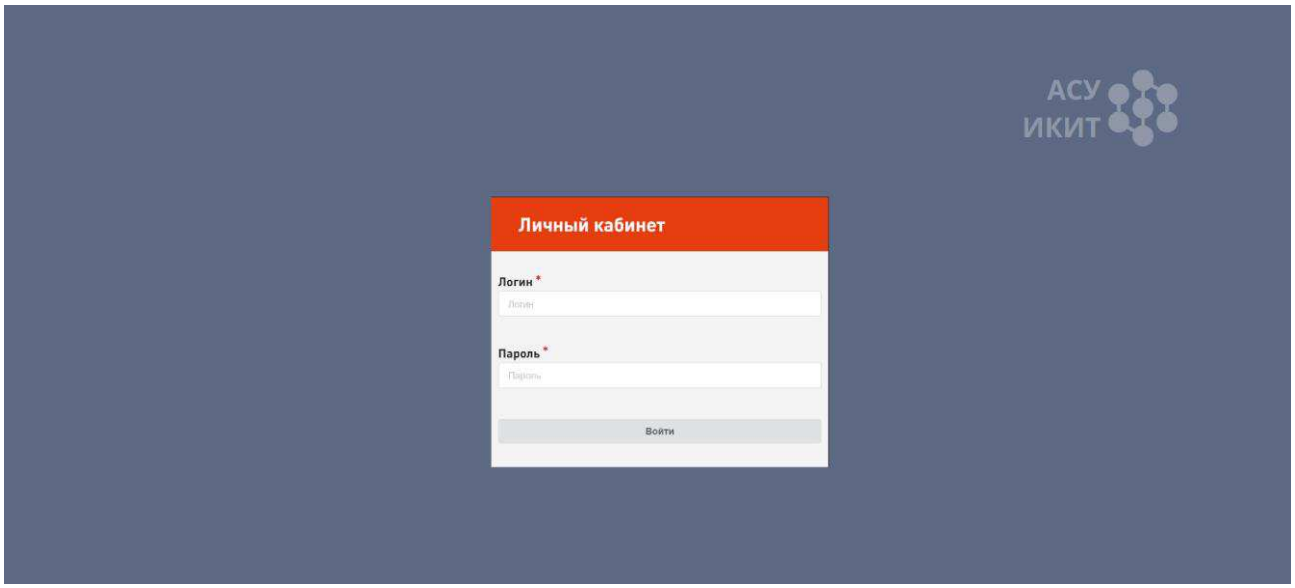

Рисунок 20 – Страница авторизации

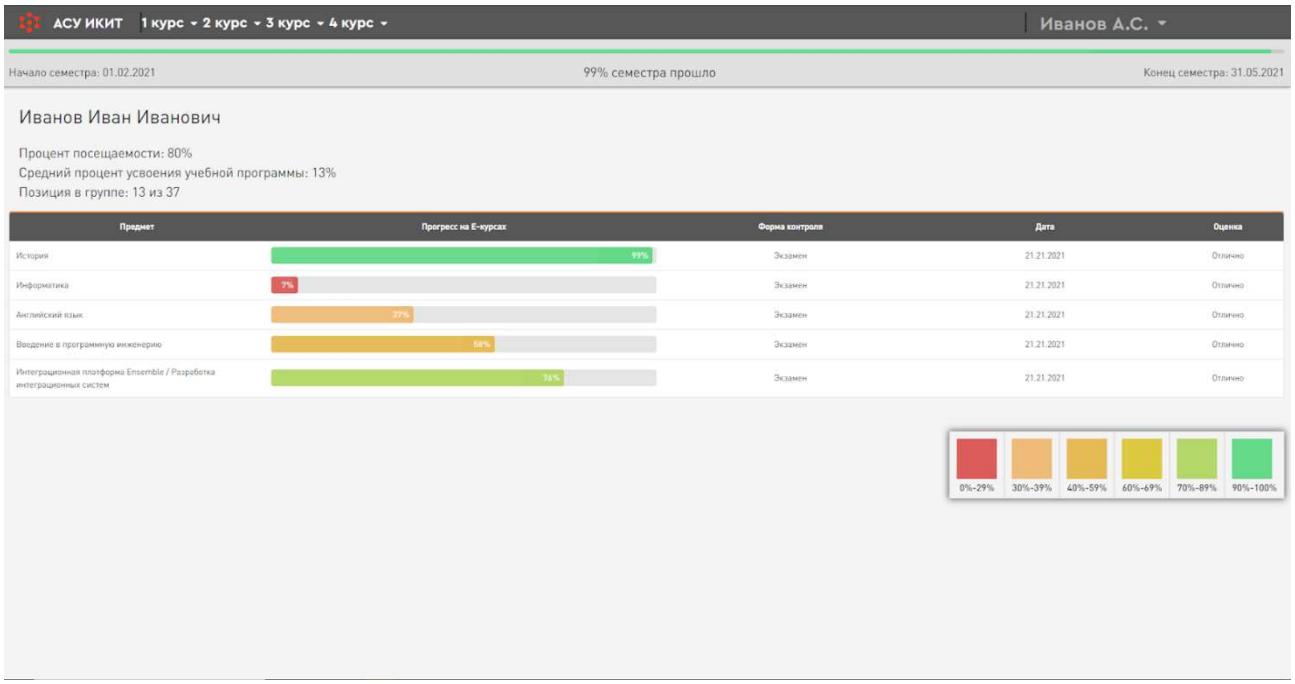

Рисунок 21 – Страница родителя

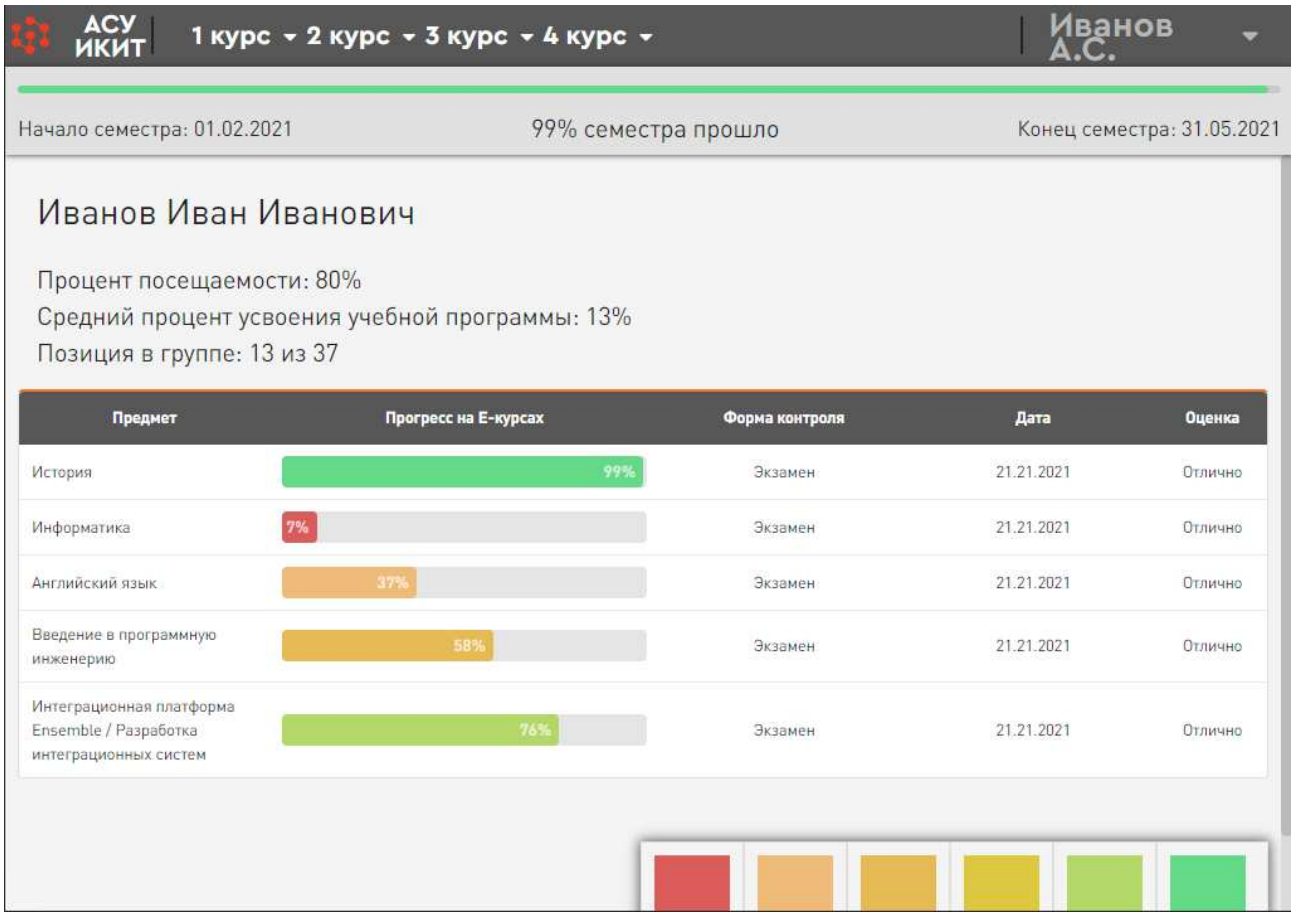

Рисунок 22 – Страница родителя (адаптивная для небольших экранов)

Данная страница содержит информацию по выбранному семестру, в виде шкалы прогресса и отображает основную информацию о студенте для родителя:

– ФИО студента;

– процент посещаемости (высчитывается из электронного журнала);

– средний процент усвоения учебной программы (высчитывается из прогресса в электронных курсах);

– позиция в группе (высчитывается из рейтинга по усвоению учебной программы внутри группы, позиция анонимна, то есть можно знать позицию только своего ребёнка);

– таблица с информацией по каждому предмету (название предмета, прогресс в электронных курсах, форма контроля, дата проведения аттестации и оценка).

Мобильная версия отличается тем, что в «шапке» находится одна вкладка «курсы», при нажатии на которую появляется меню с выбором курса, и таблица заменяется на «карусель» карточек каждого предмета (рисунок 23).

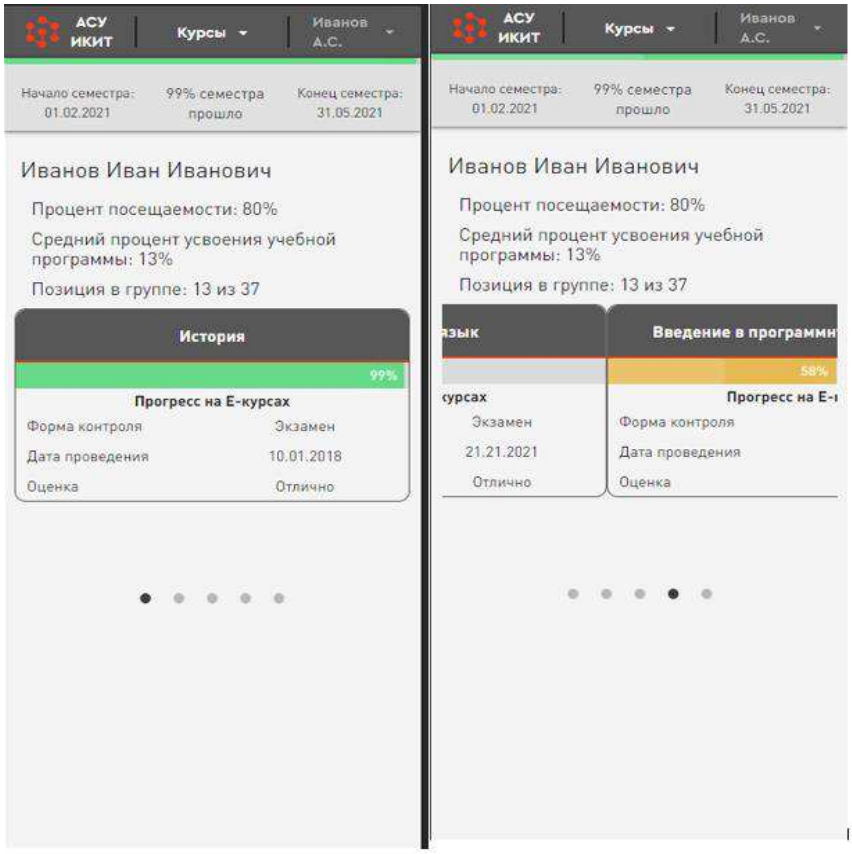

Рисунок 23 – Мобильная версия страница родителя

Страница преподавателя состоит из двух основных блоков: блок фильтров и блок отчёта (рисунок 24). Блок фильтров содержит:

– поле с выбором предмета;

– поле с выбором года обучения (присутствует поддержка множественного выбора);

– поле с выбором учебных групп (присутствует поддержка множественного выбора, группы отображаются в зависимости от выбора дат в поле выше);

– выбор между агрегированным отчётом и по группам в отдельности.

Блок отчёта зависит от выбора типа отчёта.

Окно с агрегированным отчётом содержит в себе:

– список выбранных групп;

– суммарное количество студентов;

– суммарное количество студентов, которые сдали предмет (получили оценку за экзамен\зачет);

– график со средней посещаемостью лекций каждой группы;

– график со средней посещаемостью практик каждой группы;

– гистограмму с общим прогрессом в электронных курсах;

– круговую диаграмму с количеством долгов (берутся данные о количестве просроченных заданий);

– круговую диаграмму с информацией о том, на какую оценку студенты сдали предмет.

Рисунок 24 – Страница агрегированного отчёта

В мобильной версии блок с фильтрами скрыт и появляется при нажатии на кнопку «фильтры», все графики центрируются и становятся друг под другом (рисунок 25).

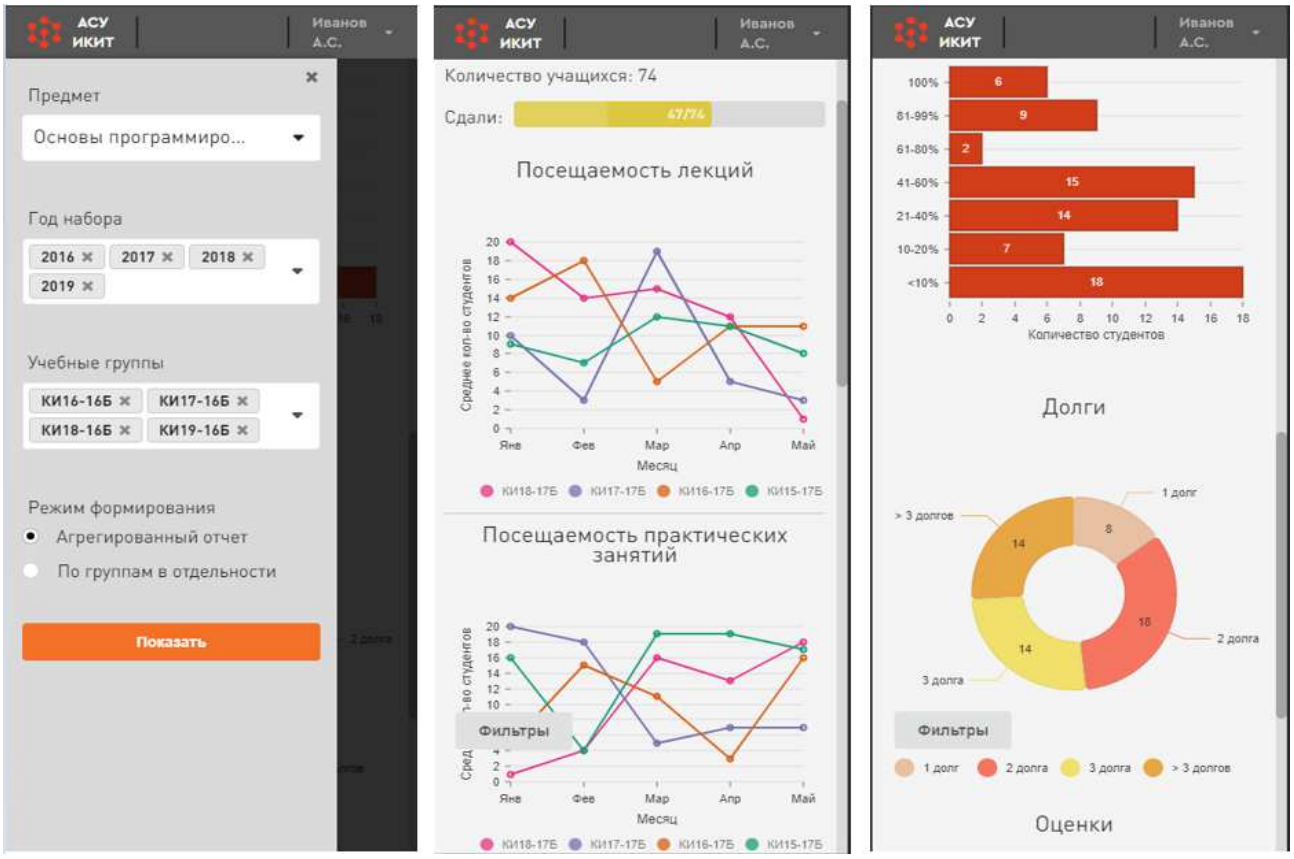

Рисунок 24 – Мобильная версия страница агрегированного отчёта

Окно с отчётом по группам отдельно (рисунок 26) содержит в себе:

– меню с выбором группы;

– количество студентов выбранной группы;

– количество студентов, которые сдали предмет (получили оценку за экзамен\зачет);

– график со средней посещаемостью лекций группы;

– график со средней посещаемостью практик группы;

– гистограмму с прогрессом в электронных курсах;

– круговую диаграмму с количеством долгов (берутся данные о количестве просроченных заданий);

– круговую диаграмму с информацией о том, на какую оценку студенты сдали предмет.

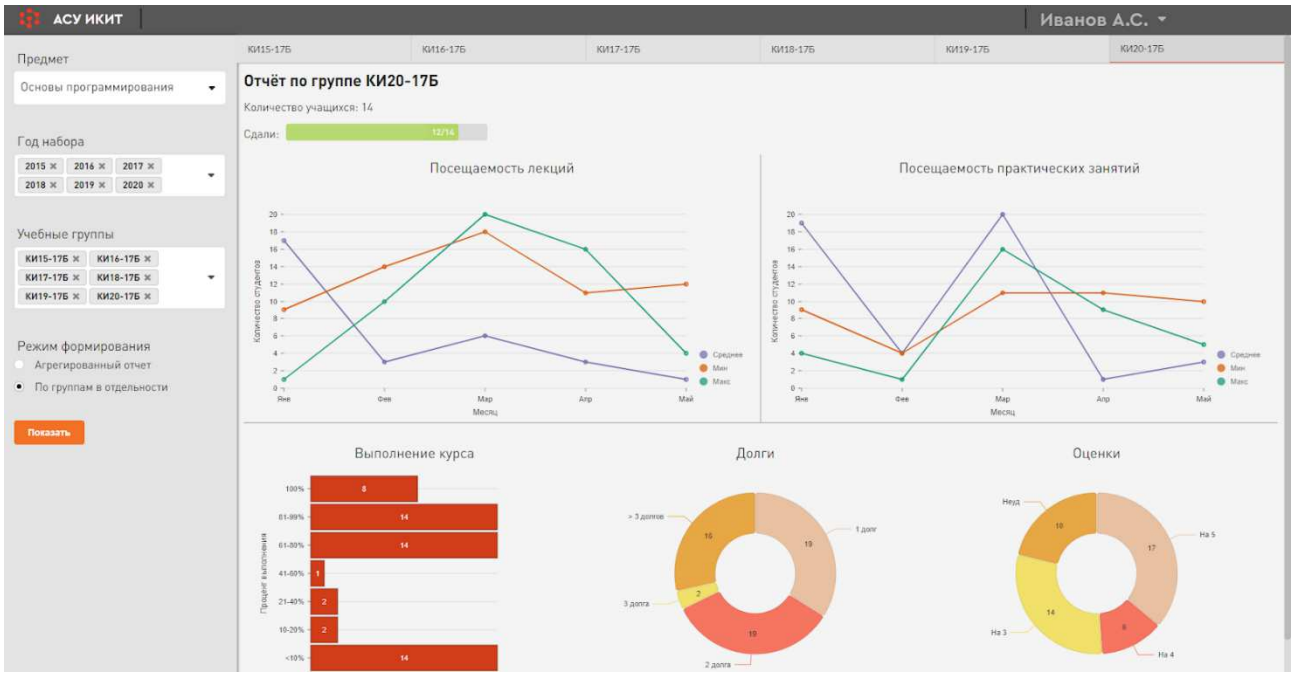

Рисунок 26 – Страница отчёта по группам в отдельности

Мобильная версия схожа с версией агрегированного отчёта. Меню с выбором группы заменяется на выпадающий список (рисунок 27).

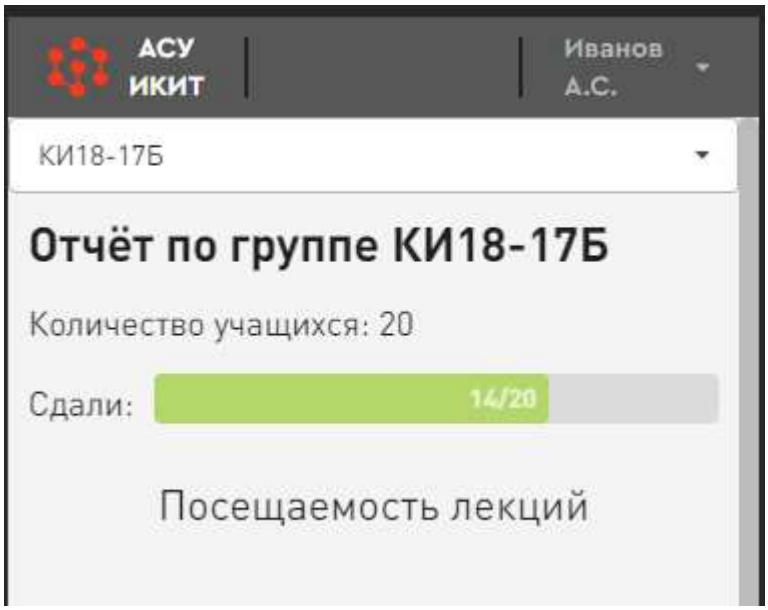

Рисунок 27 – Список с выбором групп в мабильной версии

4.2 Выводы по разделу

Целью данного раздела являлось описание результатов разработки веб-приложения.

Были представлены скриншоты интерфейса веб-приложения и описан функционал каждых элементов в разработанных страницах.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы был проведён анализ существующих решений, были составлены требования, описан выбор стека разработки, реализованы этапы проектирования и разработки клиентской части.

Были изучены прицнипы построения архитектуры веб-приложений. Также были приобретены дополнительные навыки разработки с использованием библиотеки React, в частности React Hooks и тестирования веб-приложений при помощи React Test Library.

Результатом выпускной квалификационной работы стало веб-приложение для просмотра информации о студентах на основе СЭО СФУ.

Разработанное приложение позволяет:

– авторизоваться;

– просматривать учебную информацию ребенка (для родителей);

– просматривать агрегированный отчёт по выбранным группам (для преподавателей);

– просматривать отчёт по каждой выбранной группе отдельно (для преподавателей).

Несмотря на то, что все задачи были выполнены, существуют дальнейшие способы улучшения системы, такие как реализация дополнительных фильтров для преподавателей и добавление модуля для работников деканата.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. SEO против React: Веб-краулеры умнее, чем вы думаете [Электронный ресурс] // Сайт организации freeCodeCamp. Режим доступа: https://www.freecodecamp.org/news/seo-vs-react-is-it-neccessary-to-render-reactpages-in-the-backend-74ce5015c0c9/.

2. Архитектурные особенности проектирования и разработки Веб-приложений [Электронный ресурс] // Сайт НОУ ИНТУИТ. – Режим доступа: https://intuit.ru/studies/courses/611/467/lecture/28784?page=3.

3. Бэнкс А., Порселло Е. React и Redux: функциональная веб-разработка. – СПб.: Питер, 2018. – 336 с.

4. Введение в хуки [Электронный ресурс] // Документация React. – Режим доступа: https://reactjs.org/docs/hooks-intro.html.

5. Веб-рендер [Электронный ресурс] // Портал веб-разработки Google. – Режим доступа: https://developers.google.com/web/updates/2019/02/rendering-on-theweb.

6. Диаграммы вариантов использования [Электронный ресурс] // Сайт компании Informicus. – Режим доступа: http://www.informicus.ru/default.aspx?SECTION=6&id=73&subdivisionid=4.

7. Документация Axios [Электронный ресурс] // Сайт библиотеки Axios. – Режим доступа: https://axios-http.com/docs/intro.

8. Документация Nivo Rocks [Электронный ресурс] // Сайт библиотеки Nivo. – Режим доступа: https://nivo.rocks/.

9. Документация Semantic UI React [Электронный ресурс] // Сайт библиотеки Semantic UI React. – Режим доступа: https://react.semantic-ui.com/.

10. Жизненный цикл разработки ПО. [Электронный ресурс] // Сайт компании XB Software. – Режим доступа: https://xbsoftware.ru/blog/zhiznennyj-tsykl-po-kanban/.

11. Лопатин, А. С. Языки программирования: учебное пособие / А. С. Лопатин, Л. Ю. Искакова. – Екатеринбург, 1998. – 548 с.

12. Майк К., Марк Х, Натан Р. Node.js в действии. – СПб.: Питер, 202. – 448с.

13. Многоуровневая архитектура [Электронный ресурс] // Сайт сибирского отделения Российской академии наук. – Режим доступа: http://wwwsbras.nsc.ru/Report2006/Report321/node30.html.

14. Мобильный и десктопный трафик в 2019 году [Электронный ресурс] // Сайт digital-агентства Stone Temple. – Режим доступа: https://www.stonetemple.com/mobile-vs-desktop-usage-study/.

15. Общие архитектуры веб-приложений [Электронный ресурс] // Официальный сайт Microsoft. – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures.

16. Основные концепции React.js, о которых стоит знать [Электронный ресурс] // Библиотека программиста. – Режим доступа: https://proglib.io/p/react-jsconcepts/.

17. Проектирование веб-интерфейсов [Электронный ресурс] // Сайт студии Jazz Pixels. – Режим доступа: https://jazzpixels.ru/blog/8/proektirovanieinterfeisov.

18. Результаты опроса разработчиков в 2019 году [Электронный ресурс] // Система вопросов и ответов о программировании Stack Overflow. – Режим доступа: https://insights.stackoverflow.com/survey/2019.

19. Рекомендации по включению в систему для вебмастеров [Электронный ресурс] // Справочник Google Scholar. – Режим доступа: https://scholar.google.com/intl/en/scholar/inclusion.html#indexing.

20. Фриман, А. Angular для проффесионалов. / А. Фриман. – СПб.: Питер, 2018. – 800 с

21. Якобсон, А. Унифицированный процесс разработки ПО / А. Якобсон. – СПб.: Питер, 2012. – 496 с.

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А.С. Кузнецов
подпись инициалы, фамилия
«15» 06 2021 г

БАКАЛАВРСКАЯ РАБОТА

09.03.04 Программная инженерия
код и наименование специальности

Разработка Front-end части модуля анализа успеваемости студентов на основе СЭО СФУ
тема

Руководитель ВКР 15.06.21, доцент, канд. техн. наук А. В. Хныкин
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник 15.06.21 А. А. Соколов
подпись, дата инициалы, фамилия

Красноярск 2021