МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-40 05 01 «Информационные системы и технологии»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА**

по дисциплине «Скриптовые языки программирования»

Тема: Веб-сайт «Веб-сайт факультета ИТ»

**Исполнитель**

студент 2 курса 1 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_ М. Н. Ковальчук

подпись, дата

**Руководитель**

доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н. А. Жиляк

должность, учен. степень, ученое звание подпись, дата

Допущен(а) к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата, подпись

Курсовой проект защищен с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_ Н. А. Жиляк

подпись дата инициалы и фамилия

Оглавление

[Введение 3](#_Toc185176624)

[1 Постановка задачи 4](#_Toc185176625)

[1.1 Обзор аналогичных решений 4](#_Toc185176626)

[1.2 Техническое задание 5](#_Toc185176627)

[1.3 Выбор средств реализации программного средства 6](#_Toc185176628)

[1.4 Вывод 7](#_Toc185176629)

[2 Проектирование веб-сайта 9](#_Toc185176630)

[2.1 Выбор способа вёрстки 9](#_Toc185176631)

[2.2 Выбор стилевого оформления 9](#_Toc185176632)

[2.3 Выбор шрифтового оформления 10](#_Toc185176633)

[2.4 Разработка логотипа 10](#_Toc185176634)

[2.5 Разработка пользовательских элементов 11](#_Toc185176635)

[2.6 Разработка спецэффектов 12](#_Toc185176636)

[2.7 Вывод 13](#_Toc185176637)

[3 Реализация структуры веб-сайта 14](#_Toc185176638)

[3.1 Структура React приложения 14](#_Toc185176639)

[3.2 Добавление таблиц стилей SCSS и CSS 15](#_Toc185176640)

[3.3 Программная реализация проекта 15](#_Toc185176641)

[3.3.1 Использование Redux и Redux Toolkit 15](#_Toc185176642)

[3.3.2 Использование различных хуков 17](#_Toc185176643)

[3.3.3 Использование внешних библиотек 18](#_Toc185176644)

[3.4 Вывод 19](#_Toc185176645)

[4 Описание хода работы веб-сайта 20](#_Toc185176646)

[4.1 Руководство пользователя 20](#_Toc185176647)

[4.2 Тестирование веб-сайта 22](#_Toc185176648)

[4.3 Вывод 23](#_Toc185176649)

[Заключение 24](#_Toc185176650)

[Список использованных источников 25](#_Toc185176651)

[5 Приложения 26](#_Toc185176652)

Введение

В наше время веб-приложения используются в различных сферах из-за их способности предоставлять пользователям нужный им контент в зависимости от их запросов. Веб-приложения – это такие веб-страницы, которые могут менять содержания динамически, и, чаще всего, по принципу одностраничности.

Сегодня особенной популярностью пользуется создание веб-приложений для ресурсов различных учебных заведений, с целью предоставить студентам и преподавателям легко и удобно получать доступ к образовательной информации, новостной информации об учебном заведении, множестве организационных мероприятий. Веб-приложения такого рода должны быть доступны везде и на любых устройствах – от смартфонов до компьютеров. Это ставит акцент на кроссбраузерности и кроссплатформенности при разработке таких приложений.

С учетом потребности учебных заведений в удобной и функциональной платформе, цель этого курсового проекта – создание современного веб-приложения, способного стабильно работать на разных устройствах и браузерах. В процессе разработки планируется использовать такие технологии как React и Redux Toolkit, для создания удобного интерфейса с принципом одностраничности.

Основные задачи проекта:

* Исследование существующих решений в сфере веб-приложений
* Определение ключевых характеристик разрабатываемого приложения
* Выбор подходящих технологий и инструментов
* Проектирование интерфейсов и разработка компонентов
* Создание пользовательских элементов интерфейса
* Разработка логотипа
* Разработка стилей веб-приложения
* Интеграция Redux Toolkit
* Подготовка руководства пользователя
* Подведение итого разработки.

Данный проект ориентирован на создание удобного, эстетичного и функционального решения для факультета информационных технологий.

# Постановка задачи

## Обзор аналогичных решений

В интернете существует достаточно много решений веб-приложений для факультетов в разных учебных заведениях. Были исследованы региональные решения подобных веб-приложений. В результате исследования выяснилось, что в основном такие веб-приложения несут информационный характер, в каждом из них встроен блок с новостями, и различного рода слайдеры, содержащие информацию либо о факультете, либо об учебном заведении. Один из примеров сайтов факультета может послужить сайт факультета ИТ БГТУ.

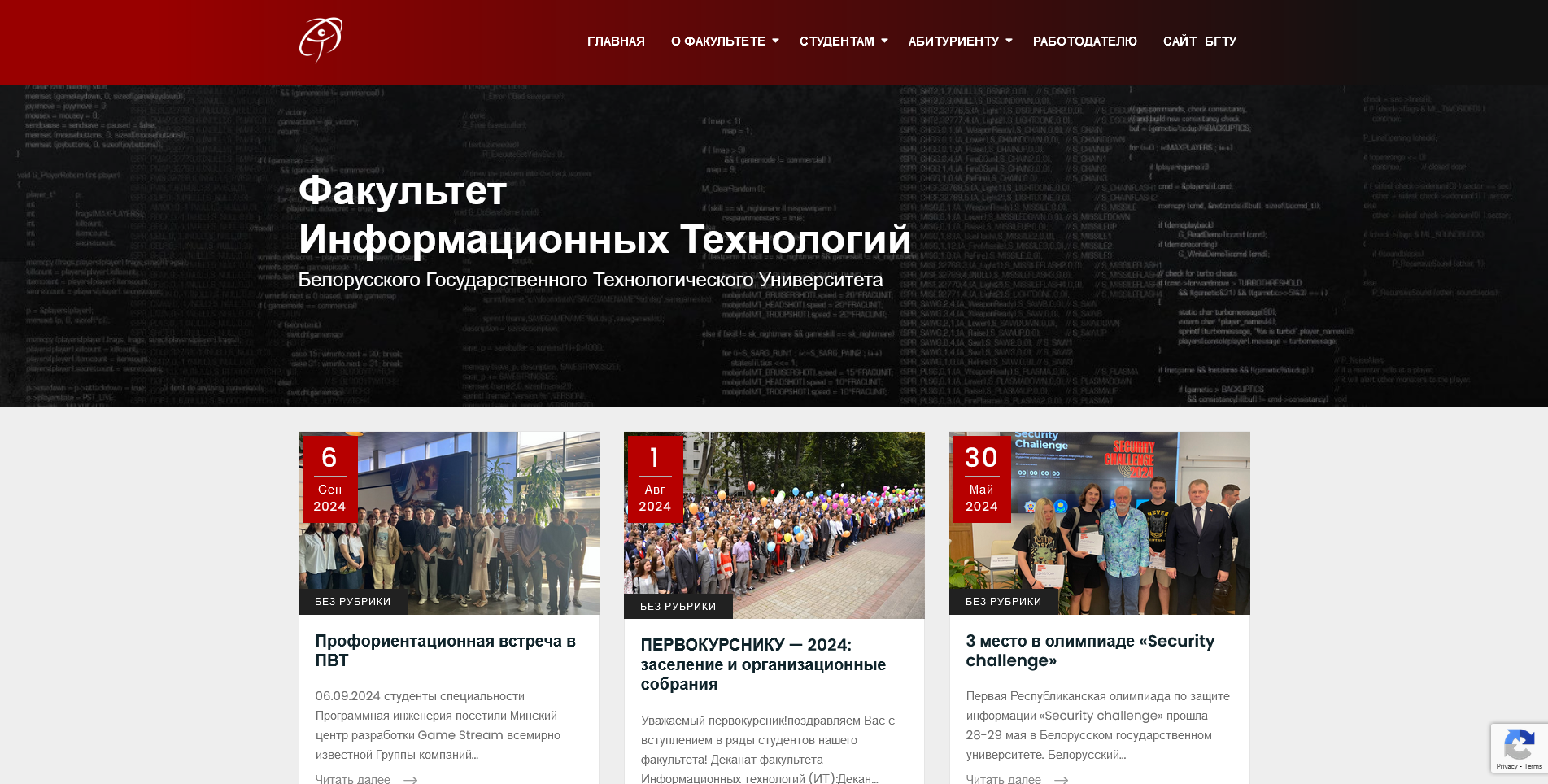


Рисунок 1.1 – сайта it.belstu.by [1]

На главной странице этого сайта можно обнаружить новостной блок, который позволяет быстро и беспрепятственно получить информацию о новостях факультета. Такое решение будет полезно, в связи с тем, что главная страница веб-приложения сразу будет выполнять свою задачу – предоставлять информацию.

Помимо сайта факультета ИТ БГТУ можно рассмотреть решения других учебных заведений. На рисунке 1.2 изображен сайт ВУЗа БНТУ.



Рисунок 1.2 – Сайт БНТУ [2]

На главной странице сайта расположен интерактивный слайдер с информацией по различным темам веб-приложения. Такое решение может быть полезно для будущего проекта, сделав его более красочнее и интекрактивней.

## Техническое задание

Для реализации проекта ключевыми критериями являются создание адаптивного и современного дизайна, улучшение структуры навигации и функционала веб-сайта, а также обеспечение его доступности для пользователей на различных устройствах. Сайт будет разработан с учётом современных стандартов UX/UI, чтобы повысить удобство взаимодействия, а также улучшить визуальное восприятие и производительность ресурса.

Основные особенности проекта:

1. Адаптивность и доступность:

Сайт будет корректно отображаться на экранах различных размеров, включая мобильные устройства, планшеты и настольные компьютеры. Интерфейс будет соответствовать современным требованиям доступности, включая использование высококонтрастных цветов, адаптивных шрифтов и чёткой структуры.

1. Интерфейс и дизайн:

Визуальный стиль будет обновлён, включая улучшение цветовой палитры, шрифтов и общего макета страниц. Упрощённая структура элементов позволит пользователям легко находить нужную информацию.

1. Функциональность:

Интерактивные элементы, такие как динамические меню, фильтры и кнопки быстрого доступа, повысят удобство использования. Реализуются формы обратной связи и возможность загрузки расписания или документов.

1. Навигация:

Оптимизированная структура меню с учётом потребностей разных категорий пользователей: студентов, абитуриентов и преподавателей. Упрощённая навигация между разделами с минимизацией количества кликов для поиска информации.

1. Производительность:

Оптимизация загрузки контента, включая изображения, скрипты и стили. Lazy loading для медиафайлов обеспечит быстрое открытие страниц даже при слабом подключении к интернету.

Для реализации проекта будут использоваться следующие технологии:

* **React**: для построения компонентного интерфейса и обеспечения быстродействия сайта;
* **Redux Toolkit**: для централизованного управления состоянием приложения и поддержки асинхронных операций;
* **TypeScript**: для минимизации ошибок и повышения надёжности кода за счёт статической типизации;
* **CSS Modules**: для структурированной и изолированной стилизации элементов;
* **Jest** и **React Testing Library**: для тестирования компонентов и проверки их корректной работы.

## Выбор средств реализации программного средства

Для разработки сайта использовался текстовый редактор Visual Studio Code. Сайт поддерживает корректное отображение в популярных браузерах, таких как Opera, Microsoft Edge, Google Chrome, Mozilla Firefox и другие.

**Visual Studio Code** – это бесплатный и многофункциональный текстовый редактор, созданный компанией Microsoft. Его возможности позволяют эффективно управлять проектом и легко редактировать компоненты. Редактор поддерживает установку расширений, обеспечивающих удобную работу с React, TypeScript, JSX, CSS, и JSON, а также инструменты для тестирования.

**React** – популярная библиотека для создания пользовательских интерфейсов. Она применяется для разработки интерфейсов на основе компонентов, которые можно многократно использовать в разных частях приложения. Ключевые особенности React включают:

* **Virtual DOM** – виртуальное представление DOM, которое позволяет обновлять только изменённые элементы интерфейса, повышая производительность приложения;
* **TSX** – синтаксис, совмещающий HTML и TypeScript, что упрощает написание и поддержку кода;
* **React Router** – инструмент для организации маршрутизации в одностраничных приложениях, обеспечивающий навигацию между страницами;
* **React Hooks** – функции для управления состоянием и жизненным циклом компонентов, которые заменяют классовую модель функциональными компонентами.

**Redux Toolkit** [6]– библиотека для централизованного управления состоянием приложения. Она обеспечивает упрощённую работу с состоянием и асинхронными запросами благодаря таким компонентам:

* **Store** – единое хранилище данных приложения;
* **Actions** – инструкции для изменения состояния;
* **Reducers** – функции, описывающие изменения состояния в ответ на действия;
* **Middleware** – промежуточный слой, который обрабатывает побочные эффекты, такие как асинхронные запросы (реализуются через redux-thunk).

**TypeScript** [3] – язык программирования, расширяющий JavaScript статической типизацией. Это позволяет минимизировать ошибки, улучшить читаемость кода и его поддержку в долгосрочной перспективе.

**CSS Modules** [4]– инструменты для создания структурированных и изолированных стилей. Эти технологии обеспечивают переиспользование кода и модульность, что особенно важно при разработке крупных проектов.

**Jest** и **React Testing Library** используются для тестирования компонентов и функциональности сайта. Эти инструменты позволяют выявлять и устранять ошибки на ранних этапах разработки, что повышает качество итогового продукта.

**JSON (JavaScript Object Notation)** – лёгкий формат обмена данными. Он активно используется для передачи данных между сервером и клиентом, предоставляя простой и понятный способ работы с информацией.

Благодаря использованию современных технологий и инструментов, разработка сайта становится более эффективной и соответствует актуальным стандартам веб-разработки.

## Вывод

На основании проведенного анализа существующего сайта факультета информационных технологий Белорусского государственного технологического университета и составленного технического задания для редизайна можно сделать следующие выводы:

* **Использование React** позволяет создавать интерактивные, динамичные и легко модифицируемые интерфейсы, обеспечивающие удобство пользователей. За счёт переиспользуемых компонентов достигается структурированность и унификация дизайна.
* **React Hooks** упрощают управление состоянием и логикой компонентов, что делает код более читаемым и поддерживаемым. Интеграция с Redux Toolkit обеспечивает централизованное управление данными и удобное решение для работы с асинхронными запросами, такими как загрузка данных для отображения на страницах сайта.
* **CSS Modules**  обеспечивают модульную стилизацию, позволяя создавать изолированные стили для каждого компонента. Это упрощает процесс масштабирования проекта, поддержания единого визуального стиля и внесения изменений.
* **React Router** предоставляет удобный способ маршрутизации, что позволяет реализовать плавную и интуитивную навигацию между страницами сайта. Это особенно важно для создания положительного пользовательского опыта.
* **JSON** используется для передачи данных между клиентом и сервером. Его легковесность и простота делают этот формат идеальным выбором для взаимодействия с сервером и обработки данных.
* **Адаптивная вёрстка** необходима для корректного отображения сайта на устройствах с разными экранами. Это обеспечивает удобство работы пользователей независимо от их устройства.

Таким образом, модернизация сайта с применением современных технологий, таких как React, Redux Toolkit, SCSS, и методов адаптивной вёрстки, позволит повысить его производительность, удобство использования и привлекательность, а также укрепит имидж факультета.

# Проектирование веб-сайта

## Выбор способа вёрстки

На сайте, представленном на фотографии, используются современные методы разработки интерфейсов, основанные на React. Каждый элемент страницы реализован как отдельный компонент, что упрощает управление его структурой и внесение необходимых изменений.

Структура сайта имеет древовидную форму: корневой компонент включает в себя дочерние компоненты, каждый из которых может содержать свои подкомпоненты. Это способствует эффективной организации контента и его легкой поддержке.

Для стилизации применяются CSS Modules, что позволяет избежать конфликтов стилей между компонентами. Вёрстка осуществляется с помощью современных технологий, таких как Flexbox. Эти методы позволяют гибко управлять макетом страницы, обеспечивая правильное отображение элементов.

Адаптивность сайта достигается через медиа-запросы и реакт-компоненты, что позволяет учесть особенности устройств, наиболее часто используемых пользователями, таких как компьютеры, ноутбуки, планшеты и смартфоны.

React Hooks играют важную роль в управлении состоянием и взаимодействии с DOM, позволяя динамически изменять отображение элементов в зависимости от размера экрана и действий пользователя, что делает интерфейс более интерактивным и удобным.

## Выбор стилевого оформления

Для сайта была подобрана такая цветовая палитра, при которой процесс посещения сайта для будет комфортным, а вся нужная информация будет удобной для чтения.



Рисунок 2.1 – Цветовая палитра сайта

Основная цветовая палитра содержит 3 основных цвета, которые взяты для минималистичного и приятного оформления сайта, чтобы у пользователя не было желания сразу покинуть сайт из-за дискомфорта.

## Выбор шрифтового оформления

**Выбор шрифтов для сайта** был осуществлен с целью повышения удобства просмотра, читабельности информации и обеспечения приятного взаимодействия пользователя с веб-ресурсом. На сайте факультета информационных технологий Белорусского государственного технологического университета важно, чтобы информация для абитуриентов, студентов и преподавателей была четкой и легко воспринимаемой. Для достижения этого подойдут такие шрифты, как **«Bebas Neue»** и **«Calvin Book»**, обладающие уникальными характеристиками.



Рисунок 2.2 – Шрифт **Bebas Neue**

Шрифт «Bebas Neue» характеризуется четкими и выразительными формами. Он идеально подходит для заголовков и ключевых акцентов, привлекая внимание благодаря своей монументальной эстетике.

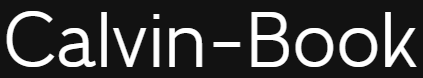


Рисунок 2.3 – Шрифт Calvin Book

«Calvin Book» отличается изысканностью и легкостью восприятия. Его тонкие линии и сбалансированные пропорции обеспечивают читабельность в длинных текстах, создавая ощущение профессионализма и современности.

Размер текста на сайте варьируется от 1 до 1.8 rem, в зависимости от контекста использования: основного текста, подписей или заголовков. Такой подход гарантирует адаптивность интерфейса под размеры различных устройств, благодаря чему пользователь получает удобное отображение информации на любых экранах.

## Разработка логотипа

**Фигура человека** изображена в движении, что символизирует развитие, прогресс и стремление вперёд — ключевые принципы современного IT-сектора. Окружающий элемент в виде дуги напоминает орбиту, что подчёркивает глобальный характер информационных технологий, их охват и влияние на все сферы жизни.

Такой дизайн олицетворяет **инновации, надёжность и стремление к совершенству** в сфере информационных технологий. Логотип создаёт впечатление современного и высокотехнологичного проекта, способного успешно развиваться в цифровой среде.



Рисунок 2.5 – Логотип веб-сайта

На изображении представлен логотип проекта, который отражает динамичный и инновационный характер информационных технологий. Логотип состоит из абстрактного изображения фигуры человека, выполненного в минималистичном стиле. Основные цвета логотипа – красный и серый, что символизирует активность, энергию и профессионализм.

## Разработка пользовательских элементов

Все пользовательские элементы и макеты были разработаны через онлайн-сервис Figma.

Изображение выглядит как Шрифт, текст, снимок экрана, Графика

Автоматически созданное описание

Рисунок 2.6 – Меню сайта

Элемент «Меню» представляет собой кнопку, которая открывает навигационное меню сайта или приложения. Обычно используется для доступа к различным разделам и функциям, обеспечивая пользователю удобный способ перемещения по интерфейсу.

Изображение выглядит как Шрифт, Графика, логотип, текст

Автоматически созданное описание

Рисунок 2.7 – Поиск по сайту

Элемент "Поиск" – это иконка, которая позволяет пользователю осуществлять поиск информации на сайте или в приложении. При нажатии на эту кнопку открывается поле для ввода запроса, что делает поиск доступным и простым.

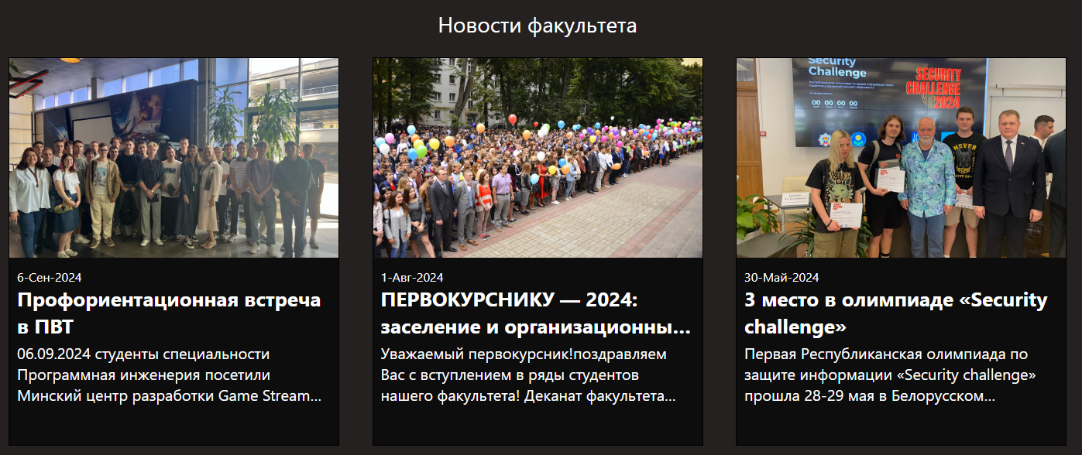


Рисунок 2.8 – Блок «Новости факультета»

Блок «Новости факультета» (рисунок 2.8) представляет собой визуальный раздел, где отображаются актуальные события и мероприятия, связанные с факультетом. Каждая карточка включает в себя заголовок, дату и изображение с кратким описанием.

## Разработка спецэффектов

Анимации и переходы придают веб-сайту и его элементам динамичности. Пользователь видит, что перед ним не простой статический “многостраничник”, а проект, над которым старались и в который вложили усилия.

Все анимации были реализованы через CSS-Modules. Само их количество не велико. Сделано это для того, чтобы быстродействие сайта оставалось на уровне и сайт не казался чем-то непонятным.

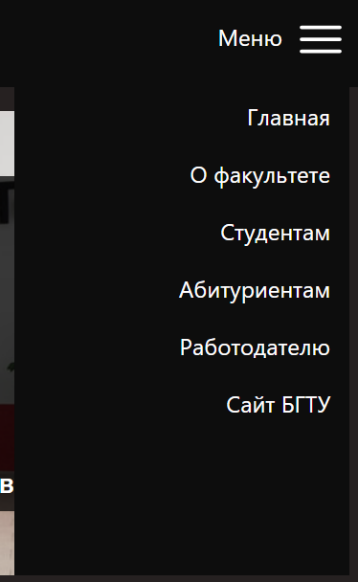


Рисунок 2.10 – Появление бокового меню

Рисунок 2.10 изображает появление бокового меню при нажатии на блок «Меню». После нажатия меню «выезжает справа налево» с плавным эффектом. В самом меню находятся ссылки на второстепенные страницы веб-сайта.

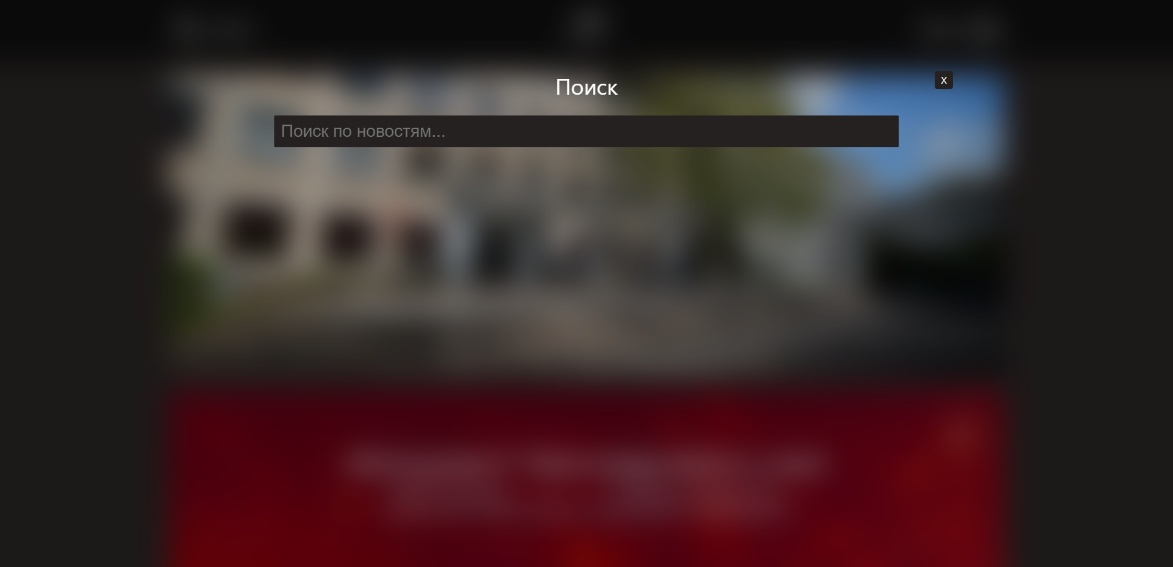


Рисунок 2.11 – Эффект при нажатии на блок «Поиск»

На рисунке 2.11 представлен эффект, который возникает при нажатии на блок «Поиск». После нажатия появляется эффект размытия заднего фона, а также появление формы ввода строки для поиска информации.

## Вывод

В ходе проектирования веб-сайта были приняты ряд важных решений, которые оказывают существенное влияние на его внешний вид и функциональность. Был выбран оптимальный способ вёрстки, который обеспечивает гибкость и адаптивность сайта к различным устройствам и экранам. Это позволяет пользователям комфортно пользоваться ресурсом с любого устройства.

Также было уделено внимание выбору стилевого оформления и шрифтового оформления. Эти аспекты оказывают значительное влияние на восприятие пользователем веб-сайта и его эстетическую привлекательность. Были выбраны соответствующие стили и шрифты, которые гармонично сочетаются с общим дизайном и создают приятную визуальную атмосферу.

Важным этапом проектирования было разработка логотипа и пользовательских элементов. Логотип является важным элементом идентификации веб-сайта, а пользовательские элементы обеспечивают удобство использования и навигации по сайту. Были созданы привлекательные и интуитивно понятные элементы, которые облегчают взаимодействие пользователя с веб-сайтом.

Эти решения в проектировании веб-сайта позволяют создать привлекательный и современный интерфейс, который привлечет и удержит пользователей. Они способствуют повышению удобства использования и обеспечивают позитивный пользовательский сеанс посещения веб-сайта.

# Реализация структуры веб-сайта

## Структура React приложения

Всё React приложение разбито на несколько частей, с которыми можно ознакомиться ниже:

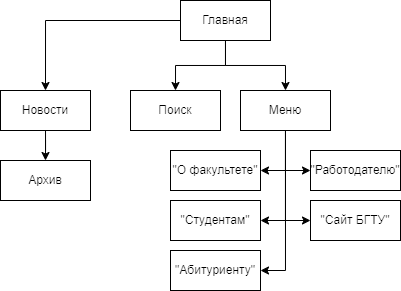


Рисунок 3.1 – Карта веб-сайта факультета информационных технологий

Каждая страница веб-сайта делится на шапку, главную часть и подвал для размещения соответственного контента или информации.



Рисунок 3.2 – Шапка веб-сайта факультета информационных технологий

В шапке расположены ключевые элементы навигации веб-сайта: логотип, поле для поиска и кнопка меню. Логотип служит для возврата на главную страницу, обеспечивая пользователю удобный доступ к основным разделам сайта. Поле поиска позволяет быстро находить необходимую информацию, улучшая пользовательский опыт. Кнопка меню открывает навигационное меню, что также упрощает перемещение по сайту. Все элементы организованы таким образом, чтобы обеспечить легкий доступ к важным функциям в любое время.

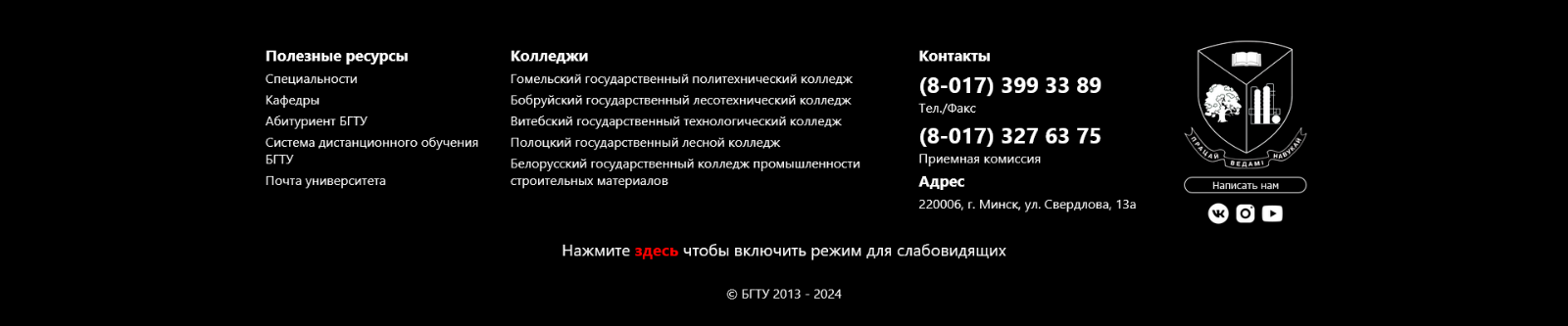


Рисунок 3.3 – Подвал веб-сайта факультета информационных технологий

В подвале размещена информация о временных рамках сайта, а также блок «Версия для слабовидящих».

## Добавление таблиц стилей SCSS и CSS

Все стили веб-сайта были написаны с использованием CSS-Modules, которые обеспечивают удобство работы с кодом за счёт изоляции стилей для каждого компонента. Это позволило организовать стили для каждого компонента сайта в отдельные модули с расширением .module.scss. Такая структура улучшает читаемость кода и упрощает его поддержку.

**CSS Modules** позволяет обеспечить изоляцию стилей для каждого компонента, исключая возможность конфликтов. Такой подход гарантирует, что стили компонентов не влияют друг на друга, и делает приложение более модульным и масштабируемым.

.input.file::file-selector-button {

    border: none;

    background-color: var(--\_red);

    color: white;

    height: 30px;

    width: 110px;

    border-radius: 5px;

    transition: 0.5s ease-in-out;

}

.input.file::file-selector-button:hover {

    background-color: var(--\_darkred);

    cursor: pointer;

}

Листинг 3.1 – CSS-код стилей для компонента input.file

Этот CSS-код стилизует кнопку выбора файла элемента <input type="file"> с помощью псевдоэлемента ::file-selector-button. Кнопке задаются пользовательские стили: убирается граница, устанавливается фоновый цвет (через переменную --\_red), белый цвет текста, размеры (110x30px), скруглённые углы (border-radius: 5px) и плавная анимация при изменении стилей (transition: 0.5s ease-in-out). При наведении (hover) кнопка меняет фон на более тёмный цвет (--\_darkred) и отображает указатель курсора (cursor: pointer), что подчеркивает её интерактивность.

## Программная реализация проекта

### Использование Redux и Redux Toolkit

Для управления состоянием приложения в проекте используется библиотека **Redux**, а также её дополнение **Redux Toolkit**, которое упрощает конфигурацию хранилища и работу с состоянием .

Состояние приложения разделено на несколько независимых срезов, каждый из которых отвечает за определённую часть данных. В данном случае используются срез **slices**, который отвечает за управление данными формы и новостей.

Срезы создаются с помощью функции createSlice из Redux Toolkit, которая позволяет описать начальное состояние, определить редьюсеры. Пример, показывающий использование Redux и Redux ToolKit, представлен в листинге 3.2.

import { configureStore } from "@reduxjs/toolkit";

import rootReducer from "./reducers";

const store = configureStore({

    reducer: rootReducer

});

export default store;

Листинг 3.2 – хранилище данных форм и новостей

В листинге 3.3 показано использование redux для организации среза для корзины сайта, в котором хранятся добавленные из каталога товары.

import { createSlice } from '@reduxjs/toolkit';

import news from '../data/news.json';

export interface newsData {

  image: string;

  date: { day: string; month: string; year: string; };

  title: string;

  content: { big: string; small: string; };

  extra: { urls: string[]; others: string[]; };

  id: number;

}

export interface State {

  form: {

    formInfo: { name: string; email: string; phone: string; theme: string; message: string; };

    news: newsData[];

  };

}

const initialState: State['form'] = {

  formInfo: { name: '', email: '', phone: '', theme: '', message: '',},

  news: news,

};

const formSlice = createSlice({

  name: 'form',

  initialState,

  reducers: {setForm: () => {},},

});

export const { setForm } = formSlice.actions;

export const formReducer = formSlice.reducer;

Листинг 3.3 – код среза для форм и новостей

### Использование различных хуков

Для управления состоянием и выполнения побочных эффектов в проекте активно используются хуки React, такие как **useState**, **useEffect**, **useReducer**, а также кастомные хуки.

Хук **useState** применяется для управления локальным состоянием в функциональных компонентах. Например, этот React-хук useLoad предназначен для отложенной загрузки изображений с использованием Intersection Observer API. Он помогает оптимизировать производительность, загружая изображения только тогда, когда они попадают в область видимости экрана.

export const useLoad = (src: string) => {

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

    const imageRef = useRef<HTMLImageElement>(null);

    useEffect(() => {

        const observer = new IntersectionObserver(([elem]) => {

            if (elem.isIntersecting) {

                setVisible(true);

                observer.disconnect();

            }

        }, {threshold: 0.1})

        if (imageRef.current) {

            observer.observe(imageRef.current);

        }

        return () => {

            if (imageRef.current) {

                observer.unobserve(imageRef.current);

            }

        }

    }, [])

    return {imageRef, isVisible, src: isVisible ? src : ''}

}

Листинг 3.5 – использования хука useLoad в ленивой загрузки изображений

Внутри хука создаётся состояние isVisible с начальным значением false, а также ссылка imageRef с помощью useRef, которая используется для привязки к изображению. Далее, с помощью useEffect создаётся наблюдатель (Intersection Observer), который следит за видимостью элемента на экране.

В конфигурации наблюдателя используется параметр threshold: 0.1, что означает, что колбэк сработает, когда 10% элемента окажется в области видимости. Если изображение становится видимым (isIntersecting равно true), хук переключает состояние isVisible на true и отключает наблюдатель с помощью observer.disconnect() для оптимизации производительности.

В листинге 3.6 useRef используется для получения и хранения ссылки на экземпляр слайдера (Slider из библиотеки react-slick). Эта ссылка позволяет программно управлять поведением слайдера, вызывая его методы напрямую, такие как slickPrev() и slickNext() для переключения слайдов.

const slider = useRef<Slider | null>(null);

<Slider ref={slider} {...settings}>

Листинг 3.6 – использование хука useRef для создания экземпляра слайдера

### Использование внешних библиотек

В данном проекте активно применяются внешние библиотеки для обеспечения функциональности серверной части приложения, включая обработку запросов, отправку писем и работу с OAuth2 .

Основные моменты и их реализация описаны ниже:

* **Создание серверного приложения**: Используется библиотека express и nodemailer [5] для создания HTTP-сервера. Она позволяет обрабатывать запросы от клиента и отправлять ответы, что представлено в листинге 3.7.

const express = require('express');

const app = express();

app.post('/send-email', async (req, res, next) => { ... }); app.listen(PORT, () => { ... });

Листинг 3.7 – использование библиотеки express для создания HTTP-сервера

* **Кросс-доменные запросы**: Для разрешения кросс-доменных запросов применена библиотека cors. Она настраивается таким образом, чтобы принимать запросы только с указанного домена (в данном случае — http://localhost:3000).

app.use(cors({ origin: 'http://localhost:3000' }));

Листинг 3.8 – использование cors для приема запросов с конкретного домена

* **Отправка электронной почты**: Для отправки писем используется библиотека nodemailer. Она настроена на работу через Gmail с использованием OAuth2 для безопасной аутентификации.

const transporter = nodemailer.createTransport({ ... });

await transporter.sendMail(mailOptions);

Листинг 3.9 – использование nodemailer для отправки сообщения

Таким образом, данные внешние библиотеки позволяют обрабатывать тексты о заказе, формировать из них сообщения и отправлять их на почту покупателя, которую он сам указывает. Каждая из библиотек помогает решить конкретные задачи в проекте, например обработка запросов, безопасная авторизация и отправка электронный писем.

## Вывод

В процессе реализации структуры веб-сайта факультета информационных технологий были достигнуты следующие ключевые результаты:

Вся архитектура приложения была организована по компонентному принципу, что облегчает поддержку и расширение функционала. Каждый компонент отвечает за свою часть интерфейса, что способствует модульности и повторному использованию кода.

Применение препроцессора CSS Modules позволяет изолировать стили для каждого компонента, что предотвращает конфликты и упрощает поддержку. Это решение улучшает читаемость и структуру стилей, делая их более управляемыми.

Была успешно внедрена библиотека Redux для управления состоянием приложения. Это решение обеспечивает централизованное хранилище данных, что позволяет легко отслеживать и управлять состоянием, особенно в контексте данных форм и новостей.

Благодаря использованию медиазапросов и реактивных компонентов, сайт имеет адаптивный дизайн, что обеспечивает комфортное взаимодействие пользователей на различных устройствах.

В целом, реализация структуры веб-сайта факультета информационных технологий продемонстрировала эффективность использования современных технологий и подходов к разработке, что способствует созданию удобного и интуитивно понятного интерфейса для пользователей.

# Описание хода работы веб-сайта

## Руководство пользователя

На главной странице веб-сайта при входе видна шапка сайта и начало содержимого главной страницы. В шапке находятся элементы для поиска по сайту и навигации через меню, а также логотип по центру шапки. В начале главной страницы расположен слайдер с фотографиями о факультете, а также баннер с информацией для абитуриентов.

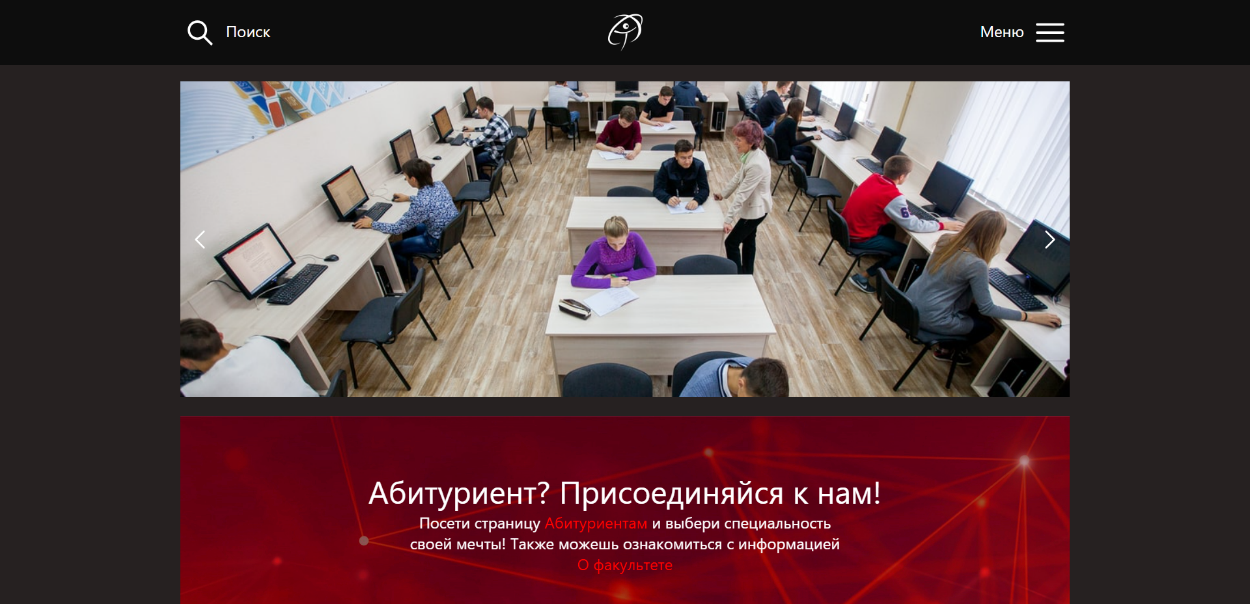


Рисунок 4.1 – Шапка и главная страница

Далее на главной странице находятся новости факультета, расположенные по три колонки. При дальнейшей прокрутке появляется кнопка Архив новостей, при нажатии на которую появляется список всех новостей.

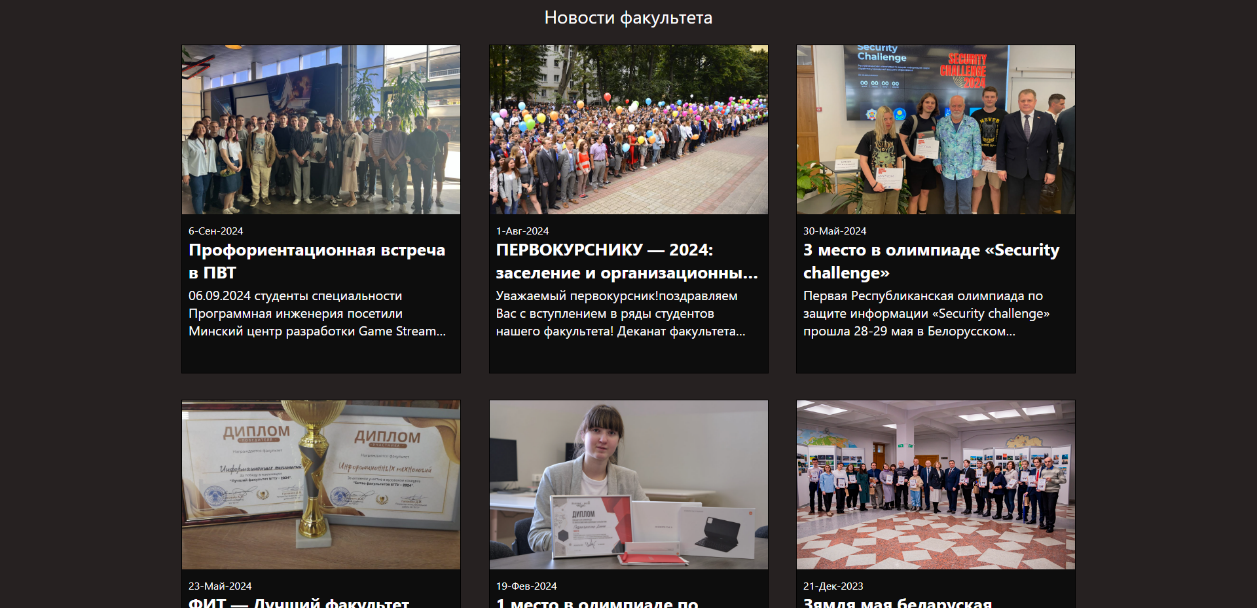


Рисунок 4.2 – Содержимое главной страницы

При нажатии на любую новость открывается соответствующая страницу с полным описание новости. Дизайн и структура зависят от самой новости напрямую.

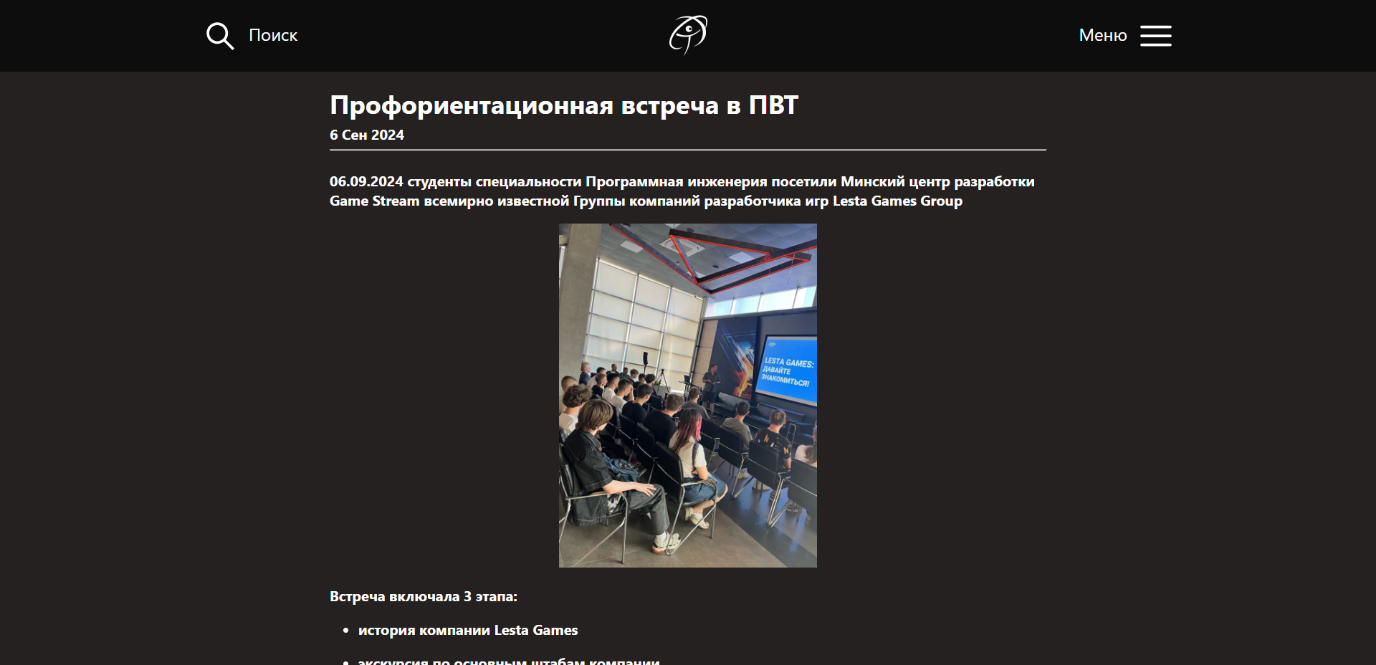


Рисунок 4.3 – Модальное окно с описанием товара

Далее будет представлена диаграмма вариантов использования пользователя, который зашёл на веб-сайт, с демонстрацией всего того, что он может сделать.

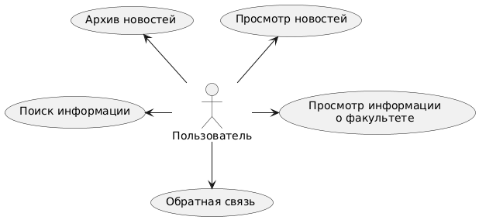


Рисунок 4.4 – Диаграмма вариантов использования Пользователя

На диаграмме вариантов использования, где актором является **Пользователь**, взаимодействующий с различными функциями веб-сайта. Пользователь может: **просматривать новости**, **просматривать архив новостей**, **искать информацию**, **просматривать информацию о факультете**, а также **оставлять обратную связь**. Эти варианты использования представляют основные действия, доступные пользователю на сайте.

## Тестирование веб-сайта

В настоящее время сайтами пользуются не только через компьютеры, но и ноутбуки, планшеты, телефоны и так далее.

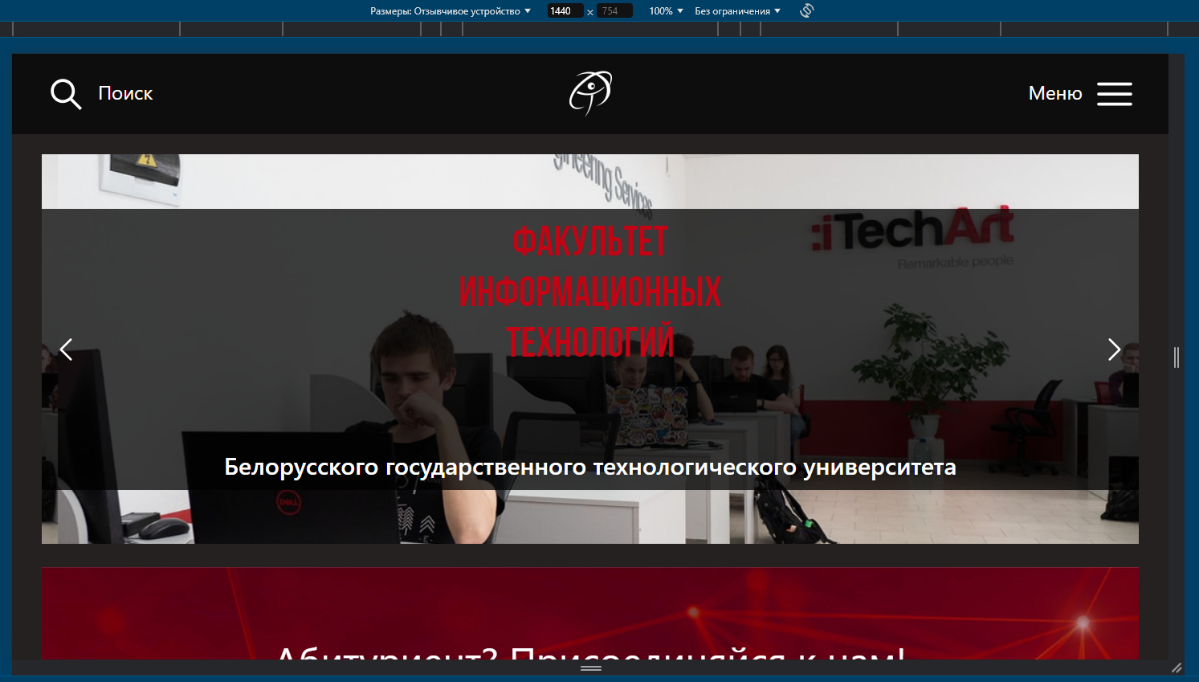


Рисунок 4.5 – Веб-сайт при ширине 1440px

На рисунке 4.5 наглядно показано, что веб-сайт при ширине экрана компьютера отображается нормально, проблем со структурой и дизайном нет.

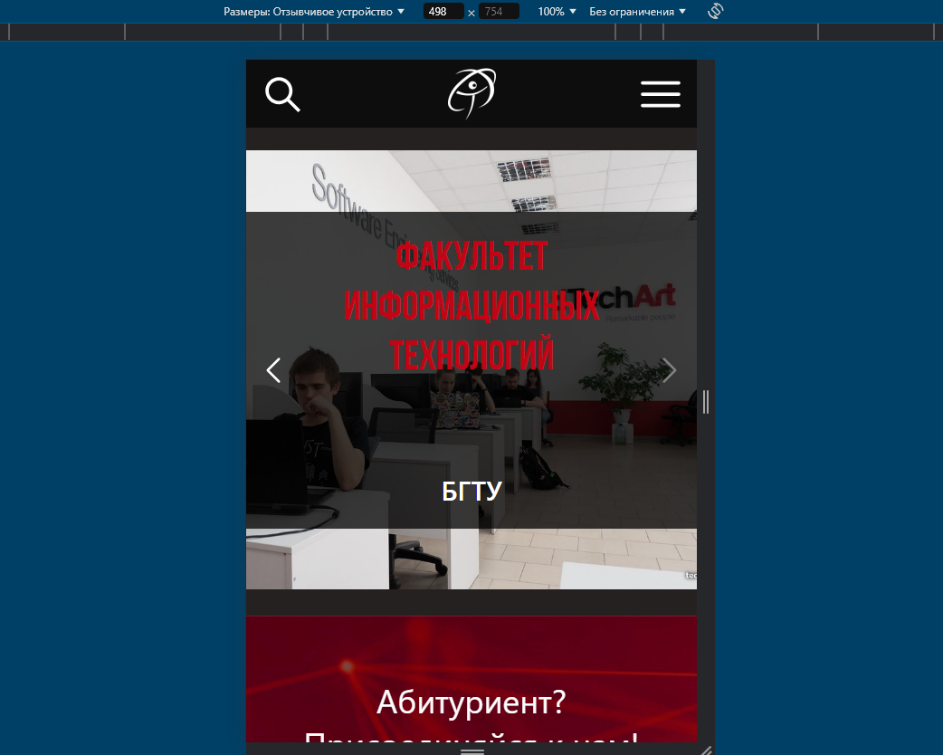


Рисунок 4.6 – Веб-сайт при ширине 500px

На рисунке 4.6 представлен адаптированный веб-сайт под мобильные устройства с маленькой диагональю экран.

Выше представленные скриншоты веб-сайта при разной ширине экрана демонстрируют хорошую адаптивность страниц веб-сайта.

## Вывод

При проведении тестирования веб-сайта была проверена его адаптивность к различным устройствам и экранам. Установлено, что сайт корректно отображается и функционирует на разных разрешениях экранов, включая мобильные устройства, планшеты и компьютеры. Это гарантирует удобство использования сайта независимо от типа устройства, используемого пользователем.

Также в данном разделе было описано подробное руководство пользования веб-приложением, которое позволит использовать сайт в полной мере его возможностей.

# Заключение

В ходе выполнения курсового проекта был реализован веб-сайт факультета информационных технологий. Основная идея сайта заключается в создании удобной платформы для студентов, что было достигнуто благодаря анализу существующих решений и внедрению современных технологий.

При разработке веб-сайта был проведён анализ аналогичных платформ с целью выявления лучших практик и недостатков, что позволило оптимизировать структуру и функциональность нашего проекта. В результате работы над курсовым проектом были достигнуты следующие цели:

* Создана структура страниц с использованием TSX и React-компонентов, обеспечивающая удобную навигацию;
* Реализована стилизация с помощью CSS Modules, что улучшило визуальное восприятие сайта;
* Написаны скрипты и логика работы сайта на TypeScript с использованием React и React Hooks, что повысило интерактивность;
* Интегрировано управление состоянием через Redux Toolkit;
* Создано краткое и понятное руководство пользователя.

Таким образом, цель по разработке веб-сайта была полностью достигнута, и все требования и задачи курсового проекта были успешно выполнены. Созданный сайт демонстрирует сочетание функциональности и эстетической привлекательности, что делает его удобным для пользователей.

# Список использованных источников

* 1. Сайт it.belstu.by [Электронный ресурс]. Режим доступа: <https://it.belstu.by> – Дата доступа: 05.09.2024
  2. Сайт bntu.by [Электронный ресурс]. Режим доступа: <https://bntu.by> - Дата доступа: 10.09.2024
  3. Документация TypeScript [Электронный ресурс]. Режим доступа: <https://www.typescriptlang.org/docs/> - Дата доступа: 1.10.2024
  4. Документация об CSS Modules [Электронный ресурс]. Режим доступа: <https://habr.com/ru/articles/335244/> - Дата доступа: 12.10.2024
  5. Документация nodemailer [Электронный ресурс]. Режим доступа: <https://www.nodemailer.com/usage/> - Дата доступа: 26.10.2024
  6. Документация Redux Toolkit [Электронный ресурс]. Режим доступа: <https://redux-toolkit.js.org/tutorials/overview> - Дата доступа: 5.11.2024

# Приложения

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

import { Provider } from 'react-redux';

import store from './redux/store';

const root = ReactDOM.createRoot(document.getElementById('root')!);

root.render(

<Provider store={store}>

<React.StrictMode>

<App />

</React.StrictMode>

</Provider>

);

reportWebVitals();

Листинг 1 – index.tsx

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

import { Faculty, Contacts } from './components/faculty/Faculty';

import { Main } from './components/main/Main';

import { Footer } from './components/footer/Footer';

import { Header } from './components/header/Header';

import { Search } from './components/search/Search';

import { useState } from 'react';

import { NewsArchive } from './components/NewsArchive/NewsArchive';

import './App.css'

import { NewsPage } from './components/NewsPage/NewsPage';

function App() {

const [search, setSearch] = useState<boolean>(false);

return (

<div className="App">

<Router>

<Search button={setSearch} search={search}/>

<Header button={setSearch} search={search}/>

<div className="content">

<Routes>

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

<Route path='/archive' element={<NewsArchive/>}/>

<Route path='/article' element={<NewsPage/>}/>

<Route path="/faculty" element={<Faculty />}>

<Route path="about" element={<div>О факультете</div>} />

<Route path="contacts" element={<Contacts />} />

<Route path="eduwork" element={<div></div>} />

<Route path="scolarship" element={<div>Стипендии</div>} />

<Route path="dormitory" element={<div>Общежитие</div>} />

</Route>

</Routes>

</div>

<Footer />

</Router>

</div>

);

}

export default App;

Листинг 2 – Main.tsx

import {useState, useRef, useEffect} from 'react'

export const useLoad = (src: string) => {

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

const imageRef = useRef<HTMLImageElement>(null);

useEffect(() => {

const observer = new IntersectionObserver(([elem]) => {

if (elem.isIntersecting) {

setVisible(true);

observer.disconnect();

}

}, {threshold: 0.1})

if (imageRef.current) {

observer.observe(imageRef.current);

}

return () => {

if (imageRef.current) {

observer.unobserve(imageRef.current);

}

}

}, [])

return {imageRef, isVisible, src: isVisible ? src : ''}

}

Листинг 3 – useLoad.tsx

import styles from "./main.module.css"

import bg\_abiturient from './bg\_abiturient.jpg'

import { MySlider } from "./Slider"

import { useSelector } from "react-redux"

import { State, newsData } from "../../redux/slices"

import { useEffect, useState } from "react"

import { useNavigate } from "react-router"

import { Link } from "react-router"

export const Main = () => {

const news = useSelector((state: State)=> state.form.news);

const [pages, setPage] = useState<number>(6);

const navigate = useNavigate();

useEffect(() => {

const handleResize = () => {

if (window.innerWidth < 600) {

setPage(3);

} else {

setPage(6);

}

};

handleResize();

window.addEventListener("resize", handleResize);

}, [])

return (

<div className={styles.main}>

<MySlider/>

<div className={styles.abiturient}>

<p>Абитуриент? Присоединяйся к нам!</p>

<span>Посети страницу <a>Абитуриентам</a> и выбери специальность своей мечты! Также можешь ознакомиться с информацией <a>О факультете</a></span>

</div>

<div className={styles.newsblock}>

<p className={styles.newshead}>Новости факультета</p>

<div className={styles.news}>

{news.slice(0, pages).map((elem) => {

return(

<div className={styles.block} onClick={()=>{navigate(`/article?id=${elem.id}`)}}>

<img className={styles.img} src={elem.image}/>

<div className={styles.text}>

<span>

{elem.date.day}-

{elem.date.month}-

{elem.date.year}

</span>

<Link to={`/article?id=${elem.id}`} title={elem.title} className={styles.head}>

{elem.title}

</Link>

<span className={styles.desc}>

{elem.content.small}

</span>

</div>

</div>

)

})}

</div>

<button className={styles.archive} onClick={() => {navigate('/archive')}}>Архив новостей</button>

</div>

</div>

)

}

Листинг 4 – Main.tsx

import React from "react";

import styles from "./header.module.css";

import Logo from "./logo.png";

import Search from "./search.svg";

import Menu from "./menu.svg";

import { useState } from "react";

import { Link, useNavigate } from "react-router";

interface ButtonHandler {

button: React.Dispatch<React.SetStateAction<boolean>>;

search: boolean;

}

interface SliderOption {

name: string,

link: string

}

export const Header: React.FC<ButtonHandler> = ({button, search}) => {

var navigate = useNavigate();

const [slider, setSlider] = useState<boolean>(false);

const sliderOptions: SliderOption[] = [

{name: "Главная", link: '/'},

{name: "О факультете", link: "/faculty/contacts"},

{name: "Студентам", link: "/student"},

{name: "Абитуриентам", link: "/abiturient"},

{name: "Работодателю", link: "/employer"},

];

return (

<div className={styles.header}>

<div className={styles.menu}>

<div className={styles.button} onClick={() => {

button(!search)

}}>

<img className={styles.option} src={Search} alt="Search Icon" />

<span>Поиск</span>

</div>

<img onClick={() => {navigate('/')}} className={styles.logo} src={Logo} alt="Logo" />

<div className={styles.button} onClick={() => setSlider(!slider)}>

<span>Меню</span>

<img className={styles.option} src={Menu} alt="Menu Icon" />

</div>

<div className={`${styles.slider} ${slider ? styles.show : styles.hide}`}>

{sliderOptions.map(elem => (

<div

className={`${styles.sliderOption} ${slider ? styles.show : styles.hide}`}

>

<Link to={elem.link} onClick={() => {setSlider(!slider)}}><div>{elem.name}</div></Link>

</div>

)

)}

<div

className={`${styles.sliderOption} ${slider ? styles.show : styles.hide}`}

>

<a href="https://belstu.by" ><div>Сайт БГТУ</div></a>

</div>

</div>

</div>

</div>

);

};

Листинг 5 – Header.tsx

import styles from "./footer.module.css"

import vk from "./vk.png"

import insta from "./insta.png"

import yt from "./yt.png"

import { useNavigate } from "react-router"

export const Footer = () => {

const navigate = useNavigate();

return (

<div className={styles.footer}>

<div className={styles.footerBlock}>

<div className={styles.list}>

<h3>Полезные ресурсы</h3>

<ul className={styles.ulfooter}>

<a href="https://www.belstu.by/universitet/divisions/uchebno-metodicheskoe-upravlenie/perechen-specialnostej.html"><li>Специальности</li></a>

<a href="https://www.belstu.by/faculties.html"><li>Кафедры</li></a>

<a href="https://abiturient.belstu.by/"><li>Абитуриент БГТУ</li></a>

<a href="https://dist.belstu.by/login/index.php"><li>Система дистанционного обучения БГТУ</li></a>

<a href="https://mail.belstu.by/mail/"><li>Почта университета</li></a>

</ul>

</div>

<div className={styles.links}>

<h3>Колледжи</h3>

<ul className={styles.ulfooter}>

<a href="https://gomel.belstu.by/"><li>Гомельский государственный политехнический колледж</li></a>

<a href="http://www.bobruisk.belstu.by/"><li>Бобруйский государственный лесотехнический колледж</li></a>

<a href="https://vitgtk.belstu.by/"><li>Витебский государственный технологический колледж</li></a>

<a href="https://pglk.belstu.by/"><li>Полоцкий государственный лесной колледж</li></a>

<a href="http://bgkpsm.belstu.by/"><li>Белорусский государственный колледж промышленности строительных материалов</li></a>

</ul>

</div>

<div className={styles.contacts}>

<h3>Контакты</h3>

<h2><a href="tel:80173993389">(8-017) 399 33 89</a></h2>

<p>Тел./Факс</p>

<h2><a href="tel:80173276375">(8-017) 327 63 75</a></h2>

<p>Приемная комиссия</p>

<h3>Адрес</h3>

<p>220006, г. Минск, ул. Свердлова, 13а</p>

</div>

<div className={styles.emblem}>

<img className={styles.emblemimg}/>

<button onClick={() => {navigate('/faculty/contacts')}}>Написать нам</button>

<div className={styles.social}>

<a href="#" ><img className={styles.socialbuttons} src={vk}></img></a>

<a href="#" ><img className={styles.socialbuttons} src={insta}></img></a>

<a href="#" ><img className={styles.socialbuttons} src={yt}></img></a>

</div>

</div>

</div>

<div className={styles.blindmode}>Нажмите<span className={styles.blind}> здесь </span>чтобы включить режим для слабовидящих</div>

<div className={styles.copyright}>

© БГТУ 2013 - 2024

</div>

</div>

// <div className={styles.footer}>

// <div className={styles.content}>

// <span className={styles.text}>© БГТУ 2013-2024</span>

// <span className={styles.text}>Версия для слабовидящих</span>

// </div>

// </div>

)

}

Листинг 6 – Footer.jsx

import { useEffect } from "react";

import { useSelector } from "react-redux";

import styles from "./search.module.css"

import {State, newsData} from "../../redux/slices"

import { useState } from "react";

import { useNavigate } from "react-router";

interface ISearch {

button: React.Dispatch<React.SetStateAction<boolean>>;

search: boolean;

}

export const Search: React.FC<ISearch> = ({ search, button }) => {

const news = useSelector((state: State)=> state.form.news);

const navigate = useNavigate();

const [news\_searched, setSearched] = useState<newsData[]>([]);

const handleSearch = (value: string) => {

setSearched(news

.filter(elem =>

elem.title

.toLowerCase()

.includes(

value

.toLowerCase()

)))

if (value.trim().length === 0) {

setSearched([])

}

console.log(news\_searched)

}

useEffect(() => {

if (search) {

document.documentElement.style.overflowY = "hidden";

} else {

document.documentElement.style.overflowY = "";

}

}, [search])

return (

<div className={`${styles.search} ${search ? styles.show : styles.hide}`}>

<div className={styles.window}>

<p className={styles.title}>Поиск</p>

<input onChange={(e) => {handleSearch(e.target.value)}}className={styles.input} placeholder="Поиск по новостям..."></input>

<div className={`${styles.suggestions} ${news\_searched.length === 0 ? styles.hide: null}`}>

{news\_searched.map((elem)=>{

return(

<div className={styles.suggestion}

onClick={()=>{

navigate(`/article?id=${elem.id}`)

button(!search);

}}>

<img className={styles.image} src={elem.image}/>

<div className={styles.desc}>

<h1>{elem.title}</h1>

<p>{elem.content.small}</p>

</div>

</div>

)

})}

</div>

<div

className={styles.close}

onClick={() => {

button(!search);

}}

>

X

</div>

</div>

</div>

);

};

Листинг 7 – Search.tsx

import Slider from "react-slick";

import "slick-carousel/slick/slick.css";

import "slick-carousel/slick/slick-theme.css";

import styles from "./main.module.css";

import "../../fonts/bebasneue.css";

import arrow\_left from "./left.svg";

import arrow\_right from "./right.svg";

import React from "react";

import { useRef } from "react";

import Slider1 from './slider.jpg'

import Slider2 from './slider2.jpg'

import Slider3 from './slider3.jpg'

export const MySlider = () => {

const slider = useRef<Slider | null>(null);

const slides = [

{

id: 1,

content: (

<div className={styles.deco} style={{backgroundImage: `url(${Slider1})`}}>

<div className={styles.line}>

<div className={styles.title}>

<span className={styles.name}>

Факультет

<br />

информационных

<br />

технологий

</span>

<br />

<span className={`${styles.title} ${styles.below}`}>

</span>

</div>

</div>

</div>

),

},

{

id: 2,

content: (

<div className={styles.deco} style={{backgroundImage: `url(${Slider2})`, backgroundPositionX: "0"}}>

</div>

),

},

{

id: 3,

content: (

<div className={styles.deco} style={{backgroundImage: `url(${Slider3})`, backgroundPositionX: "0", backgroundPositionY: "-400px"}}>

</div>

),

},

];

const settings = {

dots: false,

arrows: false,

infinite: true,

speed: 500,

autoplay: true,

autoplaySpeed: 10000,

slidesToShow: 1,

slidesToScroll: 1,

};

return (

<div className={styles.sliderContainer}>

<Slider ref={slider} {...settings}>

{slides.map((slide) => (

<div key={slide.id} className={styles.slide}>

{slide.content}

</div>

))}

</Slider>

<div className={`${styles.arrow} ${styles.left}`} onClick={() => {slider?.current?.slickPrev()}}>

<img src={arrow\_left} alt="Previous" />

</div>

<div className={`${styles.arrow} ${styles.right}`} onClick={() => {slider?.current?.slickNext()}}>

<img src={arrow\_right} alt="Next" />

</div>

</div>

);

};

Листинг 8 – Slider.tsx

import styles from './faculty.module.css'

import nav from '../navigation.module.css'

import React from 'react';

import { useForm, SubmitHandler } from "react-hook-form";

import {useState} from 'react';

import {BrowserRouter as Routes, Route, Outlet, Link} from 'react-router-dom'

declare type FormValues = {

fullName: string;

email: string;

phone: string;

subject: string;

message: string;

file: FileList;

};

interface navOption {

name: string,

link: string

}

const importAll = (requireContext: \_\_WebpackModuleApi.RequireContext): string[] =>

requireContext.keys().map((key) => requireContext(key).default || requireContext(key));

export const Contacts = () => {

const { register, handleSubmit, formState: { errors }, reset } = useForm<FormValues>();

const images = importAll(require.context('./images', false, /\.(png|jpe?g|svg)$/));

const scrollToForm = () => {

document.getElementById("form")?.scrollIntoView({ behavior: "smooth" });

}

const handleEmail: SubmitHandler<FormValues> = async (data) => {

console.log("Отправляем сообщение:", data);

const emailContent = `

<div style="width: 1024px; height: fit-content;">

<div style="height: 80px; background-color: black; display: flex">

<img

src="https://it.belstu.by/wp-content/uploads/2019/05/LogoFIT-1-e1557999163520.png"

style="height: 60px; margin: 10px"/>

<h1 style="color: white; line-height: 55px; margin: 10px">Факультет информационных технологий БГТУ</h1>

</div>

<div style="padding: 10px; font-size: 18px; background-color: #262121" color: white !important>

<p style="color: white">Здравствуйте, ${data.fullName}! Спасибо за ваше электронное обращение. Оно получено, и сейчас находится в обработке. Спасибо за ожидание.</p>

<p style="font-size: 14px; color: white;">С уважением,<br> деканат факультета информационных технологий.</p>

</div>

</div>

`

try {

const response = await fetch('http://localhost:5000/send-email', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({

email: data.email,

subject: "Мы получили ваше обращение",

html: emailContent

}),

});

if (!response.ok) {

throw new Error('Failed to send email');

}

const result = await response.json();

alert(result.message);

} catch (error) {

console.error('Error in handleEmail:', error);

alert('Не удалось отправить письмо');

}

};

const onSubmit: SubmitHandler<FormValues> = (data) => {

console.log(data);

handleEmail(data);

reset();

};

return (

<div className={styles.contacts}>

<div className={styles.persons}>

<div className={styles.person}>

<img src={images[1]}/>

<div className={styles.desc}>

<h1>Шиман Дмитрий Васильевич</h1>

<h2>Декан факультета</h2>

<p>Кандидат технических наук, доцент</p>

<p>ауд. 104 корпус 4</p>

<p>тел. <a href="tel:80173993389">8 (017) 399-3389</a></p>

<p>почта <a href="mailto:it@belstu.by">it@belstu.by</a></p>

<p>Приём: Среда с 13.00 до 16.00</p>

<p>Отправить <a onClick={scrollToForm}>электронное обращение</a></p>

</div>

</div>

<div className={styles.person}>

<img src={images[0]}/>

<div className={styles.desc}>

<h1>Салычиц Ольга Игоревна</h1>

<h2>Зам. декана по идеологической <br/>и воспитательной работе</h2>

<p>Кандидат химических наук </p>

<p>ауд. 104 корпус 4</p>

</div>

</div>

<div className={styles.person + ' ' + styles.blank}>

<img src={images[2]}/>

<div className={styles.desc}>

<h1>Жавненко Анна Александровна</h1>

<h2>Специалист</h2>

<p>Время работы:</p>

<p>Пн-Пт: 8:30 - 16:15, 12:00 - 12:45</p>

<p>Суббота: 8:30 - 13:30</p>

</div>

</div>

<div className={styles.person + ' ' + styles.blank}>

<img src={images[2]}/>

<div className={styles.desc}>

<h1>Юшкевич Екатерина Григорьевна</h1>

<h2>Секретарь</h2>

<p>Время работы:</p>

<p>Пн-Пт: 8:30 - 16:15, 12:00 - 12:45</p>

<p>Суббота: 8:30 - 13:30</p>

</div>

</div>

</div>

<div className={styles.formblock}>

<form className={styles.form} onSubmit={handleSubmit(onSubmit)} id="form">

<label className={styles.label}>Электронное обращение</label>

<input

{...register("fullName", { required: "Введите ваше ФИО" })}

className={styles.input}

placeholder="Ваше ФИО"

/>

{errors.fullName && <p className={styles.error}>{errors.fullName.message}</p>}

<input

{...register("email", {

required: "Введите ваш email",

pattern: {

value: /^\S+@\S+$/i,

message: "Некорректный email"

}

})}

className={styles.input}

placeholder="Ваш email"

/>

{errors.email && <p className={styles.error}>{errors.email.message}</p>}

<input

{...register("phone", {

pattern: {

value: /^\+?\d{10,15}$/,

message: "Некорректный телефон"

}

})}

className={styles.input}

placeholder="Ваш телефон"

/>

<input

{...register("subject")}

className={styles.input}

placeholder="Тема"

/>

<textarea

{...register("message", { required: "Введите сообщение" })}

className={`${styles.input} ${styles.textarea}`}

placeholder="Сообщение"

/>

{errors.message && <p className={styles.error}>{errors.message.message}</p>}

<input

{...register("file")}

className={`${styles.input} ${styles.file}`}

type="file"

/>

<button

type="submit"

className={`${styles.input} ${styles.button}`}>

Отправить

</button>

</form>

</div>

</div>

)

}

export const Faculty = () => {

const [activeButton, setActiveButton] = useState<number>(1);

const navOptions: navOption[] = [

{ name: "О факультете", link: "about" },

{ name: "Контакты", link: "contacts" },

{ name: "Воспитательная работа", link: "eduwork" },

{ name: "Стипендии", link: "scolarship" },

{ name: "Общежитие", link: "dormitory" },

];

return (

<div className={styles.faculty}>

<p className={nav.header}>Факультет</p>

<div className={nav.navigation}>

{navOptions.map((elem, index) => (

<Link

key={index}

to={elem.link}

className={`${nav.navbutton} ${activeButton === index ? nav.show : ''}`}

onClick={() => setActiveButton(index)}

>

{elem.name}

</Link>

))}

</div>

<div className={styles.content}>

<Outlet />

</div>

</div>

);

};

Листинг 9 – Faculty.jsx

import Parser from 'html-react-parser'

import { useSelector } from 'react-redux'

import { State } from '../../redux/slices'

import styles from '../main/main.module.css'

import { useNavigate } from 'react-router'

import { useLoad } from '../useLoad'

import { newsData } from '../../redux/slices'

import { useEffect } from 'react'

import { Link } from 'react-router'

export const NewsArchive = () => {

const news = useSelector((state: State) => state.form.news);

useEffect(() => {

window.scrollTo({ top: 0, behavior: 'smooth' });

})

return (

<div className="main">

<div className={styles.newsblock}>

<p className={styles.newshead}>Архив новостей</p>

<div className={styles.news}>

{news.map((elem) => (

<NewsItem {...elem}/>

))}

</div>

</div>

</div>

);

};

const NewsItem: React.FC<newsData> = (elem) => {

const { imageRef, isVisible, src } = useLoad(elem.image);

const navigate = useNavigate();

return (

<div

className={styles.block}

onClick={() => {

navigate(`/article?id=${elem.id}`);

}}

>

<img

className={styles.img}

src={src}

ref={imageRef}

style={{ opacity: isVisible ? 1 : 0.3 }}

alt={elem.title}

/>

<div className={styles.text}>

<span>

{elem.date.day}-

{elem.date.month}-

{elem.date.year}

</span>

<Link to={`/article?id=${elem.id}`} title={elem.title} className={styles.head}>

{elem.title}

</Link>

<span className={styles.desc}>{elem.content.small}</span>

</div>

</div>

);

};

Листинг 10 – NewsArchive.tsx

import { State, newsData } from "../../redux/slices"

import Parser from 'html-react-parser'

import {useEffect} from 'react'

import { useSelector } from "react-redux";

import { useLocation } from "react-router";

import styles from './newspage.module.css'

import './parsed.css'

export const NewsPage = () => {

const location = useLocation();

const searchParams = new URLSearchParams(location.search);

const news\_list = useSelector((state: State)=> state.form.news);

const news = news\_list.filter((elem: newsData) => elem.id.toString() === searchParams.get('id'));

useEffect(() => {

window.scrollTo({ top: 0, behavior: 'smooth' });

})

return (

<div className={styles.main}>

{news.map((elem)=>{

return (

<div>

<div className={styles.head}>

<h1 className={styles.title}>{elem.title}</h1>

<p>{`${elem.date.day} ${elem.date.month} ${elem.date.year}`}</p>

<hr/>

</div>

{Parser(elem.content.big)}

</div>

)

})}

</div>

)

}

Листинг 11 – NewsPage.tsx

import { combineReducers } from "@reduxjs/toolkit";

import { formReducer } from "./slices";

const rootReducer = combineReducers({

form: formReducer

});

export default rootReducer;

Листинг 12 – reducers.ts

import { createSlice } from '@reduxjs/toolkit';

import news from '../data/news.json';

export interface newsData {

image: string;

date: {

day: string;

month: string;

year: string;

};

title: string;

content: {

big: string;

small: string;

};

extra: {

urls: string[];

others: string[];

};

id: number;

}

export interface State {

form: {

formInfo: {

name: string;

email: string;

phone: string;

theme: string;

message: string;

};

news: newsData[];

};

}

const initialState: State['form'] = {

formInfo: {

name: '',

email: '',

phone: '',

theme: '',

message: '',

},

news: news,

};

const formSlice = createSlice({

name: 'form',

initialState,

reducers: {

setForm: () => {

},

},

});

export const { setForm } = formSlice.actions;

export const formReducer = formSlice.reducer;

Листинг 13 – slices.ts

import { configureStore } from "@reduxjs/toolkit";

import rootReducer from "./reducers";

const store = configureStore({

reducer: rootReducer

});

export default store;

Листинг 14 – store.ts

const express = require('express');

const cors = require('cors');

const nodemailer = require('nodemailer');

const { google } = require('googleapis');

require('dotenv').config();

const app = express();

const PORT = process.env.PORT || 5000;

app.use(cors({

origin: 'http://localhost:3000',

}));

app.use(express.json());

const oAuth2Client = new google.auth.OAuth2(process.env.CLIENT\_ID, process.env.CLIENT\_SECRET, process.env.REDIRECT\_URI);

oAuth2Client.setCredentials({ refresh\_token: process.env.REFRESH\_TOKEN });

async function sendEmail(email, subject, message, html) {

try {

const accessToken = await oAuth2Client.getAccessToken();

const transporter = nodemailer.createTransport({

service: 'gmail',

auth: {

type: 'OAuth2',

user: 'core.such.react@gmail.com',

clientId: process.env.CLIENT\_ID,

clientSecret: process.env.CLIENT\_SECRET,

refreshToken: process.env.REFRESH\_TOKEN,

accessToken: accessToken.token,

},

});

const mailOptions = {

from: 'core.such.react@gmail.com',

to: email,

subject: subject,

text: message,

html: html

};

const result = await transporter.sendMail(mailOptions);

return result;

} catch (error) {

console.error('Ошибка отправки сообщения:', error);

throw new Error('Не удалось отправить сообщение');

}

}

app.post('/send-email', async (req, res) => {

const { email, subject, message, html } = req.body;

try {

await sendEmail(email, subject, message, html);

res.status(200).json({ message: 'Успешно!' });

} catch (error) {

res.status(500).json({ message: 'Произошла ошибка!' });

}

});

app.listen(PORT, () => {

console.log(`Сервер запущен на порте ${PORT}`);

});

Листинг 15 – server.js