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 «Web-сайт интерент-магазин элеткроники»

Исполнитель

студент 2 курса 2 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. Р. Новожилов

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

Содержание

[ВВЕДЕНИЕ 4](#_Toc184850836)

[1. Обзор технических методов и программных средств разработки веб-сайта 5](#_Toc184850837)

[1.1 Обследование объекта и обоснование необходимости создания проекта. 5](#_Toc184850838)

[1.2 Формирование требований к проекту. 5](#_Toc184850839)

[1.3 Назначение и цели создания системы. 6](#_Toc184850840)

[1.4 Требования к системе. 6](#_Toc184850841)

[1.5 Поставновка задачи. 7](#_Toc184850842)

[2. Последовательность разработки содержания и структуры веб-сайта с использованием компонентов 8](#_Toc184850843)

[2.1 Техническое задание. 8](#_Toc184850844)

[2.2 Построение прототипов интерфейсов и макетирование страниц. 8](#_Toc184850845)

[2.3 Разработка дизайна элементов веб-сайта. 8](#_Toc184850846)

[2.4 Разработка логотипа. 9](#_Toc184850847)

[2.5 Составляющие структуры программного продукта. 9](#_Toc184850847)

[2.6 Вывод. 10](#_Toc184850848)

[3. Программная реализация проекта 11](#_Toc184850849)

[3.1 Вёрстка программного продукта. 11](#_Toc184850850)

[3.2 Сторонние компоненты Material UI. 13](#_Toc184850851)

[3.3 Написание форм с помощью React Hook Form. 14](#_Toc184850852)

[3.4 Хранение данных 15](#_Toc184850853)

[3.5 Вывод. 16](#_Toc184850854)

[4. Ключевые функции разработанного программного кода, создание значимых документов 17](#_Toc184850857)

[4.1 Руководство пользователя. 17](#_Toc184850858)

[4.2 Кроссбраузерное тестирование. 17](#_Toc184850859)

[4.3 Тестирование на адаптивность. 18](#_Toc184850860)

[4.4 Вывод. 19](#_Toc184850861)

[5. Заключение. 21](#_Toc184850862)

[6. Список использованной литературы. 22](#_Toc184850863)

[ПРИЛОЖЕНИЕ А 23](#_Toc184850864)

[ПРИЛОЖЕНИЕ Б 34](#_Toc184850865)

[ПРИЛОЖЕНИЕ В 35](#_Toc184850866)

# ВВЕДЕНИЕ

Присутствие бизнеса в интернете является критически важным для его успеха. В современном мире большинство клиентов перед покупкой обращают внимание на наличие веб-сайта, что может стать решающим фактором при выборе компании. Веб-сайт — это не просто онлайн-ресурс, а лицо бренда, которое формирует первое впечатление о компании.

Создание собственного веб-сайта открывает множество возможностей для бизнеса. Во-первых, он позволяет продемонстрировать ассортимент товаров и услуг, что помогает потенциальным клиентам лучше понять, что предлагает компания. Четкое представление о продукции способствует повышению шансов на успешные продажи.

Кроме того, веб-сайт помогает выделить компанию среди конкурентов. Он привлекает целевую аудиторию, позволяя показать уникальные предложения и сильные стороны бизнеса. С его помощью можно анонсировать новые продукты и услуги, а также поддерживать прямую связь с клиентами через контактные данные. Такой доступ увеличивает вероятность обращения к компании и, как следствие, продажи.

Одним из значительных преимуществ веб-сайта является его доступность 24/7. Это позволяет компании оперативно обновлять информацию, публиковать акции и объявления, что особенно важно в условиях быстро меняющегося рынка.

Веб-сайт также предоставляет широкие возможности для интернет-маркетинга. С помощью различных инструментов, таких как уведомления о скидках, целевая реклама и всплывающие окна, можно эффективно привлекать и удерживать клиентов.

Учитывая все эти аспекты, тема курсового проекта становится особенно актуальной.

Цель проекта заключается в разработке веб-сайта для магазина видеокарт с использованием библиотеки React.js.

Задачи, которые необходимо решить в рамках проекта, включают:

1. Изучение дизайна существующих веб-сайтов и инструментов для их создания.
2. Разработка макета и прототипа сайта.
3. Определение структуры веб-сайта.
4. Наполнение сайта информацией, соответствующей его тематике.
5. Создание дизайна и интерактивных элементов с использованием React.js.
6. Проведение тестирования веб-сайта.

Целевой аудиторией проекта станут люди, интересующиеся новинками в области техники, таких как видекарты. Этот проект не только поможет развивать бизнес, но и станет ценным ресурсом для покупателей, ищущих качественную продукцию.

# 1. Обзор технических методов и программных средств разработки веб-сайта

# 1.1 Обследование объекта и обоснование необходимости создания проекта.

В современном мире существует множество способов продвижения продукции в интернете. Некоторые компании создают привлекательные веб-сайты, которые легко запоминаются и привлекают внимание посетителей. Другие сосредотачиваются на качественном наполнении контента, а третьи предлагают товары, за которыми клиенты возвращаются снова и снова благодаря высокому качеству.

Тем не менее, вне зависимости от сильных сторон компании, наличие веб-сайта является обязательным условием для успешной торговли в сети. Когда потенциальные покупатели ищут определенные товары или магазины, они в первую очередь обращаются к интернету. Часто они используют различные критерии, такие как цена или местоположение, чтобы найти подходящий вариант. В этом контексте веб-сайт компании становится ее "лицом", от которого зависит, выберут ли клиенты именно ее продукцию.

На решение о покупке влияют множество факторов: начиная от визуальной привлекательности страниц и объема информации о компании и её товарах, и заканчивая отзывами на сторонних ресурсах, которые помогают оценить уровень обслуживания.

Все это подчеркивает важность веб-сайта как ключевого элемента любого бизнеса. Он может сыграть решающую роль в том, станет ли компания успешной или столкнется с трудностями. Без качественного онлайн-присутствия шансы на успех значительно снижаются.

# 1.2 Формирование требований к проекту.

В данной курсовой работе веб-сайт был разработан с использованием фреймворка React, а стилизация осуществлялась с помощью CSS и CSS-модулей. React предлагает удобные инструменты для работы с динамическим контентом, позволяя обновлять отдельные элементы на странице без необходимости полной перезагрузки. Это достигается благодаря встроенному DOM-дереву, которое отслеживает изменения в компонентах, что снижает нагрузку на устройство пользователя по сравнению с традиционными HTML-страницами.

Одним из ключевых аспектов проекта является лаконичность и простота дизайна. Страницы должны быть свободны от избыточной информации, но при этом содержать все основные элементы, отражающие суть проекта. Важно предоставить только значимую для посетителей информацию, чтобы облегчить поиск необходимых сведений. Простота дизайна также позволяет использовать ограниченную цветовую палитру, что помогает акцентировать внимание на важных частях сайта, способствуя увеличению продаж и достижению других целей.

Читаемость текста играет не менее важную роль. Размер шрифта должен быть оптимальным, чтобы текст легко воспринимался, что уменьшит вероятность возникновения трудностей при чтении и повысит интерес посетителей к продукции. Кроме того, навигация на сайте должна быть интуитивно понятной и легко различимой, чтобы пользователи не терялись среди страниц и быстро находили нужную информацию. Важно также избегать близкого расположения ссылок, чтобы предотвратить случайные переходы на другие страницы.

Адаптивность веб-сайта является еще одним важным аспектом. Он должен корректно отображаться на различных устройствах и экранах, обеспечивая привлекательный внешний вид вне зависимости от используемого устройства. Это создает положительное впечатление и повышает вероятность покупки, делая сайт удобным и эффективным инструментом для бизнеса.

# 1.3 Назначение и цели создания системы.

Назначение данной системы является разработка корректно работающего приложения с использованием фреймворка React, а также использование css для задания стилей.

Целями же является создание своих компонентов, которые будут корректно функционировать в приложении, а также использование готовых компонентов для получения наилучшего результата. Также одной из целей является создание работающего проекта, который содержит больше одной веб-страницы.

# 1.4 Требования к системе.

К основным требованиям к системе относятся корректная работа всего сервиса, его стабильность и отказоустойчивость. Важно, чтобы система не испытывала задержек при выполнении действий пользователями, посещении сайта или использовании его функционала. Кроме того, компоненты системы должны быть легко переносимыми на другие сервисы без изменения их первоначального состояния.

Что касается функциональных требований, система должна обеспечивать правильное выполнение всех операций, а также их простоту для восприятия и чтения. Функции должны быть отказоустойчивыми и организованными по нескольким файлам в зависимости от логики приложения и результата их выполнения. Эти функции должны иметь возможность импортироваться для использования в различных частях приложения. Основные функции могут быть расположены в тех же файлах, что и их вызовы, чтобы не перегружать код избыточным количеством файлов и облегчить понимание логики приложения.

Важные требования к среде разработки включают установку фреймворка React. Для работы системы необходим доступ к интернету и любой современный браузер. Система совместима с любыми операционными системами, имеющими графическую оболочку, включая Windows. В корневом каталоге должны быть установлены модули, такие как React Router, Redux Toolkit, react-redux, MUI, React Hook Form, Firebase и другие необходимые библиотеки для React.js.

# 1.5 Поставновка задачи.

Данный курсовой проект представляет собой приложение, которое написано с использованием фреймворка React, а также css для задания всем элементам на странице стилей и прочих библиотек (о них в разделе 3). Проект является многостраничным и написан по теме «Интернет-магазин видеокарт». В проекте реализованы функции для корректной работы фильтрации, поиска, перехода между страницами и т. д. Кроме всего прочего данный проект будет содержать компоненты, которые можно переносить на другие сервисы с минимальными изменениями в логике их работы.

# 2. Последовательность разработки содержания и структуры веб-сайта с использованием компонентов

# 2.1 Техническое задание.

В рамках данного курсового проекта будет разработан веб-сайт для магазина видеокарт с использованием фреймворка React, а стилизация всех элементов страницы будет осуществляться с помощью SCSS. React станет основой приложения, которое создается с помощью пакетного менеджера npm, используя команду npx create-react-app. Проект включает четыре основные страницы: главная, ассортимент товаров, корзина и страница о компании.

Главной целью веб-сайта является создание многостраничного приложения с использованием встроенных и пользовательских хуков, а также дополнительных компонентов для обеспечения корректной работы всего сервиса. Все страницы должны быть адаптивными и функционировать без сбоев на различных устройствах.

Каждая страница имеет унифицированный верхний и нижний колонтитулы. В верхней части размещены ссылки на все страницы приложения, а внизу находится дополнительная навигация, организованная по категориям товаров. Содержимое страниц соответствует их названиям и логике проекта.

Дизайн и функциональность этого проекта будут адаптивными и кроссбраузерными, обеспечивая корректное отображение в таких браузерах, как Google Chrome, Mozilla Firefox и Microsoft Edge.

# 2.2 Построение прототипов интерфейсов и макетирование страниц.

Для удобства визуализации и упрощения процесса исправления ошибок в проектировании и дизайне был выбран инструмент Figma. Это приложение позволило разработать прототипы всех страниц сайта и наглядно представить, как будет выглядеть конечный продукт. Использование функции соединения компонентов на одной странице с элементами на других страницах предоставляет возможность в режиме просмотра создать прототип полноценного многостраничного сайта.

# 2.3 Разработка дизайна элементов веб-сайта.

Для обеспечения адаптивности сайта была использована верстка на основе grid-элементов. Этот подход позволяет легко организовать элементы в нужном порядке и делает их адаптивными, подстраиваясь под различные размеры экранов.

Стили задаются с помощью CSS и CSS-модулей. Цветовая палитра сайта включает светлые, черные и голубые оттенки. Эта комбинация не является слишком яркой, хорошо сочетается с другими цветами и помогает акцентировать внимание на определенных элементах. Дизайн выглядит лаконично и не создает визуальной перегрузки на страницах веб-сайта.

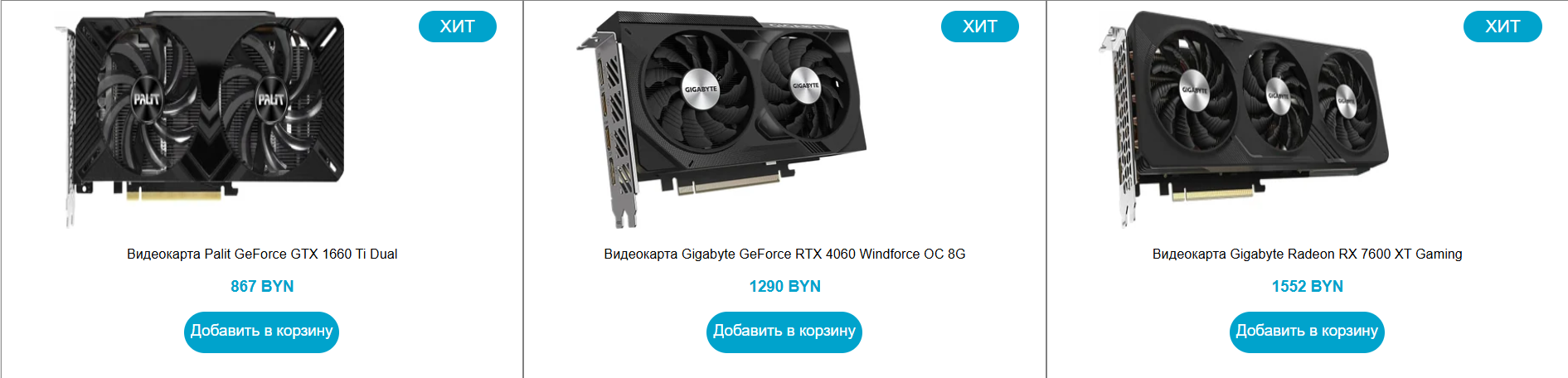


Рисунок 2.1 – Часть веб-страницы с ассортиментом магазина

Также используется большое количество картинок, которые отображают ассортимент магазина. Благодаря и можно увидеть какой товар продаётся, а также это является показателем того, какой товар получит покупатель при заказе.

Также ко всем элементам сайта на всех страницах применяются разные шрифты, в зависимости от типа устройства и браузера на котором сайт запущен, а на больших элементах применен шрифт Times New Roman.

# 2.4 Разработка логотипа.

Логотип представляет собой надпись названия магазина шрифтом RubikGlitch-Regular. Был выбран белый цвет текста и яркий голобуй цвет фона.



Рисунок 2.2 – Логотип магазина.

Логотип был наделен таким необычным шрифтом чтобы выделиться на фоне минималистичных и однотипных логотипов других фирм.

# 2.5 Составляющие структуры программного продукта.

Приложение, созданное с помощью create-react-app, структурировано в несколько папок. Основные файлы расположены в директории src, где находятся файлы всех страниц, используемых в проекте, такие как Catalog.js, Footer.js и App.js.

# 2.6 Вывод.

В данном разделе была определена последовательность разработки веб-сайта. Описан дизайн элементов, создание макетов в виде прототипа проекта. А также составлена структура программного продукта.

# 3. Программная реализация проекта

# 3.1 Вёрстка программного продукта.

Так как весь проект реализовывается с помощью приложения create-react-app, то обычный HTML использовать нельзя. Весь код, который необходимо вывести в виде страницы, написан на специальном языке разметки JSX.

<header>

        <div className="uprow">

          <ul className="hr">

            <li>

              <Link to="/">Главная</Link>

            </li>

            <li >

            <Link to="/catalog">Каталог</Link>

            </li>

            <li >

            <Link to="/basket">Корзина</Link>

            </li>

          </ul>

        </div>

        <div className="secuprow">

          <div className="seclogo">

            <img src={logo1} className="logo" />

          </div>

          <div className="search">

          <Paper

        variant="outlined"

        square={false}

        component="form"

        sx={{ p: '2px 4px', display: 'flex', alignItems: 'center',

        width:windowWidth>730?"35vw":"60vw", backgroundColor: "azure",

        borderRadius: 10,  height: "55%", alignSelf: "center",

        gridColumn: "2/3", gridRow: windowWidth>730?"1/2": "2/3"}}

        onSubmit={(e) => request(e, tempSearchValue)}

      >

        <InputBase

          sx={{ ml: 1, flex: 1 }}

          placeholder="Искать"

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

        />

        <IconButton type="button" sx={{ p: '10px'}} aria-label="search" onClick={(e)=>request(e, tempSearchValue)}>

          <SearchIcon/>

        </IconButton>

      </Paper>

          </div>

          <div className="divphone">

            <div className="number">+375 (25) 919-74-99</div>

          </div>

        </div>

      </header>

    );

Листинг 3.1 – Часть кода на языке JSX на примере компонента Header

Для использования функций не только в данном файле, но и в любых других, которые не выходят за пределы директории src, необходимо первым слово при создании функции указывать export. При это для добавления функции в какой-либо файл необходимо писать слово import.

Для создания много страничного файла использовался React Router. В нём есть элементы, которые можно экспортировать и тем самым создать многостраничный веб-сайт.

<div className="App">

      {!fetching && <Routes>

        <Route path="/"

          element={<>

              <Header></Header>

              <MainPage></MainPage>

              <Footer></Footer>

            </>}>

        </Route>

        <Route path="/catalog"

          element={<>

              <Header></Header>

              <Filters></Filters>

              <Footer></Footer></>}>

        </Route>

         <Route path="/basket"

          element={<>

              <Header></Header>

              <Basket></Basket></>}>

          </Route>

       <Route  path="/orderCreating"

          element={<>

              <Header></Header>

              <Form></Form></>}>

        </Route>

        <Route path="/orderConfirming"

        element={<>

              <Header></Header>

              <Confirm></Confirm></>} >

        </Route>

      </Routes>}

Листинг 3.2 – Код, который показывает схему маршрутов приложения.

Для корректной работы перехода между страницами на нужных страницах используется часть кода, показанная на Листинге 3.2.

Если из большого файла необходимо экспортировать только функцию, находящуюся в этом файле, то при импорте этой функции её название берут в фигурные скобки, а только после этого указывают путь к файлу, в котором лежит данная функция.

Стили подключаются также с помощью import и далее пишут путь к нужному файлу. Так как у React есть наследование стилей, то этим можно избежать избыточности кода.

# 3.2 Сторонние компоненты Material UI.

В процессе разработки данного проекта активно использовались компоненты из библиотеки Material UI (MUI). Эта популярная библиотека для React основана на принципах Material Design от Google и предлагает широкий ассортимент настраиваемых компонентов, что облегчает создание современных и привлекательных пользовательских интерфейсов. MUI включает как небольшие элементы с красивым стилем, так и более сложные компоненты, создание которых вручную заняло бы много времени.

В этом проекте были задействованы такие компоненты, как Paper, InputBase и SearchIcon. Их комбинирование позволило создать полноценную анимированную строку поиска, которая позволяет находить товары как по нажатию на иконку лупы, так и по нажатию клавиши Enter на клавиатуре.



Рисунок 3.1 – Поисковая строка, написанная с использованием MUI.

Также использовались симпатичные чек-боксы в корзине, и красивая загрузка при обновлении страницы.

Таким образом, можно подвести итог, что библиотека Material UI ускорило разработку данного проекта и помогла избежать множества ошибок. Код проекта с MUI приведен в приложении 1.

# 3.3 Написание форм с помощью React Hook Form.

Для написания почти всех форм проекта использовалась библиотека React Hook Form. **React Hook Form** – это популярная библиотека для создания форм в React. Она предоставляет инструменты для управления формами, такие как валидация, сбор данных и управление состоянием, с использованием хуков React.

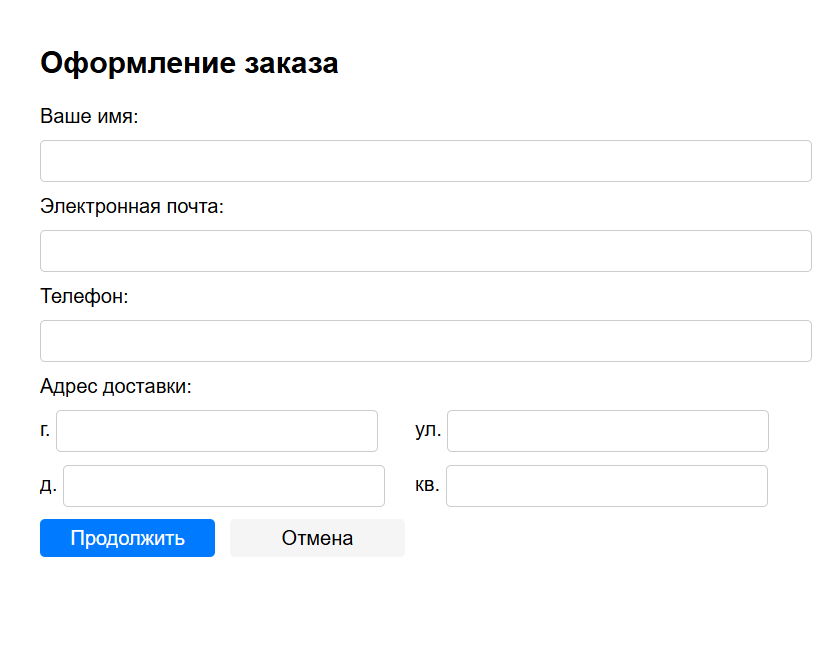


Рисунок 3.2 – Форма, написанная с помощью React Hook Form.

Форма, изображенная на рисунке 3.2 снабжена валидацией, такой как регулярные выражения и поля обязательные к заполнению. Это позволяет запретить пользователю отправить неполные данные, а также данные, не соответствующие шаблону, например электронную почту номер телефона. Для программной реализации данных ограничений используются такие возможности библиотеки как объект register и его параметры pattern и required. При вводе ошибочных данных соответствующие стили полей укажут пользователю на это, а также выведется сообщение о соответствующей ошибке.

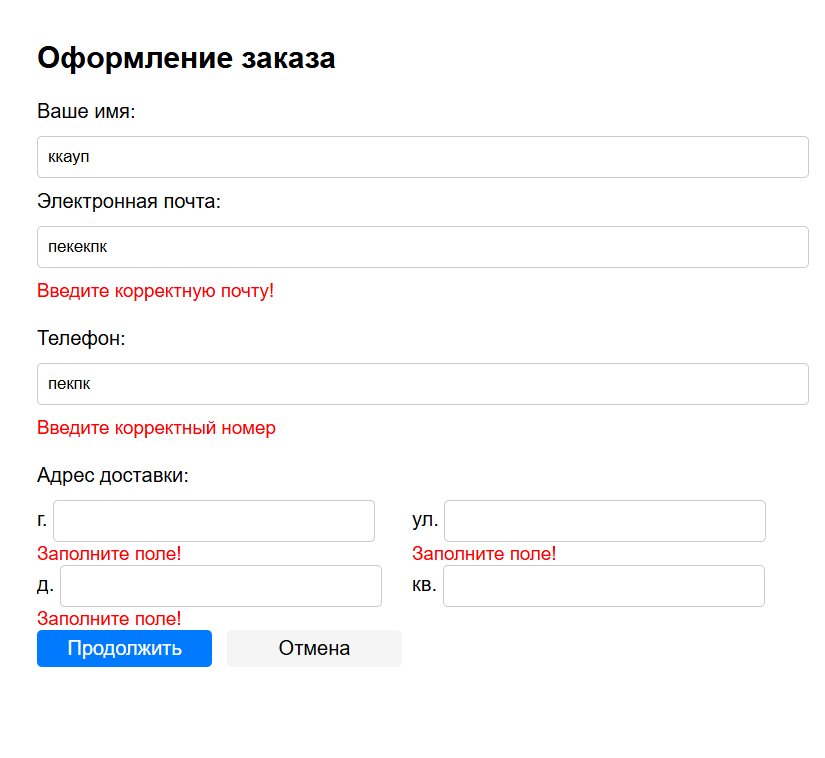


Рисунок 3.3 – Пример вида неправильно заполненной формы.

Код форм написанных с помощью библиотеки React Hook Form приведен в приложении 2.

# 3.4 Хранение данных.

Для хранения данных в проекте использовалась NoSQL база данных Firebase Realtime Database. Это облачное решение от Google предназначено для хранения и синхронизации данных в реальном времени, что позволяет разработчикам создавать приложения с мгновенной передачей информации между клиентскими устройствами. Данные сохраняются в формате JSON, что облегчает организацию и иерархическую структуру информации. Все изменения в базе данных синхронизируются немедленно с подключенными клиентами, позволяя пользователям видеть обновления в реальном времени. Кроме того, Firebase предлагает готовые решения для серверной части приложения.

В базе данных хранится информация о видеокартах магазина. Также сюда поступают данные о заказах, которые затем передаются продавцу для обработки.

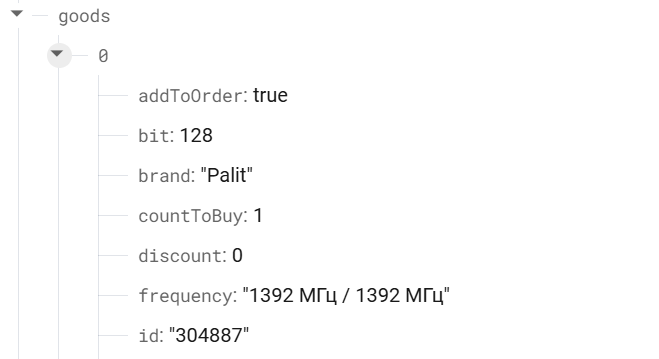


Рисунок 3.4 – Структура части базы данных, отвечающей за видеокарты.

# 3.5 Вывод.

В данном разделе было описано создание проекта. Также описаны основные части его создания и нюансы, которые могут возникнуть при создании приложения с помощью create-react-app.

# 4. Ключевые функции разработанного программного кода, создание значимых документов

# 4.1 Руководство пользователя.

Для корректного функционирования разработанного React-приложения в первую очередь необходимо выбрать директорию, в которой будет размещен проект. Затем нужно открыть командную строку и указать эту директорию, после чего создать приложение с помощью команды npx create-react-app <имя\_проекта>. После завершения процесса создания приложения необходимо перейти в директорию, которая будет носить то же имя, что и проект (<имя\_проекта>), с помощью команды cd <имя директории>. Чтобы запустить приложение, нужно ввести команду npm start, и оно откроется в браузере по адресу localhost:3000/. Если этот адрес занят, система предложит использовать другой.

Поскольку проект включает дополнительные компоненты, которые не входят в базовую установку, необходимо установить их в корневой папке проекта. Для этого снова перейдите в корневую директорию и выполните команду npm add react-router-dom, чтобы установить React Router для навигации между страницами многостраничного веб-сайта. Аналогично следует установить остальные зависимости.

После этого замените файлы в проекте на те, что были скачаны из этого проекта. Если все действия выполнены правильно, главная страница приложения должна отобразиться на экране. Если этого не происходит, вероятно, было допущено ошибка в одном из шагов.

# 4.2 Кроссбраузерное тестирование.

Это тестирование, которое основано на корректном запуске приложения в разных браузерах. Благодаря этому тестированию можно понять в каких браузерах проект работает неисправно и починить это.



Рисунок 4.1 – карточки товаров в браузере Microsoft Edge



Рисунок 4.2 – карточки товаров в браузере Google Chrome

На рисунках 4.1 и 4.2 видно что разницы в отображении каталога на двух браузерах нет.

# 4.3 Тестирование на адаптивность.

Согласно данному тестированию, все элементы должны корректно отображаться при различных размерах экрана, а также на различных устройствах.

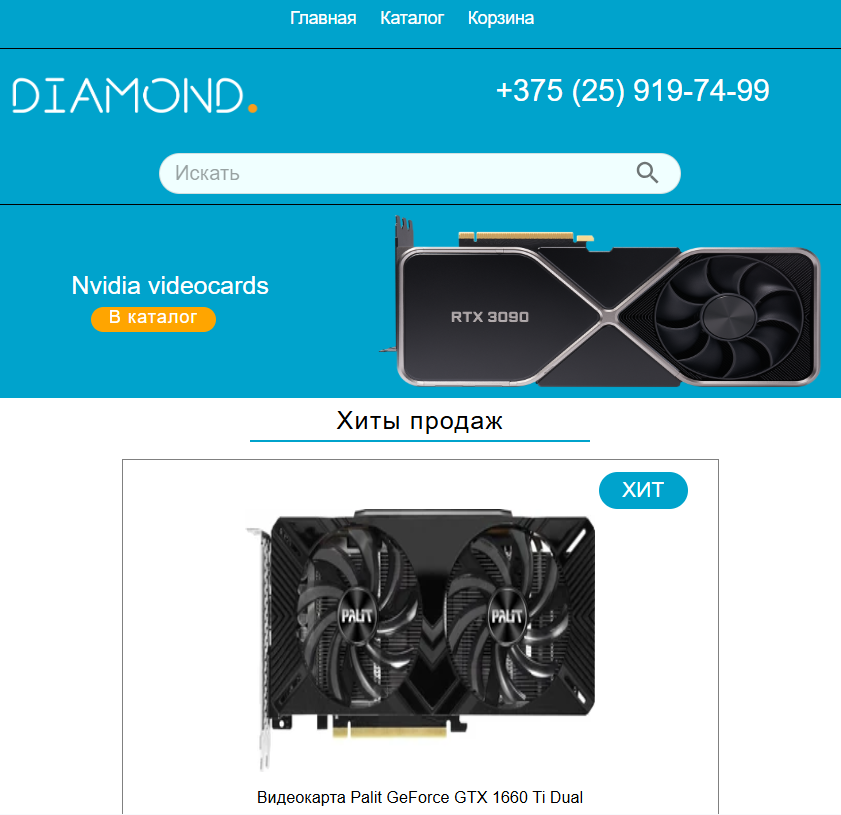


Рисунок 4.3 – Пример адаптивности при ширине экрана 680px

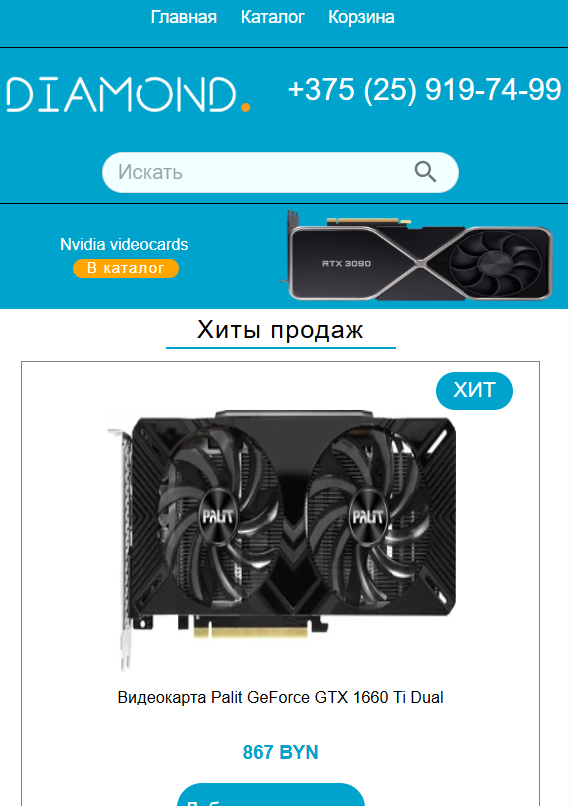


Рисунок 4.4 – Пример адаптивности при ширине экрана 460px

# 4.4 Вывод.

В результате выполнения данного проекта был успешно разработан веб-сайт интернет-магазина, который стал плодом комплексного подхода к разработке и дизайну. В процессе работы использовались современные инструменты и технологии, такие как create-react-app, что значительно упростило начальные этапы создания приложения. Для реализации функциональных возможностей были применены как встроенные, так и пользовательские компоненты, а также хуки, что позволило эффективно отслеживать и обновлять состояние на страницах.

Кроме того, в проекте активно использовались библиотеки Redux Toolkit и React Router, которые обеспечили удобное управление состоянием и навигацией между страницами. Веб-сайт включает четыре основных страницы, каждая из которых предоставляет информацию о товарах, а также включает шапку, подвал, функции сортировки и поиска. Использование grid-вёрстки позволило добиться высокой адаптивности и кроссбраузерности, что делает сайт доступным для пользователей на различных устройствах и в популярных браузерах.

# 5. Заключение.

В результате данной работы был создан веб-сайт интернет-магазина. Для его разработки использовались такие инструменты, как create-react-app, а также как встроенные, так и пользовательские компоненты и хуки, которые помогали отслеживать и обновлять состояние на страницах. Кроме того, были задействованы библиотеки Redux Toolkit и React Router. Веб-сайт состоит из четырех страниц, содержащих информацию о товарах, шапку, подвал, а также функции сортировки и поиска. Благодаря использованию grid-вёрстки была обеспечена кроссбраузерность и адаптивность сайта, что позволяет пользователям просматривать его на любых современных устройствах и в любых браузерах.

На начальном этапе были сформулированы требования к программному продукту, определены цели и задачи его создания, а также разработаны критерии для системы. Затем был создан макет сайта, в котором была выбрана основная цветовая палитра, шрифт и общий дизайн.

Разработка дизайна включала описание структуры программного продукта. Следующим шагом стала вёрстка проекта, в которой были рассмотрены ключевые моменты и особенности работы с create-react-app, поскольку вёрстка в этом случае не отличается от традиционной.

После завершения вёрстки была протестирована адаптивность и кроссбраузерность сайта. Также было подготовлено подробное руководство пользователя, позволяющее каждому желающему легко запустить проект на своем ноутбуке или компьютере.

# 6 Список использованной литературы.

1. Документация HTML5 [Электронный ресурс]. Режим доступа: [https://metanit.com/web/html5/](https://metanit.com/web/html5/%20) – Дата доступа: 10.10.2024
2. Документация CSS3[Электронный ресурс]. Режим доступа: [https://htmlbook.ru/css](https://htmlbook.ru/css%20) – Дата доступа: 10.10.2024
3. Документация React [Электронный ресурс]. Режим доступа: [https://ru.legacy.reactjs.org/docs/getting-started.html](https://ru.legacy.reactjs.org/docs/getting-started.html%20) – Дата доступа: 15.11.2024
4. Документация Redux [Электронный ресурс]. Режим доступа: [https://ru.react-redux.js.org/introduction/getting-started/](https://ru.react-redux.js.org/introduction/getting-started/%20) – Дата доступа: 15.11.2024

# ПРИЛОЖЕНИЕ А

Листинг Основные Компоненты

import "./App.css";

import Header from "./Header";

import Footer from "./Footer";

import Filters from "./Filters";

import MainPage from "./MainPage";

import Basket from "./Basket";

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

import { useEffect, useState } from "react";

import { useDispatch, useSelector } from "react-redux";

import { setBasket } from "./store/BasketSlice";

import Form from "./Form";

import { Confirm } from "./Form";

import CircularProgress from "@mui/material/CircularProgress";

import Box from "@mui/material/Box";

import { getDatabase, ref, get, push, set, query } from "firebase/database";

import { fetchGoods } from "./store/GoodsSlice";

function App() {

  const dispatch = useDispatch();

  const fetching=useSelector(state=>state.goods.fetching);

  useEffect(() => {

    dispatch(fetchGoods());

  }, []);

useEffect(()=>{

  const basket=localStorage.getItem("basket");

  if(!basket)

    return;

  return ()=>{

    dispatch(setBasket(JSON.parse(basket)));

  }

}, [])

  return (

    <div className="App">

      {!fetching && <Routes>

        <Route path="/"

          element={<>

              <Header></Header>

              <MainPage></MainPage>

              <Footer></Footer>

            </>}>

        </Route>

        <Route path="/catalog"

          element={<>

              <Header></Header>

              <Filters></Filters>

              <Footer></Footer></>}>

        </Route>

         <Route path="/basket"

          element={<>

              <Header></Header>

              <Basket></Basket></>}>

          </Route>

       <Route  path="/orderCreating"

          element={<>

              <Header></Header>

              <Form></Form></>}>

        </Route>

        <Route path="/orderConfirming"

        element={<>

              <Header></Header>

              <Confirm></Confirm></>} >

        </Route>

      </Routes>}

      { fetching && <>

      <Header></Header>

      <Box

          sx={{

            display: "flex",

            width: "100vw",

            height: "100vh",

            justifyContent: "center",

            alignItems: "center",

            position: "absolute",

            top: 0,

            zIndex: 2

          }}

        >

          <CircularProgress />

        </Box>

      </>}

    </div>

  );

}

export default App;

import { useSelector, useDispatch } from "react-redux";

import "./Catalog.css";

import {useAddToBasket} from "./hooks.js";

export default function Catalog({goods}) {

  const dispatch=useDispatch();

  const add=useAddToBasket();

  return (

    <div className="featured-products">

      {goods.length !==0 && goods.map((item) => (

        <div className="featured-product"   key={item.id}>

            {item.new && <div className="abs new">Новинка</div>}

            {!!item.discount && <div className="abs discount">Скидка {item.discount}%</div>}

            {item?.hit && <div className="abs hit">Хит</div>}

          <div className="product-image">

            <img src={item.img} />

          </div>

          <div className="product-details">

            <h2>{item.name}</h2>

            <div className="price" style={{ fontWeight: "bold" }}><div>Цена: {item.price -item.price\*item.discount\*0.01} BYN</div>

            {!!item.discount && <div className="crossed" style={{fontWeight: "initial"}}>{item.price} BYN</div>}

            </div>

            <ul>

              <li key={`${Math.floor(Math.random() \* 1000)}+memory+${item.memory}`}>{item.memory} ГБ {item.memoryType}</li>

              <li key={`${Math.floor(Math.random() \* 1000)}+freq+${item.frequency}`}>{item.frequency}</li>

              <li key={`${Math.floor(Math.random() \* 1000)}+sp+${item.sp}`}>{item.sp} sp</li>

              <li key={`${Math.floor(Math.random() \* 1000)}+bit+${item.bit}`}>{item.bit} бит</li>

              <li key={`${Math.floor(Math.random() \* 1000)}+slots+${item.slots}`}>{item.slots}</li>

              <li key={`${Math.floor(Math.random() \* 1000)}+ports+${item.sp}`}>{item.ports.join(", ")}</li>

            </ul>

          </div>

          <div className="korzborder">

              <div className="korzxit3" onClick={(e)=>add(e, item)}>Добавить в корзину</div>

          </div>

        </div>

      ))}

      { !goods.length && <div className="notFound">Товар не найден!</div> }

    </div>

  );

}

import { useSelector } from "react-redux";

import "./MainPage.css";

import mainvid from "./img/mainvid.png";

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

import { useAddToBasket } from "./hooks";

export default function MainPage() {

  return (

    <main className="mainPage">

      <Banner></Banner>

      <StringCatalog type={"hit"}></StringCatalog>

      <StringCatalog type={"new"}></StringCatalog>

      <StringCatalog type={"discount"}></StringCatalog>

    </main>

  );

}

export function Banner() {

  return (

    <div className="chart1">

      <div className="bigleft">

        <div className="zagnvid">Nvidia videocards</div>

        <div className="infovid">

        <Link to="/catalog" className="infoa">В каталог</Link>

        </div>

      </div>

      <div>

        <img src={mainvid} className="mainvid" />

      </div>

    </div>

  );

}

export function StringCatalog(props) {

  const goods = useSelector((state) => state.goods.goods);

  const add=useAddToBasket();

  if (props.type === "hit") {

    return (

      <div className="chart2" key={"chart1"}>

        <div className="row">

          <div className="hityp">Хиты продаж</div>

          <div className="line"></div>

          {goods.map((item) => {if(item?.hit) return(

            <div className="tovar" key={item.id}>

              <div className="xitborder">

                <div className="textxit">ХИТ</div>

              </div>

              <img src={item.img} className="xitimg" />

              <p className="namexit">

                {item.name}

              </p>

              <p className="pricexit">{item.price} BYN</p>

              <div className="korzborder">

                <div onClick={(e)=>add(e, item)}>

                  <p className="korzxit">Добавить в корзину</p>

                </div>

              </div>

            </div>

          )})}

        </div>

      </div>

    );

  }

  else if(props.type==="new"){

    return(

        <div className="chart2" key={"chart2"}>

        <div className="row">

          <div className="novyp">Новинки</div>

          <div className="novline"></div>

          {goods.map((item) => {if(item.new) return(

            <div className="tovar" key={item.id}>

              <div className="novborder">

                <div className="novxit">Новинка</div>

              </div>

              <img src={item.img} className="xitimg" />

              <p className="namexit">

                {item.name}

              </p>

              <p className="pricexit">{item.price} BYN</p>

              <div className="korzborder">

                <div onClick={(e)=>add(e, item)}>

                  <p className="korzxit">Добавить в корзину</p>

                </div>

              </div>

            </div>

          )})}

        </div>

      </div>

    )

  }

  else if(props.type==="discount"){

    return(

        <div className="chart2" key={"chart3"}>

        <div className="row">

          <div className="skidyp">Скидка!</div>

          <div className="novline"></div>

          {goods.map((item) => {if(!!item.discount) return(

            <div className="tovar" key={item.id}>

              <div className="xitborder">

                <div className="textxit">-{item.discount}%</div>

              </div>

              <img src={item.img} className="xitimg" />

              <p className="namexit">

                {item.name}

              </p>

              <p className="pricexit">{item.price} BYN</p>

              <div className="korzborder">

                <div onClick={(e)=>add(e, item)}>

                  <p className="korzxit">Добавить в корзину</p>

                </div>

              </div>

            </div>

          )})}

        </div>

      </div>

    )

  }

}

import { useEffect, useState } from "react";

import "./Form.css";

import { useDispatch, useSelector } from "react-redux";

import { setAllOrder } from "./store/OrderSlice";

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

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

import { deleteFromBasket } from "./store/BasketSlice";

import { getOrderInfo } from "./memoSelector";

export default function Form() {

  const {

    register,

    handleSubmit,

    formState: { errors, isValid },

  } = useForm({ mode: "onBlur" });

  const navigate = useNavigate();

  const dispatch = useDispatch();

  const order = useSelector(getOrderInfo);

  const [windowWidth, setWindowWidth] = useState(window.innerWidth);

  useEffect(() => {

    function handleResize() {

      setWindowWidth(window.innerWidth);

    }

    window.addEventListener("resize", handleResize);

    return () => window.removeEventListener("resize", handleResize);

  }, []);

  function handleFormSubmit(data) {

    console.log(isValid);

    if (!isValid) return;

    const temp = {

      name: data.name,

      mail: data.mail,

      phone: data.phone,

      city: data.city,

      street: data.street,

      house: data.house,

      flat: data.flat,

      totalPrice: order.totalPrice,

      goods: order.goods,

    };

    dispatch(setAllOrder(temp));

    navigate("/orderConfirming");

  }

  useEffect(() => {

    if (!order.totalPrice) navigate("/catalog");

  }, []);

  return (

    <div id="messageModal" className="modal">

      <div className="modal-content">

        <span className="close-button">&times;</span>

        <h2>Оформление заказа</h2>

        <form id="messageForm" onSubmit={handleSubmit(handleFormSubmit)}>

          <label>Ваше имя:</label>

          <input

            {...register("name", {

              required: "Заполните поле!",

              pattern: "^[а-яА-Я]+$|^[a-zA-Z]+$",

            })}

            className="inpsear"

          />

          {errors.name && <div className="error">{errors?.name?.message}</div>}

          <label>Электронная почта:</label>

          <input

            {...register("mail", {

              required: "Заполните поле!",

              pattern: {

                value:

                  /^(([^<>()[\].,;:\s@"]+(\.[^<>()[\].,;:\s@"]+)\*)|(".+"))@(([^<>()[\].,;:\s@"]+\.)+[^<>()[\].,;:\s@"]{2,})$/iu,

                message: "Введите корректную почту!",

              },

            })}

            className="inpsear"

          />

          {errors.mail && <div className="error">{errors?.mail?.message}</div>}

          <label htmlFor="phone">Телефон:</label>

          <input

            {...register("phone", {

              required: "Заполните поле!",

              pattern: {

                value: /^(\+375)?(29|25|44|33|17)\d{7}$/,

                message: "Введите корректный номер",

              },

            })}

            className="inpsear"

          />

          {errors.phone && (

            <div className="error">{errors?.phone?.message}</div>

          )}

          <label>Адрес доставки:</label>

          <div className="place">

            <div>

              {windowWidth > 600 ? "г." : ""}{" "}

              <input

                className="smallInput"

                style={{ marginBottom: errors.city ? 0 : "10px" }}

                {...register("city", {

                  required: "Заполните поле!",

                  pattern: /^[а-яА-Я]+$|^[a-zA-Z]+$/,

                })}

                placeholder={windowWidth <= 600 ? "Город" : null}

              />

              {errors.city && (

                <div className="error">{errors?.city?.message}</div>

              )}

            </div>

            <div>

              {windowWidth > 600 ? "ул." : ""}{" "}

              <input

                className="smallInput"

                style={{ marginBottom: errors.street ? 0 : "10px" }}

                {...register("street", {

                  required: "Заполните поле!",

                  pattern: /^[а-яА-Я]+$|^[a-zA-Z]+$/,

                })}

                placeholder={windowWidth <= 600 ? "Улица" : null}

              />

              {errors.street && (

                <div className="error">{errors?.street?.message}</div>

              )}

            </div>

            <div>

              {windowWidth > 600 ? "д." : ""}{" "}

              <input

                className="smallInput"

                style={{ marginBottom: errors.house ? 0 : "10px" }}

                {...register("house", {

                  required: "Заполните поле!",

                  pattern: /\d{1,4}$/,

                })}

                placeholder={windowWidth <= 600 ? "Дом" : null}

              />

              {errors.house && (

                <div className="error">

                  {errors?.house?.message || "Заполните корректно!"}

                </div>

              )}

            </div>

            <div>

              {windowWidth > 600 ? "кв." : ""}{" "}

              <input

                className="smallInput"

                style={{ marginBottom: errors.flat ? 0 : "10px" }}

                {...register("flat", { pattern: /\d{1,4}$|^$/ })}

                placeholder={windowWidth <= 600 ? "Квартира" : null}

              />

              {errors.flat && (

                <div className="error">{"Заполните корректно!"}</div>

              )}

            </div>

          </div>

              <div style={{display: "flex"}}>

              <input type="submit" className="butseacr submit" value="Продолжить"/>

          <button onClick={()=>{navigate('/basket')}} className="submit">Отмена</button>

              </div>

        </form>

      </div>

    </div>

  );

}

export function Confirm() {

    const navigate=useNavigate();

    const order = useSelector(getOrderInfo);

    const dispatch=useDispatch();

    useEffect(() => {

        if (!order.city) navigate("/catalog");

      }, []);

      function handleSubmit(e){

        e.preventDefault();

        order.goods.forEach((item)=>{

            dispatch(deleteFromBasket(item.id));

        })

        navigate('/catalog');

        alert("Заказ подтвержден!");

      }

  return (

    <div id="messageModal" className="modal">

      <div className="modal-content2">

        <h2>Подтверждение заказа</h2>

        <form id="messageForm" onSubmit={(e)=>handleSubmit(e)}>

            <div>{`Вы хотите сделать заказ на ${order.goods.length} ${order.goods.length>1?"товаров":"товар"}, на сумму ${order.totalPrice} руб.?`}</div>

            <div className="buttonsContainer">

                <input type="submit" value="Подтвердить"></input>

                <button onClick={()=>navigate('/orderCreating')} className="cancel">Отмена</button>

            </div>

        </form>

      </div>

    </div>

  );

}

# ПРИЛОЖЕНИЕ Б

Листинг Компоненты MUI

<Paper

        variant="outlined"

        square={false}

        component="form"

        sx={{ p: '2px 4px', display: 'flex', alignItems: 'center',

        width:windowWidth>730?"35vw":"60vw", backgroundColor: "azure",

        borderRadius: 10,  height: "55%", alignSelf: "center",

        gridColumn: "2/3", gridRow: windowWidth>730?"1/2": "2/3"}}

        onSubmit={(e) => request(e, tempSearchValue)}

      >

        <InputBase

          sx={{ ml: 1, flex: 1 }}

          placeholder="Искать"

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

        />

        <IconButton type="button" sx={{ p: '10px'}} aria-label="search" onClick={(e)=>request(e, tempSearchValue)}>

          <SearchIcon/>

        </IconButton>

      </Paper>

<Box

          sx={{

            display: "flex",

            width: "100vw",

            height: "100vh",

            justifyContent: "center",

            alignItems: "center",

            position: "absolute",

            top: 0,

            zIndex: 2

          }}

        >

          <CircularProgress />

        </Box>

<div className="thirdcol">

                <Checkbox sx={{paddingTop: 0, paddingBottom: 0,}} checked={item.addToOrder===true}

                onChange={(e)=>changeBeanByAddToOrder(e, item)}/>

              </div>

# ПРИЛОЖЕНИЕ В

Листинг Формы на React Hook Form

import { useEffect, useState } from "react";

import "./Form.css";

import { useDispatch, useSelector } from "react-redux";

import { setAllOrder } from "./store/OrderSlice";

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

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

import { deleteFromBasket } from "./store/BasketSlice";

import { getOrderInfo } from "./memoSelector";

export default function Form() {

  const {

    register,

    handleSubmit,

    formState: { errors, isValid },

  } = useForm({ mode: "onBlur" });

  const navigate = useNavigate();

  const dispatch = useDispatch();

  const order = useSelector(getOrderInfo);

  const [windowWidth, setWindowWidth] = useState(window.innerWidth);

  useEffect(() => {

    function handleResize() {

      setWindowWidth(window.innerWidth);

    }

    window.addEventListener("resize", handleResize);

    return () => window.removeEventListener("resize", handleResize);

  }, []);

  function handleFormSubmit(data) {

    console.log(isValid);

    if (!isValid) return;

    const temp = {

      name: data.name,

      mail: data.mail,

      phone: data.phone,

      city: data.city,

      street: data.street,

      house: data.house,

      flat: data.flat,

      totalPrice: order.totalPrice,

      goods: order.goods,

    };

    dispatch(setAllOrder(temp));

    navigate("/orderConfirming");

  }

  useEffect(() => {

    if (!order.totalPrice) navigate("/catalog");

  }, []);

  return (

    <div id="messageModal" className="modal">

      <div className="modal-content">

        <span className="close-button">&times;</span>

        <h2>Оформление заказа</h2>

        <form id="messageForm" onSubmit={handleSubmit(handleFormSubmit)}>

          <label>Ваше имя:</label>

          <input

            {...register("name", {

              required: "Заполните поле!",

              pattern: "^[а-яА-Я]+$|^[a-zA-Z]+$",

            })}

            className="inpsear"

          />

          {errors.name && <div className="error">{errors?.name?.message}</div>}

          <label>Электронная почта:</label>

          <input

            {...register("mail", {

              required: "Заполните поле!",

              pattern: {

                value:

                  /^(([^<>()[\].,;:\s@"]+(\.[^<>()[\].,;:\s@"]+)\*)|(".+"))@(([^<>()[\].,;:\s@"]+\.)+[^<>()[\].,;:\s@"]{2,})$/iu,

                message: "Введите корректную почту!",

              },

            })}

            className="inpsear"

          />

          {errors.mail && <div className="error">{errors?.mail?.message}</div>}

          <label htmlFor="phone">Телефон:</label>

          <input

            {...register("phone", {

              required: "Заполните поле!",

              pattern: {

                value: /^(\+375)?(29|25|44|33|17)\d{7}$/,

                message: "Введите корректный номер",

              },

            })}

            className="inpsear"

          />

          {errors.phone && (

            <div className="error">{errors?.phone?.message}</div>

          )}

          <label>Адрес доставки:</label>

          <div className="place">

            <div>

              {windowWidth > 600 ? "г." : ""}{" "}

              <input

                className="smallInput"

                style={{ marginBottom: errors.city ? 0 : "10px" }}

                {...register("city", {

                  required: "Заполните поле!",

                  pattern: /^[а-яА-Я]+$|^[a-zA-Z]+$/,

                })}

                placeholder={windowWidth <= 600 ? "Город" : null}

              />

              {errors.city && (

                <div className="error">{errors?.city?.message}</div>

              )}

            </div>

            <div>

              {windowWidth > 600 ? "ул." : ""}{" "}

              <input

                className="smallInput"

                style={{ marginBottom: errors.street ? 0 : "10px" }}

                {...register("street", {

                  required: "Заполните поле!",

                  pattern: /^[а-яА-Я]+$|^[a-zA-Z]+$/,

                })}

                placeholder={windowWidth <= 600 ? "Улица" : null}

              />

              {errors.street && (

                <div className="error">{errors?.street?.message}</div>

              )}

            </div>

            <div>

              {windowWidth > 600 ? "д." : ""}{" "}

              <input

                className="smallInput"

                style={{ marginBottom: errors.house ? 0 : "10px" }}

                {...register("house", {

                  required: "Заполните поле!",

                  pattern: /\d{1,4}$/,

                })}

                placeholder={windowWidth <= 600 ? "Дом" : null}

              />

              {errors.house && (

                <div className="error">

                  {errors?.house?.message || "Заполните корректно!"}

                </div>

              )}

            </div>

            <div>

              {windowWidth > 600 ? "кв." : ""}{" "}

              <input

                className="smallInput"

                style={{ marginBottom: errors.flat ? 0 : "10px" }}

                {...register("flat", { pattern: /\d{1,4}$|^$/ })}

                placeholder={windowWidth <= 600 ? "Квартира" : null}

              />

              {errors.flat && (

                <div className="error">{"Заполните корректно!"}</div>

              )}

            </div>

          </div>

              <div style={{display: "flex"}}>

              <input type="submit" className="butseacr submit" value="Продолжить"/>

          <button onClick={()=>{navigate('/basket')}} className="submit">Отмена</button>

              </div>

        </form>

      </div>

    </div>

  );

}

export function Confirm() {

    const navigate=useNavigate();

    const order = useSelector(getOrderInfo);

    const dispatch=useDispatch();

    useEffect(() => {

        if (!order.city) navigate("/catalog");

      }, []);

      function handleSubmit(e){

        e.preventDefault();

        order.goods.forEach((item)=>{

            dispatch(deleteFromBasket(item.id));

        })

        navigate('/catalog');

        alert("Заказ подтвержден!");

      }

  return (

    <div id="messageModal" className="modal">

      <div className="modal-content2">

        <h2>Подтверждение заказа</h2>

        <form id="messageForm" onSubmit={(e)=>handleSubmit(e)}>

            <div>{`Вы хотите сделать заказ на ${order.goods.length} ${order.goods.length>1?"товаров":"товар"}, на сумму ${order.totalPrice} руб.?`}</div>

            <div className="buttonsContainer">

                <input type="submit" value="Подтвердить"></input>

                <button onClick={()=>navigate('/orderCreating')} className="cancel">Отмена</button>

            </div>

        </form>

      </div>

    </div>

  );

}