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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. С. Позин

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 Составляющие структуры программного продукта. 9](#_Toc184850847)

[2.5 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 Слайдеры написанные с помощью React-slick. 15](#_Toc184850853)

[3.4 Хранение данных. 16](#_Toc184850854)

[3.4 Аутентификация пользователей. 17](#_Toc184850855)

[3.3 Вывод. 17](#_Toc184850856)

[4. Ключевые функции разработанного программного кода, создание значимых документов 18](#_Toc184850857)

[4.1 Руководство пользователя. 18](#_Toc184850858)

[4.2 Кроссбраузерное тестирование. 18](#_Toc184850859)

[4.3 Тестирование на адаптивность. 20](#_Toc184850860)

[4.4 Вывод. 21](#_Toc184850861)

[5. Заключение. 22](#_Toc184850862)

[6 Список использованной литературы. 23](#_Toc184850863)

[ПРИЛОЖЕНИЕ А 24](#_Toc184850864)

[ПРИЛОЖЕНИЕ Б 32](#_Toc184850865)

[ПРИЛОЖЕНИЕ В 35](#_Toc184850866)

# ВВЕДЕНИЕ

Присутствие бизнеса в Интернете оказывает большое влияние на его успех. В большинстве случаев клиенты перед совершением покупки обращают внимание на наличие веб-сайта. Его наличие присутствия в Интернете, может иметь решающее значение для бизнеса. Кроме всего этого веб-сайт является лицом бренда и компании в общем.

Бизнес должен иметь свой собственный веб-сайт по целому ряду причин. Одна из основных — демонстрация основного ассортимента товаров потенциальным клиентам. Также это помогает чётко определить, что из себя представляет компания, и повысить свои шансы на успешную продажу своего продукта клиентам.

Веб-сайт — то, что помогает выделить компанию среди конкурентов. Благодаря ему можно найти покупателей, которые заинтересованы именно в данной продукции, показать свои лучшие товары или сильные стороны. Кроме этого можно продвинуть свою новую продукцию или услугу. Также заинтересованные люди смогут связаться с компанией напрямую с помощью информации, указанной на веб-сайте, что также увеличит продажи. Поскольку веб-сайт работает круглосуточно и без выходных, то компания может легко публиковать обновления и объявления для своих клиентов, а также быстро обновлять информацию, если это будет необходимо.

Веб-сайт даёт возможность использовать различные приёмы интернет-маркетинга: уведомления об акциях, реклама, создание всплывающих окон и т.п.

Данные факты отражают актуальность тематики курсовой работы.

Целью курсового проекта являетсяразработать веб-сайт для чайного магазина с использованием библиотеки React.js.

К задачам курсового проекта относится:

1. Проанализировать дизайн существующих веб-сайтов, инструменты и библиотеки для создания веб-сайта, а также его оформления.

2. Разработать макет и прототип сайта.

3. Разработать структуру веб-сайта.

4. Наполнить веб-сайт информацией, соответствующей определённой теме.

5. Создать дизайн сайта, используя React.js, а также создать интерактивные элементы

5. Протестировать веб-сайт.

Целевой аудиторией будут люди, которые хотят приобрести новую технику или знать о новинках в мире смартфонов, наушников и других устройств.

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

# 1.1 Обследование объекта и обоснование необходимости создания проекта.

На данный момент существует огромное количество способов продвижения своей продукции в Интернете. Некоторые создают красивые сайты, которые легко бросаются в глаза и быстро запоминаются посетителям, другие же – могут похвастаться отличным наполнением своего веб-сайта, третьи – обладают отличным качеством продукции, поэтому в погоне за таким товаром покупатели возвращаются вновь после покупки какого-то товара из ассортимента.

Однако вне зависимости от того, чем может похвастаться кампания, которая продаёт свой товар в сети Интернет, ей нужен веб-сайт. Когда посетитель видит какой-то магазин или ищет какую-то определённую продукцию, он в первую очередь пользуется Интернетом и начинает искать те магазины, которые подойдут именно ему по различным критериям, например, цена или месторасположение магазина. В таких случаях веб-сайт данной кампании является главным лицом от которого зависит купит ли в итоге покупатель именно в этой кампании товар или нет. На исход влияет огромное количество факторов начиная от эстетичности страниц, количества информации о кампании и самих товарах, заканчивая отзывами на других сайтах, чтобы удостовериться в уровне обслуживания в этой кампании.

Именно учитывая всё вышеперечисленное можно понять, что веб-сайт – важная составляющая каждого бизнеса, от которой в некотором смысле будет зависеть обанкротиться ли кампания или станет популярной через время.

# 1.2 Формирование требований к проекту.

В данной курсовой работе веб-сайт создавался с помощью фреймворка React, также использовались css и css modules для задания стилей. Данный фреймворк удобен тем, что есть возможности использовать скрипты, которые позволяют, например, динамически обновлять страницу, также React имеет своё встроенное DOM-дерево, что позволяет без особых трудов сократить нагрузку на компьютер клиента, который будет посещать данный веб-сайт, это достигается путём того, что React отслеживает изменившиеся компоненты и обновляет только их а не всю страницу целиком, как это происходит в обычных HTML-страницах.

Основными требованиями к данной курсовой работе являются:

Лаконичность и простота дизайна, страницы не должны быть сильно перегружены информацией, но при этом должны отражать всю суть данной страницы и самого проекта в целом. Это достигается путём добавления только ключевой, значимой для посетителя и потенциального покупателя информации. В описаниях и на страницах веб-сайта также не должно быть излишней информации, которая усложнит поиск предполагаемо интересуемой информации посетителя веб-сайта.

Благодаря простоте дизайна есть возможность использовать базовую (основную) палитру цветов, выбранную для этого веб-сайта, а также визуально обратить внимание посетителя именно на те части веб-сайта, которые необходимо выделить, например, для повышения количества продаж или других целей.

Текст на всех страницах вне зависимости от его расположения должен иметь оптимальный размер, а также его можно было бы без затруднений прочитать. Благодаря такому тексту посетителю не надо будет долго всматриваться в текст, в попытках разобраться, что там написано, а, следовательно, это не отпугнёт потенциальных покупателей и, возможно, повысит продажи.

Навигация на веб-сайте должна быть понятна при беглом просмотре глазами. Это поможет посетителю веб-сайта не «заблудиться» во всех страницах, которые содержит посещённый им веб-сайт, а также быстро переключаться между нужными страницами.

Кроме всего эта навигация должна быть удобной. Следует избегать близкого расположения ссылок на другие страницы, так как пользователь может случайно промахнуться и, как итог, перейти на другую страницу, чего пользователю явно не хотелось.

Особое внимание стоит уделить адаптивности веб-сайта. Благодаря адаптивности веб-сайт корректно отобразиться на всех устройствах, а также размерах экрана. Как итог сайт будет выглядеть презентабельно и красиво на каком бы устройстве его не открыли. Также благодаря этого это расположит клиента, и он с большей вероятностью может приобрести продукцию.

# 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.

Основными требования к системе являются: корректная работа всего сервиса, стабильность при работе с данной системой, также сюда входит отказоустойчивость. Кроме всего прочего система не должна «тормозить» при выполнении пользователем действий, или при посещении сайта, или при использовании функционала сайта. Кроме всего прочего компоненты, которые составляют данную систему, должны иметь возможность переноситься в неизменном виде и на другие сервисы, если такое необходимо.

Требованиями к функциям, выполняемым системой, являются: правильность работы при их выполнении, также функции не должны сложно читаться. Кроме всего этого функции должны быть отказоустойчивыми и разделены на несколько файлов. Данные файлы создаются в зависимости от таких критериев как, логика самого приложения, результат выполнения функции. Данные функции должны иметь возможность импортироваться для использования в других местах приложения. Основные функции размещаются в тех же файлах, где и их вызов для простоты чтения кода и отсутствия загруженности огромным количеством файлов, что может привести к затруднению понятия логики приложения.

Основными требованиями к видам обеспечения являются: наличие установленного фреймворка React. Также необходимо подключение к сети Интернет и любой установленный браузер. Данная система может работать на всех операционных системах с графической оболочкой подобной Windows. Кроме всего прочего в корневой каталог необходимо установить модули React Router, Redux Toolkit, react-redux, MUI, React Hook Form, Firebase(модули для React.js) и другие модули.

# 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 элеткроники». 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 данном курсовом проекте необходимо создать веб-сайт Чайного магазина с использованием фреймворка React, а также использовать Scss для задания стилей всем элементам на странице. React является основой данного проекта, приложение написанное на нём создаётся с помощью пакетного менеджера npm, а именно его комнады npx create-react-app. Сам проект состоит из основных 4 страниц: главная, ассортимент с товаром, корзина, страница о кампании.

Главной задачей веб-сайта является создание многостраничного проекта используя функции, встроенные и пользовательские хуки, а также дополнительные компоненты для поддержания корректной работы приложения. Кроме всего прочего все страницы должны быть адаптивны и корректно работать.

Все страницы имеют одинаковую верхнюю и нижнюю части. На верхней части расположены ссылки на все остальные страницы приложения, а внизу расположена дополнительная навигация, основывающаяся на категориях товаров.

Наполнение страниц соответствует их названию.

Данный проект, а также дизайн является адаптивным и кроссбраузерным (корректное отображение в таких браузерах, как Google Chrome, Mozilla Firefox, Microsoft Edge).

# 2.2 Построение прототипов интерфейсов и макетирование страниц.

Для большей наглядности и простоты исправления ошибок проектирования и дизайна было выбрано приложение Figma. Данное приложение позволило создать прототип всех страниц сайта и понять, как будет выглядеть сайт в конечном итоге. Также использование функции соединения компонента на одной странице с компонентом на другой странице или же целой страницей позволяет в режиме просмотра сделать прототип функционирующего многостраничного сайта.

# 2.3 Разработка дизайна элементов веб-сайта.

Для достижения адаптивности использовалась вёрстка с использованием grid-элементов. Данная вёрстка помогает легко расположить элементы в нужном порядке, а также делает элементы адаптивными и подстраиваемыми под любой размер экрана.

Все стили задаются с помощью css, и modules css. Весь сайт сделан с использованием светлого, чёрного и голубого цветов. Такая цветовая палитра не является слишком броской, легко может сочетаться с другими цветами для обращения внимания клиента на какие-либо элементы, также данный дизайн смотрится довольно лаконично и не нагружает страницы веб-сайта.

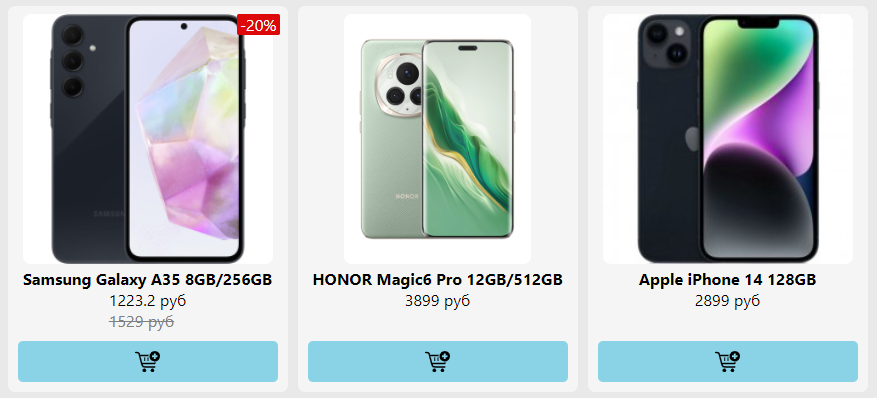


Рисунок 2.1 – Часть веб-страницы с ассортиментом магазина

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 RubikGlitch-Regular.

2.3 Разработка логотипа.

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



Рисунок 2.2 – Логотип магазина.

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

# 2.4 Составляющие структуры программного продукта.

Приложение созданное с помощью create-react-app состоит из нескольких папок. Все основные файлы находятся в папке src. В этой папке находятся файлы со всеми страницами, которые используются в данном проекте, например Catalog.js, Footer.js, App.js.

В папке hooks содержится файл hooks.js в котором написаны кастомные хуки.

В папке fonts хранится файл пользовательского шрифа RubikGlitch-Regular.ttf.

В папке Selectors содержится файл GetGoods.js в котором написан мемоизированный селектор.

# 2.5 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.

{ windowWidth>=725 && (<div className="menu">

<div className="point" ><Link to='/catalog' className="headerLink">Каталог</Link></div>

<div className="point" >

<Link to='/basket' className="headerLink">Корзина</Link>

{(!!bean.length && authorized) && <div className={bean.length>9?"beanCount outline":"beanCount"}>{bean.length>99?"99+":bean.length}</div>}

</div>

<div className="point" ><Link to='/about' className="headerLink">О нас</Link></div>

<div className="point" onClick={()=>dispatch(setShowLog(true))} >

<svg

id="key"

onMouseOver={changeFill}

onMouseLeave={changeFillBack}

xmlns="http://www.w3.org/2000/svg"

x="0px"

y="0px"

width="30"

height="30"

viewBox="0 0 24 24"

style={{ fill: svgFill}}

>

<path d="M 7 5 C 3.134 5 0 8.134 0 12 C 0 15.866 3.134 19 7 19 C 10.170669 19 12.846171 16.890989 13.707031 14 L 18 14 L 18 17 L 22 17 L 22 14 L 24 14 L 24 10 L 13.707031 10 C 12.846171 7.1090112 10.170669 5 7 5 z M 7 9 C 8.657 9 10 10.343 10 12 C 10 13.657 8.657 15 7 15 C 5.343 15 4 13.657 4 12 C 4 10.343 5.343 9 7 9 z"></path>

</svg>

</div>

</div>)

}

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.

<Header></Header>

<Routes>

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

<Route path="/basket" element={<Bean />}></Route>

<Route path="/catalog" element={<Search />}></Route>

<Route

path="/redactedCatalog/:brend/:category"

element={<Catalog />}

></Route>

<Route path="/productPage/:key" element={<ProductPage />}></Route>

<Route path="/about" element={<About />}></Route>

<Route path="/order" element={<Order />}></Route>

</Routes>

<Footer></Footer>

Рисунок 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. Например, стили из index.css файла, который создаёт автоматически при создании приложения, наследуются всеми последующими элементами в проекте.

@font-face {

font-family: RubikGlitch;

src: url(fonts/RubikGlitch-Regular.ttf);

}

Листинг 3.2 – Подключение пользовательского шрифта

# 3.2 Сторонние компоненты Material UI.

При разработке данного проекта активно использовались сторонние компоненты из библиотеки Material UI.

Material UI (MUI) -­ это популярная библиотека компонентов для React, которая следует принципам дизайна Material Design от Google. Она предоставляет обширный набор настраиваемых компонентов, которые помогают быстро создавать привлекательные и современные пользовательские интерфейсы. В MUI можно найти как обычные красиво стилизованные маленькие элементы, так и большие сложные компоненты на написание которых самостоятельно программист бы затратил большое количество времени.

В данном проекте использовались такие MUI компоненты как Paper, InputBase, SearchIcon комбинируя которые написана полноценная анимированная поисковая строка, которая начинает искать товары как по нажатию иконки лупы, так и по нажатию клавиши Enter на клавиатуре.



Риунок 3.1 – Поисковая строка написанная с использованием MUI.

Minimum distance slider, который позволяет пользователям выбирать значения из диапазона значений, перемещая ползунок по шкале. Он идеально подходит для настройки параметров, таких как громкость, яркость или применение фильтров изображений. В данном проекте использовался для настройки ценового диапазона, по которому будут фильтроваться отображаемые товары.



Рисунок 3.2 – Ползунок для фильтрации цены написанный с использованием MUI.

Также использовался компонент Pagination для разделения большого количества контента на страницы и навигации между ними. Он позволяет пользователям перемещаться по страницам контента, предоставляя удобный интерфейс для переключения между ними. Стили библиотеки MUI позволили тонко настроить визуал пагинации.



Рисунок 3.3 – Пагинация написанная с использованием MUI.

Также использовались симпатичные чекбоксы и радио-кнопки из данной библиотеки.

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 бибилиотека 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.

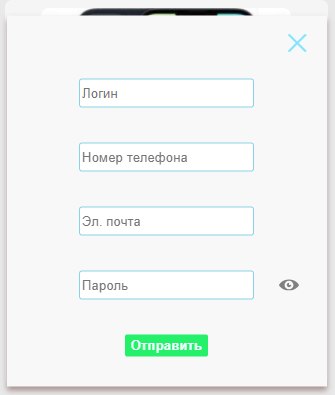


Рисунок 3.4 – Форма написанная с помощью React Hook Form.

Форма изображенная на рисунке 3.4 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 или недостаточно сложный пароль. 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.

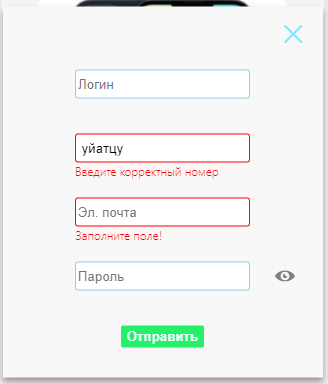


Рисунок 3.5 – Пример вида неправильно заполненной формы.

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

# 3.4 Слайдеры написанные с помощью React-slick.

На главной странице и на странице каждого продукта реализованы слайдеры написанные с помощью библиотеки React-slick. **React Slick** – это мощная библиотека для создания каруселей в React-приложениях. Она представляет собой порт популярного карусельного плагина Slick Carousel, адаптированный для использования с React. Её преимущество заключается в гибкой настройке, с помощью которой можно указать нужны ли стрелки, должен ли слайдер мотаться самостоятельно, с каким временным интервалом это делать и т. д. При чем на разных диапазонах ширины экрана настройка может быть разная. Также слайдеры в данной библиотеке адаптивны под любую ширины экрана и разработчик может сам указать сколько элементов отображать на какой ширине.

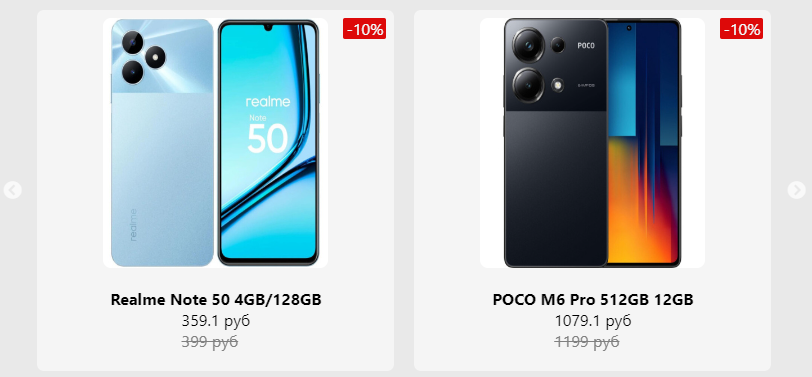


Рисунок 3.6 – Слайдер распложенный на главной странице.

На рисунке 3.6 слайдер может одновременно отображать только 2 карточки, это идеально подходит ширине экрана карточки не сжаты и не растянуты, стрелки не касаются краёв экрана.

# 3.4 Хранение данных.

Для хранения данных использовалась NoSQL СУБД Firebase Realtime Database. Firebase Realtime Database — это облачное решение для хранения и синхронизации данных в реальном времени. Оно является частью платформы Firebase, разработанной Google, и позволяет разработчикам создавать приложения, которые требуют мгновенной передачи данных между клиентскими устройствами. Данные хранятся в формате JSON, что позволяет легко организовывать и иерархически структурировать информацию. Все изменения в базе данных немедленно синхронизируются с подключенными клиентами, что позволяет пользователям видеть обновления в реальном времени. Более того Firebase предоставляет готовые решения для серверной части приложения.

В базе данных хранится информация о пользователях, их корзинах и товарах магазина. Также в базу приходят данные о заказах, для дальнейшей отправки их продавцу.

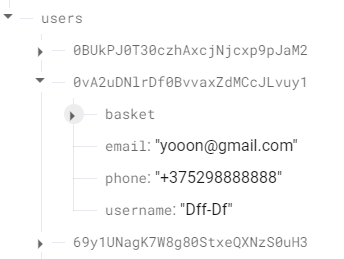


Рисунок 3.7 – Структура части базы данных отвечающей за пользователей.

На рисунке 3.7 длинная комбинация букв отвечает за Id пользователя.

# Аутентификация пользователей.

Для аутентификации пользователей использовалось готовое решении предоставленное Firebase. Аутентификация в Google Firebase — это мощный и гибкий механизм, который позволяет разработчикам управлять доступом пользователей к их приложениям. Firebase Authentication поддерживает множество методов аутентификации и упрощает процесс интеграции аутентификации в мобильные и веб-приложения. Предоставляет такие методы как: электронная почта и пароль, аутентификация через социальные сети, анонимная аутентификация и телефонная аутентификация, т.е. при входе в аккаунт будет приходить SMS с кодом.

В проекте использовался самый простой метод, это вход по электронной почте и паролю.

Таким образом Google Firebase сыграла не маловажную роль в проекте, благодаря своей базе данных Realtime Database и функциям для работы с пользователями.

# 3.3 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 корректной работы данного разработанного React-приложения в первую очередь необходимо определиться с директорией, в которой будет находиться данный проект.

После этого необходимо в командной строке указать эту директорию и создать React-приложение. Для этого в командной строке выполняется npx create-react-app <имя\_проекта>. По прошествую времени создастся React-приложение. Необходимо перейти в директорию, которая имеет такое же название как и проект (директория <имя\_прокта), с помощью команды cd <имя директории>. Для запуска приложения необходимо написать npm start. И приложение запуститься в браузере с адресом localhost:3000/. Если этот хост занят, то будет предложено открыть на другом хосте.

Так как данный проект содержит дополнения, которые не входят в основной проект, то необходимо в корневой папке данного проекта доустановить компоненты. Как было сказано необходимо перейти в корневой каталог проекта и выполнить команду npm add react-router-dom. После этого установится React Router, который используется для перехода между страницами много страничного веб-сайта. Таким же способом нужно установить остальные зависимости.

Далее необходимо заменить файл в директории проекта на файл, который был скачан из этого проекта.

После всех выполненных манипуляций на странице, где запускался проект, будет главная страница данного проекта. Если же такого не происходит, скорее всего какое-то из действий выполнено неверно.

# 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.

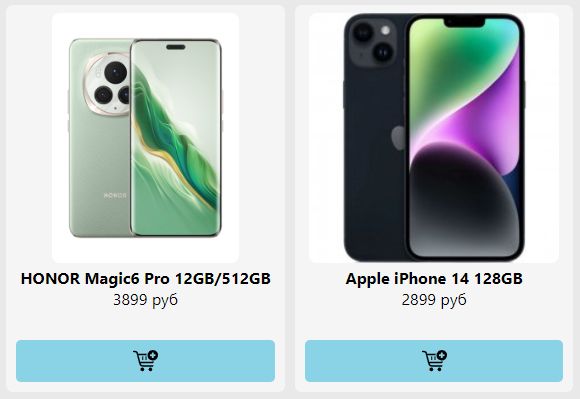


Рисунок 4.1 – карточки товаров в браузере Microsoft Edge

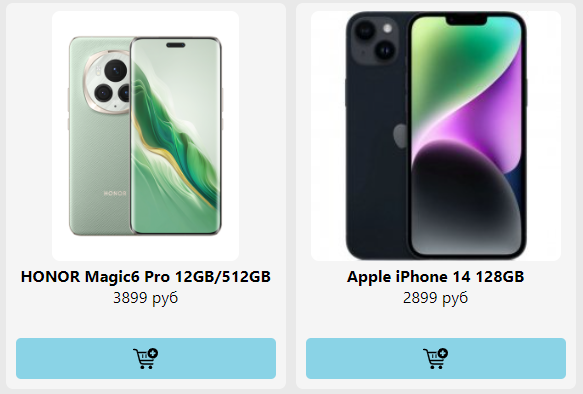


Рисунок 4.2 – карточки товаров в браузере Google Chrome

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.

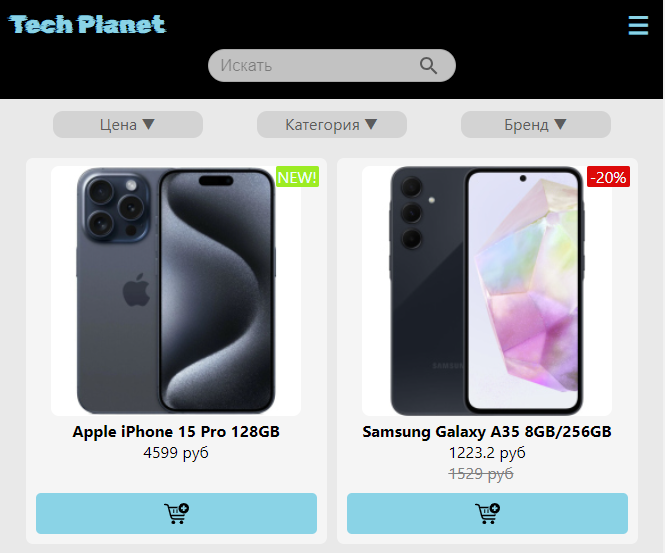


Рисунок 4.3 – Пример адаптивности при ширине экрана 680px

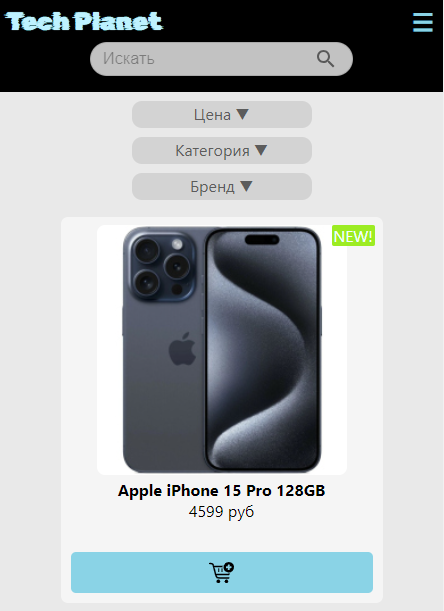


Рисунок 4.4 – Пример адаптивности при ширине экрана 460px

# 4.4 Вывод.

В данном разделе было разработано руководство пользователя, а также продемонстрировано тестирование на адаптивность и кроссбраузерность.

# 5. Заключение.

По итогам данной работы был разработан веб-сайт интернет-магазина. Для его создания были использованы: create-react-app, компоненты как встроенные, так и пользовательские, а также хуки, которые помогали отслеживать и обновлять состояние на странице. Кроме этого был использованы другие библиотеки, такие как Redux Toolkit и React Router. Данный веб-сайт состоит из 4 страниц, которые содержат информацию о товарах, шапку, подвал, сортировку и поиск. Также благодаря grid-вёрстке были реализованы кроссбраузерность и адаптивность веб-сайта, что позволяет просматривать данную работу в любом браузере и на любом современном устройстве.

Сперва были сформированы требования к программному продукту, поставлена задача, определены цели и назначения создания данного программного продукта, а также были сформированы требования к системе.

В начале работы был сделан макет сайта, а также определена основная цветовая палитра, шрифт и дизайн сайта в принципе.

Были разработаны дизайны. Также описана структура программного продукта.

Следующим шагом стала вёрстка проекта, в данном разделе были разобраны только ключевые моменты и особенности при вёрстке используя create-react-app, так как от обычной вёрстки, не учитывая данных особенностей, оно не отличается.

После этого была протестирована адаптивность и кроссбраузерность. Также описана подробное руководство пользователя, которое позволит каждому запустить данный проект у себя на 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: 6.10.2024
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: 20.11.2024
5. Дронов В. А. React 17. Разработка веб-приложений на Javascript. – Санкт-Петербург: БХВ-Петербург, 2022. – 383 с

# ПРИЛОЖЕНИЕ А

Листинг Основные Компоненты

import "./Catalog.css";

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

import { addToBean } from "./store/BeanSlice";

import { useEffect, useState } from "react";

import "./store/GoodsSlice";

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

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

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

import { reloadGoods } from "./store/chosenGoodsSlice";

import { getGoods } from "./Selectors/getGoods";

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

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

import { auth } from "./firebase";

const paginationPerPage = 12;

export default function Catalog({

request,

priceValue,

brendValue,

categoryValue,

}) {

const goods = useSelector(getGoods);

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

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

const [currPage, setCurrPage] = useState(1);

let lastItemIndex = currPage \* paginationPerPage;

let firstItemIndex = lastItemIndex - paginationPerPage;

const changePage = (e, num) => {

setCurrPage(num);

window.scrollTo({ top: 0, behavior: "instant" });

};

const dispatch = useDispatch();

console.log(bean)

if (priceValue === undefined) priceValue = [1, 5000];

if (request === undefined) request = "";

const { brend, category } = useParams();

if (brend !== undefined) brendValue = brend;

if (category !== undefined) categoryValue = category;

useEffect(() => {

changePage(null, 1);

}, [request, priceValue, brendValue, categoryValue]);

useEffect(()=>{

if(!!authorized){

const oRef=ref(getDatabase(), `users/${auth.currentUser.uid}/basket`);

set(oRef, bean);

}

}, [bean])

const add = async (e, item) => {

if(!authorized){

alert("Необходимо войти в аккаунт!");

return;

}

let container = e.target;

try {

container.querySelector("path").classList.add("cls-2");

} catch {

e.target.classList.add("cls-2");

}

let flag = true;

bean.forEach((bItem) => {

if (bItem.id === item.id) {

flag = false;

}

});

if (flag)

dispatch(addToBean(item));

setTimeout(() => {

try {

container.querySelector("path").classList.remove("cls-2");

} catch {

e.target.classList.remove("cls-2");

}

}, 70);

console.log(bean)

console.log(auth.currentUser.uid)

// const oRef=ref(getDatabase(), `users/${auth.currentUser.uid}/basket`);

// await set(oRef, [...bean, item]);//temp

};

let chosenGoods = new Array();

goods.map((item) => {

if (

item.price >= priceValue[0] &&

item.price <= priceValue[1] &&

(item.category === categoryValue || categoryValue === "all") &&

(item.brend === brendValue || brendValue === "all") &&

(item.name.toLowerCase().includes(request.toLowerCase()) ||

!request ||

request === "")

)

chosenGoods.push(item);

});

const filteredGoods = chosenGoods;

chosenGoods = chosenGoods.slice(firstItemIndex, lastItemIndex);

dispatch(reloadGoods(chosenGoods));

const pagesCount = Math.ceil(filteredGoods.length / paginationPerPage);

return (

<div className="goodsContainer">

{chosenGoods.map((item) => (

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

<div className="absol">

{item.new && <div className="new">NEW!</div>}

{item.discount !== 0 && (

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

)}

</div>

<div className="description">

<Link to={`/productPage/${item.key}`}>

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

</Link>

</div>

<div>

<div className="description name">{item.name}</div>

<div className="description price">

{`${item.price \* (100 - item.discount) \* 0.01} руб`}

{item.discount !== 0 && (

<span className="crossed">{`${item.price} руб`}</span>

)}

{item.discount === 0 && (

<span className="disp0">{`${item.price} руб`}</span>

)}

</div>

</div>

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

<svg

id="Layer\_1"

data-name="Layer 1"

xmlns="http://www.w3.org/2000/svg"

viewBox="0 0 122.88 104.43"

width="25px"

>

<defs>

<style>{`.cls-1{fill-rule:evenodd; fill: black} .cls-2{fill: #a79191}`}</style>

</defs>

<title>add-to-cart</title>

<path

className="cls-1"

d="M97,0A25.9,25.9,0,1,1,78.67,7.59,25.79,25.79,0,0,1,97,0ZM3.66,10.89a3.71,3.71,0,0,1,0-7.42H9.11A17.3,17.3,0,0,1,18,5.81c4.92,3.12,5.79,7.57,7,12.59H66.7a31,31,0,0,0-.9,7.33H27.14L35.5,57.19H94.77l0-.18c.72.05,1.44.08,2.17.08a31.59,31.59,0,0,0,5.46-.48l-1.29,5.18a3.62,3.62,0,0,1-3.57,2.82H37.47c1.32,4.88,2.63,7.51,4.42,8.74,2.16,1.4,5.92,1.5,12.21,1.4H96.64a3.67,3.67,0,1,1,0,7.33H54.19c-7.79.09-12.58-.09-16.44-2.63s-6-7.14-8.07-15.31h0L17.09,16.52c0-.09,0-.09-.09-.19a6.51,6.51,0,0,0-2.82-4.22A9.51,9.51,0,0,0,9,10.89H3.66ZM60.87,33.47a2.6,2.6,0,0,1,5.11,0V47.63a2.6,2.6,0,0,1-5.11,0V33.47Zm-15.3,0a2.6,2.6,0,0,1,5.11,0V47.63a2.6,2.6,0,0,1-5.11,0V33.47ZM85.66,86.4a9,9,0,1,1-9,9,9,9,0,0,1,9-9Zm-39.55,0a9,9,0,1,1-9,9,9,9,0,0,1,9-9Zm64.08-62.91V28.3a2.09,2.09,0,0,1-2.07,2.07h-6.66V37a2.08,2.08,0,0,1-2.07,2.07H94.58A2.07,2.07,0,0,1,92.51,37V30.37H85.85a2.08,2.08,0,0,1-2.07-2.07V23.49a2.07,2.07,0,0,1,2.07-2.07h6.66V14.76a2.07,2.07,0,0,1,2.07-2.07h4.81a2.08,2.08,0,0,1,2.07,2.07v6.66h6.66a2.08,2.08,0,0,1,2.07,2.07Z"

></path>

</svg>

</div>

</div>

))}

{chosenGoods.length === 0 && (

<div className="notFound">Ничего не найдено!</div>

)}

{(filteredGoods.length>=12 &&

<Pagination

count={pagesCount}

defaultPage={1}

boundaryCount={2}

onChange={changePage}

></Pagination>

)}

</div>

);

}

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

import { useState, useEffect } from "react";

import { changeBean, deleteFromBean } from "./store/BeanSlice";

import "./bean.css";

import Checkbox from '@mui/material/Checkbox';

import AuthorizationForm from "./AuthorizationForm";

import { setShowLog } from "./store/logMenuSlice";

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

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

import { auth } from "./firebase";

import { useUserStatus } from "./hooks/hooks";

export default function Bean() {

useUserStatus();

const dispatch = useDispatch();

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

let readyToOrder=bean.filter((item)=>{return item.addToOrder===true});

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

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

useEffect(() => {

function handleResize() {

setWindowWidth(window.innerWidth);

}

window.addEventListener("resize", handleResize);

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

}, []);

// useEffect(() => {

// const handleBeforeUnload = (e) => {//это нужно для того чтобы состояние корзины не очищалось перед перезагрузкой из-за всплытия событий

// e.preventDefault();

// };

// window.addEventListener('beforeunload', handleBeforeUnload);

// return () => {

// window.removeEventListener('beforeunload', handleBeforeUnload);

// };

// }, []);

useEffect(()=>{

if(!!authorized){

const oRef=ref(getDatabase(), `users/${auth.currentUser.uid}/basket`);

set(oRef, bean);

}

}, [bean])

function changeBeanByCountToBuy(item, dir = 1) {

const index = bean.findIndex((index) => index.id === item.id);

if (index !== -1) {

let tempCount = bean[index].countToBuy;

tempCount += dir;

dispatch(changeBean([index, tempCount, "countToBuy"]));

}

}

function changeBeanByAddToOrder(e, item) {

const index = bean.findIndex((index) => index.id === item.id);

if (index !== -1) {

let value;

e.target.checked?value=true:value=false;

dispatch(changeBean([index, value, "addToOrder"]));

}

}

return (

<div>

{!authorized && (

<div className="noAcc">

<div className="needAcc">Необходимо войти в аккаунт</div>

<div className="enter" onClick={()=>dispatch(setShowLog(true))}>Войти</div>

</div>

)}

{!!authorized && bean.length > 0 && (

<div className={ windowWidth < 850 ? "grid1 beanContainer" : "grid2 beanContainer"}>

{readyToOrder.length>0 && <div className="toOrder">

<Link to="/order"><div>К заказу</div></Link>

</div>}

{bean.map((item) => (

<div className="beanItemContainer" key={Date.now()+Math.floor(Math.random() \* 1000)}>

<div className="absol">

<div onClick={() => dispatch(deleteFromBean(item))}>

&#x2715;

</div>

</div>

<div className="description">

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

</div>

<div className="seccol smallinfo">

<div className="description nameBeanItem">

{item.name}

</div>

<div className="grid3 counter">

<div onClick={() => {item.countToBuy !== 0 ? changeBeanByCountToBuy(item, -1) : console.log(); }}>-</div>

<div>{item.countToBuy}</div>

<div onClick={() => changeBeanByCountToBuy(item)}>+</div>

</div>

<div className="beanItemPrice">{`${(item.countToBuy \*item.price \*(100 - item.discount) \* 0.01).toFixed(2)} руб`}</div>

</div>

<div className="thirdcol">

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

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

</div>

</div>

))}

</div>

)}

{!!authorized && bean.length === 0 && (

<div className="notFound empty">Корзина пуста!</div>

)}

</div>

);

}

import SliderCatalog from "./SliderCatalog";

import Footer from "./Footer";

import Header from "./Header";

import { getGoods } from "./Selectors/getGoods";

import { useSelector } from "react-redux";

export default function MainPage(){

const goods = useSelector(getGoods);

return(

<div style={{display: "grid", padding: "20px 0px"}}>

<h2 style={{justifySelf: "center"}}>Акции</h2>

<SliderCatalog filter={{prop: "discount", filter: true}} goods={goods}></SliderCatalog>

<h2 style={{justifySelf: "center"}}>Новинки</h2>

<SliderCatalog filter={{prop: "new", filter: true}} goods={goods}></SliderCatalog>

</div>

)

}

# ПРИЛОЖЕНИЕ Б

Листинг Компоненты MUI

export function MinimumDistanceSlider() {

////////////////////////////////

const value1 = useSelector((state) => state.pricefilt.priceFilter);

const dispatch = useDispatch();

const handleChange1 = (event, newValue, activeThumb) => {

if (!Array.isArray(newValue)) {

return;

}

if (activeThumb === 0) {

dispatch(

setFilter([Math.min(newValue[0], value1[1] - minDistance), value1[1]])

);

} else {

dispatch(

setFilter([value1[0], Math.max(newValue[1], value1[0] + minDistance)])

);

}

console.log("init" + value1);

};

return (

<Box

sx={{

width: 320,

backgroundColor: "rgb(253, 253, 253)",

boxShadow: "2px 3px 6px #a79191",

borderRadius: "5px",

paddingLeft: "10px",

paddingRight: "10px",

position: "absolute",

zIndex: "3",

justifySelf: "center",

}}

>

<Slider

getAriaLabel={() => "Minimum distance"}

value={value1}

onChange={handleChange1}

valueLabelDisplay="auto"

getAriaValueText={valuetext}

disableSwap

min={1}

max={5000}

style={{ color: "rgb(138, 211, 230)" }}

/>

</Box>

);

}

import { useState, useEffect, useRef } from "react";

import \* as React from 'react';

import Filters from "./Filters";

import "./Search.css";

import Paper from '@mui/material/Paper';

import InputBase from '@mui/material/InputBase';

import IconButton from '@mui/material/IconButton';

import SearchIcon from '@mui/icons-material/Search';

let text="";

function Search() {

const [filter, setFilter] = useState(""); //состояние запроса в строке поиска

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

useEffect(() => {

function handleResize() {

setWindowWidth(window.innerWidth);

}

window.addEventListener("resize", handleResize);

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

}, []);

const request=(evt, promise)=>{

evt.preventDefault();

// console.log(promise)

setFilter(promise);

}

return (

<div>

<div className="searchContainer">

<div className="forinput">

<Paper

variant="outlined"

square={false}

component="form"

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

width:windowWidth <670? "55vw": "35vw", backgroundColor: "rgb(194, 194, 194)",

borderRadius: 10, height: "55%"}}

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

>

<InputBase

sx={{ ml: 1, flex: 1 }}

placeholder="Искать"

//inputProps={{ 'aria-label': 'search google maps' }}

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

/>

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

<SearchIcon/>

</IconButton>

</Paper>

</div>

</div>

<Filters request={filter}></Filters >

</div>

);

}

export { Search };

# ПРИЛОЖЕНИЕ В

Листинг Формы на React Hook Form

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

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

import { useState, useEffect } from "react";

import {

getAuth,

createUserWithEmailAndPassword,

updateProfile,

updatePhoneNumber,

PhoneAuthProvider,

signInWithEmailAndPassword,

onAuthStateChanged,

signOut

} from "firebase/auth";

import firebaseApp from "./firebase";

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

import { setAuthorized } from "./store/AuthorizedSlice";

import { useUserStatus } from "./hooks/hooks";

import { setShowLog } from "./store/logMenuSlice";

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

export async function userRegister(email, password, phone, username) {

return async (dispatch) => {//использование redux thunk

const auth = getAuth();

try {

const userCredential = await createUserWithEmailAndPassword(

auth,

email,

password

);

const user = userCredential.user;

await updateProfile(user, {

displayName: username,

phoneNumber: phone,

});

const userRef=ref(getDatabase(), `/users/${user.uid}`);

await set(userRef, {username, email, phone, basket: []});

dispatch(setAuthorized(userCredential.user))

} catch (error) {

const errorCode = error.code;

const errorMessage = error.message;

console.log({ errorCode, errorMessage });

}

};

}

export default function AuthorizationForm(props) {

const dispatch = useDispatch();

const [logging, setLogging] = useState(true);

const [regging, setRegging] = useState(false);

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

useUserStatus();//кастомный хук

const showPass = () => {

let pass = document.getElementsByName("password")[0];

if (pass.getAttribute("type") === "text") {

pass.setAttribute("type", "password");

} else {

pass.setAttribute("type", "text");

}

};

const {

register,

handleSubmit,

reset,

formState: { errors, isValid },

} = useForm({ mode: "onBlur" });

function close() {

reset();

dispatch(setShowLog(false));

}

async function submit(data) {

if (!isValid) return;

if (!data.phone) {

close();

} else {

close();

const result = dispatch(

await userRegister(

data.email,

data.password,

data.phone,

data.username

)

);

}

}

function logIn(data) {

if (!isValid) return;

const auth = getAuth();

signInWithEmailAndPassword(auth, data.email, data.password)

.then((userCredential) => {

const user = userCredential.user;

console.log(user);

dispatch(setAuthorized(user));

})

.catch((error) => {

const errorCode = error.code;

const errorMessage = error.message;

console.log(error);

});

close();

}

async function logout(){

await signOut(getAuth());

close();

}

return (

<div className={styles.abs}>

{regging && !authorized && (

<form className={styles.autForm} onSubmit={handleSubmit(submit)}>

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

&#x2715;

</div>

<div className={styles.forInput}>

<input

className={styles.input}

{...register("username", { required: "Заполните поле!" })}

style={{

border: errors.username

? "solid red 1px"

: "1px solid rgb(138, 211, 230)",

}}

placeholder="Логин"

></input>

<div className={styles.err}>{errors?.username?.message}</div>

</div>

<div className={styles.forInput}>

<input

className={styles.input}

{...register("phone", {

required: "Заполните поле!",

pattern: {

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

message: "Введите корректный номер",

},

})}

style={{

border: errors.phone

? "solid red 1px"

: "1px solid rgb(138, 211, 230)",

}}

placeholder="Номер телефона"

></input>

<div className={styles.err}>{errors?.phone?.message}</div>

</div>

<div className={styles.forInput}>

<input

className={styles.input}

{...register("email", {

required: "Заполните поле!",

pattern: {

value:

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

message: "Введите корректную почту!",

},

})}

style={{

border: errors.email

? "solid red 1px"

: "1px solid rgb(138, 211, 230)",

}}

placeholder="Эл. почта"

></input>

<div className={styles.err}>{errors?.email?.message}</div>

</div>

<div className={styles.forInput}>

<input

type="password"

className={styles.input}

{...register("password", {

required: "Заполните поле!",

pattern: {

value:

/^(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])[A-Za-z\d@$!%\*?&]{8,}$/,

message:

"Пароль должен содежать заглавные буквы, цифры и спец. символы!",

},

})}

style={{

border: errors.password

? "solid red 1px"

: "1px solid rgb(138, 211, 230)",

}}

placeholder={

errors.password ? "Обязательно к заполнению" : "Пароль"

}

></input>

<div className={styles.err}>{errors?.password?.message}</div>

</div>

<div className={styles.eyeContainer}>

<svg

fill="gray"

width="20px"

height="20px"

viewBox="0 0 20 11"

xmlns="http://www.w3.org/2000/svg"

style={{ paddingTop: "134px" }}

className={styles.eye}

onClick={() => showPass()}>

<path d="M10 4.4C3.439 4.4 0 9.232 0 10c0 .766 3.439 5.6 10 5.6 6.56 0 10-4.834 10-5.6 0-.768-3.44-5.6-10-5.6zm0 9.907c-2.455 0-4.445-1.928-4.445-4.307S7.545 5.691 10 5.691s4.444 1.93 4.444 4.309-1.989 4.307-4.444 4.307zM10 10c-.407-.447.663-2.154 0-2.154-1.228 0-2.223.965-2.223 2.154s.995 2.154 2.223 2.154c1.227 0 2.223-.965 2.223-2.154 0-.547-1.877.379-2.223 0z" />

</svg>

</div>

<div className={styles.forInput}>

<input type="Submit" className={styles.submit}></input>

</div>

</form>

)}

{logging && !authorized && (

<form className={styles.logForm} onSubmit={handleSubmit(logIn)}>

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

&#x2715;

</div>

<div className={styles.forInput}>

<input

className={styles.input}

{...register("email", {

required: "Заполните поле!",

pattern: {

value:

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

message: "Введите корректную почту!",

},

})}

style={{

border: errors.email

? "solid red 1px"

: "1px solid rgb(138, 211, 230)",

}}

placeholder="Эл. почта"

></input>

<div className={styles.err}>{errors?.email?.message}</div>

</div>

<div className={styles.eyeContainer}>

<svg

fill="gray"

width="20px"

height="20px"

viewBox="0 0 20 11"

xmlns="http://www.w3.org/2000/svg"

className={styles.eye}

onClick={() => showPass()}>

<path d="M10 4.4C3.439 4.4 0 9.232 0 10c0 .766 3.439 5.6 10 5.6 6.56 0 10-4.834 10-5.6 0-.768-3.44-5.6-10-5.6zm0 9.907c-2.455 0-4.445-1.928-4.445-4.307S7.545 5.691 10 5.691s4.444 1.93 4.444 4.309-1.989 4.307-4.444 4.307zM10 10c-.407-.447.663-2.154 0-2.154-1.228 0-2.223.965-2.223 2.154s.995 2.154 2.223 2.154c1.227 0 2.223-.965 2.223-2.154 0-.547-1.877.379-2.223 0z" />

</svg>

</div>

<div className={styles.forInput}>

<input

type="password"

className={styles.input}

{...register("password", {

required: "Заполните поле!",

pattern: {

value:

/^(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])[A-Za-z\d@$!%\*?&]{8,}$/,

message:

"Пароль должен содежать заглавные буквы, цифры и спец. символы!",

},

})}

style={{

border: errors.password

? "solid red 1px"

: "1px solid rgb(138, 211, 230)",

}}

placeholder={

errors.password ? "Обязательно к заполнению" : "Пароль"

}

></input>

<div className={styles.err}>{errors?.password?.message}</div>

</div>

<div className={styles.forInput}>

<input

type="Submit"

className={styles.submit}

value={"Войти"}

></input>

</div>

<div className={styles.forLinks}>

<div className={styles.link} style={{ paddingRight: "30px" }}>

Забыли пароль?

</div>

<div

className={styles.link}

onClick={() => {

reset();

setLogging(false);

// dispatch(setShowLog(false))

setRegging(true);

}}

>

Зарегистрироваться

</div>

</div>

</form>

)}

{!!authorized && <div className={styles.quit} style={{backgroundColor: "black"}}>

<div className={styles.absol} onClick={() => close()}>&#x2715;</div>

<div className={styles.hello}>

<img className={styles.userPic} src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcS6mklexP7IGYmTrZKIdXlhd9D6PxkRV2PQvnfTYkJE82kWRam-89TfZ0mx1I\_U3JCVPgQ&usqp=CAU"></img>

</div>

<div className={styles.hello}>{`Здравствуйте, ${authorized.displayName}!`}</div>

<div className={styles.leave} onClick={()=>logout()}>Выйти</div>

</div>}

</div>

);}