### Частное учреждение дополнительного образования

**«НОВОСИБИРСКАЯ АКАДЕМИЯ ДИЗАЙНА И ПРОГРАММИРОВАНИЯ»**

Тема «Разработка компьютерного интернет-магазина с использованием языка программирования JavaScript и библиотек React и Redux»

Выполнил:

Ковалец Пётр Алексеевич

Руководитель дипломного проекта:

Елизарьев Никита Викторович

Красноярск, 2022

# Оглавление

1. [Технические требования 3](#_Toc9465014)
2. [Основание для разработки 3](#_Toc9465015)
3. [Назначение разработки 3](#_Toc9465016)
4. [Введение 4](#_Toc9465017)
5. [История пакета 4](#_Toc9465018)
6. [Моделирование 4](#_Toc9465019)
7. [Список модулей визуализации 6](#_Toc9465020)
8. [Плагины 7](#_Toc9465021)
9. [Цель 8](#_Toc9465022)
10. [Ход работы 8](#_Toc9465023)
11. [Моделирование шахматного коня 15](#_Toc9465024)
12. [Настройка материалов 18](#_Toc9465025)
13. [Заключение 19](#_Toc9465026)
14. [Список используемой литературы 31](#_Toc9465027)

# Технические требования

1. Компьютер;
2. Любой редактор кода;
3. Установленная Node.js;
4. Графическая операционная система;

# Основание для разработки

Основания для проведения разработки: приказ о выполнение дипломного проекта от 1 Февраля 2022 года.

Исполнитель: Ковалец Пётр Алексеевич

# Назначение разработки

Разработка интернет-магазина для работы в портфолио.

Улучшить знание в языке программирования JavaScript и библиотеках React и Redux.

# Введение

**React —** JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов. React может использоваться для разработки одностраничных и мобильных приложений.

# Цель

Передо мной была поставлена цель разработать интернет-магазин, используя полученные знания.

# Ход работы

Для выполнения поставленной задачи, я использовал программу Visual Studio Code.

1. Установка программы Node.js.
2. Установка программы Git.
3. Установка редактора кода Visual Studio Code.
4. Создание базы данных товаров в сервисе Firebase от Google
5. Запуск сервера
6. Разработка
7. Деплой на хостинг Github Pages

# Разработка

Начнем с установки библиотеки React в папке проекта:

1. Переходим в папку проекта с помощью любого терминала (на скриншоте изображён терминал Git Bash, который устанавливается вместе с Git, вместо него можно выбрать любой другой терминал).
2. Устанавливаем библиотеку React используя команду «npx create-react-app».

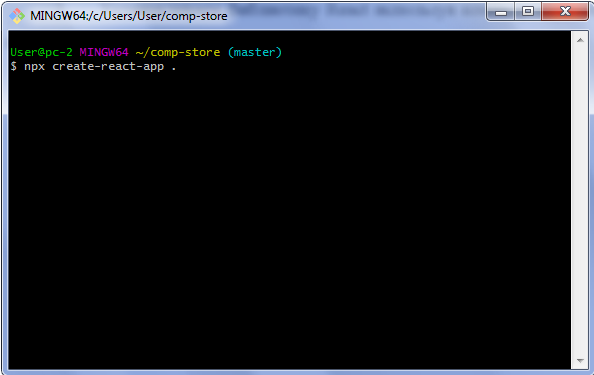


Рисунок 1 – установка библиотеки React.

Данная команда устанавливает все пакеты библиотеки React и собирает проект, npx нужен для того, чтобы не устанавливать глобально пакет create-react-app и потому что этот способ создания нового проекта часто используется.

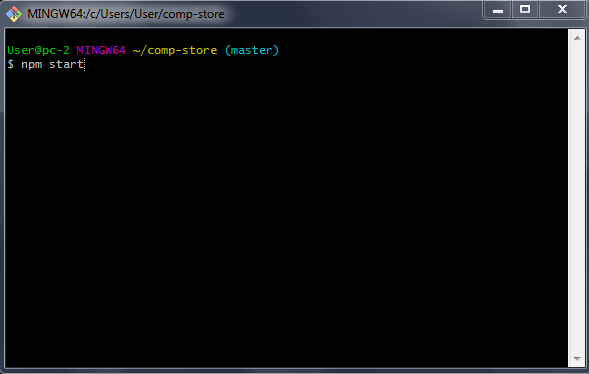


Рисунок 2 - запуск приложения.

Эта команда запустит локальный сервер на порту, который по умолчания равен 3000 и откроет страницу в браузере с проектом. Чтобы открыть сайт который запущен на локальном сервере надо перейти по ссылке <http://localhost:3000>.

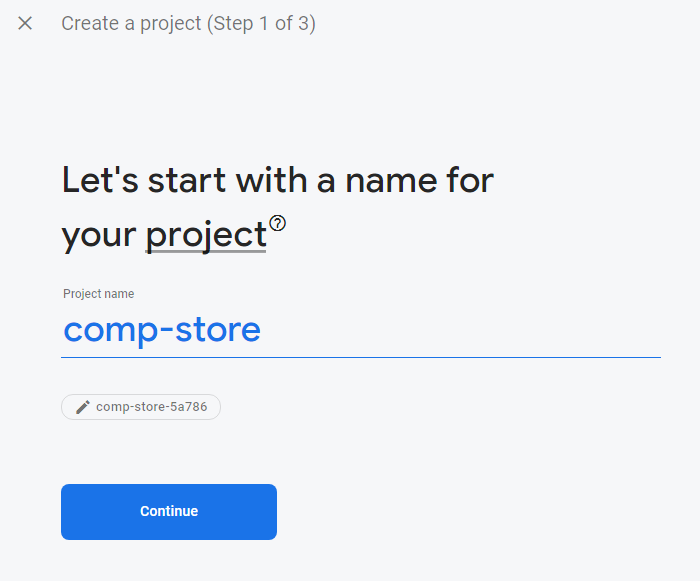


Рисунок 3 - создание проекта в Firebase.

Создадим новый проект в сервисе от Google Firebase. Firebase нужен для авторизации пользователей и создания базы данных. После создания проекта в Firebase мы должны его зарегистрировать и добавить Firebase в наш проект.

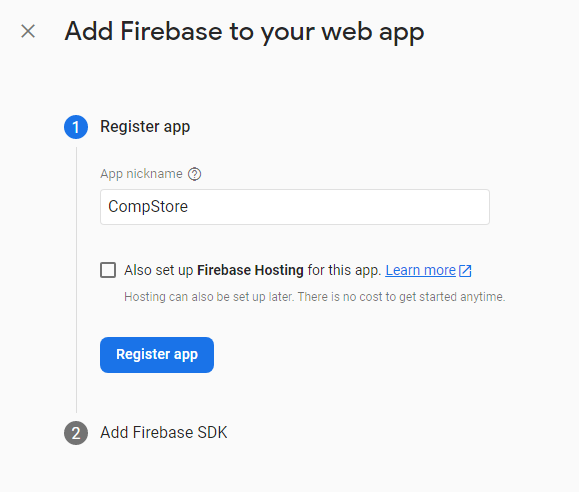


Рисунок 4 - регистрация приложения.

После регистрации проекта в Firebase, чтобы установить пакет Firebase надо выполнить в терминале команду «npm i firebase react-firebase-hooks». Эта команда установит Firebase и пакет react-firebase-hooks (react-firebase-hooks нужен для того, чтобы легче работать с Firebase в React проекте). После установки этих пакетов создадим файл firebase.js в папке src и добавим туда данные, которые были в браузере после регистрации проекта в firebase.

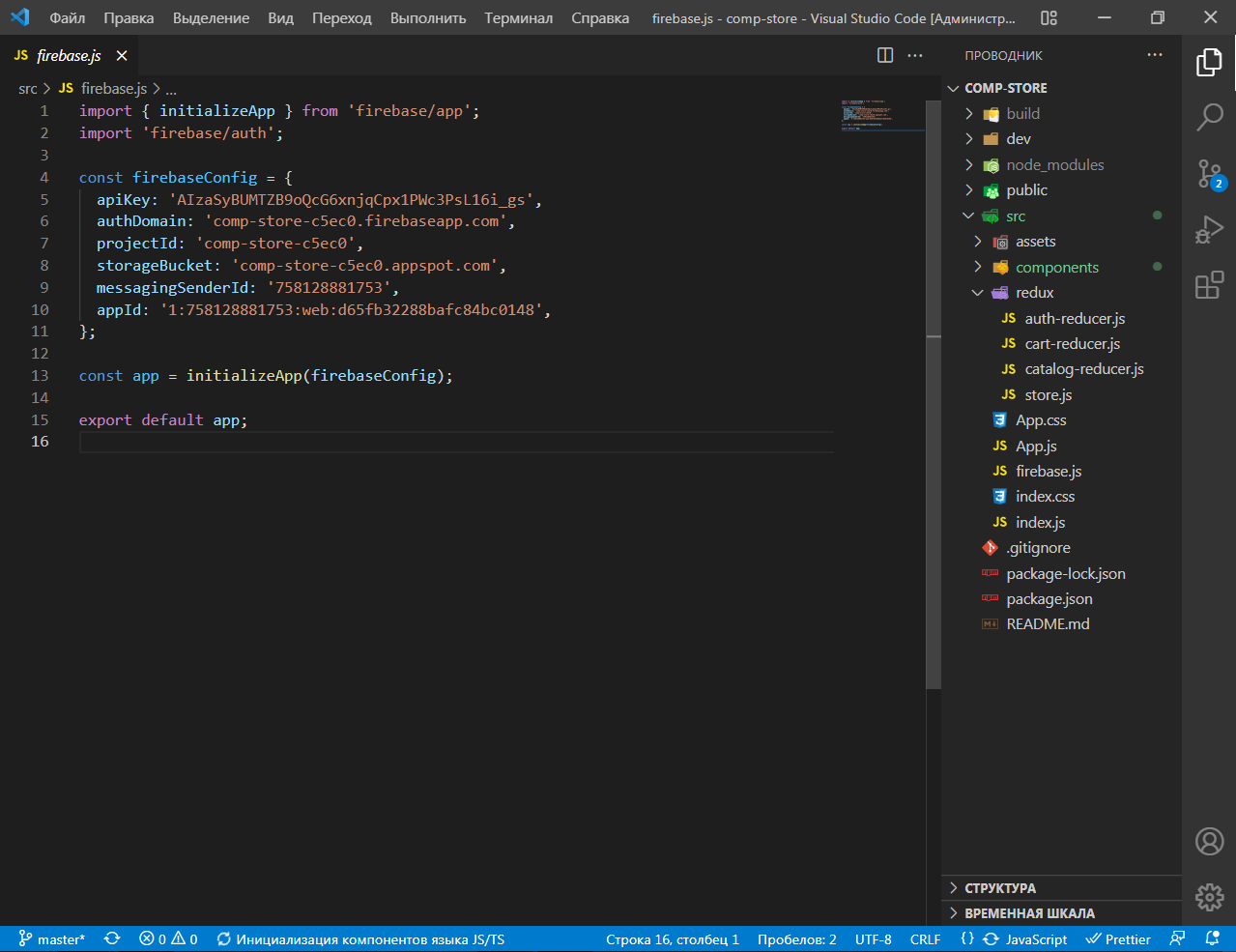


Рисунок 5 – добавление настроек Firebase в проект и экспорт функции initializeApp с теми настройками проекта из файла firebase.js.

После добавления в Firebase в проект добавим Firestore Database в наше Firebase приложение.

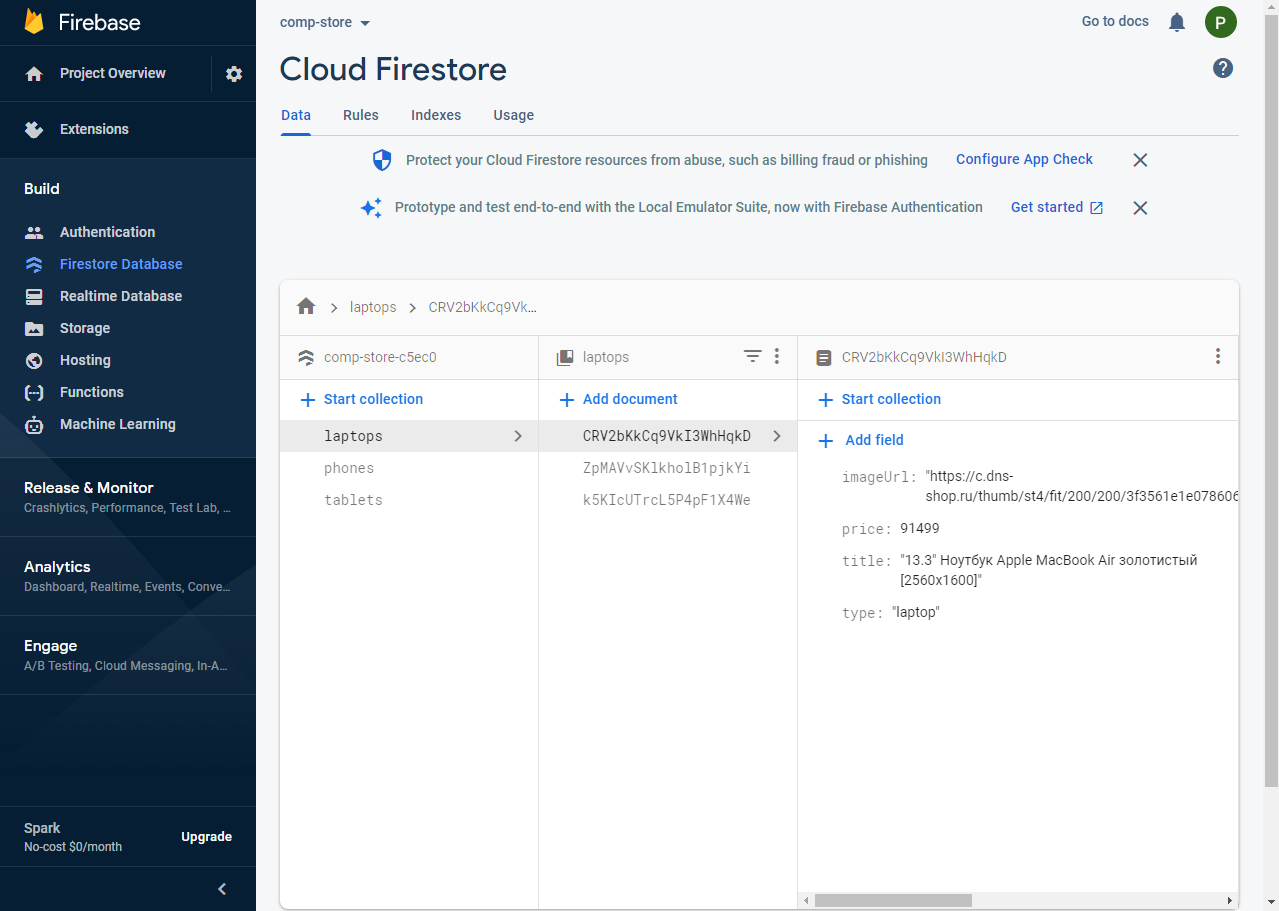


Рисунок 6 – База данных Firestore Database.

Теперь добавим маршрутизацию на наш сайт, установив два пакета и обернув компонент App компонентом HashRouter в файле index.js (HashRouter использовался для того чтобы проект нормально работал на хостинге Github Pages, потому что в проекте нет Server Side Rendering, вместо него используется Client Side Rendering, Github Pages не поддерживает СSR, если использовать хостинг, который поддерживает SSR можно было бы использовать компонент BrowserRouter).

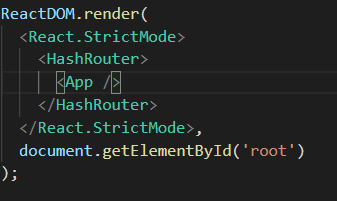


Рисунок – 7 оборачиваем компонент App компонентом HashRouter.

После добавления маршрутизации в наш проект можно создать шапку сайта с навигацией.

Создадим папку с названием Header в папке components, которая находится в src. В папке Header создадим файлы Header.js и Header.module.css. В файле Header.js находится компонент, который экспортируется из файл. Файл Header.module.css нужен для стилей компонента.

После создания файлов создадим функцию Header в файле Header.js и экспортируем её. В этом компоненте используется авторизация с помощью Google, Firebase и Redux и маршрутизация из пакета react-router-dom. В файле Header.module.css находятся стили, которые будут доступны только для компонента Header.

Если пользователь авторизован (компонент проверяет Redux данные из файла auth.js, в котором находится флаг isAuth) компонент покажет ссылку на корзину, если не авторизован на страницу авторизации (авторизация в проекте работает только с помощью Google).

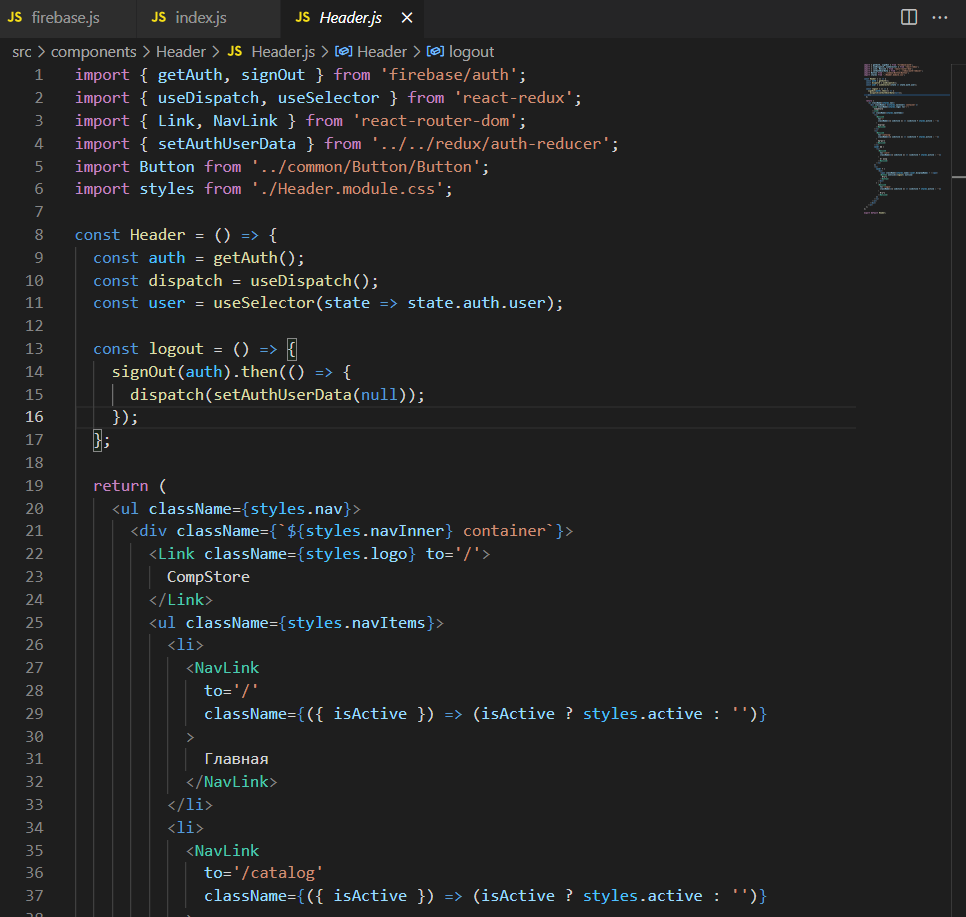


Рисунок 8 – компонент Header 1.



Рисунок 9 – компонент Header 2.

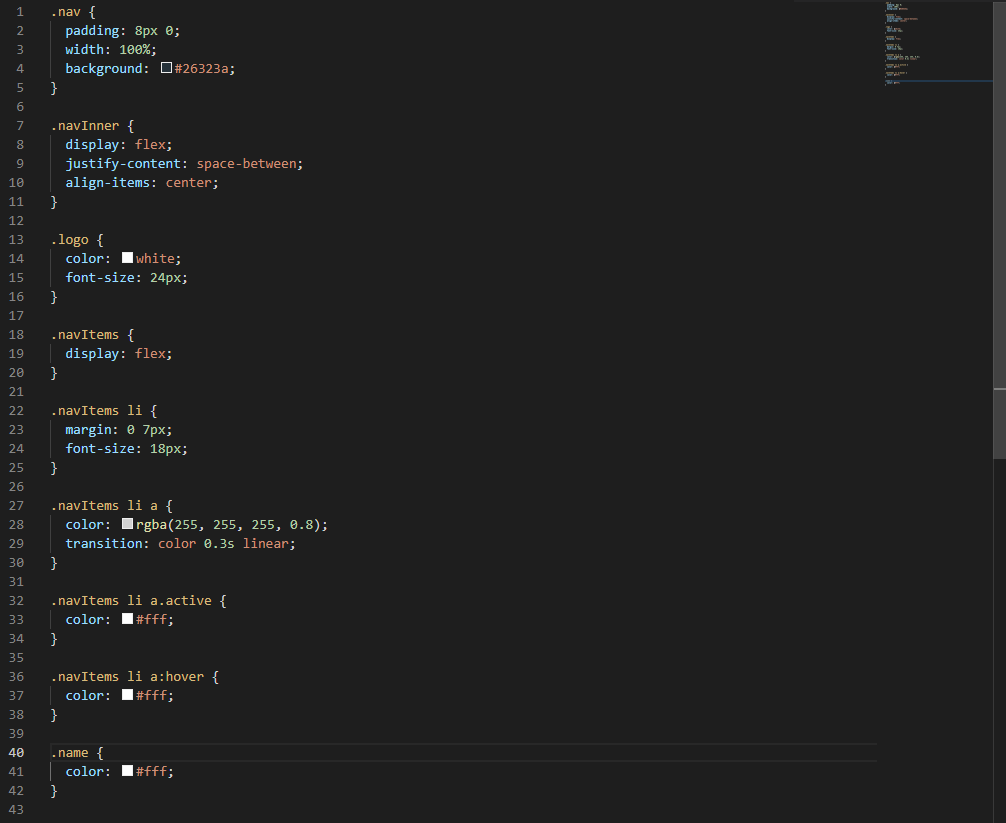
Рисунок 10 – компонент Header 3.

Рисунок 11 – стили для компонента Header.

Теперь можно создать store (глобальное хранилище данных) с помощью библиотеки Redux. Установим библиотеку Redux используя команду «npm i redux react-redux» (react-redux нужен, чтобы проще работать с Redux в React проекте). После установки надо создать папку redux в папке src и создать файл store.js, в котором будет наше глобальное хранилище данных. Создадим store используя функцию createStore, импортированную из redux и экспортируем store, чтобы импортировать его в файле index.js и обернуть компонент App компонентом Provider.

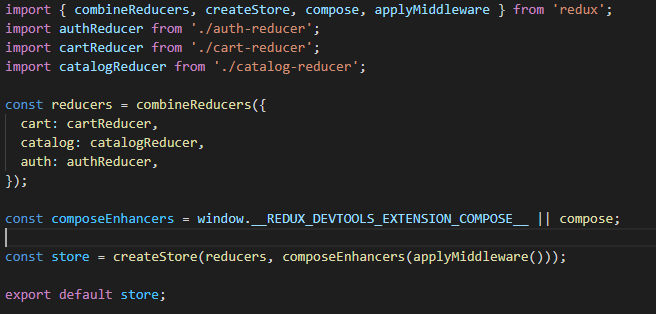


Рисунок 12 – создаём store.

В store’е также находятся reducer’ы, которые объединены в одно общее хранилище данных. Store (глобальное хранилище данных) состоит из частей этого хранилища, которые называются reducer’ами, они находятся в папке redux и называются имя-reducer. Reducer – это чистая функция которая принимает state и возвращает новый state.

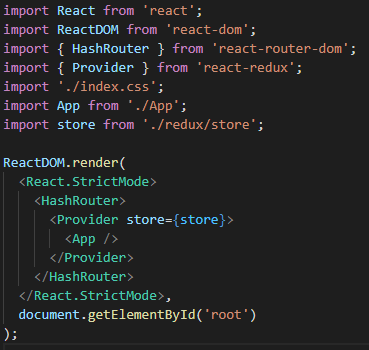


Рисунок 13 – оборачиваем компонент App.

После того как мы обернули компонент приложения нашим хранилищем, можно создать общие компоненты (компоненты которые будут использоваться в других компонентах). Создадим папку common в папке components и создадим два компонента с названиями Button (кнопка) и Preloader (картинка в формате gif, которая показывает, что сейчас идёт загрузка).

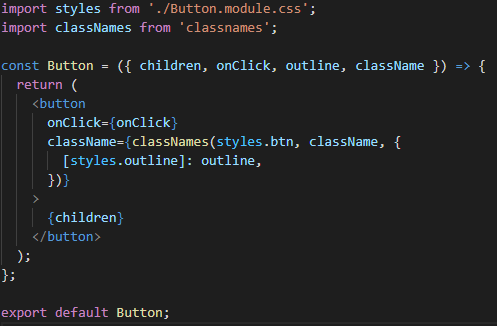
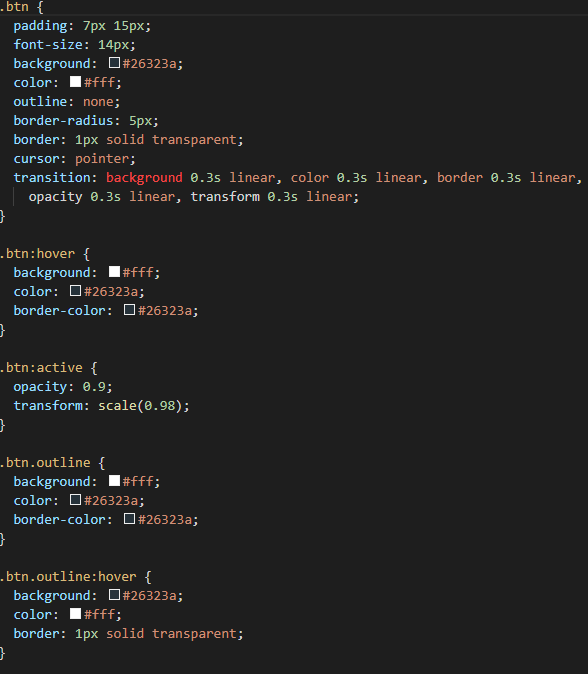


Рисунок 14 – компонент Button.

В компоненте Button мы получаем содержимое, которое передали в компонент, функцию, которая должна выполниться после нажатия, флаг outline (который нужен для того, чтобы узнать будет ли у кнопки outline стиль) и классы, которые задаются другими компонентами для задания локальных стилей компонента кнопки.

Рисунок 15 – стили для компонента Button.

Для компонента Preloader нужно создать папку assets в src и создать папку images, где будет находится изображение загрузки в формате gif.

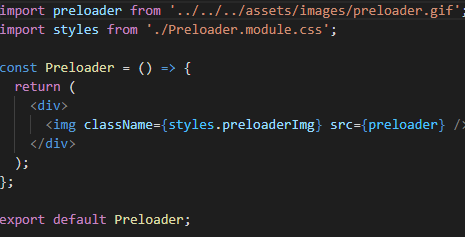


Рисунок 16 – компонент Preloader.

Для preloader’a нужен будет отступ сверху потому что он должен всегда находится в центре страницы. Preloader должен просто показывать картинку загрузки.



Рисунок 17 – стили для компонента Preloader.

Теперь можно создавать страницу каталог. Создадим reducer для каталога в папке redux и назовём его catalog-reducer.

у

Рисунок 18 – catalog-reducer.

У этого reducer’а есть одна функция, выбрать активную категорию. После создания reducer’а для хранения данных о выбранной категории, можно создавать компонент Catalog. Создадим в папке Catalog компонент CategoryPopup, который нужен будет для выбора категории. Сначала создадим массив объектов категорий в файле Catalog.js и передадим его в компонент CategoryPopup, потому что массив это ссылочный тип и если создать массив в CategoryPopup, будет происходить лишняя перерисовка компонента.

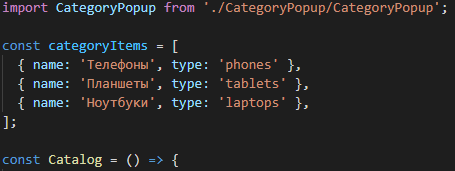


Рисунок 19 – массив объектов категорий.

Теперь создадим компонент для выбора категории CategoryPopup.



Рисунок 20 – компонент CategoryPopup 1.

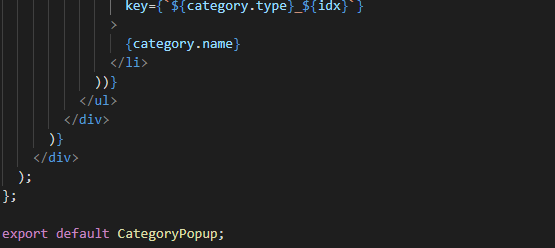


Рисунок 21 – компонент CategoryPopup 2.

Компонент CategoryPopup будет открывать меню при нажатии на компонент кнопки, который находится в нём, закрывать при повторном нажатии и закрывать при выборе категории или при нажатии на другой элемент страницы. Теперь напишем стили для компонента CategoryPopup.

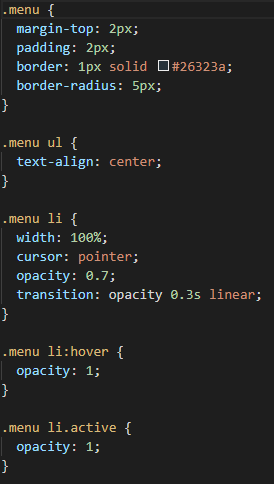
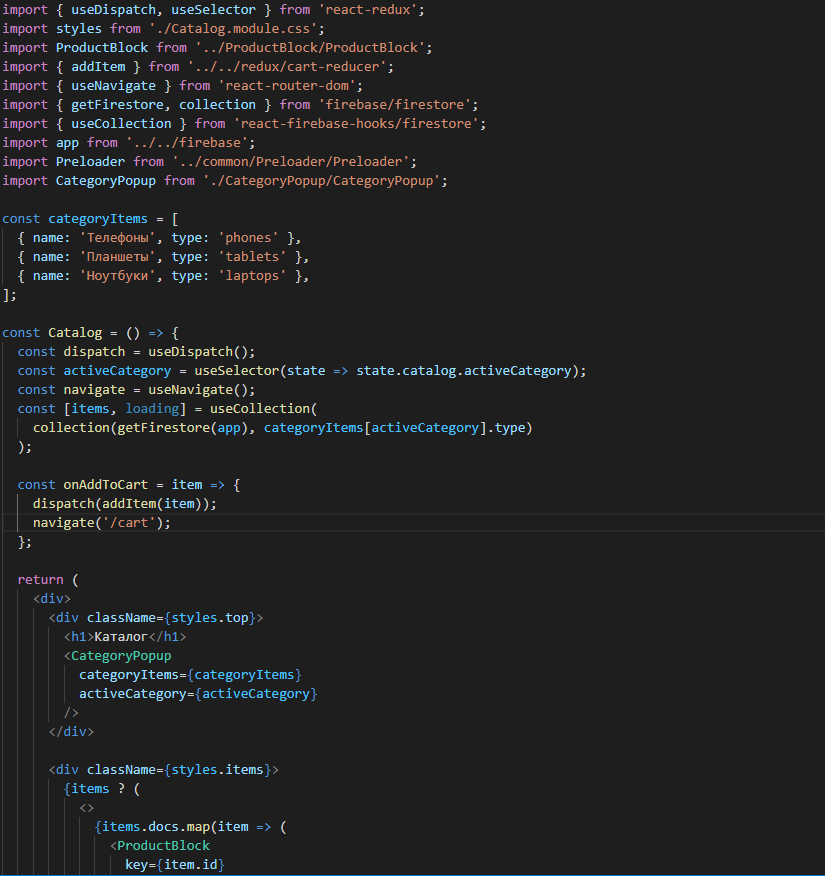
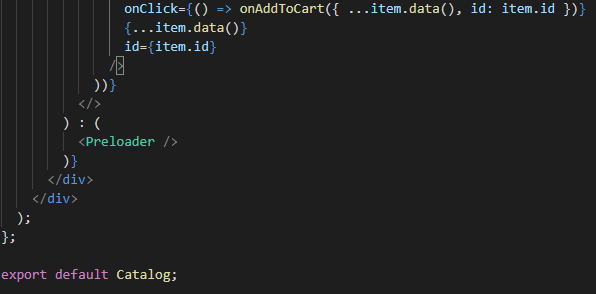


Рисунок 22 – стили для компонента CategoryPopup.

Теперь создадим компонент Catalog. При запросе на localhost:3000/catalog будет отображаться страница каталога. Компонент Catalog будет отображать товары из базы данных по выбранной категории с помощью компонента CategoryPopup. Товары из базы данных отображаются в каталоге по выбранной категории. CategoryPopup нужен только для того, чтобы менять категорию.

Рисунок 23 – компонент Catalog 1.

Рисунок 24 – компонент Catalog 2.

Напишем стили для страницы каталога в файле Catalog.module.css.

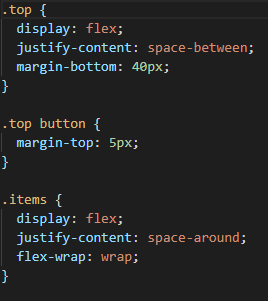


Рисунок 25 – стили для компонента Catalog.

Чтобы добавить компонент как страницу нужен компонент AppRouter, но для этого нужна авторизация. Создадим файл auth-reducer в папке Redux. В файле auth-reducer будет находится информация о пользователе. Когда страница будет загружаться, главный компонент App будет получать данные о пользователе с сервера и добавлять эти данные в объект user, который находится в auth-reducer. При авторизации флаг isAuth будет меняться на true, при выходе на isAuth будет false.



Рисунок 26 – auth-reducer.

После создания auth-reducer’a, можно создать компонент, в котором будут доступные и недоступные для пользователя страницы без авторизации. При заходе на страницу, которая не доступна пользователю с авторизацией или без авторизации, его будет перенаправлять на первую доступную страницу, которая будет указана в компоненте.



Рисунок 27 – компонент AppRouter.

Добавим его в компонент App и добавим вход в аккаунт, когда страница загрузится в функции useEffect компонента App.

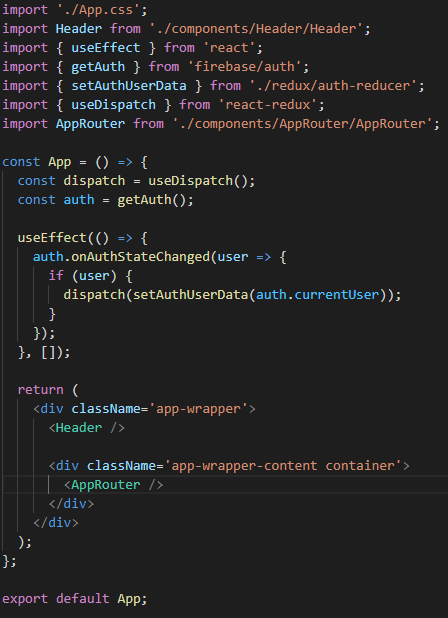


Рисунок 28 – компонент App.

Теперь добавим страницу корзины. Создадим файл cart-reducer. В этом файле будет находится информация о товарах в корзине и общей стоимости товаров в корзине. На страницу корзины нельзя будет зайти не авторизованному пользователю. С авторизацией пользователь на сможет зайти на страницу логина.



Рисунок 29 – cart-reducer 1.

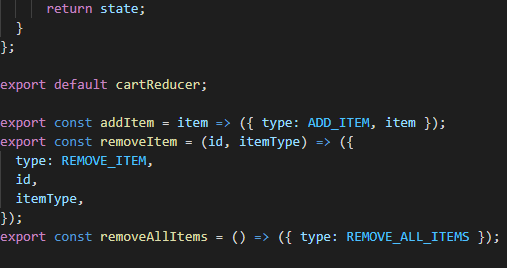


Рисунок 30 – cart-reducer 2.

Создадим страницу корзины, на которой будет отображаться таблица с заказами, суммой и кнопкой покупки товаров.



Рисунок 31 – компонент Cart 1.

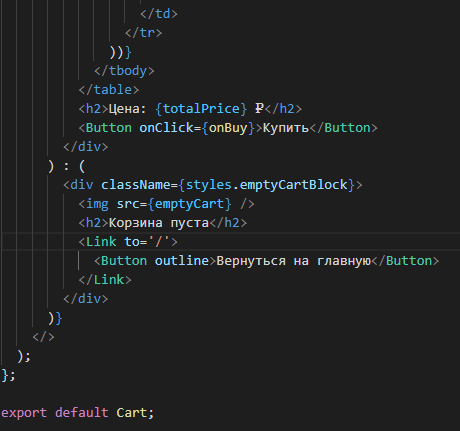


Рисунок 32 – компонент Cart 2.

Компонент корзины будет работать с таблицей заказов, он собирает данные из Redux хранилища данных в reducer’е cart и рисует каждый элемент из массива items как строку таблицы, в которой находятся информация о товаре, его изображение и цена на данный товар. Каждый товар в корзине можно убрать из корзины. После этой таблицы выводятся общая стоимость товаров в корзине, кнопка при нажатии на которую будет происходить покупка товаров (в данном проекте не будет оплаты и будет происходить очистка корзины товаров). Если товаров в корзине нет, будет отображаться картинка пустой корзины и ссылка на страницу каталога.

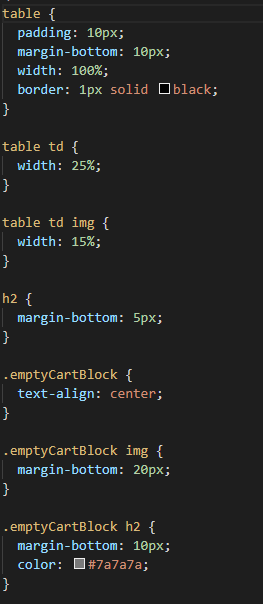


Рисунок 33 – стили для компонента Cart.

Теперь создадим главную страницу, на которой должен быть слайдер со ссылками на другие страницы сайта и информация о корзине пользователя (если пользователь не авторизован будет перенаправление на страницу авторизации). Для слайдера был выбран плагин react-responsive-carousel, который добавляет адаптивный компонент слайдера в проект. Компонент главной страницы будет называться Home. Компонент Home будет показывать пользователю сколько товаров в корзине и названия товаров при загрузке компонента и будет отображать слайдер.

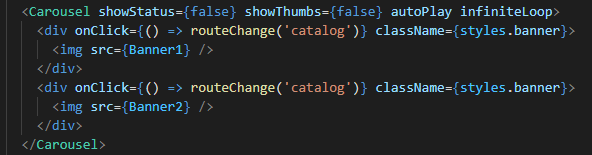


Рисунок 34 – слайдер.

Сайт готов, его можно загрузить на хостинг. Добавим все изменения в гит, используя команду «git add .», которая добавит все изменения в коммит. Зарегистрируем коммит, используя команду «git commit -m “Upload”». Добавим проект на GitHub. Для того, чтобы добавит проект на GitHub, надо зарегистрироваться, создать новый репозиторий и клонировать коммит, используя команды, которые будут показаны после создания репозитория. Добавим в проект GitHub Pages и выложим сайт на хостинг, используя ту документацию, которая будет указана на официальном сайте GitHub. Для того чтобы деплой работал без ошибок мы уже добавили в проект HashRouter вместо BrowserRouter.

# Заключение

В процессе обучения React, я решил создать интернет-магазин. Чтобы разработать интернет-магазин мне пришлось изучить базы данных сервиса от Google Firebase.

В итоге я создал интернет-магазин с работающей авторизацией с использованием Google, корзиной, каталогом с выбором категории, слайдером. Я научился использовать Redux вместе с Firebase Auth.