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 «Интернет-магазин»

**Исполнитель**

студент 2 курса 2 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н. А. Турчинович

подпись, дата

**Руководитель**

Доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н. А. Жиляк

(должность, уч. звание) (подпись, дата)

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

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

дата, подпись

Курсовой проект защищен с оценкой

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

подпись дата инициалы и фамилия

Минск 2023

**Содержание**

[Введение 5](#_Toc122376412)

[1 Постановка задач и обзор аналогичных решений 6](#_Toc122376413)

[1.1 Актуальность темы 6](#_Toc122376414)

[1.2 Обзор аналогичных решений 6](#_Toc122376415)

[1.3 Постановка задачи 9](#_Toc122376416)

[1.4 Выводы по разделу 10](#_Toc122376417)

[2 Проектирование веб-сайта 11](#_Toc122376418)

[2.1 Выбор средств реализации 11](#_Toc122376419)

[2.2 Проектирование uml-диаграммы 12](#_Toc122376420)

[2.3 Проектирование страниц 13](#_Toc122376422)

[2.4 Выбор стилевого оформления 14](#_Toc122376423)

[2.5 Разработка пользовательских элементов 14](#_Toc122376424)

[2.6 Выводы по разделу 17](#_Toc122376425)

[3 Реализация структуры веб-сайты 18](#_Toc122376426)

[3.1 Структура JSX-документа 18](#_Toc122376427)

[3.2 Структура Redux-Toolkit 23](#_Toc122376428)

[3.3 Добавление таблиц стилей SCSS 25](#_Toc122376431)

[3.4 Выводы по разделу 25](#_Toc122376432)

[4 Тестирование веб-сайта 26](#_Toc122376433)

[4.1 Адаптивный дизайн веб-сайта 26](#_Toc122376434)

[4.2 Кроссбраузерность веб-сайта 27](#_Toc122376435)

[4.3 Ручное тестирование 29](#_Toc122376436)

[4.4 Выводы по разделу 30](#_Toc122376437)

[5 Руководство пользователю 31](#_Toc122376438)

[Заключение 32](#_Toc122376439)

[Список использованных источников 33](#_Toc122376440)

[Приложение А Макет структуры веб-сайта 34](#_Toc122376441)

[Приложение Б Листинг JSX-документа 36](#_Toc122376442)

[Приложение В Листинг (SASS) 38](#_Toc122376443)

[Приложение Г Листинг Redux 42](#_Toc122376444)

# Введение

В настоящее время активно растет количество новых веб-сайтов в сфере видеоигр. Поэтому основной задачей при разработке интернет-магазина видеоигр является создание привлекательной платформы для потенциальных игроков, придание ей уникальности и стиля. В отличие от ранних дней Интернета, когда достаточно было привлечь посетителей без больших усилий и материальных затрат, сегодня, в условиях усиливающейся конкуренции, ключевую роль играют функциональность, стиль и запоминаемость магазина.

Независимо от того, создается ли сайт для представления игровой компании или для заработка на рекламе, его основной целью остается привлечение максимальной аудитории. Поэтому важен правильный подход к выбору стратегии и исполнителей. Профессиональные веб-студии предлагают разработку сайтов разного типа, включая корпоративные, визитные, интернет-магазины и сайты для бизнеса как в Интернете, так и в реальном мире.

В наше время, называемое «информационным веком», информация является ключевым ресурсом. Быстрое и четкое распространение необходимой информации через Интернет – ключевой элемент успеха в бизнесе. Интернет, как средство передачи текста, звука, изображений и видео, становится основным информационным ресурсом, обеспечивая доступ к широкому спектру данных – от прогноза погоды до личных предпочтений игроков.

С учетом постоянного роста аудитории Интернета, пользователи становятся особенно привлекательными для рекламодателей, учитывая их потенциал как клиентов. В современном мире интернет-магазины видеоигр становятся все более актуальными, учитывая, что все больше людей предпочитают покупать игры онлайн из-за более низких цен и большего выбора, чем в традиционных магазинах.

Цель курсовой работы: разработать интернет-магазин с использованием React JS, а также с применением SCSS/CSS3.

Задачи курсовой работы:

* проанализировать и описать основные принципы работы React и Redux;
* разработать приложение, используя React и Redux;
* провести тестирование приложения;
* провести анализ и оценку качества приложения.

# 1 Постановка задач и обзор аналогичных решений

## Актуальность темы

Наблюдается повышенная актуальность темы в контексте интернет-магазина видеоигр, поскольку такие магазины становятся одними из наиболее посещаемых сайтов. Для покупателей важно обеспечить удобство и оперативность процесса покупок, в то время как для продавцов ключевым является удобство и эффективность добавления товаров в каталог, а также их управление.

На сегодняшний день рынок насыщен разнообразными интернет-магазинами, которые используют различные технологии и платформы для разработки своих веб-сайтов. Однако, как правило, многие из них выбирают один из самых востребованных фреймворков для создания веб-сайтов – React JS.

## 1.2 Обзор аналогичных решений

Интернет-магазин представляет собой совокупность взаимосвязанных веб-страниц, содержащих информацию, которую пользователь может просматривать в своем браузере. Эти страницы взаимодействуют между собой посредством гипертекстовых ссылок, обеспечивая навигацию по магазину.

Для эффективного создания динамичного интернет-магазина с минимальными временными и финансовыми затратами необходимо использовать современные технологии. Важно, чтобы веб-сайт был удобен для пользователей и адаптирован под мобильные устройства. Тематика магазина должна быть востребованной и актуальной. В связи с этим было решено разработать интернет-магазин видеоигр, который станет веб-сайтом, предоставляющим возможность приобретения различных игровых товаров.

Перед началом разработки веб-сайта необходимо провести анализ аналогов, чтобы выявить используемые технологии, а также выявить преимущества и недостатки других интернет-магазинов. Особое внимание уделяется поиску уникальных особенностей в каждом аналоге, которые сделают наш интернет-магазин уникальным по сравнению с другими. В этом контексте будут рассмотрены три аналога.

Аналог №1 – сайт Steam [1].

Steam является одной из ведущих платформ для цифровой дистрибуции видеоигр и программного обеспечения. Основным преимуществом этой платформы является ее удобство и простота использования. На главной странице Steam пользователь обнаружит разделы, предназначенные для легкого поиска нужных игр и программ. На рисунке 1.1 представлен интерфейс платформы. Товары также организованы по категориям, что обеспечивает быстрый доступ к необходимым продуктам. Дизайн сайта привлекателен, с читаемым шрифтом, динамичным и интерактивным интерфейсом. При наведении на изображения происходит анимация, придающая дополнительную динамичность пользовательскому опыту.

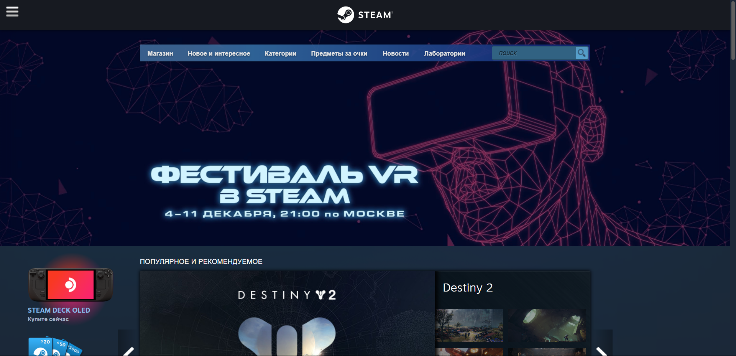


Рисунок 1.1 – Главная страница сайта Steam

Веб-сайт имеет адаптивный дизайн, который позволяет использовать сайт на мобильных устройствах, горизонтальное меню становится вертикальным, текст остаётся читабельным. Дизайн представлен на рисунке 1.2.

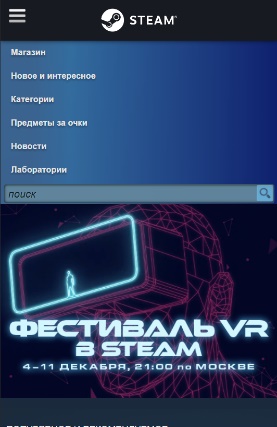


Рисунок 1.2 – Мобильная версия сайта Steam

Несмотря на преимущества, некоторые пользователи могут отмечать, что дизайн Steam иногда может быть сложным и перегруженным информацией, особенно при просмотре различных разделов и акций. Также некоторые пользователи высказывают замечания относительно неоднородности пользовательского интерфейса и ограниченных опций настройки визуального оформления.

Аналог №2 – сайт GOG [2].

GOG.com предоставляет широкий выбор видеоигр, начиная от классических до самых новых. Структура сайта обеспечивает легкость поиска, где пользователь может выбирать игры в различных разделах. Интерфейс сайта прост и интуитивно понятен, а функция поиска по названию и категориям обеспечивает быстроту поиска необходимого товара.

Каждая игра содержит подробное описание, а отзывы пользователей помогают сделать осознанный выбор. Привлекательный дизайн сайта, с яркими цветами и красочными фотографиями игр, представлен на рисунке 1.3. Уникальной особенностью является отображение ранее просмотренных игр, что облегчает пользователям быстрый доступ к интересующимся товарам.

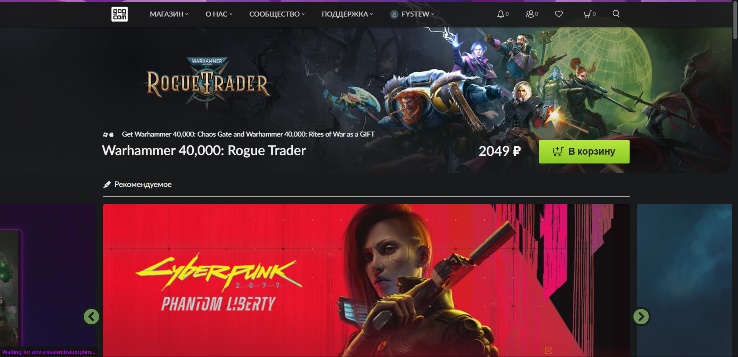


Рисунок 1.3 – Главная страница сайта GOG

Также сайт имеет адаптивный дизайн, который позволяет просматривать сайт на мобильных устройствах. У данного сайта нет никаких проблем с адаптивностью, на разных устройствах он выглядит привлекательно. На рисунке 1.4 представлено оформление на мобильных устройствах.

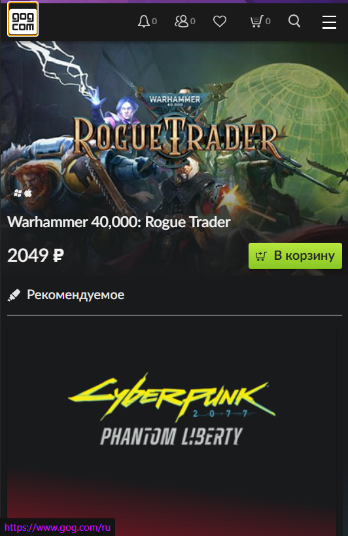


Рисунок 1.4 – Мобильная версия сайта GOG

Из недостатков сайта можно выделить слегка перегруженный интерфейс, особенно на мобильных устройствах.

Аналог №3 – сайт Epic Games Store [3].

Epic Games Store предлагает широкий ассортимент игр, включая как классические хиты, так и самые новые релизы. Сайт обеспечивает удобный поиск по различным категориям и ценовым диапазонам, а его дизайн прост и дружелюбен для пользователей. Яркие цвета не вызывают раздражения, а единообразный размер кнопок облегчает навигацию.

Однако, среди недостатков можно выделить иногда высокую загруженность сайта, что может затруднить опыт пользователя, а также не всегда легкость в поиске конкретных игр. Уникальной особенностью Epic Games Store является возможность быстрой регистрации через социальные сети. Оформление сайта Epic Games Store представлено на рисунке 1.5.

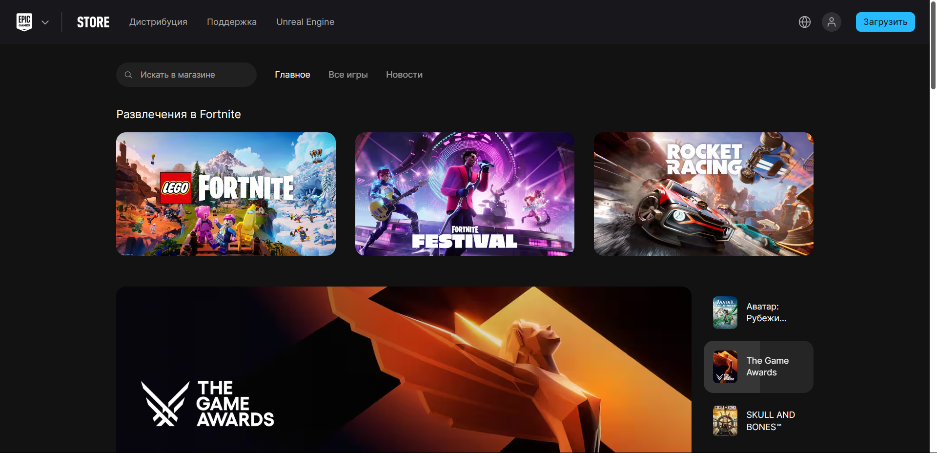


Рисунок 1.5 – Главная страница сайта Epic Games Store

Также данный сайт является кроссплатформенным и адаптирован под мобильные устройства. На рисунке 1.6 представлено оформление на мобильных устройствах.

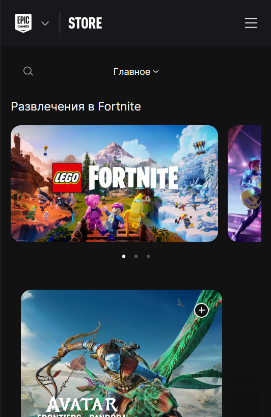


Рисунок 1.6 – Мобильная версия сайта Epic Games Store

Три аналогичных веб-сайта были подвергнуты анализу в рамках данного исследования, предоставляя обзор основных преимуществ и недостатков, а также выявляя используемые технологии в создании данных платформ. В итоге можно сделать вывод, что будущий сайт, разрабатываемый в рамках данной работы, будет обладать следующими преимуществами:

* быстрая загрузка сайта;
* простой и удобный дизайн;
* удобная навигация по сайту;
* кроссплатформенность;
* адаптивность под мобильные устройства;
* яркие и не раздражающие глаза цвета.

## Постановка задачи

Основные задачи для интернет-магазина:

* разработать структуру веб-сайта;
* разработать регистрацию и авторизацию пользователей;
* разработать возможность покупки игры;
* разработать возможность добавления и удаления товаров из списка желаемого;
* разработать возможность добавления и удаления товаров из корзины;
* разработать возможность добавления и удаления товаров из избранного;
* разработать возможность просмотра истории заказов.

## 1.4 Выводы по разделу

В этом разделе было подчеркнуто важность разработки интернет-магазина видеоигр, а также выделены ключевые преимущества и недостатки существующих платформ. Также проанализированы технологии, применяемые при создании подобных сайтов, и выявлены основные задачи, необходимые для создания интернет-магазина в данной области.

# 2 Проектирование веб-сайта

## 2.1 Выбор средств реализации

Главной задачей сайта является предоставление информации о видеоиграх, которые можно купить. Данный интернет-магазин будет предназначен для людей, которые хотят купить видеоигры. Сайт будет кроссплатформенным и адаптированным под мобильные устройства.

Интернет-магазин будет разработан в текстовом редакторе Visual Studio Code. Страницы сайта могут просматриваться в таких браузерах Google Chrome, Microsoft Edge, Safari, Opera, Mozilla Firefox и прочих современных веб-браузерах.

Visual Studio Code — редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений. Он включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса и средства для рефакторинга и многое другое.

Front-end — это часть сайта, которая видна пользователю. В данном случае это будет внешний вид сайта, его дизайн, а также навигация по сайту. В данном случае будет использованы следующие технологии: SCSS, JavaScript (TypeScript), React JS, React Router, Redux-Toolkit, Redux Persist и некоторые другие.

SASS — это препроцессор CSS, который добавляет в CSS мощные возможности, такие как переменные, вложенные правила, импорт и многое другое. Но самое главное, что SASS предоставляет механизмы, которые позволяют писать более краткий и лаконичный код.

JavaScript – это язык программирования, который используется для создания динамических веб-страниц. JavaScript позволяет создавать интерактивные веб-страницы, которые могут реагировать на действия пользователя.

TypeScript — это язык программирования, который является надмножеством JavaScript. TypeScript добавляет в JavaScript поддержку статической типизации и классов, что позволяет разработчикам писать более качественный код. TypeScript компилируется в JavaScript, поэтому его можно использовать в любом проекте, который использует JavaScript.

React JS — это библиотека JavaScript, предназначенная для разработки пользовательских интерфейсов. В отличие от других библиотек JavaScript React JS использует компонентный подход, который позволяет разрабатывать сложные интерфейсы из простых компонентов. С помощью React разработчики создают веб-приложения, которые изменяют отображение без перезагрузки страницы. Благодаря этому приложения быстро реагируют на действия пользователя, например, заполнение форм, применение фильтров, добавление товаров в корзину и так далее. React применяют для отрисовки компонентов пользовательского интерфейса. Также библиотека может полностью управлять фронтендом. В этом случае React используют с библиотеками для управления состоянием и роутинга, например, Redux и React Router.

React Router — это библиотека для маршрутизации в React-приложениях. React Router позволяет создавать маршруты, которые будут отображаться в зависимости от URL-адреса. Благодаря React Router можно создавать SPA (Single Page Application), которые не перезагружают страницу при переходе по ссылкам.

Redux — это библиотека для управления состоянием приложения.

Redux Toolkit — это набор инструментов для упрощения работы с Redux. Redux Toolkit позволяет создавать более простые и понятные редьюсеры, а также упрощает работу с асинхронными запросами.

Для хранения данных приложения будет использоваться формат JSON.

JSON — это текстовый формат обмена данными, который основан на JavaScript. JSON позволяет передавать данные между клиентом и сервером. JSON позволяет передавать данные в виде объектов, массивов, строк, чисел, логических значений и null.

Подводя итог, можно сказать, что в данном проекте будут использованы технологии, позволяющие создать кроссплатформенное приложение. Данный стек технологий позволяет создавать приложения, которые будут работать быстро и эффективно. Поэтому интернет-магази н будет работать быстро и эффективно, что позволит увеличить количество покупателей.

## 2.2 Проектирование uml-диаграммы

Роли пользователя — это набор прав, которые пользователь может получить в системе. В зависимости от роли пользователя, он может иметь доступ к различным функциям системы. В данном проекте предусмотрены следующие роли пользователей:

* Пользователь.
* Гость.

На основе предоставленного списка ролей можно построить UML диаграмму. UML диаграмма изображена на рисунке 2.1.

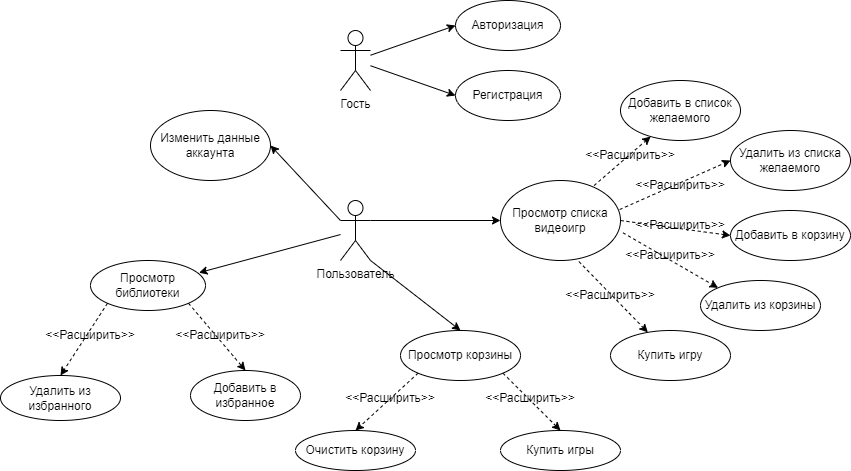


Рисунок 2.1 – UML диаграмма вариантов использования

В начале работы с сайтом пользователь является гостем. Ему будет доступен ограниченный функционал, включающий просмотр списка игр, но не их покупку. После регистрации он становится пользователем.

Пользователь может просматривать видеоигры и оформлять заказы. Пользователь может изменить свои данные или изменить пароль.

Подводя итог, можно сказать, что в данном проекте будет две роли пользователей. Пользователь будет иметь доступ к основным функциям сайта. Гость будет иметь доступ только к ограниченному функционалу.

## 2.3 Проектирование страниц

В данном проекте будет использована Flexbox-верстка (Flexbox), так как она позволяет создавать адаптивные веб-страницы, которые будут отображаться корректно на всех устройствах. С помощью этой технологии можно очень просто и гибко расставить элементы в контейнере, распределить доступное пространство между ними, и выровнять их тем или иным способом даже если они не имеют конкретных размеров. CSS Flexbox поддерживается всеми используемые на сегодняшний момент современными браузерами: Edge, Firefox, Chrome, Safari, Opera.

Исходя из выбора способа верстки были расположены все основные элементы сайта и разработаны макеты страниц. Макеты смотреть в [приложении Б](#_Приложение_Б_Макет).

Первым шагом было принято создать макет сайта на основе рассмотренных аналогов, сделать макет удобным и легким. По макету будет понятно, где будут располагаться основные блоки, картинки, текст. Макет будет представлен в [приложении А.](#_Приложение_A_Прототипы)

Вторым шагом было принято решение создать макеты страниц сайта. Макеты страниц сайта будут представлены в приложении Б.

На сайте будут представлены следующие страницы:

* + Home (Главная страница).
  + Games (Список игр)
  + Game (Игра).
  + Cart (Корзина).
  + Library (Библиотека).
  + Wishlist (Список желаемого).
  + SignIn (Вход).
  + SignUp (Регистрация).
  + Account (Профиль).
  + AccountEdit (Изменение профиля)

На странице Home будут расположены ссылки на другие разделы сайта.

На странице Games будет расположен список игр. При нажатии на карточку игры, пользователь будет перенаправлен на страницу Game.

На странице Game будет расположена информация об игре с возможностью купить ее или добавить в корзину или список желаемого.

На странице Cart будут расположены карточки продуктов, форма оформления заказа и информация об оплате.

На странице Wishlist будет находится список игр в списке желаемого.

На странице Library будут расположены игры в библиотеке с возможностью добавления их в список избранного.

На странице SignIn будут расположены форма входа, ссылка на страницу регистрации.

На странице SignUp будут расположены форма регистрации, ссылка на страницу входа.

На странице Account будут расположены данные пользователя и информация об аккаунте, а также ссылка на страницу AccountEdit.

На странице AccountEdit будет расположена форма изменения аккаунта с возможностью изменить аватарку, информацию о пользователе или пароль.

Также будет фиксированный Navbar с ссылками на главную страницу, страницу игр, страницу корзины, страницу профиля, страницу входа, страницу регистрации. Navbar будет расположен вверху страницы.

## 2.4 Выбор стилевого оформления

Для реализации стилевого оформления сайта было выбрано современное направление. Такой стиль позволяет привлечь внимание новых пользователей. Сайт не загромождается лишними деталями и большим количеством спецэффектов. Акцент на сайте ориентируется на основной контент.

Сайт будет состоять из 3-х основных цветов – зеленого, светло-серого и чёрного. Черный цвет будет использоваться для фона, а светло-серый для текста и некоторых элементов дизайна. Зеленый цвет будет использоваться в оформлении главных элементов. Эти цвета были подобраны так, потому что они будут приятны для глаз, и благодаря ним пользователь сразу обратит внимание на сайт.

В данном проекте будет использован один основной шрифт: Signate Grotesk. Шрифт подключался с помощью правила @font-face в SCSS.

## 2.5 Разработка пользовательских элементов

Логотип сайта представляет собой сочетания иконки сайта и его названия. Дизайн довольно минималистичен. Было принято решение не загромождать его множеством декоративных элементов. Результат представлен на рисунке 2.3.



Рисунок 2.3 – Логотип сайта F5Store

В проекте будет использована иконка сайта, которая будет отображаться во вкладке браузера. Иконка будет иметь формат .ico и будет иметь размер 32x32 пикселя. Иконка будет создана в программе Adobe Photoshop. Иконка представлена на рисунке 2.4.

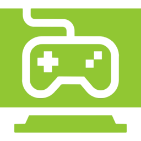


Рисунок 2.4 – Иконка сайта F5Store

В проекте будут пользовательские элементы, которые будут отвечать за взаимодействие пользователя с сайтом. Это будут следующие элементы: кнопки, модальные окна, формы и т.д. Пример кнопки будет на рисунке 2.5.



Рисунок 2.5 – Кнопка сайта F5Store

Также будет адаптивный слайдер на сайте, с помощью которого пользователь сможет просматривать изображения из интересующей игры до ее покупки. Пример слайдера будет на рисунке 2.6.

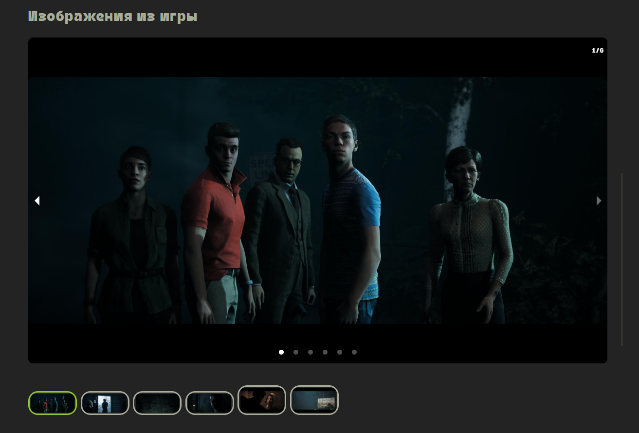


Рисунок 2.6 – Слайдер сайта F5Store

На сайте будут карточки игр, которые будут состоять из картинки и дополнительной информации с кнопкой перехода на основную страницу игры. Дизайн карточек товара будет простым и понятным. Пример карточки товара будет на рисунке 2.7.

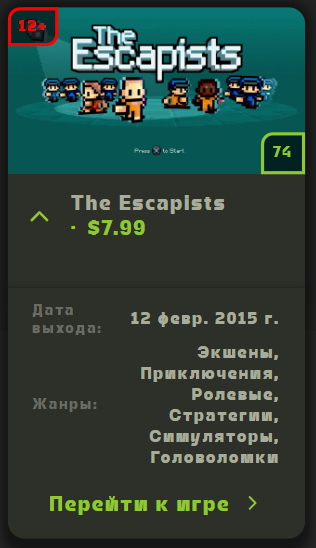


Рисунок 2.7 – Карточка игры

Также на сайте будет блок с ссылками на основные разделы сайта. Пользователь сможет легко переходить на страницы сайта используя данных элемент. Пример блока ссылок показан на рисунке 2.8.

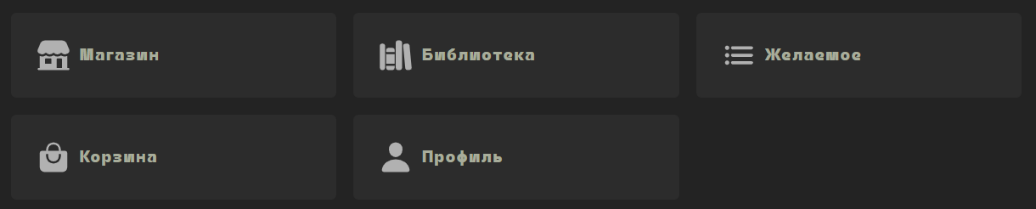


Рисунок 2.8 – Блок ссылок

Следующим основным элементом сайта будет блок корзины. Пользователь сможет добавлять продукты в корзину, удалять их из корзины и оформлять заказы. Пример блока корзины будет рисунке 2.9.

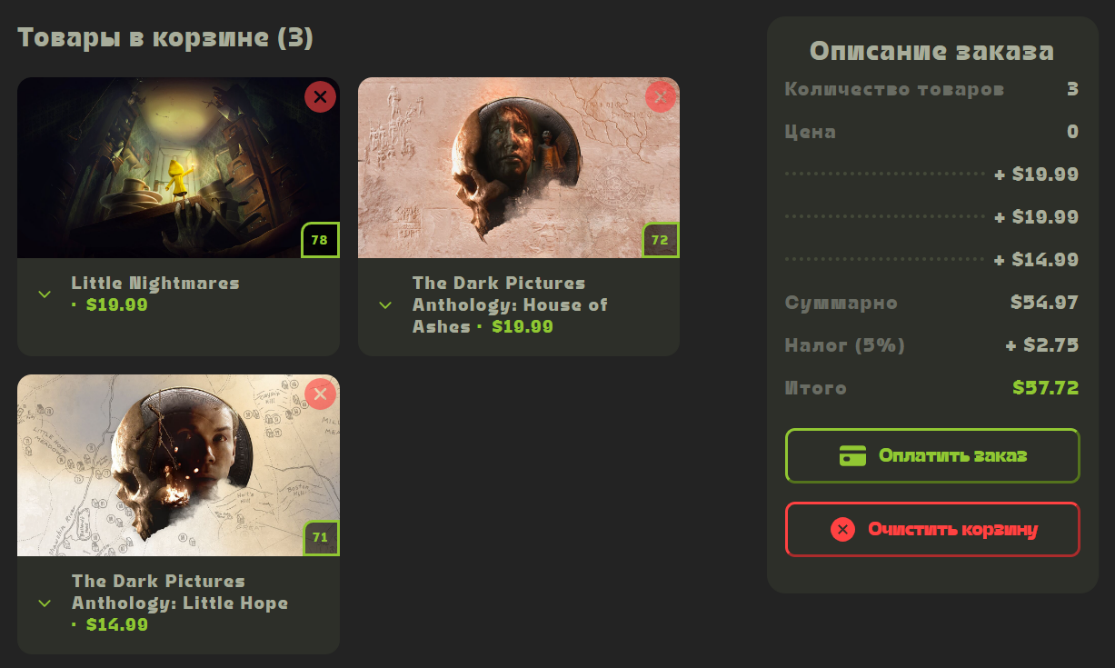


Рисунок 2.9 – Блок корзины

На сайте будут разработаны некоторые динамические эффекты и анимации, которые позволят придать сайту интерактивность и сделают его более привлекательным для пользователя. Анимация будет при наведении на кнопки и другие элементы интерфейса.

Элемент Header будет состоять из логотипа сайта, навигационного меню и блока аккаунта. В навигационном меню будет три пункта: Home, Games и Library. На рисунке 2.10 представлено оформление Navbar.



Рисунок 2.10 – Navbar сайта F5Shop

На данном этапе были созданы макеты сайта, которые будут использоваться для дальнейшей разработки сайта. Были выбраны основные цвета и шрифты, которые будут использоваться на сайте. Были разработаны некоторые динамические эффекты и анимации, которые позволят придать сайту интерактивность и сделают его более привлекательным для пользователя. Самое главное, чтобы сайт был удобным для пользователя и отвечал его потребностям.

## 2.6 Выводы по разделу

Подводя итоги этого раздела можно сказать, что были разработаны макеты сайта, которые будут использоваться для дальнейшей разработки сайта. Были выбраны основные цвета и шрифты, которые будут использоваться на сайте. Были разработаны некоторые динамические эффекты и анимации, которые позволят придать сайту интерактивность и сделают его более привлекательным для пользователя. Самое главное, чтобы сайт был удобным для пользователя и отвечал его потребностям.

# 3 Реализация структуры веб-сайты

## 3.1 Структура JSX-документа

Для реализации проекты было принято решение создавать разные компоненты. Компонент — это функция, которая возращает JSX формат.  JavaScript XML (JSX) — это расширение синтаксиса JavaScript, которое позволяет использовать HTML-подобный синтаксис для описания структуры интерфейса. Как правило, компоненты написаны с использованием JSX, но также есть возможность использования обычного JavaScript. Код примера JSX-формата показан в листинге 3.1.

|  |
| --- |
| import styles from './Footer.module.scss';  export function Footer() {  return (  <div className={styles.Footer}>  <div className={styles.Social}>  <p>Не пропускайте важные новости и события</p>  <div className={styles.Links}>  <img src='/icons/logo-discord.svg' alt='' />  <img src='/icons/logo-vk.svg' alt='' />  <img src='/icons/logo-instagram.svg' alt='' />  <img src='/icons/logo-twitch.svg' alt='' />  <img src='/icons/logo-youtube.svg' alt='' />  <img src='/icons/logo-tiktok.svg' alt='' />  </div>  </div>  <div className={styles.Apps}>  <p>F5Store также доступен в виде приложений</p>  <div className={styles.Links}>  <img src='/icons/logo-windows.svg' alt='' />  <img src='/icons/logo-tux.svg' alt='' />  <img src='/icons/logo-apple.svg' alt='' />  <img src='/icons/logo-android.svg' alt='' />  <img src='/icons/logo-google-playstore.svg' alt='' />  <img src='/icons/logo-apple-appstore.svg' alt='' />  </div>  </div>  <div className={styles.Separator}></div>  <div className={styles.Copyright}>  <p>© 2023, F5Store, Inc. Все права сохранены.</p>  <p>F5Store и логотип F5Store являются товарными знаками или зарегистрированными товарными знаками F5Store, Inc. в мире.</p>  <p>Все прочие марки и наименования продукции являются товарными знаками соответствующих владельцев.</p>  </div>  {/\* <div className={styles.Separator}></div> \*/}  <div className={styles.Agreements}>  <p>Условия предоставления услуг</p>  <p>Политика конфиденциальности</p>  <p>Правила возврата магазина</p>  </div>  </div>  );  } |

Листинг 3.1 – Структура JSX-формата

Самым главным компонентом будет index.tsx. Он будет содержать основные маршруты по сайту и провайдеры, необходимые для корректной работы приложения. Код примера маршрутов будет показан в листинге 3.2.

|  |
| --- |
| <Routes>  <Route  index  element={ <Navigate replace to='/home' /> }  />  <Route  path='home'  element={ <HomePage /> }  />  <Route  path='store'  element={ <StorePage /> }  />  <Route  path='library'  element={ <Navigate replace to='/library/all' /> }  />  <Route  path='library/:viewType'  element={ <LibraryPage /> }  />  <Route  path='cart'  element={ <CartPage /> }  />  <Route  path='wishlist'  element={ <WishlistPage /> }  />  <Route  path='game'  element={ <Navigate to='/store' /> }  />  <Route  path='game/:gameId'  element={ <GamePage /> }  />  <Route  path='account'  element={ <AccountPage /> }  />  <Route  path='account/edit'  element={ <AccountEditPage /> }  />  <Route  path='signin'  element={ <SignInPage /> }  />  <Route  path='signup'  element={ <SignUpPage /> }  />  <Route  path='signout'  element={ <SignOutPage /> }  />  <Route  path='\*'  element={ <NotFoundPage /> }  />  </Routes> |

Листинг 3.2 – Структура маршрутов

Следующим компонентом будет GameCard компонент. В нем будет находится карточка товара. Карточка игры будет состоять из:

* Фото игры.
* Названия игры.
* Цена игры.
* Кнопка перехода на страницу игры.

Код карточки игры будет показан в листинге 3.3.

|  |
| --- |
| const [ isExpanded, toggleExpanded, setExpanded ] = useToggle(false);  const cardRef = useRef<HTMLDivElement>(null);  const isHovering = useHover(cardRef);  useEffect(() => {  if (!isHovering) {  setExpanded(isHovering);  }  }, [setExpanded, isHovering]);  useOnClickOutside(cardRef, () => {  setExpanded(false);  })  const cardClickHandler = () => {  toggleExpanded();  };  const expandableClickHandler = (event: React.MouseEvent<HTMLDivElement>) => {  event.stopPropagation();  };  return (  <div className={styles.GameCard} onClick={cardClickHandler} ref={cardRef}>  <div className={styles.Cover}>  <img src={game.media.cover} alt='' />  {showAgeLimit && game.ageLimit && <span className={styles.AgeLimit}>{game.ageLimit}</span>}  {showMetacriticScore && game.metacriticScore && <span className={styles.MetacriticScore}>{game.metacriticScore}</span>}  {ActionButton && <div className={styles.ActionButton}><ActionButton game={game} /></div>}  </div>  <div className={styles.Head}>  <h3>  <ExpandIndicator isExpanded={isExpanded} />  <span>  <span className={styles.Title}>{game.title}</span>  {showPrice && <span className={styles.Price}> ·&nbsp;${game.price}</span>}  </span>  </h3>  </div>  <div className={styles.Expandable} data-expanded={isExpanded} onClickCapture={expandableClickHandler}>  <table className={styles.Info}>  <tbody>  <tr>  <th>Дата выхода:</th>  <td>{game.releaseDate}</td>  </tr>  <tr>  <th>Жанры:</th>  <td>{game.genres.join(', ')}</td>  </tr>  </tbody>  </table>  <Link to={`/game/${game.id}`}>Перейти к игре <img src='/icons/chevron-forward.svg' alt='' /></Link>  </div>  </div>  ); |

Листинг 3.3 – Структура карточки товара

Таким образом, в этом разделе были рассмотрены основные компоненты приложения, которые будут использоваться в дальнейшем. Хотелось бы отметить, что JSX-формат очень удобный для написания кода, так как он позволяет писать код в HTML-формате, что упрощает его чтение и понимание.

## 3.2 Структура Redux-Toolkit

Для того, чтобы удобно работать с данными, был использован Redux-Toolkit. В нем будет хранится информация о товарах, о пользователях, о заказах, о корзине и т.д.

Благодаря CreateSlice был реализован редьюсер для получения данных о товарах, о пользователях, о заказах, о корзине. В CreateSlice будет находиться имя редьюсера, начальное состояние, редьюсеры для изменения состояния. Код CreateSlice будет показан в листинге 3.4.

|  |
| --- |
| export type MenuState = {  isOpened: boolean,  };  const initialState: MenuState = {  isOpened: false,  }  export const menuSlice = createSlice({  name: 'menu',  initialState,  reducers: {  open: (state) => {  state.isOpened = true;  },  close: (state) => {  state.isOpened = false;  },  toggle: (state) => {  state.isOpened = !state.isOpened;  }  },  });  export const menuReducer = menuSlice.reducer;  export const menuActions = {  ...menuSlice.actions,  getState: (state: RootState) => state.menu,  getOpened: (state: RootState) => state.menu.isOpened,  }; |

Листинг 3.4 – Структура кода CreatSlice

Затем все редьюсеры нужно объединить в один редьюсер и поместить его в Store. В Store будет находиться все состояние приложения. Для того, чтобы получить данные из Store нужно использовать useSelector. Для того, чтобы изменить данные в Store нужно использовать useDispatch. Код структуры Store будет показан в листинге 3.5.

|  |
| --- |
| persistConfig = {  key: 'f5store',  storage: storage,  whitelist: ['auth'],  }  const persistReducer = createPersistReducer(persistConfig, rootReducer);  export const store = configureStore({  reducer: persistReducer,  middleware: (getDefaultMiddleware) =>  getDefaultMiddleware({  serializableCheck: {  ignoredActions: [FLUSH, REHYDRATE, PAUSE, PERSIST, PURGE, REGISTER],  },  }),  });  export const persistor = createPersistStore(store);  export type RootState = ReturnType<typeof store.getState>; |

Листинг 3.5 – Структура кода Store

В этом разделе были рассмотрены основные положения по созданию глобального хранилища данных с помощью Redux-Toolkit. Были рассмотрены основные понятия по созданию редьюсеров, асинхронных запросов к серверу, получения данных из Store.

## 3.3 Добавление таблиц стилей SCSS

Для стилизации был использован SCSS, который позволяет использовать переменные, вложенность, миксины, наследование, импорт и экспорт. Код использования SCSS показан в листинге 3.14.

|  |
| --- |
| .Header {  display: flex;  flex-direction: row;  justify-content: space-between;  align-items: center;  gap: 3em;  padding: 0.5em 1em 0.5em 1em;  background-color: var(--color-bg);  &[data-visible="false"] {  position: absolute;  top: -100%;  }  } |

Листинг 3.14 – Структура кода SCSS

В этом разделе была рассмотрена структура SCSS, что такое SASS и как он используется в проекте.

## 3.4 Выводы по разделу

Таким образом, были рассмотрены основные моменты, которые будут реализованы в проекте. В проекте будет реализована авторизация, регистрация, добавление игр в корзину, оформление заказа, получение информации об играх и о корзине.

# 4 Тестирование веб-сайта

## 4.1 Адаптивный дизайн веб-сайта

В настоящее время огромное количество пользователей предпочитают использовать интернет с мобильных устройств, и это число постоянно растет. Поэтому адаптация веб-сайта под небольшие экраны становится весьма актуальной задачей, в том числе и для моего курсового проекта.

Моя задача заключалась в создании веб-сайта, который автоматически подстраивается под различные размеры экранов. Основным условием адаптивности было отсутствие фиксированных размеров элементов страницы.

Для достижения адаптивности веб-сайта я использовал следующие средства: медиа-запросы (@media) и единицы измерения, связанные с параметрами области просмотра (эти единицы представляют собой процентные значения от текущей области просмотра браузера).

После написания основной структуры страницы на JSX и применения внешнего стиля, встал вопрос о тестировании. Веб-сайт был протестирован в различных браузерах, и результаты показали, что внешний вид сайта одинаков во всех браузерах.

При просмотре сайта на мобильных устройствах навигационное меню в верхней части экрана изменяется, превращаясь в выпадающее меню, которое открывается при нажатии на кнопку меню. На рисунке 4.1 представлен пример адаптивного дизайна для мобильных устройств.



Рисунок 4.1 – Адаптивный главной страницы для телефонов

Также при использовании мобильного устройства блоки с продуктами на странице для отображения располагаются друг под другом. На рисунке 4.2 можно увидеть пример.

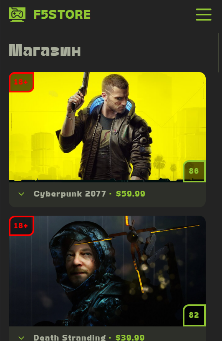
****

Рисунок 4.2 – Блоки с продуктами на мобильных устройствах

И таким образом можно сказать, что все блоки располагаются друг за другом, чтобы пользователю было все хорошо видно на его мобильном устройстве.

## 4.2 Кроссбраузерность веб-сайта

Очень важно, чтобы наш сайт был кроссбраузерным. Кроссбраузерность – это способность веб-ресурса отображаться одинаково и работать во всех популярных браузерах, без перебоев в функционировании и ошибок в верстке, а также с одинаково корректной читабельностью контента.

У каждого браузера есть свои встроенные, экспериментальные или нестандартные свойства и для того, чтобы они корректно работали было принято решения о внедрении вендорных префиксов. Они были автоматически прописаны после компиляции файла с расширением sass.

При проверке веб-сайта в Google Chrome никаких искажений текста, анимации, переходов не было выявлено. На рисунке 4.3 можно увидеть пример веб-страницы в Google Chrome.

|  |  |
| --- | --- |
|  |  |
| а) | б) |

Рисунок 4.3 – Вид веб-страницы в Google Chrome а) мобильная версия; б) компьютерная версия

В Edge сайт показал схожий результат: никаких отклонений как для мобильной версии сайта, так и для компьютерной не было выявлено, текст не съехал, не произошло его наложения на картинки. На рисунке 4.4 можно увидеть пример веб-страницы в Edge.

|  |  |
| --- | --- |
|  |  |
| а) | б) |

Рис 4.4 – Вид веб-страницы в Edge а) мобильная версия; б) компьютерная версия

В Яндекс всё так же работает без неисправностей, весь контент обладает правильным шрифтом, не наблюдается сдвиг текста или отсутствие картинок. На рисунке 4.5 можно увидеть пример веб-страницы в Яндекс.

|  |  |
| --- | --- |
|  |  |
| а) | б) |

Рис 4.5 – Вид веб-страницы в Яндекс а) мобильная версия; б) компьютерная версия

Также для тестирования сайта было использовано Linter. Linter - это инструмент, который проверяет код на наличие ошибок. В нашем случае он проверяет код на наличие ошибок в HTML и CSS.

При просмотре веб-сайта с помощью таких браузеров как Google Chrome, Яндекс и Edge общий вид веб-страницы не изменился.

Основной целью тщательной проверки веб-сайта является точная настройка всех показателей одной или всех его страниц, оперативное выявление и устранение всех дефектов, а также его адаптация для использования на различных устройствах.

Динамика веб-сайта на различных устройствах считается наиболее важной составляющей в его разработке. В данном этапе было проведено анализ поведения веб-сайта при использовании различных устройств, а также произведено тестирование в разнообразных браузерах. В результате этих мер был разработан адаптивный дизайн и обеспечена кроссбраузерность для веб-сайта. Проведенная работа позволяет пользователям легко ориентироваться на сайте, лишенном избыточных элементов, что упрощает его использование. При использовании мобильных устройств некоторые элементы сайта автоматически перестраивают свое положение, обеспечивая более удобное взаимодействие. В процессе тестирования были проверены все ссылки, включая модальные окна, а также проведена проверка форм регистрации пользователей и контента. Дополнительно была проверена совместимость с различными браузерами. (Google Chrome, Edge, Яндекс).

## 4.3 Ручное тестирование

Ручное тестирование – это тестирование, которое выполняется вручную. В данном случае ручное тестирование было проведено с целью проверки веб-сайта на соответствие требованиям, а также на наличие ошибок.

Тестирование frontend части включает в себе проверку правильности осуществления перехода между страницами, которые совершают пользователи системы, также проверку правильности отправления запросов, их обработки frontend сервером, а также правильности отображение представлений.

Рассмотрим сценарий регистрации. При заполнении формы неправильными данными должно быть получено оповещение об ошибки. Заполнение формы неправильными данными и уведомление об ошибке изображено на рисунке 4.6.

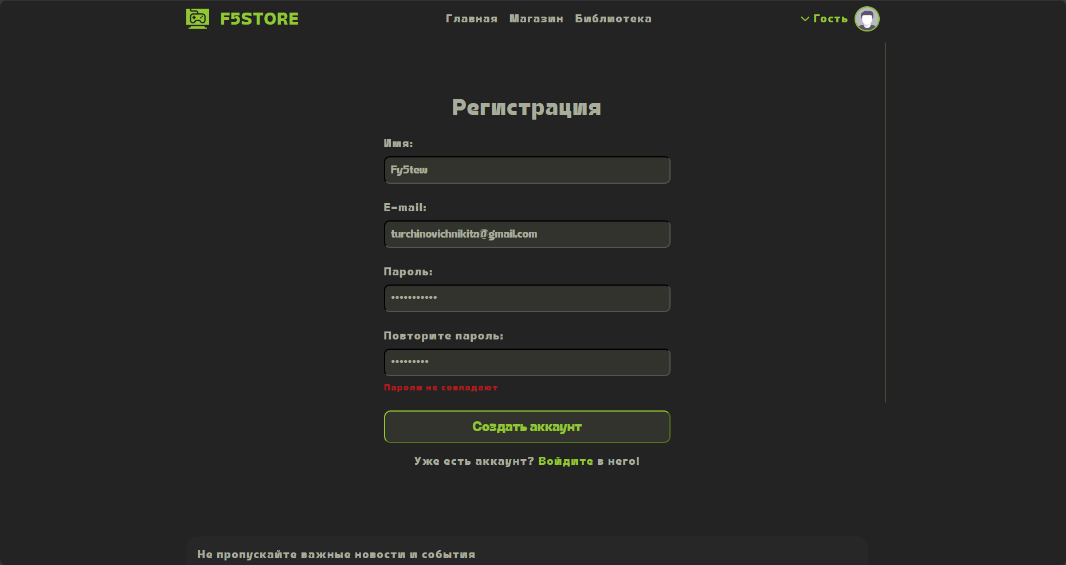


Рисунок 4.6 – Форма заполнения неверными данными

Ввод правильных данных изображен на рисунке 4.7.

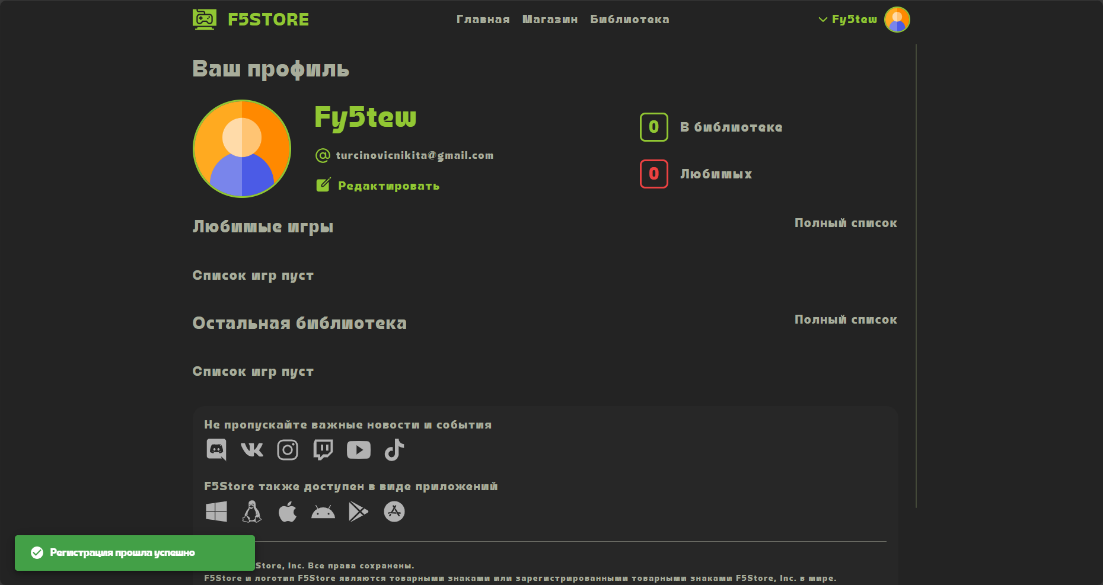


Рисунок 4.7 – Правильный ввод данных при авторизации

В ходе ручного тестирования были проведена валидация всех страниц сайта, проверены все ссылки, проверены все формы, проверен контент, проверена совместимость с браузерами. Также была проведена проверка работы корзины, где товары добавляются и удаляются из корзины. Во время тестирования не было обнаружено ошибок. Во всех браузерах работает корректно. Тестирование функционала сайта было выполнено вручную с использованием следующие браузеров Google Chrome, Edge и Яндекс.

Подводя итоги тестирования функционала сайта, можно сказать, что все функции сайта работают корректно. В ходе тестирования были проверены все ссылки (включая модальные окна), проверены формы регистрации пользователя. Была выполнена проверка контента.

## 4.4 Выводы по разделу

В этом разделе было проведено тестирование функционала сайта. Были проверены все функции сайта, которые должны работать корректно, а также были проверены все элементы, которые должны быть доступны для пользователя. Не было обнаружено никаких ошибок, которые могли бы повлиять на работу сайта.

# 5 Руководство пользователю

Данный веб-сайт предназначен для покупки игр. Для того, чтобы совершить покупку, необходимо зарегистрироваться на сайте. После регистрации необходимо авторизоваться на сайте. После авторизации пользователь может просматривать товары, добавлять их в корзину, а также удалять из корзины. После того, как пользователь добавил все товары в корзину, он может оформить заказ. После оформления заказа пользователь получает уведомление о том, что заказ оформлен. После этого пользователь может обнаружить купленные игры в своей библиотеке.

На странице Home пользователь может просматривать популярные товары и переходить на другие разделы сайта.

На странице Games пользователь может просматривать игры, которые находятся в магазине.

На странице Game пользователь может просматривать полностью описание игры, а также добавить ее в корзину.

На странице Cart пользователь может просматривать товары, которые он добавил в корзину, а также удалять их из корзины.

На странице Account пользователь может просматривать свои данные.

На странице AccountEdit пользователь может изменять свои данные.

# Заключение

Целью на курсовое проектирование было создание интернет-магазина для

продажи различных товаров для пользователей, которые хотят купить товары

в интернете.

В ходе курсовой работы были выполнены следующие задачи:

* Проанализировать и описать основные принципы работы React и Redux.
* Разработать приложение, используя React и Redux.
* Провести тестирование приложения.
* Провести анализ и оценку качества приложения.

А также были выполнены задачи по созданию интернет-магазина:

* Разработать регистрацию и авторизацию пользователей.
* Разработать возможность добавления и удаления игр из корзины.
* Разработать структуру веб-сайта.
* Разработать возможность покупки игры.

Подводя итоги курсовой работы, можно сделать вывод, что в ходе работы были выполнены все поставленные задачи. Были изучены основные принципы работы React и Redux, что позволило разработать приложение и закрепить полученные знания.

# Список использованных источников

1. Steam [Электронный ресурс]. – Режим доступа: https://store.steampowered.com/. – Дата обращения 21.11.2023.
2. GOG [Электронный ресурс]. – Режим доступа: https://www.gog.com/. – Дата обращения 21.11.2023.
3. Epic Games Store [Электронный ресурс]. – Режим доступа: https://store.epicgames.com/. – Дата обращения 21.11.2023.
4. Документация по React JS [Электронный ресурс]. - Режим доступа: https://reactjs.org/. – Дата обращения 21.11.2023.
5. Документация по Figma [Электронный ресурс]. - Режим доступа: <https://help.figma.com>. – Дата обращения 21.11.2023.
6. Документация по Redux/Redux-tolkit [Электронный ресурс]. - Режим доступа: https://redux-toolkit.js.org/ . – Дата обращения 21.11.2023.
7. Документация по Sass/Scss [Электронный ресурс]. - Режим доступа : <https://sass-scss.ru/guide/> – Дата обращения 21.11.2023.

# Приложение А Макет структуры веб-сайта

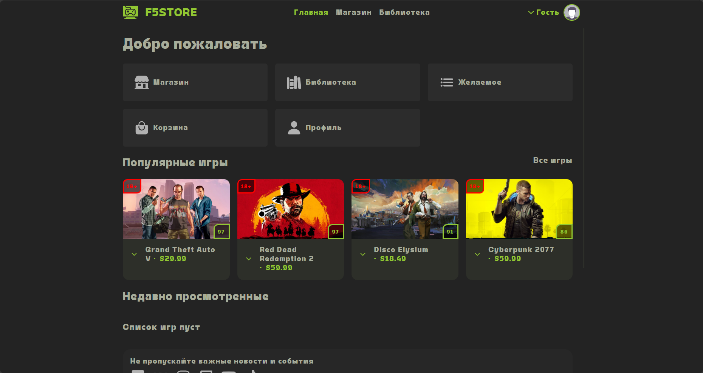
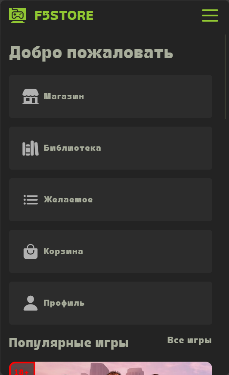
 

Рисунок 1 - макет страницы “Home”

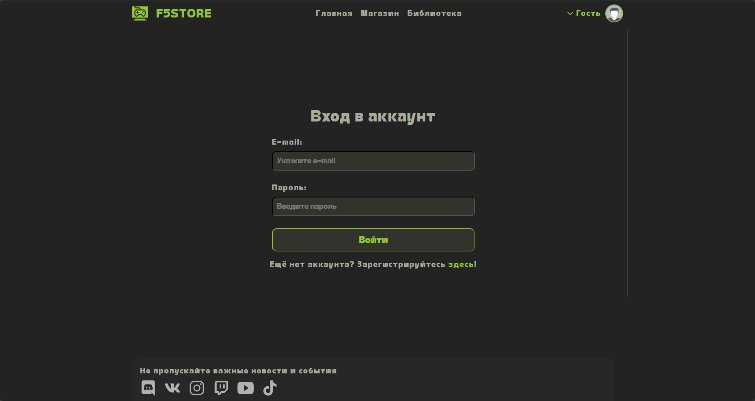
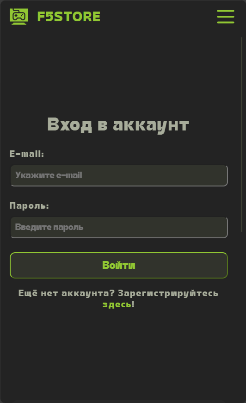
 

Рисунок 2 - макет страницы “SignIn”

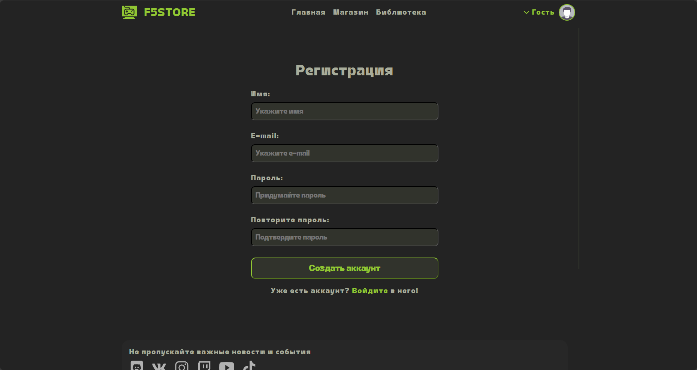
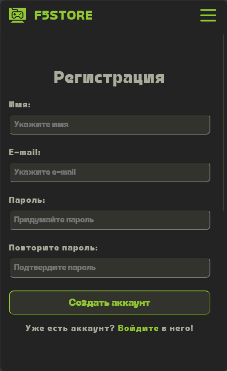
 

Рисунок 3-макет страницы “SignUp”

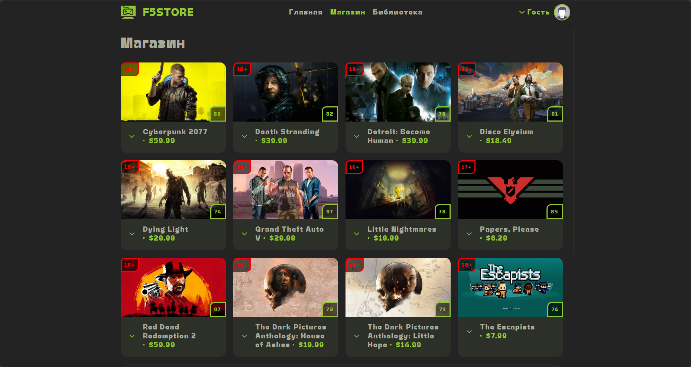
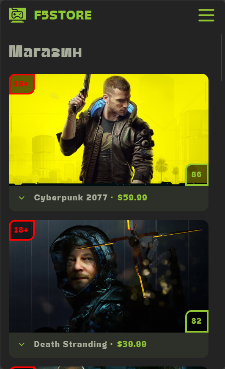
 

Рисунок 4-макет страницы “Games”

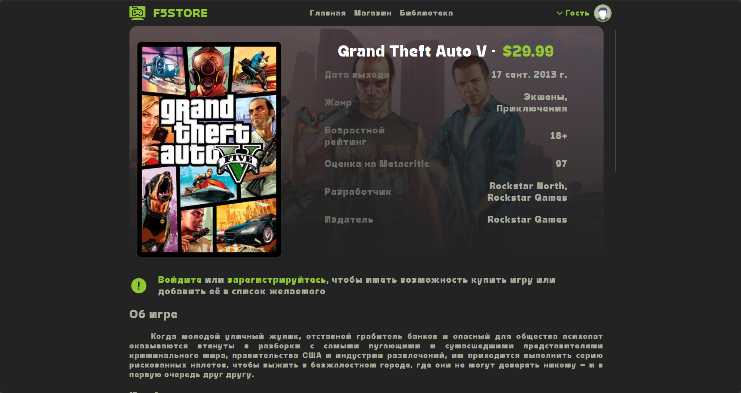
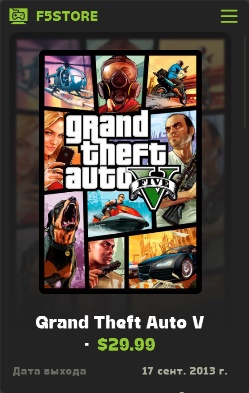
 

Рисунок 5-макет страницы “Game”

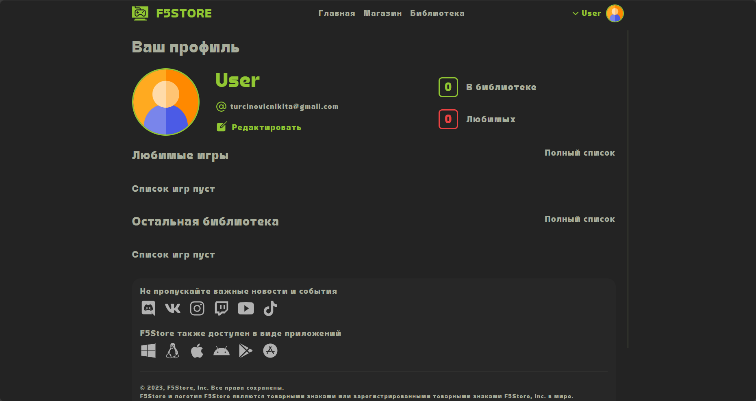
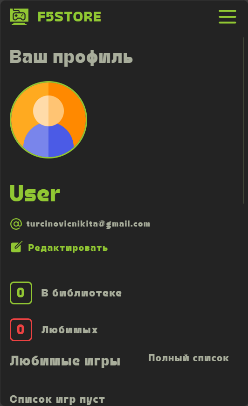
 

Рисунок 6-макет страницы “Account”

# Приложение Б Листинг JSX-документа

|  |
| --- |
| import React, { useRef, useEffect } from 'react';  import { useHover, useToggle } from 'usehooks-ts';  import { useOnClickOutside } from '../../hooks/useOnClickOutside';  import { Game } from '../../features/games/gamesSlice';  import { ExpandIndicator } from '../ExpandIndicator';  import styles from './GameCard.module.scss';  import { Link } from 'react-router-dom';  export type GameCardProps = {  game: Game,  showPrice?: boolean,  showAgeLimit?: boolean,  showMetacriticScore?: boolean,  ActionButton?: (props: { game: Game }) => JSX.Element,  }  export function GameCard({  game,  showPrice=false,  showAgeLimit=false,  showMetacriticScore=false,  ActionButton,  }: GameCardProps) {  const [ isExpanded, toggleExpanded, setExpanded ] = useToggle(false);  const cardRef = useRef<HTMLDivElement>(null);  const isHovering = useHover(cardRef);  useEffect(() => {  if (!isHovering) {  setExpanded(isHovering);  }  }, [setExpanded, isHovering]);  useOnClickOutside(cardRef, () => {  setExpanded(false);  })  const cardClickHandler = () => {  toggleExpanded();  };  const expandableClickHandler = (event: React.MouseEvent<HTMLDivElement>) => {  event.stopPropagation();  };  return (  <div className={styles.GameCard} onClick={cardClickHandler} ref={cardRef}>  <div className={styles.Cover}>  <img src={game.media.cover} alt='' />  {showAgeLimit && game.ageLimit && <span className={styles.AgeLimit}>{game.ageLimit}</span>}  {showMetacriticScore && game.metacriticScore && <span className={styles.MetacriticScore}>{game.metacriticScore}</span>}  {ActionButton && <div className={styles.ActionButton}><ActionButton game={game} /></div>}  </div>  <div className={styles.Head}>  <h3>  <ExpandIndicator isExpanded={isExpanded} />  <span>  <span className={styles.Title}>{game.title}</span>  {showPrice && <span className={styles.Price}> ·&nbsp;${game.price}</span>}  </span>  </h3>  </div>  <div className={styles.Expandable} data-expanded={isExpanded} onClickCapture={expandableClickHandler}>  <table className={styles.Info}>  <tbody>  <tr>  <th>Дата выхода:</th>  <td>{game.releaseDate}</td>  </tr>  <tr>  <th>Жанры:</th>  <td>{game.genres.join(', ')}</td>  </tr>  </tbody>  </table>  <Link to={`/game/${game.id}`}>Перейти к игре <img src='/icons/chevron-forward.svg' alt='' /></Link>  </div>  </div>  );  } |

Листинг 1 - структура JSX—документа

# Приложение В Листинг (SASS)

|  |
| --- |
| .GameCard {  $border-radius: 0.8em;  $hover-box-shadow-top: 0 0 0.5em 0.3em rgba(0, 0, 0, 0.5);  $hover-box-shadow-bottom: 0 0.5em 0.5em 0.3em rgba(0, 0, 0, 0.5);  position: relative;  min-width: 200px;  padding-bottom: 0.5em;  background-color: #2d2f29;  border-radius: $border-radius;  cursor: pointer;  transition: scale 0.1s ease-in-out;  @media (hover: hover) {  &:hover {  scale: 1.05;  box-shadow: $hover-box-shadow-top;  h3 img {  scale: 1.3;  }  }  }  &:has(.Expandable[data-expanded="true"]) {  scale: 1.05;  box-shadow: $hover-box-shadow-top;  border-bottom-left-radius: 0;  border-bottom-right-radius: 0;  z-index: 1;  h3 img {  scale: 1.3;  }  }  & > \*:not(& > .Cover) {  padding: 0.5em 1em;  }  a {  color: var(--color-primary);  text-decoration: none;  &:hover {  text-decoration: underline;  }  }  .Cover {  position: relative;  & > img {  width: 100%;  border-top-left-radius: $border-radius;  border-top-right-radius: $border-radius;  }  .AgeLimit {  position: absolute;  top: 0;  left: 0;  padding: 0.5em;  font-size: calc(1em - 0.3vw);  color: red;  background-color: rgba(0, 0, 0, 0.6);  border: 0.25em solid red;  border-top-left-radius: $border-radius;  border-bottom-right-radius: $border-radius;  }    .MetacriticScore {  position: absolute;  bottom: 4.8px;  right: 0;  padding: 0.6em;  font-size: calc(1em - 0.3vw);  color: var(--color-primary);  background-color: rgba(0, 0, 0, 0.6);  border: 0.25em solid var(--color-primary);  border-top-left-radius: $border-radius;  }  .ActionButton {  position: absolute;  top: 0;  right: 0;  width: 3em;  font-size: calc(1em - 0.3vw);  border-top-right-radius: $border-radius;  & > \* {  border-top-right-radius: $border-radius;  }  }  }  .Head {  h3 {  display: flex;  flex-direction: row;  align-items: center;  gap: 1em;  font-size: 1em;  margin: 0;  .Price {  color: var(--color-primary);  }  }  }  .Expandable {  position: absolute;  box-sizing: border-box;  width: 100%;  display: flex;  flex-direction: column;  gap: 1em;  padding-bottom: 1em !important;  background-color: inherit;  border-bottom-left-radius: $border-radius;  border-bottom-right-radius: $border-radius;  &[data-expanded="true"] {  top: 100%;  box-shadow: $hover-box-shadow-bottom;  }  &[data-expanded="false"] {  visibility: hidden;  z-index: -1;  }  a {  display: flex;  flex-direction: row;  justify-content: center;  gap: 1ch;  width: 100%;  img {  width: 1em;  height: 1em;  aspect-ratio: 1/1;  }  &:hover img {  scale: 1.5;  }  }  .Info {  font-size: 0.8em;  th {  font-size: 0.9em;  font-weight: normal;  text-align: inherit;  color: #696c63;  }    td {  text-align: right;  }  }  }  } |

Листинг 2 - структура SASS

# Приложение Г Листинг Redux

|  |
| --- |
| import { createSlice, PayloadAction } from '@reduxjs/toolkit';  import { Game } from '../games/gamesSlice';  import { RootState } from '../store';  import { guest } from './defaultUsers';  export type UserRegistration = {  name: string,  email: string,  password: string,  };  export type UnauthorizedUser = {  name: string,  avatar: string,  email: null,  password: null;  };  export type AuthorizedUser = {  name: string,  avatar: string,  email: string,  password: string,  };  export type User = UnauthorizedUser | AuthorizedUser;  type GameId = Game['id'];  export type AuthState = {  isAuthorized: boolean,  user: User,  recentlyViewed: GameId[],  library: GameId[],  favorites: GameId[],  cart: GameId[],  wishlist: GameId[],  }  const initialState: AuthState = {  isAuthorized: false,  user: guest,  recentlyViewed: [],  library: [],  favorites: [],  cart: [],  wishlist: [],  };  export const authSlice = createSlice({  name: 'auth',  initialState,  reducers: {  authorize: (state, action: PayloadAction<UserRegistration>) => {  state.isAuthorized = true;  state.user = {  ...action.payload,  avatar: 'default',  };  state.recentlyViewed = [];  state.library = [];  state.favorites = [];  state.cart = [];  state.wishlist = [];  },  unauthorize: (state) => {  state.isAuthorized = false;  state.user = guest;  state.recentlyViewed = [];  state.library = [];  state.favorites = [];  state.cart = [];  state.wishlist = [];  },  setName: (state, action: PayloadAction<string>) => {  if (!state.isAuthorized) return;  state.user.name = action.payload.slice(0, 15);  },  setAvatar: (state, action: PayloadAction<string>) => {  if (!state.isAuthorized) return;  state.user.avatar = action.payload;  },  setEmail: (state, action: PayloadAction<string>) => {  if (!state.isAuthorized) return;  state.user.email = action.payload;  },  setPassword: (state, action: PayloadAction<string>) => {  if (!state.isAuthorized) return;  state.user.password = action.payload;  },  pushRecentlyViewed: (state, action: PayloadAction<GameId>) => {  state.recentlyViewed = state.recentlyViewed.filter(gameId => gameId !== action.payload);  state.recentlyViewed.unshift(action.payload);  },  addToLibrary: (state, action: PayloadAction<GameId>) => {  if (state.cart.includes(action.payload)) {  state.cart = state.cart.filter(gameId => gameId !== action.payload);  }  if (state.wishlist.includes(action.payload)) {  state.wishlist = state.wishlist.filter(gameId => gameId !== action.payload);  }  if (!state.library.includes(action.payload)) {  state.library.push(action.payload);  }  },  removeFromLibrary: (state, action: PayloadAction<GameId>) => {  if (state.favorites.includes(action.payload)) {  state.favorites = state.favorites.filter(gameId => gameId !== action.payload);  }  if (state.library.includes(action.payload)) {  state.library = state.library.filter(gameId => gameId !== action.payload);  }  },  addToFavorites: (state, action: PayloadAction<GameId>) => {  if (!state.library.includes(action.payload)) {  return;  }  if (!state.favorites.includes(action.payload)) {  state.favorites.push(action.payload);  }  },  removeFromFavorites: (state, action: PayloadAction<GameId>) => {  if (state.favorites.includes(action.payload)) {  state.favorites = state.favorites.filter(gameId => gameId !== action.payload);  }  },  addToCart: (state, action: PayloadAction<GameId>) => {  if (state.library.includes(action.payload)) {  return;  }  if (!state.cart.includes(action.payload)) {  state.cart.push(action.payload);  }  },  removeFromCart: (state, action: PayloadAction<GameId>) => {  if (state.cart.includes(action.payload)) {  state.cart = state.cart.filter(gameId => gameId !== action.payload);  }  },  addToWishlist: (state, action: PayloadAction<GameId>) => {  if (state.library.includes(action.payload)) {  return;  }  if (!state.wishlist.includes(action.payload)) {  state.wishlist.push(action.payload);  }  },  removeFromWishlist: (state, action: PayloadAction<GameId>) => {  if (state.wishlist.includes(action.payload)) {  state.wishlist = state.wishlist.filter(gameId => gameId !== action.payload);  }  },  }  });  export const authReducer = authSlice.reducer;  export const authActions = {  ...authSlice.actions,  getState: (state: RootState) => state.auth,  getAuthorized: (state: RootState) => state.auth.isAuthorized,  getUser: (state: RootState) => state.auth.user,  getRecentlyViewed: (state: RootState) => state.auth.recentlyViewed,  getLibrary: (state: RootState) => state.auth.library,  isInLibrary: (gameId: GameId) => (state: RootState) => state.auth.library.includes(gameId),  getFavorites: (state: RootState) => state.auth.favorites,  isInFavorites: (gameId: GameId) => (state: RootState) => state.auth.favorites.includes(gameId),  getCart: (state: RootState) => state.auth.cart,  isInCart: (gameId: GameId) => (state: RootState) => state.auth.cart.includes(gameId),  getWishlist: (state: RootState) => state.auth.wishlist,  isInWishlist: (gameId: GameId) => (state: RootState) => state.auth.wishlist.includes(gameId),  }; |

Листинг 3 - структура Redux