Министерство образования и науки Республики Казахстан

Некоммерческое акционерное общество «Торайгыров университет»

**ДИПЛОМНЫЙ ПРОЕКТ**

**Создание платформы по буккросингу и обмену книгами**

По образовательной программе 6B06102 – «Информационные системы»

Павлодар

2022Министерство образования и науки Республики Казахстан

Некоммерческое акционерное общество «Торайгыров университет»

Факультет Computer Science

Допущен(а) к защите

Руководитель ОП \_\_\_\_\_\_\_\_\_\_\_\_ Н. Н. Оспанова

(подпись)

\_\_ \_\_\_\_\_\_\_\_\_ 20\_\_года

**ДИПЛОМнЫЙ ПРОЕКТ**

**Создание информационно аналитической системы по потреблению основных продуктов питания разными слоями населения на основе открытых данных РК**

**по образовательно программе 6B06102 – Информационные системы**

**Выполнил(а)** \_\_\_\_\_\_\_\_\_\_\_\_ С. А. Зайцев

(подпись)

Группа ИС-402

**Научный руководитель**

магистр, ст. преподаватель \_\_\_\_\_\_\_\_\_\_\_\_ С. Н. Талипов

(подпись)

**Нормоконтроль**

Ст. преподаватель \_\_\_\_\_\_\_\_\_\_\_\_ А. З. Даутова

(подпись)

Павлодар

2022

|  |  |  |
| --- | --- | --- |
| Задание  на выполнение ДР/П | лого 2020 | Ф МИ СМК 4.01.6/02 |

Министерство образования и науки Республики Казахстан

Некоммерческое акционерное общество «Торайгыров университет»

Факультет «Computer Science»

Специальность Информационные системы

**Утверждаю**

Руководитель образовательных программ канд. Пед. наук, профессор

Н. Н. Оспанова

20 года

**ЗАДАНИЕ**

на выполнение дипломной (го) работы/проекта

Обучающемуся Зайцеву Станиславу Алексееву

Тема ДР/П Создание платформы по буккросингу и обмену книгами

утверждена приказом по университету от 20 года №

Срок сдачи ДР/П 20 года

Исходные данные к ДР/П Методические рекомендации по выполнению ДР/П

Перечень подлежащих разработке в ДР/П вопросов или краткое содержание

выпускной работы (дипломной работы/дипломного проекта):

1 Цель и актуальность проекта

2 Обзор основных задач и функций проекта

3 Выбор средств и инструментов для разработки проекта

4 Общий алгоритм реализации проекта

5 Архитектура проекта

6 Разработка интерфейса

7 Разработка базы данных

8 Разработка API

|  |  |  |
| --- | --- | --- |
| Задание  на выполнение ДР/П | лого 2020 | Ф МИ СМК 4.01.6/02 |

9 Разработка программного кода

Перечень графического материала с точным указанием обязательных чертежей:31

Рекомендуемая основная литература: 7

**ГРАФИК**

подготовки дипломной работы/проекта

|  |  |  |
| --- | --- | --- |
| Наименование разделов, перечень  разрабатываемых вопросов | Сроки представления научному  руководителю и консультантам | Подпись |
| Описание и сравнительный анализ средств для разработки |  |  |
| Постановка цели |  |  |
| Разработка чат бота |  |  |

**Подписи**

консультантов на законченную дипломную работу (проект), с указанием относящихся к ним разделов ДР/П

|  |  |  |  |
| --- | --- | --- | --- |
| Наименования разделов | Консультанты, Ф.И.О.  (уч. степень, звание) | Дата  подписания | Подпись |
| Описание и сравнительный анализ средств для разработки |  |  |  |
| Постановка цели |  |  |  |
| Разработка чат бота |  |  |  |

Научный руководитель С. Н. Талипов

(подпись) (инициалы, фамилия)

Задание принял к исполнению

обучающийся С. А. Зайцев

(подпись) (инициалы, фамилия)

Дата 20\_\_года

**Содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 5 |
| 1 | Теоретическая часть | 6 |
| 1.1 | Цель и актуальность проекта | 6 |
| 1.2 | Обзор основных задач и функций проекта | 7 |
| 1.3 | Выбор средств и инструментов для разработки проекта | 7 |
| 1.4 | Общий алгоритм реализации проекта | 8 |
| 2. | Практическая часть | 9 |
| 2.1 | Архитектура проекта | 9 |
| 2.2 | Разработка интерфейса | 9 |
| 2.3 | Разработка базы данных |  |
| 2.4 | Разработка API | 10 |
| 2.5 | Разработка программного кода | 10 |
|  | Заключение | 29 |
|  | Список использованных источников | 30 |
|  | Приложение А | 31 |

**Введение**

В настоящее время, многие люди предпочитают чтение книг в электронном формате. Однако, существует большая группа любителей бумажных книг, которые также хотят иметь доступ к большому количеству литературы и обмениваться книгами между собой. В связи с этим, создание платформы по буккросингу и обмену книгами представляется актуальным и полезным проектом.

Цель данной работы научиться новым и современным технологиям, которые используются в компаниях мирового уровня для написания больших сайтов и приложений. А также сделать работу и добавить ее в профессиональное портфолио, для последующего показа работодателям в качестве примера и демонстрации навыков.

В рамках данного проекта была создана платформа на основе современных технологий, таких как React JS, TypeScript, Redux и Json-server. Платформа включает в себя несколько страниц, таких как страница списка всех книг, страница о компании, страница авторизации клиента, страница личного аккаунта клиента и страница конкретной книги.

Благодаря использованию современных технологий, пользователи могут быстро находить нужные книги, ознакамливаться с их основными характеристиками и связываться с владельцами книг. Это создает удобную платформу для обмена книгами между любителями чтения и способствует распространению книжной культуры.

Таким образом, разработка платформы по буккросингу и обмену книгами на основе современных технологий представляет собой важный проект, который способен показать навыки и умения студента для последующих работодателей. Также при усовершенствовании проекта для коммерческой основы платформа способна удовлетворить потребности любителей чтения и обеспечить им доступ к большому количеству литературы.

1. **Теоретическая часть**

**1.1 Цель и актуальность проекта**

В настоящее время, существует большая группа людей, которые не хотят отказываться от бумажных книг и хотят иметь доступ к широкому выбору литературы. Однако, не всегда у них есть возможность купить нужную книгу, а также не все книги доступны в их регионе. В этом случае, платформа по буккросингу и обмену книгами представляет собой идеальное решение проблемы, которое облегчает процесс обмена книгами между пользователями и позволяет им получить доступ к большому количеству книг.

Несмотря на то, что электронные книги являются более компактными и удобными в путешествиях, они не могут заменить уникальный опыт чтения бумажных книг.

Бумажные книги имеют важную значимость перед электронными, поскольку они предоставляют своим читателям уникальный опыт чтения, который невозможен сравнить с электронными форматами. Бумажные книги ощущаются в руках, позволяют насладиться ароматом бумаги и чернил, их страницы можно ощупать, перелистывать, делать пометки и выделения.

Кроме того, бумажные книги не требуют подключения к интернету и не зависят от устройств, что обеспечивает надежность и доступность. Также, многие люди предпочитают читать бумажные книги в качестве отдыха от экранов мониторов и смартфонов.

Актуальность проекта заключается в том, что он способствует развитию книжной культуры и увеличению доступности литературы для людей, проживающих в разных регионах. Кроме того, платформа представляет собой удобное пространство для обмена мнениями о книгах, что способствует развитию книжного сообщества.

**1.2 Обзор основных задач и функций проекта**

Приложение по буккросингу и обмену книгами предоставляет следующие функции:

- поиск книг по различным критериям. Это позволяет быстро и удобно найти нужные книги, используя поиск по названию, автору или жанру. Благодаря этой функции пользователи могут быстрее находить интересующие их книги, а также изучать новые книги, читать аннотации и отзывы других пользователей, тем самым создавая подборку желаемых книг.

- добавление книг в список доступных для обмена. Это позволяет пользователям добавлять книги, которые они хотели бы обменять или поделиться с другими участниками платформы. Таким образом, пользователи могут предложить к обмену книги, которые уже прочитали, или обнаружили лишние в своей коллекции, что поможет другим пользователям получить доступ к большему количеству книг.

- просмотр подробной информации о книге. Это позволяет пользователям получить всю необходимую информацию о книге, включая фотографию, описание, автора, жанр и контакты пользователя, который добавил эту книгу. Благодаря этой функции пользователи могут быстрее принимать решение о том, стоит ли им читать эту книгу, а также связываться с владельцами книг, которые они хотели бы обменять или поделиться.

- создание личного аккаунта и хранение данных о своих книгах. Это позволяет пользователям создать личный аккаунт, где они могут хранить информацию о своих книгах и управлять списком доступных книг для обмена. Благодаря этой функции пользователи могут создать свою персональную библиотеку книг, контролировать доступность книг для обмена и управлять своими обменами.

- доступ к контактам владельца книги. Это позволяет пользователям находить нужную книгу и самостоятельно связываться с владельцем в удобной социальной сети, по почте или напрямую по номеру телефона.

- просмотр информации о компании. Это позволяет пользователям ознакомиться с информацией о компании, которая разработала платформу, и получить контактные данные для обратной связи. Благодаря этой функции пользователи могут узнать больше о платформе, ее целях и задачах, а также связаться с разработчиками, чтобы получить помощь или задать вопросы.

Все эти функции позволяют пользователям нашей платформы наслаждаться чтением книг и общением с другими читателями в удобной и комфортной обстановке. Они сделают процесс поиска, обмена и чтения книг более удобным и доступным, а также позволят нашим пользователям находить новые друзья и единомышленников.

**1.3 Выбор средств и инструментов для разработки проекта**

IntelliJ IDEA - это интегрированная среда разработки для создания высококачественных приложений на Java, Kotlin, Groovy, Scala и других языках программирования. Она обладает широким спектром функциональных возможностей, интуитивно понятным интерфейсом и инструментами для улучшения производительности и эффективности разработки. IntelliJ IDEA позволяет быстро создавать и отлаживать код, а также облегчает процесс совместной работы между разработчиками.

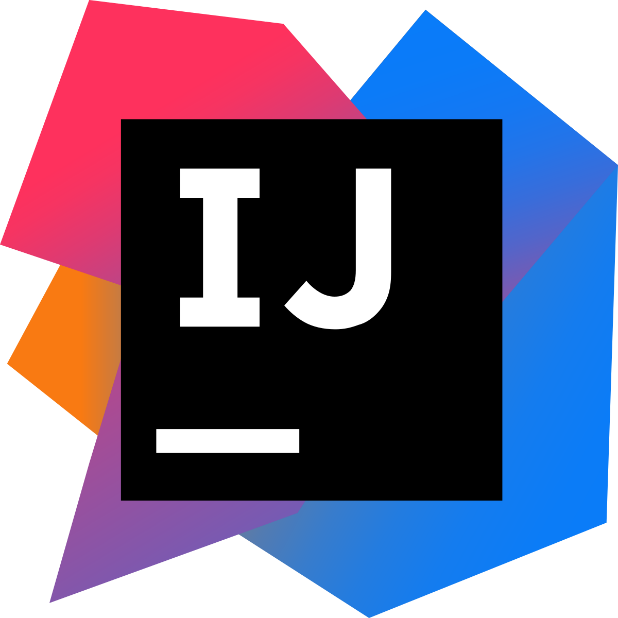


Рисунок 1 − Логотип IntelliJ Idea

Одним из принципов разработки веб-приложений является Single Page Application (SPA). Этот подход был придуман в начале 2000-х годов, когда появились первые богатые интернет-приложения (RIA) и было необходимо улучшить производительность и удобство работы с веб-сайтами.

SPA - это веб-приложение, которое загружает только одну страницу HTML и динамически обновляет ее содержимое, используя JavaScript для связи с сервером и манипулирования DOM. Этот подход позволяет создавать приложения с более быстрым откликом и более удобным пользовательским интерфейсом. Кроме того, SPA позволяет создавать более гибкие и масштабируемые приложения, поскольку они могут работать в фоновом режиме, загружать только те данные, которые нужны в данный момент, и более эффективно использовать ресурсы клиента.

Преимущества использования SPA включают:

- более быструю загрузку страниц и удобство работы с приложением;

- меньшую нагрузку на сервер и возможность кеширования данных на клиенте;

- более гибкую архитектуру приложения и возможность использования различных фреймворков и библиотек для разработки.

Использование инструментов, таких как IntelliJ IDEA, в сочетании с принципом SPA, позволяет создавать высококачественные, быстрые и масштабируемые веб-приложения, которые могут удовлетворить потребности самых требовательных пользователей.

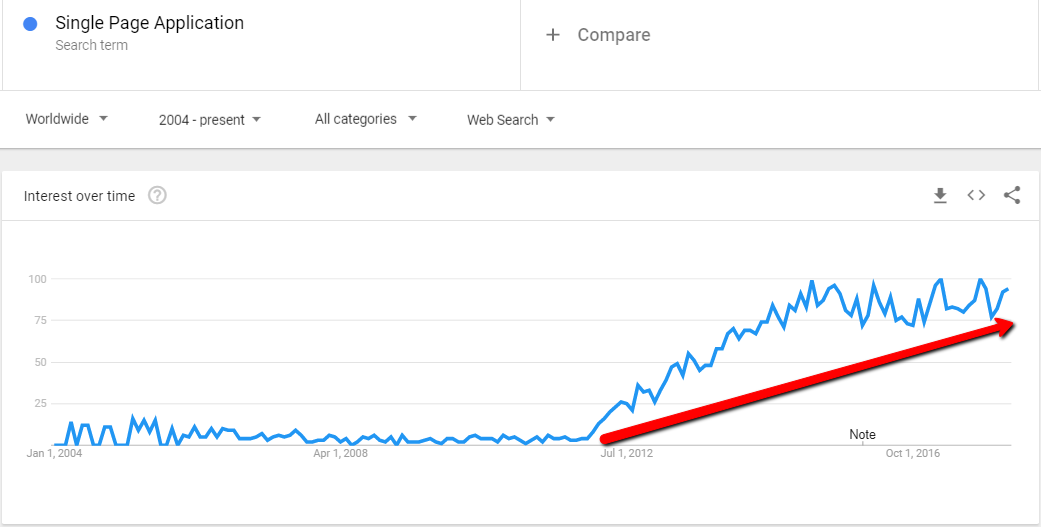


Рисунок 2 − График развития популярности SPA

Чтобы создать SPA использовались такие технологии как React JS в сочетании с TypeScript и Redux.

React JS - это библиотека JavaScript для создания пользовательских интерфейсов (UI), которая была разработана компанией Meta. Он был впервые выпущен в 2013 году и быстро стал одним из самых популярных инструментов для создания фронтенд-приложений. Он используется для создания динамических и масштабируемых интерфейсов для веб-приложений.



Рисунок 3 − Логотип компании Meta

React JS основан на принципе компонентов, что означает, что интерфейс пользователя создается из набора отдельных компонентов. Эти компоненты могут быть переиспользованы в различных частях приложения, что упрощает разработку и ускоряет процесс создания приложений. Кроме того, React использует виртуальный DOM (VDOM), который позволяет ускорить процесс отображения изменений на странице и повысить производительность приложения.

React JS имеет множество преимуществ, таких как:

- удобный и интуитивно понятный синтаксис;

- высокая производительность благодаря использованию виртуального DOM;

- возможность создавать многокомпонентные приложения, которые легко масштабировать;

- широкий выбор инструментов и библиотек для разработки.

React также обладает большим сообществом разработчиков, которые создают множество инструментов и библиотек для улучшения процесса разработки. Для создания отдельной страницы для каждой книги была использована библиотека React Router. React Router - это набор инструментов для реализации навигации в веб-приложении, который позволяет создавать SPA с множеством страниц и маршрутов.

С помощью React Router был реализован механизм маршрутизации, который позволяет перенаправлять пользователя на нужную страницу приложения в зависимости от запроса. Каждая книга имеет свой уникальный идентификатор, поэтому был создан маршрут для отображения информации о конкретной книге, основанный на этом идентификаторе.

При обращении к маршруту для отображения информации о книге, приложение загружает соответствующую страницу с информацией о книге, включая фотографию, описание, автора, жанр и контакты пользователя, который добавил эту книгу.

Таким образом, использование React Router позволило создать удобный и легко навигируемый интерфейс для отображения информации о книгах и обеспечить быстрый доступ к нужной книге с помощью уникальных маршрутов для каждой книги.

TypeScript - это язык программирования, который является надмножеством JavaScript, добавляющим статическую типизацию и другие новые функции к языку. Он был разработан Microsoft и впервые выпущен в 2012 году. TypeScript позволяет создавать более безопасный и поддерживаемый код, улучшая производительность и уменьшая количество ошибок.

TypeScript добавляет статическую типизацию к JavaScript, что означает, что переменные, функции и объекты имеют явно определенные типы. Это улучшает производительность приложения и облегчает его поддержку и разработку. TypeScript также имеет множество других возможностей, таких как поддержка классов, наследования, интерфейсов и многие другие.

React JS в сочетании с TypeScript позволяет разработчикам создавать более безопасный и поддерживаемый код, улучшая производительность и уменьшая количество ошибок. Это особенно полезно для больших проектов, где множество разработчиков работают вместе. React JS и TypeScript также интегрируются между собой очень хорошо, что обеспечивает более эффективную и безопасную разработку.

В настоящее время React JS и TypeScript считаются стандартом технологии для создания веб-приложений. Большинство крупных компаний используют React JS и TypeScript для разработки своих приложений. React JS и TypeScript имеют большое сообщество разработчиков и множество инструментов и библиотек для разработки, что делает их одними из наиболее популярных технологий для создания современных веб-приложений.

Redux - это библиотека управления состоянием для JavaScript-приложений, которая работает в связке с React JS. Redux позволяет управлять сложным состоянием приложения в единой глобальной области состояния, которая доступна для всех компонентов React. Redux помогает управлять состоянием приложения, упрощает его разработку и облегчает отладку.

Redux использует паттерн Flux, который представляет собой архитектурный шаблон для управления состоянием веб-приложения. Redux работает с единственным глобальным объектом, который содержит всю информацию о состоянии приложения. Каждый раз, когда происходит изменение состояния, создается новый объект состояния, который сохраняется в глобальном хранилище.

Redux позволяет управлять состоянием приложения с помощью actions, reducers и store. Actions - это объекты, которые описывают изменение состояния приложения. Reducers - это функции, которые принимают текущее состояние и действие и возвращают новое состояние. Store - это единственное место, где хранится состояние приложения.

React JS и Redux часто используются вместе, что обеспечивает более удобную и эффективную разработку веб-приложений. С React JS можно легко интегрировать Redux, что позволяет управлять сложным состоянием приложения с помощью простых и понятных инструментов.

Сегодня React JS и Redux считаются одними из самых популярных технологий для разработки веб-приложений. Они имеют широкое сообщество разработчиков и множество инструментов и библиотек для разработки, что делает их одними из наиболее популярных технологий для создания современных веб-приложений.

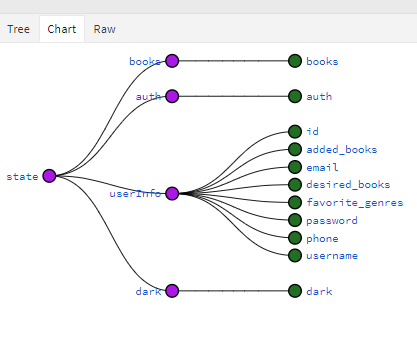


Рисунок 4 − Пример расположения данных в Redux

Для веб-приложения по буккросингу и обмену книгами был разработан современный минималистичный дизайн. Это такой подход к дизайну, который убирает все лишнее и фокусируется на существенных элементах. Этот подход создает простые, но красивые интерфейсы, которые легко воспринимаются пользователем. Такой подход позволил убрать все ненужное и сосредоточиться на функциях, которые действительно необходимы пользователям.

UI UX (user interface и user experience) - это еще один важный аспект в разработке приложений. Цель UI UX - создать интерфейс, который будет максимально простым, интуитивно понятным и удобным для пользователя. Поэтому был разработан интерфейс с интуитивно понятным пользовательским опытом, который позволяет быстро находить нужные книги и добавлять свои книги для обмена.

Также для веб-приложения по буккросингу и обмену книгами был разработана современна возможность перейти на темную тему. Темная тема, это альтернативный режим интерфейса приложения, где цветовая схема использует темные оттенки вместо светлых цветов. Такой режим имеет ряд преимуществ, которые сделали его популярным в последнее время.

Во-первых, темная тема позволяет снизить нагрузку на глаза пользователей в условиях недостаточной освещенности, таких как ночное время или темные помещения. Она также может снизить утомляемость глаз во время длительного использования приложения, что особенно важно для людей, которые проводят много времени за экраном.

Во-вторых, темная тема позволяет снизить энергопотребление смартфонов и ноутбуков с OLED-дисплеями. Такие дисплеи используют меньше энергии, когда на экране отображаются темные цвета.

В приложении по буккросингу и обмену книгами была реализована темная тема, чтобы предоставить пользователям возможность выбора цветовой схемы, которая наиболее удобна для них. Кроме того, темная тема делает приложение более современным и стильным, что может привлечь новых пользователей.

Таким образом, темная тема является полезной функцией, которая улучшает пользовательский опыт и делает приложение более удобным и доступным для пользователей.

* 1. **Общий алгоритм реализации проекта**

Проект по буккроссингу и обмену книгами был успешно реализован при помощи следующего алгоритма.

Сначала были определены требования к проекту, проведен анализ рынка и потребностей пользователей, определены функции и основные характеристики приложения.

Затем был разработан дизайн интерфейса приложения, созданы макеты и определены функциональные возможности пользовательского интерфейса.

Для разработки проекта были выбраны и установлены необходимые инструменты и технологии, включая фреймворк React, TypeScript, Redux и Json-server.

После этого была выполнена разработка серверной и клиентской частей приложения, включая разработку функций и логику приложения.

После завершения разработки приложение было тщательно протестировано на разных устройствах и браузерах, были обнаружены и исправлены ошибки.

**2 Практическая часть**

**2.1 Архитектура проекта**

Архитектура проекта была задумана таким образом, чтобы обеспечить четкую и логичную структуру приложения. В корневой папке был создан файл modules.ts, в котором содержались все текстовые константы, используемые в action и reducer, а также интерфейсы для определения формата данных в приложении. Это позволило уменьшить количество ошибок при написании кода.

Далее, в папке redux были созданы файлы, где хранился глобальный стор, а также все actions и reducers. Использование Redux обеспечивало удобное хранение состояния приложения и позволяло легко расширять его функциональность в будущем.

В папке pages были храниться все страницы приложения, в которых использовались компоненты из папки components. Компоненты были разбиты по логическим блокам, таким как header, footer, book, form, и т.д., что облегчало их использование и повторное использование в других частях приложения.

Таким образом, архитектура проекта была спроектирована так, чтобы обеспечить чистый и понятный код, удобное хранение состояния приложения и легкое масштабирование функциональности.

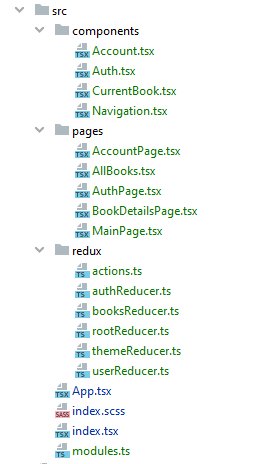


Рисунок 5 − Архитектура папок и файлов в проекте

В проекте был применен подход чистых функций в Redux, где все reducers были реализованы как чистые функции без побочных эффектов. Этот принцип показан на рисунке 6. Это позволило обеспечить стабильное и предсказуемое поведение приложения, а также упростить тестирование и отладку кода. Использование чистых функций помогло избежать проблем, связанных с разделяемым состоянием, такими как гонки данных и блокировки, и обеспечило более предсказуемое и удобное использование кода.

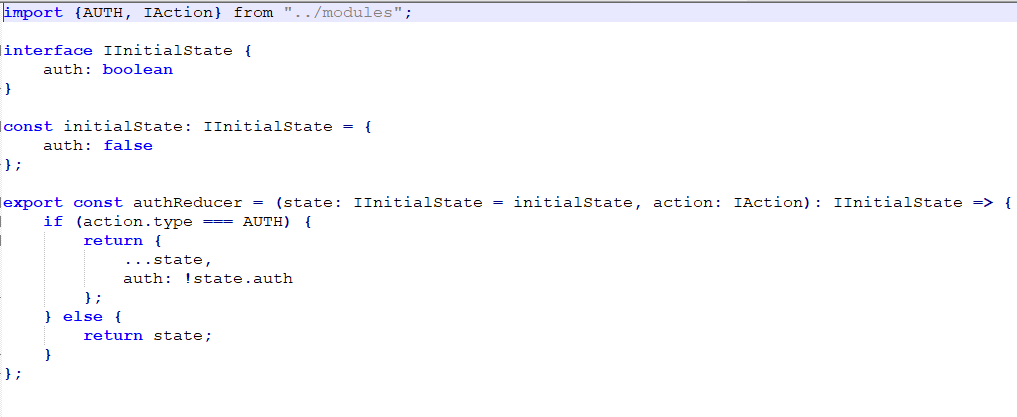


Рисунок 6 − Принцип чистых функций

Также во время разработки приложения использовался принцип тупых компонентов, которые отвечают только за отображение данных, не содержат внутри себя логики и не имеют своего собственного состояния. Такой подход к проектированию компонентов позволяет упростить код, сделать его более понятным и легким для поддержки. Этот принцип показан на рисунке 7.



Рисунок 7 − Принцип тупых компонент

При использовании тупых компонентов логика обработки данных или управления состоянием приложения остается на более высоких уровнях иерархии компонентов, что позволяет избежать появления ошибок в коде. Кроме того, такие компоненты не производят побочных эффектов, что снижает вероятность появления проблем при работе с ними.

Использование тупых компонентов в React является хорошей практикой и позволяет создавать более удобное и надежное приложение.

**2.2 Разработка интерфейса**

Разработка интерфейса програмы происходила по определенному процессу. Сначала составился подробный план разработки, где были определены функциональные и нефункциональные требования к программе, ее цели и задачи, а также ее целевая аудитория. Затем проектируется пользовательский интерфейс (UI) и пользовательский опыт (UX) с помощью специализированных программных средств, таких как Figma. На рисунках 8 и 9 изображены примеры интерфейса.

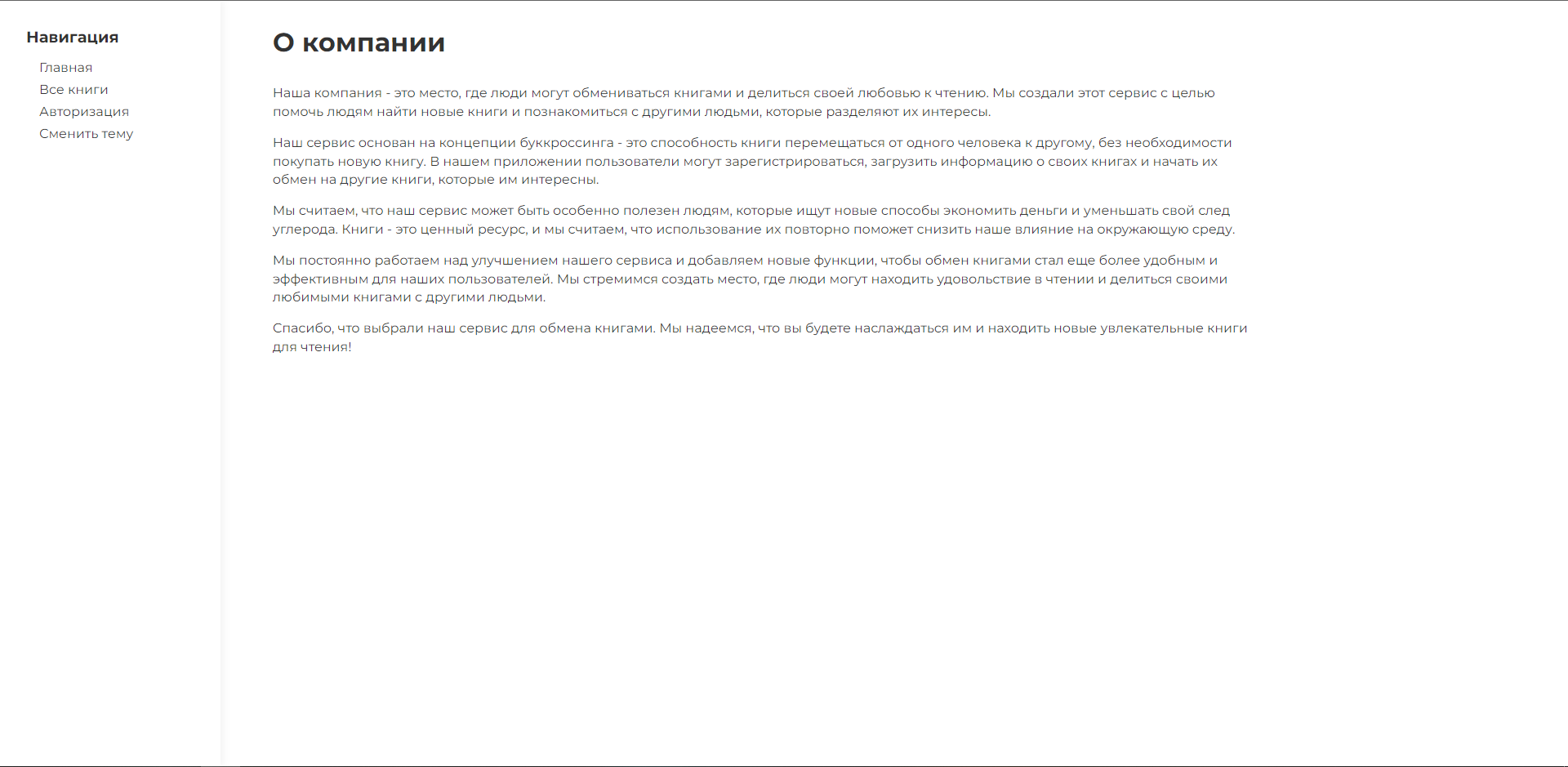


Рисунок 8 − Дизайн страницы о компании

Для обеспечения согласованности и качества интерфейса, обычно были созданы библиотеки компонентов, которые могут быть использованы в различных проектах. В этих компонентах определяются стили, расположение элементов и поведение интерфейса.

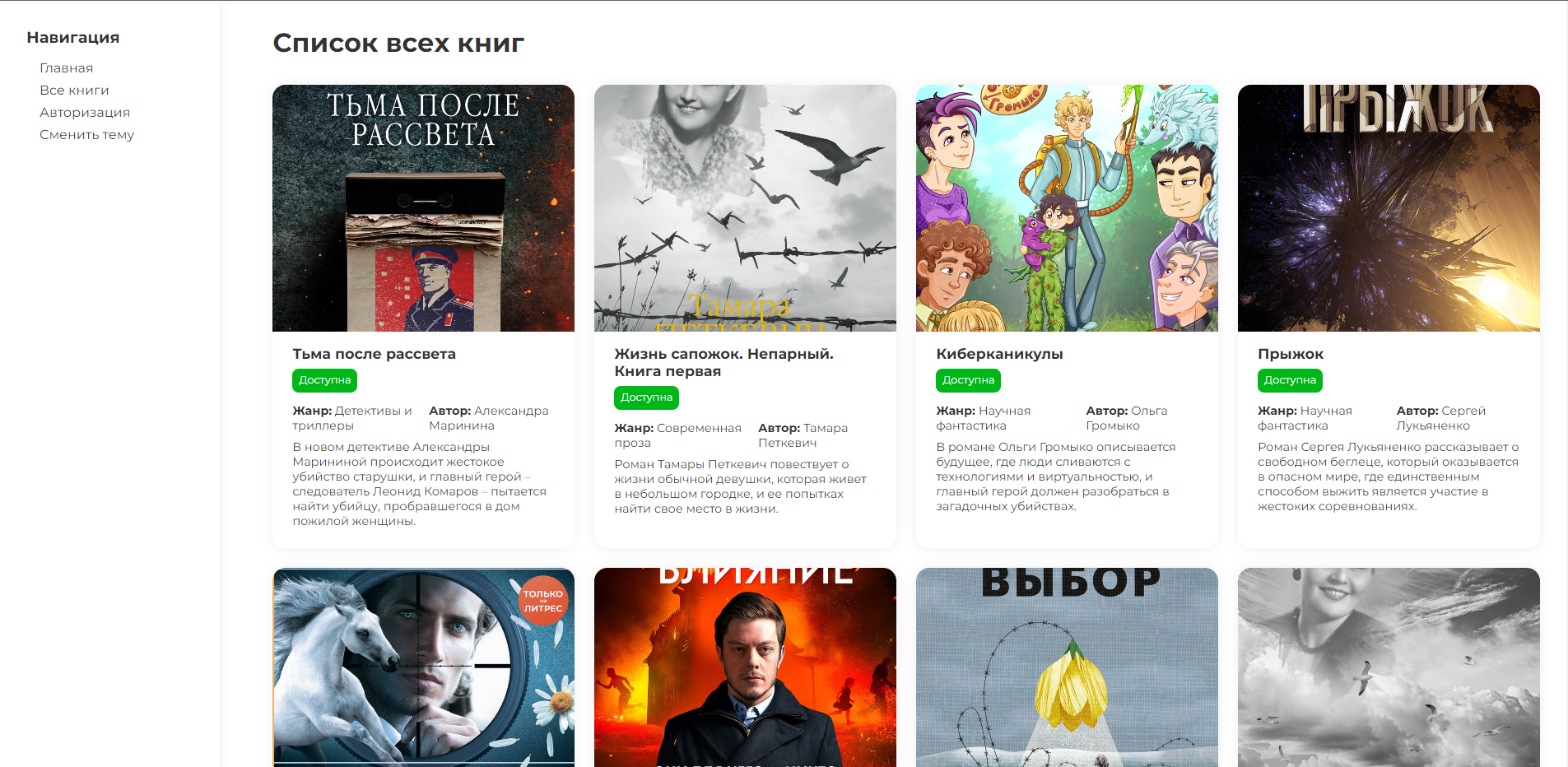


Рисунок 9 − Дизайн страницы для вывода всех книг

В процессе разработки приложения по буккросингу и обмену книгами были созданы различные компоненты, включая компонент для отображения книги, компонент формы с элементами ввода данных, а также компонент плашки, который информирует пользователя, доступна ли книга для обмена или нет.

Создание библиотеки компонентов помогает обеспечить согласованность и качество интерфейса в различных проектах. В такой библиотеке определяются стили, расположение элементов и поведение интерфейса. Это также позволяет значительно сократить время разработки и упростить процесс тестирования и обновления интерфейса.

**2.3 Разработка базы данных**

Когда дело касается разработки базы данных для проекта, необходимо определить, какая технология будет использоваться для хранения и получения данных. В данном проекте было решено использовать json-server в формате json для создания базы данных.

Json-server - это библиотека для Node.js, которая позволяет быстро и легко создать RESTful API на основе JSON файлов. Она позволяет создавать и запускать локальные серверы с фиктивными данными, что очень удобно при разработке и тестировании приложений.

Для получения данных из json-server в проекте был использован axios. Он является мощной библиотекой для работы с HTTP-запросами и поддерживает типизацию от TypeScript. Это позволяет точно определить, какие данные вы ожидаете получить от сервера, что улучшает надежность и читаемость кода.

Для обеспечения типизации данных, которые приходят с сервера, были созданы специальные интерфейсы в файле modules.ts показанные на рисунке 10 и 11. Они определяют структуру данных, что позволяет использовать эти данные в проекте без ошибок. Это также упрощает понимание того, какие данные будут получены и какие свойства они имеют.

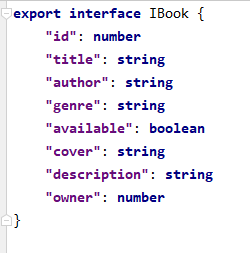


Рисунок 10 − Интерфейс IBook

Таким образом, использование json-server и axios позволило создать базу данных для проекта и получать данные из нее с помощью HTTP-запросов. Использование интерфейсов позволило обеспечить типизацию и повысить надежность проекта.

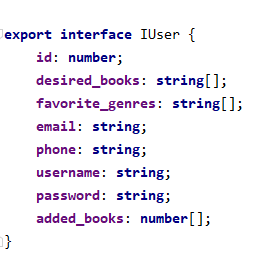


Рисунок 11 − Интерфейс IUser

**2.4 Разработка API**

Как уже было сказано, Json-server предоставляет RESTful API, который позволяет получать данные из локального файла в формате JSON. По умолчанию сервер запускается на порту 3000, но в данном проекте был использован порт 3004. Он принимает запросы GET, POST, PUT, PATCH и DELETE

Однако, если необходимо получить данные из других файлов в формате JSON, можно использовать ключевое слово "routes" в файле db.json и задать путь к этим файлам. Это хорошее качество при масштабировании приложения.

В данном проекте имеются две ссылки: http://localhost:3004/users и http://localhost:3004/books. По умолчанию, происходит GET-запрос, но можно также использовать POST, PUT, PATCH и DELETE, для добавления, обновления и удаления данных.

GET-запрос позволяет получать данные из сервера по заданному URL. Например, GET-запрос к http://localhost:3004/users вернет список пользователей, а GET-запрос к http://localhost:3004/books вернет список книг. Пример того что вернет этот запрос можно увидеть на рисунке 12 и 13.

POST-запрос используется для создания новых данных на сервере. Например, POST-запрос к http://localhost:3004/users создаст нового пользователя, а POST-запрос к http://localhost:3004/books создаст новую книгу.

PUT-запрос используется для обновления существующих данных на сервере. Например, PUT-запрос к http://localhost:3004/users/:id обновит данные пользователя с определенным идентификатором, а PUT-запрос к http://localhost:3004/books/:id обновит данные книги с определенным идентификатором.

PATCH-запрос используется для частичного обновления существующих данных на сервере. Например, PATCH-запрос к http://localhost:3004/users/:id обновит только те данные пользователя, которые были указаны в запросе, а PATCH-запрос к http://localhost:3004/books/:id обновит только те данные книги, которые были указаны в запросе.

DELETE-запрос используется для удаления существующих данных на сервере. Например, DELETE-запрос к http://localhost:3004/users/:id удалит пользователя с определенным идентификатором, а DELETE-запрос к http://localhost:3004/books/:id удалит книгу с определенным идентификатором.

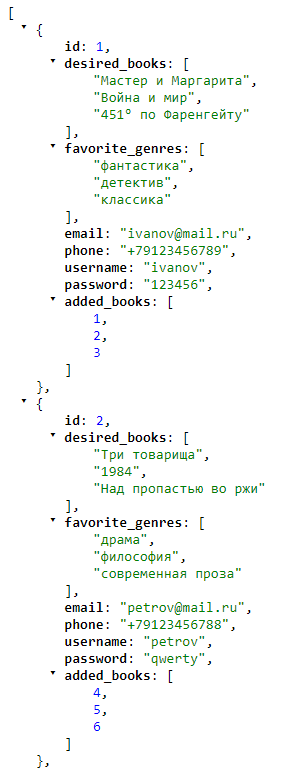


Рисунок 12 − GET запрос к users



Рисунок 13 − GET запрос к books

**2.5 Разработка программного кода**

Перед началом написания кода необходимо было установить NodeJS. Node.js это среда выполнения JavaScript, которая позволяет запускать JavaScript код на стороне сервера. Он основан на движке V8, который также используется в браузере Google Chrome.

Node.js предоставляет множество возможностей для разработки веб-приложений, включая следующие:

- создание серверов и API: Node.js позволяет создавать серверы и API на JavaScript. Это позволяет разработчикам использовать один язык программирования для написания как клиентского, так и серверного кода.

- обработка файлов: Node.js предоставляет возможности для обработки файлов, включая чтение, запись, удаление и переименование файлов.

- обработка потоков: Node.js имеет встроенную поддержку потоков, что позволяет эффективно обрабатывать ввод и вывод данных.

- работа с базами данных: Node.js поддерживает множество баз данных, включая MongoDB, MySQL и PostgreSQL.

- сборка инструментов: Node.js используется для сборки инструментов разработчиков, таких как grunt, gulp и webpack.

- разработка веб-приложений в реальном времени: Node.js используется для создания веб-приложений в реальном времени, таких как чаты, онлайн-игры и т.д.

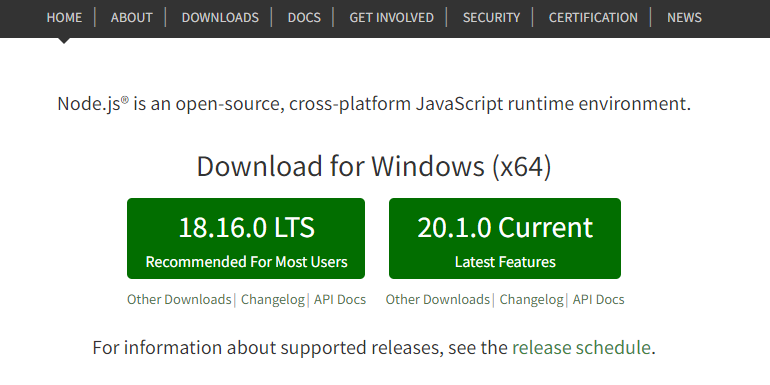


Рисунок 14 − Официальный сайт Node JS

Далее было необходимо установить все нужные пакеты и библиотеки, для этого использовались команды, показанные на рисунках 15, 16, 17. После этого можно было переходить к созданию файлов для сайта.

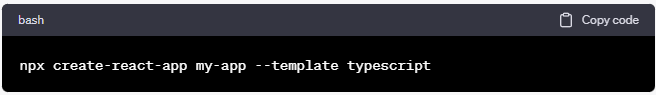


Рисунок 15 − Команда для установки React TypeScript

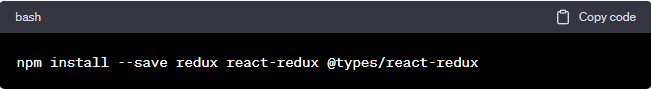


Рисунок 16 − Команда для установки Redux

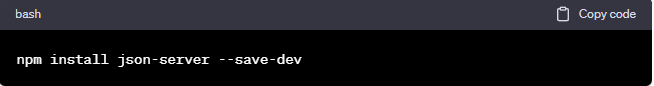


Рисунок 17 − Команда для установки JsonServer

Следующим шагом идет создание базы данных в которой будут храниться все данные о книгах и пользователях. Для того чтобы ее создать необходимо создать файл в формате json. В данной работе файл базы данных назывался db.json и содержал информацию показанную на рисунке 18

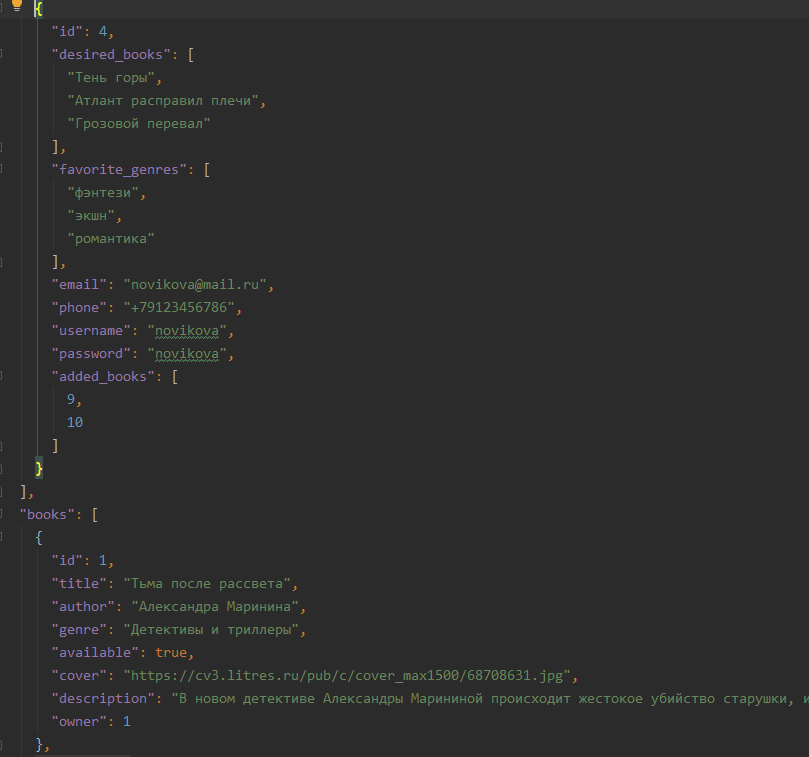


Рисунок 18 − Данные в базе данных

После того как сформированы поля и их типы, для того чтобы избежать ошибки был создан файл modules.ts, в котором прописывались все переменные и интерфейсы для приложения. Это сделано для избежания ошибок и дублирования кода. На рисунке 19 показана часть данных

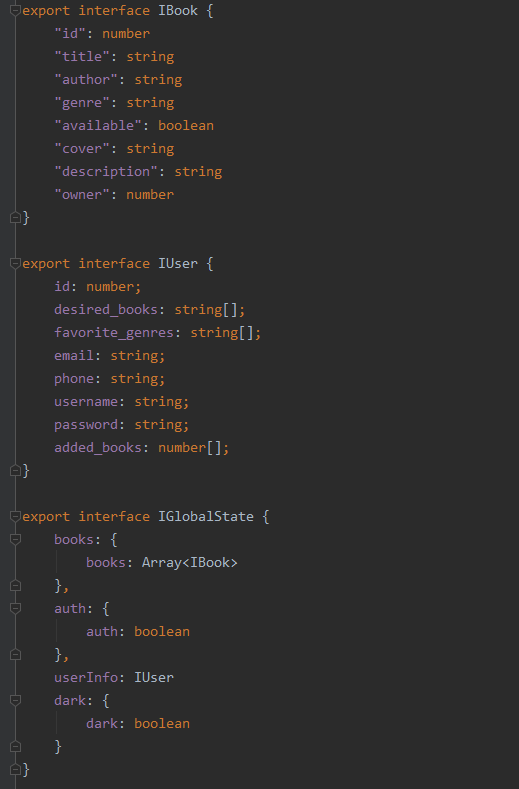


Рисунок 19 − Пример интерфейсов

После все эти данные использовались в приложении и при необходимости что-то изменить, это можно будет сделать в одном месте.

Следующим шагом было создание верстки для главной страницы. В данном проекте главной страницей является страница о компании, это сделано для того чтобы каждый новый пользователь сразу мог ознакомиться с ценностями и миссией компании, а также понимал правила использование сервисом. На рисунке 20 изображена верстка главной страницы.

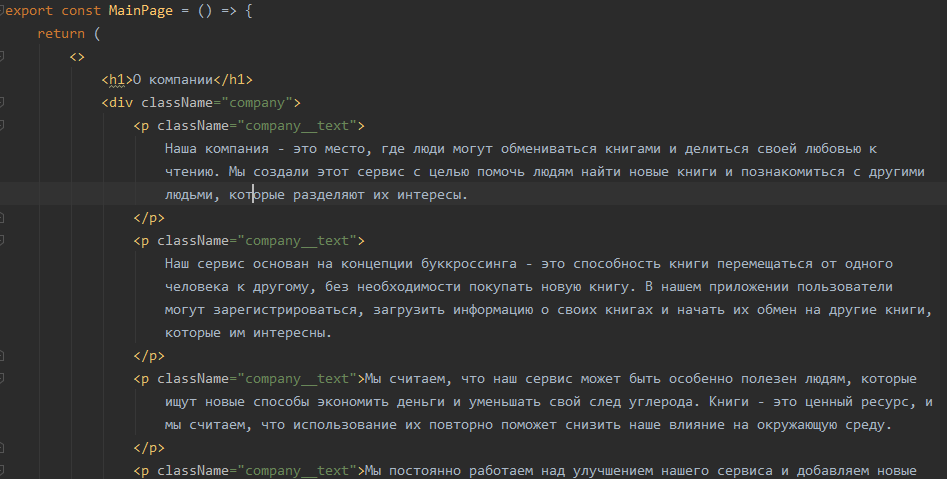


Рисунок 20 − Часть верстки главной страницы

Можно увидеть, что код это HTML код. Это так, но есть некоторые отличия. Так как этот код возвращается в теле JavaScript кода, над ним нужно произвести некоторые операции, чтобы он мог быть выведен на экраны пользователя. Такая технология называется JSX.

Любой посетитель сайта может увидеть список всех доступных книг на платформе, а также воспользоваться специальным фильтром для того чтобы найти необходимую ему книгу. Для этого была создана специальная страница под названием AllBooks. Эта страница является не простой, так как она получает данные из базы данных и рендерит их в необходимой части страницы, а также при помощи функции фильтра показывает только нужные результаты. Код страницы можно увидеть на рисунках ниже.



Рисунок 21 − Вывод всех книг на страницу



Рисунок 22 − Код верстки фильтра

Ниже представлен пример кода, который отвечает за фильтрацию книг и вывод их на страницу.



Рисунок 23 − Логика для фильтрации элементов

Однако для того чтобы добавить страницу на сервис необходимо зарегистрироваться. Для этого была создана специальная страница AuthPage.tsx.

На ней расположены две главных компоненты. Первая для пользователей, у которых уже есть аккаунт, вторая для тех, кто хочет создать новый аккаунт.

Так как при написании кода был использован стиль «тупых компонент» то всю логику необходимо было вынести в отдельное место, а верстку в соответствующие компоненты. Поэтому в AuthPage содержится вся логика, которая используется при авторизации и регистрации пользователей. Подробнее на рисунках 24 и 25.

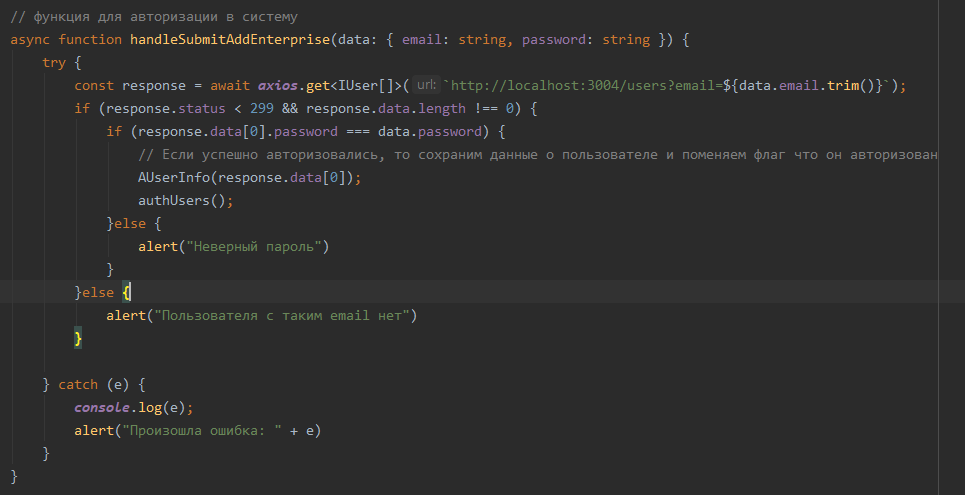


Рисунок 24 − Функция для авторизации в систему

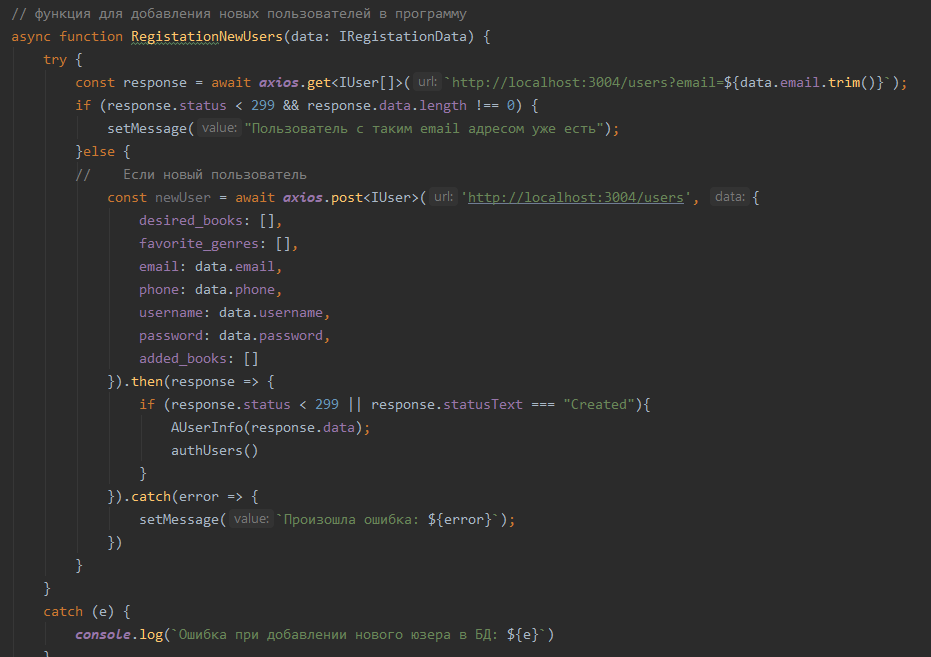


Рисунок 25 − Функция для добавления новых пользователей

Сама страница для регистрации выглядит следующим образом.

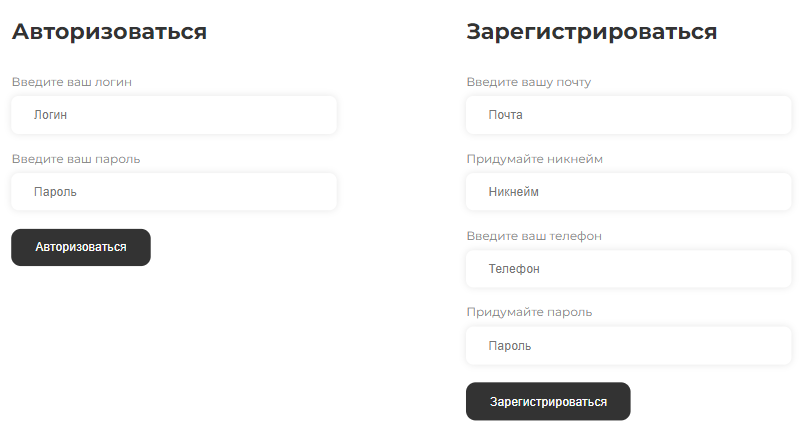


Рисунок 26 − Страница для регистрации и авторизации

После того как пользователь авторизуется или создаст новый аккаунт, у него измениться навигация сайта, появятся новые ссылки, а также его автоматически переадресует на страницу с его аккаунтом

В самом начале аккаунт будет выглядеть как показано на рисунке 27. Но после того как он добавит новые книги, в его аккаунте сразу обновиться вся информация. Также он сможет увидеть свою книгу на странице «все книги»

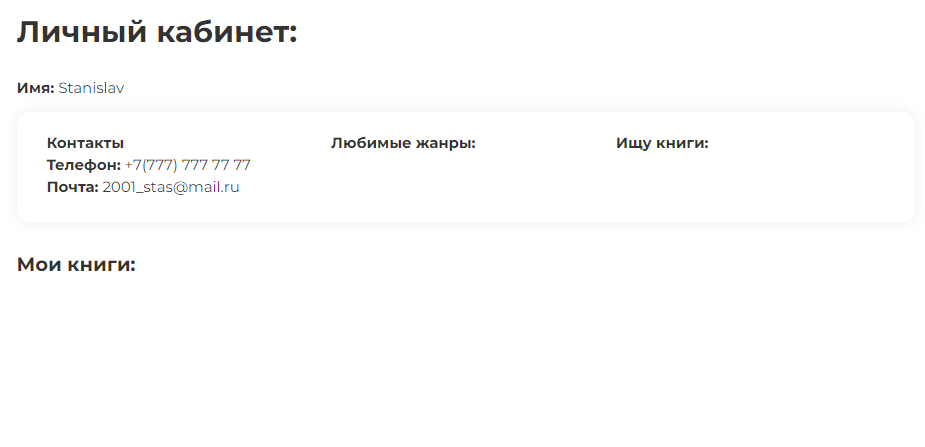


Рисунок 27 − Пустой личный кабинет

Для того чтобы добавить книгу нужно перейти на соответствующую страница. Она станет доступна только после того как пользователь авторизуется. Если же он попытается попасть на страницу не авторизованным, например, по прямой ссылке, то он увидит сообщение, показанное на рисунке 28.

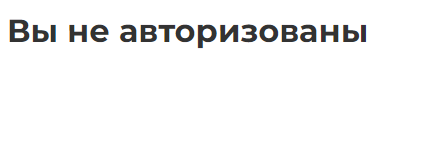


Рисунок 28 − Защита платформы

Если же пользователь авторизован, то он увидит страницу, в которой может добавить свою книгу. Как она выглядит показано на рисунке 29.

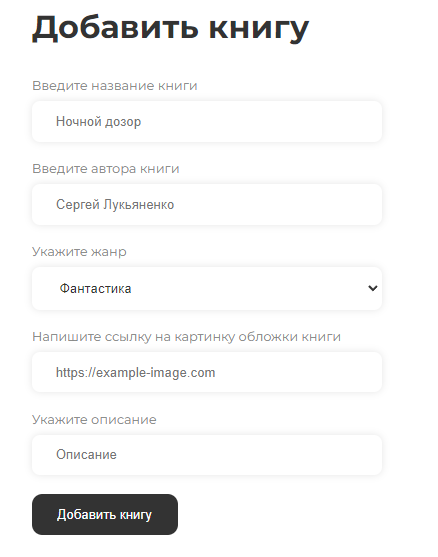


Рисунок 29 − Страница добавления книги

Если пользователь ввел все данные верно и не произошло никаких ошибок, пользователь увидит сообщение «книга успешно добавлена»

Теперь если перейти на страницу «мой аккаунт» можно увидеть все свои добавленные книги. Пример на рисунке 30.

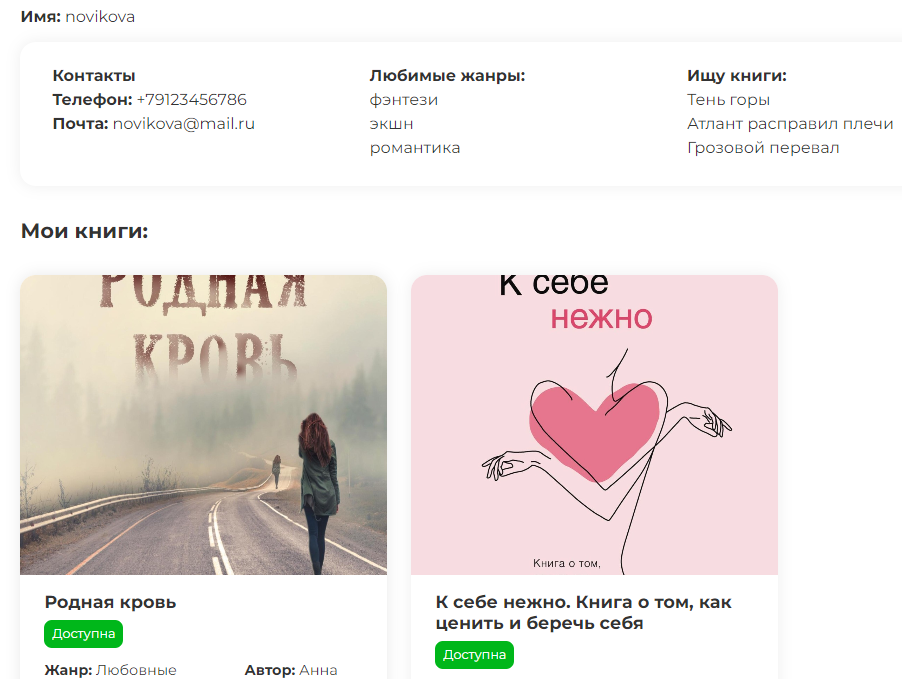


Рисунок 30 − Автоматическое обновление данных

Теперь любой пользователь сможет зайти на страницу «все книги» и увидеть книги которую вы добавили. Если он его заинтересует, то он может нажать на нее, для того чтобы перейти на страницу конкретной книги и увидеть подробное описание и контакты владельца. Как это реализовано показано на рисунке 31

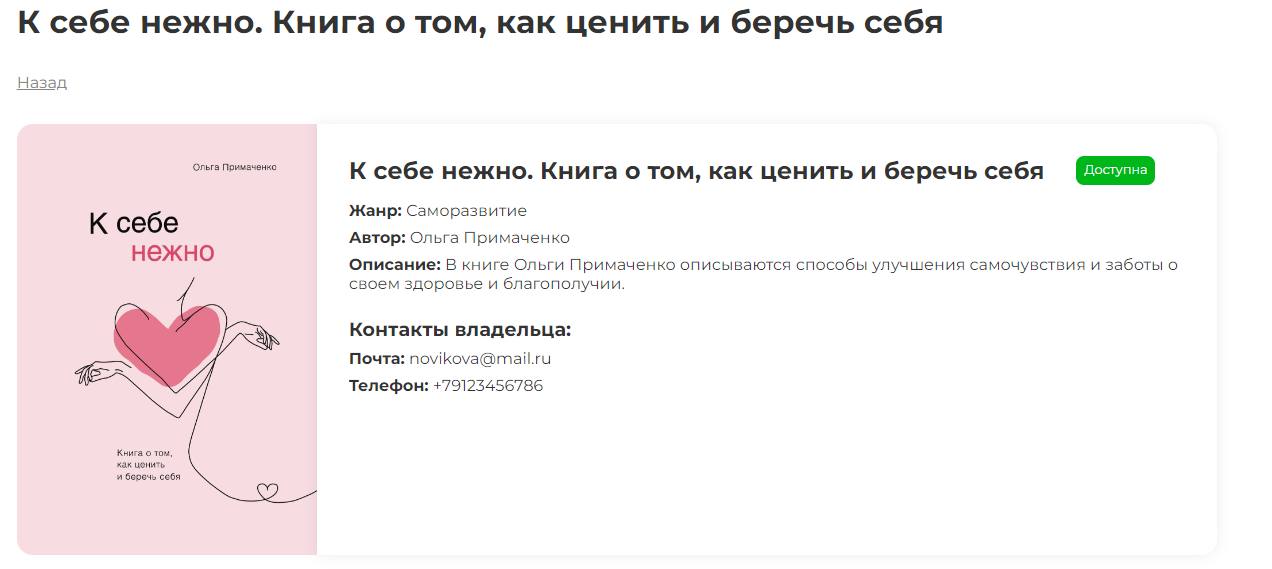


Рисунок 31 − Страница конкретной книги

**Заключение**

В данной работе была разработана платформа по буккроссингу и обмену книгами на основе современных технологий, таких как React JS, TypeScript, Redux и Json-server. Благодаря использованию этих технологий, пользователи могут быстро находить нужные книги, ознакамливаться с их основными характеристиками и связываться с владельцами книг, что создает удобную платформу для обмена книгами между любителями чтения и способствует распространению книжной культуры.

Целью данной работы было научиться новым и современным технологиям, которые используются в компаниях мирового уровня для написания больших сайтов и приложений, а также создать работу и добавить ее в профессиональное портфолио, для последующего показа работодателям в качестве примера и демонстрации навыков. В ходе работы было продемонстрировано умение применять современные технологии и инструменты для создания крупных веб-приложений, а также умение создавать удобный пользовательский интерфейс.

Таким образом, данная работа позволила не только изучить новые технологии, но и создать полезный и функциональный проект, который может быть использован в реальной жизни. Созданная платформа может стать примером для других разработчиков и продолжить свое развитие в будущем.

**Список использованных источников**

1 Adam Boduch. React and React Native: A Complete Hands-On Guide to Modern Web and Mobile Development with React.js, 4th Edition. Packt Publishing, 2022. − 586 с.

2 David Khourshid. Essential TypeScript: From Beginner to Pro. Apress, 2021. − 442 c.

3 Эрик Эванс. Предметно-ориентированное проектирование. Addison Wesley, 2003. − 656 c.

4 Борис Барский. React. Быстрый старт. Первое знакомство и создание приложения. Питер, 2021. − 240 c.

5 Официальная документация React JS [https://reactjs.org/]

6 Официальная документация TypeScript [https://www.typescriptlang.org/docs/

7 Официальная документация Redux [https://redux.js.org/

**Приложение A**