**ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ НАУК**

**Отчет по проектному семинару «Языки и системы программирования»**

На тему

«Разработка книжного интернет-магазина»

Выполнила**:**

студентка 2 курса 1 группы ИСиТ

Сушкова Дарья Сергеевна

Воронеж, 2024

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc168693833)

[1 Главная страница 4](#_Toc168693834)

[1.1 Структура главной страницы 4](#_Toc168693835)

[1.2 Выполнение JavaScript скриптов на главной странице 5](#_Toc168693836)

[2 Страница корзины 11](#_Toc168693837)

[2.1 Структура страницы корзины 11](#_Toc168693838)

[2.2 Выполнение JavaScript скриптов для страницы корзины 12](#_Toc168693839)

[3 Страница пользователя 15](#_Toc168693840)

[3.1 Структура страницы пользователя 15](#_Toc168693841)

[3.2 Выполнение JavaScript скриптов для страницы пользователя 15](#_Toc168693842)

[4 «Шапка» страницы 18](#_Toc168693843)

[ЗАКЛЮЧЕНИЕ 21](#_Toc168693844)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 22](#_Toc168693845)

ВВЕДЕНИЕ

Целью данного проекта является разработка веб-приложения для книжного интернет-магазина с использованием JavaScript.

Основными задачами для достижения поставленной цели можно выделить:

* Изучение основ синтаксиса языка JavaScript для написания основных функций приложения;
* Создание интерактивного и комфортного пользовательского интерфейса для взаимодействия с приложением;
* Обеспечение синхронизации данных между страницами приложения.

JavaScript — это полноценный динамический язык программирования, который широко используется для создания интерактивных и динамичных веб-страниц [1]. Применяемый к HTML-документам, JavaScript позволяет оживить веб-сайты, обеспечивая их взаимодействие с пользователями в реальном времени.

JavaScript был выбран для написания данного веб-приложения по нескольким причинам. Прежде всего, JavaScript является одним из наиболее распространенных языков программирования в веб-разработке, что обеспечивает обширную поддержку и доступность ресурсов для изучения и разработки. Кроме того, JavaScript обладает широким спектром возможностей для создания интерактивных и динамичных пользовательских интерфейсов, что важно для создания приложения с функциональной и привлекательной пользовательской частью.

1. Главная страница
   1. Структура главной страницы

На рисунке 1 представлено «дерево» тегов для главной страницы из файла main-page.html, оно образует его структуру. Для каждого тега можно указать класс, а при помощи них можно взаимодействовать с элементами при помощи JavaScript и CSS.



Рисунок 1 – DOM-дерево главной страницы

Согласно рисунку 1, ключевыми элементами страницы являются элемент класса header, который представляет из себя шапку сайта, а также блочный элемент shop-wrapper, который является оберткой для всего контента, содержащегося на главной странице [3].

Основной контейнер (shop-wrapper) cодержит все элементы, связанные с магазином. Он разделен на две основные части: панель с акциями и панель с товарами.

Панель с акциями (discount-panel) предназначена для отображения рекламных баннеров, которые информируют посетителей о текущих акциях и скидках. Внутри этой панели размещены карточки с акциями (discount-card). Каждая карточка содержит ссылку на соответствующую акцию и изображение баннера, что позволяет пользователю быстро перейти к интересующей акции.

Панель с товарами (shop-panel) является центральной частью магазина, где представлены все товары. Эта панель включает в себя два основных блока: блок фильтров (shop-filters) и блок с книжным контентом (shop-content).

* 1. Выполнение JavaScript скриптов на главной странице

Подключение JavaScript (далее – JS) скрипта (сценария) для главной страницы происходит в main-page.html и выглядит следующим образом: <script *type*="module" *defer* *src*="../js/main\_script.js"></script> [3]. Данная строка может быть вставлена в любую часть документа. JS код выполняется последовательно, инструкция за инструкцией, т.к. JavaScript – интерпретируемый язык программирования. Инструкция – это синтаксическая единица языка программирования, выражающая действие, выполняемое программным кодом.

Вначале в файл импортируется массив книг listOfBooks из файла module.js. Данный массив создается на основе объектов, созданных с помощью класса createBook, который в качестве параметров конструктора принимает имя (name), категории (category), происхождение (origin), тип обложки (coverType), доступность к покупке (availability), ссылку на изображение книги (imgLink), цену (price) и новую цену (newPrice).

Самой первой выполняет инструкции функция launchBooks(), которая отвечает за отрисовку актуальных книг. При первой загрузке интернет-магазина будут отрисовываться все книги из массива listOfBooks – это происходит посредством метода массива forEach().

При последующих вызовах функции launchBooks() (например, после применения фильтров), с самого начала будет очищаться весь контент в контейнере shop-content, а только потом заново будут отрисовываться книги, отфильтрованные посредством метода массива filter() – о фильтрации книг будет рассказано подробно далее [4].

Далее для каждой подходящей фильтрам книги выполняется следующее: все параметры данной книги, которые могут учитываться при дальнейшей фильтрации, заносятся в множество filters посредством функции addBookPropertiesToSet() – данный список фильтров пригодится для дальнейшей отрисовки актуальных фильтров. Далее для каждой книги создается контейнер класса product-card, т.е. карточка книги, в который добавляются данные о книге (название, изображение, цена) и кнопка add-to-cart-btn. После применения основных стилей из файла shop\_style.css для product-card и внутренних объектов данного контейнера карточка продукта (книги) выглядит следующим образом:



Рисунок 2 – Карточка товара на главной страницу

Каждый контейнер product-card добавляется на общую «полку» книг – в контейнер класса shop-content.

Далее после отрисовки всех книг выполняется функция initializeCartBtns(), она необходима для добавления обработчиков событий по клику на все кнопки добавления в корзину. Если кнопка не нажата, т.е. кнопка залита фиолетовым цветом и присутствует надпись «Купить», как на рисунке 2, то при нажатии на нее: к контейнеру данной кнопки добавляется класс pressed (что позволяет посредством стилей поменять цвет кнопки на бордовый), текст кнопки меняется на «Оформить», как на рисунке 3, в массив индексов добавленных в корзину книг (cartStates) добавляется индекс соответствующей книги, к общей стоимости товаров в корзине (totalCost) добавляется цена этой книги (причем, если у книги есть акционная цена, то добавляется она), прибавляется количество книг, добавленных в корзину (totalCount), а также на кнопку добавляется новый обработчик события – по клику пользователь переходит на страницу корзины [5].



Рисунок 3 – Карточка товара, добавленного в корзину

SessionStorage – это объект веб-хранилища, позволяющий хранить пары ключ/значение в браузере, которые сохраняются после обновления страницы или после перехода на другую страницу. В отличие от куки, объекты веб-хранилища не отправляются на сервер при каждом запросе, поэтому в них можно хранить гораздо больше данных. Стоит отметить, что totalCost, totalCount и cartStates добавляются в sessionStorage при помощи метода setItem, например: sessionStorage.setItem('total-cost', totalCost).

При переходе на другую или при обновлении текущей страницы в начале работы функции initializeCartBtns() срабатывает функция initializePressedCartBtns(), которая позволяет добавить обработчики клика на уже нажатые кнопки – при нажатии на эти кнопки пользователь перейдет на страницу корзины [6]. Такое возможно из-за сохранения cartStates в sessionStorage. Т.е. если пользователь, например, добавил несколько книг в корзину и перезагрузил главную страницу, то в функции launchBooks() для каждой книги пройдет проверка, содержится ли она в cartStates или нет, т.е. была ли до перезагрузки страницы она добавлена в корзину или нет. Если окажется, что книга была добавлена в корзину, то к контейнеру кнопки добавляется класс pressed, и карточка товара принимает вид как на рисунке 3. А затем после отрисовки всех книжных карточек, как и ранее, выполняет инструкции функция initializeCartBtns().

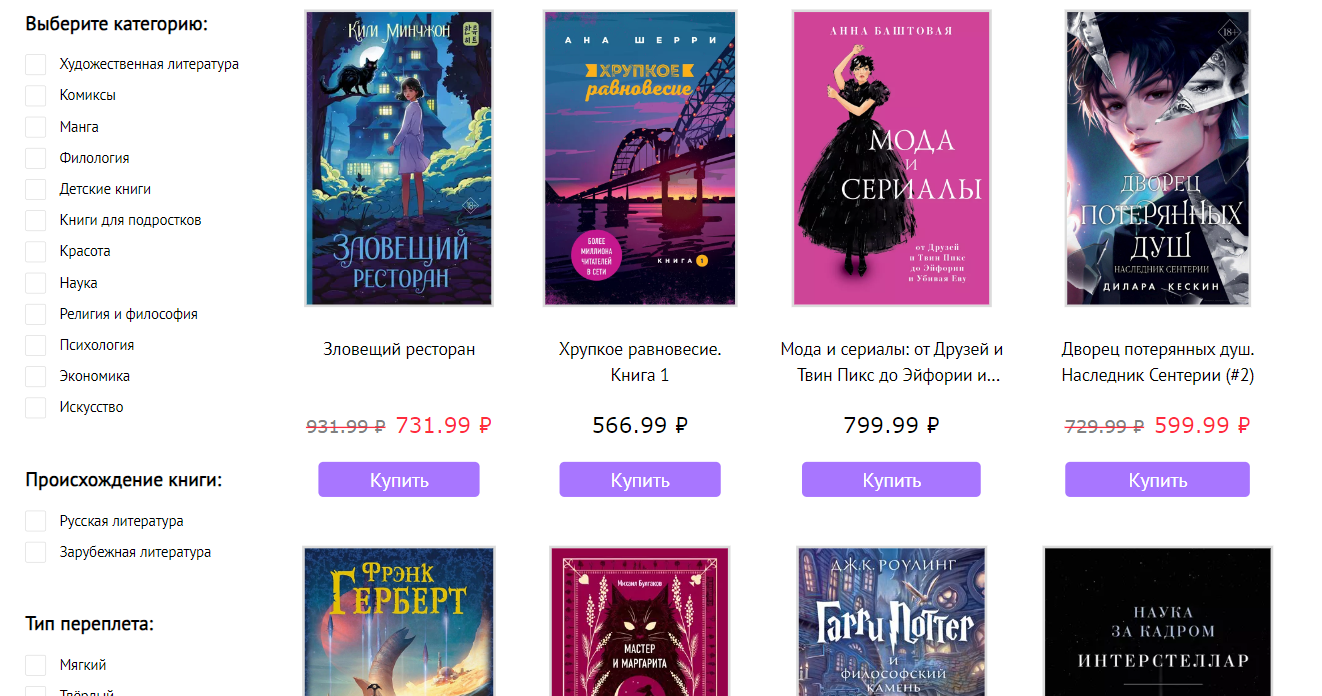
Также после отрисовки всех карточек товаров выполняется функция launchFilters(Array.from(filters)), в которую в качестве аргумента передается преобразованное в массив множество фильтров filters, которое содержит параметры (типы обложек, категории, происхождения и виды доступности), присущие данной выборке книг, а в качестве параметра данной функции выступает relevantFilters. Таким образом, мы передаем после отрисовки текущей выборки книг актуальные (релевантные) для нее фильтры в функцию launchFilters().

По аналогии с функцией launchBooks(), сначала будет очищаться весь контент в контейнере shop-filters, а затем уже заново будут отрисовываться фильтры.

Когда пользователь впервые посещает интернет-магазин, то relevantFilters будет содержать все возможные фильтры книг, поскольку изначально отрисовываются все книги.

Далее для каждой категории фильтров (Категория, Происхождение книги, Тип переплета, Доступность) создается свой контейнер filter-item, в котором и будут находиться все подходящие фильтры – каждый в собственном контейнере checkbox\_item. В каждом checkbox\_item есть инпут-контейнер с уникальным id типа checkbox, а также контейнер checkbox\_text с соответствующим наименованием фильтра.

На рисунке 4 представлен фрагмент главной страницы веб-приложения при первой загрузке пользователем. Фрагмент представляет собой блок shop-panel, включающий в себя блок фильтров слева и блок с книжным контентом справа:

Рисунок 4 – Блок shop-panel 

После добавления всех filter-item в блок фильтров shop-filters последней инструкцией функции launchFilters() выполняется вызов функции initializeFilters(). Ее функциональность заключается в том, чтобы найти все инпут-контейнеры типа checkbox и для каждого из них добавить событие «change» [6].

Как раньше упоминалось, отрисовка блока книг происходит посредством фильтрации. Для этого в файле были созданы пустые массивы categories, origins, coverTypes и availabilities, в которые теперь при изменении состояния чекбокса (отмечен  / не отмечен), соответствующее ему наименование фильтра будет либо удаляться, либо добавляться в соответствующий массив.

Также после каждого изменения состояния любого чекбокса будет заново вызываться функция launchBooks() и отрисовывать только актуальные карточки книг для текущей выборки фильтров.

На рисунке 5 представлен фрагмент главной страницы веб-приложения при задании пользователем фильтров «Книги для подростков» и «Твёрдый тип переплета», а также при добавлении им одной книги в корзину:

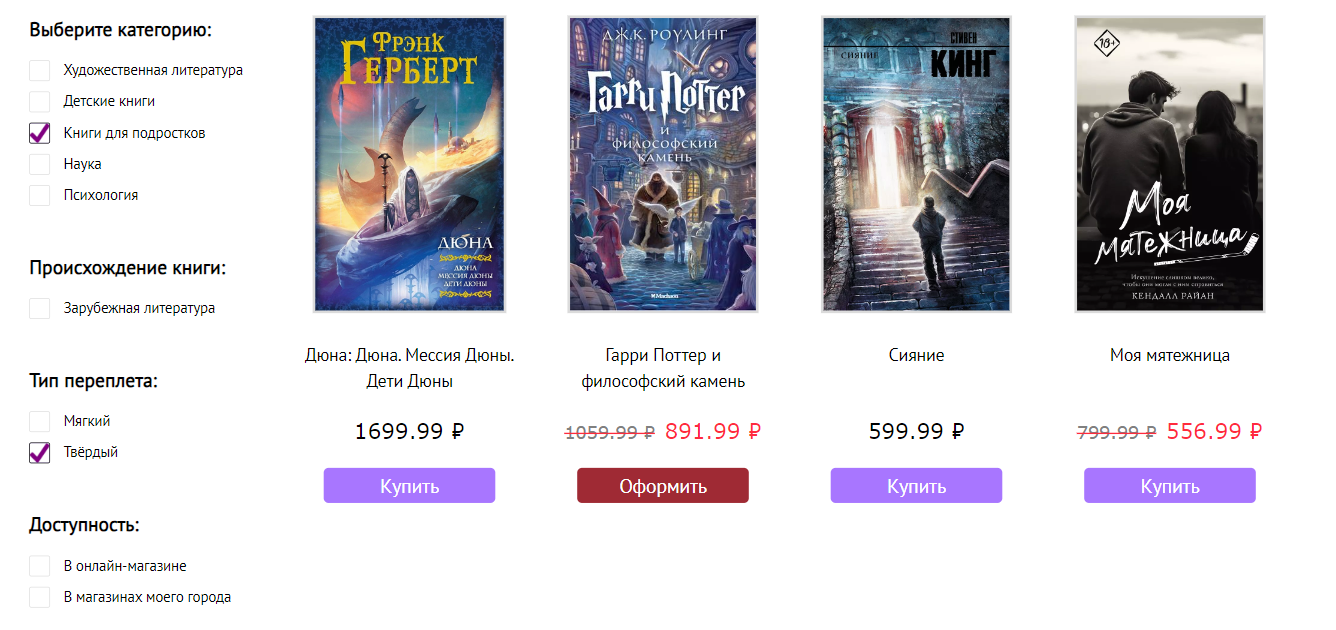


Рисунок 5 – Блок shop-panel при применении фильтров

Таким образом, функционирование главной страницы сайта представляет собой своеобразный цикл: отрисовка блока книг -> отрисовка блока фильтров по параметрам книг -> пользователь выбирает фильтры -> отрисовка блока книг по заданным фильтрам -> отрисовка блока фильтров по параметрам книг из текущей выборки и так далее.

1. Страница корзины
   1. Структура страницы корзины

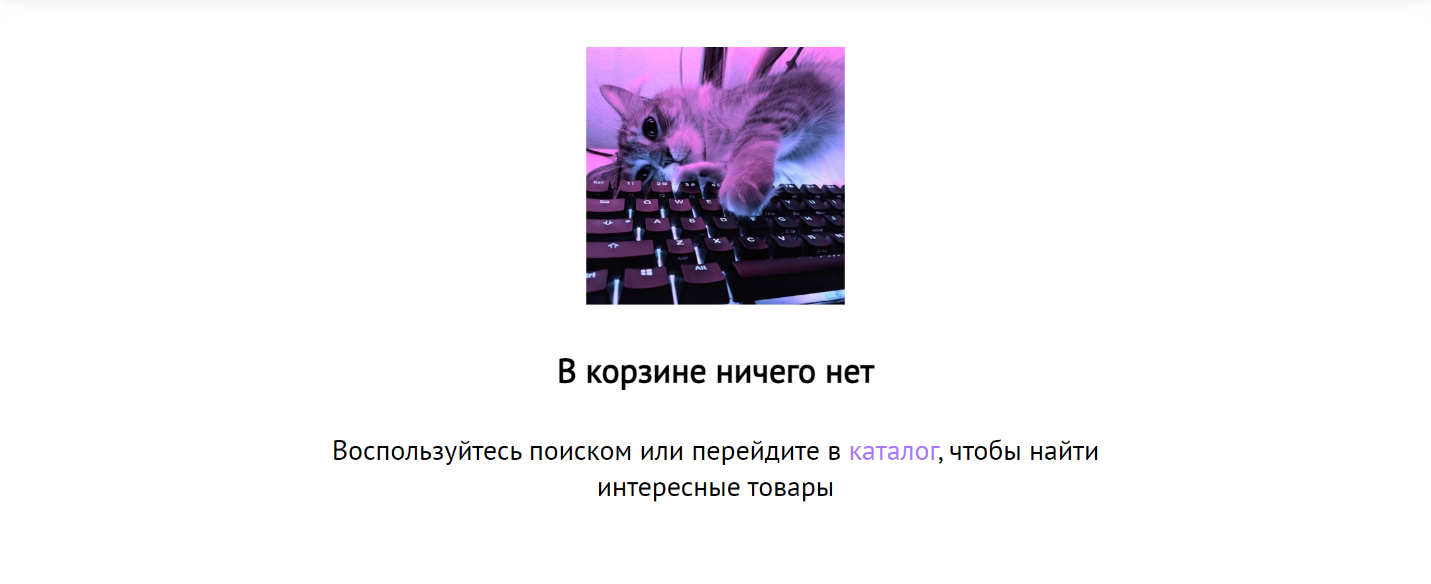
На рисунке 6 представлено структурное дерево тегов для страницы корзины из файла cart-page.html:



Рисунок 6 – DOM-дерево страницы корзины

Согласно рисунку 6 ключевыми элементами страницы являются элемент класса header, про который будет написано подробнее в следующем параграфе, а также блочный элемент cart-wrapper, который является оберткой для всего контента, содержащегося на странице корзины.

Контейнер cart-wrapper имеет два контейнера внутри: cart-panel и empty-cart-panel. Если в корзину не будет добавлено ни одного товара, то станет видимым контейнер empty-cart-panel, и тогда интерфейс корзины примет следующий вид:

Рисунок 7 – Интерфейс пустой корзины

Если в корзину будет добавлен хотя бы один товар, то станет видимым контейнер cart-panel. Основными компонентами cart-panel являются контейнер cart-content, содержащий карточки добавленных книг, и контейнер cart-summary, содержащий информацию о количестве и цене добавленных товаров.

* 1. Выполнение JavaScript скриптов для страницы корзины

Вначале, как и для главной страницы приложения, в файл cart\_script.js импортируется массив книг listOfBooks из файла module.js. Также из хранилища sessionStorage извлекаются общая стоимость (totalCost), количество (totalCount) и массив индексов (cartStates) добавленных в корзину книг.

Первой вызывается функция launchCart(), которая вызывает функции initiateCartProducts() и initiateCartSummary(), если корзина непустая, и вызывает функцию initiateEmptyCart(), если корзина пустая .

В случае вызова функции initiateEmptyCart() пользователь, в соответствии с рисунком 7, увидит сопутствующую информацию об отсутствии товаров в корзине с ссылкой на каталог (главную страницу).

В случае вызова функции initiateCartProducts() для каждого индекса из массива cartStates будет найдена соответствующая книга из массива listOfBooks, и для каждой книги будет создан контейнер product-card с основной информацией про нее, input-объект класса input-count с кнопками «+» и «-» для корректировки количества товара в корзине, а также кнопка delete-btn для удаления товара из корзины [6]. По итогу карточка добавленной книги в корзину выглядит следующим образом:

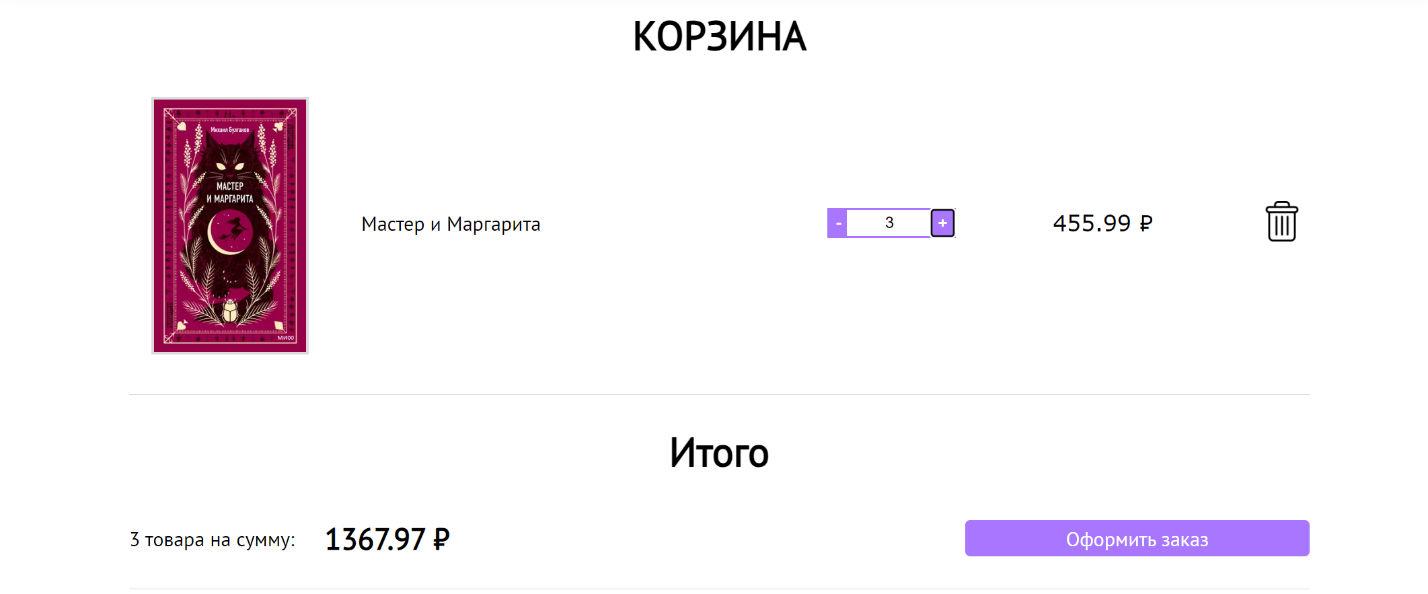
Рисунок 8 – Карточка товара в корзине

В конце эта функция вызывает еще три функции: addPlusCardBtns(), addMinusCardBtns() и addDeleteBtns(), каждая из которых накладывает на соответствующие кнопки обработчики кликов, благодаря которым итоговое количество и стоимость товаров в корзине будут меняться и перезаписываться в хранилище.

Функция initiateCartSummary() создает контейнер класса text-summary, который содержит в себе полученные totalCost и totalCount и кнопку «Оформить заказ», по нажатию на которую срабатывает встроенная js-функция alert(), которая отображает модальное диалоговое окно пользователю с сообщением «Ваш заказ принят в обработку».

Каждый раз, когда пользователь будет нажимать на кнопки «+», «-» и кнопки удаления, будет вызываться функция changeSummary(), которая будет обновлять количество и стоимость товаров в корзине в контейнере text-summary.

На рисунке 9 представлен интерфейс корзины после добавления в неё одной книги и увеличения пользователем количества единиц данного товара:

Рисунок 9 – Интерфейс корзины

Таким образом, функционирование корзины представляет собой последовательность: манипуляции с количеством товаров в корзине -> изменение данных о текущих количестве и стоимости товаров в корзине.

1. Страница пользователя
   1. Структура страницы пользователя

На рисунке 10 представлена разметка страницы пользователя.

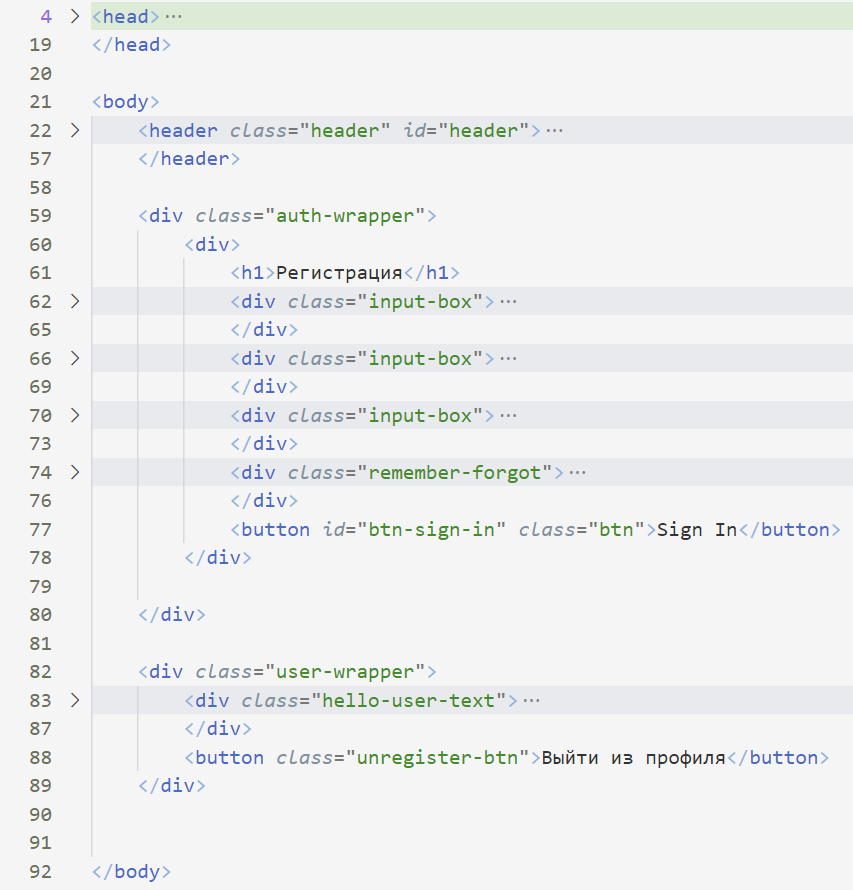


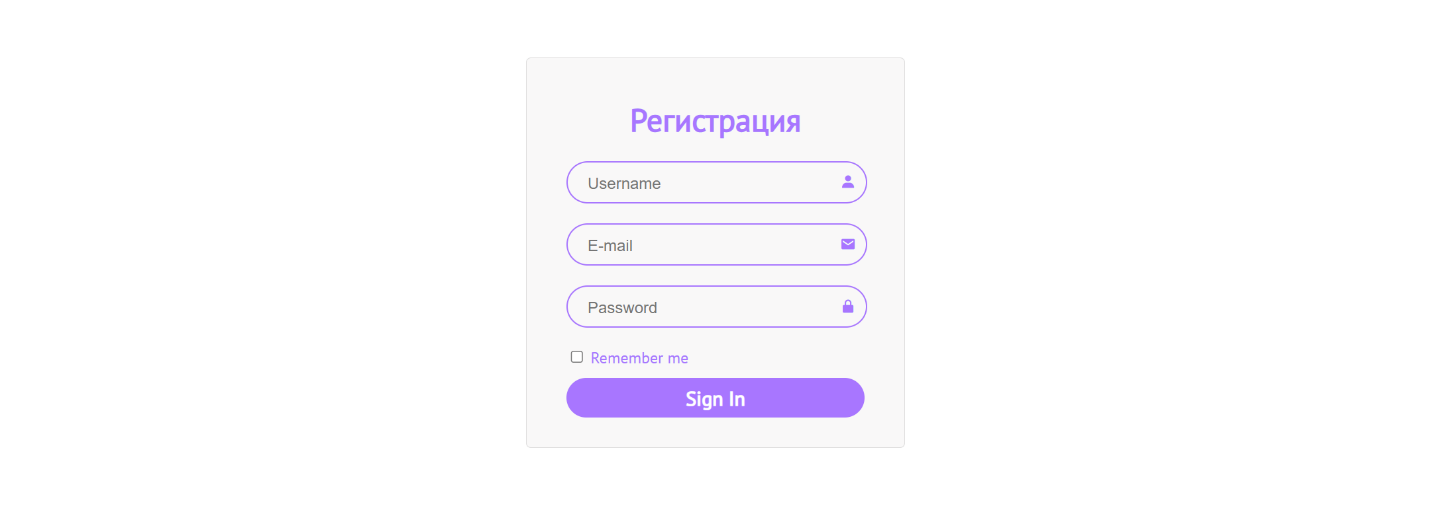
Рисунок 10 – DOM-дерево страницы пользователя

Страница пользователя – это страница, на которой пользователь может зарегистрироваться в книжном интернет-магазине. Основными компонентами данной страницы являются контейнер auth-wrapper, содержащий в себе форму регистрации, и контейнер user-wrapper, необходимый для отображения информации о регистрации пользователя.

* 1. Выполнение JavaScript скриптов для страницы пользователя

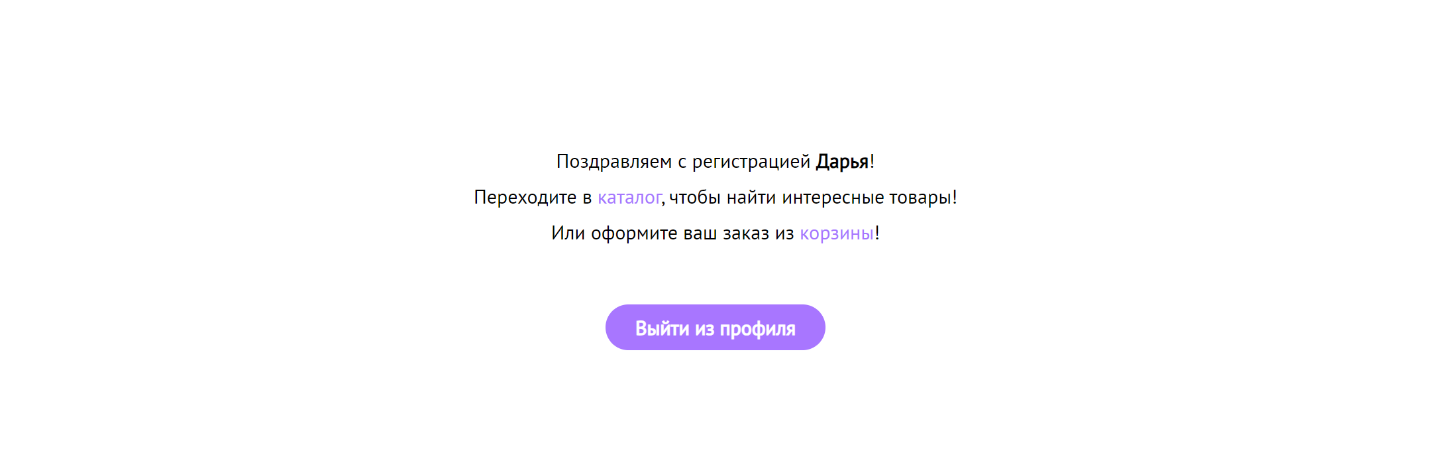
Основные скрипты для данной страницы находятся в файле authorization\_script.js.

Основная логика заключается в отслеживании состояния переменной registered, которая находится в sessionStorage. Если пользователь еще не зарегистрировался или уже вышел из своего аккаунта (registered === false || registered === undefined), то выполняется функция launchSignForm(), контейнер auth-wrapper становится видимым, а user-wrapper наоборот. Таким образом, незарегистрированный пользователь увидит на странице следующее:

Рисунок 11 – интерфейс страницы незарегистрированного пользователя

В соответствии с рисунком 11, пользователь должен заполнить форму и нажать кнопку “Sign Up”, чтобы зарегистрироваться на сайте. Для этого на кнопку поставлен обработчик событий клика. Кликнув на эту кнопку, все данные из формы будут перенесены в js для последующей обработки, имя пользователя будет занесено в sessionStorage для его отображения в header на других страницах, состояние registered будет заменено на true, панель регистрации auth-wrapper перестанет быть видимой, а панель user-wrapper станет, наоборот, видимой для пользователя.

Например, если зарегистрироваться под именем «Дарья», пользователь увидит интерфейс в соответствии с рисунком 12:

Рисунок 12 – интерфейс зарегистрированного пользователя

Теперь, когда пользователь зарегистрирован, во-первых, он увидит свое имя около иконки профиля на всех остальных страницах, во-вторых, ему станет доступна опция «Оформить заказ» в корзине.

Итак, если теперь пользователь снова зайдет на страницу регистрации, то выполнится функция launchText(), которая накладывает обработчик клика на кнопку «Выйти из профиля». Тогда, если пользователь на нее нажмет, то из sessionStorage удалится имя текущего пользователя, состояние переменной registered поменяется на false и перезапишется в sessionStorage, вновь станет видимым контейнер auth-wrapper (форма регистрации), а контейнер user-wrapper станет невидимым.

1. «Шапка» страницы

Неизменной частью всех страниц приложения является header («шапка») - элемент разметки HTML, представляющий верхнюю часть веб-страницы или секции. Общая структура header продемонстрирована на рисунке 10:

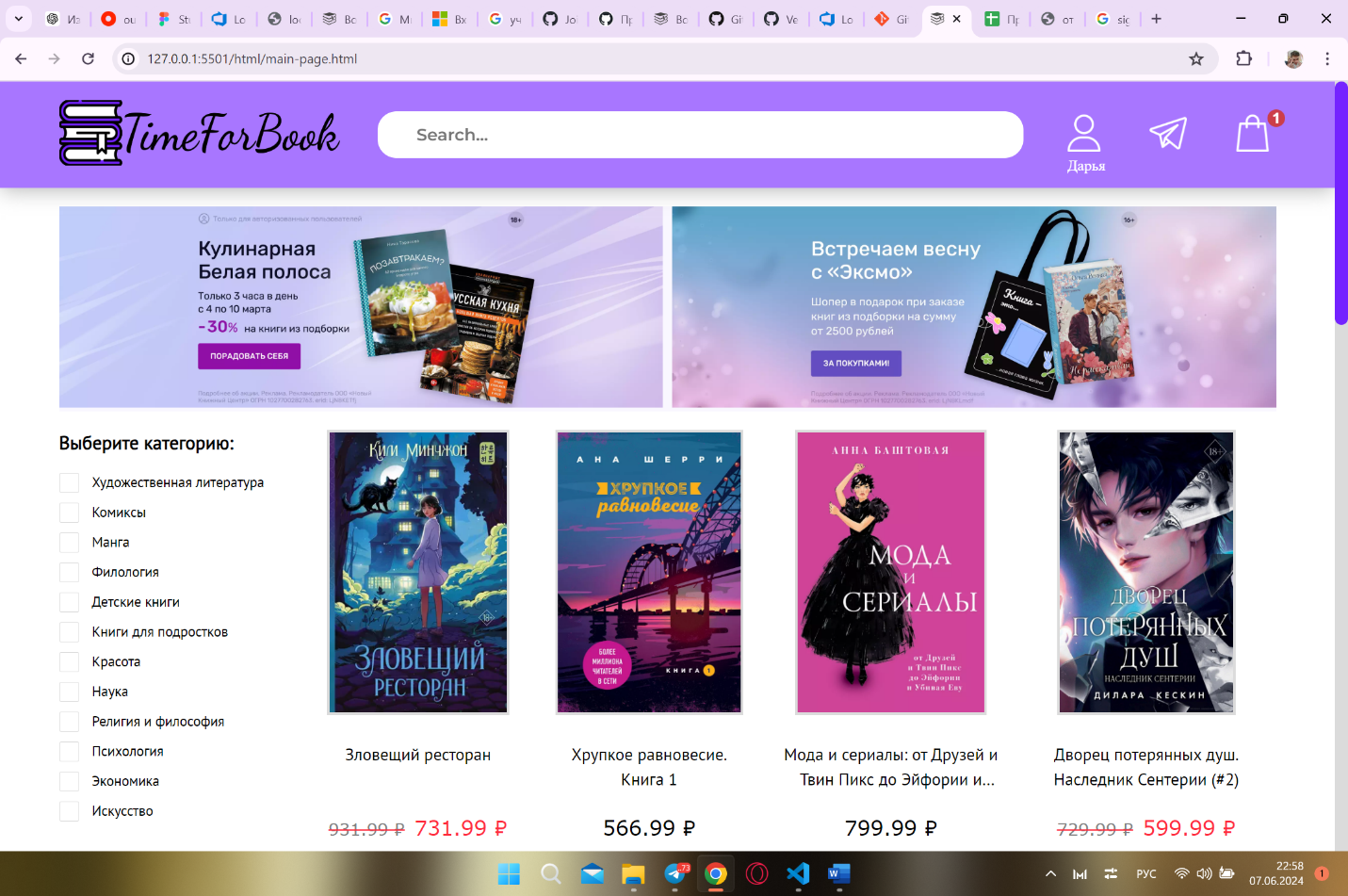


Рисунок 10 – Структура header

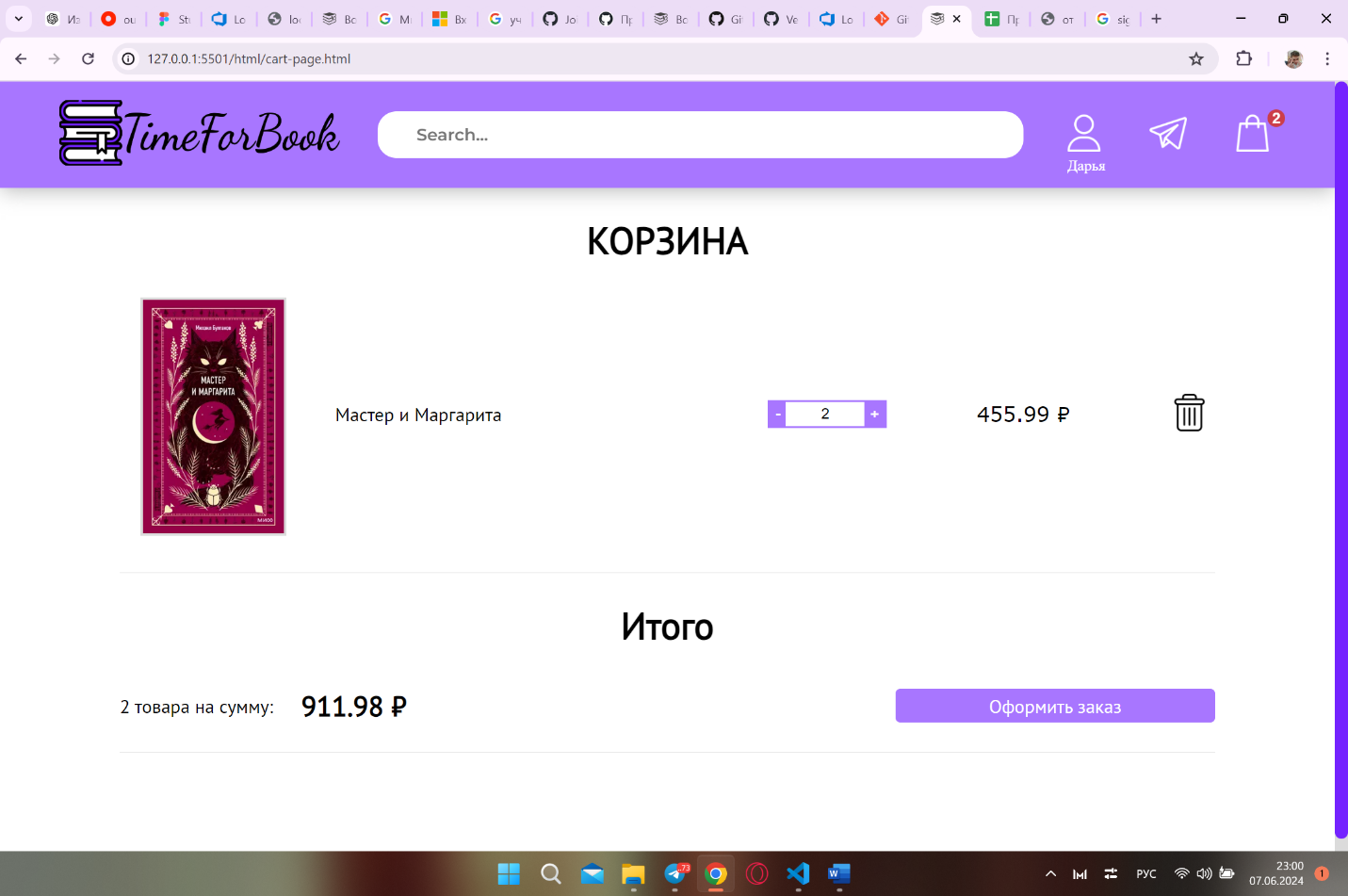
В соответствии с рисунком 10, основными элементами «шапки» являются: логотип интернет-магазина, поисковая строка, иконки профиля, корзины и др., при нажатии на которых пользователь переходит на соответствующие страницы.

Стоит также отметить, что около иконки корзины находится число добавленных в корзину книг totalCount, оно будет динамически обновляться после различных действий пользователя, связанных с добавлением или удалением книги из корзины, а около иконки пользователя, если пользователь зарегистрируется, будет находиться его имя.

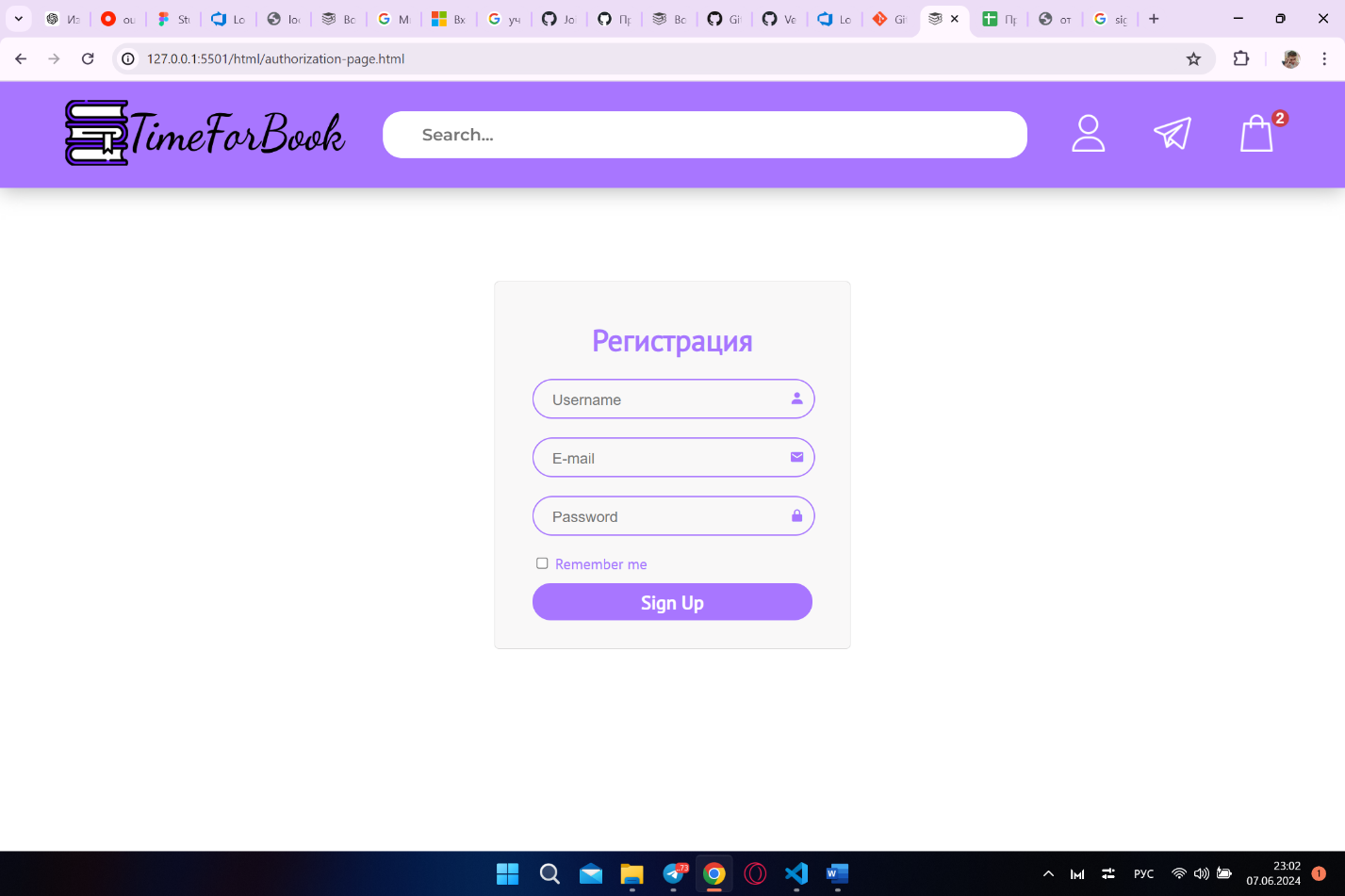
На рисунке 11 изображен конечный вид главной страницы вместе с header:

Рисунок 11 – Интерфейс главной страницы

С главной страницы, благодаря header, пользователь может переходить на страницу корзины, конечный интерфейс которой продемонстрирован на рисунке 12:

Рисунок 12 – Интерфейс страницы корзины

И наоборот, со страницы корзины пользователь может перейти на любую другую интересующую его страницу, например на страницу пользователя, на рисунке 13 представлен конечный интерфейс страницы незарегистрированного пользователя:

Рисунок 13 – интерфейс страницы пользователя

ЗАКЛЮЧЕНИЕ

В результате проделанной мной работы было создано веб-приложение для книжного интернет-магазина, в котором главная страница и страница корзины эффективно взаимодействуют друг с другом через хранение и изменение данных в sessionStorage, поставленные мной задачи были успешно выполнены. Код приложения расположен на GitHub по ссылке: <https://github.com/cutymurphy/book-site>.

Главная страница управляет состоянием корзины, добавляя выбранные пользователем книги и обновляя их количество в sessionStorage, что позволяет странице корзины автоматически получать актуальную информацию о товарах при каждом её открытии, и наоборот. Это обеспечивает синхронизацию состояния корзины с главной страницей, гарантируя, что пользователю всегда отображается актуальная информация о содержимом корзины независимо от того, на какой странице он находится.

Благодаря этому механизму, данные о товарах в корзине остаются непротиворечивыми и обновляются в реальном времени, обеспечивая удобство и целостность работы с корзиной на всех страницах приложения.

Все приложение было создано исключительно с использованием HTML, CSS и JavaScript, без привлечения дополнительных библиотек или фреймворков. Это демонстрирует гибкость JavaScript для решения комплексных задач веб-разработки, а также позволяет легко масштабировать и модифицировать приложение в будущем.

В перспективе планируется добавить бэкенд для более надёжного и масштабируемого хранения данных, а также для управления пользователями и заказами. Кроме того, запланировано переписать приложение с использованием JavaScript-библиотеки React. Эти изменения обеспечат более современную и устойчивую архитектуру, улучшат взаимодействие с пользователем и упростят дальнейшее развитие и поддержку приложения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Современный учебник JavaScript: официальный сайт. URL: <https://learn.javascript.ru/> (дата обращения: 01.06.2024). – Текст: электронный.
2. Адаптивная вёрстка: что это и как использовать / Tproger. URL: <https://tproger.ru/translations/responsive-web-design-tips> (дата обращения: 27.05.2024). – Текст: электронный.
3. Создание приложения на JavaScript, Часть 1 / YouTube – Видео: электронное. URL: <https://www.youtube.com/watch?v=XByK_MG_T_M> (дата обращения: 28.05.2024).
4. Функции JavaScript / Resources for Developers, by Developers. URL: <https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/Functions> (дата обращения: 02.06.2024). – Текст: электронный.
5. Как использовать JavaScript для создания интерактивного сайта / skypro. URL: <https://sky.pro/media/kak-ispolzovat-javascript-dlya-sozdaniya-interaktivnogo-sajta/> (дата обращения: 04.06.2024). – Текст: электронный.
6. Обработчики событий / METANIT.COM. URL: <https://metanit.com/web/javascript/9.2.php> (дата обращения: 01.06.2024). – Текст: электронный.