Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет ИТ

Кафедра Информатики и веб-дизайна

Специальность 1-47 01 02 Дизайн электронных и веб-изданий

**КУРСОВОЙ ПРОЕКТ**

по дисциплине «Современные технологии разработки веб-приложений»

Тема: «Разработка веб-сайта “BOOKHUB”»

**Исполнитель**

студент 3 курса, группы  9 Т.Л. Нгуен

подпись, дата инициалы и фамилия

**Руководитель**

ст. преп. Н.И.  Потапенко

должность, ученая степень, ученое звание подпись, дата инициалы и фамилия

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

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

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

дата, подпись

Курсовой проект защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель Н.И.  Потапенко

подпись дата инициалы и фамилия

Минск 2024

Реферат

Пояснительная записка 39 стр., 35 рис., 9 таб., 14 лист., 2 ист., 4 прил.

BOOKHUB, САЙТ, АНАЛИЗ, ДИЗАЙН, ПРОТОТИП, ФОРМА, КНИГА, ПОЛЬЗОВАТЕЛЬ, ИЗОБРАЖЕНИЕ, КНОПКА, ДИЗАЙН, ЦВЕТ, КОМПОЗИЦИЯ, МОДЕЛЬ, КОМПОНЕНТ, БД, СТРАНИЦА

Целью выполнения курсового проекта является разработка веб-сайта.

В процессе работы конкретизированы задачи для разработки приложения, выявлены группы целевой аудитории, определены функциональные элементы, прототипы типовых страниц, определены дизайн-решения и логотип онлайн-ресурса; сайт был проверен функциональным и кроссбраузерным тестированием, а также имеет адаптивный дизайн. В конце был разработан телеграм-бот, который отвечает на популярные вопросы пользователей.

В результате, в соответствии всем стандартам, разработан собственный многостраничный сайт, поддерживаемый минимум 3 браузерами, обладающим телеграм-ботом и прошедший тестирование на функциональность и кроссбраузерность.

Содержание

[Введение 5](#_Toc166687433)

[1 Аналитический раздел 6](#_Toc166687434)

[2 Проектирование 8](#_Toc166687435)

[2.1 Информационная архитектура проекта 8](#_Toc166687436)

[2.2 Проектирование базы данных 9](#_Toc166687437)

[2.3 Разработка алгоритма приложения 11](#_Toc166687438)

[2.4 Прототипирование 12](#_Toc166687439)

[3 Дизайн 15](#_Toc166687440)

[3.1 Обоснование стиля сайта 15](#_Toc166687441)

[3.2 Разработка логотипа 15](#_Toc166687442)

[3.3 Типографика и цветовая схема 16](#_Toc166687443)

[4 Программная реализация 18](#_Toc166687444)

[4.1 Верстка 18](#_Toc166687445)

[4.2 Программирование серверной части 19](#_Toc166687446)

[4.3 Программирование функциональных модулей 20](#_Toc166687447)

[5 Тестирование 26](#_Toc166687448)

[5.1 Функциональное тестирование 26](#_Toc166687449)

[5.2 Кроссбраузерное тестирование 27](#_Toc166687450)

[5.3 Адаптивность 28](#_Toc166687451)

[6 Руководство пользователя 30](#_Toc166687452)

[Заключение 34](#_Toc166687453)

[Список использованных источников 35](#_Toc166687454)

[ПРИЛОЖЕНИЕ А 36](#_Toc166687455)

[ПРИЛОЖЕНИЕ Б 37](#_Toc166687456)

[ПРИЛОЖЕНИЕ В 37](#_Toc166687457)

[ПРИЛОЖЕНИЕ Г 38](#_Toc166687458)

Введение

В 21 веке, где технологическая среда быстро меняется с каждым годом, многим компаниями приходится лучше и качественнее вовлекать клиентов, как старых, так и новоприбывших. По этой причине в мире цифровых технологий важно использовать актуальные методы по вовлечению целевой аудитории, и узнаваемостью бренда, чтобы таким образом увеличить доход. В большинстве своем главный способ повысить имидж какой-либо компании и стать конкурентноспособным в нынешнее время – это иметь свой собственный сайт, обязательно находящийся на первых строках в поисковике.

В современном мире большинство авторов и читателей переходят на онлайн ресурсы, где они могут быстро и легко начать писать свою историю или же получить доступ к своей любимой книге. По этой причине независимые компании стремятся привлечь большой поток пользователей к своему веб-ресурсу.

С целью стать конкурентноспособным в интернет-пространстве, компании хотят создать сайт таким, чтобы он был привлекателен для большинства и понятен в использовании. Цель данного курсового проекта – создать сайт для чтения и написании книг, где пользователи могут делиться своими книгами или продавать.

В качестве основного инструмента в создании сайта будет использоваться фреймворк Flask Python, который позволит разработать веб-ресурс на языке программирования Python. Также в ходе работы будет разработана база данных, где хранятся данные о книгах и пользователей.

Конечный сайт будет отвечать современным стандартам качественного сайта – отображаться в поисковых системах, иметь адаптивную верстку и обладать понятным интерфейсом. Будут предусмотрены функции для создания профилей пользователей, загрузки и публикации книг, а также система телеграмм-бота, который быстро и эффективно ответит на вопросы пользователей.

1 Аналитический раздел

Перед непосредственной разработкой сайта следует сформулировать цели и задачи к готовому продукту. Так как конечный продукт – это сайт, где пользователи могут выкладывать свои книги, читать или покупать книги других, то основная цель данной работы – это создать удобный и качественный веб-ресурс электронной библиотеки, в котором пользователи разных возрастов могут делиться своими историями.

Однако одной цели недостаточно, чтобы иметь достаточно оснований для создания веб-приложения. Для достижения поставленной цели требуется сформулировать задачи – этапы, по ходу которых можно достигнуть ту самую цель. Задач может быть много, однако, на этапе анализа можно сформулировать следующие задачи:

* *Регистрация и авторизация*. Пользователь может зарегистрировать себя на сайте и войти в него;
* *Поиск по названию книги*. С помощью поискового поля пользователь может найти книгу по названию;
* *Фильтрация*. С помощью специальной формы пользователь может изменить список книг по своему усмотрению;
* *Личный кабинет*. Перейдя в нее, пользователь может увидеть информацию о себе, а также книги, добавленные им в избранное;
* *Телеграмм-бот*. Данный чат-бот на основе ключевых слов в сообщении пользователя отвечает ему на вопросы.

На основе данного списка задач будет разрабатываться дальнейший сайт. Предметная область данного проекта – это платформа для чтения и написания электронных книг. В современном мире реальные книги стали редкостью для молодежи, так как всю информацию они могут найти на телефоне, ноутбуке и компьютере. Вслед за этим книги тоже стали выпускаться в электронном виде, по этой причине создание подобного ресурса стал необходимостью для тех, кто хочет выразить себя в творчестве, но в силу некоторых обстоятельств не может выйти на профессиональный уровень.

Подобные сайты станут полезными для начинающих писателей, которую могут обрести там свою аудиторию, получить ценный опыт и начать зарабатывать на своем творчестве. В свою очередь, читатели, которые хотят прочитать множество книг, но не могут позволить себе физическую книгу, с лёгкостью найдут интересующую их книгу на данной платформе.

Таким образом можно определить целевую аудиторию данной тематики – это люди разных возрастов, которые хотят найти платформу, которая позволит им прочитать или выложить электронную книгу, высказать своё мнение и взаимодействовать друг с другом.

На основе вышеперечисленного можно сделать вывод, что целевая аудитория на поверхности покажется однородной, однако если углубиться в эту тему, то можно достаточно точно сегментировать аудиторию на предмет интересов и т.п.

* *Читатели*. Основная группа, которая чаще всех будет покупать и читать книги. Могут разделяться по своим предпочтениям.
* *Писатели*. Люди, которые хотят продемонстрировать свое творчество и получить опыт в писательстве. Данная группа является важной, так как именно она наполняет сайт контентом, который заинтересует первую группу.
* *Партнеры компаний*. Самая маленькая группа, которая ищет талантов на веб-ресурсе, чтобы помочь им выйти на профессиональный уровень.

После того как была определена целевая аудитория и её сегментация следует анализ и обоснование инструментов разработки.

Flask – это микрофреймворк, что означает, что он имеет минимальное количество зависимостей и функций «из коробки». Это делает его очень гибким и позволяет разработчикам иметь полный контроль над своим проектом. Flask написан на Python, который является одним из самых популярных языков программирования в настоящее время. Python прост в изучении, имеет чистый синтаксис и большую поддержку сообщества, что делает его идеальным для веб-разработки [1].

Фреймворк имеет большое и активное сообщество разработчиков, которое постоянно работает над улучшением фреймворка и созданием новых расширений. Flask позволяет быстро создавать прототипы и выполнять итеративную разработку, что является ключевым аспектом в современной веб-разработке. Также Flask облегчает создание RESTful API, что является стандартом для современных веб-приложений. Flask предоставляет встроенные механизмы защиты от таких угроз, как XSS и CSRF, что обеспечивает безопасность данных пользователей. По этим причинам использование данной технологии подходит созданию сайта.

Для стилизации сайта была использована библиотека Tailwind CSS. Данная библиотека обладает обширным списком классов, которые по различному отображают контент на сайте. Разработчик может сразу в html-разметке прописывать классы Tailwind, чтобы видоизменить содержимое без надобности работать с css-файлом[2].

Также в качестве системы управления базой данных (далее СУБД) будет использоваться SQLite. SQLite – это проверенная временем система управления базами данных, которая обеспечивает высокую производительность и стабильность работы. Данная СУБД хорошо интегрируется с Python, что позволяет оперативно разработать веб-ресурс.[3]

Инструментом для создания контента на сайте была выбран язык разметки HTML. Он является стандартным языком разметки для создания веб-страниц и веб-приложений. Это означает, что практически любой веб-браузер сможет корректно отобразить контент.[4]

2 Проектирование

2.1 Информационная архитектура проекта

После аналитического раздела следует проектирование сайта. В первую очередь, разрабатывается карта сайта. Карта сайта призвана помочь планировать удобство использования для разработчиков и дизайнеров, предоставляет полный обзор, который затем помогает командам упростить, обрезать ненужные страницы и сохранить то, что важно. Однако перед непосредственным её созданием требуется определить страницы.

В рамках данной работы были выбраны следующие типовые страницы:

*Главная страница*. Лицо сайта, имеющее наибольшее количество контента. Исходя из его названия, данная страница должна иметь при себе большую часть основной информации (каталог книг, акции и предложения, рекомендованные произведения, навигационное меню, позволяющее переходить на другие страницы). Основную долю контента должны занимать непосредственно книги, представленные в виде карточек. Главная страница должна быть поделена на блоки разного размера, которые будут иметь разную категорию книг и в которых и будут располагаться карточки товара. Страница на этапе завершения загрузки должна сразу завлечь пользователя, чтобы заставить его остаться и найти понравившейся ему историю.

*Каталог*. Вторая по важности страница. Она должна иметь вид длинного списка книг, представленных в виде карточек, стиль которых должен быть идентичен, что на главной странице. Для того, чтобы пользователь не должен был каждый раз обновлять список товаров, книги должны быть представлены в виде единого списка, который можно продолжить, нажав на кнопку «Показать еще», с возможностью его фильтрации (бесплатно или платно, жанр, есть ли циклы, тэги, дата выхода и т.д.).

*Профиль*. На данную страницу можно перейти только после регистрации. Как только пользователь авторизовался, он окажется на странице профиля. На данной страницу будет располагаться основная информация о пользователе. По умолчанию она почти пустая для того, чтобы пользователь мог сам её заполнить своей биографией, библиотекой любимых книг, рецензиями, списком своих книг и т.д. Важно, чтобы вышеперечисленным пунктам был уделено свой собственный блок, чтобы пользователь мог легко по ним переходить.

*Страница книги.* Страница, описывающая электронную книгу и её характеристики. Пользователи будут очень часто посещать данную типовую страницу, по этой причине её проектированию стоит придать особое значение. Страница должна иметь следующую структуру: обложка книги, если она присутствует, далее идет кнопка «начать читать» или «купить», ниже её описание, вкратце описывающая сюжет, а также блок с комментариями, где пользователи могут поделиться своими мыслями насчет книги.

*Регистрация/авторизация*. Основные экранные формы, помогающие пользователю идентифицировать себя на сайте. Исходя из сайтов аналогов, подобные сайты не требует объемной информации о пользователе. Зачастую при регистрации они требуют его никнейм, электронную почту и пароль.

Карта сайта, приведенная на рисунке 2.1 основана на анализе сайтов-аналогов, чтобы соответствовать потребностям целевых групп. Данная карта проектирована также таким образом, чтобы оптимизировать процесс пользователя и не позволить ему запутаться на сайте.

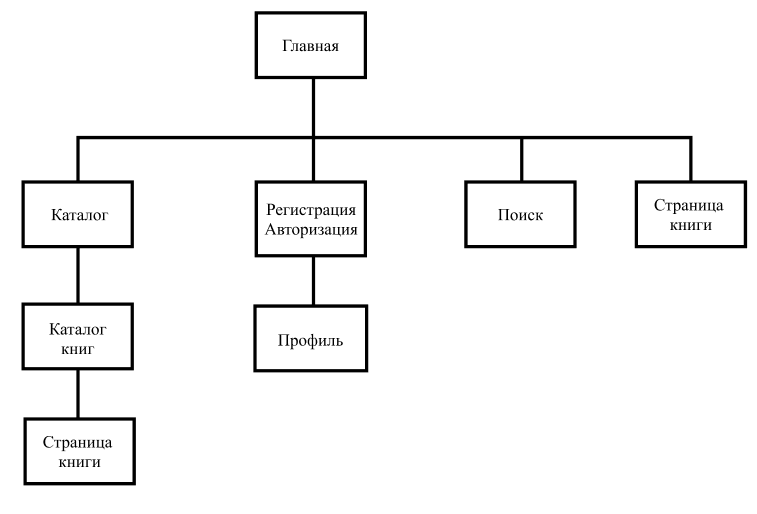


Рис 2.1 – Схема сайта

На основе данной карты будет создана навигация проекта.

2.2 Проектирование базы данных

Далее следует проектирование базы данных для сайта, что является одним из важнейших этапов при разработке ресурса. Исходя из аналитического раздела можно сделать вывод, что база данных (далее БД) должна хранить информацию о пользователе, его книги, жанры, содержимое книги, а также книги, добавленные им в избранное. В результате уже на данном этапе можно разбить будущие данные на логические группы:

* *Пользователи*. В этой таблице можно хранить информацию о пользователях, такую как их идентификатор, имя, электронную почту, пароль и дату регистрации.
* *Книга*. Здесь можно хранить информацию о книгах, такую как идентификатор, название, автор, год выпуска и описание.
* *Глава книги.* Здесь можно хранить информацию о книгах, такую как идентификатор, название, автор, год выпуска и описание.
* *Жанр книги*. Здесь можно хранить информацию о жанре.
* *Избранные книги пользователя*. Здесь можно хранить информацию о книгах, такую как идентификатор, название, автор, год выпуска и описание. Здесь можно хранить информацию о книгах, такую как идентификатор, название, автор, год выпуска и описание.

Таким образом база данных будет состоять из пяти таблиц (см. таблицы 2.1 – 2.6), каждая из которых так или иначе связана с другими. После того как были определены таблицы, следует определить в них поля, при этом учитывая методы нормализации БД.

Таблица 2.1 – Таблица пользователей

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип | Примечания |
| ID | Integer | Идентификатор, Primary key |
| Login | Varchar(255) | Имя пользователя |
| Password | Varchar(255) | Пароль |
| Email | Varchar(255) | Эл, почта |

Таблица 2.2 – Таблица жанров

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип | Примечания |
| ID | Integer | Идентификатор, Primary key |
| Genre | Varchar(255) | Жанр |

Таблица 2.3 – Таблица книги

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип | Примечания |
| ID | Integer | Идентификатор, Primary key |
| Title | Varchar(255) | Название книги |
| Cover | Varchar(255) | Ссылка на обложку |
| Genre | Integer | Жанр, Foreign Key (Жанры) |
| Author | Integer | Автор книги Foreign Key (Пользователи) |
| Date | Integer | Дата выпуска |

Таблица 2.4 – Таблица содержимого книги

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип | Примечания |
| ID | Integer | Идентификатор, Primary key |
| Book\_ID | Integer | Foreign Key (Жанры) |
| Id\_chapter | Integer | Порядок главы |
| Chapter | Varchar(255) | Содержимое главы |

Таблица 2.5 – Таблица избранных книг

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип | Примечания |
| ID | Integer | Идентификатор, Primary key |
| Book | Integer | Foreign Key (Книга) |
| User | Integer | Foreign Key (Пользователь) |

Данная БД будет впоследствии создана на основе SQLAlchemy, программной библиотекой для синхронизации Flask-приложения с данной реляционной базой данных. Его схема и диаграмма вариантов использования представлена в приложении Б и В.

2.3 Разработка алгоритма приложения

В рамках данной работы было решено разработать телеграмм-бота, который будет отвечать на популярные вопросы. Алгоритм разработки построен таким образом (см. в приложении А.), чтобы бот мог искать в сообщениях пользователя ключевые слова и на основе них выбрать соответствующий ответ на вопрос. Для этого потребуется или новая таблица БД, где будут хранятся данные, или же создать json-файл. В ходе анализа алгоритма было принято решение создать json-файл, так как в будущем будет намного легче добавлять новые вопросы и редактировать старые.

2.4 Прототипирование

После определения карты сайта и типовых страниц следует непосредственное создание прототипов самих типовых страниц. Прототипы были созданы в графическом редакторе Figma [14] Первым разработанным прототипом стала главная страницу (рисунок 2.2)

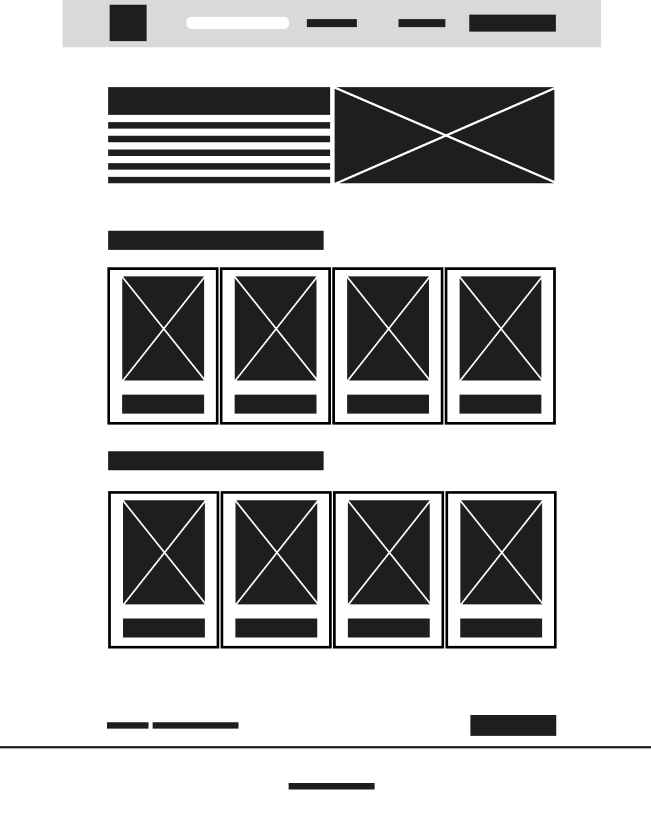


Рисунок 2.2 – Прототип главной страницы

Главная страница разработана таким образом, чтобы предоставлять как можно больше информации пользователю, но при этом не слишком его нагрузив. Как классический вариант, пользователю представлены два списка книг разных категории: популярные за последнее время и последние обновления книг.

Каталог товара (см. рисунок 2.3) спроектирован исходя из описания типовой страницы. В нем присутствует форма фильтра, где пользователь может просмотреть только те книги, которые его интересуют.

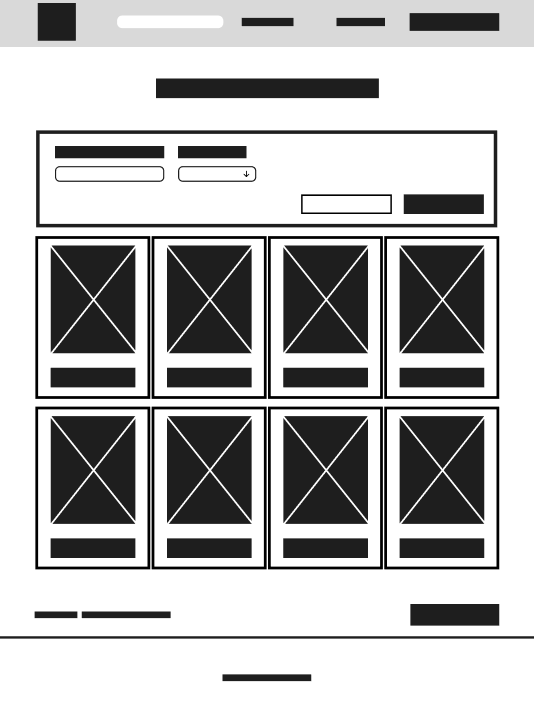


Рисунок 2.3 – Прототип страница каталога

Карта товара (см. рисунок 2.4) является подробной. Прототип представляет собой два больших блока. В одном блоке содержится обложка книги, а во второй – название книги большим шрифтом, имя автора книги, жанр произведения и его описания.

Также присутствует кнопка «Читать», которая направит пользователя на содержимое книги. Возле данной кнопки есть также «добавить в избранное», при нажатии на которой пользователь либо добавит книгу в избранное, либо удалит её из нее. В зависимости от наличия этой книги в данном списке кнопка меняет свое название.

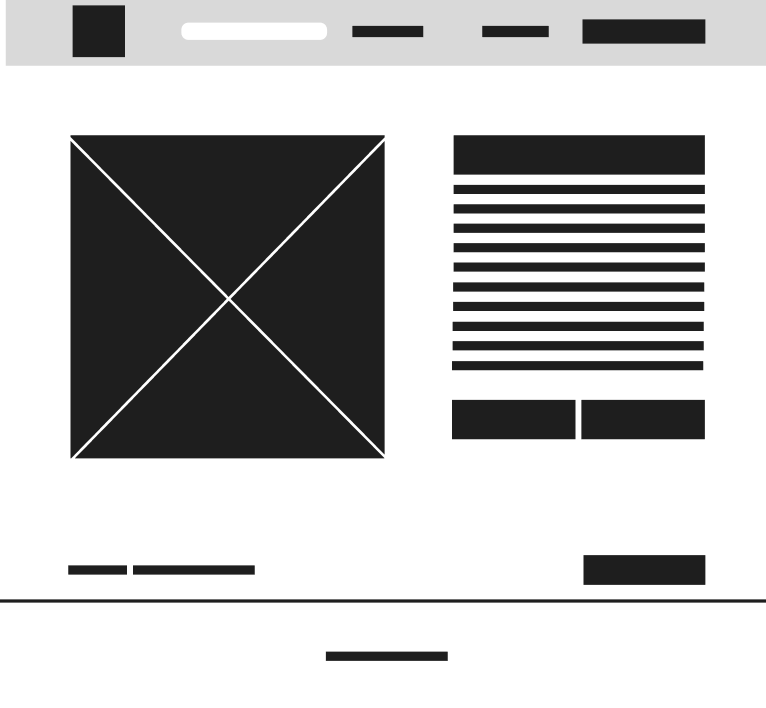


Рисунок 2.4 – Прототип страницы книги

После страницы книги была разработана страница профиля (см. рисунок 2.5). Как и предыдущий прототип, профиль также разбит на два главных блока. Наименьший содержит в себе дополнительную навигацию, которая доступна только зарегистрированному пользователю: настройки, профиль, добавить книгу и выход.

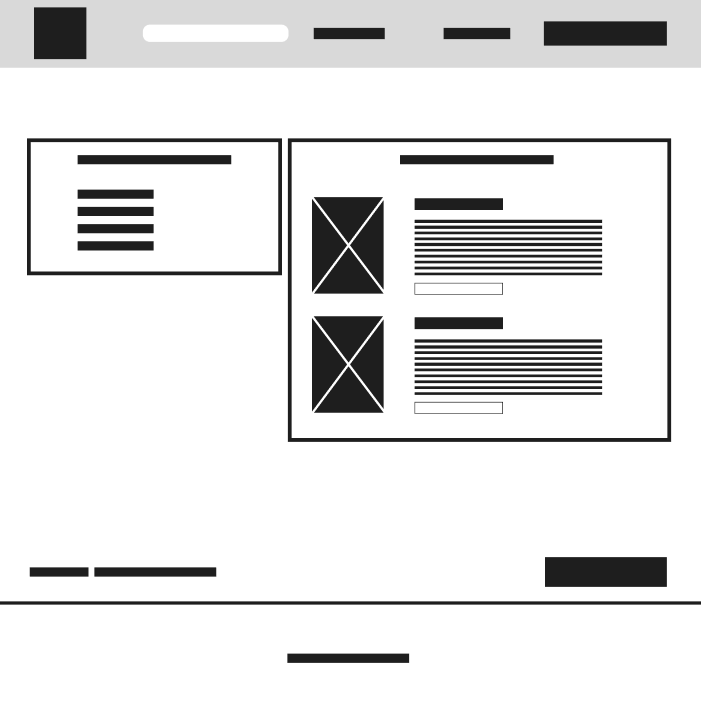


Рисунок 2.5 – Прототип страницы профиля

Во втором блоке хранится представительская информация пользователя. В частности, это его имя и описание, которое он сам может впоследствии изменить. Также в данном блоке будет храниться список его избранных книг.

Последняя страница – это страница 404 (см. рисунок 2.6), на которую переносится пользователь, если искомая страница не существует. Прототип выполнен примитивно, так как основная задача данной страницы – призвать пользователя вернуться на главную страницу или на какую-либо другую существующую.

Основной элемент прототипа – огромное число 404, которое должно первым привлечь внимание пользователя. После нее следует утешительная фраза, за которой – кнопка, предлагающая пользователю вернуться на главную страницу.

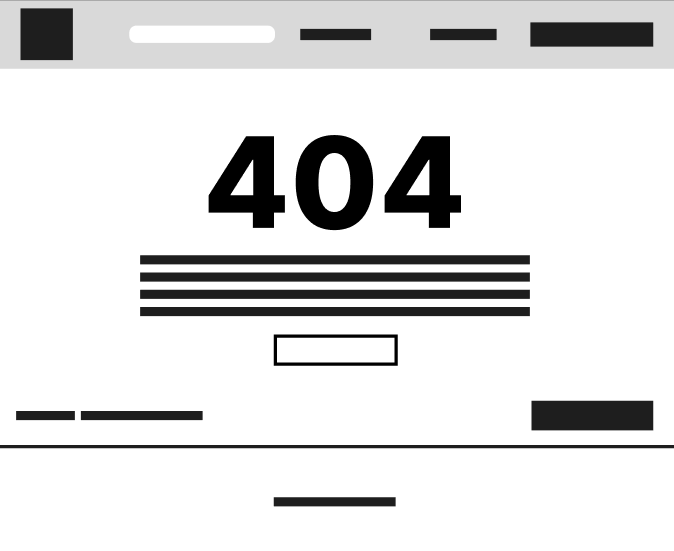


Рисунок 2.5 – Прототип страницы «404»

Немаловажным элементом каждого прототипа является интерактивные формы, позволяющие пользователям взаимодействовать с сайтом. Так как данный сайт предназначен для большинства клиентов, от новичков до заядлых читателей, интерактивные формы разработаны максимально понятно для подавляющей части.

Самый часто наблюдаемый интерактивный элемент – это хедер сайта (см. рисунок 2.6).



Рисунок 2.6 – Хедер сайта

На нем будет поисковое поле, где пользователь может написать часть или полное название книги. Также в нем будут содержаться ссылки на каталог, популярные книги и, наконец, регистрация/авторизация. При нажатии на логотип сайта, пользователь перейдет на главную страницу, что является классическим переходом каждого веб-ресурса.

3 Дизайн

3.1 Обоснование стиля сайта

Сайт BOOKHUB, как было ранее написано, сосредоточен на предоставлении пользователям возможность читать и писать книги. Таким образом с точки зрения удобства взаимодействия с сайтом было принято решение использовать неоклассический стиль. Данный стиль удобен и универсален. Его основной признак – минимум декоративных элементов, чтобы создать чистый и утонченный образ, классический шрифт и симметрия.[13]

3.2 Разработка логотипа

После этапа разработки прототипов идет разработка его дизайна. Одна из основных целей веб-интерфейса – это привлечь целевого пользователя эффективным и привлекательным видом. И самый главный элемент любого сайта – это его логотип. Основная идея логотипа «BOOKHUB» заключается в сочетании двух начальных букв слов «Book» и «Hub».



Рисунок 3.1 – Разработка логотипа

Данные буквы были стилизованы под исландские руны, которые олицетворяют преуспевание и завлекание, что являются достойными качествами каждого писателя. (см. рисунок 3.1).

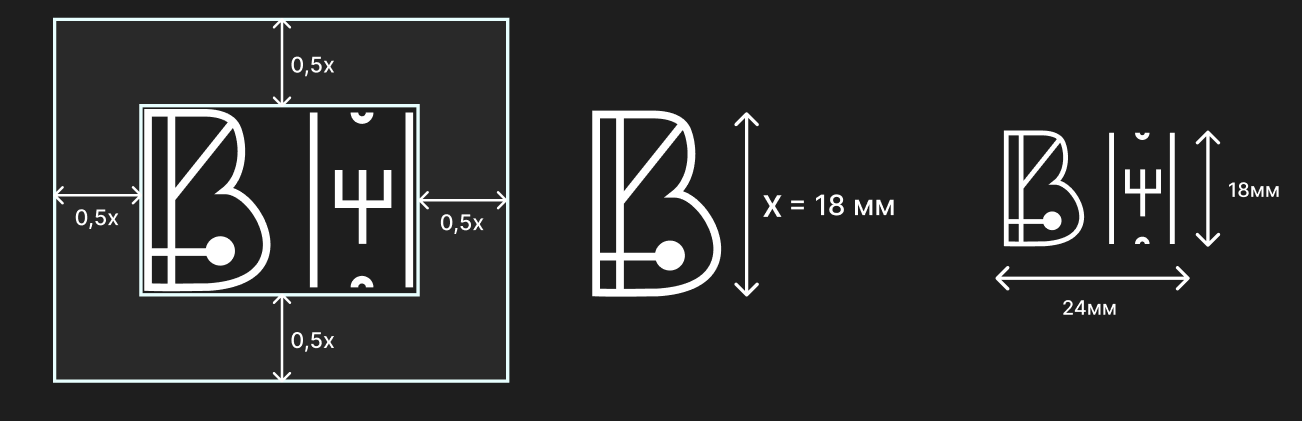


Рисунок 3.2 – Минимальный размер логотипа и его охранное поле

Основные цвета логотипа – это белый цвет для символов и черно-серый для фона. Допускается использование логотипа на различных фонах, при этом разрешается менять цвет символом с белого на черно-серный, если фон слишком ярок. Ниже представлены варианты использования логотипа на различных фонах (см. рисунок 3.3)



Рисунок 3.3 – Логотип на разных фонах

Стоит отметить, что запрещается как-либо искажать, менять фирменные цвета или расположение символов, потому что любое их проявление меняет восприятие пользователя на логотип и противоречит основной задумке.

3.3 Типографика и цветовая схема

Черный и белый цвета имеют свои уникальные преимущества в дизайне веб-сайтов, особенно для онлайн-платформ чтения книг. Черный цвет создает сильный контраст, что помогает выделить текст и другие элементы контента, делая их легко читаемыми.



Рисунок 3.4 – фирменные цвета сайта

Это особенно важно для веб-сайтов, где основной контент состоит из текста, таких как онлайн-книги. Кроме того, черный цвет придает сайту элегантный и стильный вид, что может привлечь пользователей своей привлекательностью. Белый цвет, с другой стороны, обеспечивает простоту и чистоту дизайна. Он делает веб-сайт более открытым и приветливым для пользователей, что особенно важно для платформ чтения книг, где читатели хотят, чтобы контент был доступен и легко воспринимаемым. Белый фон также помогает улучшить визуальную иерархию, делая контент более организованным и понятным для пользователей. Вместе черный и белый цвета образуют классическую цветовую гармонию.

Основной и единственный шрифт, используемый в сайте – это шрифт Inter. Шрифт (см. рисунок 3.5) был выбран с точки зрения читабельности, а также сходства с шрифтом логотипа. Шрифт достаточно прост и не выделяется выразительностью и элегантности.

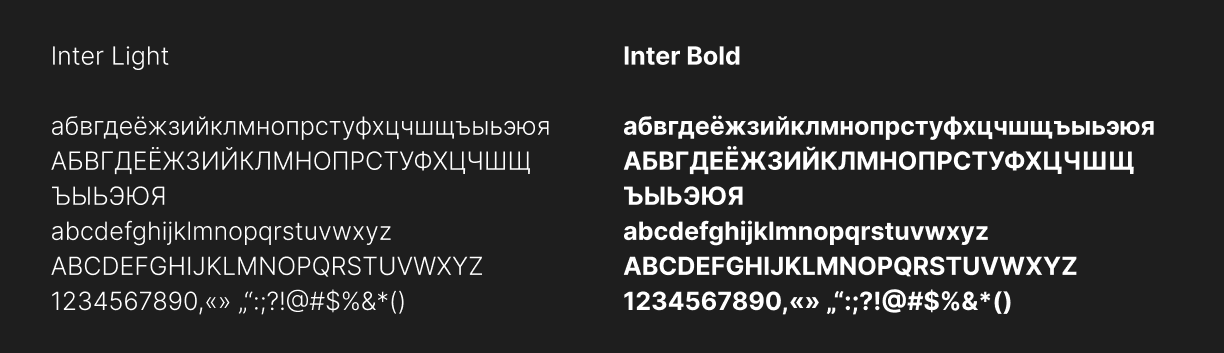
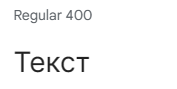
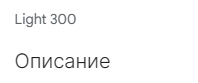


Рисунок 3.5 – Шрифт «Inter»

Текст заголовка имеет жирное начертание и размер в 60px. Обычный текст имени пользователя, а также важных элементов в хедере и футере должен иметь обычное начертание и размер в 28px. Текст описания книги – тонкое начертание и размер в 20px.

а) б) в)

Рисунки 3.6 – Размеры и начертание шрифта Inter.

а) Текст заголовка б) Обычный текст в) Текст описания

Данный логотип, цвета и шрифт будут использованы в разработке веб-приложения «BOOKHUB». Дизайн-макет приведен в приложении Г.

4 Программная реализация

4.1 Верстка

Внешний вид сайта будет реализовываться на основе html-разметки и css-кода. Первый будет инициализировать контент и структуру, а второй – их стилизовать. В качестве облегчения работы была использована библиотека tailwind css, которая стилизует контент с помощью специальных классов (flex, flex wrap, justify, items-center, px, py и т.д.).[9]

Так как весь проект разработан на основе микро-фреймворка Flask, то создание сайта отличается от классической верстки [10]. В первую очередь, в проекте определены следующие папки:



Рисунок 4.1 – Корневые каталоги проекта

В instance хранится файл .db, в котором содержатся данные базы данных, которая была описана в разделе №2. В node\_mode содержатся множество модулей js, которые помогают реализовать компиляцию css кода на основе библиотеки tailwind css. В static находятся все дополнительные файлы, которые реализуют стиль, изображения и js-код сайта. Templates хранит в себе все шаблоны, которые используется при создании типовых страница проекта. Venv, в свою очередь, выполняет внутреннюю среду проекта. В ней установлены все библиотеки, которые используются при разработке сайта.

Также в папке проекта хранится app.py – корневой файл всего проекта. В нем находится все пути, которые направляют пользователя на определенные страницы и модели для работы с таблицами БД.

Верстка в данном проекте представлена таким образом, что при создании новой страницы, сначала требуется создать html файл в папке templates. Он, в свою очередь, наследует шаблон base.html, который передает ему хедер и футер сайта. После этого в нем создаются блоки с классами tailwind, тем самым они стилизуются без непосредственного взаимодействия с css файлом.

Листинг 4.1 – Реализация адресации «/» и «/home»

{% extends 'base.html' %}

{% block title %}

HomePage

{% endblock %}

{% block body %}

 <h1 class="mb-6 text-3xl md:text-5xl lg:text-6xl leading-tight font-bold tracking-tight"> Систематизация любительских книг и переводов веб-новелл</h1>

<p class="mb-8 text-lg md:text-xl font-medium">

…

</p>

{% endblock %}

После того, как была сверстана html страница, путь к ней прописывается в файле app.py с помощью декоратора Flask. (см. листинг 4.4)

@app.route('/')

@app.route('/home')

def index():

    books = Book\_cover.query.all()

    return render\_template("index.html", books=books)

Листинг 4.2 – Реализация адресации «/» и «/home»

На основе данного алгоритма будут созданы последующие страницы с соответствующим функционалом.

4.2 Программирование серверной части

Серверная часть является неотъемлемой частью любого сайта. В проекте BOOKHUB серверной частью является файл BOOKHUB.db, в котором содержится БД и библиотека SQLAlchemy, которая помогает манипулировать БД на SQLite. Реализуется все это за счет строки подключения и объявления моделей (классов), которые были созданы в файле app.py[8]. С помощью них создавались таблицы, определялись поля и связи (см. листинг 4.3). После этого с помощью метода create\_all() в папке instance появлялся BOOKHUB.db.

class Book\_cover(db.Model):

    id = db.Column(db.Integer, primary\_key=True)

    title = db.Column(db.String(100), nullable=False)

    cover = db.Column(db.String(100), nullable=False)

    description = db.Column(db.Text, nullable=True)

    genre\_id = db.Column(db.Integer, db.ForeignKey('genre.id'), nullable=False)

Листинг 4.3 – Реализация модели «Book\_cover» в app.py

В листинге представлен пример создания таблицы «Book\_cover». С помощью метода db.Column создаются столбцы таблицы. В нем можно указать, какой тип данных имеет столб, может ли он быть пустым и является ли столб первичным ключом. Для связи с другими таблицами используется метод db.Foreign с указанием, с какой таблицей надо связать и с каким столбом.

Модели предназначены не только для создания таблиц, но и для добавления и получения строк. Таким образом они будут использоваться в дальнейшем, когда функциональные элементы потребуют добавления или отображения данных.

4.3 Программирование функциональных модулей

*Регистрация/авторизация*. Форма регистрации/авторизации находится на страницу «/registration» и «/auth». В них содержится форма заполнения, где пользователь должен во все поля ввести свои данные. В частности, при регистрации – это имя, почта, пароль и повторение пароля. При авторизации – имя и пароль. В случае если пользователь неверно ввел свои данные, в форме появляется соответствующие сообщение.

В листинге 4.4 представлен класс LoginResource, который проверяет пользователя на пароль и имя. Реализуется это с помощью класса Users, где хранятся все зарегистрированные пользователя. В нем выполняется запрос на поиск имени пользователя до первого вхождения, и если такой пользователь есть, то создается сессия, в котором помещается идентификатор. Данный подход поможет в дальнейшем отображать информацию только об этом пользователя.

Листинг 4.4 – Реализация авторизации в app.py

class LoginResource(Resource):

    def post(self):

        login\_data = request.form

        if not login\_data:

            return "Заполните все поля"

        user = Users.query.filter\_by(login=login\_data['login']).first()

        if user and pbkdf2\_sha256.verify(login\_data['password'], user.password):

            session['user\_id'] = user.id

            return "Успех"

        else:

            return "Неправильно введены данные"

Регистрация разработана схожим образом. В ней предусматривается предупреждение пользователю, если тот ввел электронную почту, которая была уже зарегистрирована. Проверяется все с помощью специального запроса в таблицу Users. В случае наличия пользователя с такой почтой, то класс возвращает сообщение о занятой почты и выводит его в форму. Если же пользователь ввел все правильно, то его переносят на главную страницу, где он уже сможет перейти на авторизацию.

class UsersResource(Resource):

    def post(self):

        new\_user\_data = request.form

        existing\_user = Users.query.filter((Users.login == new\_user\_data['login']) | (Users.email == new\_user\_data['email'])).first()

        if existing\_user:

            return "Пользователь с данной почтой или именем уже существует"

        hashed\_password = pbkdf2\_sha256.hash(new\_user\_data['password'])

        new\_user = Users(login=new\_user\_data['login'], password=hashed\_password, email=new\_user\_data['email'])

        db.session.add(new\_user)

        db.session.commit()

Листинг 4.5 – Реализация регистрации

Стоит также отметить, что в целях безопасности пароль пользователя хешируется с помощью метода pbkdf2\_sha256.hash(), который обеспечит целостность и безопасность данных.

*Поиск*. Поиск реализован более комплексным образом. В начале в html страницу создан input с идентификатором.

<form id="searchForm" method="get" class="w-1/2 xl:w-1/3">

<input type="text" class="w-full mr-6 rounded-md max-xl:hidden px-3 h-8 text-black outline-none shadow-xsm" placeholder="Найти" id="searchInput"/>

</form>

Листинг 4.6 – Форма поиска в html-шаблоне base.html

С помощью него в js-файле сайт получает данные в input и передает его на страницу каталога методом GET, одновременно с этим перенаправляя пользователя на данную страницу. Пользователя переносят с помощью события keydown. То есть пользователь должен нажать на Enter, чтобы перенестись на страницу с результатами поиска.

document.addEventListener("DOMContentLoaded", function() {

  var searchInput = document.querySelector("#searchInput");

  searchInput.addEventListener("keydown", function(event) {

    if (event.key === "Enter") {

      event.preventDefault();

      var searchTerm = searchInput.value.trim();

      window.location.href = "/catalog?search=" + encodeURIComponent(searchTerm);

    }

  });

});

Листинг 4.7 – Скрипт для получения запроса из поискового поля

Таким образом каталог получает с адреса запроса и по нему выполняет запрос по подстроке, чтобы впоследствии в каталоге показать список книг с похожим названием. Данная реализация продемонстрирована в дальнейшем.

*Фильтрация*. Код начинается с определения маршрута '/catalog' с методом GET. Это означает, что при обращении к этому маршруту с помощью GET-запроса будет выполнена функция catalog(). Внутри функции происходит извлечение параметров запроса: search\_query, genre\_id и author\_name, которые содержат информацию о поиске, жанре и авторе соответственно. Далее происходит запрос всех жанров из базы данных и подготовка запроса книг, который будет использоваться для фильтрации в зависимости от параметров запроса. Если указан genre\_id, то книги фильтруются по этому жанру.

Если указан author\_name, то происходит поиск пользователя (предположительно автора) по имени и книги фильтруются по его идентификатору. Затем, если есть search\_query, книги фильтруются по заголовку с использованием подстроки поиска. Наконец, отфильтрованный список книг передается в шаблон catalog.html вместе с списком жанров для отображения пользователю.

Листинг 4.8 – Реализация фильтрации

@app.route('/catalog', methods=['GET'])

def catalog():

search\_query = request.args.get('search')

genre\_id = request.args.get('genre')

author\_name = request.args.get('author')

genres = Genre.query.all()

books = Book\_cover.query

if genre\_id:

books = books.filter\_by(genre\_id=genre\_id)

if author\_name:

author = Users.query.filter(Users.login.ilike(f"%{author\_name}%")).first()

if author:

books = books.filter\_by(author=author.id)

if search\_query:

books = books.filter(Book\_cover.title.ilike(f"%{search\_query}%"))

books = books.all()

return render\_template("catalog.html", books=books, genres=genres)

*Личный кабинет*. Личный кабинет в проекте будет показывать информацию о пользователе и всех его книгах, добавленных в избранное. Реализуется добавление с помощью кнопки, которая передает идентификатор книги в адрес «/favorite/add/<int:book\_id>». С помощью запроса создается новый объект таблицы избранных книг, после чего пользователя перенаправляют на страницу книги. Также предусмотрено, что в зависимости от того, есть ли у пользователя данная книга в избранных или нет, меняется надпись кнопки.

@app.route('/favorite/add/<int:book\_id>', methods=['POST'])

def add\_to\_favorites(book\_id):

    if 'user\_id' not in session:

        return jsonify({'message': 'Пользователь не авторизован'}), 401

    user\_id = session['user\_id']

Листинг 4.9 – Реализация добавления в избранное

Продолжение листинга 4.9

existing\_like = Book\_like.query.filter\_by(id\_user=user\_id, id\_book=book\_id).first()

    if existing\_like:

        db.session.delete(existing\_like)

        session['isLike'] = False

    else:

        favorite = Book\_like(id\_user=user\_id, id\_book=book\_id)

        session['isLike'] = True

        db.session.add(favorite)

    db.session.commit()

    return redirect(url\_for('book', book\_id=book\_id))

Таким образом уже в личном кабинете пользователя отобразятся все книги из модели Book\_like.

*Телеграмм-бот*. Для реализации телеграмм-бота был создан отдельно файл bot.py. Перед непосредственной разработкой алгоритма бота вначале был создан сам бот. Создан он был с помощью BotFather, который позволил также редактировать его изображение, имя и описание. После этого в bot.py был передан токен бота BOOKHUB, чтобы алгоритм работал.

Далее был создан json-файл, из которого бот будет брать ключевые слова и ответы на вопрос пользователя. Ответ представлен в виде ключа, а его ключевые слова – массив из слов.

{

    "BOOKHUB - это сайт, где собраны все любительские книги и новеллы, написанные и переведенные с любовью и энтузиазмом. Вы также можете оставить свое произведение на BOOKHUB": ["портал", "сайт", "ресурс"],

    "На нашем сайте вы можете прочитать множество книг от разных авторов. Выбирайте в каталоге любую понравившуюся вам книгу": ["книг", "процесс", "чтени"],

    "Зарегистрировавшись на нашем сайте, вы можете поделиться своей собственной историей на BOOKHUB!": ["регистраци", "сайт", "польз"]

}

Листинг 4.10 – Json-файл, хранящий ответы на вопросы

С помощью декоратора @bot.message\_handler бот принимает сообщение пользователя, после чего начинает искать в сообщении ключевые слова. На основе найденных ключевых слов бот выбирает подходящий ответ и выводит его пользователю.

Листинг 4.11 – Алгоритм бота

with open('faq.json', 'r', encoding='utf-8') as faq\_file:

    faq\_data = json.load(faq\_file)

@bot.message\_handler(func=lambda message: True)

def handle\_text(message):

    text = message.text.lower()

    for question, keywords in faq\_data.items():

        if any(keyword in text for keyword in keywords):

            bot.send\_message(message.chat.id, question)

            return

    bot.send\_message(message.chat.id, "Спасибо за ваш вопрос!")

bot.polling(none\_stop=True)

Он загружает данные из файла faq.json. Затем бот просматривает каждое входящее сообщение, приводит его к нижнему регистру и проверяет, содержит ли оно какие-либо из ключевых слов из файла FAQ. Если да, то бот отправляет соответствующий вопрос из FAQ в ответ. Если нет совпадений, бот отправляет стандартное сообщение "Спасибо за ваш вопрос!".

bot.polling(none\_stop=True) запускает бота, ожидая новых сообщений от пользователей и обрабатывая их в соответствии с функцией handle\_text.

*Бургер-кнопка*. Для реализации работы кнопки был использован js-код

const burger = document.querySelector('.navbar-burger');

   const menu = document.querySelector('.navbar-menu');

   const con = document.querySelector('.navbar-content')

   burger.addEventListener('click', () => {

     menu.classList.toggle('hidden');

   });

   document.addEventListener('click', (event) => {

       if (!con.contains(event.target) && !burger.contains(event.target)) {

         menu.classList.add('hidden');

       }

     });

 });

Листинг 4.12 – Реализация бургер-кнопки

Данный код работает по логике: если пользователь нажал на кнопку, то высвечивается меню. Событие «Click» отслеживает положение, куда пользователь кликнул. В случае, если он кликнул вне мобильного меню, то меню исчезает.

5 Тестирование

5.1 Функциональное тестирование

Функциональная тестирование является важным аспектом при разработке любого веб-приложения. Благодаря нему можно обнаружить непредвиденные ошибки или недостатки проекта, которые потом впоследствии исправляются. Для сегментации тестирования ситуации(тест-кейсы) можно разделить на два типа: позитивные и негативные.[11]

Позитивные продемонстрируют работу приложения при корректно введенных данных, в то время как негативные наоборот. В качестве тест-кейсы можно взять следующие задачи:

* Авторизация на сайте (негативный, см. таблицу 5.1)
* Поиск с помощью поискового поля (позитивный, см. таблицу 5.2)
* Фильтр каталога с помощью формы фильтра (позитивный, см. таблицу 5.3)
* Переход в личный кабинет (негативный, см. таблицу 5.4)

Данные тестовые ситуации отлично продемонстрируют корректность работы системы. Кейсы будут показаны в виде таблиц, в которых описаны начальные условия для теста, тип, его цель и выполнение.

Таблица 5.1 – Тест-сьют авторизации

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Приоритет | Предусловие | Шаги | Ожидаемый результат |
| Авторизация на сайте | 1 | Неавторизованным находится на главной странице | 1. Нажать на кнопку «Войти» на главной страницу 2. В поле логина ввести «Nomore241» 3. В поле пароля ввести «234» 4. Нажать на галочку 5. Нажать на кнопку «Войти» | Форма авторизации выведет сообщение красным цветом «Неправильно введены данные» |

Таблица 5.2 – Тест-сьют поиска с помощью поискового поля

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Приоритет | Предусловие | Шаги | Ожидаемый результат |
| Поиск с помощью поискового поля | 2 | Неавторизованным находится на главной странице | 1. Ввести в поисковое поле значение «Этот мир» 2. Нажать на «Enter» | Приложение направит в каталог с результатами поиска по названию книги |

Таблица 5.3 – Тест-сьют поиска с помощью поискового поля

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Приоритет | Предусловие | Шаги | Ожидаемый результат |
| Фильтр каталога с помощью формы фильтра | 2 | Неавторизованным находится на странице каталога | 1. Написать в поле автора значение «Nomore241» 2. Выбрать из списка жанров жанр «Боевик» 3. Нажать на кнопку «Искать» | Приложение предоставит новый список книг согласно условиям, поставленных пользователем |

Таблица 5.4 – Тест-сьют поиска с помощью поискового поля

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Приоритет | Предусловие | Шаги | Ожидаемый результат |
| Переход в личный кабинет | 1 | Авторизованным находится в личном кабинете | 1. Нажать на кнопку «Выйти» 2. Вернуться на страницу личного кабинета | Приложение вместо личного кабинета направит пользователя на страницу авторизации |

Данные тесты показали надежность работы системы и корректность работы функциональных элементов. Критические ошибки не были обнаружен или допущены во время разработки приложения.

5.2 Кроссбраузерное тестирование

В данном подразделе проверяется, отображается ли веб-приложение предсказуемо и одинаково на популярных браузерах. Для теста были выбраны следующие браузеры: Google Chrome, Opera и Microsoft Edge – именно ими чаще всего пользуются рядовые пользователи. Кроссбраузерное тестирование будет происходить в три этапа. Первое – проверить, имеет ли сайт одинаковый вид на всех браузерах. Второе – отключить все изображения и оценить работу сайта. Третье – отключить поддержку JavaScript и оценить работу сайта.

В ходе тестирования было выявлено, что сайт отображается корректно на всех выбранных браузерах в независимости от типа устройства. Благодаря тому, что во всех тэгах «<img>» был прописан параметр «alt», то при отключении всех изображений, вместо картинок вставлены названия (см. рисунок 5.1)

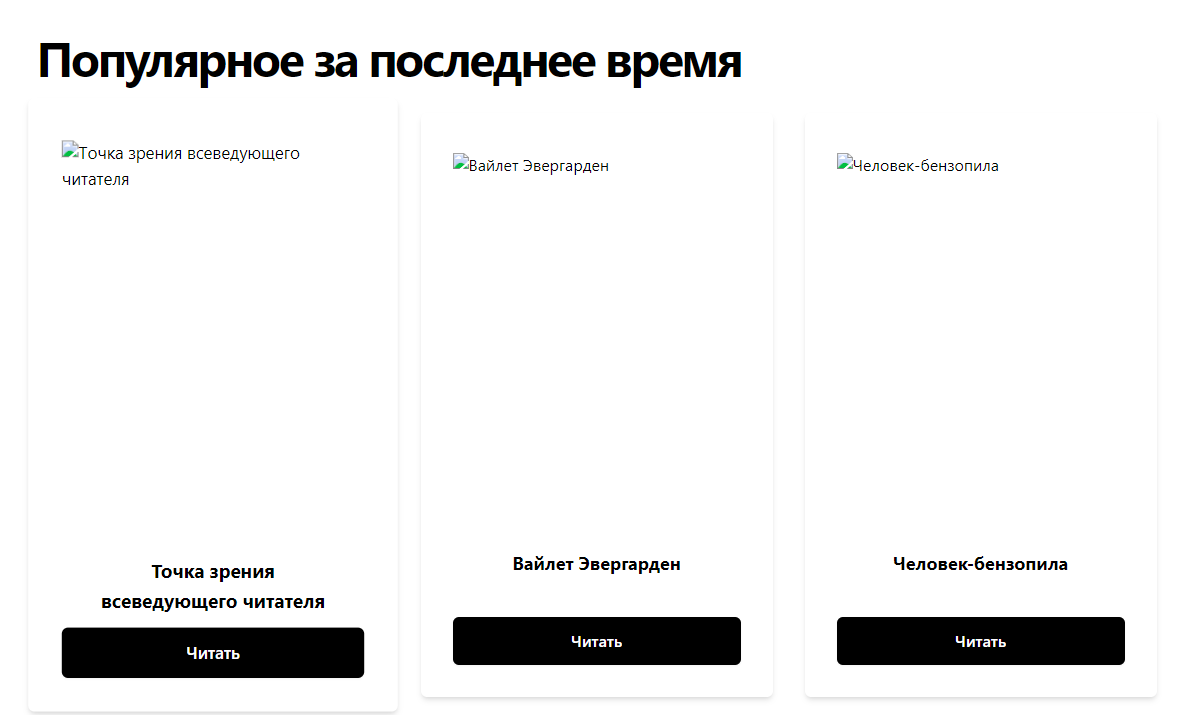


Рисунок 5.1 – Список книг при отключении изображений

Также было проверено, что при отключении JavaScript библиотека Slick Slider более не могла поддерживать слайдер на главной странице. Результат показал, что слайдер останавливается на последнем показанном изображении.

5.3 Адаптивность

Адаптивность – это способность сайта «подстраиваться» под различные размеры экрана пользовательского устройства. Адаптивный сайт хорошо смотрится и на десктопном мониторе, и на планшетном компьютере, и на экране смартфона. Причем качество отображения не зависит ни от диагонали, ни от позиционирования экрана. На сегодняшний день адаптивность является одним из ключевых требований, предъявляемых к современным сайтам.

К этапу разработки адаптива можно приступать только после того, как созданный веб-сайт полностью соответствует своему макету. Для создания адаптивного сайта «BOOKHUB» была снова использована библиотека Tailwind CSS. В ней предусмотрены специальные классы, которые уже работают с медиа-запросами при определенных ширинах:

* 2xl. Минимальная ширина 1536px;
* xl. Минимальная ширина 1280px;
* lg. Минимальная ширина 1080px;
* md. Минимальная ширина 768px;
* sm. Минимальная ширина 640px.

Для создания адаптивности достаточно указать одного из классов выше, например md, чтобы уже указать, до какой минимальной ширины будет действовать расположение объектов.

В результате сайт стал адаптивным к практически любым размерам экрана. Для проверки адаптива был использован инспектор кода (см. рисунок 5.2).

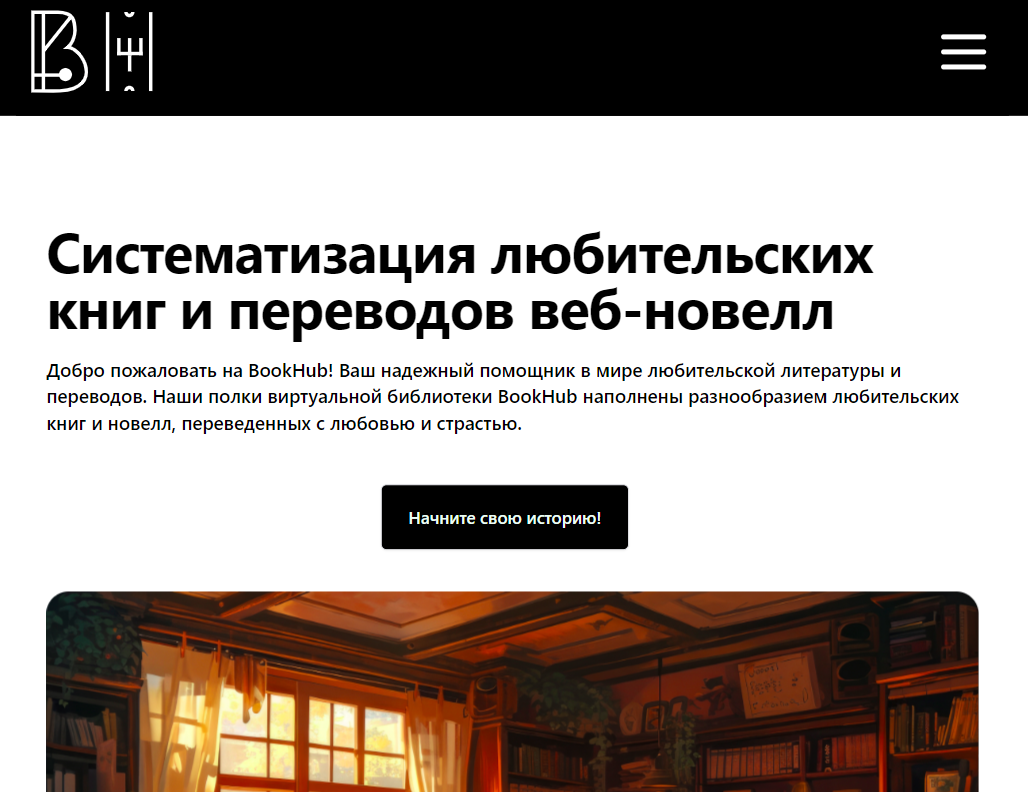


Рисунок 5.2 – Главная страница при ширине 1200px

Также в мобильной версии присутствует бургер-кнопка – альтернатива навигационной шапке (см. рисунок 5.3)

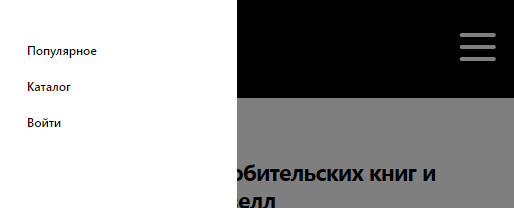


Рисунок 5.3 – Меню бургер-кнопки

. При определенной ширине для облегчения навигации по сайту она заменит шапку. Если пользователь на нее нажмет, то высветиться боковое меню, в котором присутствуют почти все элементы, находящихся на оригинальной шапке. В зависимости от того, зарегистрирован ли пользователь или нет, навигационное меню будет иметь дополнительные ссылки.

6 Руководство пользователя

BOOKHUB – сайт, предоставляющий пользователю возможность читать и писать книги. Его основная цель – дать пользователю уютную среду, где он может проводить свободное время за чтением любимых книг или продемонстрировать себя в качестве писателя.

*Вход в личный кабинет*. Работа с ресурсом осуществляется с помощью браузеров Google Chrome, Mozilla FireFox, Opera и Edge актуальных версий. Для перехода на сайт следует в строку поиска ввести ссылку «http://127.0.0.1:5000/», после чего отобразиться главная страница ресурса (см. рисунок 6.1).

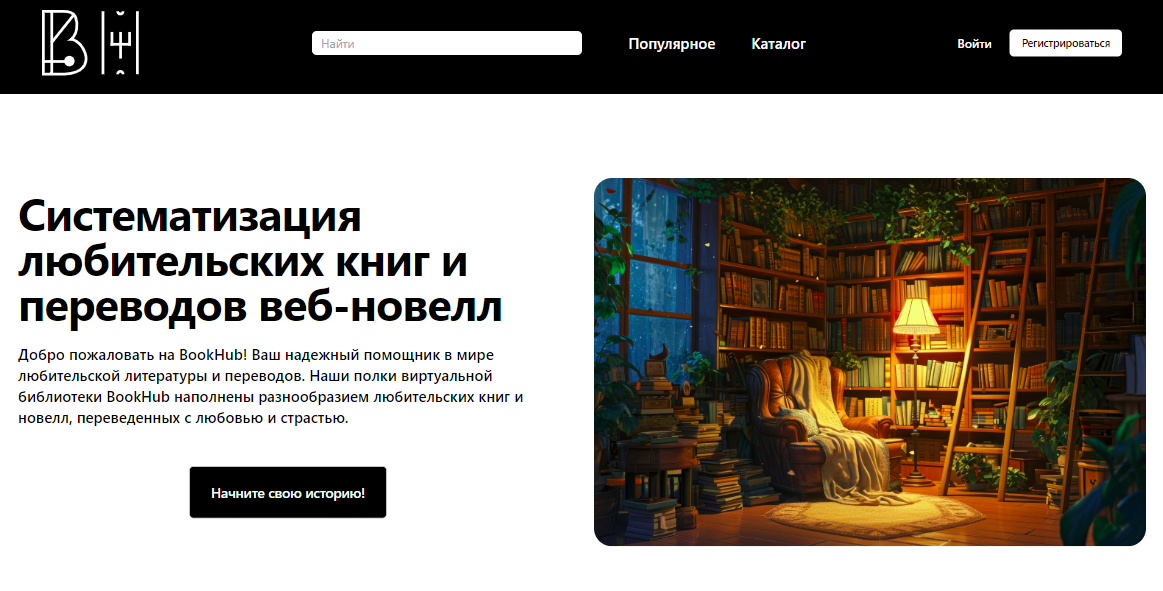


Рисунок 6.1 – Главная страница сайта

Если пользователь хочет добавлять свои книги в избранные, ему необходимо нажать на кнопку «Регистрация». После нажатия его переносят на соответствующую страницу, где будет находится модальное окно (см. рисунок 6.2)

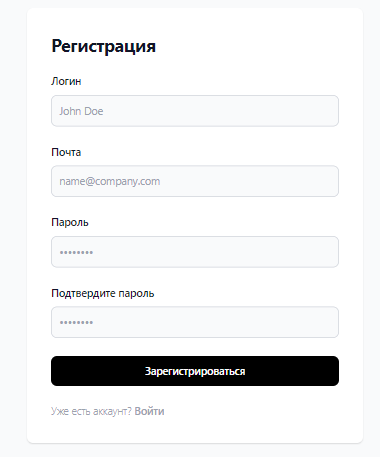


Рисунок 6.2 – Модальное окно регистрации

После заполнения полей в форме регистрации пользователю следует сохранить введенные данные с помощью кнопки «Зарегистрироваться».

Если все его данные удовлетворяют требованиями сайта, его перенаправят на главную страницу. В случае же если форма заполнена некорректно система выдаст сообщение об ошибке, а ниже названия окна будет выведено сообщение, объясняющую, где пользователь допустил ошибку.

Уже зарегистрированный пользователь осуществляет авторизацию с помощью кнопки «Войти», которая располагается рядом с «Регистрация». Авторизация представляет собой также модальное окно, схоже с регистрацией. При успешной авторизации предыдущие кнопки исчезнут, и вместо них появится ссылка на личный кабинет (см. рисунок 6.3).

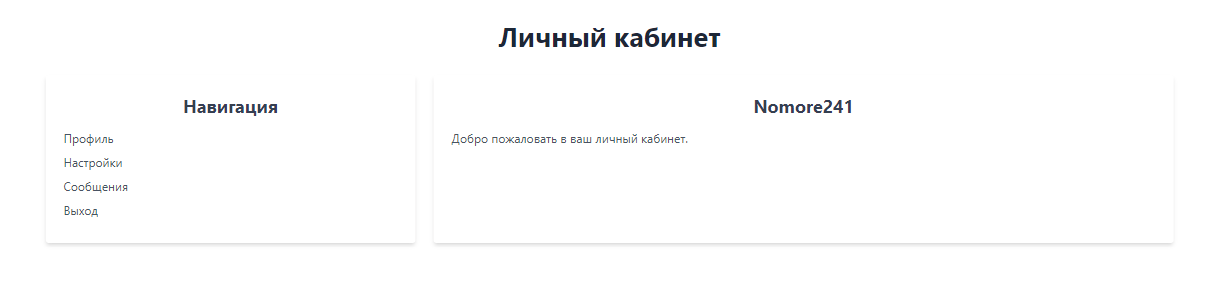


Рисунок 6.3 – Личный кабинет

Уже в личном кабинете пользователь может выйти из аккаунта, нажав в навигационном блоке кнопку «Выход».

*Фильтрация.* В независимости от того, зарегистрирован пользователь или нет, он может фильтровать каталог книг на основе двух параметров: имени автора и жанра книги (см. рисунок 6.4). Для этого ему потребуется перейти на страницу каталога, после чего ему будет представлена форма фильтра

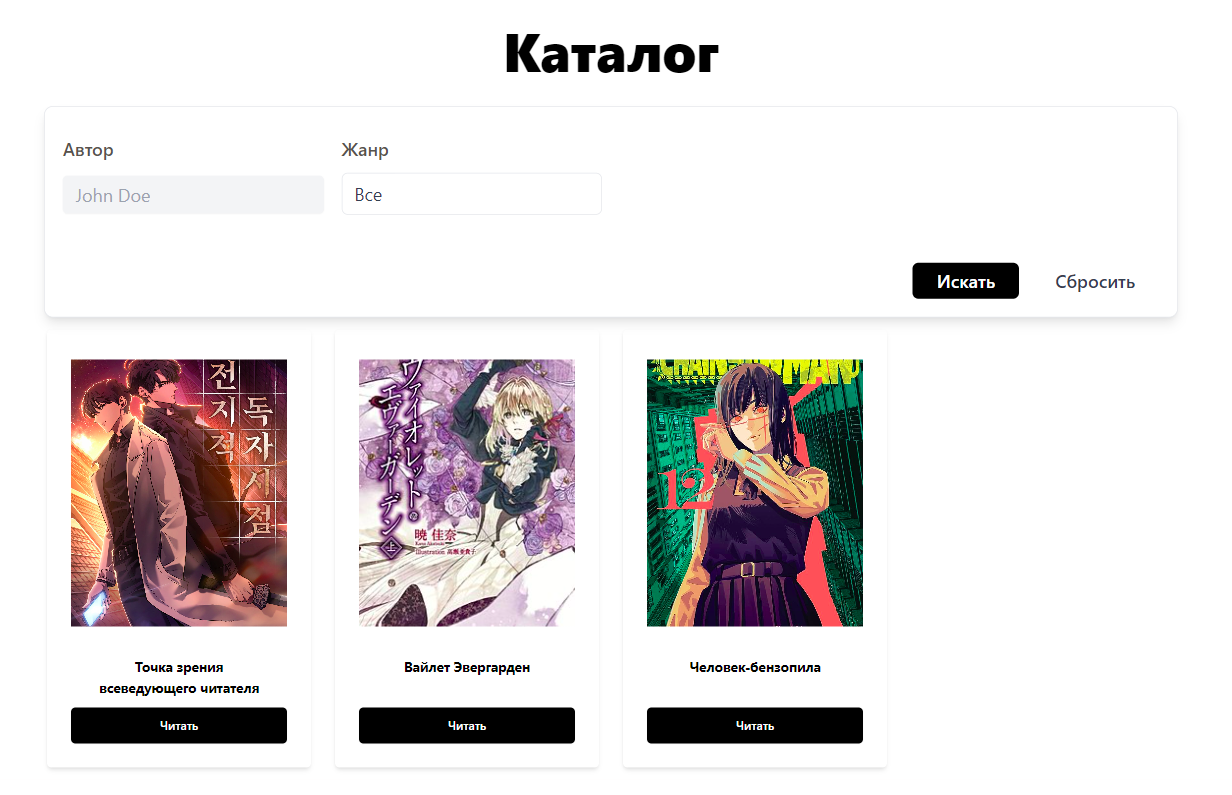


Рисунок 6.4 – Каталог «BOOKHUB»

С помощью кнопки «Искать» пользователь выполняет фильтрацию каталога, «Сбросить» – вернуться к списку всех книг. В поле автора пользователь может ввести имя автора, после чего каталог предоставит ему на основе совпадений все книги предполагаемого автора. В свою очередь, поле жанр представлен в виде списка, где пользователь выбирает понравившийся ему пункт

*Поиск*. Поле поиска находится в шапке сайте (см. рисунок 6.5), благодаря чему, на любой странице пользователь может воспользоваться поиском и перейти в каталог.



Рисунок 6.5 – Поисковое поле в шапке

При нажатии на «Enter» пользователь переходит в каталог, который уже при загрузке страницы уже отфильтрован в зависимости от содержимого. Если пользователь ничего не ввел, но при этом перешел в каталог с помощью поискового поле, то каталог предоставит ему весь список книг.

*Добавление в избранное*. Если пользователь изъявляет желание добавить понравившуюся книгу ему в кабинет. Для этого ему потребуется авторизоваться для начала, так как кнопка «добавить в избранное» (см. рисунок 6.6) будет отображена только в том случае, если пользователь уже зашел в личный кабинет. После этого он должен перейти на страницу книги

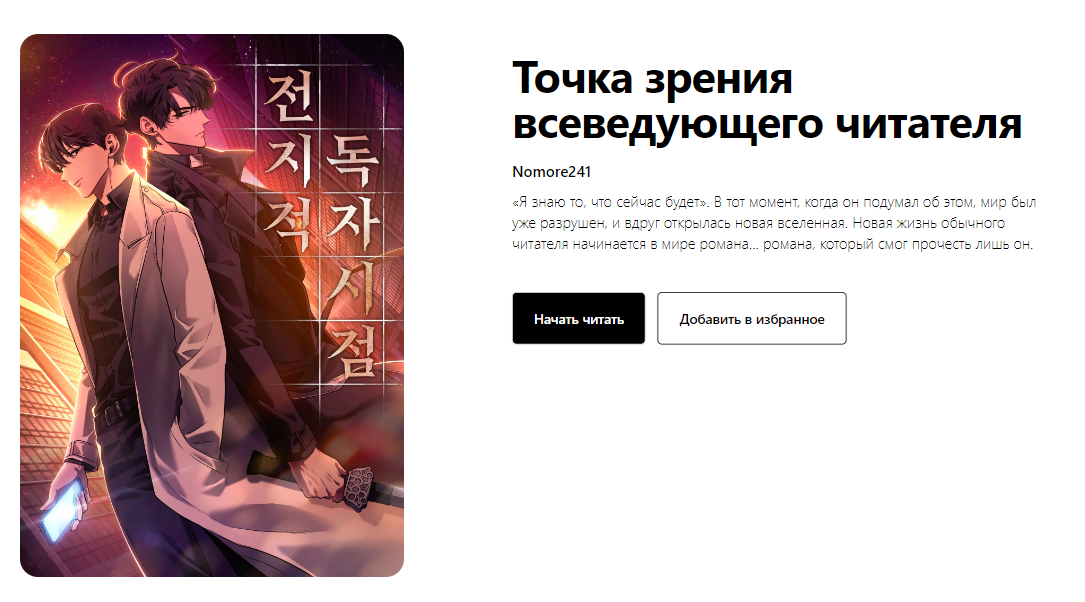


Рисунок 6.6 – Страница книги

Рядом с кнопкой «Читать» будет находится «Добавить в избранное». При нажатии на нее книга добавится в личный кабинет пользователя.

*Телеграмм-бот.* Сайт «BOOKHUB» также обладает своим собственным телеграмм ботом. Для того, чтобы взаимодействовать с ботом, пользователь в футере сайта может найти ссылка на телеграмм-бота (см. рисунок 6.7).

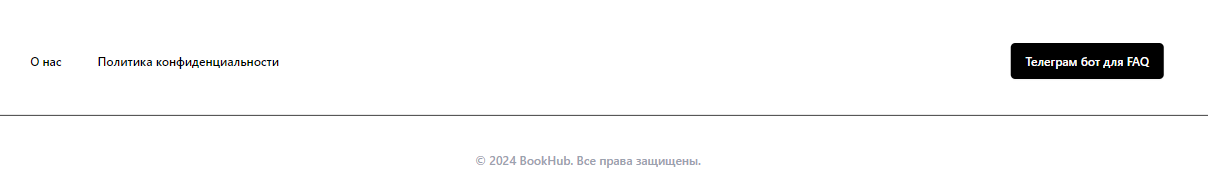


Рисунок 6.7 – Футер сайта

После этого пользователя перенаправят на телеграмм-ссылку, где ему предлагается открыть с ним чат. Бот разработан таким образом, чтобы демонстрировать имитации чата пользователя с настоящим человеком. Пользователь пишет вопрос, который его интересует, после чего бот мгновенно выводит ответ.

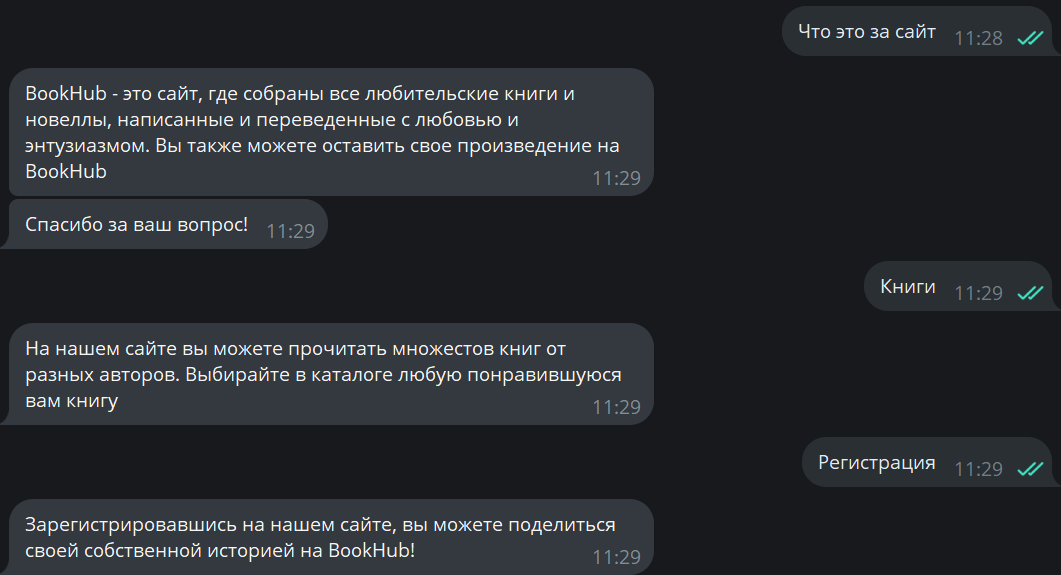


Рисунок 6.8 – Взаимодействие с чат-ботом

Если же пользователь пишет непонятный боту запрос, то он выводит статическое сообщение, которое просит уточнить или поменять вопрос.

Заключение

Целью данного курсового проекта являлось разработка адаптивного кросс-браузерного сайта, где люди могут делиться историями и читать их. Для достижения данной цели весь объем работы был подразделен на несколько основных этапов:

* анализ схожих ресурсов;
* требования к сайту;
* проектирование базы данных;
* дизайн;
* программная и серверная реализации;
* создание руководства для пользователя.

Благодаря подобной сегментации в результате курсового проекта был успешно создан сайт со всем функционалом, который был описан ранее. Данный сайт также обладает адаптивной версткой, что значит, что он корректно отображается практически на всех ширинах экрана, вплоть до мобильной.

Программная и серверная реализации были выполнены с использованием микро-фреймворка Flask, обеспечивающего гибкость и высокую производительность веб-приложения.

C помощью библиотеки Tailwind CSS была выполнена стилизация сайта. С ним не требовалось работать одновременно с HTML-файлом и CSS-файлом. Его установка позволила использовать специальные классы, которые, в свою очередь, стилизовали контент и выполняли адаптив. Благодаря нему сайт обладает резиновой версткой, что является хорошим качеством.

Slick.js позволил использовать слайдер на главной странице, за счет чего данная страница выглядит более презентабельной.

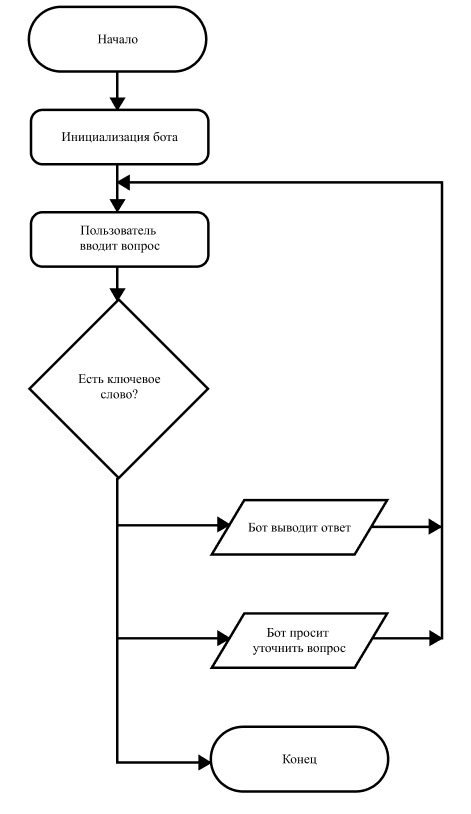
Можно сделать вывод, что поставленная цель была достигнута. Были закреплены знания использования библиотек и современного микро-фреймворка, углублено его понимание и взаимодействие, что несомненно понадобится в будущих проектах.

Список использованных источников

1. Что такое Flask и для чего он используется: SkyPro [Электронный ресурс] / Режим доступа: https://sky.pro/media/chto-takoe-flask-i-dlya-chego-on-ispolzuetsya/
2. Быстро создавайте современные веб-сайты, не выходя из HTML-кода: TailwindCSS [Электронный ресурс] / Режим доступа: <https://tailwindcss.ru/>
3. Работа с SQLite: Habr [Электронный ресурс] / Режим доступа: <https://habr.com/ru/articles/754400/>
4. HTML: Wikipedia [Электронный ресурс] / Режим доступа: https://ru.wikipedia.org/wiki/HTML#:~:text=HTML
5. Онлайн-инструмент создания диаграмм: Visual Paradigm [Электронный ресурс] / Режим доступа: https://online.visual-paradigm.com/ru/diagrams/features/uml-tool/
6. Онлайн-инструмент создания БД: Visual Paradigm [Электронный ресурс] / Режим доступа: <https://online.visual-paradigm.com/ru/diagrams/features/uml-tool/>
7. Изучение Flask / #2 - Разработка страниц и добавление стилей: YouTube [Электронный ресурс] / Режим доступа: https://www.youtube.com/watch?v=crUHP8Zo12k&list=PL0lO\_mIqDDFXiIQYjLbncE9Lb6sx8elKA&index=2
8. Изучение Flask / #3 - Работа с базой данных SQLite: YouTube [Электронный ресурс] / Режим доступа: https://www.youtube.com/watch?v=G-si1WbtNeM&t=1s
9. Tailwind CSS classes: Shuffle [Электронный ресурс] / Режим доступа: https://tailwind.build/classes
10. Quickstart: Flask [Электронный ресурс] / Режим доступа: https://flask.palletsprojects.com/en/3.0.x/quickstart/
11. Что такое тестовый набор (тест-свит): TestEngineer [Электронный ресурс] / Режим доступа: <https://testengineer.ru/test-suite/#types>
12. Проекты (работы) курсовые: elib.belstu [Электронный ресурс] / Режим доступа: <https://elib.belstu.by/bitstream/123456789/2098/1/standart-bgtu-12.05.2007.pdf>
13. Стили и направления веб-дизайна в 2024 году: SilverWeb [Электронный ресурс] / Режим доступа: <https://silverweb.by/stili-i-napravleniya-veb-dizajna-v-2024-godu/>
14. Design: Figma [Электронный ресурс] / Режим доступа: <https://www.figma.com/design/xBtCDZYWbtYynUBvMRRnrF/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF-BookHub?node-id=0-1&t=GeZUQza1thhiPHl5-0>

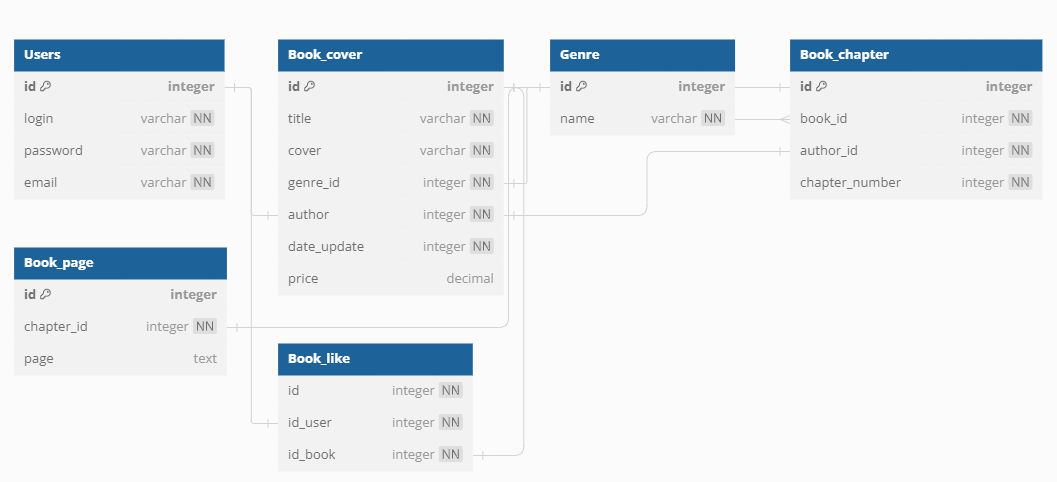
ПРИЛОЖЕНИЕ А

Блок-схема приложения



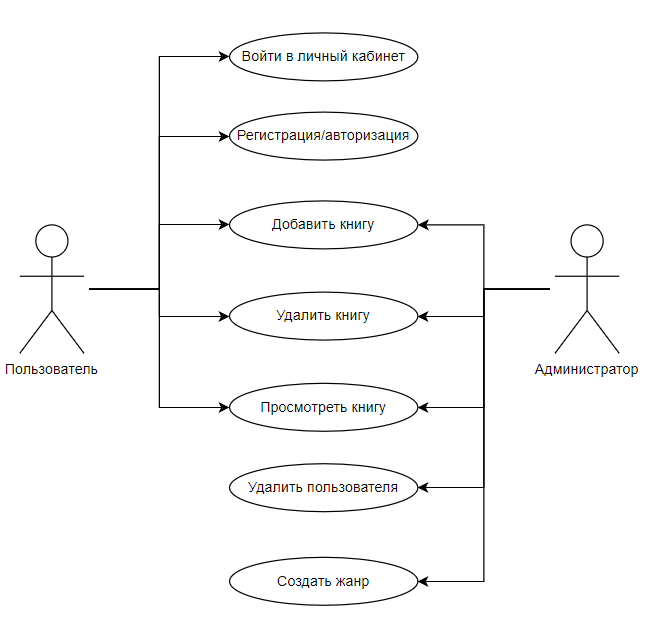
ПРИЛОЖЕНИЕ Б

Логическая схема БД



ПРИЛОЖЕНИЕ В

Диаграмма вариантов использования «BOOKHUB»



ПРИЛОЖЕНИЕ Г

Дизайн-макет «BOOKHUB»

|  |
| --- |
| Рисунок В.1 –Главная Страница |
| Рисунок В.2 –Личный кабинет |
| Рисунок В.3 –Каталог |
| Рисунок В.4 –Страница Книги |