**ГБОУ «Школа № 354 имени Д. М. Карбышева»**

**ГБОУ «Школа № 1748»**

**ГБОУ «Школа № 1584»**

**СОЗДАНИЕ САЙТА ДЛЯ ОБМЕНА КНИГАМИ – SHAREABOOK**

Участники проекта:

ученица 10 класса ГБОУ «Школа № 1748»

Саркисова Ани Кареновна

Руководитель проекта:

Гришина Арина Александровна

**Москва, 2024**

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc185800252)

[1.1 Актуальность темы работы 3](#_Toc185800253)

[1.2 Цель и задачи работы 3](#_Toc185800254)

[ОСНОВНАЯ ЧАСТЬ 4](#_Toc185800255)

[2.1 Определение целевой аудитории и её потребностей 4](#_Toc185800256)

[2.2 Разработка платформы 5](#_Toc185800257)

[Проектирование 5](#_Toc185800258)

[Дизайн 6](#_Toc185800259)

[Разработка 6](#_Toc185800260)

[2.3 Тестирование 12](#_Toc185800261)

[Функциональное тестирование 13](#_Toc185800262)

[Оценочное тестирование 15](#_Toc185800263)

[ЗАКЛЮЧЕНИЕ 16](#_Toc185800264)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 16](#_Toc185800265)

# **ВВЕДЕНИЕ**

SHAREABOOK – проект, позволяющий объединить любителей литературы и способствующий культурному обмену. На готовом сайте пользователи смогут размещать свои книги для обмена, а также искать интересующие их произведения. Простой интерфейс позволит каждому желающему легко ориентироваться в базе данных книг.

**1.1 Актуальность темы работы**

Платформа для обмена книгами является настоящей кладезью возможностей для любителей бумажных книг: для тех, чей бюджет не позволяет выделить средств на книги; для тех, кто хочет заниматься благотворительностью, передавая ненужные книги нуждающимся; для тех, кто ищет произведения-раритеты. Немаловажно, что обмен книгами способствует экологии, снижая потребность в новых ресурсах. В условиях современных экологических проблем, создание подобного сайта становится не просто желанием, а необходимостью, позволяющей сделать чтение бумажных книг доступным для всех.

**1.2 Цель и задачи работы**

Цель – создать платформу с понятным интерфейсом, где каждый сможет создавать и просматривать объявления по обмену книг, а также искать объявления по запросу.

Задачи:

1. Определить целевую аудиторию и её потребности
2. Разработать платформу
3. Провести тестирование

# **ОСНОВНАЯ ЧАСТЬ**

## **2.1** **Определение целевой аудитории и её потребностей**

В ходе обсуждения участниками было решено – нашей целевой аудиторией (далее: ЦА) является молодежь. Для определения потребностей ЦА были проведены опросы среди 50 человек.

В первом опросе была выявлена палитра цветов будущей платформы – оттенки, наиболее ассоциирующиеся с книгами у ЦА (рисунок 1).

Рисунок 1 – опрос, выявляющий цвета, ассоциирующиеся с книгами

Во втором опросе было выявлено средство связи между создателем объявления и откликнувшимся – мессенджер, наиболее удобный для ЦА (рисунок 2).

Рисунок 2 – опрос, выявляющий удобные мессенджеры

## **2.2 Разработка платформы**

### *Проектирование*

### На этом этапе было необходимо продумать структуру сайта, что и было выполнено: была создана схема переходов по страницам будущего сайта, а также UML-диаграмма взаимодействий пользователя (рисунки 3-4).

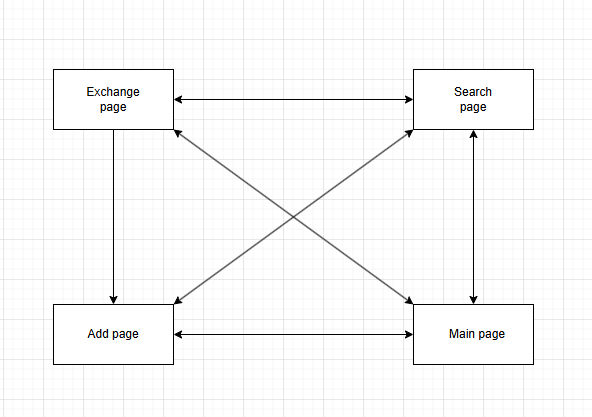


Рисунок 3 – схема переходов по страницам сайта

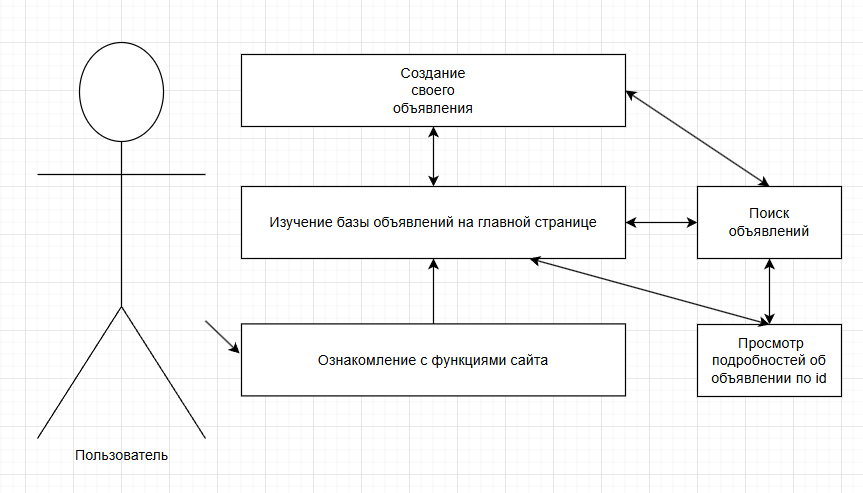


Рисунок 4 – UML-диаграмма взаимодействий пользователя

### *Дизайн*

Для создания дизайна был выбран такой инструмент, как Figma. Figma – онлайн-сервис для разработки интерфейсов и прототипирования, используется как для создания упрощённых прототипов интерфейсов, так и для детальной проработки дизайна интерфейсов, в том числе и для веб-сайтов [1]. Этот инструмент позволил создать интерактивные прототипы: разработка дизайна дала возможность представить минималистичный и функциональный интерфейс, соответствующий современным стандартам UX/UI [2], а также представить визуально функции сайта (рисунок 5). Цветовая палитра была выбрана, как и сказано выше, в соответствии с результатами опроса.

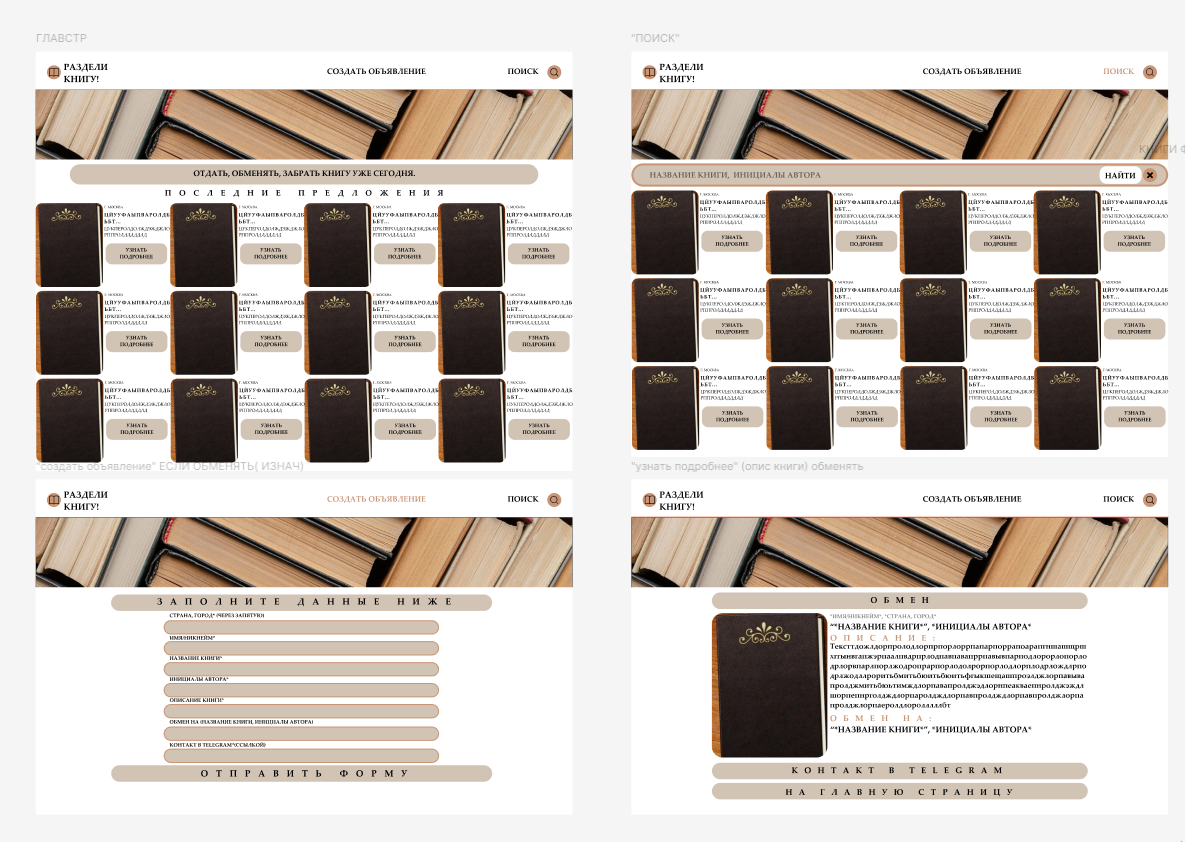


Рисунок 5 – прототипы интерфейсов, созданные с помощью Figma

*Разработка*

Было решено создавать проект в среде разработки Visual Studio Code (далее: VS Code). Для удобства работы был подключен Django – свободный фреймворк для веб-приложений на языке Python [3].

После настройки VS Code частично автоматически, частично вручную были созданы нужные папки и файлы. Далее идет разработка кода, который непосредственно относится к созданному дизайну: в папку templates\main добавляются файлы html – их код был сгенерирован Figma в dev mode, а позже правлен вручную – с добавлением необходимых ссылок и прочих изменений. Создаётся папка ‘static’: в неё подключается CSS файл, описывающий стили ссылок в сайте, а также изображения и файл favicon.ico – значок веб-сайта, отображается браузером во вкладке перед названием страницы, и в качестве картинки рядом с закладкой, а также в адресной строке в некоторых браузерах [4].

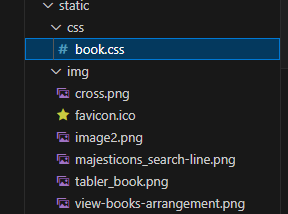


Рисунок 6 – папка ‘static’

body

{

  background-color: white;

}

A:link

{

    text-decoration: none;

    color: black;

}

A:visited

{

    text-decoration: none;

    color: black;

}

A:active

{Продолжение фрагмента на следующей странице…

    text-decoration: none;

    color: black;

}

Рисунок 7 – часть CSS кода, описывающая стиль ссылок

Затем к работе был подключен JavaScript (далее: JS), благодаря ему был осуществлен поиск на странице ‘Search page’:

<script type="text/javascript">

    function ClearField()

    {

        document.getElementById("bookname").value = "";

        const elasticItems = document.querySelectorAll('.card')

        elasticItems.forEach((el) => {

                el.classList.remove('hide')

            })

    </script>

 <div style="left: 65px; top: 420px; position: absolute; width:900px; opacity: 0.50; color: black; font-size: 28px; font-family: Palatino Linotype; font-weight: 700; word-wrap: break-word">

    <input type="text" id="bookname" maxlength=100 size=122 style="line-height:150%; border: 0px solid; background: #D1C4B4; font-size:28px; font-family: Palatino Linotype;" value="НАЗВАНИЕ КНИГИ, ИНИЦИАЛЫ АВТОРА, ГОРОД" onclick="ClearField();"/>

    </div>

    <script type="text/javascript">

    const elastic = document.querySelector('#bookname')

    elastic.addEventListener('input', (ev) => {

        const value = ev.target.value.trim()

        const elasticItems = document.querySelectorAll('.card')

        const searchRegExp = new RegExp(value, 'gi')

        if (value === '') {

Продолжение фрагмента на следующей странице…

            elasticItems.forEach((el) => {

                el.classList.remove('hide')

            })

            return

        }

        elasticItems.forEach((el) => {

            const innerTitle = el.querySelector('.b\_title')

            const innerAuthor = el.querySelector('.b\_author')

            const innerCity = el.querySelector('.b\_city')

            const elementTitle = innerTitle.textContent

            const elementAuthor = innerAuthor.textContent

            const elementCity = innerCity.textContent

            const isContainsSearchRequest1 = searchRegExp.test(elementTitle)

            const isContainsSearchRequest2 = searchRegExp.test(elementAuthor)

            const isContainsSearchRequest3 = searchRegExp.test(elementCity)

            if ((isContainsSearchRequest1)||(isContainsSearchRequest2)||(isContainsSearchRequest3))

            {

                el.classList.remove('hide')

            } else {

                el.classList.add('hide')

            }

        })

    })

    </script>

Рисунок 8-10 – части JS и HTML, сформировавшие поиск

Таким образом, мы подошли к ключевому моменту: к созданию базы данных. Сначала её следовало продумать:

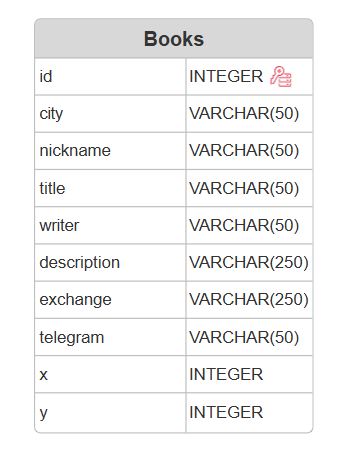
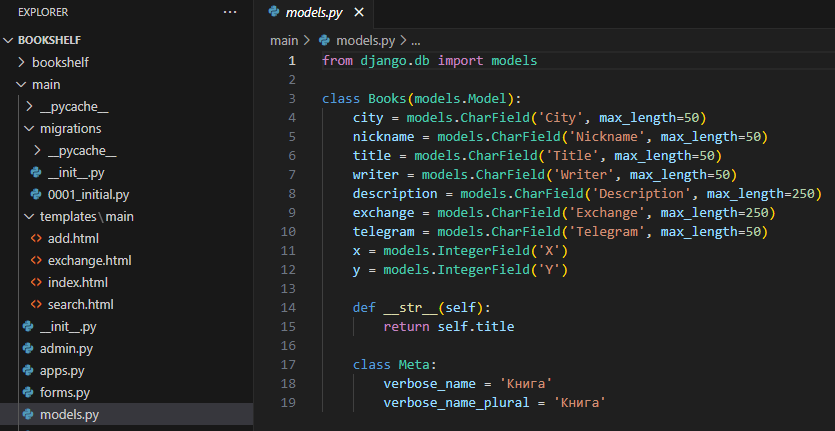


Рисунок 11 – схема базы данных

Для воплощения этой схемы в жизнь было необходимо прописать данную базу и мигрировать, используя terminal:



(нижний фрагмент – из forms.py)

from .models import Books

from django.forms import ModelForm, TextInput

Продолжение фрагмента на следующей странице…

class BooksForm(ModelForm):

    class Meta:

        model = Books

        fields = ['city', 'nickname', 'title', 'writer', 'description', 'exchange', 'telegram']

        widgets = {

            "city": TextInput(attrs={

                'class': 'input\_add',

                'size': '63',

                'maxlength': '63',

            }),

            "nickname": TextInput(attrs={

                'class': 'input\_add',

                'size': '63',

                'maxlength': '63',

            }),

            "title": TextInput(attrs={

                'class': 'input\_add',

                'size': '63',

                'maxlength': '63',

            }),

            "writer": TextInput(attrs={

                'class': 'input\_add',

                'size': '63',

                'maxlength': '50',

            }),

            "description": TextInput(attrs={

                'class': 'input\_add',

                'size': '63',

                'maxlength': '250',

            }),

            "exchange": TextInput(attrs={

                'class': 'input\_add',

                'size': '63',

                'maxlength': '63',

            }),

            "telegram": TextInput(attrs={

Продолжение фрагмента на следующей странице…

                'class': 'input\_add',

                'size': '63',

                'maxlength': '50',

            }),

        }

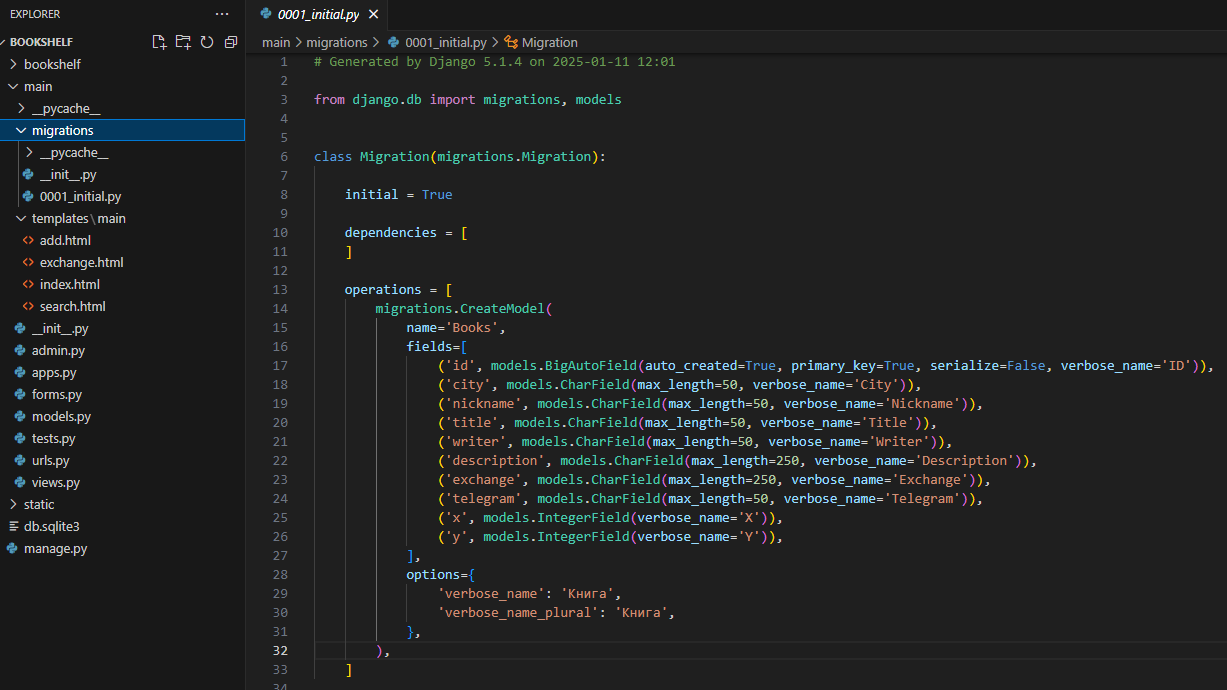


Рисунок 12-14 – миграции, создание базы данных

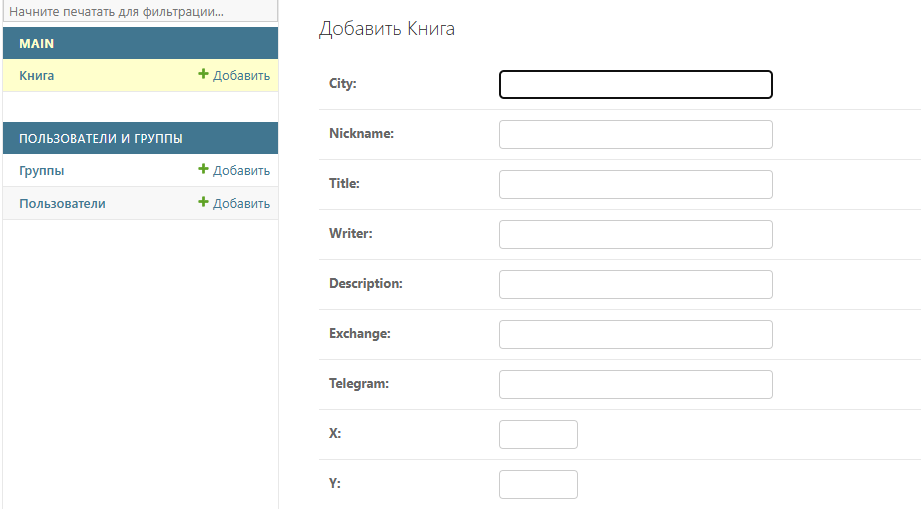
База данных – есть, так она отображена в администраторской панели Django:

Рисунок 15 – итоговая база данных в адм. панели Django

База данных «соединила» форму из страницы ‘Add page’ с выводом новых объявлений: таким образом, сайт обрёл свой полный функционал. С полным кодом программы возможно ознакомиться в открытом репозитории: <https://github.com/sarkisovaani/shareabook> .

## **2.3 Тестирование**

### *Функциональное тестирование*

Этап функционального тестирования — необходимый этап, его цель —подтвердить корректность работы разработанной программы [7].

Результаты функционального тестирования отображены в таблице 1.

Продолжение таблицы на следующей странице…

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № теста | Назначение теста | Значения исходных данных | Ожидаемый результат | Реакция программы | Вывод |
| 1 | Проверка корректности работы кнопки «СОЗДАТЬ ОБЪЯВЛЕНИЕ» | Нажатие на кнопку «СОЗДАТЬ ОБЪЯВЛЕНИЕ» | Ожидается открытие страницы «СОЗДАТЬ ОБЪЯВЛЕНИЕ» | Открытие страницы «СОЗДАТЬ ОБЪЯВЛЕНИЕ» | Программа работает корректно |
| 2 | Проверка корректности работы кнопки «ПОИСК» | Нажатие на кнопку «ПОИСК» | Ожидается открытие страницы «Search page» | Открытие страницы «Search page» | Программа работает корректно |
| 3 | Проверка корректности работы кнопки «УЗНАТЬ ПОДРОБНЕЕ» | Нажатие на кнопку «УЗНАТЬ ПОДРОБНЕЕ» | Ожидается открытие страницы «Exchange page» | Открытие страницы «Exchange page» | Программа работает корректно |
| 4 | Проверка корректности работы кнопки «ОТПРАВИТЬ ФОРМУ» | Нажатие на кнопку «ОТПРАВИТЬ ФОРМУ» | Ожидается открытие главной страницы | Открытие главной страницы | Программа работает корректно |
| 5  Продолжение таблицы на следующей странице… | Проверка корректности работы кнопки «КОНТАКТ В TELEGRAM» | Нажатие на кнопку «КОНТАКТ В TELEGRAM» | Ожидается открытие страницы перехода на аккаунт Telegram | Открытие страницы перехода на аккаунт Telegram | Программа работает корректно |
| 6 | Проверка корректности работы кнопки «НА ГЛАВНУЮ СТРАНИЦУ» | Нажатие на кнопку «НА ГЛАВНУЮ СТРАНИЦУ» | Ожидается открытие главной страницы | Открытие главной страницы | Программа работает корректно |
| 7 | Проверка корректности работы формы «НАЗВАНИЕ КНИГИ, ИНИЦИАЛЫ АВТОРА, ГОРОД» | Заполнение формы «НАЗВАНИЕ КНИГИ, ИНИЦИАЛЫ АВТОРА, ГОРОД» | Ожидается открытие страницы «Search page» с отфильтрованными соответственно запросу данными | Открытие страницы «Search page» с отфильтрованными соответственно запросу данными | Программа работает корректно |

Таблица 1 — результаты функционального тестирования

Тестирование показало, что программа работает без ошибок, то есть корректно.

### *Оценочное тестирование*

После функционального тестирования программу было решено протестировать реальными пользователями. Цель тестирования — выявить, соответствует ли программа нижеуказанным требованиям.

В оценочном тестировании участвовали 5 человек. Участники оценивали платформу по двум критериям: удобство использования и удобство эксплуатации. Выбор этих критериев обоснован их существенным влиянием на восприятие платформы ЦА: удобство использования — степень, в которой интерфейс платформы понимаем, удобство эксплуатации оценивает степень удовлетворённости в режиме реального использования платформы [7].

Результаты оценочного тестирования отображены в таблице 2.

|  |  |  |
| --- | --- | --- |
| № пользователя | Удобство использования | Удобство эксплуатации |
| 1 | 9 | 8 |
| 2 | 8 | 7 |
| 3 | 10 | 10 |
| 4 | 9 | 9 |
| 5 | 7 | 6 |
| Средний показатель | 8,6 | 8 |

Таблица 2 — результаты оценочного тестирования

Шкала оценки была определена следующим образом: «10» — максимальная степень удовлетворения, «1» — минимальная. Тестирование показало высокий уровень удовлетворенности обоими критериями, что говорит о том, что программа понимаема и удобна в использовании ЦА.

**ЗАКЛЮЧЕНИЕ**

В результате выполнения проекта была получена платформа для обмена книгами: функциональная, удобная по мнению пользователей — было выяснено тестированиями. С помощью этой платформы можно создать объявление по обмену книги, отслеживать объявления, а также искать книги по собственному запросу.

# **СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. Figma [Internet] <https://ru.wikipedia.org/wiki/Figma>

2. Шабанов, Д. Ю. "Дизайн пользовательского интерфейса." М.: Издательство, 2020.

3. Django [Internet] https://ru.wikipedia.org/wiki/Django

4. Favicon [Internet] <https://ru.wikipedia.org/wiki/Favicon>

5. Силин, П. А. "Проектирование и разработка веб-приложений." М.: Издательство, 2020.

6. Баранов, С. В. "Основы работы с Django." М.: Издательство, 2021.

7. Степанов, И. А. "Методы тестирования программного обеспечения." М.: Издательство, 2017.