**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра ИС**

**Курсовая работа**

**по дисциплине «Технологии РОД»**

**Тема: Книжный интернет-магазин**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6373 |  | Костищина А.А. |
| Студентка гр. 6373 |  | Хетеева М.Р. |
| Преподаватель |  | Нечитайленко Р.А. |

Санкт-Петербург

2021

**ЗАДАНИЕ**

**на курсовую работу**

|  |  |  |
| --- | --- | --- |
| Студентки Костищина А.А., Хетеева М.Р. | | |
| Группа 6373 | | |
| Тема работы: книжный интернет-магазин | | |
| Исходные данные:  Необходимо разработать сайт книжного интернет-магазина, в который входит отображение списка книг, просмотр информации о конкретной книге, добавление в корзину | | |
| Содержание пояснительной записки:  «Содержание», «Введение», «Требования к контенту и функционалу интернет-магазина», «Информационная архитектура интернет-магазина», «Заключение», «Список использованных источников», «Приложение» | | |
| Предполагаемый объем пояснительной записки:  Не менее 15 страниц. | | |
| Дата выдачи задания: 14.09.2021 | | |
| Дата сдачи реферата: 06.12.2021 | | |
| Дата защиты реферата: 06.12.2021 | | |
| Студентка |  | Костищина А.А. |
| Студентка |  | Хетеева М.Р. |
| Преподаватель |  | Нечитайленко Р.А. |

**Аннотация**

Курсовая работа посвящена разработке сайта интернет-магазина. Основная цель портала – осуществление купле-продажи книг. В работе описаны требования к контенту и функционалу портала, разработан макет интерфейса и сам интерфейс сайта, создана база данных, представление моделей, построена архитектура информационной системы.

**Summary**

Course work is devoted to the development of an online store website. The main goal of the portal is to carry out the sale and purchase of books. In the work, the requirements for the content and functionality of the portal were developed, the interface database and the site interface were developed, data were created, the presentation of models was created, the architecture of the information system was built.

**Содержание**

[ВВЕДЕНИЕ 5](#_Toc89643289)

[1. Цели и задачи 6](#_Toc89643290)

[2. Пользователи интернет-магазина 7](#_Toc89643291)

[3. Описание средств разработки 9](#_Toc89643292)

[4. Информационная архитектура интернет-магазина 11](#_Toc89643293)

[5. Использование цветовой схемы 13](#_Toc89643294)

[6. Страницы интернет-магазина 16](#_Toc89643295)

[6.1 Главная страница 16](#_Toc89643296)

[6.2 Магазин 22](#_Toc89643297)

[6.3 Страница просмотра полной информации о книге 25](#_Toc89643298)

[6.4 Корзина 27](#_Toc89643299)

[ЗАКЛЮЧЕНИЕ 32](#_Toc89643300)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 33](#_Toc89643301)

[ПРИЛОЖЕНИЕ 34](#_Toc89643302)

# ВВЕДЕНИЕ

В эпоху глобальной цифровизации и пандемии взаимодействие с внешним миром все чаще происходит посредством различных веб-сайтов, интернет-магазинов и мобильных приложений. Осуществлять покупки онлайн стало настолько просто, что с этим сможет разобраться любой человек, даже если он находится на начальном уровне пользования ПК. Это не только экономит время, сохраняет возможность лишний раз не выходить из дома, но и удобно с точки зрения выбора в любое время дня и ночи.

Хоть и появилось много возможностей читать книги в электронном формате, что значительно улучшает качество жизни, не нужно носить с собой огромный толстые книги, но все ещё больше половины россиян предпочитают читать бумажные книги. У кого-то не хватает времени, чтобы забежать в свой любимый книжный магазин, кому-то сложно найти книгу в магазинах своего города. Все эти проблемы можно решить с помощью книжного интернет-магазина, что и будет рассмотрено в данной курсовой работе. Далее будут описаны основные цели и задачи данного интернет-магазина, целевая аудитория, спроектирован интерфейс, функционал и база данных.

# Цели и задачи

Ввиду современной повестки дня – пандемия и повсеместное использование гаджетов, коммерчески выгоднее вывести свой магазин сразу в онлайн режиме. Отсюда вытекает основная цель создания книжного интернет-магазина:

1. предоставление возможности приобрести книги в онлайн-формате с удобными способами доставки.

Для достижения поставленных целей, необходимо решить следующие задачи:

1. просмотр всех товаров интернет-магазина;
2. отображение товаров по категориям;
3. просмотр информации о конкретном товаре;
4. добавление товара в корзину;
5. оформление заказа;
6. обеспечить возможность получить книги в пунктах самовывоза или заказать доставку на дом.

# Пользователи интернет-магазина

Пользователями интернет-магазина являются физические лица, желающие приобрести книги: любители чтения, школьники и студенты. Для формирования требований к функционалу и наполняемости портала необходимо рассмотреть потребности различных Пользователей, их интересы и опыт пользования интернет-ресурсами.

Пользователь 1

Пол: мужской

Возраст: 60 лет

Высшее образование, женат, один ребенок, два внука, начальный уровень пользования ПК

Потребности: хочет порадовать внуков подарками, но всё время забывает какую именно серию книг они читают, какие книги уже есть.

Пользователь 2

Пол: женский

Возраст: 20 лет

Студентка, уверенный пользователь ПК

Потребности: хочется купить новую книгу любимого автора, но в книжном магазине ее постоянно раскупают.

Пользователь 3

Пол: мужской

Возраст: 15 лет

Школьник, все свободное время проводит за компьютером

Потребности: ложась спать вспомнил, что на завтрашний урок нужно принести контурные карты, рано утром и поздно вечером книжные магазины не работают.

Пользователь 4

Пол: женский

Возраст: 26 лет

Мама в декрете, два ребенка, уверенный пользователь ПК

Потребности: отсутствие возможности отлучиться надолго из дома, чтобы купить детям развивающие книги и игрушки.

Количество Пользователей интернет-магазина – 10 000 человек, средний возраст Пользователя – 34 года, 58% – женщины, 42% – мужчины. Пользователи заходят в интернет-магазин сидя дома, соответственно необходимо разработать дизайн под компьютеры, ноутбуки. Так как Пользователями являются люди разных возрастов и разных навыков пользования ПК – базовый и продвинутый, следует спроектировать продукт с интуитивно понятным интерфейсом.

# Описание средств разработки

Для работы над созданием интернет-магазина необходимо сначала разобраться какие средства разработки будут использованы.

Так как итоговый формат – это веб-сайт, то не обойтись без стандартизированного языка разметки для просмотра веб-страниц HTML. Он используется, чтобы сообщить браузеру как отображать страницы, которые мы посещаем. Далее воспользуемся каскадной таблицей стилей CSS, которая определяет как созданные HTML-элементы должны отображаться. На этом шаге определяем: используемые шрифты, цвет, фоновые изображения, стили кнопок и многое другое. Также на сайте будет немного задействован JavaScript – это язык программирования, используемый для изменения содержимого веб-страниц, в конкретном случае для отображения модальных окон, пролистывания, пагинации. Чтобы сократить время на «украшение» веб-страниц нашего интернет-магазина был использован открытый и бесплатный фреймворк Bootstrap 5. Из его плюсов можно выделить не только высокую скорость создания сайтов, но и кроссбраузерность, кроссплатформенность, создание адаптивной верстки, однородность дизайна и его согласованность между компонентами.

Если говорить уже о реализации функционала интернет-магазина, то здесь был использован такой веб-фреймворк как Django версии 3.0. Django это высокоуровневый веб-фреймворк Python, способствующий быстрой разработке и созданию прагматичного, чистого кода. Фреймворк – это набор компонентов, необходимых в любых веб-приложениях. Разберем его положительные стороны, ради которых он и был выбран для разработки:

* хорошая и понятная документация;
* безопасность – Django по умолчанию предотвращает многие распространенные ошибки безопасности;
* масштабируемость – Django по своей сути является набором готовых к использованию компонентов, которые не зависят друг от друга, их можно отключать и заменять другими;
* множество полезных пакетов и утилит.

Фреймворк Django написан на языке программирования Python, поэтому его структура соответствует особенностям языка. Создатели реализовали в Django [паттерн MVC](https://ru.hexlet.io/blog/posts/chto-takoe-mvc-rasskazyvaem-prostymi-slovami) (расшифровывается как модель-представление-контроллер, от англ. model-view-controller), и он применяется в текущей версии фреймворка.

Архитектура MVC позволяет разработчику работать с визуальным представлением и бизнес-логикой приложения отдельно. Кстати, при работе с Django специалисты чаще используют термин MVT — Model-View-Template или модель-представление-шаблон. Компоненты MVT можно использовать независимо друг от друга.

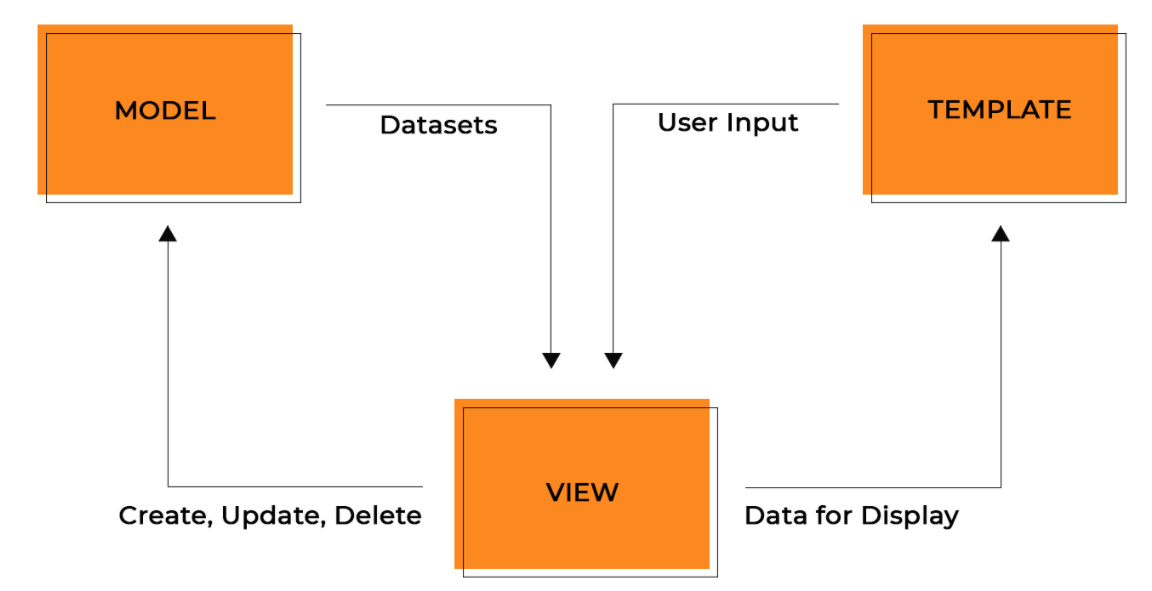


Рисунок 1 – Схема архитектуры MVT в Django

Так как для работы сайта необходима своя база данных, то была использована SQLite3, которую поддерживает Django.

# Информационная архитектура интернет-магазина

Перед началом разработки необходимо спроектировать схему интернет-магазина (см. Рисунок 2). Это поможет определиться с очередностью и приоритетом работ.

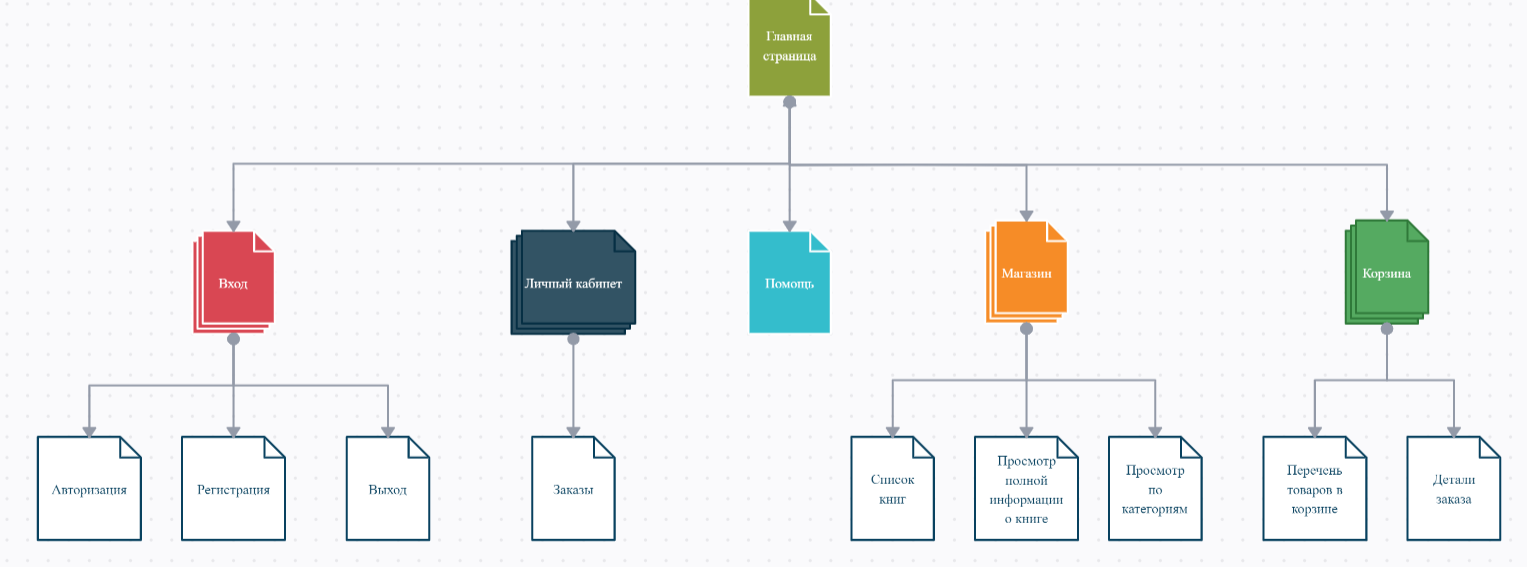
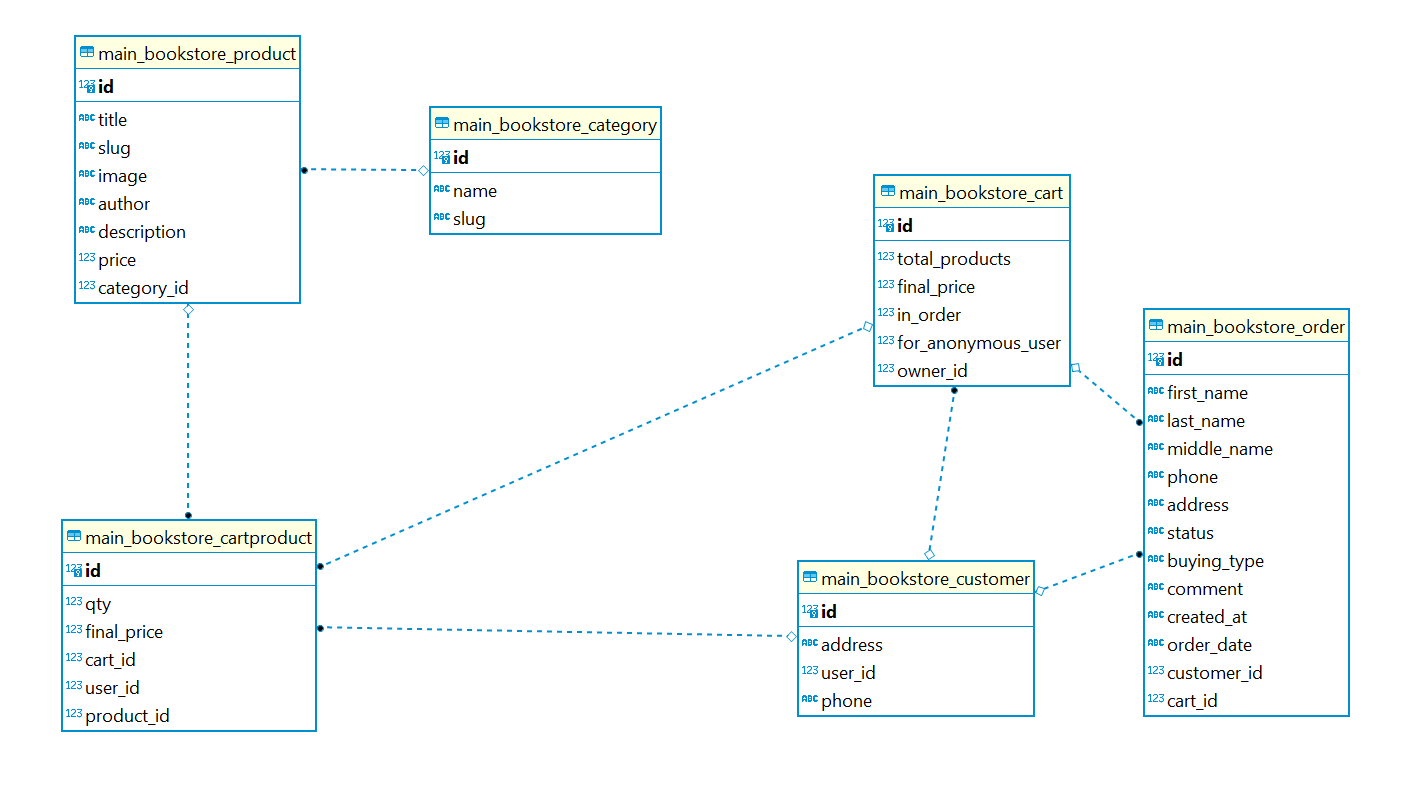


Рисунок 2 – Схема интернет-магазина

Как видно из схемы, необходимо разработать главную страницу, с которой будет произведен переход к остальному функционалу сайта, такому как авторизация/регистрация, выход, просмотр заказов авторизованного пользователя, отображение магазина товаров, в который входит просмотр не только всего перечня, но и детальный просмотр определенной книги, отображение по категориям, а также дальнейший переход к корзине.

Так как с функционалом интернет-магазина уже определились можно переходить к созданию базы данных и таблиц. Необходимо создать такие модели, как: покупатели, заказы, корзина покупателя, категории товаров, продукты и карточку продукта – эта модель подразумевает под собой промежуточный продукт, который должен относиться только к корзине. Реализация будет рассмотрена в разделе 7. Далее представлена диаграмма сущностей (см. Рисунок 3).



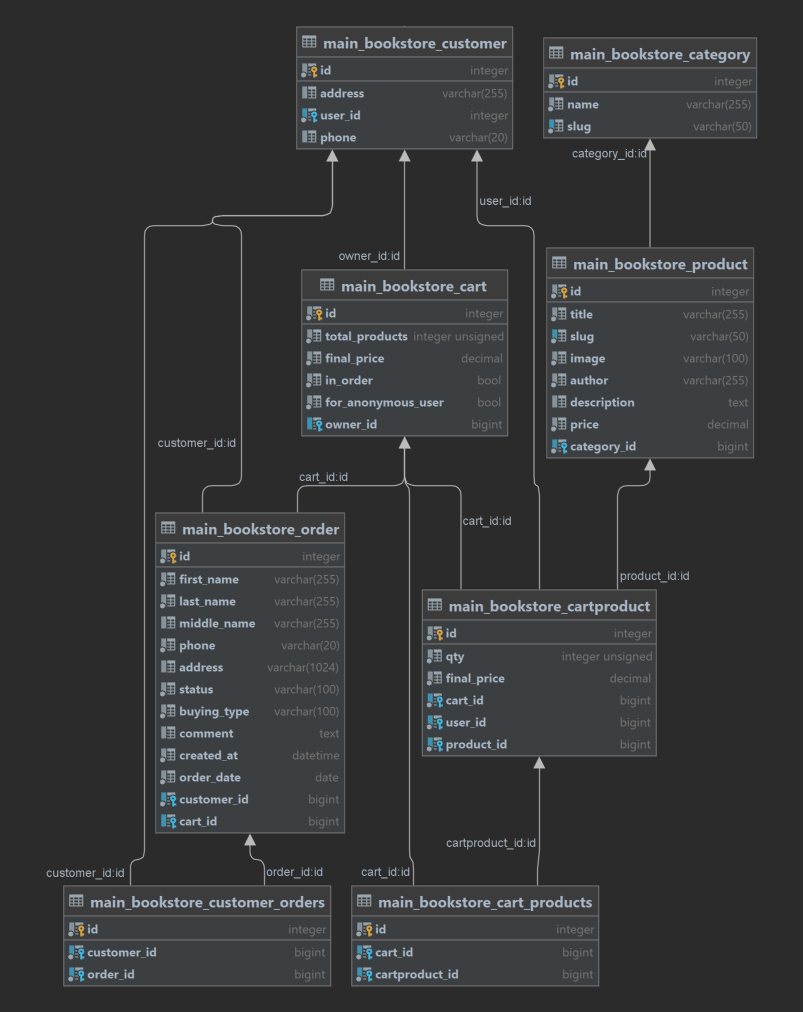


Рисунок 3 – Диаграмма сущностей

# Использование цветовой схемы

Макеты были разработаны в серых тонах специально, чтобы показать, как элементы контрастируют друг с другом. Также стоит не забывать о том, что при проектировании сайта необходимо быть ориентированным на людей в том числе с различными дефектами зрения. Для некоторых необходимы сайты, которые не потеряют части своего функционала, если их перевести в черно-белый режим. Для дальнейшей реализации интернет-магазина ниже представлена цветовая схема сайта, подходящая по контрасту.

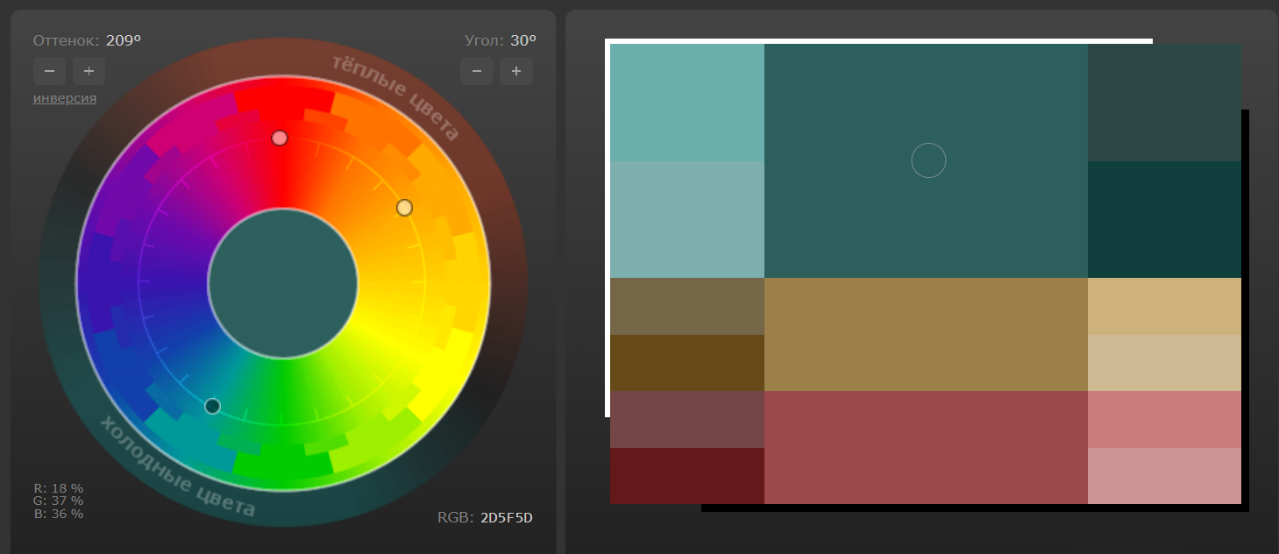


Рисунок 4 – Цветовая схема

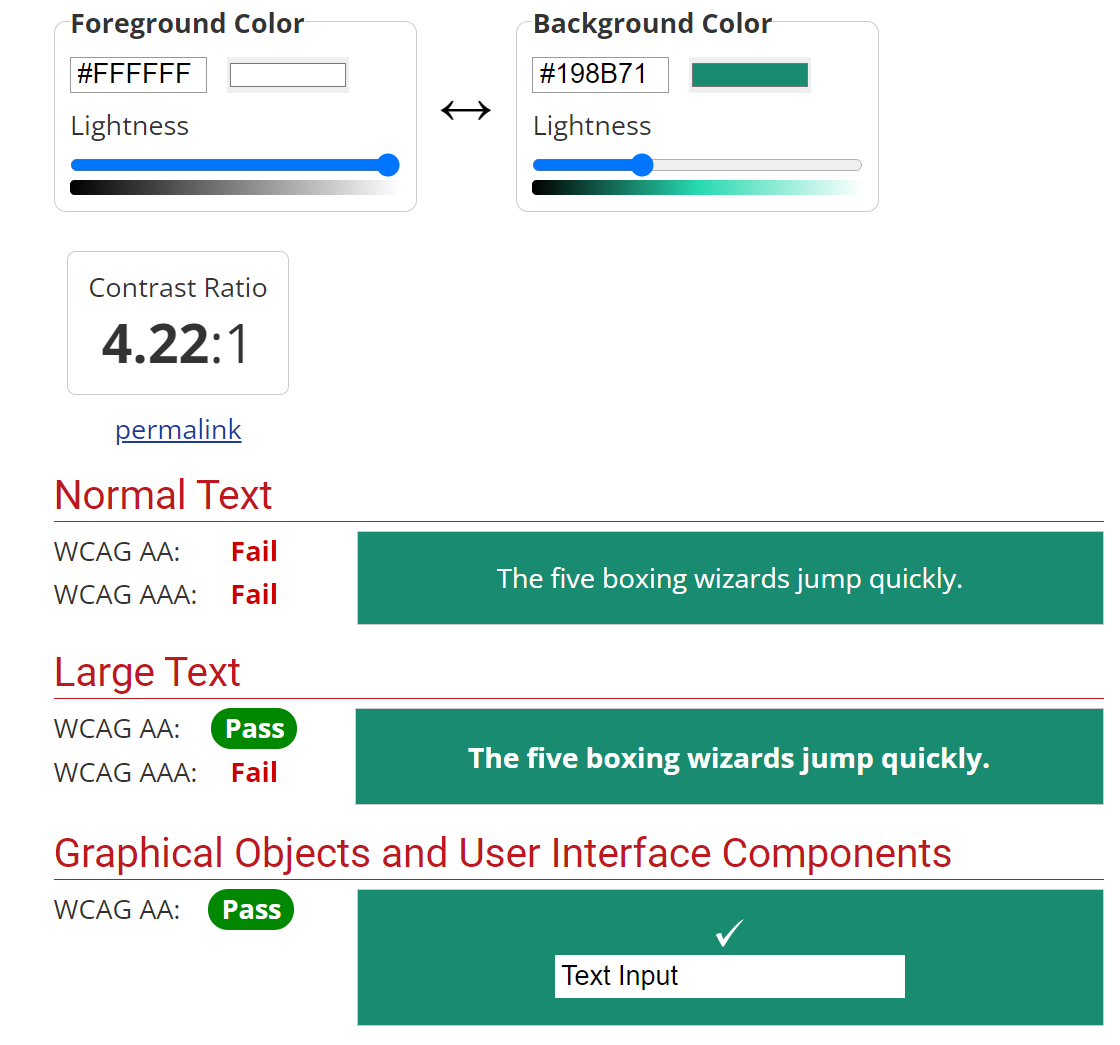


Рисунок 5 – Сочетание цветов (1)

Исходя из этого цветового сочетания видно, что лучше изначально использовать жирное выделение шрифта.

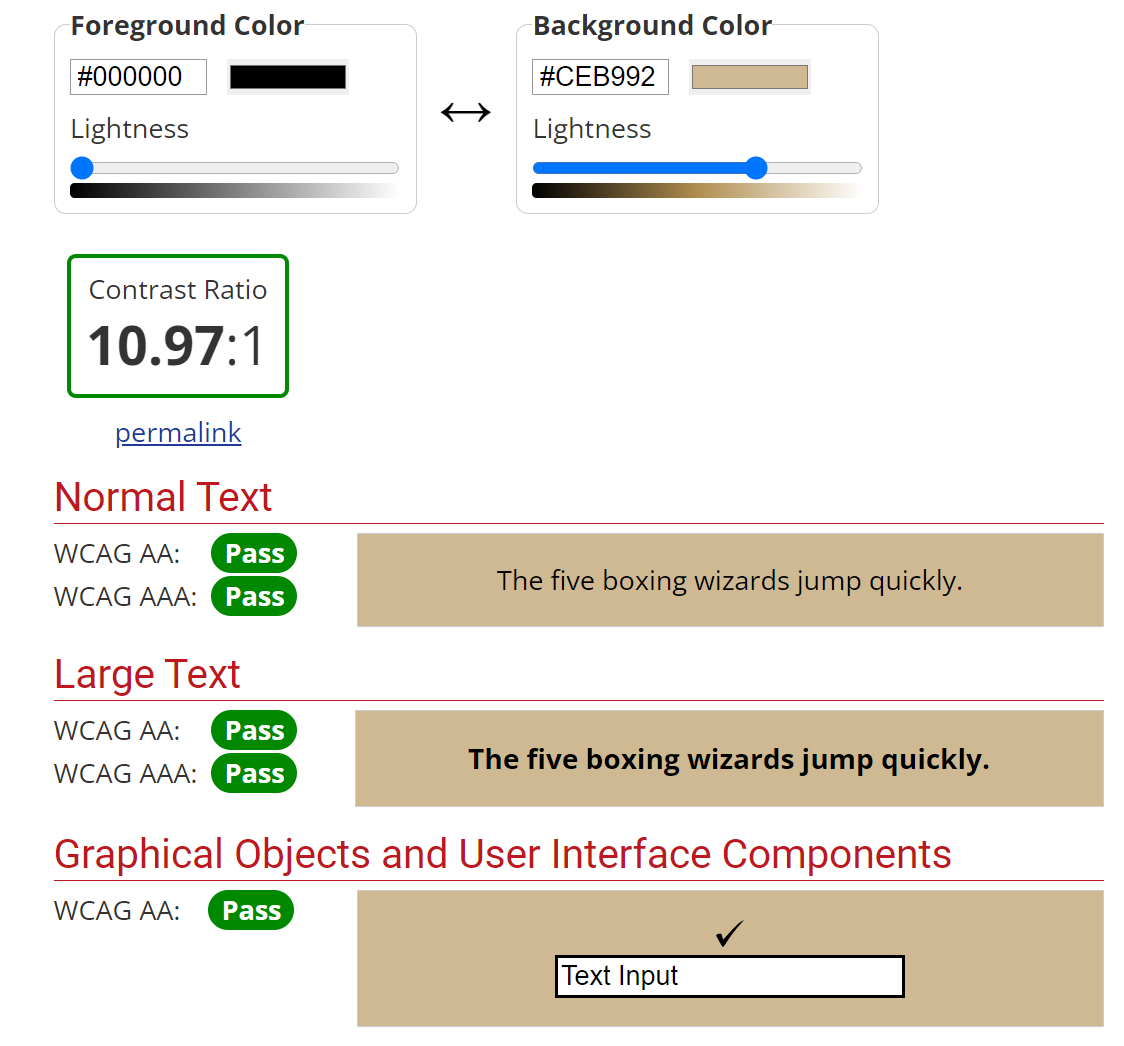


Рисунок 6 – Сочетание цветов (2)

# Страницы интернет-магазина

## Главная страница

Рабочая область Главной страницы интернет-магазина должна состоять из следующих блоков:

* шапка;
* вступительное слово;
* новые поступления книг;
* информация о владельцах магазина;
* подвал.

Шапка и подвал будут одинаковыми на всех страницах интернет-магазина, поэтому рассмотрим их только в этом пункте, далее не будем повторяться. Интерфейс главной страницы представлен на рисунке 7.

В шапке представлена информация о названии и логотипе интернет-магазина, кнопки авторизации/регистрации пользователя, переход на страницы магазина и в корзину.

В подвале отображается кнопка Помощи пользователю (при нажатии отображается модальное окно с часто-задаваемыми вопросами – см. Рисунок 8), логотип и ссылки на социальные сети, которые ведут представители интернет-магазина.

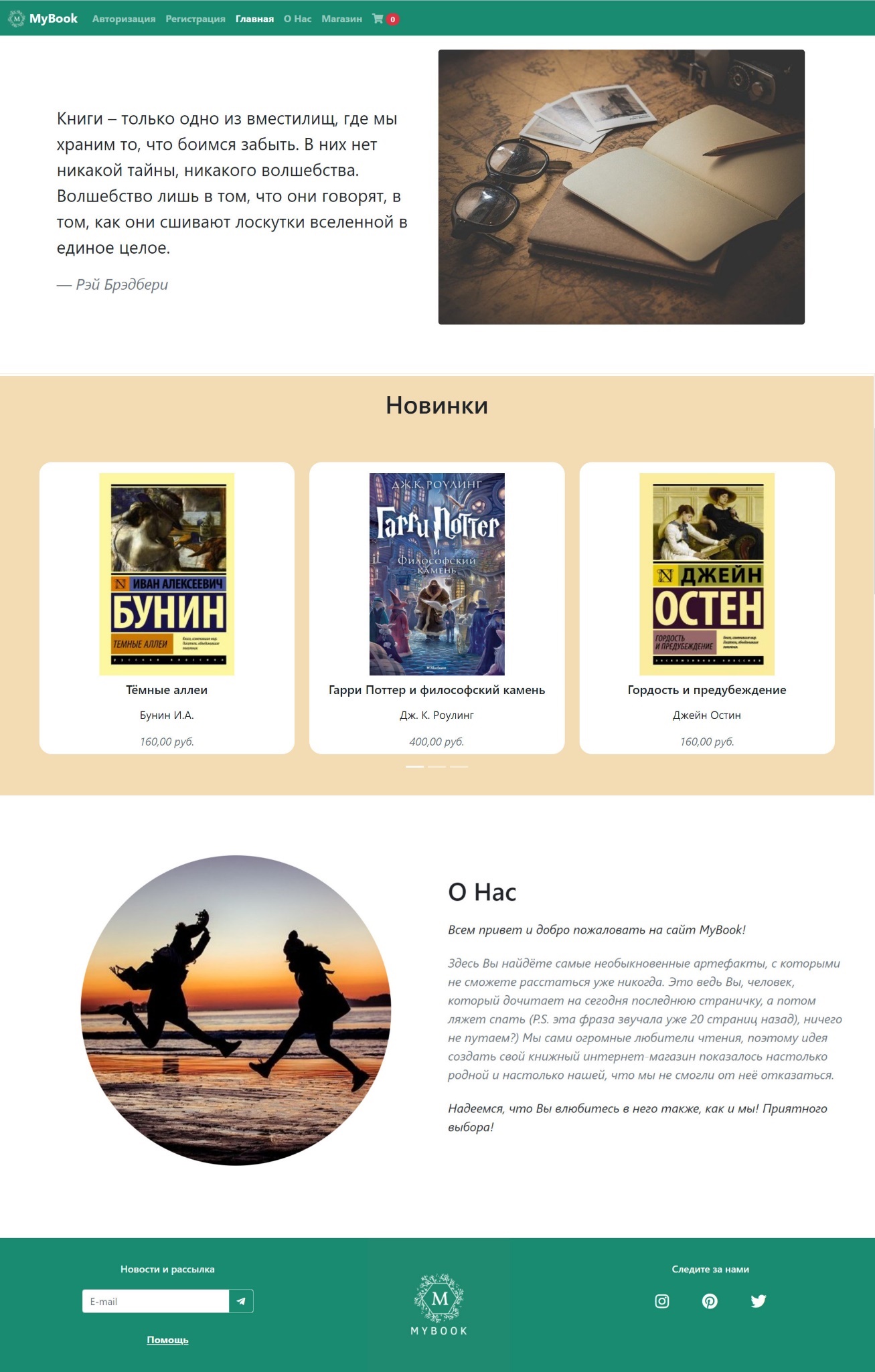


Рисунок 7 – Отображение главной страницы сайта

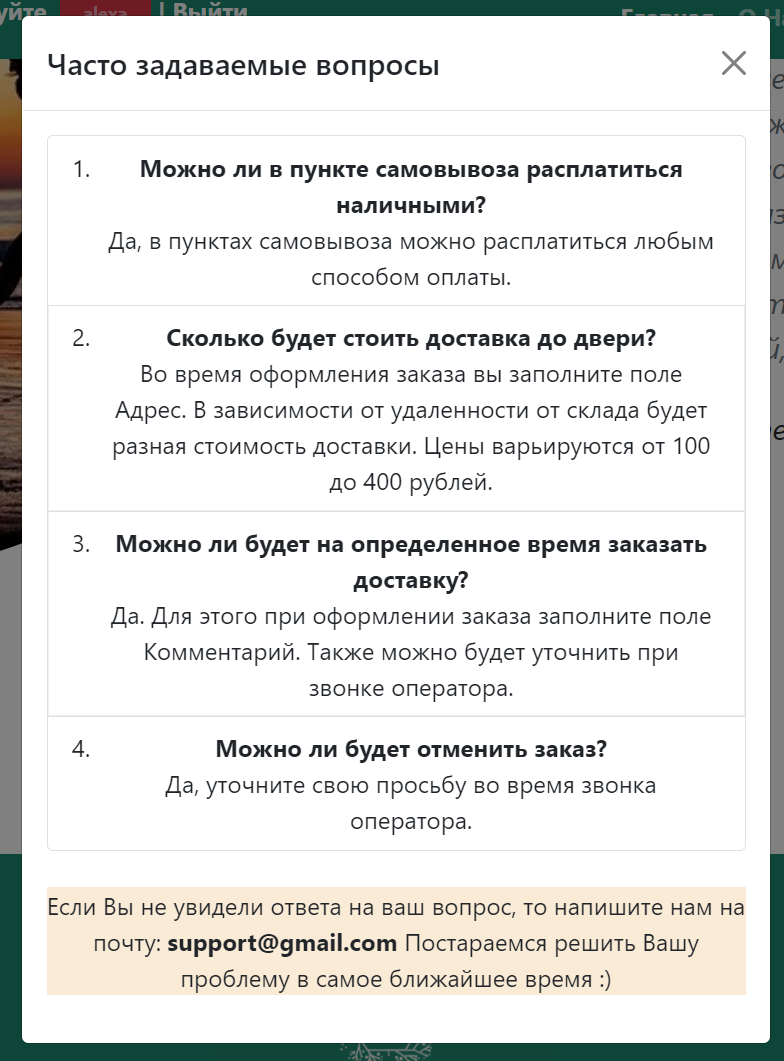


Рисунок 8 – Модальное окно Помощи

При нажатии на кнопку «Авторизация» происходит переход на страницу авторизации (см. Рисунок 9). Чтобы зайти в Систему пользователю необходимо ввести свой логин и пароль (поля являются обязательными). В файле с формами (см. Приложение 2) прописали логику – проверка вводимых данных на соответствие данных в системе: есть ли пользователь с таким логином, правильный ли пароль для этого логина, значения не пустые. Если запись в Системе найдена, то авторизуем пользователя и перенаправляем его на главную страницу, если нет, то отображаем сообщение об ошибке. Ограничения для логина: «Обязательное поле. Не более 150 символов. Только буквы, цифры и символы @/./+/-/\_». Возможные ошибки представлены на рисунках 10-11.

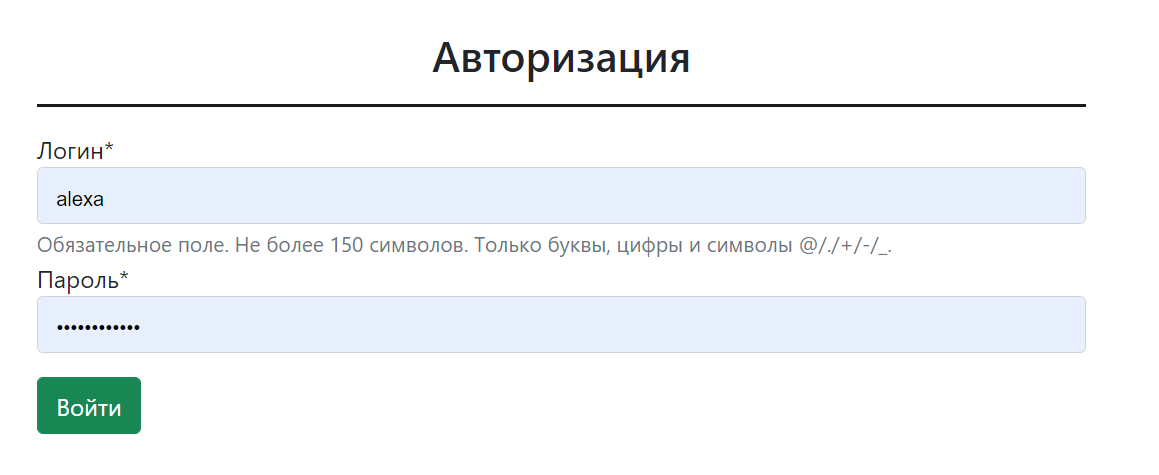
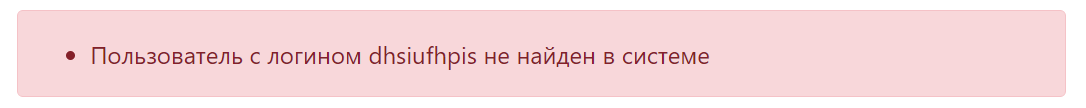


Рисунок 9 – Страница Авторизации пользователя

 Рисунок 10 – Ошибка Авторизации. Пользователя с таким логином нет в Системе

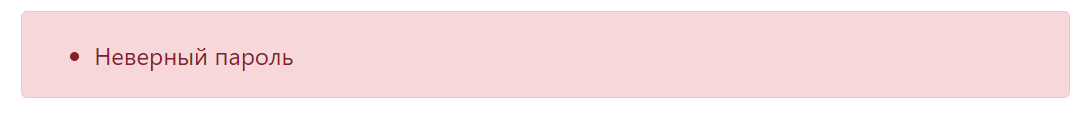


Рисунок 11 – Ошибка Авторизации. Неверный пароль

При нажатии на кнопку «Регистрация» происходит переход на страницу регистрации (см. Рисунок 12). Чтобы зайти в Систему пользователю необходимо заполнить обязательные поля «Логин», «Пароль», «Подтвердите пароль», «E-mail», и по желанию необязательные. В файле с формами (см. Приложение 2) прописали логику – проверка вводимых данных: нет ли пользователя с таким логином, нет ли Пользователя с таким же e-mail, одинаковые пароли, значения не пустые. Если запись в Системе не найдена, то регистрируем пользователя и перенаправляем его на главную страницу, если нет, то отображаем сообщение об ошибке. Ограничения для логина: «Обязательное поле. Не более 150 символов. Только буквы, цифры и символы @/./+/-/\_». Возможные ошибки представлены на рисунках 13-14.



Рисунок 12 – Страница Регистрации пользователя



Рисунок 13 – Ошибка Регистрации. Пользователь с таким логином уже существует



Рисунок 14 – Ошибка Регистрации. Пароли не совпадают

После авторизации/регистрации в шапке страницы отображается надпись с кнопкой выхода «Здравствуйте, <логин пользователя> | Выйти». При нажатии на логин происходит переход в личный кабинет пользователя, где отображается информация о его заказах (см. Рисунок 15) или же информация об отсутствии заказов с ссылкой-переходом на страницу магазина (см. Рисунок 16). Также при просмотре заказов отображается кнопка «Дополнительно», при нажатии отображается модальное окно с детализацией заказа (см. Рисунок 17)

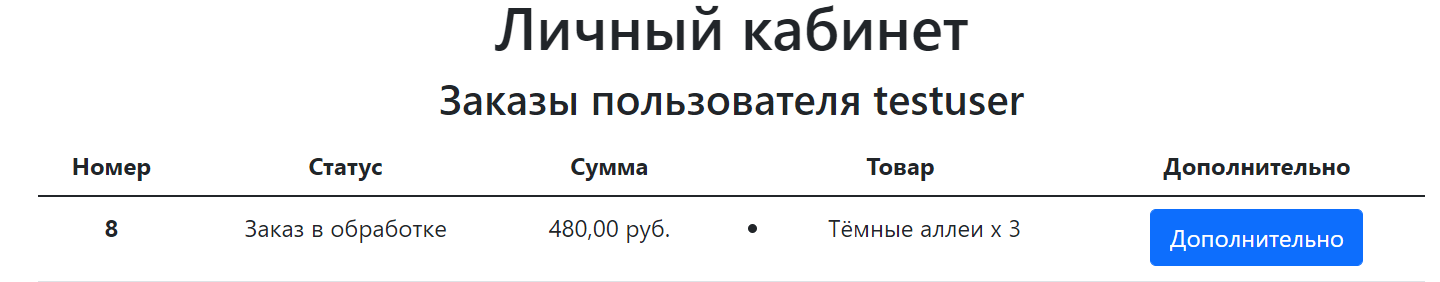


Рисунок 15 – Личный кабинет пользователя с историей заказов

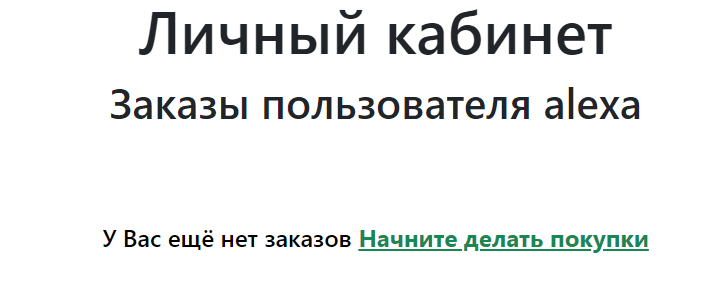


Рисунок 16 – Личный кабинет без заказов с ссылкой на магазин

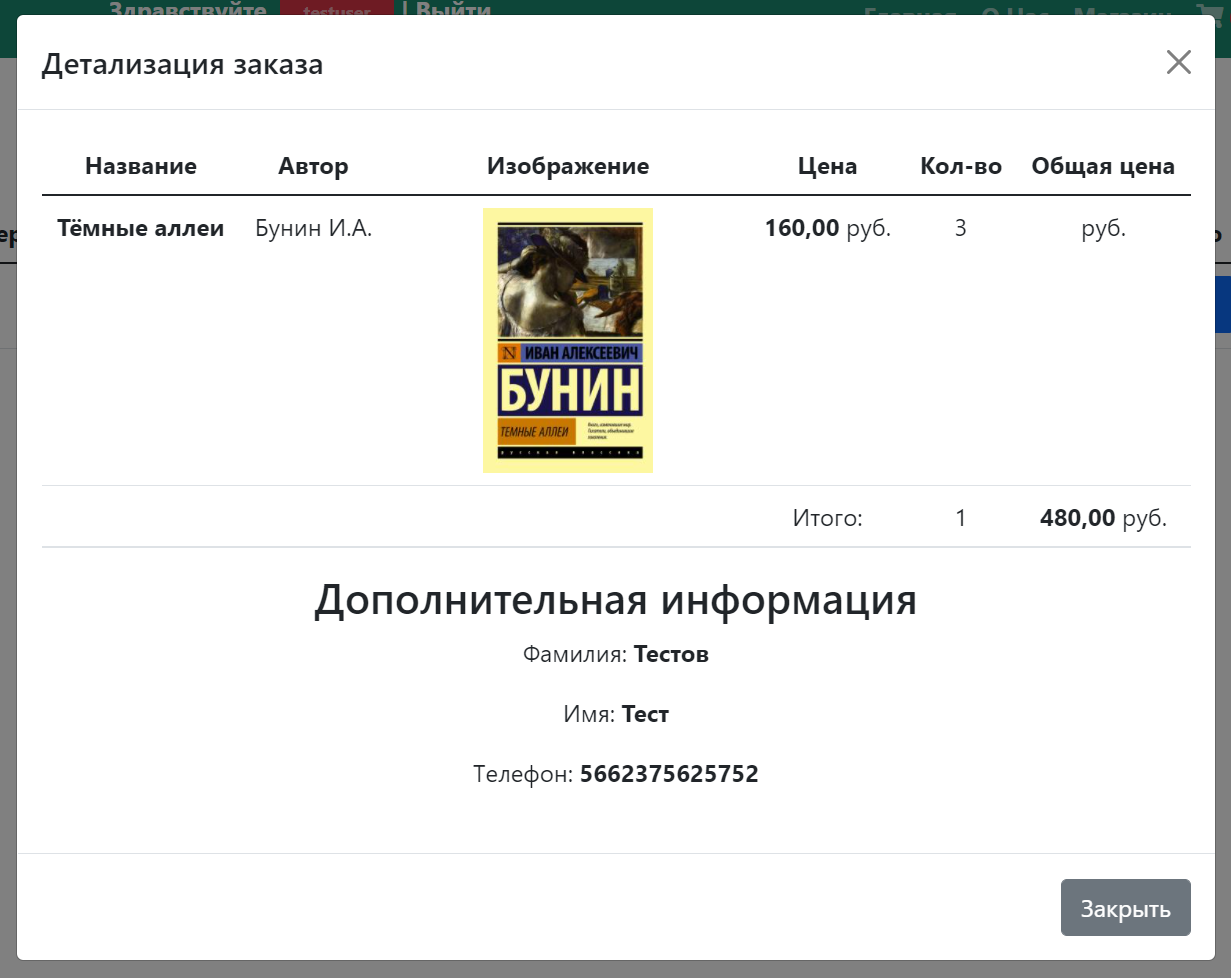


Рисунок 17 – Детализация заказа

## Магазин

Рабочая область страницы Магазина должна состоять из блоков:

* шапка – активная ссылка «Магазин»;
* заголовок страницы – «Книжный магазин MyBook»;
* список книг;
* категории;
* подвал.

Шапка и подвал были описаны в разделе Главная страница.

Интерфейс представлен на рисунке 18.

Отображение списка книг происходит по 2 в ряд, по 4 на каждой странице (в конце страницы сделана пагинация). Каждый элемент списка – карточка книги – состоит из: обложки книги, названия, автора произведения, цены и кнопки «В корзину» (при нажатии происходит переход на страницу корзины – раздел 6.4). Чтобы просмотреть полную информацию о книге необходимо нажать на ее изображение или название (см. раздел 6.3). По умолчанию отображены все книги из всех категорий.

Справа от списка книг отображаются наименования категорий и количество книг в каждой из них. При нажатии на название происходит переход на страницу соответствующей категории (см. Рисунок 19). При просмотре страницы отображается название категории по центру и список книг относящейся к ней. Список состоит из тех же элементов, только уже по 3 в ряд, что и при просмотре магазина, то есть также можно добавить книгу в корзину, отобразить полную информацию о книге.

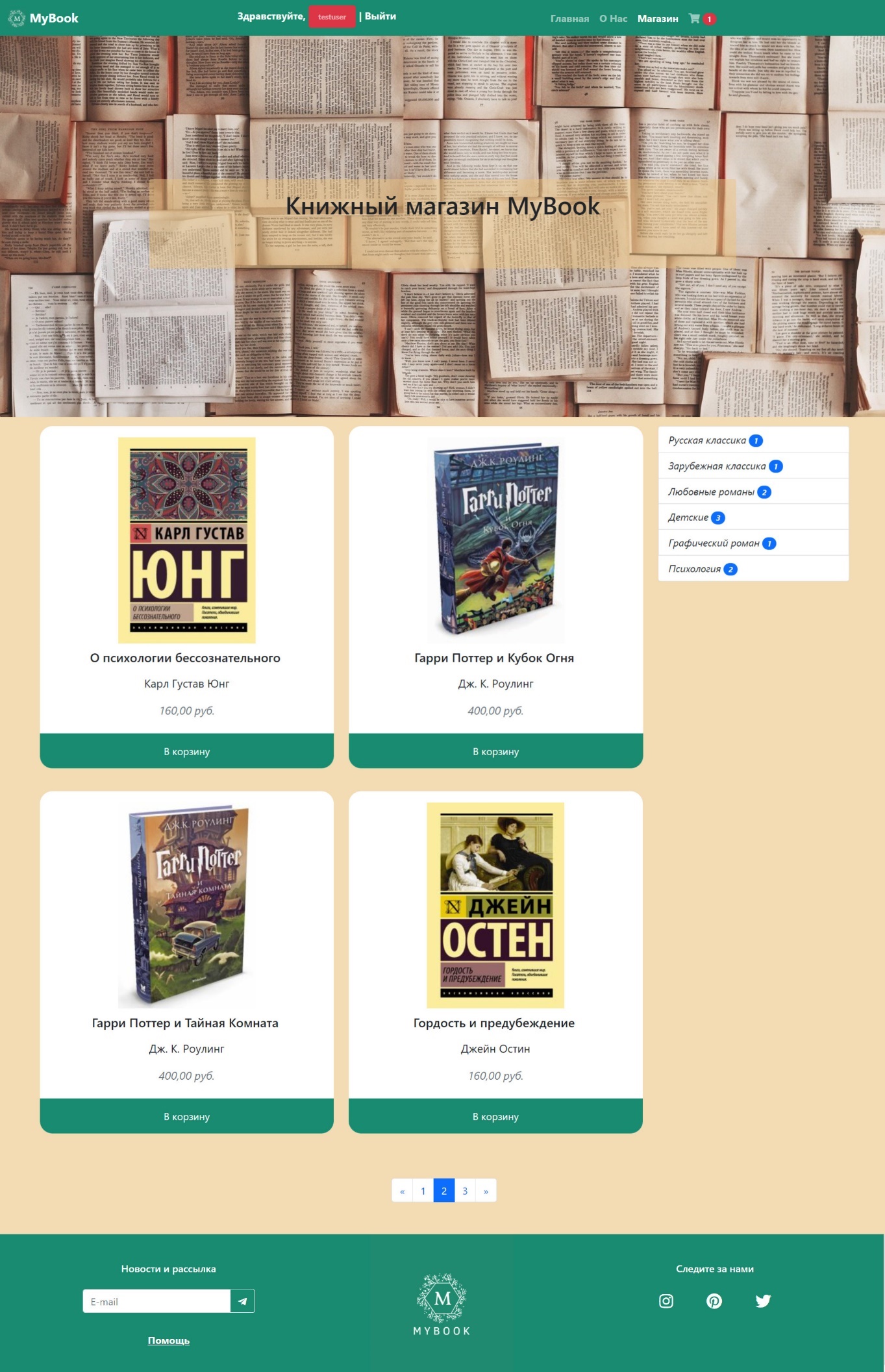


Рисунок 18 – Интерфейс страницы магазина

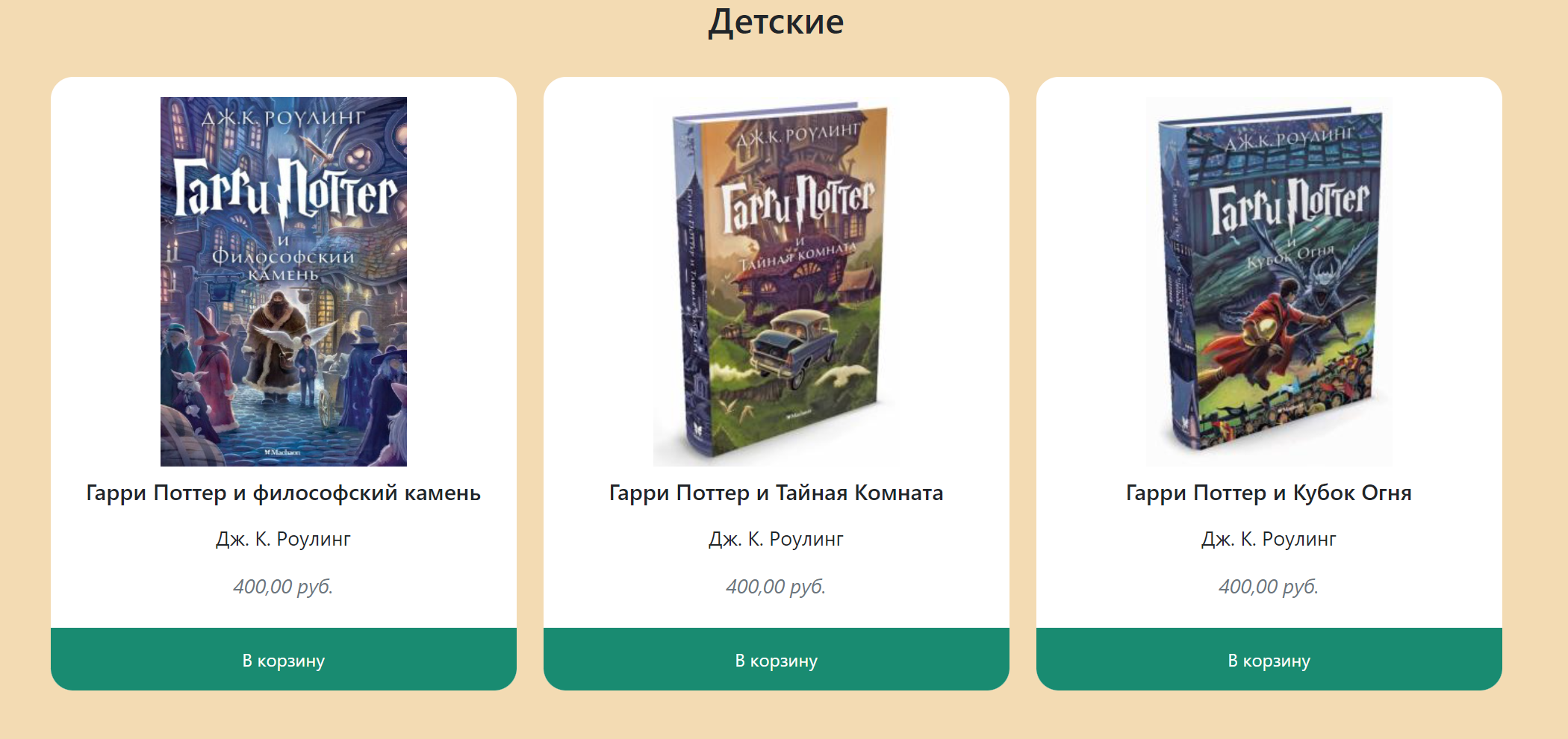


Рисунок 19 – Просмотр отдельной категории

## Страница просмотра полной информации о книге

Рабочая область страницы просмотра полной информации о книге должна состоять из блоков:

* шапка;
* заголовок страницы – «Книжный магазин MyBook»;
* информация о книге;
* подвал.

Шапка и подвал были описаны в разделе Главная страница. Назначением страницы является просмотр полной информации о выбранной книге. Интерфейс страницы представлен на рисунке 20.

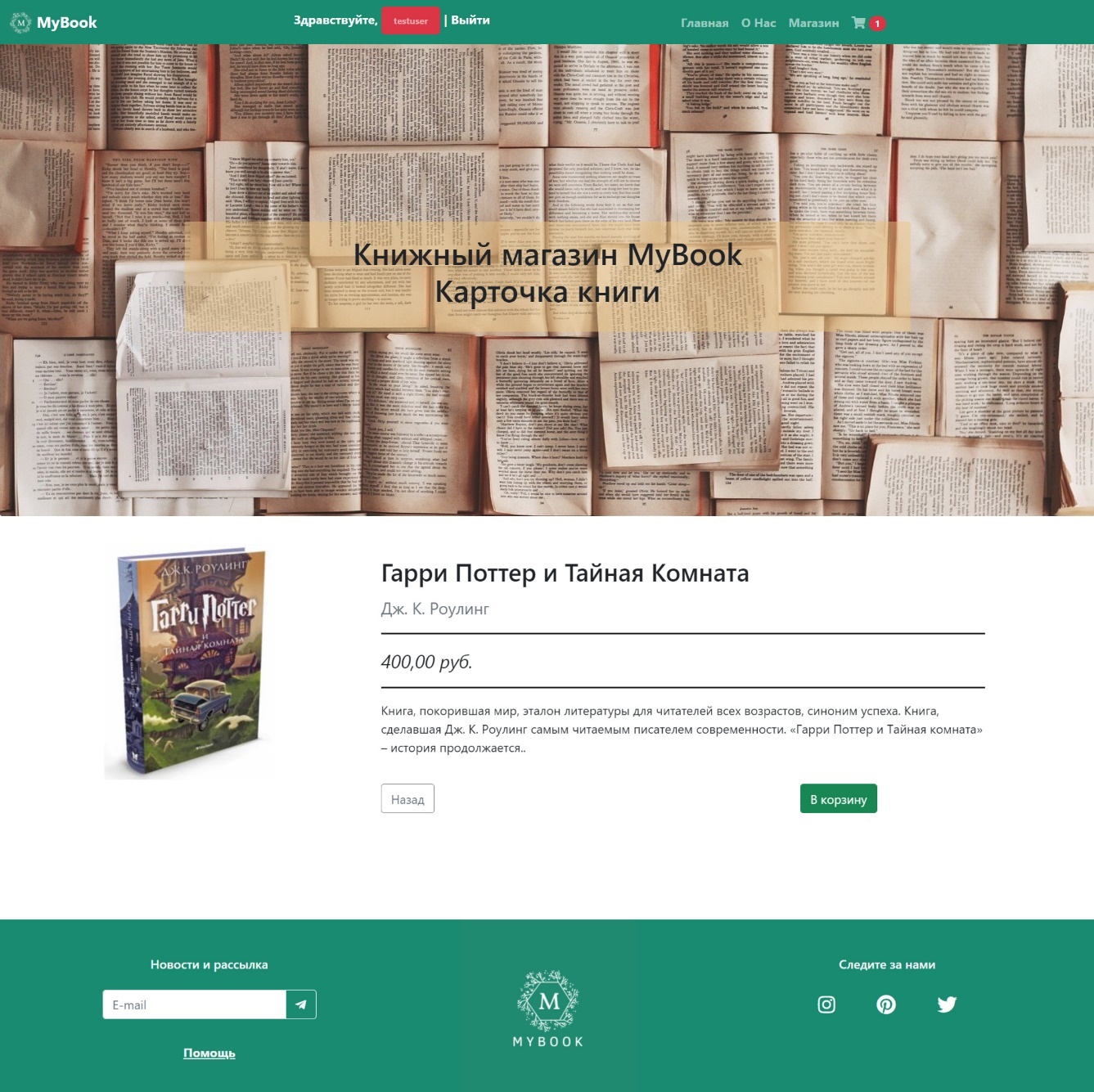


Рисунок 20 – Интерфейс страницы просмотра полной информации о книге

Рассмотрим подробнее блок страницы:

1. Информация о книге:
   1. обложка, название, автор, цена, описание книги;
   2. активные элементы управления - добавление в корзину (при нажатии на кнопку «В корзину»), возврат назад к просмотру списка книг (при нажатии на кнопку «Назад»).

## Корзина

Страница Корзины открывается, когда Пользователь хочет купить какой-либо определенный товар из интернет-магазина и нажимает на кнопки «В корзину» в момент просмотра списка книг, или на иконку «Корзина».

Покупка состоит из нескольких этапов:

* просмотр корзины (см. рисунок 21);
* детали заказа (см. рисунок 22).

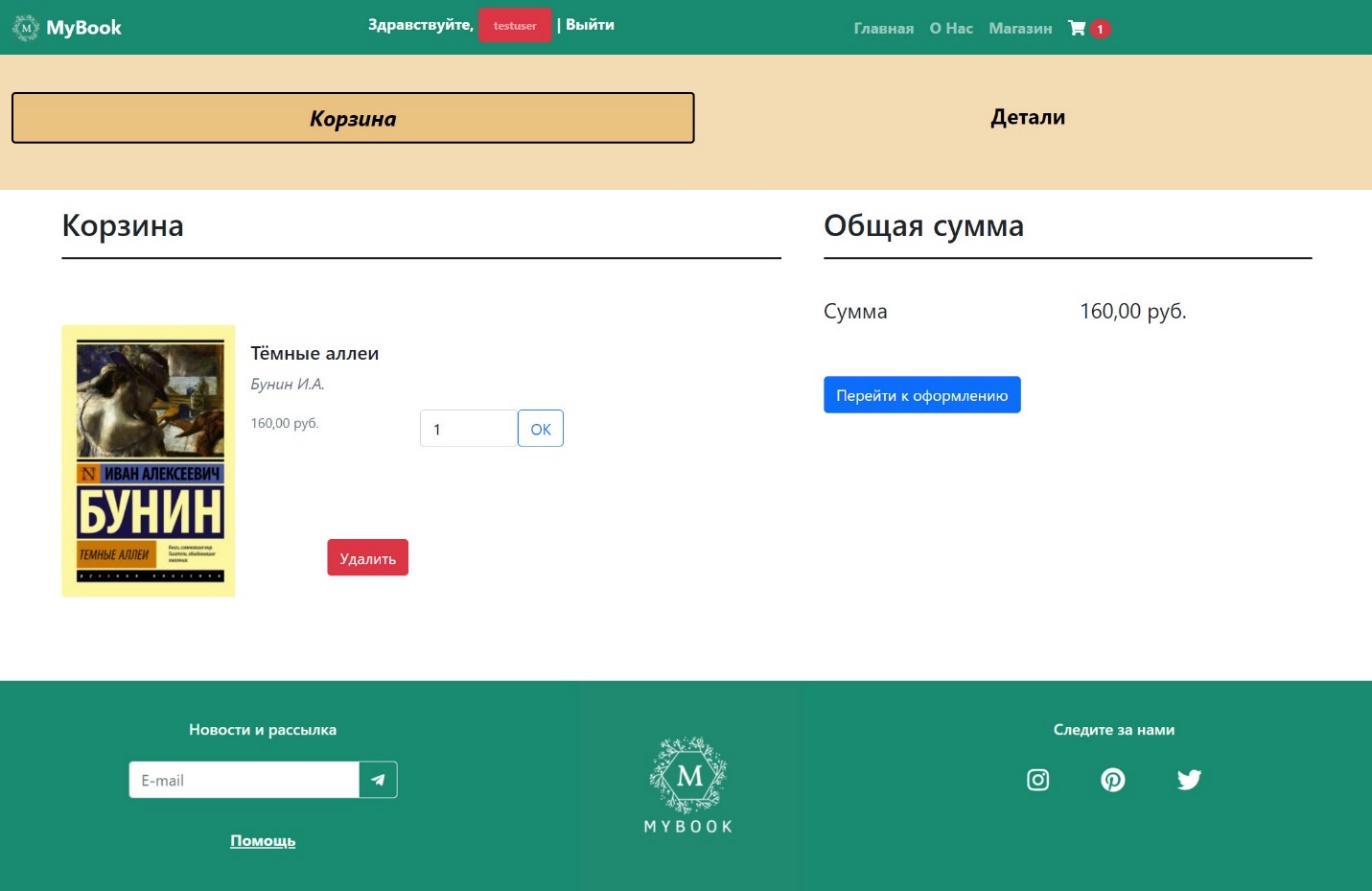


Рисунок 21 – Интерфейс страницы Корзина

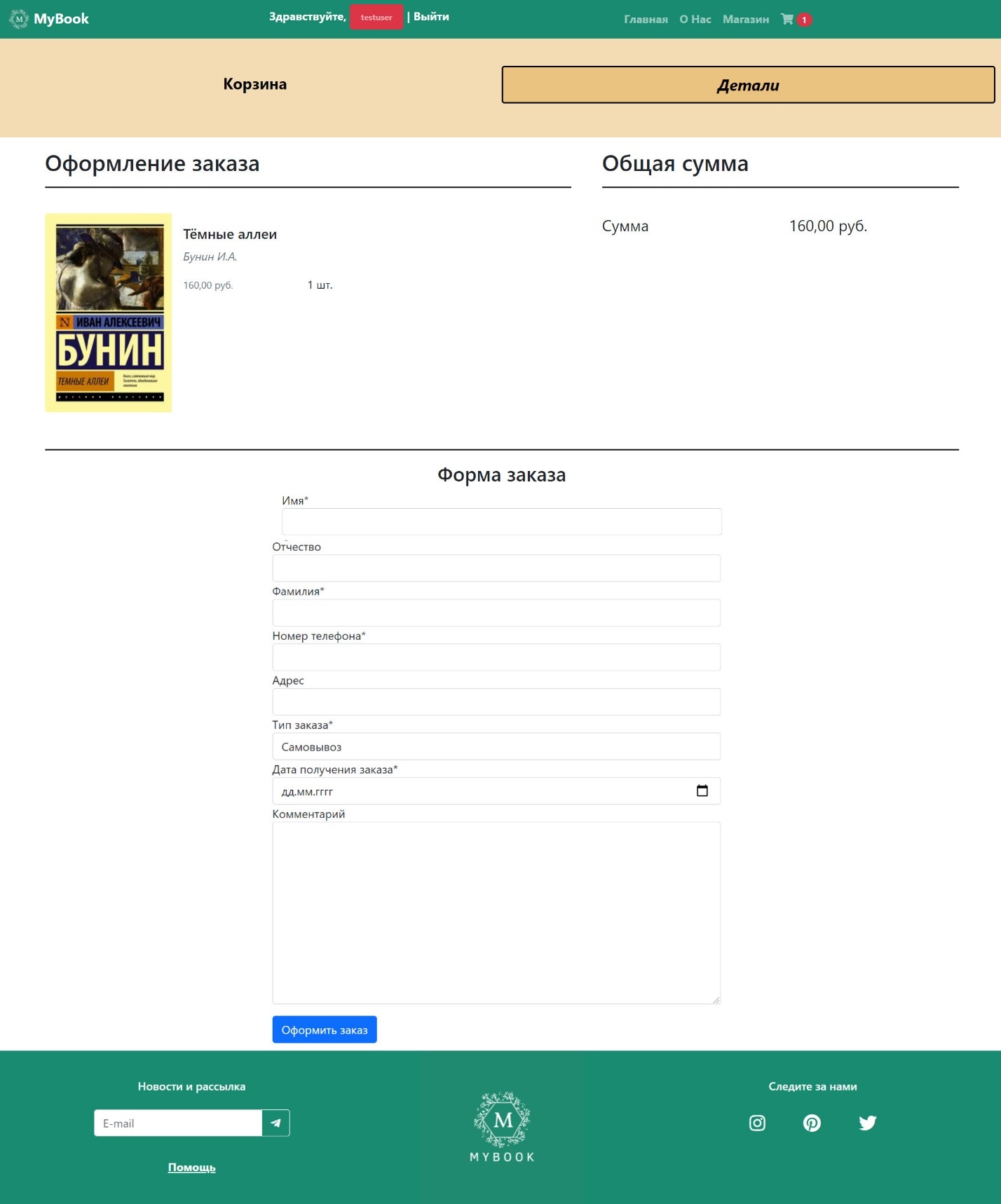


Рисунок 22 – Интерфейс просмотра деталей заказа

При добавлении товара в корзину рядом с иконкой отображается количество товаров, в корзине отображается уведомление об успешном добавлении товара, ниже список товаров (см. Рисунок 23). По умолчанию значение = 0, на странице самой корзины отображается сообщение «Ваша корзина пуста» (см. Рисунок 24).

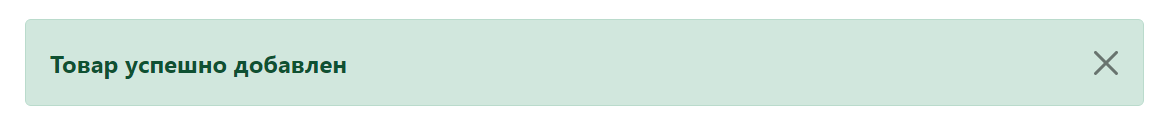


Рисунок 23 – Уведомление об успешном добавлении товара в корзину

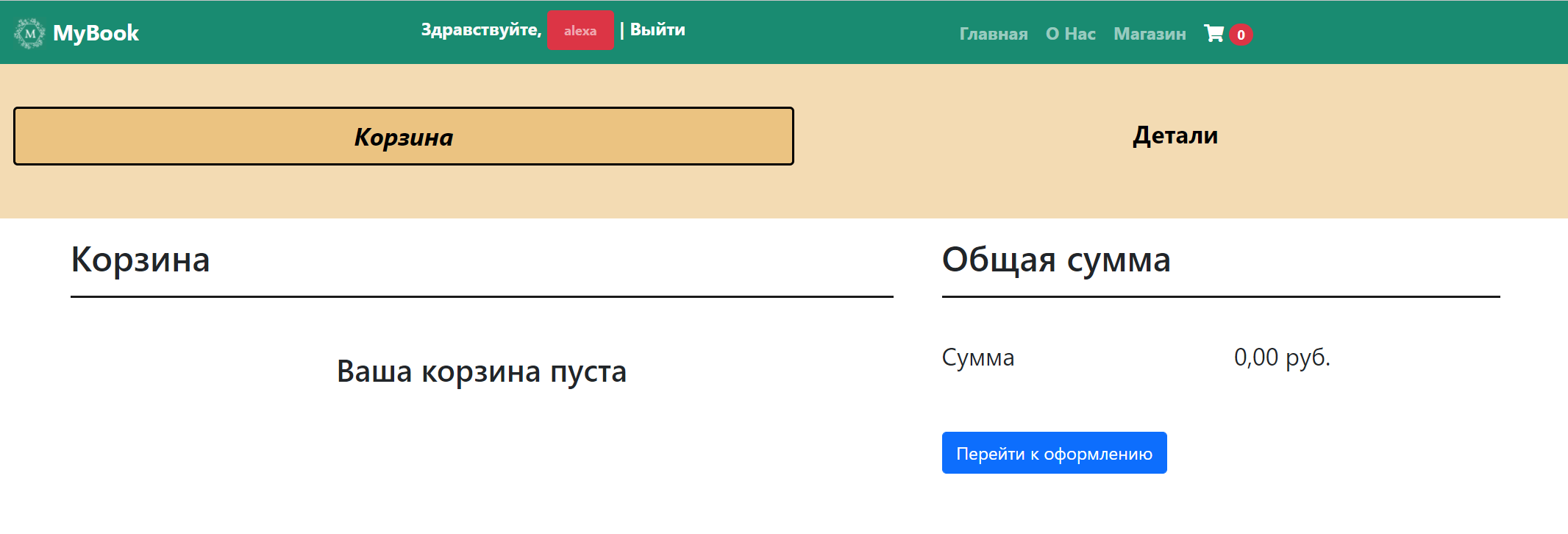


Рисунок 24 – Состояние корзины по умолчанию

При рассмотрении рисунков видно, что у всех этапов есть одинаковые блоки – шапка и подвал, которые рассмотрены в разделе Главная страница, заголовок страницы и блок «Общая сумма».

Заголовок страницы показывает на каком этапе формирования заказа находится Пользователь – активный этап выделяется рамкой.

Блок «Общая сумма» предназначен для просмотра информации об итоговой сумме заказа. Он состоит из:

* заголовка «Общая сумма»;
* итоговая сумма заказа;
* активного элемента управления – кнопки «Перейти к оформлению», при нажатии на которую происходит переход к следующему этапу формирования заказа
* Блок Корзина состоит из:
  1. заголовок – «Корзина»;
  2. элементы Корзины – краткая информация о книгах в заказе (обложка, название, автор, цена);
  3. количество товара – степпер с выбором количества товара, по умолчанию значение = 1. Изменение количества товара происходит только по нажатию кнопки «ОК». После изменения отображается уведомления об успешной операции (см. Рисунок 25);
  4. кнопка «Удалить». При нажатии происходит удаления товара из корзины. При успешной операции отображается уведомление (см. Рисунок 26).



Рисунок 25 – Уведомление об успешном изменении количества товара в корзине



Рисунок 26 – Уведомление об успешном удалении товара из корзины

1. Детали заказа:
   1. заголовок – «Детали»;
   2. поля ввода информации о покупателе (см. Приложение 2):
      1. обязательные поля «Фамилия», «Имя», «Номер телефона», «Тип заказа», «Дата получения заказа» являются обязательными для заполнения;
      2. поля «Отчество», «Адрес» и «Комментарий» являются необязательными;
   3. выбор способа доставки (обязательное поле) – возможны 2 значения:
      1. «Доставка» - при выборе данного способа заполнение поля «Адрес» является обязательным;
      2. «Самовывоз»
   4. Кнопка «Оформить заказ» - доступна только при заполнении всех обязательных полей.

После успешного формирования заказа пользователя перенаправляет на главную страницу магазина, отображается уведомление об успешном формировании заказа (см. Рисунок 27).



Рисунок 27 – Уведомление об успешном формировании заказа

Теперь информацию о заказе можно посмотреть на странице личного кабинета.

# Структура проекта и панель администрации

Поговорим немного о структуре и разработке интернет-магазина. Проект был разработан посредством среды разработки PyCharm, называется bookstore, который состоит из разделов: папки bookstore, main\_bookstore, media, static, static\_dev, база данных db.sqlite3 и файл manage.py (см. Рисунок 28).

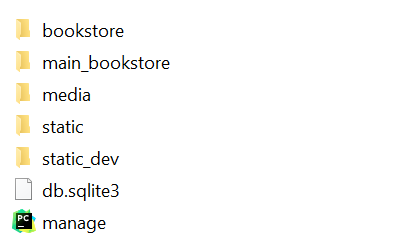


Рисунок 28 – Структура проекта

В папке bookstore хранится информация о настройках проекта, ссылка на приложение main\_bookstore и статичные и медиа файла проекта. Соответственно в папке media хранятся файлы, которые были загружены в таблицу Products в панели администрировании Django. В папке static\_dev статичные файлы, которые могут быть использованы веб-приложением без запуска сервера и файл со стилями. В папке main\_bookstore хранятся все основные файлы нашего приложения: миграции, шаблоны html-страниц, файлы форм, миксины, модели, ссылки между страницами, представления и т.д. (см. Рисунок 29).

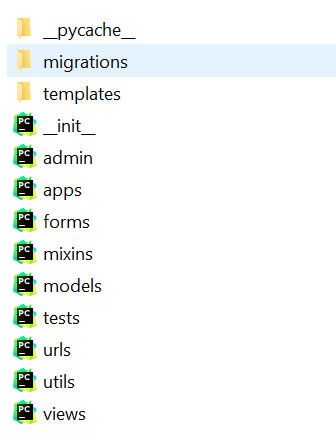


Рисунок 29 – Структура папки main\_bookstore

Информация из файлов models.py и forms.py представлена в Приложении 1 и 2, соответственно. В файле с формами описаны процессы формирования заказа, регистрации и авторизации пользователя. Далее в файле mixins.py – описан миксин (миксин – это класс, у которого есть определенный функционал, который применяется к главному классу) для отображения иконки корзины с количеством товаров в корзине конкретного пользователя на каждой странице интернет-магазина. В файле urls.py представлены все URL-адреса для нашего приложения, в который импортированы представления из файла views.py. В этом файле и размещена основная логика работы сайта. Здесь будет запрошена информация из модели, которая была создана ранее, и будет передана в шаблон. То есть это функция Python, которая принимает веб-запрос и возвращает веб-ответ, в качестве которого могут выступать html-страницы, перенаправление, ошибка 404, изображение и т.д.

Далее рассмотрим сами модели. В файле models.py описаны все модели, которые использованы в базе данных для корректной работы интернет-магазина. Модели определяют структуру хранимых данных, включая типы полей и, возможно, их максимальный размер, значения по умолчанию, параметры списка выбора, текст справки для документации, текст меток для форм и т. д. После того, как выбрана база данных, которую хотим использовать, не нужно напрямую работать с ней - просто пишем свою структуру модели и код, а Django делает всю грязную работу, связанную с базой данных за нас.

Но чтобы наглядно увидеть модели, необходимо сказать несколько слов о панели администрирования Django. Она представляет собой веб-интерфейс для создания и управления объектами базы данных. Это одна из сильных сторон Django, можно использовать прямо из коробки мощный инструмент администрирования. Рассмотрим ее более детально для нашего интернет-магазина (см. Рисунок 30).

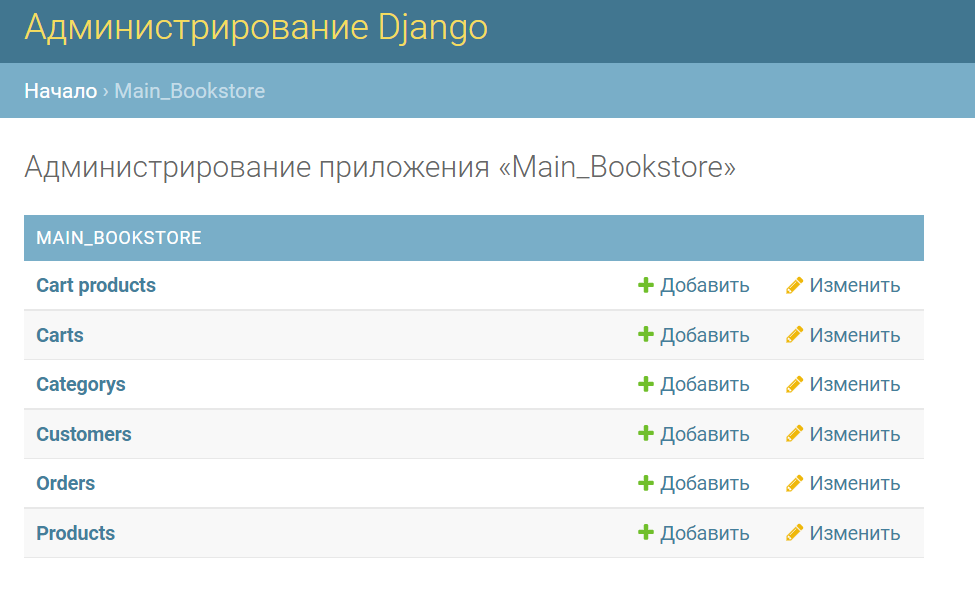


Рисунок 30 – Администрирование Django

Как видно из картинки используется 6 моделей:

* CartProduct – карточка продукта, добавленного в корзину. В ней содержится информация о покупателе, корзине, товаре, количестве товара и общей сумме.
* Cart – модель корзины. В ней содержится информация о владельце корзины, продуктах в ней, количестве товаров в корзине, общая цена, перешло ли в заказ, признак анонимного покупателя.
* Category – информация о категориях, к которым может принадлежать товар (в нашем случае это – детские, психология, графический роман, любовные романы, зарубежная и русская классика). В модели хранится как наименование категории, так и slug (это тип поля в Django для создания понятных URL на латинице).
* Customer – информация о покупателях. В модели хранится логин пользователя, номер телефона, адрес и заказы покупателя.
* Order – информация о заказе. В модели хранятся данные о покупателе, его имени, фамилии, номере телефона, идентификатор корзины, статус заказа (одно из значений: «новый заказ», «заказ в обработке», «заказ готов», «заказ выполнен»), тип заказа (самовывоз или доставка), дата получения заказа. А также необязательные поля, если они были заполнены: отчество, адрес и комментарий.
* Product – информация о товарах в магазине. Здесь хранятся данные о категории товара, названии книги, slug – чтобы просмотреть информацию о конкретной книге, изображении обложки книги, авторе, описании и цене.

Суперпользователь – администратор интернет-магазина может с помощью данного инструмента обновлять всю информацию, хранящуюся в базе данных: добавить/удалить/отредактировать книги, категории, статус заказа. Рассмотрим на примере как это выглядит (см. Рисунок 31).

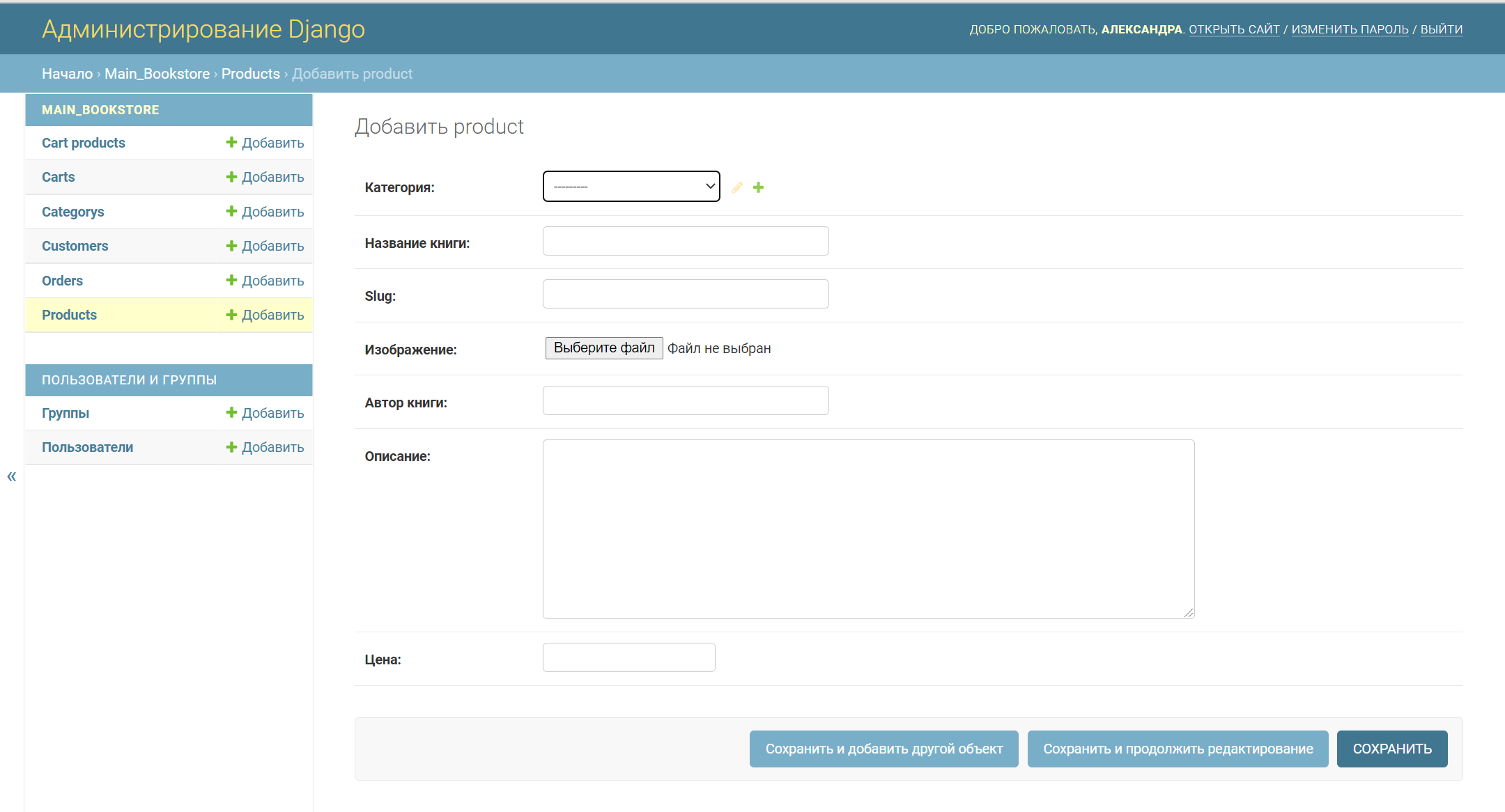


Рисунок 31 – Добавление нового продукта в интернет-магазин

Стоит также отметить, что Django предоставляет возможности аутентификации и авторизации пользователей с конфигурацией по умолчанию. Эта конфигурация покрывает требования большинства проектов, предоставляет надежный механизм работы с паролями и правами. Поэтому он и был использован посредством объекта User.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы было разработано клиент-серверное приложение книжного интернет-магазина. Описаны пользователи интернет-магазина, требования к контенту и функционалу интернет-магазина, построена информационная архитектура. Также рассмотрены необходимые средства разработки, создание базы данных, модели, шаблоны и другие возможности Django.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. <https://vec.etu.ru/moodle/course/view.php?id=925>
2. Вигерс К. Разработка требований к программному обеспечению / Пер, с англ. – М.: Издательско-торговый дом «Русская Редакция», 2004. – 576с.:
3. <https://designpub.ru/>
4. <https://webaim.org/resources/contrastchecker/>
5. <https://djbook.ru/rel3.0/>

# ПРИЛОЖЕНИЕ

Приложение 1

models.py

from PIL import Image  
  
from django.db import models  
from django.contrib.auth import get\_user\_model  
from django.utils import timezone  
  
from django.urls import reverse  
  
User = get\_user\_model()  
  
  
class MinResolutionErrorException(Exception):  
 pass  
  
  
class MaxResolutionErrorException(Exception):  
 pass  
  
  
class Category(models.Model):  
  
 name = models.CharField(max\_length=255, verbose\_name='Название категории')  
 slug = models.SlugField(unique=True)  
  
 def \_\_str\_\_(self):  
 return self.name  
  
 def get\_absolute\_url(self):  
 return reverse('category\_detail', kwargs={'slug': self.slug})  
  
 def get\_fields\_for\_filter\_in\_template(self):  
 return ProductFeatures.objects.filter(  
 category=self,  
 use\_in\_filter=True  
 ).prefetch\_related('category').value('feature\_key', 'feature\_measure', 'feature\_name', 'filter\_type')  
  
  
class Product(models.Model):  
  
 MIN\_RESOLUTION = (200, 200)  
 MAX\_RESOLUTION = (500, 500)  
 MAX\_IMAGE\_SIZE = 3145728  
  
 category = models.ForeignKey(Category, verbose\_name='Категория', on\_delete=models.CASCADE)  
 title = models.CharField(max\_length=255, verbose\_name='Название книги')  
 slug = models.SlugField(unique=True)  
 image = models.ImageField(verbose\_name='Изображение')  
 author = models.CharField(max\_length=255, verbose\_name='Автор книги')  
 description = models.TextField(verbose\_name='Описание', null=True)  
 price = models.DecimalField(max\_digits=9, decimal\_places=2, verbose\_name='Цена')  
  
 def \_\_str\_\_(self):  
 return self.title  
  
 def save(self, \*args, \*\*kwargs):  
 image = self.image  
 img = Image.open(image)  
 min\_height, min\_width = self.MIN\_RESOLUTION  
 max\_height, max\_width = self.MAX\_RESOLUTION  
 if img.height < min\_height or img.width < min\_width:  
 raise MinResolutionErrorException('Загруженное изображение меньше минимального разрешения')  
 if img.height > max\_height or img.width > max\_width:  
 raise MaxResolutionErrorException('Загруженное изображение больше максимального разрешения')  
 super().save(\*args, \*\*kwargs)  
  
 def get\_model\_name(self):  
 return self.\_\_class\_\_.\_\_name\_\_.lower()  
  
 def get\_absolute\_url(self):  
 return reverse('book\_card', kwargs={'slug': self.slug})  
  
  
class CartProduct(models.Model):  
  
 user = models.ForeignKey('Customer', verbose\_name='Пользователь', on\_delete=models.CASCADE)  
 cart = models.ForeignKey('Cart', verbose\_name='Корзина', on\_delete=models.CASCADE, related\_name='related\_products')  
 product = models.ForeignKey(Product, verbose\_name='Товар', on\_delete=models.CASCADE)  
 qty = models.PositiveIntegerField(default=1)  
 final\_price = models.DecimalField(max\_digits=9, decimal\_places=2, verbose\_name='Общая цена')  
  
 def \_\_str\_\_(self):  
 return "Продукт: {} (для корзины)".format(self.product.title)  
  
 def save(self, \*args, \*\*kwargs):  
 self.final\_price = self.qty \* self.product.price  
 super().save(\*args, \*\*kwargs)  
  
  
class Cart(models.Model):  
  
 owner = models.ForeignKey('Customer', null=True, verbose\_name='Владелец', on\_delete=models.CASCADE)  
 products = models.ManyToManyField(CartProduct, blank=True, related\_name='related\_cart')  
 total\_products = models.PositiveIntegerField(default=0)  
 final\_price = models.DecimalField(max\_digits=9, default=0, decimal\_places=2, verbose\_name='Общая цена')  
 in\_order = models.BooleanField(default=False)  
 for\_anonymous\_user = models.BooleanField(default=False)  
  
 def \_\_str\_\_(self):  
 return str(self.id)  
  
  
class Customer(models.Model):  
  
 user = models.ForeignKey(User, verbose\_name='Пользователь', on\_delete=models.CASCADE)  
 phone = models.CharField(max\_length=20,verbose\_name='Номер телефона', null=True, blank=True)  
 address = models.CharField(max\_length=255, verbose\_name='Адрес', null=True, blank=True)  
 orders = models.ManyToManyField('Order', related\_name='related\_customer', verbose\_name='Заказы покупателя')  
  
 def \_\_str\_\_(self):  
 return "Покупатель: {} {}".format(self.user.first\_name, self.user.last\_name)  
  
  
class Order(models.Model):  
  
 STATUS\_NEW = 'new'  
 STATUS\_IN\_PROGRESS = 'in\_progress'  
 STATUS\_READY = 'is\_ready'  
 STATUS\_COMPLETED = 'completed'  
  
 BUYING\_TYPE\_SELF = 'self'  
 BUYING\_TYPE\_DELIVERY = 'delivery'  
  
 STATUS\_CHOICES = (  
 (STATUS\_NEW, 'Новый заказ'),  
 (STATUS\_IN\_PROGRESS, 'Заказ в обработке'),  
 (STATUS\_READY, 'Заказ готов'),  
 (STATUS\_COMPLETED, 'Заказ выполнен')  
 )  
  
 BUYING\_TYPE\_CHOICES = (  
 (BUYING\_TYPE\_SELF, 'Самовывоз'),  
 (BUYING\_TYPE\_DELIVERY, 'Доставка')  
 )  
  
 customer = models.ForeignKey(Customer, verbose\_name='Покупатель', related\_name='related\_orders', on\_delete=models.CASCADE)  
 first\_name = models.CharField(max\_length=255, verbose\_name='Имя')  
 last\_name = models.CharField(max\_length=255, verbose\_name='Фамилия')  
 middle\_name = models.CharField(max\_length=255, verbose\_name='Отчество', null=True, blank=True)  
 phone = models.CharField(max\_length=20, verbose\_name='Номер телефона')  
 cart = models.ForeignKey(Cart, verbose\_name='Корзина', on\_delete=models.CASCADE, null=True, blank=True)  
 address = models.CharField(max\_length=1024, verbose\_name='Адрес', null=True, blank=True)  
 status = models.CharField(  
 max\_length=100,  
 verbose\_name='Статус заказа',  
 choices=STATUS\_CHOICES,  
 default=STATUS\_NEW  
 )  
 buying\_type = models.CharField(  
 max\_length=100,  
 verbose\_name='Тип заказа',  
 choices=BUYING\_TYPE\_CHOICES,  
 default=BUYING\_TYPE\_SELF  
 )  
 comment = models.TextField(verbose\_name='Комментарий', null=True, blank=True)  
 created\_at = models.DateTimeField(auto\_now=True, verbose\_name='Дата создания заказа')  
 order\_date = models.DateField(verbose\_name='Дата получения заказа', default=timezone.now)  
  
 def \_\_str\_\_(self):  
 return str(self.id)

Приложение 2

forms.py

from django import forms  
from django.contrib.auth.models import User  
  
from .models import Order, FeatureValidator, CategoryFeature, Category  
  
  
class OrderForm(forms.ModelForm):  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 super().\_\_init\_\_(\*args, \*\*kwargs)  
 self.fields['order\_date'].label = 'Дата получения заказа'  
  
 order\_date = forms.DateField(widget=forms.TextInput(attrs={'type': 'date'}))  
  
 class Meta:  
 model = Order  
 fields = (  
 'first\_name', 'middle\_name', 'last\_name', 'phone', 'address', 'buying\_type', 'order\_date', 'comment'  
 )  
  
  
class LoginForm(forms.ModelForm):  
  
 password = forms.CharField(widget=forms.PasswordInput)  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 super().\_\_init\_\_(\*args, \*\*kwargs)  
 self.fields['username'].label = 'Логин'  
 self.fields['password'].label = 'Пароль'  
  
 def clean(self):  
 username = self.cleaned\_data['username']  
 password = self.cleaned\_data['password']  
 if not User.objects.filter(username=username).exists():  
 raise forms.ValidationError(f'Пользователь с логином {username} не найден в системе')  
 user = User.objects.filter(username=username).first()  
 if user:  
 if not user.check\_password(password):  
 raise forms.ValidationError("Неверный пароль")  
 return self.cleaned\_data  
  
 class Meta:  
 model = User  
 fields = ['username', 'password']  
  
  
class RegistrationForm(forms.ModelForm):  
  
 confirm\_password = forms.CharField(widget=forms.PasswordInput)  
 password = forms.CharField(widget=forms.PasswordInput)  
 phone = forms.CharField(required=False)  
 address = forms.CharField(required=False)  
 email = forms.CharField(required=True)  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 super().\_\_init\_\_(\*args, \*\*kwargs)  
 self.fields['username'].label = 'Логин'  
 self.fields['password'].label = 'Пароль'  
 self.fields['confirm\_password'].label = 'Подтвердите пароль'  
 self.fields['phone'].label = 'Номер телефона'  
 self.fields['first\_name'].label = 'Имя'  
 self.fields['last\_name'].label = 'Фамилия'  
 self.fields['address'].label = 'Адрес'  
 self.fields['email'].label = 'E-mail'  
  
 def clean\_email(self):  
 email = self.cleaned\_data['email']  
 if User.objects.filter(email=email).exists():  
 raise forms.ValidationError(f'Данный E-mail {email} уже зарегистрирован в Системе')  
 return email  
  
 def clean\_username(self):  
 username = self.cleaned\_data['username']  
 if User.objects.filter(username=username).exists():  
 raise forms.ValidationError(f'Имя {username} занято')  
 return username  
  
 def clean(self):  
 password = self.cleaned\_data['password']  
 confirm\_password = self.cleaned\_data['confirm\_password']  
 if password != confirm\_password:  
 raise forms.ValidationError(f'Пароли не совпадают')  
 return self.cleaned\_data  
  
 class Meta:  
 model = User  
 fields = ['username', 'password', 'confirm\_password', 'last\_name', 'first\_name', 'email', 'phone', 'address'] #в каком порядке здесь, в таком и на форме