Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Сибирский федеральный университет»

**ИТОГОВАЯ АТТЕСТАЦИОННАЯ РАБОТА**   
по программе профессиональной переподготовки   
«Разработка программного обеспечения на языке Python»

на тему: Разработка web-приложения с использованием Фреймворка django

|  |  |
| --- | --- |
| Слушатель: Зубова София Павловна, Менченков Константин Евгеньевич |  |
| Группа: 16 |  |
| Руководитель итоговой аттестационной работы: Кузьмин Д.Н. |  |

Красноярск

2023

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc147659298)

[1 Обзор технологий создания web-приложений 4](#_Toc147659299)

[1.1](#_Toc147659300)  [Обзор задачи создания web-приложений 4](#_Toc147659300)

[1.2 Обзор инструментов для создания приложений 4](#_Toc147659301)

[2 Проектирование приложения 6](#_Toc147659302)

[2.1 Описание желаемых функций приложения](#_Toc147659303) интернет-магазина [6](#_Toc147659303)

[2.2 Требования к программе или программному изделию 7](#_Toc147659304)

[2.3 Стадии и этапы разработки 8](#_Toc147659305)

[2.4 Описание архитектуры приложения 9](#_Toc147659306)

[3 Реализация приложения 12](#_Toc147659307)

[3.1 Написание программного кода 12](#_Toc147659308)

[3.2 Тестирование приложения 21](#_Toc147659309)

[Заключение 33](#_Toc147659310)

[Список использованных источников 34](#_Toc147659311)

**ВВЕДЕНИЕ**

Разработка веб-сайтов на Python остается актуальной и популярной темой в IT-индустрии. Python является одним из самых гибких и мощных языков программирования, который может использоваться для создания различных типов веб-приложений, включая сайты электронной коммерции, социальные сети, блоги и многое другое.

Python имеет большое количество библиотек и фреймворков, которые облегчают процесс разработки веб-сайтов. Например, Django и Flask являются двумя из самых популярных фреймворков для веб-разработки на Python. Они предоставляют разработчикам множество инструментов и функций для быстрой и эффективной разработки веб-приложений.

Кроме того, Python также имеет большое сообщество разработчиков, которые постоянно работают над улучшением языка и созданием новых инструментов для веб-разработки. Это гарантирует, что Python останется актуальным языком программирования для веб-разработки в ближайшее время

Цель итоговой работы: разработать web-сайт для продажи товаров на Фреймворке django.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- провести анализ предметной области

- выбрать инструмент для создания приложения

- создать макет сайта

- разработать структуры базы данных

- разработать основные модули

- осуществить тестирование

**1 Обзор технологий создания web-приложений**

* 1. **Обзор задачи создания web-приложений**

Задача создания web-приложений заключается в разработке программного обеспечения, которое может быть запущено в браузере и использоваться для выполнения различных задач, таких как обработка данных, взаимодействие с базами данных, управление контентом и многое другое.

История развития технологий веб-разработки началась в 1990-х годах с появления первых статических веб-сайтов, которые состояли из HTML-страниц и не имели динамического содержимого. Затем появились CGI-скрипты, которые позволили добавлять некоторую динамическую функциональность на сайтах.[1]

В 2000-х годах были разработаны первые фреймворки для веб-разработки, такие как Ruby on Rails и Django. Они предоставляли разработчикам инструменты для быстрой разработки динамических веб-сайтов с использованием MVC-архитектуры.

Сегодня технологии веб-разработки продолжают развиваться, и новые инструменты и фреймворки появляются каждый год. Python остается одним из самых популярных языков программирования для веб-разработки благодаря своей гибкости, мощности и обширной экосистеме библиотек и фреймворков [2].

**1.2 Обзор инструментов для создания приложений**

Существует множество инструментов для создания web-приложений, от фреймворков и библиотек до платформ для разработки веб-приложений. Вот некоторые из наиболее популярных инструментов для веб-разработки:

1. Django - фреймворк для веб-разработки на языке Python, который предоставляет мощные инструменты для создания динамических веб-сайтов и веб-приложений.

2. Flask - еще один популярный фреймворк для веб-разработки на языке Python, который предоставляет минимальный набор инструментов для создания веб-приложений.

3. React - библиотека JavaScript для создания пользовательских интерфейсов, которая используется для создания динамических веб-приложений.[3]

4. Angular - еще один популярный фреймворк JavaScript для создания динамических веб-приложений.

5. Node.js - платформа для разработки серверных приложений на языке JavaScript, которая используется для создания высокопроизводительных веб-приложений.

6. Ruby on Rails - фреймворк для веб-разработки на языке Ruby, который предоставляет инструменты для быстрой разработки веб-приложений.

7. ASP.NET - фреймворк для веб-разработки на языке C#, который используется для создания масштабируемых и безопасных веб-приложений.[4]

8. Vue.js - библиотека JavaScript для создания пользовательских интерфейсов, которая используется для создания динамических веб-приложений.

Это только некоторые из инструментов, которые могут быть использованы для создания web-приложений. Фреймворк django является наиболее удобным для разработки веб-приложения, потому что не требует много времени на настройку конфигурации, позволяет программисту быстро создать тестовую версию продукта с базовым набором функций. Фреймворк из коробки содержит панель администратора, так же в Django реализовано объектно-реляционное отображение (ORM), которое обеспечивает взаимодействие приложения с базами данных (БД). ORM автоматически передаёт данные из БД, например, SQLite или MySQL, в объекты, которые используются в коде приложения.

**2. Проектирование приложения**

**2.1 Описание желаемых функций приложения интенет-магазина**

Разрабатываемое веб-приложение это интернет-магазин кроссовок. Для интернет-магазина необходимо сформулировать функциональные и нефункциональные требования.

Функциональное требование – это описание поведения системы в определенных условиях. Другими словами, это описание функций системы.[5]

Ниже перечислены требования к функциональным характеристикам для разработки веб-приложения для потенциальных покупателей и работы сотрудников магазина.

Для сотрудников магазина программа предоставляет следующие возможности:

- просмотр и дополнение списков товаров, которыми торгует магазин;

- просмотр и заполнение таблицы наличия товаров на складе (делается через панель администратора).

Для покупателей магазина программа предоставляет следующие возможности:

- просмотр меню товаров и карточки товара;

-просмотр страниц с миссией магазина, его ценностями;

-добавление товаров в корзину, оформление заказа;

-личный кабинет администратора, просмотр и добавление новых товаров, цветов, размеров.

При просмотре товаров выводится страница, на которой представлены фотокарточка, название, цена со скидкой (если такая имеется), цена без скидки, доступные цвета, размеры, кнопку добавления в корзину.

При просмотре страницы добавления товара выводятся формы для загрузки фотокарточки, добавления цены со скидкой и без нее, категории, доступных размеров, цветов,

Также для пользователей определенного типа должна быть доступна панель администратора, в которой можно редактировать записи всех таблиц базы данных приложения.

**2.2 Требования к программе или программному изделию**

Нефункциональное требование – это описание присущих свойств или характеристик, которые система должна демонстрировать, или ограничения, которые она должна соблюдать, требования к качеству и так далее.

При формулировании требований к качеству необходимо рассмотреть следующие:

- атрибуты внешнего качества: производительность, надёжность, доступность, масштабируемость, безопасность;

- атрибуты качества в использовании: результативность, эффективность, скорость обучения, точность, утомляемость;

- ограничения: совместимость с оборудованием, системным ПО, ограничения на языки и средства разработки, допущения о квалификации пользователей, состав поставки, лицензионные ограничения.

Для разрабатываемого веб-приложения можно задать следующие требования: операционная система Windows 8.1 и выше; оперативная память 512 Мб; свободное место в памяти 500 Мб. Корректное отображение (в соответствии с макетом) в браузерах: Google Chrome, Opera, ЯндексБраузер, Mozilla Firefox, Safari последних версий на момент приемки работ.

Обеспечение надежного функционирования веб-приложения интернет-магазина одежды является критически важной задачей, чтобы обеспечить удовлетворение клиентов, сохранение данных и бизнес-процессов. Ниже перечислены основные требования к обеспечению надежности такого приложения:

1.Высокая доступность и отказоустойчивость:

2. Регулярное резервное копирование данных:

3. Мониторинг и уведомления. (Непрерывный мониторинг состояния серверов и приложения; Автоматическое уведомление администраторов о проблемах или отказах.)

4. Защита от атак. (Применение мер безопасности, таких как Web Application Firewall (WAF), для предотвращения атак на приложение, такие как SQL-инъекции и кросс-сайтовый скриптинг (XSS).)[6]

5. Скорость и производительность. (Оптимизация кода и запросов к базе данных для обеспечения быстрого отклика приложения.)

6. Масштабируемость. (Возможность быстро масштабировать ресурсы при необходимости, чтобы справляться с увеличением нагрузки, например, во время праздничных распродаж или акций.)

7. Защита данных клиентов. (Соблюдение всех законов и стандартов в области защиты данных, таких как GDPR; Шифрование данных в покое и в передаче.)[7]

Данные о продукции магазина, наличии ее на складе, и пользователях хранятся в базе данных sqlite3. СУБД обеспечивает разграничение прав доступа к данным — дает клиенту права на чтение и запись данных некоторых таблиц (товар, размер скидки, цвет, размер), оператору — на чтение и запись всех таблиц (включая недоступные клиенту без прав администратора). Ввод данных в базу (логины и пароли) выполняет администратор, выполняющий поставку программного обеспечения заказчику. Выполняет он это через панель администратора.

Для разрабатываемого продукта можно выделить следующие требования: Исходный код должен быть написан на языке python c использованием фреймворка django. IDE для программирования pyCharm Community. Страницы приложения созданы с применением HTML, для описания стилей использовать CSS, для анимации используется JS. Используется база данных sqlite3.

**2.3 Стадии и этапы разработки**

Создание web сайта состоит из следующих работ:

1. Планирование: определение целей, требований и функциональности проекта, а также выбор технологий и инструментов для его реализации.

2. Проектирование: создание дизайна интерфейса, структуры базы данных, архитектуры приложения и определение логики работы приложения.

3. Разработка: создание кода приложения, его тестирование и отладка, а также интеграция с другими системами и сервисами.

4. Тестирование: проверка работоспособности приложения, выявление и исправление ошибок, проверка соответствия требованиям и ожиданиям пользователей.[9]

5. Релиз: запуск приложения на сервере и его доступность для пользователей.

6. Сопровождение: поддержка и обновление приложения, исправление ошибок и добавление новой функциональности.

**2.4 Описание архитектуры приложения**

В работе используется фреймворк Django - высокоуровневый фреймворк для веб-разработки, написанный на языке Python. Он предоставляет разработчикам множество инструментов и функций для быстрой и эффективной разработки веб-приложений.[10]

Основные преимущества Django:

1. Архитектура MTV (Model-Template-View) позволяет разделять логику приложения и представление данных.

2. Встроенная административная панель, которая позволяет управлять данными приложения без написания дополнительного кода.

3. ORM (Object-Relational Mapping) Django позволяет работать с базами данных без написания SQL-запросов.

4. Встроенная система аутентификации и авторизации пользователей.

5. Широкий выбор сторонних модулей и библиотек, которые значительно упрощают и ускоряют процесс разработки.

6. Высокая скорость работы и масштабируемость приложений на базе Django.

Django используется многими крупными компаниями, такими как Instagram, Pinterest, Mozilla и другими.

В соответствии с тем, что используется фреймворк django, проект имеет архитектуру MTV, схематично описываемую следующим образом:

Так, общая структура проекта со всеми описанными компонентами может выглядеть следующим образом.

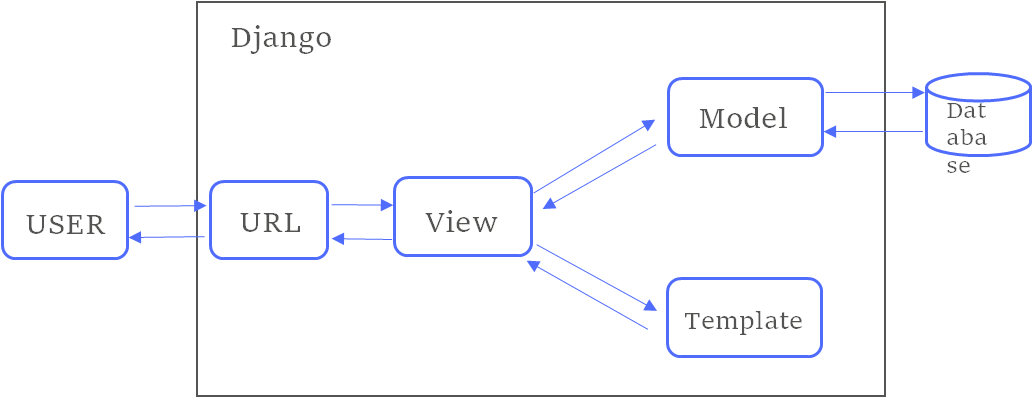


Рисунок 1. Архитектура проекта django

Таким образом, структура Django включает следующие компоненты:

- URL-маршрутизаторы, перенаправляющие HTTP-запрос от браузера или любого иного веб-клиента в представления;

- Представление View: получает запрос, обрабатывает его и отправляет пользователю некоторый ответ. Для создания ответа может применять шаблоны Template. Представление описывает основную логику и алгоритмы приложения, обращение к модели и базе данных. В архитектуре MVC этому компоненту соответствуют контроллеры.

- Model: описывает схему данных. Отдельные классы соответствуют таблицам в базе данных.

- Template: Это часть приложения, которую пользователь видит в браузере. Часто это HTML-шаблоны, которые используются представлением для демонстрации пользователю полученных от модели данных. В MVC этому компоненту соответствует View, то есть представления.

Примерная схема работы приложения на Django. Когда к приложению приходит запрос, то URL определяет, с каким ресурсом сопоставляется данный запрос. Ресурс – функция представления View – получает запрос и определенным образом обрабатывает его. При этом может обращаться к моделям и базе данных, получать из нее данные, или, наоборот, сохранять их. Результат обработки запроса отправляется обратно, и этот результат пользователь видит в своем браузере. Как правило, результат обработки запроса представляет сгенерированный html-код на основе шаблонов (Template).

**3 Реализация приложения**

**3.1 Написание программного кода**

Веб-приложение или проект Django состоит из отдельных приложений. Каждое приложение представляет какую-то определенную функциональность или группу функциональностей. Один проект может включать множество приложений. Это позволяет выделить группу задач в отдельный модуль и разрабатывать их относительно независимо от других.

Данное приложение состоит из приложений:

1.cart- приложение, содержащее логику работы с корзиной;

2. shop- функционал работы с товарами, карточками товаров, размерами, цветами;

3.djangoProject- содержит общие блоки: главная страница, контакты, страница о нас.

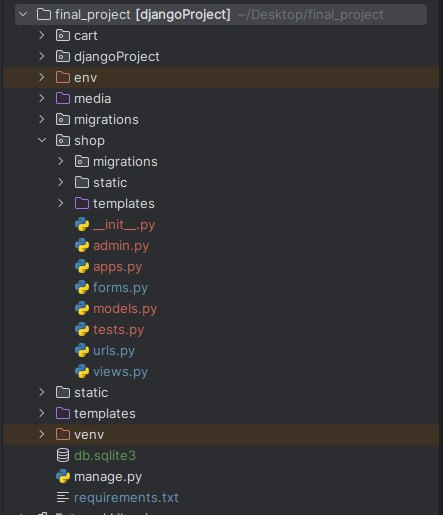


Рисунок 1 - Структура файлов проекта и приложения django

Рассмотрим его файлы:

- папка **migrations**: предназначена для хранения миграций - скриптов, которые позволяют синхронизировать структуру базы данных с определением моделей;

**- \_\_init\_\_.py**: указывает интерпретатору python, что текущий каталог будет рассматриваться в качестве пакета;

- **admin.py**: предназначен для административных функций, в частности, здесь производится регистрация моделей, которые используются в интерфейсе администратора;

- **apps.py**: определяет конфигурацию приложения;

- **models.py**: хранит определение моделей, которые описывают используемые в приложении данные;

- **tests.py**: хранит тесты приложения;

-**views.py**: определяет функции, которые получают запросы пользователей, обрабатывают их и возвращают ответ;

- **urls.py**: содержит URL-адреса;

- папка **static:** содержит статические файлы, то есть файлы css и изображения;

- папка **templates**: html файлы. которые отображаются пользователю в браузере на странице;

- файл **requirements.txt**: необходимые библиотеки для запуска сайта.

В качестве примера рассмотрим приложение магазин (shop). В соответствии с заданием требуется создать следующие функции (каждая функция будет соответствовать обработке запроса – перехода пользователя на страницу):

1. страница со всеми товарами;
2. страница товаров, отсортированных по категориям (весна, лето, зима, осень, новинки);
3. карточка товара;
4. страница добавления новой позиции и редактирования (доступна администраторам).

Содержимое функций, согласно заданию данного этапа, следующее:

- все страницы возвращают html файл, содержащий заголовок - название компании и навигация сайта – список маршрутов: ссылки для перехода на все остальные его страницы. Footer с навигацией, контактами, пользовательскими соглашениями.

- страница со всеми товарами – должна вернуть html файл, дополнительно содержащий навигацию по категориям товаров, список всех товаров. Аналогичное наполнение страницы с товарами по категориям.

- карточка товара- фотокарточка товара, цена со скидкой (если скидка имеется), цена без скидки, размеры, которые есть в наличии, цвета, кнопка добавления в корзину.

- добавление нового товара- форма, принимающая: название, категорию (доступен выбор из списка, но можно создать новую категорию), цены со скидкой и без, размеры, цвета, фото товара.

- редактирование товара- доступно изменение указанной в предыдущем пункте характеристик товара.

- успешного добавления/редактирования товара: кнопки перехода к добавлению нового товара и перехода на главную страницу.

В файле views.py функции выглядят следующим образом:

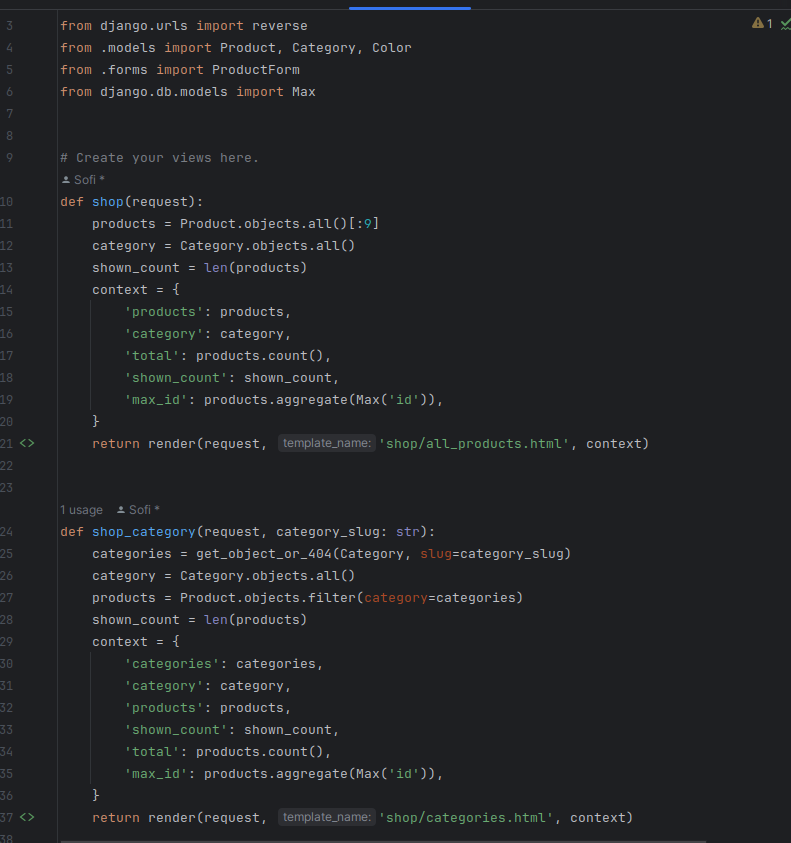


Рисунок 2 - Функции приложения

Так как данные о товарах содержатся в базе данных, каждая функция обращается к объектам в базе данных, а затем передает данные вместе с html-шаблонами.

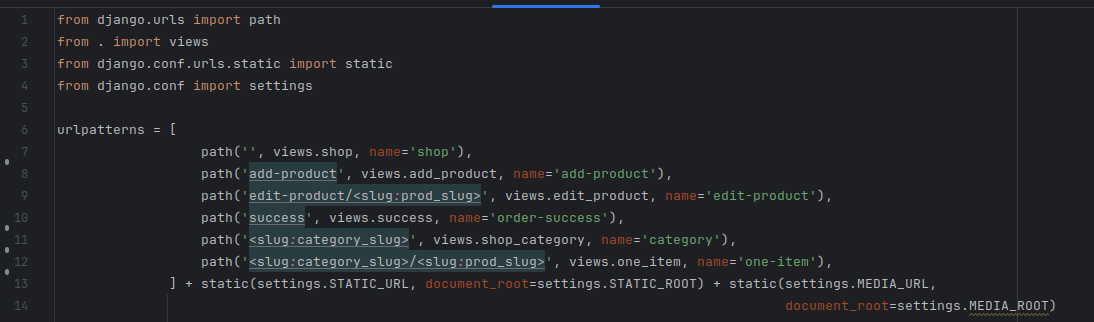


Рисунок 3 - Маршруты приложения

В файле urls.py собраны маршруты для данного приложения. Переменная urlpatterns определяет набор сопоставлений функций обработки с определенными строками запроса. Например, запрос к корню каталога магазина будет обрабатываться функцией shop.

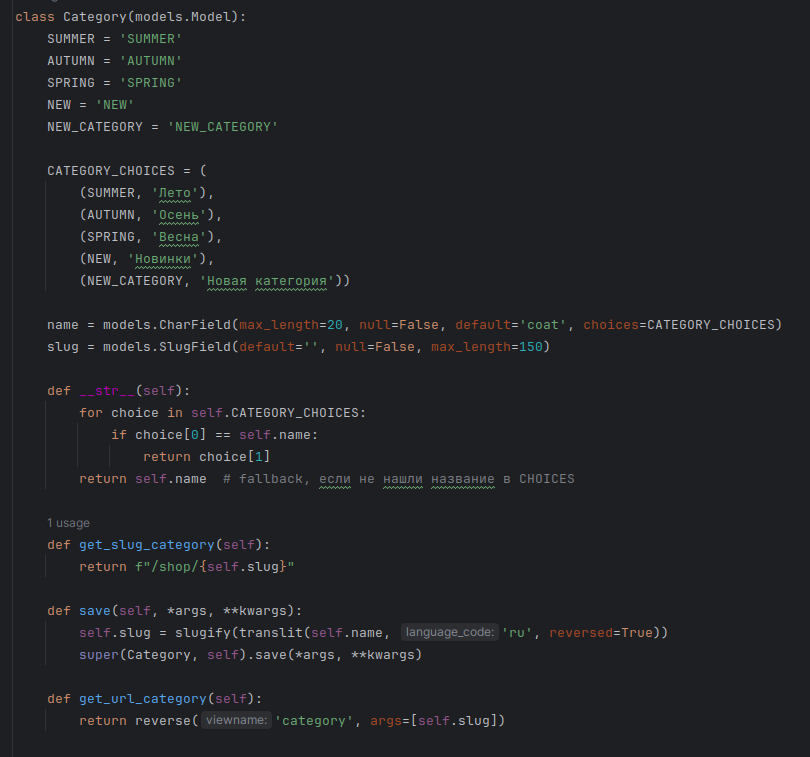


Рисунок 4 - Модель категорий приложения

На рисунке 4 представлен пример модели, содержащей категории. Здесь используется slug то есть короткая метка, которая содержит только буквы, цифры, подчеркивания или дефисы для последующего использования в URL. Так же представлены функции, в каком виде будет выводиться URL-адрес каждой категории, функция, сохраняющая новые категории.

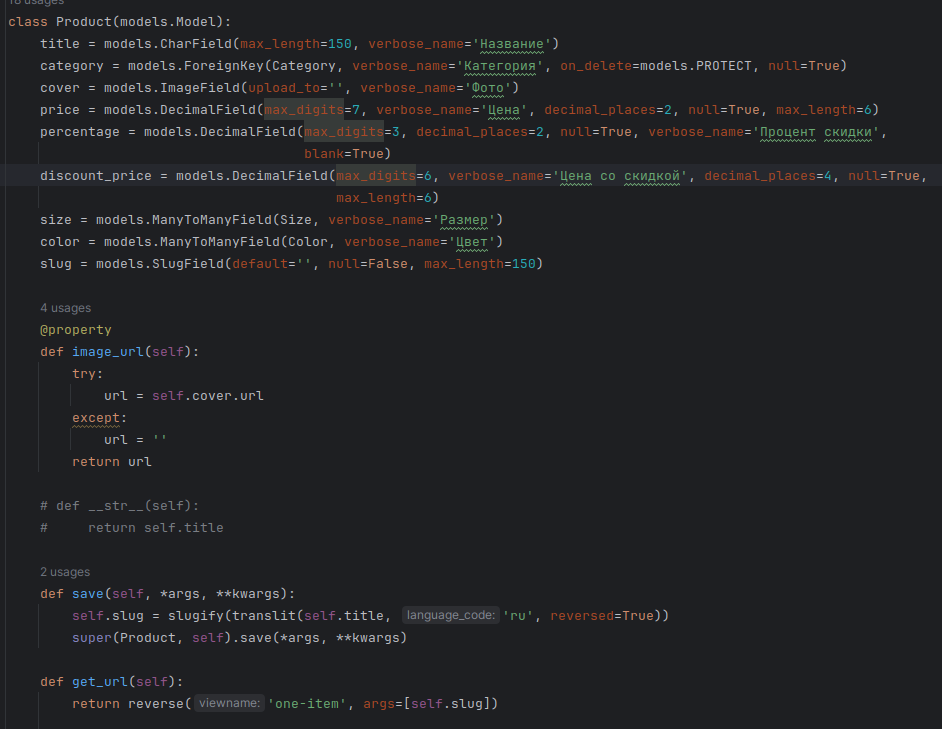


Рисунок 5 - Модель продукта приложения

В данной модели описывается, какие значения будет хранить таблица в базе данных для каждого продукта, в частности это название, фото, цену со скидкой и без, слаг продукта, размер и цвет связаны с другими таблицами с соответствующими названиями связью ManeToManyFields, так как у разных продуктов могут быть разные цвета и размеры.

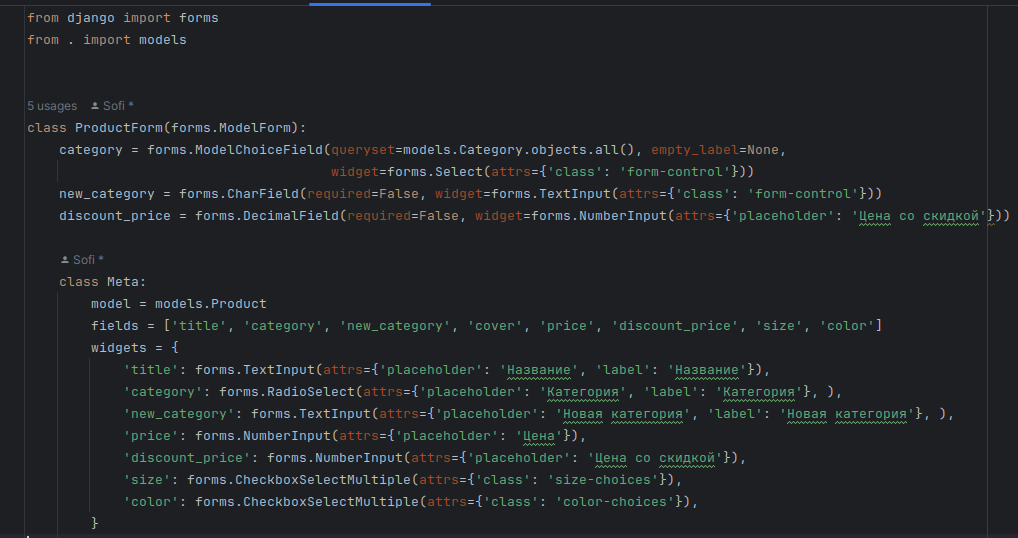


Рисунок 6 - Формы

Требуется добавить следующий функционал для заполнения таблиц базы данных и получения информации: форму для ввода данных в таблицы, функцию, позволяющую сохранить данные с формы. Затем требуется добавить на каждую страницу сайта отображения данных из соответствующих таблиц. Этот функционал доступен администраторам на страницах добавления и редактирования товаров.

Далее представлен пример html-файла.

Все файлы html в данном проекте расширяют файл main.html, в нем описаны header и footer сайта, подключены шрифты, стили.

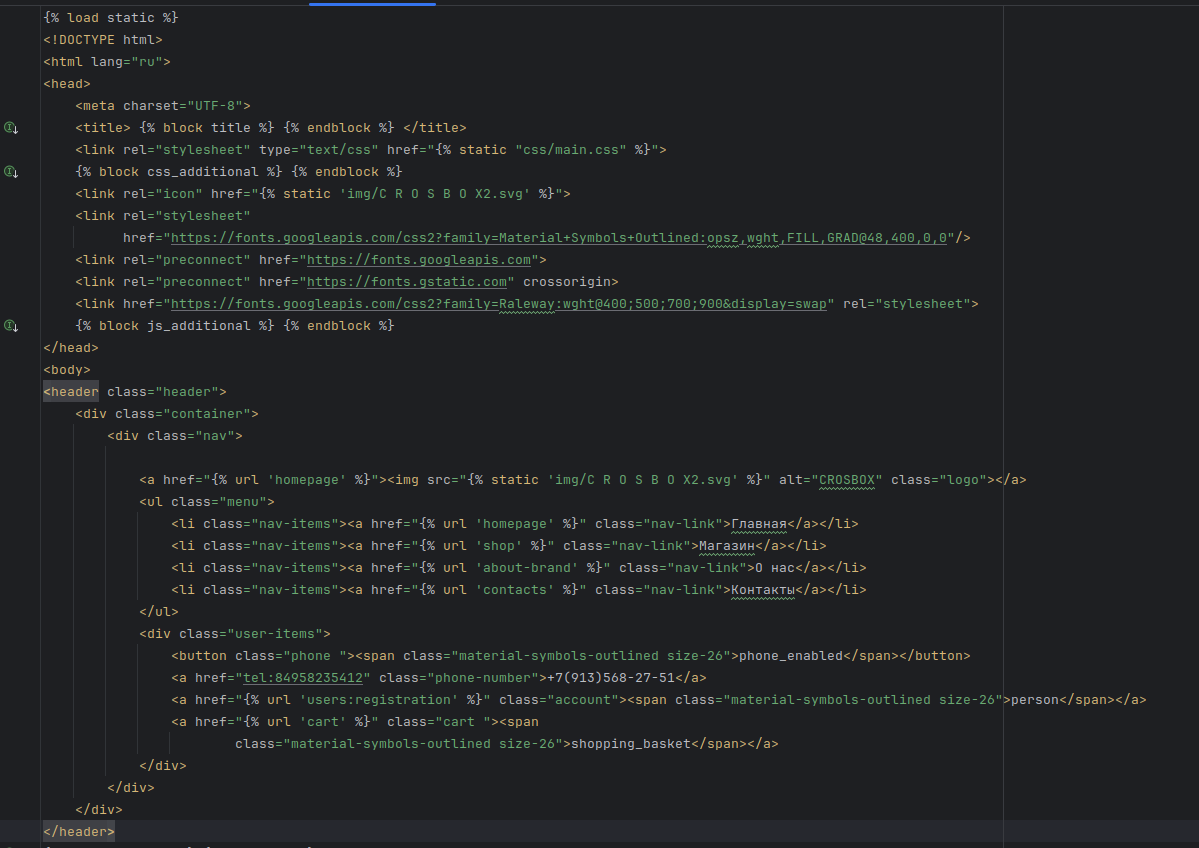


Рисунок 7 - Файл html-страницы

Поэтому в остальных файлах описывается остальной функционал и вверху написана строчка кода: {% extends 'main.html' %}.

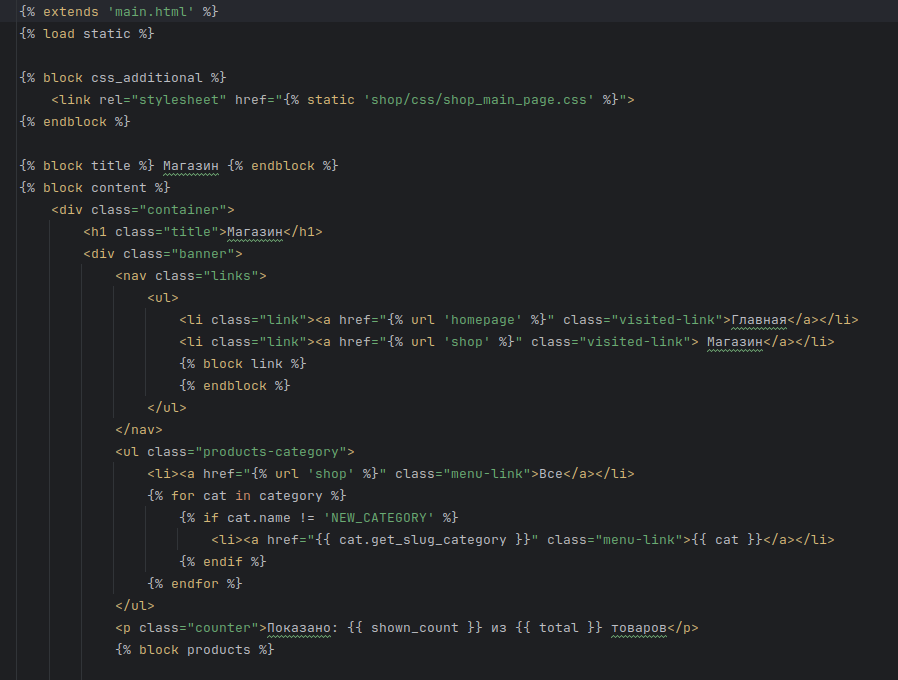


Рисунок 8 - Файл html-страницы

В данном примере указана навигация, отображение категорий, сообщение, сколько показано товаров.

Этот файл расширяется другими файлами, например, файл categories.html, в котором выводятся все товары, соответствующие определенной категории (весна, лето, зима, осень, новинки).

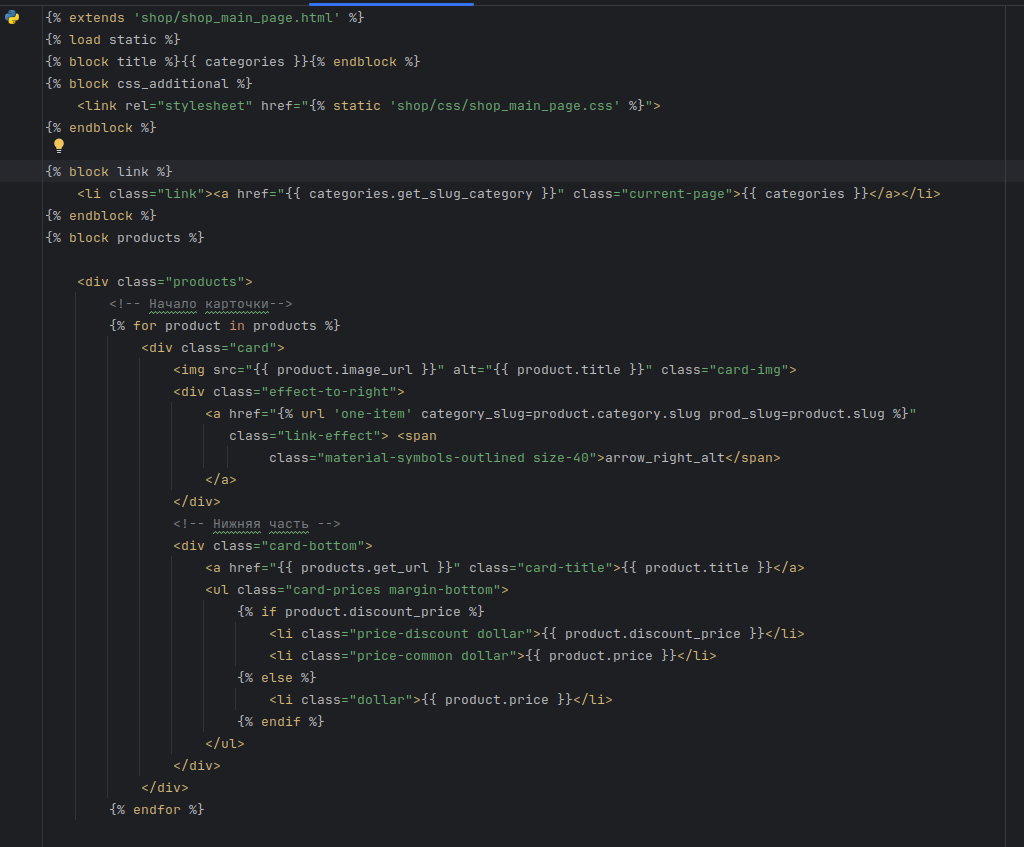


Рисунок 9 - Файл html-страницы

Зайдем в личный кабинет администратора с созданным ранее логином и паролем.

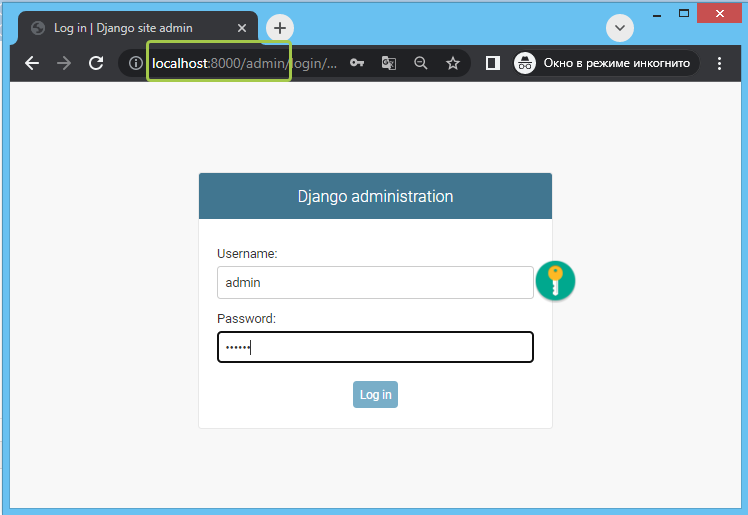


Рисунок 10 - Пример входа в личный кабинет администратора

Через интерфейс администратора добавим записи в таблицу продуктов.

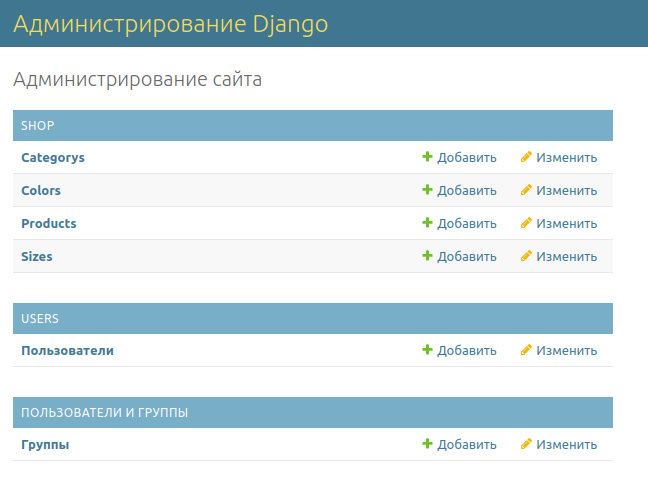


Рисунок 11 - Список доступных таблиц

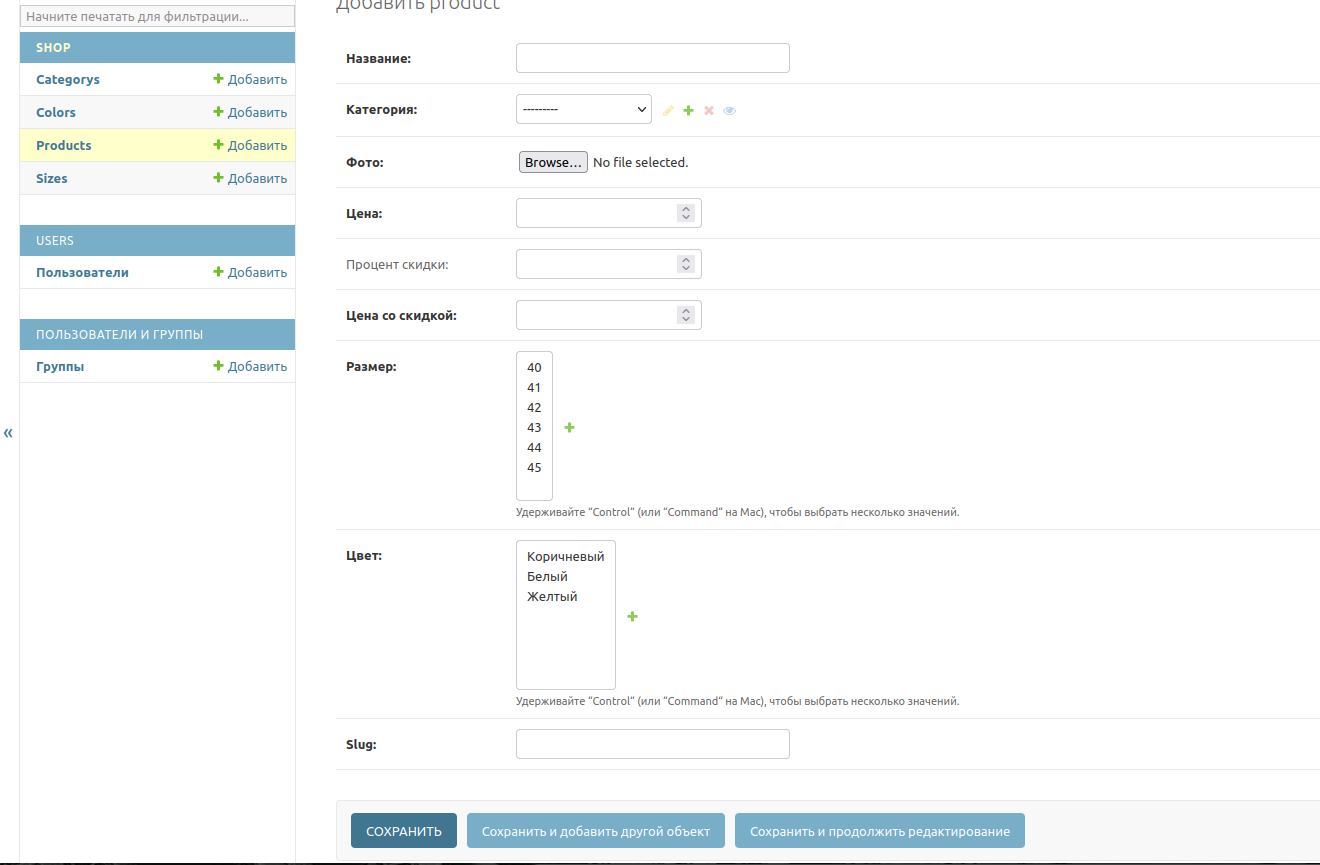


Рисунок 12 - Пример добавления записи в таблицу

Так же для удобства работы с сайтом есть страница для добавления и редактирования товаров.

Таким образом, в ходе работы написан программный код, реализующий веб-приложение, функционал которого выполняет все поставленные задачи.

**3.2 Тестирование приложения**

В результате создано веб-приложение в соответствии с заданием.

Основная информация хранятся в базе данных. Таблицы базы данных можно увидеть в СУБД или в личном кабинете администратора, где он может редактировать записи. Далее на рисунке приведен список таблиц базы данных, их отображение для администратора.

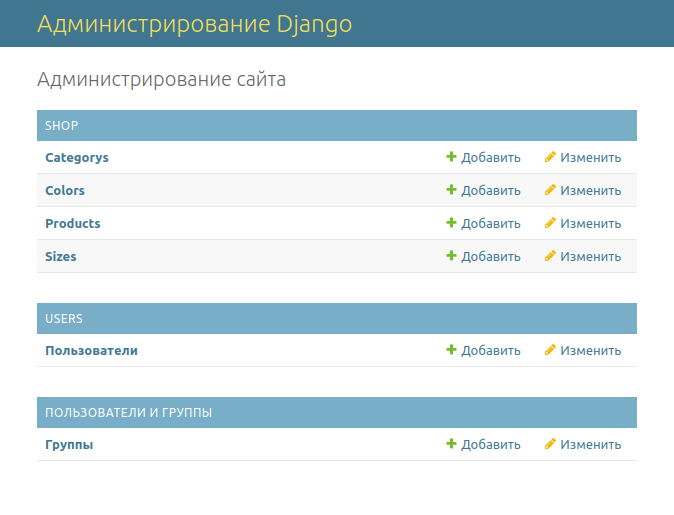


Рисунок 13 - Отображение таблиц в панели администратора

Разработанное в ходе выполнения работы веб- приложение содержит следующие страницы: главная страница, страницы контакты, о нас, магазин, страница товара, добавление и редактирование товара, корзина, оформление заказа, личный кабинет администратора.

Далее на рисунках приведены примеры внешнего вида созданных страниц в браузере.

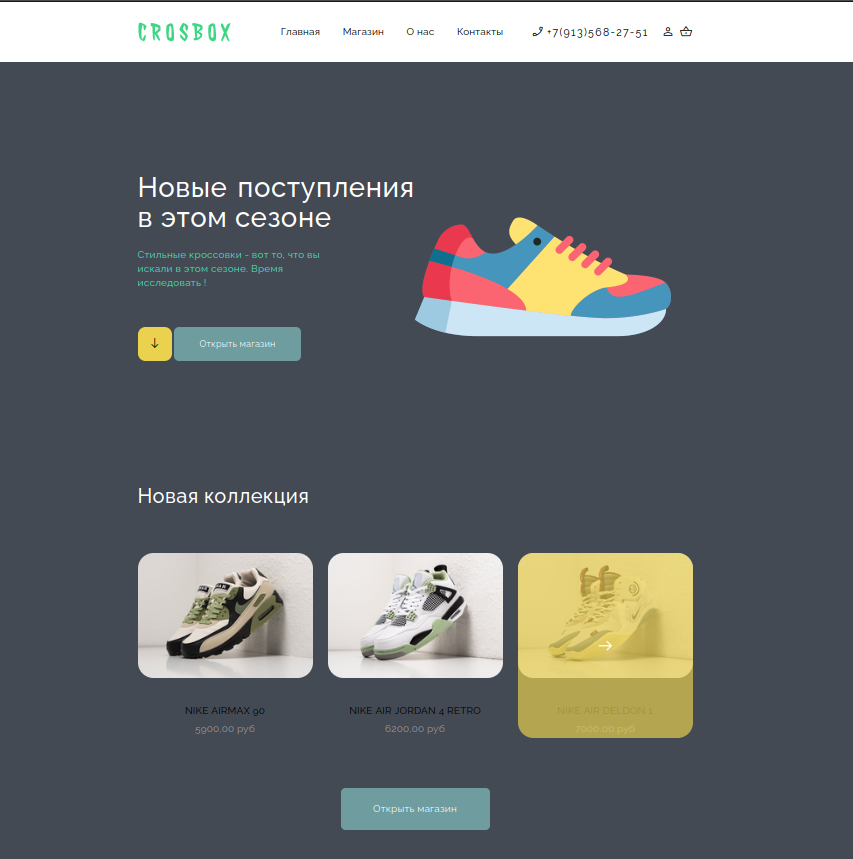


Рисунок 14 - Пример главной страницы

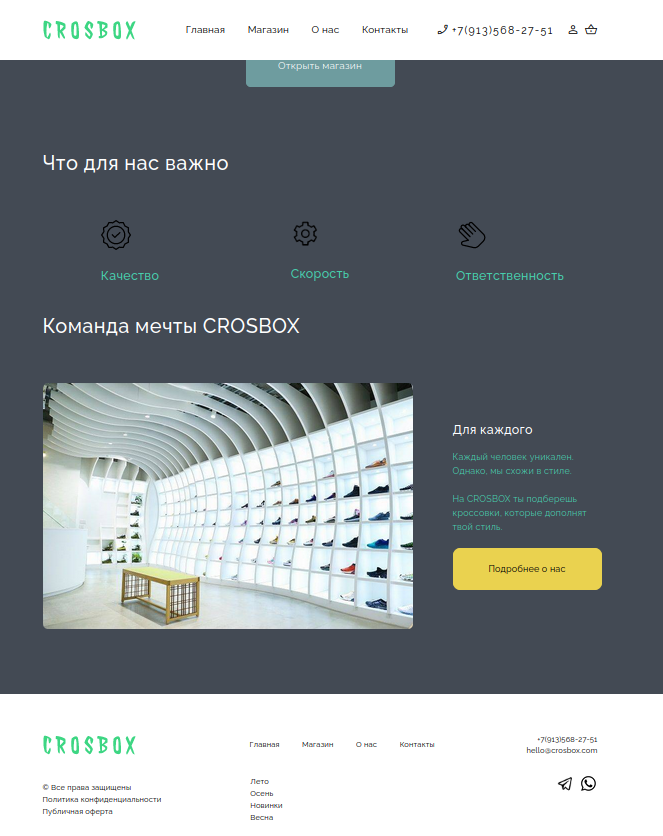


Рисунок 15 - Пример главной страницы

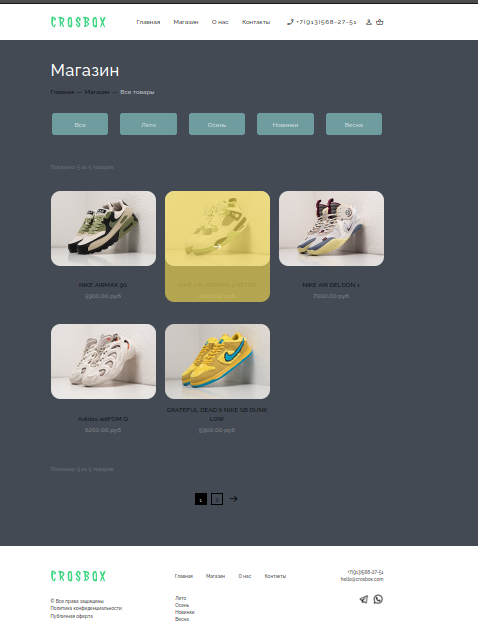


Рисунок 16 - Пример страницы магазина

Как видно из рисунков, навигация добавленная в разметке

базовой страницы, отображаются на каждой из страниц приложения.

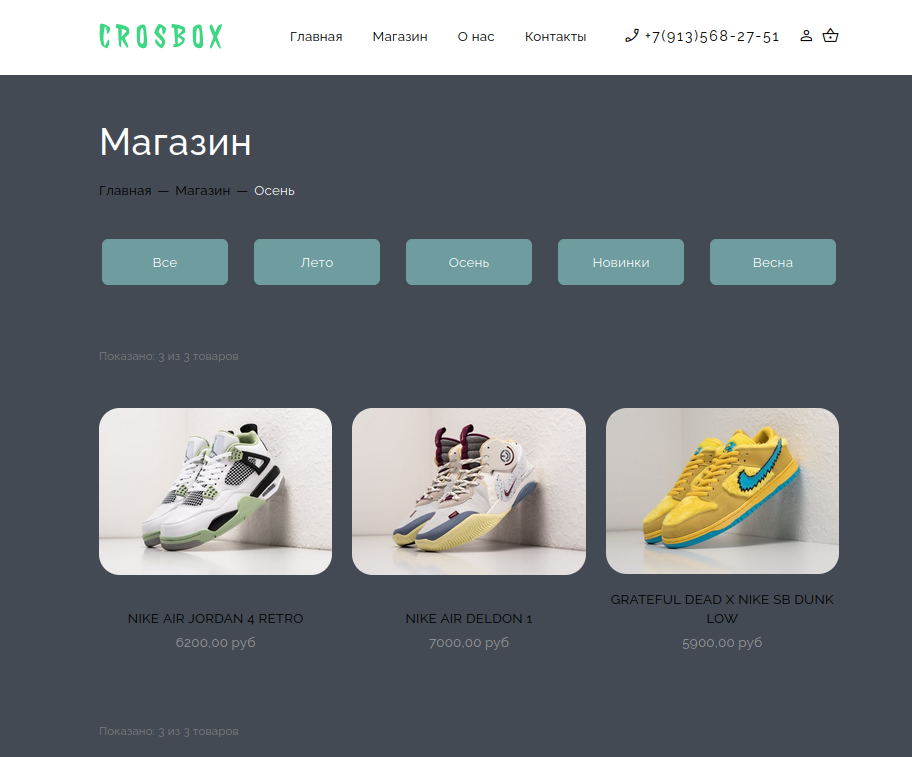


Рисунок 17 - Пример страницы магазина, категория «Осень»

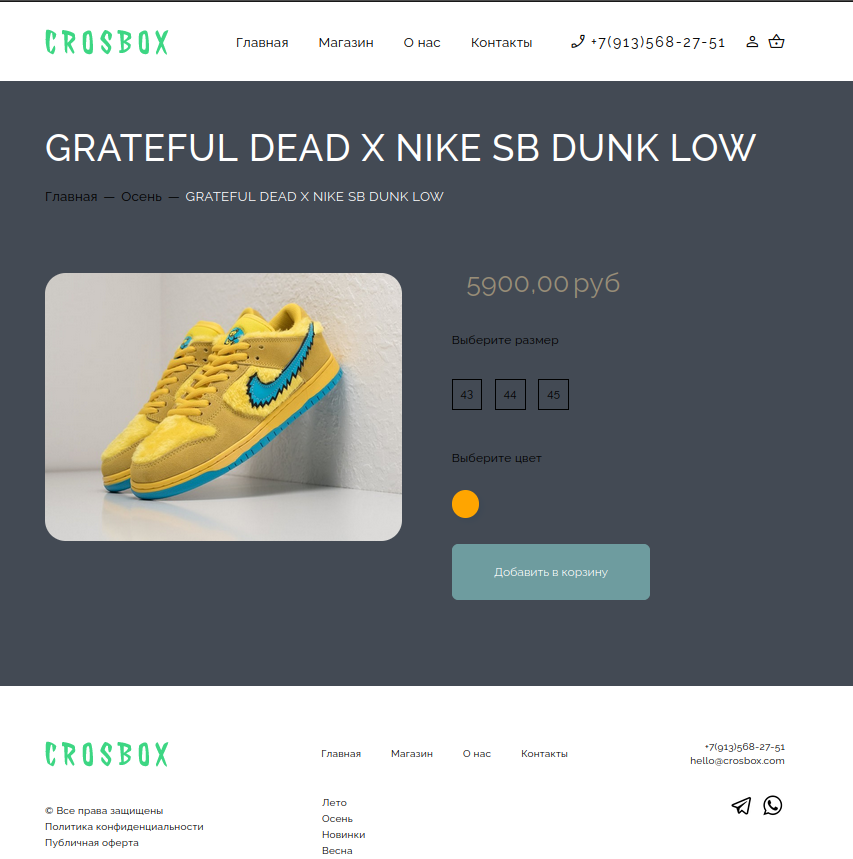


Рисунок 18 - Пример карточки товара

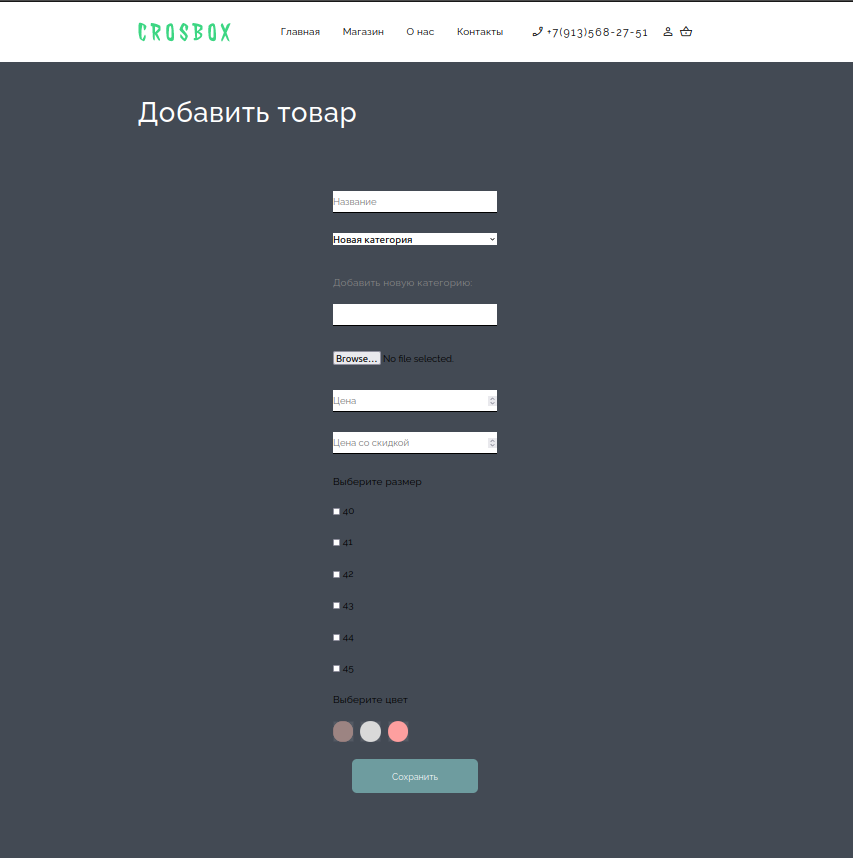


Рисунок 19 - Пример страницы добавления товара

На рисунке приведен вид страницы добавления товара в браузере. На странице размещены формы для добавления информации. Данная страница и страница редактирования товара доступна только авторизированным администраторам.

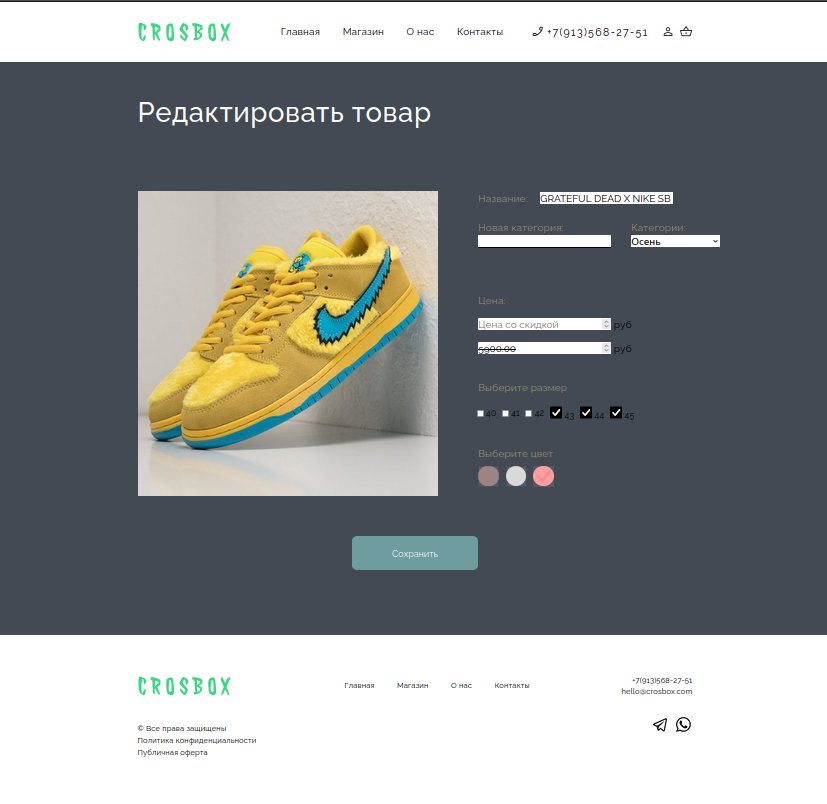


Рисунок 20 - Пример страницы редактирования товара

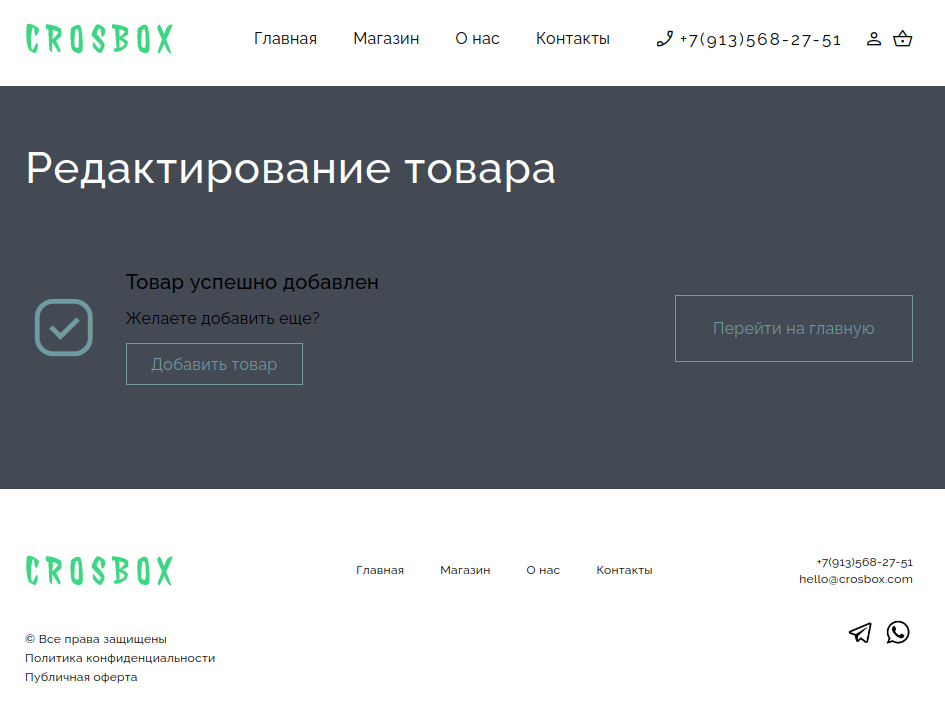


Рисунок 21 - Пример страницы успешного добавления товара в базу данных

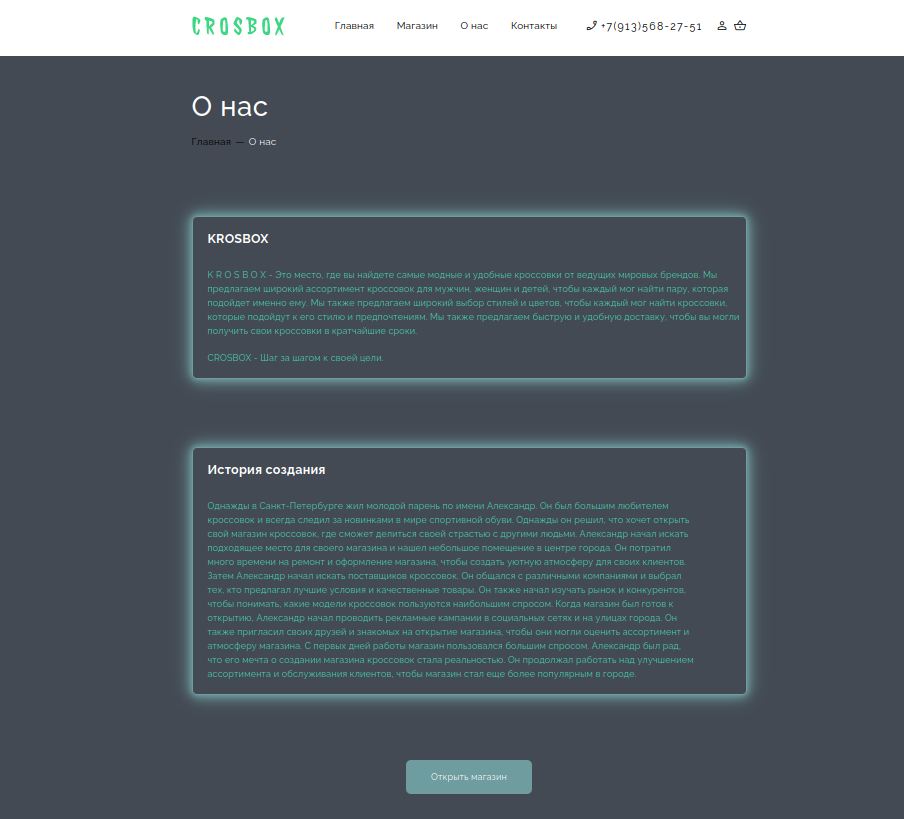


Рисунок 22 - Пример страницы «О нас»

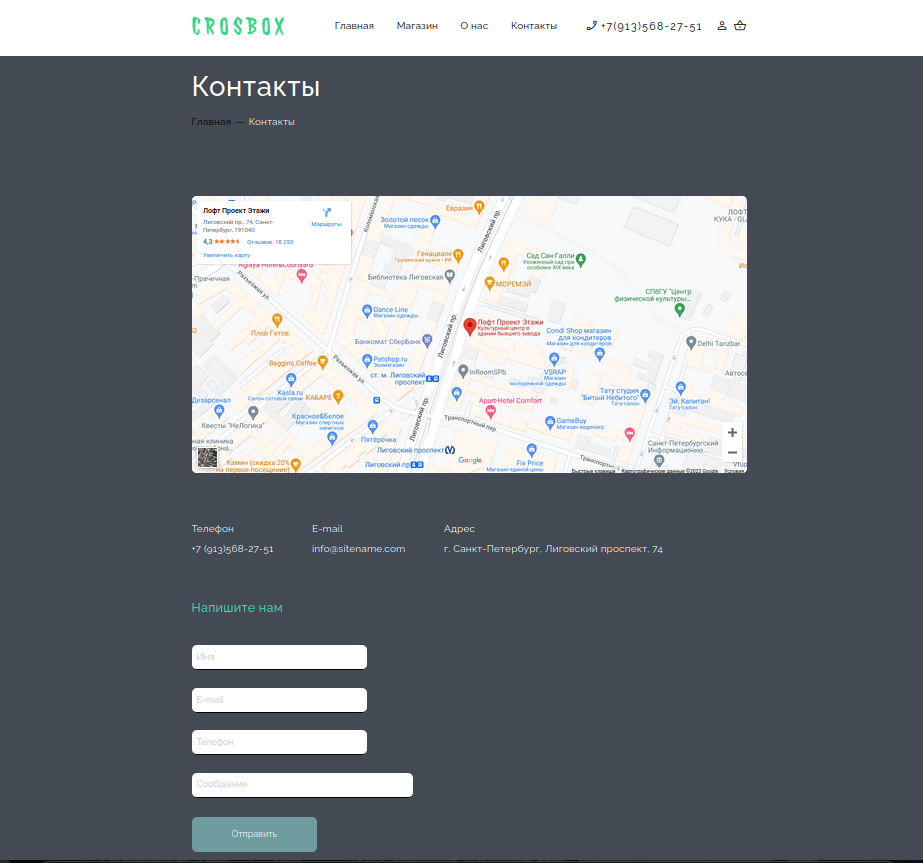


Рисунок 23 - Пример страницы «Контакты»

На странице отображается форма обратной связи, карта и контакты магазина.

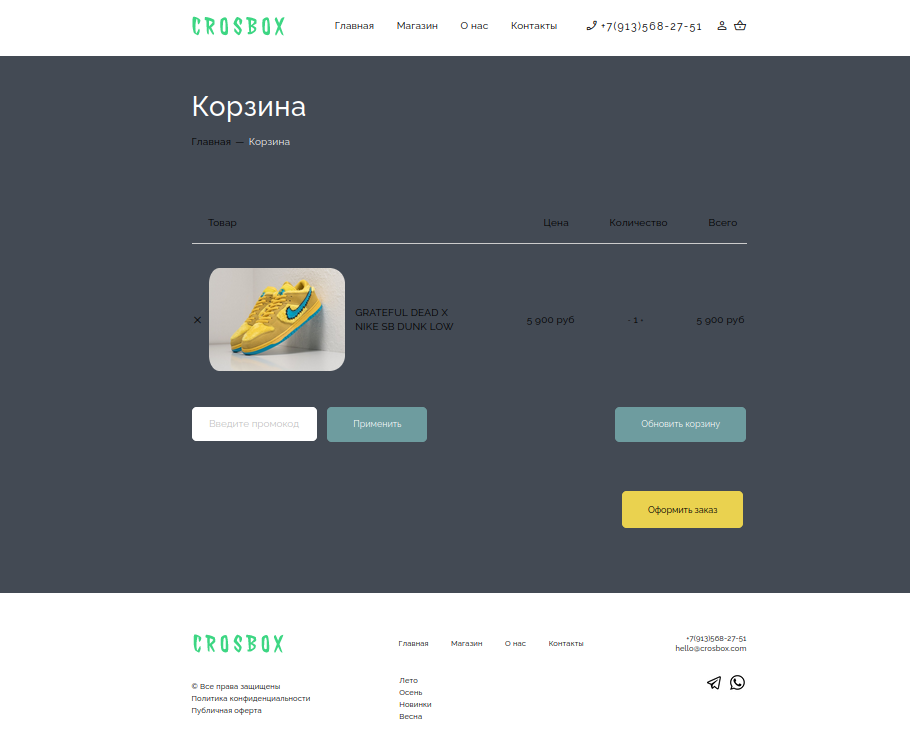


Рисунок 24 - Пример страницы «Корзина»

На данной странице показывается список того, что пользователь добавил в корзину, поле для применения промокода, кнопка обновления корзины и оформления заказа.

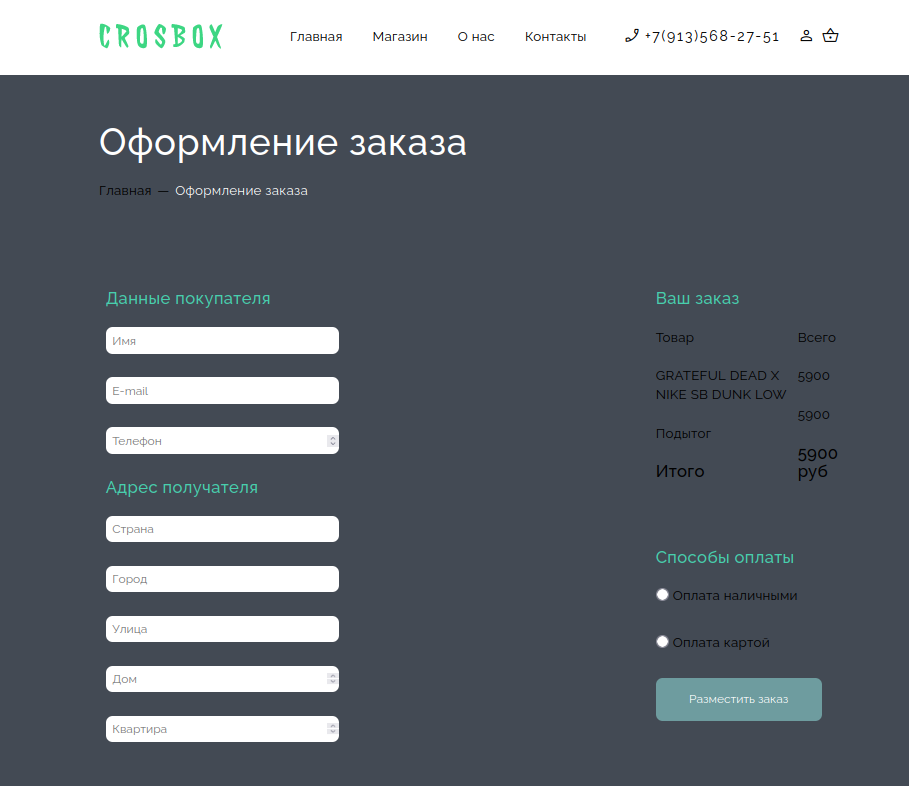


Рисунок 25- Пример страницы «Оформление заказа»

На данной странице пользователи указывают личные данные, контакты для последующей связи менеджера с ними, выбирают способ оплаты, видят общую сумму заказа.

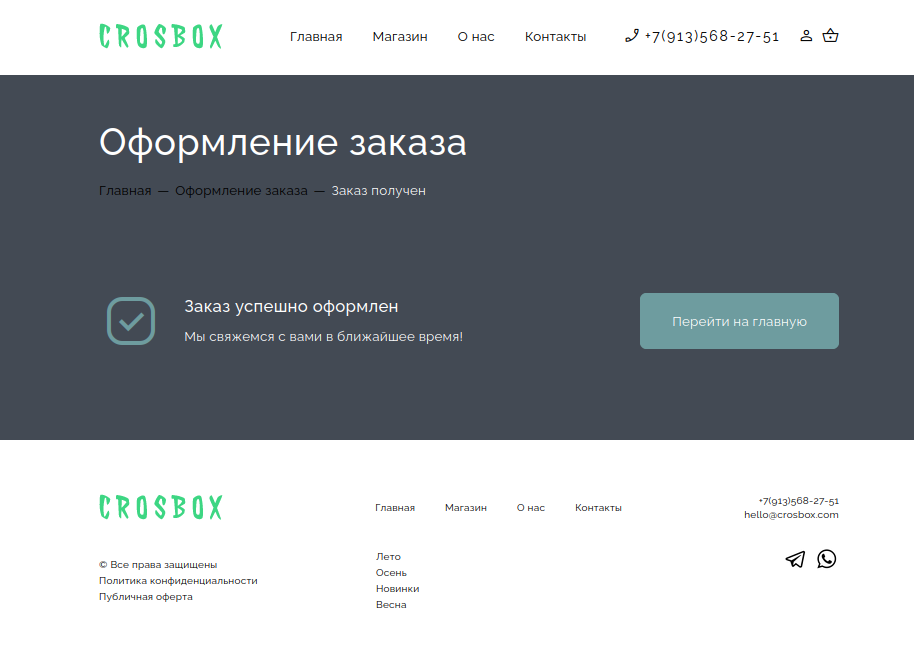


Рисунок 26- Пример страницы «Успешное оформление заказа»

После нажатия на кнопку «разместить заказ», данные пользователя добавляются в базу данных и происходит переадресация на страницу успешного оформления заказа.

Таким образом, приведенные выше рисунки иллюстрируют работоспособность созданного веб-приложения и его функционал.

**ЗАКЛЮЧЕНИЕ**

В итоговой аттестационной работе решена задача создания веб-приложения для предоставления сотрудникам предприятия (магазина) возможностей вести учет товара на складе, а также просмотр проведенных сделок по продаже продукции.

Основные функции приложения:

- Просмотр информации о предприятии и контактной информации;

- Просмотр каталога товаров;

- Добавление товара в корзину;

- Добавление данных пользователя из форм в базу данных;

- Вывод данных: веб-приложение должно отображать данные, полученные из ранее заполненных таблиц базы данных.

- Навигация по страницам: пользователь должен иметь возможность переходить по доступным страницам веб-приложения.

Основными модулями приложения являются модуль описания таблиц базы данных, модуль маршрутизации, модуль функций представления для обработки запросов пользователя, модуль шаблонов, содержащий описание html страниц.

Для создания проекта использовался фреймворк django, который предоставляет необходимые функции для работы с веб-приложениями.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Web. История одной технологии. URL: https://habr.com/ru/companies/geekbrains/articles/277957/

2. Лутц М. Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 1280 с.

3. Сенда Пек "React: Up & Running: Building Web Applications". – СПб.: Питер, 2023. – 300 с.: – (Серия «Бестселлеры O’Reilly»).

4. Набеева Э.Р., Кагиров Р.Р. Технологии создания пользовательских web-приложений // Актуальные проблемы авиации и космонавтики. 2011. №7. URL: https://cyberleninka.ru/article/n/tehnologii-sozdaniya-polzovatelskih-web-prilozheniy (дата обращения: 10.10.2023).

5. Р. Д. Гутгарц, Е. И. Провилков О формализации функциональных требований в проектах по созданию информационных систем // Программные продукты и системы. 2019. №3. URL: https://cyberleninka.ru/article/n/o-formalizatsii-funktsionalnyh-trebovaniy-v-proektah-po-sozdaniyu-informatsionnyh-sistem (дата обращения: 10.10.2023).

6. Ахмед Весам М.А. Защита информации в коммерческих Web-приложениях // Перспективы развития информационных технологий. 2015. №24. URL: https://cyberleninka.ru/article/n/zaschita-informatsii-v-kommercheskih-web-prilozheniyah (дата обращения: 10.10.2023).

7. Иконников Марк Александрович, Карманов Игорь Николаевич Меры и требования к защищенным веб-приложениям // Интерэкспо Гео-Сибирь. 2019. №2. URL: https://cyberleninka.ru/article/n/mery-i-trebovaniya-k-zaschischennym-veb-prilozheniyam (дата обращения: 10.10.2023).

8. Любанович Билл Простой Python. Современный стиль программирования. – СПб.: Питер, 2016. – 480 с.: – (Серия «Бестселлеры O’Reilly»).

9. Майорова Елена Витальевна, Соколовская Светлана Анатольевна, Черток Александр Витальевич, Преимущества гибкого подхода для сопровождения проектов разработки программного продукта // Петербургский экономический журнал. 2019. №4. URL: https://cyberleninka.ru/article/n/preimuschestva-gibkogo-podhoda-dlya-soprovozhdeniya-proektov-razrabotki-programmnogo-produkta (дата обращения: 10.10.2023).

10. Django documentation- URL <https://docs.djangoproject.com/en/4.2/>