**Приложение на Django онлайн-магазина по продаже комплектующих для ПК.**

**Оглавление**

1. **Введение**1.1 Обоснование выбора темы .............................................................. 2  
   1.2 Определение цели и задач исследования ...................................... 2
2. **Основные понятия и определения** ...................................................... 3
3. **Методы и подходы к разработке**3.1 Выбор технологий ............................................................................. 5  
   3.2 Архитектура приложения: ............................................................... 5  
   **Проектирование приложения**4.1 Планирование и анализ требований ............................................ 14  
   4.2 Основные требования .................................................................... 14  
   4.3 Технические требования ................................................................ 14
4. **Разработка в соответствии с созданной документацией**5.1 Планирование разработки ............................................................. 15  
   5.2 Разработка ....................................................................................... 15
5. **Анализ и интерпретация результатов**6.1 Тестирование функционала ........................................................... 16  
   6.2 Интерпретация результатов ........................................................... 16
6. **Заключение**7.1 Обзор выполненной работы .......................................................... 17  
   7.2 Дальнейшие планы ........................................................................ 17

**Приложения**Приложение 1. Пример файловой структуры проекта ................................ 18  
Приложение 2. Список необходимых библиотек ........................................ 19

**1. Введение**

**1.1 Обоснование выбора темы**

В современном мире онлайн-торговля продолжает набирать популярность, что создает потребность в удобных платформах для покупки товаров. Онлайн-магазины комплектующих для ПК предоставляют пользователям возможность выбирать из широкого ассортимента товаров и сравнивать цены, что делает процесс покупки более эффективным.

Выбор темы "Создание онлайн-магазина по продаже комплектующих для ПК" обусловлен актуальностью проекта и растущими потребностями пользователей, ищущих простые и надежные способы приобретения оборудования.

**1.2 Определение цели и задач исследования**

Целью моей работы является создание онлайн-магазина, который позволит пользователям легко находить и заказывать комплектующие для ПК. Для достижения этой цели я определил несколько задач:

* Изучить технологии, которые будут использоваться для разработки.
* Спроектировать архитектуру приложения и структуру базы данных.
* Реализовать пользовательский интерфейс.
* Проверить работоспособность всех функций магазина.

**2. Основные понятия и определения**

В данном разделе я приведу основные термины, которые будут использованы в работе.

Фреймворк: программная платформа, которая упрощает разработку веб-приложений. В рамках данного проекта используется Django.

Пользовательский интерфейс (UI): визуальная составляющая веб-приложения, позволяющая пользователям взаимодействовать с системой.

База данных: структурированная коллекция данных, управляющая хранилищем информации о товарах, пользователях и заказах.

Архитектура MVT: архитектурный паттерн, использующийся в Django, разделяющий приложение на три компонента: Модель (Model), Представление (View) и Шаблон (Template), что способствует лучшей организации кода и его реиспользованию.

Модель (Model): компонент, отвечающий за работу с данными приложения. Модели в Django определяют структуру таблиц базы данных и методы для извлечения, изменения и удаления данных.

Представление (View): компонент, который обрабатывает запросы от пользователя, взаимодействует с моделями и возвращает соответствующий ответ. Представления могут быть реализованы как функции или классы.

Шаблон (Template): компонент, который отвечает за отображение данных, полученных из представлений. Шаблоны используют язык разметки Django для динамической генерации HTML-кода

ORM (Object-Relational Mapping): метод, позволяющий взаимодействовать с реляционными базами данных через объектно-ориентированное программирование. В Django используется встроенная ORM, упрощающая работу с моделями данных.

Сессия: механизм хранения данных о пользователе в течение его сессии работы с приложением, что позволяет реализовать аутентификацию и управление состоянием пользователя.

CSRF (Cross-Site Request Forgery): тип атаки, при которой злоумышленник отправляет неподобающие запросы от имени аутентифицированного пользователя. Django включает защиту от CSRF-атак с помощью токенов.

Валидация данных: процесс проверки корректности и целостности данных перед их обработкой. В Django валидация осуществляется с помощью встроенных средств в моделях и формах.

Пагинация: разбиение данных (например, списка товаров) на страницы для улучшения пользовательского опыта.

**3. Методы и подходы к разработке**

**3.1 Выбор технологий**

В качестве технологии разработки был выбран фреймворк Django, известный своей простотой в использовании и мощными возможностями. Он позволяет быстро разрабатывать настраиваемые веб-приложения с использованием подхода "Model-View-Template" (MVT).

Кроме того, я решил использовать SQLite как базу данных, так как для небольших проектов она прекрасно подходит.

**3.2 Архитектура приложения: Взаимодействие моделей, представлений и шаблонов (MVT)**

**Базовый шаблон (base.html)**

Базовый шаблон (base.html) служит основой для всех страниц интернет-магазина, обеспечивая единообразие структуры и оформления. Он включает в себя загрузку статических файлов, заголовок страницы, шапку сайта и навигационное меню, которое адаптивно выводит ссылки в зависимости от состояния аутентификации пользователя.

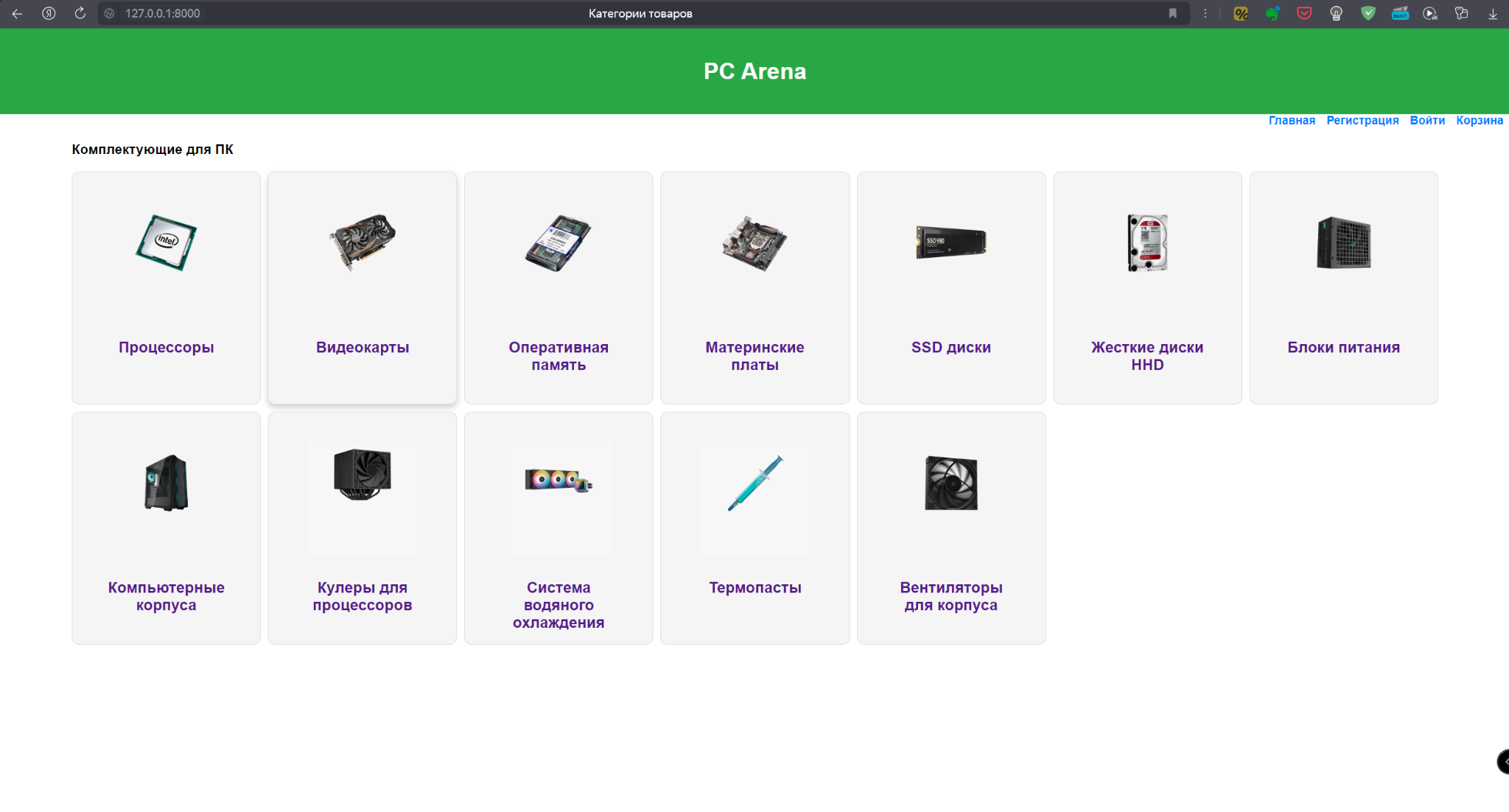
Если пользователь аутентифицирован, в навигационном меню отображается приветствие с текстом «Добро пожаловать, “имя входящего пользователя”. Также предоставляются ссылки на главную страницу , чтобы вернуться к продуктам, и ссылка на страницу выхода для завершения сеанса. Если пользователь не аутентифицирован, отображаются ссылки на главную страницу , страницу регистрации , где новый пользователь может создать учетную запись, и страницу входа для доступа к своему аккаунту.

Блок {% block content %} позволяет каждой странице вводить уникальный контент, сохраняя общую стилизацию и структуру, что делает разработку более организованной и упрощает управление интерфейсом приложения.

**Главная страница (category\_list.html)**

На главной странице используется модель Category, которая хранит информацию о категориях товаров их изображениях. Представление category\_list обрабатывает запрос, извлекая все категории из базы данных и передавая их в шаблон category\_list.html. Этот шаблон визуализирует категории, отображая их названия и изображения с помощью циклов, создавая HTML-код, который отображается в браузере пользователя.

Рис.1. Главная страница интернет-магазина “PC Arena”



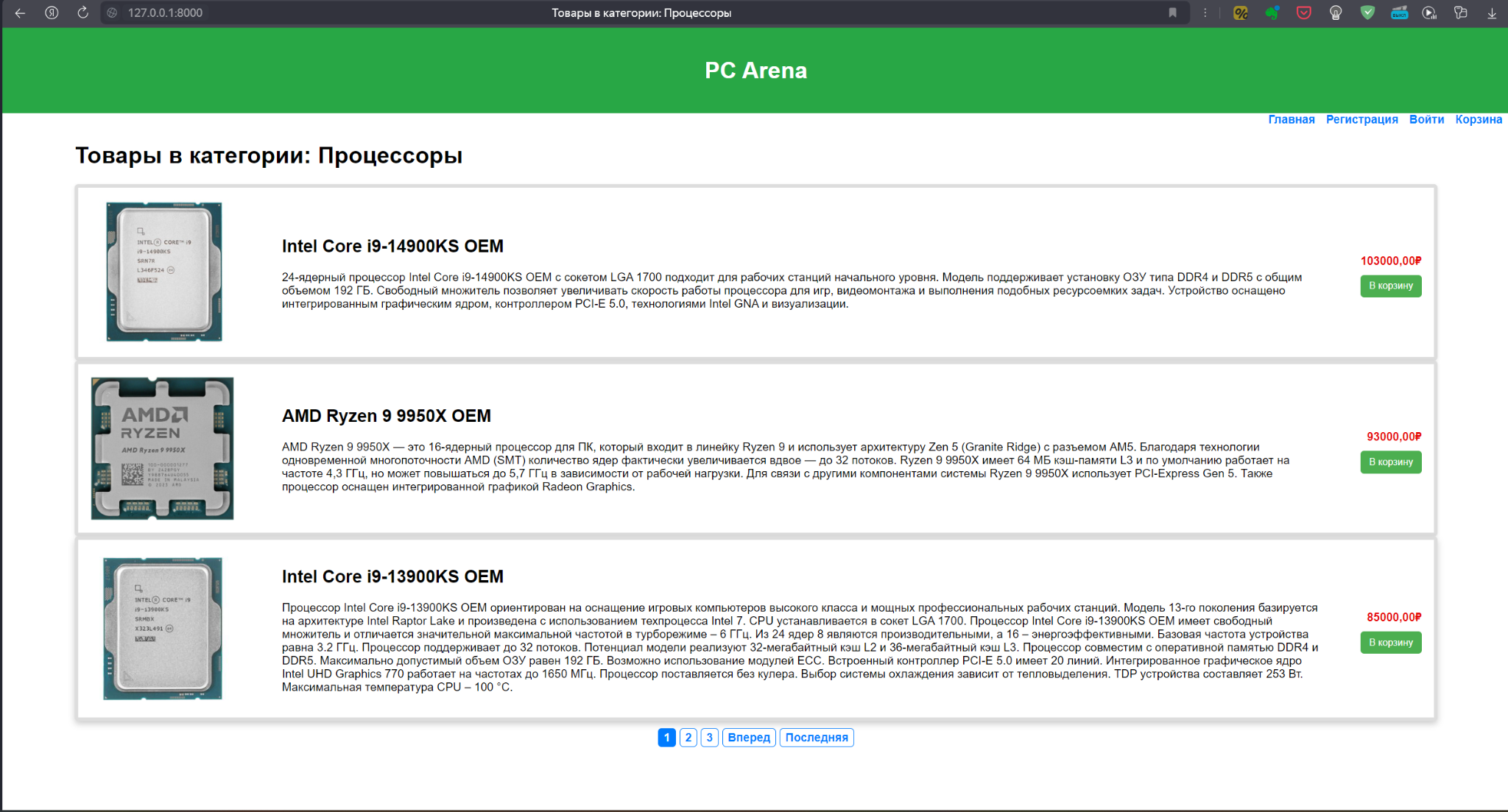
**Страница товаров определенной категории (product\_list.html)**

На странице категории используется модель Product, содержащая детали товаров. Представление product\_list фильтрует товары по выбранной категории на основе параметров запроса URL, получая данные из базы.

Для улучшения пользовательского опыта и удобства навигации по большому количеству товаров, реализована пагинация. Используя класс Paginator из Django, товары разбиваются на несколько страниц, что позволяет пользователям просматривать определенное количество товаров за один раз. При этом, в шаблоне product\_list.html отображается текущая страница товаров, а внизу страницы добавлены ссылки для перехода к другим страницам.

Шаблон product\_list.html формирует карточки товаров, отображая их изображения, названия и цены, а также включает кнопку добавления "В корзину", при нажатии которой выходит уведомление о добавление товара в корзину. Пагинация делает процесс поиска и выбора товара более удобным, позволяя пользователю легко перемещаться между различными страницами с товарами.

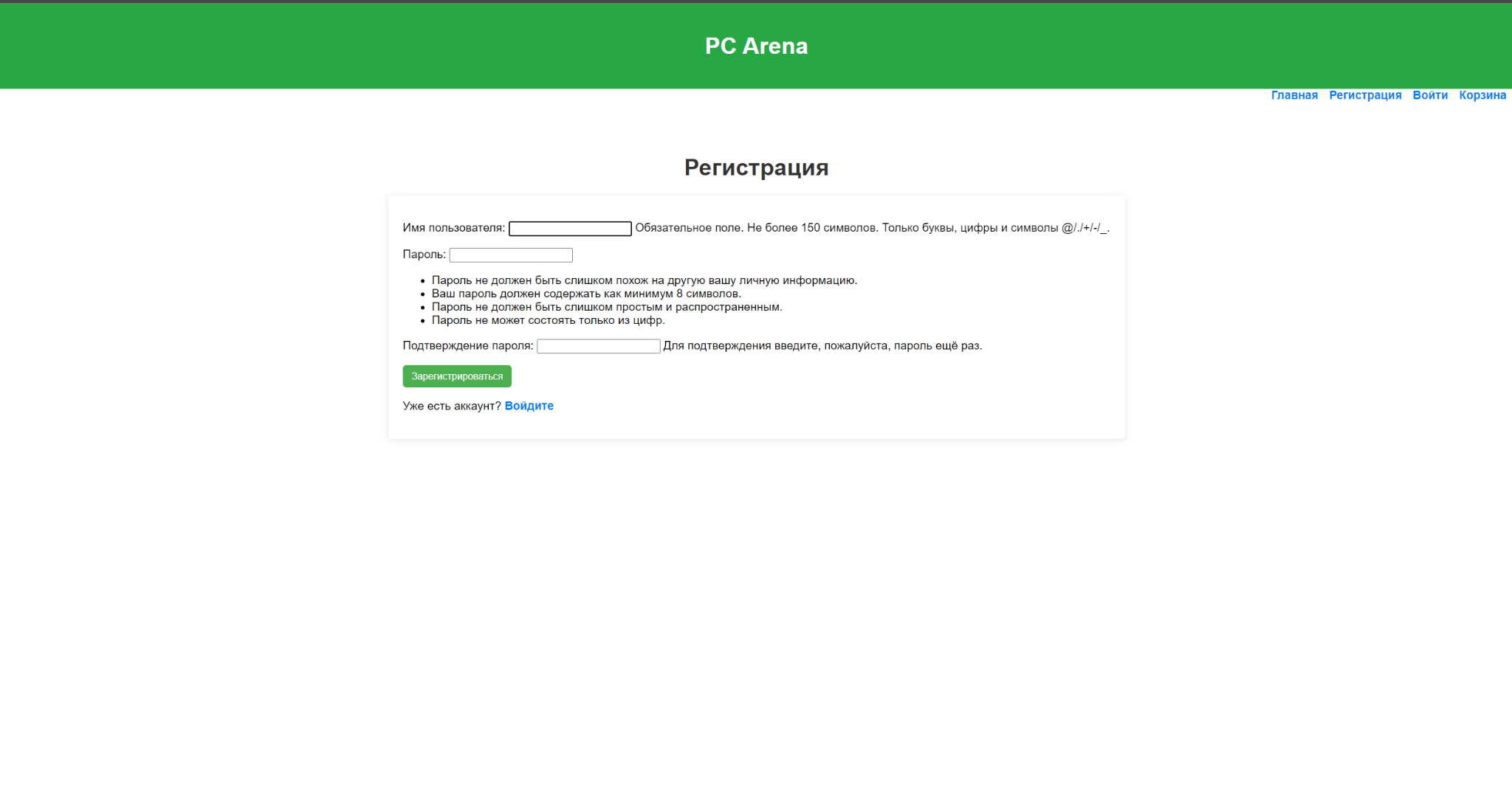
Рис.2.Страница товаров



**Страница регистрации (registration\_page.html)**

На странице регистрации используется модель User, которая хранит информацию о пользователях, включая имя пользователя, адрес электронной почты и пароль. Представление registration\_page обрабатывает запрос, позволяя пользователям заполнить форму с необходимыми данными. После отправки формы происходит валидация введенной информации: проверяются уникальность имени пользователя и адреса электронной почты, а также соблюдение минимальных требований к паролю. При успешной валидации новый пользователь создается в базе данных с хэшированием пароля для повышения безопасности. Шаблон registration\_page.html формирует интерфейс, отображая сообщения об ошибках при неправильном вводе и предлагая пользователю возможность активации учетной записи через электронную почту, если такая функция реализована.

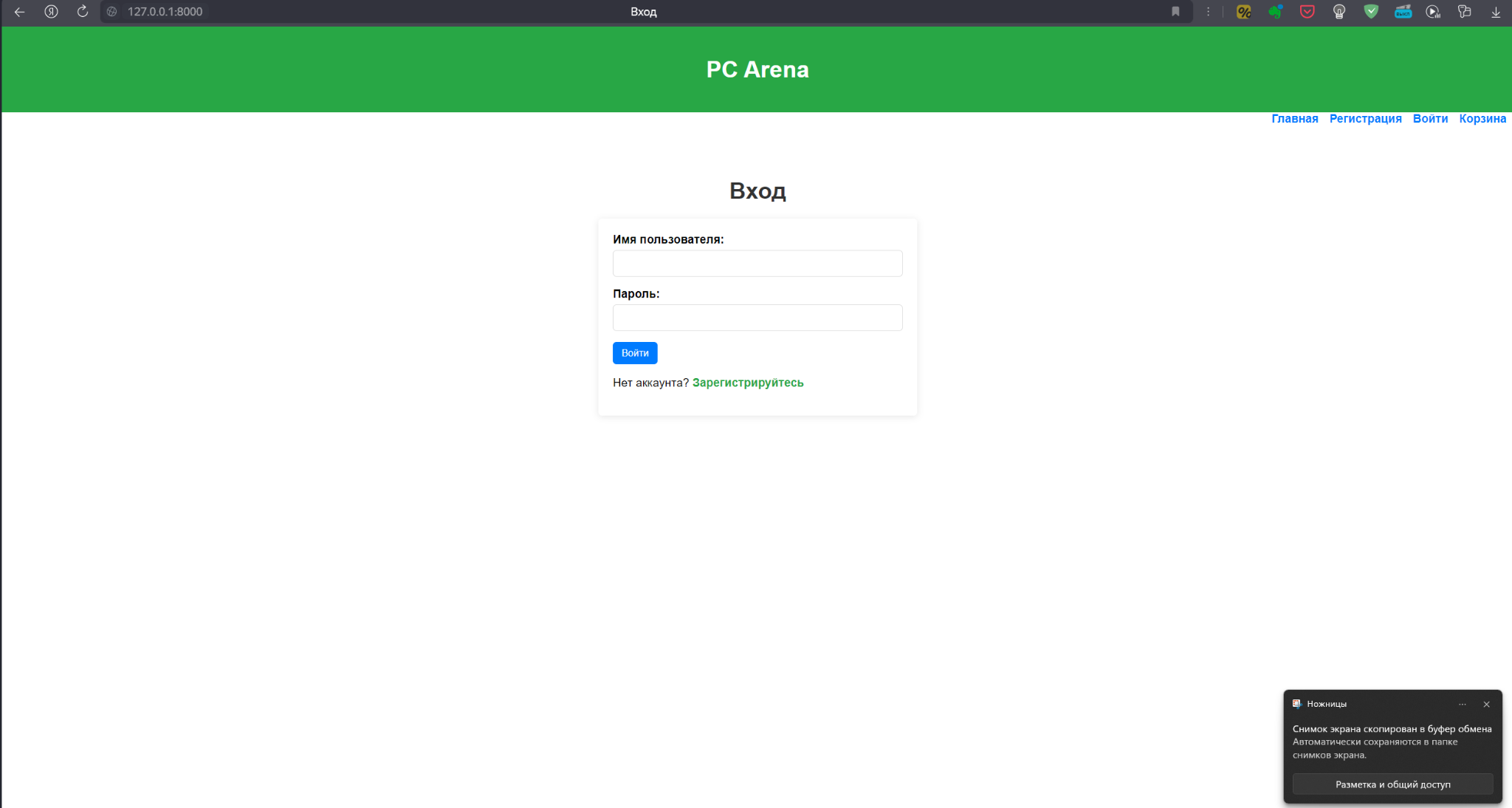
Рис.3. Страница регистрации



**Страница входа (login\_page.html)**

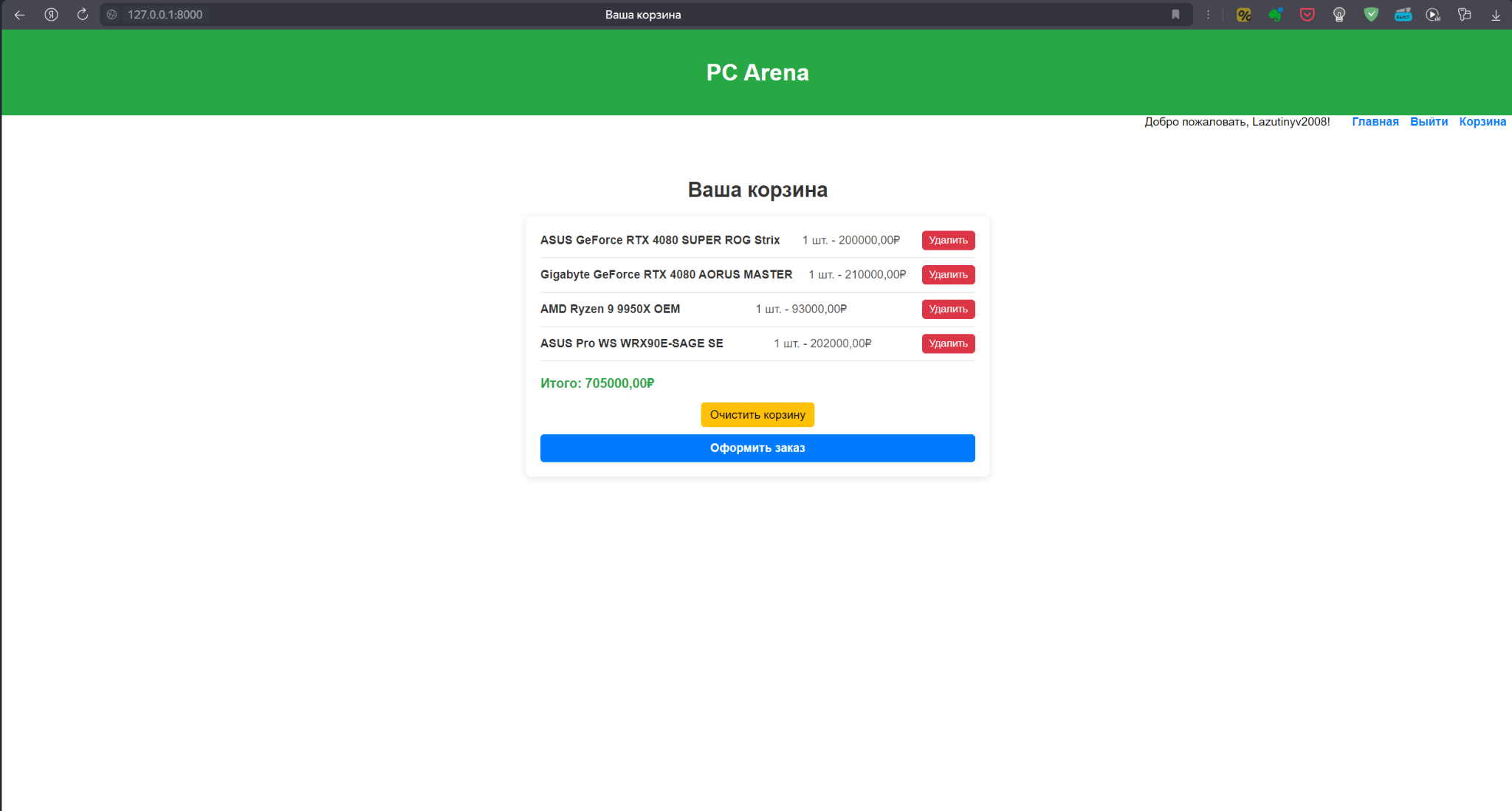
На странице входа используется модель User для аутентификации пользователей в системе. Представление login\_page обрабатывает запросы от пользователей, которые вводят свои учетные данные (имя пользователя и пароль). При отправке формы происходит проверка введенной информации с помощью метода authenticate(), позволяющего убедиться в правильности учетных данных. В случае успешной аутентификации создается сессия для пользователя, что позволяет ему оставаться в системе. Шаблон login\_page.html формирует интерфейс входа, включающий сообщения об ошибках в случае неверного ввода, а также ссылку на страницу регистрации для новых пользователей, что облегчает доступ к системе и делает процесс аутентификации интуитивно понятным.

рис.4. Страница входа



**Корзина (cart.html)**

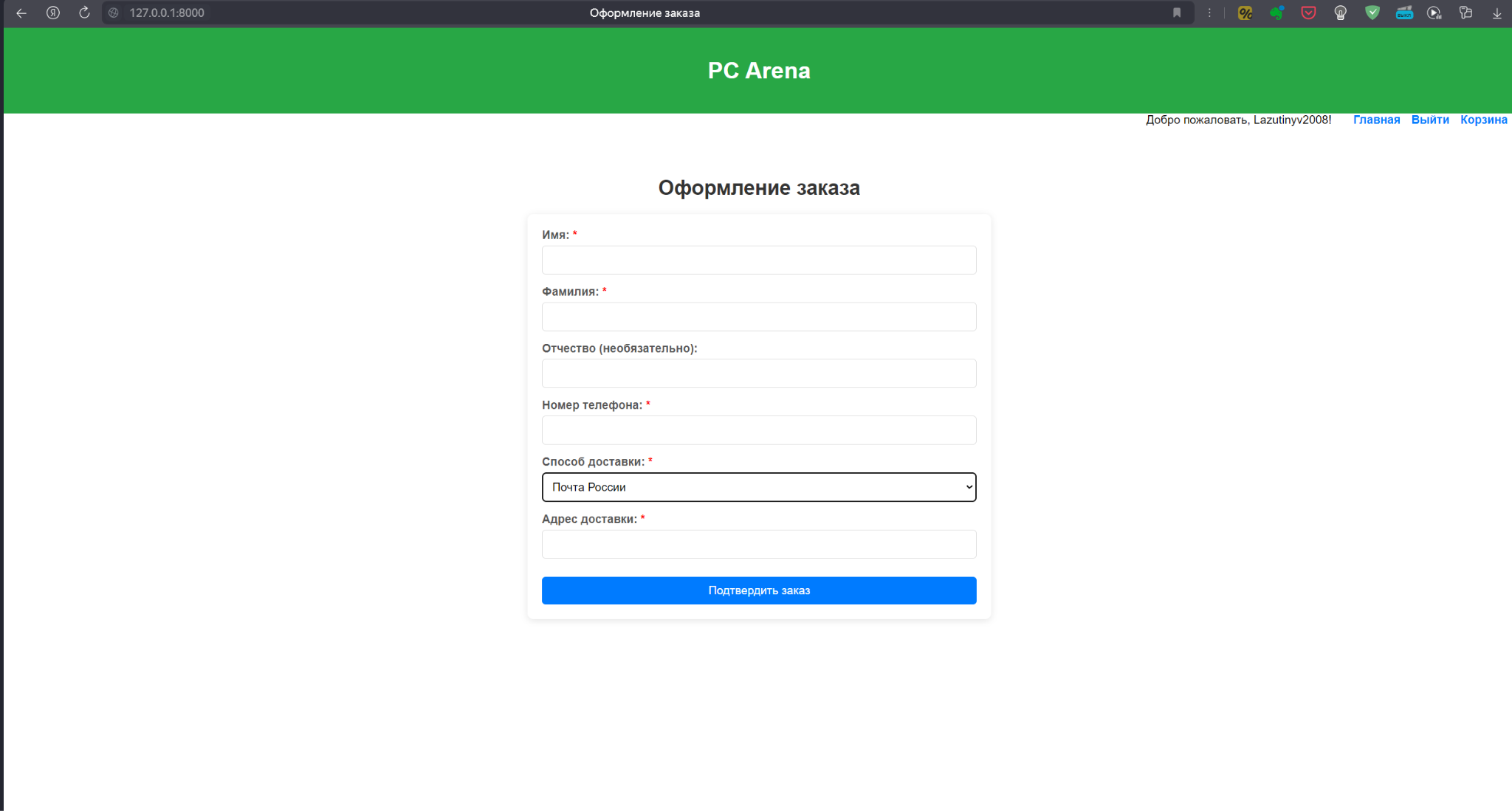
В модели Cart хранится связь между пользователем и его корзиной, а модель CartProducts управляет товарами в корзине. Представление cart извлекает данные из модели Cart и связанные с ней CartProducts для отображения всех добавленных товаров. Шаблон cart.html отвечает за визуализацию этих данных, предоставляя пользователю возможность видеть список добавленных товаров, общую сумму и включает кнопки для удаления товаров из корзины и оформления заказа, обеспечивая простоту взаимодействия.

рис.5. Страница корзины  


**Оформление заказа (checkout.html)**

На этапе оформления заказа используется модель Order, которая хранит информацию о заказах, включая данные о пользователе и его корзине. Представление checkout обрабатывает запрос на оформление, получая данные о текущем пользователе и его корзине, а также запрашивает необходимую информацию для завершения заказа и уведомляет об успешном оформлении. Шаблон checkout.html формирует форму для ввода информации для доставки, включая поля для имени, номера телефона, адреса и выбора метода доставки.Это позволяет пользователю удобно и быстро завершить процесс покупки, отправляя собранные данные для создания нового заказа и формирования его в базе данных. Важной функцией также является возможность самоочистки корзины после оформления заказа, обеспечивая тем самым удобство и контроль над покупками.

рис.5. Страница оформления заказа



**Админ-панель**

Админ-панель вашего интернет-магазина, разработанного на основе архитектуры MVT, предоставляет мощные инструменты для управления различными аспектами приложения. Вот несколько ключевых возможностей, которые можно реализовать через админ-панель:

**Возможности админ-панели**

Управление категориями товаров: Администраторы могут добавлять, редактировать и удалять категории товаров. Это позволяет поддерживать актуальность и структуру каталога, управляющего различными типами комплектующих для ПК.

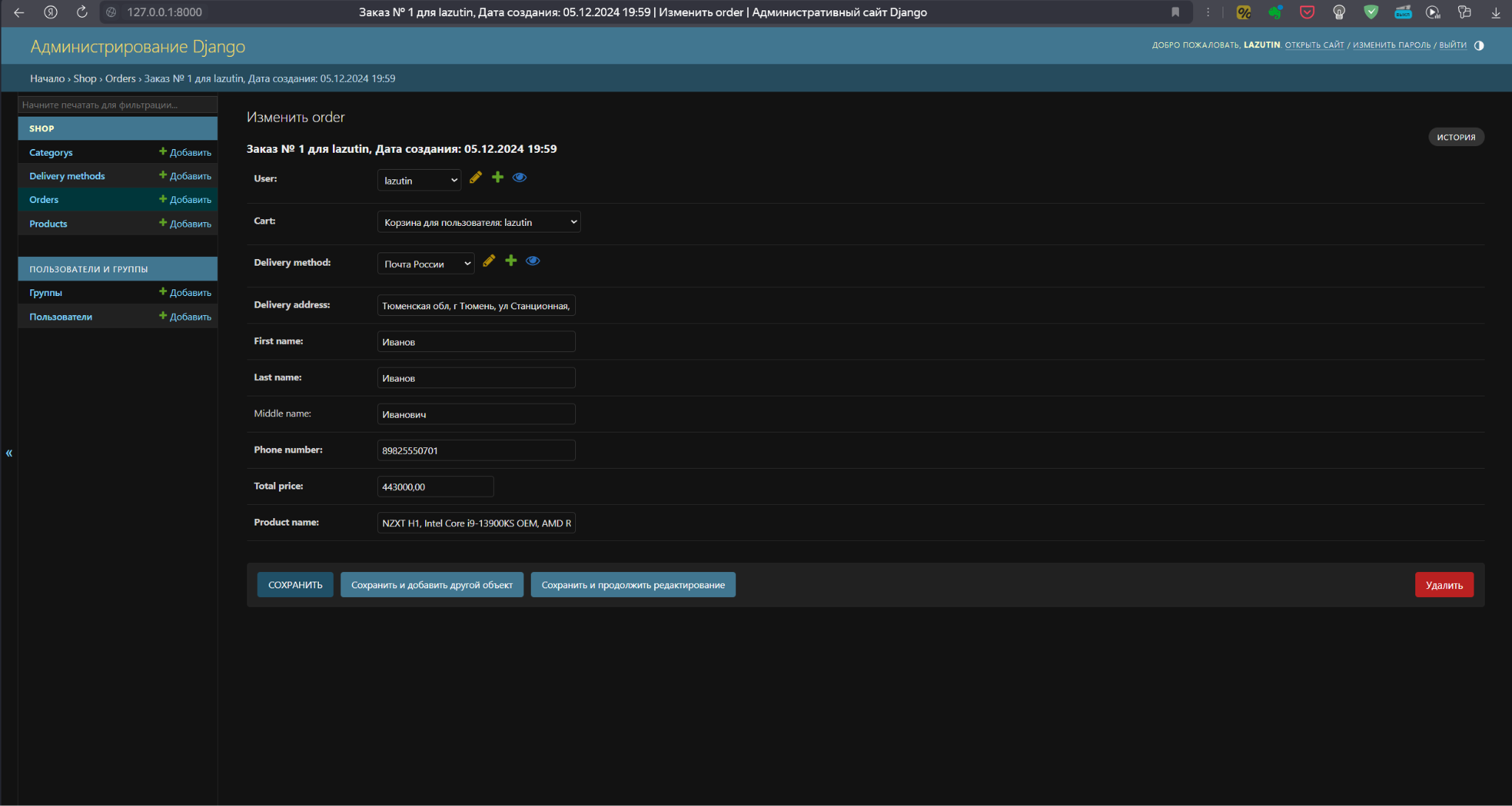
Управление товарами: Через админ-панель администраторы могут управлять списком продуктов, включая добавление новых товаров, редактирование их характеристик (таких как название, описание, цена, изображение и категории) и удаление устаревших или неподходящих товаров. Это позволяет поддерживать актуальность информации в магазине.

Управление заказами: Администраторы могут просматривать все заказы, размещенные пользователями, включая детали заказов и информацию о пользователях. Это позволяет отслеживать активность клиентов и эффективно управлять процессом выполнения заказов.

Управление пользователями: Админ-панель позволяет администраторам управлять учетными записями пользователей, включая активацию, деактивацию и изменение ролей пользователей. Это обеспечивает безопасный доступ к функционалу магазина только для уполномоченных лиц.

Аналитика и отчеты: Через админ-панель можно добавлять разделы для отчетности, где администраторы смогут просматривать статистику продаж, популярные товары и категории, активность пользователей. Это поможет принимать обоснованные решения для оптимизации бизнеса.

рис.6.Страница админ-панели



**4. Проектирование приложения**

**4.1 Планирование и анализ требований**

На этом этапе я определил, какие функции должен включать мой интернет-магазин. Это был первый шаг к созданию структуры приложения.

**4.2 Основные требования**

* Мой магазин должен обеспечивать следующие функции:
* Регистрация и аутентификация пользователей.
* Просмотр и управление товарами.
* Добавление товаров в корзину и оформление заказов.
* Удаление товаров из корзины
* управление данными с помощью админ-панели

**4.3 Технические требования**

Для работы приложения мне понадобятся:

* Python 3.6 и выше.
* Django 3.0 и выше.
* SQLite в качестве базы данных.

**5. Разработка в соответствии с созданной документацией**

**5.1 Планирование разработки**

Я разбил процесс разработки на несколько этапов: проектирование, реализация и тестирование. Каждый этап был важен, чтобы достичь конечной цели.

**5.2 Разработка**

Работа была разделена между фронтендом и бэкендом. Я использовал HTML и CSS для создания интерфейса.

**6. Анализ и интерпретация результатов**

**6.1 Тестирование функционала**

Было проведено тестирование всех функций приложения, таких как регистрация, добавление товаров,очистка корзины и оформление заказов.

**6.2 Интерпретация результатов**

Тестирование показало, что приложение работает корректно и соответствует всем требованиям. Интерфейс был интуитивно понятным, а все функции выполнялись без ошибок.

**7. Заключение**

**7.1 Обзор выполненной работы**

В рамках данной дипломной работы был успешно создан интернет-магазин комплектующих для ПК. Проект продемонстрировал основные принципы MVT и возможности фреймворка Django. Сейчас у меня есть работающий интернет-магазин, который позволяет пользователям регистрироваться, просматривать товары, добавлять, удалять товары из корзины, оформлять заказы, управлять данными с помощью админ-панели. Все поставленные задачи были успешно выполнены.

**7.2 Дальнейшие планы**

При необходимости дальнейшего развития этого проекта можно добавлять новые функции и улучшать его. Например, интеграция с платежными системами, поиска товаров,добавления в избранное,систему рекомендаций,отслеживание процесса доставки заказа и возможность оставлять отзывы и оценки товарам могут стать отличными следующими шагами.

# **Приложение 1. Пример файловой структуры проекта**

online-store

| db.sqlite3

| manage.py

| structure.txt

|

+---media

| +---categories

| |

| \---products

+---online\_store

| | asgi.py

| | settings.py

| | urls.py

| | wsgi.py

| | \_\_init\_\_.py

| |

| \---\_\_pycache\_\_

| settings.cpython-311.pyc

| urls.cpython-311.pyc

| wsgi.cpython-311.pyc

| \_\_init\_\_.cpython-311.pyc

|

\---shop

| admin.py

| apps.py

| models.py

| tests.py

| views.py

| \_\_init\_\_.py

|

+---migrations

| | 0001\_initial.py

| | \_\_init\_\_.py

| |

| \---\_\_pycache\_\_

# **Приложение 2. Список необходимых библиотек**

asgiref==3.8.1

Django==5.1.3

django-debug-toolbar==4.4.6

pillow==11.0.0

pytz==2024.2

sqlparse==0.5.2

tzdata==2024.2