Допустить к защите

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Зам. директора по УМР

\_\_\_\_\_\_\_\_\_\_ О.В. Фомичева

**ДИПЛОМНЫЙ ПРОЕКТ**

Тема:Разработка веб-приложения для работы с клиентами интернет-магазина кожаной одежды

Студент группы 9ИСП-211 / Рахманкулова А.Р.

№ группы подпись Фамилия И.О.

Специальность 09.02.07 Информационные системы и программирование

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_ / Семёнов П.Н. /

подпись Фамилия И.О.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc200900384)

[1 ТЕОРЕТИКО-АНАЛИТИЧЕСКИЙ РАЗДЕЛ 5](#_Toc200900385)

[1.1 Идентификация предметной области 5](#_Toc200900386)

[1.2 Выбор методологии и технологии концептуального моделирования ИС 9](#_Toc200900387)

[1.3 Анализ предметной области 12](#_Toc200900388)

[2 РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ 16](#_Toc200900393)

[2.1 Анализ требований и разработка спецификаций 16](#_Toc200900394)

[2.2 Проектирование и разработка дизайна интерфейса 22](#_Toc200900395)

[2.3 Разработка кода веб-приложения и программирование 27](#_Toc200900396)

[2.4 Тестирование и отладка ИС 40](#_Toc200900397)

[ЗАКЛЮЧЕНИЕ 43](#_Toc200900398)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 45](#_Toc200900399)

[ПРИЛОЖЕНИЕ А 46](#_Toc200900400)

ВВЕДЕНИЕ

Современный рынок электронной коммерции стремительно развивается, и всё больше малых и средних предприятий переходят к онлайн-формату продаж. Одной из востребованных ниш в сфере fashion-ритейла является продажа одежды, в частности изделий из кожи — уникального и нишевого товара, ориентированного на аудиторию с определёнными эстетическими предпочтениями. Для успешного взаимодействия с клиентами интернет-магазина необходим удобный, надёжный и функциональный веб-интерфейс, который обеспечивает не только удобство покупок, но и полноценную систему управления заказами, учёта клиентов и аналитики.

Актуальность темы обусловлена необходимостью повышения эффективности взаимодействия интернет-магазинов с клиентами, автоматизации рутинных операций и повышения конкурентоспособности бизнеса за счёт современных ИТ-решений. Особенно важным это становится для малых нишевых предприятий, которым требуется простое в обслуживании и гибкое в настройке решение. В условиях цифровизации бизнеса и растущей конкуренции наличие такой системы становится не только актуальным, но и необходимым условием успешной коммерческой деятельности.

Целью дипломной работы является разработка веб-приложения для интернет-магазина по продаже изделий из кожи, обеспечивающего эффективное взаимодействие с клиентами.

Для достижения этой цели необходимо решить следующие задачи:

* провести анализ предметной области и требований к системе;
* выбрать подходящую методологию и средства проектирования;
* спроектировать интерфейс и архитектуру информационной системы;
* реализовать клиентскую и серверную части веб-приложения;
* провести тестирование системы на соответствие заданным требованиям.

Объектом исследования является процесс разработки веб-приложения для интернет-магазина по продаже кожаной одежды.

Предметом исследования является проектирование и реализация клиентской и серверной частей информационной системы, включающей пользовательский интерфейс, бизнес-логику и взаимодействие с базой данных.

В ходе разработки были использованы следующие методы и подходы:

* анализ предметной области и требований к функционалу системы;
* проектирование структуры системы с учётом архитектуры Django (модель–шаблон–представление);
* создание моделей данных с использованием встроенной системы Django ORM для работы с базой данных;
* визуальное моделирование структуры системы с помощью диаграмм UML;
* использование встроенных инструментов Django для ускорения разработки (админ-панель, формы, маршрутизация и др.);

Выпускная квалификационная работа состоит из введения, двух разделов, заключения, списка источников и приложений.

В первом разделе проводится теоретико-аналитическое исследование: описание предметной области, выбор методологии, построение UML-моделей информационной системы.

Во втором разделе изложен практический процесс разработки: формализация требований, проектирование интерфейса, реализация кода, тестирование и подготовка пользовательского руководства.

В заключении подводятся итоги проделанной работы, оцениваются результаты и предлагаются направления для дальнейшего развития проекта.

1. **ТЕОРЕТИКО-АНАЛИТИЧЕСКИЙ РАЗДЕЛ**
   1. **Идентификация предметной области**

Предметной областью данного дипломного проекта является торговля кожаной одеждой компанией розничной торговли «Heart of Style». Это означает, что основное внимание уделяется процессам, связанным с закупкой, реализацией и логистикой товаров из кожи.

Компания «Heart of Style» специализируется на приобретении кожаной одежды отечественных производителей и ее последующем продвижении через собственные дистрибьюторские сети и розничные магазины. Этот аспект играет важную роль в картине общей деятельности компании «Heart of Style».

Таким образом, предметная область компании «Heart of Style» охватывает не только сферу торговли кожаной одеждой, но также включает в себя логистические процессы, контроль качества товаров, управление складами, хранение продукции, взаимодействие с различными сторонними компаниями, а также взаиморасчеты с клиентами и партнерами.

При анализе предметной области необходимо учитывать все аспекты деятельности компании «Heart of Style», начиная от взаимодействия с поставщиками и заканчивая взаимодействием с конечными потребителями веб-приложения. Важно понимать, что успешная торговля кожаной одеждой требует слаженной работы всех звеньев цепочки поставок и продаж товара, а также обслуживания клиентов.

Коммуникация с клиентами представляет собой многоаспектный процесс, который включает в себя гораздо больше, чем просто успешную продажу продукции. Этот процесс начинается с привлечения внимания покупателя к определенному интернет-ресурсу, с убеждением его в том, что именно эта компания является наиболее привлекательной и клиентоориентированной среди всего остального множества интернет-магазинов кожаной одежды. Важным аспектом является создание у покупателя ощущения, что данная организация отличается особым вниманием к их потребностям и полной лояльностью к каждому клиенту.

Однако, взаимодействие с клиентом не ограничивается только фактом завершения финансовой сделки. После завершения покупки, компания «Heart of Style» проявляет особое внимание к каждому клиенту, поддерживая с ним постоянную связь после приобретения товара, особенно в случае выявления дефекта и негативного пользования данным интернет-ресурсом. Такой подход направлен на предотвращение возможных неприятных ситуаций, на укрепление веры клиентов в лояльность компании к каждому из них и на повышение вероятности их возвращения к данному продавцу в будущем, а не к конкурентам.

На основе всего сказанного, можно выделить ключевые бизнес-процессы компании «Heart of Style»:

* закупка товара;
* хранение товара на складе;
* организация продаж;
* проведение корректных взаиморасчетов с конечным пользователем;
* проведение корректных взаиморасчетов с поставщиками.

Кроме того, компания стремится оставлять у каждого клиента настолько позитивный опыт, чтобы он создавал предпосылки для последующих повторных покупок.

Основываясь на данном анализе, специалисты компании могут разрабатывать и внедрять в жизнь клиентов эффективные стратегии управления, оптимизировать бизнес-процессы и, как следствие, значительно повышать конкурентоспособность компании на рынке кожаной одежды, что в свою очередь способствует укреплению ее позиций на рынке и привлечению новых клиентов.

Оформление поставки товара на склад требует оформление следующих документов:

1. заказ к поставщику,
2. договор с поставщиком,
3. сертификат соответствия и иные документы, подтверждающие соответствие требованиям к качеству,
4. приходная накладная,
5. акт приема товара,
6. счет поставщика.

Оформление продажи товара требует оформление следующих документов:

1. договор купли-продажи,
2. заказ клиента,
3. расходная накладная,
4. сертификат соответствия и иные документы, подтверждающие соответствие требованиям к качеству,
5. квитанция или чек об оплате.

Работа магазина кожаной одежды регламентируется следующими документами:

1. внутренняя оферта,
2. документы, подтверждающие регистрацию предпринимательской деятельности,
3. заключения Роспотребнадзора,
4. сертификат соответствия,
5. Постановление Правительства РФ от 31.12.2020 № 2463,
6. Закон «О защите прав потребителей»,
7. Закон «О защите персональных данных».

В организационной структуре интернет-магазина кожаной одежды «Heart of Style» можно выделить небольшой ряд отделов, которые играют важнейшую роль для успешной реализации автоматизации всех бизнес-процессов:

1. Менеджер по продажам – лицо, ответственное за реализацию товаров, привлечение новых клиентов, заключение договоров, управление процессом продаж, анализ конкурентов для поддержания конкурентоспособности на рынке, управление клиентской базой, разработку программ лояльности и общение с клиентами после совершения покупки. Важно поддерживать доверительные отношения с покупателями, чтобы обеспечить повторные продажи и удержание клиентов.
2. Логистики занимается планированием, координацией и контролем поставок товаров, организацией складских операций, а также управлением всей транспортной и логистической инфраструктурой компании. Это ключевой элемент эффективной работы предприятия, поскольку именно он обеспечивает бесперебойное движение товаров от поставщика к продавцу и от продавца к потребителям, оптимизируя все этапы поставочной цепочки в работе магазина.
3. Бухгалтер играет ключевую роль в обеспечении правильного бухгалтерского и налогового учёта. Её функции включают в себя не только сбор и систематизацию информации о финансовых операциях организации, но и оформление документов по расчётам с поставщиками и клиентами, а также контроль за финансовой отчётностью. Бухгалтеры также отвечают за подготовку отчётности для государственных органов, расчёт и начисление заработной платы, а также за налоговые выплаты и другие финансовые обязательства, включая оформление необходимых налоговых документов и представление бухгалтерской отчётности.
4. Специалист по складскому учёту контролирует запасы товаров на складе и организует их движение. Его задачи включают планирование и контроль запасов, приемку и отгрузку товаров, а также ведение складского учета. Он обеспечивает, чтобы товары всегда были в наличии в нужном количестве.
5. IT-отдел компании занимается обеспечением бесперебойной работы IT-инфраструктуры. Это включает в себя обслуживание аппаратного и программного обеспечения компании, резервное копирование и хранение баз данных, а также обеспечение кибербезопасности баз данных компании. Кроме того, IT-специалисты занимаются разработкой, доработкой и поддержанием уже существующих приложений, и к тому же предоставляют техническую поддержку и оперативно реагируют на возникающие проблемы с оборудованием.

Полученную организационную структуру можно увидеть на Рисунке 1.

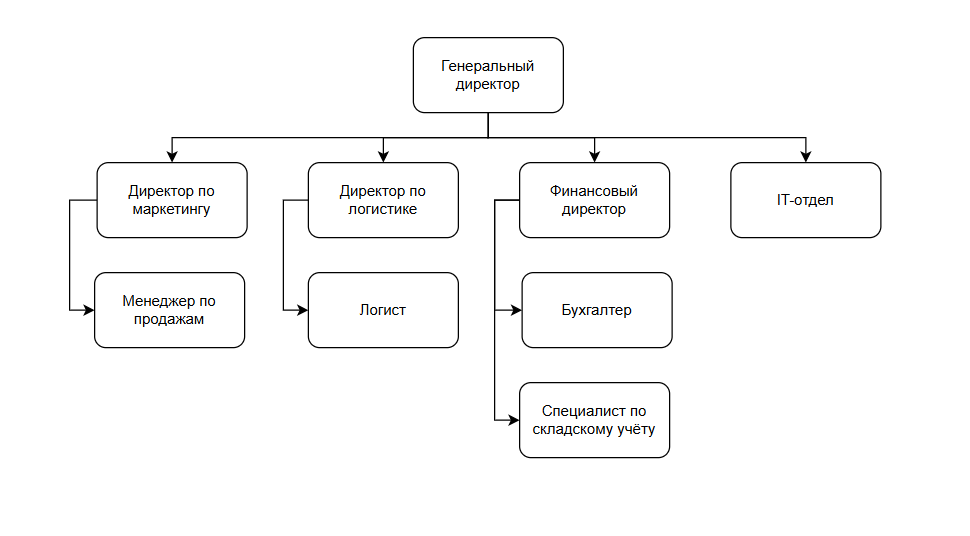


Рисунок − Организационная структура

* 1. **Выбор методологии и технологии концептуального моделирования ИС**

Концептуальное проектирование информационной системы — это этап, на котором разрабатывается обобщённое, абстрактное представление будущей системы. Его основная цель — формирование модели, отражающей ключевые бизнес-процессы, информационные потоки и взаимодействие между объектами системы, без привязки к конкретным программным или техническим средствам. Такая модель служит базисом для дальнейшего логического и физического проектирования, обеспечивая целостность и согласованность всех элементов будущей ИС.

На этапе концептуального проектирования важно выбрать подходящую методологию моделирования, которая обеспечит эффективное описание предметной области, позволит выявить и формализовать бизнес-процессы, определить роли пользователей, потоки информации и основные функции системы.

Структурный подход предполагает иерархическую декомпозицию системы на взаимосвязанные элементы. Такой подход обеспечивает строгое разделение функций, данных и управления, что особенно удобно при описании систем с чёткой логикой. Он основан на последовательном приближении: от общего к частному. Основные методологии структурного анализа:

* IDEF0 — методология функционального моделирования, направлена на выявление иерархии функций системы, определение входов, выходов, управляющих воздействий и механизмов реализации;
* DFD (Data Flow Diagrams) — диаграммы потоков данных, отображающие, как информация проходит через систему, включая процессы, внешние источники и хранилища данных;
* IDEF3 — методология моделирования сценариев и логики выполнения бизнес-процессов во времени.
* Достоинства структурного подхода:
* Наглядность представления процессов;
* Возможность пошаговой декомпозиции;
* Подходит для систем с жёстко заданной логикой.
* Недостатки:
* Плохо масштабируется при высокой сложности;
* Ограничена гибкость при изменениях требований;
* Слабая поддержка повторного использования моделей и компонентов.

Для более наглядного понимания особенностей указанных методологий приведена таблица сравнительной характеристики:

Таблица 1. Сравнительная характеристика методологий структурного моделирования бизнес-процессов:

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий | IDEF0 | DFD | IDEF3 |
| Назначение | Функциональное моделирование процессов | Моделирование потоков данных | Моделирование последовательности и логики бизнес-процессов |
| Основной вопрос | Что делает система? | Как информация проходит через систему? | Как выполняются процессы во времени? |
| Основные элементы | Функции, входы, выходы, механизмы, управления | Процессы, потоки данных, внешние сущности, хранилища данных | Сценарии, объекты, действия, переходы |
| Форма представления | Иерархическая диаграмма блоков | Диаграмма потоков с множеством уровней вложенности | Граф-сценарий выполнения операций во времени |
| Поддержка логики последовательности | Нет (описываются только зависимости и функции) | Частично (возможна логика обработки, но не строго последовательная) | Да (чётко отображается временная логика выполнения операций) |
| Поддержка декомпозиции | Да (многоуровневая детализация функций) | Да (контекстная и детализированная диаграммы) | Частично (возможна детализация сценариев) |
| Уровень формализации | Высокий | Средний | Средний |
| Сценарное описание процессов | Отсутствует | Отсутствует | Присутствует |
| Область применения | Описание функциональной структуры предприятия | Описание и анализ потоков информации в системе | Описание сложных бизнес-сценариев, процессов с временной зависимостью |

В отличие от структурного подхода, объектно-ориентированное моделирование строится вокруг понятий объектов, классов, наследования и взаимодействия между объектами. Основной целью является создание модели, в которой бизнес-объекты и их поведение описаны единым способом. Самой распространённой методологией является Unified Modeling Language (UML), которая включает множество видов диаграмм: классов, прецедентов, последовательностей и др.

Преимущества объектно-ориентированного анализа:

* высокая степень повторного использования;
* гибкость и масштабируемость;
* единый стиль моделирования от концепции до реализации;
* подходит для сложных, изменяющихся систем.
* недостатки:
* требует более высокой квалификации от аналитика;
* может быть избыточен для простых задач;
* менее нагляден на ранних этапах для бизнес-пользователей.
  1. **Анализ предметной области**

Целью данной части работы является изложение результатов анализа предметной области и разработка концептуальной модели проектируемой информационной системы. Концептуальная модель строится на базе анализа бизнес-процессов предприятия и представляет собой описание с использованием естественного языка, таблиц, схем, графов и других средств визуализации.

Анализ предметной области заключается в исследовании требований к системе, выявлении проблем текущих процессов, а также построении модели предприятия, отражающей его структуру, функциональные области и бизнес-процессы. На основе анализа формируется функциональная схема, демонстрирующая:

* Взаимодействие предприятия с внешней средой (поставщики кожаной одежды, контролирующие органы, клиенты, налоговые органы и т.д.).
* Преобразование полученных ресурсов (товаров) в результат деятельности (продажу кожаной одежды).
* Взаимодействие отделов предприятия (закупки, склад, продажи, бухгалтерия и др.) в рамках операционной деятельности.

На схеме блоками обозначаются организации, отделы и сотрудники. Связи между блоками, в том числе функциональные взаимодействия и товарно-денежные потоки, отображаются с помощью стрелок.

Для построения модели интернет-магазина кожаной одежды выделяются следующие основные функциональные области:

* планирование закупок и работа с поставщиками;
* приемка и складирование кожаной одежды;
* продажи и обслуживание клиентов;
* бухгалтерский и налоговый учет;
* управление кадрами;
* финансовое планирование и отчетность.

Каждая функциональная область включает в себя несколько бизнес-процессов. Например, область «Планирование закупок и работа с поставщиками» включает:

* формирование потребностей в товаре;
* поиск и выбор поставщика;
* оформление заказа;
* контроль качества при приемке;
* ведение документации.

Процесс продажи кожаной одежды детализируется до уровня элементарных действий (когерентных операций).

Например, процесс - оформление продажи товара:

* получение заявки клиента через интернет-сайт или в физическом магазине;
* проверка наличия выбранного товара на складе;
* подтверждение заказа клиенту;
* подготовка расходной накладной и сертификатов соответствия;
* подготовка товара к выдаче;
* выдача товара клиенту в магазине;
* обновление данных в системе учета.

Каждое действие обладает четкими границами, результатами, выполняется автономно и целостно. Например:

* действие: пользователь добавляет товар в корзину и оформляет заказ;
* результат: сформирован заказ с указанием адреса получения;
* время: с момента добавления в корзину до отправки формы;
* персонал: клиент.

Таким образом, проектируемая информационная система должна учитывать не только внутренние бизнес-процессы предприятия, но и требования законодательства в части торговли, защиты прав потребителей, персональных данных и ведения финансового учета.

К числу наиболее распространенных методологий моделирования бизнес-процессов относится методология структурного анализа и проектирования SADT и разработанные на его основе стандарт: IDEF (Рекомендацией по стандартизации Р 50.1.028-2001 «МЕТОДОЛОГИЯ ФУНКЦИОНАЛЬНОГО МОДЕЛИРОВАНИЯ»).

В работе необходимо построить контекстную диаграмму (см. рисунок 2), декомпозицию (см. рисунок 3), дерева узлов и создать глоссарий и проанализировать подробно предметную область исследуемого объекта.

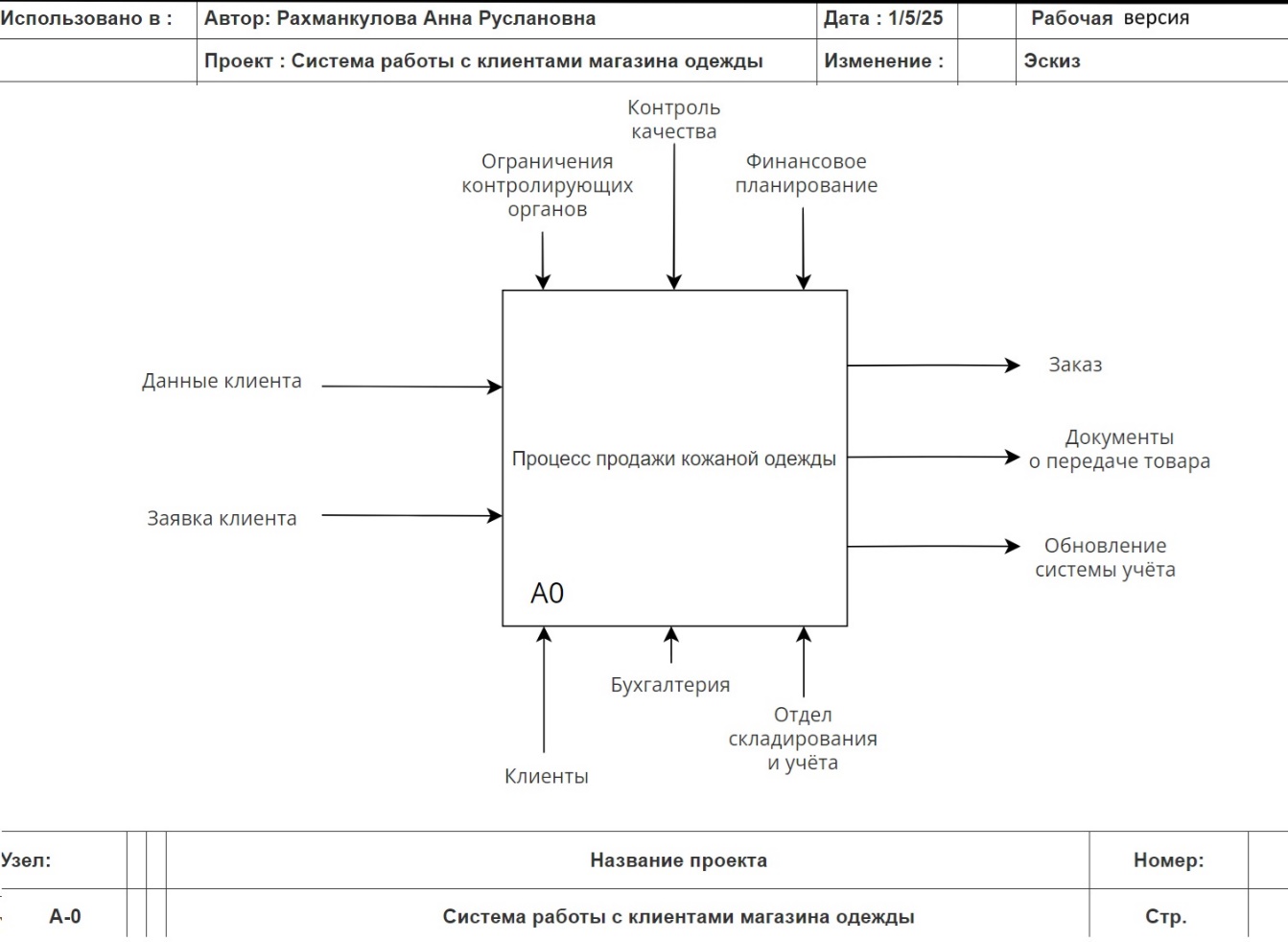


Рисунок 2 – Контектсная диаграмма IDEF0

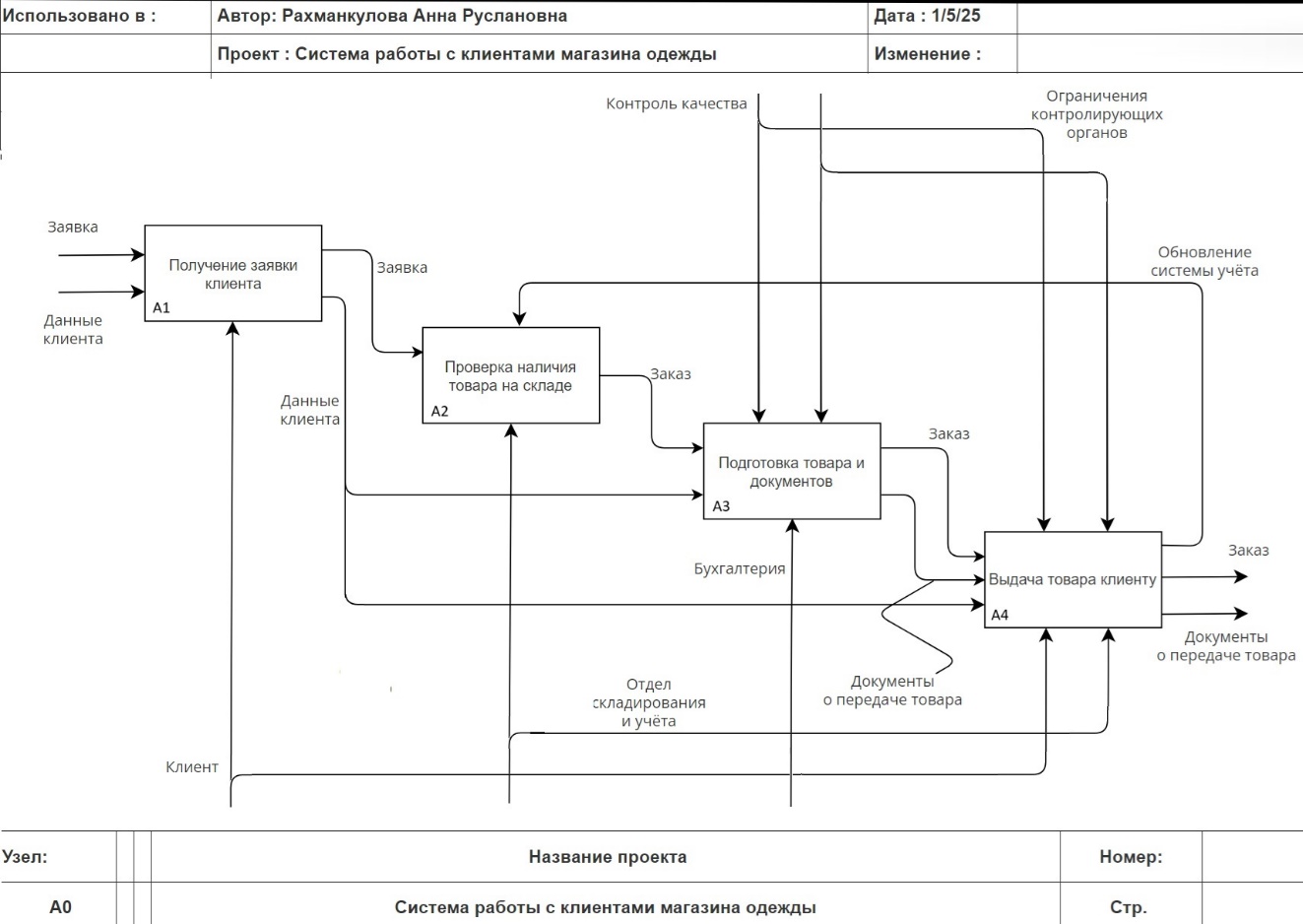


Рисунок 3 – Декомпозиция процесса продажи одежды

1. **РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ**
   1. **Анализ требований и разработка спецификаций**

На этапе проектирования веб-приложения для онлайн-магазина, специализирующегося на продаже кожаной одежды, одной из важнейших задач является всесторонний и глубокий анализ требований к системе с последующей их формализацией в виде технической и пользовательской спецификаций. Этот этап играет ключевую роль, поскольку именно на его основе строится вся дальнейшая работа — от выбора архитектуры и технологий до разработки, тестирования и внедрения готового продукта.

Главная цель этого этапа — максимально точно определить, какие именно задачи должна решать система, каким функционалом она должна обладать, кто будет её основными пользователями и в каких условиях она будет эксплуатироваться. Такой подход позволяет четко определить границы и масштабы проекта, минимизировать риски возникновения противоречий и недочётов, а также своевременно выявить потенциальные проблемные зоны, что значительно упрощает последующую разработку и снижает вероятность изменений на более поздних стадиях разработки.

Первым шагом на этом этапе является формулировка требований к системе, которые условно делятся на функциональные и нефункциональные.

Функциональные требования описывают, что система должна делать, то есть перечень функций, которые должны быть реализованы. Это конкретные действия и возможности, которые предоставляет система пользователю:

* Регистрация и авторизация: система должна предоставлять возможность регистрации нового пользователя, а также входа в систему с сохранённым логином и паролем.
* Просмотр страницы каталога: пользователь может ознакомиться с полным ассортиментом товаров, размерной сеткой и колористикой товара.
* Просмотр информации о товаре: каждый товар должен иметь подробное описание, включая доступные размеры, цветовые вариации, цену и фото с разных ракурсов.
* Просмотр страницы «О нас»: пользователь может ознакомиться с информацией о компании, её миссии, производстве и т.д.
* Добавление в корзину: пользователи должны иметь возможность добавлять товары в корзину, указывая размер. Корзина должна отображать текущие товары, с указанием выбранного цвета и размера, с возможностью удаления позиций.
* Оформление заказа: покупатели выбирают адрес доставки. Заказ должен быть подтвержден и сохранен в базе данных.
* Просмотр личных данных: пользователи должны иметь возможность просмотреть свои данные, введённые ими при регистрации
* Просмотр истории заказов: пользователи должны иметь возможность просматривать историю своих заказов, в том числе просматривать адрес получения, какие товары были заказаны, их характеристики и статус заказа
* Администрирование товарами: администратор должен иметь возможность добавлять новые товары, а также изменять информацию о товарах, уже сохранённых в базе данных
* Управление заказами и их статусами: администратор должен иметь возможность просматривать все заказы, с использованием фильтрации по статусам. Так же изменять статус заказа при переходе заказа на новый уровень

Нефункциональные требования описывают характеристики системы, которые обеспечивают комфорт и безопасность её использования. Среди них:

* Доступность: Приложение должно быть доступно 24/7 без простоев. Это позволит клиентам совершать покупки в любое время.
* Производительность: Быстрая загрузка страниц, поддержка большого количества пользователей одновременно.
* Безопасность: Хранение паролей в зашифрованном виде, защита от CSRF/XSS атак, авторизация через токены.
* Юзабилити: интуитивно понятный интерфейс, адаптивная верстка для мобильных устройств.
* Масштабируемость: Возможность расширения функционала без полной переработки архитектуры.
* Надежность и отказоустойчивость: Работа системы без сбоев даже при высокой нагрузке.
* Логирование и мониторинг: Возможность отслеживания действий пользователей и администраторов для отладки и аудита.

Действующие лица:

1. Гость – неавторизованный пользователь, может только просматривать информацию
2. Покупатель – авторизованный пользователь, желающий приобрести кожаную одежду.
3. Администратор – авторизованный пользователь, который имеет доступ к управлению товарами и заказами

Варианты использования:

1. Регистрация пользователя:

* действующее лицо: гость,
* описание:

1. Гость переходит на форму регистрации,
2. Вводит логин, пароль, e-mail, номер телефона и дату рождения,
3. Система сохраняет данные и перенаправляет на личный кабинет пользователя.

* Предпосылки: пользователь не авторизован
* Постусловие: создан личный аккаунт пользователя

1. Авторизация пользователя:

* Действующее лицо: гость
* Описание:

1. Гость вводит логин и пароль,
2. Система проверяет данные и авторизует пользователя.

* Предпосылки: пользователь зарегистрирован
* Постусловие: пользователь получает доступ к функциям аккаунта.

1. Просмотр каталога:

* Действующее лицо: гость, покупатель
* Описание:

1. Пользователь открывает каталог товаров,
2. Имеет доступ к размерной сетке и цветам.

* Предпосылки: пользователь находится на сайте
* Постусловие: пользователь видит полный ассортимент.

1. Просмотр информации о товаре:

* Действующее лицо: гость, покупатель
* Описание:

1. Пользователь выбирает товар,
2. Система отображает детальную карточку (размер, цвет, фото, цена).

* Предпосылки: пользователь выбрал товар
* Постусловие: пользователь ознакомлен с характеристиками.

1. Просмотр страницы «О нас»:

* Действующее лицо: гость, покупатель
* Описание:

1. Пользователь или гость переходит по ссылке «О нас» из любой страницы веб-приложения,
2. Система загружает и отображает страницу с информацией о компании.

* Предпосылки: пользователь находится на сайте
* Постусловие: пользователь ознакомился с информацией о компании

1. Добавление товара в корзину:

* Действующее лицо: покупатель
* Описание:

1. Пользователь указывает размер и может указать цвет, и нажимает «Добавить в корзину»,
2. Система сохраняет товар в корзине с указанными параметрами.

* Предпосылки: пользователь авторизован и выбрал товар
* Постусловие: товар добавлен в корзину.

1. Оформление заказа:

* Действующее лицо: покупатель
* Описание:

1. Пользователь открывает корзину и нажимает «Оформить заказ»,
2. Указывает адрес доставки и способ оплаты,
3. Система сохраняет заказ в БД и присваивает статус.

* Предпосылки: в корзине есть хотя бы 1 товар
* Постусловие: заказ оформлен.

1. Просмотр личных данных:

* Действующее лицо: покупатель
* Описание:

1. Пользователь переходит в личный кабинет,
2. Система отображает информацию, введённую при регистрации.

* Предпосылки: пользователь авторизован
* Постусловие: пользователь видит свои данные.

1. Просмотр истории заказов:

* Действующее лицо: покупатель
* Описание:

1. Пользователь переходит в личный кабинет,
2. Отображаются все предыдущие заказы: товары, параметры, дата, адрес, статус

* Предпосылки: пользователь авторизован
* Постусловие: пользователь видит свои заказы, оформленные ранее.

1. Администрирование товарами:

* Действующее лицо: администратор
* Описание:

1. Админ входит в панель управления товарами,
2. Может добавлять новые товары, редактировать существующие (цена, описание, фото и т.д.).

* Предпосылки: пользователь авторизован как администратор
* Постусловие: каталог товаров актуализирован.

1. Управление заказами и их статусами:

* Действующее лицо: покупатель
* Описание:

1. Админ входит в панель управления заказами,
2. Админ просматривает список заказов,
3. Может фильтровать по статусу,
4. Меняет статус заказа вручную.

* Предпосылки: пользователь авторизован как администратор, покупатели оформили хотя бы один заказ
* Постусловие: обновлены статусы заказов.

На основе приведённого анализа функциональных и нефункциональных требований, а также подробного описания вариантов использования для всех категорий пользователей — гостей, зарегистрированных пользователей и администратора — была построена диаграмма вариантов использования (Use Case Diagram) (рисунок 4). Эта диаграмма наглядно отражает взаимодействие между участниками системы и ключевыми функциями веб-приложения интернет-магазина кожаной одежды.

Диаграмма помогает визуализировать структуру системы, выделить основные сценарии поведения пользователей и их связи с функциональностью приложения. Она является важным инструментом проектирования, способствующим лучшему пониманию архитектуры и будущей реализации веб-приложения.

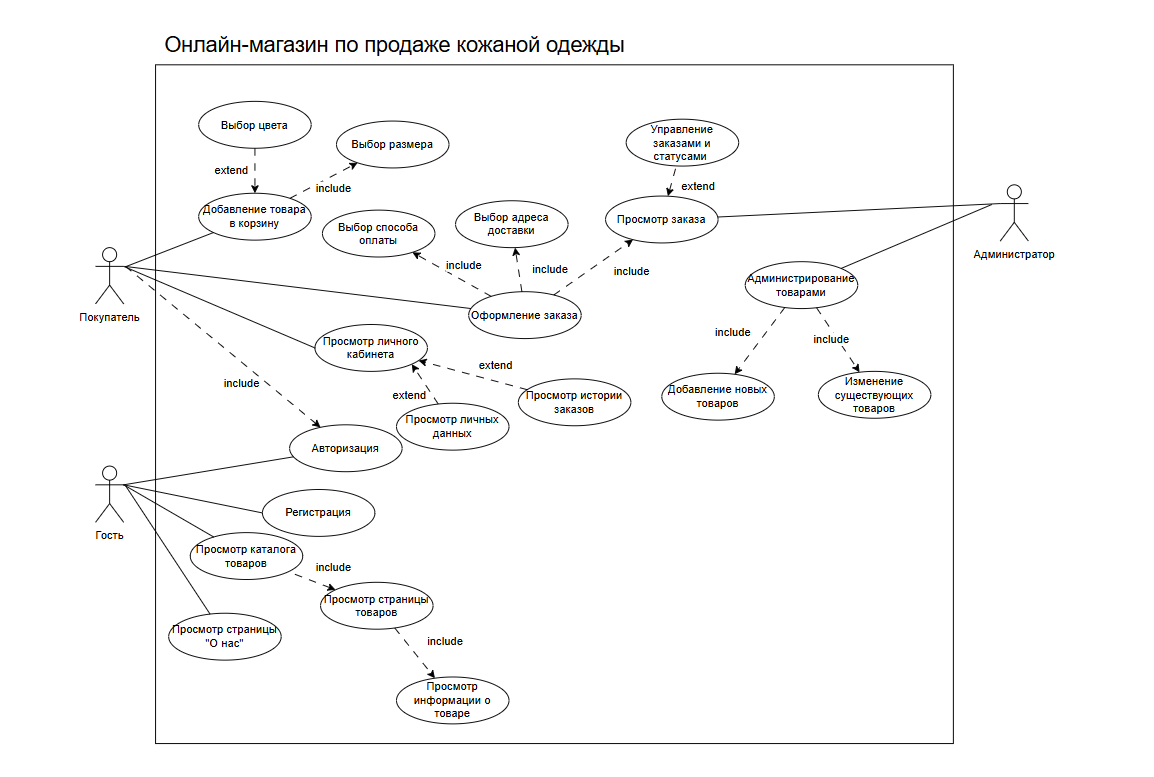


Рисунок 4 − Диаграмма вариантов использования

* 1. Проектирование и разработка дизайна интерфейса

Пользовательский интерфейс (UI) представляет собой важнейший канал коммуникации между пользователем и компьютерной системой. Именно через него происходит взаимодействие человека с веб-приложением, что в конечном итоге определяет удобство и эффективность работы пользователя с системой. Веб-интерфейс — это не просто набор визуальных элементов и кнопок, а комплексное средство, которое должно обеспечить интуитивно понятное, быстрое и комфортное выполнение пользовательских задач.

**Прозрачность интерфейса** — один из главных критериев его качества. Идеальный пользовательский интерфейс — это такой интерфейс, который пользователь практически не замечает, поскольку он органично вписывается в процесс решения его задач. Он не отвлекает, не требует долгого обучения или частых раздумий, а помогает сосредоточиться на главном — получении информации или выполнении нужных действий. В этом смысле интерфейс становится «невидимым», позволяя пользователю «смотреть сквозь него» на содержимое и функции, которые он хочет использовать.

В рамках данного проекта была реализована максимально прозрачная и ненавязчивая структура интерфейса: элементы управления (меню, кнопки, карточки товаров) лаконичны и логичны, выполнены в едином стиле, что снижает когнитивную нагрузку на пользователя. Благодаря этому пользователь интуитивно понимает, как взаимодействовать с системой, не отвлекаясь на само оформление.

Особенно важно учитывать, что посетители веб-сайта чаще всего приходят с конкретной целью — будь то поиск товара, получение информации или оформление заказа. Соответственно, интерфейс должен обеспечить максимально быстрый и удобный доступ к нужным разделам, свести к минимуму лишние действия и максимально упростить навигацию. Здесь на первый план выходят такие аспекты, как логичная структура страниц, понятные и легко различимые элементы управления, а также визуальная иерархия, которая помогает пользователю ориентироваться и не теряться.

В реализованном проекте особое внимание уделено пользовательским сценариям. Основные пути — просмотр ассортимента, добавление товаров в корзину, оформление заказа, просмотр личного кабинета — оформлены так, чтобы каждый шаг был очевидным. Меню расположено в привычных местах, фильтры работают мгновенно, важные действия (например, «Добавить в корзину») визуально выделены и доступны в один клик.

Эргономика интерфейса — ещё один важный фактор, который влияет на удовлетворённость пользователей. Она включает в себя продуманное размещение элементов, простоту навигации, логичность структуры информации и скорость отклика системы. Важно минимизировать количество шагов, необходимых для выполнения ключевых действий, таких как поиск товара, добавление его в корзину или оформление заказа. Интерфейс должен быть доступен как опытным пользователям, так и новичкам, обеспечивая при этом возможность быстрого обучения и запоминания последовательности операций.

В проекте были учтены принципы эргономики: все кнопки, формы и ссылки находятся в доступных для восприятия местах, интерфейс быстро откликается на действия пользователя, а оформление страниц позволяет легко находить нужную информацию. При этом элементы не перегружают экран и не требуют точной моторики, что особенно важно при работе с мобильных устройств.

Не менее важно при разработке учитывать адаптивность интерфейса — способность корректно отображаться и работать на самых разных устройствах: от стационарных компьютеров до смартфонов и планшетов. В современных условиях, когда значительная часть трафика приходится на мобильные устройства, адаптивность становится необходимым условием успешного веб-приложения.

В дипломном проекте реализована полная адаптивность интерфейса: все страницы сайта корректно отображаются на экранах различных размеров, элементы интерфейса автоматически перестраиваются под размер устройства, обеспечивая удобство использования. Благодаря этому сайт одинаково удобен как на ноутбуке, так и на смартфоне.

При разработке дизайна веб-приложения компании «Heart of Style» ключевым моментом является выбор тематики. Основной аудиторией данного проекта являются девушки всех возрастов, поэтому особое внимание уделяется цветовой гамме. Известно, что яркие цвета больше привлекают внимание женской аудитории, поэтому в дизайне веб-сайта были использованы оранжевые цвета. Однако, для избегания излишней вычурности и ощущения излишней яркости, был выбран более спокойный и нежный оттенок этого цвета. Важно помнить, что избыток контрастных цветов может создать впечатление закрытости и даже вульгарности, поэтому был добавлен серый цвет, который придает веб-приложению баланс и гармонию. Такой подход позволяет сделать сайт ярким и привлекательным, но в то же время сохранить его спокойствие и комфорт в использовании для пользователей.

Для обеспечения единства стиля между логотипом компании «Heart of Style» и веб-приложением, оранжевый цвет того же оттенка (#F0C57B), что и в логотипе, использован на сайте в виде небольших акцентов. Эти вставки остаются заметными на протяжении всего пребывания пользователя на веб-приложении, что создает ощущение целостности и стабильности визуального восприятия.

Кофейно-серый оттенок (#B1A59C) изначально был выбран для согласованности изображения, встречающего на главной странице сайта, а после для согласованности страниц между собой и для приятного просмотра информации в веб-приложении.

Выбранные цвета для темы сайта можно увидеть на Рисунке 5.



Рисунок 5 – Выбранные цвета для темы сайта

Для данного интернет-магазина были выбраны не стандартные шрифты, чтобы подчеркнуть его уникальность.

Alegreya Sans SC — это шрифт без засечек, который отличается своим чистым и современным дизайном. Его удобная графическая структура делает его идеальным для воплощения в жизнь различных дизайнерских идей и концепций. Alegreya Sans SC обладает высокой читаемостью и прекрасно подходит для использования в текстах любого размера. В нем представлено четырнадцать начертаний, хотя в данном случае было выбрано начертание Regular, в силу его простоты и наилучшего сочетания с тематикой проекта.

В дипломном проекте этот уникальный шрифт нашел свое применение в наименованиях страниц, размещенных в верхней части каждой страницы, в описаниях каждого товара, а также в содержании страниц. Эти элементы считаются базовыми для любого веб-приложения, именно поэтому было решено использовать именно шрифт Alegreya Sans SC — для избежания вульгарности и придания проекту стильного и запоминающегося пользователям облика.

Шрифт Alegreya Sans SC можно увидеть на Рисунке 6.

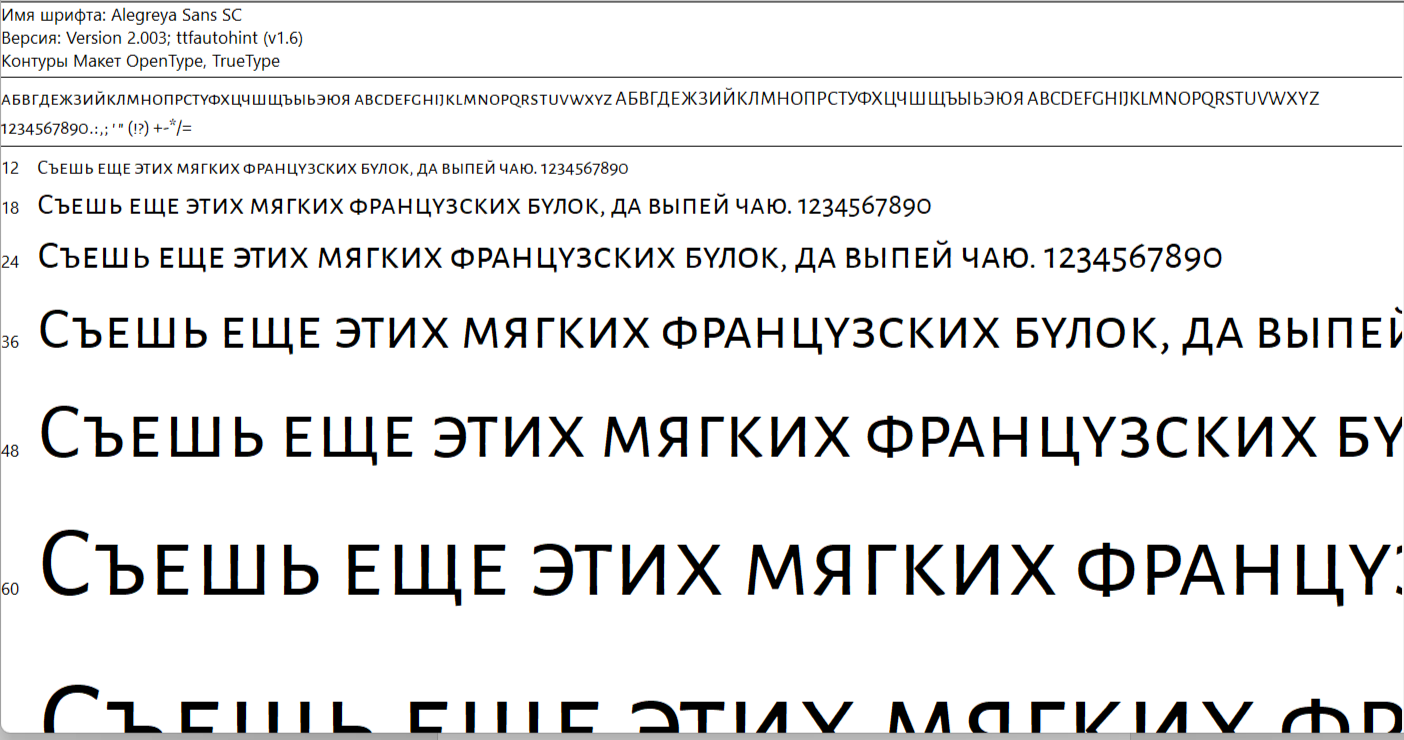


Рисунок 6 – Шрифт Alegreya Sans SC

Playfair Display SC — это шрифт с засечками, который характеризуется элегантным и стильным дизайном. Он идеально подходит для использования в заголовках, логотипах и других дизайнерских элементах, где требуется выразительный и утонченный шрифт. Playfair Display SC имеет прекрасное графическое построение и красивые пропорции символов, что делает текст читаемым и привлекательным. Этот шрифт обладает аристократическим видом и придает тексту роскошный и изысканный облик. Он обладает семью начертаниями, но в данной работе использовалось начертание Regular.

В данном проекте этот шрифт нашёл свое назначение в логотипе компании «Heart of Style», а также в названиях товара и заголовках новостей. Этот шрифт изящен и красив, он подходит для определения уникальности компании «Heart of Style». Он идеально подходит для информации, напрямую связанную с продукцией интернет-магазина.

Шрифт Playfair Display SC можно увидеть на Рисунке 7.

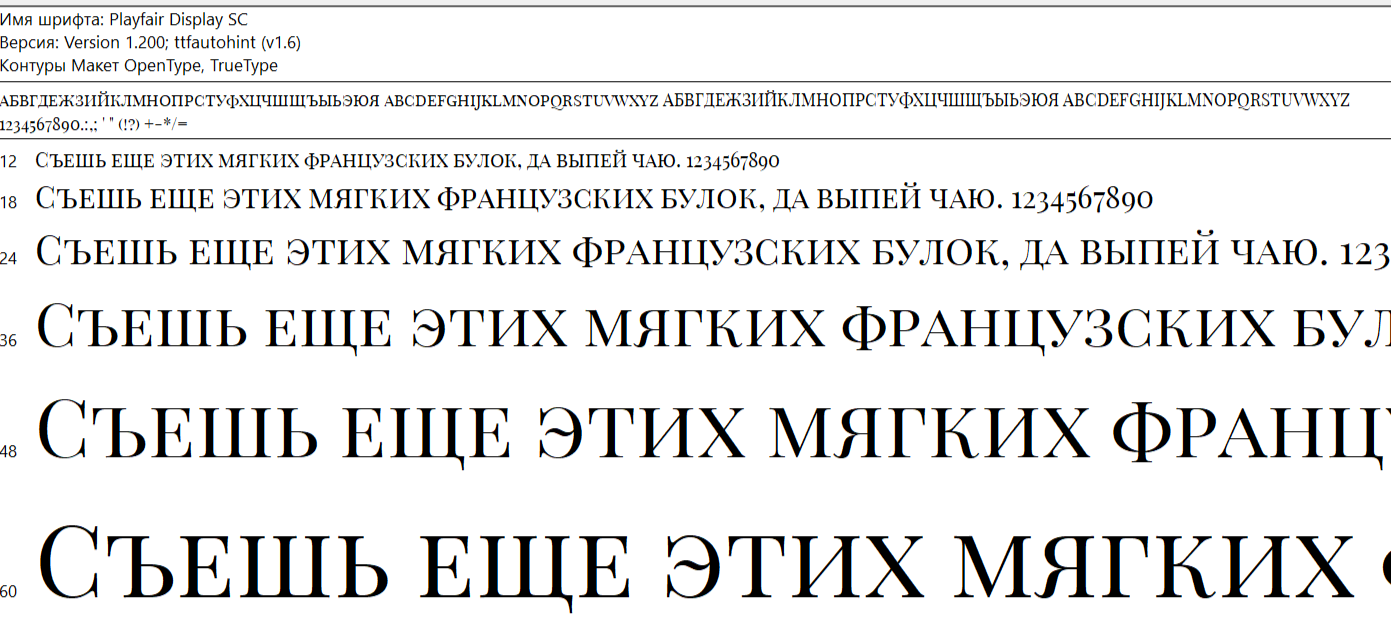


Рисунок 7 – Шрифт Playfair Display SC

Такой подход к выбору шрифтов помогает подчеркнуть уникальность компании «Heart of Style», а также вызвать у гостей интернет-магазина заинтересованность к бренду.

* 1. Разработка кода веб-приложения и программирование

**Выбор технического обеспечения и средств разработки**

Разработка веб-приложения требует тщательного выбора языков программирования, инструментов, технологий и платформ, которые будут использоваться как на этапе создания, так и при дальнейшей поддержке и сопровождении проекта. Выбор каждого компонента основывался на принципах надёжности, удобства разработки, совместимости и эффективности в рамках поставленных задач.

Основным языком программирования для разработки веб-приложения был выбран Python — современный, высокоуровневый и универсальный язык, отличающийся лаконичным синтаксисом, читаемостью и широкими возможностями. Несмотря на то, что Python не является сугубо «веб-языком», он активно используется в веб-разработке благодаря мощным фреймворкам, таким как Django, Flask и FastAPI. Развитая экосистема, обширная стандартная библиотека и активное сообщество делают Python востребованным инструментом для создания надёжных, масштабируемых и легко поддерживаемых веб-систем.

Выбор Python в качестве основного языка обусловлен рядом преимуществ по сравнению с альтернативами. В отличие от PHP, ориентированного исключительно на веб-разработку, Python обладает более широкой сферой применения — от анализа данных до искусственного интеллекта. Это делает его удобным не только для реализации серверной части, но и для последующей интеграции с другими системами. По сравнению с Java, Python позволяет быстрее разрабатывать и тестировать функциональность благодаря лаконичному синтаксису и меньшему количеству шаблонного кода. В отличие от JavaScript, традиционно применяемого для клиентской части, Python предоставляет больше возможностей для построения сложной бизнес-логики и работы с базами данных на стороне сервера.

Таким образом, Python сочетает в себе простоту, гибкость и мощный инструментарий, что делает его оптимальным выбором для создания современного веб-приложения с учётом требований к надёжности, скорости разработки и лёгкости поддержки.

Ключевым инструментом разработки стал фреймворк Django, предоставляющий разработчику все необходимые механизмы для построения модели, отображения и управления логикой приложения. Благодаря архитектуре MVT (Model–View–Template), реализация проекта была чётко структурирована по слоям, что упростило поддержку и расширение функционала на всех этапах разработки.

Для наглядного понимания взаимодействия ключевых компонентов веб-приложения, построенного на Django, диаграмма архитектуры MVT (см. рисунок 8). Она демонстрирует, как пользователь взаимодействует с приложением через представление (View), как представление обрабатывает данные с помощью модели (Model) и, наконец, как шаблоны (Template) формируют итоговую веб-страницу.

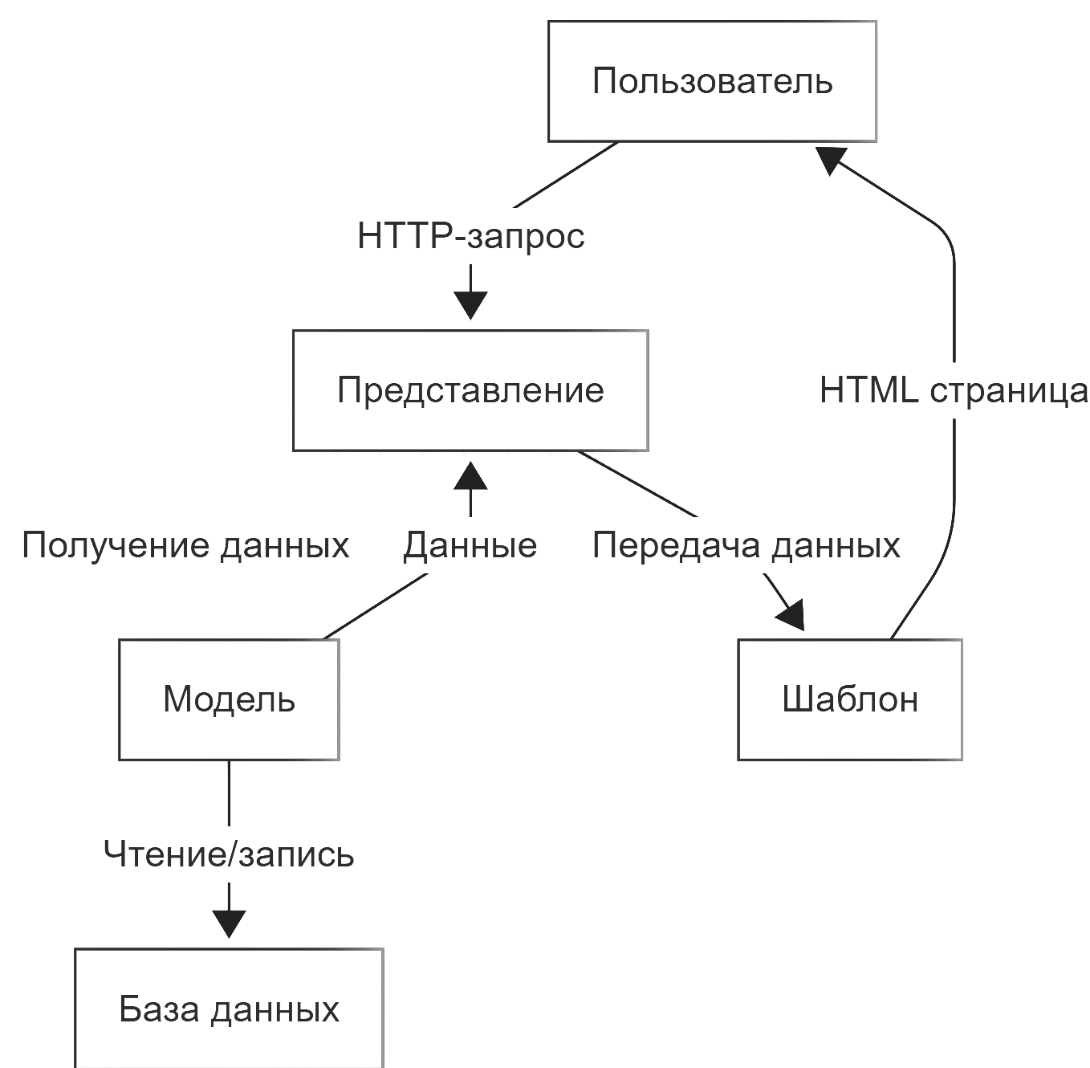


Рисунок 8 − диаграмма архитектуры MVT

Django обладает встроенной системой маршрутизации, мощной ORM (Object-Relational Mapping) для взаимодействия с базой данных, гибкой системой шаблонов, административной панелью «из коробки», а также встроенными средствами аутентификации и авторизации пользователей. Все это позволило сосредоточиться на реализации бизнес-логики, сводя к минимуму необходимость ручной настройки базовой инфраструктуры проекта.

Как уже упоминалось ранее, экосистема Python включает и другие популярные фреймворки для веб-разработки, такие как Flask и FastAPI. Flask — минималистичный микрофреймворк, предоставляющий разработчику большую свободу в построении архитектуры, но требующий ручной реализации многих компонентов, включая маршрутизацию, ORM и авторизацию. FastAPI, в свою очередь, ориентирован на разработку высокопроизводительных REST API и подходит для проектов с акцентом на скорость, асинхронную обработку и взаимодействие с внешними сервисами. Однако в случае создания функционального веб-приложения с полноценной серверной логикой, административным интерфейсом, системой учёта заказов и визуальным управлением контентом, именно Django оказался наиболее подходящим решением.

Выбор Django особенно оправдан при разработке MVP (Minimum Viable Product) — минимально жизнеспособной версии продукта. Django позволяет быстро получить рабочий прототип с полным стеком базовых функций: регистрацией пользователей, управлением данными, отображением товаров, обработкой заказов. Благодаря автоматизированной административной панели можно без лишней разработки вести управление содержимым сайта, что особенно важно для малого и среднего бизнеса, не располагающего крупной командой разработчиков. Надёжность, проверенная временем архитектура, активное сообщество и обилие документации делают Django устойчивой основой для запуска и масштабирования коммерческих проектов.

Одним из важнейших преимуществ фреймворка Django является высокий уровень встроенной безопасности, обеспечивающий надёжную защиту веб-приложений от наиболее распространённых уязвимостей. Безопасность в Django реализована на уровне архитектуры самого фреймворка и применяется по умолчанию, что позволяет разрабатывать устойчивые системы без необходимости ручной настройки большинства защитных механизмов. Такой подход особенно ценен при создании приложений, обрабатывающих персональные или финансовые данные, поскольку снижает риск уязвимостей, связанных с человеческим фактором или незнанием разработчиком тонкостей защиты.

Во-первых, Django обеспечивает защиту от SQL-инъекций благодаря использованию встроенного механизма ORM (Object-Relational Mapping). Все обращения к базе данных происходят через высокоуровневые Python-интерфейсы, и разработчику не требуется вручную писать SQL-запросы. Это исключает возможность внедрения вредоносного кода, так как все параметры автоматически экранируются и проверяются.

Кроме того, встроенные средства защиты от XSS (межсайтового скриптинга) работают по принципу автоматической обработки переменных, передаваемых в шаблоны. Все данные, которые отображаются в HTML-документах через систему шаблонов Django, по умолчанию проходят специальную обработку: потенциально опасные символы, такие как <, >, &, ', " и другие, заменяются на их безопасные HTML-сущности. Таким образом, даже если пользователь попытается ввести вредоносный JavaScript-код, он будет интерпретирован как обычный текст, поскольку все переменные в шаблонах обрабатываются с удалением потенциально опасного содержимого и выводятся в безопасной форме.

Django также включает встроенную защиту от CSRF-атак (подделки межсайтовых запросов). Каждый HTML-форме автоматически присваивается уникальный токен, проверяемый при отправке данных на сервер. Это предотвращает возможность выполнения действий от имени пользователя через подставные формы, размещённые на внешних сайтах.

Важной составляющей системы безопасности Django являются механизмы управления пользовательскими сессиями. Фреймворк использует безопасные HTTP-only куки, которые недоступны для JavaScript и, соответственно, не могут быть похищены через XSS-атаки. Также возможно использование дополнительного уровня защиты в виде флагов Secure, SameSite и автоматического завершения сессий при выходе пользователя.

Кроме перечисленных механизмов, Django содержит встроенные средства предотвращения атак через заголовки HTTP, таких как Clickjacking, за счёт настройки заголовков X-Content-Type-Options, X-Frame-Options, X-XSS-Protection и других. Эти заголовки можно легко активировать с помощью встроенных настроек или промежуточных слоёв (middleware).

Также важной особенностью Django является система миграций базы данных, позволяющая управлять изменениями в структуре моделей без потери данных. Разработчик может вносить изменения в модели Python-классов, а Django автоматически создаёт миграционные файлы, отражающие эти изменения в базе данных. Это особенно полезно в условиях активной разработки, когда структура данных может неоднократно корректироваться.

Наконец, Django хорошо масштабируется: архитектура проекта легко дополняется новыми модулями, приложениями и API-интерфейсами. Возможность разделения проекта на независимые приложения делает его пригодным не только для MVP, но и для дальнейшего роста до уровня крупного портала или интернет-магазина с множеством функций и пользователей.

Таким образом, выбор Django был обусловлен его универсальностью, функциональной полнотой, встроенной безопасностью, поддержкой миграций и возможностями масштабирования, что в совокупности делает его оптимальным решением для разработки современных надёжных веб-приложений.

В качестве основы для интерфейса используются HTML-шаблоны в сочетании с CSS, что позволяет создавать структурированные, удобные и визуально привлекательные веб-страницы. HTML отвечает за построение и разметку содержимого, а CSS — за оформление и стилизацию элементов страницы. Встроенный в Django шаблонизатор даёт возможность динамически подставлять данные из базы, реализовывать навигацию и повторно использовать общие компоненты интерфейса, такие как шапка, футер или меню.

Для придания внешнему виду современного и эстетичного дизайна применяется CSS, обеспечивающий стилизацию карточек товаров, кнопок, форм, а также адаптивность макета под различные устройства. При необходимости дополнительно используется JavaScript, который позволяет реализовывать интерактивные элементы, такие как динамическая валидация форм, всплывающие окна или обмен данными с сервером без перезагрузки страницы.

Для разработки проекта использовались две интегрированные среды разработки. Основной средой являлся PyCharm, который предоставляет полный набор инструментов для работы с языком Python и фреймворком Django, включая поддержку шаблонов, интеграцию с системой контроля версий Git, встроенный терминал и отладчик, а также удобную навигацию по структуре проекта. Вспомогательной средой был выбран Visual Studio Code, предназначенный для быстрого и удобного редактирования файлов фронтенда — HTML, CSS и JavaScript. Такой подход позволял эффективно разделить работу над backend- и frontend-частями приложения.

Для хранения данных была выбрана встроенная база SQLite, которая поставляется вместе с Django и идеально подходит для этапа разработки благодаря простой настройке и удобству использования. SQLite хранит всю базу в одном файле, не требуя установки и обслуживания отдельного сервера, что ускоряет процесс разработки и тестирования. При необходимости дальнейшего масштабирования проекта предусмотрен переход на более мощные системы управления базами данных, такие как PostgreSQL или MySQL.

После завершения разработки приложение было опубликовано на платформе PythonAnywhere — облачном хостинге, специально созданном для Python-проектов. Этот сервис позволяет легко и быстро размещать веб-приложения с поддержкой Django, предоставляя удобный графический интерфейс для настройки, доступ к консоли и возможность работы с файловой системой прямо через браузер. Выбор PythonAnywhere позволил обеспечить стабильную и доступную работу сайта без дополнительных затрат на аренду серверов.

Для управления исходным кодом и совместной работы использовалась система контроля версий Git, которая обеспечивает надежное хранение изменений, возможность отката к предыдущим версиям и удобное командное взаимодействие.

**Описание структуры базы данных**

Логическая модель предметной области представляет собой совокупность сущностей и связей между ними, отражающую структуру данных, необходимых для функционирования информационной системы интернет-магазина кожаной одежды.

Каждая сущность в данной модели описывает определённый класс объектов, а связи между сущностями определяют правила их взаимодействия. Для идентификации экземпляров сущностей используются уникальные идентификаторы (ключи), обеспечивающие однозначность записи в базе данных.

Основные сущности логической модели:

* пользователь (User / UserProfile).

Представляет зарегистрированного клиента интернет-магазина. Расширенная информация хранится в отдельной модели UserProfile, связанной с встроенной моделью User через связь "один к одному". Основные атрибуты:

* логин и пароль (модель User),
* фамилия, имя, отчество,
* телефон, электронная почта,
* дата рождения.

Товар (Product). Сущность, представляющая собой конкретную единицу кожаной продукции (куртка, юбка, пиджак и др.). Основные атрибуты:

* наименование,
* описание,
* цена,
* дата добавления.

Цвет (Color). Модель описывает доступные цветовые вариации, эта таблица вынесена отдельно для большей гибкости веб-приложения. Основные атрибуты:

* наименование,
* hex-код.

Размер (Size). Сущность определяет возможные размеры продукции, таблица вынесена отдельно так же для большей гибкости веб-приложения. Основные атрибуты:

* наименование,
* вариация товара (ProductVariation).

Модель, описывающая конкретную комбинацию товара с определённым цветом и доступными размерами. Позволяет отображать наличие и особенности отдельных товарных позиций.

Изображения товара (ProductImage). Хранит изображения товара в зависимости от цвета. Связана с конкретным товаром и цветом.

Пункт выдачи (PickupPoint). Содержит адреса пунктов самовывоза, куда может быть доставлен заказ.

Заказ (Order). Сущность, описывающая оформленный заказ пользователя. Содержит данные о типе оплаты, статусе заказа, пользователе и пункте выдачи. Атрибуты:

* тип оплаты (наличные, карта),
* статус заказа (новый, в обработке, принят и т.д.),
* дата оформления,
* причина отмены (если заказ отменён).

Связан с множеством товаров через промежуточную модель OrderItem.

Позиция заказа (OrderItem). Определяет отдельный товар внутри заказа с указанием цвета, размера, количества и цены на момент оформления.

Корзина (Cart). Представляет текущую корзину пользователя, в которой хранятся выбранные товары до момента оформления.

Позиции в корзине (CartItem). Описывает товары, находящиеся в корзине. Учитывает выбранный цвет, размер и количество.

Взаимосвязи между сущностями:

1. Один пользователь связан с одной корзиной (один к одному).
2. Один пользователь может оформить множество заказов (один ко многим).
3. Один заказ включает множество позиций (многие ко многим через OrderItem).
4. Каждый товар может иметь несколько вариаций (один ко многим).
5. Каждая вариация — это связь между товаром, цветом и набором размеров.
6. Товар может иметь много изображений, каждое из которых связано с конкретным цветом.
7. Заказ связан с одним пунктом выдачи.
8. Корзина пользователя содержит множество позиций, каждая из которых указывает на товар, его цвет, размер и количество.

Таким образом, логическая модель чётко отражает бизнес-процессы интернет-магазина кожаной одежды: от выбора и отображения товара до оформления и управления заказами. Она является основой для построения физической структуры базы данных и реализации функционала в системе на основе Django.

Для наглядного представления структуры предметной области интернет-магазина кожаной одежды логическая модель данных представлена в виде диаграммы "сущность–связь" (Entity–Relationship diagram, ER-диаграммы).  
Данная диаграмма отражает основные сущности системы, их атрибуты, а также типы связей между ними (один к одному, один ко многим, многие ко многим).

ER-диаграмма позволяет ясно продемонстрировать архитектуру базы данных, выявить логические взаимосвязи между таблицами и служит фундаментом для построения физической модели данных в реляционной СУБД. Связи, реализуемые средствами Django (например, ForeignKey, OneToOneField и ManyToManyField), также наглядно представлены, включая промежуточные таблицы и составные связи, обеспечивающие целостность и гибкость системы (рисунок 9).

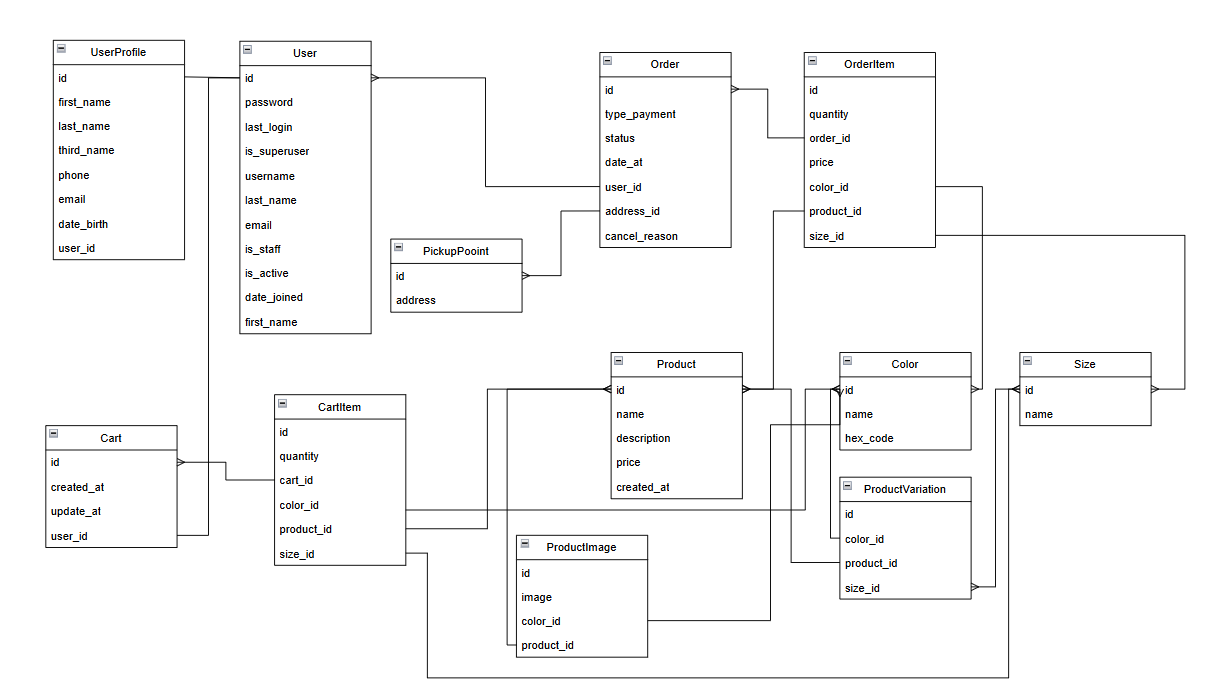


Рисунок 9 − Логическая схема данных приложения

**Программирование веб-приложения**

HTML — язык разметки, который используется для создания структуры веб-страницы. Он играет главнейшую роль в создании каркаса всех элементов на сайте — заголовков, параграфов, картинок, ссылок и других деталей контента.

Преимущества HTML:

* простота: легок в изучении даже для тех, кто начинает с нуля.
* Универсальность: поддерживается всеми браузерами и платформами.
* Готовая структура: позволяет концентрироваться на содержимом, не отвлекаясь на технические детали верстки.
* Готов к дополнениям: отлично сочетается с другими технологиями — CSS, JavaScript и серверными языками.

CSS — язык каскадных таблиц стилей, позволяющий настраивать внешний вид веб-страницы. Он отвечает за цвета, размеры элементов, выравнивание и другие аспекты дизайна.

Преимущества CSS:

* гибкость: позволяет полностью контролировать внешний вид всех элементов на сайте.
* Разделение задач: вынос дизайна из содержимого помогает концентрироваться на структуре и логике.
* Универсальность: одинаково воспроизводится на всех браузерах.
* Оптимизация: сокращает объем кода и повышает скорость загрузки страниц.

JavaScript — язык программирования, позволяющий делать веб-страницу динамичной и интерактивной. Он позволяет людям напрямую с ней взаимодействовать, вызывает изменения на сцене и отвечает на действия пользователей.

Преимущества JavaScript:

* Динамика: позволяет внедрять анимации, слайды, подсказки и диалоги без перезапроса всех данных с сервера.
* Универсальность: поддерживается всеми браузерами и отлично сочетается с другими веб-технологями.
* Готовая среда: может работать напрямую на стороне клиента, что сокращает время ожидания и повышает скорость работы приложения.
* Широкая библиотека: имеет множество готовых библиотек и каркасов (например, React, Vue), что сокращает время на создание приложений.

Django — каркас на языке Python, предназначенный для создания полноцённых веб-приложений. Использует подход Model-View-Template (MVT), что помогает эффективно организовать структуру проекта.

Преимущества Django:

* безопасность: имеет встроенные механизмы защиты, позволяющие избегать CSRF, XSS и других угроз.
* Готовая структура: каркас концентрирует логику, данные и интерфейс в разные слоя, что повышает читаемость и легкость обслуживания кода.
* Широкая библиотека: имеет множество готовых приложений и модулей (например, для работы с аутентификацией, сессями, формами), что сокращает время разработки.
* Масштабируемость: подходит для создания крупных проектов с повышенной сложностью.

SQLite — легкая реляционная база данных, позволяющая эффективно хранить структированные данные приложения.

Преимущества SQLite:

* без сервера: вся база заключается в одном файле — легкость внедрения и обслуживания.
* Простота: подходит для небольших проектов, стартапов и приложений с легкой логикой.
* Быстрота: легкость конструкции позволяет эффективно читать и записывать данные.
* Готовность: входит в стандартный комплект Python, что сокращает количество необходимых установок и зависимостей.
* Использование такого набора технологий даёт преимущества при создании веб-приложений:
* Готовая платформа: Django + SQLite отлично подходит для создания легких и средних проектов с помощью Python.
* Гибкость: HTML, CSS и JavaScript обеспечивают точное воспроизводство дизайна и динамику на стороне клиента.
* Безопасность и устойчивость: каркас Django защищает приложение, а SQLite подходит для легких задач с базами данных.
* Быстрый старт: эту комбинацию технологий проще всего внедрять и масштабировать, что особенно ценно на старте проекта.
  1. Тестирование и отладка ИС

В ходе разработки веб-приложения одним из ключевых этапов является его тестирование, которое позволяет выявить возможные ошибки, несоответствия и недостатки как в логике работы, так и во внешнем виде и удобстве использования. Тестирование обеспечивает контроль качества на всех уровнях и помогает убедиться, что разработанный продукт соответствует заявленным требованиям и ожиданиям пользователей. В рамках данного проекта тестирование проводилось вручную, что подразумевает тщательную проверку каждого элемента интерфейса и функциональных возможностей с помощью непосредственного взаимодействия с приложением и анализа его поведения в различных ситуациях.

Ручное тестирование предоставляет возможность оценить не только техническую корректность работы, но и удобство, интуитивность и общее восприятие продукта конечным пользователем. Это особенно важно при создании клиентских приложений, где комфорт и простота взаимодействия напрямую влияют на успешность и востребованность решения. Такой подход включает проверку правильности отображения всех элементов интерфейса — блоков, кнопок, изображений, а также работоспособности гиперссылок и форм, их корректной валидации и логики обработки данных. Особое внимание уделялось навигации и удобству пользования приложением, чтобы пользователь мог легко и быстро переходить между страницами, находить необходимую информацию и выполнять необходимые действия без затруднений.

В ходе тестирования также оценивалась визуальная составляющая: соответствие дизайна техническому заданию и макетам, единообразие и корректность отображения элементов на разных устройствах и браузерах, читаемость текстового контента и эстетическая привлекательность интерфейса. Это позволяет обеспечить не только функциональную, но и эстетическую целостность продукта, что является важным фактором для формирования положительного пользовательского опыта.

Для дополнительной оценки технической оптимизации веб-приложения был использован онлайн-сервис PageSpeed Insights от Google. Он позволил провести автоматизированный аудит производительности, выявить потенциальные узкие места и получить рекомендации по улучшению загрузки и взаимодействия с пользователем. Результаты тестирования продемонстрировали высокий уровень оптимизации: производительность приложения составила 93 балла для настольных устройств и 74 балла для мобильных устройств.

Были получены следующие ключевые показатели:

**Мобильные устройства:**

* время до отображения первого контентного элемента (First Contentful Paint) — 1,6 сек.
* Время до отображения крупнейшего контентного элемента (Largest Contentful Paint) — 8,6 сек.
* Индекс скорости (Speed Index) — 2,9 сек.
* Общее время блокировки (Total Blocking Time) — 0 мс.
* Совокупное смещение макета (Cumulative Layout Shift) — 0.

**Настольные устройства:**

* время до отображения первого контентного элемента (First Contentful Paint) — 0,5 сек.
* Время до отображения крупнейшего контентного элемента (Largest Contentful Paint) — 1,6 сек.
* Индекс скорости (Speed Index) — 1,2 сек.
* Общее время блокировки (Total Blocking Time) — 0 мс.
* Совокупное смещение макета (Cumulative Layout Shift) — 0.

Эти метрики свидетельствуют о хорошей скорости загрузки и стабильности интерфейса, особенно на десктопах. Более низкий показатель на мобильных устройствах связан в основном с длительным отображением крупных элементов (Largest Contentful Paint), что может быть доработано в будущем. Скриншоты результатов анализа представлены на рисунках 10 и 11:

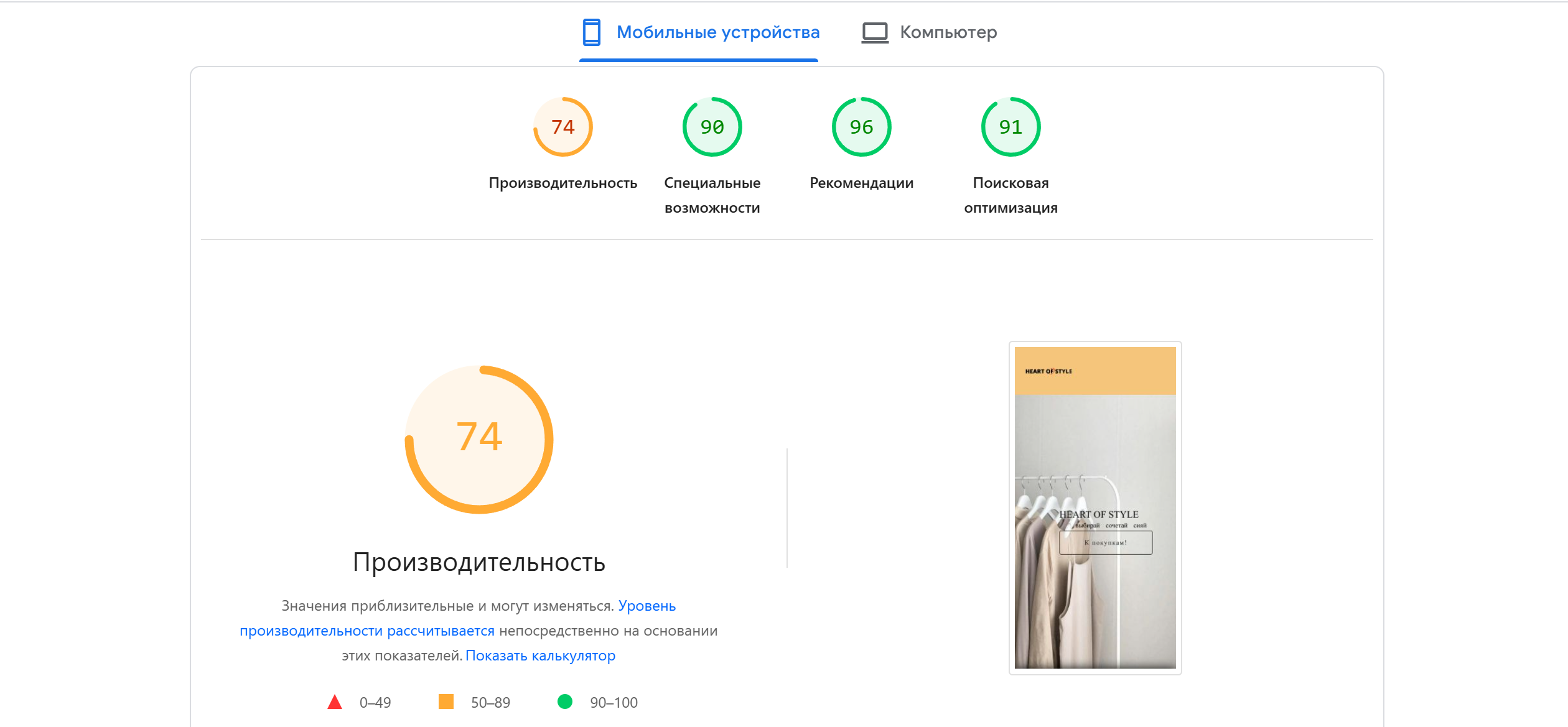


Рисунок 10 − Результаты PageSpeed Insights для мобильной версии

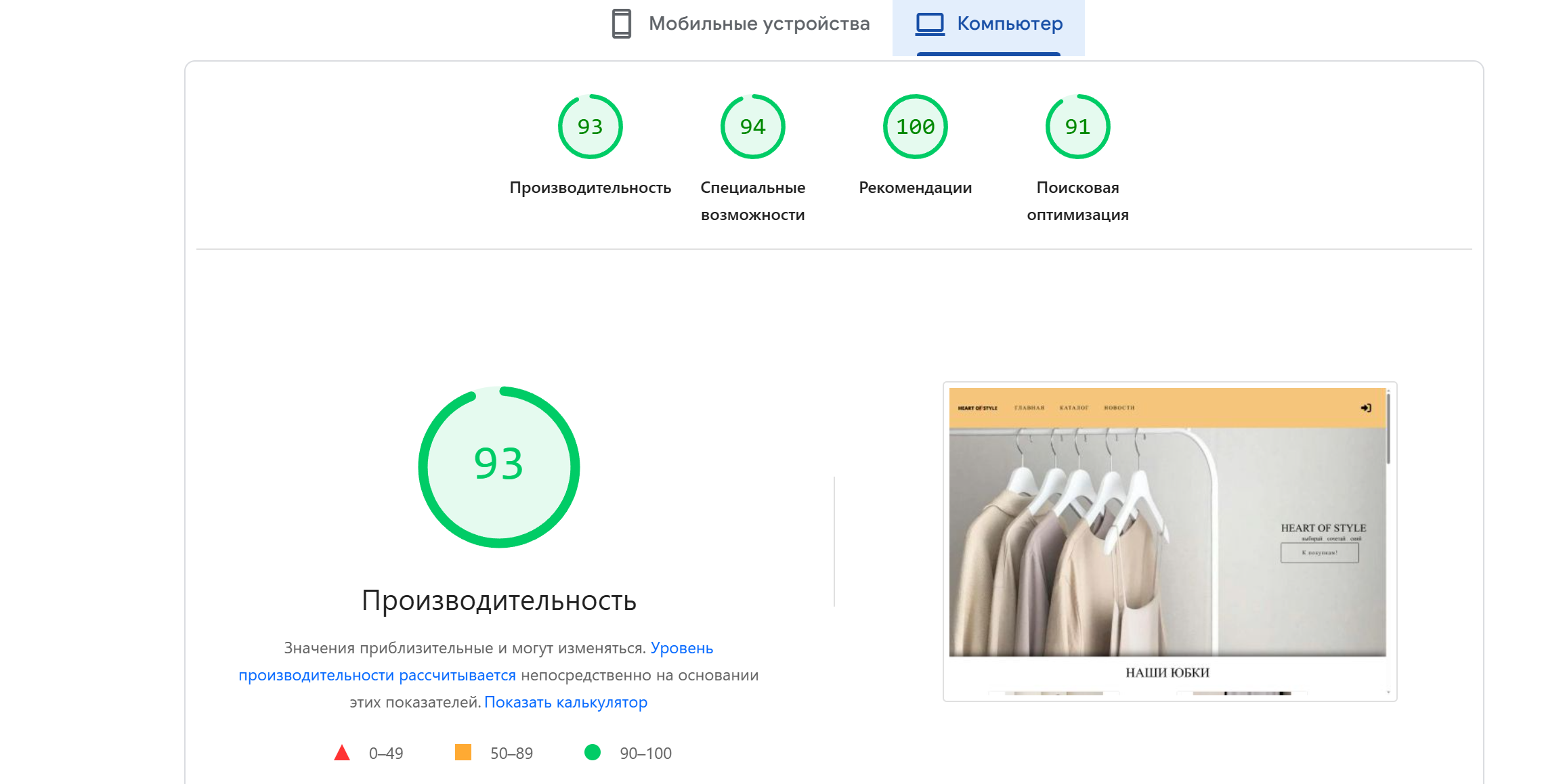


Рисунок 11 − Результаты PageSpeed Insights для десктопной версии

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения дипломного проекта было разработано современное веб-приложение для интернет-магазина кожаной одежды «Heart of Style», включающее полный цикл взаимодействия с пользователем: от выбора товара до оформления заказа и получения в пункте выдачи.

**Достигнутые результаты:**

1. реализована полная функциональность:

* удобный каталог,
* корзина с возможностью редактирования заказа,
* личный кабинет для пользователей и админ-панель для управления контентом.

1. Технологический стек:

* backend: Django (Python) для обработки бизнес-логики,
* frontend: HTML, CSS, JavaScript для динамического интерфейса,
* база данных: SQLite (с возможностью миграции на PostgreSQL/MySQL).

1. Оптимизация для бизнеса:

* SEO-оптимизация и адаптивный дизайн под мобильные устройства.

**Практическая значимость:**

Проект готов к внедрению в малый и средний бизнес. Ключевые преимущества:

* cнижение затрат на разработку,
* гибкость для масштабирования (добавление новых категорий товаров).

**Перспективы развития:**

* внедрение рекомендательной системы на основе машинного обучения;
* добавление AR-примерки товаров через камеру смартфона;
* расширение функционала веб-приложения.

**Вывод:**  
Разработанное веб-приложение соответствует современным требованиям, демонстрирует высокую производительность и безопасность, а также предоставляет удобный инструмент для увеличения продаж кожаной одежды.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Владимир Дронов Django 4. Практика создания веб-сайтов на Python – БХВ – 2024, 800с
2. Робин Никсон HTML5 и CSS3. Мастер-класс – БХВ-Петербург – 2024, 464c
3. Уолтер Шилдс SQL: быстрое погружение – Питер – 2025, 224с
4. Алексей Васильев JavaScript в примерах и задачах – Эксмо – 2025, 720с
5. Visual Studio Code – официальный сайт [Электронный ресурс]. Режим доступа: <https://code.visualstudio.com/>
6. PyCharm – официальный сайт [Электронный ресурс]. Режим доступа: https://www.jetbrains.com/pycharm/
7. Git – официальный сайт [Электронный ресурс]. Режим доступа: https://git-scm.com/
8. Font Awesome – официальный сайт [Электронный ресурс]. Режим доступа: <https://fontawesome.com/>
9. Python – официальный сайт [Электронный ресурс]. Режим доступа: <https://www.python.org/>
10. Django – официальный сайт [Электронный ресурс]. Режим доступа: <https://www.djangoproject.com/>
11. SQLite – официальный сайт [Электронный ресурс]. Режим доступа: https://www.sqlite.org/

**ПРИЛОЖЕНИЕ А**

Листинг А.1 – Код базового файла

{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

    <!-- Подключение CSS файлов -->

    <link rel="stylesheet" href="{% static 'my\_css/font\_awesome.css' %}">

    <link rel="stylesheet" href="{% static 'bootstrap.min.css' %}">

    <link rel="stylesheet" href="{% static 'my\_css/style.css' %}">

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- Блок для заголовка страницы -->

    <title>{% block title %}Магазин{% endblock %}</title>

</head>

<body class="d-flex flex-column min-vh-100">

    <!-- Навигационная панель -->

    <nav class="navbar navbar-expand-lg bg-body-tertiary">

        <div class="container-fluid">

            <!-- Логотип -->

            <img src="{% static 'img/logo.png' %}" alt="Логотип" class="logo">

            <div class="collapse navbar-collapse" id="navbarSupportedContent">

                <!-- Основные ссылки меню -->

                <ul class="navbar-nav me-auto mb-2 mb-lg-0">

                    <li class="nav-item">

                        <a class="nav-link active" href="{% url 'index' %}">Главная</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link active" href="{% url 'catalog' %}">Каталог</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link active" href="{%url 'about'%}">О нас</a>

                    </li>

                </ul>

                <!-- Иконки пользователя -->

                <form class="d-flex" role="search">

                    {% if not user.is\_authenticated %}

                        <!-- Ссылка на вход -->

                        <a class="fa-solid fa-right-to-bracket fa active" href="{% url 'login' %}" title="Войти"></a>

                    {% else %}

                        {% if user.is\_superuser %}

                            <!-- Админ-панель -->

                            <a class="fa-solid fa-book fa active" href="{% url 'admin\_order' %}" title="Заказы (админ)"></a>

                            <a class="fa-solid fa-user-tie fa active" href="{% url 'admin\_product' %}" title="Продукты (админ)"></a>

                        {% else %}

                            <!-- Корзина -->

                            <a href="{% url 'cart\_view' %}" class="fa-solid fa-basket-shopping fa active" title="Корзина">

                                (<span class="cart-counter">{{ request.user.cart.total\_items|default:0 }}</span>)

                            </a>

                            <!-- Профиль -->

                            <a class="fa-solid fa-user fa active" href="{% url 'profile' %}" title="Профиль"></a>

                        {% endif %}

                        <!-- Выход -->

                        <a class="fa-solid fa-right-from-bracket fa active" href="{% url 'logout' %}" title="Выйти"></a>

                    {% endif %}

                </form>

            </div>

        </div>

    </nav>

    <!-- Контентная часть страницы -->

    <div class="container flex-grow-1">

        {% block content %}{% endblock %}

    </div>

    <!-- Футер -->

    <footer class="text-center py-3">

        <div class="footer center">

            <!-- Социальные и контактные иконки -->

            <a href="#" class="me-3 fa-solid fa-phone fa active" title="Позвонить"></a>

            <a href="#" class="me-3 fa-brands fa-vk fa active" title="VK"></a>

            <a href="#" class="me-3 fa-brands fa-square-whatsapp fa active" title="WhatsApp"></a>

        </div>

        <!-- Форма подписки -->

        <form>

            <input type="email" placeholder="Введите ваш email">

            <button type="submit">Подписаться</button>

        </form>

    </footer>

    <!-- Подключение скриптов -->

    {% block scripts %}

        <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

    {% endblock %}

</body>

</html>

Листинг А.2 – Код главной страницы

{% extends 'base.html' %}

{% load static %}

{% block title %}Главная страница{% endblock %}

{% block content %}

<!-- Основная секция с предложением (hero section) -->

<div class="offer">

    <div class="header\_nazvanie">

        <!-- Логотип/название магазина -->

        <h2 class="header\_logo">Heart of Style</h2>

        <!-- Слоган магазина -->

        <div class="header\_devis">

            <p>выбирай</p>

            <p>сочетай</p>

            <p>сияй</p>

        </div>

        <!-- Кнопка перехода в каталог -->

        <a href="{% url 'catalog' %}" class="btn">

            <p class="text\_btn">К покупкам!</p>

        </a>

    </div>

</div>

<main>

    <section class="section">

        <!-- Секция каталога юбок -->

        <h1 class="h1">Наши юбки</h1>

        <!-- Сетка товаров (используем Bootstrap grid) -->

        <div class="row row-cols-1 row-cols-md-2 g-4 catalog">

            {% for image in products %}

            <div class="col">

                <!-- Карточка товара -->

                <div class="card card\_\_image-wrapper">

                    <!-- Ссылка на страницу товара -->

                    <a href="{{ image.product.get\_absolute\_url }}">

                        <!-- Изображение товара -->

                        <img src="{{image.image.url}}" class="card-img-top card\_image" alt="{{ image.product.name }}">

                    </a>

                    <!-- Тело карточки с информацией о товаре -->

                    <div class="card-body card\_\_text">

                        <!-- Название товара -->

                        <h5 class="card-title ubka\_nazvanie dop-text">

                            <a href="{{ image.product.get\_absolute\_url }}">{{ image.product.name }}</a>

                        </h5>

                        <!-- Описание товара -->

                        <p class="card-text card\_\_text-description">{{ image.product.description }}</p>

                        <!-- Блок с ценой и кнопками действий -->

                        <div class="sail">

                            <!-- Цена товара -->

                            <p class="card\_\_text-price">{{ image.product.price }}₽</p>

                            <!-- Кнопки действий -->

                            <div class="card\_sail">

                                <!-- Кнопка "Добавить в корзину" с data-атрибутами -->

                                <a href="#" class="add-to-cart-link"

                                   data-product-id="{{ image.product.id }}"

                                   {% if image.color %}data-color-id="{{ image.color.id }}"{% endif %}>

                                    <span class="fa-solid fa-basket-shopping"></span>

                                </a>

                            </div>

                        </div>

                    </div>

                </div>

            </div>

            {% endfor %}

        </div>

        <!-- Модальное окно для выбора размера (Bootstrap Modal) -->

        <div class="modal fade" id="sizeModal" tabindex="-1" aria-hidden="true">

            <div class="modal-dialog">

                <div class="modal-content">

                    <div class="modal-header">

                        <h5 class="modal-title">Выберите размер</h5>

                        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

                    </div>

                    <div class="modal-body">

                        <!-- Форма для выбора размера -->

                        <form id="sizeForm">

                            <div class="size-options"></div> <!-- Сюда будут добавляться варианты размеров -->

                        </form>

                    </div>

                    <div class="modal-footer">

                        <form>

                            <div class="form-row">

                                <!-- Кнопка подтверждения выбора размера -->

                                <button type="button" class="btn" id="confirmSize">Добавить в корзину</button>

                            </div>

                        </form>

                    </div>

                </div>

            </div>

        </div>

    </section>

</main>

{% endblock %}

{% block scripts %}

{{ block.super }}

<script>

$(document).ready(function() {

    // Переменные для хранения текущего состояния

    let currentProductId = null;    // ID текущего товара

    let currentColorId = null;      // ID текущего цвета (если есть)

    let $activeButton = null;       // Активная кнопка "Добавить в корзину"

    const sizeModal = new bootstrap.Modal(document.getElementById('sizeModal')); // Инициализация модального окна

    // Функция для получения CSRF-токена из cookies

    function getCookie(name) {

        let cookieValue = null;

        if (document.cookie && document.cookie !== '') {

            const cookies = document.cookie.split(';');

            for (let cookie of cookies) {

                cookie = cookie.trim();

                if (cookie.startsWith(name + '=')) {

                  cookieValue = decodeURIComponent(cookie.substring(name.length + 1));

                    break;

                }

            }

        }

        return cookieValue;

    }

    // Обработчик клика по кнопке "Добавить в корзину"

    $('.add-to-cart-link').click(function(e) {

        e.preventDefault();

        // Сохраняем данные о товаре

        currentProductId = $(this).data('product-id');

        currentColorId = $(this).data('color-id') || null;

        $activeButton = $(this); // сохраняем ссылку на активную кнопку

        // Меняем иконку на спиннер загрузки

        const $icon = $activeButton.find('span');

        $icon.removeClass('fa-basket-shopping').addClass('fa-spinner fa-spin');

        // AJAX-запрос для получения доступных размеров

        $.get(`/get\_sizes/?product\_id=${currentProductId}${currentColorId ? `&color\_id=${currentColorId}` : ''}`, function(data) {

            const $sizeOptions = $('.size-options');

            $sizeOptions.empty();

            // Если есть размеры - показываем модальное окно для выбора

            if (data.sizes && data.sizes.length > 0) {

                for (let size of data.sizes) {

                    $sizeOptions.append(`

                        <div class="form-check">

                            <input class="form-check-input" type="radio" name="size" id="size-${size.id}" value="${size.id}" required>

                            <label class="form-check-label" for="size-${size.id}">${size.name}</label>

                        </div>

                    `);

                }

                sizeModal.show();

            } else {

                // Если размеров нет - сразу добавляем в корзину

                addToCart(currentProductId, currentColorId, null, $activeButton);

            }

            // Возвращаем стандартную иконку

            $icon.removeClass('fa-spinner fa-spin').addClass('fa-basket-shopping');

        }).fail(function() {

            // В случае ошибки возвращаем стандартную иконку и показываем сообщение

            $activeButton.find('span').removeClass('fa-spinner fa-spin').addClass('fa-basket-shopping');

            alert('Ошибка загрузки размеров.');

        });

    });

    // Обработчик клика по кнопке подтверждения выбора размера

    $('#confirmSize').click(function() {

        const sizeId = $('input[name="size"]:checked').val();

        if (!sizeId) {

            alert('Пожалуйста, выберите размер');

            return;

        }

        // Добавляем товар с выбранным размером в корзину

        addToCart(currentProductId, currentColorId, sizeId, $activeButton);

        sizeModal.hide();

    });

    // Функция добавления товара в корзину

    function addToCart(productId, colorId, sizeId, $button) {

        const csrftoken = getCookie('csrftoken');

        const $icon = $button.find('span');

        // Меняем иконку на спиннер

        $icon.removeClass('fa-basket-shopping').addClass('fa-spinner fa-spin');

        // Отправляем AJAX-запрос на сервер

        $.ajax({

            url: '{% url "add\_to\_cart" %}',

            type: 'POST',

            headers: {'X-CSRFToken': csrftoken}, // CSRF-защита

            data: {

                'product\_id': productId,

                'color': colorId,

                'size': sizeId

            },

            success: function(data) {

                // При успехе меняем иконку на "галочку" на 1 секунду

                $icon.removeClass('fa-spinner fa-spin').addClass('fa-check');

                setTimeout(() => {

                    $icon.removeClass('fa-check').addClass('fa-basket-shopping');

                }, 1000);

                // Обновляем счетчик товаров в корзине (если есть в ответе)

                if (data.cart\_count !== undefined) {

                    $('.cart-count').text(data.cart\_count);

                }

                // Если сервер вернул ошибку - показываем сообщение

                if (!data.success) {

                    alert(data.message || 'Ошибка добавления в корзину');

                }

            },

            error: function(xhr) {

                // При ошибке возвращаем стандартную иконку и показываем сообщение

                $icon.removeClass('fa-spinner fa-spin').addClass('fa-basket-shopping');

                alert(xhr.responseJSON?.message || 'Ошибка на сервере при добавлении в корзину');

            }

        });

    }

});

</script>

{% endblock %}

Листинг А.3 – Код страницы каталога

{% extends 'base.html' %}

{% load static %}

{% block title %}Каталог{% endblock %}

{% block content %}

<h1 class="h1">Каталог</h1>

<div class="products-container">

  {% for entry in product\_images %}

  {% with image=entry.image product=entry.product color=entry.color sizes=entry.sizes %}

  <div class="card-wrapper">

    <div class="image-holder">

      <!-- Ссылка на страницу товара -->

        <div class="image-liquid">

          <img src="{{ image.image.url }}" alt="{{ product.name }}">

        </div>

      <!-- Кнопка добавления в корзину -->

      <div class="plus-overlay">

        <form class="add-to-cart-form">

          {% csrf\_token %}

          <input type="hidden" name="product\_id" value="{{ product.id }}">

          <input type="hidden" name="color" value="{{ color.id }}">

          <input type="hidden" name="size" class="selected-size" value="">

          <input type="hidden" name="quantity" value="1">

          <button type="button" class="plus-circle add-to-cart-button" data-url="{% url 'add\_to\_cart' %}">

            <i class="fa-solid fa-basket-shopping fa"></i>

          </button>

        </form>

      </div>

    </div>

    <!-- Информация о товаре -->

    <div class="product-description">

      <h5 class="product-title">{{ product.name }}</h5>

      <div class="row">

        <div class="col-xs-12 col-sm-8 category">{{ product.description|truncatechars:50 }}</div>

        <div class="col-xs-12 col-sm-4 price">{{ product.price }}₽</div>

      </div>

      <!-- Дополнительные детали (размеры и цвета) -->

      <a href="{% url 'product\_detail' product.id color.id %}" class="image-link">

        <div class="extra-details">

          <div class="sizes-wrapper">

            <p>Размеры</p>

            <div class="size-list">

              {% for size in sizes %}

                <button type="button" class="size-option" data-size-id="{{ size.id }}">{{ size.name }}</button>

              {% endfor %}

            </div>

          </div>

          <div class="colors-wrapper">

            <p>Цвета</p>

            <div class="color-list">

              {% for c in entry.colors %}

                <button

                  type="button"

                  class="color-sample {% if c.id == color.id %}selected{% endif %}"

                  style="background-color: {{ c.hex\_code }};"

                  data-color-id="{{ c.id }}"

                  data-product-id="{{ product.id }}">

                </button>

              {% endfor %}

            </div>

          </div>

        </a>

      </div>

    </div>

  </div>

  {% endwith %}

  {% endfor %}

</div>

<!-- Модальное окно для выбора размера (показывается, если размер не выбран) -->

<div class="modal fade" id="sizeModal" tabindex="-1" aria-labelledby="sizeModalLabel" aria-hidden="true">

  <div class="modal-dialog">

    <div class="modal-content">

      <div class="modal-header">

        <h5 class="modal-title" id="sizeModalLabel">Выберите размер</h5>

        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Закрыть"></button>

      </div>

      <div class="modal-body">

        <div id="modal-sizes" class="size-options"></div>

      </div>

      <div class="modal-footer">

        <button type="button" class="btn" id="confirmSize">Добавить в корзину</button>

      </div>

    </div>

  </div>

</div>

{% endblock %}

{% block scripts %}

{{ block.super }}

<script>

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

  // Элементы модального окна

  const sizeModal = document.getElementById("sizeModal");

  const modalSizes = document.getElementById("modal-sizes");

  const confirmBtn = document.getElementById("confirmSize");

  let currentForm = null; // Текущая форма, которая будет отправлена после выбора размера

  // Обработчик кликов по документу

  document.body.addEventListener("click", async function (e) {

    // Выбор размера в карточке товара

    if (e.target.classList.contains("size-option")) {

      const btn = e.target;

      const sizeList = btn.closest(".size-list");

      if (sizeList) {

        // Снимаем выделение со всех кнопок в этой группе

        sizeList.querySelectorAll(".size-option").forEach(b => b.classList.remove("selected"));

        // Выделяем текущую кнопку

        btn.classList.add("selected");

        // Обновляем значение в форме

        const form = btn.closest(".card-wrapper").querySelector("form.add-to-cart-form");

        if (form) {

          form.querySelector("input[name='size']").value = btn.dataset.sizeId;

        }

      }

    }

    // Выбор цвета в карточке товара

    if (e.target.classList.contains("color-sample")) {

      const card = e.target.closest(".card-wrapper");

      const form = card.querySelector("form.add-to-cart-form");

      // Снимаем выделение со всех цветов

      card.querySelectorAll(".color-sample").forEach(b => b.classList.remove("selected"));

      // Выделяем текущий цвет

      e.target.classList.add("selected");

      // Обновляем значение в форме

      form.querySelector("input[name='color']").value = e.target.dataset.colorId;

    }

    // Нажатие кнопки "Добавить в корзину"

    if (e.target.closest(".add-to-cart-button")) {

      const button = e.target.closest(".add-to-cart-button");

      const form = button.closest("form.add-to-cart-form");

      const url = button.dataset.url;

      const size = form.querySelector("input[name='size']").value;

      const product\_id = form.querySelector("input[name='product\_id']").value;

      const color = form.querySelector("input[name='color']").value;

      const quantity = form.querySelector("input[name='quantity']").value;

      const csrf = form.querySelector("input[name='csrfmiddlewaretoken']").value;

      // Если размер не выбран, показываем модальное окно

      if (!size) {

        e.preventDefault();

        currentForm = form;

        const sizeButtons = form.closest(".card-wrapper").querySelectorAll(".size-option");

        modalSizes.innerHTML = "";

        // Копируем кнопки размеров в модальное окно

        sizeButtons.forEach(btn => {

          const clone = btn.cloneNode(true);

          clone.classList.remove("selected");

          clone.classList.add("size-option");

          clone.setAttribute("data-size-id", btn.dataset.sizeId);

          modalSizes.appendChild(clone);

        });

        // Показываем модальное окно

        const bsModal = new bootstrap.Modal(sizeModal);

        bsModal.show();

        return;

      }

      // Отправка данных на сервер

      try {

        const response = await fetch(url, {

          method: 'POST',

          headers: {

            'Content-Type': 'application/x-www-form-urlencoded',

            'X-CSRFToken': csrf,

          },

          body: new URLSearchParams({

            product\_id,

            color,

            size,

            quantity

          }),

        });

        const data = await response.json();

        if (data.success) {

          // Анимация иконки корзины

          const cartIcon = document.querySelector(".cart-icon");

          if (cartIcon) {

            cartIcon.classList.add("cart-bounce");

            setTimeout(() => cartIcon.classList.remove("cart-bounce"), 500);

          }

          // Обновление счетчика товаров

          const cartCount = document.querySelector("#cart-count");

          if (cartCount && data.cart\_count !== undefined) {

            cartCount.textContent = data.cart\_count;

          }

        } else {

          console.error("Ошибка при добавлении:", data.error);

        }

      } catch (err) {

        alert("Ошибка при добавлении товара");

        console.error(err);

      }

    }

    // Выбор размера в модальном окне

    if (e.target.closest("#modal-sizes .size-option")) {

      modalSizes.querySelectorAll(".size-option").forEach(b => b.classList.remove("selected"));

      e.target.classList.add("selected");

    }

  });

  // Подтверждение выбора размера в модальном окне

  confirmBtn.addEventListener("click", function () {

    const selected = modalSizes.querySelector(".size-option.selected");

    if (!selected) return;

    const sizeId = selected.dataset.sizeId;

    // Устанавливаем выбранный размер в форме

    currentForm.querySelector("input[name='size']").value = sizeId;

    // Закрываем модальное окно

    const bsModal = bootstrap.Modal.getInstance(sizeModal);

    bsModal.hide();

    // Имитируем клик по кнопке "Добавить в корзину"

    currentForm.querySelector(".add-to-cart-button").click();

  });

});

</script>

{% endblock %}

Листинг А.4 – Код страницы «О нас»

{% extends "base.html" %}

{% load static %}

{% block title %}О нас - LederCraft{% endblock %}

{% block content %}

<div class="container py-5">

  <h1 class="text-center mb-5 display-5 fw-bold">О нас</h1>

  <!-- Наша философия -->

  <div class="row align-items-center mb-5">

    <div class="col-md-6">

      <img src="{% static 'img/about/philosophy.jpg' %}" alt="Философия LederCraft" class="img-fluid rounded-3 shadow">

    </div>

    <div class="col-md-6">

      <h2 class="h4 fw-bold mb-3">Наша философия</h2>

      <p class="lead">

        Мы верим, что кожаная одежда — это не просто мода. Это образ жизни.

        Это уверенность, индивидуальность и прочность, проверенная временем.

        В <strong>Heart of Style</strong> мы создаем вещи, которые вы будете носить с гордостью годами.

      </p>

    </div>

  </div>

  <!-- Как всё начиналось -->

  <div class="row align-items-center mb-5 flex-md-row-reverse">

    <div class="col-md-6">

      <img src="{% static 'img/about/about-workshop.jpg' %}" alt="Как всё начиналось" class="img-fluid rounded-3 shadow">

    </div>

    <div class="col-md-6">

      <h2 class="h4 fw-bold mb-3">Как всё начиналось</h2>

      <p class="lead">

        Наш бренд родился в 2018 году в мастерской из натуральной кожи, когда идея

        "шить как для себя" стала нашей главной ценностью. Сначала это были

        кастомные куртки для друзей, сегодня — тысячи довольных клиентов по всей стране.

      </p>

    </div>

  </div>

  <!-- Производство -->

  <div class="row align-items-center mb-5">

    <div class="col-md-6">

      <img src="{% static 'img/about/about-leather.jpg' %}" alt="Производство LederCraft" class="img-fluid rounded-3 shadow">

    </div>

    <div class="col-md-6">

      <h2 class="h4 fw-bold mb-3">Производство</h2>

      <p class="lead">

        Все изделия мы создаём вручную в России. Только первоклассная натуральная кожа, проверенные поставщики

        и надёжные технологии. Каждая деталь проходит строгий контроль качества.

      </p>

    </div>

  </div>

  <!-- Наши ценности -->

  <div class="mb-5">

    <h2 class="h4 fw-bold mb-3">Наши ценности</h2>

    <ul class="lead list-unstyled ps-3">

      <li>✔ Натуральные материалы</li>

      <li>✔ Прочное и честное производство</li>

      <li>✔ Честные цены без посредников</li>

      <li>✔ Поддержка ремесленного труда</li>

      <li>✔ Уникальный стиль вне времени</li>

    </ul>

  </div>

  <!-- CTA -->

  <div class="text-center">

    <h2 class="h4 fw-bold mb-3">Присоединяйтесь к Heart of Style</h2>

    <p class="lead">Мы гордимся каждым изделием и будем рады, если кожаная вещь от нас станет вашей любимой.</p>

    <button onclick="window.location.href='{% url 'catalog' %}'" class="btn">Перейти в каталог</button>

  </div>

</div>

{% endblock %}

Листинг А.5 – Код карточки товара

{% extends "base.html" %}

{% load static %}

{% block content %}

<div class="product-detail-container" style="max-width: 1000px; margin: auto; padding: 40px 20px; font-family: 'Segoe UI', sans-serif; color: #333;">

  <!-- Верхний блок: изображение + основная информация -->

  <div style="display: flex; flex-wrap: wrap; gap: 40px;">

    <!-- Галерея изображений -->

    <div style="flex: 1 1 400px;">

      <div style="display: flex; flex-direction: column; gap: 10px;">

        {% for image in images %}

          <img src="{{ image.image.url }}" alt="{{ product.name }}" style="width: 100%; border-radius: 12px; box-shadow: 0 2px 8px rgba(0,0,0,0.1);">

        {% empty %}

          <p>Изображения отсутствуют</p>

        {% endfor %}

      </div>

    </div>

    <!-- Информация о товаре -->

    <div style="flex: 1 1 400px; display: flex; flex-direction: column; justify-content: space-between;">

      <div>

        <h1 style="font-size: 32px; margin-bottom: 10px;">{{ product.name }}</h1>

        <!-- Цвет -->

        <p style="margin-bottom: 15px; font-size: 16px;">

          <strong>Цвет:</strong>  {{ variation.color.colored\_circle|safe }}

        </p>

        <!-- Описание -->

        <p style="font-size: 15px; line-height: 1.6; color: #555;">{{ product.description }}</p>

      </div>

      <!-- Цена -->

      <div style="margin-top: 25px;">

        <span style="font-size: 28px; font-weight: bold;">{{ product.price }} ₽</span>

      </div>

      <!-- Форма добавления в корзину -->

      <form method="post" action="{% url 'add\_to\_cart' %}" style="margin-top: 30px; display: flex; flex-direction: column; gap: 15px;">

        {% csrf\_token %}

        <!-- Размер -->

        <div>

          <label for="size" style="font-weight: 600;">Размер:</label><br>

          <select name="size" id="size" required style="padding: 10px; width: 100%; border-radius: 8px; border: 1px solid #ccc; margin-top: 5px;">

            {% for size in sizes %}

              <option value="{{ size.id }}">{{ size.name }}</option>

            {% endfor %}

          </select>

        </div>

        <!-- Количество -->

        <div>

          <label for="quantity" style="font-weight: 600;">Количество:</label><br>

          <input type="number" name="quantity" id="quantity" value="1" min="1"

                 style="padding: 10px; width: 100px; border-radius: 8px; border: 1px solid #ccc; margin-top: 5px;">

        </div>

        <!-- Кнопка -->

        <button type="submit"

                style="padding: 14px; background-color: #1a1a1a; color: #fff; font-weight: bold; border: none; border-radius: 8px; cursor: pointer; transition: background 0.3s;">

          🛒 Добавить в корзину

        </button>

      </form>

    </div>

  </div>

</div>

{% endblock %}

Листинг А.6 – Код страницы корзины

{% extends 'base.html' %}

{% load static %}

{% block title %}Корзина{% endblock %}

{% block content %}

<div class="cart-container">

    <h1 class="cart-title">🛒 Ваша корзина</h1>

    {% if items %}

    <div class="cart-items">

        <form method="post" action="{% url 'proceed\_to\_order' %}" id="cart-form">

            {% csrf\_token %}

            <table class="cart-table">

                <thead>

                    <tr>

                        <th></th>

                        <th>Товар</th>

                        <th>Цвет</th>

                        <th>Размер</th>

                        <th>Цена</th>

                        <th>Количество</th>

                        <th>Сумма</th>

                        <th></th>

                    </tr>

                </thead>

                <tbody>

                    {% for item in items %}

                    <tr class="cart-item">

                        <td>

                            <input type="checkbox"

                                   name="selected\_items"

                                   value="{{ item.id }}"

                                   class="select-item"

                                   data-total="{{ item.total\_price }}"

                                   checked>

                        </td>

                        <td>

                            <a href="{{ item.product.get\_absolute\_url }}" class="cart-product-link">

                                {{ item.product.name }}

                            </a>

                        </td>

                        <td>

                            {% if item.color %}

                                <div class="color-dot" style="background: {{ item.color.hex\_code }};"></div>

                                {{ item.color.name }}

                            {% else %}

                            {% endif %}

                        </td>

                        <td>{{ item.size.name|default:"—" }}</td>

                        <td>{{ item.product.price }} ₽</td>

                        <td>

                            <input type="number"

                                   name="quantity\_{{ item.id }}"

                                   value="{{ item.quantity }}"

                                   min="1"

                                   class="quantity-input"

                                   disabled>

                        </td>

                        <td>{{ item.total\_price }} ₽</td>

                        <td>

                            <button type="button"

                                    class="remove-btn"

                                    onclick="removeFromCart({{ item.id }}, '{{ csrf\_token }}')">

                                <i class="fas fa-trash-alt"></i>

                            </button>

                        </td>

                    </tr>

                    {% endfor %}

                </tbody>

            </table>

            <div class="cart-summary">

                <div class="cart-total">

                    <span>Итого:</span>

                    <span class="total-price" id="total-price-display">0 ₽</span>

                </div>

                <div class="form-row">

                    <button type="submit" class="btn" id="checkout-btn">

                        Оформить заказ

                    </button>

                </div>

            </div>

        </form>

    </div>

    {% else %}

    <div class="cart-empty">

        <p>Ваша корзина пуста</p>

        <a href="{% url 'catalog' %}" class="continue-shopping">

            Продолжить покупки

        </a>

    </div>

    {% endif %}

</div>

{% endblock %}

{% block scripts %}

{{ block.super }}

<script>

// Пересчёт общей суммы

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

    const checkboxes = document.querySelectorAll(".select-item");

    const totalDisplay = document.getElementById("total-price-display");

    const checkoutBtn = document.getElementById("checkout-btn");

    function recalculateTotal() {

        let total = 0;

        let selected = 0;

        checkboxes.forEach(cb => {

            if (cb.checked) {

                total += parseFloat(cb.dataset.total);

                selected += 1;

            }

        });

        totalDisplay.textContent = total.toFixed(2) + " ₽";

        checkoutBtn.disabled = selected === 0;

    }

    checkboxes.forEach(cb => {

        cb.addEventListener("change", recalculateTotal);

    });

    recalculateTotal();

});

// Удаление товара через POST

function removeFromCart(itemId, csrfToken) {

    const form = document.createElement('form');

    form.method = 'POST';

    form.action = `/remove\_from\_cart/${itemId}/`;

    const input = document.createElement('input');

    input.type = 'hidden';

    input.name = 'csrfmiddlewaretoken';

    input.value = csrfToken;

    form.appendChild(input);

    document.body.appendChild(form);

    form.submit();

}

</script>

{% endblock %}

Листинг А.7 – Код страницы оформления заказа

{% extends 'base.html' %}

{% load static %}

{% block title %}Оформление заказа{% endblock %}

{% block content %}

<!-- Основной контейнер с центрированием содержимого -->

<div class="d-flex justify-content-center align-items-center">

    <!-- Карточка с тенью и ограничением максимальной ширины -->

    <div class="card shadow p-4 w-100" style="max-width: 700px;">

        <div class="card-body p-4">

            <!-- Заголовок страницы -->

            <h2 class="mb-4 text-center">Оформление заказа</h2>

            <!-- Форма для оформления заказа с отключенной браузерной валидацией -->

            <form method="post" novalidate>

                {% csrf\_token %}  <!-- CSRF-токен для защиты от подделки запросов -->

                <!-- Блок для вывода системных сообщений -->

                {% if messages %}

                    {% for message in messages %}

                        <!-- Каждое сообщение выводится как alert с желтым фоном -->

                        <div class="alert alert-warning">{{ message }}</div>

                    {% endfor %}

                {% endif %}

                <!-- Секция выбора адреса доставки -->

                <div class="mb-3">

                    <!-- Лейбл для поля адреса -->

                    <label for="{{ form.address.id\_for\_label }}" class="form-label">{{ form.address.label }}</label>

                    <!-- Поле выбора адреса (рендерится Django-формой) -->

                    {{ form.address }}

                    <!-- Вывод ошибок валидации для поля адреса -->

                    {% if form.address.errors %}

                        <div class="text-danger small">{{ form.address.errors.as\_text }}</div>

                    {% endif %}

                </div>

                <!-- Секция выбора способа оплаты -->

                <div class="mb-3">

                    <!-- Лейбл для способа оплаты -->

                    <label class="form-label">{{ form.type\_payment.label }}</label>

                    <!-- Рендеринг radio-кнопок для выбора способа оплаты -->

                    {% for radio in form.type\_payment %}

                        <div class="form-check">

                            {{ radio.tag }}  <!-- Сама radio-кнопка -->

                            <!-- Лейбл для radio-кнопки -->

                            <label class="form-check-label" for="{{ radio.id\_for\_label }}">

                                {{ radio.choice\_label }}

                            </label>

                        </div>

                    {% endfor %}

                    <!-- Вывод ошибок валидации для способа оплаты -->

                    {% if form.type\_payment.errors %}

                        <div class="text-danger small">{{ form.type\_payment.errors.as\_text }}</div>

                    {% endif %}

                </div>

                <!-- Секция с перечнем выбранных товаров -->

                <h4 class="mt-4">Выбранные товары</h4>

                <!-- Список товаров в заказе -->

                <ul class="list-group mb-3">

                    {% for item in selected\_items %}

                        <!-- Каждый товар в отдельном элементе списка -->

                        <li class="list-group-item d-flex justify-content-between align-items-center">

                            <div>

                                <!-- Название товара -->

                                <strong>{{ item.product.name }}</strong>

                                <!-- Дополнительная информация: цвет, размер, количество -->

                                <div class="small text-muted">

                                    {% if item.color %}Цвет: {{ item.color.name }} {% endif %}

                                    {% if item.size %}| Размер: {{ item.size.name }}{% endif %}

                                    | Кол-во: {{ item.quantity }}

                                </div>

                            </div>

                            <!-- Цена товара с учетом количества -->

                            <span>{{ item.total\_price }} ₽</span>

                        </li>

                    {% endfor %}

                </ul>

                <!-- Блок с итоговой суммой заказа -->

                <div class="text-end fw-bold mb-3">Итого: {{ total\_price }} ₽</div>

                <!-- Кнопка отправки формы -->

                <div class="form-row text-center">

                    <button type="submit" class="btn btn-dark btn-lg px-5" id="checkout-btn">Оформить заказ</button>

                </div>

            </form>

        </div>

    </div>

</div>

{% endblock %}

Листинг А.8 – Код страницы успешного заказа

{% extends 'base.html' %}

{% load static %}

{% block title %}Заказ успешно оформлен{% endblock %}

{% block content %}

<!-- Основной контейнер с центрированием по вертикали и горизонтали -->

<div class="d-flex justify-content-center align-items-center mt-5">

    <!-- Карточка с тенью и ограничением максимальной ширины -->

    <div class="card shadow p-4 w-100" style="max-width: 600px;">

        <div class="card-body text-center">  <!-- Центрирование содержимого -->

            <!-- Иконка успеха (зеленая галочка в круге) -->

            <div class="mb-4">

                <i class="fas fa-check-circle text-success" style="font-size: 4rem;"></i>

            </div>

            <!-- Заголовок подтверждения -->

            <h2 class="mb-3">Заказ успешно оформлен!</h2>

            <!-- Блок с информацией о заказе -->

            <div class="mb-3">

                <!-- Номер заказа -->

                <p>Номер вашего заказа: <strong>№{{ order.id }}</strong></p>

                <!-- Общая сумма заказа -->

                <p>Сумма заказа: <strong>{{ order.total\_price }} ₽</strong></p>

                <!-- Статус заказа с цветным бейджем -->

                <p>Статус: <span class="badge bg-info">{{ order.get\_status\_display }}</span></p>

            </div>

            <!-- Ссылка для возврата в каталог -->

            <a href="{% url 'catalog' %}" class="continue-shopping">

                Продолжить покупки

            </a>

        </div>

    </div>

</div>

{% endblock %}

Листинг А.9 – Код страницы профиля пользователя

{% extends 'base.html' %}

{% load static %}

{% block title %}Заказ успешно оформлен{% endblock %}

{% block content %}

<!-- Основной контейнер профиля -->

<div class="profile-container">

  <!-- Заголовок страницы профиля -->

  <h1 class="profile-title">ВАШ ПРОФИЛЬ</h1>

  <!-- Секция личной информации (раскрывающийся блок) -->

  <details class="user-info">

    <summary>👤 Личная информация</summary>  <!-- Заголовок секции с иконкой -->

    <div class="user-info-content">

      <!-- Персональные данные пользователя -->

      <p><strong>Имя:</strong> {{ user.first\_name }}</p>

      <p><strong>Фамилия:</strong> {{ user.last\_name }}</p>

      <!-- Отчество с обработкой случая, когда оно не указано -->

      <p><strong>Отчество:</strong> {{ user.third\_name|default:"—" }}</p>

      <!-- Телефон с обработкой случая, когда он не указан -->

      <p><strong>Телефон:</strong> {{ user.phone|default:"—" }}</p>

      <!-- Email пользователя (обязательное поле) -->

      <p><strong>Почта:</strong> {{ user.email }}</p>

      <!-- Дата рождения в формате день.месяц.год -->

      <p><strong>День рождения:</strong> {{ user.birth\_date|date:"d.m.Y" }}</p>

    </div>

  </details>

  <!-- Секция истории заказов -->

  <h2 class="order-history-title">История заказов</h2>

  <!-- Цикл по всем заказам пользователя -->

  {% for order in orders %}

    <!-- Блок информации о заказе -->

    <div class="order-block">

      <!-- Номер и дата заказа -->

      <p><strong>Заказ №{{ order.id }}</strong> от {{ order.date\_at|date:"d.m.Y" }}</p>

      <!-- Статус заказа (используется get\_status\_display для отображения читаемого значения) -->

      <p><strong>Статус:</strong> {{ order.get\_status\_display }}</p>

      <!-- Способ оплаты -->

      <p><strong>Оплата:</strong> {{ order.get\_type\_payment\_display }}</p>

      <!-- Адрес доставки или пункт выдачи -->

      <p><strong>Пункт выдачи:</strong> {{ order.address|default:"—" }}</p>

      <!-- Таблица с товарами в заказе -->

      <table class="order-table">

        <thead>

          <tr>

            <th>Товар</th>

            <th>Цвет</th>

            <th>Размер</th>

            <th>Кол-во</th>

            <th>Цена</th>

            <th>Итого</th>

          </tr>

        </thead>

        <tbody>

          <!-- Цикл по товарам в заказе -->

          {% for item in order.items.all %}

            <tr>

              <!-- Название товара -->

              <td>{{ item.product.name }}</td>

              <!-- Цвет товара с визуальным кружком -->

              <td>

                {% if item.color %}

                  <span class="color-circle" style="background-color: {{ item.color.hex\_code }}"></span>

                  {{ item.color.name }}

                {% else %}

                  —  <!-- Если цвет не указан -->

                {% endif %}

              </td>

              <!-- Размер товара -->

              <td>

                {% if item.size %}

                  {{ item.size.name }}

                {% else %}

                  —  <!-- Если размер не указан -->

                {% endif %}

              </td>

              <!-- Количество -->

              <td>{{ item.quantity }}</td>

              <!-- Цена за единицу -->

              <td>{{ item.price }} ₽</td>

              <!-- Общая стоимость позиции -->

              <td>{{ item.total\_price }} ₽</td>

            </tr>

          {% endfor %}

        </tbody>

      </table>

      <!-- Общая сумма заказа -->

      <p class="order-total">Общая сумма: <strong>{{ order.total\_price }} ₽</strong></p>

    </div>

  {% empty %}

    <!-- Сообщение, если заказов нет -->

    <p>У вас пока нет заказов.</p>

  {% endfor %}

</div>

{% endblock %}

Листинг А.10 – Код страницы авторизации

{% extends 'base.html' %}

{% load static %}

{% block title %}Страница авторизации{% endblock %}

{% block content %}

<!-- Основной контейнер с центрированием по вертикали и горизонтали -->

<div class="d-flex justify-content-center align-items-center">

  <!-- Обертка для формы авторизации -->

  <div class="contact-wrapper">

    <!-- Блок для вывода ошибок авторизации -->

    {% if form.errors %}

      <div class="alert alert-danger" role="alert">

        Неверный логин или пароль. Попробуйте снова!

      </div>

    {% endif %}

    <!-- Заголовок формы -->

    <header class="login-cta">

      <h2>Авторизоваться</h2>

    </header>

    <!-- Форма авторизации -->

    <form method='post' action="{% url 'login' %}">

      {% csrf\_token %}  <!-- CSRF-токен для защиты формы -->

      <!-- Поле для ввода логина -->

      <div class="form-row">

        <input type="text"

               name='username'

               class="form-control"

               placeholder="Username"

               required>  <!-- Обязательное поле -->

      </div>

      <!-- Поле для ввода пароля -->

      <div class="form-row">

        <input type="password"

               name='password'

               class="form-control"

               placeholder="Password"

               required>  <!-- Обязательное поле -->

      </div>

      <!-- Кнопка отправки формы -->

      <div class="form-row">

        <button type="submit" class="btn">Войти!</button>

      </div>

      <!-- Кнопка перехода к регистрации  -->

      <div class="form-row">

        <button type="button" onclick="window.location.href='{% url 'register' %}'" class="btn">Зарегестрироваться</button>

      </div>

    </form>

  </div>

</div>

{% endblock %}

Листинг А.11 – Код страницы регистрации

{% extends 'base.html' %}

{% load static %}

{% block title %}Регистрация{% endblock %}

{% block content %}

<div class="d-flex justify-content-center align-items-center">

  <div class="contact-wrapper">

    <!-- Заголовок -->

    <header class="login-cta">

      <h2>Регистрация</h2>

    </header>

    <!-- Отображение ошибок -->

    {% if user\_form.errors or profile\_form.errors %}

      <div class="alert alert-danger" role="alert">

        Пожалуйста, исправьте ошибки в форме и попробуйте снова.

      </div>

    {% endif %}

    <!-- Форма регистрации -->

    <form method="post" action="{% url 'register' %}">

      {% csrf\_token %}

      <!-- Поля из формы пользователя -->

      <div class="form-row">

        {{ user\_form.username.label\_tag }}

        {{ user\_form.username }}

        {% if user\_form.username.errors %}

          <div class="text-danger">{{ user\_form.username.errors }}</div>

        {% endif %}

      </div>

      <div class="form-row">

        {{ user\_form.password.label\_tag }}

        {{ user\_form.password }}

        {% if user\_form.password.errors %}

          <div class="text-danger">{{ user\_form.password.errors }}</div>

        {% endif %}

      </div>

      <!-- Поля профиля -->

      <div class="form-row">

        {{ profile\_form.first\_name.label\_tag }}

        {{ profile\_form.first\_name }}

        {% if profile\_form.first\_name.errors %}

          <div class="text-danger">{{ profile\_form.first\_name.errors }}</div>

        {% endif %}

      </div>

      <div class="form-row">

        {{ profile\_form.second\_name.label\_tag }}

        {{ profile\_form.second\_name }}

        {% if profile\_form.second\_name.errors %}

          <div class="text-danger">{{ profile\_form.second\_name.errors }}</div>

        {% endif %}

      </div>

      <div class="form-row">

        {{ profile\_form.third\_name.label\_tag }}

        {{ profile\_form.third\_name }}

        {% if profile\_form.third\_name.errors %}

          <div class="text-danger">{{ profile\_form.third\_name.errors }}</div>

        {% endif %}

      </div>

      <div class="form-row">

        {{ profile\_form.phone.label\_tag }}

        {{ profile\_form.phone }}

        {% if profile\_form.phone.errors %}

          <div class="text-danger">{{ profile\_form.phone.errors }}</div>

        {% endif %}

      </div>

      <div class="form-row">

        {{ profile\_form.email.label\_tag }}

        {{ profile\_form.email }}

        {% if profile\_form.email.errors %}

          <div class="text-danger">{{ profile\_form.email.errors }}</div>

        {% endif %}

      </div>

      <div class="form-row">

        {{ profile\_form.date\_birth.label\_tag }}

        {{ profile\_form.date\_birth }}

        {% if profile\_form.date\_birth.errors %}

          <div class="text-danger">{{ profile\_form.date\_birth.errors }}</div>

        {% endif %}

      </div>

      <!-- Кнопка отправки -->

      <div class="form-row">

        <button type="submit" class="btn">Зарегистрироваться</button>

      </div>

      <!-- Кнопка входа -->

      <div class="form-row">

        <button type="button" onclick="window.location.href='{% url 'login' %}'" class="btn">Авторизоваться</button>

      </div>

    </form>

  </div>

</div>

{% endblock %}

Листинг А.12 – Код страницы редактирования товаров

{% extends 'base.html' %}

{% load static %}

{% block title %}Управление товарами{% endblock %}

{% block content %}

<div class="container mt-5">

  <h2>Управление товарами</h2>

  <div class="text-end mb-3">

    <button class="btn " data-bs-toggle="modal" data-bs-target="#addProductModal">Добавить товар</button>

  </div>

  <ul class="list-group">

    {% for product in products %}

    <li class="list-group-item d-flex justify-content-between align-items-center">

      {{ product.name }} — {{ product.price }}₽

      <button class="btn" data-bs-toggle="modal" data-bs-target="#editProductModal{{ product.id }}">Редактировать</button>

    </li>

    <!-- Модальное окно редактирования -->

    <div class="modal fade" id="editProductModal{{ product.id }}" tabindex="-1">

      <div class="modal-dialog modal-lg">

        <div class="modal-content">

          <form method="post" action="{% url 'update\_product' product.id %}" enctype="multipart/form-data">

            {% csrf\_token %}

            <div class="modal-header">

              <h5 class="modal-title">Редактировать товар</h5>

              <button type="button" class="btn-close" data-bs-dismiss="modal"></button>

            </div>

            <div class="modal-body">

              <div class="row">

                <div class="col-md-6 mb-2">

                  <label class="form-label">Название</label>

                  <input type="text" name="name" value="{{ product.name }}" class="form-control" required>

                </div>

                <div class="col-md-6 mb-2">

                  <label class="form-label">Цена</label>

                  <input type="number" name="price" value="{{ product.price }}" class="form-control" required>

                </div>

              </div>

              <label class="form-label">Описание</label>

              <textarea name="description" class="form-control mb-3">{{ product.description }}</textarea>

              <label class="form-label">Цвета</label>

              <select name="colors" multiple class="form-select mb-2">

                {% for color in colors %}

                <option value="{{ color.id }}" {% if color in product.colors.all %}selected{% endif %}>{{ color.name }}</option>

                {% endfor %}

              </select>

              <div class="input-group mb-3">

                <input type="text" name="new\_color\_name" class="form-control" placeholder="Новый цвет">

                <input type="color" name="new\_color\_hex" class="form-control form-control-color">

              </div>

              <label class="form-label">Размеры</label>

              <select name="sizes" multiple class="form-select mb-2">

                {% for size in sizes %}

                <option value="{{ size.id }}" {% if size in product.sizes.all %}selected{% endif %}>{{ size.name }}</option>

                {% endfor %}

              </select>

              <input type="text" name="new\_size\_name" class="form-control mb-3" placeholder="Новый размер">

              <label class="form-label">Изображения (выбери, чтобы заменить)</label>

              <input type="file" name="images" multiple class="form-control mb-3">

              <div class="mb-2">

                <small>Текущие изображения:</small><br>

                {% for img in product.images.all %}

                  <img src="{{ img.image.url }}" class="img-thumbnail me-1" width="70">

                {% endfor %}

              </div>

            </div>

            <div class="modal-footer">

              <button type="submit" class="btn ">Сохранить изменения</button>

            </div>

          </form>

        </div>

      </div>

    </div>

    {% endfor %}

  </ul>

</div>

<!-- Модальное окно добавления -->

<div class="modal fade" id="addProductModal" tabindex="-1">

  <div class="modal-dialog modal-lg">

    <div class="modal-content">

      <form method="post" action="{% url 'add\_product' %}" enctype="multipart/form-data">

        {% csrf\_token %}

        <div class="modal-header">

          <h5 class="modal-title">Добавить товар</h5>

          <button type="button" class="btn-lose" data-bs-dismiss="modal"></button>

        </div>

        <div class="modal-body">

          <div class="row">

            <div class="col-md-6 mb-2">

              <label class="form-label">Название</label>

              <input type="text" name="name" class="form-control" required>

            </div>

            <div class="col-md-6 mb-2">

              <label class="form-label">Цена</label>

              <input type="number" name="price" class="form-control" required>

            </div>

          </div>

          <label class="form-label">Описание</label>

          <textarea name="description" class="form-control mb-3"></textarea>

          <label class="form-label">Цвета</label>

          <select name="colors" multiple class="form-select mb-2">

            {% for color in colors %}

            <option value="{{ color.id }}">{{ color.name }}</option>

            {% endfor %}

          </select>

          <div class="input-group mb-3">

            <input type="text" name="new\_color\_name" class="form-control" placeholder="Новый цвет">

            <input type="color" name="new\_color\_hex" class="form-control form-control-color">

          </div>

          <label class="form-label">Размеры</label>

          <select name="sizes" multiple class="form-select mb-2">

            {% for size in sizes %}

            <option value="{{ size.id }}">{{ size.name }}</option>

            {% endfor %}

          </select>

          <input type="text" name="new\_size\_name" class="form-control mb-3" placeholder="Новый размер">

          <label class="form-label">Изображения</label>

          <input type="file" name="images" multiple class="form-control">

        </div>

        <div class="modal-footer">

          <button type="submit" class="btn btn-success">Добавить товар</button>

        </div>

      </form>

    </div>

  </div>

</div>

{% endblock %}

Листинг А.13 – Код страницы работы с заказами

{% extends 'base.html' %}

{% load static %}

{% block title %} Управление заказами {% endblock %}

{% block content %}

<div class="profile-container">

  <h1 class="profile-title">📦 Управление заказами</h1>

  <!-- Форма фильтрации по статусу -->

  <form method="get" class="row g-3 align-items-end mb-4">

    <div class="col-md-3">

      <label for="status" class="form-label">Фильтр по статусу:</label>

      <select name="status" id="status" class="form-select">

        <option value="">Все</option>

        <option value="0">Новая</option>

        <option value="1">В Обработке</option>

        <option value="2">Принята</option>

        <option value="3">Завершена</option>

        <option value="4">Отменена</option>

      </select>

    </div>

    <div class="col-md-auto">

      <button type="submit" class="btn">Применить фильтр</button>

    </div>

  </form>

  <!-- Список заказов -->

  {% for order in orders %}

    <div class="order-block mb-4 border p-3 rounded shadow-sm">

      <!-- Сводка по заказу -->

      <div id="summary-block-{{ order.id }}">

        <div class="d-flex justify-content-between align-items-center">

          <div>

            <p><strong>Заказ №{{ order.id }}</strong> от {{ order.date\_at|date:"d.m.Y" }}</p>

            <p><strong>Пользователь:</strong> {{ order.user.username }}</p>

            <p><strong>Статус:</strong> {{ order.get\_status\_display }}</p>

          </div>

          <div id="toggle-button-show-{{ order.id }}">

            <button class="btn" onclick="toggleDetails({{ order.id }})">Подробнее</button>

          </div>

        </div>

      </div>

      <!-- Подробности заказа (скрыты по умолчанию) -->

      <div id="details-block-{{ order.id }}" style="display: none;">

        <div class="d-flex justify-content-between align-items-center mb-2">

          <div>

            <p><strong>ФИО:</strong> {{ user.last\_name }} {{ user.first\_name }} {{ user.third\_name }}</p>

            <p><strong>Контактные данные:</strong> {{ user.phone|default:"—" }}</p>

            <p><strong>Оплата:</strong> {{ order.get\_type\_payment\_display }}</p>

            <p><strong>Пункт выдачи:</strong> {{ order.address|default:"—" }}</p>

          </div>

          <div id="toggle-buttons-hide-{{ order.id }}" class="d-flex gap-2" style="display: none;">

            <button class="btn" onclick="toggleDetails({{ order.id }})">Скрыть</button>

            <button class="btn" data-bs-toggle="modal" data-bs-target="#editModal-{{ order.id }}">Изменить</button>

          </div>

        </div>

        <!-- Таблица товаров -->

        <table class="table">

          <thead>

            <tr>

              <th>Товар</th>

              <th>Цвет</th>

              <th>Размер</th>

              <th>Кол-во</th>

              <th>Цена</th>

              <th>Итого</th>

            </tr>

          </thead>

          <tbody>

            {% for item in order.items.all %}

              <tr>

                <td>{{ item.product.name }}</td>

                <td>

                  {% if item.color %}

                    <span class="color-circle" style="background-color: {{ item.color.hex\_code }}"></span>

                    {{ item.color.name }}

                  {% else %}—{% endif %}

                </td>

                <td>{% if item.size %}{{ item.size.name }}{% else %}—{% endif %}</td>

                <td>{{ item.quantity }}</td>

                <td>{{ item.price }} ₽</td>

                <td>{{ item.total\_price }} ₽</td>

              </tr>

            {% endfor %}

          </tbody>

        </table>

        <p class="order-total">Общая сумма: <strong>{{ order.total\_price }} ₽</strong></p>

      </div>

    </div>

    <!-- Модальное окно изменения статуса -->

    <div class="modal fade" id="editModal-{{ order.id }}" tabindex="-1" aria-labelledby="editModalLabel-{{ order.id }}" aria-hidden="true">

      <div class="modal-dialog">

        <div class="modal-content">

          <form method="post" action="{% url 'update\_order\_status' order.id %}" onsubmit="return validateCancelReason({{ order.id }});">

            {% csrf\_token %}

            <div class="modal-header">

              <h5 class="modal-title" id="editModalLabel-{{ order.id }}">Изменение статуса заказа</h5>

              <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Закрыть"></button>

            </div>

            <div class="modal-body">

              <label for="status-select-{{ order.id }}" class="form-label">Статус:</label>

              <select name="status" id="status-select-{{ order.id }}" class="form-select" onchange="handleStatusChange({{ order.id }});">

                {% for value, label in order.STATUS\_CHOICES %}

                  <option value="{{ value }}" {% if value == order.status %}selected{% endif %}>{{ label }}</option>

                {% endfor %}

              </select>

              <!-- Блок причины отмены (появляется, если выбран статус "Отменён") -->

              <div id="cancel-reason-block-{{ order.id }}" class="mt-3" style="display: none;">

                <label for="cancel-reason-text-{{ order.id }}" class="form-label">Причина отмены:</label>

                <textarea name="cancel\_reason" id="cancel-reason-text-{{ order.id }}" class="form-control" rows="3" placeholder="Укажите причину отмены..."></textarea>

                <div class="invalid-feedback">Пожалуйста, укажите причину отмены.</div>

              </div>

            </div>

            <div class="modal-footer">

              <button type="button" class="btn" data-bs-dismiss="modal">Отменить</button>

              <button type="submit" class="btn">Сохранить</button>

            </div>

          </form>

        </div>

      </div>

    </div>

  {% endfor %}

  {% if orders|length == 0 %}

    <p class="text-muted">Нет заказов</p>

  {% endif %}

</div>

{% endblock %}

{% block scripts %}

{{ block.super }}

<!-- Скрипты управления отображением и проверками -->

<script>

  // Переключение подробностей заказа

  function toggleDetails(id) {

    const summary = document.getElementById(`summary-block-${id}`);

    const details = document.getElementById(`details-block-${id}`);

    const showBtn = document.getElementById(`toggle-button-show-${id}`);

    const hideBtns = document.getElementById(`toggle-buttons-hide-${id}`);

    const isOpen = details.style.display === 'block';

    if (isOpen) {

      details.style.display = 'none';

      showBtn.style.display = 'block';

      hideBtns.style.display = 'none';

    } else {

      details.style.display = 'block';

      showBtn.style.display = 'none';

      hideBtns.style.display = 'flex';

    }

  }

  // Показывать/скрывать причину отмены в зависимости от выбранного статуса

  function handleStatusChange(orderId) {

    const select = document.getElementById(`status-select-${orderId}`);

    const cancelBlock = document.getElementById(`cancel-reason-block-${orderId}`);

    const cancelReason = document.getElementById(`cancel-reason-text-${orderId}`);

    if (select.value === '4') {

      cancelBlock.style.display = 'block';

    } else {

      cancelBlock.style.display = 'none';

      cancelReason.classList.remove('is-invalid');

      cancelReason.value = '';

    }

  }

  // Валидация причины отмены при отправке формы

  function validateCancelReason(orderId) {

    const select = document.getElementById(`status-select-${orderId}`);

    const cancelReason = document.getElementById(`cancel-reason-text-${orderId}`);

    if (select.value === '4') {

      if (cancelReason.value.trim() === '') {

        cancelReason.classList.add('is-invalid');

        cancelReason.focus();

        return false;

      } else {

        cancelReason.classList.remove('is-invalid');

      }

    }

    return true;

  }

</script>

{% endblock %}

Листинг А.14 – Код работы представлений

from django.contrib.auth import logout, login  
from django.contrib.auth.decorators import login\_required  
from django.db.models import Max, Prefetch  
from django.shortcuts import render, redirect, get\_object\_or\_404  
from django.http import JsonResponse, HttpResponse  
from django.views.decorators.http import require\_POST  
from django.contrib import messages  
from django.contrib.admin.views.decorators import staff\_member\_required  
from general.forms import OrderForm, ProductForm, ProductVariationForm, UserForm, UserProfileForm  
from general.models import Product, Color, Size, ProductImage, Order, Cart, CartItem, ProductVariation, OrderItem  
# страницы  
# главная страница  
def index(request):  
 # Выявление последних 6 вариаций (тип продукта + цвет)  
 latest\_image\_ids = (  
 ProductImage.objects  
 .values('product', 'color') # Группировка по продукту и цвету  
 .annotate(latest\_image\_id=Max('id')) # Берём ID последнего изображения в группе  
 .order\_by('-latest\_image\_id')[:6] # Сортируем по новизне и берём 6  
 )  
 # 2. Получаем полные объекты ProductImage по найденным ID  
 image\_ids = [item['latest\_image\_id'] for item in latest\_image\_ids]  
 products = ProductImage.objects.filter(id\_\_in=image\_ids).select\_related('product', 'color')  
 context = {'products': products}  
 return render(request, 'index.html', context)  
# каталог  
def catalog(request):  
 product\_images = []  
 images = ProductImage.objects.select\_related('product', 'color')  
 for image in images:  
 # Получаем все вариации с тем же продуктом и цветом  
 variations = ProductVariation.objects.filter(product=image.product, color=image.color).prefetch\_related('size')  
 # Собираем размеры  
 sizes = set()  
 for var in variations:  
 sizes.update(var.size.all())  
 # Собираем все цвета для этого продукта  
 all\_variations = ProductVariation.objects.filter(product=image.product).select\_related('color')  
 color\_set = set(v.color for v in all\_variations)  
 product\_images.append({  
 'product': image.product,  
 'image': image,  
 'color': image.color,  
 'sizes': sorted(sizes, key=lambda s: s.name),  
 'colors': sorted(color\_set, key=lambda c: c.name),  
 })  
 return render(request, 'catalog.html', {  
 'product\_images': product\_images  
 })  
# страница товара  
def product\_detail(request, pk):  
 product = get\_object\_or\_404(Product, pk=pk)  
 return render(request, 'product\_detail.html', {'product': product})  
# профиль пользователя  
def profile(request):  
 orders = Order.objects.filter(user=request.user).prefetch\_related(  
 'items\_\_product', 'items\_\_color', 'items\_\_size'  
 ).select\_related('address').order\_by('-date\_at')  
 return render(request, 'profile.html', {  
 'orders': orders,  
 })  
# админка управления заказами  
def admin\_order(request):  
 status = request.GET.get('status')  
 orders = Order.objects.all().order\_by('-date\_at')  
 if status:  
 orders = orders.filter(status=status)  
 return render(request, 'admin\_order.html', {'orders': orders})  
# админка работа с товарами  
def admin\_product(request):  
 products = Product.objects.all()  
 colors = Color.objects.all()  
 sizes = Size.objects.all()  
 return render(request, 'admin\_product.html', {  
 'products': products,  
 'colors': colors,  
 'sizes': sizes  
 })  
# страница товара  
def product\_detail(request, product\_id, color\_id):  
 product = get\_object\_or\_404(Product, pk=product\_id)  
 variation = get\_object\_or\_404(ProductVariation, product=product, color\_\_id=color\_id)  
 images = ProductImage.objects.filter(product=product, color\_\_id=color\_id)  
 sizes = variation.size.all()  
 context = {  
 'product': product,  
 'variation': variation,  
 'images': images,  
 'sizes': sizes,  
 }  
 return render(request, 'product\_detail.html', context)  
# о нас  
def about(request):  
 return render(request, 'about.html')  
# регистрация  
def register(request):  
 if request.method == 'POST':  
 user\_form = UserForm(request.POST)  
 profile\_form = UserProfileForm(request.POST)  
 if user\_form.is\_valid() and profile\_form.is\_valid():  
 user = user\_form.save(commit=False)  
 user.set\_password(user.password) # хеширование пароля  
 user.save()  
 profile = profile\_form.save(commit=False)  
 profile.user = user  
 profile.save()  
 login(request, user) # автоматически входить после регистрации  
 return redirect('profile')   
 else:  
 user\_form = UserForm()  
 profile\_form = UserProfileForm()  
 return render(request, 'register.html', {  
 'user\_form': user\_form,  
 'profile\_form': profile\_form  
 })  
# корзина  
# страница корзины  
@login\_required  
def cart\_view(request):  
 # Загружаем все элементы корзины текущего пользователя  
 items = CartItem.objects.filter(cart\_\_user=request.user).select\_related('product', 'color', 'size')  
 # Получаем последние изображения по сочетаниям product + color  
 latest\_images = (  
 ProductImage.objects  
 .values('product', 'color')  
 .annotate(latest\_id=Max('id'))  
 .values\_list('latest\_id', flat=True)  
 )  
 images = (  
 ProductImage.objects  
 .filter(id\_\_in=latest\_images)  
 .select\_related('product', 'color')  
 )  
 # Создаём словарь (product\_id, color\_id) → image  
 image\_map = {  
 (img.product.id, img.color.id): img  
 for img in images  
 }  
 # Назначаем каждому item его картинку  
 for item in items:  
 key = (item.product.id, item.color.id if item.color else None)  
 item.image = image\_map.get(key)  
 # Общая сумма  
 total\_price = sum(item.total\_price for item in items)  
  
 return render(request, 'cart.html', {  
 'items': items,  
 'cart': {  
 'total\_price': total\_price  
 }  
 })  
# добавление товара в корзину  
@require\_POST  
@login\_required  
def add\_to\_cart(request):  
 try:  
 product\_id = request.POST.get('product\_id')  
 color\_id = request.POST.get('color')  
 size\_id = request.POST.get('size')  
 product = get\_object\_or\_404(Product, id=product\_id)  
 color = get\_object\_or\_404(Color, id=color\_id) if color\_id else None  
 size = get\_object\_or\_404(Size, id=size\_id) if size\_id else None  
 cart, created = Cart.objects.get\_or\_create(user=request.user)  
 cart\_item, created = CartItem.objects.get\_or\_create(  
 cart=cart,  
 product=product,  
 color=color,  
 size=size,  
 defaults={'quantity': 1}  
 )  
 if not created:  
 cart\_item.quantity += 1  
 cart\_item.save()  
 return JsonResponse({  
 'success': True,  
 'message': 'Товар добавлен в корзину',  
 'cart\_count': cart.total\_items  
 })  
 except Exception as e:  
 return JsonResponse({  
 'success': False,  
 'error': str(e)  
 }, status=400)  
# удаление товара  
@login\_required  
def remove\_from\_cart(request, item\_id):  
 cart\_item = get\_object\_or\_404(CartItem, id=item\_id, cart\_\_user=request.user)  
 cart\_item.delete()  
 return redirect('cart')  
# получение размера  
def get\_sizes(request):  
 product\_id = request.GET.get('product\_id')  
 color\_id = request.GET.get('color\_id')  
 # Здесь логика получения доступных размеров для товара и цвета  
 # Например:  
 try:  
 product\_color = ProductVariation.objects.get(product\_id = product\_id, color\_id = color\_id)  
 sizes = product\_color.size.all()  
 sizes\_data = [{'id': size.id, 'name': size.name} for size in sizes]  
 return JsonResponse({'sizes': sizes\_data})  
 except ProductVariation.DoesNotExist:  
 return JsonResponse({'sizes': []})  
# получение цвета  
def switch\_color(request):  
 product\_id = request.GET.get('product\_id')  
 color\_id = request.GET.get('color\_id')  
 product = get\_object\_or\_404(Product, id=product\_id)  
 variation = product.productvariation\_set.filter(color\_id=color\_id).first()  
 if not variation:  
 return HttpResponse("Цвет не найден", status=404)  
 image = variation.productimage\_set.first  
 context = {  
 'image': image # предполагаем, что image содержит ссылку на product и color  
 }  
 return render(request, 'product\_card.html', context)  
# выход  
def logout\_view(request):  
 logout(request)  
 return redirect('index')  
# создание заказа  
@login\_required  
def create\_order(request):  
 selected\_items\_ids = request.session.get("selected\_items")  
 if not selected\_items\_ids:  
 messages.error(request, "Вы не выбрали товары для оформления.")  
 return redirect("cart")  
 selected\_items = CartItem.objects.filter(id\_\_in=selected\_items\_ids, cart\_\_user=request.user)  
 total\_price = sum(item.total\_price for item in selected\_items)  
 if request.method == "POST":  
 form = OrderForm(request.POST)  
 if form.is\_valid():  
 order = form.save(commit=False)  
 order.user = request.user  
 order.save()  
 for item in selected\_items:  
 OrderItem.objects.create(  
 order=order,  
 product=item.product,  
 color=item.color,  
 size=item.size,  
 quantity=item.quantity,  
 price=item.product.price  
 )  
 selected\_items.delete()  
 del request.session["selected\_items"]  
 return redirect("order\_success", order\_id=order.id)  
 else:  
 print("Ошибки формы:", form.errors)  
 else:  
 form = OrderForm()  
 return render(request, "create\_order.html", {  
 "form": form,  
 "selected\_items": selected\_items,  
 "total\_price": total\_price,  
 })  
# выбор товаров в заказ  
def proceed\_to\_order(request):  
 if request.method == "POST":  
 selected\_items = request.POST.getlist("selected\_items")  
 if not selected\_items:  
 messages.warning(request, "Вы не выбрали товары для оформления.")  
 return redirect("cart")  
 request.session["selected\_items"] = selected\_items  
 return redirect("create\_order") # URL на create\_order\_view  
 return redirect("cart")  
# изменение количество товара  
@login\_required  
def update\_cart\_item(request, item\_id):  
 if request.method == 'POST':  
 cart\_item = get\_object\_or\_404(CartItem, id=item\_id, cart\_\_user=request.user)  
 quantity = int(request.POST.get('quantity', 1))  
 if quantity > 0:  
 cart\_item.quantity = quantity  
 cart\_item.save()  
 else:  
 cart\_item.delete()  
 if request.headers.get('x-requested-with') == 'XMLHttpRequest':  
 cart = cart\_item.cart  
 return JsonResponse({  
 'success': True,  
 'cart\_total': cart.total\_items,  
 'item\_total': cart\_item.total\_price,  
 'cart\_total\_price': cart.total\_price  
 })  
# успешный заказ  
def order\_success(request, order\_id):  
 order = get\_object\_or\_404(Order, id=order\_id, user=request.user)  
 return render(request, "order\_success.html", {"order": order})  
# работа в админке  
# изменение статуса заказа  
@login\_required  
def update\_order\_status(request, order\_id):  
 order = get\_object\_or\_404(Order, id=order\_id)  
 if request.method == 'POST':  
 new\_status = request.POST.get('status')  
 cancel\_reason = request.POST.get('cancel\_reason', '').strip()  
 if new\_status in dict(Order.STATUS\_CHOICES):  
 order.status = new\_status  
 if new\_status == '4':  
 if not cancel\_reason:  
 messages.error(request, "Пожалуйста, укажите причину отмены заказа.")  
 return redirect(request.META.get('HTTP\_REFERER', '/'))  
 order.cancel\_reason = cancel\_reason  
 else:  
 order.cancel\_reason = ''  
 order.save()  
 messages.success(request, f"Статус заказа №{order.id} успешно обновлён.")  
 else:  
 messages.error(request, "Недопустимый статус.")  
 return redirect(request.META.get('HTTP\_REFERER', '/'))  
# создание нового товара  
@login\_required  
def add\_product(request):  
 if request.method == 'POST':  
 name = request.POST.get('name')  
 description = request.POST.get('description')  
 price = request.POST.get('price')  
 # Новый цвет  
 new\_color\_name = request.POST.get('new\_color\_name')  
 new\_color\_hex = request.POST.get('new\_color\_hex')  
 # Новый размер  
 new\_size\_name = request.POST.get('new\_size\_name')  
 # Создаем продукт  
 product = Product.objects.create(  
 name=name,  
 description=description,  
 price=price  
 )  
 # Обработка цвета  
 color\_ids = request.POST.getlist('colors')  
 if new\_color\_name and new\_color\_hex:  
 color = Color.objects.create(name=new\_color\_name, hex\_code=new\_color\_hex)  
 color\_ids.append(str(color.id))  
 colors = Color.objects.filter(id\_\_in=color\_ids)  
 # Обработка размеров  
 size\_ids = request.POST.getlist('sizes')  
 if new\_size\_name:  
 size = Size.objects.create(name=new\_size\_name)  
 size\_ids.append(str(size.id))  
 sizes = Size.objects.filter(id\_\_in=size\_ids)  
  
 # Создаем вариацию (одну для всех сочетаний)  
 variation = ProductVariation.objects.create(product=product)  
 variation.color.set(colors)  
 variation.size.set(sizes)  
 # Обработка изображений  
 images = request.FILES.getlist('images')  
 for color in colors:  
 for image in images:  
 ProductImage.objects.create(  
 product=product,  
 color=color,  
 image=image  
 )  
 return redirect('admin\_product')  
 return redirect('admin\_product')  
# изменение существующего товара  
@login\_required  
def update\_product(request, pk):  
 product = get\_object\_or\_404(Product, pk=pk)  
 if request.method == 'POST':  
 product.name = request.POST.get('name')  
 product.description = request.POST.get('description')  
 product.price = request.POST.get('price')  
 product.save()  
 # Новый цвет  
 new\_color\_name = request.POST.get('new\_color\_name')  
 new\_color\_hex = request.POST.get('new\_color\_hex')  
 color\_ids = request.POST.getlist('colors')  
 if new\_color\_name and new\_color\_hex:  
 color = Color.objects.create(name=new\_color\_name, hex\_code=new\_color\_hex)  
 color\_ids.append(str(color.id))  
 colors = Color.objects.filter(id\_\_in=color\_ids)  
 # Новый размер  
 new\_size\_name = request.POST.get('new\_size\_name')  
 size\_ids = request.POST.getlist('sizes')  
 if new\_size\_name:  
 size = Size.objects.create(name=new\_size\_name)  
 size\_ids.append(str(size.id))  
 sizes = Size.objects.filter(id\_\_in=size\_ids)  
 # Удаляем старые вариации и изображения  
 ProductVariation.objects.filter(product=product).delete()  
 ProductImage.objects.filter(product=product).delete()  
 # Создаем новую вариацию  
 variation = ProductVariation.objects.create(product=product)  
 variation.color.set(colors)  
 variation.size.set(sizes)  
 # Обработка новых изображений  
 images = request.FILES.getlist('images')  
 for color in colors:  
 for image in images:  
 ProductImage.objects.create(  
 product=product,  
 color=color,  
 image=image  
 )  
 return redirect('admin\_product')  
 return redirect('admin\_product')

Листинг А.15 – Код работы переадресаций

from django.urls import path  
from django.contrib.auth import views as auth\_views  
from general.views import index, catalog, logout\_view, cart\_view, add\_to\_cart, remove\_from\_cart, update\_cart\_item  
from general.views import create\_order, get\_sizes, switch\_color, order\_success, proceed\_to\_order, register  
from general.views import profile, admin\_order, admin\_product, update\_order\_status, add\_product, update\_product, product\_detail, about  
urlpatterns = [  
 path('', index, name = 'index'),  
 path('catalog/', catalog, name='catalog'),  
 path('profile/', profile, name='profile'),  
 path('product\_detail/<int:pk>/', product\_detail, name='product\_detail'),  
 path('admin\_order/', admin\_order, name='admin\_order'),  
 path('admin\_product/', admin\_product, name='admin\_product'),  
 path('product\_detail/<int:product\_id>/<int:color\_id>/', product\_detail, name='product\_detail'),  
 path('about/', about, name='about'),  
 path('register/', register, name='register'),  
 path("accounts/login/", auth\_views.LoginView.as\_view(template\_name="login.html"), name='login'),  
 path('logout/', logout\_view, name='logout'),  
 path('get\_sizes/', get\_sizes, name='get\_sizes'),  
 path('switch\_color/', switch\_color, name='switch\_color'),  
 path('cart/', cart\_view, name='cart\_view'),  
 path('add\_to\_cart/', add\_to\_cart, name='add\_to\_cart'),  
 path('remove\_from\_cart/<int:item\_id>/', remove\_from\_cart, name='remove\_from\_cart'),  
 path('create\_order/', create\_order, name='create\_order'),  
 path('proceed\_to\_order/', proceed\_to\_order, name='proceed\_to\_order'),  
 path('update\_cart\_item/<int:item\_id>/', update\_cart\_item, name='update\_cart\_item'),  
 path('order\_success/<int:order\_id>/', order\_success, name='order\_success'),  
 path('update\_order\_status/<int:order\_id>/', update\_order\_status, name='update\_order\_status'),  
 path('add\_product', add\_product, name='add\_product'),  
 path('update\_product/<int:product\_id>/', update\_product, name='update\_product'),  
]

Листинг А.16 – Код моделей данных

from django.contrib.auth.models import User  
from django.db import models  
from django.urls import reverse  
from django.utils.html import format\_html  
class Color(models.Model):  
 name = models.CharField(max\_length=100)  
 hex\_code=models.CharField(max\_length=7, default='#FFFFFF')  
 def \_\_str\_\_(self):  
 return f"{self.name} ({self.hex\_code})"  
 def colored\_circle(self):  
 return format\_html(  
 '<span style="display: inline-block; width: 15px; height: 15px; border-radius: 50%; background: {}; border: 1px solid #ddd"></span> {}',  
 self.hex\_code,  
 self.name  
 )  
class Size(models.Model):  
 name = models.CharField(max\_length=100)  
 def \_\_str\_\_(self):  
 return self.name  
class Product(models.Model):  
 name = models.CharField(max\_length=100)  
 description = models.TextField()  
 price = models.DecimalField(max\_digits=10, decimal\_places=2)  
 created\_at = models.DateTimeField(auto\_now\_add=True)  
 def \_\_str\_\_(self):  
 return self.name  
 def get\_absolute\_url(self):  
 return reverse('product\_detail', args=[str(self.pk)])  
class ProductVariation(models.Model):  
 product = models.ForeignKey(Product, on\_delete=models.CASCADE)  
 color = models.ForeignKey(Color, on\_delete=models.CASCADE)  
 size = models.ManyToManyField(Size)  
 def \_\_str\_\_(self):  
 return f"{self.product.name} - {self.color.name}"  
class ProductImage(models.Model):  
 image = models.ImageField(upload\_to='product/')  
 product = models.ForeignKey(Product, on\_delete=models.CASCADE)  
 color = models.ForeignKey(Color, on\_delete=models.CASCADE)  
 def \_\_str\_\_(self):  
 return f"Изображение {self.product.name}.{self.color.name}"  
class UserProfile(models.Model):  
 user = models.OneToOneField(  
 User,  
 on\_delete=models.CASCADE  
 )  
 first\_name = models.CharField(max\_length=100)  
 second\_name = models.CharField(max\_length=100)  
 third\_name = models.CharField(max\_length=100, null=True, blank=True)  
 phone = models.CharField(max\_length=100)  
 email = models.EmailField(max\_length=100, null=True, blank=True)  
 date\_birth = models.DateField()  
class PickupPoint(models.Model):  
 address = models.CharField(max\_length=255, unique=True)  
 def \_\_str\_\_(self):  
 return self.address  
class Order(models.Model):  
 TYPE\_PAYMENT\_CHOICES = [  
 ("0", "Банковская карта "),  
 ("1", "Наличные"),  
 ]  
 STATUS\_CHOICES = [  
 ("0", "Новая"),  
 ("1", "В Обработке"),  
 ("2", "Принята"),  
 ("3", "Завершена"),  
 ("4", "Отменена"),  
 ]  
 user = models.ForeignKey(  
 User,  
 on\_delete=models.CASCADE  
 )  
 address = models.ForeignKey(  
 PickupPoint,  
 on\_delete=models.SET\_NULL,  
 null=True,  
 blank=False  
 )  
 type\_payment = models.CharField(max\_length=1, choices=TYPE\_PAYMENT\_CHOICES)  
 status = models.CharField(max\_length=1, choices=STATUS\_CHOICES, default='0')  
 date\_at = models.DateTimeField(auto\_now\_add=True, verbose\_name='Дата создания')  
 product = models.ManyToManyField(  
 Product,  
 related\_name='orders'  
 )  
 cancel\_reason = models.CharField(max\_length=200, null=True, blank=True)  
 def \_\_str\_\_(self):  
 return f"Заказ #{self.id} от {self.user.username}"  
 @property  
 def total\_price(self):  
 return sum(item.total\_price for item in self.items.all())  
class OrderItem(models.Model):  
 order = models.ForeignKey(Order, on\_delete=models.CASCADE, related\_name='items')  
 product = models.ForeignKey(Product, on\_delete=models.CASCADE)  
 quantity = models.PositiveIntegerField(default=1)  
 price = models.DecimalField(max\_digits=10, decimal\_places=2) # цена на момент заказа  
 color = models.ForeignKey(Color, on\_delete=models.SET\_NULL, null=True, blank=True)  
 size = models.ForeignKey(Size, on\_delete=models.SET\_NULL, null=True, blank=True)  
 @property  
 def total\_price(self):  
 return self.quantity \* self.price  
class Cart(models.Model):  
 user = models.OneToOneField(User, on\_delete=models.CASCADE)  
 created\_at = models.DateTimeField(auto\_now\_add=True)  
 updated\_at = models.DateTimeField(auto\_now=True)  
 def \_\_str\_\_(self):  
 return f"Корзина {self.user.username}"  
 @property  
 def total\_price(self):  
 return sum(item.total\_price for item in self.items.all())  
 @property  
 def total\_items(self):  
 return self.items.count()  
class CartItem(models.Model):  
 cart = models.ForeignKey(Cart, related\_name='items', on\_delete=models.CASCADE)  
 product = models.ForeignKey(Product, on\_delete=models.CASCADE)  
 color = models.ForeignKey(Color, on\_delete=models.SET\_NULL, null=True, blank=True)  
 size = models.ForeignKey(Size, on\_delete=models.SET\_NULL, null=True, blank=True)  
 quantity = models.PositiveIntegerField(default=1)  
 def \_\_str\_\_(self):  
 return f"{self.quantity} x {self.product.name}"  
 @property  
 def total\_price(self):  
 return self.product.price \* self.quantity

Листинг А.17 – Код отображения моделей данных в панели администратора

from django.contrib import admin  
from general.models import Color, Size, Product, ProductImage, UserProfile, Order, Cart, CartItem, ProductVariation, OrderItem, PickupPoint  
admin.site.register(Size)  
admin.site.register(Product)  
admin.site.register(ProductImage)  
admin.site.register(UserProfile)  
admin.site.register(Order)  
admin.site.register(Cart)  
admin.site.register(CartItem)  
admin.site.register(ProductVariation)  
admin.site.register(OrderItem)  
admin.site.register(PickupPoint)  
class ColorAdmin(admin.ModelAdmin):  
 list\_display = ('colored\_circle', 'name', 'hex\_code')  
 search\_fields = ('name',)  
admin.site.register(Color, ColorAdmin)

Листинг А.18 – Код работы форм

from django import forms  
from django.contrib.auth.models import User  
from .models import Order, PickupPoint, Product, Color, Size, UserProfile  
class OrderForm(forms.ModelForm):  
 """  
 Форма для оформления заказа.  
 Позволяет выбрать способ оплаты и пункт самовывоза.  
 """  
 class Meta:  
 model = Order # Используем модель Order  
 fields = ['type\_payment', 'address'] # Поля для отображения  
 widgets = {  
 'type\_payment': forms.RadioSelect, # Радиокнопки для выбора оплаты  
 'address': forms.Select, # Выпадающий список для пунктов выдачи  
 }  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 """  
 Инициализация формы с дополнительными настройками полей.  
 """  
 super().\_\_init\_\_(\*args, \*\*kwargs)  
 # Настройка поля выбора пункта самовывоза  
 self.fields['address'].queryset = PickupPoint.objects.all() # Все доступные пункты  
 self.fields['address'].required = True # Обязательное поле  
 self.fields['address'].label = "Выберите пункт самовывоза" # Кастомный лейбл  
 self.fields['address'].widget.attrs.update({'class': 'form-select'}) # CSS-класс  
 # Настройка поля выбора способа оплаты  
 self.fields['type\_payment'].required = True # Обязательное поле  
 self.fields['type\_payment'].label = "Способ оплаты" # Кастомный лейбл  
 self.fields['type\_payment'].widget.attrs.update({'class': 'form-check-input'}) # CSS-класс  
class ProductForm(forms.ModelForm):  
 """  
 Форма для создания/редактирования продукта.  
 Содержит основные поля продукта: название, описание и цену.  
 """  
 class Meta:  
 model = Product # Используем модель Product  
 fields = ['name', 'description', 'price'] # Поля для отображения  
 widgets = {  
 'name': forms.TextInput(attrs={'class': 'form-control'}),  
 'description': forms.Textarea(attrs={  
 'class': 'form-control',  
 'rows': 3 # Высота текстового поля  
 }),  
 'price': forms.NumberInput(attrs={'class': 'form-control'}),  
 }  
class ProductVariationForm(forms.Form):  
 """  
 Форма для выбора вариаций продукта (цвета и размеры).  
 Не привязана напрямую к модели, используется для обработки выбора.  
 """  
 # Поле для выбора цветов (множественный выбор)  
 color = forms.ModelMultipleChoiceField(  
 queryset=Color.objects.all(), # Все доступные цвета  
 widget=forms.CheckboxSelectMultiple, # Чекбоксы для выбора  
 label='Цвета' # Лейбл поля  
 )  
 # Поле для выбора размеров (множественный выбор)  
 size = forms.ModelMultipleChoiceField(  
 queryset=Size.objects.all(), # Все доступные размеры  
 widget=forms.CheckboxSelectMultiple, # Чекбоксы для выбора  
 label='Размеры' # Лейбл поля  
 )  
class UserForm(forms.ModelForm):  
 password = forms.CharField(widget=forms.PasswordInput)  
 class Meta:  
 model = User  
 fields = ['username', 'password']  
class UserProfileForm(forms.ModelForm):  
 class Meta:  
 model = UserProfile  
 fields = [  
 'first\_name',  
 'second\_name',  
 'third\_name',  
 'phone',  
 'email',  
 'date\_birth'  
 ]