СОДЕРЖАНИЕ

ВВЕДЕНИЕ

1 ОБЩИЙ РАЗДЕЛ

1.1 Краткая характеристика предприятия

1.2 Анализ и моделирование предметной области

1.3 Определение затрат на разработку продукта

1.4 Средства разработки проекта

2 ТЕХНИЧЕСКИЙ РАЗДЕЛ

2.1 Разработка технического задания

2.2 Проектирование и разработка ИТ-решения

2.3 Практическая реализация и тестирование

3 ОХРАНА ТРУДА И ТЕХНИКА БЕЗОПАСНОСТИ

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

ПРИЛОЖЕНИЯ

ВВЕДЕНИЕ

В современном мире интернет стал неотъемлемой частью жизни общества, бизнеса и коммуникации. Веб-сайты выступают ключевым инструментом представления информации, продвижения товаров и услуг, а также взаимодействия с пользователями. С развитием цифровых технологий требования к веб-ресурсам постоянно растут: сегодня недостаточно просто создать страницу с текстом — важно обеспечить удобный интерфейс, адаптивность, высокую скорость загрузки и безопасность.

Разработка сайта — это сложный процесс, включающий в себя множество этапов: от анализа целевой аудитории и проектирования структуры до верстки, программирования и тестирования. Современные веб-технологии, такие как HTML5, CSS3, JavaScript, React, Vue.js, Node.js, PHP, Python (Django/Flask) и другие, позволяют создавать динамичные, интерактивные и масштабируемые решения. Кроме того, важную роль играют системы управления контентом (CMS), такие как WordPress, Joomla и Bootstrap, которые ускоряют процесс разработки и упрощают поддержку сайтов.

Актуальность данной темы обусловлена несколькими факторами:

- Рост спроса на веб-разработку — компании всех уровней нуждаются в качественных сайтах для привлечения клиентов.

- Развитие новых технологий — появление фреймворков, облачных решений и инструментов автоматизации меняет подходы к созданию веб-приложений.

- Важность UX/UI-дизайна — пользователи стали более требовательными к удобству и визуальной составляющей сайтов.

Цель дипломного проекта: разработка сайта интернет-магазина по продаже товаров и услуг

1 ОБЩИЙ РАЗДЕЛ

1.1 Краткая характеристика предприятия

ИП Полищук А.В. – небольшой цветочный магазин, специализирующийся на розничной продаже свежих цветов, букетов, комнатных растений и сопутствующих товаров для флористики. Основная деятельность предприятия направлена на удовлетворение спроса частных и корпоративных клиентов, предлагая качественную продукцию и индивидуальный подход к оформлению заказов.

Основные направления деятельности:

- Продажа свежих цветов и букетов:

- Сезонные и экзотические цветы (розы, тюльпаны, хризантемы, лилии, орхидеи и др.).

- Готовые и индивидуальные букетные композиции.

- Подарочные наборы с цветами и дополнениями (мягкие игрушки, конфеты, открытки).

Комнатные растения и аксессуары:

- Декоративные растения в горшках.

- Кактусы, суккуленты, экзотические виды.

Флористические услуги:

- Составление букетов на заказ (свадьбы, юбилеи, корпоративные мероприятия).

- Оформление цветами праздников и событий.

- Доставка цветов по городу.

Основные задачи предприятия:

- Поддержание высокого качества цветочной продукции.

- Разнообразие ассортимента в соответствии с сезоном и трендами.

- Гибкая ценовая политика и акции для постоянных клиентов.

- Развитие услуг доставки и онлайн-заказов.

Клиенты и партнеры:

Основные клиенты – частные лица, желающие порадовать близких, а также корпоративные заказчики. Партнерами являются местные и региональные поставщики цветов, обеспечивающие свежесть и разнообразие ассортимента.

На предприятии используется следующее программное обеспечение:

1. Битрикс24:

- Основная система для учета товаров, управления продажами и контроля задач.

- Ведет базу клиентов, историю заказов и остатки цветов.

- Позволяет ставить задачи сотрудникам и отслеживать их выполнение.

2. 1С:Бухгалтерия:

- Автоматизирует бухгалтерский и налоговый учет.

- Интегрируется с Битрикс24 и онлайн-кассой для автоматического отражения продаж и расходов.

- Формирует отчеты для ИП и налоговой.

3. Яндекс.Доставка

- Организует доставку цветов клиентам.

- Автоматически строит маршруты для курьеров и уведомляет клиентов о статусе заказа.

4. Telegram:

- Для связи между сотрудниками.

Организационная структура ИП Полищук А.В (рисунок 1).

Основные элементы структуры:

1. Руководство:

- Осуществляет общее управление бизнесом.

- Контролирует финансовые потоки, закупки, ассортимент.

- Принимает ключевые решения по развитию магазина.

- Ведет переговоры с поставщиками и крупными клиентами.

2. Отдел продаж и обслуживания клиентов:

Продавцы-флористы:

- Консультируют покупателей.

- Формируют букеты и композиции на заказ.

- Оформляют заказы на доставку.

- Следят за витриной и выкладкой товара.

3. Отдел закупок и логистики

Ответственный за закупки (руководитель):

- Взаимодействует с поставщиками цветов и сопутствующих товаров.

- Контролирует свежесть и качество продукции.

- Оптимизирует складские запасы (особенно перед праздниками).

Курьер (заказывается через приложение):

- Доставляет заказы клиентам.

- Следит за сохранностью цветов при транспортировке.

4. Финансовый учет

Бухгалтер:

- Начисляет зарплату.

- Сдает отчетность.

- Контролирует расходы и доходы.

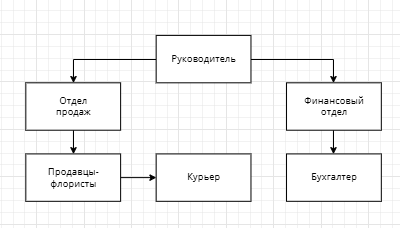


Рисунок 1 – Организационная структура предприятия

1.2 Анализ и моделирование предметной области

Создание сайта для цветочного бизнеса — это не просто размещение каталога товаров в интернете, а комплексное решение, которое помогает преодолеть ряд важных проблем, характерных для этой сферы. Рассмотрим основные из них:

1. Низкая видимость бизнеса и ограниченный охват аудитории

Проблема: Традиционный цветочный магазин работает только в пределах своего района, а реклама в соцсетях или локальных СМИ не всегда эффективна.

Решение:

- Сайт делает бизнес видимым 24/7, привлекая клиентов из разных районов города и даже других регионов (например, через доставку).

- SEO-оптимизация и контекстная реклама помогают находить новых клиентов, которые ищут цветы онлайн.

2. Сезонность спроса и необходимость быстрого реагирования

Проблема: Основные продажи цветов приходятся на праздники (8 Марта, 14 Февраля, День матери), но в остальное время трафик может падать.

Решение:

- Онлайн-заказы позволяют принимать предзаказы заранее, распределяя нагрузку

- Акции и email-рассылки стимулируют спрос в "несезон".

- Быстрое обновление ассортимента (например, добавление сезонных букетов) без перепечатывания бумажных каталогов.

3. Сложности с демонстрацией товара

Проблема: В офлайн-магазине клиент видит только часть ассортимента, а свежесть цветов не всегда очевидна.

Решение:

- Фото- и видео-презентации букетов с разных ракурсов.

- Описания с деталями (сорт цветов, упаковка, срок доставки).

- Раздел "Отзывы" с реальными фото клиентов повышает доверие.

4. Ограниченное время работы и потеря заказов

Проблема: Магазин не может принимать заказы ночью или в праздники, когда спрос максимален.

Решение:

- Круглосуточный прием заказов через сайт с возможностью выбора даты доставки.

- Чат-боты или онлайн-консультанты отвечают на частые вопросы автоматически.

5. Конкуренция с крупными маркетплейсами и агрегаторами

Проблема: Клиенты часто заказывают цветы через сервисы вроде Яндекс.Доставки, оставляя локальным магазинам минимальную прибыль.

Решение:

- Уникальный дизайн и удобство сайта создают конкурентное преимущество.

- Программы лояльности (скидки за повторные заказы, бонусы) удерживают клиентов.

У предприятия есть сайт, который выполняет свою основную функцию — демонстрирует товары и позволяет оформить заказ. Однако у него есть несколько недоработок, которые могут мешать клиентам и снижать эффективность работы:

1. Неработающие ссылки внизу страницы

2. Плохое отображение на телефонах

3. Нет описания для поисковиков

Поэтому необходимо переработать сайт с устранением недоработок.

При разработке сайта необходимо учитывать следующие нормативные документы:

1)ГОСТы:

- ГОСТ Р 52872-2019 (интернет-ресурсы),

- ГОСТ Р 57580 (безопасность платежей).

2)ISO:

- ISO 9241 (взаимодействие человека и машины),

- ISO/IEC 27001 (информационная безопасность).

3)Отраслевые стандарты:

- PCI DSS (для онлайн-платежей),

- ФЗ-152 (О персональных данных).

Требования к системе:

- Онлайн-каталог с фильтрами (по цене, типу цветов, поводу).

- Корзина с расчетом стоимости доставки.

- Личный кабинет (история заказов, бонусы).

- Панель администрирования (управление ассортиментом, заказами).

1.3 Определение затрат на разработку продукта

Весь комплекс разработки программного продукта можно разделить на этапы. Для каждого этапа указываются трудоемкость, количество исполнителей и продолжительность работ. Определяются даты начала и окончания работ. При этом длительность работ определяют по формуле 1:

 (1)

где Tц – длительность цикла, дней;

Q – трудоёмкость, человеко-дней;

R – количество исполнителей, чел.

Полученную информацию необходимо представить в таблице 1 .

Таблица 1 – Характеристика работ по разработке программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование работы | Трудоемкость | | Исполнители | Продолжи-тельность, дней |
| чел.–  дни | %к  итогу |
| 1. Анализ требований | 2 | 10 | техник-программист | 2 |
| 2. Проектирование интерфейса | 3 | 15 | техник-программист | 3 |
| 3. Верстка и фронтенд-разработка | 6 | 30 | техник-программист | 6 |
| 4. Бэкенд-разработка | 5 | 25 | техник-программист | 5 |
| 5. Тестирование и отладка | 2 | 10 | техник-программист | 2 |
| 6. Развертывание и настройка | 1 | 5 | техник-программист | 1 |
| 7. Написание документации | 1 | 5 | техник-программист | 1 |
| Итого | 20 | 100 | - | 20 |

По данным таблицы 1 составляется сводный ленточный график планирования разработки программного продукта, представляющий собой таблицу, в первом столбце которой размещены в порядке увеличения сроков начала выполнения все виды работы, а напротив – календарный период их выполнения.

Данный график приведен в таблице 2.

Таблица 2 - Сводный ленточный график планирования разработки

программного продукта

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Наименование работ | Календарный период, дни | | | | | | |
| 19.05.25 | 21.05.25 | 24.05.25 | 01.06.25 | 08.06.25 | 10.06.25 | 11.06.25 |
| 1. Анализ требований |  |  |  |  |  |  |  |
| 2. Проектирование |  |  |  |  |  |  |  |
| 3. Верстка и фронтенд |  |  |  |  |  |  |  |
| 4. Бэкенд-разработка |  |  |  |  |  |  |  |
| 5. Тестирование |  |  |  |  |  |  |  |
| 6. Развертывание |  |  |  |  |  |  |  |
| 7. Документация |  |  |  |  |  |  |  |

Для определения затрат на разработку программы составляется калькуляция сметной стоимости работ, которая включает следующие статьи:

* фонд заработной платы;
* отчисления во внебюджетные государственные фонды;
* амортизационные отчисления;
* ежемесячные материальные затраты

Затраты по этой статье состоят из планового фонда зарплаты всех категорий работников, занятых в разработке программы. Расчет заработной платы ведется на основании данных о трудоемкости, представленных в таблице 3.

Таблица 3 – Расчет фонда заработной платы

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Категории работников | Численность, чел. | Тарифная ставка,  оклад,  руб. | Сумма прямой заработной платы | В том числе | | Фонд оплаты труда, руб. |
| Премия,  руб. | Доплаты, руб. |
| техник-программист | 1 | 85000 | 60 000 | 12 750 | 8 500 | 106 250 |
| Всего: | 1 | 85000 | 60 000 | 12 750 | 8 500 | 106 250 |

Фонд заработной платы при повременной системе определяется по формуле:

 (2)

где Тст- тарифная ставка или должностной оклад, руб.;

Ч.- численность работников, чел;

Дп.- различные виды доплат, которые составляют 10% от суммы прямой заработной платы, руб.;

Зпр=85 000+12 750+8 500=106 250руб;

Фонд оплаты труда составил:106 250руб.

Определим сумму доплат к заработной плате

(3)

где Дп- размер доплат к заработной плате, руб. %;

3пр- прямой фонд оплаты труда, руб.

Дп=85 000\*0.1=8 500руб.

Сумма доплат к заработной плате составил:8 500руб.

Сп- сумма премии, которая составляет 15 % от суммы прямой заработной платы, руб. %.

Определим сумму премии

 (4)

где Сп- размер премии за достигнутые результаты, руб.;

3пр- прямой фонд оплаты труда, руб.

Сп=85000\*0,15=12750руб,

Размер премии за достигнутые результаты составляет: 12750руб.

Отчисления во внебюджетные государственные фонды составляют в 2025 году –30 % от суммы основной и дополнительной заработной платы по формуле:

 (5)

где Зсв – сумма отчислений во внебюджетные государственные фонды, руб.;

Зпл - фонд заработной платы, руб.

Зсв=0.30\*106 250руб=31875 руб.

Таким образом сумма отчислений во внебюджетные государственные фонды составляет: 31 875руб.

Амортизация начисляется на все установленное оборудование.

Расчет производиться исходя из количества единиц оборудования, их балансовой стоимости и норм амортизации.

Затраты на амортизацию основных фондов определяются по формуле:

 (6)

где Фбал.- балансовая стоимость основных фондов, руб.;

На- норма амортизации основных фондов, %.

Годовую сумму амортизационных отчислений отразим в таблице 4.

Таблица 4 - Расчет годовой суммы амортизационных отчислений

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Элементы основных фондов. | Кол-во | Балансовая  стоимость, руб. | Сумма руб. | Норма амортизации, % | Амортизационные отчисления, руб. |
| Компьютер | 1 | 40000 | 40000 | 20 | 8000 |
| Монитор | 1 | 30000 | 30000 | 10 | 3000 |
| Итого | | | | | 11000 |

Выполняем расчет фактической суммы амортизационных отчислений за период разработки программного продукта по формуле



где Агод – годовая сумма амортизационных отчислений, руб.;

Тгод – количество рабочих дней в году, д.;

Тпр – продолжительность дней выполнения работ.

Афак=11000\*20/247=890,69руб.

Фактическое сумма амортизационных отчислений за период разработки программного продукта составляет: 890,69 руб.

Расчет потребляемой энергии представлен в таблице 5

Таблица 5 - Расчет потребляемой энергии

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование | Расход кВт/час | Кол-во часов | Всего кВт/час | Тариф (руб) | Итого |
| ПК с монитором | 0.35 | 160 | 56 | 7,32 | 409,92 |
|  |  |  |  |  | 409,92 |

Определим сумму материальных затрат в таблице 6

Таблица 6 - Расчет материальных затрат.

|  |  |
| --- | --- |
| Наименование | Сумма, руб/мес. |
| Электроэнергия | 409,92 |
| Хостинг и сервер | 1500 |
| Домен | 1000 |
| Итого: | 2909,92 |
| Прочие расходы | 290,99 |
| Всего: | 3200,91 |

Сумма прочих расходов определяется в размере 10% от суммы материальных затрат.

Определим себестоимость программного продукта по формуле:

Спп = Зпл + Зсв + Афакт + Зм (7)

где: Зпл- фонд заработной платы, руб.;

Зсв - сумма отчислений во внебюджетные государственные фонды, руб.;

Афакт – фактическая амортизация используемого оборудования, руб.;

Зм- материальные затраты на период разработки программного продукта, руб.

Спп=106250+31875+890,69+3200,91=142 216,6 руб.

Себестоимость программного продукта составляет: 142 216,6 руб.

Исходя из планируемого уровня рентабельности 15%, мы можем определить цену разработанной нами программы:

 (8)

где Спп - себестоимость разработки программы;

R - планируемый уровень рентабельности.

Ц=142 216,6 +142 216,6 \*15/100=163 549,09 руб.

Таким образом цена программного продукта составила: 163 549,09 руб.

Смета затрат на создание программного продукта предприятия приведена в таблице 7.

Т а б л и ц а 7 - Смета затрат на создание программного продукта

| Наименование статьи затрат | Буквенное обозначение | Сумма, руб. | Удельный вес, % |
| --- | --- | --- | --- |
| Фонд заработной платы | Зпл | 106 250 | 74,71% |
| Сумма отчислений во  внебюджетные государственные фонды | Зсв | 31875 | 22,41% |
| Амортизация используемого оборудования | Афакт | 890,69 | 0,63% |
| Материальные затраты | Зм | 2909,92 | 2,05% |
| Прочие расходы | Зпр | 290,99 | 0,2% |
| Всего затрат | З | 142216,6 | 100,00% |

Таким образом, стоимость затрат составила 142 216,6 руб.

Диаграмма удельных затрат показана на рисунке 10.

Рисунок 2 – Диаграмма удельных затрат

1.4 Средства разработки проекта

Перед разработкой сайта необходимо определиться с технологиями, начнем с языка программирования (таблица 8):

Таблица 8 – Сравнение языков программирования для разработки сайта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерии | Python | PHP | JavaScript | Ruby |
| Подходит для | Быстрая разработка, сложная логика | Высокая Сорость, много готовых решений | Одностраничные приложения | Быстрый старт |
| Фреймворки | Django, Flask, FastAPI | Laravel, Symfony | Node, Next.js | Ruby on Rails |
| ORM | Django ORM, SQLAlchemy | Eloquent | Sequelize, TypeORM | AcriveRecord |
| Админка | Django Admin | Laravel Nova | Нужно писать с нуля | ActiveAdmin |
| Аутентификация | Django-allauth | Laravel Breeze | Passport.js, NextAuth | Devise |
| Корзина | Django-cart | Laravel Cart | Писать с нуля | Писать с нуля |
| Производительность | Средняя | Высокая | Очень высокая | Средняя |
| Базы данных | PostgreSQL, SQLite | MySQL, PostgreSQL | MongoDB, PostgreSQL | PostgreSQL, SQLite |
| Деплой | Простой | Очень простой | Сложно | Простой |
| Плюсы | Чистый код, много библиотек | Много хостингов, быстрый | Один язык для frontend+backend | Очень быстрое прототипирование |
| Минусы | Медленнее PHP/Node | Устаревает | Сложность fullstack | Меньше разработчиков |

Выводы:

- Python – лучший баланс между скоростью разработки и функциональностью.

- PHP – если нужен дешёвый хостинг и максимальная скорость.

- JavaScript – если магазин будет одностранничныйс крутым фронтендом.

- Ruby – если нравится "магия" как в Django, но с другим синтаксисом.

Далее сравним фрейморки python (таблица 9)

Таблица 9 – Сравнение Python-фреймворков для разработки сайтов

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий | Django | Flask | FastpAPI |
| Тип фреймворка | Полноценный | Микрофрейворк | Асинхронный |
| Скорость разработки | Очень высокая | Средняя | Быстрый |
| Админка | Django Admin | Flask-Admin | Писать с нуля |
| ORM | Django ORM | Нужны дополнительные библиотеки | Нужны дополнительные библиотеки |
| Аутентификация | Django-allauth | Flask-login | FastAPI Users |
| Формы | Django forms | WTForms | Писать с нуля |
| Корзина/Каталог | Django-cart | Писать с нуля | Есть только API |
| Rest API | Django REST Framework | Flask-Restful | Встроенная поддержка |
| Производительность | Средняя | Средняя | Высокая |
| Документация | Обширная | Хорошая | Удобная |

Итоговый выбор:

Django (Python) – лучший баланс:

- Готовая админка для управления товарами.

- Встроенная аутентификация и ORM.

- Поддержка платежей (Stripe, PayPal через django-payments).

2 ТЕХНИЧЕСКИЙ РАЗДЕЛ

2.1 Разработка технического задания

Функциональные требования

2.1. Каталог товаров

Категории товаров (например, "Букеты", "Комнатные растения", "Подарки").

Фильтрация и сортировка по:

Цене

Популярности

Наличию

Категориям

Поиск по названию и описанию.

Детальная страница товара с:

Фотографиями (галерея)

Описанием

Ценой

Кнопкой "Добавить в корзину"

2.2. Корзина и оформление заказа

Добавление/удаление товаров из корзины.

Изменение количества товаров.

Промокоды и скидки.

Оформление заказа:

Форма с данными покупателя (имя, телефон, email, адрес доставки).

Выбор способа доставки (курьер, самовывоз).

Оплата (онлайн или при получении).

2.3. Личный кабинет

Регистрация и авторизация (через email/соцсети).

История заказов с статусами ("В обработке", "Доставлен").

Изменение профиля (контактные данные, пароль).

2.4. Административная панель

Управление товарами (добавление, редактирование, удаление).

Управление заказами (просмотр, изменение статуса).

Управление пользователями.

Аналитика (продажи, популярные товары).

2.5. Дополнительные функции

Блог/новости (статьи о цветах, акции).

Отзывы о товарах.

Подписка на рассылку (уведомления о скидках).

3. Нефункциональные требования

3.1. Производительность

Время загрузки страницы: < 2 сек.

Оптимизация изображений (сжатие без потери качества).

3.2. Безопасность

Защита от SQL-инъекций, XSS, CSRF (в Django есть по умолчанию).

HTTPS для всех страниц.

3.3. Адаптивность

Корректное отображение на ПК, планшетах и смартфонах.

3.4. SEO

ЧПУ (человеко-понятные URL).

Мета-теги (title, description).

Sitemap.xml и robots.txt.

2.2 Проектирование и разработка ИТ-решения

При разработке интернет-магазина для начала необходимо разработать дизайн и написать фронтенд часть, а также адаптировать под мобильные устройства.

Начало главная страницы магазина показана на рисунке 3 (вид с комьютера), рисунок 4 (вид с телефона)

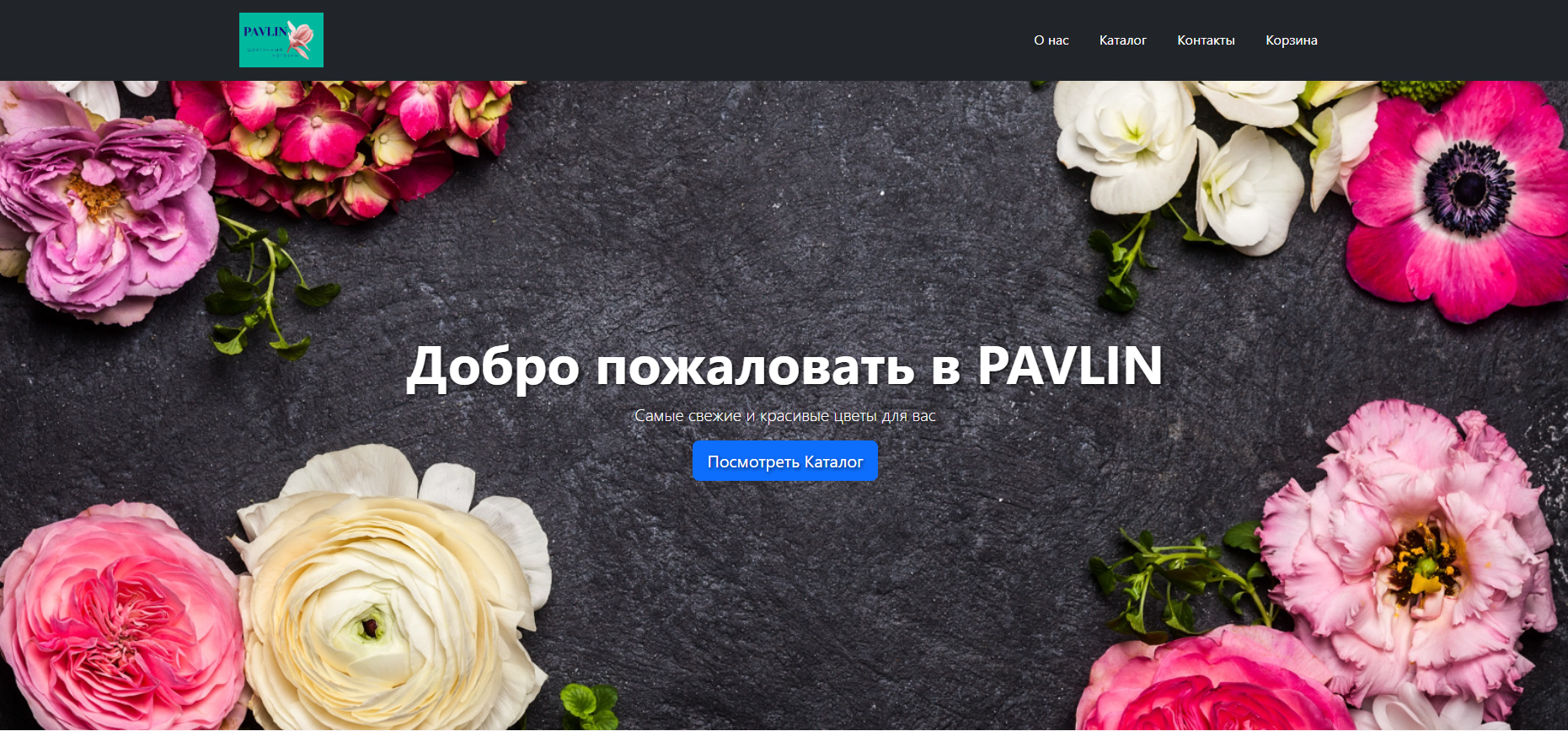


Рисунок 3 – начало главной страницы

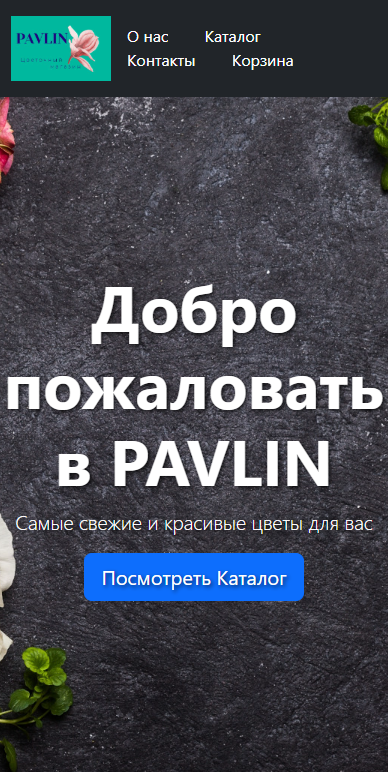


Рисунок 4 – начало главной страницы (вид с телефона)

Далее идет раздел «О нас» и каталог, рисунок 5 ( вид с компьютера) рисунок 6, 7 (вид с телефона)

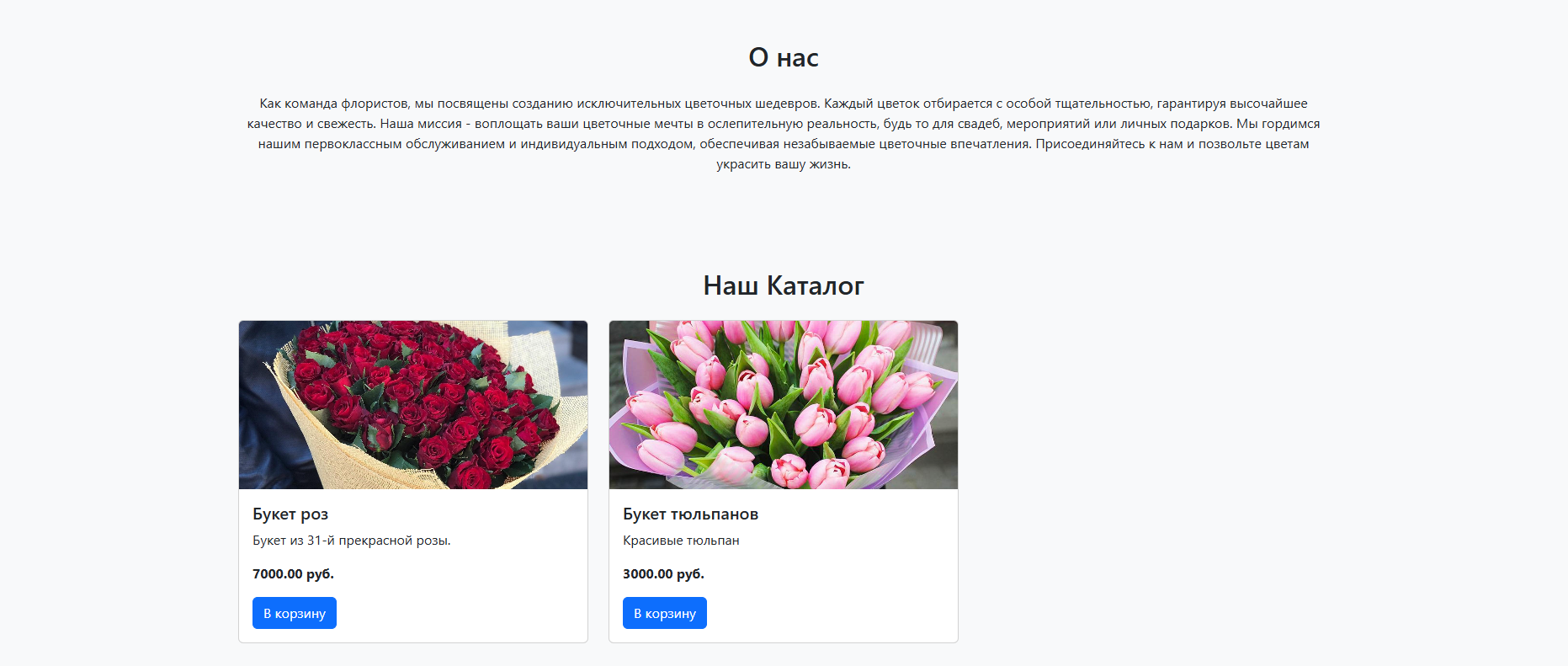


Рисунок 5 – каталог и раздел «О нас»

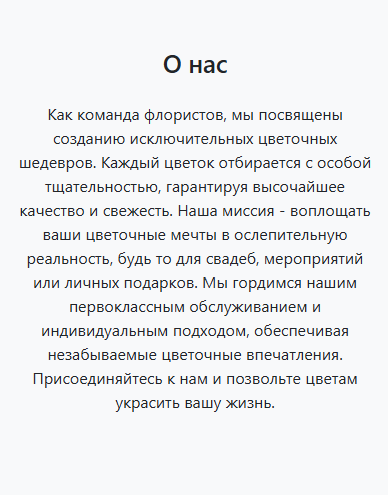


Рисунок 6 – раздел «о нас» (вид с телефона)

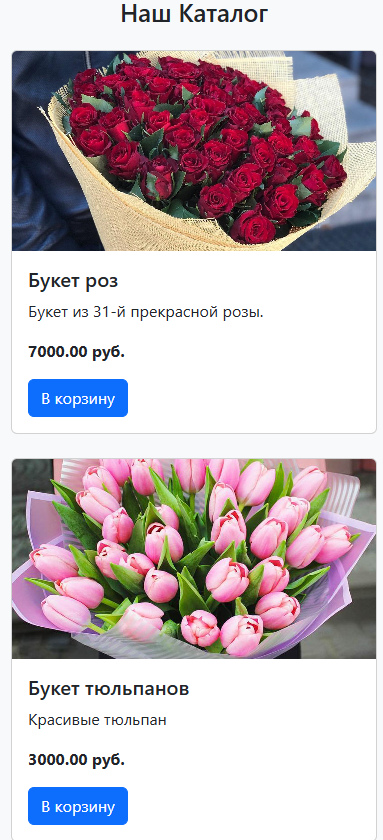


Рисунок 7 – каталог (вид с телефона)

Внизу страницы идет раздел с контактами и подвал с копирайтом, рисунок 8 (вид с компьютера), рисунок 9 (вид с телефона)

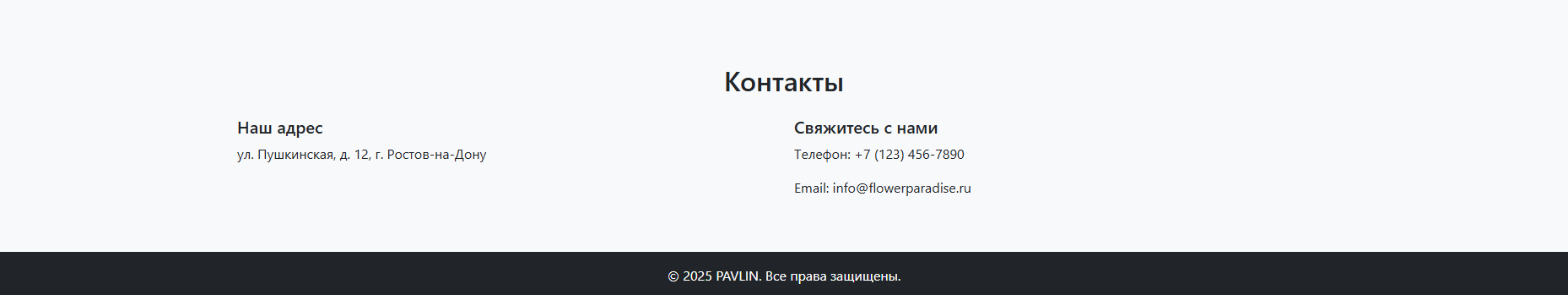


Рисунок 8 – контакты и подвал

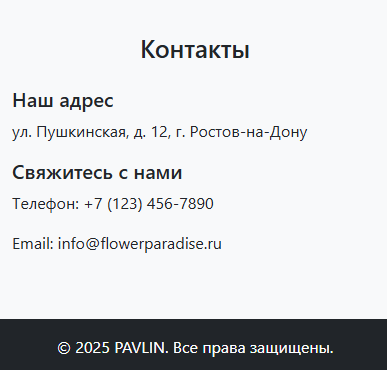


Рисунок 9 – контакты и подвал (вид с телефона)

Исходный код главной страницы:

{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

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

<title>Цветочный Магазин</title>

<link

href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"

rel="stylesheet"/>

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

</head>

<body>

<header class="bg-dark text-white py-3">

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

<a href="./index.html">

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

</a>

<nav>

<a href="#about" class="text-white mx-3 text-decoration-none"

>О нас</a

>

<a href="#products" class="text-white mx-3 text-decoration-none"

>Каталог</a

>

<a href="#contact" class="text-white mx-3 text-decoration-none"

>Контакты</a

>

<a href="{% url 'katalog:cart\_detail' %}" class="text-white mx-3 text-decoration-none"

>Корзина</a

>

</nav>

</div>

</header>

<section

class="hero d-flex align-items-center justify-content-center text-center"

style="min-height: 80vh"

>

<div>

<h1>Добро пожаловать в PAVLIN</h1>

<p class="lead">Самые свежие и красивые цветы для вас</p>

<a href="#products" class="btn btn-primary btn-lg"

>Посмотреть Каталог</a

>

</div>

</section>

<section id="about" class="py-5">

<div class="container text-center">

<h2 class="mb-4">О нас</h2>

<p>

Как команда флористов, мы посвящены созданию исключительных цветочных шедевров. Каждый цветок отбирается с особой тщательностью, гарантируя высочайшее качество и свежесть. Наша миссия - воплощать ваши цветочные мечты в ослепительную реальность, будь то для свадеб, мероприятий или личных подарков. Мы гордимся нашим первоклассным обслуживанием и индивидуальным подходом, обеспечивая незабываемые цветочные впечатления. Присоединяйтесь к нам и позвольте цветам украсить вашу жизнь.

</p>

</div>

</section>

<section id="products" class="py-5 bg-light">

<div class="container">

<h2 class="text-center mb-4">Наш Каталог</h2>

<div class="row g-4">

{% for product in products %}

<div class="col-md-4">

<div class="card">

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

<div class="card-body">

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

<p class="card-text">{{ product.description }}</p>

<p class="card-text"><strong>{{ product.price }} руб.</strong></p>

<a href="{% url 'katalog:add\_to\_cart' product.id %}" class="btn btn-primary">В корзину</a>

</div>

</div>

</div>

{% endfor %}

</div>

</div>

</section>

<section id="contact" class="py-5">

<div class="container">

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

<div class="row">

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

<h5>Наш адрес</h5>

<p>ул. Пушкинская, д. 12, г. Ростов-на-Дону</p>

</div>

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

<h5>Свяжитесь с нами</h5>

<p>Телефон: +7 (123) 456-7890</p>

<p>Email: info@flowerparadise.ru</p>

</div>

</div>

</div>

</section>

<footer class="bg-dark text-white text-center py-3">

<p class="mb-0">© 2025 PAVLIN. Все права защищены.</p>

</footer>

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

</body>

</html>

Далее идет страница с корзиной куда попадают добавленные товары. Также идет подсчет стоимости всей корзины, рисунок 10 (вид с компьютера), рисунок 11 (вид с телефона)

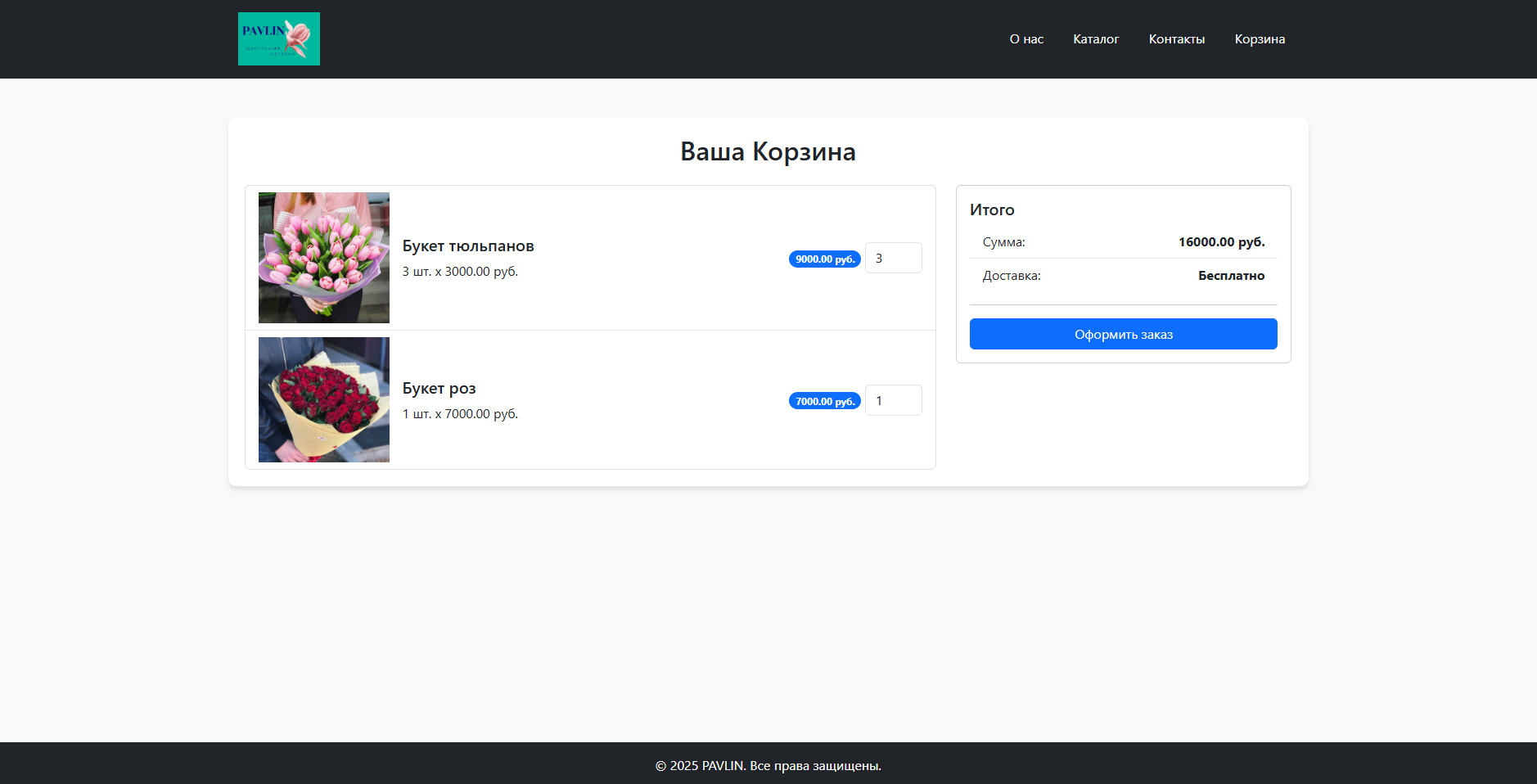


Рисунок 10 – корзина магазина

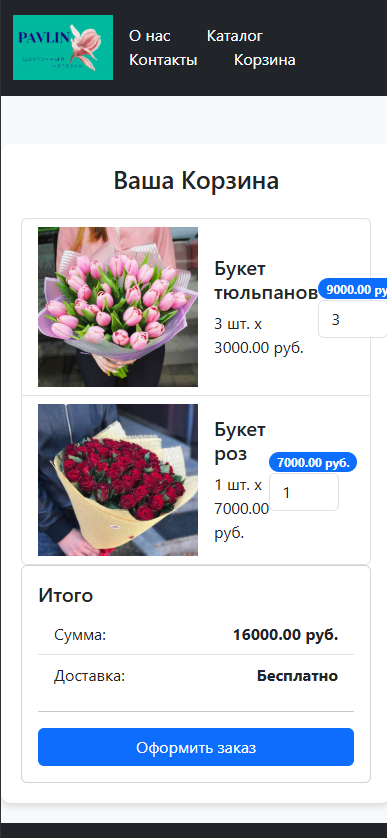


Рисунок 11 – корзина (вид с телефона)

Исходный код страницы корзины:

{% extends 'katalog/base.html' %}

{% load static %}

{% block content %}

<!DOCTYPE html>

<html lang="ru" class="h-100">

<head>

<meta charset="UTF-8">

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

<title>Корзина - Цветочный Магазин</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

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

</head>

<body class="d-flex flex-column h-100">

<header class="bg-dark text-white py-3">

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

<a href="{% url 'katalog:index' %}">

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

</a>

<nav>

<a href="{% url 'katalog:index' %}#about" class="text-white mx-3 text-decoration-none">О нас</a>

<a href="{% url 'katalog:index' %}#products" class="text-white mx-3 text-decoration-none">Каталог</a>

<a href="{% url 'katalog:index' %}#contact" class="text-white mx-3 text-decoration-none">Контакты</a>

<a href="{% url 'katalog:cart\_detail' %}" class="text-white mx-3 text-decoration-none">

<i class="bi bi-cart"></i> Корзина

</a>

</nav>

</div>

</header>

<main class="flex-grow-1 ">

<div class="container cart-container mt-5">

<h2 class="text-center mb-4">Ваша Корзина</h2>

<div class="row">

<div class="col-md-8">

<div class="list-group">

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

<div class="list-group-item d-flex align-items-center">

<img src="{{ item.product.image.url }}" alt="{{ item.product.name }}" class="me-3" width="80">

<div>

<h5>{{ item.product.name }}</h5>

<p class="mb-0">{{ item.quantity }} шт. x {{ item.product.price }} руб.</p>

</div>

<div class="ms-auto">

<span class="badge bg-primary rounded-pill">{{ item.price }} руб.</span>

<form method="post" action="{% url 'katalog:update\_cart' item.id %}" class="d-inline">

{% csrf\_token %}

<input type="number" name="quantity" value="{{ item.quantity }}" min="1" class="form-control d-inline" style="width: 70px;">

</form>

</div>

</div>

{% endfor %}

</div>

</div>

<div class="col-md-4">

<div class="card">

<div class="card-body">

<h5 class="card-title">Итого</h5>

<ul class="list-group list-group-flush">

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

<span>Сумма:</span>

<strong>{{ cart.total\_price }} руб.</strong>

</li>

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

<span>Доставка:</span>

<strong>Бесплатно</strong>

</li>

</ul>

<hr>

<div class="d-grid">

<a href="{% url 'katalog:checkout' %}" class="btn btn-primary">

Оформить заказ <i class="bi bi-arrow-right"></i>

</a>

</div>

</div>

</div>

</div>

</div>

</div>

</main>

<footer class="bg-dark text-white text-center py-3 mt-auto">

<p class="mb-0">© 2025 PAVLIN. Все права защищены.</p>

</footer>

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

<script src="{% static 'katalog/js/main.js' %}"></script>

</body>

</html>

{% endblock %}

После нажатия на кнопку «оформить заказ» пользователя перекидывает на страницу с оформление заказа, где он вводит контактные данные проверяет товар и подтверждает заказ, рисунок 12 (вид с компьютера), рисунок 13,14 ( вид с телефона)

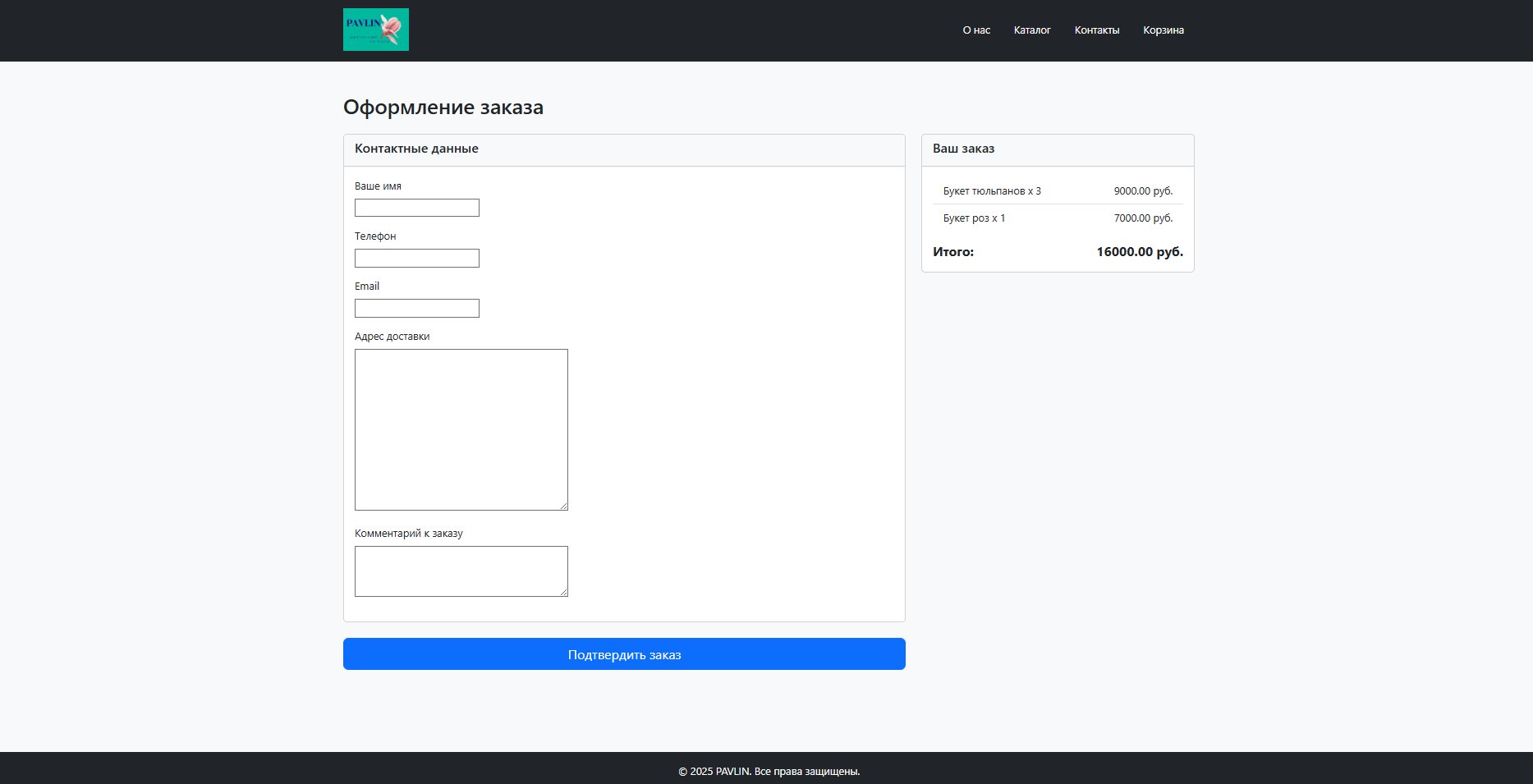


Рисунок 12 – страница оформления заказа

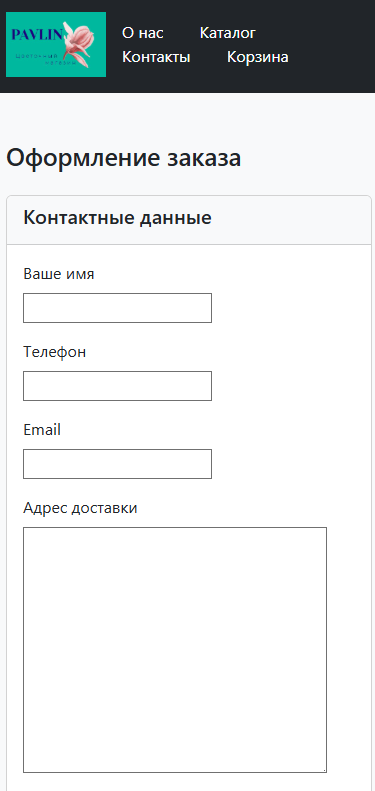


Рисунок 13 – оформление заказа (вид с телефона)

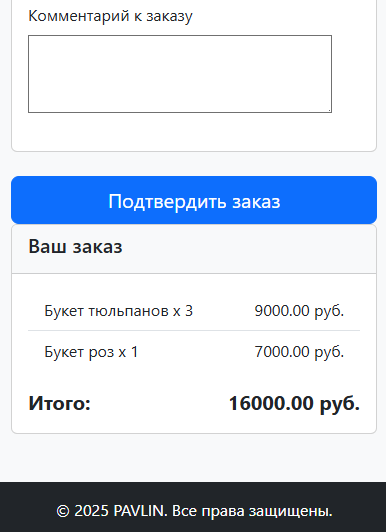


Рисунок 14 – оформление заказа (вид с телефона)

Исходный код страницы оформления заказа:

{% extends 'katalog/base.html' %}

{% load static %}

{% block content %}

<head>

<meta charset="UTF-8">

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

<title>Корзина - Цветочный Магазин</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

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

</head>

<body class="d-flex flex-column h-100">

<header class="bg-dark text-white py-3">

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

<a href="{% url 'katalog:index' %}">

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

</a>

<nav>

<a href="{% url 'katalog:index' %}#about" class="text-white mx-3 text-decoration-none">О нас</a>

<a href="{% url 'katalog:index' %}#products" class="text-white mx-3 text-decoration-none">Каталог</a>

<a href="{% url 'katalog:index' %}#contact" class="text-white mx-3 text-decoration-none">Контакты</a>

<a href="{% url 'katalog:cart\_detail' %}" class="text-white mx-3 text-decoration-none">

<i class="bi bi-cart"></i> Корзина

</a>

</nav>

</div>

</header>

<div class="container my-5">

<div class="row">

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

<div class="col-md-8">

<form method="post">

{% csrf\_token %}

<div class="card mb-4">

<div class="card-header bg-light">

<h5>Контактные данные</h5>

</div>

<div class="card-body">

<div class="mb-3">

<label for="{{ form.customer\_name.id\_for\_label }}" class="form-label">Ваше имя</label>

<br>

{{ form.customer\_name }}

<div class="invalid-feedback">

{{ form.customer\_name.errors }}

</div>

</div>

<div class="mb-3">

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

<br>

{{ form.phone }}

<div class="invalid-feedback">

{{ form.phone.errors }}

</div>

</div>

<div class="mb-3">

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

<br>

{{ form.email }}

<div class="invalid-feedback">

{{ form.email.errors }}

</div>

</div>

<div class="mb-3">

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

<br>

{{ form.address }}

<div class="invalid-feedback">

{{ form.address.errors }}

</div>

</div>

<div class="mb-3">

<label for="{{ form.comment.id\_for\_label }}" class="form-label">Комментарий к заказу</label>

<br>

{{ form.comment }}

</div>

</div>

</div>

<button type="submit" class="btn btn-primary btn-lg w-100">

Подтвердить заказ

</button>

</form>

</div>

<div class="col-md-4">

<div class="card">

<div class="card-header bg-light">

<h5>Ваш заказ</h5>

</div>

<div class="card-body">

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

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

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

<span>{{ item.product.name }} x {{ item.quantity }}</span>

<span>{{ item.price }} руб.</span>

</li>

{% endfor %}

</ul>

<div class="d-flex justify-content-between fw-bold fs-5">

<span>Итого:</span>

<span>{{ cart.total\_price }} руб.</span>

</div>

</div>

</div>

</div>

</div>

</div>

<footer class="bg-dark text-white text-center py-3 mt-auto">

<p class="mb-0">© 2025 PAVLIN. Все права защищены.</p>

</footer>

</body>

{% endblock %}

После нажатия кнопки «подтвердить заказ» пользователя переносит на «чек» где указаны детали заказа и его контактные данные, рисунок 15 ( вид с компьютера), рисунок 16 (вид с телефона)

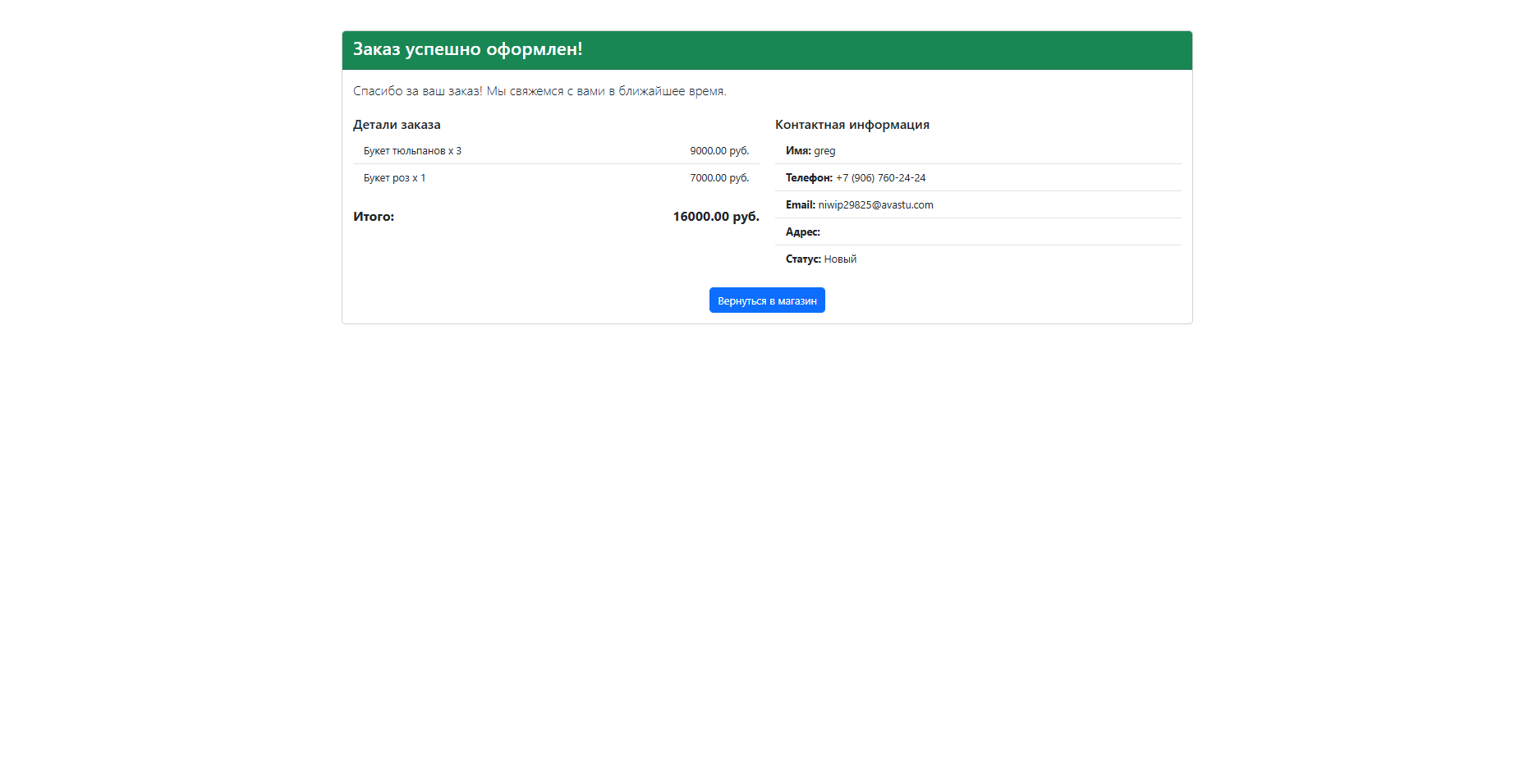


Рисунок 15 – страница заказа

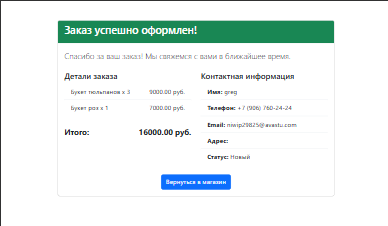


Рисунок 16 – страница заказ (вид с телефона)

На сайте также есть админ-панель (рисунок 17)

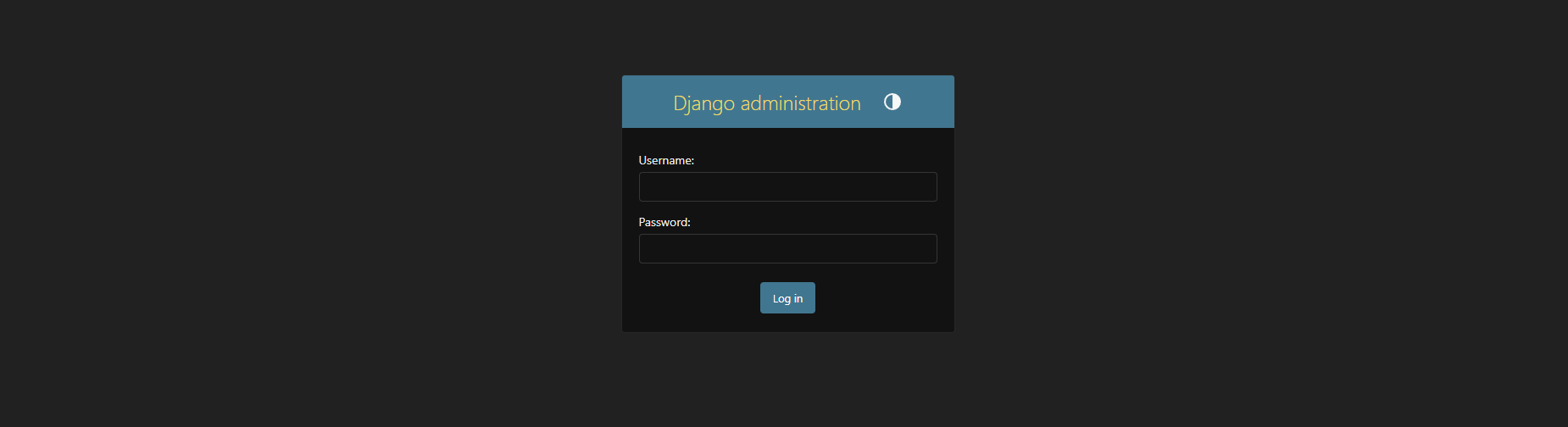


Рисунок 17 – вход в админ-панель

В админ-панеле можно смотреть заказы, добавлять товары и категории, общий вид рисунок 18.

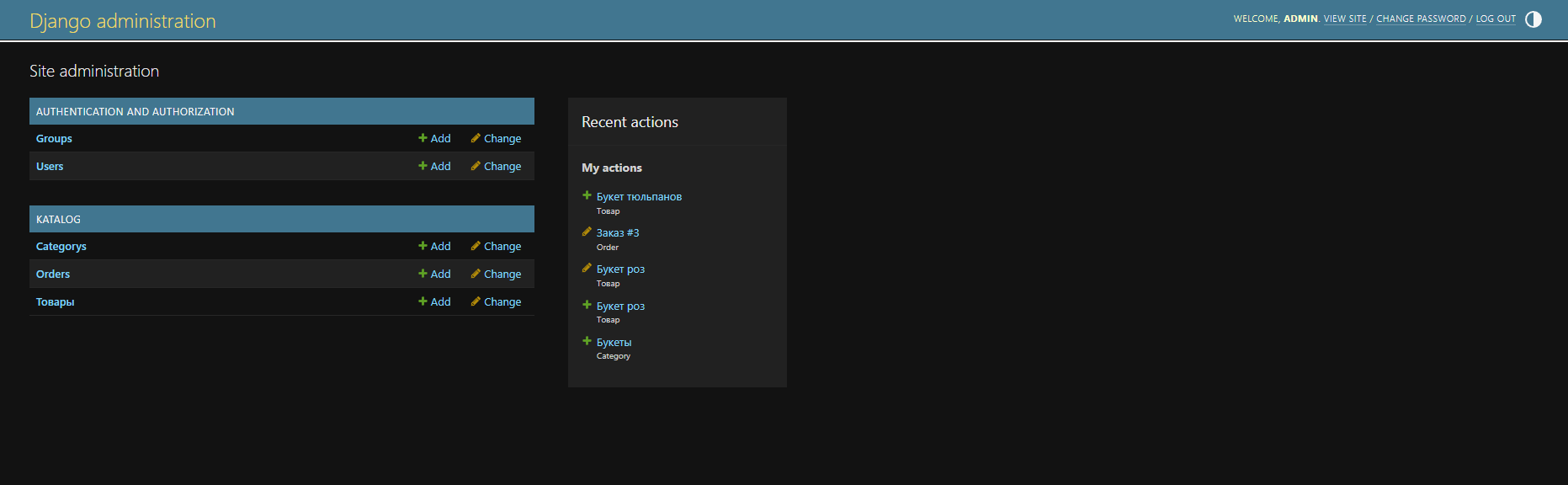


Рисунок 18 – общий вид админ-панели

Просмотр заказов (рисунок 19)

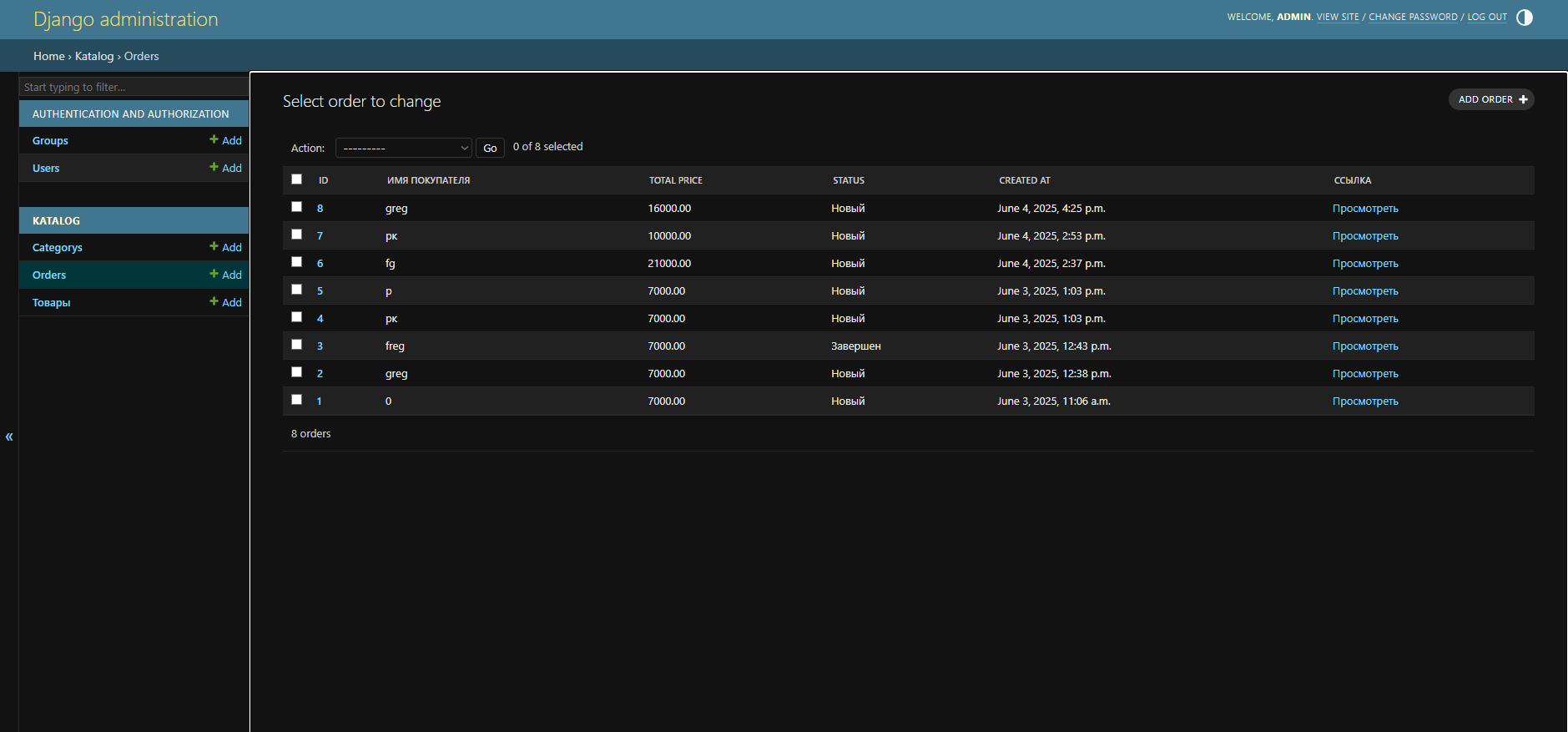


Рисунок 19 – заказы в админ-панели

Редактирование категорий (рисунок 20)

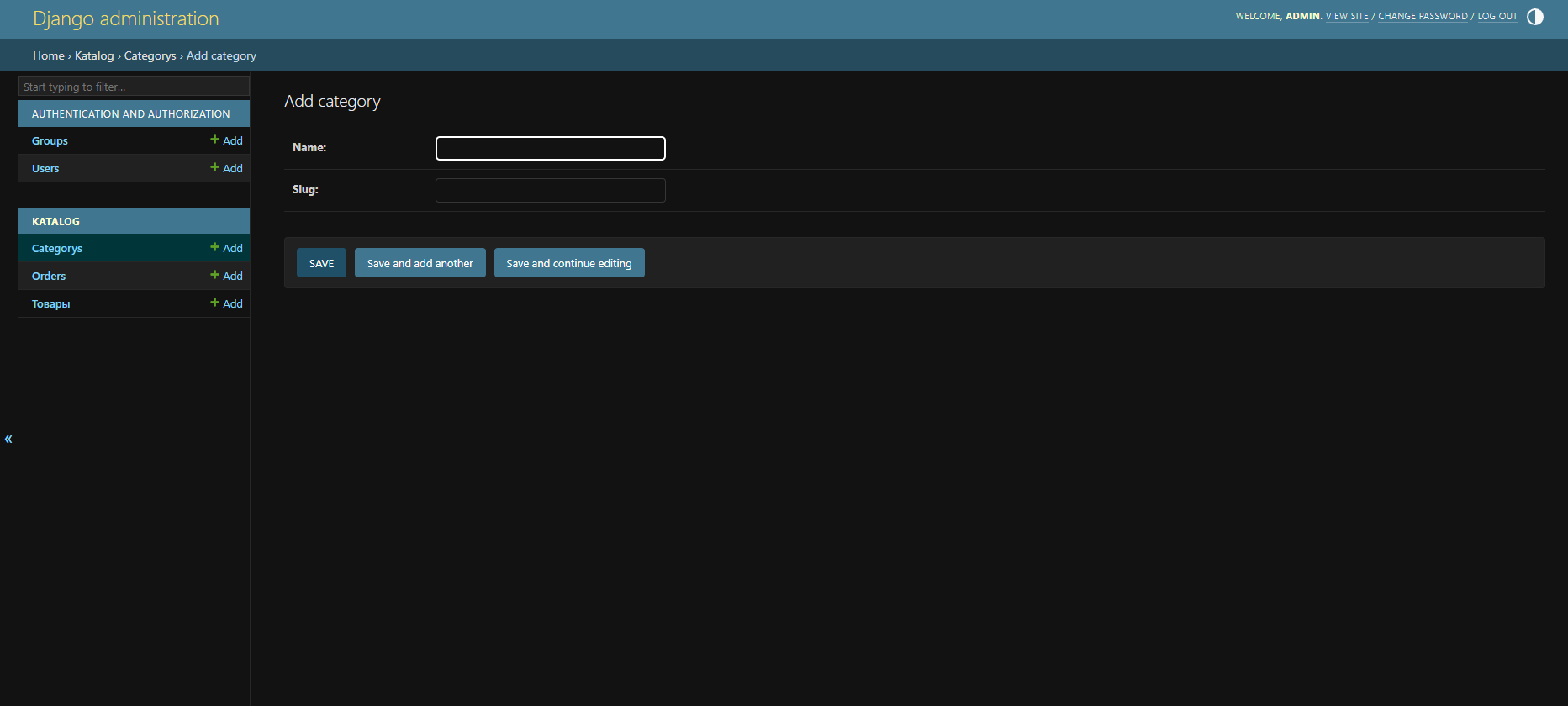


Рисунок 20 – категории товаров в админ-панели

Редактирование товаров (рисунок 21)

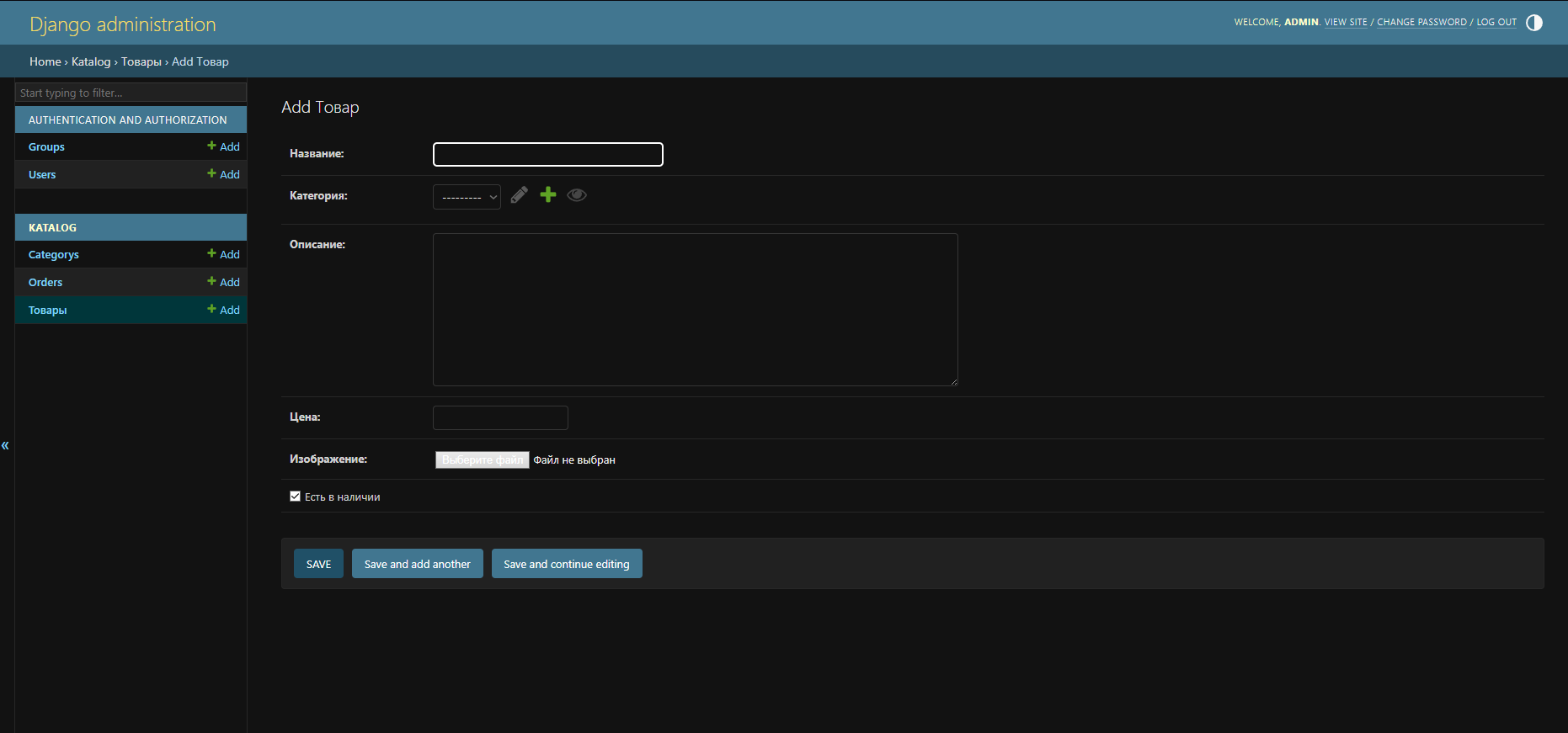


Рисунок 21 – редактирование товаров в админ-панели

Далее идет разработка backend части.

Файл models.py отвечает за структуру базы данных и бизнец-логику.

Основные модели:

1)Product – товары (цветы, букеты)

Поля: name, price, image, category, available

2)Cart – корзина пользователя

Связь: user (ForeignKey), items (ManyToMany через CartItem)

3)Order – заказы

Поля: order\_code (уникальный ID), customer\_name, phone, status, total\_price

4)OrderItem – товары в заказе

Связь: order (ForeignKey), product (ForeignKey)

Исходный код:

from django.db import models

from django.conf import settings

class Category(models.Model):

name = models.CharField(max\_length=100)

slug = models.SlugField(unique=True)

def \_\_str\_\_(self):

return self.name

class Product(models.Model):

id = models.AutoField(primary\_key=True)

name = models.CharField(max\_length=100, verbose\_name="Название")

category = models.ForeignKey(Category, on\_delete=models.CASCADE, verbose\_name='Категория')

description = models.TextField(max\_length=1500, verbose\_name="Описание")

price = models.DecimalField(max\_digits=8, decimal\_places=2, verbose\_name="Цена")

image = models.ImageField(upload\_to='products/', verbose\_name="Изображение")

available = models.BooleanField(default=True, verbose\_name="Есть в наличии")

class Meta:

verbose\_name = "Товар"

verbose\_name\_plural = "Товары"

def \_\_str\_\_(self):

return self.name

def get\_absolute\_url(self):

from django.urls import reverse

return reverse('katalog:product\_detail', args=[str(self.id)])

class Cart(models.Model):

user = models.ForeignKey(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE, null=True, blank=True)

created\_at = models.DateTimeField(auto\_now\_add=True)

updated\_at = models.DateTimeField(auto\_now=True)

@property

def total\_price(self):

return sum(item.price for item in self.items.all())

class CartItem(models.Model):

cart = models.ForeignKey(Cart, related\_name='items', on\_delete=models.CASCADE)

product = models.ForeignKey('Product', on\_delete=models.CASCADE)

quantity = models.PositiveIntegerField(default=1)

price = models.DecimalField(max\_digits=10, decimal\_places=2) # Фиксируем цену на момент добавления

def save(self, \*args, \*\*kwargs):

self.price = self.product.price \* self.quantity

super().save(\*args, \*\*kwargs)

def \_\_str\_\_(self):

return f"{self.product.name} x {self.quantity}"

class Order(models.Model):

STATUS\_CHOICES = [

('new', 'Новый'),

('processing', 'В обработке'),

('completed', 'Завершен'),

('cancelled', 'Отменен')

]

user = models.ForeignKey(

settings.AUTH\_USER\_MODEL,

on\_delete=models.SET\_NULL, # Изменено с CASCADE

null=True,

blank=True

)

customer\_name = models.CharField( # Новое поле для имени

'Имя покупателя',

max\_length=100,

blank=False # Обязательное поле

)

created\_at = models.DateTimeField(auto\_now\_add=True)

updated\_at = models.DateTimeField(auto\_now=True)

status = models.CharField(max\_length=20, choices=STATUS\_CHOICES, default='new')

total\_price = models.DecimalField(max\_digits=10, decimal\_places=2)

comment = models.TextField(blank=True)

# Контактные данные

phone = models.CharField(max\_length=20)

email = models.EmailField(

blank=True, # Разрешаем пустое значение в формах

null=True, # Разрешаем NULL в базе данных

verbose\_name='Email'

)

address = models.TextField(

blank=True, # Разрешаем пустое значение в формах

null=True, # Разрешаем NULL в базе данных

verbose\_name='Адрес доставки'

)

def \_\_str\_\_(self):

return f"Заказ #{self.id}"

class OrderItem(models.Model):

order = models.ForeignKey(Order, related\_name='items', on\_delete=models.CASCADE)

product = models.ForeignKey('Product', on\_delete=models.PROTECT)

quantity = models.PositiveIntegerField()

price = models.DecimalField(max\_digits=10, decimal\_places=2)

def \_\_str\_\_(self):

return f"{self.product.name} x {self.quantity}"

Далее идут представления views.py, файл обрабатывает HTTP-запросы и возвращает ответы. Ключевые функции показаны в таблице 10.

Таблица 10 – ключевые функции обработки

|  |  |  |
| --- | --- | --- |
| **Метод** | **URL** | **Описание** |
| product\_list | / | Главная страница с товарами |
| cart\_detail | /cart/ | Просмотр корзины |
| add\_to\_cart | /cart/add/<product\_id>/ | Добавление товара в корзину |
| update\_cart | /cart/update/<item\_id>/ | Изменение количества товара |
| checkout | /checkout/ | Оформление заказа |
| order\_detail | /orders/<order\_id>/ | Детали заказа |

Исходный код:

from django.shortcuts import render, redirect, get\_object\_or\_404

from .models import Product, Cart, CartItem, OrderItem, Order

from django.contrib.auth.decorators import login\_required

from django.contrib import messages

from django.views.decorators.http import require\_POST

from .forms import OrderForm

def index(request):

products = Product.objects.all()

return render(request, 'index.html', {'products': products})

def get\_or\_create\_cart(request):

if request.user.is\_authenticated:

cart, created = Cart.objects.get\_or\_create(user=request.user)

else:

cart\_id = request.session.get('cart\_id')

if cart\_id:

cart = get\_object\_or\_404(Cart, id=cart\_id)

else:

cart = Cart.objects.create()

request.session['cart\_id'] = cart.id

return cart

def add\_to\_cart(request, product\_id):

product = get\_object\_or\_404(Product, id=product\_id)

# Получаем или создаем корзину

if request.user.is\_authenticated:

cart, created = Cart.objects.get\_or\_create(user=request.user)

else:

cart\_id = request.session.get('cart\_id')

if cart\_id:

cart = get\_object\_or\_404(Cart, id=cart\_id)

else:

cart = Cart.objects.create()

request.session['cart\_id'] = cart.id

# Добавляем товар в корзину

cart\_item, created = CartItem.objects.get\_or\_create(

cart=cart,

product=product,

defaults={'price': product.price}

)

if not created:

cart\_item.quantity += 1

cart\_item.price = product.price \* cart\_item.quantity

cart\_item.save()

messages.success(request, f'Товар "{product.name}" добавлен в корзину')

return redirect('katalog:cart\_detail')

def remove\_from\_cart(request, item\_id):

cart\_item = get\_object\_or\_404(CartItem, id=item\_id)

cart\_item.delete()

return redirect('katalog:cart\_detail')

def cart\_detail(request):

if request.user.is\_authenticated:

cart = Cart.objects.filter(user=request.user).first()

else:

cart\_id = request.session.get('cart\_id')

cart = Cart.objects.filter(id=cart\_id).first() if cart\_id else None

if not cart:

# Создаем пустую корзину, если не существует

if request.user.is\_authenticated:

cart = Cart.objects.create(user=request.user)

else:

cart = Cart.objects.create()

request.session['cart\_id'] = cart.id

return render(request, 'katalog/cart.html', {'cart': cart})

@require\_POST

def update\_cart(request, item\_id):

cart\_item = get\_object\_or\_404(CartItem, id=item\_id)

quantity = int(request.POST.get('quantity', 1))

if quantity > 0:

cart\_item.quantity = quantity

cart\_item.price = cart\_item.product.price \* quantity

cart\_item.save()

else:

cart\_item.delete()

return redirect('katalog:cart\_detail')

def checkout(request):

cart = get\_or\_create\_cart(request) # Используем вашу существующую функцию

if not cart.items.exists():

messages.warning(request, "Ваша корзина пуста")

return redirect('katalog:cart\_detail')

if request.method == 'POST':

form = OrderForm(request.POST)

if form.is\_valid():

order = form.save(commit=False)

if not order.email and request.user.is\_authenticated:

order.email = request.user.email

if request.user.is\_authenticated:

order.user = request.user

if not order.customer\_name: # Если имя не указано, берем из профиля

order.customer\_name = request.user.get\_full\_name() or request.user.username

order.total\_price = cart.total\_price

order.save()

# Переносим товары из корзины в заказ

for item in cart.items.all():

OrderItem.objects.create(

order=order,

product=item.product,

quantity=item.quantity,

price=item.price

)

# Очищаем корзину

cart.items.all().delete()

messages.success(request, "Ваш заказ успешно оформлен!")

return redirect('katalog:order\_detail', order\_id=order.id)

else:

# Предзаполняем данные для авторизованных пользователей

initial = {}

if request.user.is\_authenticated:

initial = {

'customer\_name': request.user.get\_full\_name(),

'email': request.user.email,

'phone': request.user.phone if hasattr(request.user, 'phone') else ''

}

form = OrderForm(initial=initial)

if form.is\_valid():

order = form.save(commit=False)

order.user = request.user if request.user.is\_authenticated else None

order.total\_price = cart.total\_price

order.save()

# Переносим товары из корзины в заказ

for item in cart.items.all():

OrderItem.objects.create(

order=order,

product=item.product,

quantity=item.quantity,

price=item.price

)

# Очищаем корзину

cart.items.all().delete()

messages.success(request, "Ваш заказ успешно оформлен!")

return redirect('katalog:order\_detail', order\_id=order.id) # Вот эта ссылка

return render(request, 'katalog/checkout.html', {

'cart': cart,

'form': form

})

def order\_detail(request, order\_id):

order = get\_object\_or\_404(Order, id=order\_id)

return render(request, 'katalog/order\_detail.html', {'order': order})

Далее идет файл маршрутизации urls.py, он определяет, какой view обрабатывает запрос.

Исходный код:

from django.urls import path

from . import views

from django.views.generic.base import TemplateView

app\_name = 'katalog' # Это определяет namespace

urlpatterns = [

path('', views.index, name='index'),

path('cart/', views.cart\_detail, name='cart\_detail'),

path('add/<int:product\_id>/', views.add\_to\_cart, name='add\_to\_cart'),

path('remove/<int:item\_id>/', views.remove\_from\_cart, name='remove\_from\_cart'),

path('cart/update/<int:item\_id>/', views.update\_cart, name='update\_cart'),

path('checkout/', views.checkout, name='checkout'),

path('orders/<int:order\_id>/', views.order\_detail, name='order\_detail'),

path('robots.txt', TemplateView.as\_view(

template\_name='robots.txt',

content\_type='text/plain')

),

]

Далее идет файл с формами forms.py, он отвечает за валидацию и обработку данных от пользователей.

Исходный код:

from django import forms

from .models import Order

class OrderForm(forms.ModelForm):

class Meta:

model = Order

fields = ['customer\_name', 'phone', 'email', 'address', 'comment']

widgets = {

'comment': forms.Textarea(attrs={'rows': 3}),

}

labels = {

'customer\_name': 'Ваше имя',

}

Далее идет файл с админ-панелью admin.py, он предназначен для управления данными через Django Admin.

Исходный код:

from django.contrib import admin

from .models import Category, Product, Order

admin.site.register(Category)

admin.site.register(Product)

#@admin.register(Product)

#class ProductAdmin(admin.ModelAdmin):

# list\_display = ['name', 'category', 'price', 'available']

# list\_filter = ['category', 'available']

# search\_fields = ['name', 'description']

@admin.register(Order)

class OrderAdmin(admin.ModelAdmin):

list\_display = ('id', 'customer\_name', 'total\_price', 'status', 'created\_at', 'order\_link')

def order\_link(self, obj):

from django.urls import reverse

from django.utils.html import format\_html

url = reverse('katalog:order\_detail', args=[obj.id])

return format\_html('<a href="{}">Просмотреть</a>', url)

order\_link.short\_description = "Ссылка"

3 ОХРАНА ТРУДА И ТЕХНИКА БЕЗОПАСНОСТИ

3.1 Общие требования охраны труда

1.1. К работе программистом допускаются лица не моложе 18 лет, имеющие соответствующую выполняемой работе квалификацию, прошедшие вводный и первичный на рабочем месте инструктажи по охране труда, медосмотр, обучение и проверку знаний по охране труда.

1.2. Для выполнения работ на персональном компьютере программист должен изучить инструкцию по эксплуатации персонального компьютера, на котором работник выполняет работы, пройти инструктаж по электробезопасности и получить I группу.

1.3. Программист, выполняющий работу на персональном компьютере, независимо от квалификации и стажа работы не реже одного раза в шесть месяцев должен проходить повторный инструктаж по безопасности труда; в случае нарушения требований безопасности труда, при перерыве в работе более чем на 60 календарных дней программист должен пройти внеплановый инструктаж.

1.4. Программист, показавший неудовлетворительные навыки и знания требований безопасности при работе на персональном компьютере, к самостоятельной работе не допускается.

1.5. Программист, допущенный к самостоятельной работе, должен знать: правила эксплуатации и требования безопасности при работе с персональным компьютером, способы рациональной организации рабочего места, санитарно-гигиенические требования к условиям труда, опасные и вредные производственные факторы, которые могут оказывать неблагоприятное воздействие на программиста.

1.6. Программист, направленный для участия в несвойственных его профессии работах, должен пройти целевой инструктаж по безопасному выполнению предстоящих работ.

1.7. Во время работы на программиста могут оказывать неблагоприятное воздействие в основном следующие опасные и вредные производственные факторы:

· перенапряжение зрительного анализатора при работе за экраном дисплея;

· длительное статическое напряжение мышц спины, шеи, рук и ног, что может привести к статическим перегрузкам программиста;

· повышенный уровень шума;

· ионизирующие и неионизирующие излучения, источниками которых являются видеодисплейные терминалы;

· статическое электричество;

· электрический ток, путь которого в случае замыкания на корпус может пройти через тело человека.

1.8. Программист, работающий на персональном компьютере, должен соблюдать установленные для него режимы труда и отдыха.

1.9. Для предупреждения возможности возникновения пожара программист должен соблюдать требования пожарной безопасности сам и не допускать нарушений со стороны других работников.

1.10. Для предупреждения заболеваний программисту следует знать и соблюдать правила личной гигиены.

1.11. В случае заболевания, плохого самочувствия, недостаточного отдыха программисту следует сообщить о своем состоянии непосредственному руководителю и обратиться за медицинской помощью.

1.12. Если программист оказался очевидцем несчастного случая, он должен оказать пострадавшему первую помощь и сообщить о случившемся руководителю.

1.13. Программист должен уметь оказать первую помощь, в том числе при поражении электрическим током, пользоваться аптечкой.

1.14. Программист, допустивший нарушение или невыполнение требований инструкции по охране труда, несет ответственность согласно действующему законодательству.

2. Требования охраны труда перед началом работы

2.1. Перед началом работы программисту следует рационально организовать свое рабочее место.

2.2. Программист должен знать о том, что если в помещении расположены несколько персональных компьютеров, то для обеспечения безопасности расстояние между ними должно быть не менее 1,5 м.

2.3. Программист должен знать о том, что взаимное расположение персональных компьютеров влияет на уровень генерируемых ими излучений; для предупреждения облучения других рабочих мест следует выполнять следующие правила:

2.3.1. Левая панель персонального компьютера должна быть обращена либо к стене, либо к проходу, где нет рабочих мест.

2.3.2. Не следует располагать мониторы экранами друг к другу.

2.4. Не рекомендуется располагать монитор экраном к окну.

2.5. Для того чтобы в процессе работы не возникало перенапряжение зрительного анализатора, программисту следует проверить, чтобы на клавиатуре и экране монитора не было бликов света.

2.6. Для повышения контрастности изображения перед началом работы программист должен очистить экран монитора от пыли, которая интенсивно оседает на нем под воздействием зарядов статического электричества.

2.7. Программист должен убрать с рабочего места все лишние предметы, не используемые в работе.

2.8. Перед включением персонального компьютера программисту следует визуально проверить исправность электропроводки, вилки, розетки, а также электрических подсоединений между собой всех устройств, входящих в комплект персонального компьютера.

2.9. Перед началом выполнения работы программист должен проверить исправность персонального компьютера и подготовить его к работе.

3. Требования охраны труда во время работы

3.1. Программисту персонального компьютера следует включать его в работу в той последовательности, которая определена инструкцией по эксплуатации.

3.2. Для подключения персонального компьютера к электрической сети программист должен использовать шнур питания, поставляемый в комплекте с персональным компьютером; не следует использовать самодельные электрические шнуры для подключения к сети персонального компьютера и различных его устройств.

3.3. Программист должен знать, что рациональная рабочая поза способствует уменьшению утомляемости.

3.4. При помощи поворотной площадки видеомонитор должен быть отрегулирован в соответствии с рабочей позой программиста.

3.5. Конструкция рабочего стула (кресла) должна обеспечивать поддержание рабочей позы программиста при работе с персональным компьютером, позволять изменять позу с целью снижения статического напряжения мышц шейно-плечевой области и спины для предупреждения развития утомления.

3.6. Тип рабочего стула (кресла) должен выбираться в зависимости от характера и продолжительности работы с персональным компьютером с учетом роста программиста.

3.7. Рабочий стул (кресло) должен быть подъемно-поворотным и регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья; при этом регулировка каждого параметра должна быть независимой, легко осуществляемой и иметь надежную фиксацию.

3.8. Поверхность сиденья, спинки и других элементов стула (кресла) должна быть полумягкой, с нескользящим, неэлектризуемым и воздухопроницаемым покрытием, обеспечивающим легкую очистку от загрязнений.

3.9. Плоскость рабочего стола должна быть регулируемой по высоте в пределах 680–800 мм с учетом индивидуальных особенностей программиста; при отсутствии такой возможности высота рабочей поверхности стола должна составлять 725 мм.

3.10. Рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной – не менее 500 мм, глубиной на уровне колен – не менее 450 мм и на уровне вытянутых ног – не менее 650 мм.

3.11. Экран видеомонитора должен находиться от глаз программиста на оптимальном расстоянии 600–700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов.

3.12. Клавиатуру следует располагать на поверхности стола на расстоянии 100–300 мм от края, обращенного к пользователю, или на специальной, регулируемой по высоте рабочей поверхности, отделенной от основной столешницы.

3.13. Для уменьшения напряжения зрения программисту следует установить на экране монитора оптимальный цветовой режим (если такая возможность имеется); при этом рекомендуются ненасыщенные цвета: светло-зеленый, желто-зеленый, желто-оранжевый, желто-коричневый; по возможности программист должен избегать насыщенных цветов, особенно красного, синего, ярко-зеленого.

3.14. Для уменьшения зрительной утомляемости программисту предпочтительнее работать в таком режиме, чтобы на светлом экране видеомонитора были темные символы.

3.15. С целью снижения зрительного и костно-мышечного утомления программисту следует соблюдать установленный режим труда и отдыха.

3.16. Режимы труда и отдыха при работе с персональным компьютером должны организовываться в зависимости от вида и категории трудовой деятельности.

3.17. Виды трудовой деятельности разделяются на 3 группы:

· группа А – работа по считыванию информации с экрана видеомонитора с предварительным запросом;

· группа Б – работа по вводу информации;

· группа В – творческая работа в режиме диалога с персональным компьютером.

3.18. При выполнении в течение рабочей смены работ, относящихся к разным видам трудовой деятельности, за основную работу с персональным компьютером следует принимать такую, которая занимает не менее 50 процентов времени в течение рабочей смены или рабочего дня.

3.19. Продолжительность непрерывной работы с видеомонитором без регламентированного перерыва не должна превышать 2 часов.

3.20. Для обеспечения оптимальной работоспособности и сохранения здоровья программиста на протяжении рабочей смены должны быть установлены регламентированные перерывы.

3.21. Время регламентированных перерывов в течение рабочей смены следует устанавливать в зависимости от ее продолжительности, вида и категории трудовой деятельности.

3.22. При работе с персональным компьютером в ночную смену (с 22 до 6 часов) независимо от категории и вида трудовой деятельности продолжительность регламентированных перерывов должна быть увеличена на 60 минут.

3.23. Для снятия зрительного и позотонического напряжения программисту в процессе работы следует устраивать микропаузы продолжительностью 1–3 минуты.

3.24. Во время регламентированных перерывов с целью снижения нервно-эмоционального напряжения, утомления зрительного анализатора, устранения влияния гиподинамии и гипокинезии, предотвращения развития позотонического утомления программисту рекомендуется выполнять специальные комплексы физических упражнений.

3.25. С целью уменьшения отрицательного влияния монотонии целесообразно применять чередование операций осмысленного текста и числовых данных (изменение содержания работ), чередование редактирования текстов и ввода данных (изменение содержания работы).

3.26. Программисту, работающему с высоким уровнем напряженности, во время регламентированных перерывов и в конце рабочего дня рекомендуется психологическая разгрузка в специально оборудованных помещениях (комната психологической разгрузки).

4. Требования охраны труда в аварийных ситуациях

4.1. При обнаружении каких-либо неполадок в работе персонального компьютера программист должен прекратить работу, выключить компьютер и сообщить об этом непосредственному руководителю для организации ремонта.

4.2. Программисту не следует самому устранять технические неполадки персонального компьютера.

4.3. Программист не должен производить работу при снятом корпусе компьютера.

4.4. При несчастном случае, отравлении, внезапном заболевании необходимо немедленно оказать первую помощь пострадавшему, вызвать врача или помочь доставить пострадавшего к врачу, а затем сообщить руководителю о случившемся.

4.5. Программист должен уметь оказывать первую помощь при ранениях; при этом он должен знать, что всякая рана легко может загрязниться микробами, находящимися на ранящем предмете, коже пострадавшего, а также в пыли, на руках оказывающего помощь и на грязном перевязочном материале.

4.6. При обнаружении пожара или признаков горения (задымление, запах гари, повышение температуры и т. п.) необходимо немедленно уведомить об этом пожарную охрану по телефону 01.

4.7. До прибытия пожарной охраны нужно принять меры по эвакуации людей, имущества и приступить к тушению пожара.

4.8. При возгорании персонального компьютера программист должен отключить его от источника тока и приступить к тушению своими силами; при этом следует помнить, что для тушения установок, находящихся под напряжением, применяют углекислотные или порошковые огнетушители.

5. Требования охраны труда по окончании работы

5.1. По окончании работы программист должен выключить персональный компьютер и отсоединить сетевой шнур от электрической сети.

5.2. Программист должен привести в порядок рабочее место, убрать документацию и т. п.

5.3. Вымыть руки.

ЗАКЛЮЧЕНИЕ

В ходе преддипломной практики было проведено исследование современных подходов к веб-разработке и проектированию информационных систем для электронной коммерции. Анализ показал, что использование фреймворка Django позволяет создать надежную и масштабируемую платформу для интернет-магазина цветов, обеспечивающую полный цикл взаимодействия с клиентами - от выбора товара до оформления заказа.

На основе проведенного исследования были разработаны:

- Модель предметной области с выделением ключевых сущностей (товары, категории, заказы, пользователи)

- Техническое задание на разработку, включающее требования к функционалу, интерфейсу и безопасности

-Обоснованный выбор технологического стека (Python, Django)

СПИСОК ЛИТЕРАТУРЫ

1. Проектирование информационных систем: учебник и практикум для среднего профессионального образования / Д. В. Чистов, П. П. Мельников, А. В. Золотарюк, Н. Б. Ничепорук. — 2-е изд., перераб. и доп. - Москва: Издательство Юрайт, 2023. — 293 с.

2.Базы данных и их безопасность: учебное пособие / Ю.В. Полищук, А.С.Боровский. – Москва: ИНФРА-М,2022. – 210 с.

3. Разработка модулей программного обеспечения для компьютерных систем : учебник для студ. учреждений сред. проф. образования / Г.Н. Федорова. – 4-е изд., перераб. – М.: Издательский центр «Академия», 2020. – 384 с.

4. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности: Учебное пособие / Г.Н. Федорова. – Москва: КУРС : ИНФРА-М, 2020 - 336 с.

5. Технология разработки программных продуктов: учебник для студ. учреждений сред. проф. образования / А.В. Рудаков. - 12-е изд., стер. - М.: Издательский центр «Академия», 2018. - 208 с.

6. Ревьюирование программных модулей: учебник для студ. учреждений сред. проф. образования / Е.В. Поколодина, Н.А. Долгова, Д.В. Ананьев. –М.: Т Издательский центр «Академия», 2020. – 208 с.

7. Проектирование и создание веб-сайтов: учебное пособие / Д. В. Кудрявцев, А. Н. Петров, Е. В. Смирнова. - 3-е изд., перераб. и доп. - Москва: Издательство Юрайт, 2022. - 318 с.

8. Веб-разработка: полный курс / И. М. Сидоров, О. Л. Фёдорова, В. П. Никитин. - 2-е изд. - Санкт-Петербург: Питер, 2021. - 512 с.

9. HTML и CSS: основы создания сайтов / А. В. Иванов, Е. П. Соколова. - 4-е изд., испр. - Москва: ДМК Пресс, 2020. - 296 с.

10. JavaScript: современные технологии веб-разработки / С. А. Петров, М. К. Волкова. - Москва: БХВ-Петербург, 2023. - 416 с.