СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Управление пользователями веб-приложений - одна из важнейших задач в современной среде, особенно с учетом растущих требований к безопасности, доступности и масштабируемости системы. По мере увеличения количества пользователей и объема данных администраторы сталкиваются с необходимостью эффективно управлять учетными записями, контролировать права доступа и отслеживать активность. Это требует внедрения многофункциональных инструментов, сочетающих в себе гибкость, удобство и высокую производительность.

Для решения этой задачи мы предлагаем разработать композитное программное обеспечение на основе современного технологического стека, включающего в себя сервер и клиент. Основой системы является серверная часть, разработанная с использованием фреймворка Django. Это комплексное решение предоставляет широкие возможности для манипулирования данными, интеграции API и создания интерфейсов управления. Серверная часть обеспечивает централизованное хранение информации о пользователях, управление учетными записями и CRUD-операции для полного цикла манипулирования данными.

Для удобного взаимодействия с системой пользователи и администраторы могут использовать веб-интерфейс, построенный на HTML, CSS и JavaScript. Этот интерфейс предоставляет доступ к основным функциям системы: просмотр списков пользователей, поиск, фильтрация, редактирование учетных записей и управление ролями. Кроме того, будет реализована система пейджинга, позволяющая обрабатывать большие объемы данных без ущерба для производительности.

Интеграция API, построенного на основе Django REST Framework, позволит взаимодействовать с клиентскими приложениями. Безопасная передача данных обеспечивается за счет поддержки методов

аутентификации и авторизации для защиты учетных записей. Это особенно важно, когда необходимо обрабатывать конфиденциальную информацию. Интерфейс управления системой дополняет функциональность, предоставляя возможность отслеживать действия пользователей, анализировать записи и обрабатывать запросы на изменение данных. Разработанный инструмент полезен как для небольших проектов, так и для крупных систем с высокой нагрузкой. Таким образом, предлагаемая система управления пользователями на базе Django представляет собой надежное, удобное и безопасное решение для работы с учетными записями. Она станет эффективным инструментом для администраторов и обеспечит соответствие современным требованиям веб-разработки.

1. ОБЩАЯ ЧАСТЬ

1.1. Цель разработки

Целью разработки программного комплекса является создание многофункциональной системы, обеспечивающей эффективное, безопасное и удобное управление личными финансами пользователей через централизованную серверную платформу на базе Django и кроссплатформенное приложение Flutter. Это позволит пользователям легко отслеживать доходы и расходы, создавать бюджеты, и оптимизировать финансовые операции, а администраторам – контролировать доступ к данным и управлять системой.

1.2. Средства разработки

Программные средства, используемые для проектирования, разработки и тестирования программного комплекса представлены в таблице 1.

Таблица 1 - Программные средства

№	Тип средства	Название средства	Назначение
1	Операционная система	macOS 15.1.1	Взаимодействие программного обеспечения и пользователя
2	СУБД	PgAdmin4	Разработка базы данных
3	Инструментальные средства разработки программного решения	PyCharm	Разработка сайта
5	Браузер	Arc	Использование веб-ресурсов
6	Текстовый редактор	Microsoft Office профессиональный плюс 2022	Написание документации

В качестве средств вычислительной техники для проектирования, разработки и тестирования программного комплекса использовались MacBook PRO 14 M3 PRO

Таблица 2 - Технические характеристики

№	Тип средства	Название средства
Apple MacBook Pro 14		
1	Разрешение экрана:	3024 x 1964

2	Тип видеокарты:	Встроенная
3	Размер экрана:	14
4	Оперативная память:	8ГБ
5	Графический процессор:	Apple M3 Pro (14 ядер)

2. СПЕЦИАЛЬНАЯ ЧАСТЬ

2.1. Постановка задачи

Разработать комплексное программное решение для управления и учета товаров, заказов и платежей в онлайн-магазине киберспортивной атрибутики.

2.1.1. Входные данные предметной области

Программная система интернет-магазина киберспортивного оборудования принимает различные типы данных, которые пользователи и администраторы вводят через текстовые поля, выпадающие списки и формы загрузки файлов. К основным вводимым данным относится информация о пользователе (имя, фамилия, адрес электронной почты, логин, номер телефона, адрес доставки) и информация о товаре (название, описание, цена, количество на складе, изображение, категория, метки).

Для оформления заказа система принимает данные о заказе (выбранные товары, количество, общая цена, способ оплаты и адрес доставки). Каждое поле данных может иметь свои ограничения, такие как проверка формата электронной почты, минимальная и максимальная цена товара, ограничение на количество товаров в корзине, максимальный размер загружаемых изображений. Система также учитывает данные о транзакциях (статус платежа, идентификатор транзакции, сумма платежа и способ оплаты).

Все данные подвергаются соответствующим проверкам и верификации для обеспечения точности и безопасности покупки.

2.1.2. Выходные данные предметной области

Эти данные включают в себя различные категории, такие как информация о пользователе, детали заказа, статус выполнения транзакции, обновления продукта.

Основными выходными данными являются: Информация о пользователе: статус пользователя (активный или неактивный),

подтверждение электронной почты, история заказов и корзин, данные для входа в систему и настройки профиля. Данные о заказе: идентификатор заказа, позиции заказа, количество, цена, статус заказа (оформлен, оплачен или отправлен), способ доставки и оплаты.

Данные о транзакциях: статус платежа (успешный, неудачный), сумма транзакции, дата и время транзакции, уникальный идентификатор транзакции.

Информация о товаре: список товаров в каталоге, текущее наличие, цена, описание, количество на складе, изображение, категория. На выходные данные накладываются те же ограничения, что и на входные, включая максимальную длину текста в полях, форматирование цен и пользовательских данных, а также ограничения на отображение изображений и товаров в результатах поиска. Эти данные отображаются на странице магазина, в разделе заказов, в личном кабинете и в отчетах для менеджеров и сотрудников магазина.

2.1.3. Подробные требования к проекту

1. Серверная часть

1.1. Фреймворк и структура

Серверная часть будет реализована с использованием Django 5.1, обеспечивающего высокую производительность, безопасность и гибкость разработки. Архитектура проекта будет следовать принципам MVC (Model-View-Controller), что позволит разделить логику приложения, представление и управление данными. Структура проекта будет включать следующие компоненты:

Модели: для представления данных и взаимодействия с базой данных. Модели будут включать такие сущности, как Пользователи, Товары, Заказы, Категории, Теги и Отзывы.

Представления (Views): для обработки HTTP-запросов, управления бизнес-логикой и взаимодействия с клиентскими приложениями через RESTful API.

Контроллеры (Views): для обработки логики аутентификации, авторизации, управления сессиями и обработки пользовательских запросов.

Django ORM: для работы с реляционной базой данных и управления миграциями, что обеспечит удобное и эффективное взаимодействие с данными.

1.2. База данных

Для хранения данных будет использоваться реляционная база данных PostgreSQL, которая обеспечивает высокую надежность, масштабируемость и производительность. Основные требования к базе данных включают:

Нормализация: База данных должна быть нормализована до 3-й нормальной формы (3NF) для минимизации избыточности и предотвращения аномалий при добавлении, обновлении и удалении данных.

Таблицы: Создание следующих таблиц для учета различных сущностей:

Пользователи: хранение информации о клиентах и администраторах (ID, имя, фамилия, email, пароль, адрес доставки и т.д.).

Товары: информация о продуктах (ID, название, описание, цена, количество в наличии, изображения, категория, теги).

Категории: категории товаров (ID, название, описание).

Теги: теги для классификации товаров (ID, название).

Заказы: данные о заказах (ID, пользователь, дата создания, статус оплаты, общая сумма, способ оплаты, адрес доставки).

Позиции заказа: детали заказанных товаров (ID, заказ, товар, цена, количество).

Отзывы: отзывы пользователей о товарах (ID, пользователь, товар, рейтинг, комментарий, дата).

Ключи и индексы:

Первичные ключи (PK) для каждой записи.

Внешние ключи (FK) для связи между таблицами (например, связь заказов с пользователями и товарами).

Индексы для ускорения поиска по часто используемым полям, таким как username, email, product_title.

Транзакции: Реализация ACID-транзакций для обеспечения атомарности операций, особенно при обработке заказов и платежей.

1.3. API

Взаимодействие с клиентскими приложениями будет обеспечиваться через RESTful API, реализованный с использованием Django REST Framework. Технические требования к API включают:

REST-архитектура: Каждый API-метод будет следовать принципам REST, используя соответствующие HTTP-методы (GET для получения данных, POST для создания, PUT/PATCH для обновления и DELETE для удаления).

Аутентификация и авторизация: Использование JWT (JSON Web Token) для аутентификации пользователей. Токены будут передаваться в заголовке Authorization с префиксом Bearer. Реализация RBAC (Role-Based Access Control) для разграничения прав доступа между администраторами и клиентами.

Ограничение доступа: Внедрение механизма Rate Limiting для защиты API от DDoS-атак и чрезмерной нагрузки с использованием библиотек, таких как django-ratelimit.

Документация: Автоматическая генерация документации API с использованием инструментов, таких как Swagger или ReDoc, для удобства разработчиков и пользователей API.

Валидация данных: Встроенная валидация данных на уровне сериализаторов для обеспечения корректности и безопасности передаваемых данных.

1.4. Шифрование данных

Для обеспечения безопасности данных в системе будут реализованы следующие меры шифрования:

Шифрование паролей: Все пароли пользователей будут храниться в базе данных в зашифрованном виде с использованием алгоритмов, таких как PBKDF2, bcrypt или Argon2.

Шифрование данных в транзите: Все данные, передаваемые между клиентом и сервером, будут защищены с использованием TLS/SSL (HTTPS), что обеспечит конфиденциальность и целостность передаваемой информации.

Шифрование конфиденциальных данных: для хранения особо чувствительных данных (например, платежной информации) будет использоваться AES-256 шифрование на уровне базы данных.

1.5. Обработка ошибок и логирование

Для повышения надежности системы будут реализованы механизмы обработки ошибок и логирования:

Централизованное логирование: Использование сервисов, таких как Sentry или Logstash, для отслеживания ошибок и аномальных событий в реальном времени.

Обработка ошибок API: API будет возвращать стандартные коды ошибок HTTP (например, 400 Bad Request, 404 Not Found, 500 Internal Server Error) с подробными сообщениями о возникших ошибках.

Журналирование действий пользователей: Логирование всех критических операций (создание, обновление, удаление данных) с идентификацией пользователя и временными метками для последующего аудита и анализа.

Пользовательские страницы ошибок: Разработка пользовательских страниц ошибок (404, 500 и т.д.) для улучшения пользовательского опыта при возникновении проблем.

2. Административный интерфейс

2.1. Разработка на Django Admin

Административный интерфейс будет реализован с использованием встроенного Django Admin, что обеспечит быстрый и удобный доступ к управлению данными без необходимости разработки отдельного интерфейса.

Настройка моделей: Настройка отображения моделей в админке, добавление необходимых полей, фильтров и поиска для удобного управления.

Пользовательские действия: Разработка пользовательских действий для выполнения массовых операций, таких как массовое обновление статусов заказов или массовое удаление пользователей.

Права доступа: Настройка прав доступа для разных групп пользователей (администраторы, менеджеры, сотрудники) с использованием встроенной системы авторизации Django.

Интеграция с API: Обеспечение взаимодействия админки с RESTful API для синхронизации данных между серверной и клиентской частями.

2.2. Функциональность административного интерфейса

Административный интерфейс будет включать следующие ключевые функции:

Управление пользователями: Добавление, редактирование, блокировка и удаление пользователей. Просмотр подробной информации о пользователях, их заказах и активности.

Управление товарами: Создание, редактирование, удаление и каталогизация товаров. Управление категориями и тегами, загрузка изображений продуктов.

Управление заказами: Просмотр всех заказов, изменение статусов оплаты и доставки, управление деталями заказов.

Отчеты и аналитика: Генерация отчетов по продажам, популярности товаров, активности пользователей. Визуализация данных с помощью графиков и диаграмм.

Настройки системы: Управление общими настройками магазина, такими как способы оплаты, способы доставки, налоговые ставки и т.д.

3. Клиентское веб-приложение

3.1. Технологии

Клиентское веб-приложение будет разработано с использованием современных технологий, обеспечивающих высокую производительность и удобство использования:

Фреймворк: Использование Django для серверной части и React.js для фронтенда, что обеспечит динамичный и отзывчивый интерфейс.

Адаптивный дизайн: Реализация адаптивного интерфейса с использованием Bootstrap 5 для обеспечения корректного отображения на различных устройствах.

Интерактивные элементы: Внедрение динамических компонентов, таких как карусели, модальные окна и интерактивные формы для улучшения пользовательского опыта.

3.2. Функциональность клиентского веб-приложения

Клиентское приложение будет включать следующие основные функции:

Регистрация и аутентификация: Возможность регистрации новых пользователей, входа в систему, восстановления пароля и управления профилем.

Просмотр товаров: Каталог товаров с возможностью фильтрации по категориям и тегам, сортировки по цене, популярности и новизне.

Поиск товаров: Интеграция мощной системы поиска с поддержкой автодополнения и фильтрации результатов.

Корзина и оформление заказа: Добавление товаров в корзину, управление количеством, оформление заказа с выбором способа оплаты и доставки.

Оплата заказов: Интеграция с платежными системами для безопасной и быстрой оплаты заказов. Эмуляция процесса оплаты для тестирования.

Отслеживание заказов: Возможность просмотра статуса текущих заказов, истории заказов и деталей каждого заказа.

Отзывы и рейтинги: Возможность оставлять отзывы и рейтинги для приобретенных товаров, просмотр отзывов других пользователей.

Личный кабинет: Управление личными данными, адресами доставки, настройками уведомлений и предпочтениями.

4. Безопасность

4.1. Аутентификация и авторизация

Для обеспечения безопасности пользовательских данных и предотвращения несанкционированного доступа будут реализованы следующие меры:

RBAC (Role-Based Access Control): Разграничение прав доступа на основе ролей пользователей (администратор, менеджер, клиент) для ограничения доступа к определенным разделам и функциям системы.

Двухфакторная аутентификация (2FA): Возможность включения 2FA для повышения уровня безопасности учетных записей пользователей.

4.2. Защита от атак

Для защиты системы от различных типов атак будут применены следующие меры:

CSRF защита: Внедрение защиты от межсайтовых подделок запросов с использованием встроенных средств Django.

XSS защита: Экранирование вводимых пользователями данных и использование Content Security Policy (CSP) для предотвращения атак через межсайтовые скрипты.

SQL-инъекции: Использование ORM Django для предотвращения SQL-инъекций путем параметризации запросов.

Captcha: Внедрение CAPTCHA в формы регистрации и авторизации для предотвращения автоматических атак.

HTTPS: Обязательное использование HTTPS для всех соединений между клиентом и сервером, обеспечивающее шифрование данных в транзите.

5. Производительность и масштабируемость

5.1. Оптимизация производительности

Для обеспечения высокой производительности системы при увеличении нагрузки будут реализованы следующие подходы:

Кэширование: Использование кэширования на уровне базы данных и фронтенда с помощью Redis или Memcached для ускорения доступа к часто запрашиваемым данным.

Оптимизация запросов: Минимизация количества запросов к базе данных, использование методов select_related и prefetch_related для уменьшения числа SQL-запросов.

Асинхронная обработка: Внедрение асинхронных задач с использованием Celery для обработки длительных операций, таких как отправка email-уведомлений и генерация отчетов.

2.2. Внешняя спецификация

2.2.1. Описание задачи

Серверная часть будет разрабатываться с использованием фреймворка Django, что обеспечит высокую производительность, безопасность и масштабируемость системы. Основные компоненты серверной части включают:

Административный интерфейс:

Административный интерфейс является ключевым инструментом для управления онлайн-магазином киберспортивной атрибутики. Он предоставляет администраторам полный доступ к функционалу системы, позволяя эффективно управлять пользователями, товарами, заказами и платежами. Интерфейс будет интуитивно понятным и удобным для навигации, с возможностями фильтрации и поиска данных, что упрощает работу с большими объемами информации. Администраторы смогут:

- Управлять правами доступа для пользователей
- Настраивать параметры системы, такие как способы оплаты и доставки
- Выполнять регулярные операции по обновлению данных о товарах и категориях
- Просматривать и обрабатывать заказы, включая изменение их статусов
- Анализировать статистику продаж и активности пользователей через встроенные отчеты и графики

База данных:

Для хранения данных будет разработана база данных с не менее чем 8 таблицами, охватывающими все ключевые аспекты работы системы.

Процесс разработки базы данных включает нормализацию до 3-й нормальной формы, что обеспечит ее гибкость, минимизацию избыточности данных и оптимизацию запросов.

Основные операции над данными:

Реализованы функции добавления, редактирования и удаления данных, а также механизм обработки и сохранения изменений в базе данных с учетом транзакционной и безопасности.

API:

Взаимодействие с клиентскими приложениями будет осуществляться через API, реализованное с использованием REST-архитектуры. API будет поддерживать аутентификацию и авторизацию для безопасного доступа и управления данными.

Функциональность административного интерфейса

Сортировка, фильтрация и поиск:

В административном интерфейсе предусмотрены инструменты для сортировки, фильтрации и поиска данных, что позволяет администраторам эффективно работать с большими массивами информации и быстро находить нужные записи.

Представления (виртуальные таблицы):

Для упрощенного доступа к данным будут созданы представления, которые позволят оптимизировать запросы и повысить удобство работы с базой данных.

Шифрование данных:

В систему будет внедрена защита данных с помощью методов шифрования, как на уровне хранения. Это обеспечит высокую степень безопасности информации.

Резервное копирование и логирование:

Будет разработано параллельное приложение для автоматического резервного копирования данных и логирования действий пользователей.

Это поможет предотвратить потерю данных и повысить безопасность системы.

Экспорт и импорт данных:

Реализована возможность экспорта и импорта данных в популярных форматах, таких как SQL и JSON, что обеспечит гибкость в работе с внешними системами и удобство в анализе данных.

Статистика и диаграммы:

В административном интерфейсе будут реализованы функции для создания статистик по определённым параметрам. Также будет возможность генерировать диаграммы на основе этих данных, что поможет администраторам быстро анализировать текущие финансовые показатели системы.

Хранимые процедуры и функции:

Для повышения эффективности работы с базой данных будут разработаны хранимые процедуры и функции, а также применены транзакции для обеспечения целостности данных при выполнении операций.

Безопасность и масштабируемость

Обеспечение безопасности системы является важной частью разработки. Включены следующие механизмы защиты:

Шифрование данных: Все данные, передаваемые между сервером и клиентским приложением, будут зашифрованы с использованием современных протоколов безопасности.

Защита от атак: Реализация мер защиты от распространённых атак, таких как SQL-инъекции, CSRF, XSS и других уязвимостей.

Система спроектирована с учетом будущего роста и масштабируемости, благодаря использованию Django. Это позволит легко добавлять новые функции, а также обеспечивать стабильную работу при увеличении нагрузки.

После выполнения ролевой деятельности, пользователь может покинуть систему, путем выхода из неё.

2.2.2. Входные и выходные данные

Входные данные программного комплекса для интернет-магазина киберспортивной атрибутики.

Таблица 3 - Входные данные

№	Поле	Тип данных	Ограничения	Формат ввода	Описание
1	2	3	4	5	1
1	name	Строка	[Макс. длина: 255]	Поле ввода	Наименование
2	slug	Строка	[a-z0-9- , 3, 50]	Поле ввода	Slug для фильтра
3	category_name	Строка	[Макс. длина: 255]	Поле ввода	Наименование категории товара
4	category_slug	Строка	[a-z0-9- , 3, 50]	Поле ввода	Slug для категории
5	tag_name	Строка	[Макс. длина: 255]	Поле ввода	Наименование тега товара
6	tag_slug	Строка	[a-z0-9- , 3, 50]	Поле ввода	Slug для тега
7	tag_filters	Строка	[Связь с Filters]	Выбор из списка	Фильтр, связанный с тегом
8	product_tags	Список	[Связь с Tag, минимум 1]	Выбор из списка	Теги, связанные с продуктом
9	product_category	Строка	[Связь с Category]	Выбор из списка	Категория продукта
10	title	Строка	[Макс. длина: 255]	Поле ввода	Наименование продукта
11	product_slug	Строка	[a-z0-9- , 3, 50]	Поле ввода	Slug для продукта
12	image	Строка	[URL, путь к файлу]	Поле ввода	Изображение продукта
13	description	Текст	[Необязательно]	Текстовое поле	Описание продукта
14	price	Десятичное	[0-99999.99, 2]	Поле ввода	Цена продукта
15	available	Булево	[По умолчанию: True]	Выбор из списка	Доступность продукта
16	author	Строка	[А-Я, а-я, 3, 50]	Поле ввода	Автор отзыва
17	rating	Целое	[1, 5]	Поле ввода	Рейтинг продукта

№	Поле	Тип данных	Ограничения	Формат ввода	Описание
1	2	3	4	5	1
18	review_text	Текст	[Необязательно]	Текстовое поле	Текст отзыва
19	user	Строка	[Связь с User, Необязательно]	Выбор из списка	Пользователь, сделавший заказ
20	first_name	Строка	[А-Я, а-я, 3, 60]	Поле ввода	Имя пользователя
21	last_name	Строка	[А-Я, а-я, 3, 60]	Поле ввода	Фамилия пользователя
22	email	Строка	[user@example.com, 5, 254]	Поле ввода	Электронная почта пользователя
23	address	Строка	[Макс. длина: 150]	Поле ввода	Адрес пользователя
24	postal_code	Строка	[Макс. длина: 30]	Поле ввода	Почтовый индекс
25	city	Строка	[Макс. длина: 100]	Поле ввода	Город пользователя
26	paid	Булево	[По умолчанию: False]	Выбор из списка	Статус оплаты заказа
27	order_updated	ДатаВремя	[YYYY-MM-DD HH:MM:SS, Автоматически]	Автоматически	Дата обновления заказа
28	order_created	ДатаВремя	[YYYY-MM-DD HH:MM:SS, Автоматически]	Автоматически	Дата создания заказа
29	product	Строка	[Связь с Product]	Выбор из списка	Товар в заказе
30	price_per_item	Десятичное	[0-99999.99, 2]	Поле ввода	Цена за единицу товара
31	quantity	Целое	[Мин: 1]	Поле ввода	Количество товара в заказе
32	total_cost	Десятичное	[0-99999.99, 2, Автоматически]	Автоматически	Общая стоимость заказа

В таблице 4 представлены выходные данные программного комплекса для интернет-магазина

Таблица 4- Выходные данные

№	Поле	Тип данных	Ограничения	Описание
1	2	3	4	5
1	name	Строка	[Макс. длина: 255]	Вывод наименования

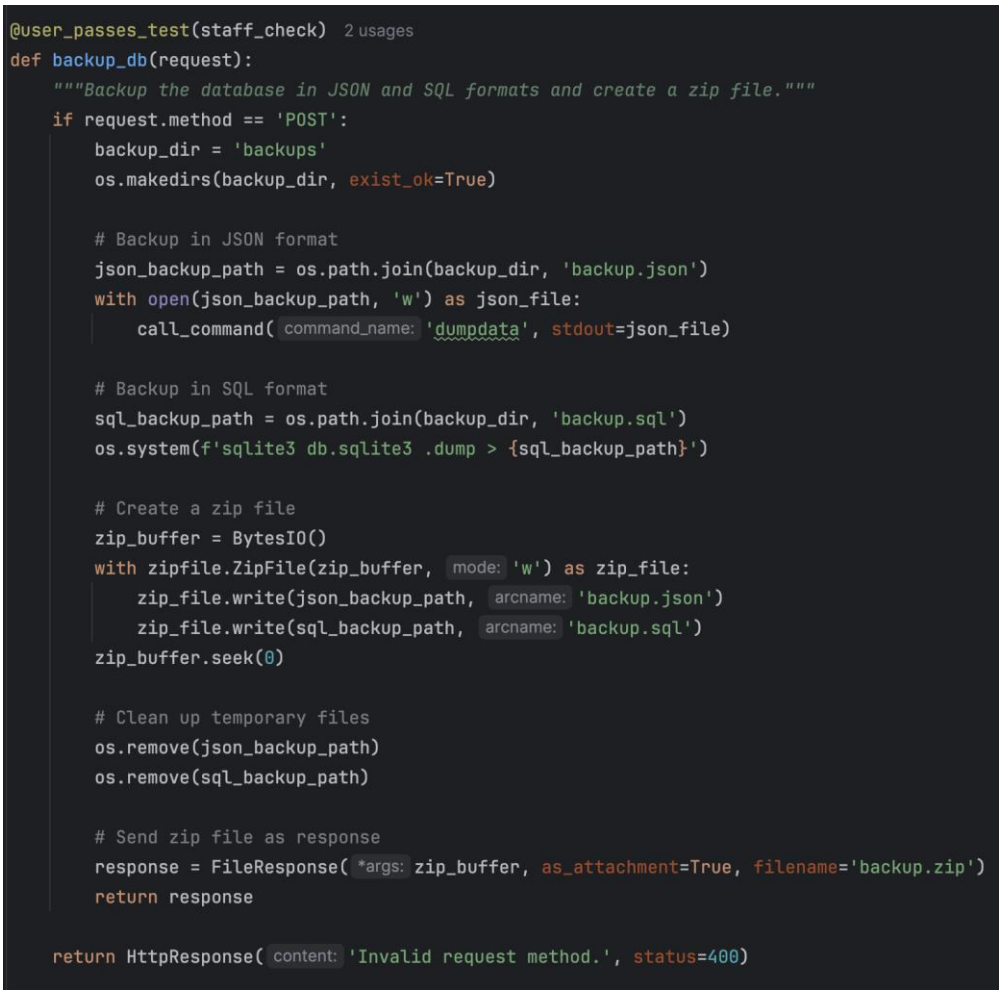
№	Поле	Тип данных	Ограничения	Описание
1	2	3	4	5
				фильтра товара
2	slug	Строка	[a-z0-9- , 3, 50]	Вывод ЧПУ (slug) фильтра
3	category_name	Строка	[Макс. длина: 255]	Вывод наименования категории товара
4	category_slug	Строка	[a-z0-9- , 3, 50]	Вывод ЧПУ (slug) категории
5	tag_name	Строка	[Макс. длина: 255]	Вывод наименования тега товара
6	tag_slug	Строка	[a-z0-9- , 3, 50]	Вывод ЧПУ (slug) тега
7	tag_filters	Строка	[Связь с Filters]	Вывод связанного фильтра тега
8	product_tags	Список	[Связь с Tag]	Вывод тегов, связанных с продуктом
9	product_category	Строка	[Связь с Category]	Вывод категории продукта
10	title	Строка	[Макс. длина: 255]	Вывод наименования продукта
11	product_slug	Строка	[a-z0-9- , 3, 50]	Вывод ЧПУ (slug) продукта
12	image	Строка	[URL, путь к файлу]	Вывод изображения продукта
13	description	Текст	[Необязательно]	Вывод описания продукта
14	price	Десятичное	[0-99999.99, 2]	Вывод цены продукта
15	available	Булево	[По умолчанию: True]	Вывод доступности продукта
16	author	Строка	[А-Я, а-я, 3, 50]	Вывод автора отзыва
17	rating	Целое	[1, 5]	Вывод рейтинга продукта

№	Поле	Тип данных	Ограничения	Описание
1	2	3	4	5
18	review_text	Текст	[Необязательно]	Вывод текста отзыва
19	user	Строка	[Связь с User, Необязательно]	Вывод пользователя, сделавшего заказ
20	first_name	Строка	[А-Я, а-я, 3, 60]	Вывод имени пользователя
21	last_name	Строка	[А-Я, а-я, 3, 60]	Вывод фамилии пользователя
22	email	Строка	[user@example.com, 5, 254]	Вывод электронной почты пользователя
23	address	Строка	[Макс. длина: 150]	Вывод адреса пользователя
24	postal_code	Строка	[Макс. длина: 30]	Вывод почтового индекса
25	city	Строка	[Макс. длина: 100]	Вывод города пользователя
26	paid	Булево	[По умолчанию: False]	Вывод статуса оплаты заказа
27	order_updated	ДатаВремя	[YYYY-MM-DD HH:MM:SS, Автоматически]	Вывод даты обновления заказа
28	order_created	ДатаВремя	[YYYY-MM-DD HH:MM:SS, Автоматически]	Вывод даты создания заказа
29	product	Строка	[Связь с Product]	Вывод товара в заказе
30	price_per_item	Десятичное	[0-99999.99, 2]	Вывод цены за единицу товара
31	quantity	Целое	[Мин: 1]	Вывод количества товара в заказе
32	total_cost	Десятичное	[0-99999.99, 2, Автоматически]	Вывод общей стоимости заказа

2.2.3. Методы

На рисунках 1-2 представлен функционал создания и восстановления резервной копии базы данных с использованием методов backup_db и restore_db. Эти методы обеспечивают удобное

резервирование и восстановление базы данных в форматах JSON и SQL, что делает их гибкими и эффективными инструментами для обеспечения целостности данных.

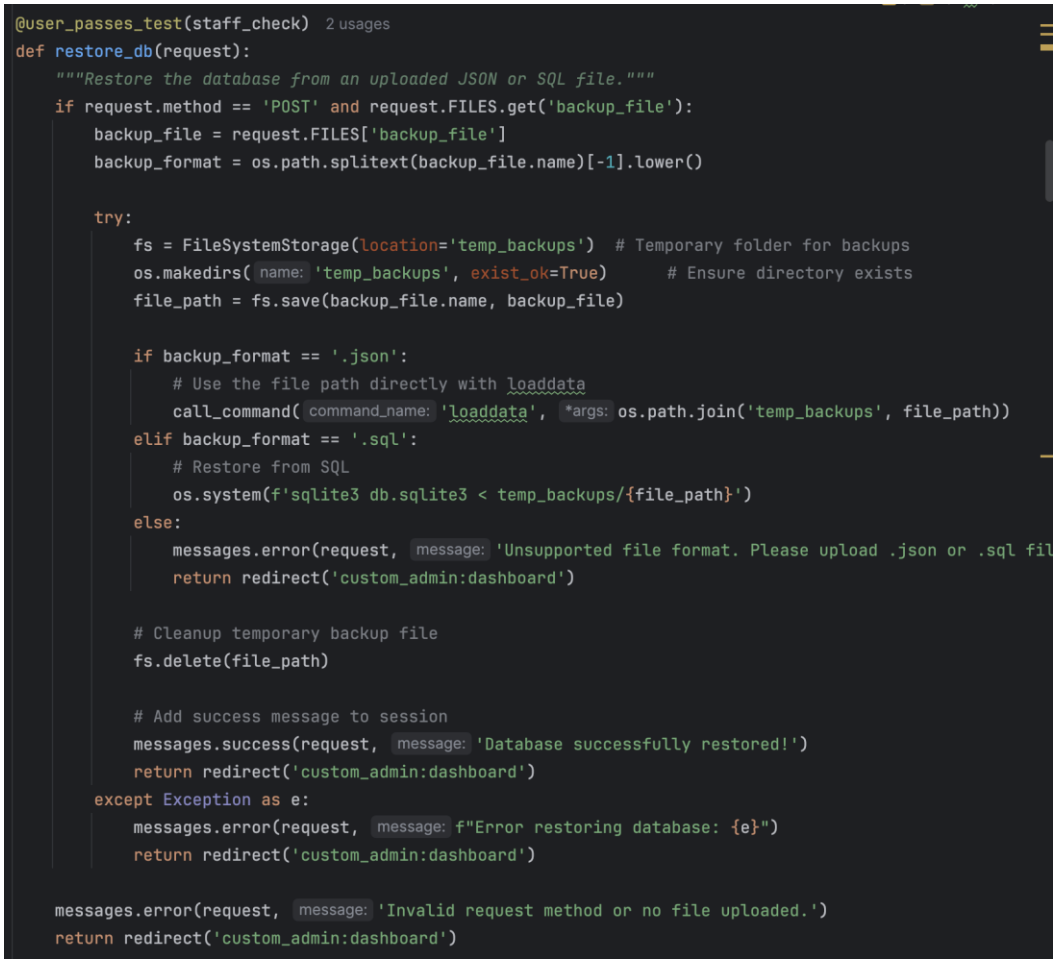


```python
@user_passes_test(staff_check)  2 usages
def backup_db(request):
    """Backup the database in JSON and SQL formats and create a zip file."""
    if request.method == 'POST':
        backup_dir = 'backups'
        os.makedirs(backup_dir, exist_ok=True)

        # Backup in JSON format
        json_backup_path = os.path.join(backup_dir, 'backup.json')
        with open(json_backup_path, 'w') as json_file:
            call_command( command_name: 'dumpdata', stdout=json_file)

        # Backup in SQL format
        sql_backup_path = os.path.join(backup_dir, 'backup.sql')
        os.system(f'sqlite3 db.sqlite3 .dump > {sql_backup_path}')

        # Create a zip file
        zip_buffer = BytesIO()
        with zipfile.ZipFile(zip_buffer,  mode: 'w') as zip_file:
            zip_file.write(json_backup_path,  arcname: 'backup.json')
            zip_file.write(sql_backup_path,  arcname: 'backup.sql')
        zip_buffer.seek(0)

        # Clean up temporary files
        os.remove(json_backup_path)
        os.remove(sql_backup_path)

        # Send zip file as response
        response = FileResponse( *args: zip_buffer, as_attachment=True, filename='backup.zip')
        return response

    return HttpResponse( content: 'Invalid request method.', status=400)
```

Рисунок 1 - backup_db

Метод backup_db предназначен для создания резервной копии базы данных. Он начинает работу с создания директории для хранения резервных копий, гарантируя, что папка backups существует или создается при необходимости. Далее данные базы данных экспортируются в двух форматах: JSON и SQL. Для создания JSON-файла используется встроенная команда dumpdata, которая сохраняет все данные в файл backup.json. Параллельно метод запускает системную команду для генерации SQL-дампа, который сохраняется в файл backup.sql. После этого оба файла резервных копий добавляются в архив backup.zip с использованием модуля zipfile, а временные файлы

backup.json и backup.sql удаляются для экономии места и поддержания чистоты системы. Полученный архив передается пользователю для скачивания с корректным заголовком, обеспечивающим удобное сохранение файла. В случае некорректного метода запроса возвращается HTTP-ответ с кодом 400, указывающий на ошибку запроса.



```
@user_passes_test(staff_check)  2 usages
def restore_db(request):
    """Restore the database from an uploaded JSON or SQL file."""
    if request.method == 'POST' and request.FILES.get('backup_file'):
        backup_file = request.FILES['backup_file']
        backup_format = os.path.splitext(backup_file.name)[-1].lower()

        try:
            fs = FileSystemStorage(location='temp_backups')  # Temporary folder for backups
            os.makedirs( name: 'temp_backups', exist_ok=True)      # Ensure directory exists
            file_path = fs.save(backup_file.name, backup_file)

            if backup_format == '.json':
                # Use the file path directly with loaddata
                call_command( command_name: 'loaddata', *args: os.path.join('temp_backups', file_path))
            elif backup_format == '.sql':
                # Restore from SQL
                os.system(f'sqlite3 db.sqlite3 < temp_backups/{file_path}')
            else:
                messages.error(request, message: 'Unsupported file format. Please upload .json or .sql fil
                return redirect('custom_admin:dashboard')

            # Cleanup temporary backup file
            fs.delete(file_path)

            # Add success message to session
            messages.success(request, message: 'Database successfully restored!')
            return redirect('custom_admin:dashboard')
        except Exception as e:
            messages.error(request, message: f"Error restoring database: {e}")
            return redirect('custom_admin:dashboard')

    messages.error(request, message: 'Invalid request method or no file uploaded.')
    return redirect('custom_admin:dashboard')
```

Рисунок 2 - restore_db

Метод restore_db отвечает за восстановление базы данных из резервной копии, загруженной пользователем в формате .json или .sql. При выполнении метода файл сохраняется во временную директорию temp_backups, после чего его формат определяется по расширению. Если формат файла — JSON, данные импортируются с использованием команды loaddata, которая загружает их обратно в базу данных. Если загружается SQL-файл, система выполняет команду для восстановления базы данных из дампа. После завершения операции временный файл

удаляется для поддержания порядка в системе. В случае успешного восстановления пользователю выводится сообщение о завершении операции, а при возникновении ошибки — сообщение с описанием проблемы для упрощения диагностики.

Таким образом, методы backup_db и restore_db обеспечивают полную поддержку резервного копирования и восстановления данных, сочетая автоматизацию, безопасность и простоту использования. Они позволяют администраторам создавать архивы с данными базы и восстанавливать их при необходимости, что критически важно для поддержания стабильности и надежности работы системы..

2.2.4. Тесты

1. По формальности тестирования.

Тестирование по тестам направлено на проверку функций мобильного приложение на соответствие написанным тест-кейсам. Используется для того, чтобы с помощью некоторых шагов, убедиться в правильной работе программы.

2. По уровню тестирования.

Системное тестирование направлено на проверку работы всей системы на соответствие заявленным требованиям к программному продукту. Позволяет в разработанном приложении выявить ошибки, связанные с взаимодействием различных компонентов системы.

3. По целям.

Функциональное тестирование направлено на проверку реализации функций программного продукта и их точности, что позволяет убедиться в правильной работе разработанного приложения в соответствии с ожиданиями пользователя.

4. По степени автоматизации.

Ручное тестирование направлено на проверку программы без использования дополнительных программных средств, которая позволяет оценить работу разработанного приложения с точки зрения пользователя.

5. По позитивности сценария.

Позитивное тестирование направлено на проверку соответствия функций мобильного приложения ожидаемому поведению. Негативное тестирование направлено на проверку работы программы в случае, когда поведение пользователя отличается от ожидаемого, что позволяет в разработанном мобильном приложении

2.2.5. Контроль целостности данных

Проверка корректности работы БД.

В процессе разработки были реализованы методы валидации входных данных в информационной системе, которые обеспечивают целостность этих входных данных.

Контроль целостности, описывающих ситуации и реакции приложения на выполнения функций представлен в таблице

Таблица 5 - Контроль целостности данных

№	Ситуация	Аномалия	Реакция	Примечание
1	2	3	4	5
1	Регистрация	Введён существующий логин и/или введены некорректные, согласно ограничениям, данные в соответствующие поля	Приложение обработает ошибку с базы данных и выведет пользователю диалоговое окно с содержанием ошибки	Регистрация возможна только с уникальным логином, почтой и валидными полями, соответствующим ограничениям полей
2	Добавление пользователя администратором	Не введены логин и/или пароль	Приложение возвращает фокус на пустое поле, данные не добавляются	Данные при добавлении нового пользователя не могут быть пустыми
		Введён существующий логин и/или	Приложение обработает ошибку с базы	Добавление нового пользователя

№	Ситуация	Аномалия	Реакция	Примечание
1	2	3	4	5
		введены некорректные, согласно ограничениям, данные в соответствующие поля	данных и выведет пользователю диалоговое окно с содержанием ошибки	возможна только с уникальным логином, почтой и валидными полями, соответствующим ограничениям полей

2.3. Проектирование

2.3.1. Схема архитектуры приложения

Компонент пользовательский интерфейс виден пользователю и взаимодействует с ним.

На Рисунке 5 представлена архитектурная схема сайта.

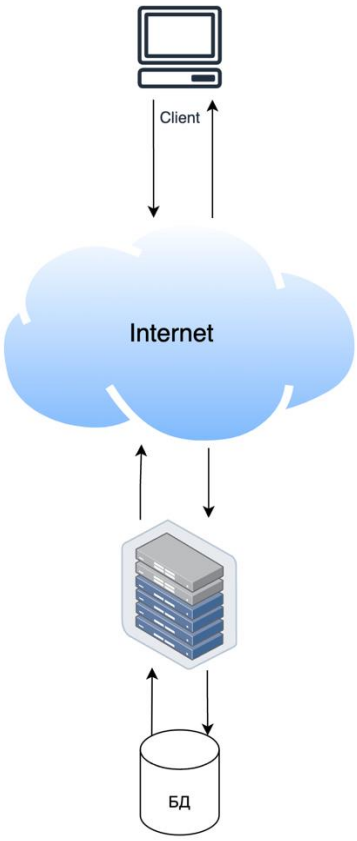


Рисунок 3 – Схема архитектуры приложения

На рисунках 6-11 изображена Idef0 диаграмма комплексного программного решения для управления и учета персональных денежных средств:

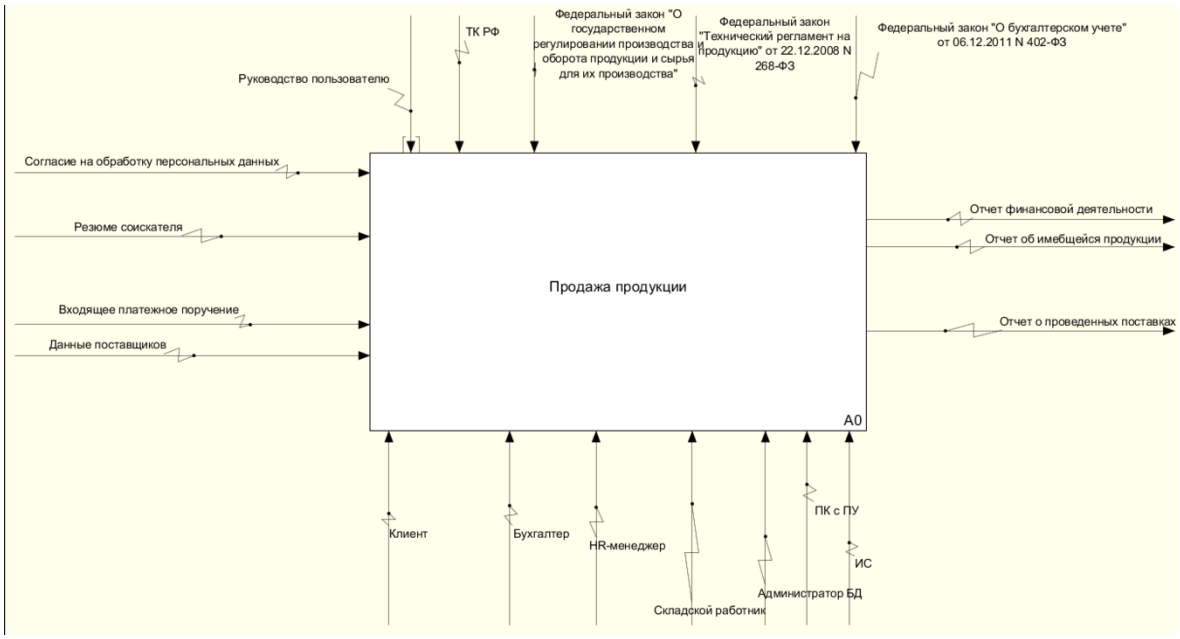


Рисунок 4 – A0 процесс «Управление личными финансами»

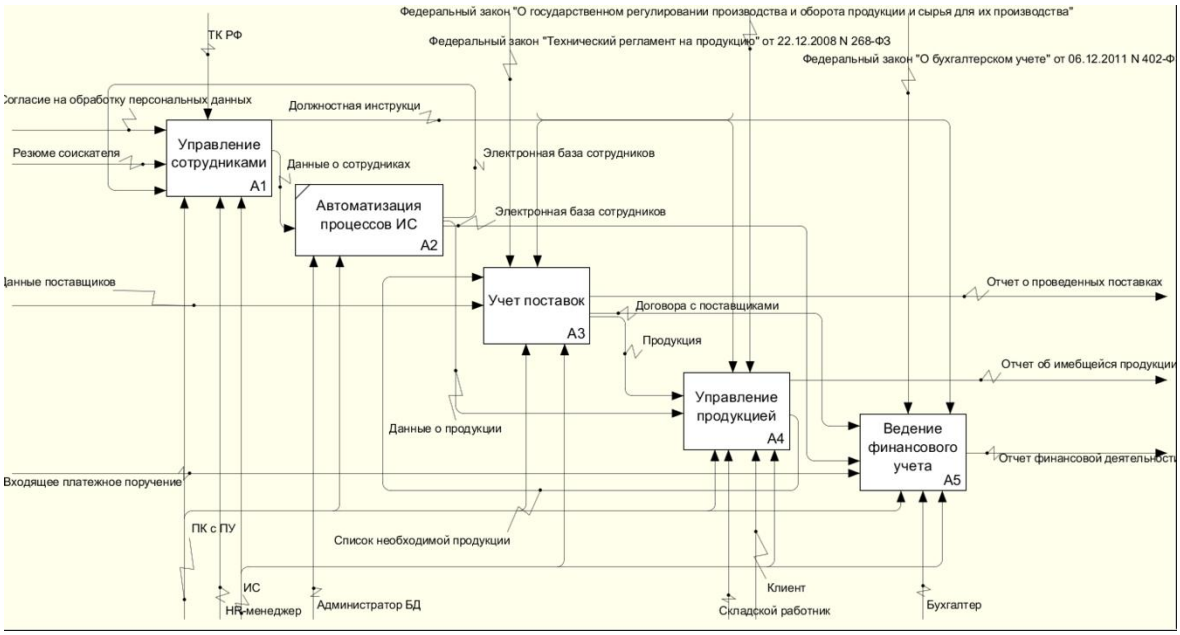


Рисунок 5 – Декомпозиция процесса «Управление личными финансами»

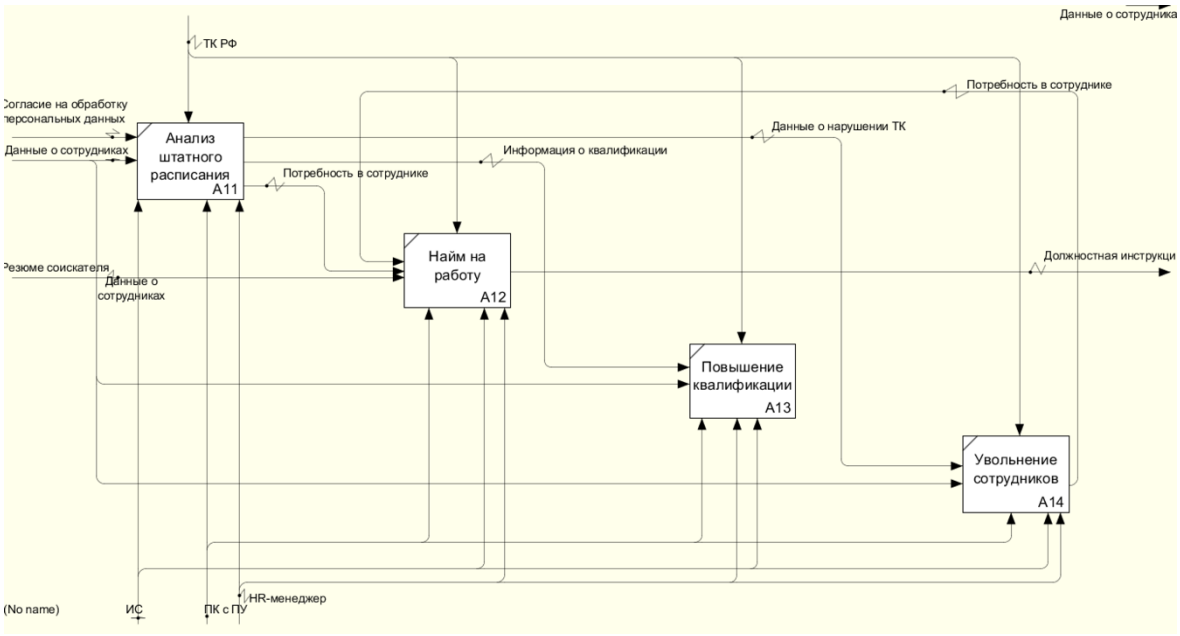


Рисунок 6 – Декомпозиция процесса «Управление сотрудками»

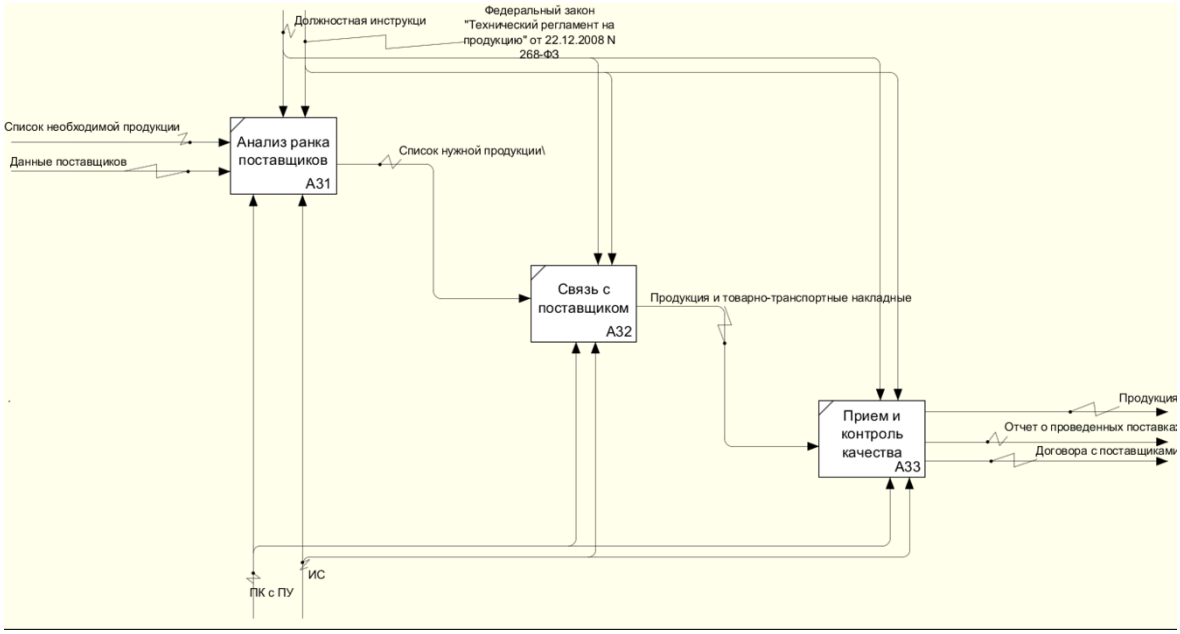


Рисунок 7 – Декомпозиция процесса «Учет поставок»

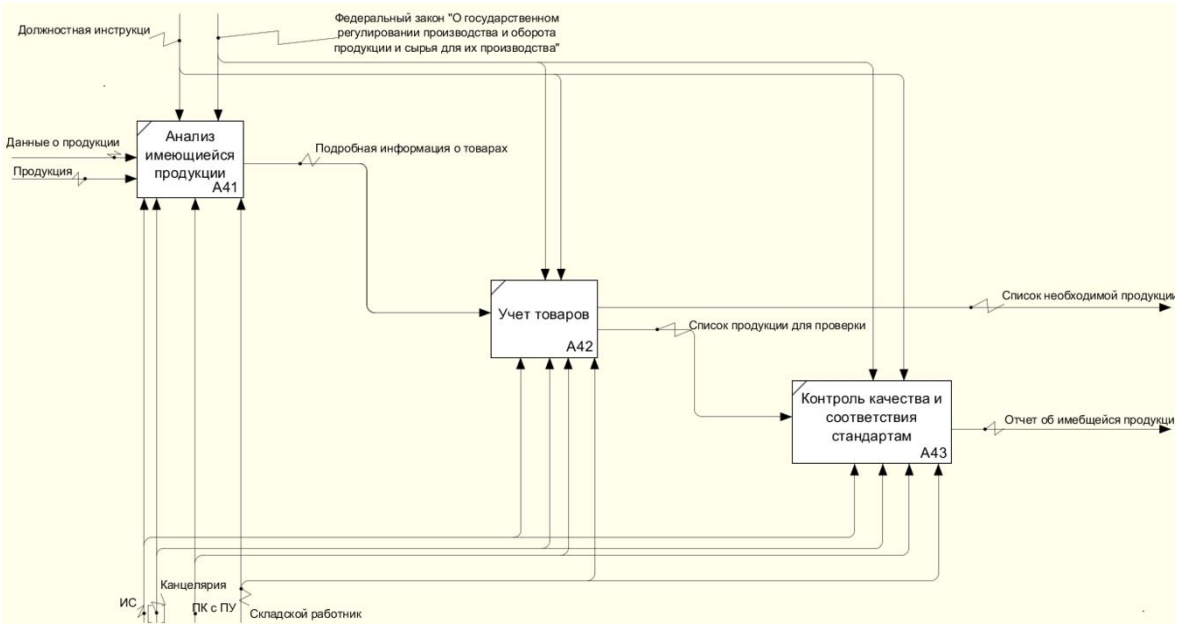


Рисунок 8 – Декомпозиция процесса «Управлением продукцией»

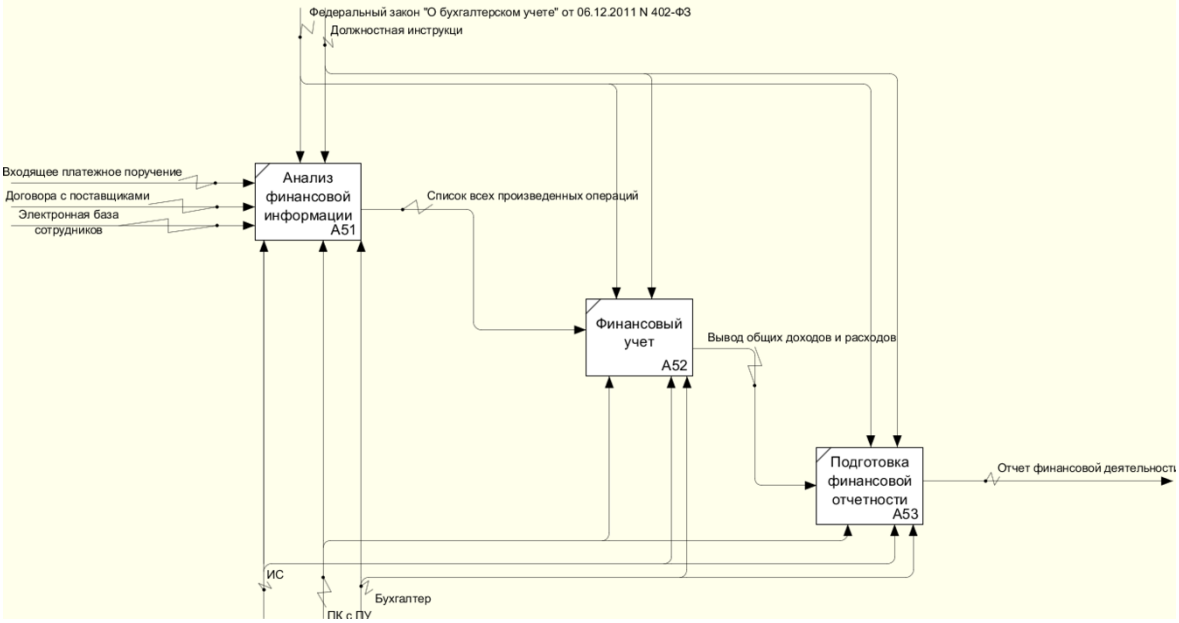


Рисунок 9 – Декомпозиция процесса «Введения финансового учёта»

На рисунке 10 представлена диаграмма прецедентов комплексного программного решения для управления и учета персональных денежных средств:

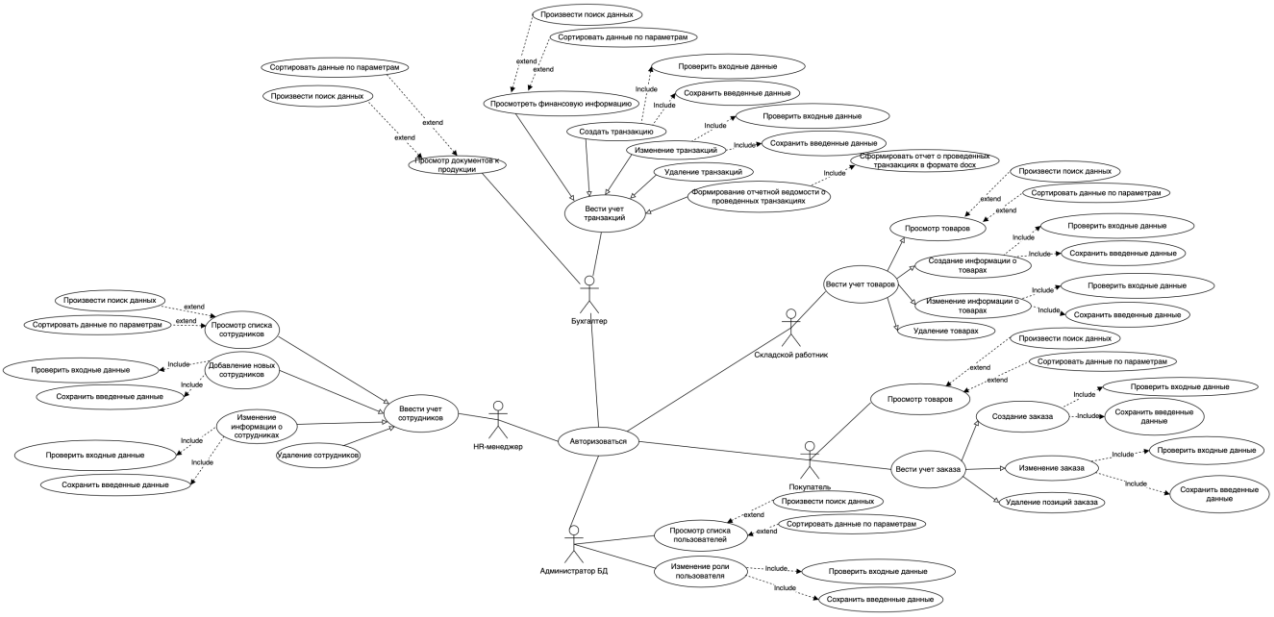


Рисунок 10 – Диаграмма прецедентов

2.3.2. Логическая схема данных

Разработанная логическая модель базы данных, представленная на Рисунке 11, послужила основой для реализации логики манипуляции данными в проектируемой базе данных.

2.3.3. Физическая схема данных

Разработанная физическая модель базы данных представлена на Рисунке 11.

Рисунок 11 – Физическая схема базы данных

2.3.4. Структурная схема

На Рисунках 12- представлен общий план структурной сайта Django, в которой графически показано взаимодействие всех контроллеров, представлений, моделей и их взаимодействие друг с другом.

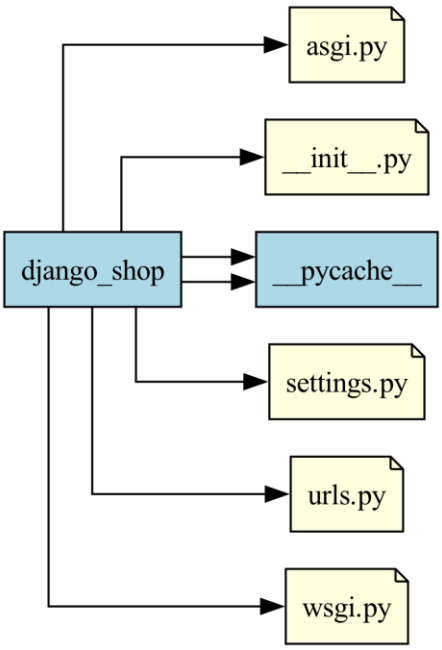


Рисунок 12 - Структурная схема главного приложения проекта

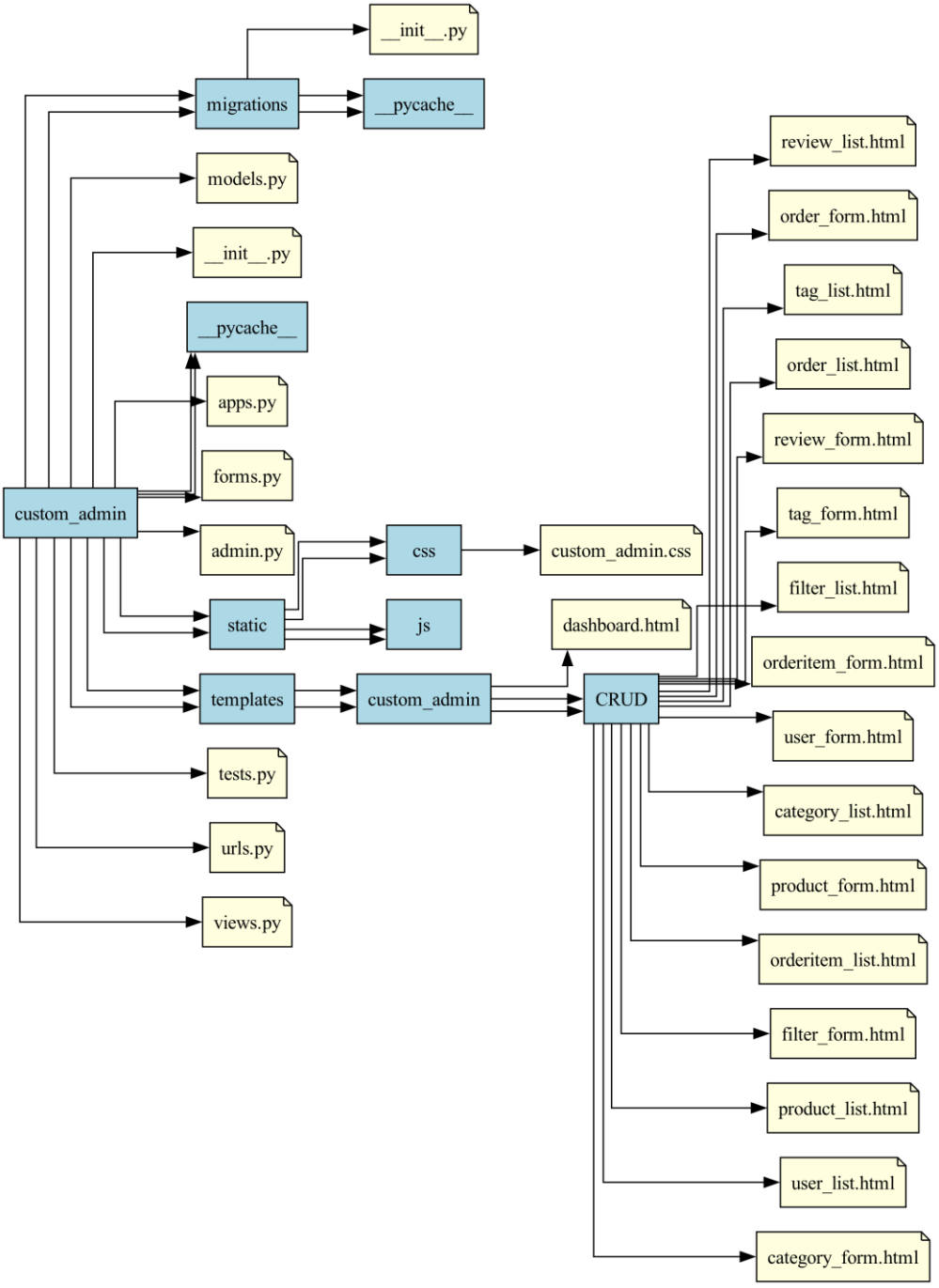


Рисунок 13 – Структурная схема приложения панели админа

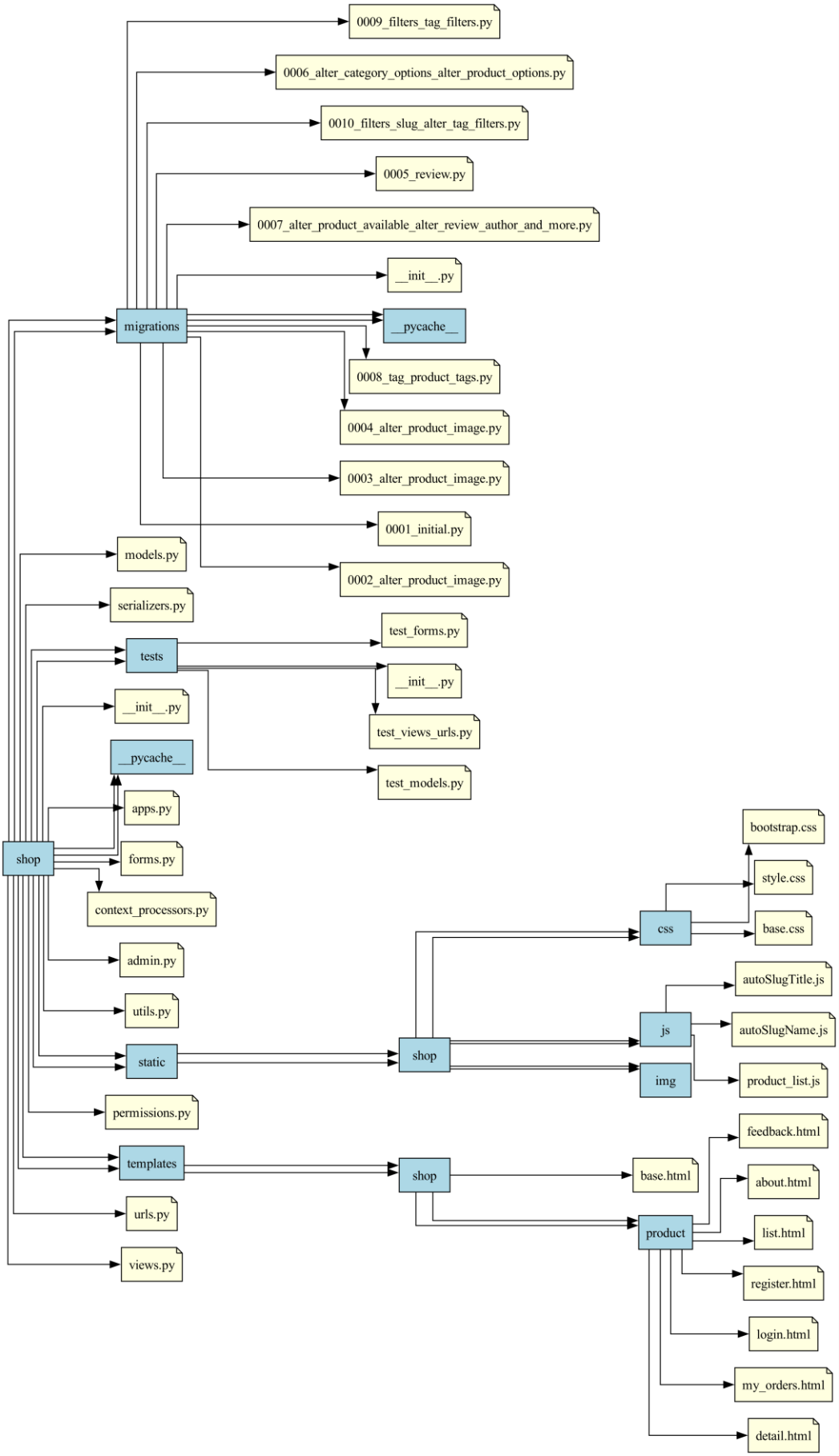


Рисунок 14 – Структурная схема приложение shop

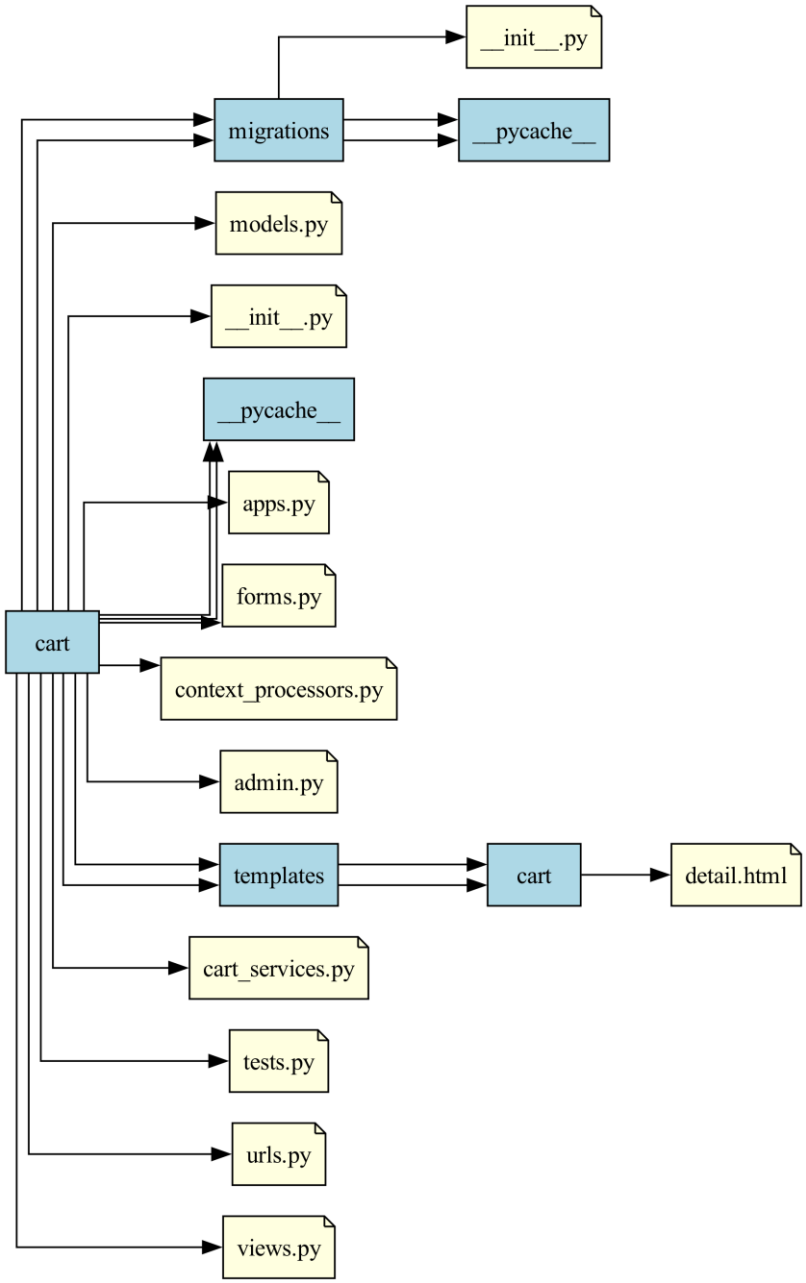


Рисунок 15 – Структурная схема приложение cart

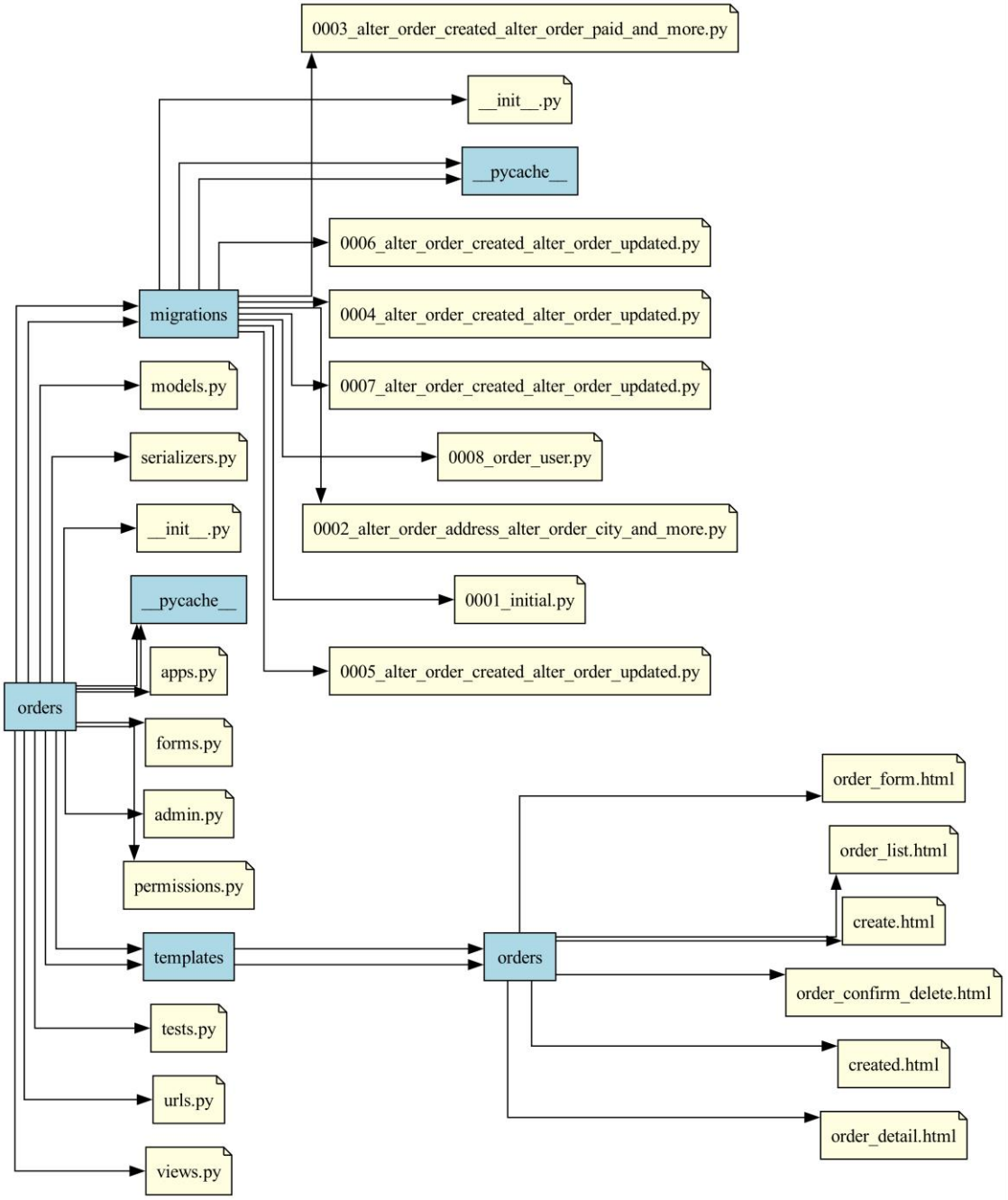


Рисунок 16 - Структурная схема приложение orders

На таблицах 6-8 представлены описания каждого модуля каждой структурной схемы.

Таблица 6 – Описание модулей сервера Django

Номер	Название модуля	Описание
1	/cart/admin.py	Настройка административного интерфейса для управления корзиной в Django admin.
2	/cart/apps.py	Конфигурация приложения корзины, включающая его

		регистрацию в проекте Django.
3	/cart/cart_services.py	Логика сервисов корзины, включая добавление, удаление и обновление товаров.
4	/cart/context_processors.py	Обеспечивает доступ к данным корзины в контексте шаблонов.
5	/cart/forms.py	Формы для обработки ввода данных, связанных с корзиной.
6	/cart/migrations/init.py	Инициализация миграций базы данных для приложения корзины.
7	/cart/models.py	Определение моделей данных для корзины, включая структуру хранимых элементов.
8	/cart/templates/cart/detail.html	Шаблон для отображения содержимого корзины пользователю.
9	/cart/tests.py	Тесты для проверки функционала приложения корзины.
10	/cart/urls.py	Маршрутизация URL для приложения корзины.
11	/cart/views.py	Обработчики представлений для управления корзиной, такие как добавление и удаление товаров.
12	/cart/init.py	Инициализация пакета приложения корзины.
13	/custom_admin/admin.py	Настройка административного интерфейса для кастомизированной панели администратора.
14	/custom_admin/apps.py	Конфигурация приложения для кастомной панели администратора.
15	/custom_admin/forms.py	Формы для обработки данных в панели администратора.
16	/custom_admin/migrations/init.py	Инициализация миграций базы данных для custom_admin.
17	/custom_admin/models.py	Определение моделей данных, используемых в

		кастомной панели администратора.
18	/custom_admin/static/css/custom_admin.css	Стили для кастомного интерфейса администратора.
19	/custom_admin/templates/custom_admin/CRUD/category_form.html	Шаблон формы для добавления/редактирования категорий.
20	/custom_admin/templates/custom_admin/CRUD/category_list.html	Шаблон списка категорий с возможностью управления ими.
21	/custom_admin/templates/custom_admin/CRUD/filter_form.html	Шаблон формы для добавления/редактирования фильтров.
22	/custom_admin/templates/custom_admin/CRUD/filter_list.html	Шаблон для отображения списка фильтров.
23	/custom_admin/templates/custom_admin/CRUD/orderitem_form.html	Шаблон формы для редактирования заказанных товаров.
24	/custom_admin/templates/custom_admin/CRUD/orderitem_list.html	Шаблон для отображения списка заказанных товаров.
25	/custom_admin/templates/custom_admin/CRUD/order_form.html	Шаблон формы для редактирования заказов.
26	/custom_admin/templates/custom_admin/CRUD/order_list.html	Шаблон для отображения списка заказов.
27	/custom_admin/templates/custom_admin/CRUD/product_form.html	Шаблон формы для добавления/редактирования продуктов.
28	/custom_admin/templates/custom_admin/CRUD/product_list.html	Шаблон для отображения списка продуктов.
29	/custom_admin/templates/custom_admin/CRUD/review_form.html	Шаблон формы для добавления/редактирования отзывов.
30	/custom_admin/templates/custom_admin/CRUD/review_list.html	Шаблон для отображения списка отзывов.
31	/custom_admin/templates/custom_admin/CRUD/tag_form.html	Шаблон формы для добавления/редактирования тегов.
32	/custom_admin/templates/custom_admin/CRUD/tag_list.html	Шаблон для отображения списка тегов.
33	/custom_admin/templates/custom_admin/dashboard.html	Шаблон панели управления для кастомного администратора.
34	/custom_admin/tests.py	Тесты для проверки функциональности

		кастомной панели администратора.
35	/custom_admin/urls.py	Маршрутизация URL для кастомного интерфейса администратора.
36	/custom_admin/views.py	Представления для обработки запросов в панели администратора.
37	/custom_admin/init.py	Инициализация пакета кастомного администратора.
38	/django_shop/asgi.py	Конфигурация ASGI для асинхронного запуска проекта Django.
39	/django_shop/settings.py	Основные настройки проекта Django, включая базы данных, приложения и middleware.
40	/django_shop/urls.py	Основная маршрутизация URL проекта Django.
41	/django_shop/wsgi.py	Конфигурация WSGI для запуска проекта Django.
42	/django_shop/init.py	Инициализация основного пакета проекта.
43	/orders/admin.py	Настройка административного интерфейса для управления заказами.
44	/orders/apps.py	Конфигурация приложения заказов.
45	/orders/forms.py	Формы для обработки данных, связанных с заказами.
46	/orders/migrations/0001_initial.py	Миграции базы данных для приложения заказов.
47	/orders/migrations/0002_alter_order_address_alter_order_city_and_more.py	Миграции базы данных для приложения заказов.
48	/orders/migrations/0003_alter_order_created_alter_order_paid_and_more.py	Миграции базы данных для приложения заказов.
49	/orders/migrations/0004_alter_order_created_alter_order_updated.py	Миграции базы данных для приложения заказов.
50	/orders/migrations/0005_alter_order_created_alter_order_updated.py	Миграции базы данных для приложения заказов.
51	/orders/migrations/0006_alter_order_created_alter_order_updated.py	Миграции базы данных для приложения заказов.

52	/orders/migrations/0007_alter_order_created_alter_order_updated.py	Миграции базы данных для приложения заказов.
53	/orders/migrations/0008_order_user.py	Миграции базы данных для приложения заказов.
54	/orders/migrations/__init__.py	Миграции базы данных для приложения заказов.
55	/orders/models.py	Определение моделей данных для заказов.
56	/orders/permissions.py	Определение правил доступа к заказам.
57	/orders/serializers.py	Сериализация данных для работы с API заказов.
58	/orders/templates/orders/create.html	Шаблоны для создания, просмотра и редактирования заказов.
59	/orders/templates/orders/created.html	
60	/orders/templates/orders/order_confirm_delete.html	Шаблоны для создания, просмотра и редактирования заказов.
61	/orders/templates/orders/order_detail.html	
62	/orders/templates/orders/order_form.html	Шаблоны для создания, просмотра и редактирования заказов.
63	/orders/templates/orders/order_list.html	Шаблоны для создания, просмотра и редактирования заказов.
4	/orders/tests.py	Тесты для проверки функциональности заказов.
65	/orders/urls.py	Маршрутизация URL для приложения заказов.
66	/orders/views.py	Обработчики представлений для управления заказами.
67	/orders/init.py	Инициализация пакета заказов.
68	/shop/admin.py	Административный интерфейс для управления магазином.
69	/shop/apps.py	Конфигурация приложения магазина.
70	/shop/context_processors.py	Предоставление глобальных переменных в шаблонах магазина.

71	/shop/forms.py	Формы для ввода данных в магазине.
72	/shop/migrations/0001_initial.py	Миграции для базы данных магазина.
73	/shop/migrations/0002_alter_product_image.py	Миграции для базы данных магазина.
74	/shop/migrations/0003_alter_product_image.py	Миграции для базы данных магазина.
75	/shop/migrations/0004_alter_product_image.py	Миграции для базы данных магазина.
76	/shop/migrations/0005_review.py	Миграции для базы данных магазина.
77	/shop/migrations/0006_alter_category_options_alter_product_options.py	Миграции для базы данных магазина.
78	/shop/migrations/0007_alter_product_available_alter_review_author_and_more.py	Миграции для базы данных магазина.
79	/shop/migrations/0008_tag_product_tags.py	Миграции для базы данных магазина.
80	/shop/migrations/0009_filters_tag_filters.py	Миграции для базы данных магазина.
81	/shop/migrations/0010_filters_slug_alter_tag_filters.py	Миграции для базы данных магазина.
82	/shop/migrations/__init__.py	Миграции для базы данных магазина.
83	/shop/models.py	Определение моделей данных для товаров и категорий магазина.
84	/shop/permissions.py	Правила доступа и сериализация данных для магазина.
85	/shop/serializers.py	Правила доступа и сериализация данных для магазина.
86	/shop/static/shop/css/base.css	Статические файлы: CSS и JS для стилей и функционала магазина.
87	/shop/static/shop/css/bootstrap.css	Статические файлы: CSS и JS для стилей и функционала магазина.
88	/shop/static/shop/css/style.css	Статические файлы: CSS и JS для стилей и функционала магазина.
89	/shop/static/shop/js/autoSlugName.js	Статические файлы: CSS и JS для стилей и функционала магазина.

90	/shop/static/shop/js/autoSlugTitle.js	Статические файлы: CSS и JS для стилей и функционала магазина.
91	/shop/static/shop/js/product_list.js	Статические файлы: CSS и JS для стилей и функционала магазина.
92	/shop/templates/shop/base.html	Шаблоны для страниц магазина: каталог, детали товара, регистрация и личный кабинет.
93	/shop/templates/shop/product/about.html	Шаблоны для страниц магазина: каталог, детали товара, регистрация и личный кабинет.
94	/shop/templates/shop/product/detail.html	Шаблоны для страниц магазина: каталог, детали товара, регистрация и личный кабинет.
95	/shop/templates/shop/product/feedback.html	Шаблоны для страниц магазина: каталог, детали товара, регистрация и личный кабинет.
96	/shop/templates/shop/product/list.html	Шаблоны для страниц магазина: каталог, детали товара, регистрация и личный кабинет.
97	/shop/templates/shop/product/login.html	Шаблоны для страниц магазина: каталог, детали товара, регистрация и личный кабинет.
98	/shop/templates/shop/product/my_orders.html	Шаблоны для страниц магазина: каталог, детали товара, регистрация и личный кабинет.
99	/shop/templates/shop/product/register.html	Шаблоны для страниц магазина: каталог, детали товара, регистрация и личный кабинет.
100	/shop/tests/test_forms.py	Тесты для проверки моделей, форм и представлений магазина.
101	/shop/tests/test_models.py	Тесты для проверки моделей, форм и представлений магазина.
102	/shop/tests/test_views_urls.py	Тесты для проверки моделей, форм и представлений магазина.
103	/shop/tests/__init__.py	Тесты для проверки моделей, форм и представлений магазина.

104	/shop/urls.py	Маршрутизация URL для приложения магазина.
105	/shop/utils.py	Утилиты и вспомогательные функции для магазина.
106	/shop/views.py	Обработчики представлений для отображения и управления данными магазина.
107	/shop/init.py	Инициализация пакета магазина.

2.3.5. Функциональная схема

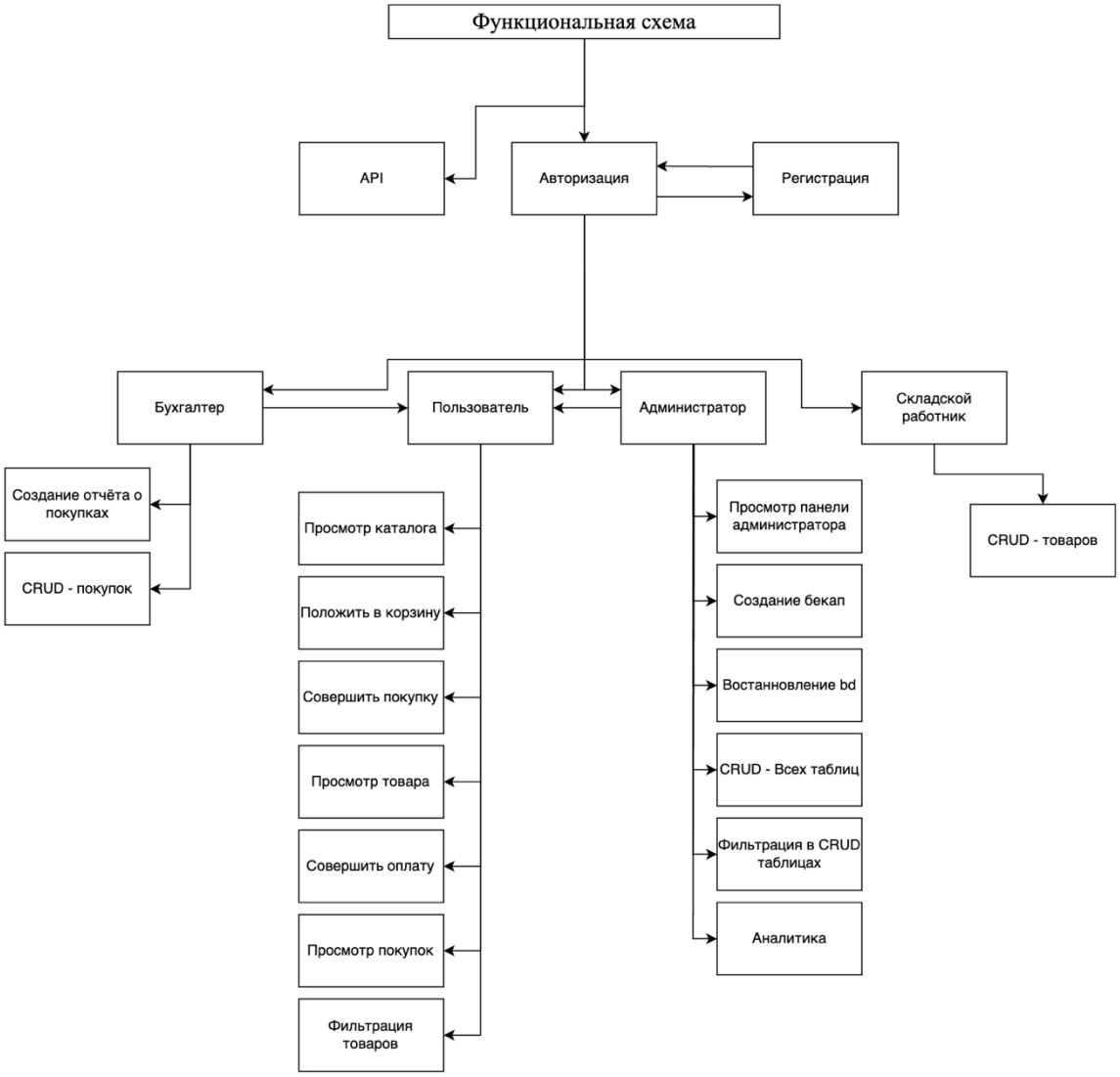


Рисунок 17 – Функциональная схема

2.3.6. Диаграмма классов

На рисунке 15 представлена диаграмма класса приложения. На ней графически изображены классы, которые состоят из полей, методов и свойств.

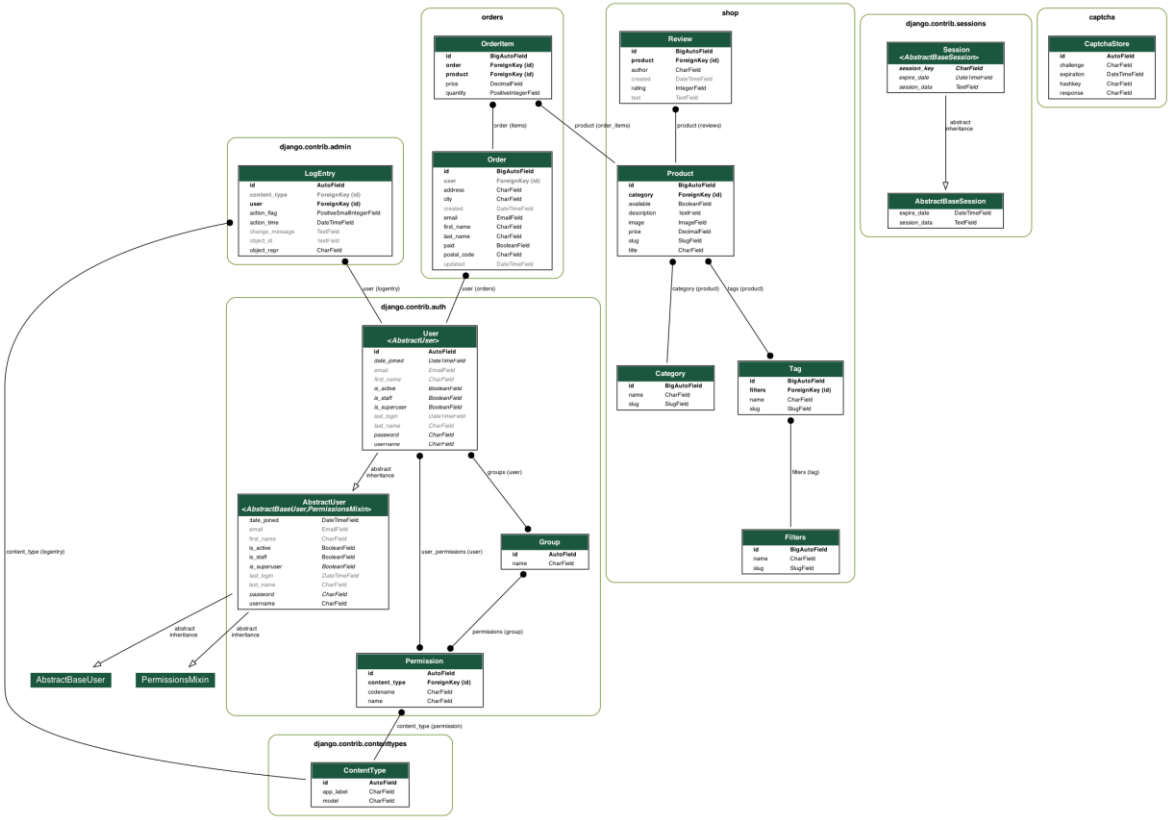


Рисунок 18 – Диаграмма классов Django

В таблицах 9-11 представлены описания классов продемонстрированных диаграмм классов.

Таблица 7 – Описание классов

Модуль	Описание	Количество строк	Размер (Кб)
1	2	3	4
/cart/admin.py	Конфигурация административной панели для модуля корзины, позволяет управлять моделями корзины через Django Admin.	3	0.06
/cart/apps.py	Конфигурация приложения корзины, содержащая настройку имени приложения для	6	0.14

Модуль	Описание	Количество строк	Размер (Кб)
1	2	3	4
	использования в проекте Django.		
/cart/cart_services.py	Содержит бизнес-логику и функции для работы с корзиной покупок: добавление, удаление, подсчет общей стоимости и управление товарами в корзине.	91	3.37
/cart/context_processors.py	Определяет контекстный процессор для передачи данных о корзине покупок в шаблоны на каждой странице.	11	0.29
/cart/forms.py	Форма для ввода и валидации данных корзины пользователем, например, изменение количества товаров.	12	0.28
/cart/migrations/__init__.py	Пустой файл для инициализации пакета миграций приложения корзины.	0	0.0
/cart/models.py	Определяет модели базы данных для корзины покупок.	1	0.1
/cart/templates/cart/detail.html	HTML-шаблон для отображения содержимого корзины покупок, включая список товаров, общую стоимость и формы для управления корзиной.	125	4.94
/cart/tests.py	Тесты для проверки функциональности модуля корзины.	3	0.06
/cart/urls.py	Определяет URL-маршруты для управления корзиной, такие как	10	0.3

Модуль	Описание	Количество строк	Размер (Кб)
1	2	3	4
	добавление и удаление товаров.		
/cart/views.py	Представления для обработки HTTP-запросов, связанных с корзиной: добавление товаров, удаление и просмотр содержимого корзины.	35	1.04
/cart/__init__.py	Пустой файл для инициализации пакета приложения корзины.	0	0.0
/custom_admin/admin.py	Конфигурация панели администратора для модуля custom_admin с кастомизированными настройками управления данными.	8	0.43
/custom_admin/apps.py	Конфигурация приложения custom_admin, определяющая имя и настройки приложения.	5	0.15
/custom_admin/forms.py	Содержит формы для ввода и валидации данных административного интерфейса, таких как добавление или редактирование данных.	40	0.87
/custom_admin/migrations/__init__.py	Пустой файл для инициализации пакета миграций приложения custom_admin.	0	0.0
/custom_admin/models.py	Определение моделей базы данных для административного интерфейса.	3	0.06

Модуль	Описание	Количество строк	Размер (Кб)
1	2	3	4
/custom_admin/static/css/custom_admin.css	Кастомные стили CSS для оформления административного интерфейса custom_admin.	25	0.27
/custom_admin/templates/custom_admin/CRUD/category_form.html	HTML-шаблон для формы создания и редактирования категорий в административном интерфейсе.	30	1.58
/custom_admin/templates/custom_admin/CRUD/category_list.html	HTML-шаблон для отображения списка категорий в административном интерфейсе с возможностью поиска и фильтрации.	121	4.75
/custom_admin/templates/custom_admin/CRUD/filter_form.html	Шаблон формы для создания и редактирования фильтров в административном интерфейсе.	54	1.82
/custom_admin/templates/custom_admin/CRUD/filter_list.html	HTML-шаблон для отображения списка фильтров с опциями управления данными в админке.	109	4.81
/custom_admin/templates/custom_admin/CRUD/order-item_form.html	Шаблон формы редактирования товаров в заказах.	19	0.76
/custom_admin/templates/custom_admin/CRUD/order-item_list.html	Шаблон отображения списка товаров в заказах с возможностью управления ими.	43	1.34
/custom_admin/templates/custom_admin/CRUD/order_form.html	Шаблон формы создания и редактирования заказов в административном интерфейсе.	39	2.12
/custom_admin/templates/custom_admin/CRUD/order_list.html	HTML-шаблон для отображения списка заказов с	123	5.78

Модуль	Описание	Количество строк	Размер (Кб)
1	2	3	4
	возможностью поиска, сортировки и управления.		
/custom_admin/templates/custom_admin/CRUD/product_form.html	Шаблон формы для добавления и редактирования продуктов в административном интерфейсе.	91	4.23
/custom_admin/templates/custom_admin/CRUD/product_list.html	Шаблон отображения списка продуктов с возможностью управления данными и фильтрацией.	130	5.8
/custom_admin/templates/custom_admin/CRUD/review_form.html	Главная страница административного интерфейса с панелью управления и сводной статистикой по данным.	36	1.51
/custom_admin/templates/custom_admin/CRUD/review_list.html	Тесты для проверки функциональности административного интерфейса.	96	4.7
/custom_admin/templates/custom_admin/CRUD/tag_form.html	Определение URL-маршрутов для административного интерфейса.	19	0.87
/custom_admin/templates/custom_admin/CRUD/tag_list.html	Представления для обработки запросов административного интерфейса, включая CRUD-операции для категорий, заказов, продуктов и фильтров.	94	4.48
/custom_admin/templates/custom_admin/dashboard.html	Конфигурация ASGI для развертывания Django-проекта в асинхронной среде.	266	12.09
/custom_admin/tests.py	Основные настройки проекта Django, включая конфигурацию приложений, баз	3	0.06

Модуль	Описание	Количество строк	Размер (Кб)
1	2	3	4
	данных, статических файлов и middleware.		
/custom_admin/urls.py	Глобальные маршруты проекта, связывающие URL-адреса с представлениями.	56	2.5
/custom_admin/views.py	Конфигурация WSGI для развертывания проекта на традиционных веб-серверах.	713	27.25
/custom_admin/__init__.py	Конфигурация административной панели для модуля корзины, позволяет управлять моделями корзины через Django Admin.	0	0.0
/django_shop/asgi.py	Конфигурация приложения корзины, содержащая настройку имени приложения для использования в проекте Django.	16	0.39
/django_shop/settings.py	Содержит бизнес-логику и функции для работы с корзиной покупок: добавление, удаление, подсчет общей стоимости и управление товарами в корзине.	148	4.94
/django_shop/urls.py	Определяет контекстный процессор для передачи данных о корзине покупок в шаблоны на каждой странице.	16	0.9
/django_shop/wsgi.py	Форма для ввода и валидации данных корзины пользователем,	16	0.39

Модуль	Описание	Количество строк	Размер (Кб)
1	2	3	4
	например, изменение количества товаров.		
/django_shop/__init__.py	Пустой файл для инициализации пакета миграций приложения корзины.	0	0.0
/orders/admin.py	Определяет модели базы данных для корзины покупок.	19	0.58
/orders/apps.py	HTML-шаблон для отображения содержимого корзины покупок, включая список товаров, общую стоимость и формы для управления корзиной.	6	0.14
/orders/forms.py	Тесты для проверки функциональности модуля корзины.	23	0.79
/orders/migrations/0001_initial.py	Определяет URL-маршруты для управления корзиной, такие как добавление и удаление товаров.	46	1.86
/orders/migrations/0002_alter_order_address_alter_order_city_and_more.py	Представления для обработки HTTP-запросов, связанных с корзиной: добавление товаров, удаление и просмотр содержимого корзины.	43	1.29
/orders/migrations/0003_alter_order_created_alter_order_paid_and_more.py	Пустой файл для инициализации пакета приложения корзины.	50	1.84
/orders/migrations/0004_alter_order_created_alter_order_updated.py	Конфигурация панели администратора для модуля custom_admin с кастомизированными настройками	22	0.61

Модуль	Описание	Количество строк	Размер (Кб)
1	2	3	4
	управления данными.		
/orders/migrations/0005_alter_order_created_alter_order_updated.py	Конфигурация приложения custom_admin, определяющая имя и настройки приложения.	22	0.64
/orders/migrations/0006_alter_order_created_alter_order_updated.py	Содержит формы для ввода и валидации данных административного интерфейса, таких как добавление или редактирование данных.	22	0.61
/orders/migrations/0007_alter_order_created_alter_order_updated.py	Пустой файл для инициализации пакета миграций приложения custom_admin.	22	0.64
/orders/migrations/0008_order_user.py	Определение моделей базы данных для административного интерфейса.	27	0.77
/orders/migrations/__init__.py	Кастомные стили CSS для оформления административного интерфейса custom_admin.	0	0.0
/orders/models.py	HTML-шаблон для формы создания и редактирования категорий в административном интерфейсе.	48	2.06
/orders/permissions.py	HTML-шаблон для отображения списка категорий в административном интерфейсе с возможностью поиска и фильтрации.	13	0.67
/orders/serializers.py	Шаблон формы для создания и редактирования	24	0.68

Модуль	Описание	Количество строк	Размер (Кб)
1	2	3	4
	фильтров в административном интерфейсе.		
/orders/templates/orders/create.html	HTML-шаблон для отображения списка фильтров с опциями управления данными в админке.	50	2.15
/orders/templates/orders/created.html	Шаблон формы редактирования товаров в заказах.	20	0.54
/orders/templates/orders/order_confirm_delete.html	Шаблон отображения списка товаров в заказах с возможностью управления ими.	13	0.5
/orders/templates/orders/order_detail.html	Шаблон формы создания и редактирования заказов в административном интерфейсе.	17	0.78
/orders/templates/orders/order_form.html	HTML-шаблон для отображения списка заказов с возможностью поиска, сортировки и управления.	60	2.39
/orders/templates/orders/order_list.html	Шаблон формы для добавления и редактирования продуктов в административном интерфейсе.	46	2.02
/orders/tests.py	Шаблон отображения списка продуктов с возможностью управления данными и фильтрацией.	3	0.06
/orders/urls.py	Главная страница административного интерфейса с панелью управления и сводной статистикой по данным.	21	0.85

Модуль	Описание	Количество строк	Размер (Кб)
1	2	3	4
/orders/views.py	Тесты для проверки функциональности административного интерфейса.	100	3.46
/orders/__init__.py	Определение URL-маршрутов для административного интерфейса.	0	0.0
/shop/admin.py	Представления для обработки запросов административного интерфейса, включая CRUD-операции для категорий, заказов, продуктов и фильтров.	44	1.78
/shop/apps.py	Конфигурация ASGI для развертывания Django-проекта в асинхронной среде.	10	1.0
/shop/context_processors.py	Основные настройки проекта Django, включая конфигурацию приложений, баз данных, статических файлов и middleware.	11	0.78
/shop/forms.py	Глобальные маршруты проекта, связывающие URL-адреса с представлениями.	85	3.45
/shop/migrations/0001_initial.py	Конфигурация WSGI для развертывания проекта на традиционных веб-серверах.	42	1.63
/shop/migrations/0002_alter_product_image.py	Конфигурация административной панели для модуля корзины, позволяет управлять моделями корзины через Django Admin.	18	0.41
/shop/migrations/0003_alter_product_image.py	Конфигурация приложения корзины, содержащая	18	0.42

Модуль	Описание	Количество строк	Размер (Кб)
1	2	3	4
	настройку имени приложения для использования в проекте Django.		
/shop/migrations/0004_alter_product_image.py	Содержит бизнес-логику и функции для работы с корзиной покупок: добавление, удаление, подсчет общей стоимости и управление товарами в корзине.	18	0.41
/shop/migrations/0005_review.py	Определяет контекстный процессор для передачи данных о корзине покупок в шаблоны на каждой странице.	29	1.05
/shop/migrations/0006_alter_category_options_alter_product_options.py	Форма для ввода и валидации данных корзины пользователем, например, изменение количества товаров.	21	0.59
/shop/migrations/0007_alter_product_available_alter_review_author_and_more.py	Пустой файл для инициализации пакета миграций приложения корзины.	45	1.57
/shop/migrations/0008_tag_product_tags.py	Определяет модели базы данных для корзины покупок.	31	0.99
/shop/migrations/0009_filters_tag_filters.py	HTML-шаблон для отображения содержимого корзины покупок, включая список товаров, общую стоимость и формы для управления корзиной.	31	1.01
/shop/migrations/0010_filters_slug_alter_tag_filters.py	Тесты для проверки функциональности модуля корзины.	24	0.68
/shop/migrations/__init__.py	Определяет URL-маршруты для	0	0.0

Модуль	Описание	Количество строк	Размер (Кб)
1	2	3	4
	управления корзиной, такие как добавление и удаление товаров.		
/shop/models.py	Представления для обработки HTTP-запросов, связанных с корзиной: добавление товаров, удаление и просмотр содержимого корзины.	105	5.1
/shop/permissions.py	Пустой файл для инициализации пакета приложения корзины.	14	0.66
/shop/serializers.py	Конфигурация панели администратора для модуля custom_admin с кастомизированными настройками управления данными.	33	0.74
/shop/static/shop/css/base.css	Конфигурация приложения custom_admin, определяющая имя и настройки приложения.	366	4.54
/shop/static/shop/css/boot-strap.css	Содержит формы для ввода и валидации данных административного интерфейса, таких как добавление или редактирование данных.	278	4.93
/shop/static/shop/css/style.css	Пустой файл для инициализации пакета миграций приложения custom_admin.	14	0.35
/shop/static/shop/js/au-toSlugName.js	Определение моделей базы данных для	9	0.37

Модуль	Описание	Количество строк	Размер (Кб)
1	2	3	4
	административного интерфейса.		
/shop/static/shop/js/autoSlugTitle.js	Кастомные стили CSS для оформления административного интерфейса custom_admin.	21	0.84
/shop/static/shop/js/product_list.js	HTML-шаблон для формы создания и редактирования категорий в административном интерфейсе.	45	1.98
/shop/templates/shop/base.html	HTML-шаблон для отображения списка категорий в административном интерфейсе с возможностью поиска и фильтрации.	155	8.22
/shop/templates/shop/product/about.html	Шаблон формы для создания и редактирования фильтров в административном интерфейсе.	15	0.28
/shop/templates/shop/product/detail.html	HTML-шаблон для отображения списка фильтров с опциями управления данными в админке.	127	6.32
/shop/templates/shop/product/feedback.html	Шаблон формы редактирования товаров в заказах.	28	0.73
/shop/templates/shop/product/list.html	Шаблон отображения списка товаров в заказах с возможностью управления ими.	120	7.06
/shop/templates/shop/product/login.html	Шаблон формы создания и редактирования заказов в административном интерфейсе.	29	0.89
/shop/templates/shop/product/my_orders.html	HTML-шаблон для отображения списка	46	1.77

Модуль	Описание	Количество строк	Размер (Кб)
1	2	3	4
	заказов с возможностью поиска, сортировки и управления.		
/shop/templates/shop/product/register.html	Шаблон формы для добавления и редактирования продуктов в административном интерфейсе.	31	1.05
/shop/tests/test_forms.py	Шаблон отображения списка продуктов с возможностью управления данными и фильтрацией.	174	7.01
/shop/tests/test_models.py	Главная страница административного интерфейса с панелью управления и сводной статистикой по данным.	198	6.96
/shop/tests/test_views_urls.py	Тесты для проверки функциональности административного интерфейса.	236	9.3
/shop/tests/__init__.py	Определение URL-маршрутов для административного интерфейса.	0	0.0
/shop/urls.py	Представления для обработки запросов административного интерфейса, включая CRUD-операции для категорий, заказов, продуктов и фильтров.	27	1.32
/shop/utils.py	Конфигурация ASGI для развертывания Django-проекта в асинхронной среде.	17	0.8
/shop/views.py	Основные настройки проекта Django, включая конфигурацию приложений, баз	285	12.02

Модуль	Описание	Количество строк	Размер (Кб)
1	2	3	4
	данных, статических файлов и middleware.		
/shop/__init__.py	Глобальные маршруты проекта, связывающие URL-адреса с представлениями.	0	0.0

2.3.7. Схема тестирования

На Рисунке 16 представлена схема тестирования приложения, на котором графически изображены возможности пользователя в процессе эксплуатации.

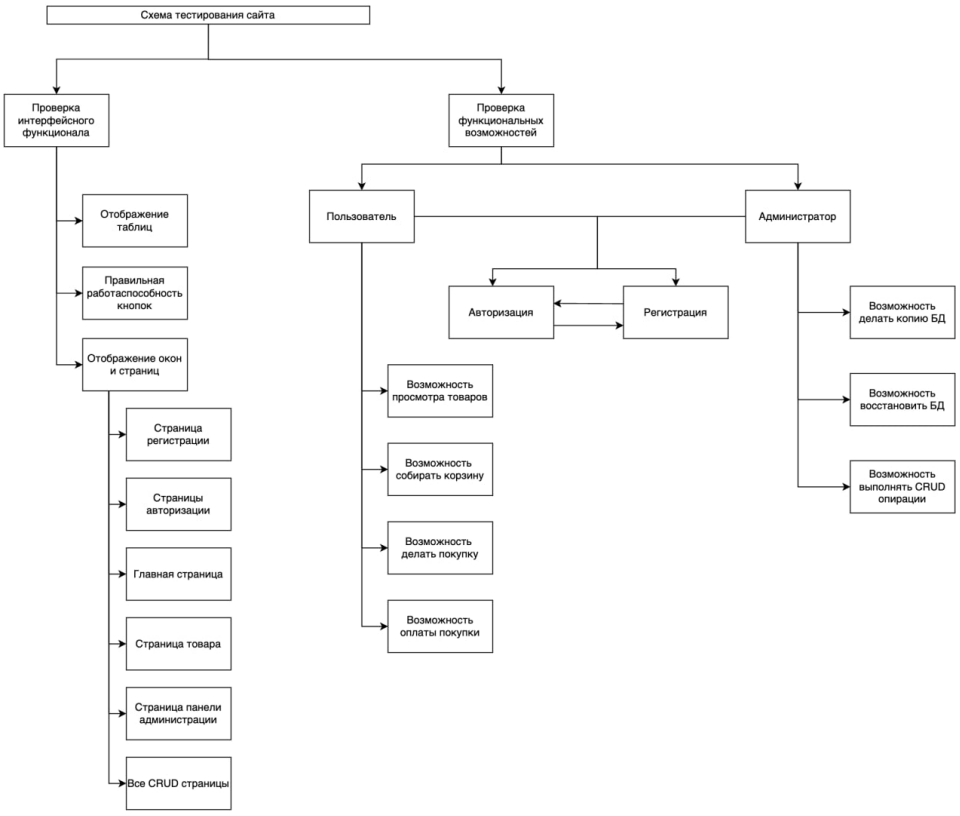


Рисунок 19 – Схема тестирования приложения

2.3.8. Схема пользовательского интерфейса

На рисунке 15 представлена схема пользовательского интерфейса приложения.

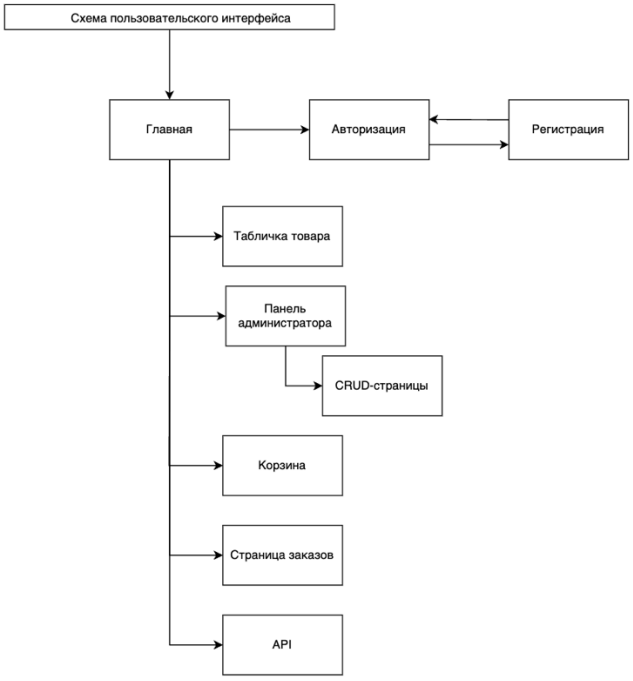


Рисунок 20 – Схема пользовательского интерфейса

2.4. Результат работы программы

В результате поставленной задачи было разработано требуемое программное решение в виде flutter приложения, приложения администратора и сервера Django.

На Рисунках 15-18 представлена некоторая часть работы сайта. Более развёрнуто результат работы программы описан в ПРИЛОЖЕНИИ Г «Руководство пользователя»

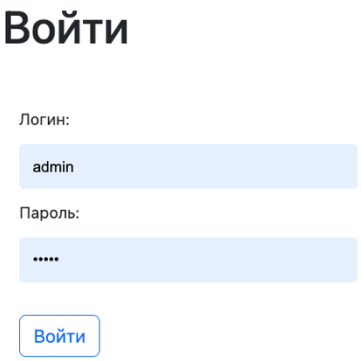


Рисунок 21 – Окно авторизации

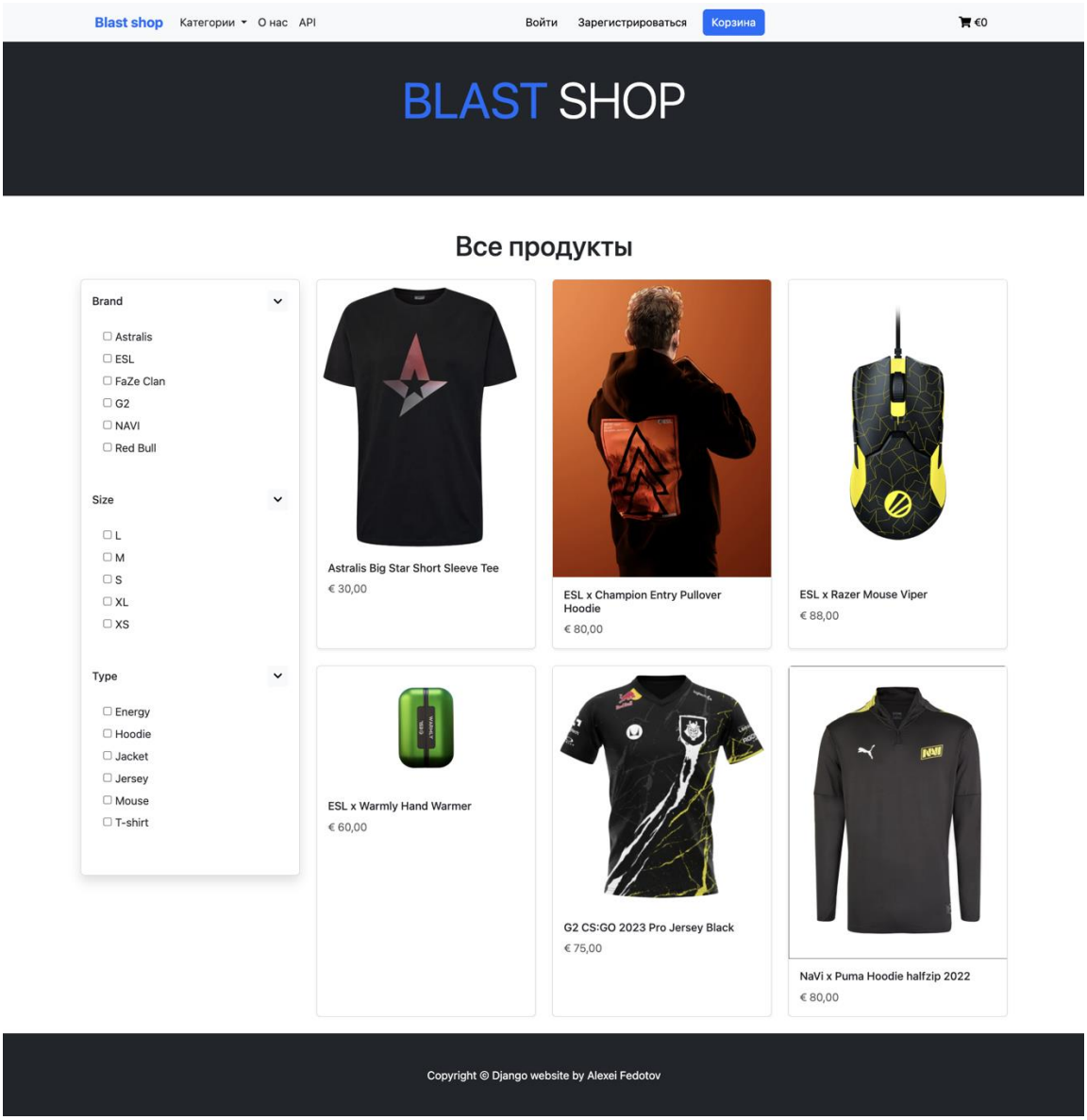


Рисунок 22 – Основная страница

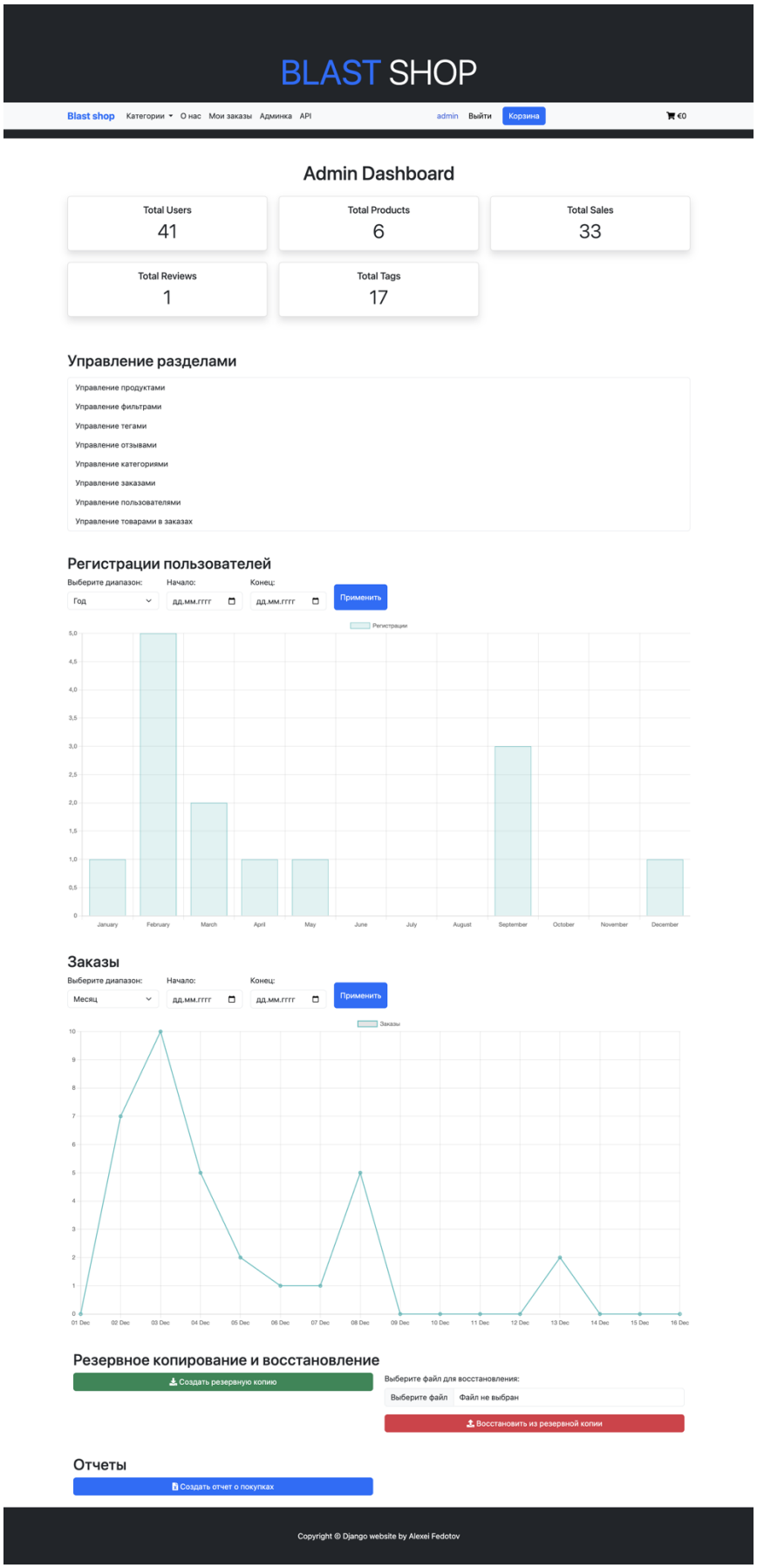


Рисунок 23 – Графики и администрирования

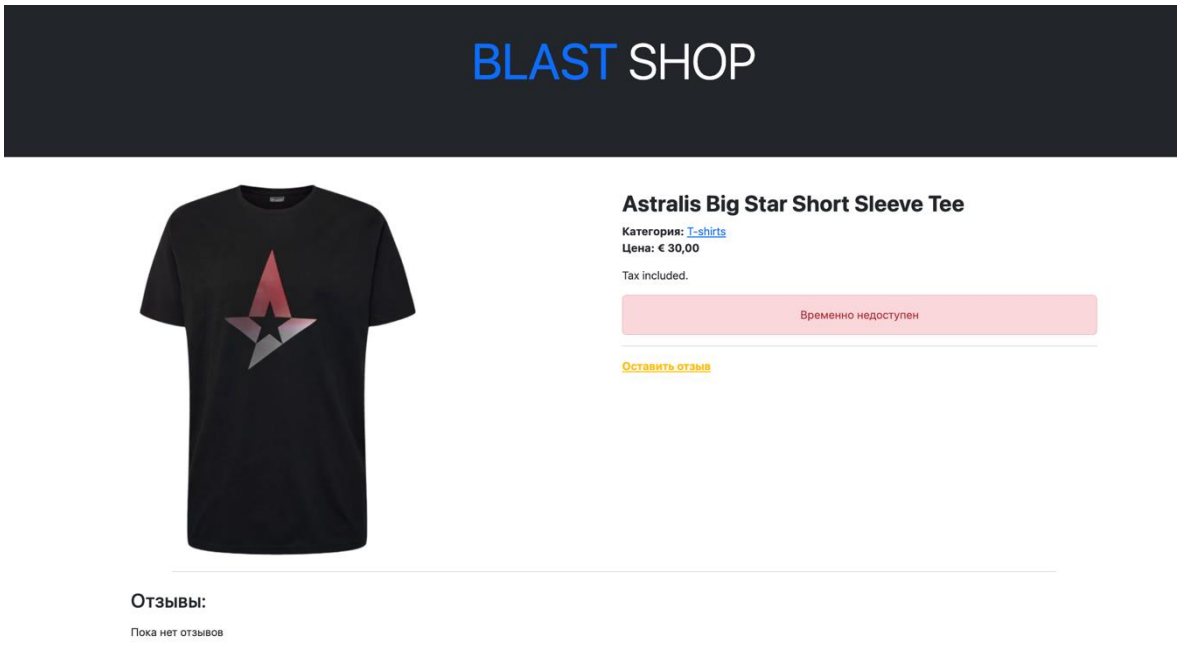


Рисунок 24 – Страница товара

3. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

3.1. Инструментальные средства

В данном курсовом проекте разработан многофункциональный онлайн-магазин киберспортивной атрибутики, включающий в себя серверную часть, удобный пользовательский интерфейс и административный модуль для управления системой. Проект реализован с использованием современных технологий и инструментов, обеспечивающих производительность, безопасность и удобство использования.

1. Используемые IDE и платформы разработки:

Django – мощный фреймворк на Python для разработки серверной части веб-приложения. Django предоставляет встроенные инструменты для создания моделей данных, обработки запросов, а также реализации бизнес-логики.

HTML5, CSS3 и JavaScript – использовались для создания адаптивных и интерактивных веб-страниц.

Bootstrap – фреймворк для быстрой и гибкой разработки адаптивного интерфейса. С его помощью был реализован современный дизайн, поддерживающий отображение на разных устройствах.

JavaScript – применен для динамического взаимодействия с пользователем, включая обновление корзины товаров без перезагрузки страницы и создание интерактивных элементов.

PyCharm – использовался для разработки серверной части на фреймворке Django. Django обеспечивает быструю разработку, безопасность и гибкость в управлении бизнес-логикой проекта. Серверная часть отвечает за обработку HTTP-запросов, управление базой данных и предоставление REST API клиентскому приложению.

Visual Studio Code – применялся для верстки интерфейсов и работы с HTML, CSS и JavaScript, включая кастомные скрипты для

динамического поведения элементов на страницах, таких как автогенерация слагов и управление отображением товаров.

Postman – инструмент для тестирования API запросов, проверки корректности работы эндпоинтов и их соответствия требованиям бизнес-логики.

2. База данных:

Для хранения данных была выбрана реляционная база данных PostgreSQL, что обеспечило высокую производительность и надежность. База данных была спроектирована и реализована с учетом нормализации до 3NF, что позволило минимизировать избыточность и повысить целостность данных. Структура базы данных включает основные таблицы, отвечающие за следующие сущности:

- Пользователи – хранение данных о зарегистрированных пользователях магазина.
- Товары – таблица для хранения информации о киберспортивной атрибутике (название, описание, цена, категория, изображение).
- Категории – категории товаров для удобной навигации и фильтрации.
- Заказы – хранение информации о заказах, включая статус и дату оформления.
- Отзывы – отзывы пользователей о товарах для повышения уровня доверия клиентов.
- Корзина – отдельная таблица для хранения данных о товарах, добавленных пользователями в корзину.

В базе данных реализовано 15 таблиц, которые охватывают все аспекты работы с личными денежными средствами, включая пользователей, транзакции, категории расходов, статистику и отчеты.

Основные особенности работы с базой данных:

Нормализация до 3NF: Структура базы данных спроектирована так, чтобы избежать избыточности и аномалий при обновлении данных, что улучшает поддержку целостности базы данных и упрощает дальнейшее расширение системы.

Хеширование паролей: для повышения безопасности, пароли пользователей хранятся в зашифрованном виде с использованием алгоритма хеширования. Это предотвращает их утечку в случае взлома базы данных.

Логирование действий пользователей: В системе реализована функция логирования, которая фиксирует действия пользователей в системе (например, создание транзакций, изменение данных и т.д.). Это позволяет отслеживать активность пользователей и обеспечивать безопасность системы.

Резервное копирование базы данных: Встроенная функция резервного копирования позволяет регулярно сохранять состояние базы данных, что важно для защиты данных и восстановления работы системы в случае сбоя.

3. Функциональность работы с данными:

Для обеспечения полноты функционала приложения были реализованы следующие операции:

CRUD операции: В системе поддерживаются все основные операции с данными — создание (Create), чтение (Read), обновление (Update) и удаление (Delete). Эти операции реализованы через API, обеспечивающее взаимодействие клиента с сервером.

Поиск, сортировка и фильтрация данных: В системе предусмотрены механизмы поиска, сортировки и фильтрации транзакций и других данных, что помогает пользователям быстро находить нужную информацию. Например, поиск по дате, сумме, категории расходов и т.д.

Регистрация и авторизация пользователей: используется хеширование паролей и дополнительные проверки от SQL-инъекций аутентификация для защиты данных пользователей и безопасного взаимодействия с сервером.

Просмотр каталога товаров: Пользователи могут просматривать полный список киберспортивной атрибутики с возможностью поиска, сортировки и фильтрации по категориям, ценам и дате добавления.

Добавление товаров в корзину: Реализована функциональность управления корзиной, где пользователь может добавлять, изменять количество или удалять товары перед оформлением заказа.

Оформление заказов: Пользователь может оформить заказ, указав контактные данные и адрес доставки. Информация отправляется на сервер, где происходит валидация и сохранение заказа в базе данных.

Административная часть:

CRUD-операции: Администраторы могут управлять товарами, заказами, категориями и пользователями. Реализованы функции создания, чтения, обновления и удаления данных через административную панель.

Управление заказами: Администратор может отслеживать статус заказов, изменять их состояние (например, "в обработке", "отправлено" или "доставлено") и уведомлять пользователей об изменениях.

Статистика и аналитика: Разработаны отчеты по продажам, категориям товаров и активности пользователей. Визуализация данных представлена в виде диаграмм и графиков для упрощенного анализа.

4. Работа с данными и API

В проекте реализованы механизмы для полного управления данными, обеспечивающие как взаимодействие с пользователями, так и внутреннюю обработку информации на сервере.

- CRUD-функции: Все основные операции с данными (создание, чтение, обновление и удаление) были разработаны для моделей товаров, заказов, пользователей и отзывов.
- Товары: Администраторы могут добавлять новые товары, загружать изображения, изменять описание, цену и категорию. Пользователи получают актуальные данные через страницы каталога.
- Заказы: Обработка заказов включает в себя создание записей с данными пользователя и статусом заказа, а также возможность изменения статуса на этапах «в обработке», «отправлено» и «доставлено».
- Отзывы: Реализована функциональность добавления и модерации пользовательских отзывов с использованием форм Django.
- Фильтрация и поиск: для удобства пользователей и администраторов реализованы фильтры и поиск:
- Категории товаров: Фильтрация товаров по категориям и диапазону цен.
- Статусы заказов: Отбор заказов по статусу («обработка», «отправка», «завершён»).
- Поиск по ключевым словам: Встроенная поддержка поиска по названию и описанию товаров, что ускоряет работу пользователей с большим каталогом.
- Экспорт и импорт данных: Данные о заказах и товарах можно выгружать в excel для анализа и отчетности. Это особенно полезно для интеграции с внешними аналитическими системами.

Импорт данных позволяет массово загружать информацию о товарах из подготовленных файлов, что упрощает управление большим ассортиментом.

API для обмена данными: Хотя проект не включает полное REST API, взаимодействие между модулями обеспечено внутренними

вызовами методов и формами Django, что позволяет быстро и гибко обрабатывать запросы.

5. Безопасность системы

Для улучшения пользовательского опыта и удобства управления магазином реализованы статистические отчёты и графические визуализации данных. Это позволяет как администраторам, так и пользователям анализировать ключевые метрики работы магазина.

Отчёты по продажам: ссистема генерирует отчёты о продажах за выбранный период. Например, администратор может получить данные о продажах на уровне товаров, категорий и общего дохода.

Построение графиков и диаграмм: В проекте используются графические диаграммы для наглядного отображения статистики. Например, гистограммы отображают количество заказов по дням, а круговые диаграммы показывают распределение продаж по категориям.

Визуализация выполнена с помощью JavaScript и сторонних библиотек, таких как Chart.js, что позволяет представить данные в удобной и интуитивно понятной форме.

Аналитические панели: В административной части проекта создана панель аналитики, которая позволяет отслеживать текущую ситуацию в магазине. Панель включает ключевые метрики, такие как общее количество заказов, доходы за текущий месяц и активность пользователей.

6. Визуализация данных и аналитика

Безопасность является приоритетной задачей при разработке онлайн-магазина. В проекте реализованы следующие механизмы защиты данных и пользователей:

Хеширование паролей: Все пароли пользователей хранятся в базе данных в зашифрованном виде с использованием встроенного в Django

алгоритма PBKDF2. Это предотвращает компрометацию данных в случае взлома базы.

Валидация форм и данных: Все данные, отправляемые через формы, проходят проверку на стороне сервера для исключения некорректного ввода или возможных атак.

Защита от CSRF (Cross-Site Request Forgery): Реализована защита на уровне форм и запросов, чтобы предотвратить несанкционированные действия от имени пользователя.

Ограничение доступа: Администраторы имеют доступ к закрытой панели управления с использованием встроенной системы аутентификации Django. Пользовательские роли (администратор, пользователь) разграничивают возможности доступа к определённым страницам и действиям.

Логирование: В системе внедрено логирование действий пользователей и ошибок, что позволяет отслеживать подозрительную активность и оперативно реагировать на сбои.

Регулярное резервное копирование: База данных регулярно сохраняется в формате SQL и JSON для восстановления в случае сбоев.

7. Тестирование и отладка проекта

Модульное тестирование: для проверки корректности работы основных компонентов использовались юнит-тесты на базе Django TestCase.

Логирование ошибок: Внедрено логирование действий системы для отслеживания ошибок и анализа сбоев.

Дебаггинг: При разработке активно использовался debug-toolbar и инструменты отладки PyCharm, что позволило выявлять и устранять ошибки на всех этапах разработки.

8. Итоговые характеристики проекта

Тестирование системы проводилось для обеспечения стабильности и надёжности работы всех компонентов проекта. Были реализованы различные виды тестов:

- Модульное тестирование: С помощью встроенного тестового фреймворка Django выполнено тестирование моделей, форм и представлений. Это позволило проверить корректность выполнения всех операций и обработку возможных ошибок.
- Функциональное тестирование: тестировались ключевые функции магазина, включая добавление товаров в корзину, оформление заказов, просмотр каталога и авторизацию пользователей.
- Интеграционное тестирование: Проверено взаимодействие между различными компонентами системы – базой данных, серверной частью и пользовательским интерфейсом.
- Отладка кода: использовались встроенные инструменты отладки (debug-toolbar, print-запросы) и анализ логов для выявления и устранения ошибок. Логирование позволило фиксировать все исключения и помогло оптимизировать работу системы.

Благодаря тщательному тестированию удалось добиться стабильной работы приложения даже при высоких нагрузках и большом количестве данных.

3.2. Характеристики программы

Характеристики Django приложения представлены в приложении Е «Текст программы» в таблице 1-3 «Модули».

ЗАКЛЮЧЕНИЕ

В рамках курсового проекта была разработана система для онлайн-магазина киберспортивной атрибутики, включающая клиентскую часть для пользователей и административный интерфейс для управления магазином. Основная цель разработки заключалась в создании удобного и функционального инструмента для эффективного управления продажами, заказами и каталогом товаров, а также обеспечения простоты взаимодействия пользователей с магазином и повышения их удовлетворенности.

Цель проекта состояла в автоматизации процессов продаж и управления магазином киберспортивной продукции. Для её достижения были поставлены задачи: проектирование архитектуры системы, разработка пользовательского интерфейса, реализация функциональных модулей, тестирование и обеспечение безопасности работы приложения. В качестве технологий были выбраны Django для серверной части и административного интерфейса, а также HTML, CSS и JavaScript для фронтенда, что обеспечило стабильную и производительную работу системы. В качестве базы данных использовался PostgreSQL, что позволило достичь высокой производительности и надёжности хранения данных.

В процессе проектирования были определены входные и выходные данные, разработаны детализированные требования к системе, а также подготовлены функциональные и структурные схемы приложения. Дополнительно была спроектирована логическая модель базы данных, включающая таблицы для товаров, заказов, пользователей, отзывов и категорий. Эти этапы позволили четко структурировать работу и обеспечить планомерное выполнение задач на каждом этапе разработки.

Результатом проекта стал полнофункциональный онлайн-магазин, включающий возможности для добавления, редактирования и удаления товаров, обработки заказов и управления пользователями. Для пользователей реализованы функции регистрации и авторизации, просмотра каталога, добавления товаров в корзину, оформления заказов и оставления отзывов. В административном интерфейсе разработаны механизмы для управления товарами, заказами и пользователями, а также аналитические инструменты для отслеживания продаж и популярности товаров. Система успешно прошла функциональное и интеграционное тестирование, продемонстрировав высокую стабильность и производительность при работе с большими объемами данных.

В процессе разработки использовались современные инструменты: PyCharm для написания серверного кода, PostgreSQL для хранения данных и инструменты отладки Django для выявления ошибок и оптимизации производительности. Благодаря внедрённым механизмам логирования и тестирования, система показывает надёжную работу и устойчивость к сбоям, а оптимизация запросов и кеширование обеспечивают высокую скорость загрузки страниц.

Проект можно считать успешным, так как он достиг поставленных целей и задач. Разработанная система предоставляет удобный и интуитивно понятный интерфейс для пользователей и администратора, а также обладает гибкостью для дальнейшего масштабирования и расширения. Онлайн-магазин способен удовлетворить потребности клиентов в приобретении киберспортивной атрибутики, оптимизировать бизнес-процессы и повысить эффективность управления продажами.

Внедрение данного проекта позволит увеличить доступность и популярность киберспортивных товаров среди целевой аудитории, улучшить клиентский опыт и автоматизировать процессы работы магазина, что подчеркнёт его конкурентоспособность на рынке.

СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ

1. ГОСТ 19404-79 ЕСПД. Пояснительная записка. Переиздание января 2010 г.

2. ГОСТ 7.80-2000 СИБИД. Библиографическая запись. Заголовок. Общие требования и правила составления.

3. ГОСТ Р 7.0.5-2008 Библиографическая ссылка. Общие требования и правила составления.

4. ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов.

5. ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов.

6. ГОСТ 19.105-78 ЕСПД. Общие требования к программным документам.

7. Django Project Documentation [Электронный ресурс]. URL: https://docs.djangoproject.com/ (Дата обращения: 16.12.2024).

8. PostgreSQL Documentation [Электронный ресурс]. URL: https://www.postgresql.org/docs/ (Дата обращения: 16.12.2024).

9. Руководство по Python 3.12 [Электронный ресурс]. URL: https://docs.python.org/3/ (Дата обращения: 16.12.2024).

10. W3Schools HTML и CSS [Электронный ресурс]. URL: https://www.w3schools.com/ (Дата обращения: 16.12.2024).

11. Bootstrap Documentation [Электронный ресурс]. URL: https://getbootstrap.com/ (Дата обращения: 16.12.2024).

12. Metanit Django [Электронный ресурс]. URL: https://metanit.com/python/django/ (Дата обращения: 16.12.2024).

13. SQL Учебник [Электронный ресурс]. URL: https://sql-academy.org/ (Дата обращения: 16.12.2024).

14. Современные методы хеширования данных в Django [Электронный ресурс]. URL: https://djangoproject.com/ (Дата обращения:

16.12.2024).

15. Визуализация данных в веб-приложениях [Электронный ресурс]. URL: https://chartjs.org/ (Дата обращения: 16.12.2024).

16. MDN Web Docs [Электронный ресурс]. URL: https://developer.mozilla.org/ (Дата обращения: 16.12.2024).

17. Интерактивная документация по REST API [Электронный ресурс]. URL: https://swagger.io/ (Дата обращения: 16.12.2024).

18. Гайд по безопасности Django [Электронный ресурс]. URL: https://owasp.org/ (Дата обращения: 16.12.2024).

19. Руководство по PL/pgSQL [Электронный ресурс]. URL: https://www.postgresql.org/docs/current/plpgsql.html (Дата обращения: 16.12.2024).

20. Создание и оптимизация индексов в PostgreSQL [Электронный ресурс]. URL: https://postgresql.org/docs/current/indexes.html (Дата обращения: 16.12.2024).

21. Использование ORM в Django для работы с базами данных [Электронный ресурс]. URL: https://docs.djangoproject.com/en/4.2/topics/db/ (Дата обращения: 16.12.2024).

22. Работа с моделями и миграциями в Django [Электронный ресурс]. URL: https://docs.djangoproject.com/en/4.2/topics/migrations/ (Дата обращения: 16.12.2024).

23. GitHub – Репозиторий с проектами на Django [Электронный ресурс]. URL: https://github.com/ (Дата обращения: 16.12.2024).

24. Интерактивные таблицы с DataTables [Электронный ресурс]. URL: https://datatables.net/ (Дата обращения: 16.12.2024).

25. Система прав доступа в Django Rest Framework [Электронный ресурс]. URL: https://www.django-rest-framework.org/api-

guide/permissions/ (Дата обращения: 16.12.2024).

26. Принципы построения функциональной архитектуры веб-приложений [Электронный ресурс]. URL: https://medium.com/ (Дата обращения: 16.12.2024).

27. Современные методы тестирования Django-проектов [Электронный ресурс]. URL: https://pytest-django.readthedocs.io/ (Дата обращения: 16.12.2024).

28. Основы создания RESTful API в Django [Электронный ресурс]. URL: https://realpython.com/ (Дата обращения: 16.12.2024).

29. Кэширование данных в Django [Электронный ресурс]. URL: https://docs.djangoproject.com/en/4.2/topics/cache/ (Дата обращения: 16.12.2024).

30. Руководство по логированию в Python [Электронный ресурс]. URL: https://docs.python.org/3/howto/logging.html (Дата обращения: 16.12.2024).