**Введение**

В условиях современного рынка многие предприниматели проявляют интерес к сфере флористической торговли. Однако, в связи с активным развитием информационных технологий, данная деятельность сопряжена с рядом сложностей, включая высокую конкуренцию и значительные трудности при создании собственного программного обеспечения с нуля.

Основная цель разработки дипломного проекта заключалась в создании эффективной платформы для начинающих предпринимателей в сфере цветочного бизнеса, а также в предоставлении потребителям возможности приобретать их продукцию. В процессе реализации проекта были использованы следующие технологии:

* **Python 3.13 (Основной ЯП)**
* **Django 5.2.3**
* **PostgreSQL** (основная СУБД)
* **Django ORM** (работа с базой через модели)
* **Django Templates** + Bootstrap‑стилизация (HTML/CSS)
* **JavaScript (ES6+)** с Fetch API и AJAX для динамики (обновление счётчика корзины, управление количеством в корзине и т. д.)
* **django-taggit** (управление тегами товаров)
* **django-imagekit** (обработка картинок: ресайз, webp‑конверсия)
* **HMAC + hashlib** (генерация цифровой подписи заказов)
* **UUIDField** (уникальные идентификаторы транзакций)
* **Session & Cookies** (хранение CSRF‑токена, сессий авторизации)
* **Bootstrap Toasts** (всплывающие уведомления через JavaScript)
* **CSS-анимации и переменные — Custom properties** (плавные hover‑эффекты, анимации удаления)
* **Font-Awesome - иконочный шрифт для красивой иконочной деталищации сайта**

В рамках проекта «Floreal Paris» определены следующие ключевые цели исследования:

1. **Создание монолитной архитектуры клиент-сервер**Мы используем Django как единую платформу для фронтенда и бэкенда. Логика разделена на модули: товары, корзина, заказы, отзывы, чат. Все страницы оформлены единым шаблоном.
2. **Управление товарами через админку и интерфейс**Продавцы могут создавать, редактировать и удалять объявления о цветочных композициях. Возможна загрузка изображений (Django-imagekit) и добавление тегов (django-taggit).
3. **Динамическая корзина с оформлением заказа**Корзина обновляется асинхронно при изменении количества и суммы товаров. Оформление заказа включает генерацию UUID-транзакции и HMAC-подписи.
4. **Система отзывов без дублирования**Пользователи могут оставить только один отзыв на товар. При попытке повторного оставления система ловит IntegrityError и уведомляет об этом через Django-сообщения (Bootstrap Toasts). Администраторы могут удалять некорректные отзывы.

@login\_required

def add\_review(request, product\_id):

product = get\_object\_or\_404(Product, pk=product\_id, is\_active=True)

if request.method == 'POST':

form = ReviewForm(request.POST)

if form.is\_valid():

rev = form.save(commit=False)

rev.product = product

rev.user = request.user

try:

rev.save()

messages.success(request, "Спасибо за ваш отзыв!")

except IntegrityError:

messages.error(request, "Вы уже оставили отзыв на этот товар.")

return redirect('product\_detail', product\_id=product.id)

else:

form = ReviewForm()

return render(request, 'products/reviews/add\_review.html', {

'form': form,

'product': product

})

1. **Полнотекстовый поиск по товарам и пользователям**Поиск осуществляется по товарам (SQL-шаблоны в PostgreSQL) и пользователям (по имени и email). Результаты поиска отображаются в едином шаблоне с ссылками на детализацию.
2. **Личные и публичные профили**На личной странице пользователя видны контактные данные, список его объявлений и история заказов. Публичный профиль продавца позволяет отправлять ему сообщения.
3. **Модуль обмена сообщениями**При первом сообщении создаётся чат. Переписка хранится и выводится в шаблоне. В будущем планируется добавить WebSocket-уведомления или Celery-фоновые задачи.
4. **Ролевая модель и кастомная админ-панель**В модель пользователя добавлено поле role (покупатель/продавец/администратор). Права разграничены на уровне view-функций и шаблонов. Администрирование ведётся через отдельное приложение dashboard. Покупатель - пользователь без товаров; продавец - пользователь с товарами; администратор - назначается ГАдмином (главный админ)
5. **Оптимизация запросов и масштабируемость**Мы используем Django ORM с селектами (select\_related, prefetch\_related), индексами в PostgreSQL и простым кешированием. Организована пагинация и выборка популярных и новых товаров.
6. **Безопасность и сохранность данных**Обеспечена защита от CSRF-атак. Проверены права доступа на уровне views и шаблонов, предоставляемыми Django. Автоматически уведомляются пользователи при удалении их товаров продавцом.

Проект заложил основу для дальнейшего развития и внедрения. Чтобы система стала полностью функциональной и применимой на практике, нужно интегрировать проверенную систему оплаты товаров. Она должна исключать использование симуляционных или тестовых платежных механизмов.

# **Терминология**

| **Термин** | **Определение** |
| --- | --- |
| **User** | Пользователь платформы. Имеет поля username, email, role (buyer/seller/admin), avatar, phone и т. д. |
| **Buyer** | Роль пользователя (role='buyer'): может просматривать товары и оформлять заказы. |
| **Seller** | Роль пользователя (role='seller'): может добавлять, редактировать и удалять свои товары. |
| **Admin** | Роль пользователя (role='admin'): имеет расширенные права модерирования (удаление товаров, отзывов, пользователей). |
| **Superuser** | Главный администратор (is\_superuser=True): всё то же, что и Admin, плюс управление другими админами. |
| **Product** | Модель товара с полями title, description, price, status (in\_stock/on\_order), image, tags, views, created\_at, updated\_at, is\_active. |
| **Cart** | Активная корзина пользователя, хранит CartItem и является основой для оформления заказа. |
| **CartItem** | Элемент корзины: связь cart↔product и поле quantity. |
| **Order** | Оформленный заказ, содержит ссылку на корзину, поле transaction\_id (UUID), total\_amount, status (pending/processing/completed/cancelled) и digital\_signature (HMAC‑SHA256). |
| **Review** | Отзыв пользователя на товар: связь user↔product, поля rating (1–5) и comment. |
| **IntegrityError** | Исключение Django при попытке дублирования отзыва (по уникальному ограничению). |
| **Digital Signature** | Цифровая подпись заказа: HMAC‑SHA256 от строки transaction\_id + total\_amount с секретным ключом. |
| **UUIDField** | Поле для уникальных идентификаторов (транзакций). |
| **Django‑ORM** | Система моделей и запросов Django для работы с базой данных (PostgreSQL). |
| **SQL‑template** | Ручные SQL‑запросы через connection.cursor() и ILIKE для полнотекстового поиска. |
| **PostgreSQL** | СУБД, в которой хранятся все данные (товары, пользователи, заказы и т. д.). |
| **django‑taggit** | Библиотека для управления тегами (tags) у товаров. |
| **django‑imagekit** | Обработка и генерация изображений при загрузке у товаров и аватаров. |
| **Bootstrap Toasts** | UI‑компонент для всплывающих уведомлений (сообщения об успехе/ошибке). |
| **Fetch/AJAX** | Нативный JavaScript fetch() для асинхронного обновления корзины и других действий. |
| **CSRF** | Встроенная защита Django от межсайтовой подделки запросов в формах и AJAX. |
| **Cart‑update logic** | Клиентская логика increment/decrement, полное удаление товара, очистка корзины. |
| **Custom Admin** | Собственное приложение /dashboard/ с моделями пользователей, товаров и отзывов. |
| **UserProfile** | Дополнительная модель профиля (phone\_number, address, favorite\_flowers, birth\_date). |
| **ChatRoom** | Комната чата между покупателем и продавцом по конкретному товару. |
| **Message** | Сообщение в чате с полями sender, content, timestamp, read, attachment. |
| **UUID** | Универсальный уникальный идентификатор, используется для транзакций заказов. |
| **HMAC + hashlib** | Механизм генерации цифровой подписи заказа. |
| **Views Counter** | Счётчик просмотров (views) для сортировки популярных товаров. |
| **SQL ILIKE** | Оператор PostgreSQL для нечувствительного к регистру поиска по title и description. |
| **Polymorphic Roles** | Разграничение доступа в шаблонах и представлениях на основе user.role и is\_superuser. |
| **Messages Framework** | Встроенный Django‑модуль для хранения и отображения временных уведомлений (success/error). |
| **Home Page Widgets** | "Популярные" и "Новые" товары на главной странице, выбираются через ORM и SQL. |
| **Permissions** | Проверка прав в @login\_required, @user\_passes\_test, а также в views и шаблонах. |
| **Profile Pages** | Личный (/profile/) и публичный (/users/<username>/) профили с действиями по пользователю. |
| **Receipt Generation** | Генерация HTML‑чека по URL order/<uuid>/receipt/ |

# Глава 1. Создание веб-сервиса «Floreal Paris»: теория и технологии

1.1. Актуальные тенденции в развитии онлайн‑платформ электронной коммерции  
 В последние годы рынок электронной коммерции демонстрирует стабильно высокий рост, а вместе с ним — и требования к удобству и надёжности онлайн‑сервисов. Для малого и среднего бизнеса, в том числе в сегменте цветочного ритейла, ключевыми тенденциями являются:

* **Мобильная доступность** – адаптивный дизайн под любые устройства, быстрый отклик интерфейса.
* **Интерактивность без перезагрузок** – AJAX‑запросы и динамические обновления (количество товаров в корзине, отзывы, списки) повышают скорость взаимодействия пользователя с платформой.
* **Персонализация предложений** – система тегов, категориями и просмотра истории позволяет предлагать клиенту релевантные товары.
* **Удобные методы оплаты и оформления заказа** – цифровая подпись, интеграция платёжных шлюзов, генерация чеков повышают доверие и лояльность покупателей.
* **Социальное подтверждение качества** – система рейтингов и отзывов, а также уведомления о доступности товара мотивируют к покупке.
* **Безопасность и целостность данных** – использование CSRF‑защиты, HMAC‑подписей для заказов и строгих прав доступа для разных ролей пользователей.

Платформа «Floreal Paris» учитывает эти тенденции, обеспечивая современный UX, гибкую кастомизацию и надёжность.

1.2. Технологический стек проекта

1.2.1. Фронтенд

* **Django Templates** Используются для генерации HTML на сервере, с возможностью вставки динамических данных (товары, корзина, отзывы).
* **Bootstrap 5** Фреймворк для адаптивной и кросс‑браузерной вёрстки, включает утилиты для сеток, кнопок, модальных окон и уведомлений (Toasts).
* **Vanilla JavaScript + Fetch API** Асинхронное добавление/удаление товаров из корзины, изменение количества, отправка форм отзывов и авторизации без перезагрузки страницы.
* **Bootstrap Toasts** Всплывающие уведомления о результате операций (успех/ошибка) для интуитивно понятной обратной связи.
* **Font-Awesome**

Иконочный шрифт, позволяющий задекорировать сайт красивыми минималистическими иконками.

1.2.2. Бэкенд

* **Python 3.13** Современная версия языка, повышающая безопасность и производительность.
* **Django 5.2 Web Framework** MVC‑архитектура, ORM для работы с БД, встроенная система аутентификации и разграничения прав.
* **django‑taggit** Управление тегами у товаров, гибкая фильтрация и поиск по категориям.
* **django‑imagekit** Обработка и ресайзинг загружаемых изображений (товары, аватары) для оптимизации хранения и отображения.
* **PostgreSQL** Реляционная СУБД с поддержкой полнотекстового поиска, транзакций и надёжностью для хранения основной информации.
* **HMAC + hashlib** Генерация цифровой подписи заказов, гарантирующей неизменность данных о сумме и идентификаторе транзакции.
* **UUIDField** Для уникальных идентификаторов заказов (transaction\_id), безопасных встраиваний в URL и чек‑документы.

1.2.3. Инструменты разработки

* **PyCharm** Основная IDE для серверной части (Django), отладки и написания тестов.
* **Web Storm** Для правки фронтенд‑части, JavaScript и стилей.
* **Git & GitHub** Контроль версий.

1.3. Архитектура веб‑сервиса «Floreal Paris»

1.3.1. Паттерн MVC (MVT в Django)

* **Model** – модели User, Product, Cart, CartItem, Order, Review, UserProfile, ChatRoom, Message.
* **View** – функции‑представления на Django (views.py), обрабатывающие HTTP‑запросы, выполняющие бизнес‑логику и возвращающие JSON для AJAX или шаблоны HTML.
* **Template** – HTML‑шаблоны с расширениями {% extends … %} и динамическими вставками {{ … }}, стилизованные через Bootstrap и CSS‑переменные.

1.3.2. Разграничение прав доступа

* **Покупатель (buyer)** – добавляет товары в корзину, оставляет отзывы (по одному на товар), оформляет заказы.
* **Продавец (seller)** – создаёт, редактирует и удаляет собственные товары, просматривает заказы и чаты по ним.
* **Администратор (admin)** – модерирует отзывы и товары чужих пользователей и удаляет учётные записи..
* **Главный администратор (superuser)** – дополнительно управляет ролями администраторов

1.4. Особенности работы с данными

1.4.1. Каталог товаров и фильтрация

* Поиск через ORM (поиск по названию и описанию) и ручные SQL‑шаблоны с ILIKE для нечувствительности к регистру.

Пример кода представления для поиска информации на веб-сервисе:

**def search\_view(request):**

**query = request.GET.get('q', '').strip()**

**product\_results = []**

**user\_results = []**

**if query:**

**# --- Поиск товаров через SQL-шаблон ---**

**sql\_products = """**

**SELECT id, title, description, price, image, views**

**FROM app\_of\_floreal\_paris\_product**

**WHERE title ILIKE %s OR description ILIKE %s**

**ORDER BY views DESC**

**"""**

**pattern = f'%{query}%'**

**with connection.cursor() as cursor:**

**cursor.execute(sql\_products, [pattern, pattern])**

**rows = cursor.fetchall()**

**for id, title, description, price, image, views in rows:**

**product\_results.append({**

**'id': id,**

**'title': title,**

**'description': description,**

**'price': price,**

**'image': image,**

**'views': views,**

**})**

**sql\_users = """**

**SELECT id, username, email, date\_joined**

**FROM app\_of\_floreal\_paris\_user**

**WHERE username ILIKE %s OR email ILIKE %s**

**ORDER BY date\_joined DESC**

**"""**

**with connection.cursor() as cursor:**

**cursor.execute(sql\_users, [pattern, pattern])**

**rows = cursor.fetchall()**

**for id, username, email, date\_joined in rows:**

**user\_results.append({**

**'id': id,**

**'username': username,**

**'email': email,**

**'joined': date\_joined,**

**})**

**return render(request, 'search.html', {**

**'query': query,**

**'product\_results': product\_results,**

**'user\_results': user\_results,**

**})**

* Теги (django‑taggit) позволяют организовать гибкую навигацию по жанрам и стилям цветочных композиций.

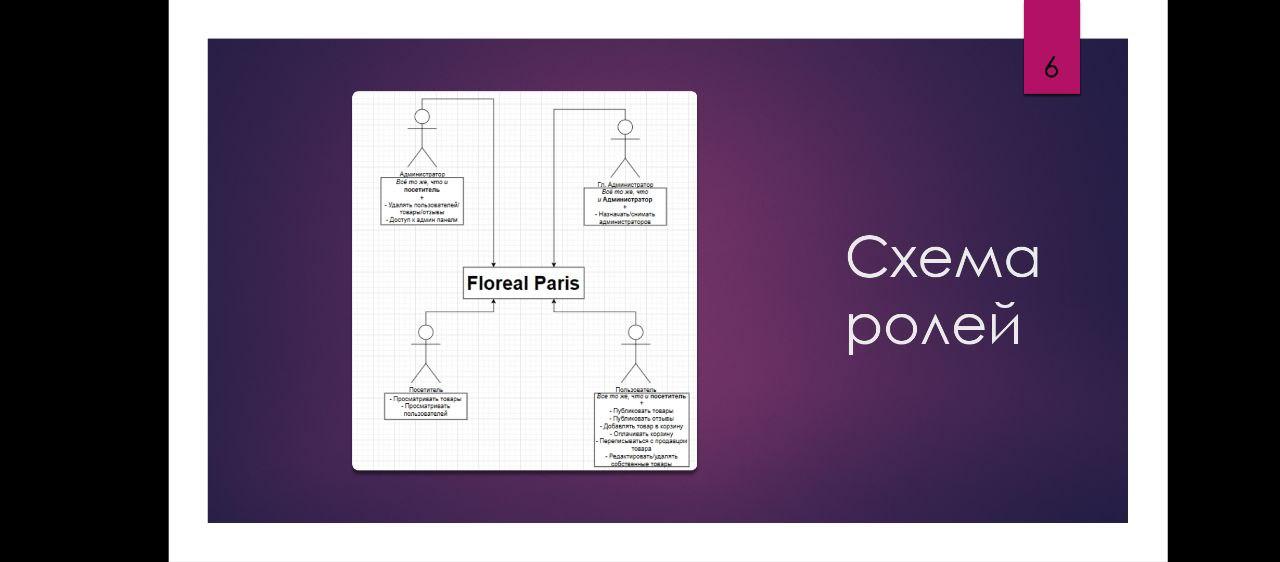
1.4.2. Корзина и оформление заказа

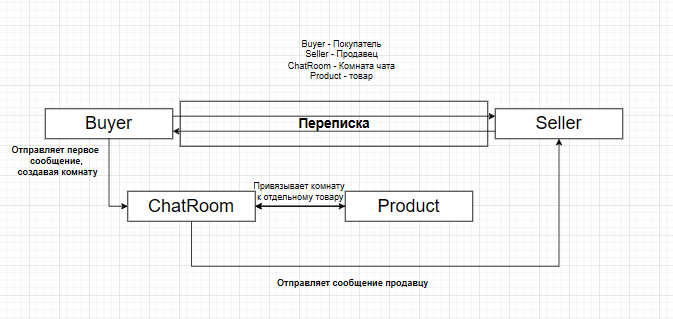
* **Cart** – создаётся автоматически при первом добавлении товара, хранит активные позиции (CartItem).
* **AJAX‑логика** для повышения отзывчивости:  
  + increment/decrement количества без перезагрузки
  + полное удаление конкретного товара
  + очистка всей корзины
* **Order** – после «чекаута» корзина деактивируется, создаётся заказ с цифровой подписью (HMAC), формируется HTML‑чек для скачивания.

1.4.3. Система отзывов и рейтингов

* Модель Review с уникальным ограничением (user, product), предотвращающим дублирование.
* Обработка IntegrityError для пользовательского предупреждения о повторном отзыве.
* Средняя оценка товара выводится на странице и влияет на сортировку «популярных» предложений.

Схема ролей:



Коммуникация (Чат румы по товару)

Как было и как стало диаграмма:  
 Изображение выглядит как текст, снимок экрана, Шрифт, Параллельный

Содержимое, созданное искусственным интеллектом, может быть неверным.

Схема моделей:

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

1.5. Выводы по главе 1

1. Платформа «Floreal Paris» спроектирована с учётом современных требований e‑commerce (Электронная Коммерция (e-commerce) - это любая коммерческая деятельность, осуществляемая через интернет): адаптивности, интерактивности и безопасности.
2. Выбранный стек (Django 5.2 + PostgreSQL + Bootstrap + Vanilla JS) обеспечивает скорость разработки, простоту сопровождения и масштабируемость.
3. Использование AJAX и цифровой подписи заказов повышает доверие пользователей и снижает нагрузку на сервер за счёт отказа от полных перезагрузок страниц.
4. Чёткое разделение ролей (buyer/seller/admin/superuser) и встроенные средства Django позволяют строго контролировать доступ к критичным операциям (удаление товаров, отзывов, пользователей).

# Создание и запуск веб-сервиса «Floreal Paris»

2.1. Общая архитектура системы

2.1.1. Клиент‑серверная модель

* **Клиентская часть**
  + Выполняется в браузере и состоит из HTML‑шаблонов Django, стилизованных с помощью Bootstrap 5 и CSS‑переменных.
  + Взаимодействие с сервером для динамических операций (корзина, отзывы, профиль) происходит через Fetch API и JSON‑ответы.
* **Серверная часть**
  + Django 5.2 — обработка URL, маршрутизация (urls.py), представления (views.py), ORM‑модели (models.py).
  + Контекст‑процессор cart\_summary добавляет в каждый шаблон текущее состояние корзины.
  + Сообщения (django.contrib.messages) используются для всплывающих предупреждений/успехов (Bootstrap Toasts).

2.2. Проектирование базы данных

2.2.1. Сущности и связи

* **User** (наследник AbstractUser)  
   – роли (buyer, seller, admin), проверка прав;
* **Product** – seller → User (FK), поля: title, description, price, status, image, tags, created\_at, updated\_at, is\_active, views;
* **Cart** – user → User (FK), is\_active, created\_at;
* **CartItem** – cart → Cart (FK), product → Product (FK), quantity;
* **Order** – user → User (FK), cart → Cart (FK), transaction\_id (UUID), total\_amount, status, digital\_signature;
* **Review** – user → User (FK), product → Product (FK), rating 1–5, comment, created\_at; уникальный индекс (user, product);
* **UserProfile** – user → User (OneToOne), доп. поля phone\_number, address, favorite\_flowers, birth\_date;
* **ChatRoom** и **Message** – для обмена сообщениями между buyer и seller по конкретному product.

2.2.2. Ключевые ограничения

* При удалении товара (Product.delete()) автоматически удаляются связанные CartItem, Review, ChatRoom и Message (CASCADE DEL).
* Уникальность отзыва по (user, product) защищает от дублей на уровне БД.

2.3. Реализация ключевого функционала

2.3.1. Каталог и карточка товара

* **product\_list** – отображает все is\_active=True товары, сортировка по дате (новинки) или по количеству просмотров (популярные).
* **product\_detail**
  1. Инкремент счётчика views при каждом GET‑запросе.
  2. Вывод подробностей: изображение, описание, статус, цена, дата/время создания и редактирования (с учётом отступа в 3 минуты для флага was\_edited).
  3. Кнопки:  
     + Для автора и администратора – «Редактировать»/«Удалить».
     + Для всех остальных – AJAX‑форма «В корзину».

2.3.2. Корзина и оформление заказа

* **AJAX‑эндпойнты** (add\_to\_cart, update\_cart\_item, remove\_from\_cart, clear\_cart):  
  + Возвращают JSON { success, cart\_count, cart\_total, item\_quantity }.
  + В шаблоне view\_cart.html кнопки «+»/«−» мгновенно изменяют количество, а «тест» полностью удаляет позицию.
* **checkout**
  + При POST – атомарное создание Order, генерация HMAC‑подписи и деактивация корзины.
  + В личном кабинете пользователя (profile/detail.html) выводится таблица my\_orders с кнопкой «Скачать чек» (генерируется view generate\_receipt).

2.3.3. Отзывы и рейтинги

* **add\_review**
  + Форма ReviewForm, проверка уникальности (user, product) через ловлю IntegrityError и сообщение пользователю.
* **Вывод**
  + На странице товара список отзывов с цветовой индикацией звёзд, кнопка «Удалить» для администратора.

2.3.4. Управление профилем

* **profile\_view**
  + Форма ProfileForm для редактирования контактных данных и адреса; блок «Мои объявления» с карточками товаров.
* **public\_profile**
  + Просмотр чужих профилей по URL /users/<username>/, вывод активных объявлений и кнопка «Написать в чат» (если авторизован).

2.3.5. Чат между покупателем и продавцом

* При клике «Написать» создаётся или открывается ChatRoom(product, buyer, seller).
* Шаблон chat\_room.html отображает историю Message и форму отправки нового сообщения через AJAX.

2.4. Интеграции и расширения

2.4.1. Администрирование

* **Кастомная админ‑панель** dashboard/ с собственным оформлением:  
  + Список пользователей (только для GA), переключение ролей, удаление;
  + Список товаров и отзывов с возможностью удаления и модерации;
* **Стандартный Django Admin** оставлен для продвинутых задач.

2.4.2. Безопасность

* **CSRF** – встроенная защита для всех POST и AJAX‑запросов.
* **Проверка прав** в декораторах @login\_required, @user\_passes\_test(is\_admin).
* **HMAC‑подпись** защищает от подделки суммы заказа.

2.5. Тестирование и оптимизация

2.5.1. Юнит‑ и интеграционные тесты

* **Тесты моделей**: создание/удаление товаров, корзины, заказов, отзывов.
* **Тесты представлений**: корректность HTTP‑ответов, JSON для AJAX‑операций, редиректы и запрещённые доступы.

2.5.2. Производительность

* **Пагинация** списка товаров и отзывов при большом объёме записей.
* **Select\_related / prefetch\_related** в представлениях product\_list и view\_cart для снижения числа запросов к базе.

2.6. Выводы по главе 2

1. Спроектированная архитектура чётко разделяет слои данных, бизнес‑логики и представлений, что упрощает сопровождение и расширение.
2. Комбинация Django Templates + AJAX позволяет сохранять быстрый отклик интерфейса без перехода к полноценному SPA.
3. Реализованный функционал корзины, оформления заказа, отзывов и модерации покрывает основные бизнес‑требования цветочного ритейла.
4. Встроенные тесты и оптимизации на уровне ORM обеспечивают стабильность и масштабируемость системы.

Roadmap:

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.