Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ярославский государственный технический университет»

Кафедра «Информационные системы и технологии»

УДК 004.42 ДОПУСКАЕТСЯ К ЗАЩИТЕ

Заведующий кафедрой

канд. техн. наук

\_\_\_\_\_\_\_\_\_\_С.Ю. Бойков

«\_\_»\_\_\_\_\_\_\_\_\_\_\_2025

РАЗРАБОТКА СИСТЕМЫ ПРИЛОЖЕНИЙ ДЛЯ УПРАВЛЕНИЯ ЦВЕТОЧНЫМ МАГАЗИНОМ

Пояснительная записка к выпускной квалификационной работе по направлению подготовки «Информационные системы и технологии»

ЯГТУ 09.03.02 – 016 ВКР

СОГЛАСОВАНО

|  |  |
| --- | --- |
| Руководитель  ассистент кафедры ИСТ  \_\_\_\_\_\_\_\_Е.В. Александрова  «\_\_»\_\_\_\_\_\_\_\_\_\_\_2025 | Нормоконтролер  ассистент кафедры ИСТ  \_\_\_\_\_\_\_\_Е.В. Александрова  «\_\_»\_\_\_\_\_\_\_\_\_\_\_2025 |
| Консультант по экономике и организации производств  канд. полит. наук, доцент  \_\_\_\_\_\_\_\_Е.А. Страдина  «\_\_»\_\_\_\_\_\_\_\_\_\_\_2025 | Проект выполнил  студент группы ЦИС-49  \_\_\_\_\_\_\_\_\_\_\_ А.А. Иванов  «\_\_»\_\_\_\_\_\_\_\_\_\_\_2025 |

РЕФЕРАТ

73 с., 66 рис., 11 табл., 29 источников.

ВЕБ-ПРИЛОЖЕНИЕ, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ЦВЕТОЧНЫЙ МАГАЗИН, АВТОМАТИЗАЦИЯ ТОРГОВЛИ, УЧЕТ ТОВАРОВ, УЧЕТ ПРОДАЖ, ИНФОРМАЦИОННАЯ СИСТЕМА, БАЗА ДАННЫХ.

Объектом исследования являются информационные процессы управления ассортиментом и заказами в розничном цветочном магазине.

Цель работы – разработка веб-приложения и мобильного приложения для автоматизации управления цветочным магазином.

Для достижения цели выполнены анализ требований, проектирование архитектуры, разработка и тестирование системы.

Методы исследования – анализ предметной области и проектирование информационных систем; для реализации использованы современные технологии веб и мобильной разработки, а также СУБД.

В результате разработана информационная система, включающая веб-приложение и мобильное приложение, обеспечивающая автоматизацию основных процессов магазина (учет товаров, продаж и заказов).

Практическая значимость работы заключается в повышении эффективности управления магазином: система снижает трудозатраты на учет и число ошибок, повышает качество обслуживания.

Апробация проведена в условиях цветочного магазина: тестовая эксплуатация системы подтвердила ее работоспособность и эффективность.

Содержание

[Введение 5](#_Toc199749170)

[1 Аналитическая часть 6](#_Toc199749171)

[1.1 Характеристика объекта исследования 6](#_Toc199749172)

[1.1.1 Ассортимент и сезонность 6](#_Toc199749173)

[1.1.2 Организация работы и персонал 6](#_Toc199749174)

[1.1.3 Ключевые бизнес-процессы 6](#_Toc199749175)

[1.1.4 ИТ-инфраструктура и выявленные проблемы 7](#_Toc199749176)

[1.2 Анализ существующих решений 7](#_Toc199749177)

[1.2.2 Критерии сравнения 8](#_Toc199749178)

[1.3 Постановка цели и задачи 15](#_Toc199749179)

[1.4 Вывод по аналитической части 15](#_Toc199749180)

[2. Проектная часть 17](#_Toc199749181)

[2.1 Обоснование проектных решений 17](#_Toc199749182)

[2.2 Архитектура системы 17](#_Toc199749183)

[2.2.1 Внешние пользователи 18](#_Toc199749184)

[2.2.2 Уровень Frontend 18](#_Toc199749185)

[2.2.3 Уровень Backend 18](#_Toc199749186)

[2.2.4 Уровень внешних сервисов 19](#_Toc199749187)

[2.2.5 Потоки взаимодействий 19](#_Toc199749188)

[2.3 Концептуальное проектирование 20](#_Toc199749189)

[2.3.1 Диаграммы вариантов использования 20](#_Toc199749190)

[2.3.2 Диаграммы последовательности 22](#_Toc199749191)

[2.3.3 Диаграмма компонентов 25](#_Toc199749192)

[2.4 Проектирование базы данных 28](#_Toc199749193)

[2.4 Разработка пользовательского интерфейса 29](#_Toc199749194)

[2.5 Вывод по проектной части 30](#_Toc199749195)

[3. Технологическая часть 32](#_Toc199749196)

[3.1 Используемые программные продукты 32](#_Toc199749197)

[3.1.1 Серверная платформа: Java 17 и Spring Boot 3 32](#_Toc199749198)

[3.1.2 СУБД PostgreSQL 16 32](#_Toc199749199)

[3.1.3 Фронтенд: React 18 c Vite 33](#_Toc199749200)

[3.1.4 Мобильная админ-панель: Flutter 3 33](#_Toc199749201)

[3.1.5 Генерация изображений: Kandinsky API 33](#_Toc199749202)

[3.1.6 Инфраструктура и развёртывание: Docker + GitHub Actions 34](#_Toc199749203)

[3.1.7 Инструменты разработки и тестирования 34](#_Toc199749204)

[3.2 Описание работы приложений 34](#_Toc199749205)

[3.2.1 Описание работы клиентского Web-приложения 34](#_Toc199749206)

[Экономическая часть 45](#_Toc199749207)

[4.1 Решаемая проблема и цель проекта 45](#_Toc199749208)

[4.1.1 Проблема 45](#_Toc199749209)

[4.1.2 Цель проекта 45](#_Toc199749210)

[4.1.3 Оценка конкурентоспособности в сравнении с аналогами 46](#_Toc199749211)

[4.2 Заинтересованные стороны проекта 46](#_Toc199749212)

[4.3 Методология управления проектом 51](#_Toc199749213)

[4.3.1 Существующие методологии 51](#_Toc199749214)

[4.3.2 Выбор подходящей методологии для проекта 51](#_Toc199749215)

[4.4 Структурная декомпозиция работ 52](#_Toc199749216)

[4.4.1 Жизненный цикл проекта 52](#_Toc199749217)

[4.4.2 Состав команды с учетом выполнения работ 55](#_Toc199749218)

[4.4.3 Структурная декомпозиция работ 57](#_Toc199749219)

[4.5 Риски проекта 66](#_Toc199749220)

[4.6 Бизнес-модель 70](#_Toc199749221)

[4.7 Стоимость проекта 71](#_Toc199749222)

[4.7.1 Анализ структуры затрат 71](#_Toc199749223)

[Список литературы 72](#_Toc199749224)

# Введение

Развитие электронной коммерции остаётся одним из ключевых трендов мировой и российской розничной торговли. По данным Data Insight, объём рынка электронной коммерции в России вырос с ≈ 235 млрд ₽ в 2011 году до ≈ 1,72 трлн ₽ в 2019-м, при этом среднегодовой темп прироста составлял порядка 28 %.[[1]](Диплом_АА.docx) Рост числа пользователей сети, повышение их цифровой грамотности и распространение мобильного доступа обеспечили дальнейший подъём: в 2024 году онлайн-продажи превысили 3 трлн ₽, а доля электронной коммерции достигла 16.2 % общего оборота непродовольственного ритейла.[2]

Практически у каждого федерального или локального розничного игрока сегодня есть собственное веб-приложение для приёма онлайн-заказов. Такое приложение расширяет географию продаж, повышает скорость обслуживания и лояльность клиентов. Тем не менее цветочный ритейл обладает спецификой: сезонность спроса, быстрый оборот свежей продукции и критичная важность визуальной составляющей букета. Эти факторы диктуют особые требования к информационной системе цветочного магазина.

Научная новизна работы заключается в интеграции двух технологических решений:

ИИ-конструктор букета — модуль, генерирующий фотореалистичный предпросмотр композиции на основе выбранных цветов и аксессуаров;

Гибридный офлайн-режим для персонала, позволяющий сохранять детали активных заказов в локальное хранилище мобильного или настольного клиента, что гарантирует непрерывность доставки даже при отсутствии интернет-связи.

Цель исследования — разработать веб-приложение для управления цветочным магазином с поддержкой упомянутых функций, которое будет одновременно удобным для покупателей и надёжным для сотрудников.

Актуальность, практическая значимость и новизна предложенного подхода подтверждают целесообразность выполнения данной выпускной квалификационной работы.

# 1 Аналитическая часть

## 1.1 Характеристика объекта исследования

Объектом исследования является розничный цветочный магазин «Веточка», расположенный в центральной части г. Ярославля (население ≈ 600 тыс. человек). Магазин работает с 2021 года и позиционируется как «семейная мастерская флористики», ориентированная на индивидуальные букеты и быструю доставку внутри города.

### 1.1.1 Ассортимент и сезонность

Ассортимент магазина насчитывает примерно двести–двести двадцать товарных позиций, сгруппированных в четыре основные категории. Примерно половину оборота (около 50 %) дают срезанные цветы: их завозят дважды в неделю, а срок реализации составляет от пяти до десяти дней. Около 35 % продаж обеспечивают готовые букеты и композиции; в ассортименте порядка двадцати пяти стандартных схем плюс возможность собрать «индивидуальный букет» под запрос клиента. Горшечные растения формируют примерно десятую часть выручки (10 %); здесь акцент сделан на неприхотливых видах, подходящих для квартир. Оставшиеся пять процентов приходится на сопутствующие товары — упаковку, открытки и небольшие сувениры, которые продаются как дополнение к основным покупкам.

Спрос выраженно сезонный: пики 14 февраля, 8 марта, выпускные (конец мая) и свадебный период (июнь-сентябрь). В «пиковые» даты дневной оборот превышает среднегодовой в 4-5 раз.

### 1.1.2 Организация работы и персонал

Штат небольшой — 3 человека:

* владелец-управляющий (закупки, партнёрские программы);
* 1 флорист-универсал (оформление витрины, сбор букетов);
* 1 курьер на частичной занятости (доставка по городу).

Учёт рабочей смены ведётся вручную, на бумаге; касса подключена к ОФД, но аналитические отчёты формируются вручную.

### 1.1.3 Ключевые бизнес-процессы

Цветы закупаются на оптовой базе Московского региона, логистика — рефрижератором раз в 3-4 дня. Поступление фиксируется вручную: дата, поставщик, артикул, количество, закупочная цена. На основании остатков флористы формируют 8-10 актуальных «витринных» букетов; фото выкладываются во «ВКонтакте», сообщество @ve.tochka.flowers. Каналы для приёма заказов: сообщения во «ВКонтакте» (≈ 30 %), телефон (≈ 25 %), офлайн-зал (≈ 45 %). Администратор заносит данные в общую таблицу: имя, состав, цена, адрес, время доставки. Онлайн-эквайринг отсутствует; клиенты переводят на СБП или оплачивают курьеру по QR-коду. Курьер получает заказы через общий чат. При нестабильном мобильном интернете (спальные районы, пригород) статусы «В пути/Доставлен» могут запаздывать до конца смены. В конце дня администратор формирует сводную таблицу продаж и остатков. Инвентаризация проводится раз в неделю вручную.

### 1.1.4 ИТ-инфраструктура и выявленные проблемы

В ходе описания характеристики объекта было выявлено, что магазин «Веточка» сталкивается с теми же трудностями, что и большинство небольших цветочных точек: ИТ-инфраструктура разрознена, учёт ассортимента и заказов ведётся вручную, а полноценного онлайн-канала продаж вовсе нет. Отсюда проистекают основные слабые места. Во-первых, ручной ввод данных в таблицы неминуемо порождает ошибки и отнимает много времени у сотрудников. Во-вторых, отсутствие собственного сайта с корзиной приводит к тому, что часть потенциальных покупателей, привыкших оформлять покупки целиком в интернете, уходит к конкурентам. В-третьих, курьеры зависят от нестабильного мобильного интернета: если связь обрывается, заказ может оказаться просроченным, потому что статус доставки не обновился вовремя. Наконец, при создании клиентом собственной композиции онлайн, он может увидеть конечный результат непосредственно после оплаты заказа, это создает проблему, поскольку если ему не понравится внешний вид, он получит негативный опыт от покупки.

Эти проблемы убедительно показывают, что магазину необходимо единое веб-приложение, которое будет автоматизировать учёт остатков и заказов, поддерживать онлайн-оплату, хранить активные доставки в офлайн-кэше и генерировать предварительные изображения настраиваемых букетов автоматически. Такая система не только повысит лояльность клиентов и снизит операционные риски, но и задаст чёткие требования к будущей реализации, которые далее предстоит детально сформулировать.

## 1.2 Анализ существующих решений

Прежде чем приступать к проектированию собственной системы электронной коммерции для продажи цветов, необходимо провести тщательный анализ уже существующих решений. Такой анализ позволяет выявить лучшие практики, понять ожидания пользователей и избежать повторения ошибок конкурентов. Изучение чужого опыта помогает определить, какие функции уже реализованы на рынке, а каких не хватает, чтобы новая платформа выгодно отличалась. Кроме того, анализ конкурентов дает представление о технических подходах и архитектурных решениях, используемых в отрасли, что способствует более обоснованному выбору технологий и инструментов при разработке собственной системы.

### 1.2.2 Критерии сравнения

В рамках технического анализа были выделены четыре ключевых критерия, по которым сравниваются существующие сервисы продажи цветов онлайн. Эти критерии отражают функциональные возможности и удобство использования систем как для клиентов, так и для администраторов магазина:

а) Наличие и удобство веб-сайта для оформления заказа: оценивается, есть ли у сервиса полноценный сайт для онлайн-заказа цветов и насколько удобен интерфейс для покупателей (навигация, фильтры, мобильная адаптация и пр.).

б) Наличие панели администратора для управления магазином: проверяется, предоставляет ли система инструментарий для администрирования — например, личный кабинет продавца или администратора, где можно управлять каталогом товаров, отслеживать поступившие заказы, обновлять цены и т.д.

в) Возможность офлайн-доступа к заказам: анализируется, может ли магазин работать с данными заказов без постоянного подключения к интернету. Иными словами, есть ли режим офлайн-работы (например, мобильное приложение с локальным хранением данных), позволяющий просматривать или обрабатывать заказы при временном отсутствии сети.

г) Наличие конструктора букета или генерации изображений: определяется, реализован ли на платформе интерактивный конструктор букета, позволяющий пользователю самостоятельно сформировать композицию (выбрать виды цветов, упаковку и сразу увидеть итоговый вариант), либо есть ли функция автоматической генерации изображений букетов (например, с помощью ИИ) для визуализации настраиваемых букетов. Такая функциональность демонстрирует степень инновационности сервиса в плане персонализации продукта.

Далее рассматривается, как указанные критерии реализованы у четырех ведущих российских игроков рынка онлайн-торговли цветами: Flowwow, Flor2U, FloraExpress и СоюзЦветТорг. Анализ сосредоточен на технической составляющей – то есть, как сделаны те или иные функции и какие имеются плюсы и минусы с точки зрения технологии и удобства.

1.2.3 Обзор основных конкурентов

Flowwow – это международная онлайн платформа, объединяющая тысячи цветочных магазинов и покупателей на одном ресурсе [3]. С технической точки зрения Flowwow реализован как экосистема, включающая веб-сайт и мобильные приложения.

Веб-сайт и приложения: Платформа предоставляет удобный веб-интерфейс для оформления заказов: пользователи могут легко найти букет по различным фильтрам (по случаю, типу цветов, городу доставки и т.д.), добавить его в корзину и оформить доставку. Интерфейс зарекомендовал себя как интуитивно понятный и функциональный [3]. Помимо сайта, Flowwow предлагает мобильное приложение для покупателей на iOS и Android, что повышает доступность сервиса. Приложение позволяет получать пуш-уведомления о статусе заказа и упрощает процесс повторного заказа. Особенностью Flowwow является функция прикрепления фотографии готового букета перед отправкой: продавец загружает фото собранного букета, и клиент может его подтвердить перед доставкой. Эта техническая реализация повышает доверие покупателей и качество обслуживания.

Панель администратора: Так как Flowwow – маркетплейс, он обеспечивает полноценную панель для сторонних продавцов. Реализация сделана в виде отдельного приложения Flowwow Seller для флористических магазинов-партнеров [4]. Через него продавцы регистрируют свой магазин, управляют товарами (букетами), ценами, временем работы и обрабатывают поступающие заказы. Фактически, приложение продавца выступает в роли административной панели: оно предоставляет аналитику по продажам, инструменты продвижения, прием и подтверждение новых заказов и пр. Такой подход с упором на мобильное приложение упрощает работу магазина — владельцы могут оперативно управлять своими онлайн-заказами прямо со смартфона. С точки зрения архитектуры, система построена таким образом, что все операции (добавление товара, обновление статуса заказа) происходят через API маркетплейса, а продавец взаимодействует через удобный интерфейс приложения. Преимущество такого решения — высокая мобильность и возможность мгновенно реагировать на новые заявки; недостаток — зависимость от стабильности мобильного приложения и необходимости постоянного интернет-подключения.

Офлайн-доступ к заказам: Flowwow не предусматривает автономный офлайн-режим для работы с заказами. Поскольку все заказы хранятся на сервере и оперативно обновляются, продавцу или курьеру требуется подключение к интернету, чтобы получить актуальные данные. Мобильное приложение может кешировать некоторую информацию (например, список текущих заказов, если он был загружен ранее), однако полноценно обрабатывать новые заказы или изменять статус доставки без сети невозможно. Таким образом, в критерии офлайн-доступа Flowwow имеет ограничение — система рассчитана на постоянное онлайн-взаимодействие.

Конструктор букета / генерация изображений: На Flowwow отсутствует онлайн-конструктор букетов. Модель маркетплейса предполагает, что пользователи выбирают из готовых позиций, загруженных разными магазинами (на выбор представлено множество букетов, каждый с фотографией и описанием). Пользователь не может интерактивно изменить состав букета на сайте – вместо этого он выбирает из каталога уже сформированный букет. Также платформа не использует генеративные технологии для создания изображений букетов: все фото загружаются флористами и представляют реальные товары. Из плюсов такого подхода – клиент видит именно тот букет, который продается, с реальными фотографиями; из минусов – отсутствует возможность индивидуальной конфигурации композиции онлайн. В случае, если покупателю нужен особый букет, ему приходится связываться с поддержкой или выбирать схожий вариант из готовых.

Flor2U – это крупный российский интернет-магазин доставки цветов, работающий с 2020 года и ставший одним из лидеров в отрасли. В отличие от Flowwow, Flor2U не является маркетплейсом, а выступает как единый магазин/служба, то есть все заказы выполняются самой компанией (через собственные точки или партнеров, но под единым брендом).

Веб-сайт: Официальный сайт Flor2U.ru предоставляет современный и удобный интерфейс для выбора букета и оформления заказа [5]. Каталог разбит по категориям (повод, вид цветов, цена), есть функция поиска, фильтрации по цене и др. Особое внимание уделено визуальной части: каждое предложение сопровождается качественным фото. Пользователь может просмотреть детали букета, выбрать размер вариации (например, стандарт / премиум), добавить сопутствующие товары (открытки, подарки) и указать адрес/время доставки. Сайт оптимизирован под мобильные устройства, обеспечивая комфортный заказ со смартфона без отдельного приложения. Технически можно предположить, что сайт построен на базе типового решения для e-commerce (в России популярны платформы вроде 1С-Битрикс, Insales и т.п.), что дает готовый набор функций: личный кабинет пользователя, онлайн-оплата, отслеживание статуса заказа и др. Интерфейс Flor2U получил высокие оценки за понятность: пользователи отмечают, что оформить заказ легко и быстро [3], а информация о товарах и условиях доставки предоставлена исчерпывающе.

Панель администратора: Как и любой интернет-магазин, Flor2U имеет административную систему для управления контентом и заказами. Эта панель недоступна для сторонних продавцов (поскольку магазин единый), она используется внутренними менеджерами компании. Техническая реализация панели, вероятно, интегрирована в платформу сайта: администраторы могут добавлять новые букеты, обновлять цены, видеть входящие заказы, менять их статус (например, «собирается», «отправлен курьером», «доставлен»). Также админ-панель обрабатывает поступление оплаты, генерацию накладных и прочие бек-офисные функции. С точки зрения внешних пользователей, отдельного кабинета продавца нет (в отличие от Flowwow), но для внутренних целей все необходимые инструменты управления присутствуют. Достоинство такого закрытого решения – полный контроль над системой и возможность тонкой настройки под свои бизнес-процессы; недостаток – меньшая гибкость к масштабированию (невозможно просто подключить новый независимый магазин без значительной доработки).

Офлайн-доступ к заказам: Flor2U не предлагает режима офлайн-работы. Размещение заказа и его обработка требуют соединения с интернетом, так как все данные хранятся на сервере. В случае, например, временного сбоя в сети, курьер или менеджер Flor2U не смогут обновить информацию о заказе или получить новые заказы до восстановления соединения. Тем не менее, компания частично компенсирует это круглосуточной поддержкой и мультиканальными сервисами: принять заказ можно не только через сайт, но и по телефону. Фактически, альтернативный канал – звонок оператору – играет роль резервного способа оформления заказа, что было типично до повсеместного распространения мобильного интернета. Но самостоятельного приложения с офлайн-хранилищем данных у Flor2U нет, и основной упор делается на онлайн-сервис.

Конструктор букета / генерация изображений: Онлайн-конструктора на сайте Flor2U нет – клиентам предлагается выбирать из более 3000 готовых букетов и композиций, представленных в каталоге. Однако сервис позиционирует себя как гибкий в удовлетворении запросов: есть возможность составить индивидуальный букет с помощью консультанта. Это означает, что покупатель может связаться с флористом (по телефону или через чат) и обсудить создание букета по своим предпочтениям. Флорист вручную подберет цветы и составит уникальную композицию, а затем предоставит клиенту фото готового результата (по запросу, перед доставкой). С технической точки зрения, такая индивидуальная работа не автоматизирована на фронтенде – она ложится на процессинг через CRM/чат с оператором. Это полуавтоматическое решение: плюс в том, что практически любое пожелание клиента может быть выполнено человеком, минус – отсутствие интерактивности и мгновенной визуализации на самом сайте. Что касается генерации изображений с помощью ИИ, у Flor2U такого функционала не отмечается – все фотографии букетов реальные, снятые заранее. В перспективе это оставляет простор для улучшений: например, можно внедрить модуль, позволяющий клиенту собрать букет онлайн и сразу увидеть сгенерированное изображение, чего у текущих конкурентов нет.

FloraExpress – одна из старейших служб экспресс-доставки цветов, работающая не только по Москве, но и по другим городам России и даже за рубежом. Этот сервис делает упор на срочность и широту географического охвата: позиционируется как международная служба, готовая доставить букет даже в удаленный город, через локальных партнеров.

Веб-сайт: Официальный сайт FloraExpress представляет собой типичный онлайн-витринный каталог, где пользователь может выбрать город доставки, просмотреть доступные букеты и подарки, и оформить заказ с указанием адреса и времени. Интерфейс достаточно удобен и понятен, хотя визуально может уступать более современным конкурентам. Тем не менее, сайт поддерживает все базовые функции: категории букетов по случаям (свадьба, день рождения, соболезнования и т.п.), фильтрацию по типу цветов (розы, лилии, тюльпаны и т.д.), выбор дополнений (конфеты, игрушки) и разные варианты оплаты. Особенностью FloraExpress является опция международной доставки – на сайте можно выбрать страну назначения, и сервис обеспечит вручение цветов через партнера в той стране. Технически это означает интеграцию с внешними флористическими службами: заказ, сделанный на FloraExpress, передается в локальную службу в нужном городе. Такая схема требует надежного обмена данными (возможно, через API или даже по электронной почте/CRM) и контроля качества исполнения на удаленной стороне. С точки зрения покупателя, интерфейс старается скрыть сложность – клиент просто выбирает город, а дальше каталог адаптируется под местность (валюта, ассортимент). Весь процесс заказа происходит онлайн; сайт работает 24/7, принимая заявки в любое время суток [6].

Панель администратора: Внутренне FloraExpress должна обладать системой управления заказами, особенно учитывая распределенную модель работы. Можно предположить наличие административной панели, где операторы видят все заказы, перенаправляют их соответствующим локальным исполнителям, отслеживают подтверждение доставки. Вероятно, эта система включает модули для управления каталогом (разные каталоги по странам/городам), ценообразования (в разных регионах цены отличаются), а также для мониторинга логистики. Для внешних флористов-партнеров может существовать партнерский интерфейс: например, локальные магазины получают заказы через специальное приложение или веб-доступ к системе FloraExpress. Прямых сведений об этом нет в открытых источниках, но иначе было бы трудно координировать международные доставки. Скорее всего, партнеры получают уведомления о новом заказе и должны подтвердить и выполнить его, после чего отметить в системе факт доставки. Таким образом, административная часть FloraExpress более сложная, чем у обычного магазина: она выполняет функции распределенного оркестратора заказов. С точки зрения технологии, это сложность, но и ключевое преимущество сервиса.

Офлайн-доступ к заказам: Как и конкуренты, FloraExpress не имеет явно выраженного офлайн-режима работы. Для принятия и координации заказов требуется связь с сервером. Операторы кол-центра работают круглосуточно, поэтому в случае проблем с интернетом клиенты могут позвонить – однако сами сотрудники FloraExpress все равно нуждаются в подключении к системе для размещения заказа. Нет данных о том, что FloraExpress предлагает какое-либо автономное приложение, которое бы позволяло просматривать заказы офлайн. Вероятно, критические подразделения (например, курьеры) работают через мобильный интернет или получают задания через СМС/звонки, когда это необходимо, что является ручным резервным решением. В целом же, система полагается на онлайн-доступ, а офлайн-функциональность не заявлена как часть сервиса.

Конструктор букета / генерация изображений: На платформе FloraExpress нет интерактивного конструктора букетов. Концепция сервиса – максимально быстрая доставка готовых букетов, поэтому упор делается на заранее подготовленные композиции. Клиент выбирает из предложенного списка стандартных букетов (разработанных профессиональными флористами). Если нужно что-то особенное, в FloraExpress, как и в Flor2U, можно связаться с операторами для индивидуального заказа, но сделать это через интерфейс сайта нельзя – потребуется общение через телефон или почту. Автоматической генерации изображений также не используется: все товары имеют реальные фотографии. В контексте новых трендов это означает, что FloraExpress пока не внедряла AR/VR или AI-инструменты для персонализации. Технический упор сделан на надежность доставки и широту сети, а не на пользовательские эксперименты в интерфейсе.

СоюзЦветТорг – российская компания, представляющая собой одну из крупнейших сетей цветочных магазинов (более 60 торговых точек в Москве и области) [3]. Фактически, это офлайн-ритейлер (цветочные гипермаркеты), который активно развивает и онлайн-продажи через собственный сайт и приложения. СоюзЦветТорг сочетает преимущества большой физической сети (собственная инфраструктура, склады, курьеры) с современными e-commerce технологиями.

Веб-сайт: На официальном сайте представлен полный каталог продукции, схожий с классическими интернет-магазинами цветов. Интерфейс предоставляет разнообразные категории товаров (монобукеты, авторские композиции, цветы по типу, по цене, акции и т.д.), что упрощает навигацию для клиента. Сайт оснащен удобной корзиной, возможностью указать пожелания к заказу, выбрать дату/время доставки и оплатить онлайн. Интересной технической особенностью является наличие на сайте раздела «Собери свой букет» – это реализация конструктора букета [7]. Пользователь может самостоятельно выбрать разные виды цветов (розы, хризантемы, тюльпаны и проч.), указать желаемое количество, добавить элементы декора и упаковки, фактически сформировав уникальный букет под свой вкус. Сайт в интерактивном режиме подсчитывает стоимость и позволяет добавить такой кастомный букет в корзину. Однако стоит отметить, что визуализация при этом остаётся ограниченной: система, по сути, отображает список выбранных компонентов, а не генерирует фотографию собранного букета. Тем не менее, это шаг вперед по сравнению с другими конкурентами, у которых нет даже такой функции. Помимо веб-сайта, СоюзЦветТорг интегрирован с крупными агрегаторами доставки (например, их услуги представлены в приложениях «Яндекс Еда» и «Delivery Club» для экспресс-доставки цветов из ближайшего магазина. Также у компании есть собственное мобильное приложение для заказов, чтобы удерживать постоянных клиентов в своей экосистеме (приложение доступно для iOS/Android, упоминается на сайте и в маркетах приложений).

Панель администратора: Поскольку СоюзЦветТорг управляет десятками точек и онлайн-каналом, у них должна быть мощная система управленения. Вероятно, используется корпоративная платформа, позволяющая централлизованно обновлять каталог на сайте (и в приложении), следить за остатками товара в разных магазинах, получать онлайн-заказы и перенаправлять их на исполнение в ту торговую точку, которая ближе всего к адресу доставки клиента. Административная панель, скорее всего, разграничена по ролям: центральный офис управляет глобальными настройками (ассортимент, ценообразование, маркетинговые акции), а менеджеры отдельных магазинов имеют доступ к своим заказам (например, видят заказы, адресованные к выдаче из их филиала, и отмечают статус выполнения). Такая распределенная система администрирования требует хорошей синхронизации данных и, возможно, частично работает офлайн внутри магазинов (например, через локальные POS-системы). Однако онлайн-заказы все равно приходят через интернет, и общая база данных находится в облаке или на сервере компании. В целом, наличие панели администратора очевидно – без ИТ-системы управлять столькими точками и заказами невозможно – и можно заключить, что она есть как на уровне всего магазина (CMS для сайта), так и на уровне отдельных филиалов (система приема и выполнения заказов).

Офлайн-доступ к заказам: У СоюзЦветТорг, несмотря на развитость инфраструктуры, нет полной офлайн-работы с онлайн-заказами. Магазины, конечно, могут продолжать работу офлайн, продавая цветы напрямую покупателям в зале, но если говорить про заказы, поступившие через сайт или приложение, для их обработки нужно подключение к центральной системе. В случае временного отключения интернета отдельный магазин не сможет получить новые онлайн-заказы или обновить статусы существующих. Таким образом, как и остальные конкуренты, СоюзЦветТорг полагается на онлайн-связь. Но его сильная сторона – наличие физических магазинов – позволяет, например, выполнять заказы даже при сбоях: курьер может быть направлен с ближайшей точки, если другая на связи недоступна. Это скорее организационная устойчивость, чем техническая.

Конструктор букета / генерация изображений: Как отмечалось, СоюзЦветТорг – единственный из рассматриваемых сервисов, кто имеет встроенный онлайн-конструктор букета на своем сайте [7]. Эта функция дает пользователям больше свободы: можно собрать букет по своему желанию. Реализация технически представляет собой каталог компонентов (цветы поштучно, зелень, упаковка) с возможностью выбора количества каждого. Система в реальном времени суммирует цену и показывает, доступен ли компонент. Однако, отсутствует автоматическая генерация изображения итогового букета – пользователь видит позиции списка, но не их визуальное сочетание. Для опытного пользователя этого может быть достаточно, но новичку сложно представить результат. Здесь имеется потенциал улучшения: интеграция генеративной нейросети. Сейчас же СоюзЦветТорг компенсирует это за счет консультаций: при необходимости клиент может уточнить детали с флористом, а при сборке букета магазин, вероятно, высылает фото готовой работы (по запросу, через мессенджеры).

Таблица №1 - Сравнение альтернативных решений

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Критерий** | **Flowwow** | **Flor2U** | **FloraExpress** | **СоюзЦветТорг** | **Vetochka** |
| Веб-сайт для заказа | Да | Да | Да | Да | Да |
| Админ-панель | Да | Да | Да | Да | Да |
| Офлайн-доступ | Нет | Нет | Нет | Нет | **Да** |
| Конструктор AI-генерация | Нет | Нет | Нет | Частично | **Да** |

Таким образом у всех перечисленных решений есть свои сильные стороны (широкий охват и узнаваемость у Flowwow, круглосуточный сервис у Flor2U и т.д.), но заметны и технические ограничения. Ни один из конкурентов не сочетает сразу маркетплейс и офлайн‑режим, а также не реализует генерацию изображений букетов (Таблица 1). Новая система Vetochka, напротив, позиционируется как универсальная платформа: она предоставляет удобный веб‑интерфейс и мобильные приложения с офлайн‑доступом к заказам (что позволяет принимать заказы даже при отсутствии сети) и включает современный конструктор букетов с визуализацией (в том числе на базе генеративных моделей), чего нет ни у Flowwow, ни у других перечисленных сервисов. Благодаря этим технологиям Vetochka устраняет ключевые «узкие места» текущих решений и предлагает более гибкую модель работы для флористов и курьеров.

## 1.3 Постановка цели и задачи

Разработать и внедрить для цветочного магазина «Веточка» комплексную систему приложений, которая:

* предоставляет покупателю интерактивный конструктор букетов c мгновенным ИИ-предпросмотром композиции;
* обеспечивает сотрудникам магазина надёжную работу с заказами в гибридном (онлайн / офлайн) режиме;
* автоматизирует учёт ассортимента, оплату и доставку без привлечения сторонних маркетплейсов и комиссий.

## 1.4 Вывод по аналитической части

Проведённое исследование объекта («Веточка») и обзор существующих решений подтвердили, что у малого цветочного ритейла остаются незакрытыми два критически важных сценария:

Индивидуальная визуализация букета в режиме реального времени. Ни одно из проанализированных решений не предлагает генеративного конструктора, способного показывать покупателю фотореалистичную картинку композиции.

Надёжная работа с заказами при нестабильном интернете. Готовые продукты хранят данные только в облаке; при обрыве связи курьер или администратор теряет доступ к актуальной информации, что повышает риск срыва доставки.

Таким образом, аналитическая часть сформировала чёткое обоснование необходимости создания специализированного веб-приложения, которое:

* сохраняет контроль бренда и снижает издержки;
* внедряет AI-конструктор букетов как конкурентное преимущество;
* обеспечивает гибридный офлайн-режим для персонала;
* автоматизирует складской учёт и онлайн-оплату.

На основе выявленных требований и критериев переход к стадии технического проектирования представляется целесообразным и своевременным.

# 2. Проектная часть

## 2.1 Обоснование проектных решений

В рамках проектной части принято решение интегрировать внешний сервис для генерации превью собранных букетов вместо разработки собственной нейросети. Во-первых, создание и обучение собственных моделей требует значительных вычислительных ресурсов, лицензий и штата специалистов по машинному обучению, что выходит за рамки бюджета и сроков реализации проекта. Внешний сервис же предоставляет готовый API с понятным интерфейсом: при изменении состава букета фронтенд отправляет параметры в сервис, а в ответ получает высококачественное изображение композиции за доли секунды. Это позволяет быстро и надёжно визуализировать любые сочетания цветов без дополнительных затрат на инфраструктуру. Для управления магазином было решено разработать отдельное мобильное приложени. Такой подход обеспечивает максимальную мобильность сотрудников: флористы и курьеры получают в своём кармане весь функционал админ-панели (список активных заказов, их статусы, контактные данные клиентов) с возможностью работы офлайн. При отсутствии интернета приложение хранит данные локально и автоматически синхронизирует изменения при восстановлении соединения. Мобильный интерфейс упрощает приём и обработку заказов на ходу и снижает зависимость от постоянного доступа к компьютеру, что особенно важно в условиях курьерской доставки и оперативного управления в точке продаж.

## 2.2 Архитектура системы

Диаграмма общей архитектуры представлена на рисунке №1.

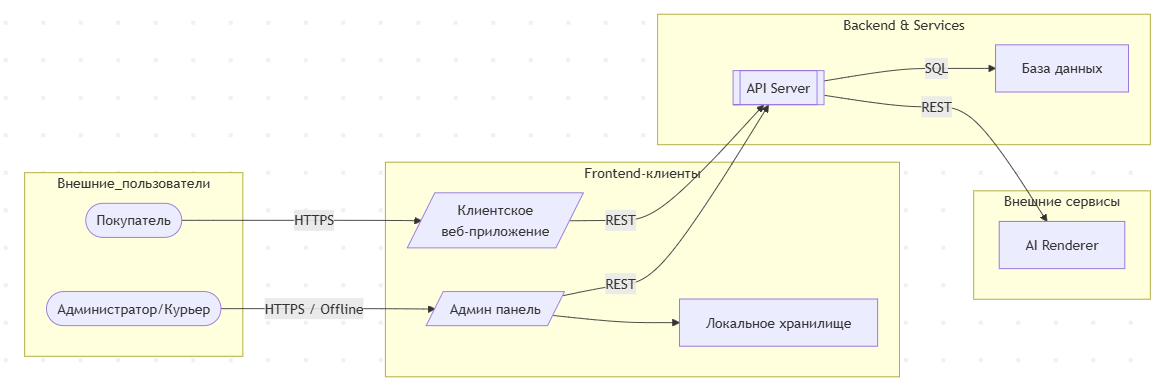


Рисунок №1 – «Общая архитектура ИС»

Диаграмма фиксирует логическую структуру разрабатываемой информационной системы и основные каналы взаимодействия между её компонентами. Для наглядности использована нотация Mermaid flowchart, где блоки-актеры представлены прямоугольниками/плашками, а потоки данных — направленными дугами с подписью протокола.

### 2.2.1 Внешние пользователи

**Покупатель — это любой клиент, который оформляет заказ посредством официального сайта магазина.**

**Администратор, флорист и курьер — сотрудники, одновременно работающие в специализированном мобильном приложении: администратор располагает полными правами доступа; флорист отвечает за подтверждение и тщательную комплектацию заказов; курьер своевременно переводит статусы на «В пути» и «Доставлено» соответственно.**

### 2.2.2 Уровень Frontend

Описание Frontend-компонентов включает три ключевых элемента, каждый из которых выполняет свою специфическую роль в общей системе:

Компонент Клиентское веб-приложение обеспечивает публичный интерфейс: каталог товаров, конструктор букетов, корзину и весь процесс оформления заказа. Он запускается непосредственно в браузере и автоматически подстраивается под ширину экрана от 360 до 1920 пикселей, что гарантирует удобный и понятный пользовательский опыт как на смартфонах, так и на больших десктопах. Адаптивная верстка позволяет сохранить целостность дизайна и функциональности при любых размерах экрана.

Админ панель приложение, которое представляет единый интерфейс для трёх типов внутренних пользователей: администратора, флориста и курьера. Приложение предоставляет доступ ко всем необходимым инструментам: приём и распределение заказов для администратора, подтверждение и комплектование для флориста, а также управление статусами доставки для курьера.

Локальное хранилище интегрировано приложение и отвечает за сохранение данных о заказах при отсутствии интернет-соединения. Даже в офлайн-режиме сотрудники могут просматривать детали текущих и прошлых заказов. Как только устройство восстанавливает связь с сервером, все изменённые или новые данные автоматически синхронизируются, обеспечивая актуальность информации в системе.

### 2.2.3 Уровень Backend

Backend-уровень системы состоит из двух основных компонентов, которые вместе обеспечивают хранение данных, реализацию бизнес-логики и взаимодействие с фронтенд-приложениями:

Сервер приложений предоставляет REST-эндпоинты для работы с каталогом товаров и оформлением заказов, а также управляет всеми основными бизнес-процессами системы. Через эти API клиенты получают информацию о доступных цветах и готовых букетах, создают новые заказы и отслеживают их статус. Кроме того, сервер отвечает за проверку и подтверждение полномочий пользователей, реализуя все правила авторизации и аутентификации, что гарантирует безопасность и разделение прав доступа для администратора, флориста и курьера.

В качестве основного хранилища данных системы будет использоваться SQL база данных. В ней будут храниться все ключевые сущности приложения: информация о цветах (flowers), готовых композициях (bouquets), заказах (orders) и пользователях (users). Для автоматизации работы с продуктами, имеющими ограниченный срок годности, в базе настроены триггеры: как только цветы достигают конца своего срока годности, соответствующая запись автоматически помечается или удаляется из доступного запаса. Это позволяет всем компонентам системы работать с актуальными данными, исключая риск продажи испортившихся товаров.

### 2.2.4 Уровень внешних сервисов

**AI Renderer** — облачный сервис генеративной графики. Через REST-API принимает текстовый промпт + параметры букета и возвращает фотореалистичное изображение.

### 2.2.5 Потоки взаимодействий

**Покупатель оформляет заказ через веб-интерфейс (Web) по защищённому протоколу HTTPS. В браузере клиент собирает букет в конструкторе, добавляет его в корзину и подтверждает покупку. После этого сформированные данные заказа отправляются по REST-запросу напрямую на API Server, где они обрабатываются дальше.**

**Администратор, флорист и курьер работают через единую админ панель. Когда устройство находится в сети, все запросы и обновления статусов (приём заказа, комплектация, отправка в доставку, подтверждение доставки) происходят по HTTPS/REST к тому же API Server. Если связь временно отсутствует, приложение переключается на локальное хранилище, откуда берёт информацию об активных заказах и позволяет продолжать работу в офлайн-режиме. Как только соединение восстанавливается, все накопленные изменения автоматически синхронизируются с сервером через REST-запросы.**

**И веб-клиент, и админ панель используют единый набор эндпоинтов API Server, реализованный по принципу REST. Это позволяет централизовать всю бизнес-логику и точки входа: каталог цветов, управление корзиной и заказами, авторизация пользователей (покупателей и сотрудников), а также маршрутизация запросов к другим внутренним сервисам.**

**Для постоянного хранения и чтения данных об объектах приложения (цветы, готовые букеты, заказы, пользователи) API Server взаимодействует с реляционной базой данных (SQL). Все операции записи и выборки выполняются через оптимизированные SQL-запросы — это обеспечивает консистентность данных и надёжность при одновременном обслуживании большого числа пользователей.**

**Отдельный внешний сервис AI Renderer служит для генерации изображений букетов по запросам конструктора на фронтенде. При каждом обращении в конструктор сервер формирует REST-запрос к сервису, получает оттуда сгенерированное изображение и возвращает его обратно клиентскому приложению в составе ответа API. Таким образом, пользователь сразу видит визуализацию своего букета, созданную с помощью ИИ.**

## 2.3 Концептуальное проектирование

### 2.3.1 Диаграммы вариантов использования

Диаграмма вариантов использования представляет собой графическое отображение ключевых функций системы, доступных для взаимодействия с пользователями. Она служит отправной точкой при проектировании и разработке программного обеспечения, отражая концептуальное понимание будущей системы.

Основные задачи построения диаграммы вариантов использования включают:

* определение границ системы и основных аспектов предметной области на начальной стадии проектирования;
* выявление требований к функциональности разрабатываемого программного продукта;
* создание базовой модели, которая впоследствии детализируется в логическом и физическом представлении системы.

Смысл диаграммы заключается в отображении сценариев взаимодействия между системой и внешними участниками процесса (актёрами). Под актёрами подразумеваются любые сущности, которые инициируют взаимодействие с системой — это могут быть люди, программные компоненты или технические устройства. Каждый вариант использования описывает последовательность операций, выполняемых системой в ответ на действия актёра.

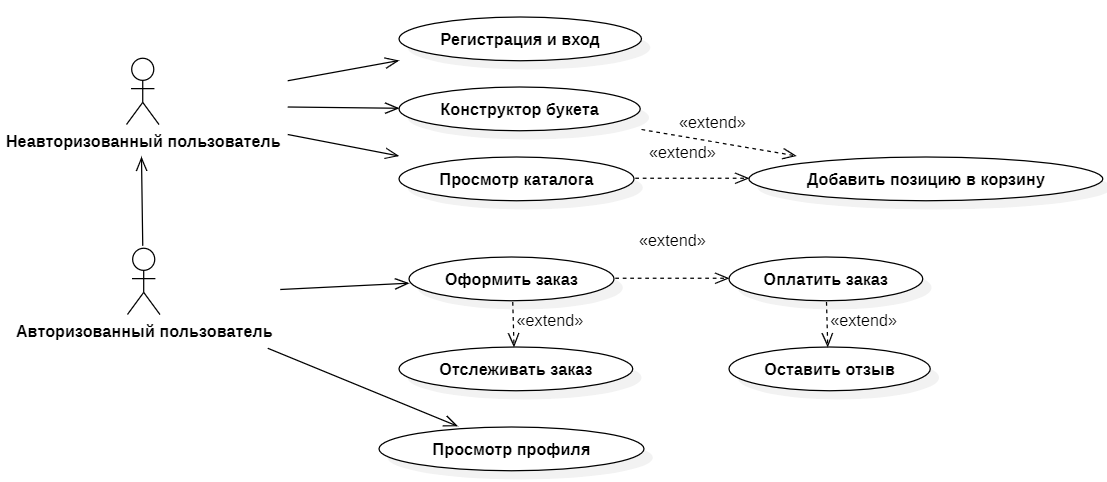


Рисунок №2 – «Диаграмма вариантов использования авторизованного/неавторизованного пользователя»

В рамках данной системы разработаны две отдельные диаграммы:

* диаграмма вариантов использования пользовательского интерфейса, охватывающая поведение как авторизованных, так и неавторизованных пользователей (см. Рисунок №2);
* диаграмма взаимодействий в мобильном приложении, предназначенная для ролей администратора, флориста и курьера (см. Рисунок).

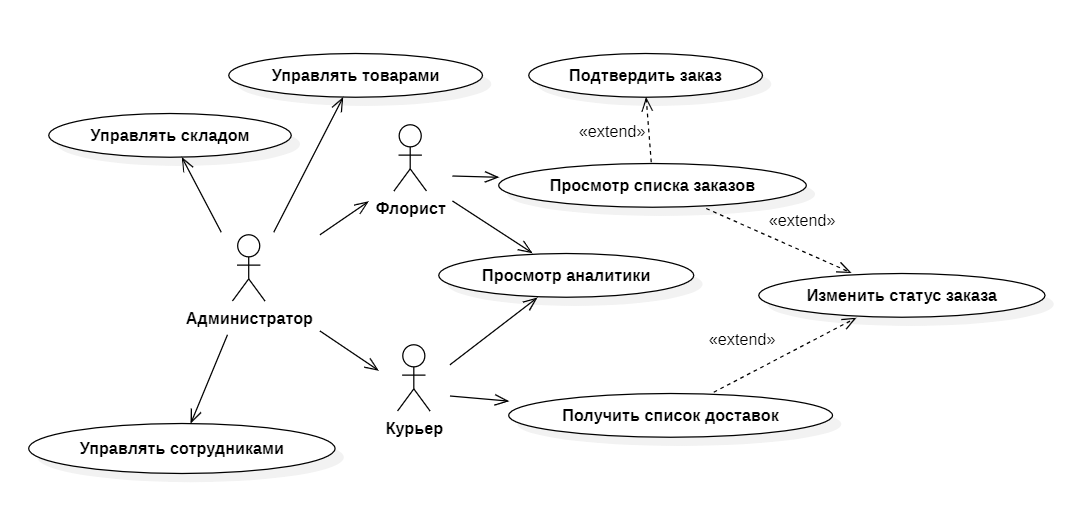


Рисунок №3 – «Диаграмма вариантов использования Администратора/Курьера/Флориста»

Основные варианты использования:

Веб-сайт (две роли):

**Регистрация / вход / выход** – базовая точка доступа ко всем персональным функциям.

**Просмотр каталога** – свободный обход витрины, фильтр по цене.

**Конструктор букета** – интерактивная сборка композиции; вызывает ИИ-превью.

**Добавить в корзину** («include» к каталогу и конструктору) – кладёт выбранный товар в локальный список.

**Оформление заказа** – ввод адреса и времени доставки.

**Оплата заказа**(«include» к оформлению) – после подтверждения статус «Оплачен».

**Отслеживание заказа –** лента статусов «Принят – Сборка – В пути – Доставлен».

**Оставить отзыв** («extend» к отслеживанию, активируется после «Доставлен») – оценка букета и курьера.

**Управление профилем** – изменение контактов, список прошлых заказов.

Мобильное административное приложение (три роли):  
- Администратор

Управление товарами – может добавлять, редактировать и удалять товары, которые будут отображаться в каталоге у пользователя

Управление складом – оформление поставок и списаний магазина.

Управление сотрудниками – добавление новых сотрудников, назначение им ролей

- Флорист

**Список заказов** – все текущие заявки подгружаются и кешируются офлайн.

**Подтвердить заказ** (include к списку), перевод «Новый» – «Принят».

**Управление каталогом** – добавление / изменение позиций (только онлайн).

**Просмотр аналитики** – быстрые сводки продаж.

- Курьер

**Мои доставки** – Список назначенных заказов с адресами.

**Изменить статус** – «В пути» и «Доставлен»; изменения синхронизируются при появлении сети (extend к «Мои доставки»).

**Просмотр аналитики** – быстрые сводки продаж.

### 2.3.2 Диаграммы последовательности

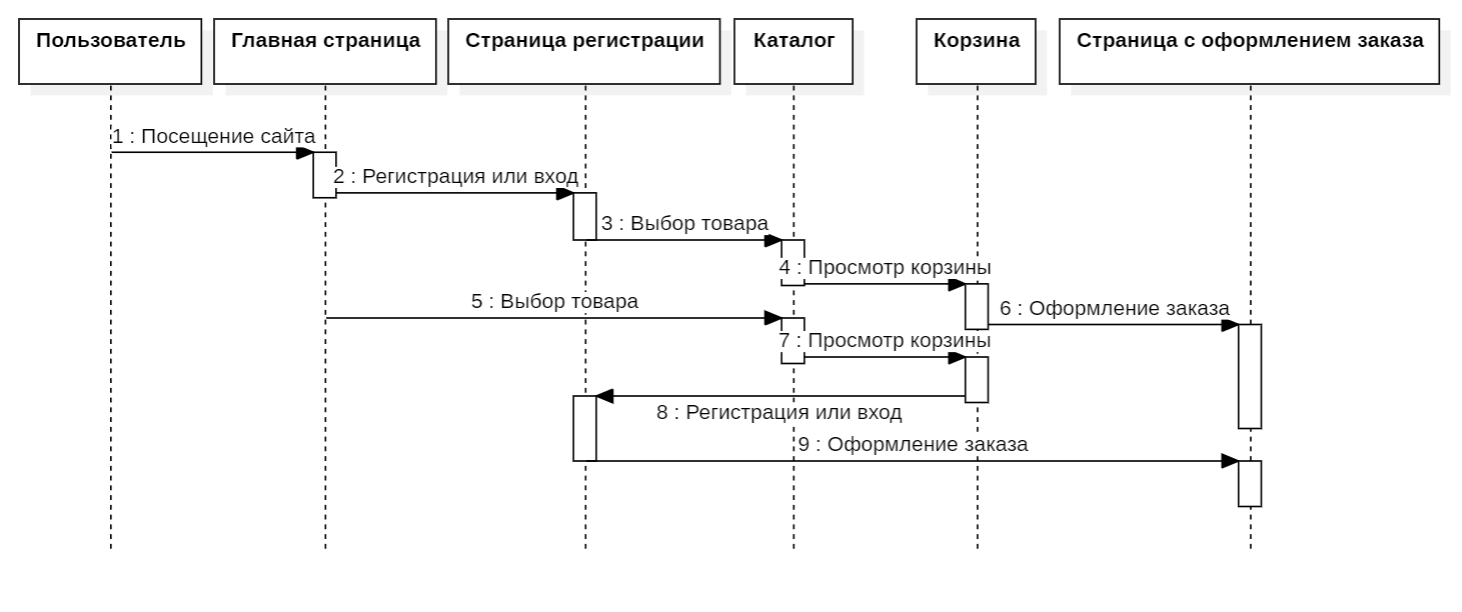


Рисунок №4 – «Последовательность действий при заказе товара»

Диаграммы последовательностей служат для отображения процесса взаимодействия между объектами в рамках одного сценария использования. Они показывают, каким образом компоненты системы обмениваются сообщениями для реализации определённой функциональности, а также фиксируют порядок этих взаимодействий. Такие диаграммы позволяют наглядно проследить логику выполнения операций в конкретной ситуации. Для полноты описания системы необходимо составить диаграммы последовательностей, отражающие действия каждого типа пользователя. Ниже представлена диаграмма последовательности для покупателя при оформлении заказа (Рисунок №4):

После того как пользователь открывает сайт, управление переходит к главной странице, где система сразу предлагает авторизоваться: если аккаунт отсутствует, посетитель заполняет форму регистрации и автоматически возвращается на главную уже в статусе «вошёл». Далее покупатель переходит в каталог, просматривает ассортимент и кладёт понравившийся букет в корзину; далее пользователь переходит на страницу корзины, чтобы посмотреть содержание заказа и его итоговую сумму. Нажатие кнопки «Оформить» переносит его на страницу оформления заказа, где нужно подтвердить адрес и время доставки. Если на этот момент пользователь оказался неавторизованным (например, собирал корзину как гость), система делает обязательный возврат к форме входа / регистрации, а после успешной авторизации снова перенаправляет на страницу оформления. Заполнив все поля и подтвердив заказ, клиент видит финальное сообщение об успешном оформлении; на этом последовательность завершается и дальнейшее взаимодействие переходит в режим отслеживания статуса заказа.

Диаграмма последовательности действий для флориста при сборке заказа представлена на рисунке №5.

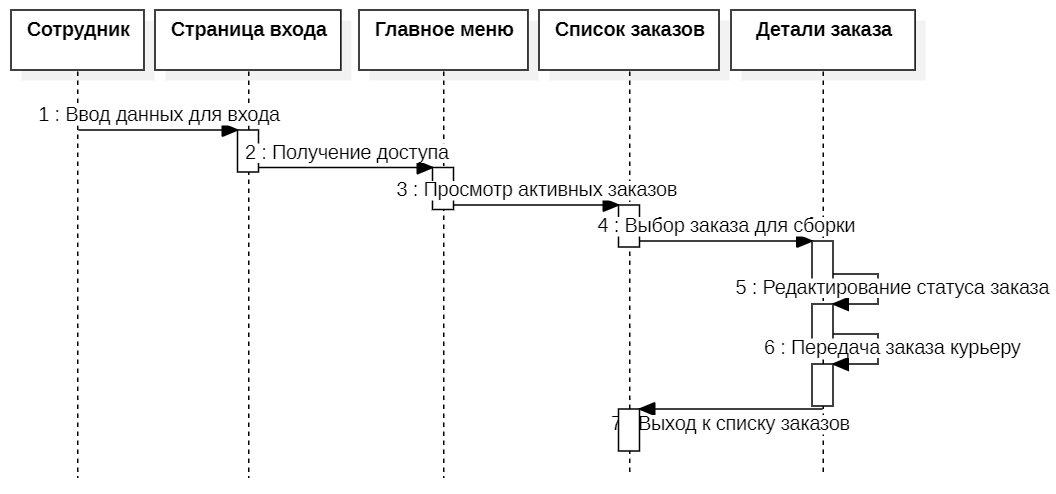
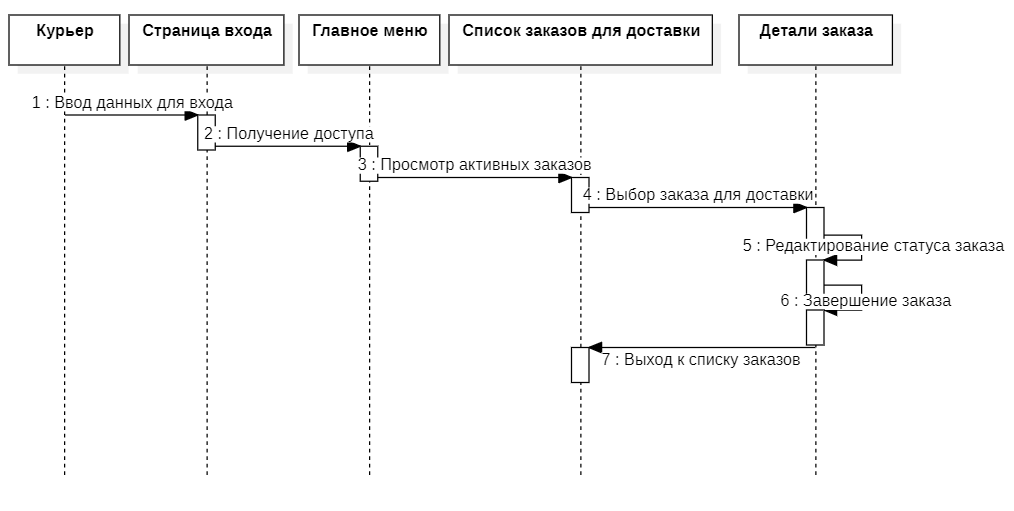


Рисунок №5 – «Диаграмма последовательности действий при сборке заказа»

После того как сотрудник-флорист запускает мобильное приложение, он попадает на страницу входа и вводит свои учётные данные. Приложение подтверждает права доступа и переводит пользователя в главное меню (шаг 2 на диаграмме). Из меню флорист открывает список заказов; в этот момент приложение подгружает (или считывает из офлайн-хранилища) перечень всех активных заявок на сборку — шаг 3 «Просмотр активных заказов». Обнаружив нужную позицию, сотрудник нажимает на карточку — происходит переход к экрану деталей заказа (шаг 4). Здесь ему отображается состав букета, комментарии клиента и целевое время доставки.

После того как флорист выбрал заказ для сборки композиции, он меняет статус на «Принят на сборку» (шаг 5 «Редактирование статуса заказа»). Событие фиксируется локально и, если связь доступна, сразу уходит на сервер. Далее он нажимает кнопку «Собран» (шаг 6). Система назначает исполнителя и отправляет push-уведомление на устройство курьера. На этом работа с конкретным заказом завершена, и приложение автоматически возвращает флориста назад к списку заказов, чтобы можно было выбрать следующую заявку (последняя стрелка «Выход к списку заказов»).

Диаграмма последовательности действий для доставки заказа курьером представлена на рисунке №6.

Рисунок №6 – «Диаграмма последовательности действий при доставке заказа»

При запуске мобильного приложения курьер попадает на страницу входа и вводит логин/код подтверждения (шаг 1 «Ввод данных для входа»). После успешного входа система подтверждает права и переводит пользователя в главное меню (шаг 2 «Получение доступа»). Из главного меню курьер открывает список заказов для доставки. Приложение поднимает данные из локального кеша, а при наличии сети запрашивает свежие данные с сервера — таким образом на шаге 3 он видит полный перечень назначенных ему адресов даже офлайн. Выбрав конкретную запись (шаг 4 «Выбор заказа для доставки»), курьер переходит к экрану деталей заказа: адрес места доставки, телефон получателя, таймер желаемого времени вручения. По выезду к клиенту он нажимает кнопку «В пути», что фиксируется как изменение статуса (шаг 5 «Редактирование статуса заказа»). После вручения букета курьер нажимает «Доставлен» — шаг 6 «Завершение заказа». Статус обновляется до финального, и клиент в профиле видит заказ как «Завершенный». Завершив операцию, приложение автоматически возвращается к списку заказов (шаг 7 «Выход к списку заказов»), чтобы курьер мог перейти к следующей адресной точке. Благодаря такой последовательности он не теряет данные при плохом интернете и всегда знает актуальную очередь доставок.

### 2.3.3 Диаграмма компонентов

Диаграммы компонентов предоставляют обобщённое представление о взаимодействии между различными частями системы, отображая ключевые модули приложения и связи между ними. Такой тип диаграмм позволяет сформировать общее понимание архитектуры программного обеспечения. В рассматриваемом случае используется обзорная форма диаграммы, где связи между компонентами представлены как зависимости. При этом интерфейсы и порты в данной визуализации не отображаются, поскольку акцент сделан именно на логических взаимосвязях между модулями.

Диаграмма компонентов Frontend части web-приложения представлена на рисунке №7.

В верхней части показана иерархия страниц клиентского Web-приложения. Центральным узлом является главная страница, от которой расходятся основные разделы: «Контакты», «О магазине», «Корзина», «Каталог товаров» и «Профиль» пользователя.

* ветвь «Корзина» ведёт к отдельной странице оформления заказа;
* ветвь «Каталог» раскрывается в карточки конкретных товаров;
* ветвь «Профиль» содержит информацию с историей и статусами заказов.

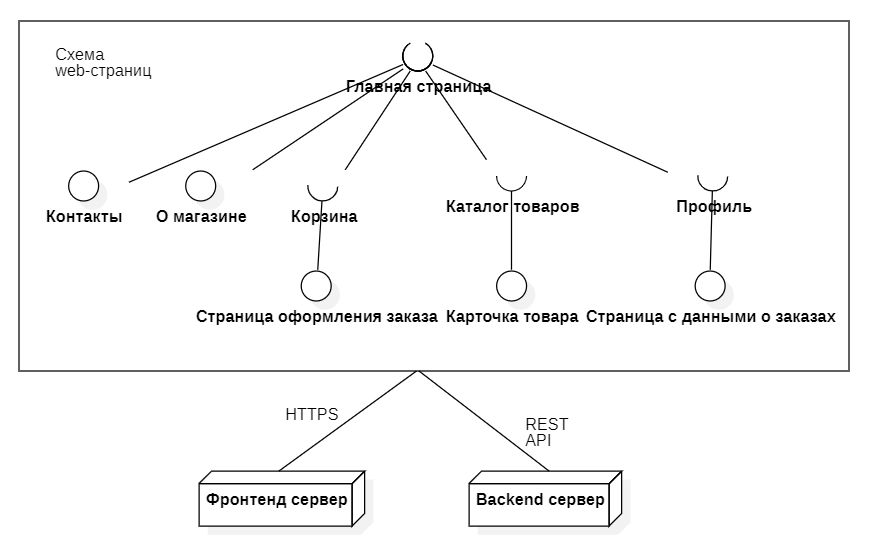


Рисунок 7 – «Диаграмма компонентов Frontend»

Нижний блок иллюстрирует разделение по слоям: фронтенд-сервер (статические файлы React-приложения), доступный клиенту по HTTPS, и отдельный backend-сервер c REST API, который обслуживает бизнес-логику и связь с базой данных. Связь между слоями осуществляется через JSON-запросы, а пользователь взаимодействует только c фронтендом.

Диаграмма компонентов мобильного приложения представлена на рисунке №8.

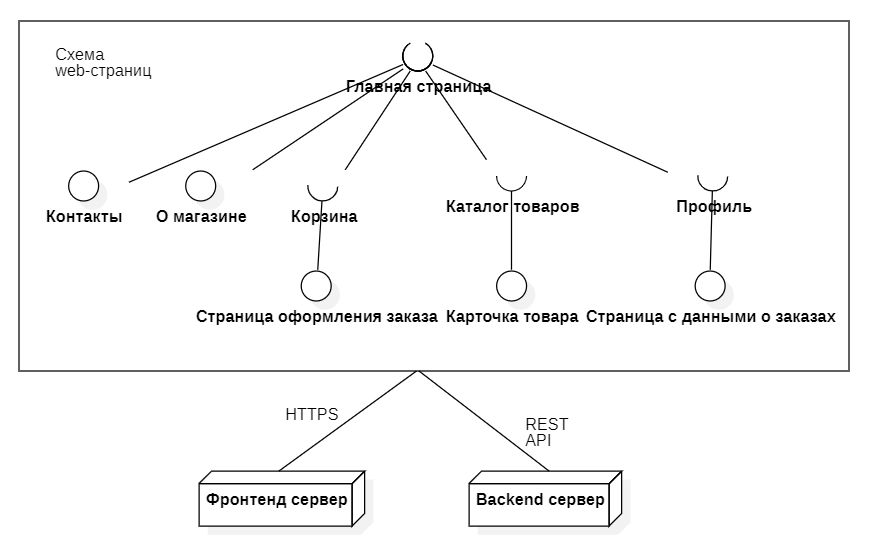


Рисунок 7 – «Диаграмма компонентов Frontend»

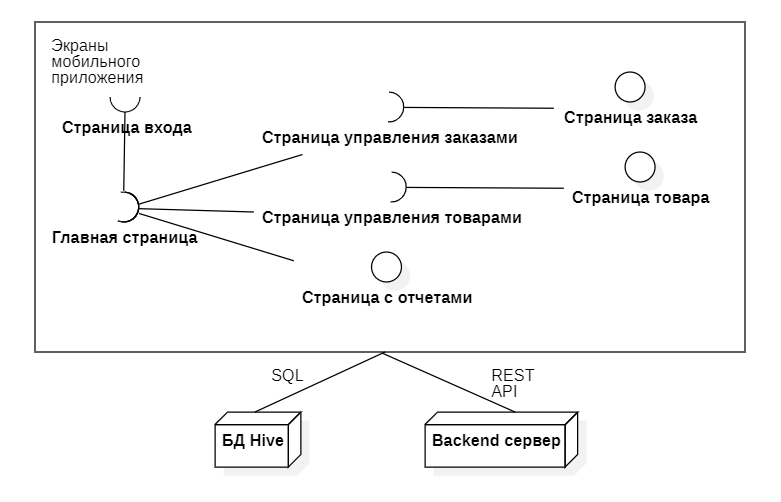


Рисунок №8 – «Диаграмма компонентов мобильного приложения»

В верхнем блоке показана навигация мобильного приложения. Стартовым узлом является страница входа: после авторизации пользователь попадает на главную страницу.

Из главной доступны три функции:

Управление заказами — список активных заявок; отдельный переход ведёт к странице конкретного заказа с деталями и кнопкой смены статуса.

Управление товарами — таблица каталога; из неё открывается страница товара, где можно изменить цену или описание.

Отчёты — экран графиков и экспортных таблиц.

Нижний блок иллюстрирует два источника данных приложения:

БД SQLite — локальное хранилище, в которое выгружаются активные заказы и товары, оно обеспечивает работу офлайн.

Backend-сервер — удалённый REST API, с которым приложение синхронизирует изменения и получает свежие данные при наличии сети.

Таким образом диаграмма объединяет навигационную карту экранов с техническими каналами — локальной БД и облачным API.

Диаграмма компонентов backend части системы представлена на рисунке №9

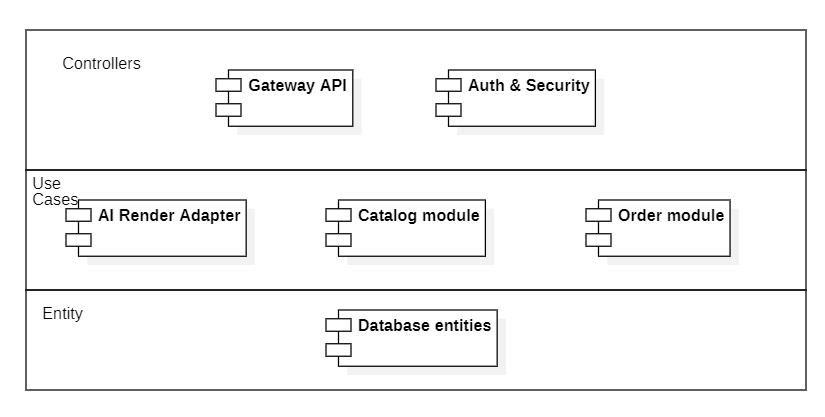


Рисунок №9 – «Диаграмма компонентов backend»

На схеме показаны три горизонтальных слоя серверного приложения.

**а) Controllers** — самый внешний уровень.

Gateway API принимает HTTP-запросы от сайта и мобильного клиента, превращает их в DTO и передаёт внутрь системы. Auth & Security — фильтр, который до выполнения бизнес-логики проверяет JWT-токен и роли пользователя.

б) **Use Cases** (прикладной слой) содержит конкретные сценарии работы магазина. Catalog module управляет цветами и букетами: добавление, изменение цены, поиск. Order module создаёт заказ, меняет его статус и рассчитывает итоговую стоимость. AI Render Adapter инкапсулирует обращение к Kandinsky-API — от бизнес-кода это скрыто за интерфейсом.

**в) Entity** — внутренний, наиболее стабильный слой. Здесь живут JPA-сущности и базовые правила предметной области (параметры букета, ограничения по количеству, расчёт цены позиции).

Зависимости направлены «снаружи к центру»: контроллеры знают о сценариях, сценарии — о сущностях, но сущности не зависят ни от веб-фреймворка, ни от внешних API.

**Преимущества такого деления, заключаются в следующем:**

а) Чистая ответственность – веб-детали, бизнес-логика и данные не смешиваются.

б) Тестируемость – use-case-слой можно тестировать, подменив репозитории заглушками.

в) Гибкость – смена СУБД или переход с REST на GraphQL затрагивает только внешний слой, не ломая домен.

г) Безопасность – все операции проходят через проверенный поток «контроллер – сценарий – сущность», прямого доступа к БД извне нет.

д) Командная работа – фронтенд опирается только на контракт Gateway API, в то время как бэкенд-команда независимо улучшает внутренние модули. В контексте нынешней разработки это не имеет значания, но в будущем, если проектом будет заниматься команда разработчиков, это будет весомым преимуществом.

## 2.4 Проектирование базы данных

Логическое проектирование – это описание и построение схем связей между элементами данных, не затрагивая их содержание и среду хранения*.*

Логическая модель, представленная на рисунке ниже, была доведена до состояния третьей нормальной формы (3NF):

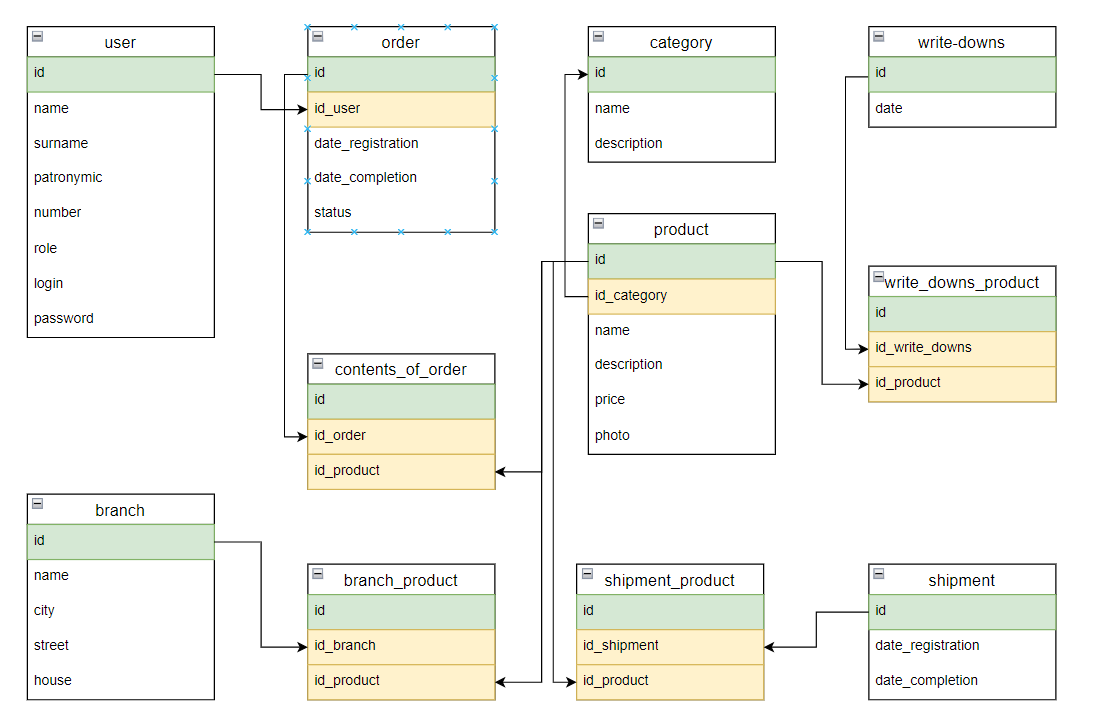


Рисунок 10 – «Логическая модель БД в 3NF»

В предлагаемой дипломной работе логическая модель базы данных веб-приложения «цветочный магазин» строится на выделении ключевых предметных сущностей и их взаимосвязей, без привязки к конкретной реализации СУБД. Логическая диаграмма данных является схематическим отображением сущностей и связей между ними. В нашей системе в числе основных объектов – пользователи (клиенты магазина), заказы, товары (цветы, букеты, сопутствующие продукты) и категории товаров. Эти таблицы составляют основу структуры БД и отражают ключевые бизнес-процессы приложения. Так, таблица User содержит информацию о зарегистрированных пользователях – покупателях цветочного магазина (контактные данные, учётные записи и т.п.). Каждый пользователь может формировать множество заказов (отношение «один ко многим»), поэтому в таблице Order хранится внешний ключ, ссылающийся на соответствующую запись в User. Это обеспечивает связь «многие заказы – один пользователь». любые данные о покупках конкретного клиента связываются с его профилем через этот ключ. Кроме того, в таблице Order могут содержаться ссылки на филиалы (Branch), через которые осуществляется доставка или самовывоз, что также организовано через внешний ключ.

Детали каждого заказа оформляются в отдельной связующей таблице contents\_of\_order, аналогично таблице purchase\_item из примеров реляционных. Эта таблица содержит записи о конкретных товарах и их количестве в рамках каждого заказа: ее строки связывают один заказ с одним товаром и указывают, сколько единиц данного товара было куплено. Таким образом, реализуется связь многие-ко-многим между заказами и товарами: один заказ может включать много разных товаров, а один товар (наименование цветка или букета) может присутствовать во множестве заказов разных клиентов. Для организации этой связи таблица contents\_of\_order содержит два внешних ключа – order\_id и product\_id, указывающие на первичные ключи таблицы Order и Product соответственно. Например, учебная схема базы сети магазинов бытовой техники включает аналогичную таблицу purchase\_item, «содержащую информацию о товарах, купленных в рамках одной покупки». Сущность Product описывает каждый товар магазина: разные сорта цветов, цветочные композиции, сопутствующие товары и услуги. У каждой записи товара есть атрибуты наименования, описания, цены и т.д.

Важным элементом является связь товаров с их категориями – таблица Category. Категории служат справочником для классификации продукции (например, «Розы», «Тюльпаны», «Букеты» и т.п.) и позволяют группировать похожие товары. В таблице Product предусмотрено поле category\_id как внешний ключ на Category, что реализует отношение «многие товары – одна категория» (категория содержит множество товаров). Примерно аналогичная структура представлена в описании учебной БД магазина бытовой техники: там также отдельно хранятся данные о товарах и о категориях, к которым они относятся. Справочник категорий облегчает фильтрацию, поиск и другие операции над товарами. Кроме того, для букетов реализуется раздельный учёт их составов: если товар является букетом, то связь «букет – цветы» моделируется через дополнительную связующую таблицу. Такая таблица может содержать идентификатор букета и идентификаторы входящих в него сортов цветов с указанием количества каждого. Это опять же отношение «многие ко многим» (каждый букет состоит из множества цветов, и каждый сорт может быть частью разных букетов), которое реализуется через внешний ключ на таблицу букетов и внешний ключ на таблицу товаров-цветов.

Таким образом, структура поддерживает учёт состава букетов и позволяет при продаже автоматически списывать со склада соответствующее количество каждого входящего в состав цветка. Для учёта запасов товара в разных точках продаж выделена сущность Branch (филиал или магазин). Таблица Branch хранит информацию о каждом филиале – его название, адрес и прочие характеристики. Операции прихода товара оформляются через таблицу Shipment (аналогично «поставкам»): каждая запись в Shipment указывает на конкретный товар и конкретный филиал (внешние ключи product\_id и branch\_id), а также количество доставленного товара и дату поступления. Это означает, что для каждого филиала может быть множество поставок – связь «один филиал – многие поставки», и для каждого товара – множество записей о поставках «товар доставлен в филиал»

Аналогичным образом организовано и списание просроченных или испорченных товаров: таблица Write\_downs содержит записи о выбытии товара из учёта, ссылаясь на товар и на филиал, где произошло списание. Таким образом, и поступление, и списание товара моделируются через внешние ключи к таблицам Product и Branch, что позволяет детально отслеживать остатки каждого вида товара по филиалам. В готовой учебной базе данных это соответствует таблице deliveries – поставки товаров в магазины.

Все взаимосвязи между перечисленными таблицами реализуются через внешние ключи (foreign keys), указывающие на первичные ключи других таблиц. В результате формируется типичная реляционная структура: один пользователь может иметь много заказов (1:M), один заказ – много позиций товара (1:M), а один товар может попадать в разные заказы и браться на поставку в разные филиалы (многие-ко-многим, реализуемое через связующие таблицы).

Также связь один-ко-многим связывает категории с товарами и филиалы с операциями поставки/списания. Такая организация обеспечивает целостность и непротиворечивость данных: например, в каждом заказе гарантированно существует ссылка на конкретного пользователя и филиал через внешние ключи, а описанная модель позволяет гибко формировать запросы на анализ продаж, проверку остатков, учет доходов от продаж и т.п. Соответственно, описанная логическая схема охватывает всю необходимую предметную область – хранение информации о покупателях, ассортименте магазинов, заказах, поставках и списаниях, а также отделенном учёте составов букетов – и отражает все требуемые бизнес-объекты и справочники.

## 2.5 Вывод по проектной части

В ходе проектного этапа я сформировал целостную архитектурную модель веб-системы и зафиксировал ключевые артефакты, которые будут служить основой для последующей реализации.

Функциональная декомпозиция.

На уровне пользовательских сценариев определены две независимые зоны: публичное клиентское приложение (каталог, конструктор букета, корзина, оплата, личный кабинет) и админ-панель (заказы, сборка, доставка, каталог, отчёты). Для каждой зоны подготовлена карта экранов с навигационными связями; макеты главных интерфейсов подтверждают, что поток действий пользователя остаётся линейным и не перегружен переходами.

Логическая архитектура backend.

Принято трёхслойное деление — Controllers → Use-Cases → Entity.

Gateway API и Auth & Security обрабатывают HTTP-запрос, а бизнес-модули (Catalog, Order, AI Render) инкапсулируют предметную логику. Такое расслоение упрощает тестирование сервисов и даёт возможность при необходимости вынести модули в отдельные микросервисы без переписывания домена.

Интеграции и офлайн-механизм.

Под офлайн-работу мобильного клиента выбран Embedded-SQL движок Hive (Isar на Flutter). Фоновый энд-пойнт /orders/delta отдаёт изменения по метке времени, что минимизирует трафик и позволяет курьеру видеть заказы даже в «зоне без сети». Связка Spring Boot + PostgreSQL + Hive обеспечит консистентность данных за счёт двусторонней синхронизации статусов.

Концепция базы данных.

Предложенная ER-схема (рис. 10) содержит 12 сущностей и охватывает учёт пользователей, заказов, товарных позиций, поставок, списаний и филиалов.

связь order - contents\_of\_order - product описывает состав заказа;

branch\_product отражает распределение товара по точкам;

write\_downs и write\_downs\_product позволяют фиксировать списание просроченных цветов, что важно для реальной флористики.

Такая структура покрывает как онлайн-продажи, так и внутренний учёт магазина.

Визуальная часть.

Разработаны прототипы web-страниц для дальнейшей реализации.

Итог: проектная часть сформировала полный набор проектных решений — от пользовательских сценариев до ER-модели и границ модулей. Эти материалы позволяют без изменения требований перейти к технологическому этапу: настройке окружения, написанию кода контроллеров и сервисов, разработке экранов и тестированию.

# 3. Технологическая часть

## 3.1 Используемые программные продукты

Каждая из используемых технологий приносит собственную ценность: Java + Spring Boot 3 с Spring Data JPA ходят рука об руку, обеспечивая надёжный и масштабируемый бэкенд; PostgreSQL предоставляет мощную и гибкую СУБД для хранения всех сущностей; React с HTML5/CSS3/ES6+ создаёт адаптивный, быстрый и поддерживаемый веб-интерфейс; Kotlin + Room позволяют мобильному приложению работать независимо от соединения и сохранять данные локально; а интеграция с Kandinsky API даёт пользователю мгновенное и качественное превью будущего букета без дополнительных затрат на нейросетевую инфраструктуру. Все эти компоненты вместе формируют современную, отказоустойчивую и расширяемую систему управления магазином «Веточка». Дальше написано подробнее о выбранной технологии для реализации каждого компонента системы.

### 3.1.1 Серверная часть

Java уже давно зарекомендовала себя как надёжная платформа для создания масштабируемых и производительных серверных приложений. Spring Boot, в свою очередь, упрощает всё, что связано с настройкой и развёртыванием: благодаря автоконфигурации в пару строк кода вы получаете встроенный веб-сервер (например, Tomcat) и возможность запускать приложение без внешнего контейнера. Именно поэтому мы выбрали связку Java 17 + Spring Boot 3. В проекте «Веточка» Spring Boot отвечает за весь REST-слой: каждый контроллер помечается аннотациями @RestController и @RequestMapping, а сам запуск с main() сразу разворачивает HTTP-сервер на нужном порту. Для взаимодействия с базой данных мы используем Spring Data JPA, что даёт несколько важных преимуществ. Во-первых, интерфейсы JpaRepository уже содержат готовые методы для CRUD-операций (findAll(), findById(), save(), delete()) без необходимости писать SQL вручную. Во-вторых, при желании сделать более сложный запрос достаточно объявить метод с правильным именем — Spring Data сам сгенерирует нужный SQL. В «Веточке» сущности (User, Product, Order и другие) аннотированы @Entity и связаны между собой через @ManyToOne, @OneToMany и так далее. Как результат, при старте контейнера Hibernate автоматически создаёт схему в нашей базе данных PostgreSQL, и нам даже не пришлось вручную писать DDL-скрипты для таблиц. Кроме того, Spring Data JPA позволяет легко обозначить транзакционные границы через @Transactional — это даёт гарантию, что любая операция с БД будет либо полностью выполнена, либо полностью отменена при ошибке.

### 3.1.2 Основная СУБД

В качестве основной СУБД была выбрана PostgreSQL, так как она отличается высокой надёжностью, строгой поддержкой SQL-стандартов и продвинутыми возможностями (JSONB, массивы, UUID и пр.). В «Веточке» PostgreSQL хранит все данные о товарах, заказах, категориях, филиалах, списаниях цветов и т. д. Мы настроили пул соединений HikariCP, указали диалект Hibernate (PostgreSQLDialect) и включили автоматическую генерацию миграций с помощью Flyway (или Liquibase, если требуется). Одно из важных мест заняли триггеры на уровне БД, которые автоматически списывают срезанные цветы по дате годности, когда меняется запись в таблице «write\_downs». Благодаря этому бизнес-логика остаётся консистентной даже при прямом доступе к БД или при одновременной работе нескольких сервисов.

3.1.3 Веб-клиент

Для клиентской части мы выбрали традиционное сочетание HTML5 и CSS3, обеспечивающее семантическую разметку и адаптивную стилизацию. JavaScript версии ES6+ позволяет писать более короткий и понятный код благодаря стрелочным функциям, деструктуризации объектов и другим современным возможностям. Основу интерфейса составляет React, поскольку эта библиотека поддерживает компонентный подход: каждая часть страницы (каталог, карточка товара, корзина, форма оформления заказа, админ-панель) оформлена как отдельный React-компонент. Это не только облегчает поддержку, но и упрощает тестирование, — при необходимости мы можем подключить Jest или React Testing Library для unit-тестов. Маршрутизацию реализует React Router: перемещение между "/catalog", "/cart", "/admin" происходит без перезагрузки браузера, что создаёт SPA-опыт. Для AJAX-запросов к серверному API используется Axios, хотя можно ограничиться встроенным fetch. Например, при загрузке списка товаров компонент Catalog вызывает axios.get('/api/products') и обновляет состояние с помощью хука useState, а сам запрос оформлен внутри useEffect, чтобы выполняться один раз при монтировании. Стили оформлены с помощью CSS-модулей (имена классов автоматически изолируются по компонентам), что исключает конфликты названий, и при желании можно было подключить какую-нибудь CSS-библиотеку (Tailwind, Bootstrap), но в нашем случае потребовалось лишь минимальное количество правил: .catalog\_\_grid, .product\_\_title и т. д.

### 3.1.4 Админ-панель

Android-часть написана полностью на Kotlin, потому что этот язык сокращает количество «шумового» кода (встроенная проверка на null, data-классы, расширения функций). Для управления локальной базой данных мы применили библиотеку Room, которая представляет собой оболочку над SQLite. Room автоматически генерирует код для CRUD-операций на основе аннотаций @Entity и @Dao, а также проверяет SQL-запросы во время компиляции — это значительно снижает риск ошибок «в рантайме». В «Веточке» сущность OrderEntity описана как @Entity(tableName = "orders"), а DAO-интерфейсы (OrderDao, ProductDao) содержат методы @Insert, @Delete и @Query("SELECT \* FROM orders WHERE status != 'COMPLETED'"), что обеспечивает получение всех активных заказов. Локальные данные кешируются в SQLite, и при отсутствии интернета приложение сохраняет новые заказы и изменения статусов в БД. При восстановлении соединения через Android ConnectivityManager инициируется синхронизация: suspend fun syncOrders() { … } внутри viewModelScope.launch { … }, что возвращает актуальные статусы на сервер и очищает локальный кеш. Компоненты MVVM (ViewModel + LiveData) гарантируют, что UI (Activity/Fragment) автоматически обновляется при изменении записей в БД.

### 3.1.5 Внешний AI-сервис (Kandinsky API)

Для генерации превью «конструируемого» букета мы интегрировали внешний нейросетевой API (Kandinsky), чтобы не тратить время и ресурсы на обучение собственных моделей. Преимущество этого подхода в том, что все вычисления происходят на стороне провайдера: достаточно отправить POST-запрос с текстовым описанием состава (например, «розы, лилии, зелень, бумажная упаковка») и получить в ответ URL готового изображения размером 512×512 за 1–3 секунды. На сервере Spring Boot реализован компонент (Рисунок № 11),



Рисунок №11 – Компонент для отправки запроса во внешний сервис

который формирует prompt на базе выбранных пользователем товаров, отправляет его на Kandinsky и сохраняет возвращённый URL в базе данных. В React-клиенте и в мобильном приложении при первом открытии конструктора мы запрашиваем /api/preview?ids=…, получаем ссылку и отображаем <img src={previewUrl} />. При последующих обращениях, если пользователь сидит в офлайн-режиме, приложение берёт уже сохранённый URL из Room или из локального кеша браузера (IndexedDB), что гарантирует корректное отображение даже без сети.

## 3.2 Описание работы приложений

### 3.2.1 Описание работы клиентского Web-приложения

При переходе по URL адресу магазина, клиента встречает главный экран (Рисунок №12). В верхней части страницы изображены логотип магазина, его название, контакт для связи, панель поиска, а также иконки для перехода в профиль или корзину. Ниже расположена навигационная панель, по которой пользователь может перейти в нужный ему раздел. При нажатии на разделы «О нас» и «Отзывы» происходит переход к соответствующим панелям снизу страницы (Рисунок №13), «Каталог» и «Сборка букета» ведут на соответствующие страницы приложения. Далее в главной части экрана изображены действующие на данный момент акции, их может редактировать управляющий магазина в соответствующем разделе административной панели (см. рисунок №). За акциями следуют готовые букеты (Витринные букеты, которые можно доставить незамедлительно). После того как пользователь ознакомился с главной страницей, он может добавить готовый букет в корзину, либо же перейти в каталог, где предоставлен более широкий спектр товаров.

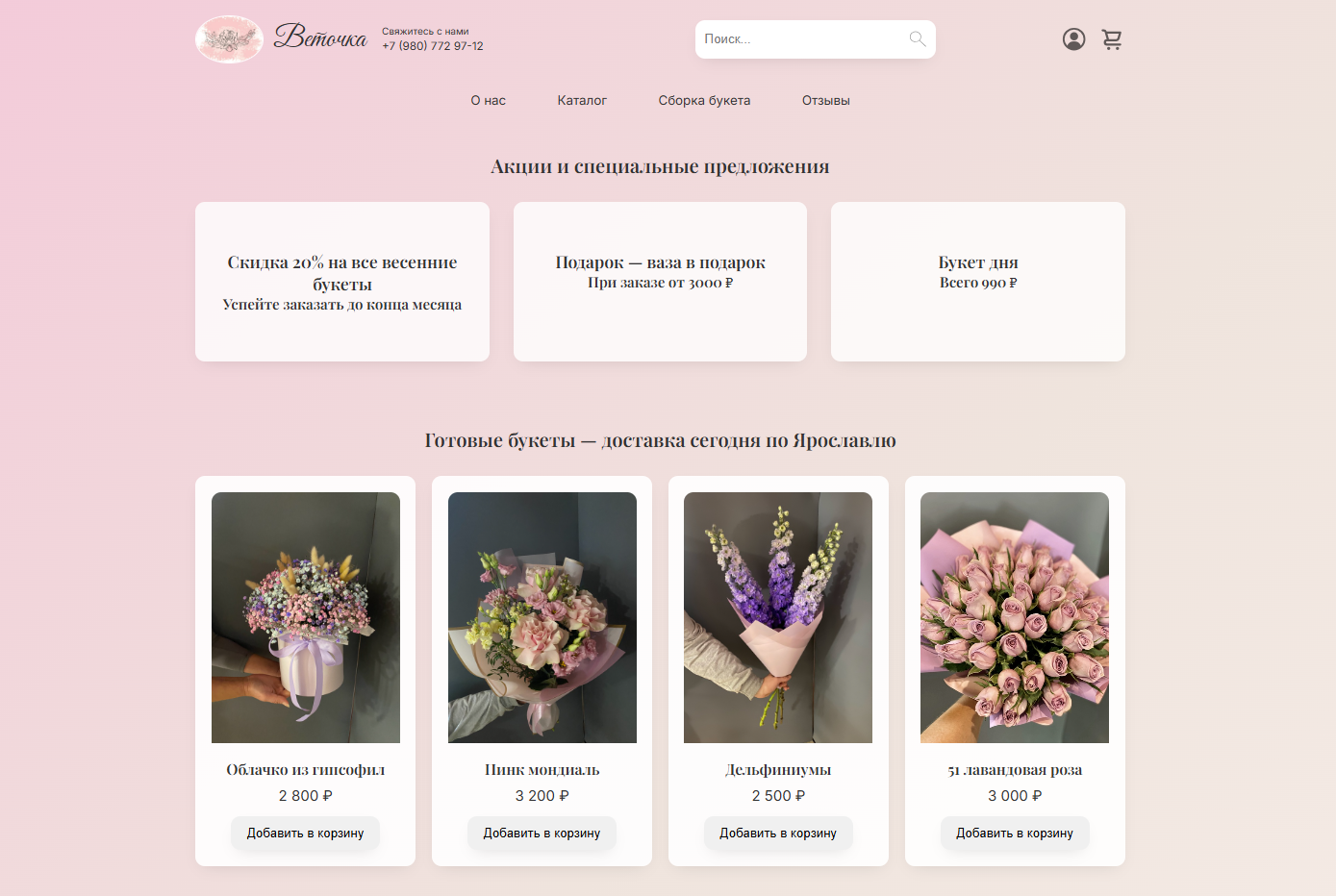


Рисунок №12 - «Главная страница»

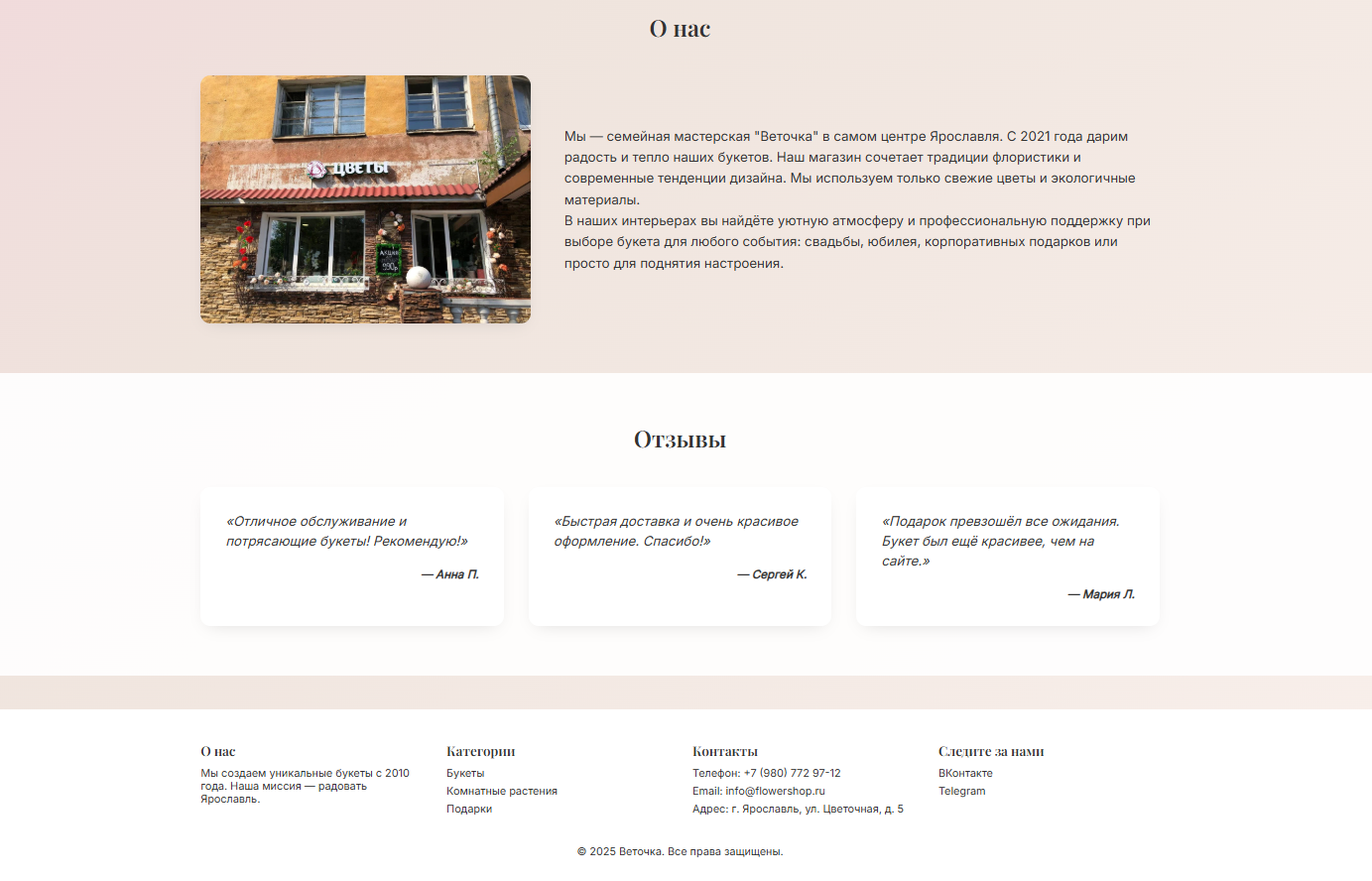


Рисунок №13 - Разделы «О нас» и «Отзывы»

При переходе на страницу каталога товаров (рисунок №14), пользователю предоставляется выбор категорий товаров («Цветы», «Букеты», «Комнатные растения», «Дополнительно»). Так же в каждой категории он может отфильтровать позиции по цене, по популярности, по названию и по наличию в магазине.



Рисунок №14 - Верхняя часть страницы каталога



Рисунок №15 - Нижняя часть страницы каталога

Например, пользователь хочет купить букет «Дельфиниумы», тогда он нажимает кнопку «Добавить в корзину», после чего в корзине появится соответствующая позиция, а возле её иконки и в карточке товара появится количество с возможностью изменения (Рисунок №16).

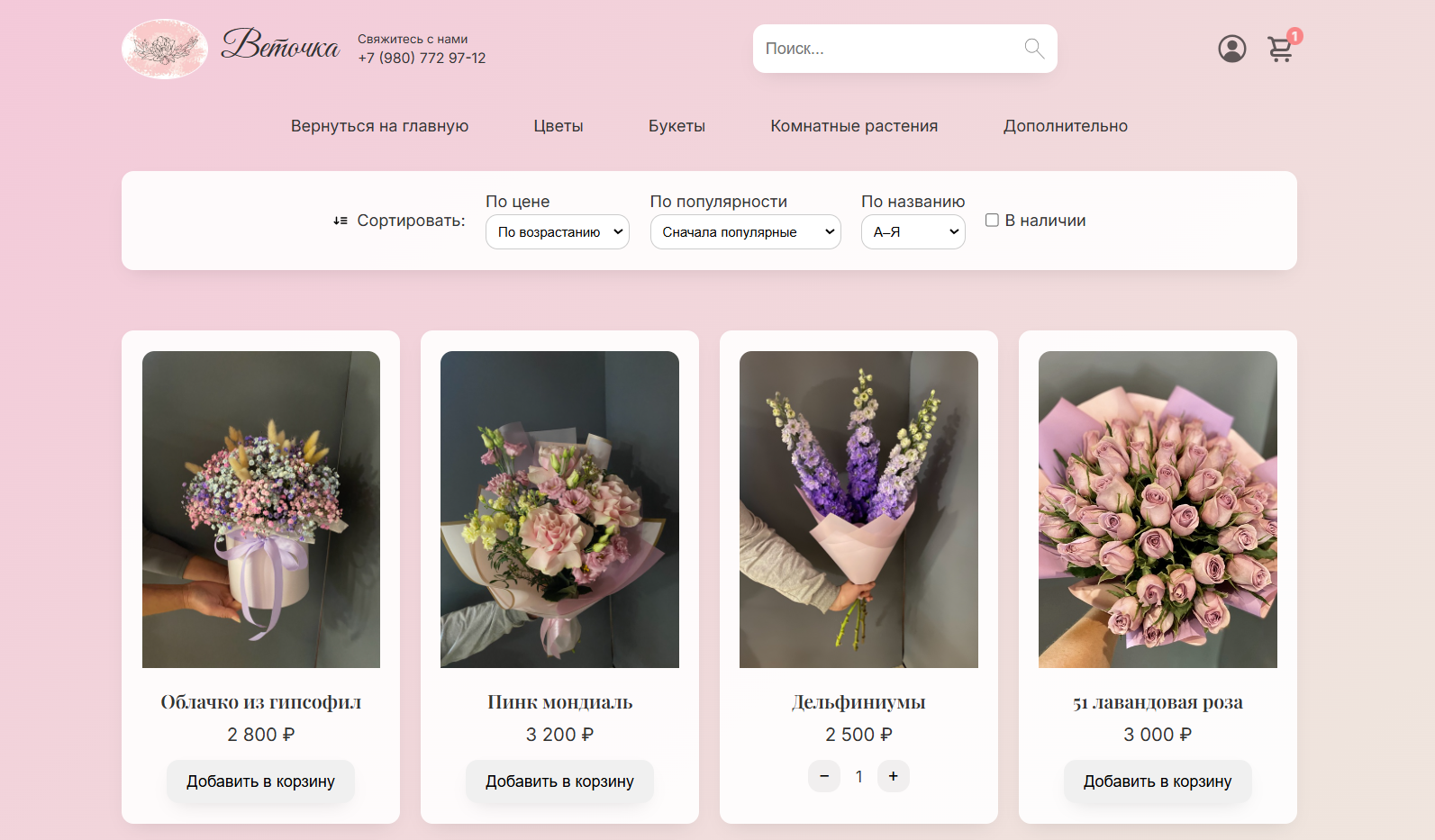


Рисунок №16 - Страница каталога после выбора товара

Далее пользователь может выбрать другие товары, например, он может создать собственный букет. Для этого ему нужно нажать на соответствующую кнопку на главной странице. Раздел с созданием собственного букета показан на рисунке №17.

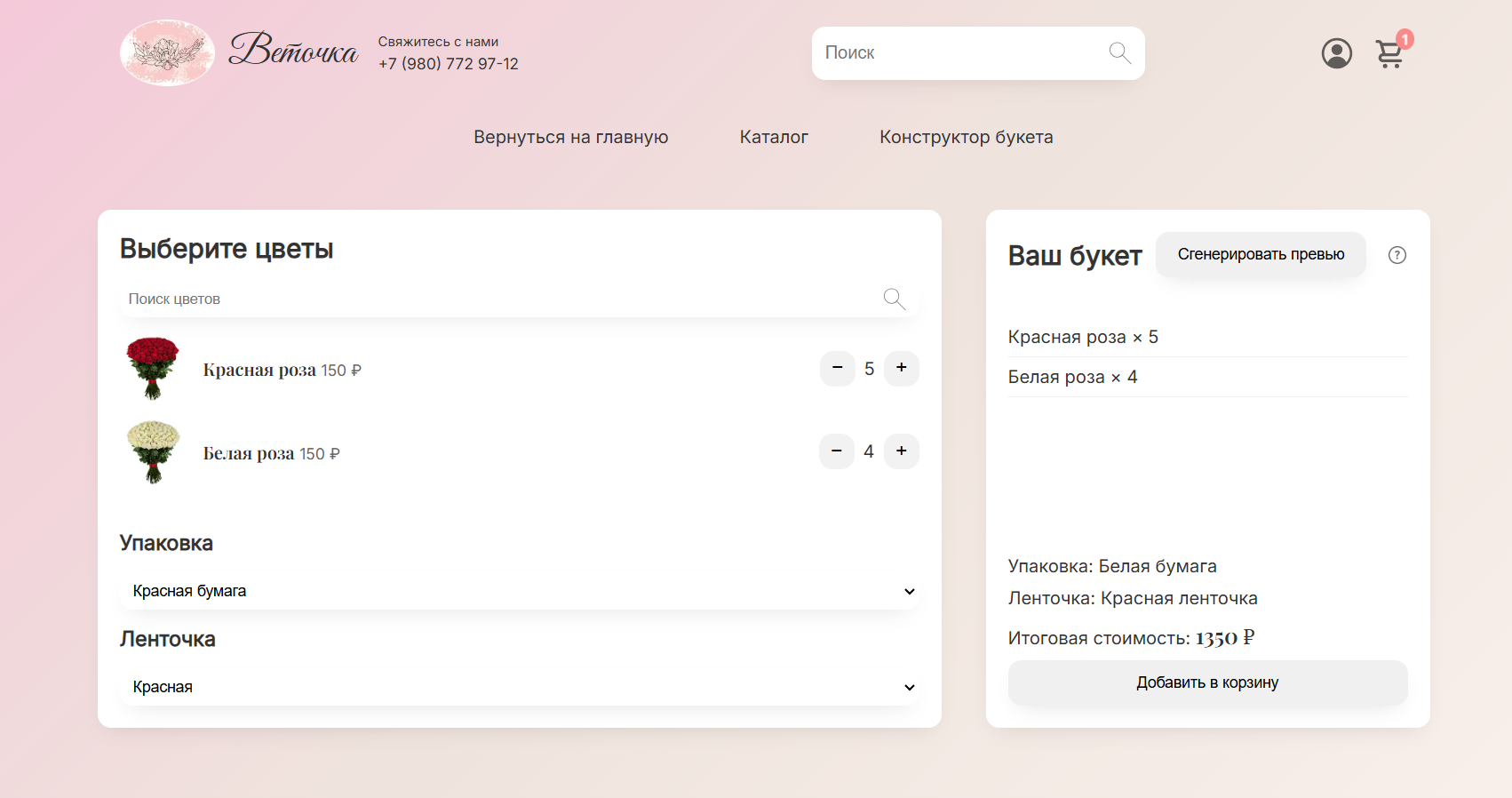
На нем предлагается выбрать цветы, упаковку и ленту. После чего в панели справа отобразится состав и итоговая стоимость букета. Отличительной функцией на данной панели является возможность сгенерировать предположительное изображение результата сборки. Для этого необходимо нажать на кнопку «Сгенерировать». 

Рисунок №17 - Создание собственного букета

В новом модальном окне после ожидания генерации (Рисунок №18) можно будет увидеть картинку, сгенерированную нейросетью (Рисунок №19).



Рисунок №18 - Ожидание загрузки фото

Поле того как пользователь был удовлетворен своим собственным букетом, он нажимает кнопку добавить в корзину, где появляется новая позиция.

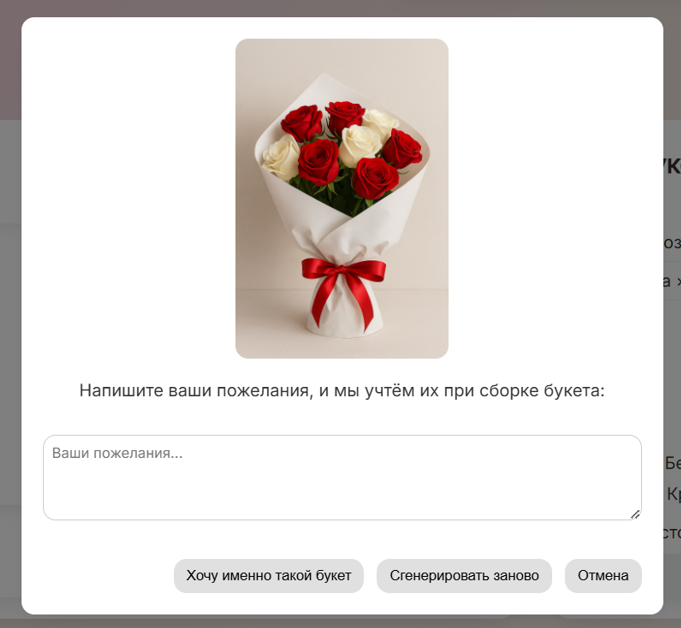


Рисунок №19 - Окно с картинкой букета

На следующем шаге пользователь переходит в корзину, в которой представлены все позиции текущего заказа, их количество, стоимость, а также итоговая сумма (Рисунок №20).

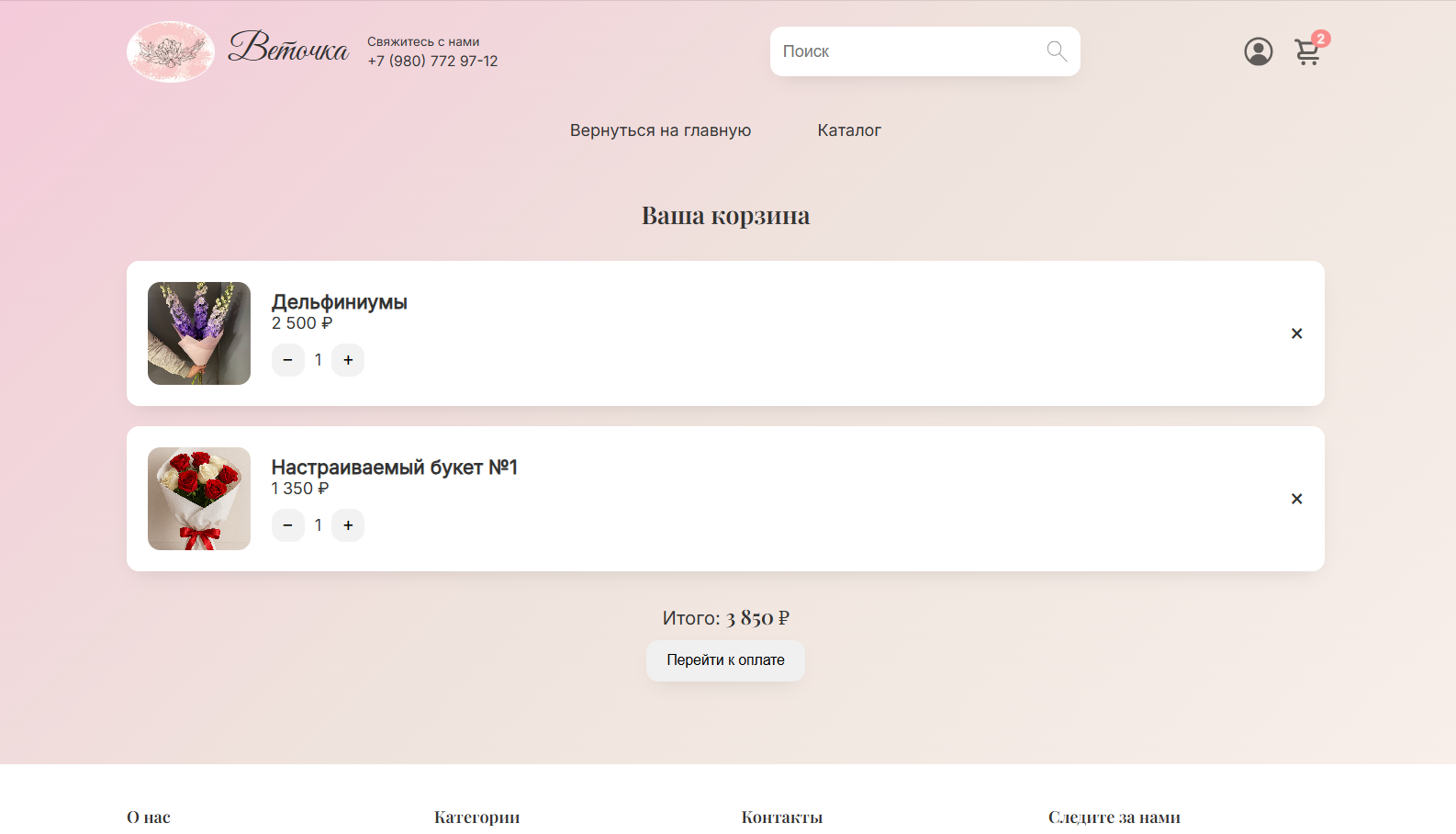


Рисунок №20 - Состав корзины

Дальше пользователь проверяет состав своего заказа и нажимает кнопку «Перейти к оплате». В зависимости от того, авторизован пользователь или нет, он переходит сразу к оформлению заказа или попадает на страницу входа.

На странице входа (Рисунок №21) пользователю предлагается ввести учетные данные либо нажать кнопку «Регистрация» для перехода на соответствующую страницу создания аккаунта (Рисунок №22).

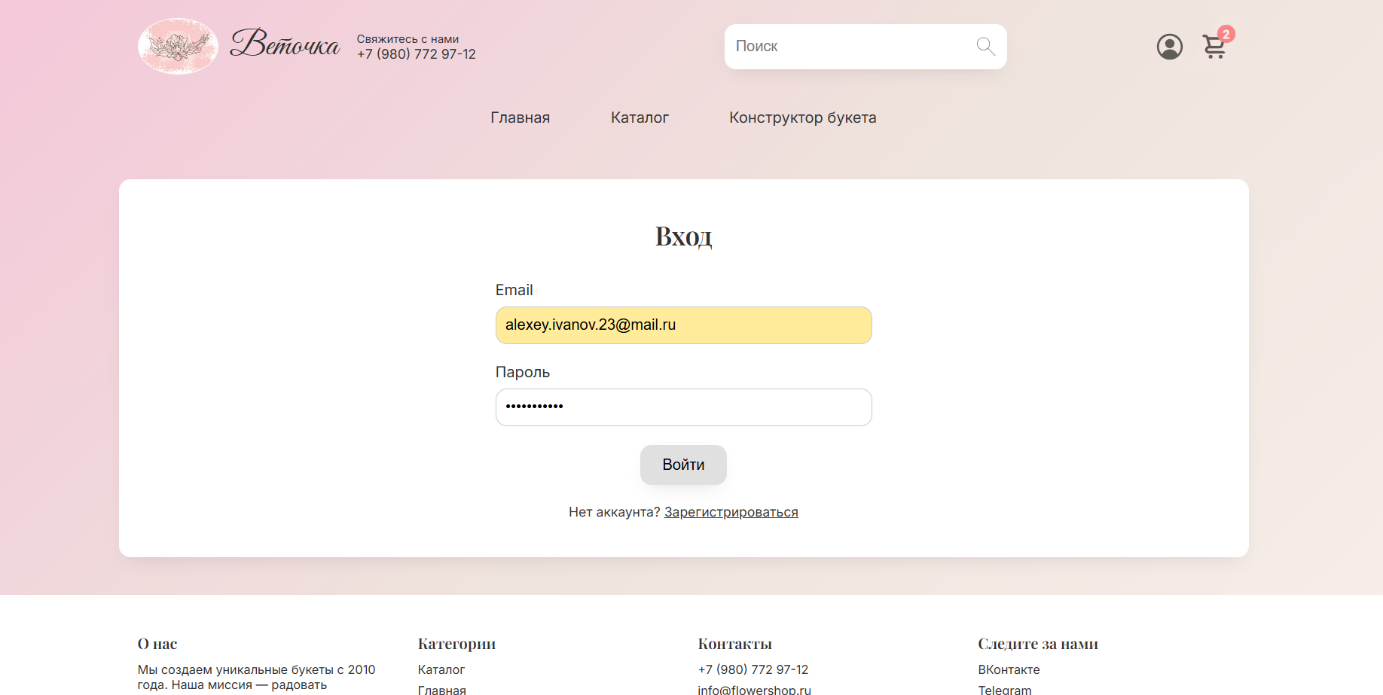


Рисунок №21 - Страница входа

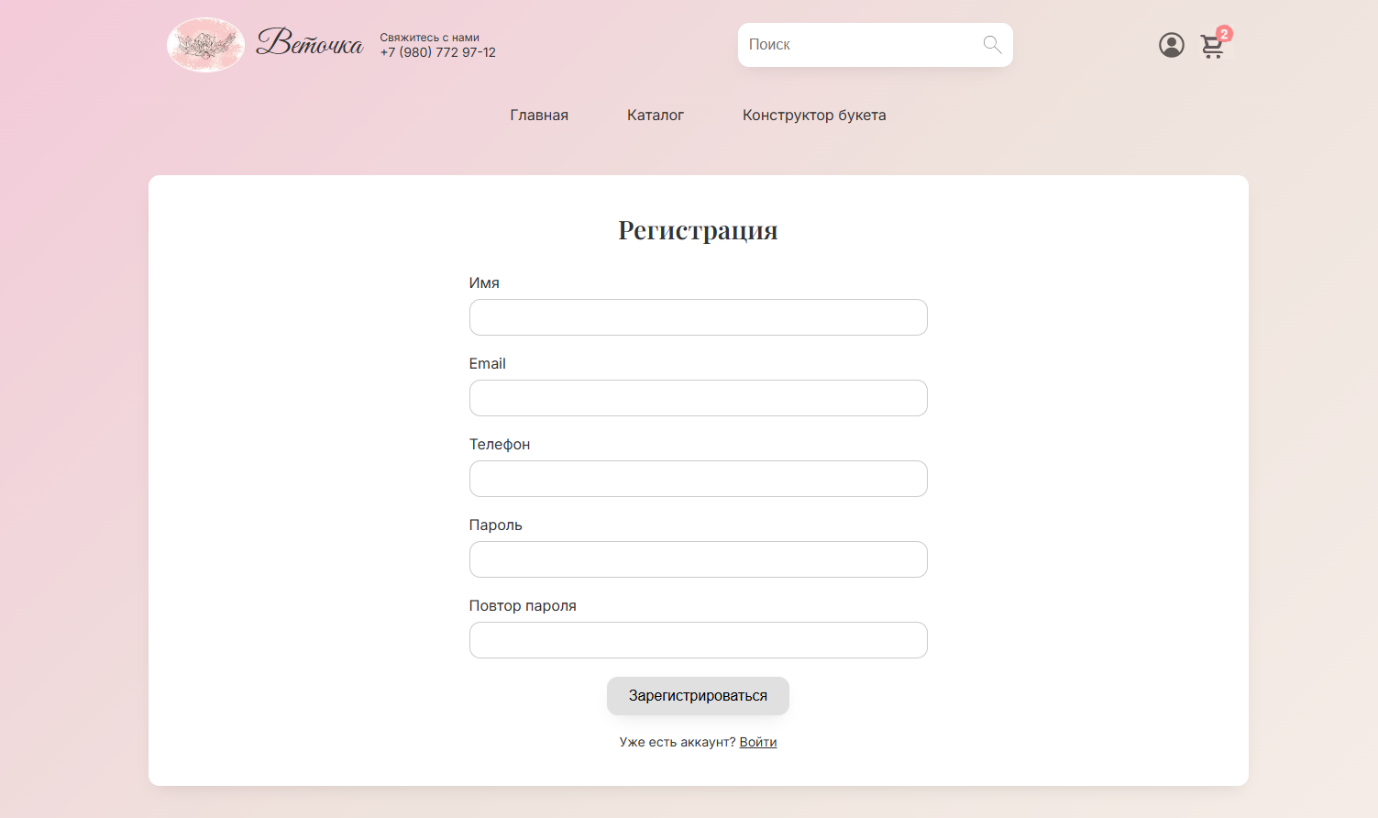


Рисунок №22 - Страница регистрации

После авторизации клиента переадресует на страницу, где он находился до авторизации, с соответствующим уведомлением, в данном сценарии в корзину.

Для завершения заказа пользователь переходит на страницу оформления (Рисунок №23). Здесь ему необходимо ввести данные необходимые для доставки и нажать кнопку «Подтвердить заказ»



Рисунок №23 - Страница оформления

После оформления заказа, пользователя переадресует на страницу профиля, где он может посмотреть статус заказа (Рисунок №24).

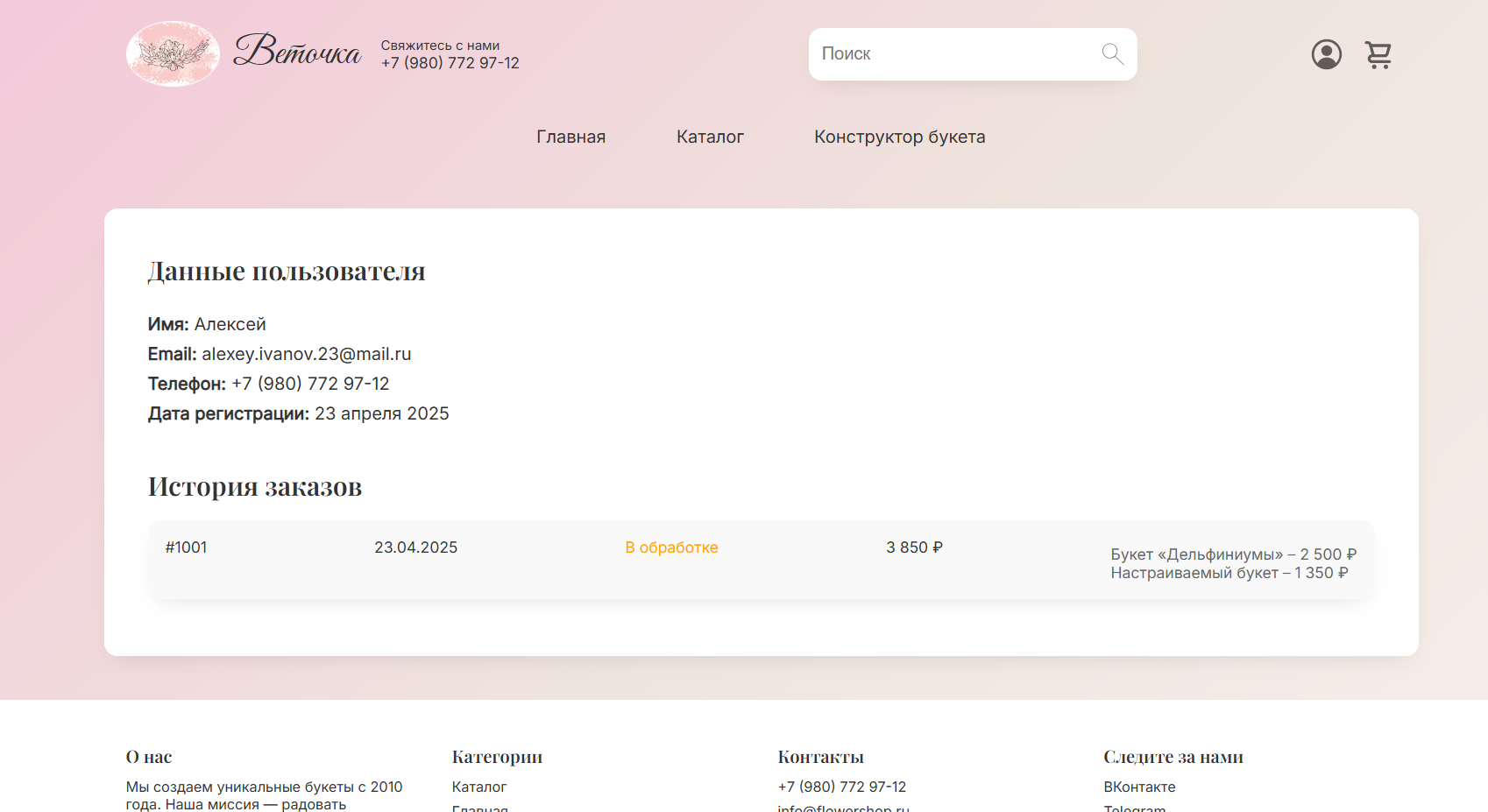


Рисунок №24 - Профиль пользователя

3.2.2 Описание работы панели управления

При открытии приложения сотрудника встречает экран входа (Рисунок №25), где он может ввести свои данные для получения доступа.

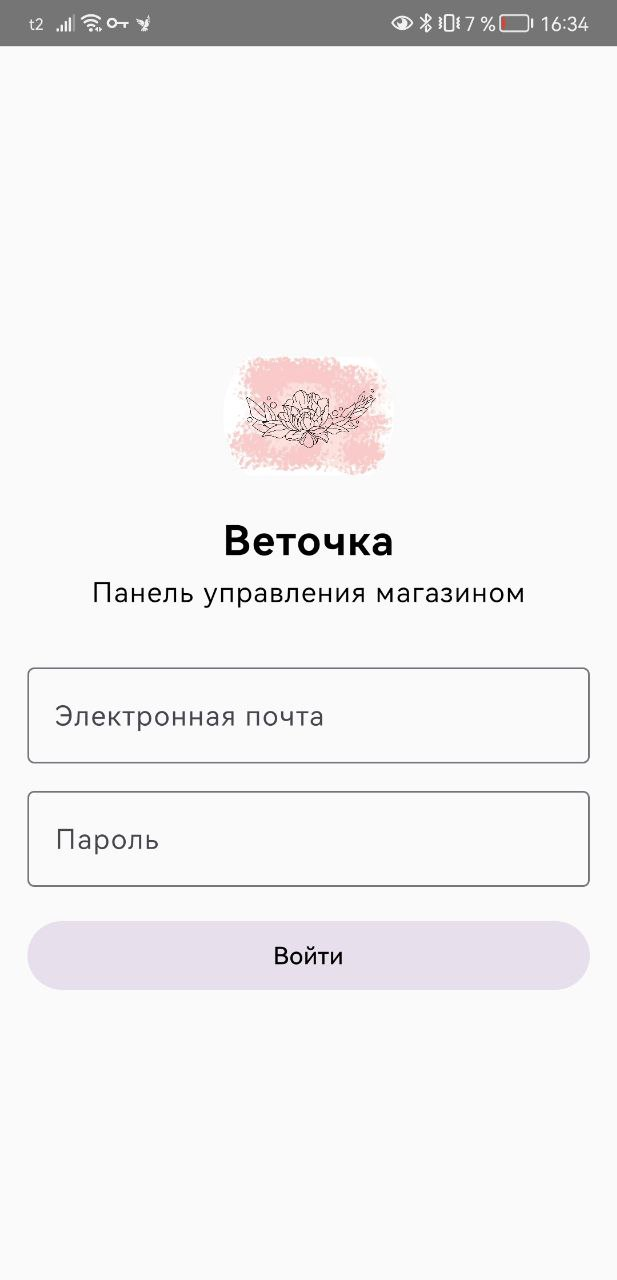


Рисунок №25 - Экран входа в админ панель

После успешной авторизации, сотрудник переходит на экран с активными заказами (Рисунок №26). На данном экране представлены заказы с краткой информацией о них. Новый заказ от клиента помечается статусом «В обработке». Чтобы принять заказ, нужно изменить статус, нажав на панель с необходимым заказом. После нажатия откроется экран с детальным описанием заказа (Рисунок №26). Здесь указана такая информация как состав заказа, итоговая сумма, адрес доставки, дата и время доставки, комментарий к заказу и статус. Статус отображается в виде выпадающего списка, нажав на который можно выбрать новый. Так же на этой панели можно посмотреть описание товара, для его сборки.

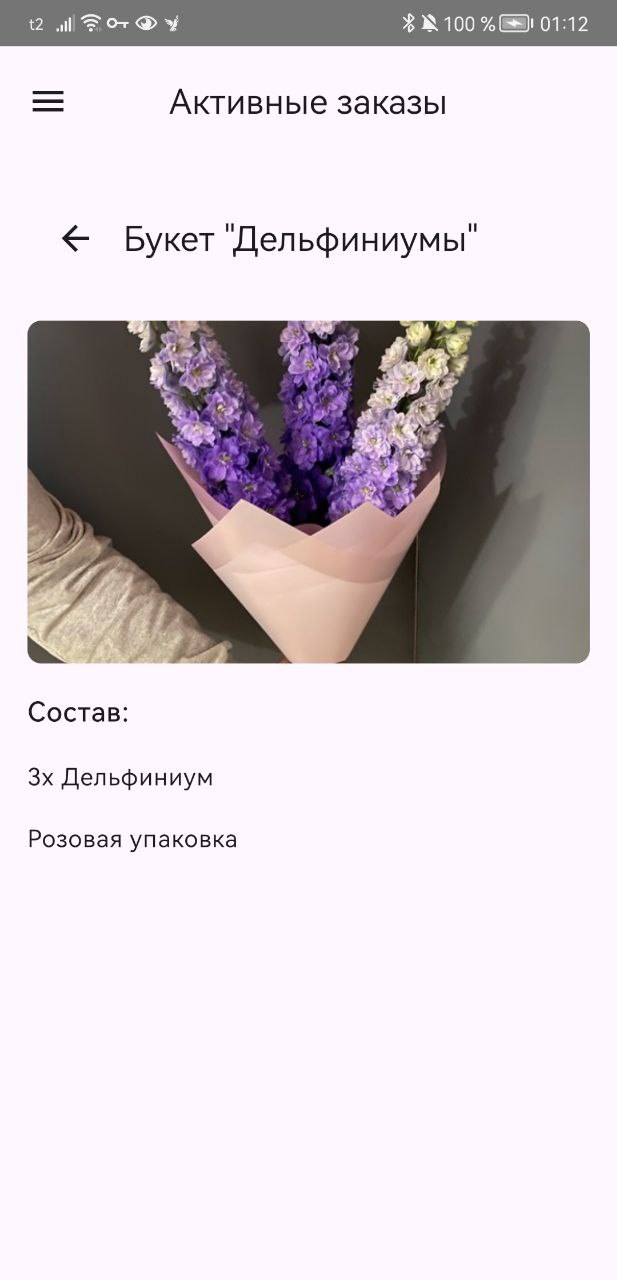
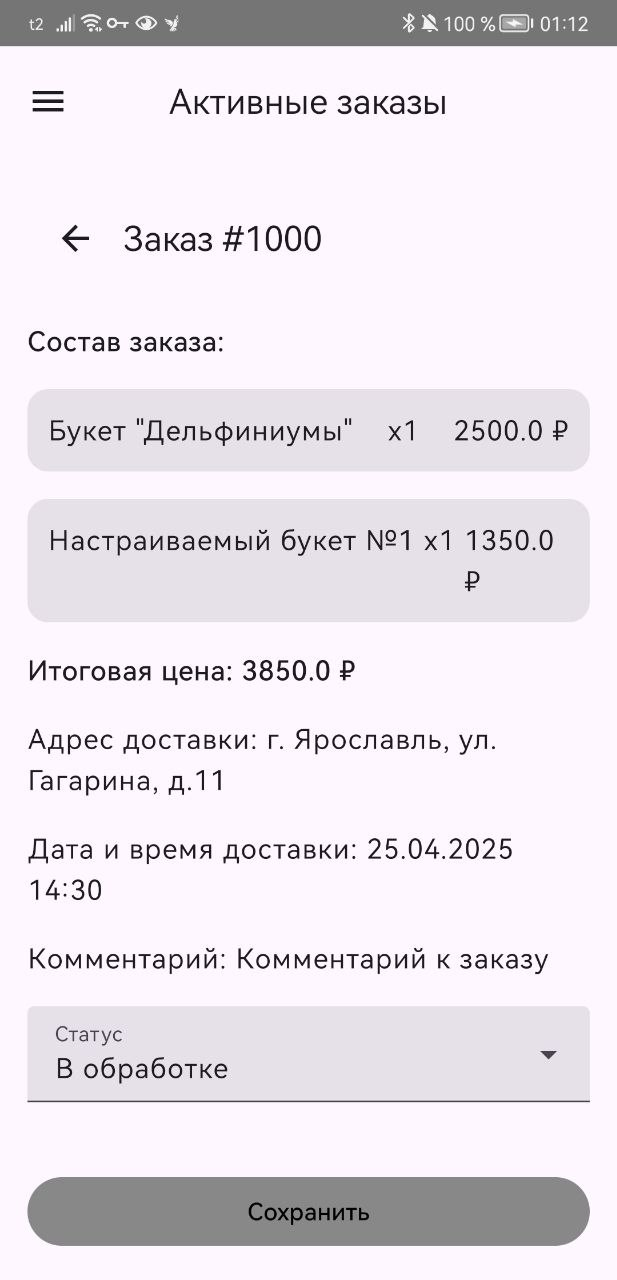
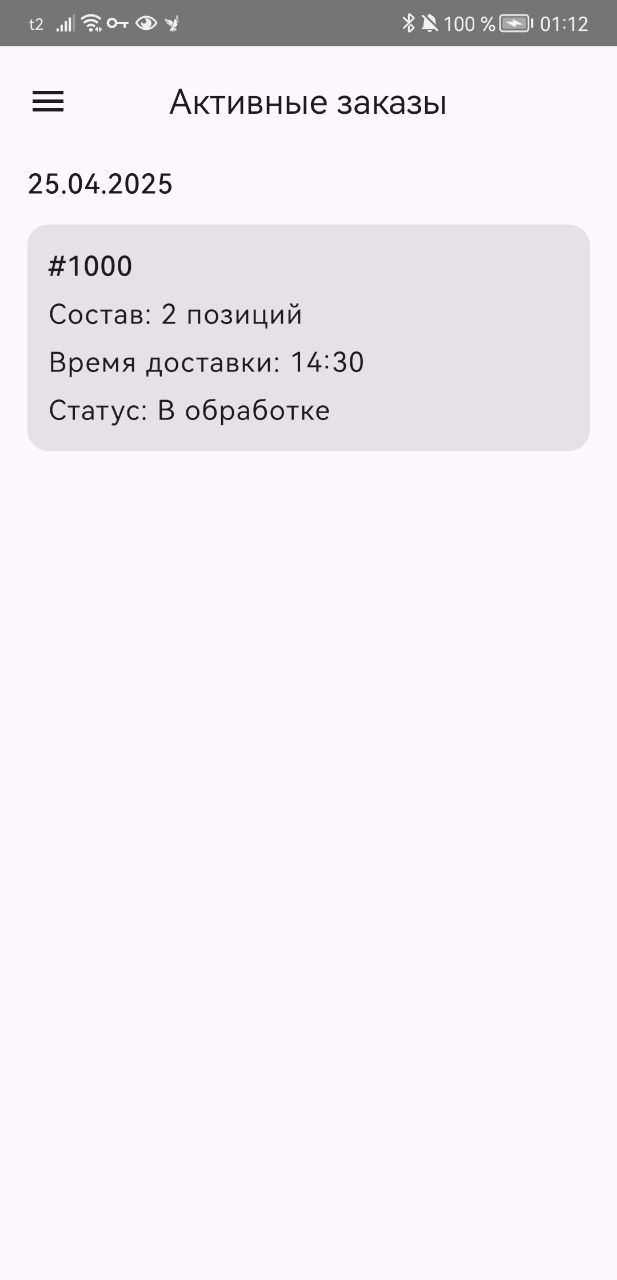


Рисунок №26 - Экран активных заказов и детального описания заказа

После просмотра заказа, любой сотрудник может поменять статус на «Принят» для подтверждения оформления заказа. Далее флорист, меняет статус на «Собирается», собирает все необходимые позиции и переводит статус в «Передан в доставку». Курьер, зайдя в приложение увидит активные заказы, которые переданы в доставку, возьмет необходимые товары и поедет на адрес, который указан в детальном описании товара. В случае отключения мобильного интернета, информация по заказу будет получаться из локального хранилища устройства, а на экране будет выведено соответствующее уведомление (Рисунок №27). После фактической доставки и оплаты, курьер переведёт статус заказа в «Завершён» и заказ можно будет считать выполненным.

Помимо работы с активными заказами, в панели управления имеются функции управления магазином, показа статистики, просмотр профиля сотрудника.

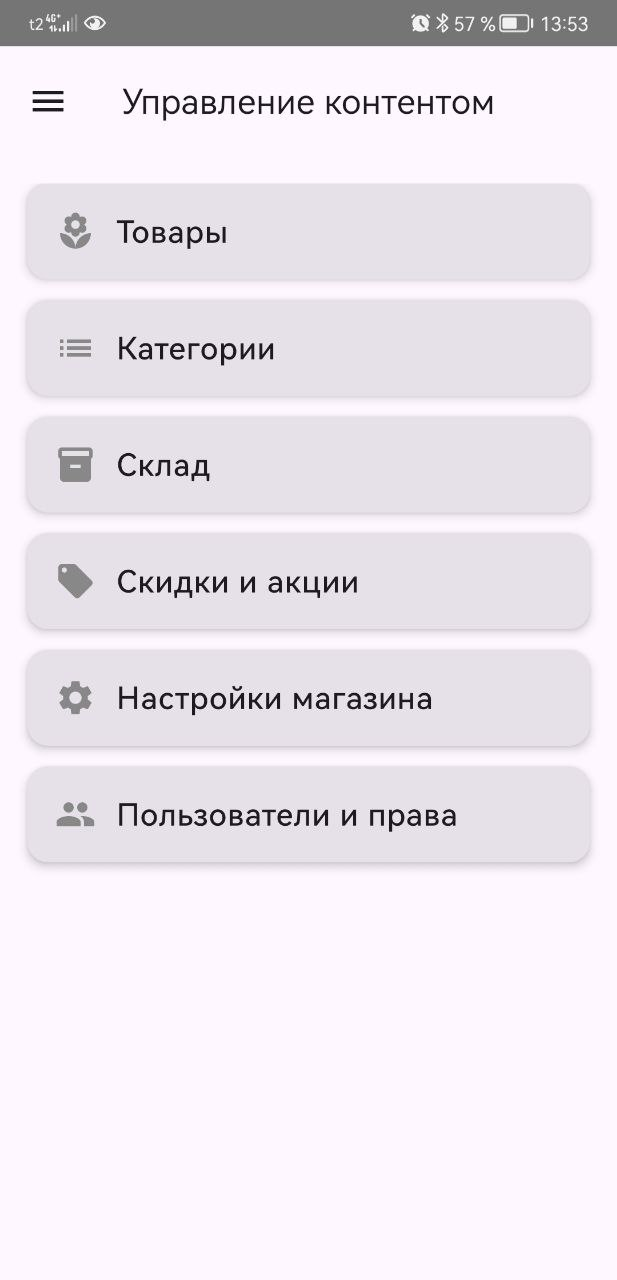
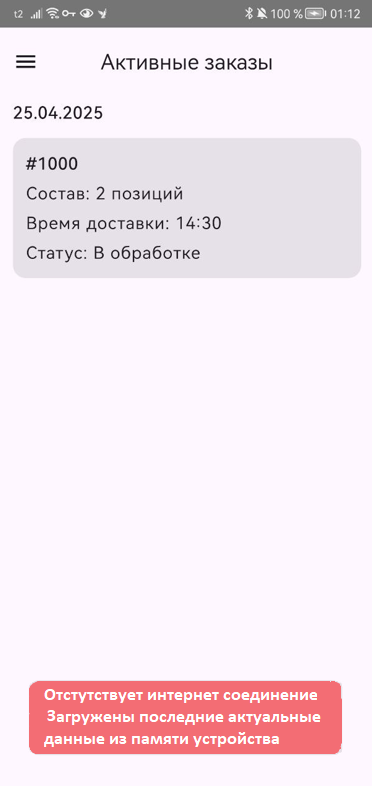


Рисунок 27 – Экраны с ошибкой и категориями управления магазином

Выбрать соответствующие разделы можно в боковом меню приложения (Рисунок №27). Панель управления магазином имеет в свою очередь несколько дочерних панелей, предназначенных для управления определённой сферой деятельности магазина (Рисунок №27). Раздел «Товары» предназначен для формирования витрины: здесь администратор добавляет или удаляет позиции, загружает фотографии, заполняет название, описание, цену и при необходимости скрывает товар. Изменения мгновенно появляются во внутреннем приложении и видны флористам для работы с заказами. Раздел «Склад» отвечает за фактические остатки. Сюда заносят приход от поставщиков и списывают цветы, ушедшие в работу или брак. Хранятся история движений и минимальные остатки. В «Настройках магазина» собраны общие данные: контактные телефоны, e-mail и соцсети, адрес магазина и часы работы, а также настройки доступа к генерации предпросмотра настраиваемого букета. Все изменения сразу применяются в клиентских интерфейсах. «Пользователи и права» позволяет управлять аккаунтами сотрудников: создавать флористов и курьеров, назначать им роли и уровни доступа.

В разделе аналитики и отчетов можно посмотреть статистику выручки за сегодня, неделю или месяц, а также линейный график показывающий, как выручка менялась по дням за последний месяц. На таком графике сразу видно растёт прибыль, падаем или стоит на месте. В профиле можно посмотреть актуальную информацию о сотруднике.

### 3.3 Вывод по технологической части

# Экономическая часть

## 4.1 Решаемая проблема и цель проекта

### 4.1.1 Проблема

Цветочный магазин «Веточка» в Ярославле сегодня сталкивается с рядом технологических ограничений, которые негативно влияют на обслуживание клиентов и эффективность работы.

Во-первых, вся онлайн-деятельность ведётся через группу во «ВКонтакте», без собственного веб-сайта, где покупатель мог бы выбрать букет, оплатить его и отследить доставку.

Во-вторых, учёт товаров и заказов ведётся вручную, что неизбежно приводит к ошибкам при списании просроченных цветов и потере информации о текущих остатках.

В-третьих, отсутствие мобильного приложения с офлайн-режимом лишает сотрудников возможности получать данные о заказах и статусе доставки без постоянного интернета.

В-четвертых, при желании клиента собрать свой букет, у него нет визуального представления о том, что он получит, что ведёт к негативным отзывам при несоответствии ожиданий и конечного продукта.

На сегодняшний момент существуют технологии, которые позволяют решить данные проблемы, поэтому требуется разработка и внедрения комплексной системы: веб-интерфейса, мобильного приложения и модуля AI-визуализации букетов.

### 4.1.2 Цель проекта

Целью проекта является разработка и внедрение единой информационной системы, которая позволит устранить существующие технологические ограничения магазина «Веточка» и обеспечить качественно новый уровень обслуживания клиентов.

В результате реализации проекта предполагается получить:

- Цель проекта должна быть реализована в течение 12 месяцев с начала разработки проекта.

- Веб-интерфейс для покупателей, доступный круглосуточно, позволяющий выбирать букеты, оплачивать их платёжными картами и электронными кошельками, отслеживать статус заказа в режиме реального времени и оставлять отзывы.

- Мобильное приложение для сотрудников с офлайн-режимом хранения данных, которое обеспечивает получение и подтверждение заказов без постоянного интернет-соединения (автономно хранить до 100 активных заказов на устройстве) и автоматическую синхронизацию с сервером системы при восстановлении связи.

- Модуль AI-визуализации (Kandinsky), интегрированный в веб интерфейс, позволяющий в режиме реального времени генерировать превью клиентского «конструируемого» букета с точностью не менее 85 % к финальному варианту. Генерация изображения должна занимать не более 3 секунд при стандартных параметрах (разрешение 512 × 512).

### 4.1.3 Оценка конкурентоспособности в сравнении с аналогами

Ниже приведён сводный расчёт конкурентоспособности с использованием индексного метода. В таблице указаны пять ключевых показателей качества, их весовые коэффициенты B, оценки X для каждого продукта и произведения B × X. В конце приведены обобщённые показатели качества J по каждому из конкурентов и для проекта Vetochka.

Таблица №2 - Расчет показателя качества балльно-индексным методом

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Показатели качества | B | Flowwow | | Flor2U | | FloraExpress | | СоюзЦветТорг | | Vetochka | |
| X | B × X | X | B × X | X | B × X | X | B × X | X | B × X |
| 1. Веб-интерфейс | 0,25 | 4 | 1,00 | 4 | 1,00 | 3 | 0,75 | 4 | 1,00 | 5 | 1,25 |
| 2. Мобильное приложение | 0,20 | 4 | 0,80 | 4 | 0,80 | 3 | 0,60 | 3 | 0,60 | 5 | 1,00 |
| 3. Офлайн-режим | 0,20 | 1 | 0,20 | 1 | 0,20 | 1 | 0,20 | 1 | 0,20 | 5 | 1,00 |
| 4. AI-визуализация | 0,15 | 1 | 0,15 | 1 | 0,15 | 1 | 0,15 | 2 | 0,30 | 5 | 0,75 |
| 5. Админ-панель | 0,20 | 4 | 0,80 | 3 | 0,60 | 2 | 0,40 | 3 | 0,60 | 5 | 1,00 |
| **Обобщённый показатель качества J** |  |  | **2,95** |  | **2,75** |  | **2,10** |  | **2,70** |  | **5,00** |

По итогам расчёта проект **Vetochka** получает максимальный результат (J = 5,00), тогда как ближайшие конкуренты достигают лишь J = 2,95 (Flowwow) и ниже. Это подтверждает, что по выбранным критериям (удобный веб-интерфейс, мобильное приложение с офлайн-режимом, AI-визуализация и админ-панель) Vetochka обладает наивысшим обобщённым показателем качества и, следовательно, обладает наилучшими возможностями для успешной конкуренции на рынке.

## 4.2 Заинтересованные стороны проекта

Реестр заинтересованных сторон отражает все группы, которые участвуют в проекте или могут повлиять на его успех. Ниже приведена таблица с объединёнными столбцами «Цели и интересы сторон, степень их проявления» и «Степень влияния/Вероятные риски от стороны».

Таблица №3 - Реестр заинтересованных лиц проекта

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Описание стороны** | **Цели и интересы сторон, степень их проявления** | **Степень влияния / Вероятные риски от стороны** |
| **Владелец магазина** | Собственник «Веточки», отвечающий за стратегию развития, вложение средств и контроль над всеми бизнес-процессами. | Желаемый результат – увеличение выручки за счет роста онлайн-продаж и минимизации потерь от просроченных цветов. Проявляется очень сильно: все ключевые решения о бюджете и функциях системы принимаются владельцем. | Высокая степень влияния: он распоряжается ресурсами и утверждает конечный функционал. Риски – в случае задержки финансирования, требования изменений функций в процессе разработки или отказ от инноваций без жёстких гарантий рентабельности могут парализовать проект. |
| **Клиенты (покупатели)** | Посетители веб-сайта и пользователи мобильных приложений, оформляющие заказы, ожидающие удобного и надёжного сервиса. | Основной интерес – получить понятный и удобный интерфейс: выбрать букет, оплатить, отследить доставку, увидеть превью в конструкторе. Проявление высокое, поскольку именно клиенты формируют выручку магазина. | Средняя степень влияния: они непосредственно не управляют разработкой, но негативные отзывы и снижение лояльности при неудобствах способны существенно ухудшить репутацию и показатели продаж. |
| **Флористы (персонал)** | Сотрудники, собирающие букеты и поддерживающие ассортимент; через мобильное приложение принимают и подтверждают заказы. | Заинтересованы в быстром приёме заявок, удобном отображении состава букета и точных данных об остатках цветов. Проявление высокое, так как их эффективность напрямую влияет на качество исполнения заказов. | Высокая степень влияния: ошибки или отказ использовать мобильное приложение могут приводить к задержкам сборки. Если интерфейс сложен или синхронизация ненадёжна, могут возникнуть сбои в работе, снижение скорости обслуживания и увеличение брака. |
| **Курьеры (доставка)** | Сотрудники, доставляющие готовые букеты, используют мобильное приложение для получения информации о маршруте и статусе заказа. | Интерес – получить точные данные о заказе (адрес, контакт), работать в офлайн-режиме при слабом интернете и быстро подтверждать доставку. Проявление среднее, так как курьеры влияют на конечный пользовательский опыт, но не на стратегию проекта. | Средняя степень влияния: сбои в приложении у курьеров приводят к задержкам доставки, что негативно влияет на репутацию. Риски – потеря офлайн-данных, дублирование статусов, снижение удовлетворённости клиентов из-за ошибок курьеров. |
| **Администраторы** | Менеджеры, отвечающие за управление каталогом, контроль остатков, обработку рекламаций, взаимодействие с платежными системами. | Заинтересованы в удобной админ-панели, позволяющей изменять ассортимент, отслеживать заказы, запускать акции и контролировать списания просроченных товаров. Проявляется высоко: от их работы зависит оперативность решения внутренних задач. | Высокая степень влияния: некорректная работа или ошибки администраторов ведут к неточным остаткам, финансовым потерям и срывам выполнения заказов. Риски – сопротивление покупке новых решений, требующих изменения привычных процедур, что может затормозить внедрение. |
| **Поставщики цветов** | Оптовые поставщики свежих цветов и сопутствующих товаров; от их работы зависит наличие ассортимента. | Цель – своевременная поставка качественных сортов, минимизация возвратов из-за просрочки, прогнозирование потребностей. Проявление среднее: поставщики не участвуют в разработке, но влияют на доступность товаров в приложениях. | Низкая степень влияния: задержки поставок или некорректные прогнозы ведут к дефициту товаров и невозможности выполнить заказы. Риски – отсутствие интеграции API для автоматического обновления данных о поступлениях, что усложняет синхронизацию информации об остатках. |
| **Провайдер AI-сервиса** | Внешний сервис Kandinsky, предоставляющий API для генерации изображений «виртуальных» букетов на основе параметров. | Цель – обеспечить высокое качество и скорость генерации превью (≥ 85 % точности, ≤ 3 с на создание 512×512 px). Проявление среднее: животворность проекта зависит от стабильности их API, но они не участвуют в деталях бизнес-логики. | Средняя степень влияния: сбои в работе API Kandinsky или ухудшение качества генерации приведут к неудовлетворённости клиентов. Риски – изменение тарифов, увеличение стоимости запросов, появление ограничений на количество запросов, ухудшение релевантности изображений. |
| **Платежные системы** | Банковские шлюзы и электронные кошельки, через которые проходят транзакции по оплате заказов. | Заинтересованы – в соблюдении стандартов безопасности (PCI DSS), минимизации отказов при совершении платежей (≤ 2 %). Проявление низкое: их задача в целом корпоративна, но качество интеграции влияет на удовлетворённость покупателей. | Средняя степень влияния: сбои в платежных системах приведут к отказу клиентов от оплаты и росту корзин-брошенок. Риски – необходимость доработки по изменению API, рост комиссии, некорректная интеграция может привести к сбоям при оплате. |
| **IT-команда** | Разработчики и тестировщики, создающие код веб- и мобильных приложений, отвечающие за создание архитектуры и контроль качества. | Цель – реализовать функционал в соответствии с ТЗ, соблюдать сроки и стандарты code review, обеспечить покрытие тестами. Проявление высокое: именно от их производительности зависят сроки выпуска и стабильность кода. | Высокая степень влияния: нарушение сроков, низкое качество кода или недостаточное тестирование приведут к отлагательным исправлениям, сбоям при демонстрации, необходимости переработок и срывам плана. Риски – текучка кадров, неправильное понимание требований, недостаток ресурсов и опыта. |

В этой таблице объединены цели и интересы с уровнем их проявления в одной колонке, а степень влияния и вероятные риски — в другой, что позволяет наглядно увидеть, насколько каждая сторона вовлечена в проект и какие последствия возможны при негативном сценарии их участия. На основе этих данных была составлена детализированная таблица (Таблица №4).

Таблица №4 - Определение степени влияния-интереса

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Стейкхолдер** | **Степень влияния** | **Интерес** |
| 1 | Владелец магазина | 3 | 3 |
| 2 | Клиенты (покупатели) | 2 | 3 |
| 3 | Флористы (персонал) | 3 | 3 |
| 4 | Курьеры (доставка) | 2 | 2 |
| 5 | Администраторы (менеджеры) | 3 | 3 |
| 6 | Поставщики цветов | 1 | 2 |
| 7 | Провайдер AI-сервиса | 2 | 2 |
| 8 | Платежные системы | 2 | 1 |
| 9 | IT-команда разработчиков | 3 | 3 |

Далее была составлена матрица степени влияния-интереса (Рисунок №28)

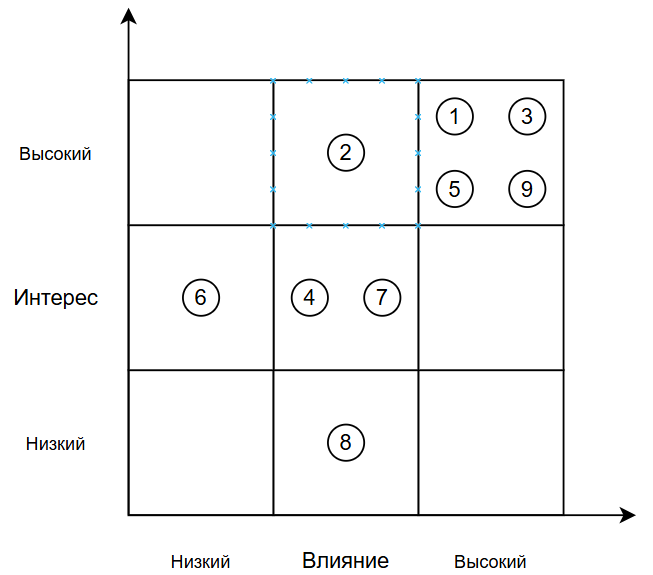


Рисунок №28 - Матрица влияния-интереса

## 4.3 Методология управления проектом

Методология управления проектом — это совокупность обоснованных принципов, методов и процессов, предназначенных для организации, планирования и контроля всех аспектов проекта с целью достижения его поставленных целей в установленные сроки и в рамках бюджета. Она включает в себя структурированный подход к управлению проектами, охватывающий различные этапы жизненного цикла проекта: от инициации и планирования до исполнения, мониторинга и завершения. Методология управления проектом должна обеспечивать эффективное распределение ресурсов, управление рисками, коммуникацию между всеми заинтересованными сторонами и контроль качества выполнения задач.

### 4.3.1 Существующие методологии

Для успешной реализации проектов в сфере разработки программного обеспечения необходимо учитывать не только методологию управления проектами, но и действующие стандарты, которые регламентируют процессы создания ПО. Эти стандарты формируют основу для структурирования и упорядочивания всех этапов жизненного цикла программного обеспечения — от первоначального планирования до окончательной эксплуатации и дальнейшего сопровождения.

- ГОСТ Р ИСО/МЭК 12207-2010 — стандарт, который определяет процессы жизненного цикла программных средств, включая этапы планирования, разработки, эксплуатации и сопровождения

- ГОСТ Р 57193-2016 — охватывающий жизненный цикл систем, который также включает в себя аппаратно-программные комплексы.

- ГОСТ Р 59793-2021 — регламентирует стадии создания автоматизированных систем, обеспечивая единые подходы к разработке.

- ГОСТ Р 53622-2009 — устанавливающий требования к информационно-вычислительным системам, что позволяет развивать гарантии качества и надежности.

Эти стандарты не только обеспечивают соблюдение необходимых технических и методологических требований, но и способствуют повышению качества разрабатываемого ПО, что особенно важно для реализации проектов для государственных и корпоративных заказчиков.

## 4.3.2 Выбор подходящей методологии для проекта

Для разработки системы «Веточка» наиболее подходящей является гибридная методология, основанная на принципах Agile (Scrum).

Во-первых, проект включает несколько взаимозависимых компонентов — веб-интерфейс, мобильное приложение с офлайн-режимом и AI-модуль визуализации — и требует быстрой реакции на изменения требований. Scrum позволяет разбивать работу на короткие спринты (2–3 недели), по окончании каждого из которых демонстрируется работоспособная часть функционала (например, готовый каталог товаров, работающая синхронизация в мобильном приложении или базовый AI-рендер). Это даёт гибкость: при необходимости можно оперативно скорректировать приоритеты или внести уточнения, не дожидаясь завершения всего проекта.

Во-вторых, у проекта есть ключевые заинтересованные стороны (владелец магазина, флористы, курьеры, поставщики, IT-команда, провайдер AI-сервиса), и для них важно видеть промежуточные результаты. В рамках Scrum назначается Product Owner, который координирует взаимодействие с владельцем магазина и конечными пользователями. Регулярные демонстрации (sprint review) позволяют всем участникам оценивать качество реализации, уточнять детали конструирования букетов и работу офлайн-режима, благодаря чему снижается риск «не того» решения, которое будет трудно исправить на финальной стадии.

Третьим аргументом является объём команды — в проекте намерены участвовать 3–5 специалистов, поэтому избыточная документация и формальные процессы пожизненного цикла (например, полное соответствие ISO/IEC 12207) создадут лишнюю административную нагрузку и замедлят разработку. Scrum задаёт минимальный набор ритуалов (планирование спринта, ежедневный стендап, демонстрация и ретроспектива), что упрощает коммуникацию, позволяет быстро выявлять и устранять узкие места и концентрироваться на создании работающего кода, а не на бюрократии.

Наконец, в гибкой методологии легко заложить автоматизированное тестирование (юнит-тесты REST-API, интеграционные тесты мобильного приложения и проверки AI-модуля), что обеспечивает уверенность в качестве на каждом шаге. В результате Scrum обеспечивает предсказуемую скорость внедрения функционала, прозрачность для всех участников и возможность своевременно адаптироваться к новым требованиям, что идеально подходит для проекта «Веточка».

## 4.4 Структурная декомпозиция работ

### 4.4.1 Жизненный цикл проекта

На представленной схеме (Рисунок №29) жизненный цикл проекта «Веточка» показан в виде сочетания единовременных (каскадных) этапов и повторяющегося (итеративного) цикла спринтов.

Весь процесс разработки можно разбить на следующие основные фазы:

а) Инициация и общее планирование

Эта стартовая фаза начинается сразу после того, как проект формально одобрен и выделен бюджет. На данном этапе собираются ключевые бизнес-требования, превращаются в первоначальное техническое задание и формируется общий список задач (Product Backlog). Здесь же определяются заинтересованные стороны, назначаются ответственные роли (владелец проекта, Product Owner, Scrum Master, команда разработчиков) и уточняются сроки первого цикла работы. Результатом этой фазы является согласованный план итераций, подготовленные черновые макеты интерфейсов и базовые архитектурные наработки.

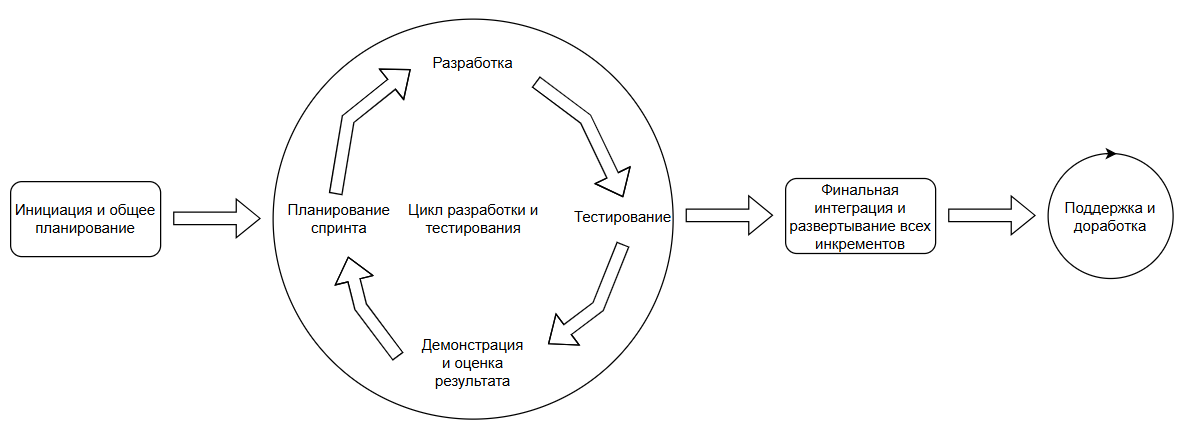


Рисунок №29 - Жизненный цикл продукта и проекта

б) Итеративный цикл разработки и тестирования (спринты)

В центре схемы находится большой круг, иллюстрирующий повторяющийся процесс спринта. Каждый спринт включает четыре ключевых шага:

- Планирование спринта (Sprint Planning). На этом этапе из общего списка Product Backlog выбираются конкретные задачи, которые команда будет реализовывать в течение ближайших 2-3 недель. Для каждого пункта определены требования, критерии приёмки и прогнозируемый объём работы (story points).

- Разработка (Development). После окончания планирования команда приступает к написанию кода: создаются REST-эндпоинты на сервере (Spring Boot), разрабатываются компоненты веб-приложения на React, строятся экраны мобильного приложения на Kotlin и формируются запросы к API Kandinsky для визуализации букетов. В ходе спринта разработчики тесно взаимодействуют, регулярно обмениваются статусами и корректируют подход по мере необходимости.

- Тестирование (Testing). Параллельно с основными задачами команда проводит модульные и интеграционные тесты. Это позволяет сразу же выявлять ошибки в контроллерах, бизнес-логике и интеграции с внешними сервисами. Например, проверяется, что мобильное приложение действительно хранит до 100 активных заказов в офлайн-режиме и верно синхронизируется при появлении сети, а AI-модуль Kandinsky успевает выдавать превью букета за не более чем 3 секунды.

- Демонстрация и оценка результата (Sprint Review). В конце спринта готовый инкремент (например, новый функционал каталога, готовый экран мобильного приложения или вариант AI-визуализации) показывается владельцу магазина и другим заинтересованным лицам. Здесь же собирается обратная связь: если что-то оказалось не совсем удобным или не до конца проработанным, соответствующие user stories помечаются для доработки в одном из следующих спринтов.

По завершении демонстрации команда проводит ретроспективу (Sprint Retrospective), во время которой обсуждаются сильные и слабые стороны прошедшего цикла, выявляются узкие места (например, слишком длительное тестирование или сложная интеграция с Kandinsky) и формулируются предложения по улучшению рабочих процессов к следующему спринту. Затем цикл повторяется, забирая из Product Backlog следующие приоритетные задачи.

в) Финальная интеграция и развёртывание всех инкрементов

После того как завершены все запланированные спринты и в Product Backlog не осталось критических задач, начинаются работы по сборке и интеграции всех частей системы. В эту фазу входят:

- Объединение кода веб-приложения, серверной части, мобильного клиента и AI-модуля в единую сборку.

- Проведение полного end-to-end тестирования, охватывающего все пользовательские сценарии: от выбора и генерации превью букета до приёма заказа в офлайн-режиме и его синхронизации.

- Подготовка инфраструктуры для условного «пуска» (production-like окружение): настройка серверов, базы данных PostgreSQL, DNS-записей и сертификатов.

- Разработка финальных инструкций для флористов и курьеров: как устанавливать мобильное приложение, как работать с офлайн-данными, как проверять сгенерированные AI-превью.

Результатом этой фазы является «объединённая» система, готовая к демонстрации в максимально близкой к боевой среде без фактического привлечения реальных клиентов.

г) Поддержка и доработка

Заключительная фаза жизненного цикла начинается сразу после «условного запуска» системы. Команда продолжает работать над устранением критических багов, возникающих в процессе пилотного использования, а также собирает запросы на дополнительные функции, которые будут включены в будущие спринты. Основные задачи этой стадии:

- Мониторинг работоспособности сервисов (логирование запросов, контроль производительности).

- Оперативное исправление ошибок, критичных для реального использования (например, сбой синхронизации офлайн-заказов или падение AI-модуля).

- Постоянное обновление Product Backlog на основе идей, поступивших от владельца магазина и сотрудников (флористов, курьеров).

- Подготовка и проведение следующих инкрементов, если появляются новые требования (например, расширение каталога более чем до 500 SKU или улучшение качества AI-визуализации).

Хотя в диаграмме эта фаза обозначена кругом, в реальности она также строится по принципу повторяющихся небольших спринтов: каждую новую задачу команда оценивает и закладывает в очередной спринт, после чего цикл разработки и тестирования повторяется.

Таким образом, жизненный цикл проекта представлен не как классическая «каскадная модель» от начала до конца, а как сочетание единовременных этапов (инициация и финальная интеграция) с непрерывным итеративным циклом разработки и тестирования. Такой подход позволяет получать рабочие инкременты уже после первого спринта, демонстрировать их бизнес-заказчику и вносить необходимые корректировки в ходе последующих итераций, что особенно важно при работе над сложными комплексными системами с множеством взаимозависимых компонентов.

### 4.4.2 Состав команды с учетом выполнения работ

В рамках реализации проекта «Веточка» целесообразно сформировать междисциплинарную команду, способную обеспечить проектирование, разработку, тестирование и последующее сопровождение системы. Ниже приведено описание ключевых ролей, необходимых для полного цикла работ:

1. Руководитель проекта (1 человек)

Отвечает за стратегическое управление и координацию всех этапов разработки. В его задачи входит: формирование и контроль графика работ, мониторинг выполнения задач, управление рисками, распределение ресурсов и взаимодействие со всеми заинтересованными сторонами (владелец магазина, поставщики AI-сервиса, платёжные системы и прочие). Руководитель проекта обеспечивает, чтобы результаты каждой итерации (спринта) соответствовали целям бизнеса и требованиям к качеству.

2. Бизнес-аналитик (1 человек)

Собирает и документирует подробные требования к системе, исходя из сценариев использования, специфики цветочного магазина «Веточка» и ожиданий конечных пользователей (покупателей, флористов, курьеров, администраторов). Аналитик проводит интервью с персоналом, систематизирует бизнес-процессы, готовит User Stories и Acceptance Criteria, контролирует согласование результатов итераций с заказчиком и уточняет приоритеты в Product Backlog.

3. Архитектор (1 человек)

Проектирует общую техническую архитектуру решения, отвечающего за взаимодействие веб-клиента, серверной части и мобильного приложения с AI-модулем визуализации. Он выбирает стек технологий (React, Spring Boot, Kotlin, PostgreSQL, интеграция с Kandinsky API), формирует схему слоёв приложения (Presentation, Business Logic, Data Access), разрабатывает логику API и модели данных. Задача архитектора — обеспечить масштабируемость, отказоустойчивость и возможность дальнейшего расширения системы.

4. Разработчик серверной части (1 человек)

Создаёт и поддерживает REST-API на Spring Boot (Java 17). Этот специалист реализует бизнес-логику (операции с каталогом товаров, оформление и изменение заказов, авторизация/аутентификация пользователей), настраивает взаимодействие с базой данных PostgreSQL и интегрирует сторонние сервисы (AI-модуль Kandinsky, платёжные шлюзы). В рамках работы разработчик сервера создаёт модель данных, настраивает триггеры для автоматического списания просроченных цветов и обеспечивает корректное хранение и обновление статусов заказов.

5. Frontend-разработчик (1 человек)

Отвечает за реализацию веб-интерфейса на React. Его обязанности включают проектирование компонентов каталога, конструктора букетов, корзины, страниц оформления заказа и админ-панели (управление товарами, просмотр заказов, аналитика остатков). Frontend-разработчик отвечает за адаптивный дизайн (работа на мобильных и десктопных устройствах), интеграцию с REST-API и отрисовку AI-превью букетов, получаемых из Kandinsky. Важной задачей является обеспечение стабильной производительности (время отклика <200 мс при типовой нагрузке) и удобного UX.

6. Android (Kotlin)-разработчик (1 человек)

Создаёт мобильное приложение для флористов и курьеров, которое работает в офлайн-режиме. Он разрабатывает локальную базу данных (Room/SQLite) для хранения до 100 активных заказов, реализует экраны списка заказов, деталей заказа, изменённые статусы («собран», «передан курьеру», «доставлен») и взаимодействие с веб-API при восстановлении соединения (синхронизация в течение 60 секунд). Kotlin-разработчик также интегрирует AI-модуль, чтобы флористы видели предварительное изображение букета на своём устройстве, и настраивает механизм кеширования справочных данных (остатков по филиалам).

7. Инженер DevOps / DBA (1 человек)

Отвечает за настройку инфраструктуры разработки, тестирования и демонстрации (staging) приложения. Он развертывает контейнеризированные сервисы (Docker/Kubernetes), настраивает CI/CD-пайплайны для автоматической сборки и деплоя веб- и мобильных модулей, а также администрирует базу данных PostgreSQL (резервное копирование, настройка репликации, оптимизация запросов). DevOps-инженер следит за мониторингом производительности (Prometheus/Grafana), логированием (ELK) и безопасностью (настройка брандмауэров, SSL-сертификатов).

8. Тестировщик (1 человек)

Планирует и проводит модульное, интеграционное и системное тестирование. Он разрабатывает сценарии для Unit-тестов (JUnit/Mockito для backend, JUnit/AndroidX для мобильного клиента), интеграционные тесты API (Postman/Insomnia), а также end-to-end тесты (Selenium/Playwright для веб-части). Тестировщик документирует найденные дефекты и контролирует их исправление во время каждого спринта, участвует в приёмочном тестировании (UAT) с финальным демонстрациям инкрементов.

9. Технический писатель (1 человек)

Создаёт и актуализирует полную документацию проекта:

Описание архитектуры и технического решения (архитектурные схемы, ER-модель базы).

Руководства пользователя и администратора («как собрать букет в конструкторе», «как подтверждать заказ офлайн»).

API-документацию (спецификации REST-эндпоинтов, форматы запросов/ответов, примеры JSON для вызова AI-модуля).

Инструкции по развертыванию и поддержке (DevOps-гайд).

10. Специалист по поддержке пользователей (1 человек)

Организует обучение флористов и курьеров работе с новой системой: проводит тренинги по использованию мобильного приложения (в том числе по офлайн-режиму), объясняет интерфейс админ-панели и работу конструкторского модуля. После «условного Go-Live» отвечает за техническую поддержку: регистрирует запросы в баг-трекер, консультирует по возникающим вопросам, координирует исправление нештатных ситуаций.

### 4.4.3 Структурная декомпозиция работ

Таблица №5 – Структурная декомпозиция работа

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | Работа | Длительность (дни) | Начало | Окончание | Предшественник(и) | Ресурсы (люди, оборудование, ПО) | Ответственный | Планируемый результат |
| Стадия 1 | Инициация проекта |  |  |  |  |  |  |  |
| 1.1 | Сбор бизнес-требований и определение целей | 5 | 01.01.2025 | 05.01.2025 | - | Руководитель проекта, Бизнес-аналитик; ноутбук, доступ в интернет | Руководитель проекта | Первичное ТЗ, список стейкхолдеров |
| 1.1.1 | Интервью с владельцем магазина и ключевыми пользователями (флористы, курьеры) | 2 | 01.01.2025 | 02.01.2025 | - | Руководитель проекта, Бизнес-аналитик; Zoom, ноутбук | Бизнес-аналитик | Фиксация потребностей и приоритетов |
| 1.1.2 | Формулирование проблематики, целей и задач проекта | 3 | 03.01.2025 | 05.01.2025 | 1.1.1 | Бизнес-аналитик; Word/Google Docs, ноутбук | Бизнес-аналитик | Документ «Цели и задачи проекта» |

Продолжение таблицы №5

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1.2 | Назначение ролей и формирование команды | 3 | | 06.01.2025 | | 08.01.2025 | | 1.1.2 | | Руководитель проекта; e-mail, внутренний чат | | Руководитель проекта | Утверждённый состав команды, распределение ролей |
| 1.3 | Разработка предварительного плана проекта | | 5 | | 09.01.2025 | | 15.01.2025 | | 1.2 | | Руководитель проекта, Бизнес-аналитик; Microsoft Project или Jira | Руководитель проекта | График работ с основными этапами и предполагаемыми сроками |
|  | Итого по Стадии 1 | | 11 | | 01.01.2025 | | |  | | --- | | 15.01.2025 |  |  | | --- | |  | | |  | |  |  |  |
| Стадия 2 | Анализ | |  | |  | |  | |  | |  |  |  |
| 2.1 | Сбор и анализ функциональных требований | | 10 | | 16.01.2025 | | 27.01.2025 | | 1.3 | | Бизнес-аналитик; ноутбук, доступ в интернет, Figma | Бизнес-аналитик | Документ «Функциональные требования» |
| 2.2 | Сбор и анализ нефункциональных требований | | 10 | | 28.01.2025 | | 10.02.2025 | | 2.1 | | Бизнес-аналитик; ноутбук, доступ в интернет | Бизнес-аналитик | Документ «Нефункциональные требования» |
| 2.3 | Формирование Product Backlog и приоритизация задач | | 5 | | 11.02.2025 | | 15.02.2025 | | 2.2 | | Бизнес-аналитик, Архитектор; Jira/Trello | Бизнес-аналитик | Приоритизированный Product Backlog |
|  | Итого по Стадии 2 | | 25 | | 16.01.2025 | | 15.02.2025 | |  | |  |  |  |

Продолжение таблицы №5

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стадия 3 | Проектирование |  |  |  |  |  |  |  |
| 3.1 | Архитектурное проектирование системы | 7 | 16.02.2025 | 24.02.2025 | 2.3 | Архитектор, Разработчик сервера; ноутбук, draw.io | Архитектор | Архитектурная схема (C4) |
| 3.2 | Декомпозиция логической модели БД и проект ER-диаграммы | 7 | 25.02.2025 | 05.03.2025 | 3.1 | Архитектор, Разработчик сервера; pgAdmin, ноутбук | Архитектор | ER-модель базы данных |
| 3.3 | Прототипирование UI/UX веб-интерейса и мобильных экранов | 10 | 06.03.2025 | 19.03.2025 | 3.2 | Frontend-разработчик, Android-разработчик; Figma, ноутбуки | Frontend-разработчик | Макеты страниц и экранов (Figma) |
| 3.4 | Определение спецификаций REST-API и форматов взаимодействия с AI-модулем | 5 | 20.03.2025 | 26.03.2025 | 3.3 | Архитектор, Backend-разработчик; Swagger/OpenAPI, ноутбуки | Архитектор | Документ «Спецификация REST-API» |
|  | Итого по Стадии 3 | 29 | 16.02.2025 | 26.03.2025 |  |  |  |  |

Продолжение таблицы №5

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стадия 4 | Разработка |  |  |  |  |  |  |  |
| 4.1 | Спринт 1: Настройка окружения и «Hello World» | 7 | 27.03.2025 | 29.03.2025 | 3.4 | Backend-разработчик, Frontend-разработчик; IntelliJ IDEA, VSCode, ноутбуки | Backend-разработчик | Skeleton проекта |
| 4.2 | Спринт 2: CRUD для каталога и пользователей | 10 | 30.03.2025 | 06.04.2025 | 4.1 | Backend-разработчик, Frontend-разработчик; PostgreSQL, React, ноутбуки | Backend-разработчик | CRUD для users и products, UI каталога |
| 4.3 | Спринт 3: Корзина и оформление заказа | 10 | 07.04.2025 | 14.04.2025 | 4.2 | Frontend-разработчик, Backend-разработчик; Stripe sandbox, React, ноутбук | Frontend-разработчик | Функция корзины, оформление заказа |
| 4.4 | Спринт 4: Офлайн-хранение заказов в мобильном приложении | 10 | 15.04.2025 | 21.04.2025 | 4.3 | Android-разработчик; Android Studio, Room/SQLite | Android-разработчик | Экран списка заказов, локальное хранилище |

Продолжение таблицы №5

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4.5 | Спринт 5: Синхронизация мобильного клиента с сервером | 7 | 21.04.2025 | 28.04.2025 | 4.4 | Android-разработчик, Backend-разработчик; Retrofit, PostgreSQL | Android-разработчик | Офлайн-передача статусов, синхронизация |
| 4.6 | Спринт 6: Интеграция AI-модуля (Kandinsky) | 7 | 29.04.2025 | 06.05.2025 | 4.5 | Backend-разработчик, Android-разработчик; API-ключ Kandinsky, ноутбуки | Backend-разработчик | Превью букета (≤3 c) |
| 4.7 | Спринт 7: Админ-панель: управление товарами и заказами | 7 | 07.05.2025 | 14.05.2025 | 4.6 | Frontend-разработчик, Backend-разработчик; React/Admin-UI, PostgreSQL | Frontend-разработчик | Готовая админ-панель |
| 4.8 | Спринт 8: Доработка и оптимизация функционала | 7 | 15.05.2025 | 20.05.2025 | 4.7 | Вся команда; ноутбуки, тестовый сервер | Scrum-команда | Исправлены баги, оптимизирован офлайн-режим и AI |
|  | Итого по Стадии 4 | 65 | 27.03.2025 | 20.05.2025 |  |  |  |  |

Продолжение таблицы №5

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стадия 5 | Тестирование |  |  |  |  |  |  |  |
| 5.1 | Модульное тестирование backend и frontend | 7 | 20.05.2025 | 23.05.2025 | 4.8 | Тестировщик; JUnit, Jest, Mockito, ноутбук | Тестировщик | Набор Unit-тестов |
| 5.2 | Интеграционное тестирование веб ↔ сервер ↔ AI | 10 | 24.05.2025 | 28.05.2025 | 5.1 | Тестировщик; Postman, ноутбук | Тестировщик | Отчёт об интеграционных тестах |
| 5.3 | Интеграционное тестирование мобильного приложения ↔ сервер | 7 | 29.05.2025 | 30.05.2025 | 5.2 | Тестировщик; смартфоны, ноутбук | Тестировщик | Отчет о стаб. офлайн-режима и синхр. |
| 5.4 | Системное (E2E) тестирование | 7 | 31.05.2025 | 03.08.2025 | 5.3 | Тестировщик, Бизнес-аналитик; различные устройства | Тестировщик | Протестированы от заказа до доставки |
| 5.5 | Приёмочное (UAT) тестирование | 7 | 31.07.2025 | 08.08.2025 | 5.4 | Тестировщик, Бизнес-аналитик; ноутбук, смартфон | Бизнес-аналитик | Отчёт по замечаниям конечных пользователей |
|  | Итого по Стадии 5 | 38 | 22.06.2025 | 08.08.2025 |  |  |  |  |

Продолжение таблицы №5

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Стадия 6 | Развёртывание и сопровождение |  |  |  |  |  |  |  |
| 6.1 | Подготовка тестового окружения (staging) | 5 | 09.08.2025 | 15.08.2025 | 5.5 | DevOps-инженер; серверное оборудование, Docker/K8s | DevOps-инженер | Настроена staging-среда, CI/CD |
| 6.2 | Пилотное развёртывание и обучение персонала | 7 | 16.08.2025 | 24.08.2025 | 6.1 | DevOps-инженер, Специалист по поддержке; ноутбуки, смартфоны | Специалист по поддержке | Опробована система, обучен персонал |
| 6.3 | Корректировка по результатам пилотного развёртывания | 7 | 25.08.2025 | 02.09.2025 | 6.2 | Scrum-команда; ноутбуки, сервер | Scrum-команда | Внесены правки по замечаниям |
| 6.4 | Условный Go-Live и «горячая» поддержка | 10 | 03.09.2025 | 16.09.2025 | 6.3 | DevOps-инженер, Специалист по поддержке; production-like окружение | DevOps-инженер | Система «в работе» |

Продолжение таблицы №5

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 6.5 | Текущая поддержка и мелкие доработки | 30 | 17.09.2025 | 17.10.2025 | 6.4 | Scrum-команда, Специалист по поддержке; ноутбуки, удалённый доступ | Scrum-команда | Исправлены баги, реализованы новые запросы |
|  | Итого по Стадии 6 | 59 | 09.08.2025 | 17.10.2025 |  |  |  |  |
| Всего по проекту |  | 222 | 01.01.2025 | 17.10.2025 |  |  |  |  |

## 4.5 Риски проекта

Любой инновационный проект, особенно в сфере высоких технологий, неизбежно сталкивается с различными рисками, которые могут повлиять на его реализацию. Поэтому рассмотрим основные риски для проекта, которые представлены в таблице 6.

Таблица №6 - Риски проекта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Источник риска | Описание риска | Код | Последствия (в т. ч. негативные и позитивные) |
| 1 | Технологические | Отключение или изменение API сервиса Kandinsky (генерация предварительных изображений букетов) | Р1 | Негативные: AI-превью перестанут работать, потребуется поиск альтернативы или освоение собственной нейросети, что отодвинет сроки и повысит затраты. |
| 2 | Поставщики | Задержка со стороны облачного хостинга или облачной БД (PostgreSQL), сбои в работе сервера | Р2 | Негативные: тестирование и пилотное развёртывание откладываются, возможны перебои в демонстрации заказчику. Позитивные: стимулирует внедрение локального тестового стенда. |
| 3 | Финансовые | Недостаточное выделение средств на аренду серверного оборудования и лицензии (Spring Boot, React) | Р3 | Негативные: придётся урезать функционал (убрать AI-превью или отсрочить мобильное приложение), возможны задержки проекта. |
| 4 | Управленческие | Уход одного из ключевых разработчиков (например, Android-разработчика) | Р4 | Негативные: снижение скорости разработки мобильного приложения, может потребоваться обучение нового специалиста. Позитивные: пересмотр сроков, улучшение документации. |
| 5 | Производственные | Ошибки в реализации офлайн-синхронизации мобильного приложения (некорректное сохранение заказов) | Р5 | Негативные: курьеры не будут видеть новые заказы, возникнут задержки доставки, повысятся обращения в техподдержку. |
| 6 | Маркетинговые / Рыночные | Появление конкурента с уже существующим интернет-магазином и встроенным конструктором букетов | Р6 | Негативные: снижение конкурентоспособности, необходимость доработать дополнительные функции. Позитивные: усиление мотивации ускорить выпуск MVP и активировать маркетинг. |
| 7 | Социальные | Нехватка квалифицированных флористов, неспособных работать с новым модулем AI-визуализации | Р7 | Негативные: букет, собранный «по превью», не будет соответствовать ожиданиям, возрастёт число возвратов и жалоб. |

Продолжение таблицы №6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 8 | Информационные / Безопасности | Уязвимости в платёжной системе (XSS, SQL-инъекции) | Р8 | Негативные: возможны утечки персональных данных клиентов, финансовые потери, штрафы, потеря доверия. Позитивные: усиление комплекта защит, повышение уровня надежности. |
| 9 | Организационные | Слабая коммуникация между веб- и мобильными разработчиками, несогласованность API-спецификаций | Р9 | Негативные: ошибки интеграции, рост числа багов, задержки на этапе тестирования. |
| 10 | Внешние (регуляторные) | Новые требования к защите персональных данных (ФЗ-152) | Р10 | Негативные: придётся перерабатывать модуль авторизации, шифрования и хранения данных, увеличатся сроки и расходы. Позитивные: повышение доверия клиентов, лучшие юридические гарантии. |

Экспертную оценку выявленных рисков с определением ранга на основании вероятности его проявления и воздействия на цель проекта представлена в таблице 7.

Таблица №7 - Экспертная оценка рисков

|  |  |  |
| --- | --- | --- |
| Код риска | Вероятность наступления (1–3) | Степень влияния на достижение цели (1–3) |
| R1 | 2 | 3 |
| R2 | 2 | 2 |
| R3 | 1 | 3 |
| R4 | 2 | 2 |
| R5 | 2 | 3 |
| R6 | 1 | 2 |
| R7 | 1 | 2 |
| R8 | 2 | 3 |
| R9 | 2 | 2 |
| R10 | 1 | 2 |

По результатам оценки рисков была составлена карта рисков, изображённая на рисунке 30.

Карта рисков разделена на несколько зон, выделенных по-разному:

- риски критического уровня обозначены на диаграмме толстой линией – это те риски, которые представляют наибольшую угрозу для организации из-за высокой вероятности возникновения или значительного возможного ущерба;

- риски среднего уровня отмечены двойной линией – они характеризуются средней вероятностью наступления или умеренным влиянием на финансовые показатели и репутацию компании;

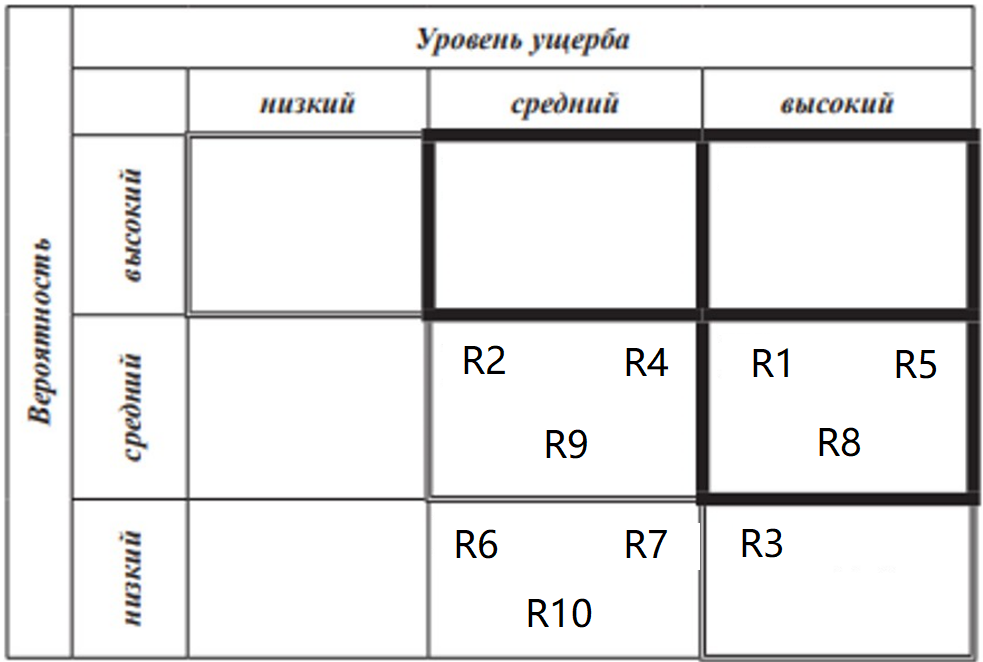


Рисунок №30 - Карта рисков проекта

- риски низкого уровня обозначены тонкой линией – это риски с низкой вероятностью возникновения и/или не оказывающие существенного влияния на деятельность организации.

На основании проведенной экспертной оценки и карты рисков все выявленные угрозы были классифицированы по уровням критичности. Для критического и среднего уровней разработаны соответствующие стратегии реагирования, представленные в таблице 8.

Таблица №8 - Стратегии и методы реагирования на риски

|  |  |  |
| --- | --- | --- |
| Код риска | Стратегия | Конкретные меры |
| R1 | Минимизация + Передача | Поддерживать резервную интеграцию с альтернативным AI. |
| R2 | Минимизация | Настроить автоматические бэкапы и резервное облако. |
| R3 | Принятие + Минимизация | Отложить малокритичные функции и заложить резерв в бюджете. |
| R4 | Минимизация | Документировать API и иметь пул замен Android-разработчиков. |
| R5 | Минимизация | Добавить unit-тесты офлайн-синхронизации и логирование. |
| R6 | Мониторинг | Вести базу кандидатов и проводить перекрёстное обучение. |
| R7 | Мониторинг | Внедрить обязательные code review и статический анализ. |
| R8 | Минимизация + Принятие | Регулярно проводить аудит безопасности и обновлять зависимости. |
| R9 | Минимизация | Утвердить единый OpenAPI-спецификации и ежедневные синхроны. |
| R10 | Мониторинг | Отслеживать изменения законодательства и готовить инструкции. |

## 4.6 Бизнес-модель

Для проекта «Веточка» оптимальной будет модель индивидуального сопровождения и разовой продажи лицензии с последующим техподдержкой. Поскольку система разрабатывается единственно для конкретного магазина «Веточка» и полностью заточена под его бизнес-процессы. Магазин «Веточка» оплачивает полный объём работ по проектированию, разработке, тестированию и внедрению системы: веб-приложения, мобильного клиента и AI-модуля. Стоимость рассчитывается на основе оценённого объёма задач и почасовой ставки исполнителей. В результате «Веточка» получает готовый программный комплекс и исключительные права на его использование без ограничения по времени.

Поскольку проект разбит на спринты (8 основных итераций), расчёт стоимости ведётся поэтапно: каждый спринт имеет фиксированную стоимость, и оплата происходит по завершении и приёмке результатов спринта владельцем магазина. Это позволяет «Веточке» гибко контролировать бюджет, а команде — получать своевременную плату за сданные фичи. По итогам каждого этапа заказчик подписывает Акт выполненных работ, и проект движется к следующему спринту..

После передачи основной системы оформляется отдельный договор техподдержки: он может быть как ежемесячным (фиксированная ежемесячная плата за гарантийное сопровождение, исправление критических ошибок), так и почасовым (оплата только фактически потраченного времени на мелкие доработки). При необходимости «Веточка» может запросить новые модули (например, интеграцию с локальными курьерскими службами, расширенные отчёты по продажам) — тогда выполнение этих работ оплачивается отдельно по заранее согласованной ставке.

Для генерации превью «конструируемого» букета используется внешнее API Kandinsky, лицензионные расходы на которое оплачивает сам магазин. Команда-посредник интегрирует соответствующий модуль и обучает владельцев «Веточки» пользоваться настройками. В будущем, если «Веточка» захочет перенести AI-рендер на собственные нейросети, это можно будет оформить как отдельный контракт на доработку.

Итого, бизнес-модель строится как «Разовая разработка + Договор техподдержки/доработок».

Такая схема наилучшим образом соответствует формату «индивидуального корпоративного решения», запрашиваемого магазином «Веточка»: она гарантирует полную адаптацию под бизнес-процессы, прямой контроль над финансированием и возможность оперативных улучшений по мере роста и изменения потребностей клиента.

## 4.7 Стоимость проекта

### 4.7.1 Анализ структуры затрат

Затраты на выполнение проекта состоят из затрат на заработную плату исполнителям, затрат на закупку или аренду оборудования, затрат на организацию рабочих мест, и затрат на накладные расходы (1):

С = Сзп + Сэл + Соб + Сорг + Снакл , (1) где: Сзп – заработная плата исполнителей;

Сэл – затраты на электроэнергию;

Соб – затраты на обеспечение необходимым оборудованием; Сорг – затраты на организацию рабочих мест;

Снакл – накладные расходы.

Затраты на выплату исполнителям заработной платы определяется следующим соотношением (2):

*Сзп = Сз.осн + Сз.доп + Сз.отч* , (2) где: Сз.осн – основная заработная плата;

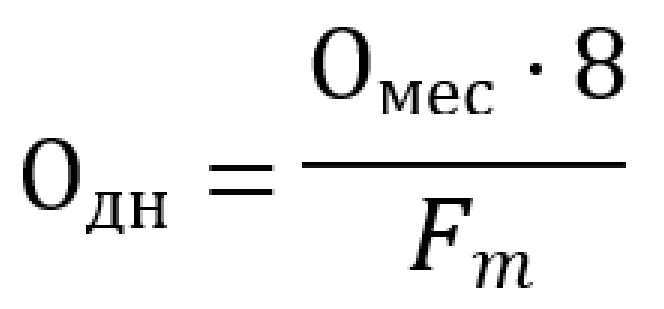
Сз.доп – дополнительная заработная плата; Сз.отч – отчисление с заработной платы.

Расчет основной заработной платы при дневной оплате труда исполнителей проводится на основе данных по окладам и графику занятости исполнителей (3):

Сз.осн = *Одн* ×*Тзан ,* (3)

где: Одн – дневной оклад исполнителя;

Тзан – число дней, отработанных исполнителем проекта. При 8-и часовом рабочем дне оклад рассчитывается:



где: Омес – месячный оклад;

Fм – месячный фонд рабочего времени.

В таблице 13 можно увидеть расчет заработной платы с перечнем исполнителей и их месячных и дневных окладов, а также времени участия в проекте для каждого исполнителя. (Источник данных является: hh.ru, средние зарплаты для Москвы в IT-сфере.)

Расчёт дневного оклада для проекта:

# Список литературы

1. Маркетинговое исследование «Электронная торговля 2020–2024, прогноз Data Insight» [Электронный ресурс]. Режим доступа: <https://datainsight.ru/DI_eCommerce2020_2024> (Дата обращения: 26.02.2025).

2. Аналитика сайта AKIT. Режим доступа: <https://akit.ru/analytics/analyt-data> (Дата обращения: 26.02.2025).

3. ТОП-14 лучших сервисов доставки букетов и цветов на дом в 2024–2025 году в Москве: рейтинг и обзор цветочных интернет-магазинов. DTF. Режим доступа: <https://dtf.ru/onlinekursy/2733351-top-14-luchshih-servisov-dostavki-buketov-cvetov-na-dom-v-2024-2025-godu-v-moskve-reiting-i-obzor-cvetochnyh-internet-magazinov> (Дата обращения: 26.02.2025).

4. Официальный сайт Flowwow. Режим доступа: <https://about.flowwow.com> (Дата обращения: 26.02.2025).

5. О компании Flor2U. Режим доступа: <https://flor2u.ru/o-kompanii/> (Дата обращения: 26.02.2025).

6. О компании FloraExpress. Режим доступа: <https://www.floraexpress.ru/ru/page/about> (Дата обращения: 27.02.2025).

7. О компании СоюзЦветТорг. Режим доступа: <https://www.souzcvettorg.ru/about/> (Дата обращения: 27.02.2025).