**СОДЕРЖАНИЕ**

[Термины и определения 2](#_Toc198816483)

[Перечень сокращений и обозначений 3](#_Toc198816484)

[Введение 4](#_Toc198816485)

[1 общая часть 6](#_Toc198816486)

[1.1 анализ предметной области 6](#_Toc198816487)

[1.2 постановка задачи 7](#_Toc198816488)

[1.3 требования к системе 7](#_Toc198816489)

[1.4 анализ существующих решений 8](#_Toc198816490)

[1.5 архитектура приложения 8](#_Toc198816491)

[2 специальная часть 11](#_Toc198816492)

[2.1 проектирование системы 11](#_Toc198816493)

[2.2 реализация системы 16](#_Toc198816494)

[2.3 тестирование системы 20](#_Toc198816495)

[3 экономическая часть 25](#_Toc198816496)

[4 техника безопасности и охрана труда 27](#_Toc198816497)

[4.1 анализ условий труда специалиста по информационным технологиям 27](#_Toc198816498)

[4.2 расчёт искусственного освещения 28](#_Toc198816499)

[4.3 электробезопасность 29](#_Toc198816500)

[4.4 пожарная безопасность 29](#_Toc198816501)

[Заключение 31](#_Toc198816502)

[Список использованных источников 32](#_Toc198816503)

[Приложение А (справочное) Исходные коды 34](#_Toc198816504)

[Приложение Б (справочное) Словарь данных 43](#_Toc198816505)

[Приложение В (справочное) Тест-кейсы 45](#_Toc198816506)

**ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ**

В настоящем отчете о НИР применяют следующие термины с соответствующими определениями, которые представлены в таблице 1.

Таблица – Расшифровка терминов и определений

|  |  |
| --- | --- |
| **ТЕРМИН** | **ОПРЕДЕЛЕНИЕ** |
| Telegram-бот | Программное обеспечение, работающее в мессенджере Telegram, предназначенное для автоматизации взаимодействия с пользователями. |
| Администратор | Пользователь с расширенными правами, управляющий каталогом, рассылками и настройками бота. |
| База данных | Организованная совокупность данных, хранящая информацию о товарах и пользователях. |
| Каталог товаров | Структурированный перечень товаров (цветов, букетов, подарков) с их описанием и стоимостью. |
| Рассылка уведомлений | Автоматическая отправка сообщений подписанным пользователям о новинках, акциях или изменениях в ассортименте. |
| Хостинг | Размещение интернет-проектов на физических и виртуальных серверах, подключенных к Интернету, для непрерывного обеспечения присутствия сайтов в Сети |

**ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ**

В настоящем отчете о НИР применяют следующие сокращения и обозначения, расшифровка которых приведена в таблице 2.

Таблица – Расшифровка сокращений

| **СОКРАЩЕНИЕ/ОБОЗНАЧЕНИЕ** | **РАСШИФРОВКА** |
| --- | --- |
| API | Application Programming Interface — интерфейс программирования приложений. |
| CRM | Customer Relationship Management — система управления взаимодействием с клиентами. |
| SQL | Structured Query Language — язык структурированных запросов для работы с БД. |
| СУБД | Система управления базами данных. |
| БД | База данных. |

**ВВЕДЕНИЕ**

В современных условиях цифровизации бизнеса автоматизация торговли приобретает особое значение. Использование информационных систем позволяет повысить эффективность управления, улучшить взаимодействие с клиентами и минимизировать ручной труд. Цветочные магазины, как часть розничной торговли, также сталкиваются с необходимостью автоматизации процессов, таких как каталогизация товаров и управление клиентской базой.

В рамках данного проекта разрабатывается информационная система на основе Telegram-бота для цветочного магазина. Telegram-бот обеспечит удобный доступ клиентов к каталогу товаров, возможность получения информации о новинках, актуальном наличии и контактных данных магазина. Это позволит улучшить клиентский сервис и увеличить вовлеченность пользователей.

Для разработки проекта использовался язык программирования Python версии 3.10 [1], а также библиотека Aiogram 2.25.0 для работы с Telegram Bot API [2]. База данных реализована с использованием SQLite [3].

Целью проекта является автоматизация процесса ведения каталога, оптимизация создания рассылок. Для достижения данной цели необходимо выполнить следующие задачи:

* Провести анализ предметной области и выявить основные потребности пользователей;
* Определить функциональные и нефункциональные требования к системе;
* Спроектировать архитектуру системы, включая базу данных и взаимодействие компонентов;
* Реализовать Telegram-бота, обеспечивающего доступ к каталогу товаров и актуальной информации;
* Разработать механизм рассылки уведомлений для клиентов;
* Провести тестирование и отладку системы;
* Подготовить документацию для пользователей и администраторов.

Краткое описание структуры отчета. Отчет включает следующие разделы:

* Введение – описание актуальности, целей и задач проекта;
* Общая часть – анализ предметной области, обзор существующих решений и обоснование выбора инструментов;
* Специальная часть – проектирование, реализация и тестирование системы;
* Заключение – подведение итогов и перспективы дальнейшего развития;
* Приложения – дополнительные материалы и кодовые фрагменты.

1. Общая часть
   1. Анализ предметной области

ИП «АБРАМОВИЧ АНАСТАСИЯ АБРАМОВНА» – это заказчик приложения для более удобной работы магазина цветов «FLOWER PRINCESS» по адресу город Санкт-Петербург улица Караваевская дом 26 корпус 1. Основные требования от заказчика:

* управление ассортиментом товаров;
* ведение клиентской базы;
* продвижение продукции через цифровые каналы.

Основной проблемой магазина является необходимость автоматизации процессов продаж и взаимодействия с клиентами. На данный момент рассылка производится вручную клиентам, что затрачивает очень много времени сотрудников магазина.

Цветочные магазины занимаются продажей цветов, букетов и сопутствующих товаров, таких как подарочные упаковки и мягкие игрушки. Современные технологии позволяют автоматизировать эти процессы, повысив эффективность и удобство работы. Telegram бот будет составлен исходя из запросов и требований цветочного магазина.

Основные проблемы, с которыми сталкиваются владельцы цветочных магазинов:

* Отсутствие удобного механизма информирования клиентов о наличии товаров;
* Трудности в управлении ассортиментом и его обновлении;
* Необходимость уведомления клиентов о специальных предложениях и новинках;
* Ручное ведение клиентской базы и истории запросов;
* Ручное ведение базы данных магазина.

Использование Telegram-бота позволит решить эти проблемы, предоставив автоматизированный инструмент взаимодействия с клиентами, а также увеличит свободное время сотрудника.

* 1. Постановка задачи

Цель проекта: повышение эффективности работы цветочного магазина за счет автоматизации процессов взаимодействия с клиентами на основе Telegram-бота. Telegram-бот — это инструмент для достижения цели (автоматизации).

Основные задачи проекта:

* Разработка каталога товаров с возможностью просмотра характеристик;
* Реализация системы уведомлений о новинках и акциях;
* Интеграция базы данных для хранения информации о товарах и клиентах;
* Создание панели администратора для управления ассортиментом;
* Внедрение механизма подписки на рассылку.
  1. Требования к системе

Функциональные требования:

* 1. Работа с каталогом,
* Клиент может просматривать список товаров, их характеристики и стоимость;
* Администратор может добавлять, редактировать и удалять позиции каталога.
  1. Система уведомлений,
* Пользователи могут подписаться на рассылку акций и новинок;
* Бот отправляет автоматизированные уведомления зарегистрированным пользователям.
  1. Права доступа,
* Пользователи имеют ограниченный доступ (просмотр каталога);
* Администраторы могут управлять товарами и рассылками.
  1. Анализ существующих решений

Обзор аналогичных решений. На рынке представлены различные CRM-системы для магазинов, однако их стоимость часто высока, а функциональность избыточна для небольших цветочных магазинов. Telegram-боты применяются для автоматизации бизнеса, но требуют индивидуальной настройки.

Сравнение и обоснование разработки. Сравнение готовых решений с предлагаемой разработкой показывает, что:

* Готовые CRM – сложны в интеграции, требуют абонентской платы.
* Самостоятельная разработка – позволяет разработать гибкое решение, адаптированное под специфику бизнеса решение, соответствующее потребностям магазина, поэтому целесообразно разработать собственное решение, ориентированное на специфику бизнеса.
  1. Архитектура приложения

Проектируемая система включает:

* 1. Telegram-бот [15] – интерфейс взаимодействия с пользователем,

Telegram-бот [16] является основным инструментом взаимодействия клиентов с системой. Он предоставляет удобный интерфейс для просмотра каталога товаров и подписки на уведомления. Бот получает команды от пользователей и перенаправляет запросы в серверную часть для обработки.

Функциональные возможности бота включают:

* Обработку пользовательских команд (/start, /db /log);
* Отправку динамически формируемых сообщений с информацией о товарах;
* Реализацию inline-кнопок для удобной навигации;
* Подписки на рассылки.

Бот работает на основе библиотеки Aiogram, обеспечивающей асинхронное взаимодействие с Telegram API [4]

* 1. Серверная часть – обработка запросов и взаимодействие с базой данных,

Серверная часть отвечает за обработку всех входящих запросов от бота, выполнение бизнес-логики и взаимодействие с базой данных. Она реализована на языке Python с использованием библиотеки Aiogram и SQL-запросов для работы с данными.

Основные задачи серверной части:

* Прием и обработка запросов от бота;
* Выполнение операций с базой данных (чтение, запись, обновление информации);
* Логирование действий пользователей для последующего анализа;
* Обеспечение разграничения прав доступа между пользователями и администраторами.

Серверная часть может быть развернута локально или на облачном хостинге для обеспечения бесперебойной работы.

* 1. База данных – хранение информации о товарах и подписках пользователей,

Для хранения данных используется реляционная база данных SQLite [5], поскольку она легковесна и не требует развертывания отдельного сервера.

Основные таблицы базы данных:

* users – информация о зарегистрированных пользователях;
* flowers – каталог цветов;
* bouquets – каталог букетов;
* gifts – каталог подарков;
* contacts – контактная информация о магазине.

База данных обеспечивает хранение, быстрый поиск информации и возможность последующего анализа данных. Взаимодействия проектируемой системы показана на рисунке 1.

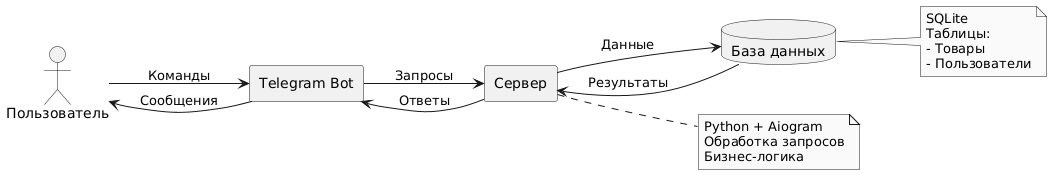


Рисунок – Схема клиент-серверного взаимодействия

1. Специальная часть
   1. Проектирование системы

На этапе проектирования были разработаны диаграмма вариантов использования, диаграмма последовательностей, диаграмма активности и диаграмма классов.

Диаграмма вариантов использования показывает (Рисунок 2), разделение доступа между клиентом и администратором, показывая визуализации различий в правах доступа и понимания архитектуры взаимодействия с ботом. Администратор имеет доступ к тем же данным, что и пользователь (просмотр каталога и контактной информации), а также имеет дополнительные возможности редактирования данных, просмотр логов и базы данных

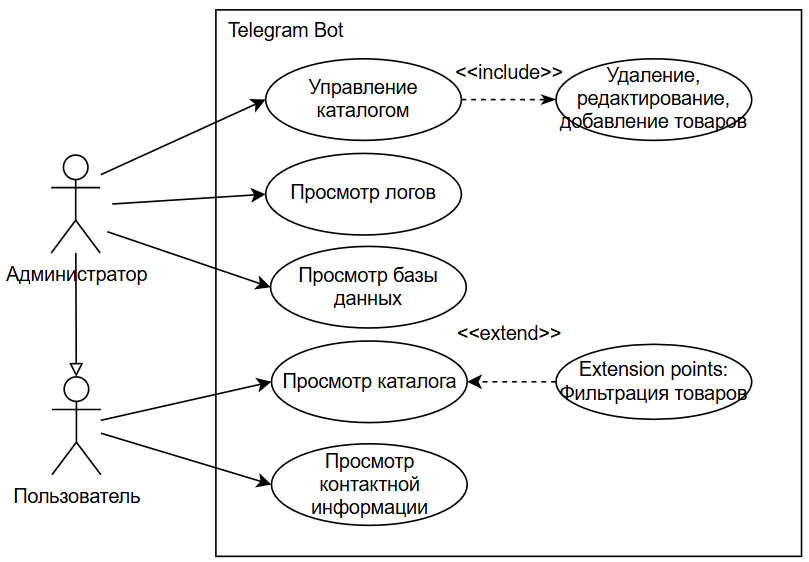


Рисунок – Диаграмма вариантов использования

Диаграмма активности (Рисунок 3) детализирует процесс взаимодействия пользователя с ботом. Например, после запуска бота (/start) клиент может выбрать просмотр каталога или просмотр контактной информации. Эти потоки соединяют выходы одного элемента с входами другого, показывая порядок выполнения действий использования бота.

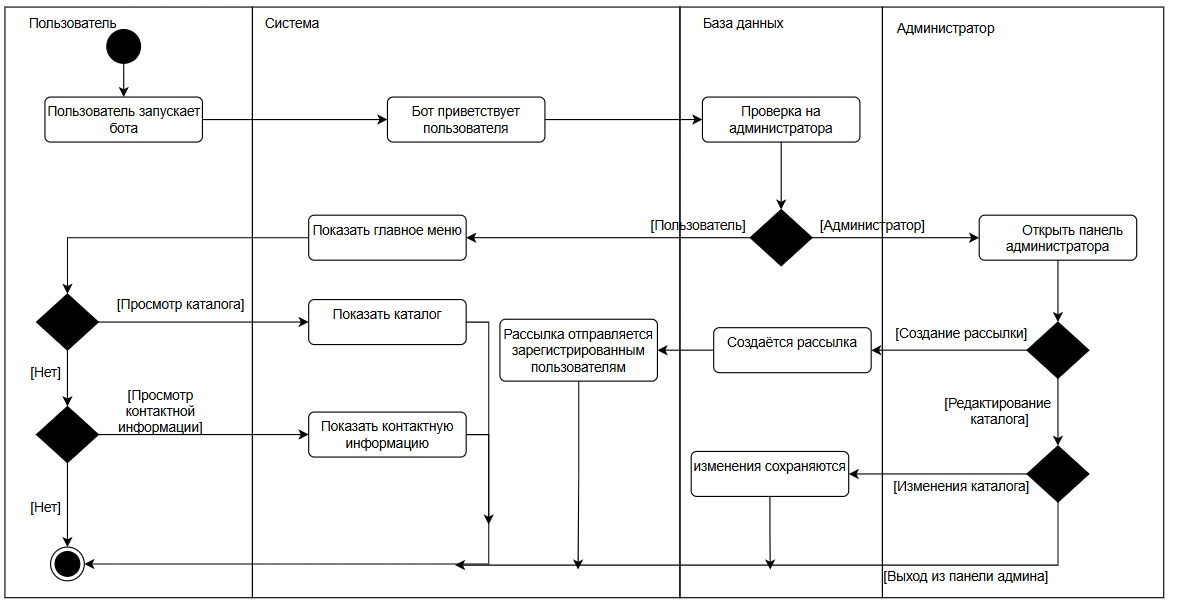


Рисунок – Диаграмма активности

Диаграмма последовательности (Рисунок 4) демонстрирует порядок выполнения команд между пользователем, ботом и базой данных. Например, при добавлении новой рассылки администратором, бот отправляет запрос в базу, вносит изменения и подтверждает операцию.

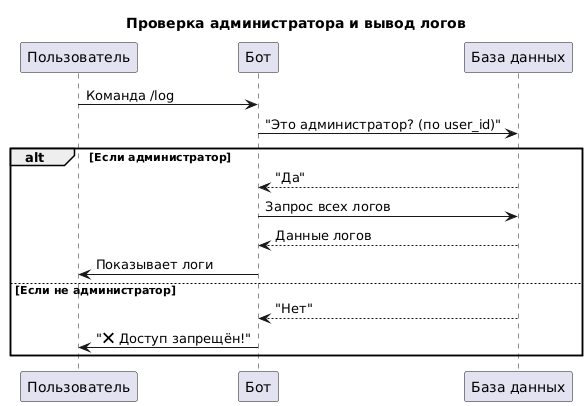


Рисунок – Диаграмма последовательности

Диаграмма классов (Рисунок 5) представляет структуру Telegram-бота для управления ассортиментом товаров (цветы, букеты, подарки) и контактами магазина. Основные компоненты включают самого бота (работающего на библиотеке Aiogram), базу данных для хранения информации, админ-панель для управления правами и логирования, а также модуль клавиатур для взаимодействия с пользователем. Бот взаимодействует с базой данных для обработки запросов и использует клавиатуры для формирования интерфейса, в то время как админ-панель контролирует доступ и настройки системы.

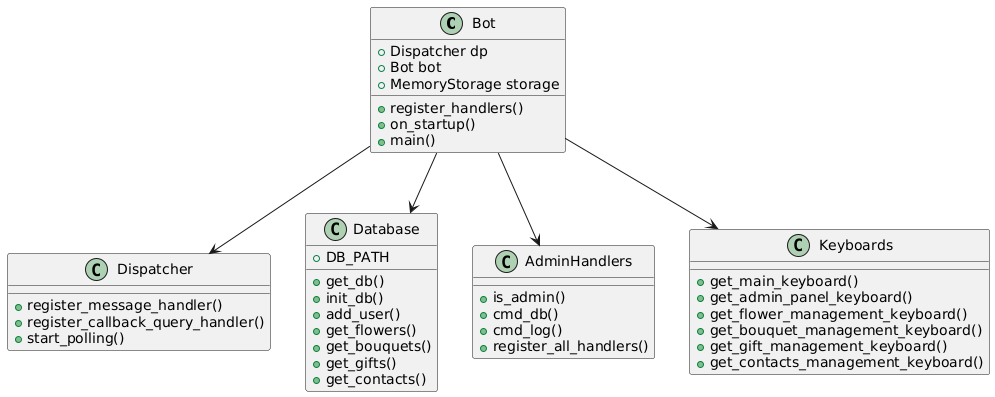


Рисунок – Диаграмма классов

Проектирование клиентской части. Telegram-бот предоставляет пользователям интерфейс для просмотра каталога, получения информации о наличии товаров и подписки на рассылку обновлений.

Архитектура Telegram-бота [6] разработана с учетом рекомендаций официальной документации Telegram [5] и принципов построения масштабируемых ботов [8].

Проектирование серверной части. Сервер обрабатывает запросы пользователей, взаимодействует с базой данных и выполняет автоматическую рассылку уведомлений.

Структура базы данных включает таблицы users, flowers, bouquets, gifts и contacts, обеспечивающие хранение и обработку данных (Рисунок 6).

На диаграмме отражены логические связи между таблицами, демонстрирующие взаимодействие пользователей с элементами каталога. Так, пользователи могут просматривать товары из различных категорий — цветов, букетов и подарков. Это отображается с помощью связей вида users одним ко многим цветы, что означает: один пользователь может взаимодействовать с множеством цветов, при этом конкретный цветок может быть не связан ни с одним пользователем.

Также реализована связь контакты один ко многим users, визуально объединяющая таблицу контактной информации с пользователями. Она предполагает, что одна запись с контактными данными (например, телефон магазина) может быть общей для нескольких пользователей, что удобно для обратной связи. База данных реализована с использованием SQLite [4]. Словарь базы данных описывает все типы и ограничения данных в таблицах (Таблица Б.1).

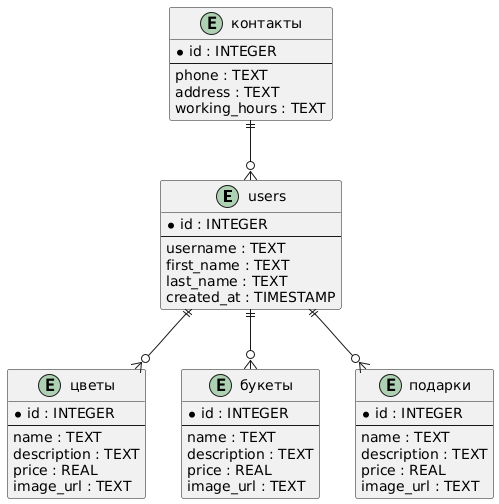


Рисунок – Схема базы данных

Основные таблицы:

* users – информация о пользователях;
* flowers – данные о цветах;
* bouquets – данные о букетах;
* gifts – данные о подарках;
* contacts – контактная информация магазина.

Пример кода базы данных в приложение А.

* 1. Реализация системы

Для разработки Telegram-бота использованы:

* Язык программирования: Python 3.10; [2]
* Библиотека для работы с Telegram API: Aiogram 2.25.0;
* База данных: SQLite3 [7]; [4]

Основные функции:

* Просмотр каталога товаров (цветы, букеты, подарки);
* Получение контактной информации магазина;
* Подписка на уведомления о новинках и акциях.

Алгоритм взаимодействия с пользователем:

1. Пользователь отправляет команду /start,
2. Бот загружает главное меню и предлагает пользователю выбрать действие,
3. Если Пользователь выбирает «Просмотр каталога» – бот загружает данные из базы и отправляет список товаров,
4. Если Пользователь начинает пользоваться ботом на – его ID заносится в базу подписчиков,
5. Администратор может войти в панель управления (/admin) и редактировать каталог и рассылку,
6. При создании рассылки бот отправляет уведомления подписчикам.

Для обработки команд пользователя использована библиотека Aiogram, которая позволяет асинхронно взаимодействовать с Telegram Bot API [3].

Пример реализации функции для добавления нового пользователя показан на рисунке 7.

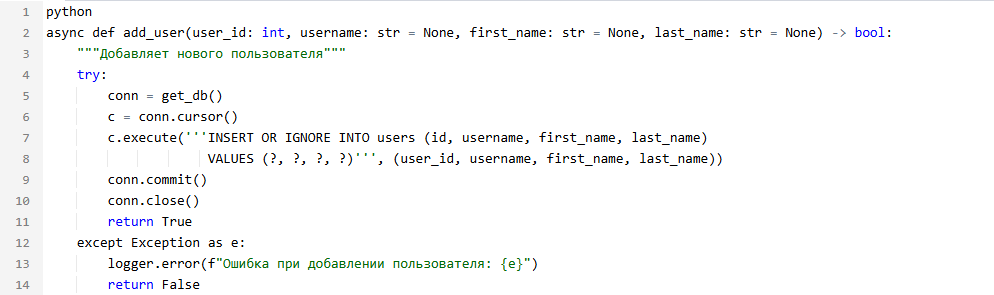


Рисунок – Добавление нового пользователя

Права доступа в приложении делятся на две основные категории: права пользователя и права администратора.

1. Пользователи могут выполнять базовые действия, такие как:

* Запуск бота с помощью команды /start или кнопки;
* Просмотр каталога товаров;
* Получение контактной информации магазина;
* Подписка на уведомления.

1. Администраторы имеют расширенные возможности:

* Управление каталогом (добавление, редактирование и удаление товаров);
* Создание и отправка рассылок;
* Настройка контактной информации магазина;
* Доступ к логам и базе данных.

Управление администраторами. Для создания и настройки бота используется BotFather – официальный бот Telegram для управления чат-ботами. Через BotFather можно задать основные функции и возможности бота. Для создания пустого бота необходимо ввести определенные команды, показанные на рисунке 8.

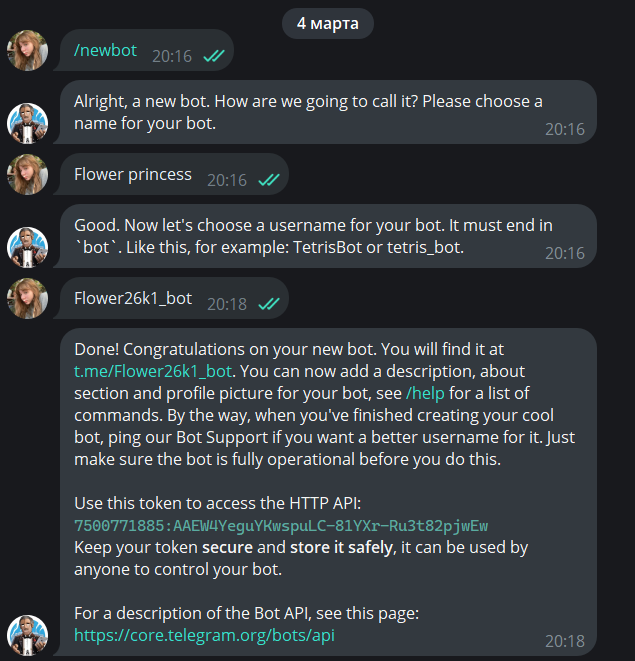


Рисунок – Создание бота и получение токена

Разграничение прав доступа реализовано на уровне кода. Для подключения проекта из PyCharm к Telegram-боту необходимо внести в файл config.py следующие данные:

* API-токен бота;
* ID администратора (ADMIN\_ID).

Это обеспечивает:

* Авторизацию бота через Telegram API;
* Разграничение прав доступа между пользователями и администраторами;
* Полноценное использование всех функций системы.

Управление администраторами в config.py Доступ к функциям администратора определяется уникальным идентификатором ADMIN\_ID, указанным в файле config.py.

Для добавления или удаления администратора текущий администратор должен вручную изменить значение ADMIN\_ID в config.py, указав новый идентификатор.

Узнать свой Telegram ID можно в профиле приложения Telegram.

Файл config.py хранится на сервере и доступен только разработчику или через панель управления хостингом, что обеспечивает безопасность [17] и предотвращает несанкционированные изменения.

В текущей версии бота отсутствует интерфейс для динамического управления администраторами – все изменения выполняются вручную.

Настройка бота в PyCharm

Для подключения бота через PyCharm выполните следующие шаги:

* Создайте бота через BotFather и получите API-токен.
* Внесите API-токен и ADMIN\_ID в файл config.py (см. рисунок 9).

С точки зрения безопасности доступ к файлу config.py ограничен файловой системой сервера, что предотвращает несанкционированное изменение настроек и обеспечивает безопасность работы бота.

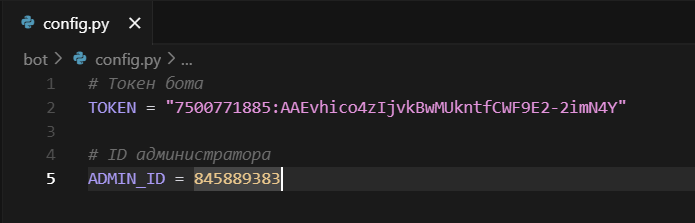


Рисунок – Указание ID человека, который является администратором

* 1. Тестирование системы

Для верификации функциональности системы разработаны тест-кейсы, охватывающие ключевые сценарии взаимодействия пользователей и администратора (см. Таблицу В.1).

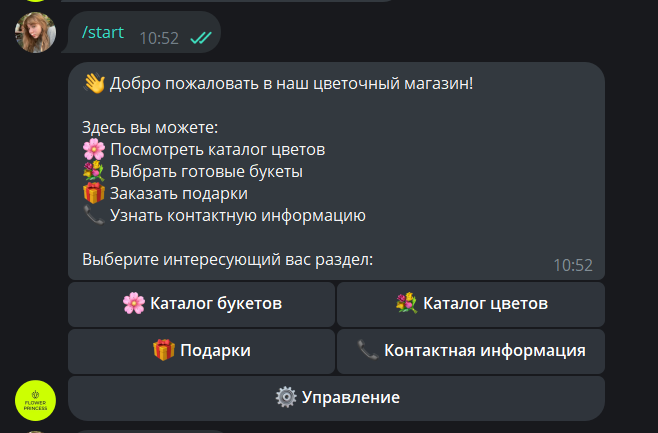


Рисунок – Тест 1, проверка команды /start

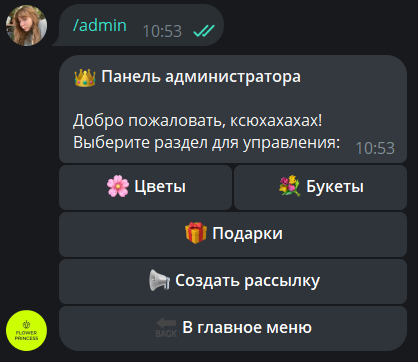


Рисунок – Тест 2, проверка команды /admin Администратором

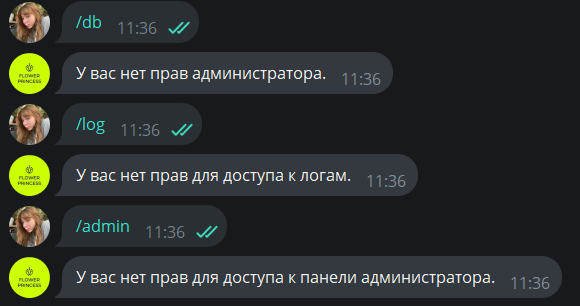


Рисунок – Тест 3, проверка команды /admin Пользователем

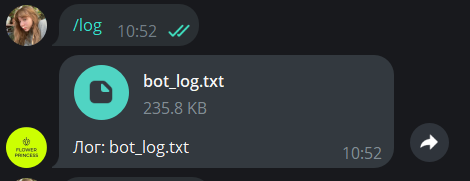


Рисунок – Тест 4, проверка команды /log Администратором

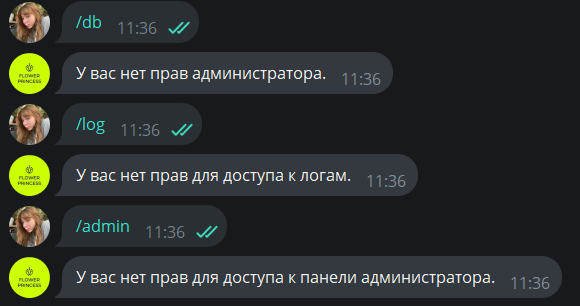


Рисунок – Тест 5, проверка команды /log Пользователем

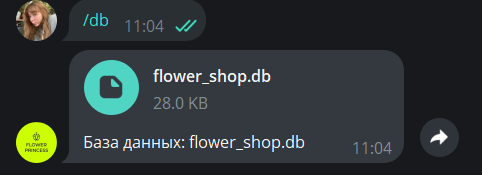


Рисунок – Тест 6, проверка команды /db Администратором

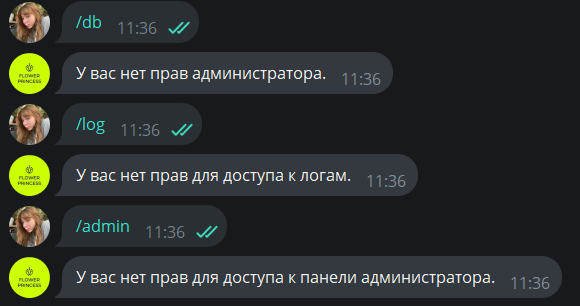


Рисунок – Тест 7, проверка команды /db Пользователем

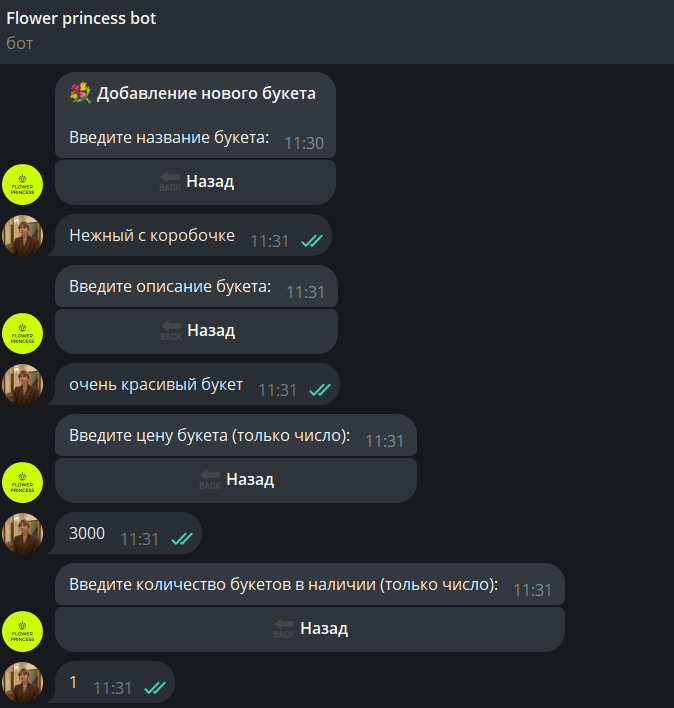


Рисунок – Тест 8, добавление букета

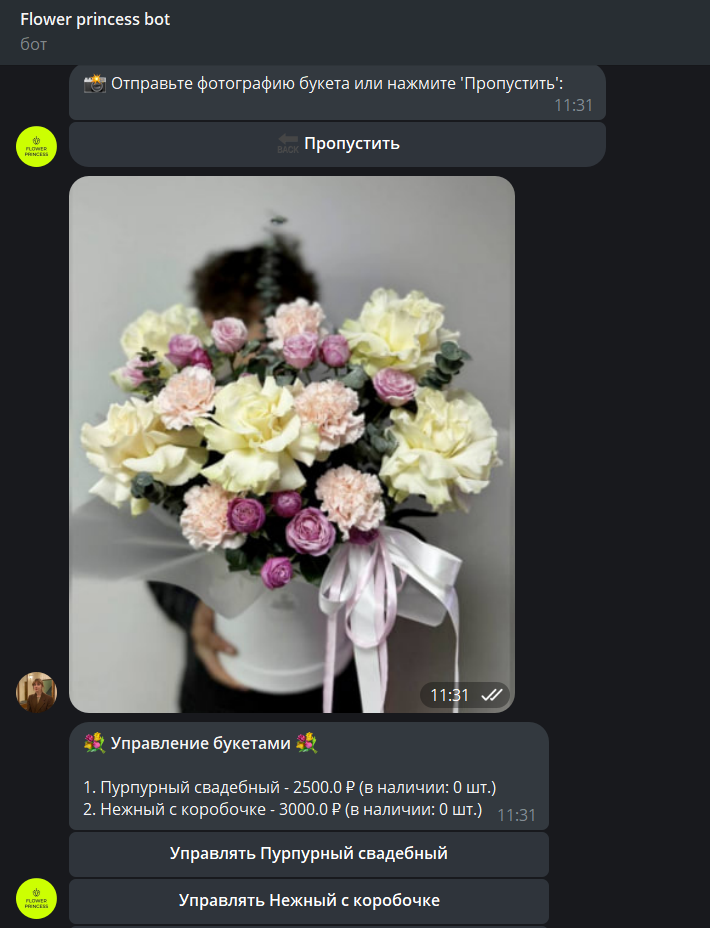


Рисунок – Тест 8, добавление букета

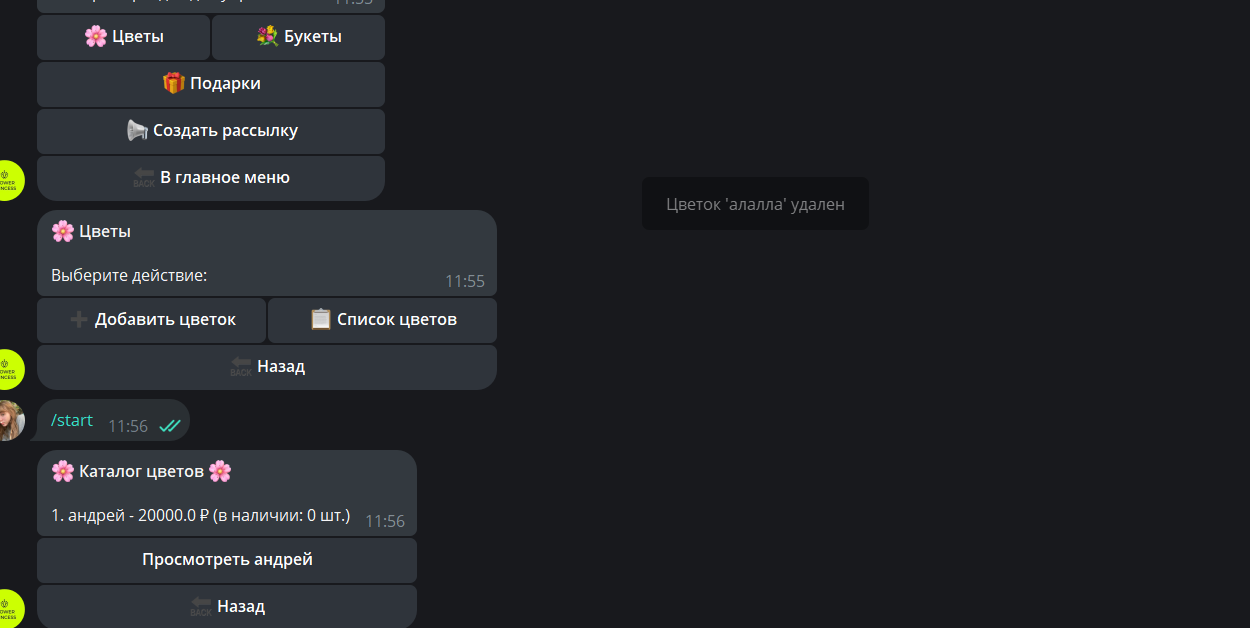


Рисунок – Тест 9, удаление цветка

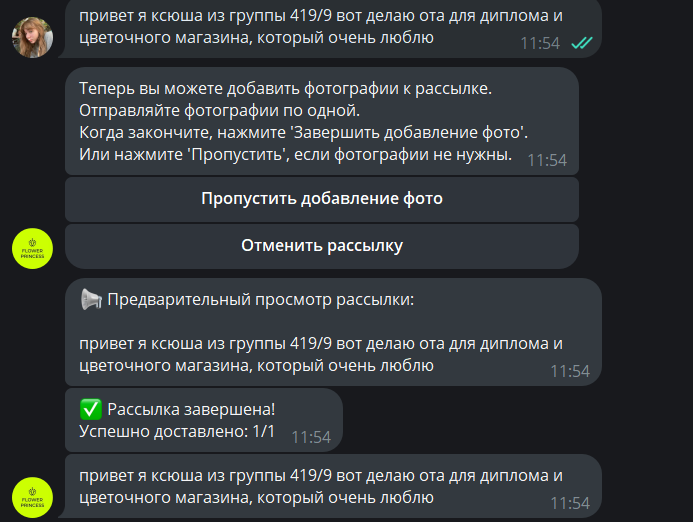


Рисунок – Тест 10, создание рассылки

1. Экономическая часть (переделать)
   1. Область применения программного продукта и его преимущества перед аналогичным программным продуктом

Разработанный Telegram-бот для цветочного магазина «Flower Princess» предназначен для автоматизации ключевых бизнес-процессов, таких как:

* Управление каталогом товаров (цветы, букеты, подарки);
* Взаимодействие с клиентами через удобный интерфейс мессенджера;
* Автоматическая рассылка уведомлений о новинках и акциях.

Преимущества перед аналогами:

* Простота и доступность: В отличие от сложных CRM-систем, бот не требует дополнительного обучения сотрудников и легко интегрируется в текущие процессы.
* Экономия времени: Автоматизация рутинных задач (рассылка, обновление каталога) сокращает трудозатраты.
* Гибкость: Решение адаптировано под специфику малого бизнеса, что делает его более эффективным по сравнению с универсальными системами.
* Низкая стоимость: Отсутствие абонентской платы и минимальные затраты на поддержку.
  1. Трудоемкость разработки программного продукта, квалификация исполнителя и его оклад

Разработка программного продукта велась одним разработчиком в течение 10 рабочих дней по 4 часа в день. Общая трудоёмкость составила 40 часов.

Таблица 3 – Трудоёмкость разработки программного продукта

| **НАИМЕНОВАНИЕ ЭТАПА** | **ОБОЗНАЧЕНИЕ** | **ТРУДОЁМКОСТЬ, Ч.** |
| --- | --- | --- |
| Постановка задачи | Тпз | 4 |
| Проектирование архитектуры | Тпа | 6 |
| Разработка интерфейса | Тпи | 8 |
| Программирование бизнес-логики | Трф | 12 |
| Тестирование и отладка | Ттс | 6 |
| Подготовка документации и развёртывание | Тд | 4 |
| Итого | Тобщ | 40 |

Таблица 4 – Разработчик программного продукта

|  |  |  |  |
| --- | --- | --- | --- |
| **ИСПОЛНИТЕЛЬ** | **СТАВКА ЗА СМЕНУ (4 Ч.)** | **КОЛ-ВО ДНЕЙ** | **СУММА, РУБ.** |
| Разработчик-программист | 1500 | 10 | 15 000 |

* 1. Расходы на технические и расходные средства

Таблица 5 – Стоимость технических средств

|  |  |  |  |
| --- | --- | --- | --- |
| **НАИМЕНОВАНИЕ** | **ЦЕНА, РУБ.** | **КОЛ-ВО** | **СУММА, РУБ.** |
| Ноутбук | 42 000 | 1 | 42 000 |
| Мышь | 1 500 | 1 | 1 500 |
| Принтер | 10 000 | 1 | 10 000 |
| Итого | — | — | 53 500 |

Таблица 6 – Затраты на расходные материалы

|  |  |  |  |
| --- | --- | --- | --- |
| **НАИМЕНОВАНИЕ** | **СТОИМОСТЬ** | **КОЛ-ВО** | **СУММА, РУБ.** |
| Интернет | 500 руб./мес. | 1 мес. | 500 |
| Электричество | 7,83 руб./КВт\*ч | 30 КВт\*ч | 234,9 |
| Бумага | 800 руб. | 1 уп. | 800 |
| Ручка | 50 руб. | 2 шт. | 100 |
| Картридж | 1200 руб. | 1 | 1200 |
| Итого | — | — | 2834,9 |

* 1. Расчёт затрат на разработку

Затраты на оплату труда с доплатами (1):

, ()

где, 115 000 – основная зарплата;

0,15 – коэффициент дополнительных выплат (премии, отпуск и пр.).

Страховые взносы (2):

(2)

где, 0,3 – норматив отчислений на страхование;

17 250 – сумма зарплаты с учётом доплат.

Затраты на расходные материалы с учётом доставки (3):

(3)

где, 0,13 – коэффициент транспортных расходов.

Накладные расходы (4):

, (4)

где, 0,56 – норматив накладных расходов на аренду, амортизацию, орг. поддержку.

Расчет полных затрат на разработку проектного решения (программного продукта) осуществляется по формуле (5):

(5)

Подставляем всё в формулу (6):

(6)

* 1. Расчёт цены, прибыли и налогов

Плановая прибыль (7):

, (7)

где, 0,3 – коэффициент рентабельности, обеспечивающий прибыль.

Налог на прибыль (8):

(8)

где, 0,25 – ставка налога на прибыль.

Чистая прибыль – это разность между прибылью и налогом на неё. (9):

(9)

Поступления в бюджет – это сумма всех налогов, выплачиваемых государству (10):

(10)

Таблица 7 – Итоговая таблица экономических результатов

|  |  |  |
| --- | --- | --- |
| **ПОКАЗАТЕЛЬ** | **ОБОЗНАЧЕНИЕ** | **ЗНАЧЕНИЕ, РУБ.** |
| Плановая прибыль | Ппл | 26 635,93 |
| НДС | НДС | 23 884,47 |
| Цена продукта | Цпп | 139 306,84 |
| Налог на прибыль | НП | 6 658,98 |
| Чистая прибыль | ЧП | 19 976,95 |
| Поступления в бюджет | ПБ | 30 543,45 |

Произведя вычисления, было выявлено, что чистая прибыль составит 19 976,95 рублей.

На разработку программного продукта было потрачено 88 786,44 рубля, а итоговая цена составила 139 306,84 рубля, что даёт чистую прибыль в 19 976,95 рубля.

Разработанный Telegram-бот обладает конкурентными преимуществами за счёт удобства использования, автоматизации заказов, простоты администрирования и адаптивности к нуждам малого бизнеса. Он может быть применён как в учебных целях, так и в рамках реальной деятельности небольших торговых точек, особенно цветочных магазинов.

1. Техника безопасности и охрана труда
   1. Анализ условий труда программиста (специалиста по информационным технологиям)

Рабочее место программиста – это часть складского помещения, где специалист проводит большую часть рабочего времени. Совокупность показателей технических и программных средств определяет качество и производительность работы, наряду с габаритными показателями мебели и её удобством.

Помещение для работы представляет собой зону длиной 5 м, шириной 4 м и высотой потолков 3 м. Общая площадь рабочей зоны составляет 20 м². Согласно санитарным правилам экран ноутбука нужно располагать на расстоянии 55-60 см от глаз пользователя, но не ближе 50 см. При использовании жидкокристаллического дисплея на рабочее место должно приходиться не менее 4,5 м² площади. На экран не должен попадать прямой солнечный свет во избежание бликов и повышенной нагрузки на зрение.

Для снижения нагрузки на технические средства, влекущей за собой перегрев ноутбука, используются программные средства, позволяющие снизить нагрузку на технические средства за счёт отказа в использование ненужных служб.

При идентификации вредных производственных факторов было установлено, что опасными могут являться:

* электромагнитные поля и излучения от технических средств;
* статическое электричество;
* статические перегрузки, вызываемые неподвижной работой за компьютером;
* перенапряжение органов зрения.

В целях выявления вредных или опасных факторов производственной среды и трудового процесса проводится специальная оценка условий на рабочих местах. Оценке подлежат все имеющиеся в организации рабочие места.

Работа, по специальной оценке, условий труда заключается в исследовании следующих факторов на рабочих местах организации:

* химические факторы;
* физические факторы;
* биологические факторы;
* тяжесть и напряженность трудового процесса.

Для уменьшения нагрузки на органы зрения, а также снятия мышечного напряжения во время рабочего дня проводится проветривание в течении 10-15 минут каждые 3 часа.

* 1. Расчёт искусственного освещения

Для качественной и безопасной работы важно обеспечить нормативную освещённость. Согласно СП 52.13330.2016 [10], минимальный уровень освещённости для работы с компьютером составляет 300 лк. Нормативные данные по освещённости и коэффициентам взяты из СП 52.13330.2016 «Естественное и искусственное освещение». Использованы Таблицы 5.1, 9 и 10.

Индекс помещения по формуле (1)

(1)

где a – длина помещения – 4 метра;

b – ширина помещения – 5 метров;

h1 – высота помещения – 3 метра;

h2 – высота рабочей поверхности – стандартно 0,8 метра.

Коэффициенты отражения:

* Потолок: 70% (бетон с побелкой);
* Стены: 50% (светло-серые);
* Пол: 20% (бетонное покрытие).

Определение коэффициента использования (U):

По таблице для светильника PTF/R UNI LED при:

φ = 1,01

Отражение: потолок 70%, стены 50%, пол 20%

Методом интерполяции между φ=1,0 и φ=1,25:

Для φ=1,0: U=69% (по таблице).

Для φ=1,25: U=78% (по таблице).

Значит для φ=1,01: U≈70% (0,7).

Для помещений общественных зданий с нормальными условиями: коэффициент запаса равен 1,4.

Формула для расчёта количества светильников (2):

, (2)

где N – количество светильников, шт;

Е – нормируемая освещенность, 300 лк, по Таблица 5.1 из СП 52.13330.2016 [10];

S – площадь помещения: = 20 м²;

Кз – коэффициент запаса = 1,5, Таблица 9 СП 52.13330.2016: для светодиодов при офисных условиях — 1,5;

Фл – световой поток одной лампы = 2900 лм;

n – количество ламп в светильнике = 1;

U – коэффициент использования = 0,7. По таблице 10 коэффициентов использования СП 52.13330.2016— зависит от индекса помещения.

Требуется установка 5 светодиодных светильников PTF/R UNI LED (33 Вт). Полученное количество светильников совпадает с числом светильников, установленных в помещении, что соответствует нормативным показателям искусственного освещения согласно СП 52.13330.2016 «Естественное и искусственное освещение».

* 1. Электробезопасность на предприятии

Общие положения: все сотрудники при поступлении на работу обязаны пройти вводный и первичный инструктажи по электробезопасности. Особое внимание уделяется правилам работы с электрооборудованием, присутствующим в конкретных рабочих зонах. Для программистов и сотрудников цветочного отдела перечень оборудования включает:

1. Основное оборудование,

* Ноутбук/компьютер с подставкой;
* Беспроводные устройства (мышь, клавиатура);
* Принтер/сканер/МФУ;
* Роутер и сетевое оборудование;
* Кассовый аппарат;
* Холодильник для цветов с системой вентиляции.

1. Вспомогательное оборудование.

* Удлинители и сетевые фильтры;
* Зарядные устройства;
* Осветительные приборы;
* Системы вентиляции.

Требования к организации рабочего места:

1. Электропроводка,

* Запрещается использование поврежденных проводов и розеток;
* Минимальное расстояние между проводами и отопительными приборами 0,5 метра.

Запрещается:

* Прокладывать провода под коврами;
* Вешать провода на гвозди;
* Использовать самодельные удлинители.

1. Размещение оборудования.

* Ноутбук должен располагаться на специальной подставке с вентиляционными отверстиями;
* Беспроводные устройства должны находиться в зоне уверенного приема сигнала;
* Принтер и другое периферийное оборудование размещается на расстоянии не менее 0,7 метра от основного рабочего места.

Правила эксплуатации оборудования:

1. Компьютерная техника,

Запрещается:

* Прикасаться к задней панели системного блока при включенном питании;
* Допускать попадание жидкости на устройства;
* Закрывать вентиляционные отверстия.

Требуется:

* Использовать сетевые фильтры;
* Обеспечивать свободную циркуляцию воздуха вокруг устройств.

1. Холодильник для цветов,

Ежедневная проверка:

* Исправность вентиляционной системы;
* Отсутствие наледи на охлаждающих элементах;
* Целостность изоляции проводов.

Запрещается:

* Устанавливать посторонние предметы на холодильник;
* Блокировать вентиляционные решетки.

1. Сетевое оборудование.

* Роутер должен быть установлен в хорошо вентилируемом месте;
* Запрещается накрывать оборудование;
* Требуется регулярная очистка от пыли.

Меры безопасности при работе:

1. Общие требования,

Запрещается:

* Самостоятельный ремонт оборудования;
* Использование неисправных устройств;
* Одновременное касание оборудования и заземленных предметов.

Обязательно:

* Отключать технику при длительном простое
* Использовать только сертифицированные зарядные устройства

1. Действия в аварийных ситуациях,

При обнаружении неисправности:

1. Немедленно отключить устройство от сети,
2. Сообщить ответственному лицу,
3. Вывесить предупредительный знак.

При поражении электрическим током:

1. Отключить источник питания,
2. Оказать первую помощь,
3. Вызвать медицинскую службу.

Соблюдение данных требований обеспечивает безопасную эксплуатацию электрооборудования и предотвращает электротравматизм на рабочем месте.

* 1. Пожарная безопасность на предприятии

Инструктаж по пожарной безопасности проводится согласно программе, утвержденной ответственным за охрану труда, с соблюдением всех действующих норм и правил. Его продолжительность определяется утвержденной программой. Обычно такой инструктаж совмещают с инструктажем по технике безопасности и проводят в аналогичные сроки.

Прохождение противопожарного инструктажа обязательно для всех сотрудников организации без исключения - постоянных и временных работников, командированных специалистов, практикантов и студентов, независимо от их образования, стажа или занимаемой должности.

Факт проведения инструктажа фиксируется в специальном журнале учета с обязательными подписями как инструктирующего, так и инструктируемого. Для внеплановых инструктажей в журнале обязательно указывается причина их проведения.

Важно отметить, что к работе допускаются только те сотрудники, которые прошли противопожарный инструктаж. При изменении характера работы или условий труда проводится дополнительное обучение по предупреждению и тушению пожаров в порядке, установленном руководством предприятия.

Первичный инструктаж должен длиться не менее одного часа. В ходе него сотрудники должны ознакомиться:

* с действующими на предприятии правилами пожарной безопасности и соответствующими инструкциями;
* с наиболее пожароопасными участками производства, где категорически запрещено курить и использовать открытый огонь;
* с потенциальными причинами возникновения пожаров и мерами их предотвращения;
* с практическими действиями при возникновении пожара: вызовом пожарной охраны, использованием первичных средств пожаротушения, расположением ближайшего телефона, а также правилами поведения при пожаре и эвакуации людей и материальных ценностей.

Во время первичного инструктажа ответственный сотрудник должен подробно рассказать о производственных установках с повышенной пожарной опасностью, мерах предотвращения возгораний, указать специально отведенные места для курения, ознакомить новых работников с имеющимися средствами пожаротушения, показать ближайший телефон и разъяснить правила поведения при пожаре.

Обязательным условием проведения противопожарного инструктажа является практическая демонстрация способов использования имеющихся на предприятии средств пожаротушения, включая огнетушители, противогазы, респираторы и другое оборудование.

Соблюдение всех перечисленных мероприятий способствует снижению утомляемости персонала, уменьшению травматизма, повышению производительности труда и созданию комфортных условий для трудовой деятельности.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения выпускной квалификационной работы была достигнута поставленная цель – разработка Telegram-бота для автоматизации работы цветочной лавки. В соответствии с целью были успешно решены все поставленные задачи:

* проведён анализ предметной области и существующих решений;
* выбраны инструменты и технологии реализации;
* спроектирована архитектура приложения;
* реализован Telegram-бот с разделением прав пользователей и администраторов;
* разработан удобный интерфейс команд и добавлены возможности взаимодействия с пользователями;
* проведено тестирование и оформлены результаты.

В результате был создан полностью функционирующий Telegram-бот, который обеспечивает базовые потребности бизнеса в автоматизации каталога, взаимодействии с клиентами и администрировании. Данный бот может быть внедрён в реальной среде.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Python 3.10 Documentation: офиц. сайт / Python Software Foundation. — URL: https://docs.python.org/3/ (дата обращения: 19.03.2025).
2. Aiogram 2.25.0 Documentation: офиц. сайт / Aiogram Developers. — URL: https://docs.aiogram.dev/en/latest/ (дата обращения: 19.03.2025).
3. SQLite Documentation: офиц. сайт / Richard Hipp. — URL: https://www.sqlite.org/docs.html (дата обращения: 19.03.2025).
4. Telegram Bot API Documentation: офиц. сайт / Telegram. — URL: https://core.telegram.org/bots/api (дата обращения: 19.03.2025).
5. Иванов, И. И. Разработка Telegram-ботов: практическое руководство / И. И. Иванов. — Москва: Пример, 2023. — 256 с.
6. Петров, П. П. Базы данных для начинающих / П. П. Петров. — Санкт-Петербург: Пример, 2022. — 320 с.
7. Smith, J. Building Scalable Telegram Bots / J. Smith // Journal of Software Engineering. — 2021. — № 2. — С. 45—56.
8. GitHub: репозиторий с примерами кода: сайт. — URL: https://github.com (дата обращения: 19.03.2025).
9. СП 52.13330.2016. Естественное и искусственное освещение: свод правил / Минстрой России. — Москва, 2016.
10. ГОСТ 12.1.019—79. Система стандартов безопасности труда. Электробезопасность. Общие требования и номенклатура видов защиты. — Москва: Стандартинформ, 1979.
11. ГОСТ 12.4.026—2015. Система стандартов безопасности труда. Цвета сигнальные, знаки безопасности и разметка сигнальная. — Москва: Стандартинформ, 2015.
12. Беляев, С. В. Информационные технологии в малом бизнесе / С. В. Беляев. — Москва: Инфра-М, 2021.
13. Сидоров, Д. А. Базы данных: проектирование, реализация, оптимизация / Д. А. Сидоров. — Санкт-Петербург: Питер, 2022.
14. Мельникова, Н. А. Проектирование интерфейсов чат-ботов / Н. А. Мельникова. — Москва: НИУ ВШЭ, 2020.
15. Федотов, И. Ю. Практика разработки Telegram-ботов / И. Ю. Федотов. — Казань: Лаборатория знаний, 2023.
16. Российская Федерация. Законы. О персональных данных: Федеральный закон № 152-ФЗ. — Москва, 2006.
17. Лебедев, А. И. Защита информации и кибербезопасность / А. И. Лебедев. — Москва: Академия, 2022.
18. Сафонов, К. М. Санитарно-гигиенические условия труда: учебник / К. М. Сафонов. — Москва: Юрайт, 2021.
19. PyPI – Python Package Index: сайт. — URL: https://pypi.org/ (дата обращения: 19.03.2025).

ПРИЛОЖЕНИЕ А

(справочное)

**Исходный код**

А.1 MAIN.PY

*import* asyncio

*import* logging

*import* os

*from* aiogram *import* Bot, Dispatcher, types

*from* aiogram.contrib.fsm\_storage.memory *import* MemoryStorage

*from* aiogram.dispatcher.middlewares *import* BaseMiddleware

*from* bot.config *import* TOKEN

*from* bot.utils *import* setup\_logging, log\_action

*from* bot.handlers *import* register\_all\_handlers

*from* bot.handlers.base *import* register\_handlers *as* register\_base\_handlers

*from* bot.admin\_handlers *import* register\_all\_handlers *as* register\_admin\_handlers

*from* bot.admin\_handlers.broadcast *import* register\_handlers *as* register\_broadcast\_handlers

*# Настройка логирования*

logging.basicConfig(

*level*=logging.INFO,

*format*='%(asctime)s - %(name)s - %(levelname)s - %(message)s'

)

logger = logging.getLogger(\_\_name\_\_)

*# Создаем директории для данных и логов*

os.makedirs("data", *exist\_ok*=True)

os.makedirs("logs", *exist\_ok*=True)

*# Настраиваем логирование*

setup\_logging()

*# Инициализация бота и диспетчера*

bot = Bot(*token*=TOKEN, *parse\_mode*='HTML')

storage = MemoryStorage()

dp = Dispatcher(bot, *storage*=storage)

*# Middleware для логирования всех callback\_query*

class LoggingMiddleware(BaseMiddleware):

    async def **on\_pre\_process\_callback\_query**(*self*, *callback\_query*: types.CallbackQuery, *data*: dict):

        logger.info(f"Получен callback\_query: {*callback\_query*.data} от пользователя {*callback\_query*.from\_user.id}")

async def **on\_startup**(*dp*: Dispatcher):

    """Действия при запуске бота"""

    me = *await* bot.get\_me()

    logger.info(f"Бот @{me.username} запущен и готов к работе!")

    log\_action(f"Бот @{me.username} запущен и готов к работе!")

async def **main**():

    """Главная функция"""

*try*:

*# Регистрация middleware*

        dp.middleware.setup(LoggingMiddleware())

*# Регистрация обработчиков*

        register\_admin\_handlers(dp)      *# Сначала регистрируем обработчики команд администратора*

        register\_broadcast\_handlers(dp)   *# Затем обработчики рассылки*

        register\_all\_handlers(dp)        *# Затем обработчики модулей*

        register\_base\_handlers(dp)       *# Затем базовые обработчики*

*# Запуск бота*

*await* on\_startup(dp)

        print("Бот запущен! Нажмите Ctrl+C для остановки")

*await* dp.start\_polling()

*finally*:

*await* bot.close()

*await* storage.close()

*if* \_\_name\_\_ == '\_\_main\_\_':

*# Запуск бота*

*try*:

        print("Запуск бота...")

        asyncio.run(main())

*except* (KeyboardInterrupt, SystemExit):

        print("\nБот остановлен!")

*except* Exception *as* e:

        logger.exception(f"Произошла ошибка: {e}")

А.2 database.py:

import sqlite3

import logging

from typing import Dict, List, Optional, Any

logger = logging.getLogger(\_\_name\_\_)

# Путь к базе данных

DB\_PATH = "flower\_shop.db"

def get\_db():

    """Создает подключение к базе данных"""

    conn = sqlite3.connect(DB\_PATH)

    conn.row\_factory = sqlite3.Row

    return conn

def init\_db():

    """Инициализирует базу данных"""

    conn = get\_db()

    c = conn.cursor()

    # Создаем таблицу пользователей

    c.execute('''CREATE TABLE IF NOT EXISTS users

                 (id INTEGER PRIMARY KEY,

                  username TEXT,

                  first\_name TEXT,

                  last\_name TEXT,

                  created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP)''')

    # Создаем таблицу цветов

    c.execute('''CREATE TABLE IF NOT EXISTS flowers

                 (id INTEGER PRIMARY KEY AUTOINCREMENT,

                  name TEXT NOT NULL,

                  description TEXT,

                  price REAL NOT NULL,

                  quantity INTEGER DEFAULT 0,

                  image\_url TEXT)''')

    # Создаем таблицу букетов

    c.execute('''CREATE TABLE IF NOT EXISTS bouquets

                 (id INTEGER PRIMARY KEY AUTOINCREMENT,

                  name TEXT NOT NULL,

                  description TEXT,

                  price REAL NOT NULL,

                  image\_url TEXT)''')

    # Создаем таблицу подарков

    c.execute('''CREATE TABLE IF NOT EXISTS gifts

                 (id INTEGER PRIMARY KEY AUTOINCREMENT,

                  name TEXT NOT NULL,

                  description TEXT,

                  price REAL NOT NULL,

                  image\_url TEXT)''')

    # Создаем таблицу контактов

    c.execute('''CREATE TABLE IF NOT EXISTS contacts

                 (id INTEGER PRIMARY KEY CHECK (id = 1),

                  phone TEXT,

                  address TEXT,

                  working\_hours TEXT)''')

    # Добавляем начальные контакты, если их нет

    c.execute('INSERT OR IGNORE INTO contacts (id, phone, address, working\_hours) VALUES (?, ?, ?, ?)',

              (1, '+7 (XXX) XXX-XX-XX', 'г. Санкт-Петербург, Караваевская 26к1', '9:00 - 21:00'))

    # Проверяем наличие столбца quantity в таблице flowers

    c.execute("PRAGMA table\_info(flowers)")

    columns = [column[1] for column in c.fetchall()]

    if 'quantity' not in columns:

        c.execute('ALTER TABLE flowers ADD COLUMN quantity INTEGER DEFAULT 0')

    conn.commit()

    conn.close()

# Функции для работы с пользователями

async def add\_user(user\_id: int, username: str = None, first\_name: str = None, last\_name: str = None) -> bool:

    """Добавляет нового пользователя"""

    try:

        conn = get\_db()

        c = conn.cursor()

        c.execute('''INSERT OR IGNORE INTO users (id, username, first\_name, last\_name)

                     VALUES (?, ?, ?, ?)''', (user\_id, username, first\_name, last\_name))

        conn.commit()

        conn.close()

        return True

    except Exception as e:

        logger.error(f"Ошибка при добавлении пользователя: {e}")

        return False

async def get\_all\_users() -> List[int]:

    """Возвращает список ID всех пользователей"""

    try:

        conn = get\_db()

        c = conn.cursor()

        c.execute('SELECT id FROM users')

        users = [row['id'] for row in c.fetchall()]

        conn.close()

        return users

    except Exception as e:

        logger.error(f"Ошибка при получении списка пользователей: {e}")

        return []

# Функции для работы с цветами

def get\_flowers() -> List[Dict]:

    """Возвращает список всех цветов"""

    conn = get\_db()

    c = conn.cursor()

    c.execute('SELECT \* FROM flowers')

    flowers = [dict(row) for row in c.fetchall()]

    conn.close()

    return flowers

def get\_flower\_by\_id(flower\_id: int) -> Optional[Dict]:

    """Возвращает цветок по ID"""

    conn = get\_db()

    c = conn.cursor()

    c.execute('SELECT \* FROM flowers WHERE id = ?', (flower\_id,))

    flower = c.fetchone()

    conn.close()

    return dict(flower) if flower else None

def add\_flower(flower\_data: Dict) -> bool:

    """Добавляет новый цветок"""

    try:

        conn = get\_db()

        c = conn.cursor()

        c.execute('''INSERT INTO flowers (name, description, price, quantity, image\_url)

                     VALUES (?, ?, ?, ?, ?)''',

                  (flower\_data['name'], flower\_data.get('description'),

                   flower\_data['price'], flower\_data.get('quantity', 0),

                   flower\_data.get('image\_url')))

        conn.commit()

        conn.close()

        return True

    except Exception as e:

        logger.error(f"Ошибка при добавлении цветка: {e}")

        return False

def update\_flower(flower\_id: int, flower\_data: Dict) -> bool:

    """Обновляет данные цветка"""

    try:

        conn = get\_db()

        c = conn.cursor()

        c.execute('''UPDATE flowers

                     SET name = ?, description = ?, price = ?, quantity = ?, image\_url = ?

                     WHERE id = ?''',

                  (flower\_data['name'], flower\_data.get('description'),

                   flower\_data['price'], flower\_data.get('quantity', 0),

                   flower\_data.get('image\_url'), flower\_id))

        conn.commit()

        conn.close()

        return True

    except Exception as e:

        logger.error(f"Ошибка при обновлении цветка: {e}")

        return False

def delete\_flower(flower\_id: int) -> bool:

    """Удаляет цветок"""

    try:

        conn = get\_db()

        c = conn.cursor()

        c.execute('DELETE FROM flowers WHERE id = ?', (flower\_id,))

        conn.commit()

        conn.close()

        return True

    except Exception as e:

        logger.error(f"Ошибка при удалении цветка: {e}")

        return False

def update\_flower\_quantity(flower\_id: int, quantity: int) -> bool:

    """Обновляет количество цветка"""

    try:

        conn = get\_db()

        c = conn.cursor()

        c.execute('UPDATE flowers SET quantity = ? WHERE id = ?', (quantity, flower\_id))

        conn.commit()

        conn.close()

        return True

    except Exception as e:

        logger.error(f"Ошибка при обновлении количества цветка: {e}")

        return False

# Функции для работы с букетами

def get\_bouquets() -> List[Dict]:

    """Возвращает список всех букетов"""

    conn = get\_db()

    c = conn.cursor()

    c.execute('SELECT \* FROM bouquets')

    bouquets = [dict(row) for row in c.fetchall()]

    conn.close()

    return bouquets

def get\_bouquet\_by\_id(bouquet\_id: int) -> Optional[Dict]:

    """Возвращает букет по ID"""

    conn = get\_db()

    c = conn.cursor()

    c.execute('SELECT \* FROM bouquets WHERE id = ?', (bouquet\_id,))

    bouquet = c.fetchone()

    conn.close()

    return dict(bouquet) if bouquet else None

def add\_bouquet(bouquet\_data: Dict) -> bool:

    """Добавляет новый букет"""

    try:

        conn = get\_db()

        c = conn.cursor()

        c.execute('''INSERT INTO bouquets (name, description, price, image\_url)

                     VALUES (?, ?, ?, ?)''',

                  (bouquet\_data['name'], bouquet\_data.get('description'),

                   bouquet\_data['price'], bouquet\_data.get('image\_url')))

        conn.commit()

        conn.close()

        return True

    except Exception as e:

        logger.error(f"Ошибка при добавлении букета: {e}")

        return False

def update\_bouquet(bouquet\_id: int, bouquet\_data: Dict) -> bool:

    """Обновляет данные букета"""

    try:

        conn = get\_db()

        c = conn.cursor()

        c.execute('''UPDATE bouquets

                     SET name = ?, description = ?, price = ?, image\_url = ?

                     WHERE id = ?''',

                  (bouquet\_data['name'], bouquet\_data.get('description'),

                   bouquet\_data['price'], bouquet\_data.get('image\_url'),

                   bouquet\_id))

        conn.commit()

        conn.close()

        return True

    except Exception as e:

        logger.error(f"Ошибка при обновлении букета: {e}")

        return False

def delete\_bouquet(bouquet\_id: int) -> bool:

    """Удаляет букет"""

    try:

        conn = get\_db()

        c = conn.cursor()

        c.execute('DELETE FROM bouquets WHERE id = ?', (bouquet\_id,))

        conn.commit()

        conn.close()

        return True

    except Exception as e:

        logger.error(f"Ошибка при удалении букета: {e}")

        return False

# Функции для работы с подарками

def get\_gifts() -> List[Dict]:

    """Возвращает список всех подарков"""

    conn = get\_db()

    c = conn.cursor()

    c.execute('SELECT \* FROM gifts')

    gifts = [dict(row) for row in c.fetchall()]

    conn.close()

    return gifts

def get\_gift\_by\_id(gift\_id: int) -> Optional[Dict]:

    """Возвращает подарок по ID"""

    conn = get\_db()

    c = conn.cursor()

    c.execute('SELECT \* FROM gifts WHERE id = ?', (gift\_id,))

    gift = c.fetchone()

    conn.close()

    return dict(gift) if gift else None

def add\_gift(gift\_data: Dict) -> bool:

    """Добавляет новый подарок"""

    try:

        conn = get\_db()

        c = conn.cursor()

        c.execute('''INSERT INTO gifts (name, description, price, image\_url)

                     VALUES (?, ?, ?, ?)''',

                  (gift\_data['name'], gift\_data.get('description'),

                   gift\_data['price'], gift\_data.get('image\_url')))

        conn.commit()

        conn.close()

        return True

    except Exception as e:

        logger.error(f"Ошибка при добавлении подарка: {e}")

        return False

def update\_gift(gift\_id: int, gift\_data: Dict) -> bool:

    """Обновляет данные подарка"""

    try:

        conn = get\_db()

        c = conn.cursor()

        c.execute('''UPDATE gifts

                     SET name = ?, description = ?, price = ?, image\_url = ?

                     WHERE id = ?''',

                  (gift\_data['name'], gift\_data.get('description'),

                   gift\_data['price'], gift\_data.get('image\_url'),

                   gift\_id))

        conn.commit()

        conn.close()

        return True

    except Exception as e:

        logger.error(f"Ошибка при обновлении подарка: {e}")

        return False

def delete\_gift(gift\_id: int) -> bool:

    """Удаляет подарок"""

    try:

        conn = get\_db()

        c = conn.cursor()

        c.execute('DELETE FROM gifts WHERE id = ?', (gift\_id,))

        conn.commit()

        conn.close()

        return True

    except Exception as e:

        logger.error(f"Ошибка при удалении подарка: {e}")

        return False

# Функции для работы с контактами

def get\_contacts() -> Dict:

    """Возвращает контактную информацию"""

    conn = get\_db()

    c = conn.cursor()

    c.execute('SELECT \* FROM contacts WHERE id = 1')

    contacts = c.fetchone()

    conn.close()

    return dict(contacts) if contacts else {

        'phone': '+7 (XXX) XXX-XX-XX',

        'address': 'г. Санкт-Петербург, Караваевская 26к1',

        'working\_hours': '9:00 - 21:00'

    }

def update\_contacts(contacts\_data: Dict) -> bool:

    """Обновляет контактную информацию"""

    try:

        conn = get\_db()

        c = conn.cursor()

        c.execute('''UPDATE contacts

                     SET phone = ?, address = ?, working\_hours = ?

                     WHERE id = 1''',

                  (contacts\_data.get('phone'),

                   contacts\_data.get('address'),

                   contacts\_data.get('working\_hours')))

        conn.commit()

        conn.close()

        return True

    except Exception as e:

        logger.error(f"Ошибка при обновлении контактов: {e}")

        return False

# Инициализируем базу данных при импорте

init\_db()

А.3 keyboards.py:

*from aiogram.types import InlineKeyboardMarkup, InlineKeyboardButton*

*from bot.config import ADMIN\_ID*

*def get\_main\_keyboard(user\_id: int = None) -> InlineKeyboardMarkup:*

*"""Создает главную клавиатуру с 4 кнопками"""*

*keyboard = InlineKeyboardMarkup(row\_width=2)*

*keyboard.add(*

*InlineKeyboardButton("🌸 Каталог букетов", callback\_data="bouquets"),*

*InlineKeyboardButton("💐 Каталог цветов", callback\_data="flowers"),*

*InlineKeyboardButton("🎁 Подарки", callback\_data="gifts"),*

*InlineKeyboardButton("📞 Контактная информация", callback\_data="contacts")*

*)*

*# Добавляем кнопку управления только для администратора*

*if user\_id == ADMIN\_ID:*

*keyboard.add(InlineKeyboardButton("⚙️ Управление", callback\_data="admin\_panel"))*

*return keyboard*

*def get\_admin\_panel\_keyboard() -> InlineKeyboardMarkup:*

*"""Создает клавиатуру для панели администратора"""*

*keyboard = InlineKeyboardMarkup(row\_width=2)*

*keyboard.add(*

*InlineKeyboardButton("🌸 Цветы", callback\_data="admin\_flower"),*

*InlineKeyboardButton("💐 Букеты", callback\_data="admin\_bouquet"),*

*InlineKeyboardButton("🎁 Подарки", callback\_data="admin\_gift")*

*)*

*keyboard.add(InlineKeyboardButton("📢 Создать рассылку", callback\_data="broadcast"))*

*keyboard.add(InlineKeyboardButton("🔙 В главное меню", callback\_data="back\_to\_main"))*

*return keyboard*

*def get\_back\_keyboard() -> InlineKeyboardMarkup:*

*"""Создает клавиатуру с кнопкой назад"""*

*keyboard = InlineKeyboardMarkup()*

*keyboard.add(InlineKeyboardButton("🔙 Назад", callback\_data="cmd\_start"))*

*return keyboard*

*# Клавиатуры для управления цветами*

*def get\_flower\_management\_keyboard() -> InlineKeyboardMarkup:*

*"""Создает клавиатуру для управления цветами"""*

*keyboard = InlineKeyboardMarkup(row\_width=2)*

*keyboard.add(*

*InlineKeyboardButton("➕ Добавить цветок", callback\_data="add\_flower"),*

*InlineKeyboardButton("📋 Список цветов", callback\_data="list\_flowers")*

*)*

*keyboard.add(InlineKeyboardButton("🔙 Назад", callback\_data="back\_to\_admin"))*

*return keyboard*

*# Клавиатуры для управления букетами*

*def get\_bouquet\_management\_keyboard() -> InlineKeyboardMarkup:*

*"""Создает клавиатуру для управления букетами"""*

*keyboard = InlineKeyboardMarkup(row\_width=2)*

*keyboard.add(*

*InlineKeyboardButton("➕ Добавить букет", callback\_data="add\_bouquet"),*

*InlineKeyboardButton("📋 Список букетов", callback\_data="list\_bouquets")*

*)*

*keyboard.add(InlineKeyboardButton("🔙 Назад", callback\_data="back\_to\_admin"))*

*return keyboard*

*# Клавиатуры для управления подарками*

*def get\_gift\_management\_keyboard() -> InlineKeyboardMarkup:*

*"""Создает клавиатуру для управления подарками"""*

*keyboard = InlineKeyboardMarkup(row\_width=2)*

*keyboard.add(*

*InlineKeyboardButton("➕ Добавить подарок", callback\_data="add\_gift"),*

*InlineKeyboardButton("📋 Список подарков", callback\_data="list\_gifts")*

*)*

*keyboard.add(InlineKeyboardButton("🔙 Назад", callback\_data="back\_to\_admin"))*

*return keyboard*

*# Клавиатура для управления контактами*

*def get\_contacts\_management\_keyboard() -> InlineKeyboardMarkup:*

*"""Создает клавиатуру для управления контактами"""*

*keyboard = InlineKeyboardMarkup(row\_width=1)*

*keyboard.add(*

*InlineKeyboardButton("📱 Изменить телефон", callback\_data="edit\_phone"),*

*InlineKeyboardButton("📍 Изменить адрес", callback\_data="edit\_address"),*

*InlineKeyboardButton("⏰ Изменить время работы", callback\_data="edit\_work\_hours")*

*)*

*keyboard.add(InlineKeyboardButton("🔙 Назад", callback\_data="back\_to\_admin"))*

*return keyboard*

ПРИЛОЖЕНИЕ Б

(справочное)

**Словарь данных**

Таблица Б. - Словарь базы данных

| **НАЗВАНИЕ** | **ТИП** | **NULL** | **ОГРАНИЧЕНИЯ** |
| --- | --- | --- | --- |
| Users | Таблица | - | Хранение информации о пользователях. |
| Id | INTEGER | NOT NULL | PRIMARY KEY. Уникальный идентификатор пользователя. |
| Username | TEXT | NULL | Имя пользователя в Telegram. |
| First\_name | TEXT | NULL | Имя пользователя. |
| Last\_name | TEXT | NULL | Фамилия пользователя. |
| Created\_at | TIMESTAMP | NOT NULL | DEFAULT CURRENT\_TIMESTAMP. Дата и время регистрации пользователя. |
| Flowers | Таблица | - | Хранение информации о цветах. |
| Id | INTEGER | NOT NULL | PRIMARY KEY. Уникальный идентификатор цветка. |
| Name | TEXT | NOT NULL | Название цветка |
| Description | TEXT | NOT NULL | Описание цветка |
| Price | REAL | NOT NULL | Стоимость цветка |
| Image\_url | TEXT | NULL | Ссылка на изображение цветка |
| Bouquets | Таблица | - | Хранение информации о букетах. |
| Id | INTEGER | NOT NULL | PRIMARY KEY. Уникальный идентификатор букета. |

Продолжение таблицы Б.1

|  |  |  |  |
| --- | --- | --- | --- |
| **НАЗВАНИЕ** | **ТИП** | **NULL** | **ОГРАНИЧЕНИЯ** |
| Name | TEXT | NOT NULL | Название букета |
| Description | TEXT | NOT NULL | Описание букета |
| Price | REAL | NOT NULL | Стоимость букета |
| Image\_url | TEXT | NULL | Ссылка на изображение букета |
| Gifts | Таблица | - | Хранение информации о подарках. |
| Id | INTEGER | NOT NULL | PRIMARY KEY. Уникальный идентификатор подарка. |
| Name | TEXT | NOT NULL | Название подарка |
| Description | TEXT | NOT NULL | Описание подарка |
| Price | REAL | NOT NULL | Стоимость подарка |
| Image\_url | TEXT | NULL | Ссылка на изображение подарка |
| Contacts | Таблица | - | Хранение контактной информации магазин. |
| Id | INTEGER | NOT NULL | PRIMARY KEY CHECK. Уникальный идентификатор контакта |
| Phone | TEXT | NULL | Номер телефона магазина. |
| Address | TEXT | NULL | Адрес магазина. |
| working\_hours | TEXT | NULL | Время работы магазина. |

ПРИЛОЖЕНИЕ В

(справочное)

**Тест-кейсы**

Таблица В. - Тест кейсы

| **№** | **НАЗВАНИЕ ТЕСТ-КЕЙСА** | **ПРЕДУСЛОВИЯ** | **ШАГИ** | **ОЖИДАЕМЫЙ РЕЗУЛЬТАТ** | **ФАКТИЧЕСКИЙ РЕЗУЛЬТАТ** |
| --- | --- | --- | --- | --- | --- |
| 1 | Проверка команды /start | Бот запущен, Пользователь не авторизован как Администратор | 1.Отправить команду /start в чат с ботом. | Бот отправляет приветственное сообщение: "👋 Добро пожаловать в наш цветочный магазин! Выберите интересующий вас раздел:" и отображает главное меню. (Рисунок 10). | Соответствует ожидаемому |
| 2 | Проверка команды /admin от администратора | Бот запущен, Пользователь имеет права Администратора | 1.Отправить команду /admin в чат с ботом. | Бот открывает меню администратора и отправляет сообщение: "👑 Панель администратора. Добро пожаловать, [имя администратора]! Выберите раздел для управления:". (Рисунок 11). | Соответствует ожидаемому |

Продолжение таблицы В.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **НАЗВАНИЕ ТЕСТ-КЕЙСА** | **ПРЕДУСЛОВИЯ** | **ШАГИ** | **ОЖИДАЕМЫЙ РЕЗУЛЬТАТ** | **ФАКТИЧЕСКИЙ РЕЗУЛЬТАТ** |
| 3 | Проверка команды /admin от обычного Пользователя | Бот запущен, Пользователь не является Администратором | 1.Отправить команду /admin в чат с ботом. | Бот отправляет сообщение: "У вас нет прав для доступа к панели администратора." (Рисунок 12). | Соответствует ожидаемому |
| 4 | Проверка команды /log от Администратора | Бот запущен, Пользователь является Администратором | 1.Отправить команду /log в чат с ботом. | Бот отправляет файл с логами.  (Рисунок 13). | Соответствует ожидаемому |
| 5 | Проверка команды /log от обычного Пользователя | Бот запущен, Пользователь не является Администратором | 1.Отправить команду /log в чат с ботом. | Бот отправляет сообщение: "У вас нет прав для доступа к логам." (Рисунок 14). | Соответствует ожидаемому |
| 6 | Проверка команды /db от администратора | Бот запущен, Пользователь является Администратором | 1.Отправить команду /db в чат с ботом. | Бот отправляет файл с базой данных.  (Рисунок 15). | Соответствует ожидаемому |
| 7 | Проверка команды /db от обычного пользователя | Бот запущен, Пользователь не является Администратором | 1.Отправить команду /db в чат с ботом. | Бот отправляет сообщение: "У вас нет прав администратора." (Рисунок 16). | Соответствует ожидаемому |

Продолжение таблицы В.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **НАЗВАНИЕ ТЕСТ-КЕЙСА** | **ПРЕДУСЛОВИЯ** | **ШАГИ** | **ОЖИДАЕМЫЙ РЕЗУЛЬТАТ** | **ФАКТИЧЕСКИЙ РЕЗУЛЬТАТ** |
| 8 | Проверка добавления нового букета через панель администратора | Бот запущен, Пользователь является Администратором | 1.Открыть панель администратора. Ввести данные нового букета. 3. Подтвердить добавление. | Бот отправляет сообщение: "Букет '[название цветка]' успешно добавлен!" и добавляет товар в каталог (Рисунок 17-18). | Соответствует ожидаемому |
| 9 | Проверка удаления цветка через панель администратора | Бот запущен, Пользователь является Администратором, цветок присутствует в каталоге | 1.Открыть панель администратора. 2.Выбрать цветок для удаления. 3. Подтвердить удаление. | Бот отправляет сообщение: "Цветок '[название цветка]' удален." и удаляет цветок из базы данных (Рисунок 19). | Соответствует ожидаемому |
| 10 | Проверка создания рассылки через панель администратора | Бот запущен, Пользователь является Администратором, есть подписчики | 1.Открыть панель администратора. 2. Создать новое сообщение для рассылки. 3. Подтвердить отправку. | Бот отправляет сообщение: "✅ Рассылка завершена! Успешно доставлено: [количество]/[общее количество пользователей]" (Рисунок 20). | Соответствует ожидаемому |