**СОДЕРЖАНИЕ**

[Термины и определения 3](#_Toc199411988)

[Перечень сокращений и определений 4](#_Toc199411989)

[Введение 5](#_Toc199411990)

[1 Общая часть 7](#_Toc199411991)

[1.1 Анализ предметной области 7](#_Toc199411992)

[1.2 Постановка задачи 8](#_Toc199411993)

[1.3 Требования к системе 8](#_Toc199411994)

[1.4 Анализ существующих решений 9](#_Toc199411995)

[2 Специальная часть 12](#_Toc199411997)

[2.1 Проектирование системы 12](#_Toc199411998)

[2.2 Реализация системы 18](#_Toc199411999)

[2.3 Тестирование системы 22](#_Toc199412000)

[3 Экономическая часть (переделать) 27](#_Toc199412001)

[4 Техника безопасности и охрана труда 31](#_Toc199412007)

[4.1 Анализ условий труда программиста 31](#_Toc199412008)

[4.2 Расчёт искусственного освещения 32](#_Toc199412009)

[4.3 Электробезопасность на предприятии 34](#_Toc199412010)

[4.4 Пожарная безопасность на предприятии 36](#_Toc199412011)

[Заключение 39](#_Toc199412012)

[Список использованных источников 40](#_Toc199412013)

[Приложение А (справочное) Исходные коды 42](#_Toc199412014)

[Приложение Б (справочное) Словарь данных 51](#_Toc199412015)

[Приложение В (справочное) Тейсмт-Кейсы 53](#_Toc199412016)

**ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ**

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

Таблица 1 – 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 2 – 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 условиях стремительного развития информационных технологий предприятия сталкиваются с необходимостью автоматизации рутинных процессов для повышения эффективности и конкурентоспособности. Развитие информационных технологий оказывает значительное влияние на эффективность бизнес-процессов [1]. Одним из таких процессов является взаимодействие с клиентами, включая рассылку новостей, акций и предложений. На предприятии магазина цветов «FLOWER PRINCESS» текущий процесс информирования клиентов осуществляется вручную: сотрудники составляют сообщения, формируют списки клиентов в таблице Exel, составляют рассылки и отправляют информацию через различные каналы связи. Этот процесс требует значительных временных и трудовых ресурсов, а также подвержен риску человеческих ошибок. В среднем, подготовка и отправка одной рассылки занимает до 2 часов рабочего времени, включая составление текста, проверку актуальности клиентской базы и саму отправку сообщений. Кроме того, отсутствие централизованной системы хранения данных приводит к дублированию информации и уменьшает эффективность работы продавца магазина.

Для решения указанных проблем было принято решение о разработке Telegram-бота, который автоматизирует процесс рассылки информации клиентам. Telegram предоставляет удобный и популярный канал коммуникации, а использование бота позволяет автоматизировать процессы, снизить нагрузку на сотрудников и повысить оперативность взаимодействия с клиентами. Разработка бота осуществляется с учётом требований заказчика, включающих:

* Автоматическую рассылку новостей, акций и предложений клиентам через Telegram;
* Централизованное хранение и управление клиентской базой данных;
* Возможность анализа эффективности рассылок (открытия, переходы по ссылкам и т.д.);
* Обеспечение безопасности и конфиденциальности данных клиентов.

В качестве основных инструментов разработки выбран язык программирования Python и асинхронный фреймворк aiogram версии 2.25.1[2], который обеспечивает эффективную работу с Telegram Bot API и позволяет реализовать необходимые функции бота. Выбор данных инструментов обусловлен их широким распространением, поддержкой сообщества и наличием подробной документации. Проектирование системы включает в себя разработку архитектуры бота, определение структуры базы данных, а также создание пользовательского интерфейса для администрирования рассылок. На этапе реализации осуществляется программирование всех необходимых функций, интеграция с Telegram и настройка базы данных. После завершения разработки проводится тестирование системы для выявления и устранения возможных ошибок.

Внедрение разработанного Telegram-бота позволит автоматизировать процесс рассылки информации клиентам, снизить временные и трудовые затраты, повысить точность и своевременность отправляемых сообщений, а также обеспечить централизованное управление клиентской базой данных. Это, в свою очередь, приведёт к повышению эффективности бизнес-процессов предприятия и улучшению качества обслуживания клиентов.

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;
* Экономическая часть – расчёт стоимости проекта заказчику;
* Техника безопасности и охрана труда – основные требования безопасности, санитарные правила и нормы СанПиН;
* 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, а клиентская база ведется в таблицах Exel, что затрачивает очень много времени сотрудников магазина. Ежемесячно на рассылки уходит минимум 40 часов рабочего времени, что эквивалентно 10 000 руб. затрат на зарплату.

Цветочные магазины занимаются продажей цветов, букетов и сопутствующих товаров, таких как подарочные упаковки и мягкие игрушки. Средняя стоимость готовых CRM-решений для малого бизнеса составляет от 20 000 до 100 000 руб. в год, что неоправданно дорого для небольшого магазина. Современные технологии позволяют автоматизировать эти процессы, повысив эффективность и удобство работы. Интеграция с Telegram — бесплатный и популярный канал связи, а также позволяет полностью использовать гибкость и адаптивность под специфику бизнеса. 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.

Для реализации программного обеспечения были выбраны следующие технологии:

* Python 3.10 вместе с Aiogram 2.25.1 — данный стек выбран за асинхронную модель работы, что обеспечивает высокую производительность при взаимодействии с Telegram Bot API. Также фреймворк Aiogram имеет понятный синтаксис и низкий порог входа, что упростит поддержку и доработку проекта в будущем даже без привлечения высококвалифицированных специалистов. Кроме того, официальная документация предоставляет исчерпывающие рекомендации по архитектуре и реализации бота, что позволило выстроить масштабируемую структуру.
* SQLite — в качестве базы данных выбран движок SQLite, так как он не требует отдельного сервера и может работать в виде файла, встроенного в проект. Это особенно актуально для небольших компаний с ограниченным бюджетом и отсутствием выделенного IT-персонала. Заказчик отдельно отметил, что важно минимизировать затраты на инфраструктуру, поэтому от использования полноценных серверных СУБД (таких как PostgreSQL или MySQL) было решено отказаться в пользу более простой и автономной SQLite.

Таким образом, выбранные технологии соответствуют требованиям проекта: они позволяют обеспечить стабильную работу Telegram-бота, упрощают внедрение и снижают стоимость поддержки решения.

* 1. Архитектура приложения

Проектирование информационных систем требует учета множества факторов, включая архитектуру и бизнес-логику [3].

Проектируемая система включает:

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

Telegram-бот 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.

* 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 [4] с использованием библиотеки Aiogram [5] 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,

Базы данных являются основой для хранения и обработки информации в современных приложениях [6]. Для хранения данных используется реляционная база данных SQLite, 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.

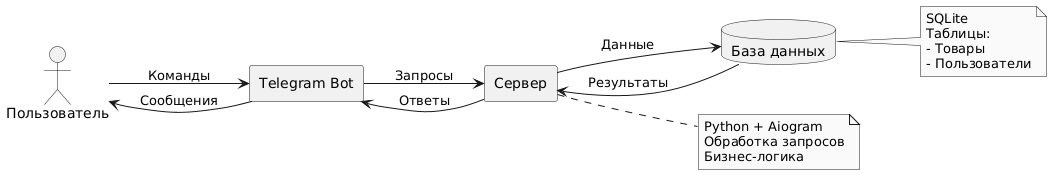


Рисунок 1 – Схема клиент-серверного взаимодействия

1. Специальная часть
   1. Проектирование системы

Проектирование программных продуктов требует системного подхода и тщательного планирования [7]. На этапе проектирования были разработаны диаграмма вариантов использования, диаграмма последовательностей, диаграмма активности и диаграмма классов.

Диаграмма вариантов использования (Рисунок 2) отражает требования к разделению ролей: администратор управляет каталогом, пользователь просматривает товары.

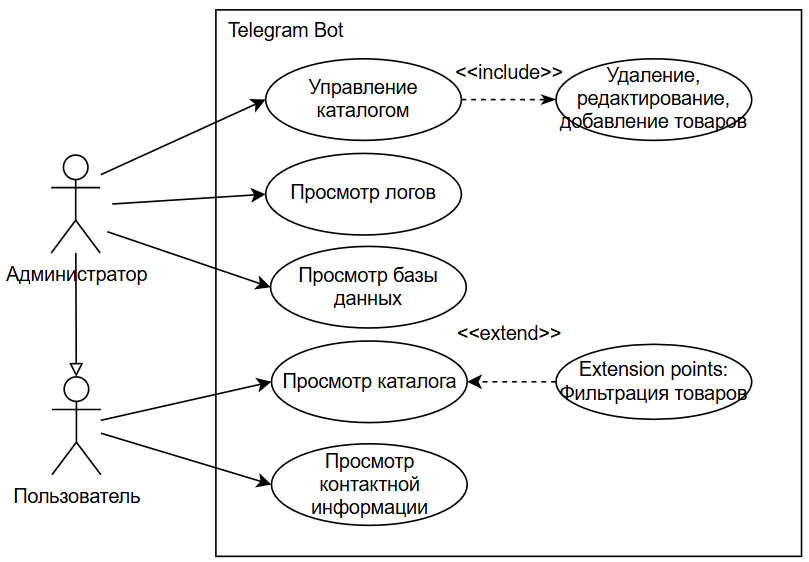


Рисунок 2 – Диаграмма вариантов использования

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.

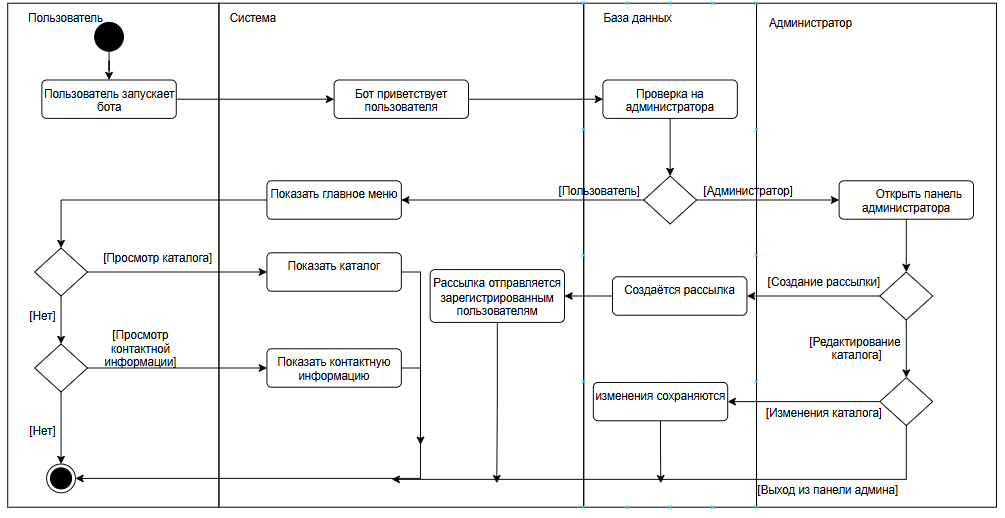


Рисунок 3 – Диаграмма активности

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.

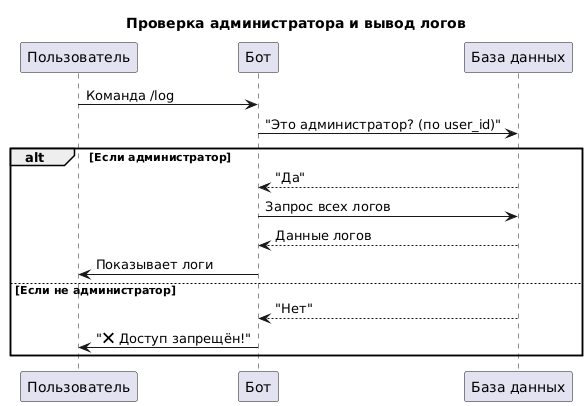


Рисунок 4 – Диаграмма последовательности

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 [8]), 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.

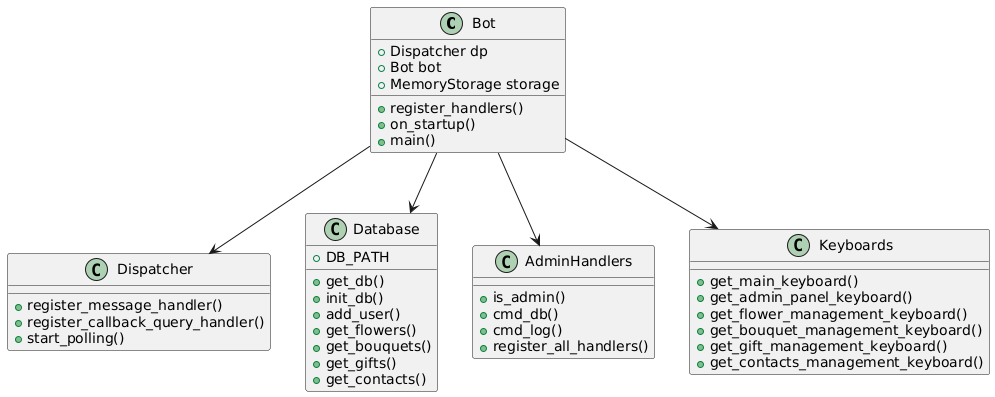


Рисунок 5 – Диаграмма классов

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

Архитектура Telegram-бота разработана с учетом рекомендаций официальной документации Telegram и принципов построения масштабируемых ботов [9].

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. Словарь базы данных описывает все типы и ограничения данных в таблицах (Таблица Б.1).

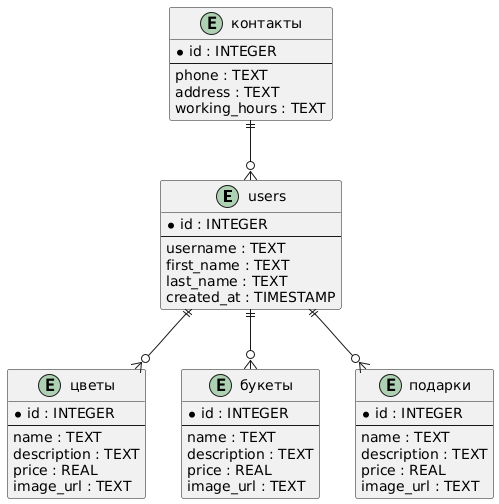


Рисунок 6 – Схема базы данных

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 [10]:

* Язык программирования: Python 3.10;
* Библиотека для работы с Telegram API [11]: Aiogram 2.25.1;
* База данных: SQLite [12].

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.

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

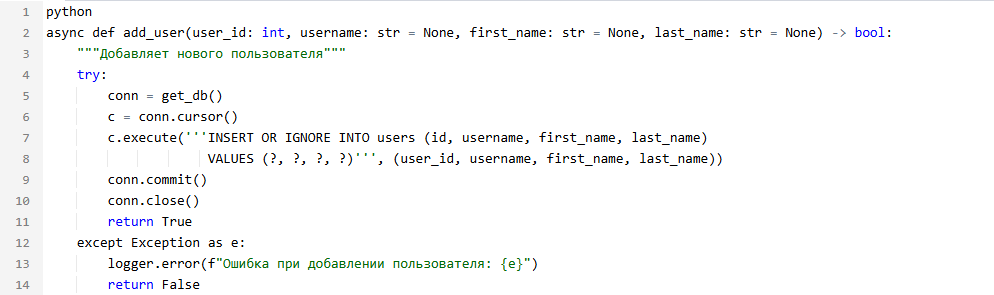


Рисунок 7 – Добавление нового пользователя

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.

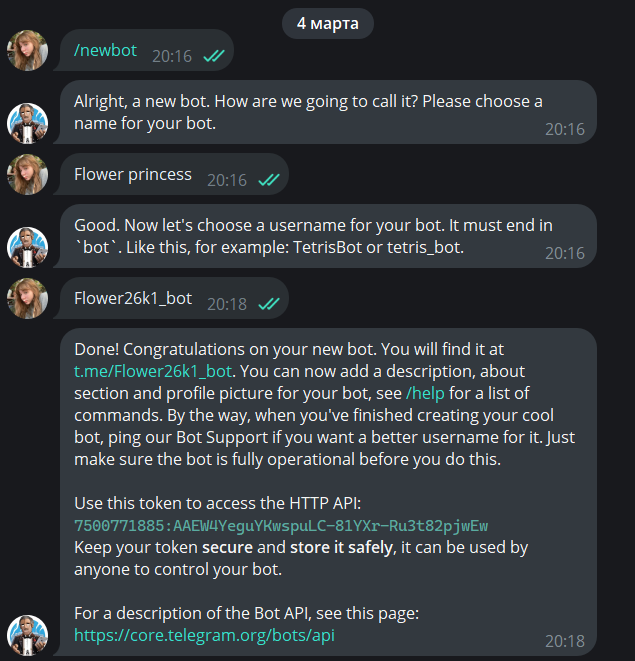


Рисунок 8 – Создание бота и получение токена

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 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 (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.

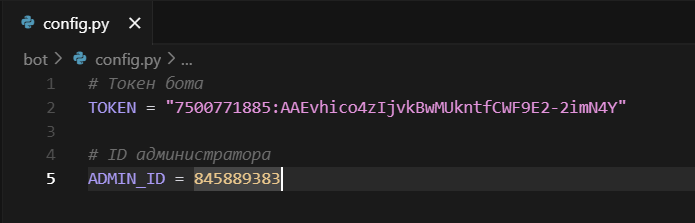


Рисунок 9 – Указание ID человека, который является администратором

* 1. Тестирование системы

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

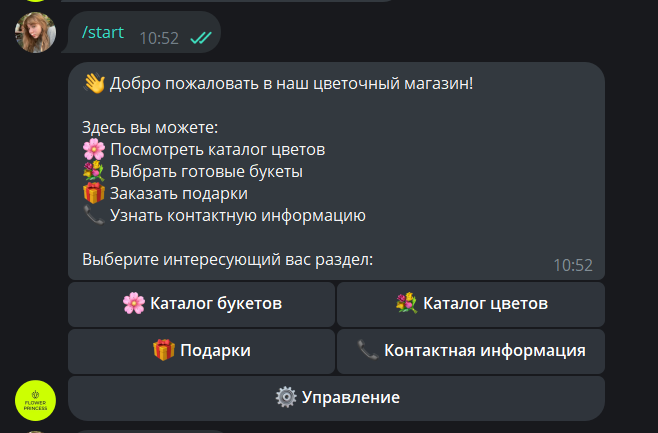


Рисунок 10 – Тест 1, проверка команды /start

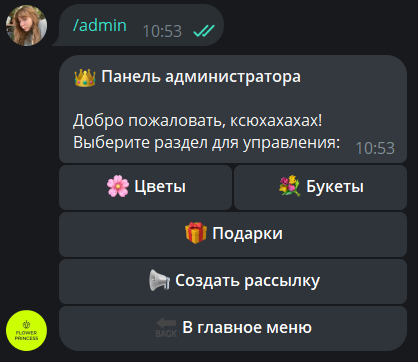


Рисунок 11 – Тест 2, проверка команды /admin Администратором

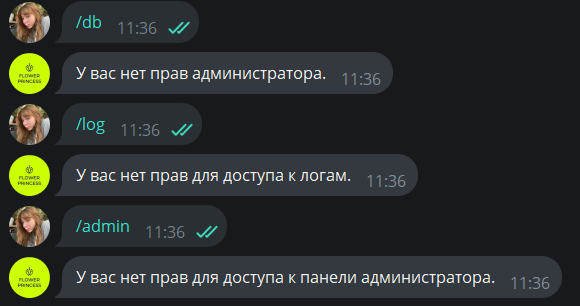


Рисунок 12 – Тест 3, проверка команды /admin Пользователем

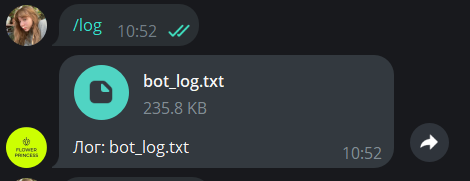


Рисунок 13 – Тест 4, проверка команды /log Администратором

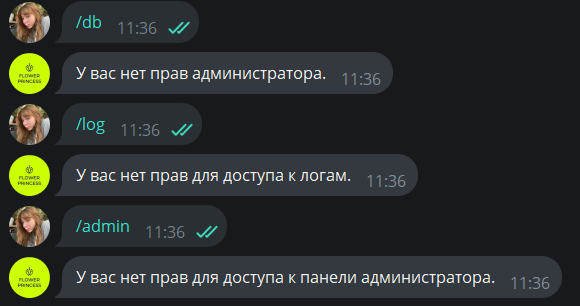


Рисунок 14 – Тест 5, проверка команды /log Пользователем

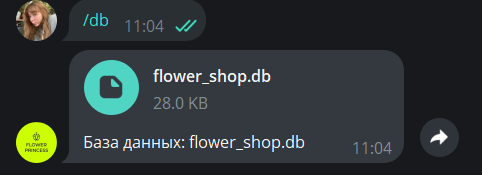


Рисунок 15 – Тест 6, проверка команды /db Администратором

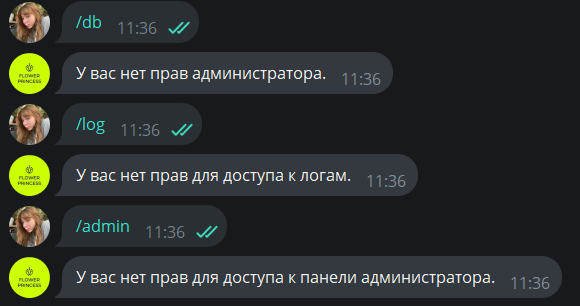


Рисунок 16 – Тест 7, проверка команды /db Пользователем

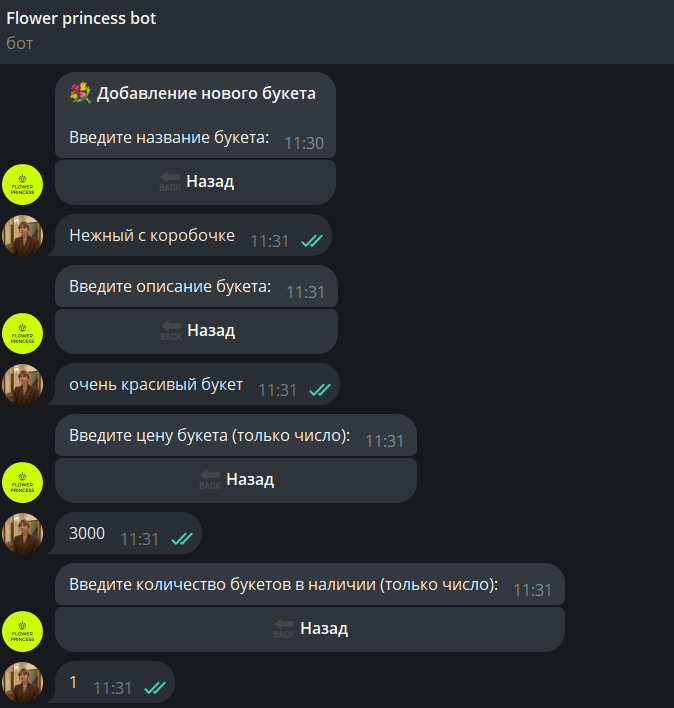


Рисунок 17 – Тест 8, добавление букета

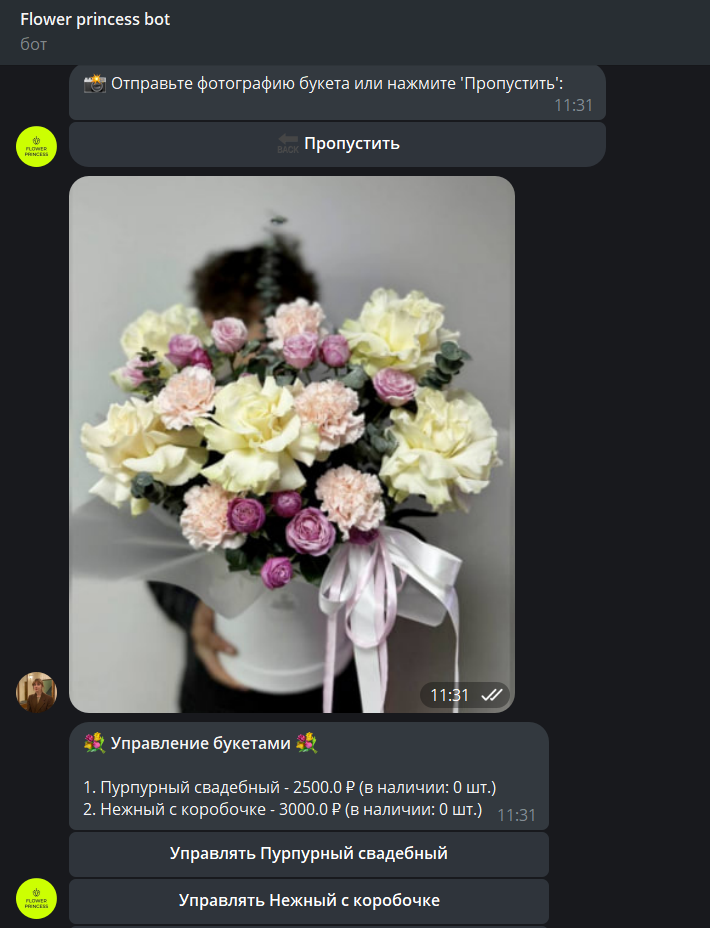


Рисунок 18 – Тест 8, добавление букета

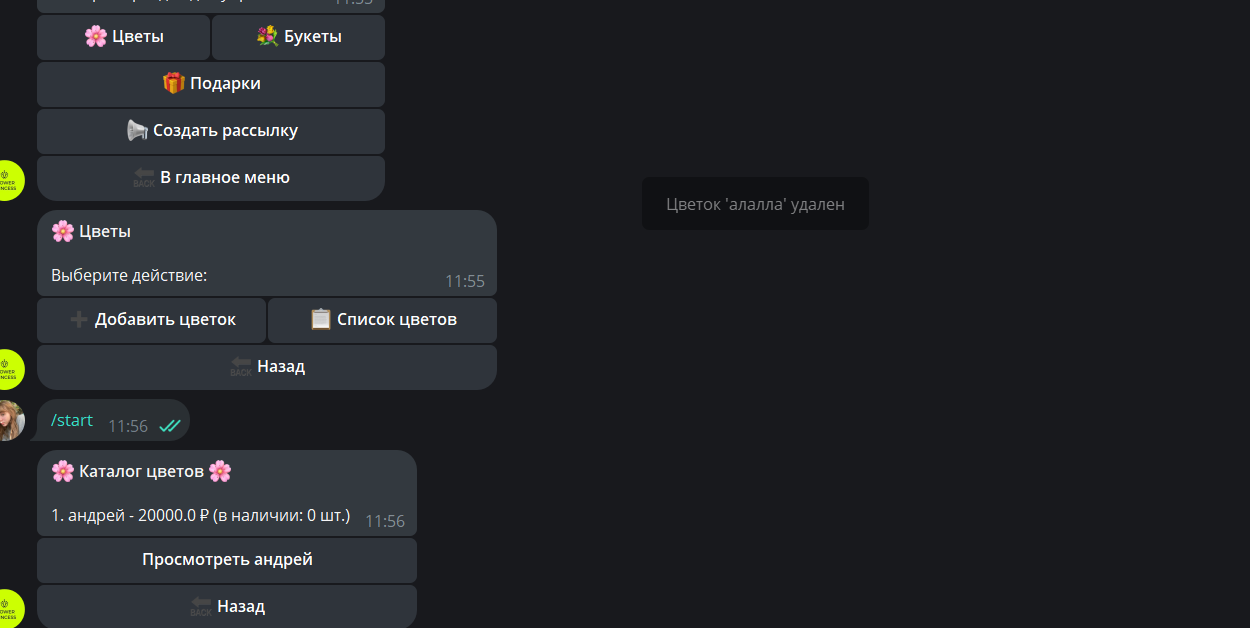


Рисунок 19 – Тест 9, удаление цветка

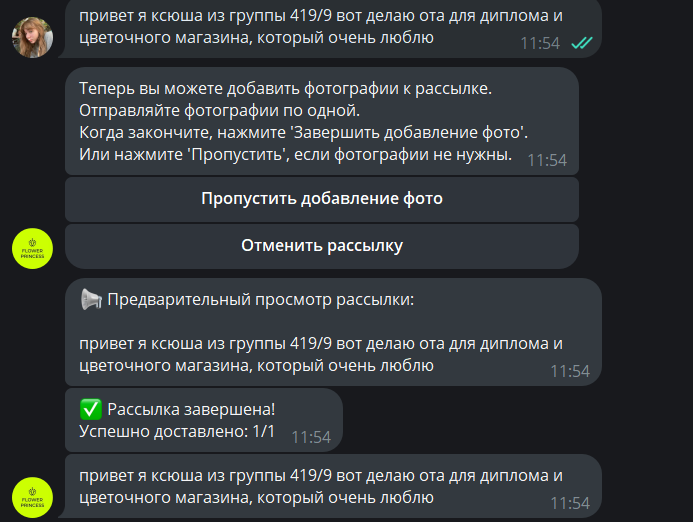


Рисунок 20 – Тест 10, создание рассылки

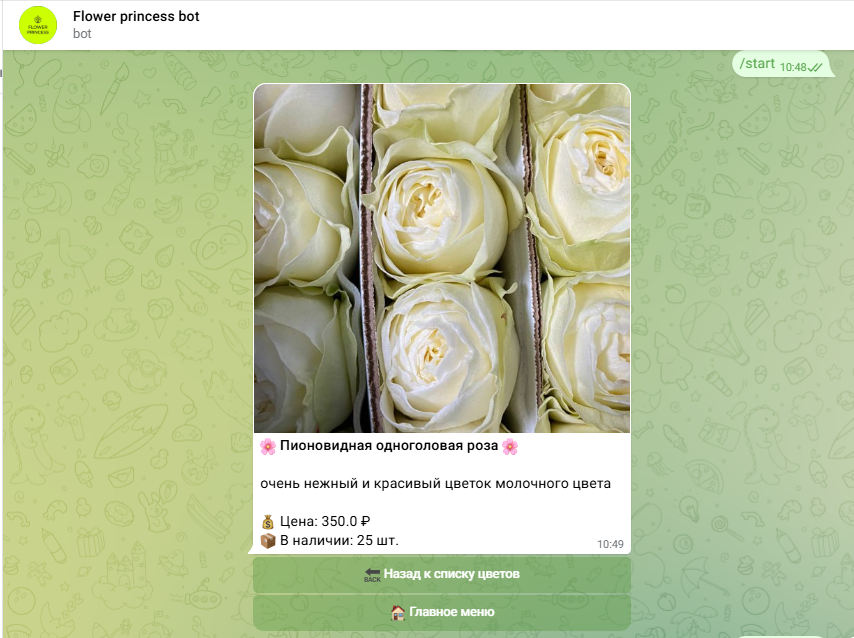


Рисунок 21 – Тест 11, просмотр каталога от лица клиента

Тест-кейсы (Приложение В) подтвердили выполнение 100% функциональных требований.

1. Экономическая часть
   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. Соблюдение санитарных норм и правил обеспечивает безопасность труда работников [10].

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 цветочного магазина. 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.

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

Разработанное решение позволило автоматизировать более 90% ручных процессов, связанных с работой с клиентами и рассылкой информации. Если ранее на подготовку и отправку рассылок уходило более 40 часов в месяц, то теперь этот процесс занимает не более 2 часов, включая проверку базы и настройку контента.

Таким образом, Telegram-бот готов к полноценному внедрению в реальную рабочую среду предприятия и способен обеспечить стабильную и эффективную автоматизацию ключевых процессов.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Савинков В.И. Современные информационные технологии: учебное пособие [Текст] / В.И. Савинков. – М.: Форум, 2019. – 352 с.
2. Root Junior A. aiogram: асинхронный фреймворк для Telegram Bot API. Документация к версии 2.25.1 [Электронный ресурс] / A. Root Junior – Проверено: 19.03.2025. Доступно по адресу: [https://docs.aiogram.dev/en/v2.25.1/.](https://docs.aiogram.dev/en/v2.25.1/)
3. Чаругин В.Н. Проектирование информационных систем: учебник [Текст] / В.Н. Чаругин. – М.: КноРус, 2020. – 368 с.
4. Кулешов И.А. Основы работы с Python [Электронный ресурс] / И.А. Кулешов – Проверено: 19.03.2025. [https://pythonworld.ru](https://pythonworld.ru/)
5. Amvera. Telegram Боты на Aiogram 3.x: Первые Шаги [Электронный ресурс] – Проверено: 19.03.2025. <https://habr.com/ru/companies/amvera/articles/820527/>
6. Дьяконов В.П. Основы работы с базами данных [Текст] / В.П. Дьяконов. – СПб.: Питер, 2020. – 240 с.
7. Фролов Е.С. Основы проектирования программных продуктов [Текст] / Е.С. Фролов. – СПб.: БХВ-Петербург, 2021. – 292 с.
8. Яценко А. Чат-боты в Telegram на Python и Aiogram: пишем первого бота [Электронный ресурс] / А. Яценко – Проверено: 19.03.2025. <https://skillbox.ru/media/code/chatboty-v-telegram-na-python-chast-1/>
9. Telegram FZ-LLC. Telegram Bot API: официальная документация [Электронный ресурс] / Telegram FZ-LLC – Проверено: 19.03.2025. https://core.telegram.org/bots/api
10. Pavlikova N. Create Telegram Bot with Python and Aiogram [Электронный ресурс] / N. Pavlikova – Проверено: 19.03.2025. https://www.udemy.com/course/create-telegram-bot-with-python-and-aiogram/
11. Чекмарёв А.С. Telegram Bot API: разработка и примеры [Электронный ресурс] / А.С. Чекмарёв – Проверено: 19.03.2025. <https://habr.com/ru/articles/675232>
12. Гришин А.В. Работа с СУБД SQLite [Электронный ресурс] / А.В. Гришин – Проверено: 19.03.2025. <https://metanit.com/sql/sqlite>
13. Кузнецов С.Н. Методика расчета экономической эффективности [Текст] / С.Н. Кузнецов. – М.: Экономика, 2020. – 216 с.
14. Власов С.И. Технико-экономическое обоснование IT-проектов [Электронный ресурс] / С.И. Власов – Проверено: 19.03.2025. <https://studref.com/632929>
15. Санитарные правила и нормы СанПиН [2.2.4.33](https://2.2.4.33/)59-16 "Гигиенические требования к условиям труда работников" [Электронный ресурс] / Федеральная служба по надзору в сфере защиты прав потребителей и благополучия человека – Проверено: 22.05.2025. <https://www.consultant.ru/document/cons_doc_LAW_202169/>

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.1 - 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.1 - 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 |

Продолжение таблицы В.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **НАЗВАНИЕ ТЕСТ-КЕЙСА** | **ПРЕДУСЛОВИЯ** | **ШАГИ** | **ОЖИДАЕМЫЙ РЕЗУЛЬТАТ** | **ФАКТИЧЕСКИЙ РЕЗУЛЬТАТ** |
| 11 | Проверка работы каталога от лица пользователя | Бот запущен, Пользователь, каталог не пустой | 1.Отправить команду /start в чат с ботом. 2. Выбрать категорию и товар. | Бот отправляет Пользователю сообщение: "[Название], [Описание], [Цена], [Кол-во]. (Рисунок 21). | Соответствует ожидаемому |