КОМИТЕТ ПО ОБРАЗОВАНИЮ ПРАВИТЕЛЬСТВА САНКТ-ПЕТЕРБУРГА

**САНКТ- ПЕТЕРБУРГСКОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ   
ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ   
«КОЛЛЕДЖ ЭЛЕКТРОНИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»**

ОТЧЁТ

**ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

Специальность 09.02.07 «Информационные системы и программирование»

Квалификация «Программист»

**ПМ 01 РАЗРАБОТКА МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ КОМПЬЮТЕРНЫХ СИСТЕМ**

Выполнил:

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Буркин

Группа 21

Наименование профильной организации: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ООО «ИТ В СЗ»

Руководитель практики от СПб ГБПОУ КИТ:\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.С. Михайлова

Санкт-Петербург 2025

**Содержание**

[Введение 3](#_Toc193800729)

[1. Формирование алгоритмов разработки программных модулей в соответствии с техническим заданием 4](#_Toc193800730)

[2. Разработка программных модулей в соответствии с техническим заданием 8](#_Toc193800731)

[3. Выполнение отладки программных модулей с использованием специализированных программных средств 9](#_Toc193800732)

[4. Выполнение тестирования программных модулей 12](#_Toc193800733)

[5. Осуществление рефакторинга и оптимизации программного кода 13](#_Toc193800734)

[6. Разработка модулей программного обеспечения для ассистента 15](#_Toc193800735)

[Заключение 19](#_Toc193800736)

[Список литературы 20](#_Toc193800737)

# Введение

Производственная практика является важным этапом в становлении студентов как начинающих специалистов в профессиональной среде, где все полученные знания используются и в дальнейшем развиваются. Цель производственной практики — освоение вида профессиональной деятельности, формирование профессиональных и общих компетенций, а также приобретение необходимых умений и опыта практической работы.

Задачи и опыт работы:

* формировал алгоритмы разработки программных модулей в соответствии с техническим заданием;
* разрабатывал программные модули в соответствии с техническим заданием для проекта на python с использованием Sql в среде Pycharm.
* выполнял отладку программных модулей с использованием специализированных программных средств и тестирование программных модулей с помощью pytest;
* осуществлял рефакторинг и оптимизацию программного кода для улучшения его производительности;
* разрабатывал модули программного обеспечения для системы технической поддержки, подключенной к хостингу;

В рамках производственной практики я работала над проектом, направленным на создание системы технической поддержки с использованием Python и SQL. Проект был реализован в среде PyCharm, что позволило мне получить практический опыт в разработке программных модулей для технической поддержки. Работа в ООО «ИТ В СЗ» стала ценным опытом для моего профессионального роста и развития.

# Формирование алгоритмов разработки программных модулей в соответствии с техническим заданием

Ассистент для Telegram разработан для автоматизации обработки заявок пользователей, предоставления справочной информации и передачи сложных запросов специалистам. Реализация включает интеграцию с CRM-системами, базами данных и обеспечение стабильной работы под высокой нагрузкой.

2. Функциональные требования

2.1 Основные функции

Приём обращений:

* текстовые сообщения, фото, файлы;

Кнопочное меню с обязательными вопросами (например, категория проблемы, контактные данные).

Маршрутизация запросов:

* автоматическая категоризация обращений по ключевым словам;
* передача сложных запросов специалистам через интеграцию с CRM (Битрикс24) или email;

Хранение истории:

* запись всех обращений в базу данных postgresql с метками времени и статусами;

Оповещения:

* уведомления пользователей о смене статуса заявки («принято», «в работе», «решено»);

2.2 Взаимодействие с пользователем

* главное меню с кнопками: «создать заявку», «проверить статус», «справочная информация»;
* динамические формы для сбора обязательных данных (например, описание проблемы, приоритет);

Обратная связь:

* возможность прикрепления скриншотов и документов;
* автоматические ответы при отсутствии операторов;

2.3 Панель администратора

Управление заявками:

* просмотр/фильтрация обращений по статусам, категориям, датам;
* ручное перенаправление запросов между операторами;

Настройки:

* редактирование списка обязательных вопросов;
* добавление/удаление операторов поддержки;

Аналитика:

* генерация отчётов по количеству обращений, среднему времени решения;

3. Нефункциональные требования

Производительность:

* обработка до 1000 запросов в минуту (с использованием асинхронной библиотеки aiogram).

Надёжность:

* резервное копирование базы данных ежечасно;
* мониторинг доступности через healthcheck-запросы;

Безопасность:

* шифрование конфиденциальных данных (логины, токены) через aws kms;
* валидация входящих файлов на запрещённые форматы;

4. Интеграции

CRM-система:

* синхронизация с битрикс24 через rest api: создание сделок, обновление статусов;

Уведомления:

* отправка email-оповещений через smtp-сервер;

База данных:

* хранение истории обращений в postgresql с использованием orm sqlalchemy;

5. Технологический стек

Язык: Python 3.10 + Aiogram.

Библиотеки:

* psycopg2 (для PostgreSQL);
* requests (для API Битрикс24);
* python-dotenv (для управления переменными окружения);

База данных:

* postgresql;
* таблицы users, tickets, attachments, logs.

Хостинг:

* развёртывание на heroku с использованием веб-хуков telegram;
* резервный сервер на aws ec2;

6. Развёртывание и сопровождение

* настройка веб-хука telegram через ngrok для локального тестирования;
* миграция базы данных с использованием alembic;
* запуск бота на heroku с автоматическим деплоем из git-репозитория;

Обслуживание:

* еженедельное обновление зависимостей;
* мониторинг ошибок через sentry;

7. Контроль качества

* юнит-тесты для обработчиков сообщений (например, проверка корректности парсинга введённых данных);
* интеграционные тесты с моками api битрикс24 и telegram;

Логирование:

* запись всех действий в elasticsearch для анализа инцидентов;

CI/CD:

* автоматический запуск тестов при пуше в ветку main (github actions);

8. Дополнительные возможности

* кластеризация обращений с помощью ml-алгоритмов (библиотека scikit-learn);
* поддержка мультиязычности;
* динамическое переключение языков через json-конфиги;

# 2. Разработка программных модулей в соответствии с техническим заданием

**1. Декомпозиция требований ТЗ**

* + разработал структуру бота с использованием библиотеки python-telegram-bot, которая поддерживает асинхронное программирование (asyncio);
  + реализовал систему состояний (states) для управления процессом создания заявки: выбор конфигурации, ввод данных (организация, фио, телефон, описание), обработка вложений;
  + настроил базу данных sqlite для хранения заявок (таблица tickets), администраторов (admins) и отзывов (feedback);
  + реализовал отправку email-уведомлений администратору с использованием smtplib, включая прикрепление файлов (вложения пользователя).
  + разработал панель администратора для управления заявками: просмотр новых, в работе и решенных заявок, изменение статуса заявок;
  + добавил систему обратной связи: после решения заявки пользователь может оценить качество поддержки (от 1 до 5);

Создан функциональный Telegram-бот, который соответствует техническому заданию. Пользователи могут создавать заявки, администраторы получают уведомления по email, а также могут управлять заявками через панель администратора.



Рисунок 1 – «Процесс создания заявки»

# 3. Выполнение отладки программных модулей с использованием специализированных программных средств

**Цель:**

Обеспечить стабильную работу Telegram-ассистента, устранив ошибки в коде.

**Выполненные задачи:**

Использовал библиотеку logging для записи логов (уровень INFO и ERROR), что позволило отслеживать выполнение программы и выявлять ошибки. Например, логировал успешную отправку email и ошибки при прикреплении файлов.

Отладил обработку callback-запросов в функции button\_click: устранил ошибку с некорректным парсингом callback\_data для оценки заявки (rate\_), добавив проверку на количество элементов в split('\_').

Исправил проблему с некорректной валидацией номера телефона в функции handle\_input: добавил проверку на длину номера и наличие только цифр после символа "+".

Отладил работу с базой данных SQLite: устранил проблему с некорректным закрытием соединений, что приводило к блокировке базы данных при параллельных запросах.

Использовал PyCharm для отладки: установил точки останова (breakpoints) в функциях handle\_media и send\_email, чтобы проверить корректность обработки вложений и отправки email.

**Результат:**

Устранены основные ошибки, связанные с обработкой пользовательского ввода, взаимодействием с базой данных и отправкой email. Бот работает стабильно, без сбоев при выполнении ключевых функций.

Использованные инструменты:

PyCharm (отладчик), logging, SQLite Browser (для проверки базы данных).

Пример отладки в функции button\_click изображен на рисунке 2.

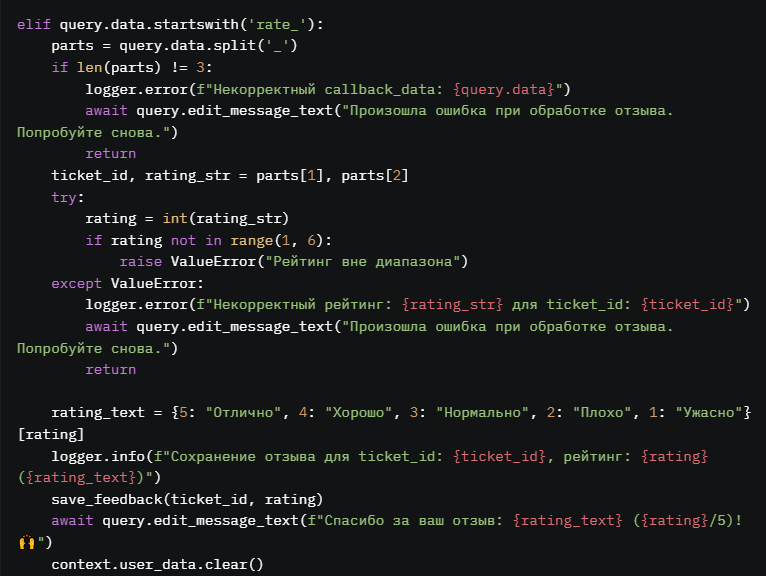


Рисунок 2 – «Проверка корректности данных»

# Выполнение тестирования программных модулей

**Цель:**

Проверить корректность работы всех функций Telegram-бота и убедиться, что он соответствует требованиям.

Выполненные задачи:

* провел модульное тестирование функции save\_ticket: проверил, что данные корректно сохраняются в базу данных sqlite, включая все поля (user\_id, config, org\_dept, name, phone, description);
* тестировал функцию send\_email: отправлял тестовые заявки с разным количеством вложений (0, 1, 3), чтобы убедиться, что email отправляется без ошибок, а вложения корректно прикрепляются;
* проверил работу функции handle\_media: отправлял различные типы вложений (фото, видео, документы, стикеры), чтобы убедиться, что они корректно обрабатываются и добавляются в описание заявки;
* тестировал сценарий с максимальным количеством вложений (max\_attachments = 3): убедился, что после достижения лимита пользователь получает соответствующее сообщение;
* провел тестирование ux: проверил, что пользовательский ввод (например, некорректный номер телефона) обрабатывается с выводом понятных сообщений об ошибке;
* тестировал панель администратора: проверил, что заявки отображаются в правильных категориях ("принято", "в работе", "решено") и статусы обновляются корректно;

**Результат:**

Все модули бота работают корректно: заявки создаются, отправляются по email, вложения обрабатываются, панель администратора функционирует без ошибок. Пользовательский интерфейс интуитивно понятен, ошибки ввода обрабатываются корректно.

Использованные инструменты:

1. Ручное тестирование через Telegram, SQLite Browser, Gmail (для проверки email).

Пример функции save\_ticket (тестирование сохранения заявки) предоставлен на рисунке 3:

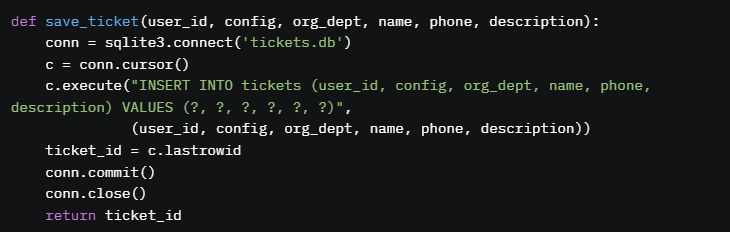


Рисунок 3 – «Проверка корректности всех полей и сохранение»

# Осуществление рефакторинга и оптимизации программного кода

**Цель:**

Улучшить читаемость, производительность и пользовательский опыт (UX) Telegram-бота.

**Выполненные задачи:**

* провел рефакторинг функции handle\_media: изначально бот отправлял три последовательных сообщения при добавлении вложений, что ухудшало ux. я изменил логику, чтобы бот отправлял одно сообщение и редактировал его с помощью edit\_message\_text, используя сохраненный message\_id в context.user\_data['description\_message\_id']. это сократило количество сообщений и улучшило восприятие интерфейса;
* оптимизировал функцию send\_email: добавил обработку исключений для удаления временных файлов (finally), чтобы избежать накопления файлов на диске в случае ошибки;
* рефакторил функцию button\_click: вынес логику обработки конфигураций в словарь config\_map, что сделало код более читаемым и упростило добавление новых конфигураций;
* улучшил читаемость кода: добавил комментарии к ключевым функциям (start, handle\_input, handle\_media), разбил длинные функции на более мелкие блоки;
* оптимизировал работу с базой данных: добавил индексы для таблицы tickets по полю status, что ускорило запросы в функции get\_tickets\_by\_status;
* улучшил обработку ошибок: добавил логирование и пользовательские сообщения для случаев, когда пользователь вводит некорректные данные (например, некорректный номер телефона);

**Результат:**

Код стал более читаемым и поддерживаемым, производительность бота улучшилась за счет оптимизации запросов к базе данных. Пользовательский опыт стал лучше благодаря замене множественных сообщений на редактирование одного сообщения.

Использованные инструменты:

* PyCharm (рефакторинг), SQLite Browser (оптимизация базы данных).

Пример рефакторинга в функции handle\_media (оптимизация UX) предоставлен на рисунке 4.

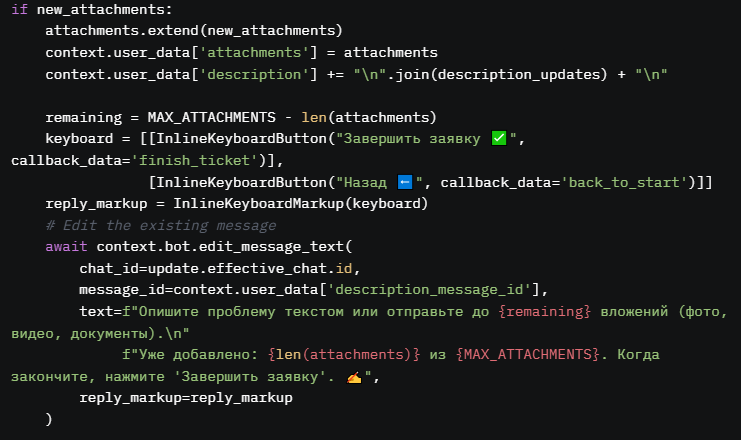


Рисунок 4 – «Замена отправки новых сообщений»

# Разработка модулей программного обеспечения для ассистента

Разработка программных модулей для ассистентов, таких как Telegram-боты, является важным аспектом создания эффективных и масштабируемых систем автоматизации. Модульное проектирование позволяет разбить функционал ассистента на независимые компоненты, каждый из которых решает конкретную задачу, что упрощает поддержку, тестирование и расширение возможностей бота.

1. Типы модулей в ассистентах

1.1 Модули функций (Feature Modules)

* отвечают за конкретные функции ассистента, такие как создание заявки, обработка пользовательского ввода или отправка уведомлений;
* позволяют разрабатывать и тестировать функции независимо друг от друга, что ускоряет процесс разработки и снижает риск ошибок;

1.2 Модули плагинов (Plugin Modules)

* предоставляют общую логику, которая используется в нескольких модулях функций, например, работа с базой данных или отправка email-уведомлений;
* упрощают управление общим кодом и облегчают его обновление без влияния на другие части системы;

1.3 Модули утилит (Utility Modules)

* содержат повторно используемый код, такой как логирование, обработка ошибок или форматирование данных;
* позволяют избежать дублирования кода и способствуют его многократному использованию в разных частях проекта;

2. Преимущества модульного проектирования

* упрощает добавление новых функций, таких как новые команды или интеграции с другими сервисами;
* снижает затраты на разработку за счет повторного использования кода (например, модуль отправки email);
* позволяет более тщательно тестировать и отлаживать отдельные модули, минимизируя ошибки в системе;

3. Инструменты и технологии

* python и python-telegram-bot: основной стек для разработки telegram-бота с поддержкой асинхронного программирования (asyncio);
* используется для хранения данных (заявок, отзывов, администраторов);
* для отправки email-уведомлений администратору;
* для разработки, отладки и рефакторинга кода;

4. Лучшие практики

* независимость модулей: каждый модуль (например, обработка заявок или отправка email) должен быть самодостаточным и не зависеть от других;
* стандартизация: использование единых соглашений по именованию функций и переменных, а также документирование кода;
* тестирование: регулярное тестирование каждого модуля (например, тестирование функции сохранения заявки в базу данных);

Пример реализации предоставлен на рисунке 5-6.

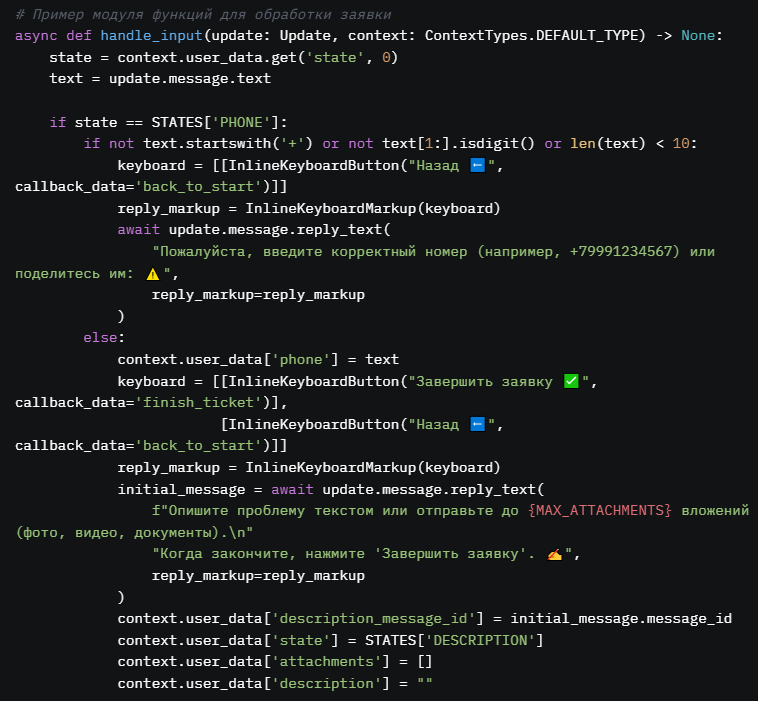


Рисунок 5 – «Реализация логики»



Рисунок 6 – «Реализация логики»

Модульное проектирование позволило создать гибкого и масштабируемого Telegram-бота, который эффективно решает задачи автоматизации техподдержки. Разделение на модули функций, плагинов и утилит упростило разработку, тестирование и поддержку проекта, что особенно важно для ассистентов с большим количеством функций.

# Заключение

В заключение хочу отметить, что производственная практика стала важным этапом в моем профессиональном развитии, позволив мне применить полученные теоретические знания в реальной разработке программного обеспечения. Целью практики было освоение вида профессиональной деятельности по ПМ01, формирование профессиональных и общих компетенций, а также приобретение необходимых умений и опыта практической работы.

В ходе практики я выполнил следующие задачи:

* сформировал алгоритмы разработки программных модулей в соответствии с техническим заданием;
* разработал программные модули для telegram-бота в соответствии с техническим заданием;
* выполнил отладку программных модулей с использованием специализированных программных средств;
* провел тестирование программных модулей;
* осуществил рефакторинг и оптимизацию программного кода;
* разработал модули программного обеспечения для ассистентов (telegram-бота);

В рамках самостоятельной работы я был вовлечен в разработку Telegram-бота для автоматизации техподдержки, что позволило мне углубить навыки программирования на Python, работы с базами данных SQLite, асинхронного программирования с использованием библиотеки python-telegram-bot, а также отправки email-уведомлений через smtplib. Этот опыт дал мне практические навыки в разработке, отладке, тестировании и оптимизации программного обеспечения, а также научил эффективно организовывать модульную структуру проекта. Работа над проектом, который решает реальные задачи автоматизации, стала ценным опытом для моего профессионального роста и развития, укрепив мои компетенции в области разработки программного обеспечения и взаимодействия с пользователями через современные платформы, такие как Telegram.

# Список литературы

1. «Fluent Python: Clear, Concise, and Effective Programming» — Luciano Ramalho. 2022 год.
2. «Learning Python» — Mark Lutz. 2013 год.
3. «PHP and MySQL Web Development» — Luke Welling, Laura Thomson. 2016 год.
4. «Python 3 Reference Manual» — Guido van Rossum, Fred L. Drake. 2009 год.
5. «Python in easy steps» — Mike McGrath. 2018 год.