**Разработка и внедрение системы автоматизации для ускорения и улучшения обработки заказов.**

Программа: Разработчик Data - Engineer

Технологическая специализация

Поплавский Сергей Валериевич

г. Нижний Новгород

2025 г.

Оглавление

[**1.** **Введение:** 3](#_Toc197771568)

[**2.** **Теоретическая часть** 4](#_Toc197771569)

[**3.** **Анализ существующих решений в области автоматизации обработки заказов** 4](#_Toc197771570)

[**1.** **Ключевые инсайты для разработки** 5](#_Toc197771571)

[**2.** **Проектирование архитектуры решения** 7](#_Toc197771572)

[**2.1.** **Общая архитектура системы** 7](#_Toc197771573)

[**2.2.** **Компоненты** **системы** 7](#_Toc197771574)

[**2.3.** **Функциональные** **возможности** 8](#_Toc197771575)

[**2.3.1.1.** **Основные функции системы:** 8](#_Toc197771576)

[**2.3.1.2.** **Модуль управления записями:** 8](#_Toc197771577)

[**2.3.1.3.** **Административный модуль:** 8](#_Toc197771578)

[**2.4.** **Взаимодействие компонентов** 8](#_Toc197771579)

[**2.4.1.1.** **Процесс взаимодействия компонентов:** 8](#_Toc197771580)

[**3.** **Обзор библиотек и инструментов, используемых в проекте** 10](#_Toc197771581)

[**3.1.** **Основные библиотеки** 10](#_Toc197771582)

[**3.2.** **Вспомогательные инструменты** 10](#_Toc197771583)

[**3.3.** **Системные инструменты** 10](#_Toc197771584)

[**3.4.** **Дополнительные инструменты** 11](#_Toc197771585)

[**3.5.** **Специализированные инструменты** 11](#_Toc197771586)

[**3.6.** **Интеграция и взаимодействие** 12](#_Toc197771587)

[**4.** **Практическая часть** 13](#_Toc197771588)

[**4.1.** **Описание разработанного приложения** 13](#_Toc197771589)

[**4.1.1.1.** **Общее описание приложения** 13](#_Toc197771590)

[**4.1.1.2.** **Основные функции приложения** 13](#_Toc197771591)

[**4.1.1.3.** **Интерфейс пользователя** 13](#_Toc197771592)

[**4.1.1.4.** **Преимущества** 14](#_Toc197771593)

[**4.2.** **Структура проекта** 14](#_Toc197771594)

[**4.3.** **Взаимодействие между различными компонентами приложения** 14](#_Toc197771595)

[**4.3.1.1.** **Модуль Downloads** 14](#_Toc197771596)

[**4.2.1.2.** **Модуль Initialization** 16](#_Toc197771601)

[**4.2.1.3.** **Модуль Master** 18](#_Toc197771609)

[**4.2.1.4.** **Модуль Paths** 19](#_Toc197771620)

[**4.2.1.5.** **Модуль Services** 20](#_Toc197771621)

[**4.2.1.6.** **Модуль Schedule** 22](#_Toc197771622)

[**4.3.** **Описание взаимодействия компонентов телеграмм бота** 23](#_Toc197771623)

[**4.3.1.2.** **Общее описание бота** 23](#_Toc197771624)

[**1.1.1.1.** **Обработчик команды Start** 24](#_Toc197771625)

[**1.1.1.2.** **Обработчик нажатие кнопок главного меню** 25](#_Toc197771626)

[**1.1.1.3.** **Кнопка регистрации** 25](#_Toc197771627)

[**1.1.1.4.** **Кнопка авторизации** 26](#_Toc197771628)

[**1.1.1.5.** **Кнопка смены пароля** 28](#_Toc197771629)

[**1.1.1.6.** **Обработчик запросов из меню для авторизованных пользователей** 29](#_Toc197771630)

[**1.1.1.7.** **Обработчик нажатия кнопки регистрации** 29](#_Toc197771631)

[**1.1.1.8.** **Обработчик нажатия кнопки просмотра записей** 32](#_Toc197771632)

[**1.1.1.9.** **Обработчик нажатия кнопки удаления записи** 32](#_Toc197771633)

[**5.** **Тестирование разработанного решения** 35](#_Toc197771640)

[**5.1.** **Тестирование системы управления автокалендарём** 35](#_Toc197771641)

[**5.2.** **Тестирование меню установки MySQL** 40](#_Toc197771642)

[**5.3.** **Тестирование модуля настройки службы MySQL и создания базы данных** 45](#_Toc197771643)

[**5.4.** **Тестирование модуля управления списком мастеров** 50](#_Toc197771644)

[**5.5.** **Тестирование модуля управления списком услуг** 56](#_Toc197771645)

[**5.6.** **Тестирование модуля управления рабочим расписанием** 62](#_Toc197771646)

[**5.7.** **Тестирование Telegram-бота системы управления записями** 68](#_Toc197771647)

[**6.** **Заключение:** 74](#_Toc197771648)

[**Список используемой литературы и источников** 77](#_Toc197771649)

[**7.** **Приложения** 0](#_Toc197771650)

# **Введение:**

В современных условиях ведения бизнеса эффективность обработки заказов является одним из ключевых факторов успеха компании. Система автоматизации обработки заказов представляет собой комплексное программное решение, направленное на оптимизацию бизнес-процессов, связанных с приемом, обработкой и выполнением заказов клиентов.

Актуальность темы обусловлена растущей потребностью предприятий в повышении скорости обработки заказов и снижении количества ошибок при их выполнении. В условиях высокой конкуренции компании вынуждены искать новые способы оптимизации бизнес-процессов, что делает разработку автоматизированных систем особенно востребованной.

Целью проекта является разработка и внедрение системы автоматизации обработки заказов, которая позволит существенно ускорить процесс обработки заказов и повысить качество обслуживания клиентов.

**Задачи:**

* Анализ существующих решений в области автоматизации обработки заказов
* Проектирование архитектуры системы
* Разработка и тестирование программного обеспечения
* Внедрение системы
* Оценка эффективности внедренного решения

**Инструменты разработки:** Git, PyCharm, MySQL

**Состав команды:** Поплавский Сергей Валериевич (Руководитель проекта). Также на проекте были задействованы: аналитик, разработчик и тестировщик.

# **Теоретическая часть**

# **Анализ существующих решений в области автоматизации обработки заказов**

На сегодняшний день рынок предлагает довольно обширный выбор различных готовых решений для малого, среднего и крупного бизнеса. Поэтому на основе открытых источников и публичной информации проведем сравнение нескольких решений по следующим критериям:

**Модель монетизации**

* Стоимость внедрения
* Ежемесячная/годовая подписка
* Дополнительные платные модули
* Стоимость поддержки
* Условия масштабирования

**Функциональность**

* Базовые бизнес-процессы
* Интеграции с другими системами
* Мобильная версия
* Аналитические инструменты
* Отчётность

Сравнение проведем по первым пяти предложениям в поисковике Yandex, а именно “КонтурМаркет”, “Sabyretail”, “LiteBox”, “Битрикс24”, “1C:Предприятие”.

Сравнительные таблицы представлены в [Приложении 1](#Приложение_1) и [Приложении 2](#Приложение_2) текущей работы.

На основании проведенного анализа можно сделать следующие выводы:

1. Для малого бизнеса наиболее целесообразным является выбор между “КонтурМаркет”, “LiteBox” и “Битрикс24”. Эти системы предлагают оптимальное соотношение цены и функционала, что делает их наиболее практичными решениями для начинающих предпринимателей и небольших торговых предприятий.  
   При этом важно отметить, что по мере роста бизнеса может возникнуть необходимость перехода на более мощные решения, такие как “Sabyretail” или “1C:Предприятие”, которые предоставляют расширенные возможности для управления и аналитики, но требуют значительных инвестиций в внедрение и поддержку.
2. Насыщенность рынка различными решениями свидетельствует о высоком уровне конкуренции. В таких условиях успешное внедрение нового продукта возможно при наличии следующих факторов: **уникальное торговое предложение** - внедрение инновационных пользовательских функций, которые выделят продукт среди конкурентов, **целевая ниша** - концентрация на определенной категории бизнеса для создания максимально релевантного решения, **специализация** - фокус на конкретных потребностях целевой аудитории

## **Ключевые инсайты для разработки**

Ресурсные ограничения проекта составляют 1 человека-разработчика, что определяет необходимость выбора оптимальной стратегии развития.

1. В качестве целевой аудитории определены индивидуальные предприниматели, для которых планируется разработка специализированного решения.
2. Техническое решение предусматривает создание CRM-системы на базе Telegram-бота, что обеспечивает следующие преимущества: **доступность** - использование уже установленного мессенджера, **простота интеграции** - минимальные затраты на внедрение, **удобство использования** - привычный интерфейс для пользователей, **масштабируемость** - возможность постепенного расширения функционала.
3. Также необходимо продумать простые и доступные способы реализации пользовательского интерфейса. Важно создать **минималистичный дизайн** с использованием чистого, современного стиля оформления. Особое внимание следует уделить **интуитивной навигации**: разработать логичную структуру меню, создать панель быстрого доступа и внедрить систему подсказок для новых пользователей. Интерфейс должен быть максимально простым в использовании: минимизировать количество кликов до основных функций, добавить превью изменений перед сохранением и обеспечить визуальную обратную связь при взаимодействии. С технической стороны важно использовать готовые компоненты, придерживаться модульной структуры кода и тщательно оптимизировать производительность.
4. Необходимо предусмотреть базовое логирование информации о клиентской базе, и наиболее доступным источником для этого кажется база данных MySQL. Для эффективной реализации следует создать отдельную схему для логирования, оптимизировать структуру таблиц и настроить индексацию ключевых полей. В структуре данных важно хранить информацию о **взаимодействиях с клиентами**, добавлять **временные метки операций**. Безопасность системы требует **шифрования конфиденциальных данных**, контроля доступа к логам и регулярного резервного копирования. Техническая реализация должна включать использование триггеров и хранимых процедур, настройку автоматического удаления устаревших данных и оптимизацию запросов для быстрой работы системы.
5. Необходимо предусмотреть минимальную отчетность для упрощения проведения постанализа динамики продаж пользователям. Система должна включать базовые отчеты: ежедневные, ежемесячные и ежегодные сводки, отчеты по категориям товаров и анализ продаж по регионам. Аналитические возможности должны предоставлять **графики динамики** **продаж**, **сравнение различных периодов** и **выявление трендов**. Важно обеспечить возможность экспорта данных в форматы Excel и PDF, настроить различные форматы отчетов и сохранить шаблоны для повторного использования. Пользователи должны иметь полный контроль над отчетами через фильтры и сортировку, настройку параметров и просмотр истории.

## **Проектирование архитектуры решения**

### **Общая архитектура системы**

Архитектура решения представляет собой клиент-серверную модель, где серверная часть реализована на Python с использованием MySQL для хранения данных, а клиентская часть представлена Telegram-ботом для взаимодействия с конечными пользователями.

### **Компоненты** **системы**

**Система состоит из следующих основных компонентов:**

**Backend-приложение на Python**

* Управление базой данных
* Обработка запросов от Telegram-бота
* Валидация данных
* Бизнес-логика системы

**База данных MySQL**

* Хранение информации о:
* Мастерах и их расписании
* Клиентах и их записях
* Услугах и их характеристиках
* Записях и бронированиях

**Telegram-бот**

* Фронтенд для взаимодействия с клиентами
* Реализация пользовательского интерфейса
* Обработка команд пользователя

### **Функциональные** **возможности**

#### **Основные функции системы:**

* Модуль регистрации и аутентификации:
* Регистрация новых пользователей
* Авторизация существующих пользователей
* Смена пароля

#### **Модуль управления записями:**

* Просмотр доступных мастеров и их расписания
* Бронирование времени у мастера
* Просмотр списка своих записей
* Отмена существующих записей

#### **Административный модуль:**

* Управление списками мастеров
* Настройка графика работы
* Добавление новых услуг
* Управление ценами

### **Взаимодействие компонентов**

#### **Процесс взаимодействия компонентов:**

**Клиент взаимодействует с Telegram-ботом через**

* Текстовые команды
* Кнопочные меню
* Inline-клавиатуры

**Telegram-бот:**

* Формирует запросы к backend-приложению
* Получает данные от сервера
* Отображает информацию пользователю

**Backend-приложение:**

* Обрабатывает запросы от бота
* Работает с базой данных
* Выполняет бизнес-логику
* Возвращает результаты боту

Данная архитектура обеспечивает надежное и эффективное решение для автоматизации приема и обработки заказов, с возможностью дальнейшего развития и масштабирования.

## **Обзор библиотек и инструментов, используемых в проекте**

### **Основные библиотеки**

В проекте используется **Telebot** - официальная библиотека для создания и управления ботами в Telegram. Она предоставляет удобный интерфейс для работы с Telegram Bot API, позволяя отправлять и получать сообщения, обрабатывать команды пользователей, работать с медиафайлами и реализовывать интерактивные элементы через кнопочные меню и inline-клавиатуры.

Для взаимодействия с базой данных MySQL применяется **mysql.connector** - официальный Python-драйвер, обеспечивающий надежное подключение к базе данных, выполнение SQL-запросов и работу с результатами. Библиотека позволяет эффективно управлять данными и обеспечивает безопасность при работе с базой.

### **Вспомогательные инструменты**

**SQLAlchemy** представляет собой мощную библиотеку для работы с базами данных, которая предоставляет высокоуровневый API для взаимодействия с различными СУБД. Она включает в себя систему ORM (Object-Relational Mapping), которая упрощает работу с базами данных, позволяя разработчикам оперировать объектами Python вместо написания прямых SQL-запросов.

Для работы с датами и временем в проекте используется стандартная библиотека **datetime**. Она предоставляет широкий набор инструментов для создания, форматирования и манипуляции объектами даты и времени, что особенно важно при работе с расписанием и временем записи клиентов.

### **Системные инструменты**

**Logging** - стандартная библиотека Python для логирования, которая позволяет эффективно отслеживать работу приложения. Она предоставляет гибкие возможности настройки уровней логирования, форматирования сообщений и записи их в различные источники, что существенно упрощает отладку и мониторинг работы системы.

Для работы с файлами и файловой системой используются модули **os** и **sys**. Они предоставляют набор функций для взаимодействия с операционной системой, включая управление файлами, каталогами и процессами, а также доступ к различным системным параметрам и функциям интерпретатора Python.

### **Дополнительные инструменты**

**Subprocess** - модуль для запуска и управления внешними процессами. Он позволяет выполнять системные команды, запускать другие программы и получать результаты их выполнения, что может быть полезно при необходимости интеграции с внешними сервисами или выполнения системных задач.

**Pathlib** представляет собой современный модуль для работы с путями файлов, который обеспечивает удобный и безопасный способ работы с файловой системой. Он предлагает объектно-ориентированный подход к работе с путями, что делает код более читаемым и упрощает навигацию по файловой системе.

### **Специализированные инструменты**

**Charset\_normalizer** используется для автоматического определения кодировки текстовых данных. Библиотека помогает корректно обрабатывать текстовые файлы различных форматов, автоматически определяя и нормализуя их кодировку.

**Pycparser** - это компилятор препроцессора C, который предоставляет возможности для парсинга C-кода и работы с абстрактным синтаксическим деревом (AST). Хотя в контексте данного проекта его использование может показаться необычным, он может быть полезен при необходимости работы с C-кодом или его анализа.

### **Интеграция и взаимодействие**

Все используемые библиотеки и инструменты хорошо интегрируются между собой, что позволяет создавать надежные и масштабируемые решения. Выбор данных инструментов обусловлен их стабильностью, широким распространением и активной поддержкой сообщества разработчиков, что обеспечивает долгосрочную поддержку проекта и его дальнейшее развитие.

Каждая библиотека выполняет свою специфическую функцию, при этом они эффективно взаимодействуют друг с другом, обеспечивая надежную работу всей системы. Такой набор инструментов позволяет создавать производительные и безопасные решения, которые легко поддерживать и развивать в будущем.

# **Практическая часть**

## **Описание разработанного приложения**

#### **Общее описание приложения**

Приложение представляет собой консольное решение для автоматизации управления базой данных и настройки системы. Оно разработано на Python и предназначено для работы с MySQL. Основной функционал приложения реализуется через главное меню, которое позволяет пользователю выполнять различные действия, связанные с настройкой и управлением данными.

#### **Основные функции приложения**

**Установка и настройка MySQL.** Приложение предоставляет возможность установки и базовой конфигурации MySQL, что упрощает процесс подготовки базы данных для работы.

**Настройка службы MySQL и базы данных.** Пользователь может управлять службой MySQL, включая её запуск, остановку и настройку параметров базы данных.

**Редактирование таблиц мастеров.** Приложение позволяет добавлять, изменять и удалять записи в таблице мастеров, что удобно для управления персоналом.

**Редактирование таблицы услуг.** Пользователь может добавлять, изменять и удалять услуги, доступные в системе.

**Редактирование графика работы.** Приложение предоставляет инструменты для настройки графика работы мастеров, что помогает оптимизировать расписание.

**Выход.** Возможность завершить работу с приложением.

#### **Интерфейс пользователя**

Приложение запускается через консоль и выводит главное меню, оформленное в виде таблицы с пунктами выбора. Пользователь может перемещаться по меню с помощью клавиш или ввода номеров пунктов. После выбора действия приложение выполняет соответствующую операцию и возвращает пользователя в главное меню.

#### **Преимущества**

Простота использования благодаря интуитивно понятному интерфейсу. Возможность выполнения всех необходимых операций через консоль. Гибкость настройки и управления данными.

## **Структура проекта**

* Program:
  + Main.py – основной исполняемый файл для открытия главного меню;
  + Директория Downloads – для выгрузки и экспорта csv файлов;
  + Директория Modules – все компоненты основного приложения
  + Директория Telebot – директория содержащая скрипт для работы бота
    - Директория Modules\_bot – директория с дополнительными компонентами, которые использует bot
    - Bot.py – файл для запуска бота телеграмма

## **Взаимодействие между различными компонентами приложения**

При запуске файла **main.py** пользователю предлагается выбрать пункты меню: 1 – Вызов модуля **Download**, 2 – вызов модуля **initialization**, 3 – вызов модуля **Masters**, 4 – вызов модуля **services**, 5 – вызов модуля **schedule**.

#### **Модуль Downloads**

Модуль **download.py** представляет собой важный компонент системы, отвечающий за автоматизированную установку MySQL. Он объединяет в себе функционал для скачивания, установки и настройки базы данных, обеспечивая пользователя удобным интерфейсом для выполнения этих задач.

В основе работы модуля лежит несколько ключевых функций:

1. Функция **downmysql()** отвечает за загрузку установщика MySQL с официального сайта. Прежде чем начать загрузку, модуль проверяет наличие файла на компьютере пользователя, что предотвращает ненужное дублирование данных. Процесс загрузки сопровождается наглядным индикатором прогресса, который реализуется через вспомогательную функцию **bar\_progress()**. Эта функция выводит информацию о текущем состоянии загрузки, показывая процент выполнения, текущие и общие байты.
2. Для установки MySQL используется функция **installmysql()**. Она запускает процесс установки через системный msiexec, предварительно проверив наличие скачанного установщика. В случае успешного выполнения операции пользователь получает соответствующее уведомление, в противном случае выводится подробная информация об ошибке.
3. Важной частью модуля является функция **show\_instruction()**, которая предоставляет пользователю доступ к инструкции по установке MySQL. Функция ищет специальный файл в директории Modules и выводит его содержимое на экран, что помогает пользователям корректно настроить систему.
4. Управление всеми функциями осуществляется через консольное меню, реализованное в функции **show\_menu()**. Оно предоставляет пользователю простой интерфейс с пунктами для скачивания, установки MySQL, просмотра инструкции и выхода в главное меню. Модуль тщательно обрабатывает ввод пользователя, проверяя корректность введённых данных и предупреждая о возможных ошибках.

**Download.py** отличается продуманной системой обработки ошибок. Он проверяет существование необходимых файлов, корректно обрабатывает исключения при установке и выводит понятные сообщения об ошибках. Такой подход делает процесс установки MySQL максимально простым и безопасным для конечного пользователя.

Модуль **download.py** активно использует стандартные библиотеки Python: **wget** для загрузки файлов, **os** и **sys** для работы с операционной системой, а также **subprocess** для выполнения системных команд. Это обеспечивает его надёжность и совместимость с различными системами.

В целом **download.py**, представляет собой комплексное решение для автоматизации процесса установки MySQL, которое значительно упрощает работу пользователя и минимизирует вероятность ошибок при настройке системы.



#### **Модуль Initialization**

Модуль **initialization.py** представляет собой ключевой компонент системы, отвечающий за настройку и инициализацию базы данных MySQL. Он объединяет в себе функционал управления службами MySQL и создания необходимых таблиц для работы приложения.

**Управление службами MySQL**

Модуль предоставляет комплексный инструментарий для работы с службами MySQL на Windows-системах. Он включает в себя:

1. Мониторинг состояния служб - функция **get\_service\_status()** позволяет отслеживать текущий статус службы MySQL
2. Управление запуском - функция **start\_service()** обеспечивает корректное включение службы с проверкой успешности операции
3. Управление остановкой - функция **stop\_service()** отвечает за безопасное завершение работы службы
4. Общее управление - функция **service\_info()** объединяет все операции управления в единый интерфейс

**Создание базы данных:**

Функция **create\_database()** отвечает за инициализацию всей необходимой структуры данных:

1. Создание базы данных Autocalendar с проверкой существования

**Инициализация таблиц:**

1. **Clients** - хранит информацию о клиентах
2. **Masters** - содержит данные о мастерах
3. **Services** - перечень предоставляемых услуг
4. **Schedule** - расписание работы
5. **Bookings** - записи клиентов

**Создание представления (VIEW)**

Модуль создает представление записанные пользователи, которое объединяет данные из нескольких таблиц для получения информации о забронированных услугах.

**Реализовано два EVENT для автоматической работы с данными:**

1. **update\_view\_event** - обновляет представление каждые 5 минут
2. **auto\_check\_slots** - проверяет и закрывает свободные слоты каждые 15 минут

**Управление статусом записей**

Модуль включает два триггера для автоматического управления статусом записей в таблице Schedule:

1. **update\_status\_on\_insert** - устанавливает статус при вставке новой записи
2. **update\_status\_on\_update** - обновляет статус при изменении записи

Управление модулем осуществляется через интуитивно понятное консольное меню **show\_menu()**, которое предоставляет пользователю доступ к основным функциям: получению статуса службы MySQL, её запуску и остановке, созданию базы данных. Такой подход делает работу с модулем доступной даже для пользователей без глубоких технических знаний.

Модуль **initialization.py** является критически важным компонентом системы, обеспечивающим корректную инициализацию и настройку базы данных, что является фундаментом для работы всего приложения.



#### **Модуль Master**

Модуль Master представляет собой важный компонент системы, отвечающий за управление информацией о мастерах в базе данных. Он обеспечивает полный цикл работы с данными: от добавления новых записей до их редактирования и удаления.

**Основные функции модуля**

1. Добавление мастеров реализовано через функцию **add\_master()**, которая позволяет вводить данные о новых специалистах. Функция запрашивает у пользователя ФИО, логин и номер телефона, после чего выполняет массовую вставку записей в базу данных. Особое внимание уделяется валидации данных - все поля обязательны для заполнения, что предотвращает появление неполных записей.
2. Редактирование информации осуществляется через функцию **edit\_master()**. Она предоставляет возможность обновить данные существующего мастера. Пользователь может изменить ФИО, логин или номер телефона, при этом система позволяет редактировать только выбранные поля, оставляя остальные без изменений. Перед внесением изменений происходит проверка существования мастера в базе данных.
3. Удаление мастеров выполняется функцией **delete\_master().** Она сначала выводит список всех существующих мастеров с их идентификаторами, после чего пользователь может выбрать конкретного специалиста для удаления. Такой подход предотвращает случайное удаление данных.
4. Просмотр мастеров реализован через функцию **show\_master()**, которая выводит полный список всех специалистов с их данными в удобном формате.
5. Экспорт данных осуществляется функцией **export\_masters\_to\_csv()**, которая сохраняет информацию о мастерах в CSV-файл с уникальным именем, содержащим временную метку. Файл сохраняется в специальную директорию загрузок.
6. Импорт данных выполняется функцией **import\_masters\_from\_csv()**, которая позволяет загружать информацию из CSV-файлов, находящихся в директории загрузок. Система показывает список доступных файлов и позволяет выбрать нужный для импорта.
7. Управление модулем осуществляется через интуитивно понятное консольное меню **show\_menu()** управления мастерами, которое обеспечивает удобный интерфейс для работы с данными.

Модуль Master является важным инструментом для администрирования базы данных мастеров, обеспечивая полный контроль над их информацией и поддерживая целостность данных.



#### **Модуль Paths**

Модуль **Paths** представляет собой вспомогательный инструмент для работы с путями в файловой системе. Его основная задача - обеспечить корректное формирование пути к директории Downloads относительно директории Program.

**Принцип работы**

Функция **path\_downloads()** выполняет следующие действия:

1. Получает текущий путь к директории с помощью os.getcwd()
2. Разбивает путь на составные части с помощью разделителя директорий
3. Находит индекс директории Program в полученном списке
4. Формирует новый путь до директории Downloads

#### **Модуль Services**

Модуль Services представляет собой важный компонент системы, отвечающий за управление информацией об услугах в базе данных. Он обеспечивает полный цикл работы с данными: от добавления новых записей до их редактирования.

**Основные функции модуля**

1. Добавление услуг реализовано через функцию **add\_service(),** которая позволяет массово вводить данные о новых услугах. Функция запрашивает у пользователя количество записей, которые нужно добавить, а затем собирает информацию о названии услуги, её длительности и стоимости. Длительность по умолчанию устанавливается в 1 час, остальные поля обязательны для заполнения.
2. Редактирование услуг осуществляется через функцию **edit\_service().** Она предоставляет возможность обновить информацию об уже существующих услугах. Сначала выводится список всех услуг с их ID, после чего пользователь может выбрать конкретную услугу для редактирования. Система позволяет менять отдельные поля, оставляя остальные без изменений.
3. Функция **delete\_services()** отвечает за удаление услуг из базы данных. Она обеспечивает безопасный процесс удаления с подтверждением действий. Включает в себя - **получение списка услуг** - выводит все доступные услуги с их ID, названием, длительностью и ценой, **выбор услуги** - пользователь указывает ID услуги для удаления.
4. Функция **show\_service()** предназначена для просмотра всех услуг в базе данных. Включает в себя - **получение данных** - запрашивает информацию об услугах из базы, **вывод списка** - показывает ID, название, длительность и цену каждой услуги.
5. Функция **export\_services\_to\_csv()** позволяет экспортировать данные об услугах в CSV-файл. Включает в себя - **получение данных** - запрашивает информацию об услугах, **формирование пути** - использует модуль **Paths** для определения директории сохранения, **создание файла** - генерирует уникальное имя файла с временной меткой, **экспорт данных** - записывает информацию в CSV-файл с заголовками, **информирование** - сообщает о успешном экспорте с указанием пути к файлу
6. Функция **import\_services\_from\_csv()** представляет собой мощный инструмент для массового обновления данных об услугах через CSV-файлы. Сначала она сканирует директорию загрузок на наличие CSV-файлов и выводит их список для выбора пользователем. Система обеспечивает надежную валидацию ввода, проверяя корректность выбранного номера файла и обрабатывая возможные ошибки преобразования данных.

При обработке файла происходит обновление существующих записей в базе данных. Функция тщательно проверяет каждую строку на наличие всех необходимых полей и корректно обрабатывает пустые значения. После успешного импорта система автоматически удаляет обработанный файл из директории, что помогает поддерживать порядок в системе.

1. В центре модуля находится удобное консольное меню **show\_menu()**, которое предоставляет полный набор инструментов для работы с услугами. Меню оформлено в привлекательном визуальном стиле с использованием декоративных элементов и иконок, что делает его интуитивно понятным для пользователя.

#### **Модуль Schedule**

Модуль Schedule представляет собой комплексный инструмент для работы с расписанием мастеров в системе. Он обеспечивает создание, редактирование и импорт расписания с учетом существующих записей и валидации данных.

**Основные функции модуля**

1. Функция **list\_csv()** автоматически сканирует директорию загрузок, предоставляет выбор из доступных файлов и выполняет валидацию ввода.
2. Для работы с расписанием необходимо иметь актуальную информацию о мастерах. Функция **get\_masters()** отвечает за получение списка всех мастеров из базы данных, включая их ФИО, логин и номер телефона.
3. Процесс создания расписания организован максимально эффективно через функцию **create\_schedule().** При формировании расписания происходит- предварительная загрузка существующего расписания, формирование новых слотов с учетом рабочих часов, проверка отсутствия пересечений с существующими записями, обновление статуса слотов, автоматическое предотвращение дублирования записей.
4. Функция **import\_schedule()** обеспечивает гибкий механизм импорта расписания из CSV-файлов. При этом происходит чтение данных из CSV-файлов, выполняется конвертация форматов даты, осуществляется массовое обновление записей, обрабатываются возможные ошибки при конвертации, подтверждается успешное обновление данных.
5. Функция **export\_schedule()** представляет собой инструмент для выгрузки расписания конкретного мастера в CSV-файл. При вызове функции происходит следующее - система устанавливает безопасное соединение с базой данных и запрашивает все доступные слоты указанного мастера со статусом “open”. После получения данных формируется путь сохранения файла в директорию загрузок с использованием уникального имени, содержащего ID мастера. Процесс экспорта включает создание CSV-файла с заголовками столбцов и последовательную запись всех записей расписания. При успешном завершении операции пользователь получает информативное сообщение с указанием пути к созданному файлу. Функция обеспечивает надежную обработку ошибок и корректное завершение работы при возникновении проблем с базой данных.
6. В центре модуля находится удобное консольное меню **show\_menu()**, которое предоставляет полный набор инструментов для работы с услугами. Меню оформлено в привлекательном визуальном стиле с использованием декоративных элементов и иконок, что делает его интуитивно понятным для пользователя.

Модуль Schedule является важным компонентом системы управления, обеспечивающим корректное формирование и редактирование расписания работы мастеров с учетом всех бизнес-требований и технических ограничений. Его продуманная архитектура и надежная реализация делают его незаменимым инструментом для управления расписанием в системе.

## **Описание взаимодействия компонентов телеграмм бота**

#### **Общее описание бота**

Telegram-бот представляет собой удобный инструмент для взаимодействия пользователей с сервисом записи к мастерам. Бот обеспечивает полный цикл работы с аккаунтом и записями, включая регистрацию, авторизацию и управление записями.

#### **Обработчик команды Start**

**Start** является одним из фундаментальных обработчиков бота, который активируется при получении команды /start от пользователя. Этот обработчик представляет собой точку входа в систему и отвечает за инициализацию первого взаимодействия с пользователем.

Использует функцию **send\_main\_menu()** представляющую собой ключевой элемент пользовательского интерфейса бота, отвечающий за отображение начального экрана взаимодействия с пользователем. При вызове функции происходит последовательная инициализация всех необходимых компонентов интерфейса.  
В первую очередь функция создает пустую клавиатуру типа InlineKeyboardMarkup, которая является контейнером для размещения интерактивных элементов управления. Это специальный тип разметки, позволяющий размещать кнопки непосредственно под сообщением, что делает интерфейс более компактным и удобным для пользователя.

Далее происходит создание трех основных кнопок управления. Каждая кнопка инициализируется как объект InlineKeyboardButton с двумя обязательными параметрами: текстовым содержимым, которое будет отображаться пользователю, и callback\_data - внутренним идентификатором, который будет передан боту при нажатии на кнопку. Такой подход позволяет системе корректно обрабатывать действия пользователя и направлять его в нужное подменю.

После создания кнопок они добавляются в ранее инициализированную клавиатуру с помощью метода add(). Этот метод позволяет группировать кнопки в одну строку, что обеспечивает оптимальное расположение элементов управления на экране мобильного устройства.

Финальным этапом работы функции является отправка приветственного сообщения пользователю с прикрепленной клавиатурой. Сообщение содержит важную информацию о назначении бота, его статусе как учебного проекта и предупреждении о необходимости воздержаться от использования реальных персональных данных. Такой подход обеспечивает безопасность пользователей и предотвращает возможные проблемы с конфиденциальностью.

Для отправки сообщения используется метод send\_message() с тремя обязательными параметрами: идентификатор чата для определения получателя, текст сообщения с форматированием для лучшей читаемости и объект разметки reply\_markup, содержащий созданную клавиатуру с кнопками.

Благодаря такой структуре функция обеспечивает единообразное отображение главного меню всем пользователям бота, предоставляя им базовые возможности для начала взаимодействия с системой. При этом интерфейс остается гибким и может быть легко модифицирован путем добавления или изменения кнопок управления.

#### **Обработчик нажатие кнопок главного меню**

**callback\_message** является центральным обработчиком callback-запросов бота, который отвечает за обработку основных действий пользователя в главном меню. Функция активируется при нажатии на кнопки **регистрации**, **авторизации**, **смены пароля** или **отмены**.

При запуске обработчика происходит:

1. Извлечение идентификатора пользователя из полученного callback-запроса
2. Установление соединения с базой данных через функцию **get\_connection()**
3. Создание базового набора кнопок с возможностью отмены операции
4. Инициализация клавиатуры с кнопкой “Отмена”

#### **Кнопка регистрации**

Реализует полный цикл регистрации нового пользователя в системе. Процесс разделен на четыре последовательных этапа, каждый из которых представлен отдельной функцией: проверка уникальности аккаунта, ввод ФИО, ввод номера телефона и сохранение данных в базе данных.

Функция **registration()** является точкой входа в процесс регистрации. Она принимает callback-запрос, курсор базы данных и идентификатор пользователя. Первым делом функция проверяет, не зарегистрирован ли уже пользователь с таким идентификатором в таблице clients. При обнаружении существующего аккаунта процесс прерывается с соответствующим сообщением.

Если пользователь уникален, система последовательно запрашивает дополнительную информацию:

**get\_fio()** отвечает за получение ФИО пользователя. Функция создает интерфейс с кнопкой отмены и ожидает ввода данных. После получения ФИО система переходит к следующему этапу.

**get\_phone()** запрашивает номер телефона пользователя. Как и предыдущие функции, она обеспечивает возможность отмены операции и передает собранные данные дальше по цепочке.

**save\_registration()** представляет собой финальный этап процесса регистрации пользователя в системе, который отвечает за сохранение собранных данных в базе данных. Функция принимает четыре параметра: объект сообщения с введённым паролем, идентификатор пользователя, его ФИО и номер телефона, полученные на предыдущих этапах регистрации. Основной частью функции является выполнение SQL-запроса на добавление новой записи в таблицу clients. Запрос использует параметризованные значения, что защищает систему от SQL-инъекций и делает код более безопасным. В таблицу заносятся все собранные данные: идентификатор пользователя в качестве логина, ФИО, введённый пароль и номер телефона.

#### **Кнопка авторизации**

При нажатии на кнопку “Авторизация” вызывается функция **process\_password()**, которая представляет собой критически важный компонент системы аутентификации, отвечающий за процесс авторизации пользователя в системе. Функция принимает два параметра: объект сообщения с введённым паролем и идентификатор пользователя, который используется в качестве логина.

Первым делом функция извлекает введённый пользователем пароль из текста сообщения. Затем она выполняет базовую валидацию, проверяя, был ли пароль введён вообще. Если поле пароля пустое, система информирует пользователя об ошибке и возвращает его в главное меню, что предотвращает дальнейшие попытки авторизации без ввода пароля.

Для обеспечения безопасного взаимодействия с базой данных функция использует контекстный менеджер with, который автоматически закрывает соединение после выполнения всех операций. Это гарантирует корректное освобождение ресурсов даже при возникновении ошибок. Внутри блока создаётся курсор для выполнения SQL-запросов.

Основной частью функции является выполнение SQL-запроса, который извлекает сохранённый пароль из таблицы clients по идентификатору пользователя. Использование параметризированного запроса защищает систему от SQL-инъекций и делает код более безопасным.

После получения данных из базы данных система проверяет, был ли найден пользователь. Если запись не обнаружена, пользователю отправляется сообщение о том, что пользователь не найден, и он возвращается в главное меню. Это предотвращает несанкционированный доступ и информирует пользователя о возможной ошибке в вводе идентификатора.

При успешном нахождении пользователя происходит сравнение введённого пароля с сохранённым в базе данных. Если пароли совпадают, система подтверждает успешную авторизацию, отправляет пользователю соответствующее сообщение с его идентификатором и отправляет меню для авторизованных пользователей **auto\_menu().**

В случае несовпадения паролей пользователь получает информативное сообщение о неверном пароле и предложение либо повторить попытку, либо сменить пароль. Система возвращает его в главное меню, где он может выбрать соответствующий пункт для смены пароля.

#### **Кнопка смены пароля**

При нажатии на кнопку “Сменить пароль” вызывается функция **change\_password(),** которая представляет собой функцию, отвечающую за процесс смены пароля пользователя в системе. При получении callback-запроса на изменение пароля система ожидает ввода нового пароля от пользователя. Первым делом функция проверяет, был ли введен новый пароль. Если поле пароля пустое, система информирует пользователя об ошибке и возвращает его в главное меню, предотвращая дальнейшие действия без ввода пароля.

Для обеспечения безопасности и корректной работы с базой данных функция использует контекстный менеджер with, который автоматически закрывает соединение после выполнения всех операций. Внутри блока создаётся курсор для выполнения SQL-запросов. Основной частью функции является проверка существования пользователя в базе данных. Система выполняет SQL-запрос, который ищет запись с указанным идентификатором пользователя в таблице clients.

Если пользователь найден, функция делегирует процесс обновления пароля вспомогательной функции **update\_password(),** передавая ей новый пароль и идентификатор пользователя. В случае отсутствия пользователя в базе данных система информирует пользователя об ошибке и предлагает либо повторить попытку, либо зарегистрироваться, после чего возвращает его в главное меню.

**update\_password()** является вспомогательной функцией, непосредственно отвечающей за обновление пароля в базе данных. Она принимает два параметра: новый пароль и идентификатор пользователя. Первым делом функция устанавливает соединение с базой данных и создаёт курсор для выполнения SQL-запросов. Основной частью функции является выполнение SQL-запроса на обновление пароля для указанного пользователя.

#### **Обработчик запросов из меню для авторизованных пользователей**

**callback\_message()** представляет собой основной обработчик меню для авторизованных пользователей. Функция реагирует на три основных действия: создание новой записи, просмотр существующих записей и отмена записи. В зависимости от выбранного пункта меню система выполняет соответствующие действия: показывает список услуг, выводит список записей пользователя или предлагает выбрать запись для отмены.

#### **Обработчик нажатия кнопки регистрации**

При выборе создания новой записи система отображает клавиатуру с доступными услугами.

**select\_service()** отвечает за обработку выбора услуги пользователем. Когда пользователь выбирает услугу из списка, система сохраняет ID выбранной услуги в словарь user\_data и показывает список доступных мастеров. После этого система отправляет пользователю подтверждение о выбранном ID услуги.

В обработчике вызывается функция **get\_service\_list()** представляет собой функцию, которая отвечает за получение и формирование списка доступных услуг для пользователя. Функция не принимает никаких параметров и возвращает готовую клавиатуру с кнопками выбора услуг.

После получения данных функция начинает формировать интерактивную клавиатуру. Для каждой услуги из базы данных создаётся отдельная кнопка, которая содержит название услуги и её стоимость в формате “Название услуги - Стоимость руб.”. Каждая кнопка также получает уникальный callback\_data, который содержит ID услуги в формате “select\_service\_ID”, что позволяет системе корректно обрабатывать выбор пользователя.

**select\_master()** обрабатывает выбор мастера пользователем. При выборе мастера система сохраняет его ID в словарь user\_data и проверяет наличие свободных слотов. Если слоты доступны, система показывает клавиатуру с датами для записи. Если слотов нет, пользователю предлагается выбрать другого мастера. В конце система отправляет подтверждение о выбранном мастере.

В обработчике вызывается функция **get\_masters\_list()** представляет собой функцию, которая отвечает за получение и формирование списка доступных мастеров для пользователя. Функция не принимает никаких параметров и возвращает готовую клавиатуру с кнопками выбора мастеров.

После получения данных функция начинает формировать интерактивную клавиатуру. Для каждого мастера из базы данных создаётся отдельная кнопка, которая содержит его ФИО. Каждая кнопка также получает уникальный callback\_data, который содержит ID мастера в формате “select\_master\_ID”, что позволяет системе корректно обрабатывать выбор пользователя.

**select\_date()** отвечает за обработку выбора даты пользователем. При выборе даты система извлекает ID мастера из словаря user\_data и показывает клавиатуру с доступными часами для записи. После этого система отправляет пользователю подтверждение о выбранной дате.

В обработчике вызывается функция **get\_free\_hours()** представляет собой функцию, которая отвечает за получение и формирование списка свободных временных слотов для записи к мастеру в конкретный день. Функция принимает два параметра: строку с датой (date\_str) и ID мастера (master\_id), на основе которых формирует список доступных часов.

После получения данных функция начинает формировать интерактивную клавиатуру. Для каждого доступного временного слота создаётся отдельная кнопка, которая содержит только час начала приёма. Каждая кнопка также получает уникальный callback\_data, который содержит ID временного слота в формате “select\_time\_ID”, что позволяет системе корректно обрабатывать выбор пользователя.

**select\_time()** является финальным обработчиком, который завершает процесс записи. При выборе времени система извлекает необходимые данные из словаря user\_data и создаёт новую запись в базе данных.

В обработчике вызывается функция **get\_client\_id()** представляет собой функцию, которая отвечает за получение уникального идентификатора клиента по его логину. Функция принимает один параметр — client\_id (логин пользователя) и возвращает его ID из базы данных или None, если пользователь не найден.

Далее выполняется SQL-запрос на добавление новой записи, связывая ID клиента, расписания и услуги. После успешной записи система подтверждает создание записи и возвращает пользователя в главное меню.

Все обработчики включают механизмы обработки ошибок, которые информируют пользователя о возможных проблемах и возвращают его в главное меню. Система использует словарь user\_data для временного хранения выбранных параметров, что позволяет сохранять контекст между различными этапами выбора записи. Такой подход обеспечивает плавный и интуитивно понятный процесс бронирования услуг для пользователя, делая взаимодействие с системой максимально удобным.

Каждый обработчик включает подтверждение выбора через метод answer\_callback\_query, что даёт пользователю обратную связь о его действиях. При этом система следит за доступностью слотов и информирует пользователя о невозможности записи, если выбранные параметры недоступны. Такой подход обеспечивает корректную работу системы бронирования и делает процесс записи максимально прозрачным для пользователя.

#### **Обработчик нажатия кнопки просмотра записей**

Если пользователь хочет проверить свои записи, система запрашивает список его бронирований и показывает их в текстовом формате посредством вызова функции **get\_appointments\_list()** которая представляет собой функцию, отвечающую за получение и форматирование списка записей пользователя. Функция принимает один параметр — идентификатор пользователя (chat\_id) и возвращает структурированную информацию о его записях.

После получения данных функция проверяет, есть ли у пользователя записи. Если записи отсутствуют, функция возвращает сообщение “У вас нет записей”, информируя пользователя об отсутствии бронирований.

При наличии записей система формирует структурированное текстовое сообщение. Она начинает с заголовка “Ваши записи:” и последовательно добавляет информацию о каждой записи, включая дату, время, ФИО мастера, название услуги и её стоимость. Каждое поле выводится на отдельной строке для лучшей читаемости, а между записями добавляются пустые строки для визуального разделения информации.

#### **Обработчик нажатия кнопки удаления записи**

При выборе отмены записи система проверяет наличие записей у пользователя посредством вызова функции **get\_booking\_id()**

**get\_booking\_id()** представляет собой функцию, которая отвечает за получение и формирование списка записей пользователя для их последующей отмены. Функция принимает один параметр — chat\_id (идентификатор пользователя) и возвращает клавиатуру с доступными для отмены записями или сообщение об отсутствии записей.

Для каждой записи создаётся отдельная кнопка, содержащая - время записи, название услуги, скрытый ID записи в callback\_data формате “select\_appointment\_ID”.

Созданная клавиатура возвращается в качестве результата функции, после чего может быть использована для отправки пользователю через мессенджер. Если записей у пользователя нет, то ему возвращается сообщение "У вас нет записей для отмены".

Если они есть, предлагает выбрать запись для отмены через специальную клавиатуру и передает выбор в функцию **select\_appointment()** представляющую собой обработчик, который отвечает за обработку запросов на удаление записей в системе. Она активируется каждый раз, когда пользователь нажимает на кнопку с callback\_data, начинающейся на ‘select\_appointment\_’, что указывает на намерение отменить конкретную запись.

При активации обработчика происходит извлечение уникального идентификатора записи из полученного callback\_data. Для этого используется метод split(), который разбивает строку по символу подчеркивания, после чего извлекается последний элемент массива (ID записи). Преобразование в целочисленный тип данных обеспечивает корректную работу с идентификатором в последующих операциях.

Далее функция пытается удалить запись, вызывая метод **delete\_booking()** из класса A\_key и передавая ему извлеченный ID. Для обеспечения стабильности работы используется блок try-except, который перехватывает возможные ошибки при выполнении операции удаления.

**delete\_booking()** представляет собой надежный механизм удаления записей из базы данных, который обеспечивает корректную обработку операций и защиту от возможных ошибок. Функция принимает один параметр - уникальный идентификатор записи (id\_booking), которую необходимо удалить.

В случае успешного удаления записи (если метод delete\_booking возвращает True), пользователю отправляется сообщение с подтверждением об успешном удалении. При неудачной попытке удаления (если метод возвращает False) пользователь получает соответствующее уведомление об ошибке. Дополнительно предусмотрен перехватчик ValueError, который обрабатывает возможные ошибки преобразования типов данных и также информирует пользователя о возникшей проблеме.

После выполнения операции удаления (независимо от её результата) функция вызывает auto\_menu, передавая идентификатор чата. Это обеспечивает возврат пользователя в главное меню системы, что является удобной практикой для поддержания целостности пользовательского интерфейса.



# **Тестирование разработанного решения**

Для обеспечения качества и надежности приложения, а также подтверждения корректности его работы, был проведен комплексный цикл тестирования. В процессе тестирования особое внимание уделялось проверке всех функциональных возможностей системы, включая работу с базой данных MySQL, корректность обработки пользовательского ввода через консольное меню и стабильность взаимодействия между различными компонентами приложения.

Тестирование проводилось с учетом специфики консольного интерфейса и особенностей работы с базой данных, что позволило всесторонне оценить работоспособность приложения в различных сценариях использования. Особое внимание было уделено проверке обработки исключительных ситуаций и корректности обработки ошибок, что является критически важным для систем управления данными.

В ходе тестирования были проверены все основные сценарии использования приложения, включая базовые операции с данными, настройку параметров системы и работу с различными типами данных в базе MySQL. Результаты тестирования позволили убедиться в соответствии приложения заявленным требованиям и его готовности к практическому применению.

## **Тестирование системы управления автокалендарём**

Данный раздел посвящен проверке работоспособности основного интерфейса приложения и всех его функциональных модулей. Особое внимание уделяется тестированию главного меню, механизма обработки пользовательского ввода и корректности взаимодействия между различными компонентами системы.

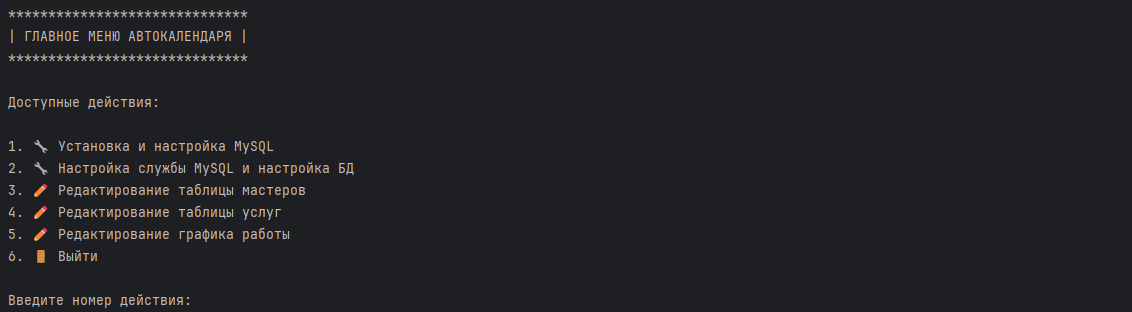
**В процессе тестирования проверяется:**

* корректность отображения меню
* валидация пользовательского ввода
* обработка исключительных ситуаций
* корректность перехода между модулями
* работа механизма выхода из приложения

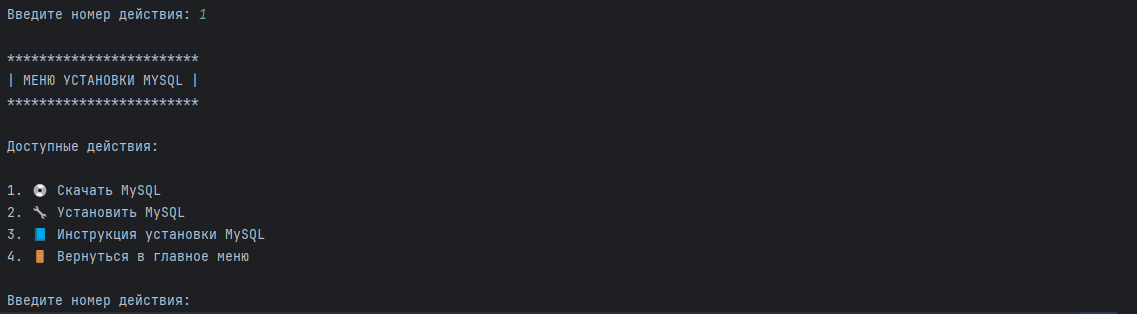
Тестирование проводится с учетом всех возможных сценариев использования, включая корректные и некорректные варианты ввода данных пользователем.

**Тесты**

Главное меню:



Переход в первый раздел:



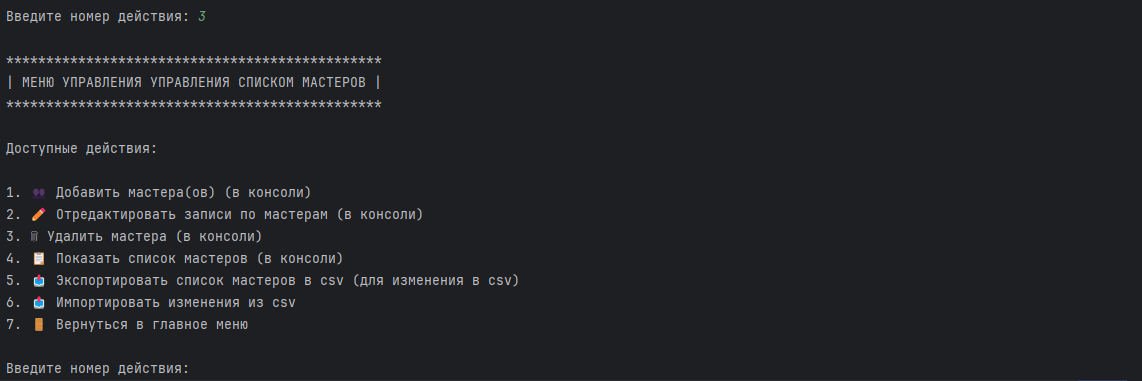
Возврат в главное меню:



Переход во второй раздел:



Переход в третий раздел:



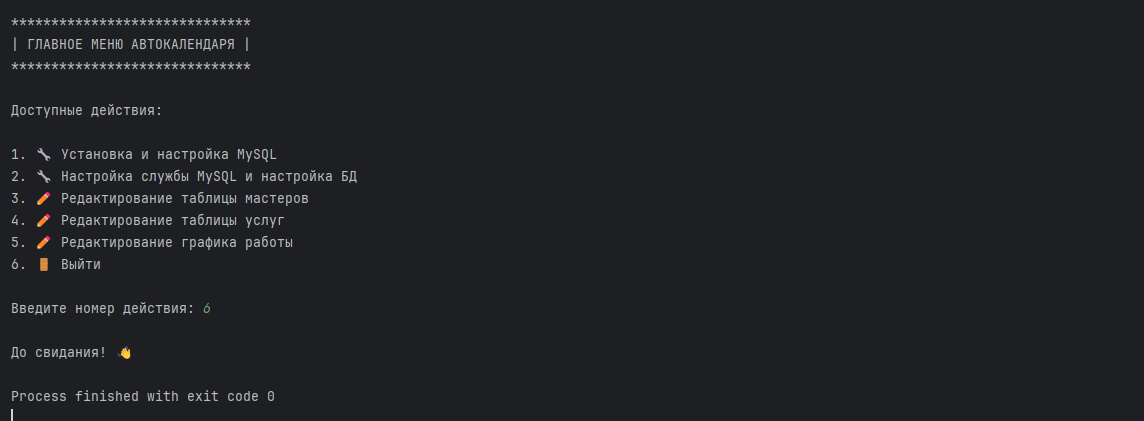
Переход в четвертый раздел:



Переход в пятый раздел:



Выход:



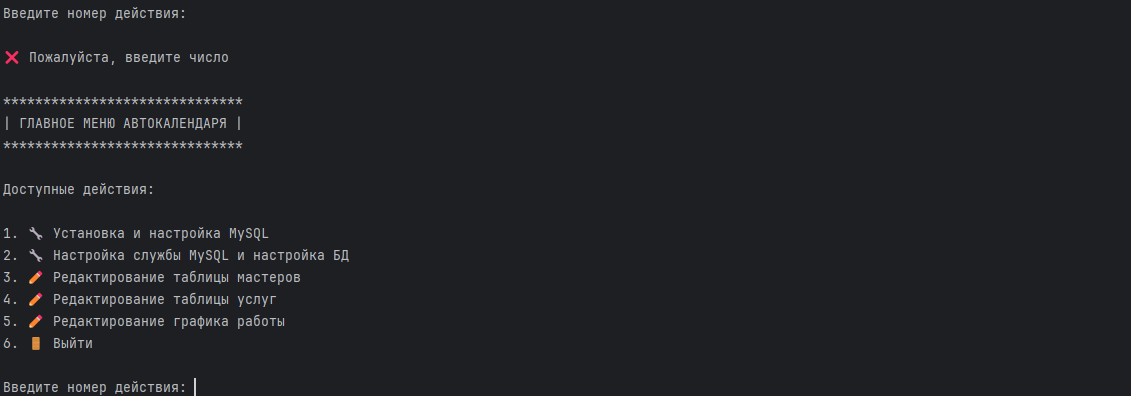
Отработка чисел больше 6:



Буквенный ввод:



Пустые значения:



**По итогам тестирования** - все основные сценарии использования приложения были успешно протестированы:

* Корректное отображение главного меню
* Правильная работа системы валидации ввода
* Успешный переход между модулями
* Корректная работа механизма выхода из приложения
* Правильное отображение разделителей и форматирование

**Негативные сценарии:**

Проведена проверка обработки некорректных данных:

* Успешная обработка невалидных чисел (вне диапазона 1-6)
* Корректная обработка буквенного ввода
* Правильная обработка пустых значений

В процессе тестирования критических ошибок не обнаружено. Все выявленные незначительные проблемы были устранены до завершения тестирования.

Приложение работает стабильно и соответствует заявленным требованиям. Все функциональные модули взаимодействуют корректно, система обработки ошибок работает должным образом.

## **Тестирование меню установки MySQL**

Необходимо проверить корректность работы меню установки MySQL, включая отображение элементов меню, обработку пользовательского ввода и выполнение соответствующих действий.

**Тестируемые функции:**

* Отображение меню с доступными действиями
* Валидация ввода пользователя
* Выполнение соответствующих функций в зависимости от выбора
* Обработка ошибок ввода

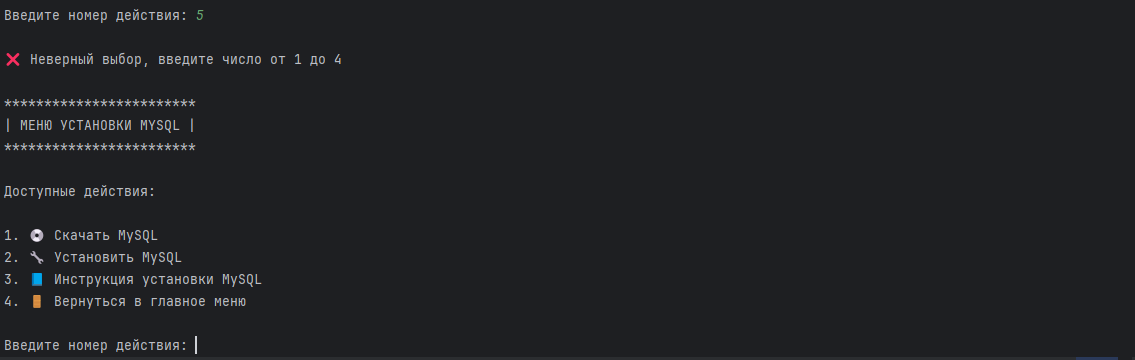
Таким образом, тестирование направлено на всестороннюю проверку работоспособности модуля установки MySQL, его интерфейса и взаимодействия с пользователем, что является ключевым фактором для обеспечения качественной работы всей системы управления автокалендарём.

**Тесты**

Главное меню:

****

Ввод числа больше 4:



Пустые значения:



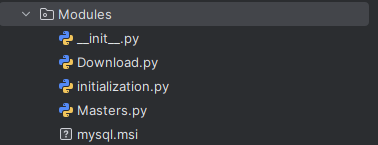
Текстовые значения:



Загрузка MySQL:



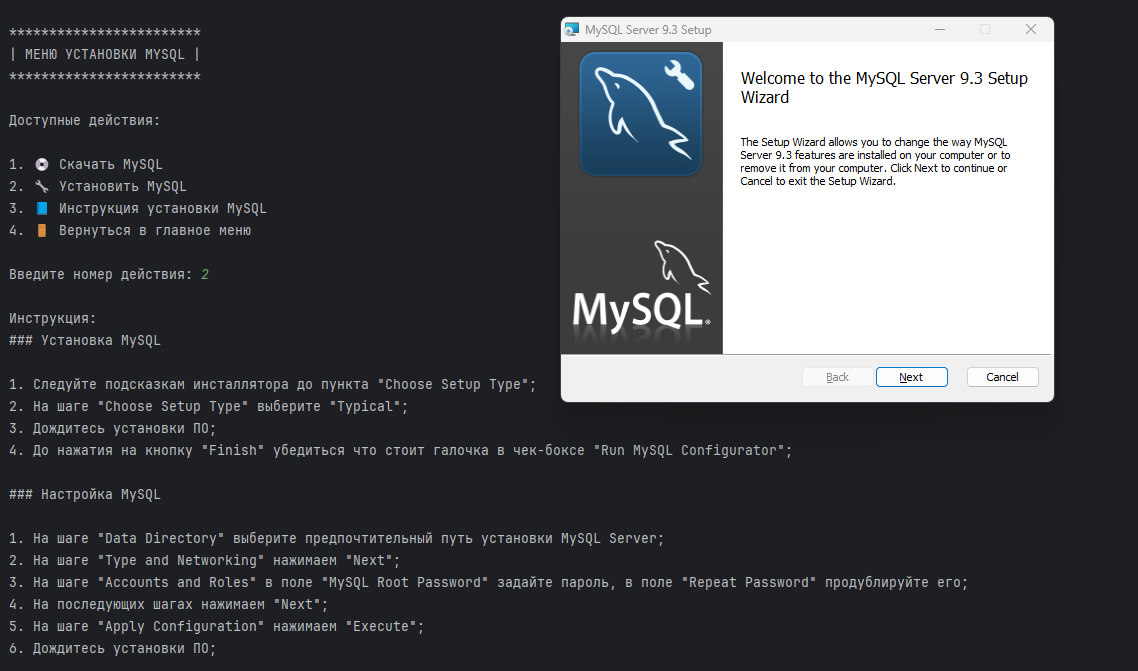
Файл успешно сохраняется в директории:



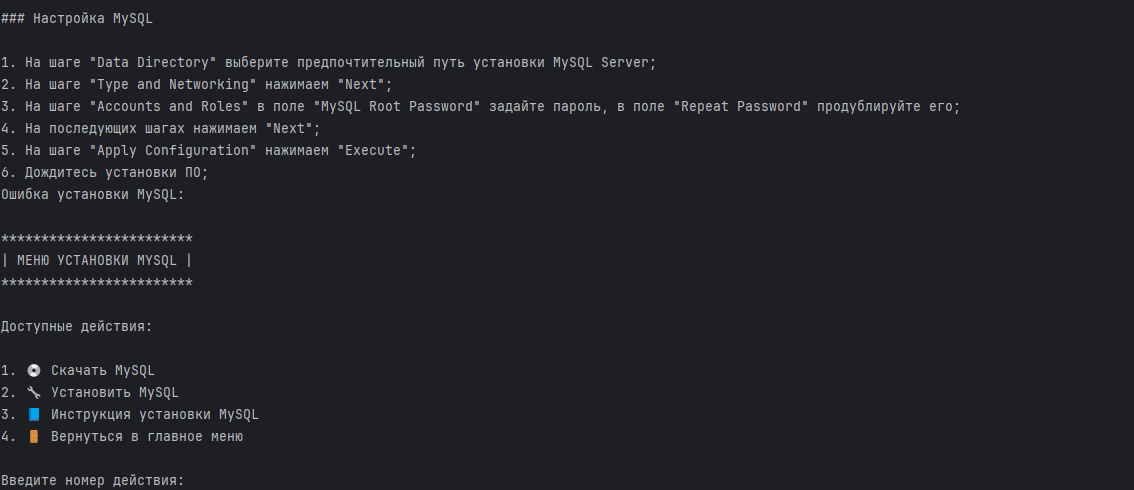
Если MySQL уже загружен:



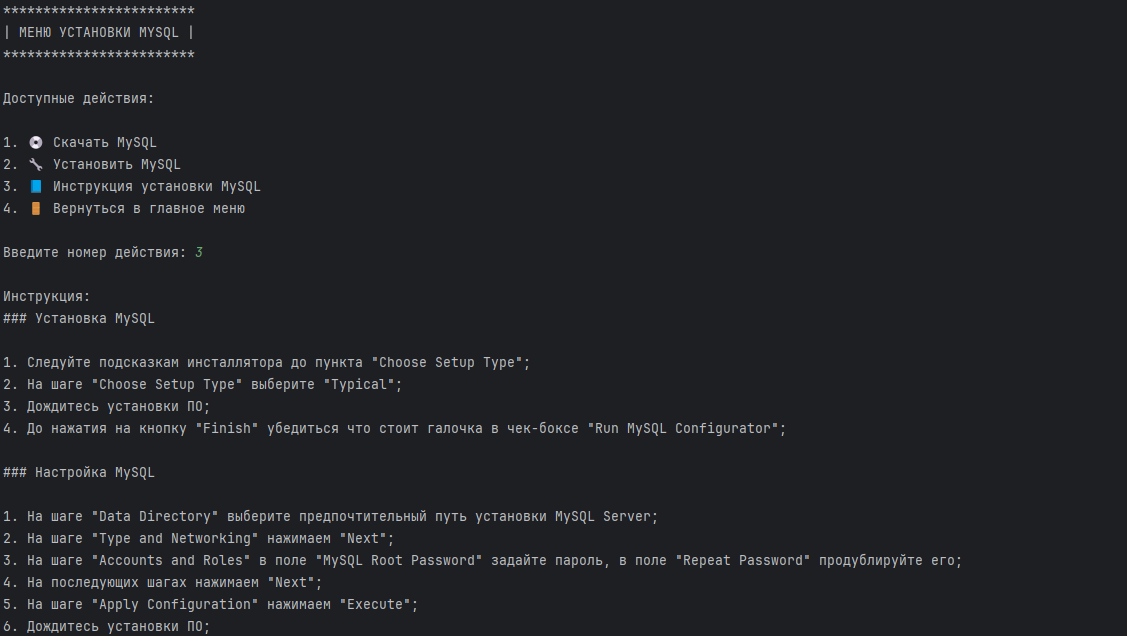
Установка MySQL:



Возврат в меню после завершения установки:



Вызов инструкции:



Возврат в главное меню:



В процессе проверки было установлено, что меню установки MySQL отображается корректно, с соблюдением заданного форматирования и визуальной структуры. Все доступные действия (скачивание, установка MySQL, просмотр инструкции, возврат в главное меню) отображаются в соответствии с требованиями дизайна.

Система валидации пользовательского ввода продемонстрировала стабильную работу. При вводе корректных числовых значений (от 1 до 4) происходит ожидаемое выполнение соответствующих функций. В случаях некорректного ввода (буквы, числа вне диапазона) система корректно обрабатывает ошибки и информирует пользователя о необходимости повторного ввода.

## **Тестирование модуля настройки службы MySQL и создания базы данных**

Данный раздел посвящен проверке функциональности меню, отвечающего за настройку службы MySQL и создание базы данных. Основное внимание уделяется корректности отображения меню, обработке пользовательского ввода и выполнению соответствующих действий при выборе различных опций.

Тестирование направлено на проверку следующих ключевых аспектов:

* корректность отображения меню настроек
* работоспособность функций управления службой MySQL
* процесс создания базы данных
* механизм валидации пользовательского ввода
* обработку ошибок при некорректном вводе
* корректность работы с параметром service

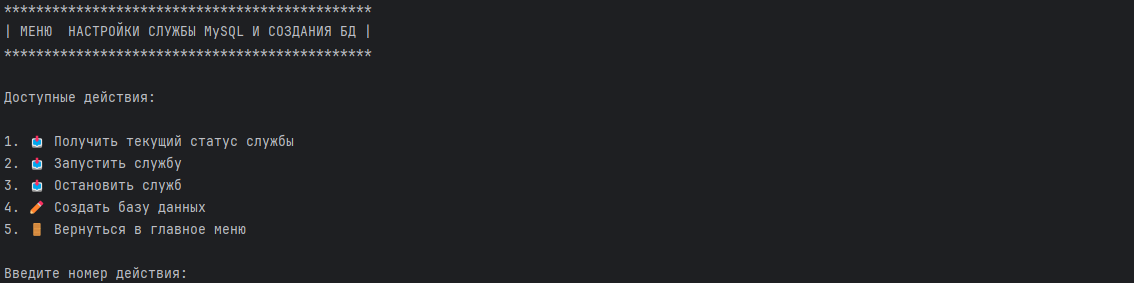
Особое внимание уделяется проверке взаимодействия между компонентами модуля, включая корректную передачу параметров в функции service\_info() и create\_database(), а также обработку различных сценариев использования меню.

В процессе тестирования проверяется как корректная работа при правильном вводе данных пользователем, так и обработка ошибок при вводе некорректных значений или символов. Также тестируется механизм возврата в главное меню и корректность отображения предупреждающих сообщений.

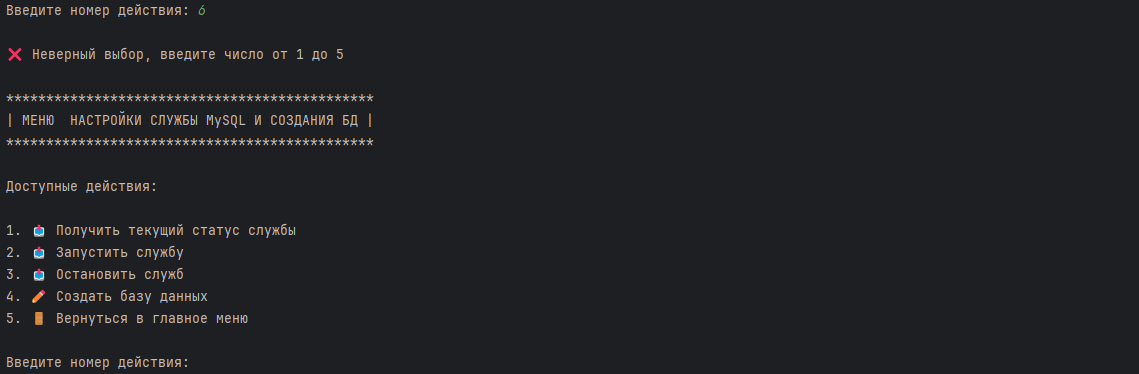
Успешное прохождение всех тестовых сценариев подтверждает готовность модуля к практическому применению и его соответствие заявленным требованиям по настройке службы MySQL и созданию базы данных.

**Тесты**

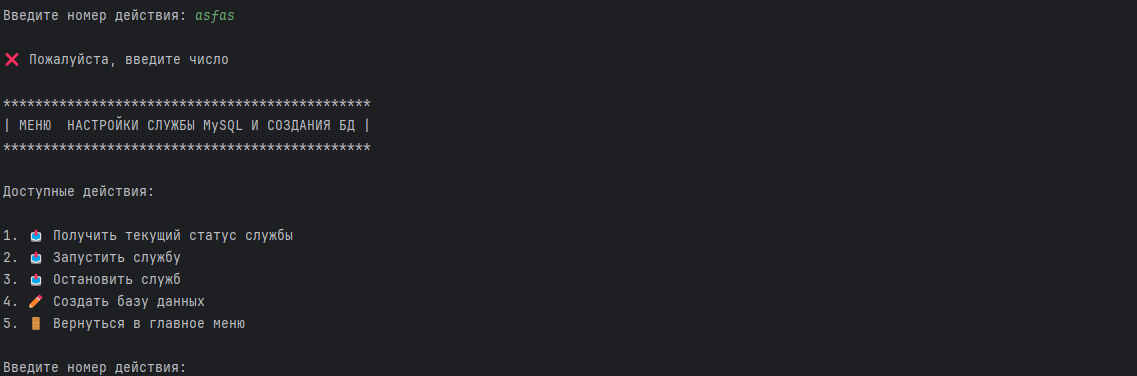
Главное меню:



Число больше 5:



Текст:



Пустое значение:



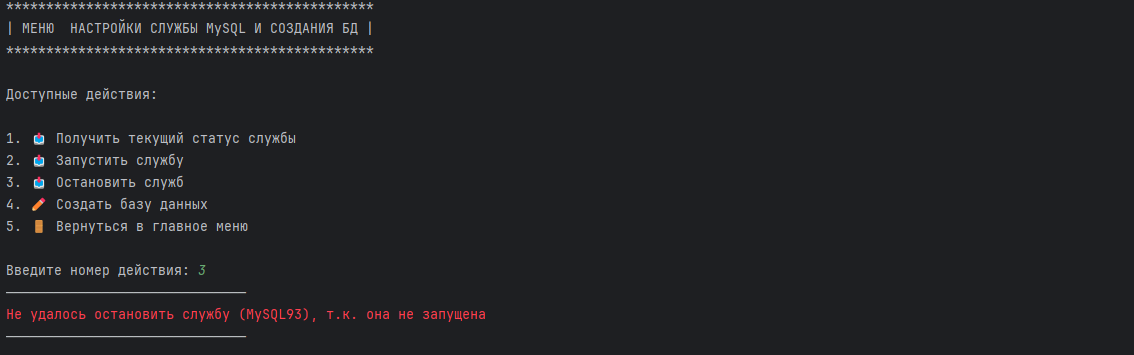
Получение текущего статуса службы:



Остановка службы:



Повторная остановка службы:

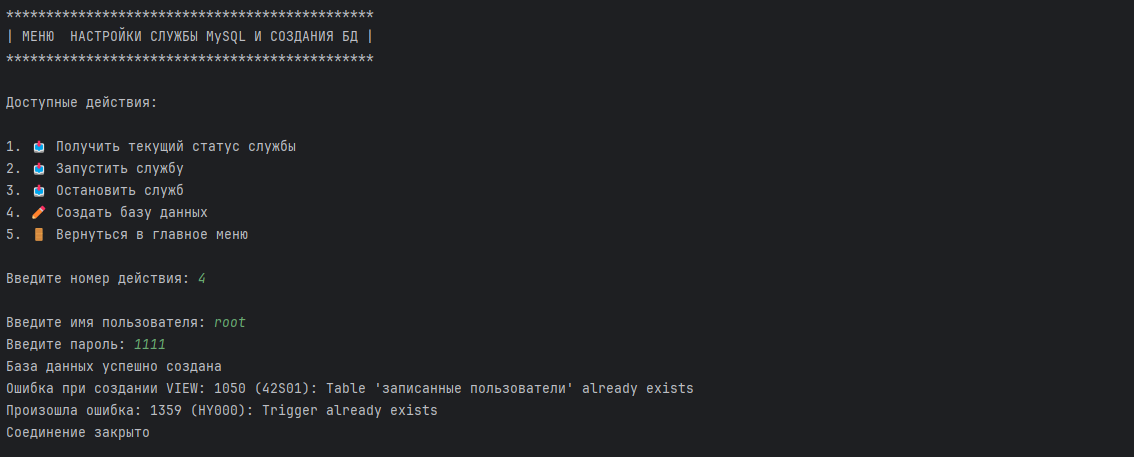


Запуск службы:

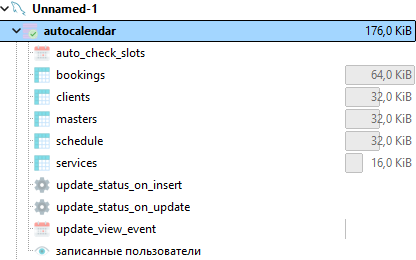


Создание базы данных:

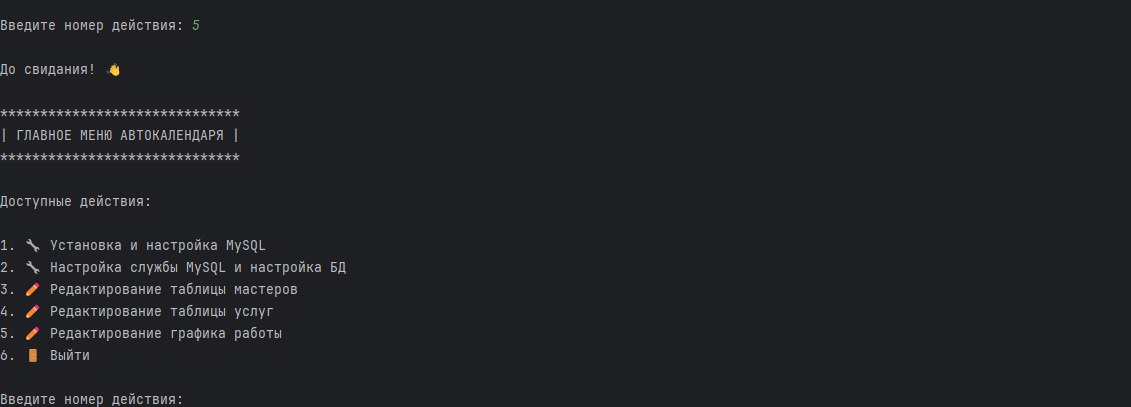
\*создавалась ранее



Проверка структуры на сервере через HeidiSQL:



Возврат в главное меню:



Тестирование проводилось с целью проверки корректности работы меню настройки службы MySQL и создания базы данных. В ходе проверки были проанализированы различные сценарии использования модуля, включая как стандартные, так и нестандартные ситуации взаимодействия с пользователем.

По результатам проведенного тестирования можно сделать вывод о стабильной работе модуля настройки службы MySQL. Все функциональные компоненты демонстрируют ожидаемое поведение, система обработки ошибок работает корректно, интерфейс обеспечивает удобное взаимодействие с пользователем.

## **Тестирование модуля управления списком мастеров**

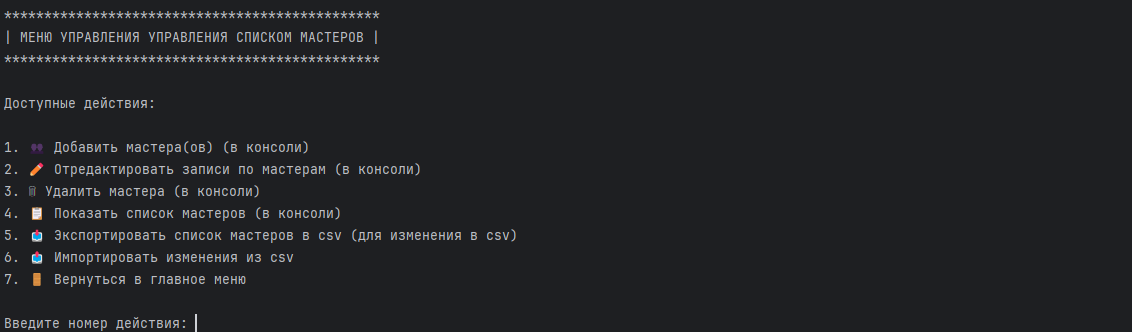
Данный раздел посвящен проверке функциональности меню управления списком мастеров, которое включает в себя полный набор операций по работе с данными: добавление новых записей, редактирование существующих, удаление мастеров, просмотр списка, а также экспорт и импорт данных в формате CSV.

Тестирование направлено на проверку следующих ключевых аспектов:

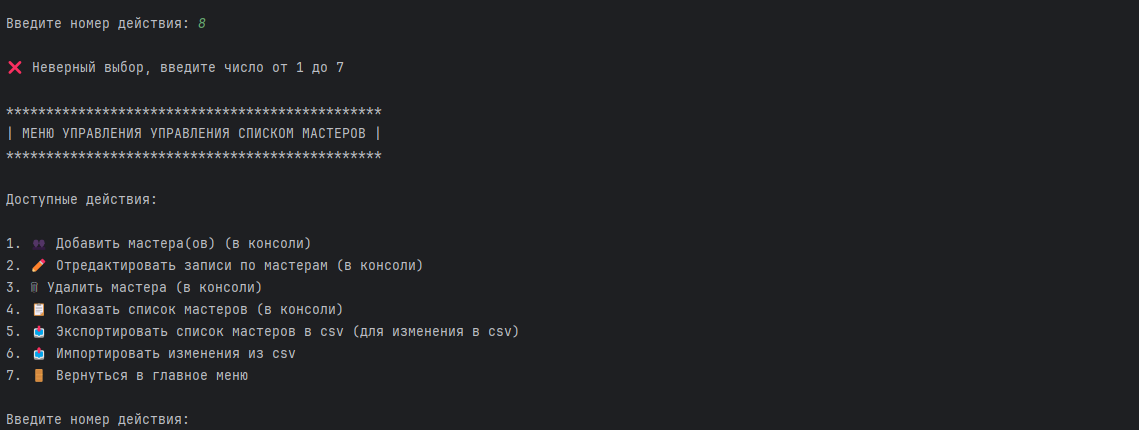
* корректность отображения меню управления
* работоспособность функций добавления, редактирования и удаления мастеров
* корректность отображения списка мастеров
* механизмы экспорта и импорта данных в CSV
* систему валидации пользовательского ввода
* обработку ошибок при некорректном вводе
* взаимодействие между различными функциональными компонентами

**Тесты**

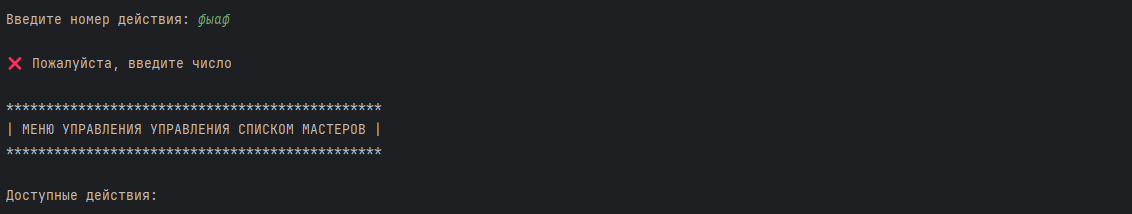
Главное меню:



Число больше 7:



Ввод текстового значения:



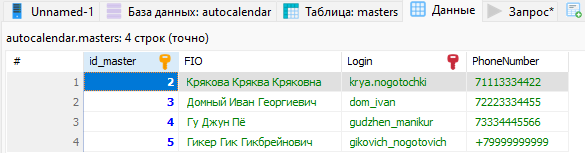
Ввод пустого значения:



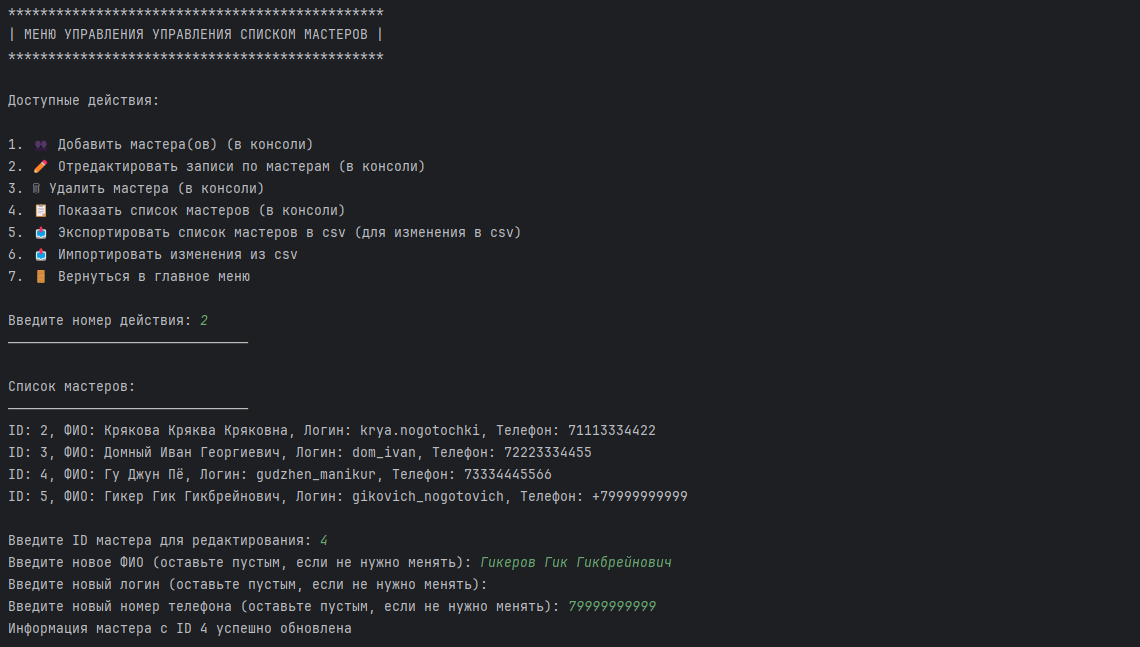
Добавление мастеров:



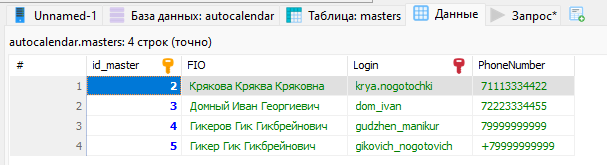
Проверяем запись в БД:



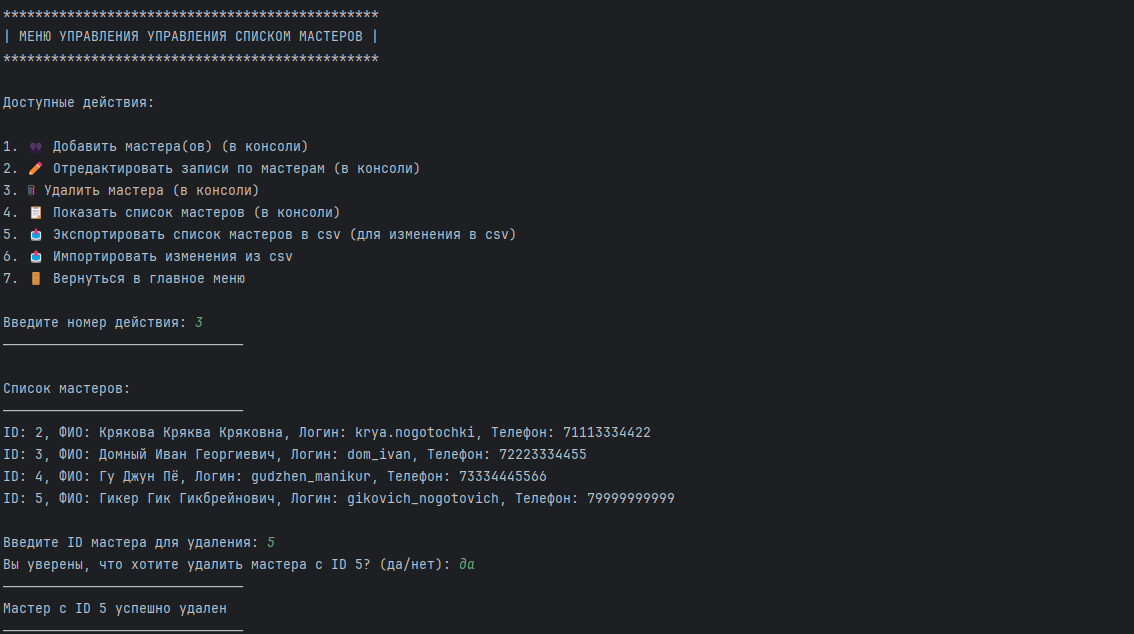
Редактирование мастера:



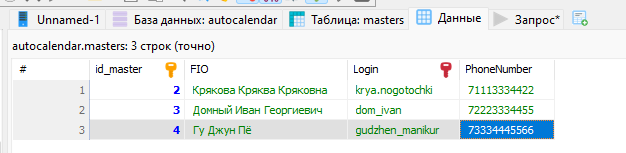
Проверяем изменения в БД:



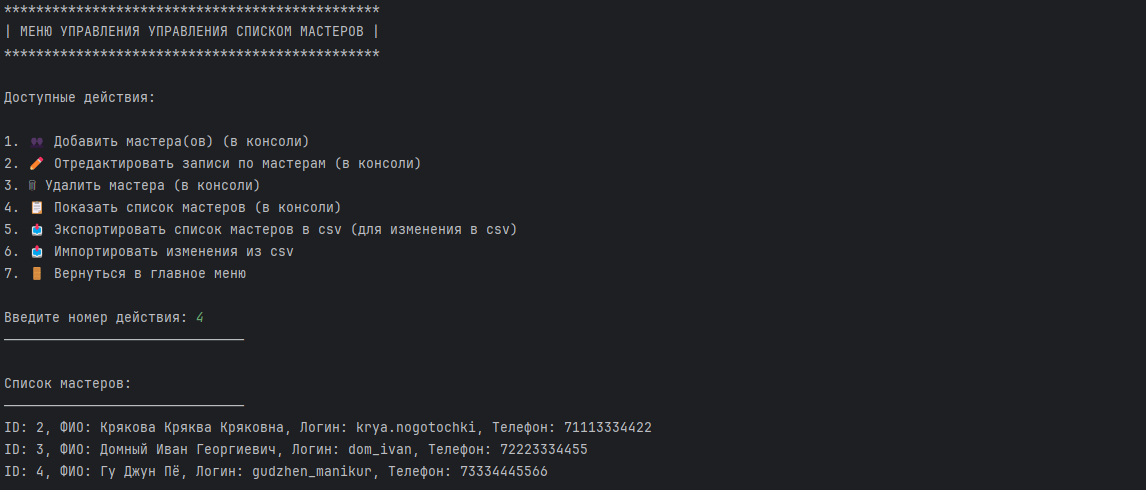
Удаление мастера:



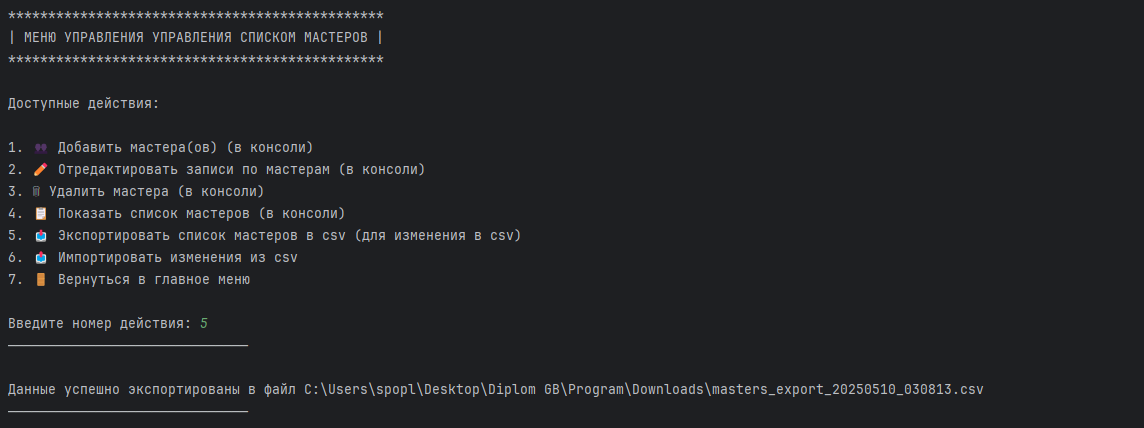
Проверяем в БД:



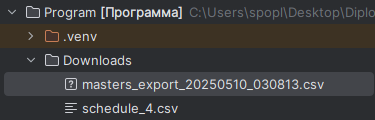
Вывод списка мастеров:



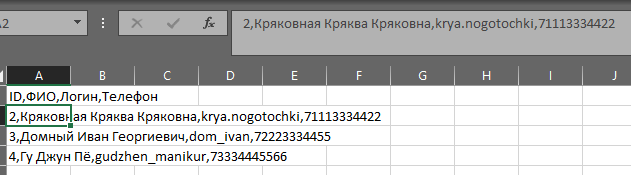
Экспорт в csv:



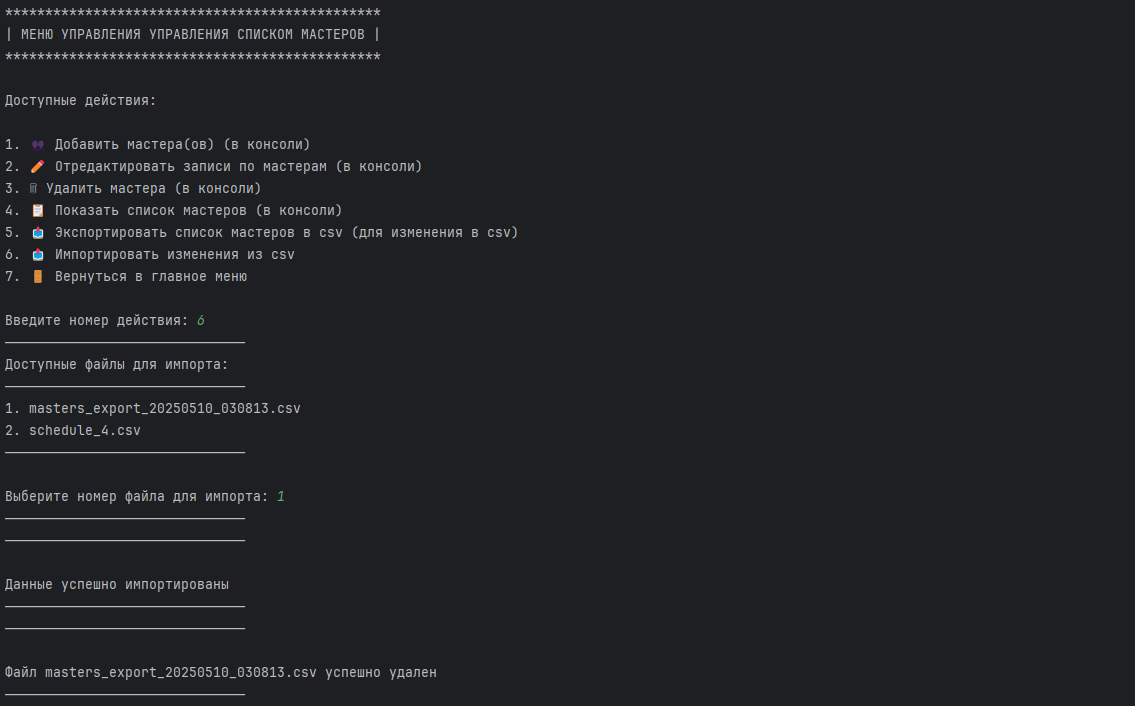
Файл в директории:



Внесение изменений в CSV:



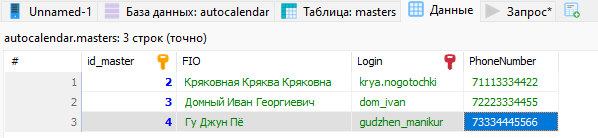
Импорт изменений в БД:



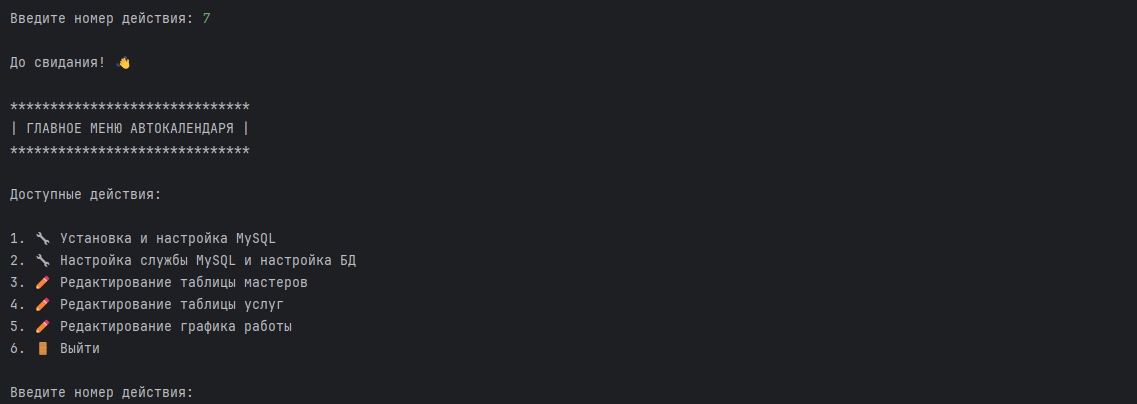
Директория очищена:



Проверка изменений в БД:



Возврат в главное меню:



В процессе проверки было установлено, что меню управления отображается корректно, с соблюдением заданного форматирования и визуальной структуры. Все доступные действия (добавление, редактирование, удаление мастеров, просмотр списка, экспорт и импорт в CSV, возврат в главное меню) отображаются в соответствии с требованиями дизайна.

Система валидации пользовательского ввода продемонстрировала стабильную работу. При вводе корректных числовых значений (от 1 до 7) происходит ожидаемое выполнение соответствующих функций. В случаях некорректного ввода (буквы, числа вне диапазона) система корректно обрабатывает ошибки и информирует пользователя о необходимости повторного ввода.

По результатам проведенного тестирования можно сделать вывод о стабильной работе модуля управления списком мастеров. Все функциональные компоненты демонстрируют ожидаемое поведение, система обработки ошибок работает корректно, интерфейс обеспечивает удобное взаимодействие с пользователем.

## **Тестирование модуля управления списком услуг**

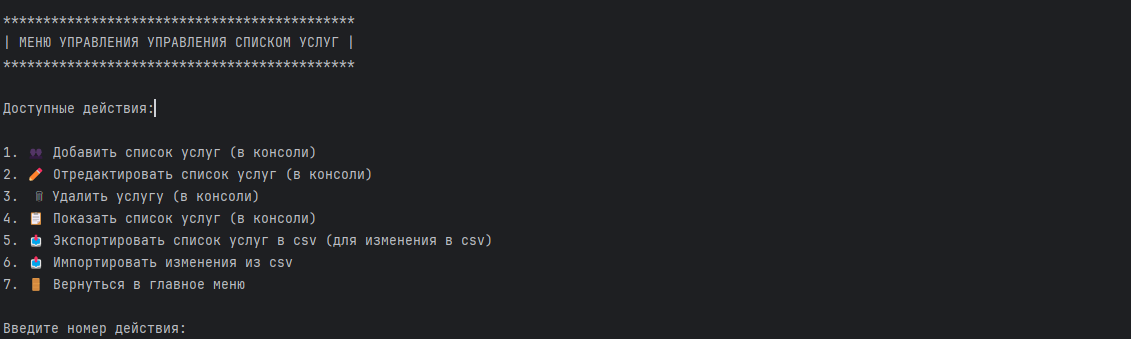
Данный раздел посвящен проверке функциональности меню управления списком услуг, которое включает в себя полный набор операций по работе с данными: добавление новых услуг, редактирование существующих, удаление услуг, просмотр списка, а также экспорт и импорт данных в формате CSV.

Тестирование направлено на проверку следующих ключевых аспектов:

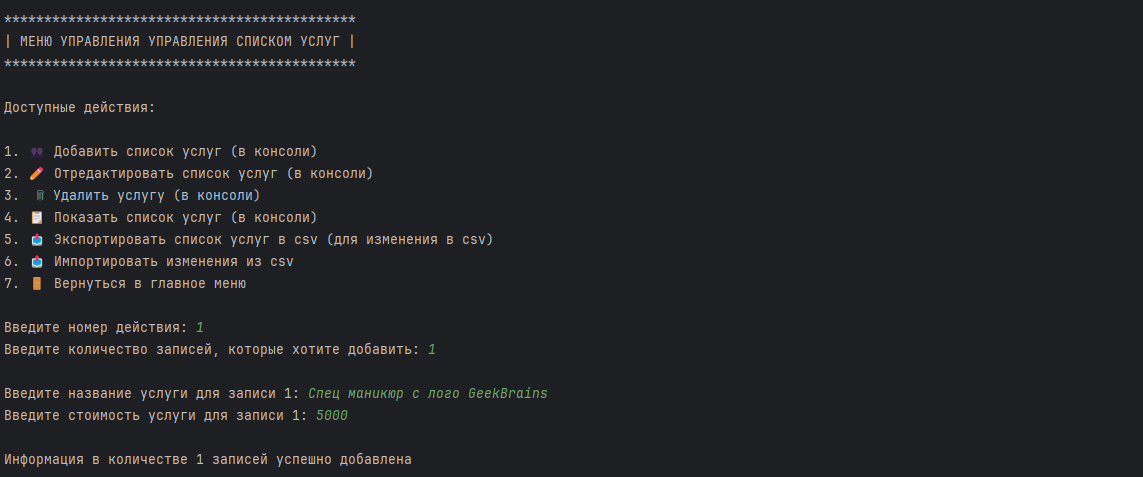
* корректность отображения меню управления
* работоспособность функций добавления, редактирования и удаления услуг
* корректность отображения списка услуг
* механизмы экспорта и импорта данных в CSV
* систему валидации пользовательского ввода
* обработку ошибок при некорректном вводе
* взаимодействие между различными функциональными компонентами

**Тесты**

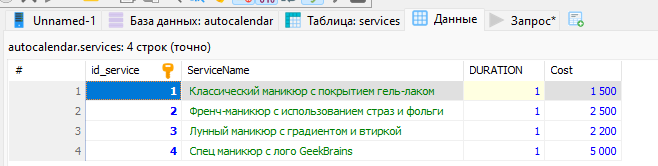
Главное меню:



Добавление услуг:



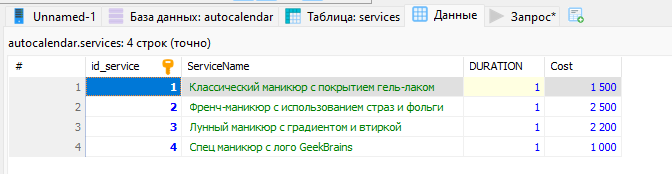
Проверка в БД:



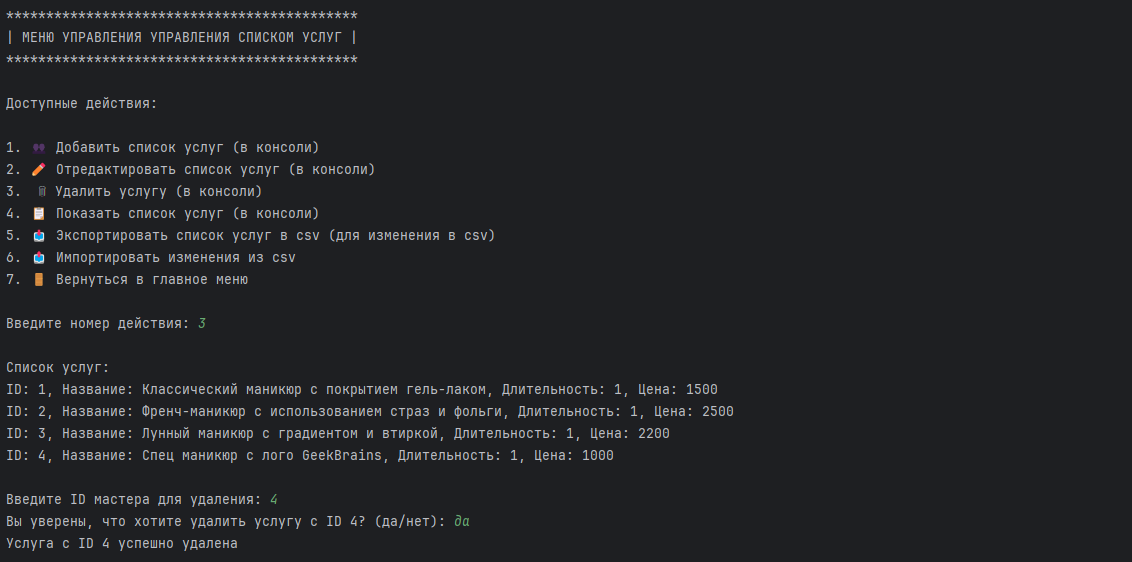
Редактирование услуг:



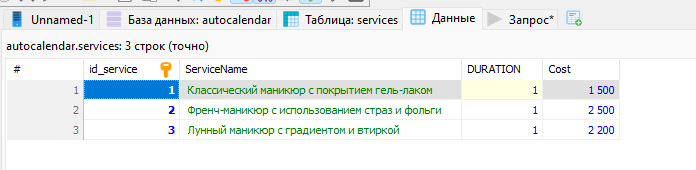
Проверка в БД:



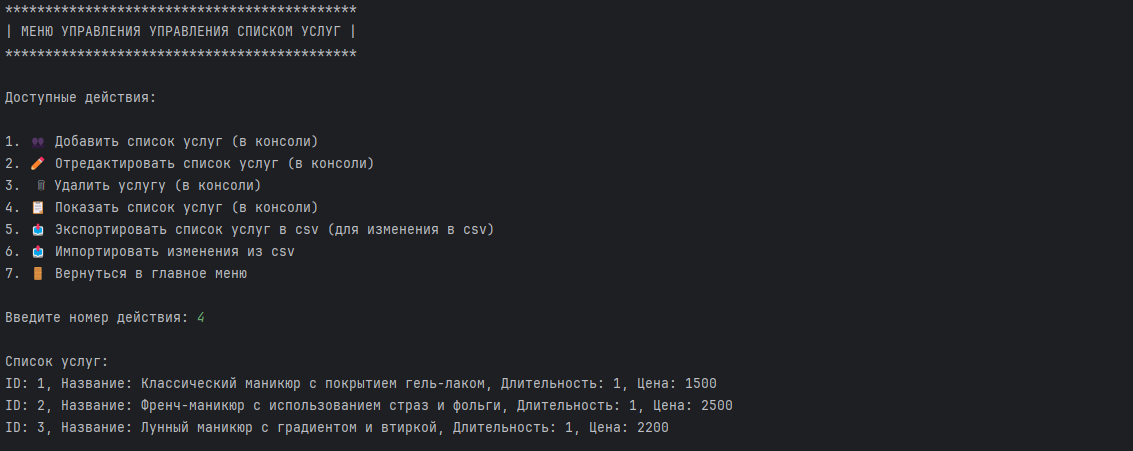
Удаление услуги:



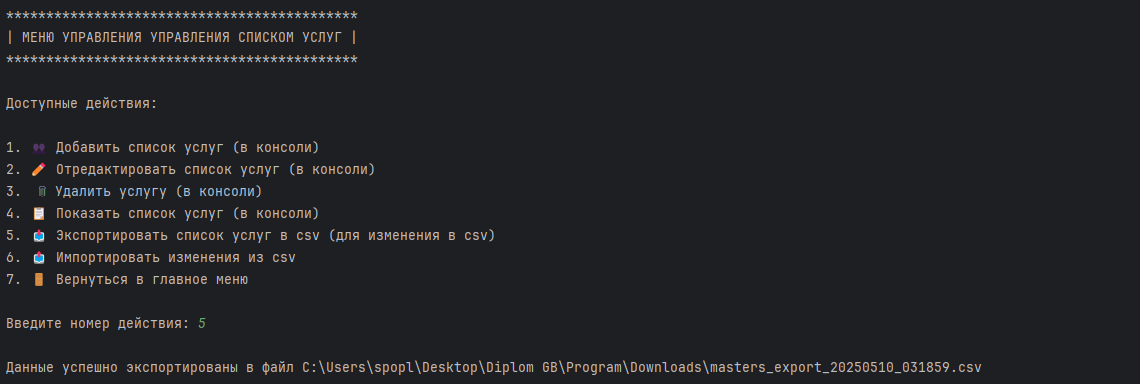
Проверка в БД:



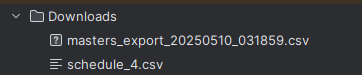
Отображение списка услуг:



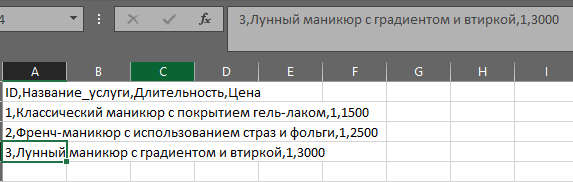
Экспорт списка услуг:



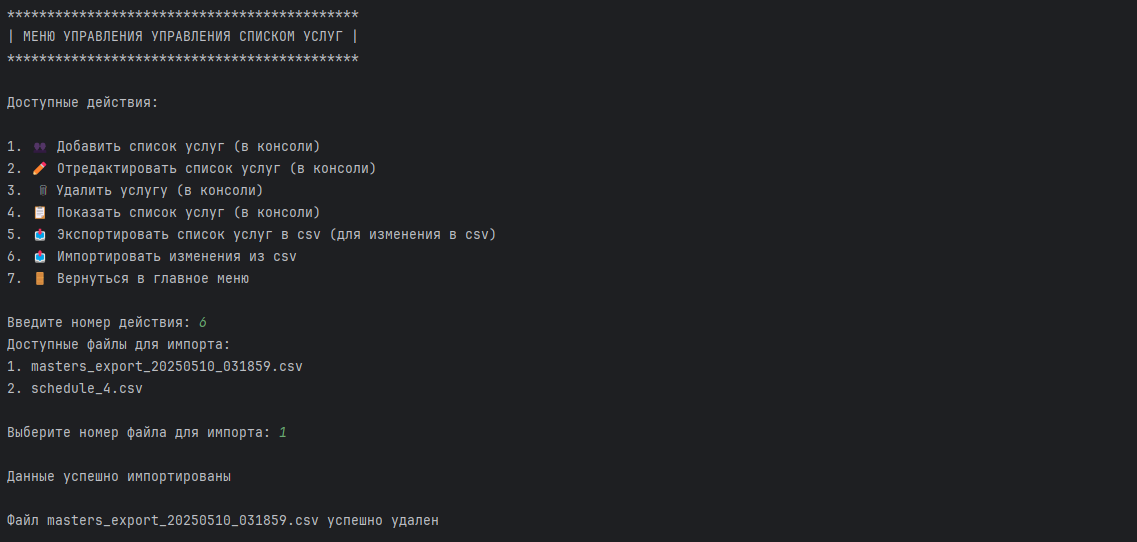
В директории:



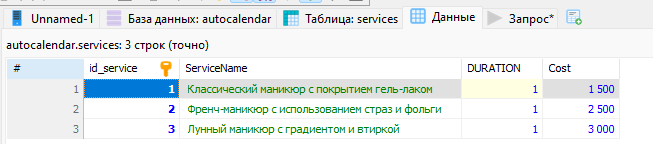
Внесение изменений в csv (цена в id 3):



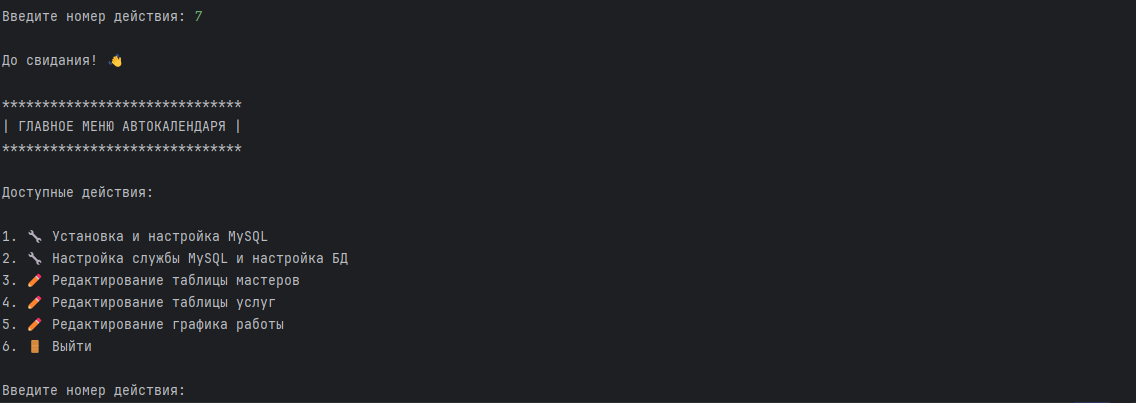
Импорт изменений из csv:



Проверка в БД:



Возврат в главное меню:



Тестирование проводилось с целью проверки корректности работы меню управления списком услуг и всех его функциональных компонентов. В ходе проверки были проанализированы различные сценарии использования модуля, включая как стандартные, так и нестандартные ситуации взаимодействия с пользователем.

В процессе проверки было установлено, что меню управления отображается корректно, с соблюдением заданного форматирования и визуальной структуры. Все доступные действия (добавление, редактирование, удаление услуг, просмотр списка, экспорт и импорт в CSV, возврат в главное меню) отображаются в соответствии с требованиями дизайна.

По результатам проведенного тестирования можно сделать вывод о стабильной работе модуля управления списком услуг. Все функциональные компоненты демонстрируют ожидаемое поведение, система обработки ошибок работает корректно, интерфейс обеспечивает удобное взаимодействие с пользователем.

## **Тестирование модуля управления рабочим расписанием**

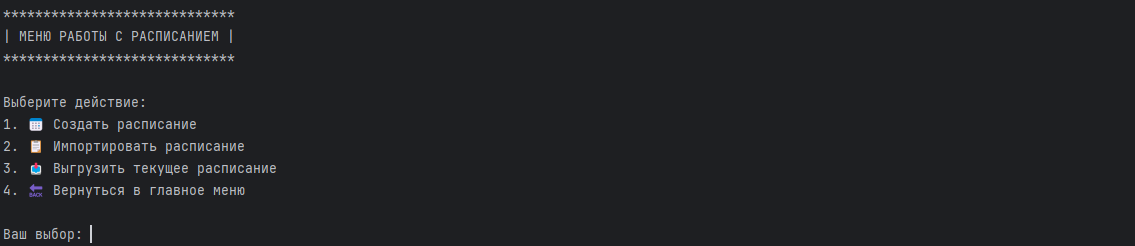
Данный раздел посвящен проверке функциональности меню управления рабочим расписанием, которое включает в себя ключевые операции по работе с графиками мастеров: создание нового расписания, импорт существующих графиков, экспорт текущих расписаний, а также валидацию пользовательского ввода и обработку ошибок.

Тестирование направлено на проверку следующих ключевых аспектов:

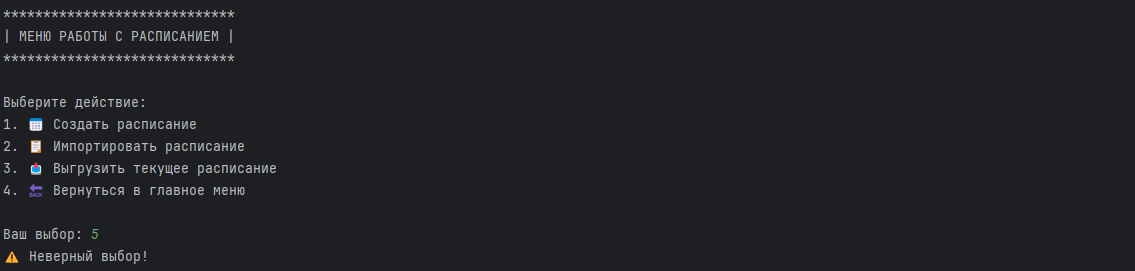
* корректность отображения меню управления расписанием
* работоспособность функции создания нового расписания с валидацией дат и времени
* механизмы импорта и экспорта расписаний в формате CSV
* систему обработки пользовательского ввода
* корректность работы с данными мастеров
* обработку ошибок при некорректном вводе
* взаимодействие между различными функциональными компонентами

**Тесты**

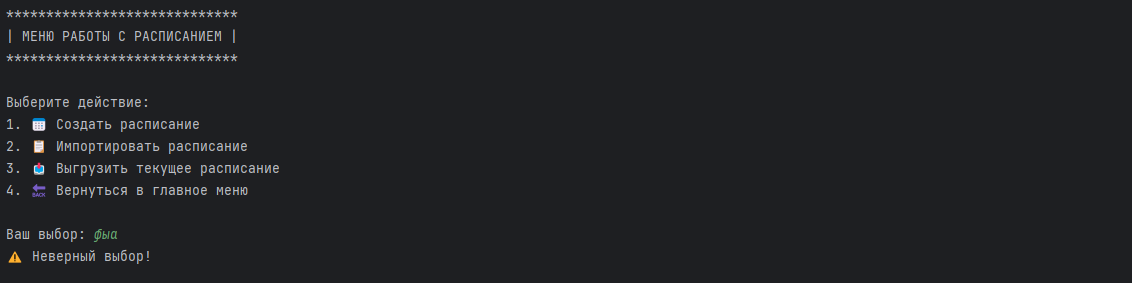
Главное меню:



Ввод больше 4:



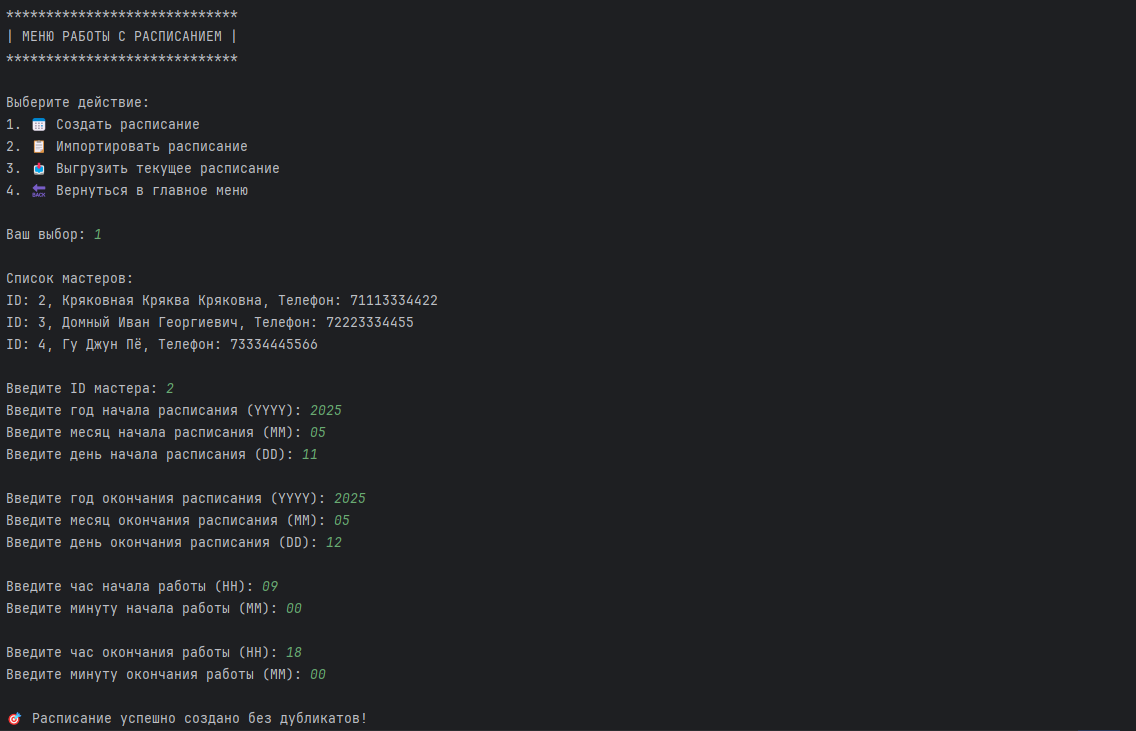
Ввод текста:



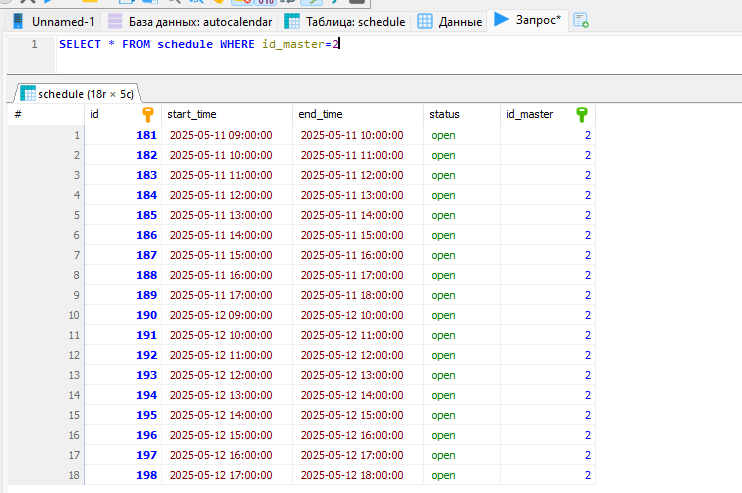
Ввод пустого значения:



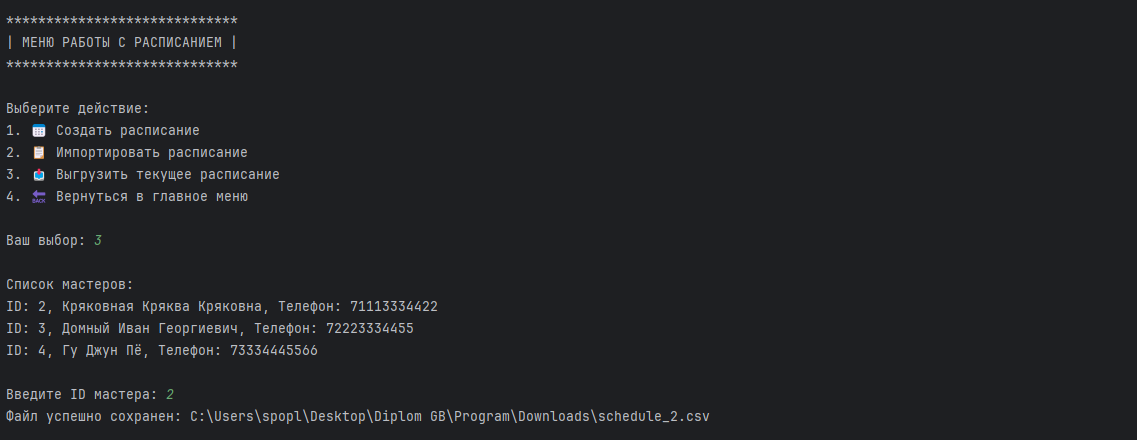
Создание расписания:



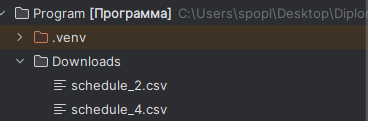
Проверка в БД:



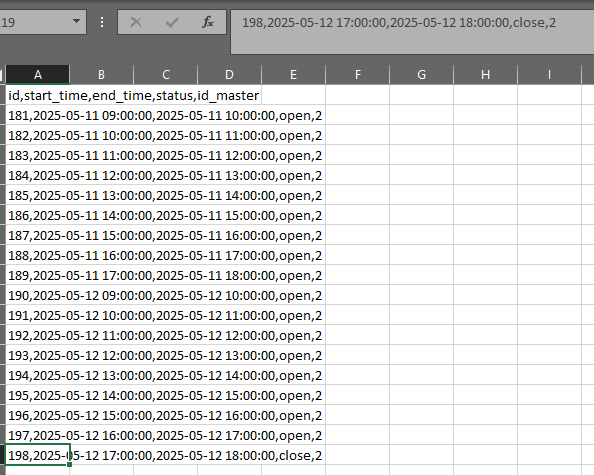
Экспорт расписания:



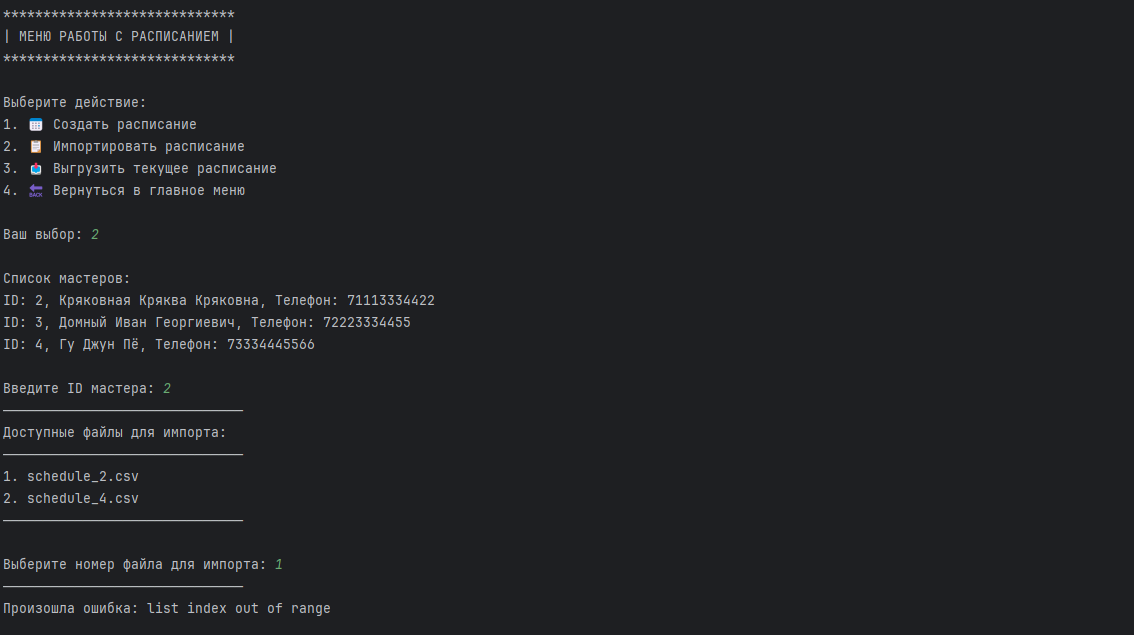
В директории:



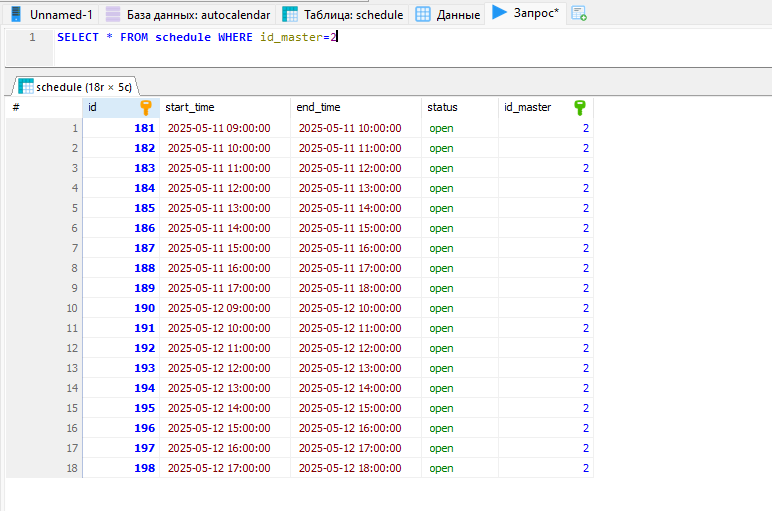
Изменение файла:



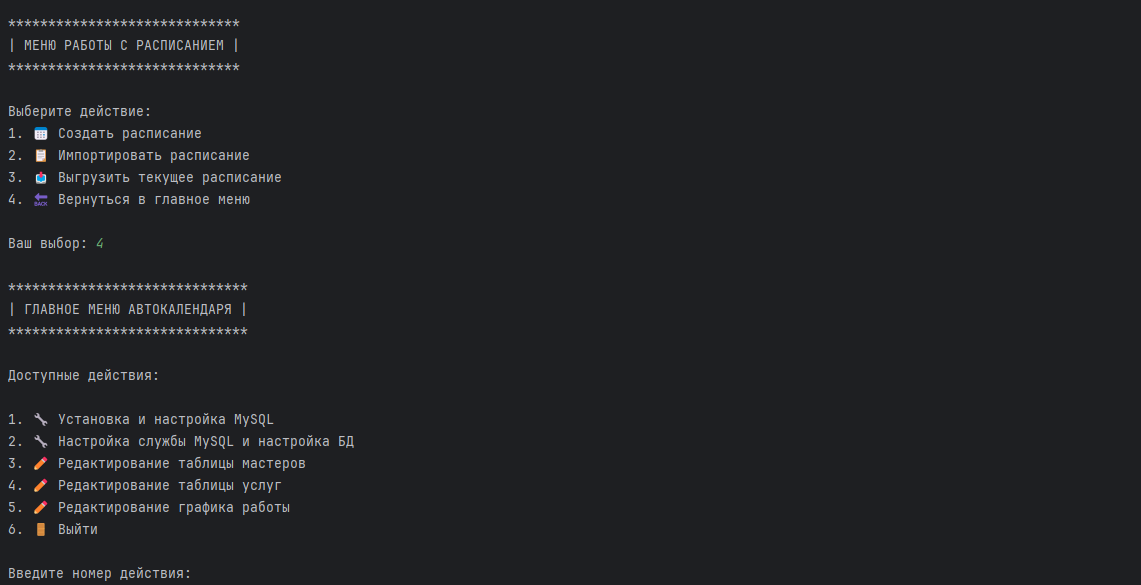
Импорт расписания:



Проверка в БД:



Возврат в главное меню:



Тестирование проводилось с целью проверки корректности работы меню управления расписанием и всех его функциональных компонентов. В ходе проверки были проанализированы различные сценарии использования модуля, включая как стандартные, так и нестандартные ситуации взаимодействия с пользователем.

В процессе проверки было установлено, что меню управления расписанием отображается корректно, с соблюдением заданного форматирования и визуальной структуры. Однако при тестировании функции импорта расписания были выявлены проблемы в работе данного компонента.

При попытке импортировать расписание (вариант 2 в меню) система корректно отображает список мастеров и запрашивает ввод ID. После указания ID мастера и выбора CSV-файла возникает ошибка несмотря на то, что файл существует и имеет корректный формат.

По результатам проведенного тестирования можно сделать вывод о стабильной работе большинства функциональных компонентов модуля. Однако функция импорта расписания требует доработки.

**Общие итоги:** Приложение находится в работоспособном состоянии и готово к использованию. Большинство критических ошибок устранены, базовые функции работают стабильно.

## **Тестирование Telegram-бота системы управления записями**

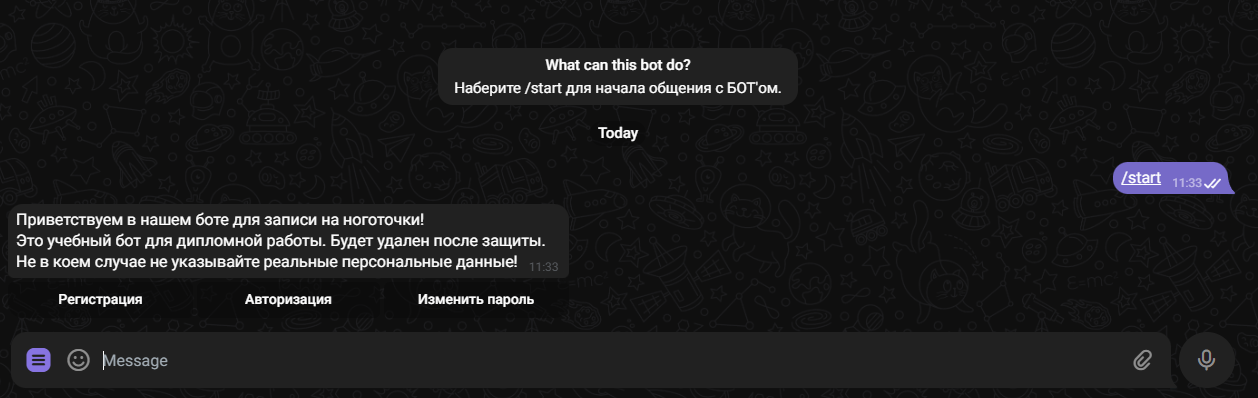
Бот предоставляет пользователям комплексный набор функций, включающий регистрацию новых аккаунтов, процедуру авторизации с подтверждением личности, возможность смены учетных данных, а также полный спектр операций по работе с записями к мастерам: от создания новой записи до ее отмены и проверки статуса. Тестирование охватывает базовые сценарии, проверку корректность работы системы уведомлений.

Тестирование направлено на проверку следующих ключевых аспектов:

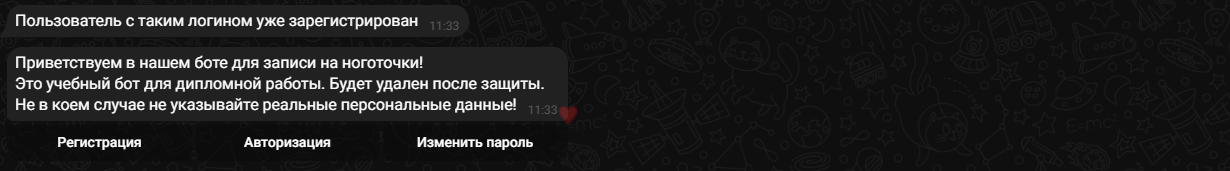
* Тестирование сценария регистрации
* Тестирование сценария авторизации
* Тестирование сценария смены пароля
* Тестирование сценария при неверном вводе пароля
* Тестирование сценария записи
* Тестирование отображения записей для авторизованных пользователей
* Тестирование сценария отмены записи

**Тесты**

Инициализация посредством отправки команды /start:



Тестирование сценария регистрации:

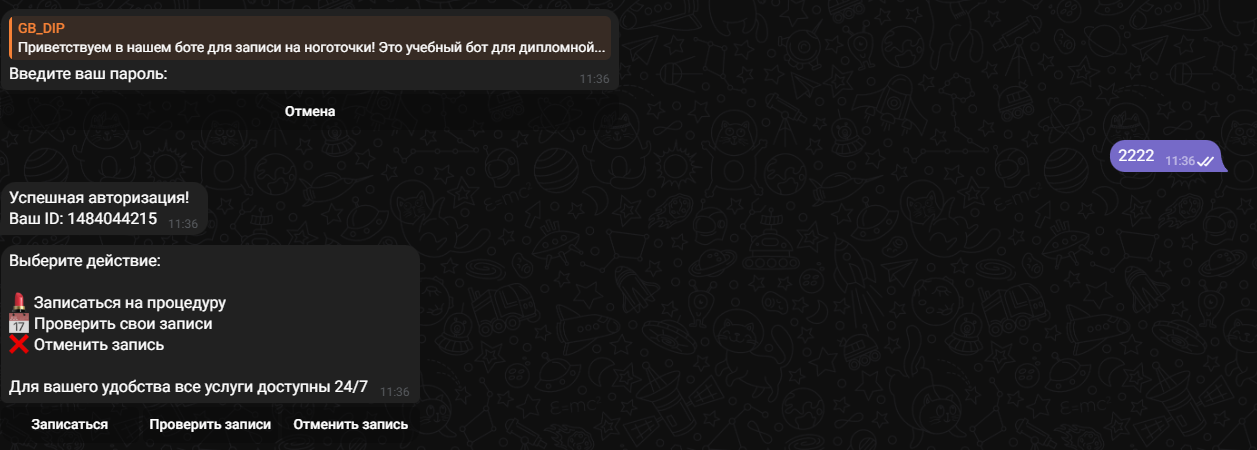


Т.к. в системе зарегистрирован пользователь с моим user\_id, то выводится сообщение о том, что пользователь уже зарегистрирован

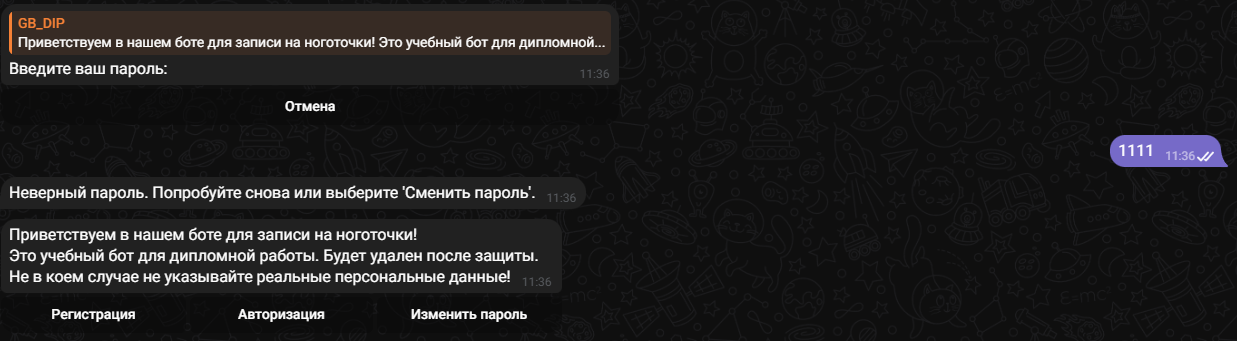
Проверка наличия данных в БД:



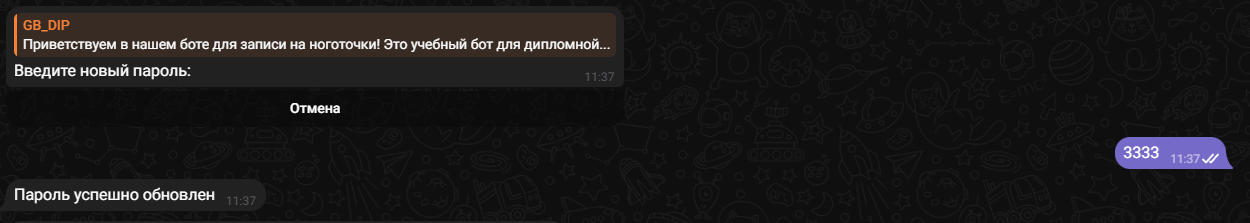
Тестирование сценария авторизации:



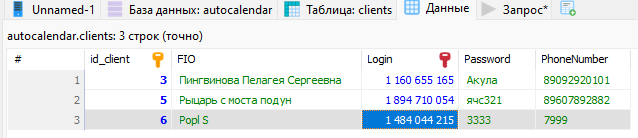
Тестирование сценария при неверном вводе пароля:



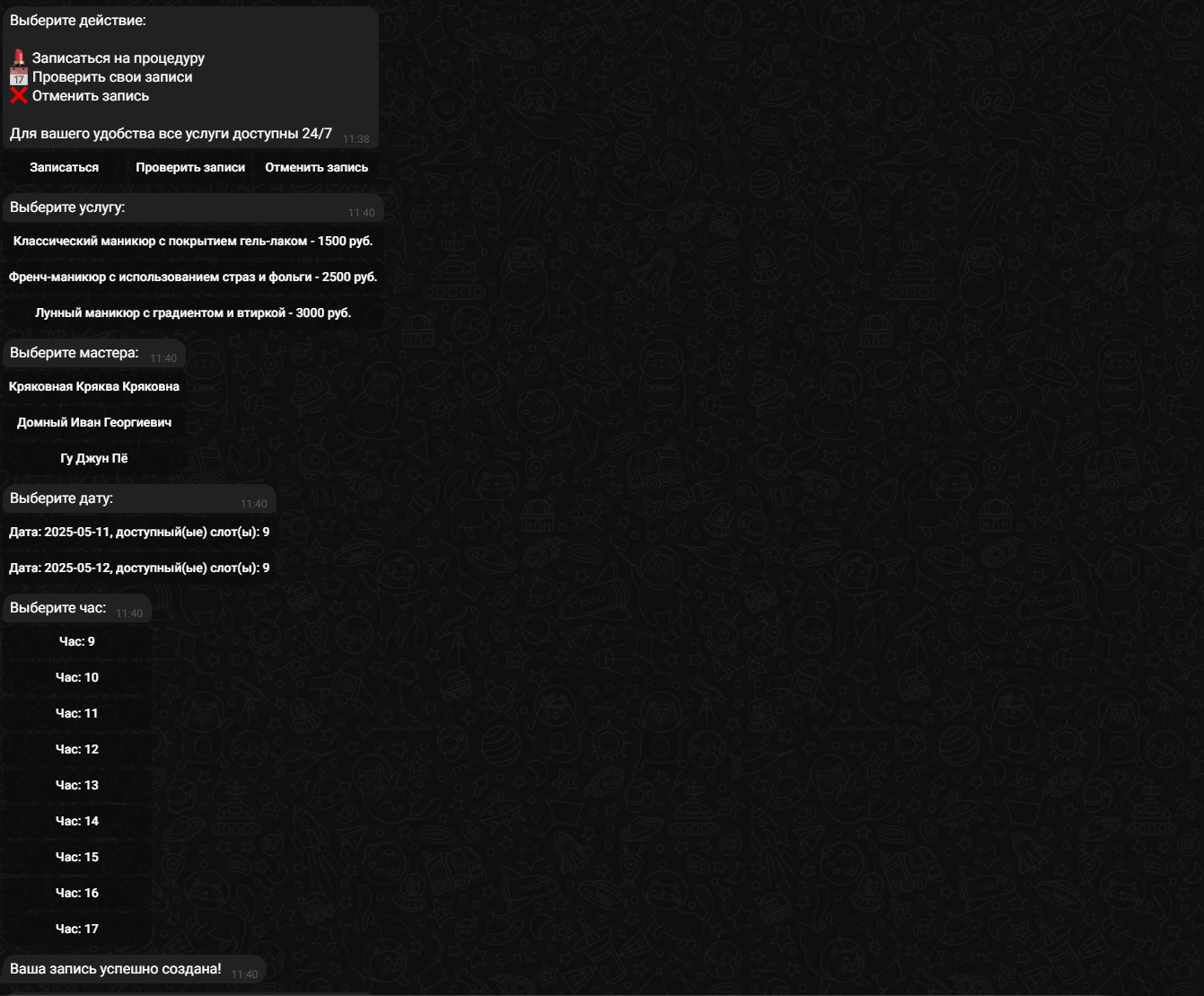
Тестирование сценария смены пароля:



Проверяем запись в БД:



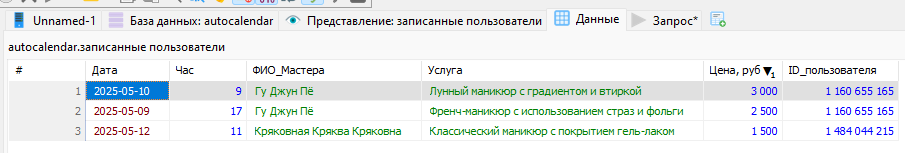
Тестирование сценария записи (выбран мастер Кряковная Кряква Кряковна):



При каждом нажатии на кнопку пользователь информируется о выборе:



Проверка в БД:

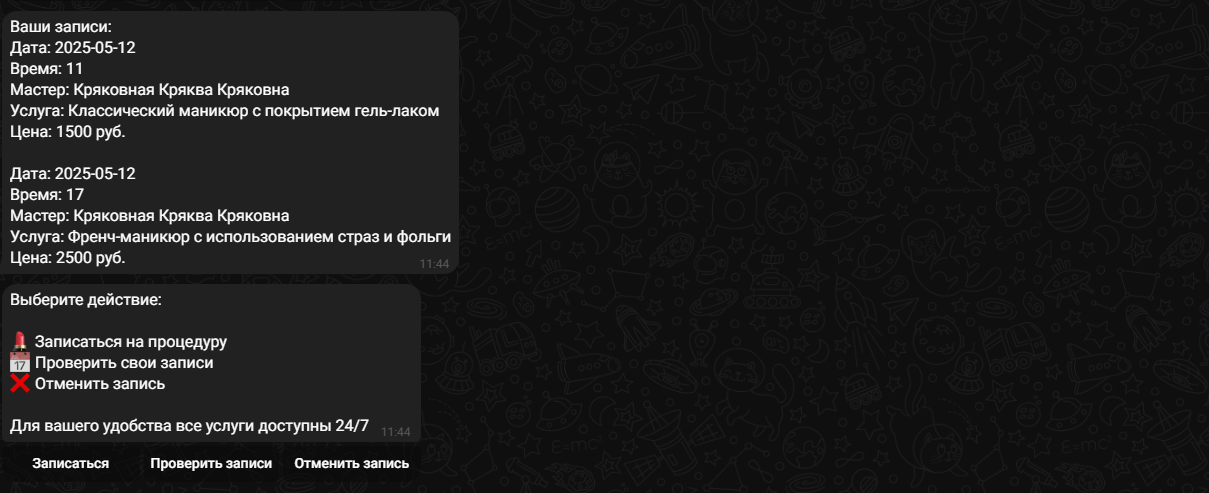


Тестирования сценария повторной записи на тоже время:

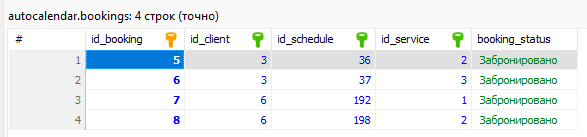


Как видно повторная запись на 11 час недоступна.

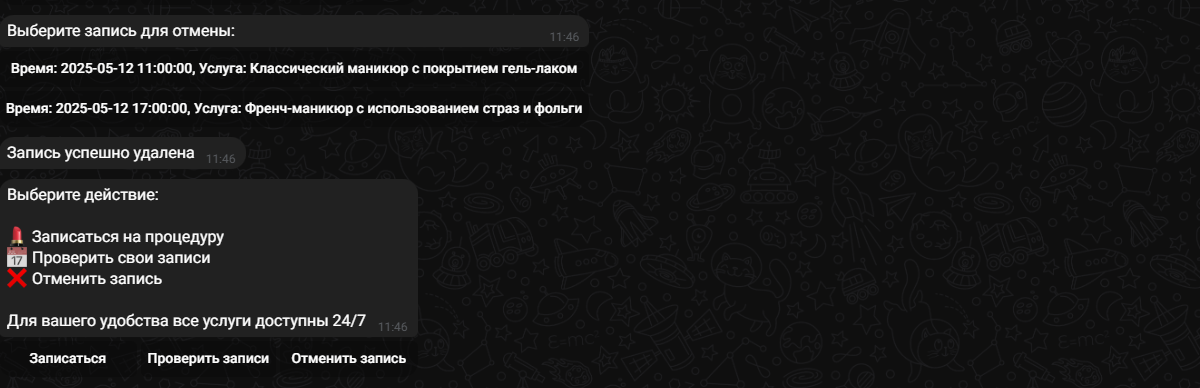
Тестирование отображения записей для авторизованных пользователей



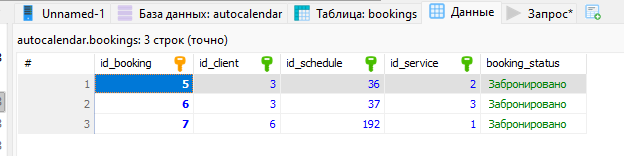
Проверка в БД (в БД у моего пользователя id\_client = 6):



Тестирования сценария отмены записи:



Проверка в БД:



**Общий итог:** в результате проведения комплексного тестирования всех функциональных компонентов Telegram-бота можно сделать вывод о его стабильной и корректной работе во всех тестируемых сценариях. По результатам тестирования можно сделать вывод, что Telegram-бот полностью готов к эксплуатации и соответствует всем заявленным требованиям к функциональности, безопасности и удобству использования.

# **Заключение:**

**Теоретические выводы**

В ходе проведенного исследования были изучены существующие системы автоматизации заказов, проанализированы их функциональные возможности и выявлены ключевые преимущества и недостатки. На основе сравнительного анализа были определены основные направления для разработки собственного решения, учитывающего современные требования к системам управления клиентскими заказами.

**Результаты практической разработки**

Разработанная система автоматизации заказов представляет собой инновационное решение, где в качестве CRM-системы выступает Telegram-бот, интегрированный с базой данных MySQL. Такой подход позволил создать гибкую и масштабируемую платформу для управления заказами, которая может быть адаптирована под различные бизнес-процессы.

**Достигнутые цели и задачи**

В процессе работы были успешно решены следующие задачи:

* Анализ существующих решений - проведен детальный анализ существующих систем автоматизации обработки заказов. Были изучены ключевые функциональные возможности, преимущества и недостатки конкурирующих решений. На основе проведенного анализа были выделены основные требования к разрабатываемой системе и определены направления для создания собственного решения.
* Проектирование архитектуры системы - разработана архитектура системы, включающая интеграцию Telegram-бота с базой данных MySQL. Создана схема взаимодействия компонентов системы, определены основные модули и их функциональность. Проведен выбор технологий и инструментов разработки.
* Разработка и тестирование программного обеспечения. Реализован базовый функционал системы, включающий - работу с пользовательскими данными, управление заказами, интеграцию с базой данных, базовую систему аутентификации
* Проведено тестирование всех реализованных компонентов, выявлены основные проблемы и направления для улучшения.
* Внедрение системы - разработанное решение было внедрено в тестовую среду, где была проверена его работоспособность и совместимость с различными компонентами. Проведена настройка окружения и развертывание системы.
* Оценка эффективности - проведена предварительная оценка эффективности внедренного решения. Определены основные показатели производительности системы, выявлены ключевые преимущества и ограничения. Сформированы рекомендации по дальнейшему развитию и улучшению системы.

**Практическая значимость**

Разработанное решение демонстрирует возможность создания эффективной системы автоматизации заказов с использованием современных технологий. Важно отметить, что разработка подобной системы не требует значительных временных и финансовых затрат, что делает её привлекательной для малого и среднего бизнеса.

**Рекомендации по внедрению**

Для успешного внедрения и эксплуатации системы рекомендуется:

* Провести полный аудит безопасности
* Реализовать систему резервного копирования данных
* Разработать документацию для пользователей и администраторов
* Организовать техническую поддержку
* Провести тестирование производительности

**Общие выводы**

Проведенное исследование и разработка собственного решения позволили сделать следующие выводы:

* Современные технологии, такие как Telegram API и MySQL, предоставляют широкие возможности для создания эффективных CRM-систем
* Интеграция мессенджера в систему автоматизации заказов повышает удобство использования
* Базовые функции управления заказами могут быть реализованы в короткие сроки
* Для конкурентоспособности системы необходимо расширение функционала и улучшение пользовательского опыта

**Перспективы развития проекта**

Разработанная система имеет значительный потенциал для дальнейшего развития. При условии реализации запланированных улучшений, система может стать конкурентоспособным решением на рынке автоматизации заказов. Особое внимание следует уделить:

* Интеграции с другими сервисами и платформами
* Внедрению искусственного интеллекта для автоматизации процессов
* Развитию мобильного приложения
* Созданию модульной архитектуры для легкого расширения функционала - частично реализована

В результате проведенной работы была успешно разработана система автоматизации заказов, которая демонстрирует перспективность использования современных технологий для решения бизнес-задач. Несмотря на текущие ограничения, система представляет собой работоспособное решение, которое может быть доработано до уровня конкурентоспособного продукта.

Практическая значимость работы заключается в создании основы для дальнейшего развития системы, которая может быть адаптирована под различные бизнес-процессы и потребности пользователей. Результаты исследования могут быть использованы как для практического применения, так и для дальнейших научных исследований в области автоматизации бизнес-процессов.

Таким образом, цель дипломной работы достигнута, а поставленные задачи успешно решены. Разработанное решение подтверждает возможность создания эффективных систем автоматизации с использованием современных технологий, что открывает новые перспективы для развития данного направления.

# **Список используемой литературы и источников**

* Статья: «[Автоматизация магазина: Рейтинг лучших программ автоматизации магазина розничной торговли в 2024 году](https://vc.ru/services/1396503-avtomatizaciya-magazina-reiting-luchshih-programm-avtomatizacii-magazina-roznichnoi-torgovli-v-2024-godu)» . Автор: @luchshie
* Сайт: «[КонтурМаркет](https://kontur.ru/market/price/52?group=53&count=1&period=1)»
* Сайт: «[SabyRetail](https://saby.ru/retail/food)».
* Сайт: «[1С:Предприятие 8](https://v8.1c.ru/price/#fe26f070-7409-11e6-853d-0050569f2de3)»
* Сайт: «[LiteBox](https://litebox.ru/catalog/avtomatizatsiya-magazina/)»
* Сайт: «[Эватор](https://evotor.ru/category/avtomatizatsiya/)»
* Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015.
* Рихтер Дж. Программирование на платформе Microsoft .NET Framework 4.5. СПб.: Диалектика, 2013.
* Документация по программированию:
* Макконнелл С. Совершенный код. 2-е изд. СПб.: Символ-Плюс, 2017.
* Мартин Р. Чистый код. Создание, анализ и рефакторинг. СПб.: Питер, 2019.
* Official Telebot Documentation. Telegram Bot API. Москва: Telegram LLC, 2023. Bot API | Telegram Developers. URL: https://core.telegram.org/bots/api
* MySQL Reference Manual. Oracle Corporation, 2023.
* MySQL Connector/Python Developer Guide. Oracle Corporation, 2023.
* SQLAlchemy Documentation. URL: https://www.sqlalchemy.org/
* SQLAlchemy: The Database Toolkit for Python. URL: https://docs.sqlalchemy.org/

# **Приложения**

Приложение 1. Сравнение модели монетизации для актуальных бизнес-решений.   
**\*Цены для Н.Новгорода**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Модель монетизации** | | | | | |
| **Показатели** | **Контур.Маркет** | **Saby Retail** | **LiteBox** | **Битрикс24** | **1С:Предприятие** |
| Стоимость внедрения | от 4 410 руб. | от 68 500 руб. | от 2 890 руб. | Бесплатно | от 20 100 руб. |
| Тип подписки | Ежегодная | Ежегодная | Ежемесячная | Ежемесячная | Ежемесячная |
| Дополнительные модули | Есть | Есть | Есть | Есть | Есть |
| Поддержка | Включена в подписку | Включена в подписку | Включена в подписку | Включена в подписку | Платная |
| Масштабирование | 1 точка | До 500 точек | 1 лицензия | 5 гб | За 1 пользователя |

Приложение 2. Сравнение функциональности для актуальных бизнес-решений.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Функциональность** | | | | | |
| **Показатели** | **Контур.Маркет** | **Saby Retail** | **LiteBox** | **Битрикс24** | **1С:Предприятие** |
| Базовые процессы | Учет товаров, склад, продажи, закупки | Торговля, склад, закупки, финансы | Торговля, склад, закупки | CRM, продажи, закупки, склад | Полный учет, бухгалтерия, торговля |
| Интеграции | 1С, банки, онлайн-кассы | 1С, банки, онлайн-кассы, CRM | Онлайн-кассы, банки | Более 1000 сервисов | Все популярные сервисы |
| Мобильная версия | Есть | Есть | Есть | Есть | Есть |
| Аналитические инструменты | Отчеты по продажам, склад, финансы | Глубокий анализ продаж, маржинальность | Базовые отчеты | Глубокая аналитика | Глубокая аналитика |
| Отчетность | Налоговая | Налоговая, управленческая | Налоговая | Гибкая настройка | Полная, включая управленческую |