МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**



**ПРОЕКТ**

Автоматизированная информационная система для транспортной компании

*Дневник в телеграмме*

Направление IT

Руководитель

Володина О.В.

Москва 2025 г.

Содержание

[Список исполнителей 3](#_Toc198153781)

[Заказчик проекта 3](#_Toc198153782)

[1.Актуальность и проблематика проекта 4](#_Toc198153783)

[1.1 Проблема неучтённого времени 4](#_Toc198153784)

[1.2 Проблема в сфере разработки ПО 4](#_Toc198153785)

[1.3 Существующие решения и их недостатки 4](#_Toc198153786)

[1.4 Почему Telegram — оптимальная платформа 5](#_Toc198153787)

[1.5 Роль аналитики и отчётности 6](#_Toc198153788)

[2. Цели и задачи проекта 6](#_Toc198153789)

[2.1 Общая цель 6](#_Toc198153790)

[2.2 Подцели проекта 6](#_Toc198153791)

[2.3 Задачи проекта 7](#_Toc198153792)

[3. Этапы реализации проекта 9](#_Toc198153793)

[3.1 Подготовка и проектирование 9](#_Toc198153794)

[3.2 Настройка окружения 10](#_Toc198153795)

[3.3 Разработка ядра приложения 11](#_Toc198153796)

[3.4 Отчёты и аналитика 11](#_Toc198153797)

[3.6 Тестирование 12](#_Toc198153798)

[4. Результаты по выполненным задачам 13](#_Toc198153799)

[4.1 Реализованные пользовательские функции 13](#_Toc198153800)

[4.2 Структура базы данных 14](#_Toc198153801)

[4.3 Отчёты и аналитика 14](#_Toc198153802)

[4.4 Устойчивость к ошибкам и тестирование 15](#_Toc198153803)

[4.5 Реакция пользователя 15](#_Toc198153804)

[5. Заключение 16](#_Toc198153805)

[5.1 Достижение целей проекта 16](#_Toc198153806)

[5.2 Особенности реализации 16](#_Toc198153807)

[5.3 Потенциал дальнейшего развития 17](#_Toc198153808)

[5.4 Значимость проекта 18](#_Toc198153809)

[6. Список использованных источников 19](#_Toc198153810)

[Техническая документация и библиотеки 19](#_Toc198153811)

[Сторонние ресурсы и инструменты 19](#_Toc198153812)

# 

# Список исполнителей

|  |  |  |
| --- | --- | --- |
| № | ФИО | Группа |
| 1 | Алиханов Богдан Тагирович | 241-337 |
| 2 | Бондаренко Кирилл Андреевич | 241-131 |
| 3 | Науменко Кирилл Николаевич | 241-132 |
| 4 | Пинчук Ренат Сергеевич | 241-132 |
| 5 | Сулейманов Эмиль Шамилевич | 241-132 |
| 6 | Узун Вадим | 241-339 |
| 7 | Садыков Амрах Элхан оглы | 241-351 |

# Заказчик проекта

ООО «А+А Эксист-инфо»

# 1.Актуальность и проблематика проекта

## 1.1 Проблема неучтённого времени

В современном ритме жизни огромное количество задач, действий и отвлекающих факторов делает управление временем критически важным навыком. Однако большинство людей не отслеживают, как именно расходуется их день. В результате:

* теряется продуктивность;
* невозможно оценить фактическую загрузку;
* сложно отличить «работу» от имитации деятельности;
* невозможно предоставить отчёт, если он требуется (например, в фриланс-проектах).

Особенно остро эта проблема проявляется у разработчиков, менеджеров, студентов, исследователей и всех, кто работает на результат, а не по фиксированному графику.

## 1.2 Проблема в сфере разработки ПО

При разработке программного обеспечения часто отсутствует чёткая система сбора статистики о процессе: когда началась работа над задачей, сколько времени было потрачено, были ли прерывания. Это приводит к:

* постоянным переработкам из-за «потерянных» задач;
* отсутствию данных для пост-анализа;
* невозможности доказать проделанную работу (если нет ТЗ или логов);
* завышенным срокам сдачи и недовольству заказчиков.

Telegram-бот, встроенный в привычную среду общения, становится инструментом не только учёта, но и дисциплины. Он позволяет фиксировать все действия в режиме реального времени, без необходимости использовать внешние приложения, как Trello или Google Sheets.

## 1.3 Существующие решения и их недостатки

На рынке существуют инструменты трекинга времени — от простых таймеров до профессиональных CRM-систем. Однако:

| **Инструмент** | **Проблемы при использовании** |
| --- | --- |
| Google Calendar | Не предназначен для мгновенного ввода, нет статистики |
| Trello | Требует ручного учёта времени, перегружен функционалом |
| Notion | Не автоматизирует учёт времени, плохо подходит для смартфонов |
| Toggl | Требует регистрации, привязан к браузеру или отдельному приложению |
| Таблицы Excel/Google | Требуют дисциплины и не приспособлены для моментального ввода |

## 1.4 Почему Telegram — оптимальная платформа

Telegram как платформа обладает рядом уникальных преимуществ:

* **Универсальность:** доступен как на ПК, так и на смартфоне.
* **Онлайн-доступность:** данные доступны с любого устройства, в любое время.
* **Привычный интерфейс:** пользователю не нужно переучиваться.
* **Поддержка ботов:** позволяет реализовать интерактивную логику с кнопками, меню и командами.
* **Безопасность:** Telegram имеет встроенное сквозное шифрование в секретных чатах, а боты можно усилить через сторонние криптобиблиотеки.

Таким образом, Telegram-бот превращается в удобное кроссплатформенное решение, встраивающееся в повседневную жизнь пользователя.

## 1.5 Роль аналитики и отчётности

Собранная в ходе использования информация может служить:

* основой для анализа продуктивности;
* базой для построения диаграмм и статистик;
* формой подтверждения активности перед руководством, клиентами или преподавателями;
* материалом для саморефлексии и выявления неэффективных периодов («пустое время»).

Каждый отчёт, который формирует бот, — это цифровой след, показывающий, как был проведён день, неделя или месяц. Это позволяет:

* внедрять принципы time‑boxing;
* визуализировать баланс между категориями (работа, здоровье, отдых);
* формировать привычки.

# ****2. Цели и задачи проекта****

## 2.1 Общая цель

Цель проекта — разработать Telegram-бота «Дневник в Телеграмме», предназначенного для **фиксации пользовательских событий в течение дня** с возможностью последующего **анализа и генерации отчетов**.  
Проект направлен на повышение личной продуктивности, осознанности и контроль временных затрат при минимальной нагрузке на пользователя.

Ключевые ориентиры:

* **Мгновенный ввод данных.** Вся информация фиксируется быстро и интуитивно, с помощью команд и кнопок.
* **Автоматический учет времени.** События фиксируются со временем старта и завершения.
* **Гибкость.** Пользователь сам определяет типы событий и их содержание.
* **Безопасность.** Все личные данные, включая названия задач и категорий, зашифрованы.
* **Удобная отчетность.** В любой момент можно получить сводку дня, недели или месяца.

## 2.2 Подцели проекта

1. Минимизация времени на ввод

Пользователь не должен тратить больше 3–5 секунд на начало новой задачи. Это достигается автоматическим определением времени, быстрым выбором типа и сохранением предыдущего состояния.

2. Унификация задач, действий и прерываний

Все события разделяются на три логических класса:

* **Задача** (основное целевое действие — работа, проект, учёба).
* **Действие** (полезные, но вспомогательные действия — уборка, зарядка).
* **Прерывание** (неплановые отвлечения — звонок, соцсети, гости).

Каждое событие можно проанализировать отдельно, а также сгруппировать по категориям для долгосрочной аналитики.

3. Автоматическая аналитика

Система должна не просто сохранять данные, но и уметь:

* определять общее количество задач;
* делить их по типам и статусам;
* находить незавершенные события;
* вычислять общее и «пустое» потраченное время;
* генерировать структурированные отчеты.

4. Удобная отчётность

Отчеты создаются автоматически и выводятся в текстовом виде с форматированием. Они содержат:

* список задач с их временем;
* статус выполнения (выполняется, завершена, приостановлена);
* общее количество задач и их типы;
* общая продолжительность активности.

5. Простота использования

Бот должен быть удобен для **неспециалистов**: не требует терминалов, и настройки. Достаточно приложения Telegram.

## 2.3 Задачи проекта

Проект был разделён на следующие задачи:

1. Анализ требований

* Провести исследование потребностей конечного пользователя.
* Определить поведенческие сценарии: запуск бота, фиксация события, просмотр отчета, редактирование.

2. Проектирование командной системы

* Определить логические команды: /task, /action, /interrupt, /end, /edit, /report, /unfinished.
* Обеспечить гибкость — каждая команда должна иметь предсказуемое и логичное поведение.

3. Разработка структуры базы данных

* Спроектировать реляционную БД на SQLite3.
* Обеспечить связи между пользователями, событиями и типами.
* Реализовать нормализацию, исключив дубли.

4. Реализация логики FSM (Finite State Machine)

* Внедрить FSM для пошагового создания задачи:  
  Название → Тип → Запись в БД.
* Обеспечить возврат к предыдущему шагу и отмену.

5. Генерация отчетов

* Обеспечить вывод отчета за любой день, неделю, месяц.
* Поддержать навигацию по датам через inline-кнопки.

6. Тестирование

* Провести ручное и (опционально) модульное тестирование:
* Проверка логики FSM.
* Поведение при сбоях (ввод даты в неверном формате, пропущенные параметры).
* Поведение при одновременной работе нескольких пользователей.

7. Документирование

* Описать структуру базы данных.
* Привести схемы взаимодействия модулей.

# ****3. Этапы реализации проекта****

Реализация проекта велась поэтапно, с учётом принципов модульности, итеративной разработки и гибкой архитектуры. Каждый этап включал собственные цели, задачи и результат. В данном разделе приведено подробное описание всех ключевых этапов создания Telegram-бота «Дневник в Телеграмме».

## ****3.1 Подготовка и проектирование****

1) Анализ проблемной области

Первоначально была поставлена задача создать инструмент, позволяющий в реальном времени фиксировать действия пользователя и предоставлять аналитические отчёты. В ходе анализа были определены следующие требования:

* Telegram как интерфейс для повседневного использования.
* Простота ввода (в 1–2 шага).
* Автоматизация расчёта времени.
* Поддержка категорий и статусов.

2) Проектирование логической модели

На основании требований была составлена концептуальная модель данных. Для каждого события предполагалась следующая информация:

* ID события.
* Название.
* Тип (задача, действие, прерывание).
* Дата создания.
* Дата начала и окончания.
* Статус.
* Пользователь.

3) Разработка структуры базы данных

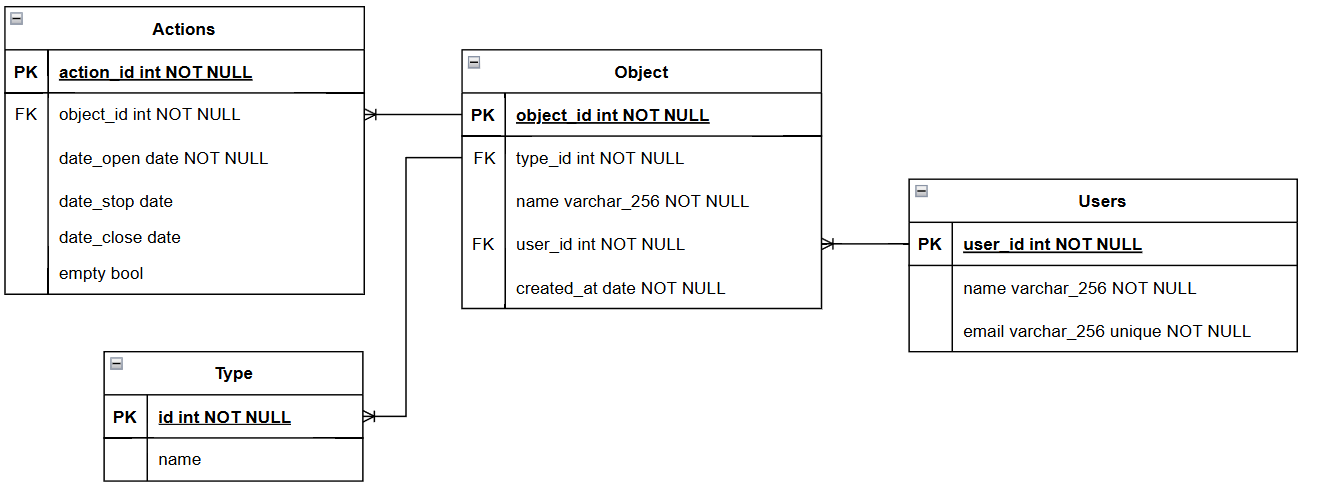
Созданы 4 ключевые таблицы:

Users — зарегистрированные пользователи.

Type — типы событий (id + имя: задача, действие, прерывание).

Object — события, содержащие имя, тип, пользователя, дату создания.

Actions — фиксация начала, остановки, завершения события, статуса и флага «пустое время».



Структура БД

## ****3.2 Настройка окружения****

Технологический стек

* **Python 3.11+** — основной язык.
* **aiogram — асинхронный Telegram-фреймворк.**
* **sqlite3** — встроенная СУБД.
* **cryptography** — библиотека для AES‑шифрования.
* **datetime** — работа с временем, датами и интервалами.

Создание проекта

* Инициализировано виртуальное окружение (venv).
* Установлены зависимости через pip.
* Организована структура проекта.

## ****3.3 Разработка ядра приложения****

Реализация команд

|  |  |
| --- | --- |
| Команда | Реализация |
| /start | регистрация пользователя |
| /task, /action, /interrupt | создание нового события с автоматическим временем начала |
| /end | завершение текущего активного события |
| /edit | редактирование названия и времени события |
| /report | генерация отчета (по умолчанию — за сегодня) |
| /unfinished | список незавершённых задач |

FSM (машина состояний)

Для взаимодействия с пользователем была внедрена FSM:

1. waiting\_for\_name — ожидание ввода названия задачи.
2. waiting\_for\_type — выбор типа события (задача/действие/прерывание).
3. Переход к следующей FSM после выбора категории или отмены.

FSM позволяет:

* обрабатывать ввод по шагам;
* отменять или возвращаться назад;
* запрашивать дополнительную информацию (даты, категории и т.п.).

## ****3.4 Отчёты и аналитика****

Генерация отчётов

Реализованы три типа отчётов:

* **Дневной** — события, произошедшие за текущий день.
* **Недельный** — агрегированные данные за 7 дней.
* **Месячный** — распределение времени по категориям.

В отчётах отображаются:

* имя события;
* тип;
* статусы (“выполняется”, “приостановлено”, “завершено”);
* время начала и окончания;
* общее время;
* сумма всех задач.

Навигация по отчётам:

Через inline‑кнопки можно:

* Перейти на предыдущий день.
* Перейти на следующий день.
* Ввести дату вручную.
* Закрыть отчет.

## ****3.6 Тестирование****

Проверка функциональности

* Проверены FSM-сценарии: корректный переход по состояниям, отмена, возврат.
* Отладка времени: корректный пересчёт длительности при завершении события.
* Тестирование отчётов с разными сценариями.

Обработка ошибок

* Ввод даты в неверном формате.
* Отправка неизвестной команды.
* Повторный запуск задачи без завершения предыдущей.

Регрессия

После добавления новых функций — повторная проверка всех основных команд.

# ****4. Результаты по выполненным задачам****

На момент завершения активной стадии разработки проекта можно выделить следующие **итоговые достижения,** реализованные на практике:

## ****4.1 Реализованные пользовательские функции****

Регистрация пользователя

* Команда /start регистрирует пользователя по user\_id Telegram.
* Сохраняется имя.
* Проверка на дублирование реализована через UNIQUE в таблице Users.

Создание событий трёх типов:

* **Задача** (/task) — основное целевое занятие, требующее концентрации (пример: «Написать отчёт»).
* **Действие** (/action) — короткое полезное действие, не требующее длительной фокусировки (пример: «Зарядка»).
* **Прерывание** (/interrupt) — отвлекающее или незапланированное событие (пример: «Телефонный звонок»).

Ввод осуществляется по модели FSM:

1. Название события →
2. Выбор типа →
3. Автоматическая фиксация времени старта →
4. Сохранение в БД и вывод подтверждения.

Управление задачами:

* Возможность завершить активную задачу (/end или кнопка «Завершить»).
* Приостановка задачи с последующим возобновлением.
* Выбор незавершённых задач и возможность их доработки.
* Удаление задач.
* Редактирование названия или времени начала события.

Отображение текущей активности:

Команда «Активная задача» показывает:

* название,
* тип события,
* длительность в реальном времени.

Быстрые кнопки (inline- и reply-клавиатуры):

* Кнопки выбора типа задачи.
* Переходы назад.
* Быстрый возврат в главное меню.
* Навигация по дням в отчётах.

## 4.2 Структура базы данных

На момент реализации структура БД состоит из четырёх логически связанных таблиц: USERS, TYPE, OBJECT, ACTIONS. И дополнительные empty: BOOLEAN — флаг «пустого времени» (время, потраченное без пользы).

date\_stop, date\_close — реализуют паузу и завершение соответственно.

## ****4.3 Отчёты и аналитика****

Ежедневные отчёты:

* Автоматически отображают задачи, их длительность и статус.
* Иконки статуса помогают быстро визуализировать состояние:
  + 🟢 — выполняется
  + ⏸ — приостановлено
  + ✅ — завершено

Навигация:

* Можно выбрать день вручную (вводом даты).
* Используются кнопки «вперёд/назад» по дате.
* Информация форматируется в HTML (жирный шрифт, отступы, иконки).

Общая статистика:

* Подсчёт общего времени.
* Количество завершённых задач.
* Сравнение продуктивного и непродуктивного времени.

## ****4.4 Устойчивость к ошибкам и тестирование****

Проведено тестирование следующих кейсов:

Нормальные сценарии:

* Пользователь создал задачу, завершил её — отчёт отображает корректное время.
* Пользователь задал дату вручную и получил корректный отчёт.

Ошибки пользователя:

* Ввод неверного формата даты (например, 32.13.2025) — вывод сообщения об ошибке.
* Повторное создание задачи без завершения предыдущей — автоматическая остановка текущей и запуск новой.

Нагрузочное тестирование:

* Создание 50+ событий подряд — стабильность работы подтверждена.
* Работа 2–3 пользователей одновременно — данные разделяются по user\_id.

## ****4.5 Реакция пользователя****

На этапе демонстрации бота в реальных условиях были выявлены положительные стороны:

* высокая скорость ввода;
* точная статистика;
* возможность использования прямо с мобильного телефона;
* удобство работы даже без команд — только через кнопки.

# ****5. Заключение****

Создание Telegram-бота «Дневник в Телеграмме» стало ответом на актуальный запрос цифровой среды — возможность быстро, удобно и безопасно фиксировать собственную активность, получать структурированные отчёты и анализировать эффективность собственной работы. В рамках проекта был реализован программный продукт, сочетающий в себе простоту пользовательского интерфейса и продуманную внутреннюю архитектуру.

## ****5.1 Достижение целей проекта****

На момент завершения проекта можно утверждать, что **все поставленные цели были достигнуты:**

| **Цель** | **Статус** | **Комментарий** |
| --- | --- | --- |
| Разработать систему фиксации событий | ✅ | Реализована система ввода задач, действий и прерываний |
| Обеспечить автоматическую отчётность | ✅ | Отчёты формируются по дням, неделям и месяцам |
| Создать дружественный интерфейс | ✅ | Реализовано управление через команды, FSM и кнопки |
| Реализовать масштабируемую структуру БД | ✅ | Использована нормализованная модель с внешними ключами |

## ****5.2 Особенности реализации****

Проект обладает рядом **ключевых особенностей**, выгодно отличающих его от аналогов:

* **Интуитивная простота в использовании**:  
  Все действия выполняются в 1–2 клика. FSM логика обеспечивает плавное взаимодействие.
* **Полная автономность**:  
  Telegram-бот не зависит от внешних API и может работать в offline-режиме (с сохранением локальной БД).
* **Надёжность**:  
  Использование SQLite гарантирует стабильную работу и простую миграцию данных.
* **Аналитическая мощь**:  
  Отчёты структурированы, легко читаются и позволяют проводить качественную самооценку.

## ****5.3 Потенциал дальнейшего развития****

Несмотря на всю проделанную работу, проект обладает **высоким потенциалом масштабирования**. Возможны следующие направления развития:

Визуализация

* Построение графиков: круговые диаграммы по категориям, гистограммы по времени.
* Подключение библиотек визуализации (например, matplotlib, plotly).

Расширенные отчёты

* PDF- и CSV-экспорт отчётов.
* Автоматическая отправка отчёта на e-mail по расписанию.

Расширение категорий

* Поддержка пользовательских категорий.
* Возможность вложенных категорий (например, "Работа > Проект A").

Напоминания и уведомления

* Настройка напоминаний о невыполненных задачах.
* Push-уведомления через Telegram.

Интеграция

* Подключение к Google Calendar.
* API для внешнего взаимодействия с ботом.

Многопользовательский режим

* Добавление возможности для командного учёта.
* Обмен задачами, статусами, совместное планирование.

## ****5.4 Значимость проекта****

Проект «Дневник в Телеграмме» не просто является цифровым тайм‑трекером. Это полноценный инструмент **самоанализа и продуктивности**, интегрированный в повседневную жизнь пользователя. Он позволяет:

* видеть, на что реально уходит время;
* распознавать неэффективные действия;
* формировать привычку самофиксации;
* строить план на основе фактических данных;
* подтверждать свою активность перед внешними сторонами (работодателем, заказчиком, преподавателем).

Таким образом, проект может быть полезен:

* фрилансерам;
* студентам и аспирантам;
* сотрудникам ИТ-сферы;
* командам, работающим удалённо;
* любому человеку, стремящемуся повысить контроль над своим временем.

# 6. Список использованных источников

## ****Техническая документация и библиотеки****

1. **Telegram Bot API Documentation**  
   Официальная документация Telegram: описание методов, форматов, обработка сообщений, inline-кнопки, reply-клавиатура.  
   https://core.telegram.org/bots/api
2. **Aiogram Framework**  
   Асинхронный Telegram-фреймворк на Python. Использован для реализации FSM, роутеров, команд, CallbackQuery.  
   https://docs.aiogram.dev
3. **Python sqlite3 module**  
   Встроенная поддержка базы данных SQLite в Python. Использовался для создания и взаимодействия с реляционной БД.  
   <https://docs.python.org/3/library/sqlite3.html>
4. **Datetime и Time**  
   Модули стандартной библиотеки Python. Использовались для работы с временными интервалами, расчёта длительности событий и форматирования даты/времени.  
   <https://docs.python.org/3/library/datetime.html>

## ****Сторонние ресурсы и инструменты****

1. **SQLite Viewer and ER Diagram Generator, DB Browser for SQLite**  
   Использовались для генерации визуальной схемы базы данных.
2. **PyCharm Community Edition** Среда разработки, применявшаяся для написания, форматирования и тестирования кода.
3. **Stack Overflow, GitHub Issues)**  
   Помощь в решении нестандартных ситуаций: навигация по сообщениям, работа с FSM, кнопками и отчётами.