Федеральное государственное автономное образовательное учреждение высшего образования

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий  
Кафедра «Инфокогнитивных технологий»

Направление подготовки/ специальность: информатика и вычислительная техника

ОТЧЕТ

по проектной практике

Студент: Колотыгин Даниил Алексеевич Группа: 241-3210

Место прохождения практики: Московский Политех, кафедра Инфокогнитивных технологий

Отчет принят с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики: Чернова В.М.

Москва 2025

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

1. Общая информация о проекте:

* Название проекта
* Цели и задачи проекта

1. Общая характеристика деятельности организации *(заказчика проекта)*

* Наименование заказчика
* Организационная структура
* Описание деятельности

1. Описание задания по проектной практике
2. Описание достигнутых результатов по проектной практике

ЗАКЛЮЧЕНИЕ *(выводы о проделанной работе и оценка ценности выполненных задач для заказчика)*

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

ПРИЛОЖЕНИЯ *(при необходимости)*

ВВЕДЕНИЕ

Проектная практика направлена на закрепление теоретических знаний, полученных в ходе обучения, и их применение в реальных проектах. В рамках практики я участвовал в разработке социально значимого проекта — платформы для помощи людям с ограниченными возможностями здоровья (ОВЗ), а также реализовал дополнительный проект — Telegram-бота для напоминания о приеме лекарств. Оба проекта имеют практическую ценность и соответствуют современным требованиям к IT-решениям.

**1. Общая информация о проекте**

**1.1 Название проекта**

* Основной проект: **«Поддержка рядом»** — веб-платформа для взаимодействия волонтеров и людей с ОВЗ.
* Вариативная часть: **Telegram-бот «PillReminder»** — сервис для управления расписанием приема лекарств.

**1.2 Цели и задачи проекта**

**Основной проект:**

* **Цель:** Создание удобного инструмента для коммуникации между людьми с ОВЗ и волонтерами.
* **Задачи:**
  + Разработка адаптивного веб-интерфейса.
  + Реализация функционала личного кабинета.
  + Обеспечение доступности для пользователей с разными формами инвалидности.

**Вариативная часть (бот):**

* **Цель:** Автоматизация напоминаний о приеме лекарств для людей с хроническими заболеваниями.
* **Задачи:**
  + Создание базы данных для хранения расписаний.
  + Реализация команд для управления напоминаниями.
  + Развертывание бота на облачной платформе Railway.

**2. Общая характеристика деятельности организации (заказчика проекта)**

**2.1 Наименование заказчика**

Заказчиком проекта выступила **кафедра Инфокогнитивных технологий Московского Политехнического Университета**.

**2.2 Организационная структура**

* **Руководитель практики:** Чернова В.М.
* **Куратор проекта:** Соловей Л.Б.
* **Партнер:** Организация **«Мосволонтер»** (оказание методической поддержки).

**2.3 Описание деятельности**

Кафедра занимается разработкой IT-решений в области социальной интеграции и поддержки людей с ОВЗ. В рамках сотрудничества с «Мосволонтером» были определены требования к функционалу платформы и бота.

**3. Описание задания по проектной практике**

Задание состояло из двух частей:

1. **Базовая часть:**
   * Создание статического сайта на HTML/CSS.
   * Настройка Git-репозитория и документирование проекта с использованием Markdown.
2. **Вариативная часть:**
   * Разработка Telegram-бота на Python с использованием библиотеки python-telegram-bot.
   * Реализация базы данных (SQLite) для хранения данных пользователей.
   * Развертывание бота на облачном сервисе Railway.

**4. Описание достигнутых результатов по проектной практике**

**4.1 Основной проект: платформа «Поддержка рядом»**

* Разработаны ключевые страницы сайта:
  + Главная (адаптивная верстка).
  + Личный кабинет волонтера.
  + Журнал активности.
* Внедрены элементы доступности:
  + Контрастная цветовая схема.
  + Поддержка screen readers.

**4.2 Вариативная часть: бот «MedicineReminder»**

* **Реализованные функции:**
  + Добавление/удаление лекарств (/add, /del).
  + Просмотр расписания (/list).
  + Поддержка часовых поясов (/timezone).
* **Технические аспекты:**
  + Использована СУБД SQLite для хранения данных.
  + Настроен автоматический деплой на Railway.
* **Ссылки:**
  + Репозиторий: https://github.com/Daniil888-m/ProjectPractice.
  + Бот в Telegram: @PillReminder\_Polytech\_bot.

**ЗАКЛЮЧЕНИЕ**

В ходе проектной практики были достигнуты следующие результаты:

1. Разработан прототип платформы для помощи людям с ОВЗ, соответствующий требованиям заказчика.
2. Создан и развернут Telegram-бот, решающий проблему дисциплинированного приема лекарств.
3. Приобретены навыки работы с Git, HTML/CSS, Markdown, Python и облачными сервисами.

Оба проекта имеют социальную значимость и могут быть доработаны для внедрения в реальную эксплуатацию.

**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Документация Python-telegram-bot.
2. Руководство по развертыванию на Railway.
3. ГОСТ Р 7.0.97-2016 (требования к оформлению отчетов).

**ПРИЛОЖЕНИЯ**

1. Скриншоты интерфейса бота (Рисунок 1, 2).
2. Фрагмент кода (Листинг 1).

**Приложение 1. Скриншоты интерфейса бота**

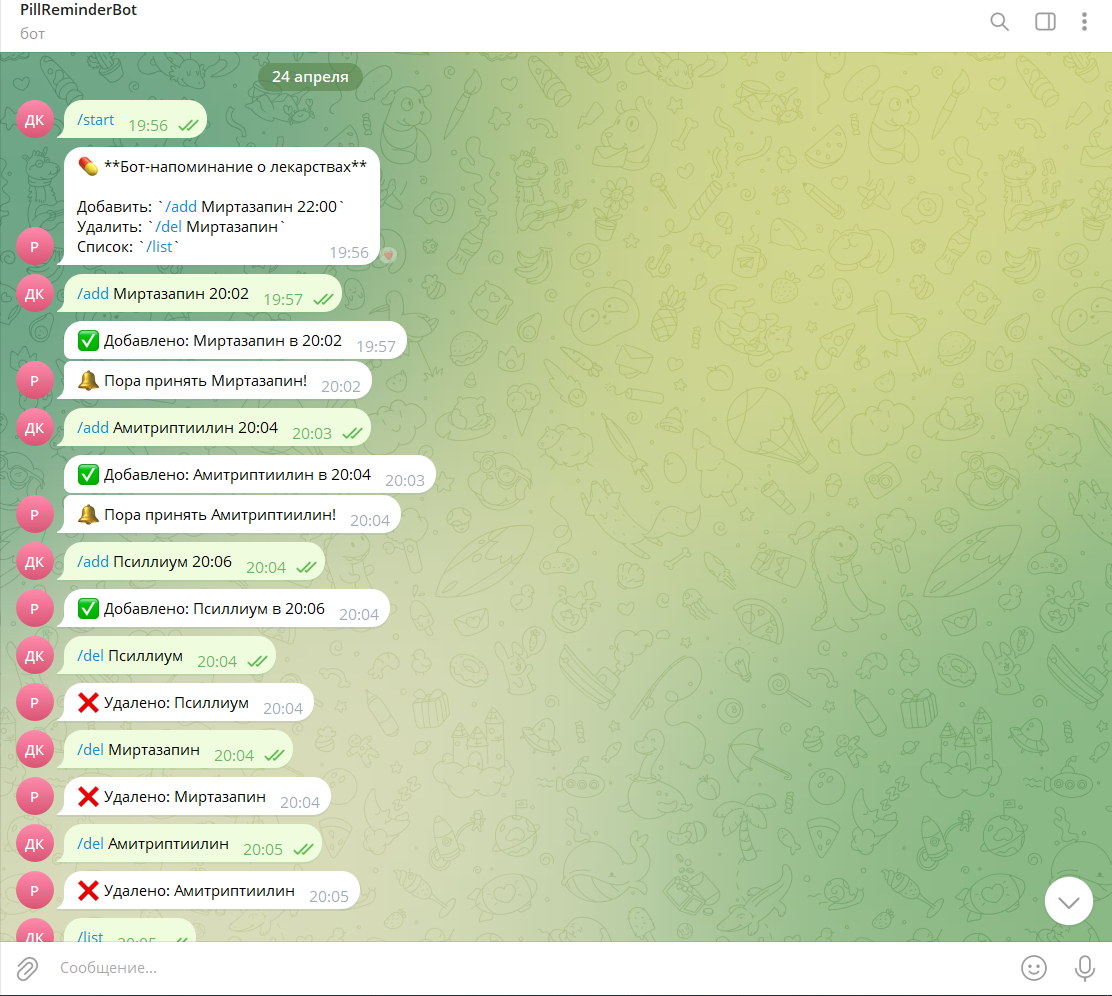


Рисунок 1 – Скриншот интерфейса бота

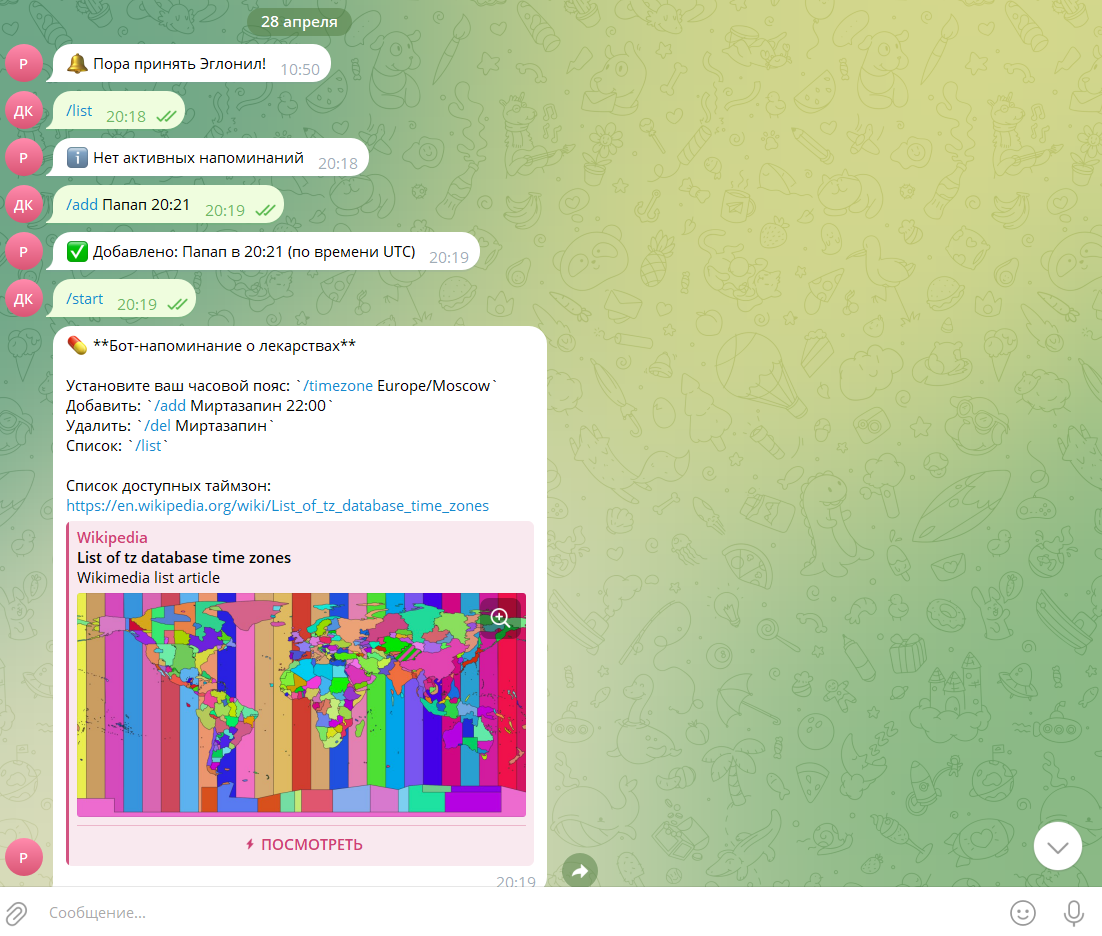


Рисунок 2 – Скриншот интерфейса бота

Приложение 2. Листинг программы телеграмм-бота

from telegram.ext import Application, CommandHandler, ContextTypes

from telegram import Update

import sqlite3

import datetime

import pytz

import asyncio

import os

TOKEN = "7392929368:AAEiMeWKDSQ8dYUBqr8ekYy4J1ilagtYuQo"

# --- Функции для работы с БД ---

def **init\_db**():

    conn = sqlite3.**connect**('pills.db')

    cursor = conn.**cursor**()

    cursor.**execute**('''

        CREATE TABLE IF NOT EXISTS reminders (

            chat\_id INTEGER,

            drug\_name TEXT,

            time TEXT,

            timezone TEXT DEFAULT 'UTC',

            PRIMARY KEY (chat\_id, drug\_name)

        )

    ''')

    conn.**commit**()

    conn.**close**()

def **add\_to\_db**(chat\_id, drug\_name, time\_str, timezone='UTC'):

    conn = sqlite3.**connect**('pills.db')

    cursor = conn.**cursor**()

    cursor.**execute**("INSERT OR REPLACE INTO reminders VALUES (?, ?, ?, ?)",

                 (chat\_id, drug\_name, time\_str, timezone))

    conn.**commit**()

    conn.**close**()

def **del\_from\_db**(chat\_id, drug\_name):

    conn = sqlite3.**connect**('pills.db')

    cursor = conn.**cursor**()

    cursor.**execute**("DELETE FROM reminders WHERE chat\_id=? AND drug\_name=?",

                 (chat\_id, drug\_name))

    conn.**commit**()

    conn.**close**()

def **get\_reminders**(chat\_id):

    conn = sqlite3.**connect**('pills.db')

    cursor = conn.**cursor**()

    cursor.**execute**("SELECT drug\_name, time, timezone FROM reminders WHERE chat\_id=?",

                 (chat\_id,))

    result = cursor.**fetchall**()

    conn.**close**()

    return result

def **get\_all\_reminders**():

    conn = sqlite3.**connect**('pills.db')

    cursor = conn.**cursor**()

    cursor.**execute**("SELECT chat\_id, drug\_name, time, timezone FROM reminders")

    result = cursor.**fetchall**()

    conn.**close**()

    return result

def **update\_timezone**(chat\_id, timezone):

    conn = sqlite3.**connect**('pills.db')

    cursor = conn.**cursor**()

    cursor.**execute**("UPDATE reminders SET timezone=? WHERE chat\_id=?",

                 (timezone, chat\_id))

    conn.**commit**()

    conn.**close**()

# --- Обработчики команд ---

async def **start**(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    await update.message.**reply\_text**(

        "💊 \*\*Бот-напоминание о лекарствах\*\*\n\n"

        "Установите ваш часовой пояс: `/timezone Europe/Moscow`\n"

        "Добавить: `/add Миртазапин 22:00`\n"

        "Удалить: `/del Миртазапин`\n"

        "Список: `/list`\n\n"

        "Список доступных таймзон: https://en.wikipedia.org/wiki/List\_of\_tz\_database\_time\_zones"

    )

async def **set\_timezone**(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    chat\_id = update.message.chat\_id

    try:

        timezone = context.args[0]

        # Проверяем валидность таймзоны

        if timezone not in pytz.all\_timezones:

            await update.message.**reply\_text**("⛔ Неверная таймзона. Пример: `/timezone Europe/Moscow`")

            return

**update\_timezone**(chat\_id, timezone)

        await update.message.**reply\_text**(f"✅ Часовой пояс установлен: {timezone}")

    except IndexError:

        await update.message.**reply\_text**("⛔ Используйте: `/timezone Europe/Moscow`")

async def **add\_reminder**(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    chat\_id = update.message.chat\_id

    try:

        drug\_name = context.args[0]

        drug\_time = context.args[1]

        try:

            datetime.datetime.**strptime**(drug\_time, "%H:%M")

        except ValueError:

            await update.message.**reply\_text**("⛔ Формат времени: `22:00`")

            return

        # Получаем текущую таймзону пользователя или используем UTC по умолчанию

        timezone = 'UTC'

        reminders = **get\_reminders**(chat\_id)

        if reminders and reminders[0][2]:  # Если уже есть записи с таймзоной

            timezone = reminders[0][2]

**add\_to\_db**(chat\_id, drug\_name, drug\_time, timezone)

        await update.message.**reply\_text**(f"✅ Добавлено: {drug\_name} в {drug\_time} (по времени {timezone})")

    except IndexError:

        await update.message.**reply\_text**("⛔ Используйте: `/add Лекарство 22:00`")

async def **del\_reminder**(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    chat\_id = update.message.chat\_id

    try:

        drug\_name = context.args[0]

**del\_from\_db**(chat\_id, drug\_name)

        await update.message.**reply\_text**(f"❌ Удалено: {drug\_name}")

    except IndexError:

        await update.message.**reply\_text**("⛔ Используйте: `/del Лекарство`")

async def **list\_reminders**(update: Update, context: ContextTypes.DEFAULT\_TYPE):

    chat\_id = update.message.chat\_id

    reminders = **get\_reminders**(chat\_id)

    if reminders:

        message = "📋 Ваши напоминания:\n"

        for drug, time\_str, timezone in reminders:

            message += f"- {drug} в {time\_str} (по времени {timezone})\n"

    else:

        message = "ℹ️ Нет активных напоминаний"

    await update.message.**reply\_text**(message)

async def **check\_reminders**(context: ContextTypes.DEFAULT\_TYPE):

    reminders = **get\_all\_reminders**()

    for chat\_id, drug\_name, time\_str, timezone in reminders:

        try:

            # Получаем текущее время в указанной таймзоне

            tz = pytz.**timezone**(timezone)

            now = datetime.datetime.**now**(tz).**strftime**("%H:%M")

            if now == time\_str:

                await context.bot.**send\_message**(chat\_id, text=f"🔔 Пора принять {drug\_name}!")

        except Exception as e:

**print**(f"Ошибка при проверке напоминания: {e}")

def **main**():

    try:

        # Удаляем старую БД

        if os.path.**exists**('pills.db'):

            os.**remove**('pills.db')

**print**("Старая БД удалена")

**init\_db**()

**print**("База данных успешно инициализирована")

        app = Application.**builder**().**token**(TOKEN).**build**()

        # Добавляем обработчики команд

        app.**add\_handler**(CommandHandler("start", **start**))

        app.**add\_handler**(CommandHandler("timezone", **set\_timezone**))

        app.**add\_handler**(CommandHandler("add", **add\_reminder**))

        app.**add\_handler**(CommandHandler("del", **del\_reminder**))

        app.**add\_handler**(CommandHandler("list", **list\_reminders**))

        # Настраиваем проверку напоминаний

        job\_queue = app.job\_queue

        job\_queue.**run\_repeating**(**check\_reminders**, interval=60.0)

**print**("Бот успешно запущен")

        app.**run\_polling**()

    except Exception as e:

**print**(f"Ошибка при запуске: {str(e)}")

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

    # Проверяем установку зависимостей

    try:

        from telegram.ext import Application, CommandHandler, ContextTypes

        from telegram import Update

        import pytz

**main**()

    except ImportError:

**print**("Ошибка: Не установлены необходимые библиотеки.")

**print**("Установите их командой: pip install python-telegram-bot==20.3 pytz")