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
* 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

ВВЕДЕНИЕ

В ходе проектной практики мной был разработан Telegram-бот "PillReminder", предназначенный для помощи людям с хроническими заболеваниями в соблюдении режима приема лекарств. Данный проект представляет собой практическую реализацию полученных в университете знаний в области программирования на Python, работы с базами данных и облачными технологиями.

Бот обладает следующим функционалом:

* Управление расписанием приема лекарств через простые команды
* Автоматические напоминания в установленное время
* Поддержка различных часовых поясов
* Хранение истории назначений

Разработка велась с использованием современных технологий:

* Язык программирования Python 3.10
* Библиотека python-telegram-bot для работы с Telegram API
* SQLite для хранения данных пользователей
* Облачная платформа Railway для развертывания

Проект имеет социальную значимость, так как решает актуальную проблему соблюдения медицинских рекомендаций, особенно важную для пожилых людей и пациентов с хроническими заболеваниями.

**1. Общая информация о проекте:**

**1.1 Название проекта**  
Telegram-бот **«PillReminder»** — сервис для автоматизации напоминаний о приеме лекарств.

**1.2 Цели и задачи**

* **Цель:** Создание удобного инструмента для людей с хроническими заболеваниями, обеспечивающего своевременный прием лекарств.
* **Задачи:**
  + Реализация базы данных (SQLite) для хранения расписаний.
  + Разработка команд управления (/add, /del, /list).
  + Настройка автоматических напоминаний с учетом часовых поясов.
  + Деплой бота на облачной платформе Railway.

#### ****2. Характеристика деятельности заказчика****

#### ****2.1 Наименование заказчика****

**Московский политехнический университет** (Московский Политех)

* **Факультет:** Информационных технологий
* **Кафедра:** Инфокогнитивных технологий

#### ****2.2 Описание деятельности****

1. **Образовательная деятельность:**
   * Подготовка специалистов в области IT, когнитивных технологий и инженерии.
   * Внедрение проектного обучения для решения реальных задач.
2. **Научно-исследовательская работа:**
   * Разработка социально значимых IT-решений (например, для людей с ОВЗ).
   * Исследования в области искусственного интеллекта, человеко-машинного взаимодействия.
3. **Социальная миссия:**
   * Сотрудничество с организациями (например, «Мосволонтер») для создания технологий, улучшающих качество жизни.
   * Поддержка студенческих инициатив, направленных на помощь уязвимым группам населения.
4. **Инфраструктура:**
   * Лаборатории для разработки и тестирования ПО.

#### ****3. Описание задания по проектной практике****

**Цель разработки:**  
Создание Telegram-бота «PillReminder» для автоматизации напоминаний о приеме лекарств с возможностью гибкой настройки расписания.

**Технические требования:**

1. **Язык программирования:** Python 3.10
2. **Библиотеки:**
   * python-telegram-bot (версия 20.3) — для работы с Telegram API
   * sqlite3 — встроенная СУБД для хранения данных пользователей
   * pytz — обработка часовых поясов
   * datetime — работа с временем и датами
3. **Хранение данных:** SQLite (локальная база данных)
4. **Деплой:** Облачная платформа Railway
5. **Дополнительные требования:**
   * Поддержка командного интерфейса
   * Логирование ошибок
   * Защита персональных данных

**Функциональные требования к боту:**

1. **Основные команды:**
   * /start — приветственное сообщение с инструкцией
   * /add <лекарство> <время> — добавление препарата в расписание
   * /del <лекарство> — удаление препарата из расписания
   * /list — просмотр текущего списка лекарств
   * /timezone <часовой пояс> — установка часового пояса
2. **Дополнительные функции:**
   * Автоматическая проверка времени и отправка уведомлений
   * Поддержка разных часовых поясов
   * Валидация вводимых данных
   * Обработка ошибок ввода
3. **Особенности реализации:**
   * Использование контекстных обработчиков команд
   * Регулярная проверка времени (каждую минуту)
   * Журналирование всех операций

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

4.1. Анализ требований

**Цель:** разработать Telegram-бота для управления расписанием приема лекарств с функциями:

- Добавление/удаление препаратов

- Настройка часового пояса

- Автоматические напоминания

- Хранение истории назначений

**Схема базы данных в SQLite:**

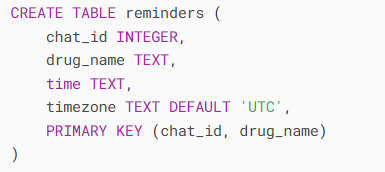


Рис 1 – Схема базы данных в SQLite

**Диаграмма компонентов:**

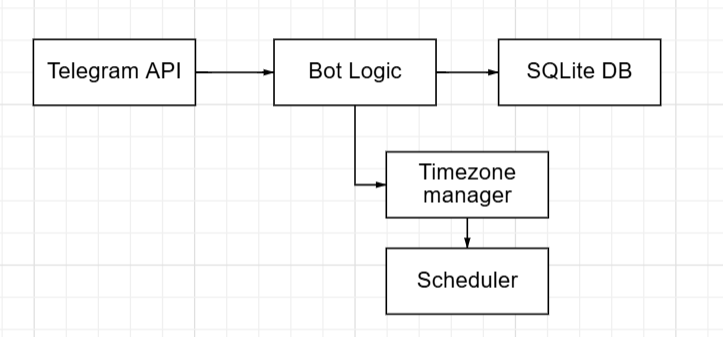


Рис. 2 – Диаграмма компонентов

#### 4.2. Реализация (ключевые этапы)

4.2.1. Базовая структура бота

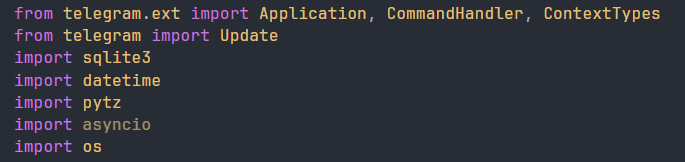


Рис. 3 – Подключение зависимостей проекта

4.2.2. Работа с базой данных

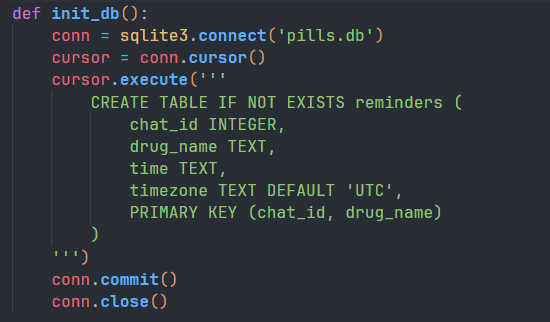


Рис. 4 – Работа с базой данных

4.3 Основные команды бота

*Команда /start:*

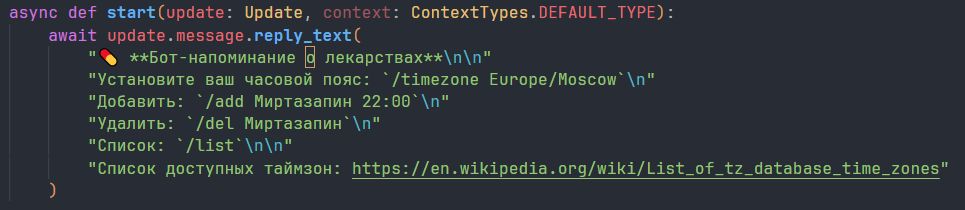


Рис. 5 – Код для работы команды start

*Добавление лекарства (/add):*

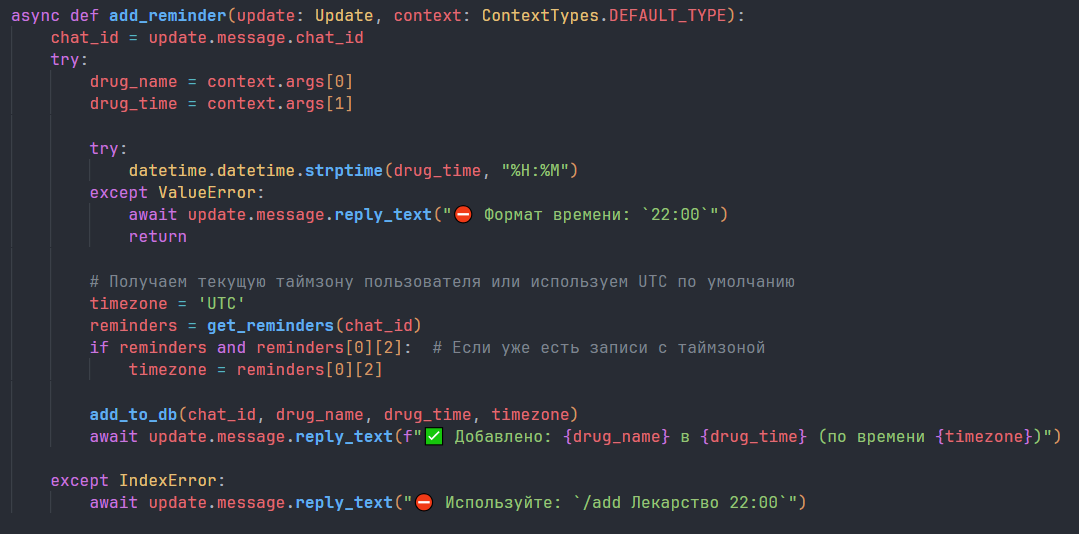
**

Рис. 6 – Код для работы команды add

Система напоминаний

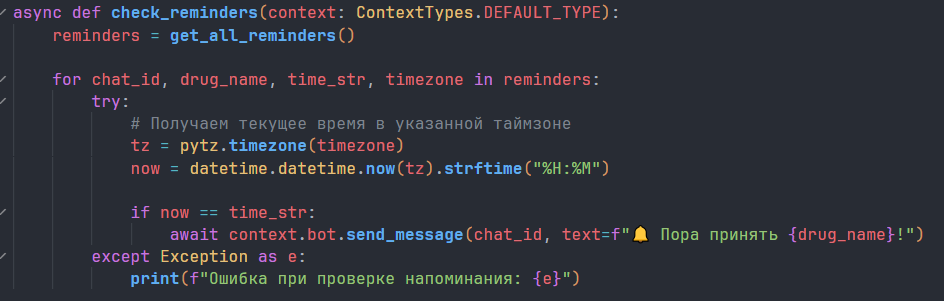
****

Рис. 7 – Код для автоматической проверки времени и отправки напоминаний

* 1. **Развертка бота на стороннем ресурсе**

В качестве платформы для деплоя проекта был выбран сервис Railway.

**Этапы развертывания:**

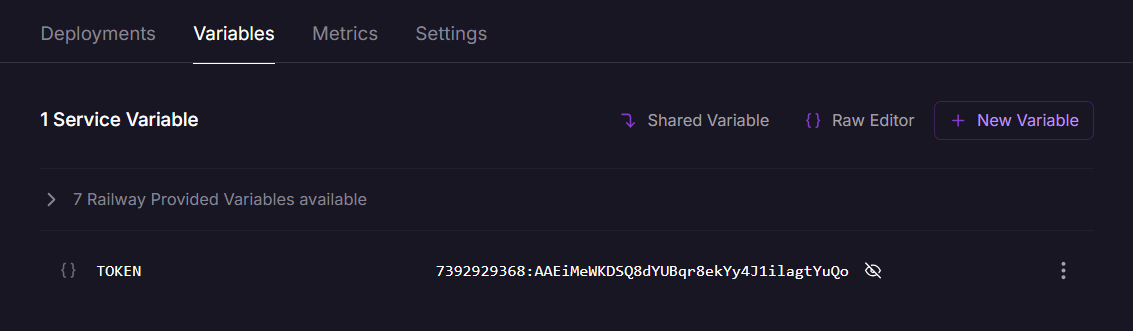
* Создание аккаунта на Railway
* Настройка переменных окружения: 

Рис. 8 – Настройка переменных окружения на платформе Railway

* Автоматический деплой из GitHub-репозитория:

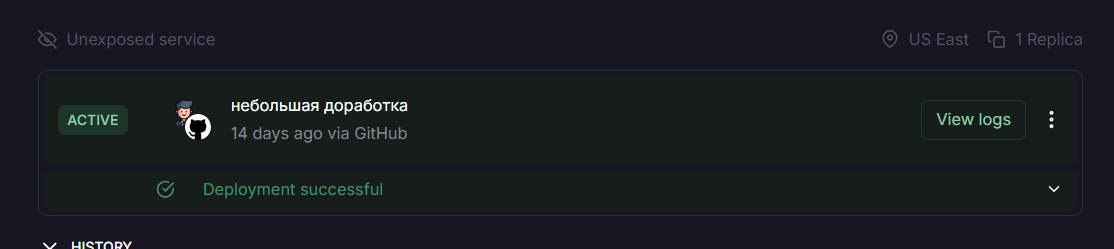


Рис. 9 – Результат успешной развертки проекте на платформе

**Скриншоты интерфейса бота**

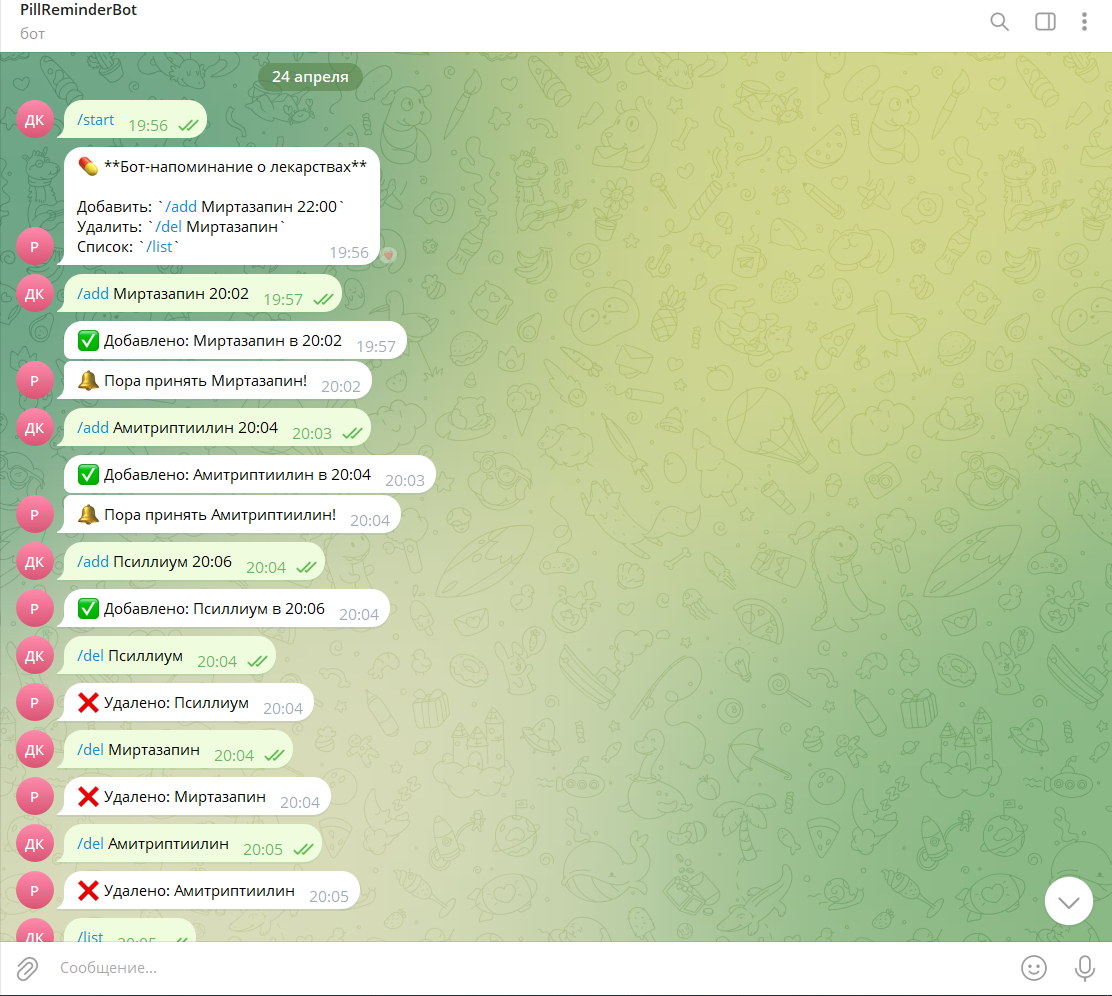


Рисунок 10 – Скриншот интерфейса бота

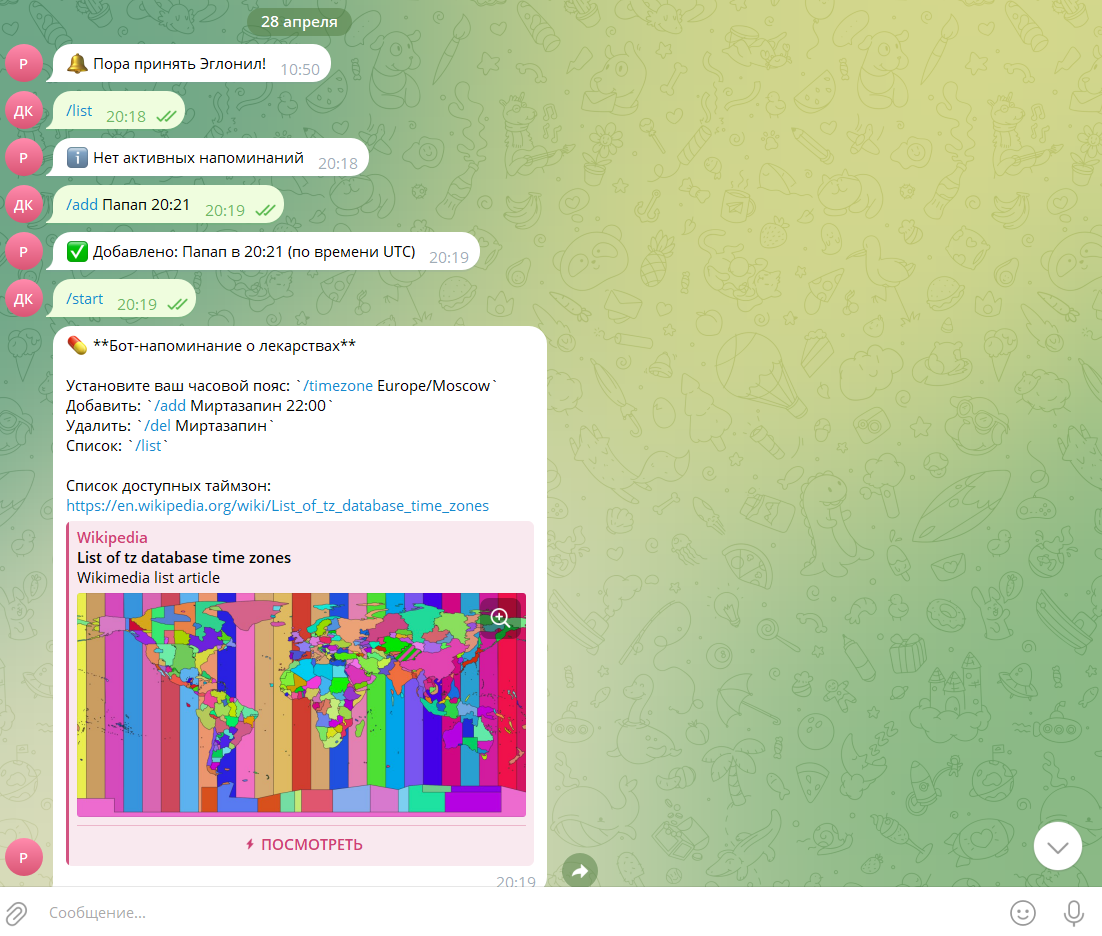


Рисунок 11 – Скриншот интерфейса бота

**Ссылки:**

* [Репозиторий проекта](https://github.com/Daniil888-m/ProjectPractice)
* [Бот в Telegram](https://t.me/PillReminder_Polytech_bot)

**ЗАКЛЮЧЕНИЕ**

В ходе проектной практики были достигнуты следующие результаты:

1. Создан и развернут Telegram-бот, решающий проблему дисциплинированного приема лекарств.

Проект имеют социальную значимость и может быть доработан 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**

Приложение 1. 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")