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
2. Общая характеристика деятельности организации *(заказчика проекта)*
    - Наименование заказчика
    - Описание деятельности
3. Описание задания по проектной практике
4. 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:

```
CREATE TABLE reminders (
    chat_id INTEGER,
    drug_name TEXT,
    time TEXT,
    timezone TEXT DEFAULT 'UTC',
    PRIMARY KEY (chat_id, drug_name)
)
```

Рис 1 – Схема базы данных в SQLite

Диаграмма компонентов:

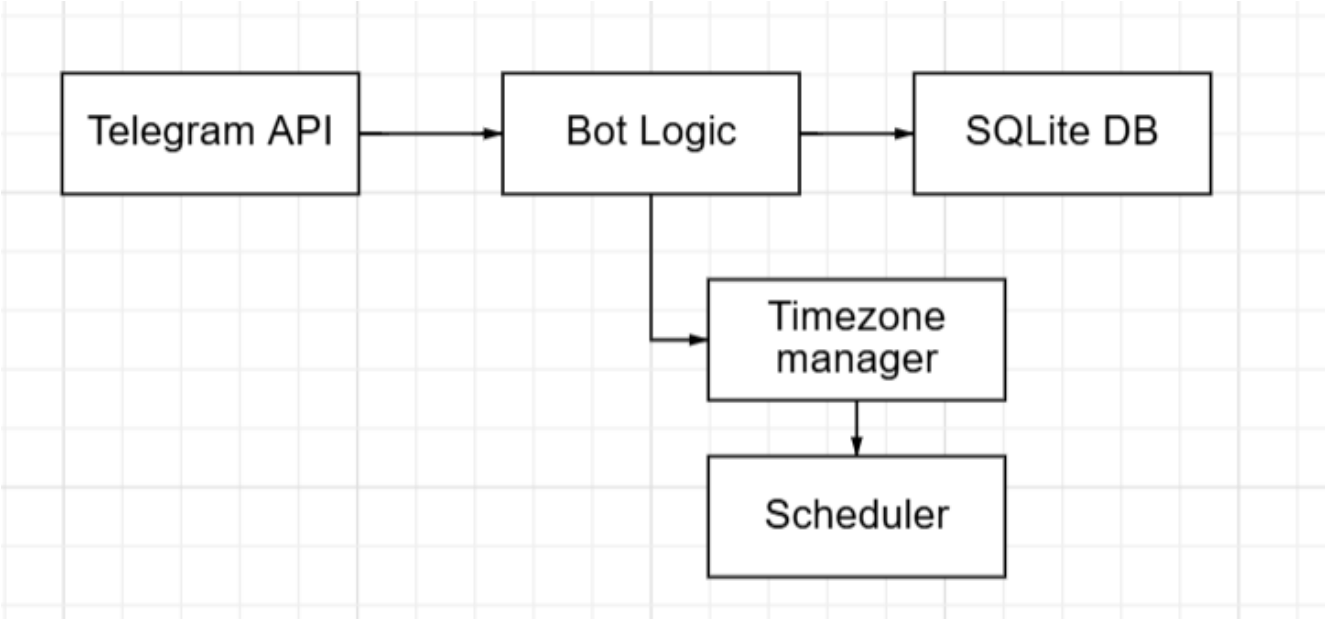

Рис. 2 – Диаграмма компонентов

4.2. Реализация (ключевые этапы)

4.2.1. Базовая структура бота

```
from telegram.ext import Application, CommandHandler, ContextTypes
from telegram import Update
import sqlite3
import datetime
import pytz
import asyncio
import os
```

Рис. 3 – Подключение зависимостей проекта

4.2.2. Работа с базой данных

```python
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()
```

Рис. 4 – Работа с базой данных

4.3 Основные команды бота

Команда /start:

```python
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"
    )
```

Рис. 5 – Код для работы команды start

Добавление лекарства (/add):

```python
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`")
```

Рис. 6 – Код для работы команды add

Система напоминаний

```python
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}")
```

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

4.4 Развертка бота на стороннем ресурсе

В качестве платформы для деплоя проекта был выбран сервис Railway.

Этапы развертывания:

- Создание аккаунта на Railway

- Настройка переменных окружения:

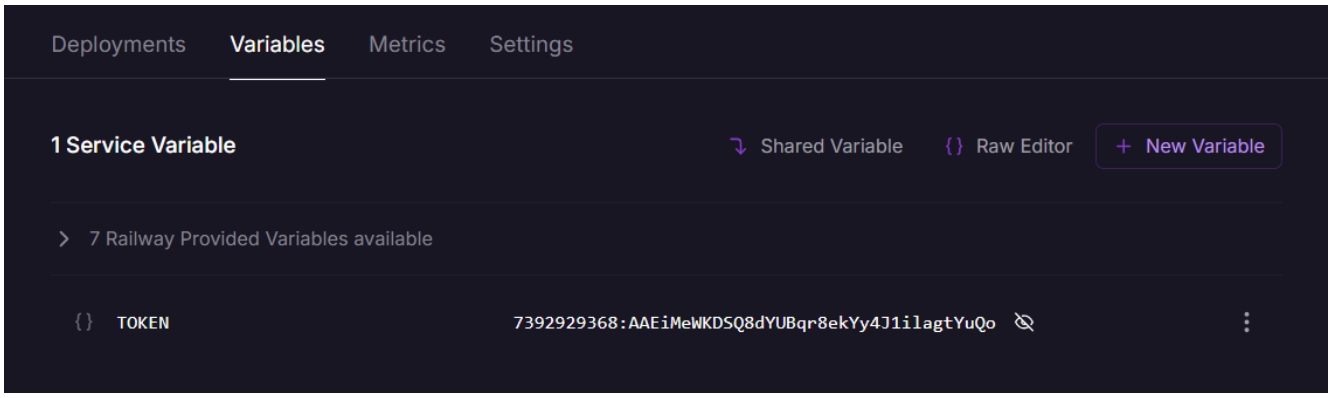

Рис. 8 – Настройка переменных окружения на платформе Railway

- Автоматический деплой из GitHub-репозитория:

Рис. 9 – Результат успешной развертки проекте на платформе

Скриншоты интерфейса бота

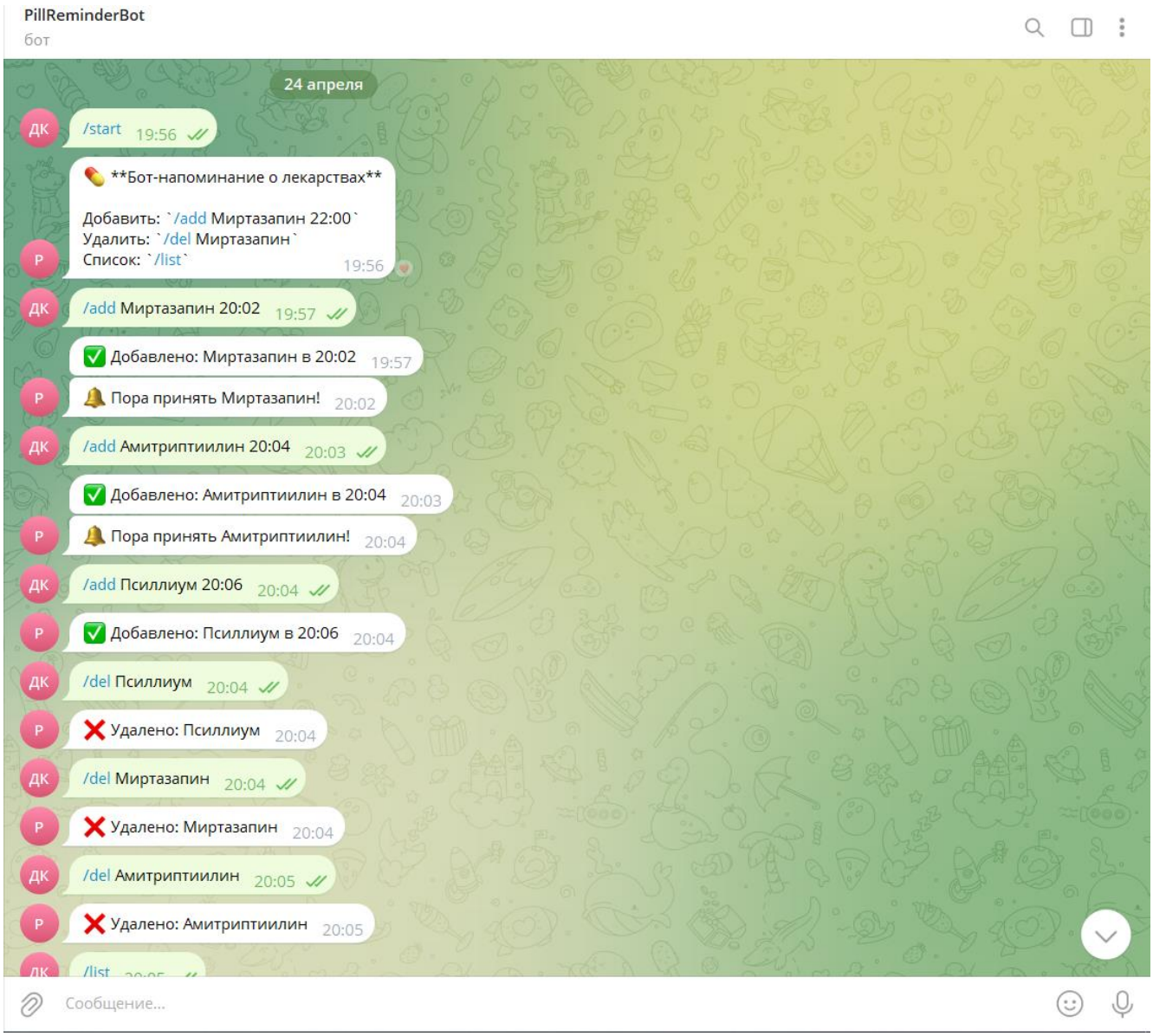

Рисунок 10 – Скриншот интерфейса бота

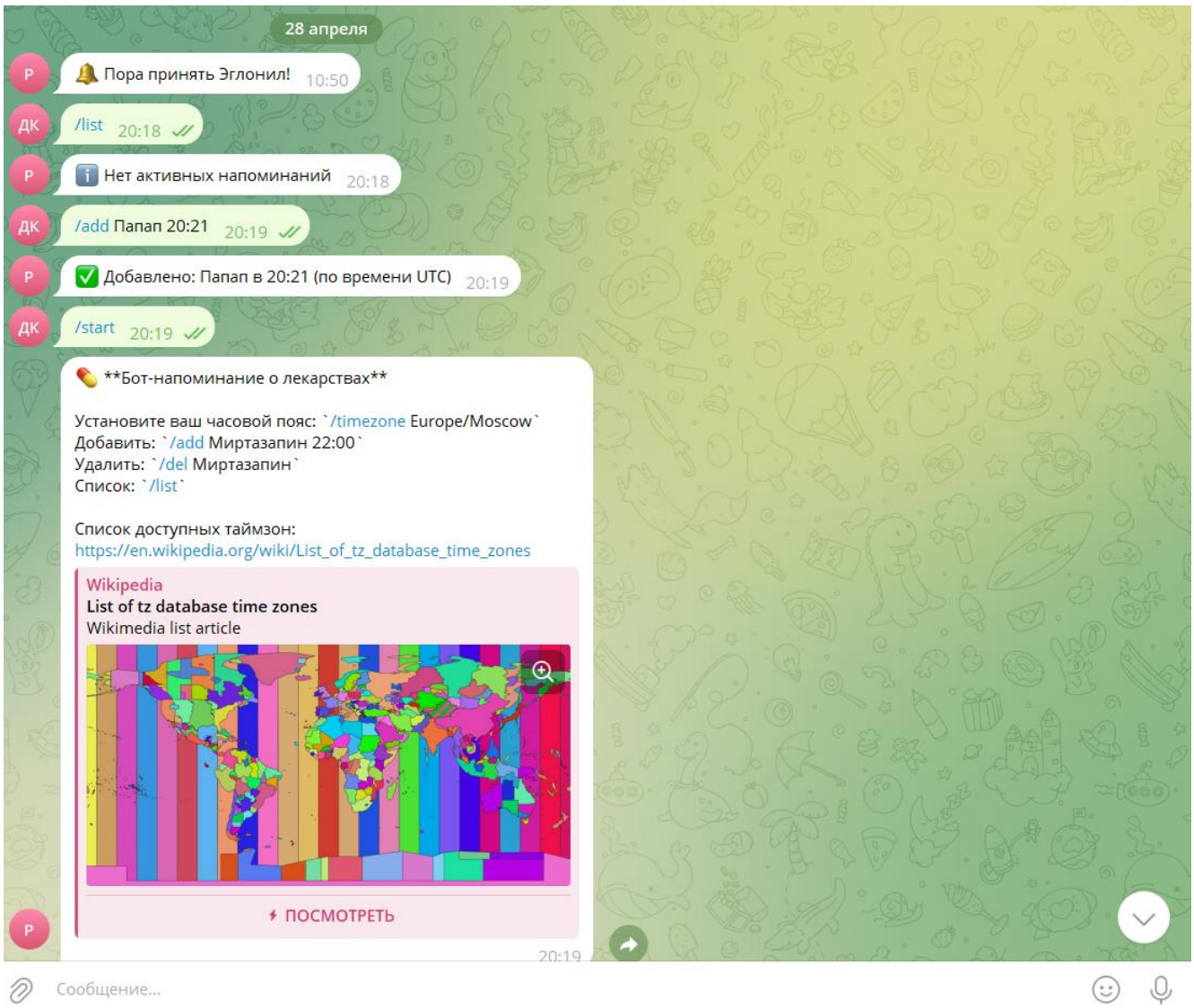

Рисунок 11 – Скриншот интерфейса бота

Ссылки:

- Репозиторий проекта
- Бот в Telegram

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")
```