**Московский государственный технический университет им. Н.Э. Баумана**

**Факультет ИУ Кафедра ИУ5**

**Курс «Основы информатики» Отчет по домашнему заданию**

Выполнил студент группы ИУ5-33Б: Козлов А. А.

Подпись и дата:

Проверил преподаватель каф.: Гапанюк Ю. Е.

Подпись и дата:

Москва, 2024 г

**Постановка задачи**

Реализовать небольшой проект на языке программирования, который ранее не был изучен.

1. **Выбранный язык программирования**

Python

1. **Подробности реализации задачи**

Изучить асинхронность на примере написания асинхронного кода для телеграмм-бота.

1. **Работа бота**

Пользователь вводит сообщение о запланированном событии с упоминанием даты. Бот по вызванной команде отправляет сообщение с напоминанием о запланированном событии, если дата соответствует дате события. Также в программе присутствует команда по очистке событий, чей срок уже вышел.

Доступные команды:

/add\_event - добавить новое событие (пример: /add\_event Название события, Дата события)

/remind - напомнить о событиях на сегодня

/clear - очистить все события до сегодняшнего дня

/help - показать это меню

**Текст** **программы**

import logging

import datetime

import json

from telegram.ext import CommandHandler, Application

logging.basicConfig(level=logging.INFO)

TOKEN = ‘\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*’

async def start(update, context):

    await context.bot.send\_message(chat\_id=update.effective\_chat.id, text='Бот запущен! \n\n'

                                                                        'Доступные команды: \n'

                                                                        '/add\_event - добавить новое событие (пример: /add\_event Название события, Дата события) \n'

                                                                        '/remind - напомнить о событиях на сегодня \n'

                                                                        '/clear - очистить все события до сегодняшнего дня \n'

                                                                        '/help - показать это меню')

def save\_events(events):

    with open('2ndSem-HW/events.json', 'w') as f:

        json.dump(events, f, indent=4)

def load\_events():

    try:

        with open('2ndSem-HW/events.json', 'r') as f:

            events = json.load(f)

            for event\_name, event\_date in events.items():

                events[event\_name] = datetime.datetime.strptime(event\_date, '%Y-%m-%d').date()

            return events

    except json.JSONDecodeError:

        return {}

    except FileNotFoundError:

        return {}

events = load\_events()

def get\_events():

    return events

async def add\_event(event\_name, event\_date):

    events[event\_name] = event\_date.strftime('%Y-%m-%d')  # Преобразовать дату в строку

    save\_events(events)

async def add\_event\_handler(update, context):

    message\_text = update.message.text

    if message\_text.startswith('/add\_event '):

        message\_text = message\_text.replace('/add\_event ', '')

    parts = message\_text.split(', ')

    if len(parts) != 2:

        await context.bot.send\_message(chat\_id=update.effective\_chat.id, text='Неправильный формат сообщения. Пожалуйста, отправьте сообщение в формате "Название события, Дата события"')

        return

    event\_name = parts[0]

    event\_date = parts[1]

    try:

        event\_date = datetime.datetime.strptime(event\_date, '%Y-%m-%d').date()

    except ValueError:

        await context.bot.send\_message(chat\_id=update.effective\_chat.id, text='Неправильный формат даты. Пожалуйста, отправьте дату в формате YYYY-MM-DD')

        return

    try:

        await add\_event(event\_name, event\_date)

        await context.bot.send\_message(chat\_id=update.effective\_chat.id, text='Событие добавлено!')

    except Exception as e:

        await context.bot.send\_message(chat\_id=update.effective\_chat.id, text='Ошибка добавления события: ' + str(e))

async def remind(update, context):

    events = load\_events()

    today = datetime.date.today()

    today\_events = [event for event, date in events.items() if date == today]

    if not today\_events:

        await context.bot.send\_message(chat\_id=update.effective\_chat.id, text='Событий нет сегодня')

    else:

        for event in today\_events:

            await context.bot.send\_message(chat\_id=update.effective\_chat.id, text=f'Напоминание: {event}')

async def clear\_before\_today(update, context):

    eventsForClear = load\_events()

    today = datetime.date.today()

    new\_events = {}

    for event, date in eventsForClear.items():

        if date >= today:

            new\_events[event] = date.strftime('%Y-%m-%d')

    save\_events(new\_events)

    await context.bot.send\_message(chat\_id=update.effective\_chat.id, text='Записи до сегодняшнего дня очищены')

async def help(update, context):

    await context.bot.send\_message(chat\_id=update.effective\_chat.id, text='Доступные команды: \n'

                                                                        '/add\_event - добавить новое событие (пример: /add\_event Название события, Дата события) \n'

                                                                        '/remind - напомнить о событиях на сегодня \n'

                                                                        '/clear - очистить все события до сегодняшнего дня \n'

                                                                        '/help - показать это меню')

def main():

    aplication = Application.builder().token(TOKEN).build()

    start\_handler = CommandHandler('start', start)

    gadd\_event\_handler = CommandHandler('add\_event', add\_event\_handler)

    remind\_handler = CommandHandler('remind', remind)

    clear\_handler = CommandHandler('clear', clear\_before\_today)

    help\_handler = CommandHandler('help', help)

    aplication.add\_handler(start\_handler)

    aplication.add\_handler(gadd\_event\_handler)

    aplication.add\_handler(remind\_handler)

    aplication.add\_handler(clear\_handler)

    aplication.add\_handler(help\_handler)

    try:

        aplication.run\_polling()

    except Exception as e:

        print('Ошибка запуска приложения: ' + str(e))

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

    main()

**Подробное описание программы**

Импорт библиотек:

import logging

import datetime

import json

from telegram.ext import CommandHandler, Application

Настраиваем сообщения в командной строке:

logging.basicConfig(level=logging.INFO)

Функция обработки команды /start:

async def start(update, context):

    await context.bot.send\_message(chat\_id=update.effective\_chat.id, text='Бот запущен! \n\n'

                                                                        'Доступные команды: \n'

                                                                        '/add\_event - добавить новое событие (пример: /add\_event Название события, Дата события) \n'

                                                                        '/remind - напомнить о событиях на сегодня \n'

                                                                        '/clear - очистить все события до сегодняшнего дня \n'

                                                                        '/help - показать это меню')

Вспомогательные функции сохранения записей в файл и выгрузки их из файла:

def save\_events(events):

    with open('2ndSem-HW/events.json', 'w') as f:

        json.dump(events, f, indent=4)

def load\_events():

    try:

        with open('2ndSem-HW/events.json', 'r') as f:

            events = json.load(f)

            for event\_name, event\_date in events.items():

                events[event\_name] = datetime.datetime.strptime(event\_date, '%Y-%m-%d').date()

            return events

    except json.JSONDecodeError:

        return {}

    except FileNotFoundError:

        return {}

Выгружаем файл:

events = load\_events()

Вспомогательная функция, возвращающая массив с записями:

def get\_events():

    return events

Функция для преобразования даты в строку:

async def add\_event(event\_name, event\_date):

    events[event\_name] = event\_date.strftime('%Y-%m-%d')  # Преобразовать дату в строку

    save\_events(events)

Функция для обработки получаемых записей и добавления их в фаил:

async def add\_event\_handler(update, context):

    message\_text = update.message.text

    if message\_text.startswith('/add\_event '):

        message\_text = message\_text.replace('/add\_event ', '')

    parts = message\_text.split(', ')

    if len(parts) != 2:

        await context.bot.send\_message(chat\_id=update.effective\_chat.id, text='Неправильный формат сообщения. Пожалуйста, отправьте сообщение в формате "Название события, Дата события"')

        return

    event\_name = parts[0]

    event\_date = parts[1]

    try:

        event\_date = datetime.datetime.strptime(event\_date, '%Y-%m-%d').date()

    except ValueError:

        await context.bot.send\_message(chat\_id=update.effective\_chat.id, text='Неправильный формат даты. Пожалуйста, отправьте дату в формате YYYY-MM-DD')

        return

    try:

        await add\_event(event\_name, event\_date)

        await context.bot.send\_message(chat\_id=update.effective\_chat.id, text='Событие добавлено!')

    except Exception as e:

        await context.bot.send\_message(chat\_id=update.effective\_chat.id, text='Ошибка добавления события: ' + str(e))

Функция обработки команды напоминания:

sync def remind(update, context):

    events = load\_events()

    today = datetime.date.today()

    today\_events = [event for event, date in events.items() if date == today]

    if not today\_events:

        await context.bot.send\_message(chat\_id=update.effective\_chat.id, text='Событий нет сегодня')

    else:

        for event in today\_events:

            await context.bot.send\_message(chat\_id=update.effective\_chat.id, text=f'Напоминание: {event}')

Функция обработки ошибки:

async def clear\_before\_today(update, context):

    eventsForClear = load\_events()

    today = datetime.date.today()

    new\_events = {}

    for event, date in eventsForClear.items():

        if date >= today:

            new\_events[event] = date.strftime('%Y-%m-%d')

    save\_events(new\_events)

    await context.bot.send\_message(chat\_id=update.effective\_chat.id, text='Записи до сегодняшнего дня очищены')

Функция обработки команды очистки просроченных напоминаний:

async def help(update, context):

    await context.bot.send\_message(chat\_id=update.effective\_chat.id, text='Доступные команды: \n'

                                                                        '/add\_event - добавить новое событие (пример: /add\_event Название события, Дата события) \n'

                                                                        '/remind - напомнить о событиях на сегодня \n'

                                                                        '/clear - очистить все события до сегодняшнего дня \n'

                                                                        '/help - показать это меню')

Функция main и ее запуск:

def main():

    aplication = Application.builder().token(TOKEN).build()

    start\_handler = CommandHandler('start', start)

    gadd\_event\_handler = CommandHandler('add\_event', add\_event\_handler)

    remind\_handler = CommandHandler('remind', remind)

    clear\_handler = CommandHandler('clear', clear\_before\_today)

    help\_handler = CommandHandler('help', help)

    aplication.add\_handler(start\_handler)

    aplication.add\_handler(gadd\_event\_handler)

    aplication.add\_handler(remind\_handler)

    aplication.add\_handler(clear\_handler)

    aplication.add\_handler(help\_handler)

    try:

        aplication.run\_polling()

    except Exception as e:

        print('Ошибка запуска приложения: ' + str(e))

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

    main()

**Анализ результатов**

