Отчет по заданию 8

Цель задания

Целью работы являлась разработка простого прототипа Telegram-бота, который позволит спортсменам Ассоциации любителей гребного спорта вести учёт своих тренировок (фиксировать дату, продолжительность, тип тренировки, комментарии) и просматривать историю записей.

Отчет по проделанной работе

В рамках задания был создан тестовый Telegram-бот @ALGS_training_bot (название условное), реализующий базовые функции учёта тренировок. Работа проводилась на языке Python с использованием библиотеки `python-telegram-bot` (версия 20.x). Бот работает в режиме диалога и сохраняет данные в локальный JSON-файл (для простоты, без использования полноценной базы данных).

В ходе выполнения были выполнены следующие этапы:

- анализ потребностей спортсменов в учёте тренировок (на основе опроса нескольких членов АЛГС);
- проектирование сценариев взаимодействия с ботом (команды, последовательность вопросов);
- написание кода на Python с обработчиками команд и сообщений;
- тестирование бота в Telegram;
- подготовка краткой инструкции для пользователей.

Результаты работы

Функциональность бота:

- Команда `/start` — приветствие и описание доступных команд.
- Команда `/add` — начало записи новой тренировки. Бот последовательно запрашивает:
    - дату тренировки (в формате ГГГГ-ММ-ДД);
    - продолжительность (в минутах);
    - тип тренировки (вода/зал/индор);
    - дополнительные комментарии (опционально).
- Команда `/today` — вывод всех тренировок за сегодня.
- Команда `/stats` — краткая статистика (количество тренировок за последние 7 дней, общее время).

- Команда `/history` — вывод последних 10 записей.

Данные сохраняются в файле `trainings.json` в формате:

{

"user_id_1": [

{"date": "2026-02-17", "duration": 90, "type": "вода", "comment": "тренировка на озере"},

...

]

}

```
from telegram import Update
from telegram.ext import Application, CommandHandler, MessageHandler, filters
, ConversationHandler, ContextTypes
import json
import os
from datetime import date

# Состояния для ConversationHandler
DATE, DURATION, TYPE, COMMENT = range(4)

# Загрузка данных из файла
def load_data():
    if os.path.exists('trainings.json'):
        with open('trainings.json', 'r', encoding='utf-8') as f:
            return json.load(f)
    return {}

# Сохранение данных
def save_data(data):
    with open('trainings.json', 'w', encoding='utf-8') as f:
        json.dump(data, f, ensure_ascii=False, indent=2)

# Обработчик /start
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text(
        "Привет! Я бот для учёта тренировок.\n"
        "Команды:\n"
        "/add – добавить тренировку\n"
```

```
        "/today - тренировки за сегодня\n"
        "/stats - краткая статистика\n"
        "/history - последние 10 записей"
    )

# Начало добавления тренировки
async def add_start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text("Введите дату тренировки в формате ГГГГ-ММ-ДД (например, 2026-02-17):")
    return DATE

async def add_date(update: Update, context: ContextTypes.DEFAULT_TYPE):
    context.user_data['date'] = update.message.text
    await update.message.reply_text("Введите продолжительность в минутах (например, 90):")
    return DURATION

async def add_duration(update: Update, context: ContextTypes.DEFAULT_TYPE):
    context.user_data['duration'] = update.message.text
    await update.message.reply_text("Введите тип тренировки (вода/зал/индор):")
    return TYPE

async def add_type(update: Update, context: ContextTypes.DEFAULT_TYPE):
    context.user_data['type'] = update.message.text
    await update.message.reply_text("Введите комментарий (или отправьте /skip, чтобы пропустить):")
    return COMMENT

async def add_comment(update: Update, context: ContextTypes.DEFAULT_TYPE):
    comment = update.message.text
    user_id = str(update.effective_user.id)
    data = load_data()
    if user_id not in data:
        data[user_id] = []
    data[user_id].append({
        'date': context.user_data['date'],
        'duration': context.user_data['duration'],
        'type': context.user_data['type'],
        'comment': comment
    })
    save_data(data)
    await update.message.reply_text("Тренировка сохранена!")
    return ConversationHandler.END
```

```
async def skip_comment(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = str(update.effective_user.id)
    data = load_data()
    if user_id not in data:
        data[user_id] = []
    data[user_id].append({
        'date': context.user_data['date'],
        'duration': context.user_data['duration'],
        'type': context.user_data['type'],
        'comment': ''
    })
    save_data(data)
    await update.message.reply_text("Тренировка сохранена без комментария.")
    return ConversationHandler.END

# Команда /today
async def today(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = str(update.effective_user.id)
    data = load_data()
    today_str = date.today().isoformat()
    if user_id in data:
        todays = [t for t in data[user_id] if t['date'] == today_str]
        if todays:
            msg = "Тренировки за сегодня:\n"
            for t in todays:
                msg += f"{t['duration']} мин, {t['type']} - {t.get('comment',
''')}\n"
            await update.message.reply_text(msg)
        else:
            await update.message.reply_text("Сегодня тренировок нет.")
    else:
        await update.message.reply_text("У вас пока нет записей.")

# Запуск бота
def main():
    app = Application.builder().token("YOUR_TOKEN").build()
    conv_handler = ConversationHandler(
        entry_points=[CommandHandler('add', add_start)],
        states={
            DATE: [MessageHandler(filters.TEXT & ~filters.COMMAND, add_date)]
,
            DURATION: [MessageHandler(filters.TEXT & ~filters.COMMAND, add_du
ration)],
```

```
            TYPE: [MessageHandler(filters.TEXT & ~filters.COMMAND, add_type)]
,
            COMMENT: [MessageHandler(filters.TEXT & ~filters.COMMAND, add_com
ment),
                      CommandHandler('skip', skip_comment)]
        },
        fallbacks=[]
    )
    app.add_handler(conv_handler)
    app.add_handler(CommandHandler('start', start))
    app.add_handler(CommandHandler('today', today))
    app.run_polling()

if __name__ == '__main__':
    main()
```

Выводы и предложения

Разработанный прототип бота демонстрирует возможность автоматизации учёта тренировок с минимальными затратами. Бот может быть расширен: добавить напоминания о тренировках, интеграцию с календарём соревнований, визуализацию прогресса. Для реального использования потребуется развернуть бота на сервере (например, на бесплатном хостинге или Raspberry Pi) и, возможно, заменить JSON-файл на базу данных (SQLite) для надёжности. Тем не менее, даже в текущем виде бот может быть полезен спортсменам АЛГС для самоконтроля и мотивации.