МУНИЦИПАЛЬНОЕ АВТОНОМНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ГОРОДА РОСТОВА-НА-ДОНУ   
«КЛАССИЧЕСКИЙ ЛИЦЕЙ № 1»

**Проектная работа**

Telegram-бот “напоминалка” для олимпиад

Выполнил:

Лазимов Д. И., лицеист 10 ω класса МАОУ "Классический лицей № 1"

Научный руководитель:

Пусева О.Н., учитель информатики МАОУ "Классический лицей № 1"

2023г.

ОГЛАВЛЕНИЕ

[Введение 3](#_Toc103286896)

[Цели и задачи 3](#_Toc103286897)

[Актуальность 4](#_Toc103286898)

[Анализ существующих решений 4](#_Toc103286899)

[Дорожная карта 5](#_Toc103286900)

[Описание разработанного решения 6](#_Toc103286901)

[Заключение 14](#_Toc103286902)

# Введение

С каждым годом ЕГЭ все усложняется и усложняется. Чтобы перестраховаться от непредвиденных случаев, и чтобы поступить в университет мечты (вероятно, на вашем факультете будет очень большая конкуренция) многие люди решают учувствовать в олимпиадах, чтобы в дальнейшем получим преференции при поступлении.

Данный проект был создан для информирования людей об олимпиадах, которые они планируют писать. Он состоит из мобильного Тelegram-бота, напоминающего о последних новостях олимпиадного мира. За информационную основу был взят сайт, откуда я и беру сами олимпиады и новости.

Приложение написано на языке программирования Python в среде Visual Studio. Также использовал базу данных SQLite 3 для хранения информации о пользователе и его олимпиадах. И написал свою api, к которой обращаюсь для удобного получения информации с сайта. К api прилагаются и парсеры.

## Цели и задачи

Цель: создать Telegram-бота для информирования людей о последних новостях олимпиад

Для этого были поставлены и решены следующие задачи:

1. Написание алгоритма работы программы.
2. Написание парсеров для удобного сбора информации с сайта
3. Написание api для удобного обращения к собранной информации
4. Разработка понятного интерфейса для бота.
5. Разработка возможности просмотра своих олимпиад.
6. Разработка алгоритма для сообщения новостей пользователям
7. Создание БД

# 

# Актуальность

1. **Человеческий фактор:** Многие люди, которые интересуются олимпиадами иногда забывают пройти отборочный тур на олимпиаде, не успевают подать заявление на финальный отбор, поздно узнают про опубликованный список прошедших на какой-то из туров, из-за чего опять-таки не могут победить в олимпиаде и т.д. Никто не отменял человеческий фактор, поэтому на помощь придет бот помощник, который будет напоминать вам о новостях олимпиадного мира.
2. **Вся информация в одном месте:** По большему счету нет какого-то единого сайта, на котором бы хранились все олимпиады и последние новости, но даже если и есть (а он есть), то сталкиваешься с неудобным интерфейсом, который не понятен пользователю, неструктурированной информацией, когда заходишь на сайт, а там миллион вложений, в которых легко запутаться. Но мой бот также решает эту проблему. Интерфейс просто, поэтому понятен любому пользователю, в нем представлен только необходимый минимум функций, а пользователю чтобы не пропустить свою олимпиаду надо просто их выбрать и больше ни о чем не думать, кроме подготовки.

### Анализ существующих решений

На данный момент есть следующие решения для удобного отслеживания последних новостей олимпиад:

1. Веб-сайт
2. Теlegram-бот
3. Приложение на телефон и/или компьютер

Данный проект относится к решениям второго типа. Преимущества данной программы в том, что:

1. Telegram популярная соц-сеть, где многие люди проводят много времени, поэтому интегрируя данную идею в бота мы делаем удобнее пользователю, т.к все находиться в одном месте
2. Для разработки бота нужно меньше времени и сил, чем на другие решения
3. Отсутствует реклама и другие отвлекающие факторы.
4. Открытый исходный код.

### 

### Дорожная карта

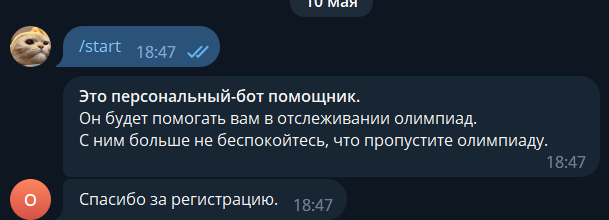
|  |  |
| --- | --- |
| Тема проекта | Telegram-бот “напоминалка” для олимпиад |
| ФИО | Лазимов Даниил Ибрагимович |

|  |  |  |  |
| --- | --- | --- | --- |
| № | Содержание деятельности | Даты | В чем результат? |
|  | Анализ конкурентов и выявление слабых мест уже существующих продуктов. | Ноябрь 2022 | Выявлены недостатки конкурентов. |
|  | Составление плана разработки и списка требуемых библиотек. | Декабрь 2022 | Составлен план и список библиотек. |
|  | Начало разработки продукта. | Январь 2023 | Начата активная разработка продукта. |
|  | Завершение разработки и тестирование получившегося продукта. | Март 2023 | Выявлены недочеты в работе продукта, а также получен отзыв пользователя. |
|  | Доработка продукта после выявленных недочетов | Май 2023 | Добавлены недостающие функции, переработан дизайн и улучшена стабильность. |

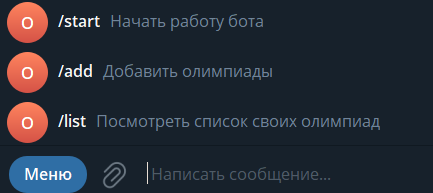
### 

### Описание разработанного решения

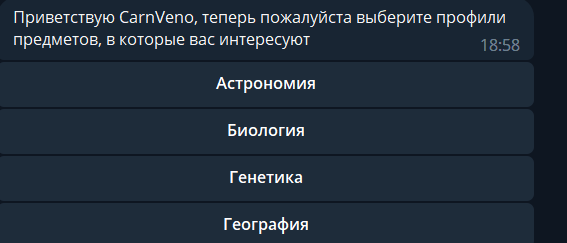
1. После первого запуска командой /start бота вы уже будете зарегистрированы в боте

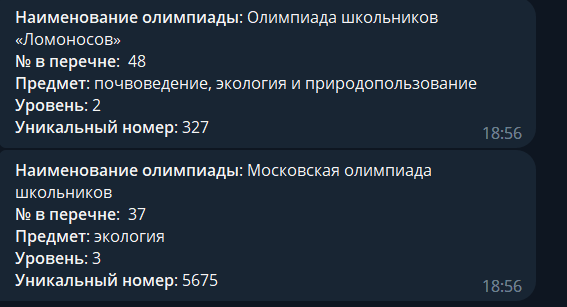


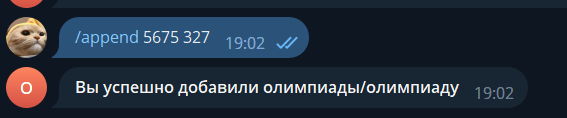
1. После успешной авторизации мы можем посмотреть список добавленных олимпиад или добавить новые



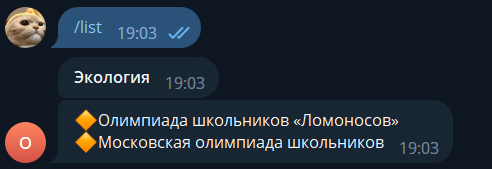
1. Далее, когда мы решим добавить олимпиады появится список из их профилей.



1. Потом бот выводит список из олимпиад по выбранному профилю
2. Теперь с помощью команды /append можно добавить олимпиады к себе в профиль



1. Также можно посмотреть список своих олимпиад с помощью команды /list



**Используемые технологии**

**Используемые библиотеки:**

1. aiogram
2. aioschedule
3. asyncio
4. bs4
5. emoji
6. Flask
7. lxml
8. re
9. requests
10. datetime

Реализация:

* + - 1. Для написания бота использовал библеотеку aiogram

from aiogram import Bot, types

from aiogram.dispatcher import Dispatcher

from aiogram.utils import executor

from aiogram.types import CallbackQuery

from aiogram.types import InlineKeyboardMarkup, InlineKeyboardButton, ReplyKeyboardMarkup, ReplyKeyboardRemove, \

ReplyKeyboardMarkup, KeyboardButton

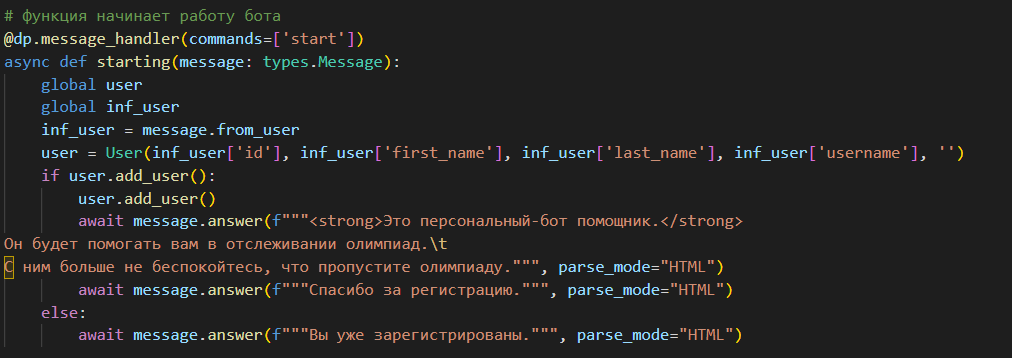
from aiogram.utils.callback\_data import CallbackData

* + - 1. Создаем бота

bot = Bot(token=TOKEN)

dp = Dispatcher(bot)

* + - 1. Функция начинает работу бота



* + - 1. Функция выводит профили олимпиад

@dp.message\_handler(commands=['add'])

async def olimpiads(message: types.Message):

    usern = user.usernam()

    url = 'http://127.0.0.1:8000/list\_olimpix'

    response = requests.get(url=url).json()

    keyboard = types.InlineKeyboardMarkup()

    for elem in response:

        keyboard.add(types.InlineKeyboardButton(text=elem, callback\_data=f"p:{elem}:{response.index(elem)}"))

    await message.answer(f"""Приветствую {usern}, теперь пожалуйста выберите профили предметов, в которые вас интересуют""", parse\_mode="HTML", reply\_markup=keyboard)

* + - 1. Функция выводит список олимпиад

@dp.callback\_query\_handler(text\_startswith="p")

async def find\_in\_prof(query: CallbackQuery):

    global prof, ind

    await query.answer()

    await query.message.delete()

    data = query.data.split(':')

    prof = data[-2]

    ind = int(data[-1])

    url = 'http://127.0.0.1:8000/olimpix'

    response = requests.get(url=url).json()

    await query.message.answer("""Когда будете выбирать олимпиады пожалуйста введите команду /append и введите цифры уникального кода олимпиады""", parse\_mode="HTML")

    await query.message.answer("""Например /append 5251 1324 4314 и т.д""", parse\_mode="HTML")

    for i in response[ind][prof]:

        dataset = i["".join(list(i.keys()))]

        num = dataset[0]

        desc = dataset[1]

        level = dataset[2]

        uniq = dataset[-1].split('/')[-1]

        await query.message.answer(f"""<strong>Наименование олимпиады</strong>: {"".join(list(i.keys()))}

<strong>№ в перечне</strong>:  {num}

<strong>Предмет</strong>: {desc}

<strong>Уровень</strong>: {level}

<strong>Уникальный номер</strong>: {uniq} """, parse\_mode="HTML")

* + - 1. Функция добавляет олимпиады к пользователю

@dp.message\_handler(commands=['append'])

async def appending(message: types.Message):

    try:

        com = "".join(message.get\_full\_command()[1])

        com = re.findall(r'(?i)([0-9]+)', com)

        url = 'http://127.0.0.1:8000/olimpix'

        data = []

        response = requests.get(url=url).json()[ind][prof]

        for i in response:

            dataset = i["".join(list(i.keys()))]

            uniq = dataset[-1].split('/')[-1]

            if str(uniq) in com:

                data.append(uniq)

        user.update\_info\_user(data)

        await message.answer(f"""<strong>Вы успешно добавили олимпиады/олимпиаду</strong>""", parse\_mode="HTML")

    except Exception:

        await message.answer(f"""<strong>Введен некорректный запрос</strong>""", parse\_mode="HTML")

* + - 1. Функция выводит список олимпиад пользователю

@dp.message\_handler(commands=['list'])

async def list\_olimpiads(message: types.Message):

    lst = user.get\_list()

    try:

        if lst == [] or lst == ['']:

            raise Exception

        if lst[0] == '':

            del lst[0]

        url = 'http://127.0.0.1:8000/list'

        response = requests.get(url=url).json()

        a = []

        for i in response:

            dictt = {}

            for key, val in i.items():

                names = []

                for j in val:

                    if j[0] in lst:

                        names.append(j[1])

                        del lst[lst.index(j[0])]

                if names != []:

                    dictt[key] = names

            if dictt != {}:

                a.append(dictt)

        for i in a:

            for key, val in i.items():

                await message.answer(f"""<strong>{key}</strong>""", parse\_mode="HTML")

                await message.answer('\n'.join(list(map(lambda x: emoji.emojize(':large\_orange\_diamond:') + x, val))))

    except Exception:

        await message.answer(f"""<strong>Пока что вы не добавили ни одной олимпиады</strong>""", parse\_mode="HTML")

* + - 1. Функция новости сравнивает по времени

@dp.message\_handler()

async def choose\_your\_dinner():

    lst = user.get\_list()

    try:

        if lst == [] or lst == ['']:

            raise Exception

        if lst[0] == '':

            del lst[0]

        time = datetime.datetime.now()

        for i in lst:

            url = f'http://127.0.0.1:8000/news/{i}'

            response = requests.get(url=url).json()

            tm = response[0]

            date = int(tm[0])

            month = int(tm[1])

            year = int(tm[2])

            # для демонтстрации

            # date = time.day

            # month = time.month

            # year = time.year

            print(time.year, time.day, time.month)

            if int(time.year) == year and int(time.day) == date and int(time.month) == month:

                await bot.send\_message(chat\_id=inf\_user['id'], text=f"""<strong>Новость по олимпиаде {response[-1]}</strong>""", parse\_mode="HTML")

                await bot.send\_message(chat\_id=inf\_user['id'], text=f"""{response[1]}""", parse\_mode="HTML")

    except Exception:

        pass

* + - 1. Функция которая в определенное время выводит новости пользователю

async def scheduler():

    aioschedule.every().day.at("17:30").do(choose\_your\_dinner)

    while True:

        await aioschedule.run\_pending()

        await asyncio.sleep(1)

* + - 1. Для удобства обращения к собранной информации сделал api

from flask import Flask, jsonify

from find\_list import find\_list

from profiles\_parser import profiles

from list\_profiles import list\_profiles

from news\_parser\_olimp import newss

from news\_parser import lsst

app = Flask(\_\_name\_\_)

app.config['SECRET\_KEY'] = 'abobus'

app.config['JSON\_AS\_ASCII'] = False

headers = {

        "accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.9",

        'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.84 Safari/537.36'

    }

class IDError(Exception):

    pass

@app.route('/find\_list')

def find\_listt():

    return jsonify(find\_list(headers=headers))

@app.route('/olimpix')

def olimpix():

    return jsonify(profiles())

@app.route('/list\_olimpix')

def list\_olimpix():

    return jsonify(list\_profiles())

@app.route('/news/<int:num>')

def news(num):

    return jsonify(newss(str(num)))

@app.route('/list')

def listt():

    return jsonify(lsst())

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

    app.run(port=8000, host="127.0.0.1", debug=True)

* + - 1. Для сбора информации написал парсеры
      2. Для удобства каждый для каждого пользователя создается класс

class User:

    def \_\_init\_\_(self, id, first\_name, last\_name, username, olimpiads):

        self.con = sqlite3.connect("Bot\_users.db")

        self.id = id

        self.fr = first\_name

        self.lr = last\_name

        self.username = username

        self.olimps = olimpiads

    # добавляем нового пользователя

    def add\_user(self):

        cur = self.con.cursor()

        users\_id = list(map(lambda x: x[0], cur.execute(f"SELECT user\_id from users").fetchall()))

        if self.id not in users\_id:

            cur.execute(f"INSERT INTO users VALUES(?, ?, ?, ?, ?)", (self.id, self.fr, self.lr , self.username, self.olimps))

            self.olimps = []

            self.con.commit()

            return True

        else:

            self.olimps = list(set("".join(map(lambda x: x[0], cur.execute("SELECT olimpiads from users WHERE user\_id = (?)", (str(self.id), )).fetchall())).split(';')))

            return False

    # обновляем информацию об олимпиадах пользователя

    def update\_info\_user(self, olimps):

        self.olimps += list(set(olimps))

        st = ";".join(set(self.olimps))

        cur = self.con.cursor()

        cur.execute(f"""UPDATE users SET olimpiads = (?) WHERE user\_id = (?)""", (st, self.id))

        self.con.commit()

    # берем список из олимпиад пользователя

    def get\_list(self):

        cur = self.con.cursor()

        return list(set("".join(map(lambda x: x[0], cur.execute("SELECT olimpiads from users WHERE user\_id = (?)", (str(self.id), )).fetchall())).split(';')))

    # получаем ник пользователя

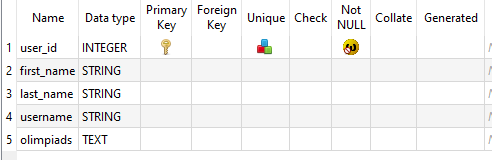
    def usernam(self):

        cur = self.con.cursor()

        usersn = list(map(lambda x: x[0], cur.execute(f"""SELECT username from users WHERE user\_id = '{self.id}'""").fetchall()))

        return usersn[0]

13 Создание БД для хранения информации о пользователе



Ссылка на полный код:

<https://github.com/zelell1/bot_olimp>

### 

### Заключение

В ходе работы над индивидуальным проектом была достигнута поставленная цель, которая заключалась в создании Telegram-бота для напоминания об олимпиадах и об их полседних новостях

Для реализации поставленной цели мною были достигнуты задачи:

1) Написание парсеров для удобного сбора информации и бд для хранения информации о пользователях

2) Написание своего api для удобного обращения к собранной информации

3) Создание Telegram-бота и описание технологии напоминания

Этот Telegram-бот отличный помощник для информирования людей об олимпиадах. С ним можно будет больше не беспокоиться о том, что вы не будете в курсе последних новостей о ваших олимпиадах

**Список источников**

* + - 1. https://docs.python.org/3/
      2. https://docs.aiogram.dev/en/latest/
      3. https://olimpiada.ru/
      4. https://core.telegram.org/