МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Дополнительное образование «Основы промышленного программирования»

проект по теме

«Бот для поиска информации об олимпиадах»

Ученик Коростелев М.А.

Методист Копытина Е.А.

Воронеж 2024 – 2025

Оглавление

[Введение 3](#_Toc192533132)

[Основная часть. Возникновение идеи проекта. 4](#_Toc192533133)

[Основная часть. Использование телеграм-бота 5](#_Toc192533134)

[Выводы 9](#_Toc192533135)

[Список литературы 10](#_Toc192533136)

[Приложение A 11](#_Toc192533137)

# Введение

В настоящее время все чаще видим большое количество телеграмм-ботов. Телеграмм боты могут использоваться для автоматизации определенных задач, таких как:

* обработка заказов,
* уведомлений и организации массовых рассылок.

Боты могут также использоваться для:

* игр
* обучения
* поиска информации и предоставления других сервисов и функций

Они позволяют пользователям взаимодействовать с ботами на удобном для них языке, просто набрав определенные команды в чате. Телеграмм боты также позволяют массово выполнять однотипные задачи, такие как составление списка дел, отправка сообщений на определенное время или в определенном формате.

Для того чтобы написать такой бот используются различные технологии, в том числе и язык программирования Python и среду программирования Visual Studio Code. Современное общество испытывает все больший интерес к олимпиадам. Это связано не только со стремлением поступить в ВУЗ, но и с жаждой информации об этих мероприятиях, их участниках, расписании и результатах. В условиях информационной перегрузки люди нуждаются в эффективном средстве доступа к актуальным данным о олимпиадах.

Предмет – телеграмм-бот. Целью данного проекта является разработка бота, который может осуществлять поиск информации о олимпиадах, предоставляя пользователям доступ к актуальным данным о различных событиях. Конкретные задачи включают в себя:

1. Сбор и систематизацию информации о различных олимпиадах.

2. Разработка алгоритма эффективного поиска информации, основанного на потребностях пользователя.

3. Создание бота, обеспечивающего удобный и быстрый доступ к собранным данным. Для достижения этой цели мы будем использовать методы анализа данных и системные подходы.­­­

Важность проекта заключается в разработке удобного инструмента для школьников, который позволяет пользователям быть в курсе последних событий в мире олимпиад.

# ****Основная часть. Возникновение идеи проекта**.**

**Идея проекта возникла из-за моего активного образа жизни и участия в различных школьных олимпиадах. Столкнувшись с проблемой отслеживания олимпиад, я понял, что необходим удобный способ поиска информации о них.**

**Будучи школьником, я часто сталкивался с тем, что поиск олимпиад, соответствующих моим интересам и уровню подготовки, отнимал у меня много времени, и это не только отнимало много времени, но и мешало мне эффективно планировать свою учебную и внеклассную деятельность.**

**В этот момент мне пришла в голову идея создать бота для автоматизации процесса поиска олимпиад и соревнований. Я понял, что такой бот не только сэкономит мне массу времени, но и поможет быть в курсе всех текущих событий и предложений в мире образования и науки. Поэтому мой активный образ жизни и желание максимально использовать свое время привели меня к созданию проекта, который мог бы помочь не только мне, но и другим школьникам, которые хотят активно участвовать в школьных и научных мероприятиях. Чтобы это сделать мне пришлось углубиться в этой теме. Я начал читать книги на эту теме, и наткнулся на книгу «**Изучаем Python» [1]. Она мне помогла мне написать основную часть кода, но также нельзя забывать про книги [2], [3], [4], [5] и [6], они мне помогли дописать оставшуюся часть кода [Приложение A].

# ****Основная часть****. Использование телеграм-бота

Но как его использовать? Выбираешь тему, отправляешь ссылку на сообщество и получаешь результат!

Но конкретнее:

**@dp.message\_handler(commands=["start"])**

**async def start(message: types.Message):**

**keyboard = InlineKeyboardMarkup(row\_width=2)**

**buttons = [InlineKeyboardButton(text=topic, callback\_data=topic) for topic in TOPICS.keys()]**

**keyboard.add(\*buttons)**

**await message.reply('Выберите тему для поиска:', reply\_markup=keyboard)**

**Эта функция отвечает за обработку команды /start и отправляет пользователю клавиатуру с кнопками для выбора темы поиска.**

**@dp.callback\_query\_handler(lambda callback\_query: callback\_query.data in TOPICS.keys())**

**async def choose\_topic(callback\_query: types.CallbackQuery):**

**topic = callback\_query.data**

**await bot.answer\_callback\_query(callback\_query.id)**

**await bot.send\_message(callback\_query.message.chat.id,**

**'Введите ссылку на сообщество ВКонтакте: ')**

**url = message.text[15:]**

**bot.current\_topic = topic**

**Эта функция обрабатывает выбор темы пользователем через кнопки и запрашивает ссылку на сообщество ВКонтакте.**

**@dp.message\_handler(lambda message: message.text.startswith('https://vk.com/'))**

**async def handle\_community\_link(message: types.Message):**

**community\_link = message.text**

**process\_url(community\_link)**

**community\_id = re.search(r'/(\d+)', community\_link).group(1)**

**vk\_api = VkApi(token=VK\_TOKEN)**

**vk\_api = vk\_api.get\_api()**

**results = get\_posts\_by\_topic(vk\_api, community\_id, bot.current\_topic)**

**if results:**

**await bot.send\_message(message.chat.id, 'Вот ссылки на посты, найденные по вашему запросу:')**

**for post in results:**

**keyboard = InlineKeyboardMarkup()**

**keyboard.add(InlineKeyboardButton("Ссылка", url=post['link']))**

**await bot.send\_message(message.chat.id, f"Заголовок: {post['title']}", reply\_markup=keyboard)**

**else:**

**await bot.send\_message(message.chat.id, 'К сожалению, ничего не найдено по вашему запросу.')**

**Эта функция обрабатывает сообщение с ссылкой на сообщество ВКонтакте и запускает процесс получения постов по выбранной теме из этого сообщества.**

**@dp.message\_handler(commands=["start"])**

**async def start(message: types.Message):**

**Эта функция является обработчиком команды /start. Когда пользователь отправляет эту команду, бот отправляет сообщение с клавиатурой для выбора темы поиска.**

**@dp.callback\_query\_handler(lambda callback\_query: callback\_query.data in TOPICS.keys())**

**async def choose\_topic(callback\_query: types.CallbackQuery):**

**Эта функция обрабатывает выбор пользователя через кнопки с темами поиска. После выбора темы пользователем, бот запрашивает ссылку на сообщество ВКонтакте.**

**@dp.message\_handler(lambda message: message.text.startswith('https://vk.com/'))**

**async def handle\_community\_link(message: types.Message):**

**Эта функция обрабатывает сообщение пользователя, содержащее ссылку на сообщество ВКонтакте. Бот извлекает идентификатор сообщества из ссылки и запускает процесс получения постов по выбранной теме из этого сообщества.**

Телеграм боты для поиска олимпиад могут быть полезными в различных ситуациях, например:

1. Уменьшение времени поиска олимпиад в интернете. Из-за переизбытка информации сложно найти нужную и достоверную, поэтому бот поможет делать это быстрее и качественнее.
2. При отправке команды «/start» бот отправляет нам сообщение с выбором темы запроса:

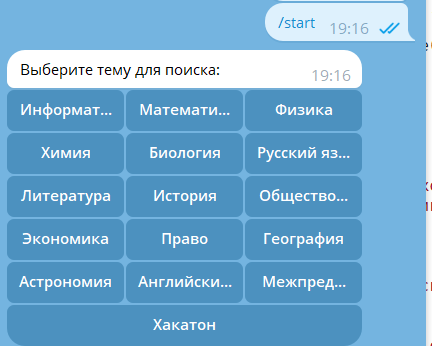
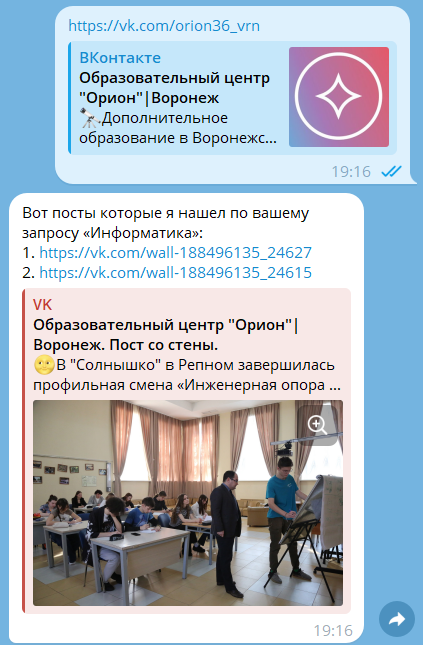


Рис.1 – Старт

При выборе темы нам нужно отправить ссылку на источник, от куда бот будет брать информацию:

****

**Рис.2 – Отправка ссылки и получение ответа**

1. **Мы получили корректный и нужный нам ответ.**

**.**

# Выводы

* 1. Понимание принципов работы с языком программирования Python: для создания бота, способного искать информацию, необходимо понимать, как работает этот метод поиска.
  2. Разработка алгоритма: для создания бота, необходимо разработать алгоритм, который сможет выполнять поиск постов.
  3. Написание кода: после того, как разработался алгоритм, нужно был написать код для его реализации. Это может потребовать определенных навыков программирования.
  4. Тестирование: после того, как написан код, нужно было протестировать его, чтобы убедиться, что бот работает правильно.
  5. Проблемы с интеграцией бота с платформой, на которой он должен работать.
  6. Нехватка времени и ресурсов. Создание бота может занять много времени и требовать значительных ресурсов, таких как вычислительной мощности, знаний и памяти.
  7. Поставленные цели достигнуты. Задачи выполнены. Бот создан, оттестирован. И он работает!

При разработке бота были использованы современные технологии программирования, а также были учтены особенности интеграции с платформой Телеграмм, что позволило создать простой и удобный в использовании инструмент для поиска информации.

В итоге, проект демонстрирует, что использование бота может быть полезным и эффективным способом обеспечения информации в цифровой среде.

**Перспективные направления развития работы:**

* Улучшить код, его стабильность и производительность;
* Сделать возможность поиска по глобальной сети Интернет

Примечание: чтоб запустить бот нужно перейти по ссылке. <https://olimpfind.glitch.me/>

# ****Список литературы****

1. Лутц М. «Изучаем Python», 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 1280 с.
2. Рейтц К., Шлюссер Т. «Автостопом по Python». – СПб.: Питер, 2017. – 336 с.: ил. – (Серия «Бестселлеры O’Reilly»).
3. Карлсон К., Мартинес М. «Python для детей: самоучитель по программированию». – СПб.: Питер, 2019. – 352 с.
4. Лангтон Д., Моуэн К., Майклсон Д. «Python для детей: самоучитель по программированию». – СПб.: Питер, 2019. – 336 с.
5. Джон Скит. "Совершенный код". - Москва: Вильямс, 2017. - 544 с.
6. Аллен Дауни. "Чистый код: Создание, анализ и рефакторинг". - Москва: ДМК Пресс, 2019. - 464 с.

**Приложение A**

**Код проекта**

from aiogram.types import InlineKeyboardMarkup, InlineKeyboardButton

from aiogram.contrib.fsm\_storage.memory import MemoryStorage

from aiogram import Bot, Dispatcher, types, executor

import aiohttp, logging, re

from os import environ

# Настройка логирования

logging.basicConfig(level=logging.INFO)

logging.basicConfig(

    format="%(filename)s[LINE:%(lineno)d]# %(levelname)-8s [%(asctime)s]  %(message)s",

    level=logging.DEBUG,

    filename="mylog.log",

)

# Ваши токены

TELEGRAM\_TOKEN = (environ['TELEGRAM\_TOKEN'])

bot = Bot(token=TELEGRAM\_TOKEN)

dp = Dispatcher(bot, storage=MemoryStorage())  # Добавляем хранилище состояний

VERSION = "5.199"

kol\_post = 200

TOKEN\_USER = (environ['TOKEN\_USER'])

# Словарь с темами для выбора

TOPICS = {

    "Информатика": r"(?:информатика|инфо|комп\'" + r"|программирование)",

    "Математика": r"(?:математика|математик)",

    "Физика": r"(?:физика|физик)",

    "Химия": r"(?:химия|химик)",

    "Биология": r"(?:биология|биолог)",

    "Русский язык": r"(?:русский язык|русский|язык)",

    "Литература": r"(?:литература|лит)",

    "История": r"(?:история|историк)",

    "Обществознание": r"(?:Обществознание|общество)",

    "Экономика": r"(?:экономика|экономист)",

    "Право": r"(?:право|правовед)",

    "География": r"(?:география|географ)",

    "Астрономия": r"(?:астрономия|астроном)",

    "Английский язык": r"(?:английский язык|английский|язык)",

    "Межпредметная": r"(?:межпредметная)",

    "Хакатон": r"(?:хакатон|программирование)",

}

# Создаем клавиатуру для выбора темы

keyboard = InlineKeyboardMarkup(row\_width=3)

buttons = [

    InlineKeyboardButton(text=topic, callback\_data=topic) for topic in TOPICS.keys()

]

keyboard.add(\*buttons)

# Обработчик команды /start

@dp.message\_handler(commands=["start"])

async def process\_start\_command(message: types.Message):

    user\_id = message.from\_user.id

    user\_name = message.from\_user.username

    keyboard = InlineKeyboardMarkup(row\_width=3)

    buttons = [

        InlineKeyboardButton(text=topic, callback\_data=topic) for topic in TOPICS.keys()

    ]

    keyboard.add(\*buttons)

    await message.answer("Выберите тему для поиска: ", reply\_markup=keyboard)

# Обработчик инлайн кнопок с выбором темы

@dp.callback\_query\_handler(lambda callback\_query: callback\_query.data in TOPICS.keys())

async def choose\_topic(callback\_query: types.CallbackQuery):

    await bot.answer\_callback\_query(callback\_query.id)

    global current\_topic

    current\_topic = callback\_query.data

    chat\_id = callback\_query.message.chat.id

    await bot.send\_message(chat\_id, f"Введите ссылку на сообщество ВКонтакте: ")

# Обработчик ввода ссылки на сообщество ВКонтакте

@dp.message\_handler(content\_types=types.ContentType.TEXT)

async def receive\_community\_link(message: types.Message):

    if len(message.text) <= len("https://vk.com/"):  # Проверка ссылки на корректность

        await message.answer("Не коректный ввод:(", reply\_markup=keyboard)

    elif message.text.startswith("https://vk.com/"):

        global community\_id

        global current\_topic

        community\_id = message.text.split("/")[-1]

        result\_with\_olimp, result\_whiout\_olimp = await vkpost(

            community\_id, current\_topic

        )

        if result\_with\_olimp:

            message\_for\_user = (

                f"Вот посты которые я нашел по вашему запросу «{current\_topic}»:\n"

            )

            message\_for\_user += "\n".join(

                [f"{i + 1}. {post}" for i, post in enumerate(result\_with\_olimp)]

            )

        else:

            message\_for\_user = "Постов не найдено. "

            if result\_whiout\_olimp:

                message\_for\_user += "Но может вам подайдет что-то из этого?:\n"

                message\_for\_user += "\n".join(

                    [f"{i + 1}. {post}" for i, post in enumerate(result\_whiout\_olimp)]

                )

        await message.answer(message\_for\_user)

        await message.answer("Выберите тему для поиска: ", reply\_markup=keyboard)

# Функция для поиска постов в сообществе ВКонтакте

async def vkpost(community\_id, current\_topic):

    DOMAIN = community\_id

    links\_with\_olimp = []

    links\_without\_olimp = []

    async with aiohttp.ClientSession() as session:

        response = await session.get(

            "https://api.vk.com/method/wall.get",

            params={

                "access\_token": TOKEN\_USER,

                "v": VERSION,

                "domain": DOMAIN,

                "count": kol\_post,

            },

        )

        data = await response.json()

    for post in data.get("response", {}).get("items", []):

        text = post.get("text", "")

        if current\_topic == "Хакатон":

            if any(

                    re.search(rf"\b{current\_topic}\b", word, re.IGNORECASE)

                    for word in re.split(r"\W+", text)

            ):

                links\_with\_olimp.append(

                    f"https://vk.com/wall{post['owner\_id']}\_{post['id']}"

                )

        else:

            if any(

                    re.search(rf"\b{current\_topic}\b", word, re.IGNORECASE)

                    for word in re.split(r"\W+", text)

            ):

                if "олимпиад" in text.lower():

                    links\_with\_olimp.append(

                        f"https://vk.com/wall{post['owner\_id']}\_{post['id']}"

                    )

                else:

                    links\_without\_olimp.append(

                        f"https://vk.com/wall{post['owner\_id']}\_{post['id']}"

                    )

    return links\_with\_olimp, links\_without\_olimp