**Муниципальное автономное общеобразовательное учреждение**

**«Школа №17 города Белогорск»**

**Индивидуальный итоговый проект**

**«Создание виртуального помощника**

**по школьному расписанию**

**(чат-бота в Telegram)»**

**Выполнил:**

обучающийся 10А класса

Попов Максим Александрович

**Руководитель:**

учитель информатики

Ивченко Ирина Дмитриевна

г. Белогорск, 2024

Оглавление

[Введение. 3](#_Toc163471930)

[Актуальность: 3](#_Toc163471931)

[Гипотеза: 3](#_Toc163471932)

[Цель проекта: 3](#_Toc163471933)

[Задачи: 3](#_Toc163471934)

[Теоретические основы 4](#_Toc163471935)

[Что такое чат-бот? 4](#_Toc163471936)

[Преимущества использования чат-ботов: 4](#_Toc163471937)

[Виртуальных помощников можно создать с помощью: 5](#_Toc163471938)

[Выводы: 6](#_Toc163471939)

[Как все начиналось 6](#_Toc163471940)

[Разработка. 6](#_Toc163471941)

[1. Зависимости 6](#_Toc163471942)

[2. Изучение серверных запросов 7](#_Toc163471943)

[3. Пользовательский интерфейс 8](#_Toc163471944)

[Заключение 15](#_Toc163471945)

[Список используемых источников 15](#_Toc163471946)

[Приложения 17](#_Toc163471947)

# Введение.

## Актуальность:

После создания сайта-расписания в 9 классе, написание телеграмм бота было неминуемо. Я хотел сделать экосистему расписания, чтобы всегда иметь возможность родителям, и детям узнать его

После неудачного запуска сайта, понял, что людям будет удобнее получать информацию из мессенджеров, нежели через сайт.

## Гипотеза:

Если я создам Telegram бота с расписанием, это облегчит жизнь школьников, их родителей и сотрудников образовательной организации

## Цель проекта:

Разработка Телеграм бота для учащихся моей школы.

## Задачи:

Написать Чат-бот для Телеграма, подробно изучить API запросы к Google Sheets

# Теоретические основы

## Что такое чат-бот?

Чат-бот — это программа, предназначенная для автоматизации общения с людьми через текстовые или голосовые сообщения. В случае с Telegram, чат-боты могут использоваться для различных целей: от предоставления информации и консультаций до выполнения определенных задач, оформления заказов и многое другое. Они обычно работают по заранее заданным сценариям или ключевым словам, что позволяет им взаимодействовать с пользователями, отвечая на их вопросы или выполняя команды.

## Преимущества использования чат-ботов:

Круглосуточная доступность: Чат-боты могут работать 24/7, обслуживая пользователей в любое время.

Снижение затрат: Чат-боты могут выполнять задачи, которые раньше выполнялись людьми, что приводит к экономии на персонале.

Повышенная эффективность: Чат-боты могут обрабатывать несколько запросов одновременно, что повышает эффективность обслуживания клиентов.

Улучшение взаимодействия с пользователем: Чат-боты могут предоставлять персонализированное и удобное обслуживание, что улучшает взаимодействие с пользователем.

Сбор данных: Чат-боты могут собирать данные о взаимодействиях с пользователями, которые можно использовать для улучшения обслуживания и маркетинговых кампаний.

В целом, чат-боты используются для улучшения обслуживания клиентов, повышения эффективности и автоматизации задач, требующих взаимодействия с текстом или речью.

Мы живем во времена расцвета программистов и ПО, в моем распоряжении были разные статьи, видео на Ютубе и искусственный интеллект, хочу добавить, что я являюсь студентом в IT Академии TOP, там меня научили как писать чат боты и делать запросы к API различных сервисов.

API *(Application Programming Interface) - это набор правил и инструментов, который позволяет различным программам взаимодействовать друг с другом. API определяет способы, как различные компоненты программного обеспечения могут общаться друг с другом, обмениваться данными и выполнять определенные функции. Он предоставляет структурированный способ для приложений использовать функциональность других программ или сервисов без необходимости знать детали их реализации.*

По моему мнению, чтобы написать любой код на языке программирования Python нужно иметь: хотя бы 1 руку и интернет, к счастью, я счастливый обладатель 2х рук и интернета и написание робота для телеграма было не трудной задачей.

## Виртуальных помощников можно создать с помощью:

Платформ для создания чат-ботов (например, Dialogflow, IBM Watson Assistant)

Разработки с нуля с использованием фреймворков и библиотек NLP/ML

Готовых решений для голосовых помощников (например, Amazon Alexa Skills Kit, Google Assistant Actions SDK, Unisender)

Выбор способа зависит от сложности проекта, функций и ресурсов.

Руководствуясь желанием сотворить нечто уникальное и углубить свои познания в этой области, я смело избрал путь самостоятельной разработки виртуального помощника

Python *— это высокоуровневый интерпретируемый язык программирования, который отличается чистотой синтаксиса, простотой в изучении и широким спектром применения, включая веб-разработку, научные вычисления, автоматизацию задач и многое другое.*

*Python был разработан в конце 1980-х и начале 1990-х годов Гвидо ван Россумом в Нидерландах. Язык получил свое название в честь популярного шоу Monty Python's Flying Circus. Первая версия Python, 0.9.0, была выпущена в феврале 1991 года. Python был создан как простой, но мощный язык программирования, который был легким для изучения и использования. Он быстро приобрел популярность благодаря своей ясности и удобству, и с течением времени стал одним из самых популярных языков программирования в мире. Различные версии Python были выпущены с улучшениями и новыми возможностями, делая его востребованным как среди начинающих, так и опытных программистов.*

Конечно же просто зная пайтон написать бота с легкостью не получится. Кроме Python придется освоить несколько библиотек. В моем случае это “telebot”.

## Выводы:

Проанализировав все что у меня есть, а именно уже работающая на локальном сервере расписание на Google таблицах, логично было предположить, что бот будет брать данные из той же таблицы, а благодаря запросам к API это будет гораздо быстрее (среднее время отклика ~ 100 мс)

Я разбил работу на несколько частей:

* Применение теоретических основ
* Изучение возможных практических решений
* Разработка
* Распространение или интеграция

## Как все начиналось

Поставив перед собой задачу написать текстового бота, я сразу же открыл интегрированная среду разработки (IDE). К этому моменту у меня накопилось много набросков, которые я с радостью применил.

# Разработка.

## Зависимости

Как я сказал ранее, я уже имел навыки написания чат-ботов, поэтому повторно ознакомившись с наработками с занятий, я начал свою работу

Фрагменты кода для подробного изучения:

# Импорт необходимых библиотек  
import telebot as bot  
from telebot import types  
import requests  
import datetime  
# Импорт необходимых зависимостей  
import CheckDateSheets1  
import CheckDateSheets2  
import ALL  
import ALL2  
import Zamen  
import Day  
import settings

Код выше – это код, которые импортирует необходимые для него файлы и библиотеки

*В Python библиотеки представляют собой наборы модулей и функций, которые позволяют расширить возможности языка программирования. Они созданы для повторного использования кода и облегчения разработки программ. Библиотеки содержат реализации различных алгоритмов, структур данных, инструменты для работы с сетью, базами данных, графикой и другими задачами. Используя библиотеки, разработчики могут сократить время написания программ и улучшить их функциональность. В Python существует множество сторонних библиотек, которые можно устанавливать и подключать к своим проектам для решения конкретных задач.*

## Изучение серверных запросов

Самая первая и самая последняя трудность, с которой я столкнулся это подключение к таблицам. Простые запросы типа requests.get не помогали, перечитывать запутанные статьи на английском сайте с их официальной документации было достаточно долго. Тогда я нашел на YouTube’e маленький канал (TheLookin) с 17-ти минутным видео, где все было рассказано слишком просто, чтобы не разобраться.

### Пример API запроса на получение информации о дате

import os.path  
from google.auth.transport.requests import Request  
from google.oauth2.credentials import Credentials  
from google\_auth\_oauthlib.flow import InstalledAppFlow  
from googleapiclient.discovery import build  
from googleapiclient.errors import HttpError  
  
# If modifying these scopes, delete the file token.json.  
SCOPES = ["https://www.googleapis.com/auth/spreadsheets.readonly"]  
  
# The ID and settings.py of a sample spreadsheet.  
SAMPLE\_SPREADSHEET\_ID = "1O75BM0QxHI2l8n5rMewLm9LABKHeAUc1eKMeEtW7gEU"  
SAMPLE\_RANGE\_NAME = "Расписание замен 1 смены!A2:A2"  
  
mas = []  
count = 0  
def clear():  
 global mas, count  
 mas.clear()  
 count = 0  
def main():  
 global mas, count  
 """Shows basic usage of the Sheets API.  
 Prints values from a sample spreadsheet.  
 """  
 creds = None  
 # The file token.json stores the user's access and refresh tokens, and is  
 # created automatically when the authorization flow completes for the first  
 # time.  
 if os.path.exists("pas/token.json"):  
 creds = Credentials.from\_authorized\_user\_file("pas/token.json", SCOPES)  
 # If there are no (valid) credentials available, let the user log in.  
 if not creds or not creds.valid:  
 if creds and creds.expired and creds.refresh\_token:  
 creds.refresh(Request())  
 else:  
 flow = InstalledAppFlow.from\_client\_secrets\_file(  
 "pas/credentials.json", SCOPES  
 )  
 creds = flow.run\_local\_server(port=0)  
 # Save the credentials for the next run  
 with open("pas/token.json", "w") as token:  
 token.write(creds.to\_json())  
  
 try:  
 service = build("sheets", "v4", credentials=creds)  
  
 # Call the Sheets API  
 sheet = service.spreadsheets()  
 result = (  
 sheet.values()  
 .get(spreadsheetId=SAMPLE\_SPREADSHEET\_ID, range=SAMPLE\_RANGE\_NAME)  
 .execute()  
 )  
 values = result.get("values", [])  
  
 if not values:  
 print("No data found.")  
 mas.append('Ошибка даты')  
 return mas  
  
 for row in values:  
 try:  
 answer = (f"{row[0]}")  
 mas.append(answer)  
 except:  
 answer = (f'Ошибка в получении даты')  
 mas.append(answer)  
 print(mas)  
 return mas  
 except HttpError as err:  
 print(err)

## Пользовательский интерфейс

Гораздо удобнее разговаривать с ботом, если под рукой есть заготовленные фразы. И я ради удобства сделал их и себе получилось можно сказать приложение

def keys(info):  
 global IsAll, IsZamen  
 kb = types.ReplyKeyboardMarkup(row\_width=2)  
 but1 = types.KeyboardButton('Полное расписание')  
 but2 = types.KeyboardButton('Изменения')  
 but3 = types.KeyboardButton('Покажи кота😼')  
 kb.add(but1, but2, but3)  
 bot.send\_message(info.chat.id, 'Привет', reply\_markup = kb)

Код, которые вы только что посмотрели отвечает за появление главного меню

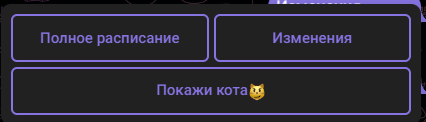


Рисунок 1. Главное меню

### Подробнее про каждый пункт меню

#### Полное расписание.

При нажатии на кнопку, они заменяются для выбора класса



Рисунок 2. Выбор классов

И вот, после получения номера класса, бот берет из файла диапазон ячеек, который ему нужен, делает запрос к таблице, та в свою очередь возвращает ему список, который после становится обычным текстом

elif 'расписание 5а' in msg.text or (IsAll == True and '5а' in msg.text):  
 SAMPLE\_RANGE\_NAME = settings.ПолноеРасписание5А  
 fullGen(msg, SAMPLE\_RANGE\_NAME)

def fullGen(msg, SAMPLE\_RANGE\_NAME):  
 ALL.clear()  
  
 res = ALL.main(SAMPLE\_RANGE\_NAME)  
 for i in range(len(res)):  
 ress = '\n'.join(res)  
 bot.send\_message(msg.chat.id, ress, parse\_mode='HTML')  
 keys(msg)

Это результат на примере 8Б класса



Рисунок 3. Полное расписание

#### Изменения.

Практически тоже самое, что и на полном расписании, но код сильно различается.

elif 'изменения 5а' in msg.text or (IsZamen == True and '5а' in msg.text):  
 liter = '5А'  
 zamenGen(msg, liter)

Получение информации о классе, и запуск функции

def zamenGen(msg, liter):

today = datetime.date.today() # получает текущую дату

weekday = today.weekday() # Достает день недели

weekday\_name = daysOfWeek[weekday] # Берет название дня недели

bot.send\_message(msg.chat.id, 'Смотрю везде, где только можно🔎')

if weekday\_name == 'Понедельник':

attribute\_name = 'Понедельник' + liter

SAMPLE\_RANGE\_NAME = getattr(settings, attribute\_name)

elif weekday\_name == 'Вторник':

attribute\_name = 'Вторник' + liter

SAMPLE\_RANGE\_NAME = getattr(settings, attribute\_name)

elif weekday\_name == 'Среда':

attribute\_name = 'Среда' + liter

SAMPLE\_RANGE\_NAME = getattr(settings, attribute\_name)

elif weekday\_name == 'Четверг':

attribute\_name = 'Четверг' + liter

SAMPLE\_RANGE\_NAME = getattr(settings, attribute\_name)

elif weekday\_name == 'Пятница':

attribute\_name = 'Пятница' + liter

SAMPLE\_RANGE\_NAME = getattr(settings, attribute\_name)

elif weekday\_name == 'Суббота':

attribute\_name = 'Понедельник' + liter

SAMPLE\_RANGE\_NAME = getattr(settings, attribute\_name)

elif weekday\_name == 'Воскресенье':

attribute\_name = 'Понедельник' + liter

SAMPLE\_RANGE\_NAME = getattr(settings, attribute\_name)

Day.clear()

pol = Day.main(SAMPLE\_RANGE\_NAME) # Получает полное расписание на этот день недели

for i in range(len(pol)):

ress = '\n'.join(pol)

bot.send\_message(msg.chat.id, 'Жду ответа⏳')

attribute\_name = 'Замены' + liter

SAMPLE\_RANGE\_NAME = getattr(settings, attribute\_name)

Zamen.clear()

zam = Zamen.main(SAMPLE\_RANGE\_NAME) # Получает измененное расписание на этот день недели

for i in range(len(zam)):

resss = '\n'.join(zam)

bot.send\_message(msg.chat.id, 'Ответ получен✅')

CheckDateSheets1.clear()

today = CheckDateSheets1.main() # Чтобы не запутаться берет дату, на которое сделано изменение

try:

for i in range(len(today)):

ressss = '\n'.join(today)

ressss = f"<b>{ressss}</b>"

except:

ressss = f"<i>Ошибка в поиске даты</i>" # Все мы люди и можем ошибаться и не написать дату, а это позволит избежать критической ошибки

Выше показал большую часть в открытом виде и для вашего удобства подписал комментарии.

if weekday\_name != 'Суббота' and weekday\_name != 'Воскресенье':  
 if weekday\_name != 'Четверг' and weekday\_name != 'Понедельник' and weekday\_name != 'Вторник':  
 tDay = weekday\_name[:-1] + 'у'  
 else:  
 tDay = weekday\_name  
 else:  
 tDay = 'Понедельник'

А вот эта часть кода склоняет дни недели. Заменяет последнюю букву на «у»

То, что ниже отправляет уже готовое сообщение пользователю

zamen = f'Вот твое обычное расписание на <b>{tDay}</b> \n{ress} \n\nПосмотри изменения ниже \n{ressss} \n\n{resss}'  
bot.send\_message(msg.chat.id, zamen, parse\_mode='HTML')

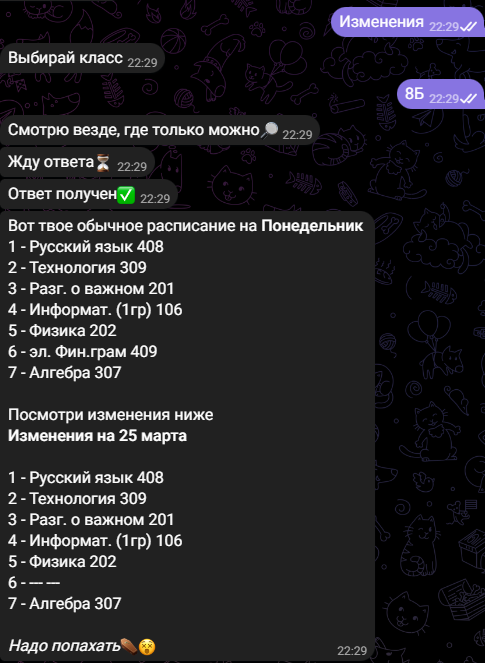
Вот что получит человек в ответ на свой запрос

Рисунок 4. Изменения в расписании

Покажи кота😼.

Забавная кнопка, добавил, потому что могу.

def catGen(msg):  
 url = "https://cat14.p.rapidapi.com/v1/images/search"  
  
 headers = {  
 "x-api-key": "live\_2NaPD4EWeLRfElyfgRyoAh4Ep9vDksAMo0fRDGItbPfahmyK",  
 "X-RapidAPI-Key": "UGHnXl5vvJ4ImjV49gEOERJN8EXmzLWfAiH5MurYSUi2Kn",  
 "X-RapidAPI-Host": "cat14.p.rapidapi.com"  
 }  
  
 response = requests.get(url, headers=headers)  
  
 catJs = response.json()  
 bot.send\_photo(msg.chat.id, catJs[0]['url'])

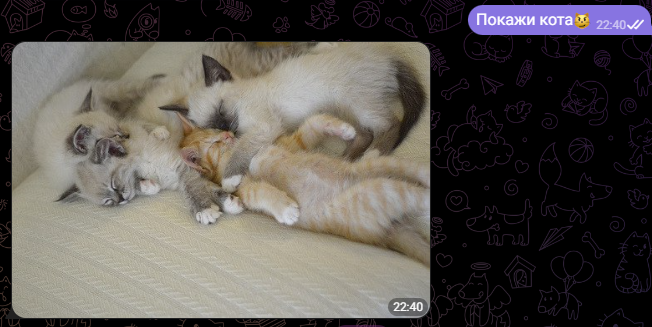
Бот обращается к апи, получает ответ в формате json, бот достает url картинки, и отправляет пользователю. Автор этого апи сам не знает какое количество фотографий котов у него есть.

*JSON (JavaScript Object Notation) — это текстовый формат обмена данными, основанный на JavaScript. Несмотря на происхождение, JSON является независимым от языка платформой и может использоваться практически с любым языком программирования. Основные особенности JSON:*

* *Легкость чтения и написания: для человека формат достаточно прост и понятен, а для машины — легко анализируем и генерируем.*
* *Структура: JSON представляет собой набор пар ключ/значение (похож на объекты в JavaScript)*
* *Использование: JSON широко используется для передачи данных в веб-приложениях между клиентом и сервером. Также его часто применяют для хранения конфигураций, настроек и других данных.*

Пример JSON объекта, который приходит после генерации кота:

[  
 {  
 "id": "a4s",  
 "url": "https://cdn2.thecatapi.com/images/a4s.jpg",  
 "width": 500,  
 "height": 331  
 }  
]

Рисунок . Покажи кота

#### Команда для администратора

Для очень важных уведомлений, я создал команду «Глобальное сообщение». Суть кроется в названии, отправляет всем пользователям сообщение, которое написал администратор.

def read\_unique\_ids(msg):  
 unique\_ids = set()  
 with open('bd.txt', 'r') as file:  
 for line in file:  
 # Удаление пробельных символов и переносов строки  
 id = line.strip()  
 # Добавление id в множество, если строка не пустая  
 if id:  
 unique\_ids.add(id)  
 # Преобразование множества в список для возможного дальнейшего использования  
 text = msg.text.replace('глобальное сообщение ', '<b>Глобальное сообщение\n</b>') #Замена "командного слова" нужным и красивым"  
 for one in unique\_ids:  
 global smscount  
 try:  
 bot.send\_message(one, text, parse\_mode='HTML')  
 smscount += 1  
 except:  
 pass  
 globalsms = f'Всего отправленно {smscount} людям'  
 bot.send\_message(824176864, globalsms)

### Настройки

Так как я планирую, что мой бот в школе еще долго будет существовать, структура таблиц может поменяться, и чтобы бот показывал все ровно и как надо, я сделал файл с настройками диапазонов, конечно же на 5 классе все не остановилось

# Дни недели смена сильных  
Понедельник1 = 2  
Вторник1 = 12  
Среда1 = 21  
Четверг1 = 30  
Пятница1 = 40  
  
# Дни недели смена слабых  
  
Понедельник2 = 2  
Вторник2 = 11  
Среда2 = 19  
Четверг2 = 27  
Пятница2 = 36  
  
  
# Настройка диапозонов таблицы  
  
# 5 А Класс  
ПолноеРасписание5А = 'Смена1!C3:D48'  
Понедельник5А = 'Смена1!C3:D11'  
Вторник5А = 'Смена1!C13:D20'  
Среда5А = 'Смена1!C22:D29'  
Четверг5А = 'Смена1!C31:D39'  
Пятница5А = 'Смена1C41:D48'  
Замены5А = 'Расписание замен 1 смены!C3:D10'

Когда боту нужно показать расписание, он берет из файла значение переменное и делает запрос к апи, исходя из заданных значений

elif 'расписание 5а' in msg.text or (IsAll == True and '5а' in msg.text):  
 SAMPLE\_RANGE\_NAME = settings.ПолноеРасписание5А  
 ALL.clear()  
  
 res = ALL.main(SAMPLE\_RANGE\_NAME)

# Заключение

По заключению проекта я сделал адаптивного Телеграм бота для школы, потратив для этого 3 дня.

Поставленная цель – создание Чат бота была выполнена

# Список используемых источников

<https://chat.openai.com/> - ИИ, выполняющий роль корректора и словаря.

<https://www.youtube.com/@thelookin8237> – Рассказал про Подключение к апи

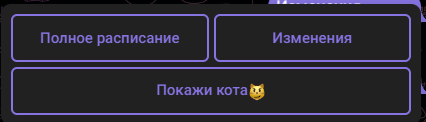
Приложения  


Рисунок 1. Главное меню



Рисунок 2. Выбор классов



Рисунок 3. Полное расписание

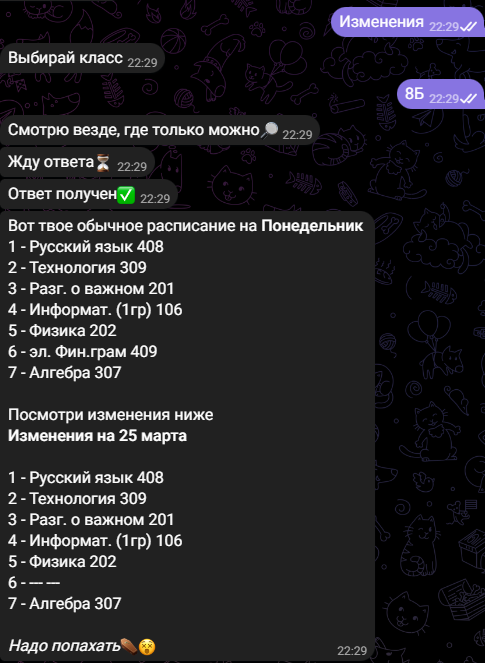
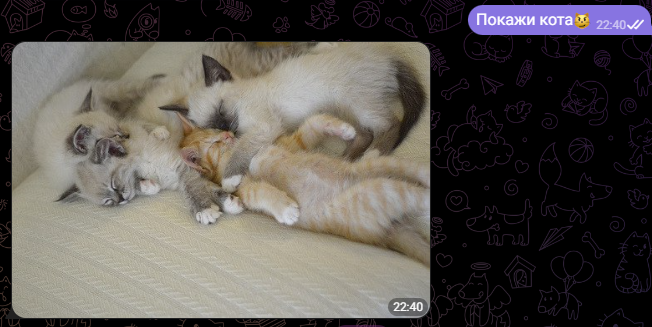


Рисунок 4. Изменения в расписании

Рисунок 5. Покажи кота