**1. Введение**

Тема проекта: Telegram-бот

Цель проекта - изучить принципы создания Telegram-бота с использованием языка Python.

**2. Выбор технологий**

|  |  |  |
| --- | --- | --- |
| Компонент | Технология | Описание |
| Язык программирования | Python | Современный язык с мощной экосистемой |
| Telegram API | Telegram.Bot | Популярная .NET-библиотека для работы с Bot API |
| IDE | PyCharm | Для разработки |
| Хранение данных | JSON-файлы | Выбирается в зависимости от проекта |
| Геолокация | Яндекс.карты | Бесплатные API для ссылок, интеграция без ключей |
| Документация | Markdown, Draw.io, PlantUML | Для технического описания и UML |

**3. Исследование: как работает Telegram-бот на C#**

**3.1 Как работает Telegram Bot API**

Telegram Bot API — это HTTP-интерфейс, через который можно получать и отправлять сообщения, обрабатывать команды и взаимодействовать с пользователями. Есть два основных способа получения обновлений:

• Long Polling — бот запрашивает новые сообщения с интервалом

• Webhook — Telegram сам отправляет обновления на указанный URL

Для простоты мы будем использовать Long Polling, так как он не требует публичного сервера на первом этапе.

**3.2 Основы библиотеки Telegram.Bot для C#**

Telegram.Bot (https://github.com/TelegramBots/Telegram.Bot) — официальная .NET-библиотека для Telegram Bot API. Возможности:

• Отправка и получение сообщений.

• Обработка команд.

• Поддержка клавиатур и inline-кнопок.

• Работа с медиа и API Telegram.

**4. Пошаговое создание Telegram-бота на C#**

**Шаг 1: Регистрация бота в Telegram**

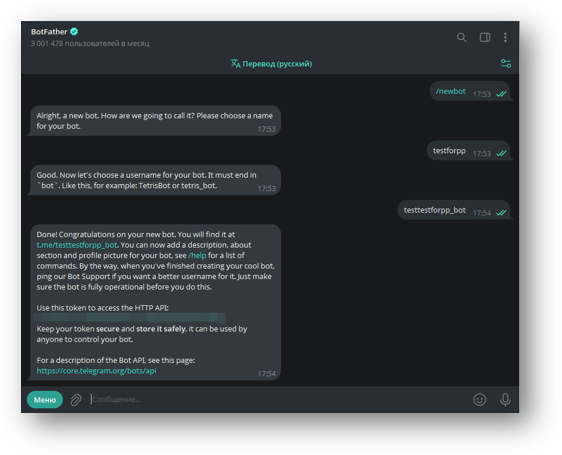
1. Откройте Telegram и найдите бота @BotFather (https://t.me/BotFather)

2. Отправьте команду /newbot

3. Укажите имя и username (например, MyTestBot)

4. Скопируйте токен, он выглядит так:

123456789:ABCDefGhIJKLmNoPQRsTUVwxyZ



**Шаг 2: Создание проекта**

1. Откройте Python
2. Создайте виртуальное окружение
3. Установите необходимые библиотеки:

pip install pyTelegramBotAPI requests python-dotenv

Создайте файл со структурой route.json со структурой:

[

{

"id": "center\_moscow",

"name": "По центру Москвы",

"stops": [

{

"name": "Красная площадь",

"short\_desc": "Главная площадь страны...",

"long\_history": "Подробное описание...",

"photo": "photos/red\_square.jpg",

"yandex\_map\_url": "https://yandex.ru/maps/..."

}

]

}

]

**Шаг 3: Базовая реализация бота**

Вставьте следующий код:

import telebot

from telebot.types import InlineKeyboardMarkup, InlineKeyboardButton, ReplyKeyboardMarkup, KeyboardButton

import json

TOKEN = '7624076910:AAFtnSHskqlY\_Aak\_mfMkKwSB76c2fmm62s'

bot = telebot.TeleBot(TOKEN)

# Загружаем все маршруты

with open("route.json", encoding="utf-8") as f:

ROUTES = json.load(f)

# Словари для хранения состояния

user\_route = {} # user\_id -> route\_id

user\_step = {} # user\_id -> stop\_index

# Получить маршрут по ID

def get\_route(route\_id):

for route in ROUTES:

if route["id"] == route\_id:

return route

return None

@bot.message\_handler(commands=['start'])

def main(message):

markup = InlineKeyboardMarkup()

markup.row(InlineKeyboardButton("Конечно", callback\_data="start\_walk"))

markup.row(

InlineKeyboardButton("Да", callback\_data="start\_walk"),

InlineKeyboardButton("Само собой", callback\_data="start\_walk")

)

bot.send\_message(

message.chat.id,

"""🗺️ Привет!

Я — ваш персональный гид по Москве. Готов провести вас по увлекательным маршрутам, рассказать о тайнах старинных улиц, памятниках и скрытых жемчужинах столицы.

📍 Для начала выберите район Москвы, где вы находитесь. Я предложу маршруты и помогу пройти их шаг за шагом.

🚶‍♀️ Готовы начать прогулку?""",

reply\_markup=markup

)

@bot.callback\_query\_handler(func=lambda c: c.data == "start\_walk")

def choose\_route(call):

bot.answer\_callback\_query(call.id)

kb = ReplyKeyboardMarkup(resize\_keyboard=True)

kb.row(

KeyboardButton("По центру Москвы"),

KeyboardButton("10 главных достопримечательностей Москвы")

)

kb.row(

KeyboardButton("От Кремля до Арбата"),

KeyboardButton("ВДНХ: от арки до космоса")

)

bot.send\_message(call.message.chat.id, "Выберите маршрут:", reply\_markup=kb)

@bot.message\_handler(func=lambda m: m.text in ["По центру Москвы", "10 главных достопримечательностей Москвы","От Кремля до Арбата","ВДНХ: от арки до космоса"])

def start\_selected\_route(message):

route\_map = {

"По центру Москвы": "center\_moscow",

"10 главных достопримечательностей Москвы": "top\_10\_moscow",

"От Кремля до Арбата": "kremlin\_to\_arbat",

"ВДНХ: от арки до космоса": "vdnh\_tour"

}

route\_id = route\_map[message.text]

user\_route[message.chat.id] = route\_id

user\_step[message.chat.id] = 0

send\_stop(message.chat.id)

def send\_stop(chat\_id):

route\_id = user\_route.get(chat\_id)

step = user\_step.get(chat\_id, 0)

route = get\_route(route\_id)

if not route or step >= len(route["stops"]):

ikb = InlineKeyboardMarkup()

ikb.add(InlineKeyboardButton("🔁 Начать заново", callback\_data="start\_walk"))

bot.send\_message(chat\_id, "🎉 Экскурсия завершена! Хотите пройти другую?", reply\_markup=ikb)

return

stop = route["stops"][step]

with open(stop["photo"], "rb") as img:

bot.send\_photo(

chat\_id,

img,

caption=f"📍 \*{stop['name']}\*\n\n{stop['short\_desc']}",

parse\_mode='Markdown'

)

ikb = InlineKeyboardMarkup()

ikb.add(

InlineKeyboardButton("📜 Подробнее", callback\_data=f"more\_{step}"),

InlineKeyboardButton("➡️ Далее", callback\_data=f"next\_{step}")

)

ikb.add(InlineKeyboardButton("🗺 Открыть в Яндекс.Картах", url=stop["yandex\_map\_url"]))

bot.send\_message(chat\_id, "Выберите действие:", reply\_markup=ikb)

@bot.callback\_query\_handler(func=lambda c: c.data.startswith("more\_"))

def show\_more(call):

step = int(call.data.split("\_")[1])

route\_id = user\_route.get(call.message.chat.id)

route = get\_route(route\_id)

history = route["stops"][step]["long\_history"]

parts = [history[i:i + 4000] for i in range(0, len(history), 4000)]

for part in parts:

bot.send\_message(call.message.chat.id, part)

ikb = InlineKeyboardMarkup()

ikb.add(InlineKeyboardButton("➡️ Далее", callback\_data=f"next\_{step}"))

bot.send\_message(call.message.chat.id, "Готовы продолжить?", reply\_markup=ikb)

bot.answer\_callback\_query(call.id)

@bot.callback\_query\_handler(func=lambda c: c.data.startswith("next\_"))

def next\_step(call):

step = int(call.data.split("\_")[1]) + 1

user\_step[call.message.chat.id] = step

bot.answer\_callback\_query(call.id)

send\_stop(call.message.chat.id)

bot.infinity\_polling()

**Шаг 4: Запуск**

1. Вставьте свой токен в строку:

var botClient = new TelegramBotClient("YOUR\_TOKEN\_HERE");

2. Напишите боту в Telegram команду /start

**Итог:**

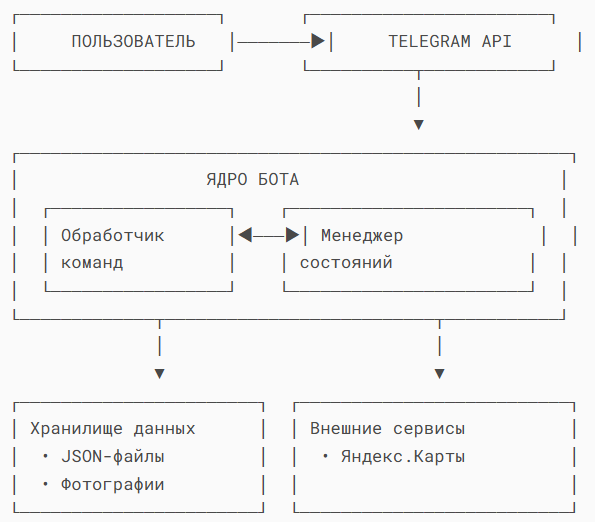
На этом этапе у нас есть базовый Telegram-бот, который:

• Обрабатывает команды /start, /help, /info

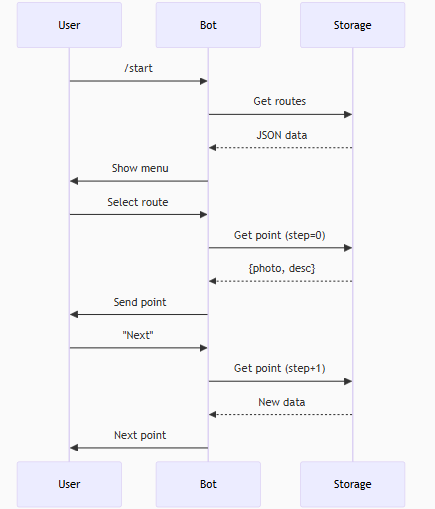
• Отвечает на любые текстовые сообщения

**5. Иллюстрации и диаграммы**

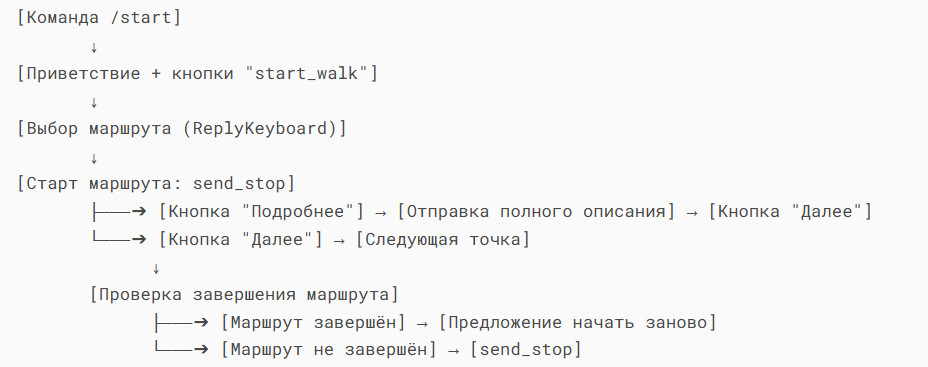
**5.1. UML-диаграмма компонентов**



**5.2. Диаграмма последовательности обработки сообщений**



**5.3. Схема логики обработки команд**



**6. Модификация проекта: Ведение дневника**

**Цель модификации**

Добавить в Telegram-бота голосовые подсказки и аудиогид, так как только тестовый формат может быть неудобен в движении.

**Реализация:**

* Использован gTTS (Google Text-to-Speech) для генерации аудио.
* Кнопка "🔊 Прослушать" в интерфейсе точки маршрута.

**Пример кода**

from gtts import gTTS

def generate\_voice(text, filename="temp.mp3"):

tts = gTTS(text=text, lang='ru')

tts.save(filename)

return filename

@bot.callback\_query\_handler(func=lambda c: c.data.startswith("audio\_"))

def send\_audio\_desc(call):

step = int(call.data.split("\_")[1])

route = get\_route(user\_route[call.message.chat.id])

text = route["stops"][step]["short\_desc"]

audio\_file = generate\_voice(text)

bot.send\_audio(call.message.chat.id, open(audio\_file, 'rb'))

**7 и 8 пункты по подключению кода к гиту и записыванию видео**

****