Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №3
«Разработка простого бота для Telegram с использованием языка Python»

Выполнил:
студент группы ИУ5-33Б
Смирнов Артём

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.

Москва, 2023 г.

Задание:

Разработайте простого бота для Telegram. Бот должен использовать функциональность создания кнопок

Текст Программы:

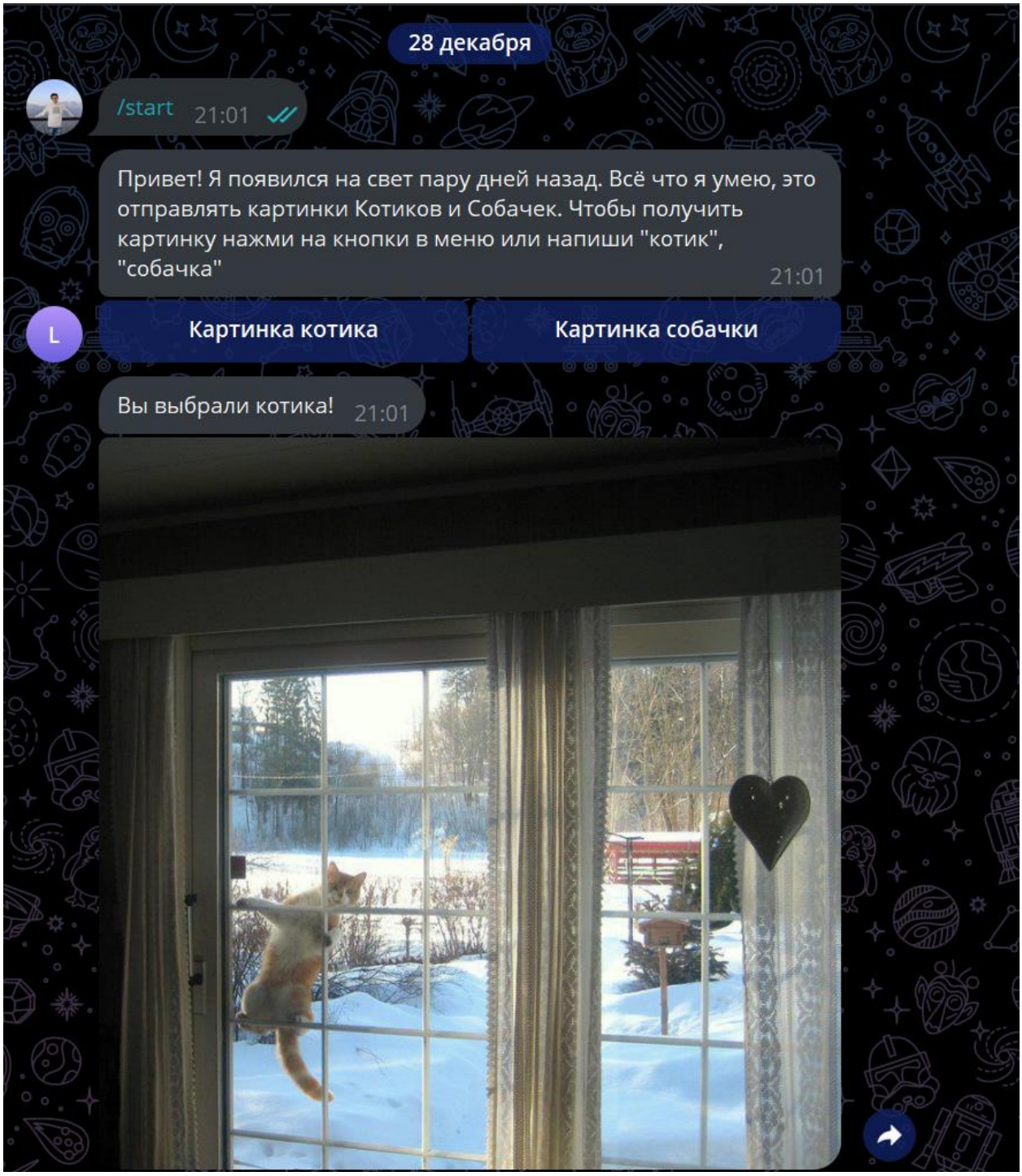
```
from typing import Final

import requests

from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup, CallbackQuery

from telegram.ext import Application, CommandHandler, MessageHandler, filters, ContextTypes, CallbackQueryHandler, CallbackContext



bot_token: Final = ""

bot_name: Final = ""



#Commands

async def start_command(update: Update, context: ContextTypes.DEFAULT_TYPE):

    response = '''Привет! Я появился на свет пару дней назад. Всё что я умею, это отправлять картинки Котиков и Собачек. Чтобы получить картинку нажми на кнопки в меню или напиши "котик", "собачка" '''

    keyboard = [

                InlineKeyboardButton("Картинка котика", callback_data="cat"),

                InlineKeyboardButton("Картинка собачки", callback_data="dog"),

            ]

    reply_markup = InlineKeyboardMarkup([keyboard])

    await update.message.reply_text(response, reply_markup=reply_markup)



async def help_command(update: Update, context: ContextTypes.DEFAULT_TYPE):

    await update.message.reply_text("Привет! Чем могу помочь ?")



async def custom_command(update: Update, context: ContextTypes.DEFAULT_TYPE):

    await update.message.reply_text("Привет! Чем могу помочь ?")



#Callbacks
```

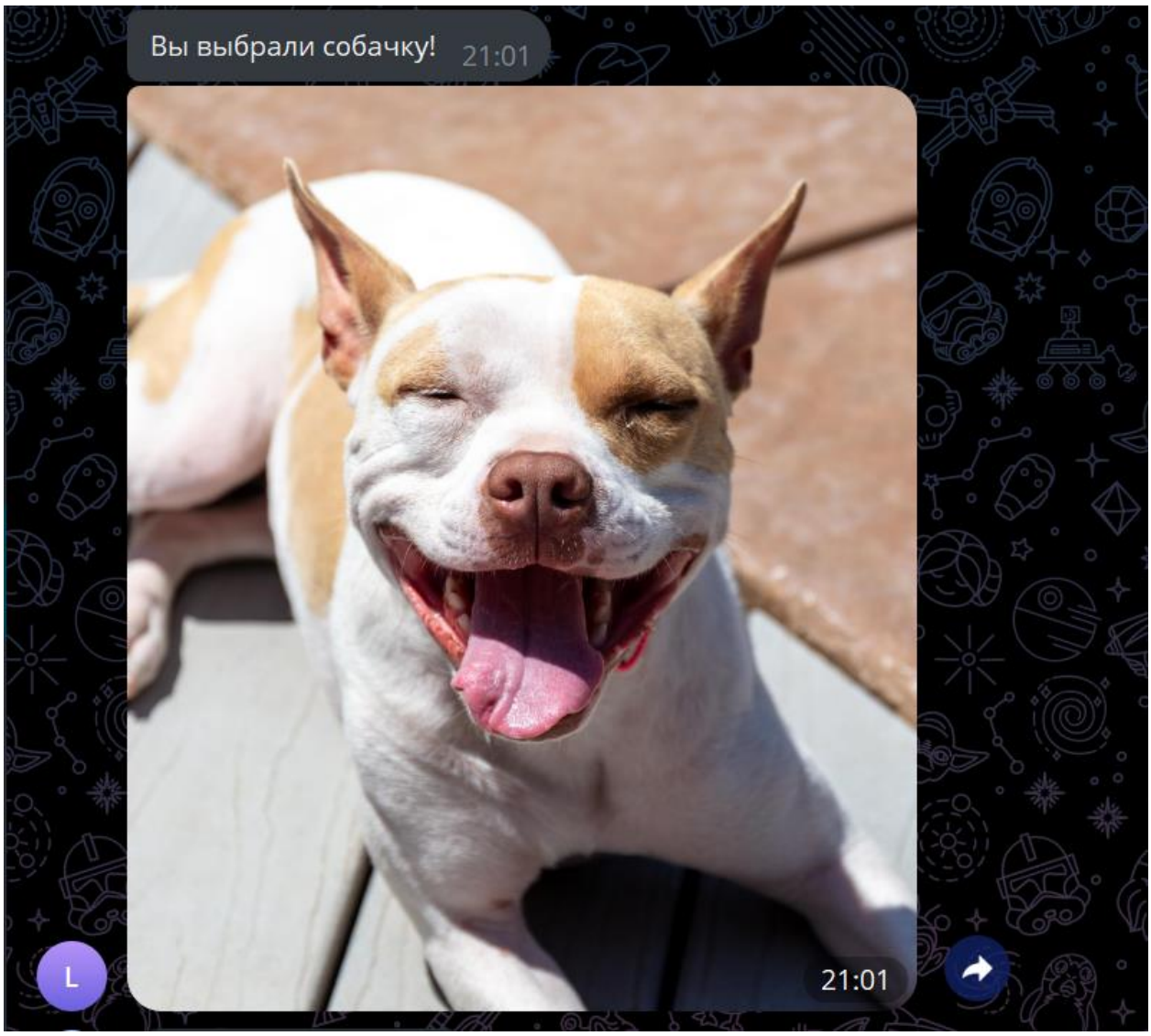
```
async def handle_callback(update: Update, context):
    query = update.callback_query
    button_pressed = query.data
    if button_pressed == "cat":
        await query.answer()  # Acknowledge the button press
        await update.effective_message.reply_text("Вы выбрали котика!")
        data = requests.get("https://api.thecatapi.com/v1/images/search")
        cat_pic = data.json()[0]["url"]
        await update.effective_message.reply_photo(cat_pic)
    elif button_pressed == "dog":
        await query.answer()  # Acknowledge the button press
        await update.effective_message.reply_text("Вы выбрали собачку!")
        data = requests.get("https://api.thedogapi.com/v1/images/search")
        dog_pic = data.json()[0]["url"]
        await update.effective_message.reply_photo(dog_pic)



#Responses
def handle_response(text: str) :
    processed: str = text.lower()
    if 'котик' in processed :
        return 'Вы выбрали котика!'
    elif 'собачк' in processed:
        return 'Вы выбрали собачку!'
    else:
        return 'Я вас не понял. Попробуйте ввести запрос еще раз или выберите вариант из меню'



#messages
async def handle_message(update: Update, context: ContextTypes.DEFAULT_TYPE):
    message_type: str = update.message.chat.type
    text: str = update.message.text


    print(f'User ({update.message.chat.id}) in {message_type}: "{text}"')


    if message_type == 'group':
```

```
        if bot_name in text:
            new_text: str = text.replace(bot_name, '').strip()
            response: str = handle_response(new_text)
        else:
            return
    else:
        response: str = handle_response(text)
    if response == 'Вы выбрали котика!':
        await update.effective_message.reply_text("Вы выбрали котика!")
        data = requests.get("https://api.thecatapi.com/v1/images/search")
        cat_pic = data.json()[0]["url"]
        await update.effective_message.reply_photo(cat_pic)
    elif response == 'Вы выбрали собачку!':
        await update.effective_message.reply_text("Вы выбрали собачку!")
        data = requests.get("https://api.thedogapi.com/v1/images/search")
        dog_pic = data.json()[0]["url"]
        await update.effective_message.reply_photo(dog_pic)
    else:
        print('Bot:', response)
        await update.message.reply_text(response)


#errors
async def error(update: Update, context: ContextTypes.DEFAULT_TYPE):
    print(f'Update {update} caused error {context.error}')


if __name__ == '__main__':
    print("Starting bot ...")
    app = Application.builder().token(bot_token).build()

    #Commands
    app.add_handler(CommandHandler('start',start_command))
    app.add_handler(CommandHandler('help',help_command))
    app.add_handler(CommandHandler('custom',custom_command))
    app.add_handler(CallbackQueryHandler(handle_callback))
    #Messages
```

```
app.add_handler(MessageHandler(filters.TEXT, handle_message))

#Error
app.add_error_handler(error)

#Polls the bot
print("Poling ...")
app.run_polling(poll_interval=0.1)
```

Результат выполнения:




Вы выбрали собачку! 21:01
21:01
L

Хочу котика 21:01

Вы выбрали котика! 21:01

L

Отправь собачку 21:01

Вы выбрали собачку! 21:01

L