Московский государственный технический университет им. Н.Э. Баумана

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

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №6 «Разработка бота на основе конечного автомата для Telegram с использованием языка Python»

Выполнил:
студент группы ИУ5-34Б

Угрюмов Михаил

Проверил:
преподаватель каф. ИУ5

Гапанюк Ю. Е.

2022 г.

Задание:

Разработайте бота для Telegram. Бот должен реализовывать конечный автомат из трех состояний.

Текст программы.

config.py

```
# токен бота
TOKEN = 'тут находится токен бота'

# путь к файлу с данными о состояниях пользователей (при первом запуске бота в
файле записано одно число: 0 - количество пользователей
PATH_USERS = 'database\\users.txt'

# возраст пользователя по умолчанию (нужен для поиска собеседников)
DEFAULT_USER_AGE = 14

# типы контента, которыми могут обмениваться пользователи в диалоге
CONTENT_TYPES = ['text', 'audio', 'document', 'photo', 'sticker', 'video',
'video_note', 'voice', 'location', 'contact', 'animation']

# эхо-режим для отладки (меняется только Application.find_companion → условие в
цикле for и условие перед return)
# MODE = 'SELF_TO_SELF'

# обычный режим
MODE = 'WORK'
```

User.py

```
from config import *

# класс для хранения информации об одном пользователе
class User:
    def __init__(self):
        self.sex = 'n'                  # пол
        self.want = 'n'                 # предпочтения поиска (по полу)
        self.age = DEFAULT_USER_AGE     # возраст
        self.state = 0                  # состояние использования бота
        self.companion = 0              # id собеседника
```

Database_worker.py

```
from User import *

# класс для хранения состояний пользователей
class Database_worker:
    def __init__(self, path_users):
        self.path_users = path_users

    # чтение из файла
    def get_users(self):
        users = {}
        with open(self.path_users) as file:
            n = int(next(file))
            for i in range(n):
                line = file.readline()
                user = User()
                user_id, user.sex, user.want, user.age, user.state,
user.companion = map(str, line.split())
                user.age = int(user.age)
                user.state = int(user.state)
                user.companion = int(user.companion)
                users[int(user_id)] = user
        return users
```

```
    # запись в файл
    def write_users(self, users):
        with open(self.path_users, 'w') as file:
            file.write(str(len(users)) + '\n')
            for user_id, user in users.items():
                file.write('{} {} {} {} {} {}\n'.format(user_id, user.sex,
user.want, user.age, user.state, user.companion))
```

Application.py

```
import telebot
from telebot import types
from Database_worker import *

# класс для работы бота
class Application:
    def __init__(self, token):
        self.bot = telebot.TeleBot(token)                    # сам бот
        self.database_worker = Database_worker(PATH_USERS)   # ввод-вывод
состояний пользователей в файл
        self.users = self.database_worker.get_users()        # словарь
пользователей { user_id : User }

    # выбора пола пользователя
    def choose_sex(self, user_id, message):
        correct = True
        if message.text == '🙍‍♂️ Мужской':
            self.users[user_id].sex = 'm'
        elif message.text == '🙍‍♀️ Женский':
            self.users[user_id].sex = 'w'
        elif message.text == 'Неважно':
            self.users[user_id].sex = 'n'
        else:
            correct = False
        self.database_worker.write_users(self.users)
        return correct

    # выбор возраста пользователя
    def choose_age(self, user_id, message):
        correct = True
        if message.text == 'Неважно':
            self.users[user_id].age = 0
        else:
            try:
                self.users[user_id].age = int(message.text)
            except:
                correct = False
        self.database_worker.write_users(self.users)
        return correct

    # выбор предпочтений поиска пользователя
    def choose_want(self, user_id, message):
        correct = True
        if message.text == '🙍‍♂️ Мужского':
            self.users[user_id].want = 'm'
        elif message.text == '🙍‍♀️ Женского':
            self.users[user_id].want = 'w'
        elif message.text == 'Не имеет значения':
            self.users[user_id].want = 'n'
        else:
            correct = False
        self.database_worker.write_users(self.users)
        return correct

    # старт либо изменение настроек пользователя
```

```
    def to_state_1(self, user_id, text):
        self.users[user_id].state = 1
        self.database_worker.write_users(self.users)
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
        btn1 = types.InlineKeyboardButton('👨 Мужской')
        btn2 = types.InlineKeyboardButton('👩 Женский')
        btn3 = types.InlineKeyboardButton('Неважно')
        markup.add(btn1, btn2, btn3)
        self.bot.send_message(user_id, text, reply_markup=markup)

    # изменение возраста
    def to_state_2(self, user_id, text):
        self.users[user_id].state = 2
        self.database_worker.write_users(self.users)
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
        btn = types.InlineKeyboardButton('Неважно')
        back = types.InlineKeyboardButton('Назад')
        markup.add(btn, back)
        self.bot.send_message(user_id, text, reply_markup=markup)

    # изменение предпочтений поиска
    def to_state_3(self, user_id, text):
        self.users[user_id].state = 3
        self.database_worker.write_users(self.users)
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
        btn1 = types.InlineKeyboardButton('👨 Мужского')
        btn2 = types.InlineKeyboardButton('👩 Женского')
        btn3 = types.InlineKeyboardButton('Не имеет значения')
        back = types.InlineKeyboardButton('Назад')
        markup.add(btn1, btn2, btn3, back)
        self.bot.send_message(user_id, text, reply_markup=markup)

    # готовность поиска собеседников
    def to_state_4(self, user_id, text, **kwargs):
        # нужно ли создавать кнопку 'Назад'
        add_back = True
        if len(kwargs) != 0:
            add_back = kwargs['back']

        self.users[user_id].state = 4
        self.database_worker.write_users(self.users)
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
        btn = types.InlineKeyboardButton('🔎 Поиск собеседника')
        if add_back:
            back = types.InlineKeyboardButton('Назад')
            markup.add(btn, back)
        else:
            markup.add(btn)
        self.bot.send_message(user_id, text, reply_markup=markup)

    # переход в состояние поиска собеседника
    def to_state_5(self, user_id, text):
        self.users[user_id].state = 5
        self.database_worker.write_users(self.users)
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
        btn = types.InlineKeyboardButton('Остановить поиск')
        markup.add(btn)
        self.bot.send_message(user_id, text, reply_markup=markup)

    # переход в состояния диалога
    def to_state_6(self, user_id, text):
        self.users[user_id].state = 6
        self.database_worker.write_users(self.users)
        markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
        btn1 = types.InlineKeyboardButton('Завершить диалог')
```

```
        btn2 = types.InlineKeyboardButton('Следующий собеседник')
        markup.add(btn1, btn2)
        self.bot.send_message(user_id, text, reply_markup=markup)

    # поиск индекс пользователя из списка prior, у которого |age - prior_age| -
минимально
    @staticmethod
    def find_best_age_in_list(prior, prior_age):
        answer = 0
        prev = abs(prior[0][1].age - prior_age)
        for i in range(1, len(prior)):
            curr = abs(prior[i][1].age - prior_age)
            if curr < prev:
                answer = i
                prev = curr
        return answer

    # поиск собеседника
    def find_companion(self, user_id):
        prior_sex = self.users[user_id].want
        prior_want = self.users[user_id].sex
        prior_age = self.users[user_id].age
        prior_1 = []
        prior_2 = []
        prior_3 = []
        for id, user in self.users.items():
            if MODE == 'WORK' and user.state == 5 and user.companion == 0 and
user_id != id or MODE == 'SELF_TO_SELF' and user.state == 5 and user.companion
== 0:
                if user.sex == prior_sex and user.want == prior_want:
                    prior_1.append((id, user))
                elif (user.sex != prior_sex or user.want != prior_want) and not
(
                        user.sex != prior_sex and user.want != prior_want):
                    prior_2.append((id, user))
                else:
                    prior_3.append((id, user))
        if len(prior_1) > 0:
            id = prior_1[self.find_best_age_in_list(prior_1, prior_age)][0]
        elif len(prior_2) > 0:
            id = prior_2[self.find_best_age_in_list(prior_2, prior_age)][0]
        elif len(prior_3) > 0:
            id = prior_3[self.find_best_age_in_list(prior_3, prior_age)][0]
        else:
            return False
        self.users[user_id].companion = id
        if MODE == 'WORK':
            self.users[user_id].n = 1
            self.users[id].companion = user_id
            self.to_state_6(id, 'Собеседник найден!')
        self.database_worker.write_users(self.users)
        return True

    # выполнение поиска собеседника
    def process_state_5(self, user_id):
        self.to_state_5(user_id, '🔎 Ищу тебе собеседника...')
        found = self.find_companion(user_id)
        if found:
            self.to_state_6(user_id, 'Собеседник найден!')

    # завершение диалога
    def stop_command(self, user_id, text_1, text_2):
        companion_id = self.users[user_id].companion
        self.users[user_id].companion = 0
        self.users[companion_id].companion = 0
        self.to_state_4(user_id, text_1, back=False)
```

```
        self.to_state_4(companion_id, text_2, back=False)
        self.database_worker.write_users(self.users)

    # следующий собеседник
    def next_command(self, user_id):
        self.stop_command(user_id, 'Диалог завершён.', 'Собеседник завершил
диалог.')
        self.process_state_5(user_id)

    # отправка сообщений между собеседниками
    def send_content(self, message):
        user_id = message.from_user.id

        reply_id = 0
        if message.reply_to_message:
            reply_id = message.reply_to_message.message_id

        if message.content_type not in ['text', 'photo', 'location', 'contact']:
            eval('self.bot.send_' + message.content_type +
'(self.users[user_id].companion, message.' + message.content_type + '.file_id,
reply_to_message_id=reply_id)')
        elif message.content_type == 'text' and message.text not in ['Следующий
собеседник', 'Завершить диалог']:
            self.bot.send_message(self.users[user_id].companion, message.text,
reply_to_message_id=reply_id)
        elif message.content_type == 'photo':
            self.bot.send_photo(self.users[user_id].companion,
message.photo[len(message.photo) - 1].file_id, reply_to_message_id=reply_id)
        elif message.content_type == 'location':
            self.bot.send_location(self.users[user_id].companion,
message.location.latitude, message.location.longitude,
reply_to_message_id=reply_id)
        elif message.content_type == 'contact':
            self.bot.send_contact(self.users[user_id].companion,
message.contact.phone_number, message.contact.first_name,
message.contact.last_name, reply_to_message_id=reply_id)
```

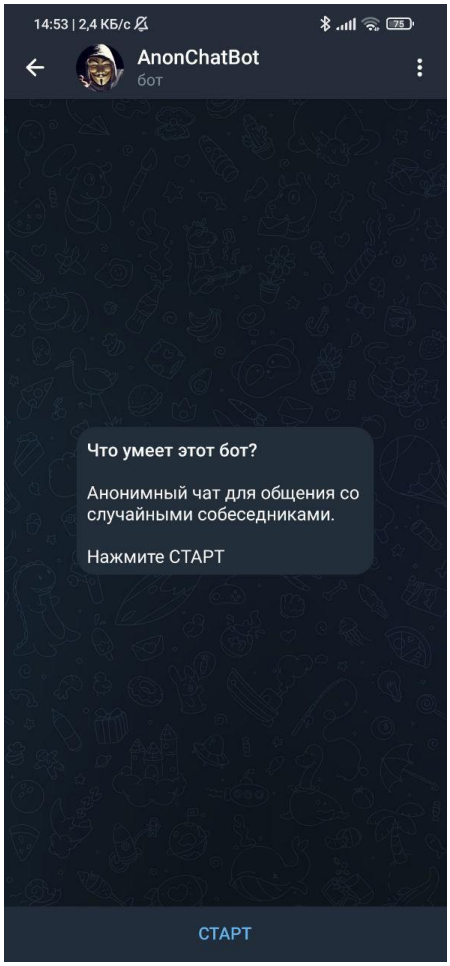
main.py

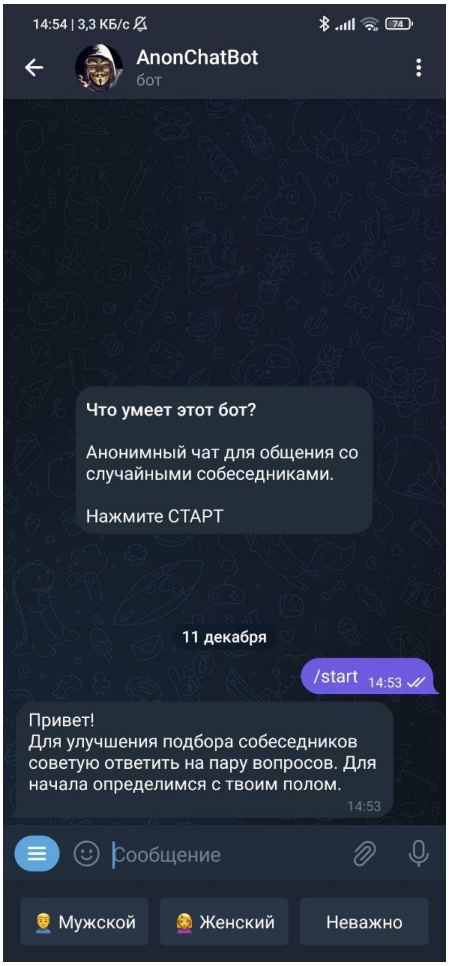
```
from Application import *

# создание приложения ( бот + работа с файлом + словарь { user_id : User } )
app = Application(TOKEN)

# старт
@app.bot.message_handler(commands=['start'])
def start(message):
    user_id = message.from_user.id
    app.users[user_id] = User()
    app.to_state_1(user_id, 'Привет!\nДля улучшения подбора собеседников советую
ответить на пару вопросов. Для начала определимся с твоим полом.')

# поиск собеседника
@app.bot.message_handler(commands=['search'])
def search(message):
    user_id = message.from_user.id
    if app.users[user_id].companion != 0:
        app.to_state_5(user_id, 'В данный момент пообщаться не с кем :(')
    else:
        app.bot.send_message(user_id, 'У тебя уже есть собеседник.')

# завершение диалога
@app.bot.message_handler(commands=['stop'])
def stop(message):
    user_id = message.from_user.id
    if app.users[user_id].companion != 0:
        app.stop_command(user_id, 'Диалог завершён.', 'Собеседник завершил
```

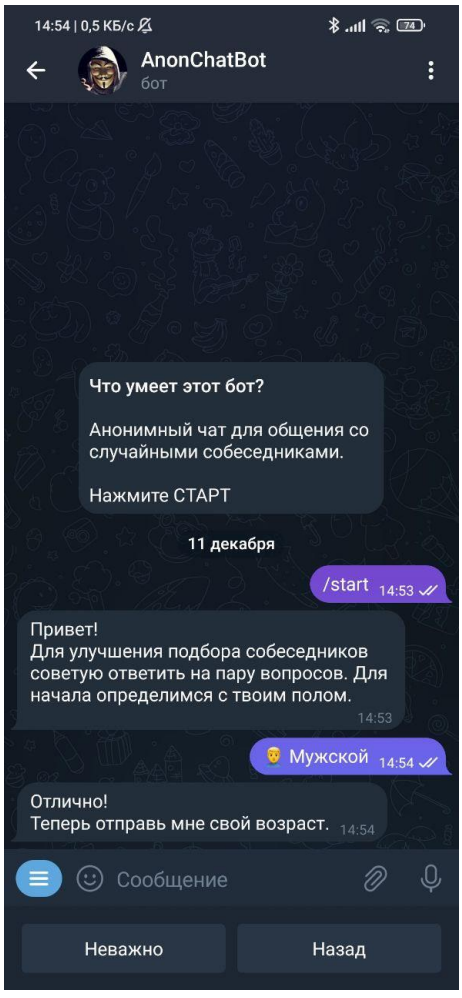
```
диалог.')
    else:
        app.bot.send_message(user_id, 'У тебя ещё нет собеседника.\nНажми
/search.')

# следующий собеседник
@app.bot.message_handler(commands=['next'])
def next(message):
    user_id = message.from_user.id
    if app.users[user_id].companion != 0:
        app.next_command(user_id)
    else:
        app.bot.send_message(user_id, 'У тебя ещё нет собеседника.\nНажми
/search.')

# отправить собеседнику ссылку на свой аккаунт
@app.bot.message_handler(commands=['link'])
def next(message):
    user_id = message.from_user.id
    if app.users[user_id].companion != 0:
        app.bot.send_message(user_id, 'Ссылка на мой
аккаунт:\nhttps://t.me/{}'.format(message.from_user.username))
    else:
        app.bot.send_message(user_id, 'У тебя ещё нет собеседника.\nНажми
/search.')

# изменение настроек поиска
@app.bot.message_handler(commands=['reset'])
def reset(message):
    user_id = message.from_user.id
    if app.users[user_id].companion == 0:
        app.to_state_1(user_id, 'Выбери свой пол.')
    else:
        app.bot.send_message(user_id, 'Опция недоступна во время диалога.')

# помощь по боту
@app.bot.message_handler(commands=['help'])
def help(message):
    user_id = message.from_user.id
    app.bot.send_message(user_id, 'Здесь будет информация о боте.')

# правила общения в чате
@app.bot.message_handler(commands=['rules'])
def rules(message):
    user_id = message.from_user.id
    app.bot.send_message(user_id, 'Здесь будут правила общения в чате.')

# взаимодействие с пользователем в зависимости от его состояния (state)
@app.bot.message_handler(content_types=CONTENT_TYPES)
def text_processing(message):
    user_id = message.from_user.id

    # state 1 - выбор пола
    if app.users[user_id].state == 1:
        correct = app.choose_sex(user_id, message)
        if correct:
            app.to_state_2(user_id, 'Отлично!\nТеперь отправь мне свой
возраст.')
        else:
            app.bot.send_message(user_id, 'Просто нажми на одну из кнопок.')

    # state 2 - выбор возраста
    elif app.users[user_id].state == 2:
        correct = app.choose_age(user_id, message)
        if message.text == 'Назад':
            app.to_state_1(user_id, 'Выбери свой пол.')
```

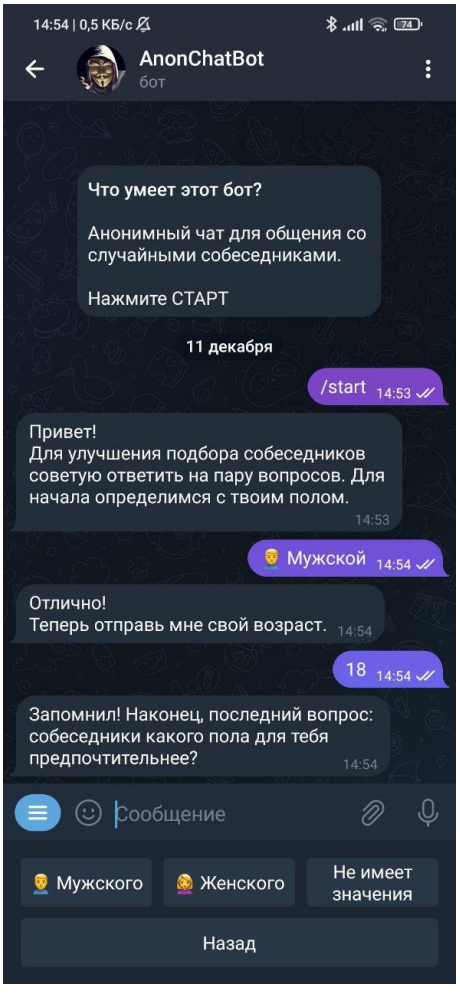
```
        elif correct:
            app.to_state_3(user_id, 'Запомнил! Наконец, последний вопрос:
собеседники какого пола для тебя предпочтительнее?')
        else:
            app.bot.send_message(user_id, 'Введи одно число, либо нажми
"Неважно".')

    # state 3 - выбор предпочтений поиска
    elif app.users[user_id].state == 3:
        correct = app.choose_want(user_id, message)
        if message.text == 'Назад':
            app.to_state_2(user_id, 'Отправь свой возраст.')
        elif correct:
            app.to_state_4(user_id, 'Готово! Теперь ты можешь общаться, для
этого нажми\n🔎 Поиск собеседника".')
        else:
            app.bot.send_message(user_id, 'Просто нажми на одну из кнопок.')

    # state 4 - готовность искать собеседника
    elif app.users[user_id].state == 4:
        if message.text == '🔎 Поиск собеседника':
            app.process_state_5(user_id)
        elif message.text == 'Назад':
            app.to_state_3(user_id, 'Собеседники какого пола для тебя
предпочтительнее?')
        else:
            app.bot.send_message(user_id, 'Нажми "🔎 Поиск собеседника".')

    # state 5 - поиск собеседника
    elif app.users[user_id].state == 5:
        if message.text == 'Остановить поиск':
            app.to_state_4(user_id, 'Вы завершили поиск.', back=False)
        else:
            app.bot.send_message(user_id, 'Собеседник ещё не найден.')

    # state 6 - диалог
    elif app.users[user_id].state == 6 and app.users[user_id].companion != 0:
        if message.content_type == 'text':
            if message.text == 'Завершить диалог':
                app.stop_command(user_id, 'Диалог завершён.', 'Собеседник
завершил диалог.')
            elif message.text == 'Следующий собеседник':
                app.next_command(user_id)
        app.send_content(message)

if __name__ == '__main__':
    app.bot.polling(none_stop=True, interval=0)
```

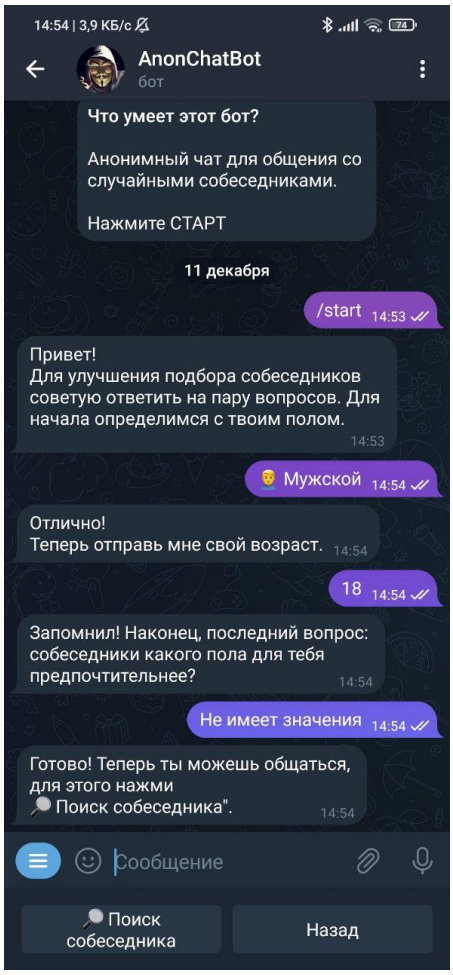
Результаты выполнения программы:

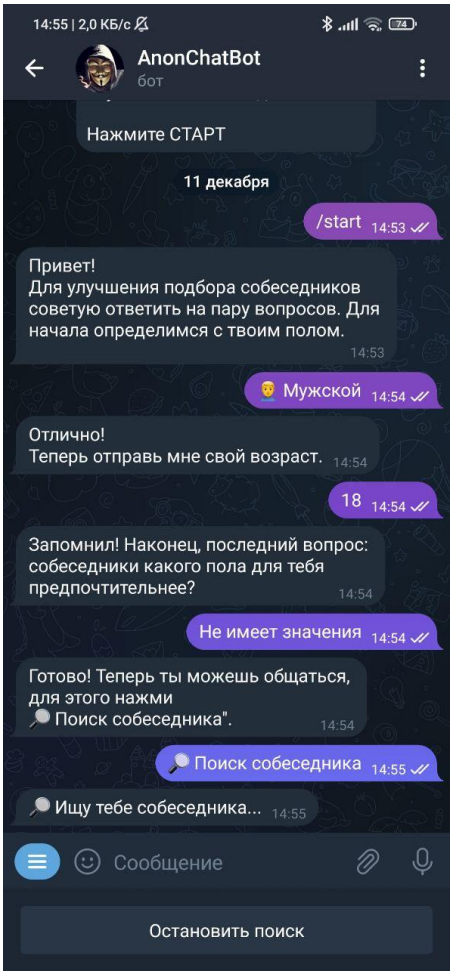




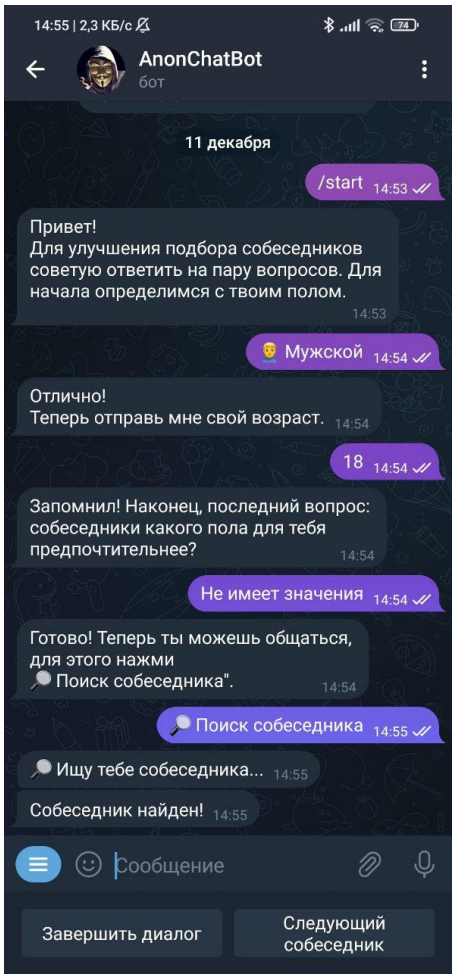


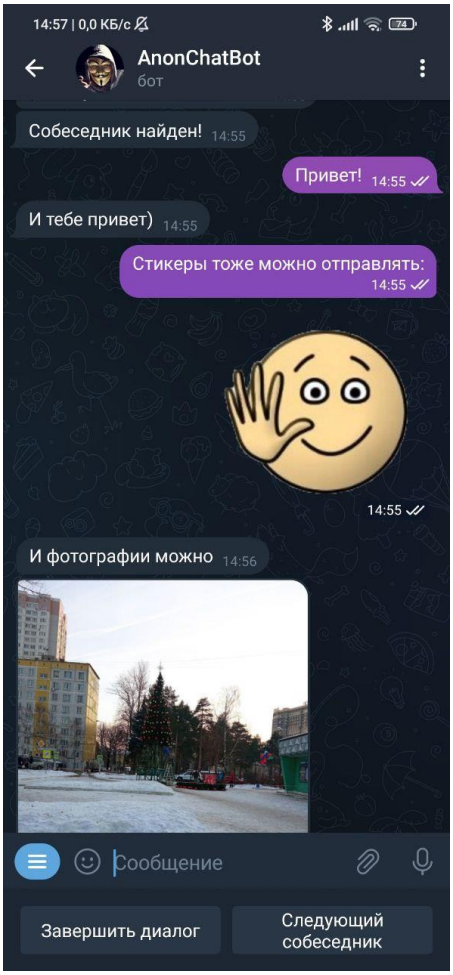




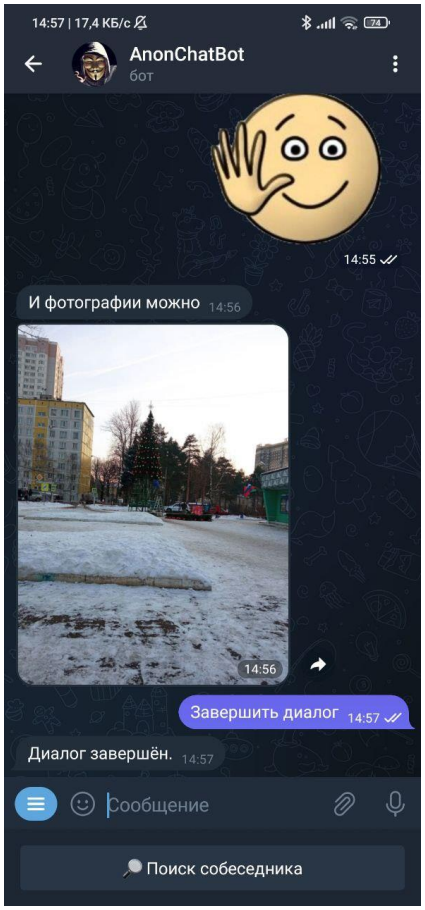


Когда другой пользователь тоже нажёмет «Поиск собеседника» :





Завершение диалога:



Вот так диалог выглядит у собеседника: