**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: |  | Проверил: |
| студент группы ИУ5-34Б |  | преподаватель каф. ИУ5 |
| Угрюмов Михаил |  | 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))

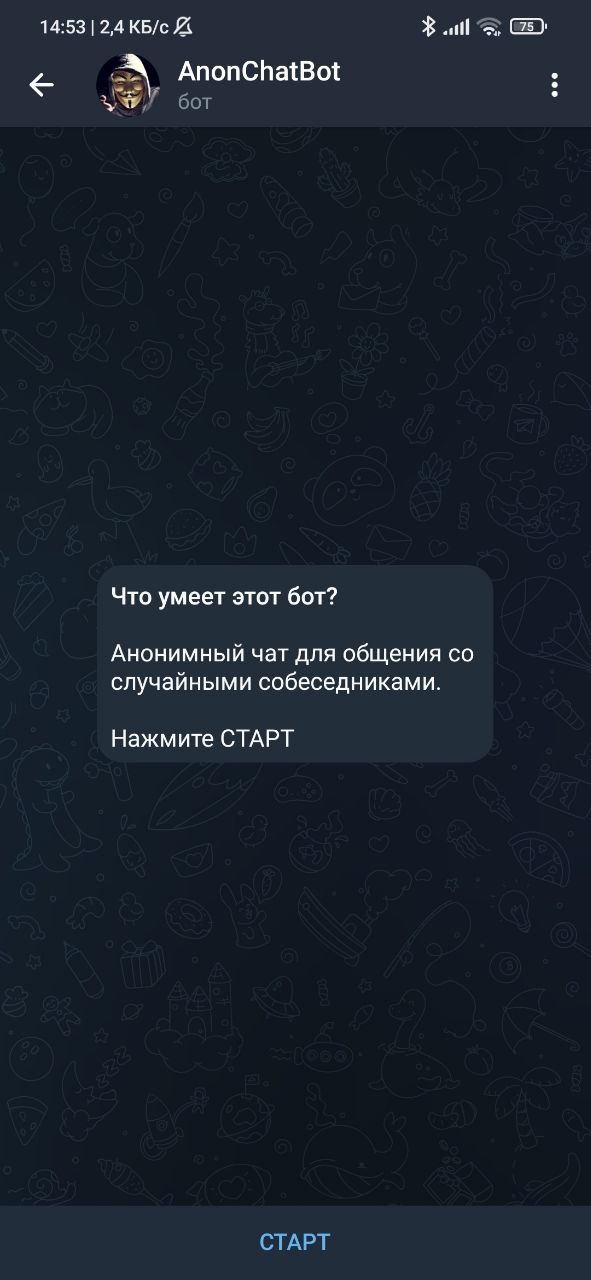
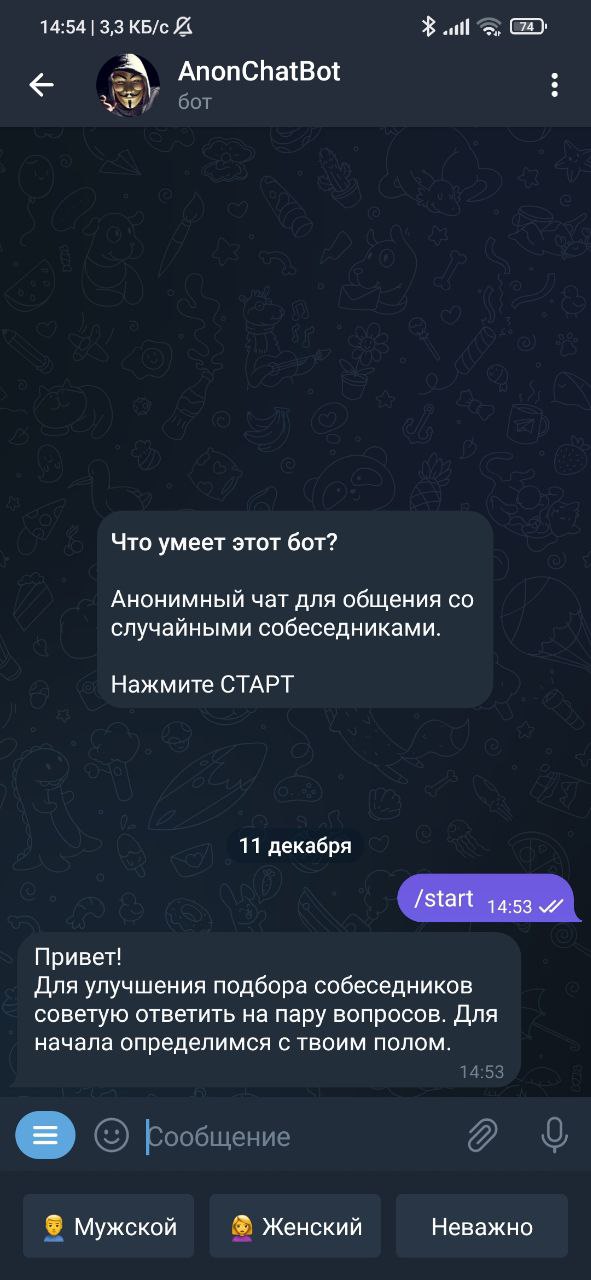
**Application.py**

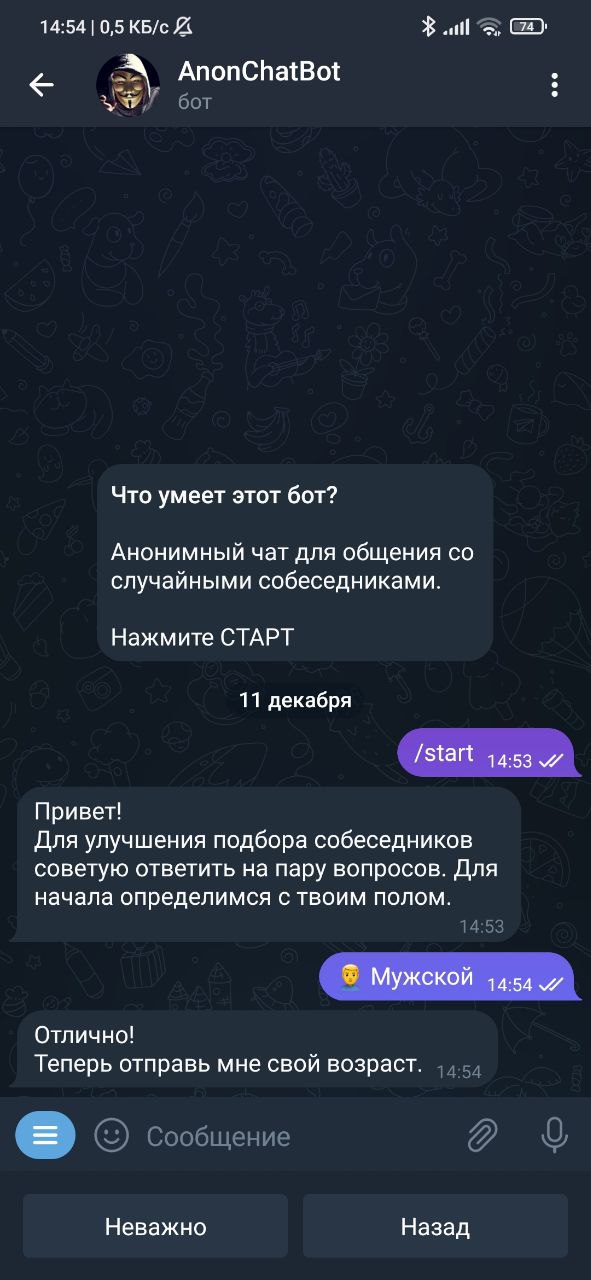
import telebot  
from telebot import types  
from Database\_worker import \*  
  
# класс для работы бота  
class Application:  
 def \_\_init\_\_(self, token):  
 self.bot = telebot.TeleBot(token) # сам бот  
 self.database\_worker = Database\_worker(PATH\_USERS) # ввод-вывод состояний пользователей 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)

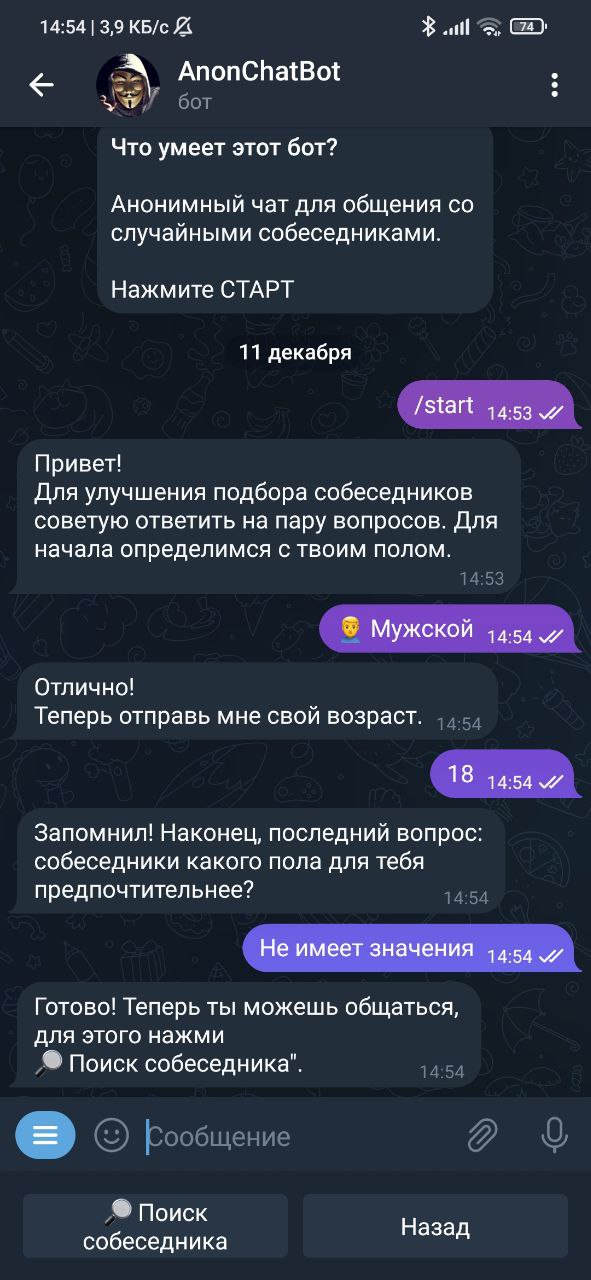
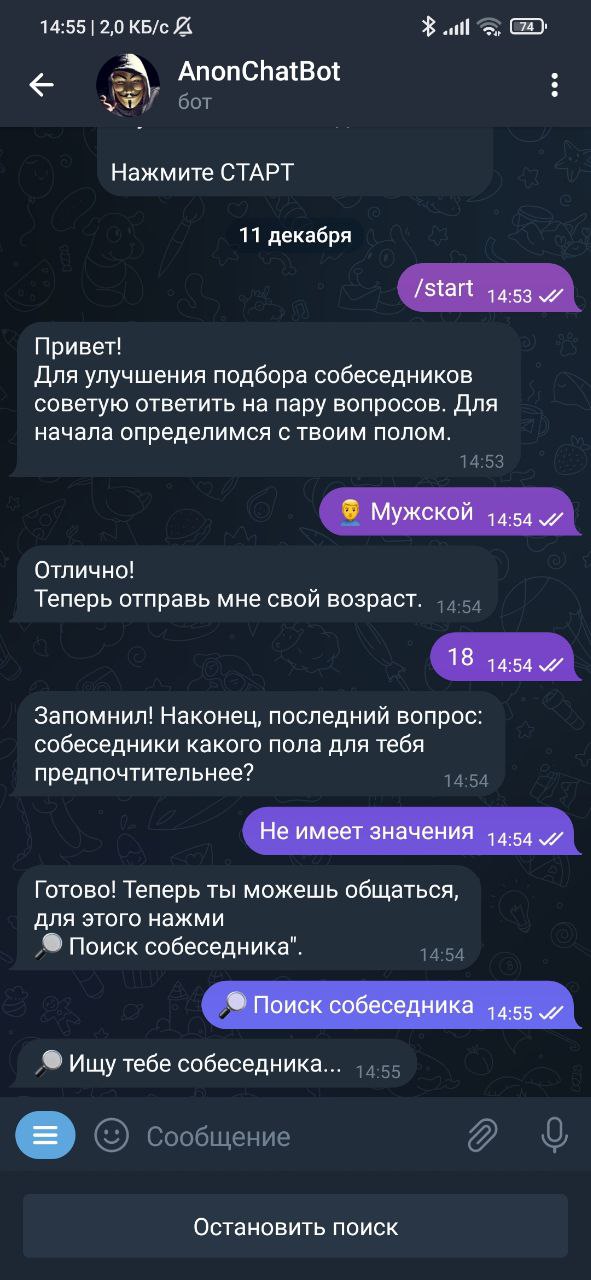
**main.py**

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, 'Cобеседники 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:

