|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Московский технологический университет»**  (МГУПИ) |

Институт КБСП направление 09.03.02

Кафедра КБ-4 «Автоматизированные системы управления»

Дисциплина «Технология программирования в среде Python»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе на тему:**

**РАЗРАБОТКА ПРИЛОЖЕНИЯ С ГРАФИЧЕСКИМ ИНТЕРФЕЙСОМ ДЛЯ МГНОВЕННОГО ОБМЕНА ТЕКСТОВЫМИ СООБЩЕНИЯМИ**

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Леонов С.А.

подпись, дата

Группа БСБО-02-15

Работа защищена на оценку\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев А.С.

подпись, дата

Члены комиссии\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата инициалы и фамилия

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата инициалы и фамилия

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата инициалы и фамилия

|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Московский технологический университет»**  (МГУПИ) |

Институт КБСП направление 09.03.02

Кафедра КБ4 «Автоматизированные системы управления»

Дисциплина «Технология программирования в среде Python»

**ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ**

Студент: 3 курса группа: БСБО-02-15

1 Тема:

**ТЕМА**

2 Срок представления проекта (работы) к защите 23.12.2017 г.

3 Исходные данные для разработки

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4 Содержание пояснительной записки:

Титульный лист

Задание

Содержание

Введение

1 Составляющие проекта

2 Техническое задание

3 Основания для разработки.

4 Назначение разработки.

5 Требования к программе.

6 Стадии и этапы разработки.

Заключение

Список использованных источников

Руководитель работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев А.С.

подпись, дата

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Леонов С.А.

подпись, дата

Содержание

[Введение 4](#_Toc501696480)

[Составляющие проекта 6](#_Toc501696481)

[Техническое задание 7](#_Toc501696482)

[Основания для разработки. 7](#_Toc501696483)

[Назначение разработки. 7](#_Toc501696484)

[Требования к программе. 7](#_Toc501696485)

[1. Требования к функциональным характеристикам. 7](#_Toc501696486)

[2. Требования к составу и параметрам технических средств. 7](#_Toc501696487)

[3. Требования к информационной и программной совместимости. 7](#_Toc501696488)

[Стадии и этапы разработки. 8](#_Toc501696489)

[Заключение 8](#_Toc501696490)

[Приложение 9](#_Toc501696491)

[Список использованной литературы 14](#_Toc501696492)

Введение

Система мгновенного обмена сообщениями, Система обмена мгновенными сообщениями (англ. Instant messaging, IM) — службы мгновенных сообщений (Instant Messaging Service, IMS), программы онлайн-консультанты (OnlineSaler) и программы-клиенты (Instant Messenger, IM) для обмена сообщениями в реальном времени через Интернет. Многие из таких программ-клиентов могут применяться для организации групповых текстовых.

Для подобного рода коммуникации необходима клиентская программа, так называемый мессенджер (англ. messenger — курьер). Отличие от электронной почты здесь в том, что обмен сообщениями идёт в реальном времени (англ. instant — мгновенно). Большинство IM-клиентов позволяет видеть, подключены ли в данный момент абоненты, занесённые в список контактов. В ранних версиях программ всё, что печатал пользователь, тут же передавалось. Если он делал ошибку и исправлял её, это тоже было видно. В таком режиме общение напоминало телефонный разговор. В современных программах сообщения появляются на мониторе собеседника уже после окончания редактирования и отправки сообщения.

Как правило, мессенджеры не работают самостоятельно, а подключаются к центральному компьютеру сети обмена сообщениями, называемому сервером. Поэтому мессенджеры и называют клиентами (клиентскими программами). Термин является понятием из клиент-серверных технологий.

Цель

Разработать систему мгновенного обмена текстовыми сообщениями, позволяющую пользователям отправлять и принимать простые текстовые сообщения.

Список задач:

Разработать метод передачи текстовых сообщений между пользователями через сервер.

Разработать графический интерфейс ввода-вывода текстовой информации.

Составляющие проекта

Server.py – описание класса сервера, функций приёма, обработки и пересылки текстовых сообщений.

Client.py – описание функций графического интерфейса ввода-вывода, функции передачи и приёма сообщений,

Текст исходного кода приведен в приложении 1 к пояснительной записке.

Техническое задание

Основания для разработки.

Учебный план по дисциплине «Технология программирования в среде Python», направление бакалавриата «Информационные системы и Технологии» кафедры КБ4 «Автоматизированные системы управления», Московский технологический университет (МТУ), 2017 г.

Назначение разработки.

Разработанное Desktop-приложение предназначено обмена сообщениями, их корректного отображения, ввода и вывода.

Требования к программе.

К разработанной программе предъявляются следующие требования:

1. Требования к функциональным характеристикам.

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

* 1. Передача имени пользователя в текстовом виде на серверную часть, последующее отображение для каждого сообщения имени пользователя, отправившего его, в виде: Имя пользователя: Текст сообщения. При вводе пробела в качестве имени пользователя, использовать имя «Anonimous».
  2. Передача текстовых сообщений от пользователя на серверную часть путём ввода сообщения в предназначенную для этого часть клиентского окна и последующего нажатия соответствующей кнопки.
  3. Отображение в клиентском окне всех сообщений, принятых серверной частью за время с момента подключения пользователя к серверной части и до его отключения
  4. Отображение в клиентском окне имен пользователей, подключившихся к серверной части с целью использования чата частью за время с момента подключения пользователя к серверной части и до его отключения

1. Требования к составу и параметрам технических средств.

2.1 Компьютер, поддерживающий интерпретатор языка Python версии 3.0 и выше.

2.2 Клавиатура.

2.3 Интерпретатор языка Python версии 3.0.

* 1. Оперативная память объемом от 1 Гб.

2.5 Свободное место на SSD или HDD диске объемом не менее 80 Мб.

1. Требования к информационной и программной совместимости.

3.1 Требования к операционной системе

В состав технических средств должен входить персональный компьютер.

Программные средства и языки программирования: Python версии 3.Х, библиотеки socket, threading, tkinter, sys, интерпретатор языка Python версии 3.Х.

Стадии и этапы разработки.

1. Определение направления работы (тематики)

2. Постановление задачи

3. Проведение исследовательской работы, поиск оптимальных методов решения

4. Создание алгоритма работы программного изделия

5. Выбор рабочей среды

6. Написание серверной части программного изделия Server.py

7. Написание клиентской части программного изделия Client.py

8. Тестирование программного изделия на локальной машине

9. Тестирование программного изделия на OS Linux

10. Исправление возникших неисправностей

11. Финальное тестирование

12. Сдача программного изделия заказчику

Сроки разработки: работоспособное программное изделие необходимо предоставить заказчику не позднее 23.12.2017

Исполнители: студенты группы БСБО-02-15 кафедры КБ-4 факультета КБ института Комплексной Безопасности и Специального Приборостроения Московского Технологического Университета Леонов Сергей и Логвинов Александр.

Разработка программного кода должна происходить на языке программирования Python версии 3.6.4 в среде разработки PyCharm от JetBrains s.r.o.

Заключение

Изучены возможности языка Python.

Разработан метод передачи текстовых сообщений между пользователями через сервер.

Разработан графический интерфейс ввода-вывода текстовой информации.

Получен готовый программный продукт – система мгновенного обмена текстовыми сообщениями.

Приложение 1. Исходный код программы.

Server.py:

import socket  
import sys  
import threading  
  
  
PORT = 9009  
  
  
class ChatServer(threading.Thread):  
 def \_\_init\_\_(self, port, host='localhost'):  
 threading.Thread.\_\_init\_\_(self)  
 self.port = port  
 self.host = host  
 self.server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
 self.users = set() # current connections  
 self.users\_lock = threading.Lock()  
 try:  
 self.server.bind((self.host, self.port))  
 except socket.error:  
 print('Bind failed %s' % socket.error)  
 sys.exit()  
  
 self.server.listen(10)  
  
 def send\_to\_all(self,message):  
 with self.users\_lock:  
 for c in self.users:  
 try:  
 c.sendall(message.encode())  
 except ConnectionResetError:  
 print("connection closed")  
 with self.users\_lock:  
 self.users.remove(c)  
  
 def client\_name(self,conn):  
 with self.users\_lock:  
 self.users.add(conn)  
 print("got to name")  
 name = conn.recv(1024).decode()  
 if name == "{close}":  
 with self.users\_lock:  
 self.users.remove(conn)  
 conn.close()  
 else:  
 if name == " ":  
 name = "Anonimous"  
 new\_user\_annnouncement = ("New user has joined")  
 else:  
 print("correct name "+name)  
 new\_user\_annnouncement = (name +" has joined")  
 self.send\_to\_all(new\_user\_annnouncement)  
 self.client\_messg(conn,name)  
  
 def client\_messg(self,conn,name):  
 while True:  
 message = conn.recv(4096).decode()  
 if message == "{close}":  
 exit\_message = (name+" has left the chat")  
 self.send\_to\_all(exit\_message)  
 with self.users\_lock:  
 self.users.remove(conn)  
 conn.close()  
 break  
 message = (name + ": "+message)  
 self.send\_to\_all(message)  
  
 def handle\_connection(self, conn, addr):  
 print('Client connected with ' + addr[0] + ':' + str(addr[1]))  
 self.client\_name(conn)  
  
  
 def run(self):  
 print('Waiting for connections on port %s' % (self.port))  
 while True:  
 conn, addr = self.server.accept()  
 threading.Thread(target=self.handle\_connection, args=(conn, addr)).start()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 server = ChatServer(PORT)  
 server.run()

Client.py:

from socket import AF\_INET, socket, SOCK\_STREAM  
from threading import Thread  
import tkinter  
import sys  
  
  
def receive():  
 while True:  
 try:  
 msg = client\_socket.recv(BUFSIZ).decode("utf8")  
 try:  
 msg\_list.insert(tkinter.END, msg)  
 except RuntimeError:  
 break  
 except OSError:  
 break  
  
  
def send(event=None):  
 msg = my\_msg.get()  
 my\_msg.set("")  
 client\_socket.send(bytes(msg, "utf8"))  
 if msg == "{close}":  
 client\_socket.close()  
 top.quit()  
  
def send\_close(event=None):  
 client\_socket.sendall(bytes("{close}", "utf8"))  
 # client\_socket.close()  
 top.quit()  
  
def on\_closing(event=None):  
 my\_msg.set("{close}")  
 send()  
 top.quit()  
  
top = tkinter.Tk()  
top.title("Chat Client")  
  
messages\_frame = tkinter.Frame(top)  
my\_msg = tkinter.StringVar()  
my\_msg.set("")  
scrollbar = tkinter.Scrollbar(messages\_frame)  
msg\_list = tkinter.Listbox(messages\_frame, height=15, width=50, yscrollcommand=scrollbar.set)  
scrollbar.pack(side=tkinter.RIGHT, fill=tkinter.Y)  
msg\_list.pack(side=tkinter.LEFT, fill=tkinter.BOTH)  
msg\_list.pack()  
login\_details = tkinter.Label(text="First message is username;\nwhitespace for anonimous.")  
login\_details.pack()  
messages\_frame.pack()  
  
entry\_field = tkinter.Entry(top, width=50, textvariable=my\_msg)  
entry\_field.bind("<Return>", send)  
entry\_field.pack()  
send\_button = tkinter.Button(top, text="Send", command=send)  
send\_button.pack()  
leave\_button = tkinter.Button(top, text="Exit",command=send\_close)  
leave\_button.pack()  
  
top.protocol("WM\_DELETE\_WINDOW", on\_closing)  
if len(sys.argv) == 3:  
 print(sys.argv)  
 HOST = str(sys.argv[1])  
 PORT = int(sys.argv[2])  
 print("Connecting to ip",HOST,", port",PORT)  
else:  
 print("Running with default settings: localhost,9009")  
 PORT = 9009  
 HOST = "localhost"  
BUFSIZ = 1024  
ADDR = (HOST, PORT)  
  
client\_socket = socket(AF\_INET, SOCK\_STREAM)  
client\_socket.connect(ADDR)  
receive\_thread = Thread(target=receive)  
receive\_thread.start()  
tkinter.mainloop()

# Список использованной литературы

[1] <http://habrahabr.ru>. - 2017

[2] Rossum G. Python tutorial. – 1995.

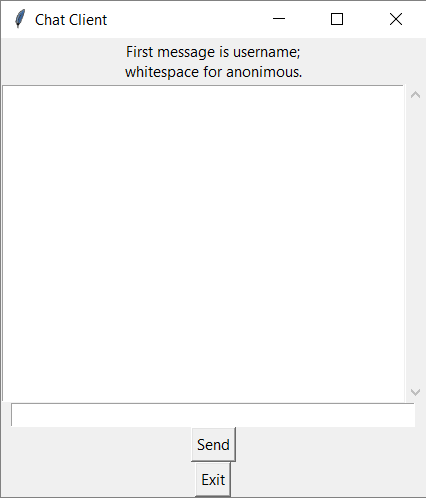
[3] <http://younglinux.info>. - 2017

[4] Саммерфилд М. Программирование на Python 3. – 2011.

[5] Сузи Р. А. Язык программирования Python //М.: Бином. Лаборатория знаний. – 2006.

# Скриншоты

Окно ввода-вывода при запуске программы



Окна ввода-вывода двух пользователей в процессе общения:

