Секция:

Прикладное программирование

Тема проектной работы:

Мессенджер для локальной сети

Выполнил:

Круглов Александр Александрович
ученик 7 класса
Руководитель:
Пинчук Наталья Николаевна
Педагог дополнительного образования кружка "Юный программист"
ГБОУ ДО РК "Дворца детского юношеского творчества"

г. Симферополь, 2023

Содержание:

Введение ……2

1. История развития мессенджеров ……......3

1.1 Что такое мессенджер ……...3

1.2 Самый первый мессенджер ……..3

1.3 Локальная сеть ……..4

1.4 IP адрес и сетевой порт…….4

2. Мой мессенджер……...5

2.1 Инструкции……5

2.2 Как работает мой мессенджер.....6

Заключение.……..7

Список использованных источников……..8

Приложения……..9

Введение

В современном мире необходимо всегда быть на связи. Связь нужна всем. Во всяких чрезвычайных ситуациях нужно обязательно иметь связь. Одним из таких способов коммуникации является мессенджер. Особенность мессенджеров в том, что информация передаётся мгновенно. Так же мессенджеры нужны для общения в коллективах и в организациях. Изучая возможности Python я решил узнать, возможно ли создать программу, которая позволяет объединить пользователей для общения по внутренней (локальной) сети.

Цель: создание простого средства общения для локальной сети.

Задачи:

1. Познакомиться с мессенджерами: изучить историю развития, принципы работы локальной сети.
2. Изучить возможности Python для создания сетевых программ.
3. Выстроить структуру программы для мессенджера.
4. Отладка работы и апробация ее на практике.

1. История развития мессенджеров

1.1 Что такое мессенджер

Мессенджер - программа для передачи сообщений мгновенно, в какой-либо сети. Можно передавать текстовые сообщения. Большинство таких программ могут использоваться для групповых текстовых чатов или видеоконференций.

Мессенджеры отличаются от электронной почты тем, что обмен сообщениями происходит прямо сейчас. 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, так как общение происходит в данный момент.

Обычно мессенджеры не работают сами, а подключаются к специальному компьютеру, через который проходят сообщения. Такой компьютер называют сервером.

1.2 Самый первый мессенджер

IRC (Internet Relay Chat) - протокол для обмена сообщениями в режиме реального времени.

Разработан в основном для группового общения, также позволяет общаться через личные сообщения и обмениваться данными, в том числе файлами.

IRC-сеть — это группа серверов, соединённых между собой. Простейшей сетью является один сервер.

IRC предоставляет возможность как группового, так и приватного общения, то есть в личных сообщениях.

Пользователь может отправить сообщение списку пользователей, при этом серверу отправляется список, сервер выбирает из него пользователей и отправляет сообщение каждому из них.

Так же можно использовать каналы. Сообщение отправляется серверу, а сервер отправляет его всем остальным пользователям в канале.

Можно так же отправить широковещательное сообщение. Сообщения пользователей, означающие изменение состояния сети (режима канала или статуса пользователя и т.п.), должны отправляться всем серверам, входящим в сеть. Сообщения от сервера тоже должны быть отправлены всем серверам.

1.3 Локальная сеть

Мессенджер, который я написал, использует локальную сеть. Локальная сеть (LAN) — компьютерная сеть, покрывающая небольшую территорию или небольшую группу зданий.

Компьютеры могут соединяться между собой, используя различные среды доступа: медные провода (витая пара), оптические провода (оптические кабели) и через радиоканал. Проводные, оптические связи устанавливаются через Ethernet и другие средства. Отдельная локальная сеть может иметь связь с другими локальными сетями через шлюзы, а также быть частью глобальной сети (например, Интернет) или быть подключённой к ней.

Глобальная сеть — компьютерная сеть, охватывающая большие территории, возможно находящиеся в различных городах и странах.

1.4 IP адрес и сетевой порт

IP-адрес — уникальный адрес устройства в компьютерной сети. В 4-й версии IP-адрес записывается в виде четырёх десятичных чисел значением от 0 до 255, разделённых точками, например, 192.168.0.3. В 6-й версии IP-адрес записывается в виде восьми четырёхзначных шестнадцатеричных чисел, разделённых двоеточиями, например, 2001:0db8:85a3:0000:0000:8a2e:0370:7334. Ведущие нули допускается в записи опускать. Нулевые группы, идущие подряд, могут быть опущены, вместо них ставится двойное двоеточие (fe80:0:0:0:0:0:0:1 можно записать как fe80::1). Больше одного такого пропуска в адресе написать нельзя.

IP-адрес состоит из двух частей: номера сети и номера узла. Для выхода в глобальную сеть необходимо, чтобы IP-адрес был из другого блока адресов, или в локальной сети должен быть сервер, заменяющий внутренний IP-адрес (серый) на внешний IP-адрес (белый).

Порт — целое неотрицательное число. В основном на компьютере в пространстве пользователя исполняется несколько процессов, в каждом из которых выполняется какая-либо программа. В случае если несколько программ используют компьютерную сеть, то ОС периодически получает пакет, предназначенный для одной из программ. То есть порт - ячейка, в которой выполняется какая-либо программа.

2. Мой мессенджер

2.1 Инструкции

При входе в серверную часть нужно ввести сетевой порт, на котором будет находиться наш чат.

Enter the port of the server:

Затем мы видим IP адрес и порт сервера, сервер ждёт подключений.

Server bind at host <ip адрес сервера> and port <порт сервера>

Зайдём в клиентскую часть. Нужно ввести IP адрес сервера и порт, который мы вводили.

Enter the ip of the server. If ip is wrong restart the programm and try again:

Enter the port of the server. If port is wrong restart the programm and try again:

Подключение произведено успешно! Осталось ввести имя:

Connected to <ip адрес сервера>, <порт сервера>

Enter your nickname: <ваше имя>

И можно общаться. Сообщение выглядит так:

<дата и время (с точностью до миллисекунды)>: <имя отправителя>: <сообщение>

Зайдём на сервер. Выводятся сообщения о подключении пользователей:

User <его ip адрес> connected!

Сообщения от пользователей кодируются в кодировку utf-8 и шифруются. Шифрование проходит так: к номеру каждого символа в таблице unicode (у каждого символа есть номер, и эти номера записаны в таблице

unicode) прибавляется ключ - двузначное число. Этот ключ есть у всех пользователей этого мессенджера. А дешифрование - вычесть из номера зашифрованного символа этот ключ. На сервере это выглядит так:

User <его ip адрес> sent <зашифрованное сообщение>

Попробуем закрыть клиентскую программу. На сервере вывелось сообщение о выходе этого пользователя:

User <его ip адрес> removed!

2.2 Как работает мой мессенджер

Мой мессенджер написан на языке программирования Python. При его написании я использовал библиотеки socket - для работы с сетью, threading - для создания потоков (нескольких процессов в одной программе), rich - для цветного текста в консоли, datetime - вывод времени и os - для очистки консоли.

Сервер:

У нас есть класс Server, основанный на классе Socket. Класс Socket находится в отдельном файле. В нём прописано, что объект класса Socket является, так сказать, сетевым.

Есть 5 функций: инициализация, настройка, отправка данных, получение данных, ожидание подключений.

В функции инициализации создаётся список пользователей.

Сначала вызывается функция настройки - нужно ввести порт. После ввода создаётся сервер с ip адресом компьютера и портом, который мы ввели. И сервер готов принимать подключения!

Затем вызывается функция ожидания подключений. Сервер принимает пользователей, создаётся поток (процесс) принятия сообщений и запускается. Сервер принимает сообщение и вызывает функцию отправки сообщения всем пользователям.

Если пользователь отключается, то выводится сообщение о его выходе.

Клиент:

Есть класс Client, тоже основанный на классе Socket.

Импортируется файл Encrypting - для шифрования.
Здесь уже 4 функции - инициализация, настройка, отправка и принятие.
Вызывается функция настройки - спрашивается ip адрес и порт сервера.
Создаётся 2 потока на отправку и принятие данных и запускаются.
Отправка данных - один раз запрашивается имя, потом формируется сообщение - <Имя>: <Ввод пользователя>. Потом оно отправляется.
Принятие - принимает данные с сервера, расшифровывает их, прибавляет к списку сообщений, очищает консоль и выводит список. Программный код представлен скриншотами в Приложениях 6,7,8.

Заключение

Выполняя эту работу, я получил следующие новые знания: историю появления мессенджеров, особенности их создания, какие бывают сети и что такое локальная сеть, что такое IP адрес, из чего он формируется и как правильно его использовать, что такое сетевой порт и как его использовать для общения в глобальной и локальной сети, в чем разница между глобальной и локальной сетью. Изучил новые для меня возможности языка программирования Python. Кроме того, мне стало понятным как пользователи могут общаться по локальной сети с помощью созданного программного обеспечения. У такого способа общения есть достоинства – никто из другой сети не подключится, и можно общаться без интернета, но есть и недостатки – нельзя связаться с кем-то на удаленном расстоянии, участники чата находятся в пределах одной аудитории или одного здания. Для налаживания такого общения достаточно ПК, подключенного в локальную сеть и программного продукта такого как этот мессенджер.
В данной работе возможно дальнейшее развитие проекта, как-то - улучшение дизайна, добавление функций оповещения и настройки объема передаваемой информации. Это станет мне возможным при дальнейшем изучении возможностей Python. Так же можно придумать свое оригинальное название мессенджеру.

Мессенджеры очень важны в нашей жизни, много информации, которую необходимо передавать сотрудникам одного учреждения во время работы или, например, студентам во время обучения и др. Это является актуальным в момент отключения Интернета и такой вид связи поможет пользователям оставаться на связи без подключения к глобальной сети.

Список использованных источников

Список литературы:

1. Библиотека журнала Хакер. Python Глазами Хакера, – СПб: BHV-Петербург, 2022. - 172 с.
2. Метиз Э. Изучаем Python, – СПб.: Питер, 2021. – 511 с.
3. Доусон М. Программируем на Python, – СПб.: Питер, 2020. – 414 с.
4. Бриггс Д. Python для детей, – М: «Манн, Иванов и Фербер», 2020 – 317с.
5. Пэйн Б. Python для детей, – М: «Эксмо», 2019, – 350 с.

Интернет источники:

https://ru.wikipedia.org/wiki/Система_мгновенного_обмена_сообщениями

https://ru.wikipedia.org/wiki/IRC

https://ru.wikipedia.org/wiki/Локальная_вычислительная_сеть

https://ru.wikipedia.org/wiki/Глобальная_вычислительная_сеть

https://ru.wikipedia.org/wiki/IP-адрес

https://ru.wikipedia.org/wiki/Порт_(компьютерные_сети)

Приложение 1

Рис.1 Ввод порта сервера

Рис.2 Сервер настроен и ждёт подключений

Приложение 2

Рис.1 Ввод ip адреса сервера в клиентской части программы

Рис.2 Ввод порта в клиентской части программы

Приложение 3

Рис.1 Подключение произведено успешно! Ввод имени

Рис.2 Имя введено! Можно писать сообщения

Приложение 4

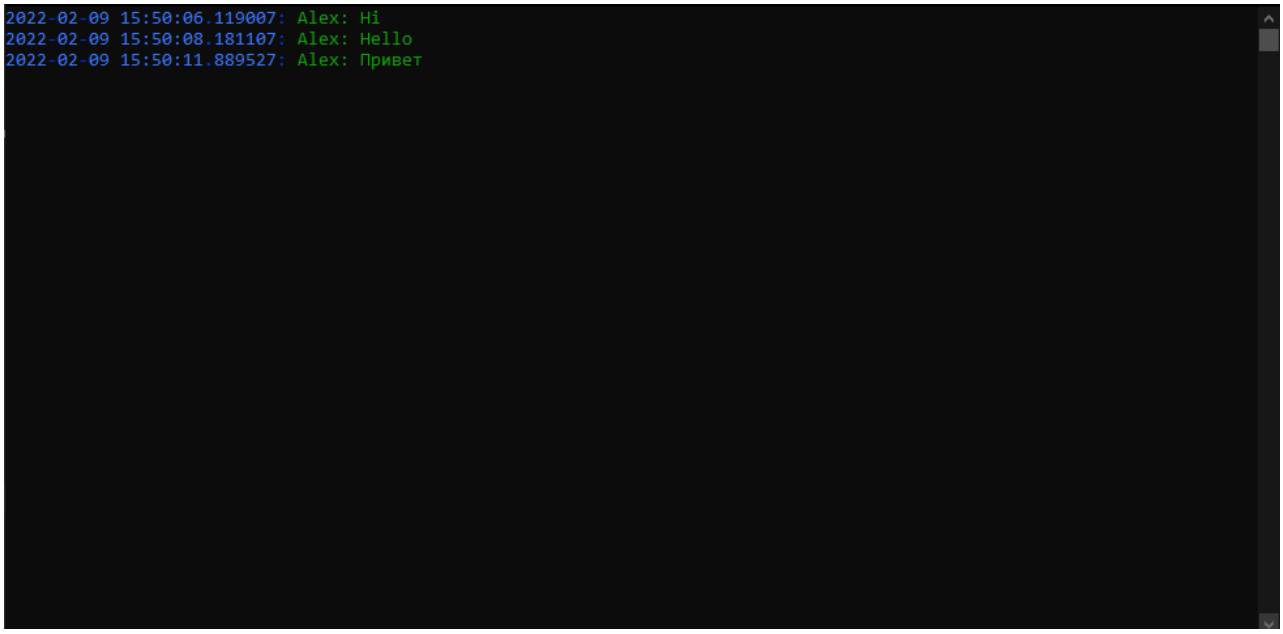


Рис.1 Сообщения в клиентской программе

Приложение 5

Рис.1 Сообщение о подключении. Зашифрованные сообщения

Рис.2 Сообщение о выходе пользователя

Приложение 6

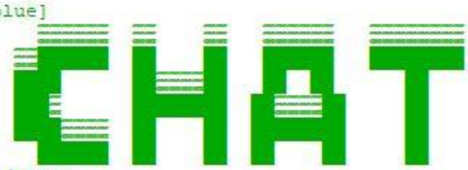
Программный код серверной части мессенджера:

main.py - C:\НН Работа\КОНКУРСЫ\шаг в науку 2022\main.py (3.7.5)

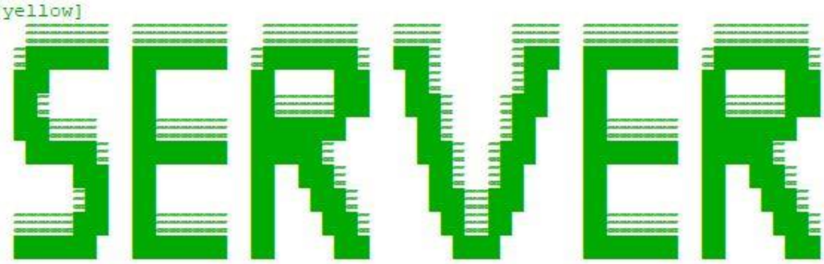
File Edit Format Run Options Window Help

```python
from Socket import Socket
import socket
import threading
from datetime import datetime
from rich import print
from rich import inspect
from rich.console import Console
console = Console()
class Server(Socket):
        def __init__(self):
                super(Server, self).__init__()
                self.users = []

        def set_up(self):

                host = socket.gethostbyname_ex(socket.gethostname())[-1][-1]
                port = int(console.input("[blue]Enter the port of the server: [/blue]"))

                self.bind((host, port))
                print(f"[yellow]Server bind at host[/yellow] [blue]<{host}>,[/blue] [yellow]and port[/yellow] [blue]<{port}>[/blue]")
                self.listen(5)
                print("[yellow]Server is listening...[/yellow]")
                self.accept_sockets()

        def send_data(self, data):
                for user in self.users:
                        user.send(data)

        def listen_socket(self, listened_socket=None):
                global address
                while True:
                        try:
                                data = listened_socket.recv(2048)
                                print(f"[green]User[/green] [blue]<{address[0]}>[/blue] [green]sent[/green] [blue]<{data}>[/blue] [green]at[/green] [blue]<{datetime.now()}>[/blue]")

                                self.send_data(data)
                        except ConnectionResetError:
                                print(f"[red]User[/red] [blue]<{address[0]}>[/blue] [red]removed![/red]")
                                self.users.remove(listened_socket)
                                return
        def accept_sockets(self):
                global address
                while True:
                        user_socket, address = self.accept()
                        print(f"[green]User[/green] [blue]<{address[0]}>[/blue] [green]connected![/green]")

                        self.users.append(user_socket)
                        listen_accepted_user = threading.Thread(
                                target=self.listen_socket,
                                args=(user_socket,)
                                )
                        listen_accepted_user.start()
if __name__ == '__main__':
        print("""
                [blue]
```

```python
                [blue]
```



```python
                [/blue]
                [yellow]
```



```python
                [/yellow]
                """)
        server = Server()
        server.set_up()
```

Приложение 7

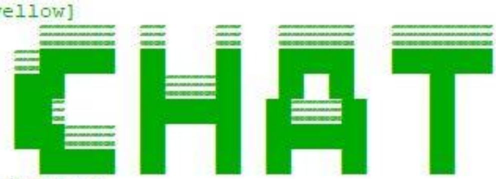
Программный код клиентской части мессенджера:

client.py - C:\НН Работа\КОНКУРСЫ\шаг в науку 2022\client.py (3.7.5)

File Edit Format Run Options Window Help

```python
from Socket import Socket
import socket
from threading import Thread
from datetime import datetime
import time
from Encrypting import Encryptor
from os import system
from rich import print
from rich import inspect
from rich.console import Console
console = Console()

class Client(Socket):
	def __init__(self):
		super(Client, self).__init__()
		self.messages = ""
		self.encryptor = Encryptor()
	def set_up(self):
		try:
			host = console.input("[blue]Enter the ip of the server, if ip is wrong, restart the programm and try again: [/blue]")
			port = int(console.input("[blue]Enter the port of the server, if port is wrong, restart the programm and try again: [/blue]"))
			self.connect(
				(host, port)
				)
			print(f"[green]Connected to[/green] [blue]<{host}>, <{port}>[/blue]")
			listen_thread = Thread(target=self.listen_socket)
			listen_thread.start()

			send_thread = Thread(target=self.send_data, args=(None,))
			send_thread.start()

		except ConnectionRefusedError:
			print("[red]Sorry, server is offline[/red]")
			time.sleep(3)
			exit(0)
		except OSError:
			print("[red]Problem with connection[/red]")
			time.sleep(3)
			exit(0)
		except ValueError:
			print("[red]Sorry, something from entry is wrong[/red]")
			time.sleep(3)
			exit(0)
	def listen_socket(self, listened_socket=None):
		global nick
		while True:
			data = self.recv(2048)
			message = data.decode("utf-8")

			clean_data = self.encryptor.decrypt(message)

			self.messages += f"[blue]{datetime.now()}:[/blue] [green]{clean_data}[/green] \n"
			system("cls")
			print(self.messages)

	def send_data(self, data):
		global nick
		nick = console.input("[green]Enter your nickname: [/green]")
		while True:
			mess = str(nick) + str(": ") + str(input(" "))
			encrypted data = self.encryptor.encrypt(mess)
```

```python
if __name__ == '__main__':
	print("""
		[yellow]
```



```python
		[/yellow]
		[blue]
```

```python
		[/blue]
		""")
	client = Client()
	client.set_up()
	#window.mainloop()
```

Приложение 8

Программный код шифратора/дешифратора сообщений:

Encrypting.py - C:\НН Работа\КОНКУРСЫ\шаг в науку 2022\Encrypting.py (3.7.5)

File Edit Format Run Options Window Help

```
import time
import random
class Encryptor:
        def __init__(self):
                self.key = 67

        def encrypt(self, text):
                encrypted_text = ""

                for symb in text:
                        encrypted_text += chr(ord(symb) + self.key)

                return encrypted_text

        def decrypt(self, text):

                decrypted_text = ""

                for symb in text:
                        decrypted_text += chr(ord(symb) - self.key)

                return decrypted_text
```