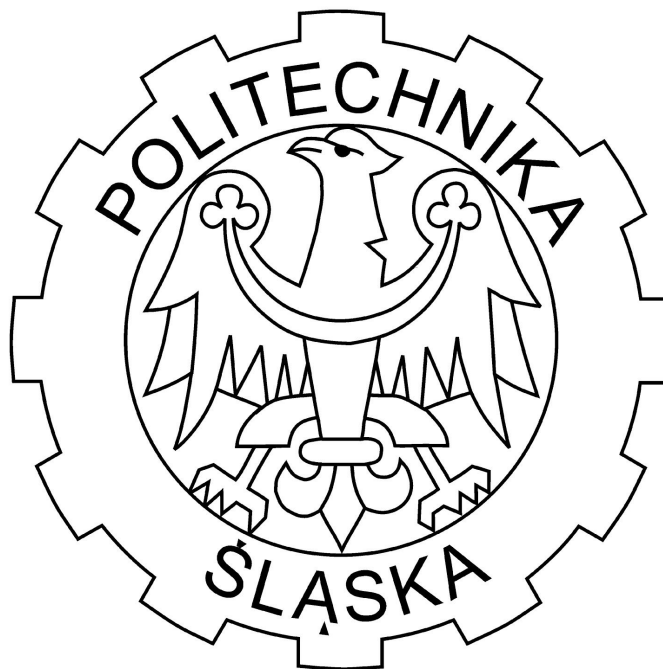


**Politechnika Śląska w Gliwicach**  
**Wydział Automatyki, Elektroniki i Informatyki**



**Programowanie Komputerów**

**Projekt - Komunikator Sieciowy**

Autor:	Damian Pietryja
Prowadzący:	dr inż. Tomasz Moroń
Rok akademicki:	2018/2019
Kierunek:	Teleinformatyka
Rodzaj studiów:	SSI
Semestr:	3
Grupa dziekańska:	2

## 1.Wstęp- opisy wykonywanego projektu:

Wykonywany przeze mnie projekt to komunikator sieciowy, który przekształcił się w czat wieloużytkownikowy. W projekcie tym znajdują się 2 programy:

- Serwer- jest to serwer TCP/IP, który służy do komunikowania się pomiędzy użytkownikami oraz nadzorowaniu przesyłanych informacji. Jest to aplikacja konsolowa.
- GUIClient- jest to program okienkowy dla użytkownika, który służy do rozmowy w danym "oknie" czatowym.

Aplikacja okienkowa, czyli GUIClient była stworzona przez zalecaną przez prowadzących bibliotekę QT. Obie aplikację, czyli GUIClient oraz Serwer opierają się na tej samej bibliotece połączeniowej (tzw. "networkowej") WinSocket, która umożliwia połączenie ze sobą aplikacji dzięki adresom IP.

## 2. Teoretyczne podstawy projektu

### 2.1 Krótki opis najważniejszych struktur w programach

W programie Serwer można wyróżnić najważniejszą strukturę, **TcpListener**.

**TcpListener** jest to klasa, która implementuje różne metody do inicjalizacji całego serwera, jak i komunikacji między użytkownikami, która lista połączonych użytkowników z serwerem ,przechowuje się w tablicy mieszającej. Klasa ta również dziedziczy metody czysto wirtualne z klasy Sck, która jest wspólna dla serwera jak i GUIClienta.

Pozostałe struktury mają mniejsze znaczenie. Służą do sprawdzenie parametrów wpisanych przez osobę, która tworzy serwer. Szczegółowa dokumentacja klas znajduje się w pliku: Dokumentacja\_wewnętrzna\_serwera.pdf

W aplikacji GUIClient możemy wyróżnić 3 najważniejsze struktury:

- **GUIClient**-klasa QDialog, czyli służąca do pokazania się okna dialogowego. Służy ona do uruchomienia użytkownika i połączenia z serwerem. Jest ona tzw.barierą, czyli przy poprawnym dołączeniu serwera pozwala na dalsze czynności.

- ChatBox**- klasa QWidget, czyli klasa służąca do pokazania się okienka, bardziej rozbudowanego niż okno dialogowe. Służy ona do wyświetlania, pisania wiadomości z innymi użytkownikami serwera czatowego.

- TcpClient**- jest to klasa, która jest odpowiedzialna za komunikację z serwerem. Implementuję różne metody służące do tej komunikacji i łączeniu się z serwerem

Szczegółowa dokumentacja aplikacji znajdzie się w pliku:Dokumentacja\_wewnętrzna\_GUIClient.pdf

### 2.2 Schemat działania programów

Program serwer jak wcześniej został powiedziany służy do komunikacji między użytkownikami, ale jak to wygląda? Serwer ten poprzez wcześniej inicjalizowany przez użytkownika adres IP oraz port, tworzy gniazdo do nasłuchiwanie, czy ktoś chce dołączyć do serwera. Jeśli jest już kilka użytkowników serwer ten (jeśli jest wiadomość odebrana przez niego) rozsyła po innych wiadomości odebrane. Po prostu jest tzw. "mózgiem" całego czatu.

Program GUIClient (zwany dalej klientem) służy jako interfejs graficzny dla użytkownika. Po wpisaniu odpowiednich parametrów (adresu IP i portu serwera) ujawnia się

oczom okienko czatowe. Zaczynamy to tak wszystko ładnie zadziała, w kliencie również musi powstać gniazdo internetowe użytkownika. Dzięki niemu serwer może podłączyć użytkownika do siebie i komunikować się.

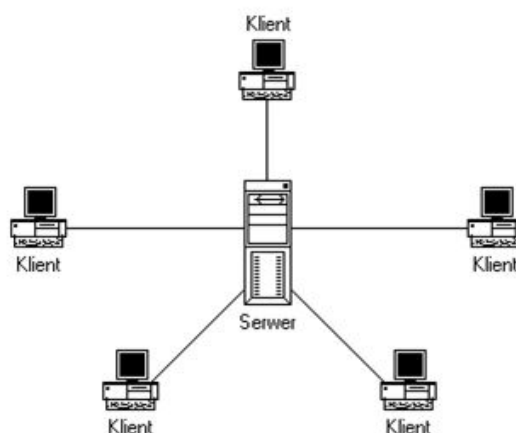
Komunikacja ta działa według modelu TCP/IP przedstawionego poniżej na obrazku.

Model OSI	Model TCP/IP	PROTOKÓŁ / urządzenie sieciowe
Warstwa aplikacji	Warstwa aplikacji	HTTP, TELNET SSH, FTP POP3, SMTP IMAP, DNS
Warstwa prezentacji		
Warstwa sesji		
Warstwa transportowa	Warstwa transportowa	TCP, UDP
Warstwa sieci	Internet	IP, RIP OSPF, EIGRP /Router, Switch
Warstwa łącza danych	Warstwa dostępu do sieci	ETHERNET / Switch, Most, Hub
Warstwa fizyczna		Media transmisyjne (skrętka, światłowód), kodowanie

*Rys.2.2.1 Przedstawia różne modele sieciowe.*

Serwer zajmuje się głównie warstwą transportową, a klient warstwą aplikacji oraz warstwą transportową.

Połączenie między serwerem, a klientami jest siecią typu gwiazda. Poniżej również przedstawiam przykładowy schemat jak wygląda z strony fizycznej sieć.



*Rys.2.2.2. Schemat sieci z strony fizycznej*

### 3. Prezentacja działania programów.

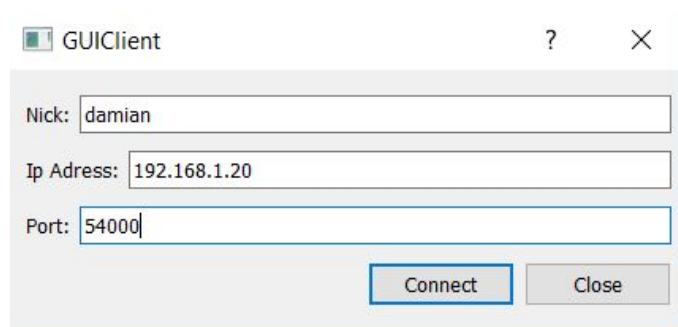
Prezentacja działania programu została przeprowadzona między dwoma komputerami połączonymi przez router Wi-Fi w tej samej sieci lokalnej. Instrukcja dla użytkownika znajduje się w pliku: dokumentacja\_zewnetrzna.pdf

Przykładowe inicjalizacja czatu dla adresu ip-192.168.1.20 oraz portu-54000 .  
Najpierw trzeba włączyć serwer podanymi parametrami.

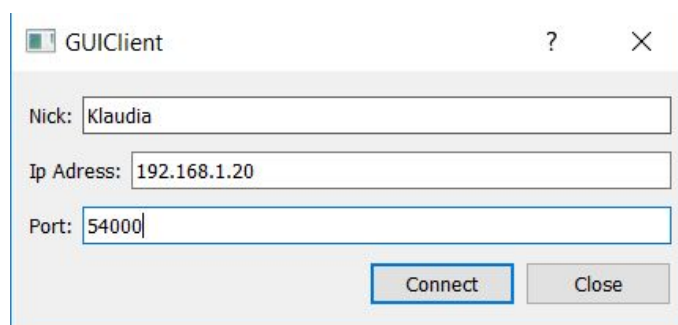
```
C:\Users\damian\Source\Repos\d5b347fc-gr21-repo\Projekt\Serwer\x64\Debug>serwer.exe -i
Visual Leak Detector read settings from: C:\Program Files (x86)\Visual Leak Detector\vld.ini
Visual Leak Detector Version 2.5.1 installed.
Witam Cie w serwerze. Zaraz musisz podaś parametry serwera!
Podaj numer adresu IP ,na którym ma zostac utworzony serwer (przemysl to): 192.168.1.20
Podaj numer portu ,na którym ma zostac utworzony serwer (przemysl to): 54000
Started...
>>
```

*Rys.3.1 Inicjalizacja serwera za pomocą podanych parametrów pokazanych na obrazku.*

Gdy serwer wyświetli komunikat “started” oznacza to, że serwer działa w sieci i następnie włączamy klienta.

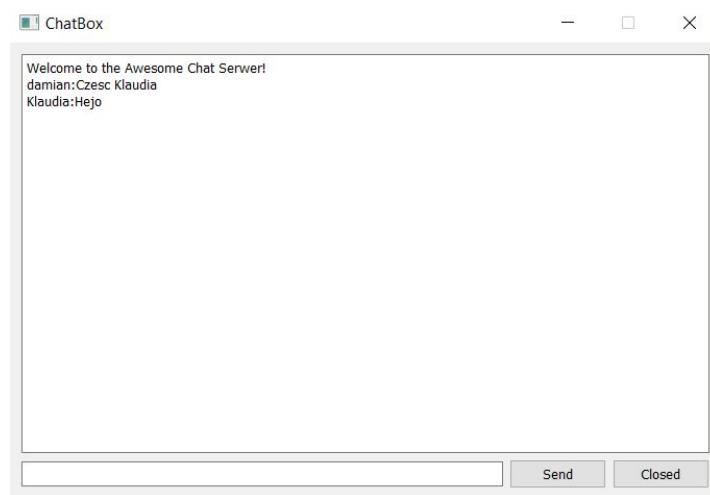


*Rys.3.2 Okno dialogowe klienta 1 służące połączeniu z serwerem (czatem).*



*Rys.3.3 Okno dialogowe klienta 2 służące połączeniu z serwerem (czatem).*

Następnie po pozytywnym połączeniu otwiera się okno czatowe, która umożliwia komunikację.



*Rys.3.4 Okno czatowe użytkownika.*

Dzięki przyciskowi *send* wysyłamy wiadomość do użytkownika, a przycisk *closed* rozłącza nas z oknem czatowym.

## 4. Wnioski

Temat projektu, który wybrałem był inspirowany tym, że jestem na kierunku Teleinformatyk, czyli będę miał dużo styczności z komunikacją sieciową. Jest to mój pierwszy poważny projekt pisany obiektowo, czyli schemat pisania obiektowo był mi nieznany. Ten problem był najmniejszy, największym problemem okazało się poznanie biblioteki WinSock, które zapoznanie się z nią i walczenie z wyciekami pamięci (biblioteka bierze bardzo dużo pamięci i trzeba dobrze operować pamięcią) zajęło ponad miesiąc. Również bardzo ważna jest synchronizacja między użytkownikami, gdzie stanowiło również to duży problem (nie na darmo się mówi, że jeśli pisząc jakiś projekt w grupie, odejdzie osoba odpowiedzialna za synchronizację to cały projekt się sypie). Po poznaniu tej biblioteki i napisaniu prototypu klienta jako aplikacji konsolową, trzeba było przenieść to na aplikację okienkową. Tu nastąpiła kolejna trudność z poznanie biblioteki QT. Stworzenie przycisków, okien itd, nie było problemem, ale problemem było połączenie ze sobą biblioteki QT i biblioteki WinSock.

Nie udało mi się zrealizować wszystkiego co zaplanowałem (ze względu na wyżej wymienione problemy), czyli:

- wyświetlanie okna z osobami, które są dostępne w czacie
- funkcji disconnect (miała być zamiast przycisku closed), czyli gdybyśmy nacisnęli przycisk *disconnect*, okno czatowe by się zamknęło, a otworzyło okno dialogowe tj. na początku oraz rozłączyło z tym czatem.
- wyświetlanie polskich znaków na serwerze
- wysyłania wiadomości od serwera (nie dotyczy komunikatów, tylko pisanie wiadomości przez administratora serwera)

-idealnej synchronizacji serwera, czyli przechwytywania każdej wiadomości wysyłanej przez użytkowników. Przy dużej ilości wiadomości, serwer może nie nadążać z odbieraniem i wysyłaniem.

Mimo długiej listy nie zrealizowanych rzeczy, moim skromnym zdaniem projekt wyszedł bardzo dobrze, jak na osobę, która nie ma dużego doświadczenia z programowaniem obiektowym oraz trudnością projektu. Projekt ten nauczył mnie zapoznać z biblioteką WinSock oraz QT. Również nauczyłem się, jak bardzo ważne jest zarządzanie pamięcią komputera (przez pojawianie się podczas projektu tzw. bluescreen'ów) oraz jak duże ułatwienie jest pisanie programów obiektowo.