

Politechnika Poznańska

Telefonia IP

Dokumentacja projektowa

No Title Call

Juliusz Horowski 136247

juliusz.horowski@student.put.poznan.pl

Marcin Złotek 136334

marcin.zlotek@student.put.poznan.pl

Poznań, 2020

Spis treści

1	Ogólna charakterystyka	2
2	Architektura systemu	2
3	Wymagania	2
3.1	Funkcjonalne	2
3.1.1	Użytkownik niezalogowany/niezarejestrowany	2
3.1.2	Użytkownik zalogowany	2
3.1.3	Serwer	3
3.1.4	Administrator serwera	4
3.2	Pozafunkcjonalne	4
4	Narzędzia, środowisko, biblioteki	5
5	Diagramy UML	6
5.1	Przypadków użycia	6
5.2	Stanów	7
5.3	Klas	8
5.4	Sekwencji	9
6	GUI	11
7	Testy i przebiegi	14
8	Podsumowanie	16
9	Podział pracy	16
9.1	Cele zrealizowane	16
9.2	Cele niezrealizowane	16
9.3	Problemy	16
9.4	Perspektywy rozwoju	17
9.5	Dokończenie samodzielne	17

1 Ogólna charakterystyka

Tematem projektu jest system komunikacji głosowej poprzez protokół IP, przeznaczona dla rozmów 1:1. System będzie umożliwiał bezpieczne prowadzenie rozmów, a także czatu tekstowego. Aplikacja klienta będzie posiadała interfejs graficzny. Projekt nosi nazwę *NoTitleCall*, sloganem jest: „*listen a lot, talk more*”.

2 Architektura systemu

Architektura systemu jest w postaci klient-serwer. Użytkownik będzie się łączył z serwerem w celu nawiązania połączenia z innym użytkownikiem. Cała transmisja będzie prowadzona poprzez serwer; zarówno rozmowa, czat tekstowy jak i pakiety informacyjne dla serwera.

Do komunikacji z serwerem służy protokół TCP oraz własna implementacja protokołów do nawiązywania połączenia. Do przechowywania danych o użytkownikach serwer wykorzystuje pliki *XML*. Poufność przekazywanych danych jest zapewniona przez protokół *TLS*.

3 Wymagania

3.1 Funkcjonalne

Wymagania funkcjonalne dla aplikacji klienta i serwera pośredniczącego.

3.1.1 Użytkownik niezalogowany/niezarejestrowany

Wymagania dla aplikacji klienta niezalogowanego/niezarejestrowanego:

- utworzenie konta użytkownika,
- utworzenie hasła do konta,
- wybór pseudonimu użytkownika,

3.1.2 Użytkownik zalogowany

Wymagania dla aplikacji klienta zalogowanego:

- prowadzenie czatu głosowego,
- prowadzenie czatu tekstowego (z tym samym użytkownikiem co czat głosowy),
- wybór osoby z którą chcemy nawiązać kontakt,

- nawiązanie połączenia z innym użytkownikiem,
- zawieszenie połączenia,
- zakończenie połączenia,
- wybór motywu graficznego,
- zmiana głośności rozmówcy,
- wyciszenie rozmówcy,
- przechowywanie historii połączeń w formie listy ostatnich 20 połączeń,
- wyświetlanie listy kontaktów, w tym przypiętych na samej górze,
- opisywanie kontaktów użytkownika (np. zmiana nicku).

3.1.3 Serwer

Wymagania dla serwera pośredniczącego w transmisji:

- informowanie o dostępności użytkownika (dostępny, niedostępny, nie przeszkadzać),
- informowanie użytkownika o błędach w komunikacji,
- umożliwienie nawiązania połączenia,
- utrzymanie połączenia,
- przechowywanie listy kontaktów,
- przechowywanie informacji dotyczącej użytkowników (hasła, pseudonimy),
- rejestrowanie nowych użytkowników,
- logowanie istniejących użytkowników,
- weryfikacja nadawcy danych przy pomocy jednorazowych tokenów,
- możliwość zalogowania się administratora systemu.

3.1.4 Administrator serwera

Wymagania dla administratora zarządzającego serwerem:

- zalogowanie się na serwer,
- sprawdzenia aktualnie zalogowanych użytkowników,
- usuwanie kont użytkowników,
- sprawdzenie logów z powiadomieniami serwera,
- wyłączenia serwera.

3.2 Pozafunkcjonalne

Wymagania pozafunkcjonalne odnoszące się do całego systemu. Są to wymagania dotyczące wydajności, bezpieczeństwa i użyteczności systemu.

- system musi posiadać serwer wielowątkowy,
- serwer posiada stały, znany aplikacji klienckiej, adres IP,
- hasło służące do zalogowania administratora na serwerze musi być niewidoczne podczas wpisywania,
- aplikacja użytkownika posiada graficzny interfejs użytkownika,
- aplikacja użytkownika powinna umożliwić wybór jednego z motywów graficznych: ciemny, jasny,
- komunikacja klient-serwer jest szyfrowana przy pomocy SSL/TLSv1.2,
- komunikacja głosowa pomiędzy użytkownikami musi być szyfrowana,
- komunikacja tekstowa pomiędzy użytkownikami musi być szyfrowana,
- komunikacja odbywa się 1:1,
- czas przesyłu informacji pomiędzy użytkownikami nie powinien być dłuższy niż 2 sekundy,
- system musi działać na systemie operacyjnym Windows 10 lub nowszym,
- system powinien przechowywać hasła w postaci skrótu utworzonego funkcją SHA-256,

- system powinien obliczać czas połączenia z dokładnością do 1 sekundy,
- system wymaga połączenia internetowego o przepustowości 100kB/s (kilobajtów na sekundę) i większej,
- system nie powinien retransmitować danych dźwiękowych,
- nazwą identyfikacyjną użytkownika jest jego adres e-mail,
- nazwa identyfikacyjna użytkownika musi być unikalna,
- dane dotyczące użytkowników powinny być przechowywane w plikach XML.

4 Narzędzia, środowisko, biblioteki

Zbiór używanych do stworzenia projektu narzędzi i bibliotek. Wymienione zostały także używane środowiska programistyczne (*IDE*), które umożliwiły stworzenie całego systemu.

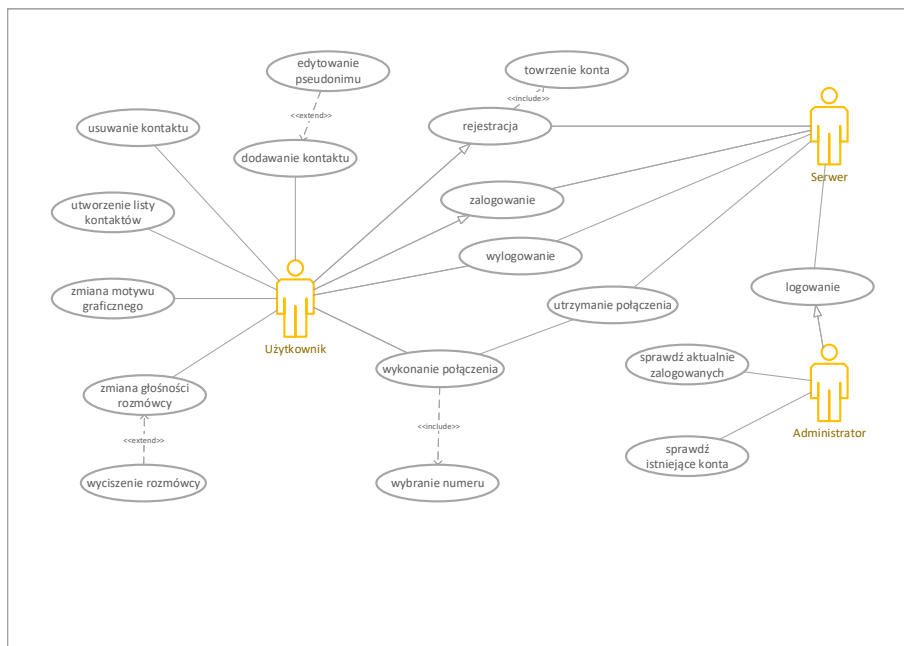
- Narzędzia
 - C#,
 - XML,
 - LINQ.
- Środowisko
 - MS Visual Studio 2015 oraz 2019,
 - RawCap,
 - Wireshark
 - MS Visio 2016
 - TeXStudio.
- Biblioteki / Standardy
 - X.509,
 - SSL/TLSv1.2.

5 Diagramy UML

Diagramy UML przedstawiające budowę i działanie systemu.

5.1 Przypadków użycia

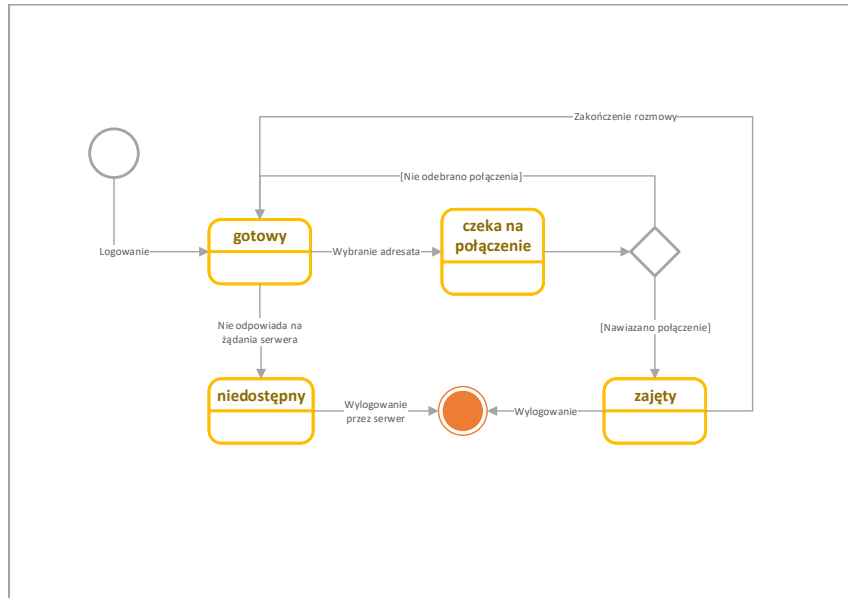
Diagram przypadków użycia z podziałem na aktorów.



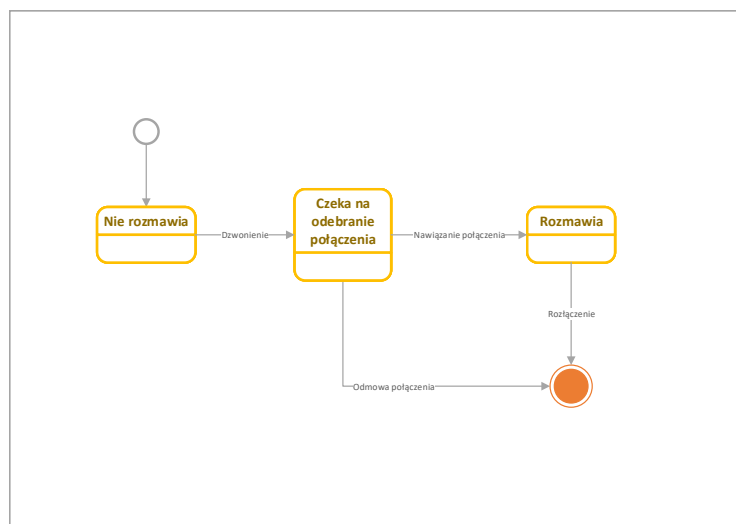
Rysunek 1: Diagram UML przypadków użycia

5.2 Stanów

Diagramy stanów: stanu użytkownika zalogowanego i rozmowy pomiędzy użytkownikami.



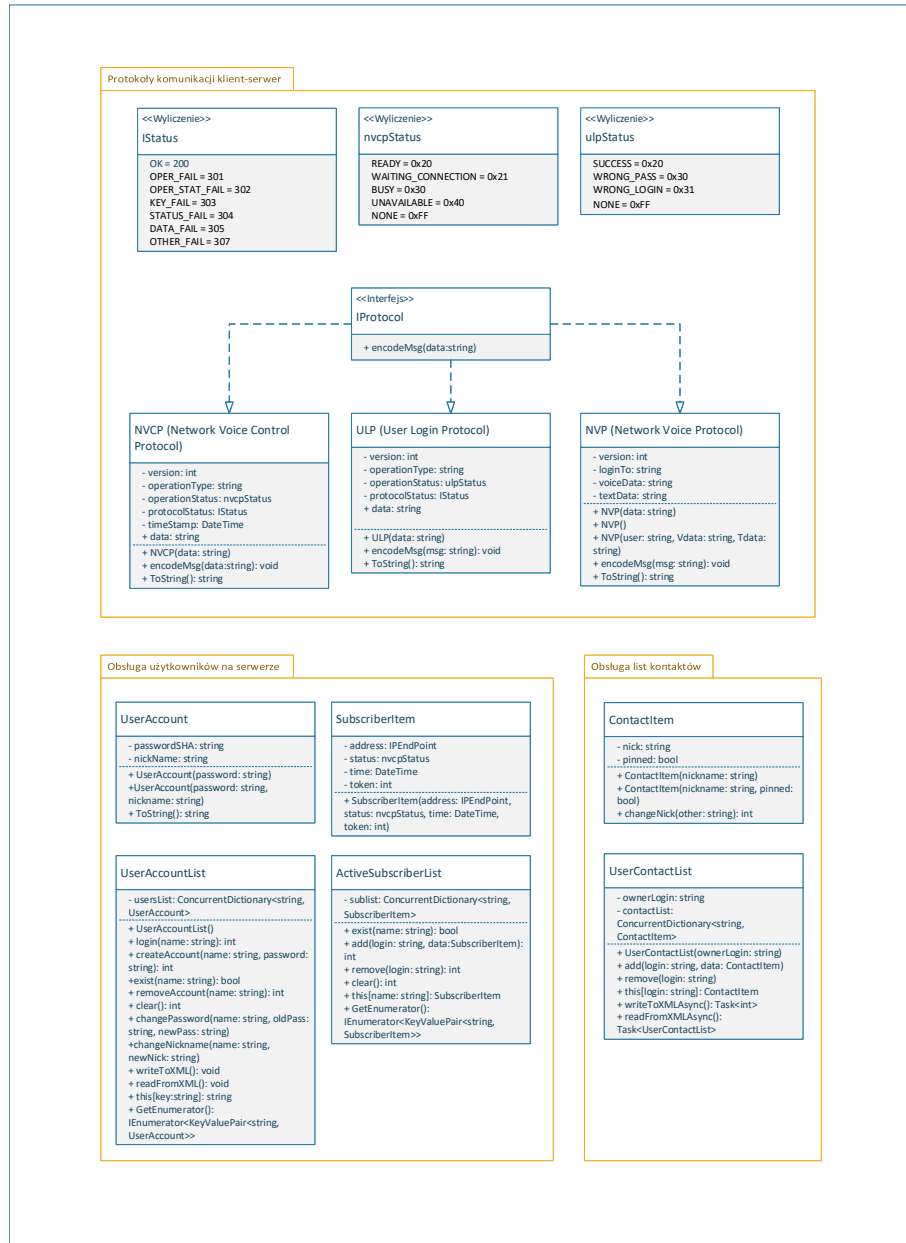
Rysunek 2: Diagram UML stanów - użytkownik zalogowany



Rysunek 3: Diagram UML stanów - rozmowa użytkownika z innym

5.3 Klas

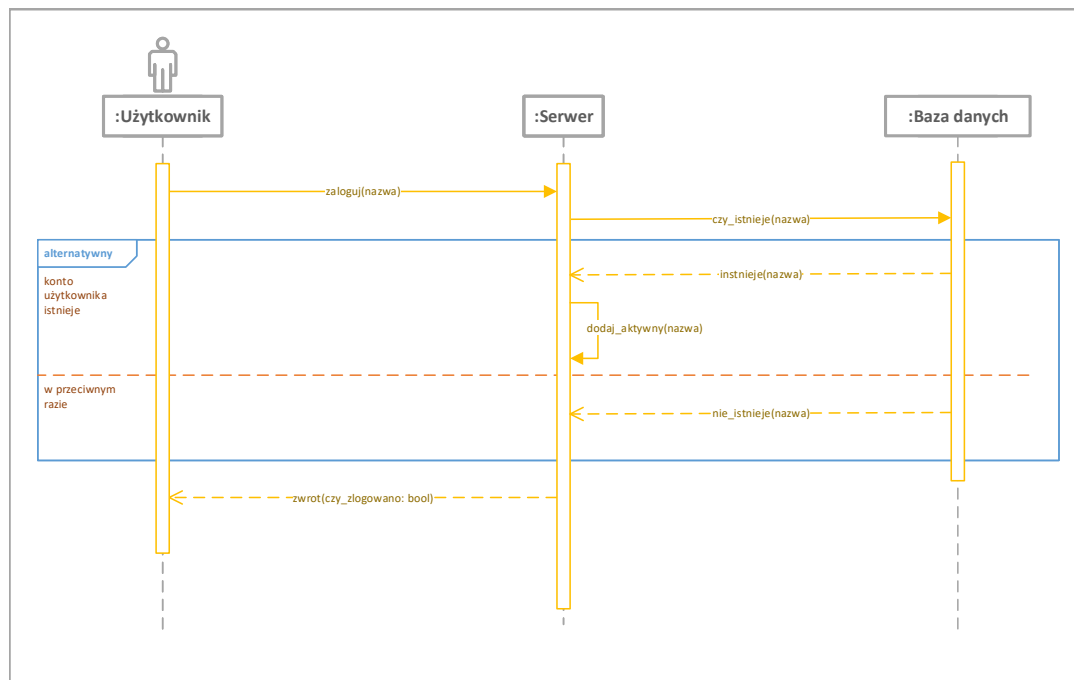
Diagramy klas protokołów, aplikacji serwera i klienta.



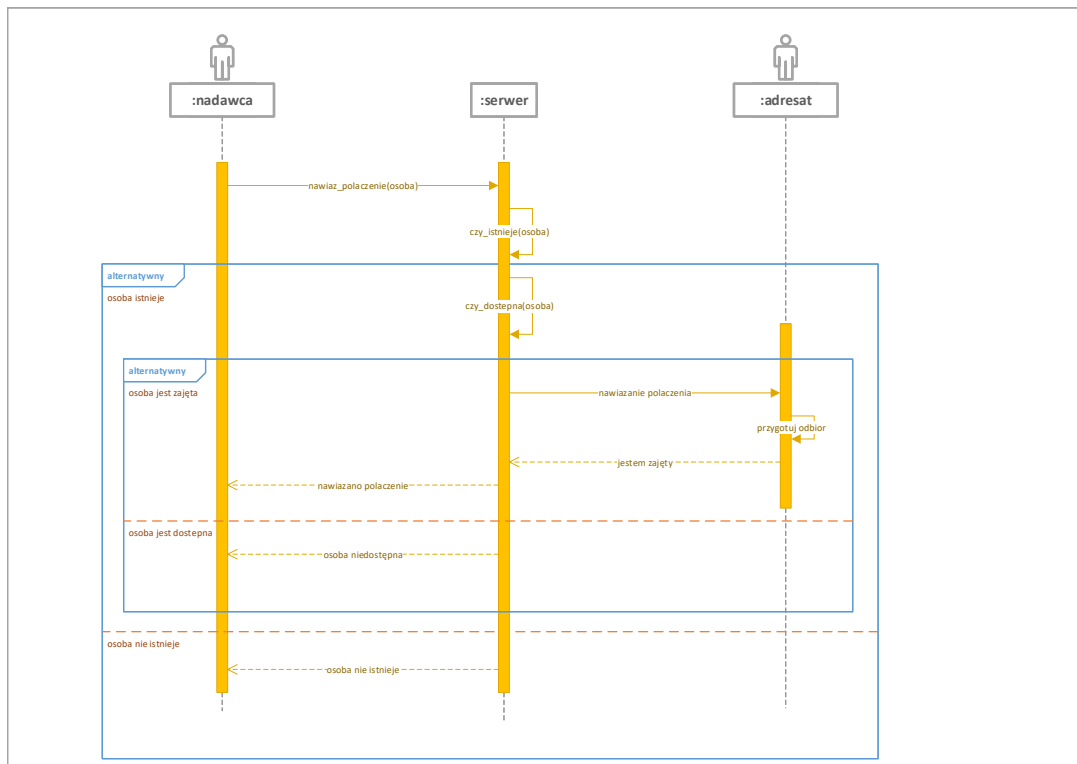
Rysunek 4: Diagram UML klas - serwer

5.4 Sekwencji

Diagramy sekwencji: próby nawiązania połączenia oraz logowania od systemu.



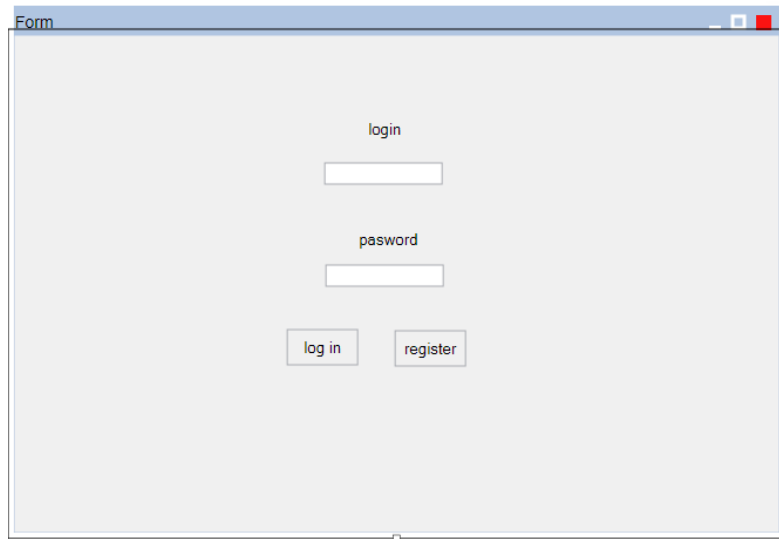
Rysunek 5: Diagram UML sekwencji - logowanie



Rysunek 6: Diagram UML sekwencji - nawiązanie połączenia

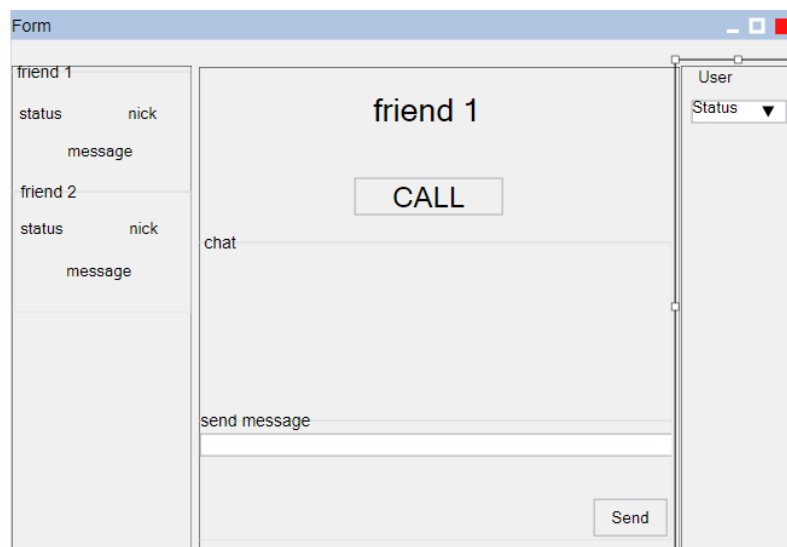
6 GUI

Projekt interfejsu graficznego.



A screenshot of a graphical user interface window titled "Form". The window has a light gray background and a standard Windows-style title bar with minimize, maximize, and close buttons. In the center of the window, there is a login form. It consists of the label "login" above a text input field, followed by the label "password" above another text input field. Below these fields are two buttons: "log in" and "register".

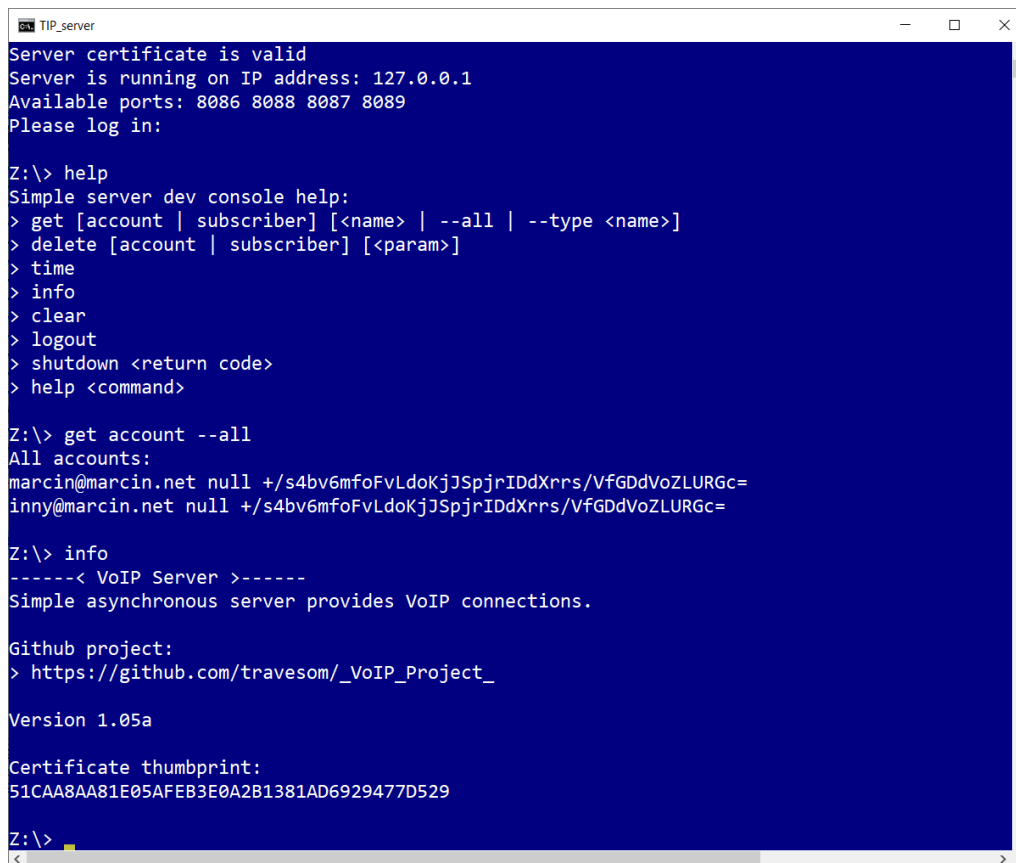
Rysunek 7: Ekran logowania



A screenshot of a graphical user interface window titled "Form", which appears to be a chat application. The window is divided into several sections. On the left, there is a sidebar with two friend lists. The first list is for "friend 1" and contains fields for "status", "nick", and "message". The second list is for "friend 2" and also contains fields for "status", "nick", and "message". In the center of the window, there is a large chat area. At the top of this area, it says "friend 1" and "chat". Below this, there is a "CALL" button. At the bottom of the chat area, there is a "send message" label, a text input field, and a "Send" button. On the right side of the window, there is a vertical panel with a "User" label and a "Status" dropdown menu.

Rysunek 8: Ekran główny

Poniżej umieszczono zrzuty ekranu konsoli serwera. Przedstawiono przykładowe komendy dostępne dla administratora po zalogowaniu się. Hasło logowania nie pojawia się przy wpisywaniu, jest ot zabieg celowy.



```
TIP_server
Server certificate is valid
Server is running on IP address: 127.0.0.1
Available ports: 8086 8088 8087 8089
Please log in:

Z:\> help
Simple server dev console help:
> get [account | subscriber] [<name> | --all | --type <name>]
> delete [account | subscriber] [<param>]
> time
> info
> clear
> logout
> shutdown <return code>
> help <command>

Z:\> get account --all
All accounts:
marcin@marcin.net null +/s4bv6mfoFvLdoKjJSpjrIDdXrrs/VfGDdVoZLURGc=
inny@marcin.net null +/s4bv6mfoFvLdoKjJSpjrIDdXrrs/VfGDdVoZLURGc=

Z:\> info
-----< VoIP Server >-----
Simple asynchronous server provides VoIP connections.

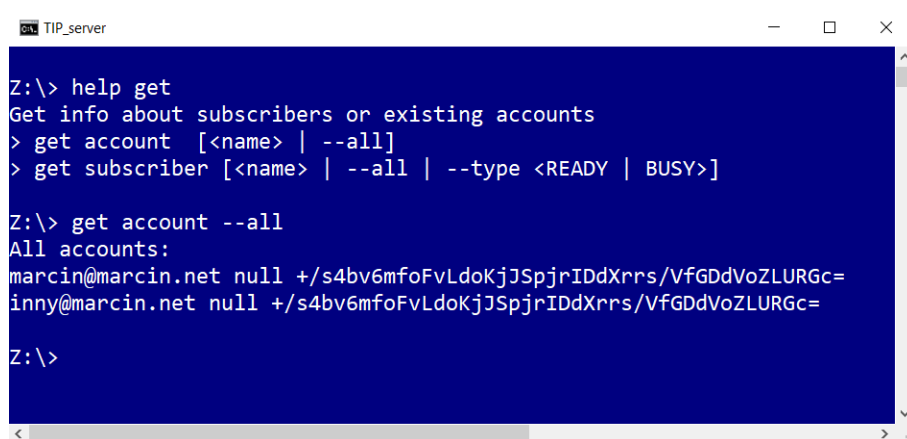
Github project:
> https://github.com/travesom/_VoIP_Project_

Version 1.05a

Certificate thumbprint:
51CAA8AA81E05AFEB3E0A2B1381AD6929477D529

Z:\>
```

Rysunek 9: Konsola serwera

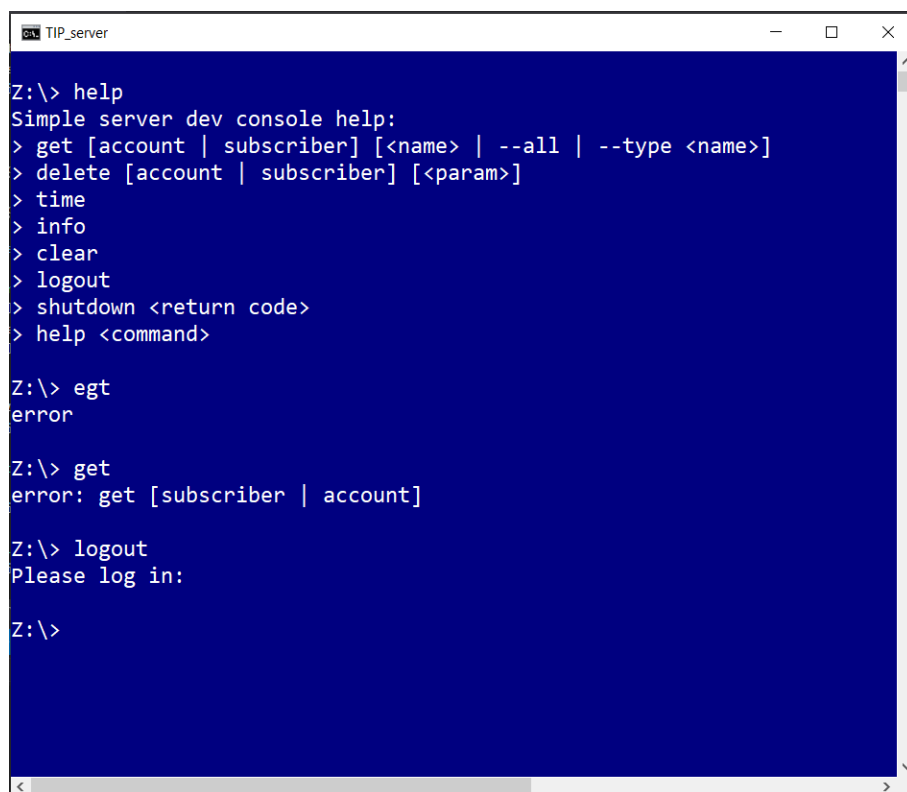


```
Z:\> help get
Get info about subscribers or existing accounts
> get account [<name> | --all]
> get subscriber [<name> | --all | --type <READY | BUSY>]

Z:\> get account --all
All accounts:
marcin@marcin.net null +/s4bv6mfoFvLdoKjJSpjrIDdXrrs/VfGDdVoZLURGc=
inny@marcin.net null +/s4bv6mfoFvLdoKjJSpjrIDdXrrs/VfGDdVoZLURGc=

Z:\>
```

Rysunek 10: Konsola serwera



```
Z:\> help
Simple server dev console help:
> get [account | subscriber] [<name> | --all | --type <name>]
> delete [account | subscriber] [<param>]
> time
> info
> clear
> logout
> shutdown <return code>
> help <command>

Z:\> egt
error

Z:\> get
error: get [subscriber | account]

Z:\> logout
Please log in:

Z:\>
```

Rysunek 11: Konsola serwera

7 Testy i przebiegi

Poniżej przedstawiono fragmenty transmisji klient-serwer. Całość przygotowanej transmisji znajduje się w pliku *tip_spr.pcap*.

```
C:\Windows\system32\cmd.exe
Uwierzytelniono
Wysłano: ver#1' oper#REGISTER' status#NONE' ulp#OK' data#marcin@marcin.net'
Odebrano: ver#1' oper#REGISTER' status#NONE' ulp#DATA_FAIL' data#''

Uwierzytelniono
Wysłano: ver#1' oper#REGISTER' status#NONE' ulp#OK' data#marcin@marcin.1234 123qwe'
Odebrano: ver#1' oper#REGISTER' status#NONE' ulp#DATA_FAIL' data#''

Uwierzytelniono
Wysłano: ver#1' oper#REGISTER' status#NONE' ulp#OK' data#marcin@marcin.net 123qwe'
Odebrano: ver#1' oper#REGISTER' status#SUCCESS' ulp#OK' data#''

Uwierzytelniono
Wysłano: ver#1' oper#LOGIN' status#NONE' ulp#OK' data#marcin@marcin.net'
Odebrano: ver#1' oper#LOGIN' status#NONE' ulp#DATA_FAIL' data#''

Uwierzytelniono
Wysłano: ver#1' oper#LOGIN' status#NONE' ulp#OK' data#marcin@marcin.net 1qaz2wsx'
Odebrano: ver#1' oper#LOGIN' status#WRONG_PASS' ulp#OK' data#''

Uwierzytelniono
Wysłano: ver#1' oper#LOGIN' status#NONE' ulp#OK' data#marcin@marcin.net 123qwe'
Odebrano: ver#1' oper#LOGIN' status#SUCCESS' ulp#OK' data#9854416'

Uwierzytelniono
Wysłano: ver#1' oper#MY_STATUS' status#READY' nvcp#OK' time#21.09.2020 14:48:22' data#marcin@marcin.net 9854416'
Odebrano: ver#1' oper#MY_STATUS' status#READY' nvcp#OK' time#21.09.2020 14:48:22' data#7312906'

Uwierzytelniono
Wysłano: ver#1' oper#MY_STATUS' status#BUSY' nvcp#OK' time#21.09.2020 14:48:22' data#marcin@marcin.net 7312906'
Odebrano: ver#1' oper#MY_STATUS' status#BUSY' nvcp#OK' time#21.09.2020 14:48:22' data#5840324'

Uwierzytelniono
Wysłano: ver#1' oper#REGISTER' status#NONE' ulp#OK' data#inny@marcin.net 123qwe'
```

Rysunek 12: Przebieg w programie testującym serwer

```
C:\Windows\system32\cmd.exe
Wysłano: ver#1' oper#REGISTER' status#NONE' ulp#OK' data#inny@marcin.net 123qwe'
Odebrano: ver#1' oper#REGISTER' status#SUCCESS' ulp#OK' data#''

Uwierzytelniono
Wysłano: ver#1' oper#LOGIN' status#NONE' ulp#OK' data#inny@marcin.net 123qwe'
Odebrano: ver#1' oper#LOGIN' status#SUCCESS' ulp#OK' data#1826232'

Uwierzytelniono
Wysłano: ver#1' oper#MY_STATUS' status#READY' nvcp#OK' time#21.09.2020 14:48:22' data#inny@marcin.net 1826232'
Odebrano: ver#1' oper#MY_STATUS' status#READY' nvcp#OK' time#21.09.2020 14:48:22' data#3582895'

Uwierzytelniono
Wysłano: ver#1' oper#AVABILITY' status#NONE' nvcp#OK' time#21.09.2020 14:48:22' data#inny@marcin.net marcin@marcin.net'
Odebrano: ver#1' oper#AVABILITY' status#BUSY' nvcp#OK' time#21.09.2020 14:48:22' data#''

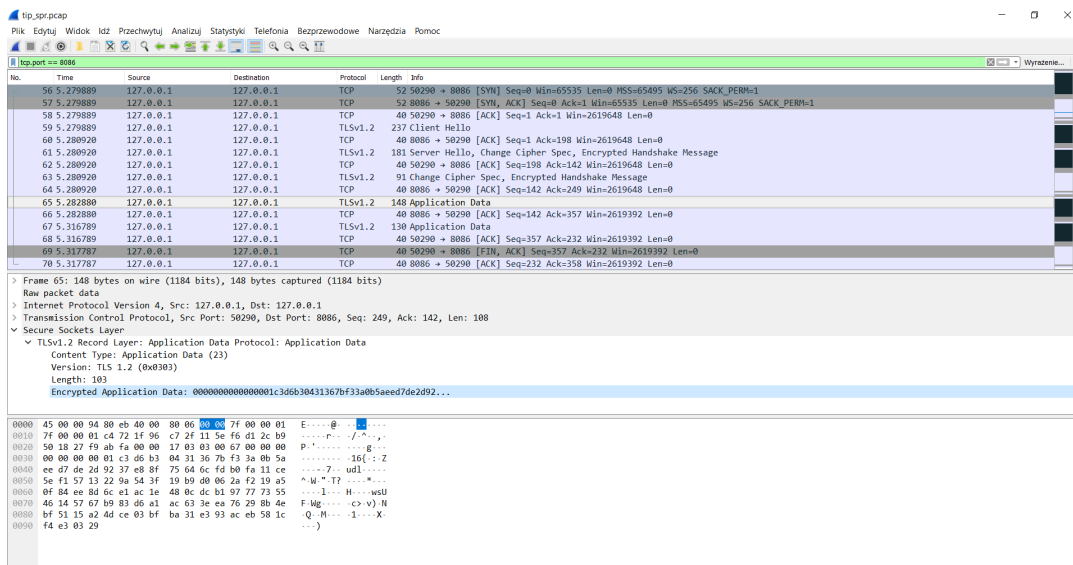
Uwierzytelniono
Wysłano: ver#1' oper#MY_STATUS' status#READY' nvcp#OK' time#21.09.2020 14:48:22' data#marcin@marcin.net 5840324'
Odebrano: ver#1' oper#MY_STATUS' status#READY' nvcp#OK' time#21.09.2020 14:48:22' data#1041386'

Uwierzytelniono
Wysłano: ver#1' oper#AVABILITY' status#NONE' nvcp#OK' time#21.09.2020 14:48:22' data#inny@marcin.net marcin@marcin.net'
Odebrano: ver#1' oper#AVABILITY' status#READY' nvcp#OK' time#21.09.2020 14:48:22' data#''

Uwierzytelniono
Wysłano: ver#1' oper#MY_STATUS' status#UNAVAILABLE' nvcp#OK' time#21.09.2020 14:48:22' data#marcin@marcin.net 1041386'
Odebrano: ver#1' oper#MY_STATUS' status#UNAVAILABLE' nvcp#OK' time#21.09.2020 14:48:22' data#2798048'

Uwierzytelniono
Wysłano: ver#1' oper#MY_STATUS' status#UNAVAILABLE' nvcp#OK' time#21.09.2020 14:48:22' data#inny@marcin.net 3582895'
Odebrano: ver#1' oper#MY_STATUS' status#UNAVAILABLE' nvcp#OK' time#21.09.2020 14:48:22' data#6027293'
```

Rysunek 13: Przebieg w programie testującym serwer



Rysunek 14: Fragment przebiegu w programie Wireshark

8 Podsumowanie

Projektu nie udało się dokończyć z powodu braku zaangażowania ze strony współautora pracy. Ukończono prace nad serwerem, protokołami komunikacyjnymi, szyfrowaniem połączeń. Brakuje aplikacji klienta i obsługi dźwięku.

9 Podział pracy

Juliusz Horowski	aplikacja klienta (w tym GUI) - nie zrobiono
	obsługa dźwięku - nie zrobiono
	sprawozdanie końcowe
Marcin Złotek	aplikacja serwera
	protokoły komunikacyjne
	mechanizm logowania i rejestracji
	szyfrowanie komunikacji klient-serwer
	sprawozdanie końcowe

9.1 Cele zrealizowane

Udało się zrealizować własne protokoły realizujące zadania obsługi użytkowników i połączeń. Jeśli chodzi o aplikacje klienta to zostało ukończone częściowo GUI oraz teoretyczne założenia dotyczące działania. Chodzi m.in. o diagramy UML komunikacji, klas itp.

9.2 Cele niezrealizowane

Nie udało się ukończyć projektu ze powodu braku współpracy ze współautorem projektu.

9.3 Problemy

Podczas tworzenia aplikacji napotkaliśmy na trudności:

- problem z implementacją szyfrowanej komunikacji z serwerem przy pomocy protokołu SSL,
- brak współpracy.

9.4 Perspektywy rozwoju

Przede wszystkim dokończenie projektu. Następnie perspektywy rozwoju, które zwiększą funkcjonalność systemu:

- rozszerzenie funkcjonalności rejestracji/logowania o możliwość odzyskiwania zapomnianego hasła,
- rozszerzenie funkcjonalności administratora po zalogowaniu.

9.5 Dokończenie samodzielne

Z powodów omówionych wcześniej, funkcjonalność systemu została ograniczona. W nowym kształcie system spełnia funkcję poczty głosowej. Można z jego pomocą nagrać dla kogoś wiadomość. Następnie ta osoba logując się może sprawdzić ilość nieodebranych wiadomości oraz je odsłuchać. Nagrania po odsłuchaniu są usuwane z serwera. Pojedynczą wiadomość można wysłać do jednej osoby.

Tylko osoba do której została wysłana wiadomość, może ją odsłuchać.

Funkcje związane z logowaniem/wylogowaniem pozostają bez zmian.