

**POLITECHNIKA POZNAŃSKA**

**INFORMATYKA**

**Voice over Internet Protocol projekt**

**Autor :**

**Cezary Czekalski**  
**cezaryczekalski@gmail.com**

**Marzec 2019**

## **Spis treści:**

|  |           |
|--|-----------|
| <b>1. Charakterystyka ogólna projektu</b>                  | <b>3</b>  |
| <b>2. Architektura systemu</b>                             | <b>3</b>  |
| <b>3. Wymagania funkcjonalne i pozafunkcjonalne</b>        | <b>5</b>  |
| <b>3.1 Wymagania funkcjonalne</b>                          | <b>5</b>  |
| <b>3.2 Wymagania pozafunkcjonalne</b>                      | <b>5</b>  |
| <b>4. Model związków encji</b>                             | <b>6</b>  |
| <b>5. Kamienie milowe</b>                                  | <b>6</b>  |
| <b>6. Narzędzia, środowisko, biblioteki, kodeki</b>        | <b>7</b>  |
| <b>7. Opis najważniejszych protokołów</b>                  | <b>7</b>  |
| <b>8. Diagram przypadków użycia</b>                        | <b>8</b>  |
| <b>9. Diagram przebiegu</b>                                | <b>9</b>  |
| <b>10. Diagram stanów</b>                                  | <b>10</b> |
| <b>11. Diagram klas</b>                                    | <b>11</b> |
| <b>12. Projekt interfejsu graficznego</b>                  | <b>13</b> |
| <b>13. Najważniejsze metody i fragmenty kodu aplikacji</b> | <b>16</b> |
| <b>14. Testy i przebieg sesji</b>                          | <b>16</b> |
| <b>15. Analiza bezpieczeństwa</b>                          | <b>17</b> |
| <b>16. Podsumowanie</b>                                    | <b>17</b> |

## 1. Charakterystyka ogólna projektu

Projekt zostanie zrealizowany poprzez stworzenie aplikacji o nazwie “Voip P2P” przeznaczonej na komputery osobiste. Realizującej transmisję głosową typu P2P(*Peer-to-Peer*), czyli polegającej na komunikacji pomiędzy dwoma osobami jednocześnie i napisaniu do niej dokumentacji.

Ze względu na zdecentralizowaną naturę P2P, zdecydowano się na zastosowanie zdecentralizowanej instancji bazy danych, a więc mieszczącej się lokalnie na każdym z komputerów użytkowników. Przy tworzeniu tabel bazy danych zostanie uwzględniona rejestracja czasów trwania rozmów pomiędzy użytkownikami jak i adresy samych rozmówców.

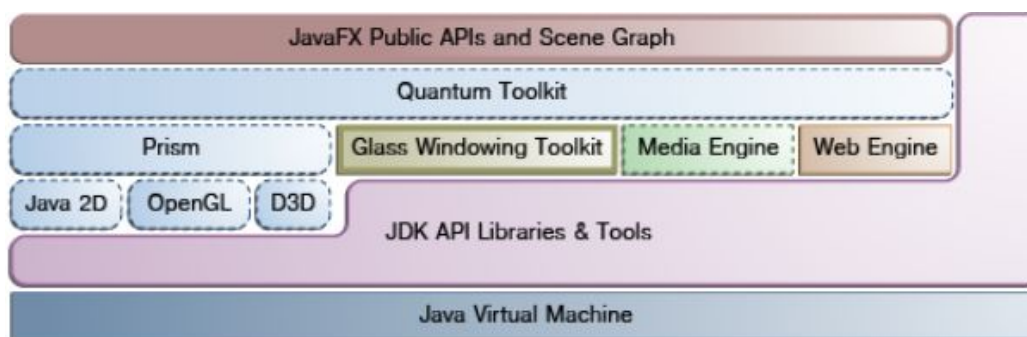
Po włączeniu “Voip P2P” ukaże się panel pozwalający na założenie lokalnego konta użytkownika lub zalogowanie się na już istniejące konto. Po zalogowaniu możliwe będzie wyszukiwanie aktywnych użytkowników w bezprzewodowej sieci lokalnej, a także dzwonenie na znane adresy, sprawdzanie historii rozmów lub oczekiwanie na przychodzące połączenia. W przypadku przychodzących połączeń odbiorca jest informowany kto do niego dzwoni poprzez wyświetlenie adresu osoby inicjującej połączenie. Przyjęto że osoba która zainicjowała połączenie jak i jego odbiorca ma możliwość zakończenia aktywnego połączenia.

## 2. Architektura systemu

Architektura aplikacji składa się z elementów napisanych w języku java z wykorzystaniem platformy JavaFX, interfejsu programistycznego aplikacji JAIN SIP API służącego do zrealizowania fazy sygnalizacji w przypadku połączenia. W skład architektury wchodzi także SQLite jako baza danych przeznaczona na przechowywane lokalnych kont użytkowników i historii rozmów.

### JavaFX

JavaFX stanowi zbiór bibliotek przeznaczonych do tworzenia zaawansowanych i nowoczesnie wyglądających aplikacji działających na wielu dostępnych współcześnie systemach operacyjnych. Pierwotnie była częścią JDK(*Java Development Kit*)8. W nowszej wersji javy stanowi już samodzielny zbiór bibliotek. JavaFX jest aktualnie rekomendowana przez firmę Oracle jako aktualne rozwiązanie w przypadku aplikacji desktopowych dla systemów operacyjnych z zainstalowanym JRE(*Java Runtime Environment*)8+ lub JDK8+. Na zdjęciu 1. poniżej przedstawione komponenty składowe platformy JavaFX.



Zdjęcie 1. Przedstawiający diagram architektury JavaFx

### JavaFX Public APIs and Scene Graph

Na górnej części zdjęcia 1 przedstawiającej stos komponentów JavaFX ukazany jest komponent JavaFX Public APIs and Scene Graph zaznaczony brązowym kolorem. JavaFX Public APIs dostarcza zbiór metod pozwalających na tworzenie wyjątkowo zaawansowanych aplikacji.

Scene Graph który jest hierarchicznym drzewem liści, którego każdy liść odpowiada widocznemu elementowi dla użytkownika końcowego. Każdy liść ma własne ID, możliwość dopasowania odpowiedniego kaskadowego arkusza stylów i wiele innych właściwości, modyfikatorów.

### Graphic System

Na zdjęciu 1. poniżej elementu JavaFX Public APIs and Scene Graph, Graphic System oznaczony kolorem niebieskim i składający się z Quantum Toolkit, Prism, Java2D, OpenGL i D3D przyspiesza renderowanie sprzętowe za pomocą renderowania programowego.

### Glass Windowing Toolkit

Kolorem beżowym na zdjęciu 1. zaznaczono komponent Glass Windowing Toolkit będący w najniższej warstwie stosu graficznego JavaFX. Jego głównym zadaniem jest dostarczanie usług takich jak zarządzanie zegarami, powierzchnie i zarządzanie oknami aplikacji. Odpowiada za połączenie między platformą JavaFX, a systemem operacyjnym na którym działa. JavaFX używa pojedynczego wątku na obsługę przerwań pochodzących od użytkownika korzystającego z aplikacji tak zwanego wątku aplikacji. Dzięki temu użytkownik nie odczuwa "zacinania się" aplikacji.

### Pulse

Pulse jest wydarzeniem i odpowiada za synchronizację stanu liści z użyciem Prism. Limit klatek na sekundę jest ustawiony na 60 FPS (*Frames per second*). Więcej szczegółów dotyczących architektury technologii javaFX można znaleźć na stronie:

<https://docs.oracle.com/javafx/2/architecture/jfxpub-architecture.htm>

## Systemem zarządzania bazą danych

Systemem zarządzania bazą danych (SZBD) jest SQLite, z logiem przedstawionym na zdjęciu 2. SZBD to oprogramowanie lub system informatyczny służący do zarządzania bazą danych. Pozwala on użytkownikom na definiowanie, tworzenie, utrzymywanie i kontrolę dostępu do bazy danych.



Zdjęcie 2. Logo SQLite

SQLite jest relacyjną bazą danych która jest darmowa zarówno w przypadku użycia komercyjnego jak i prywatnego ze względu na domenę publiczną. Jest też wbudowaną bazą danych (*embedded database*), nie używa serwera. Jest to tylko plik z którego SQLite czyta i zapisuje. Więcej informacji na <https://www.sqlite.org/about.html>

## Sygnalizacja

*The Java APIs for Integrated Networks* (JAIN) wykorzystywana do sygnalizacji w Voip wykorzystuje ono SIP jako standardowy protokół do sygnalizacji. Więcej informacji na: <https://www.oracle.com/technetwork/java/introduction-jain-sip-090386.html>

## 3. Wymagania funkcjonalne i pozafunkcjonalne

Jednym ze sposobów opisu wymagań funkcjonalnych, czyli sprecyzowanych funkcji programu jest lista wymagań przedstawiona w podpunkcie 3.1. Wymagania pozafunkcjonalne przedstawione w podpunkcie 3.2. opisują wymagania od strony technicznej jak łatwość przystosowania aplikacji do systemu operacyjnego na którym ma działać, wymagania odnośnie utrzymywania aplikacji, między innymi język w którym dany program ma być napisany, szybkość opanowania aplikacji, użyte protokoły.

### 3.1. Wymagania funkcjonalne

- Użytkownik tworzy swoje konto przez podanie loginu, hasła.
- Użytkownik loguje się na konto użytkownika.
- Użytkownik może wyszukać innych użytkowników poprzez odpytanie sieci.
- Użytkownik ma wgląd w swoją historię połączeń.
- Użytkownik który chce porozmawiać z drugim zaprasza go do rozmowy poprzez naciśnięcie przycisku odpowiadającego za rozpoczęcie rozmowy.
- Użytkownik który chce zakończyć rozmowę naciska przycisk zakończenia rozmowy.

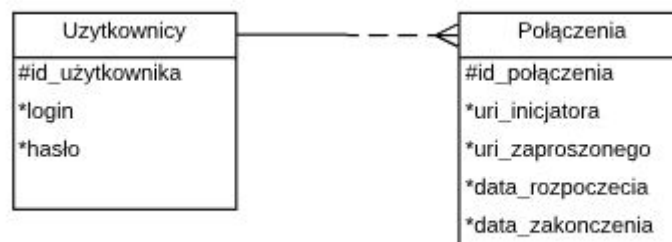
### 3.2. Wymagania pozafunkcjonalne

- Aplikacja napisana w języku Java.
- Faza komunikacji odbywa się przy użyciu udp.

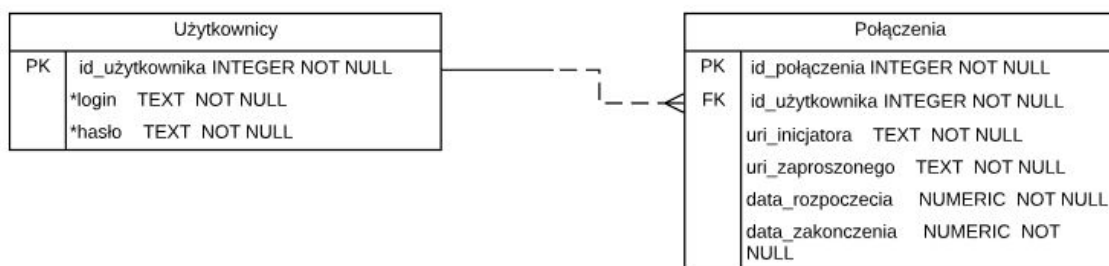
- Faza dialogu odbywa się poprzez protokół SIP.
- Aplikacja wykorzystuje bazę danych do przechowywania trwałych danych.
- Aplikacja pozwala na stworzenie lokalnych kont użytkownika.
- Aplikacja powinna być bezpieczna, dzięki wykorzystaniu bazowych technik zabezpieczeń takich jak jednokierunkowa funkcja skrótu.
- Aplikacja umożliwia jej wykorzystanie przez dwóch użytkowników jednocześnie.
- Zapewnienie działania na wersji Java >=8.
- Zapewnienie działania na Windows 10.
- Do implementacji sygnalizacji zostanie wykorzystane JAIN SIP API.
- Aplikacja pozwala na wyszukanie aktualnie aktywnych użytkowników.

#### 4. Model związków encji

Model związków encji dla lokalnego użytkownika. W tej sekcji został przedstawiony model związków encji zgodny z notacją Barkera na zdjęciu 3., natomiast model relacyjny ukazany został na zdjęciu 4.



Zdjęcie 3. Model związków encji



Zdjęcie 4. Model relacyjny

#### 5. Kamienie milowe

Kamienie milowe oznaczają znaczący postęp w tworzeniu aplikacji. W przypadku tego projektu kamienie milowe oznaczone są za pomocą listy wypunktowanej z kolejnością od początkowego do końcowego. Przekreślona linia oznacza jego wykonanie.

- ~~1. Napisanie komunikatora udp peer to peer i nawiązanie połączenia głosowego P2P jednocześnie~~
- ~~2. Napisanie modułu odpowiedzialnego za dialog~~
- ~~3. Połączenie modułów.~~
- ~~4. Testowanie i usunięcie błędów.~~
- ~~5. Dodanie GUI.~~
- ~~6. Rozszerzenie o bazę użytkowników i ich znajomych którzy też korzystają z danego komputera.~~

## ~~7. Testowanie i usunięcie błędów.~~

## 6. Narzędzia, środowisko, biblioteki, kodeki

- I. Narzędzia
  - A. IntelliJ IDEA
- II. Środowisko
  - A. Java,
  - B. sqlite
- III. Biblioteki
  - A. JavaFx
  - B. JAIN SIP API

## 7. Opis najważniejszych protokołów

### SIP(*Session Initiation Protocol*)

W celu zrealizowania funkcji dzwonienia do użytkownika końcowego zastosowany zostanie protokół SIP. Jest on przeznaczony do sygnalizacji, inicjalizacji i fazy kończenia połączenia w VOIP(*Voice Over Internet Protocol*). Protokół ten może być użyty w połączeniu z protokołami warstwy transportowej jak TCP lub UDP, lecz ze względu na brak przesyłanych potwierdzeń w UDP, niepotrzebnych w przypadku rozmowy komunikatów potwierdzających, w projekcie został wykorzystany protokół UDP.

### UDP(*User Datagram Protocol*)

Protokół UDP przeznaczony jest do przesyłania pakietów pomiędzy hostami. Nie gwarantuje dostarczenia datagramu, nie zapewnia on także retransmisji i kontroli przesyłanych datagramów. Jego prostsza budowa przekłada się jednak na większą prędkość transmisji.

W projekcie do transportu datagramów UDP został użyty protokół IPv4, przedstawiony na zdjęciu 5. poniżej. Na zdjęciu część przedstawiająca UDP jest zaznaczona kolorem szarym.

### IPv4(*Internet Protocol version 4*)

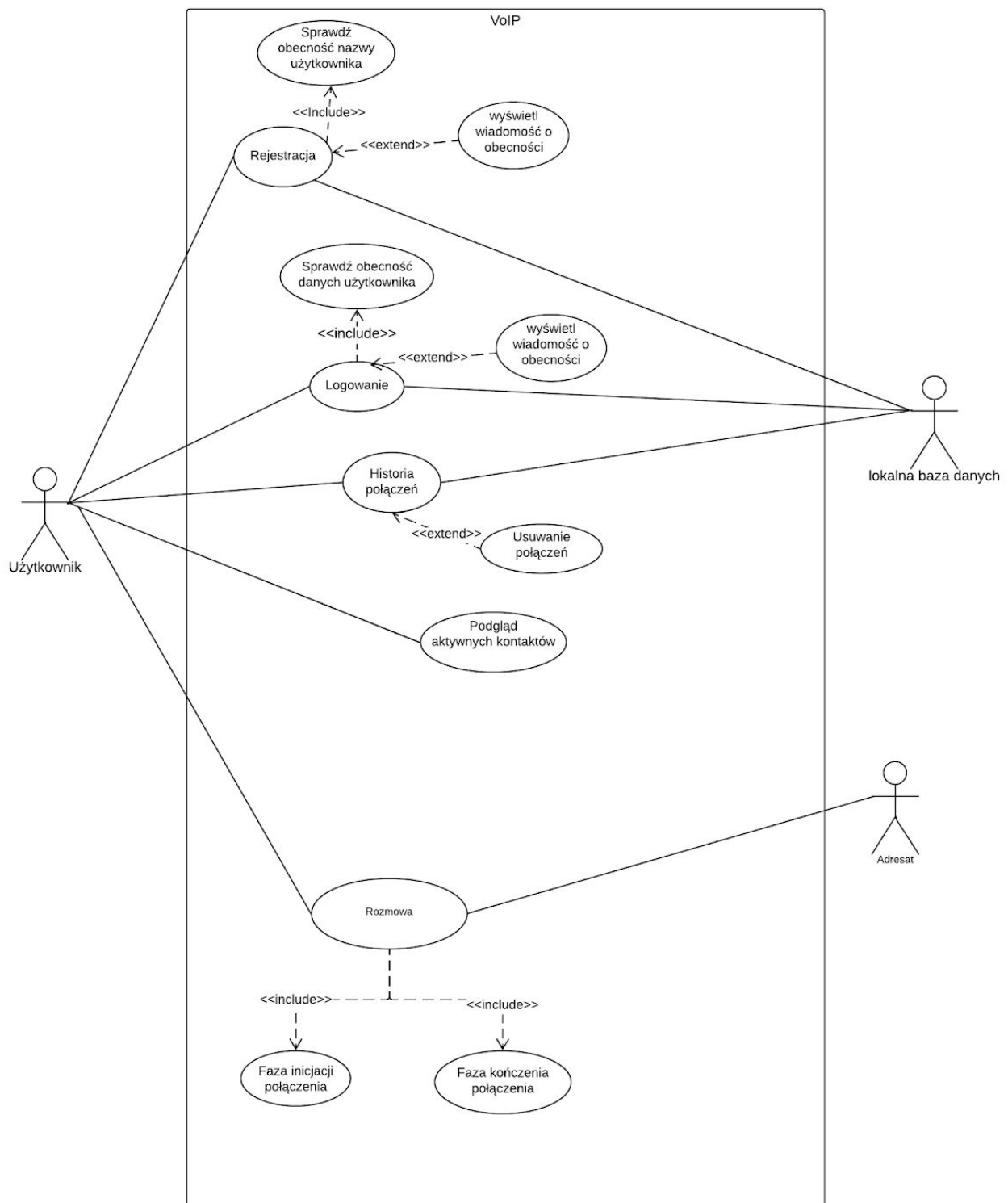
W warstwie międzysieciowej stosowany jest protokół IPv4 przeznaczony między innymi do przesyłania datagramów pomiędzy hostami w sieci. Uproszczony pakiet IPv4 transportujący datagram UDP ukazany na zdjęciu 5.

| +   | Bity 0 – 7     | 8 – 15   | 16 – 23        | 24 – 31 |
|-----|----------------|----------|----------------|---------|
| 0   | Adres źródłowy |          |                |         |
| 32  | Adres docelowy |          |                |         |
| 64  | Zera           | Protokół | Długość UDP    |         |
| 96  | Port źródłowy  |          | Port docelowy  |         |
| 128 | Długość        |          | Suma kontrolna |         |
| 160 | Dane           |          |                |         |

Zdjęcie 5. datagramu UDP wewnątrz pakietu IPv4

## 8. Diagram przypadków użycia

Diagram przypadków użycia – graficzne przedstawienie przypadków użycia, aktorów oraz związków między nimi, przedstawiony został na zdjęciu 6. Aktorzy na diagramie podzieleni są na użytkownika korzystającego z aplikacji lokalnie na swoim komputerze, adresata z którym użytkownik próbuje nawiązać połączenie i lokalną bazę danych użytkownika.

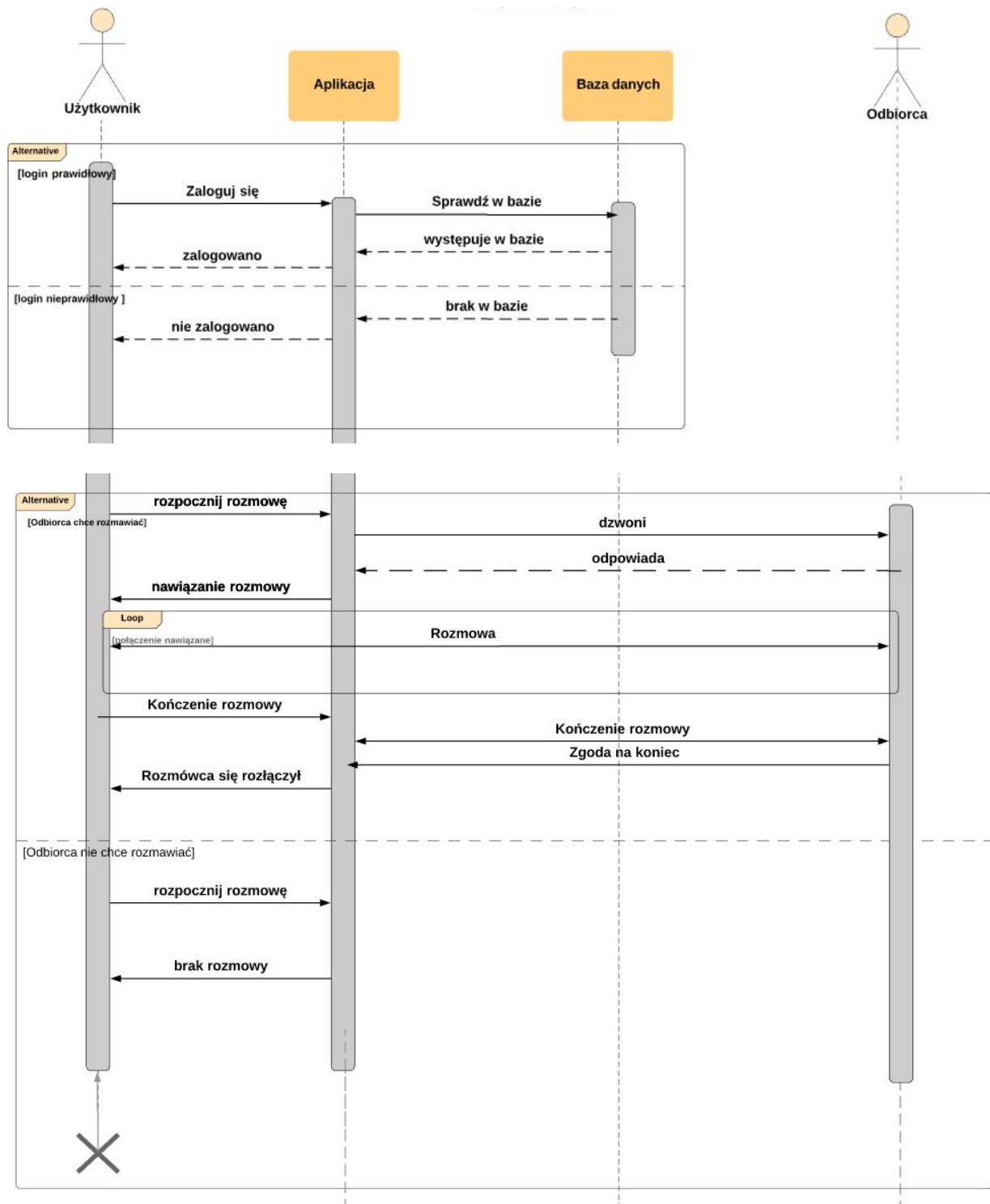


Zdjęcie 6. Diagramu UML przypadków użycia



## 9. Diagram przebiegu

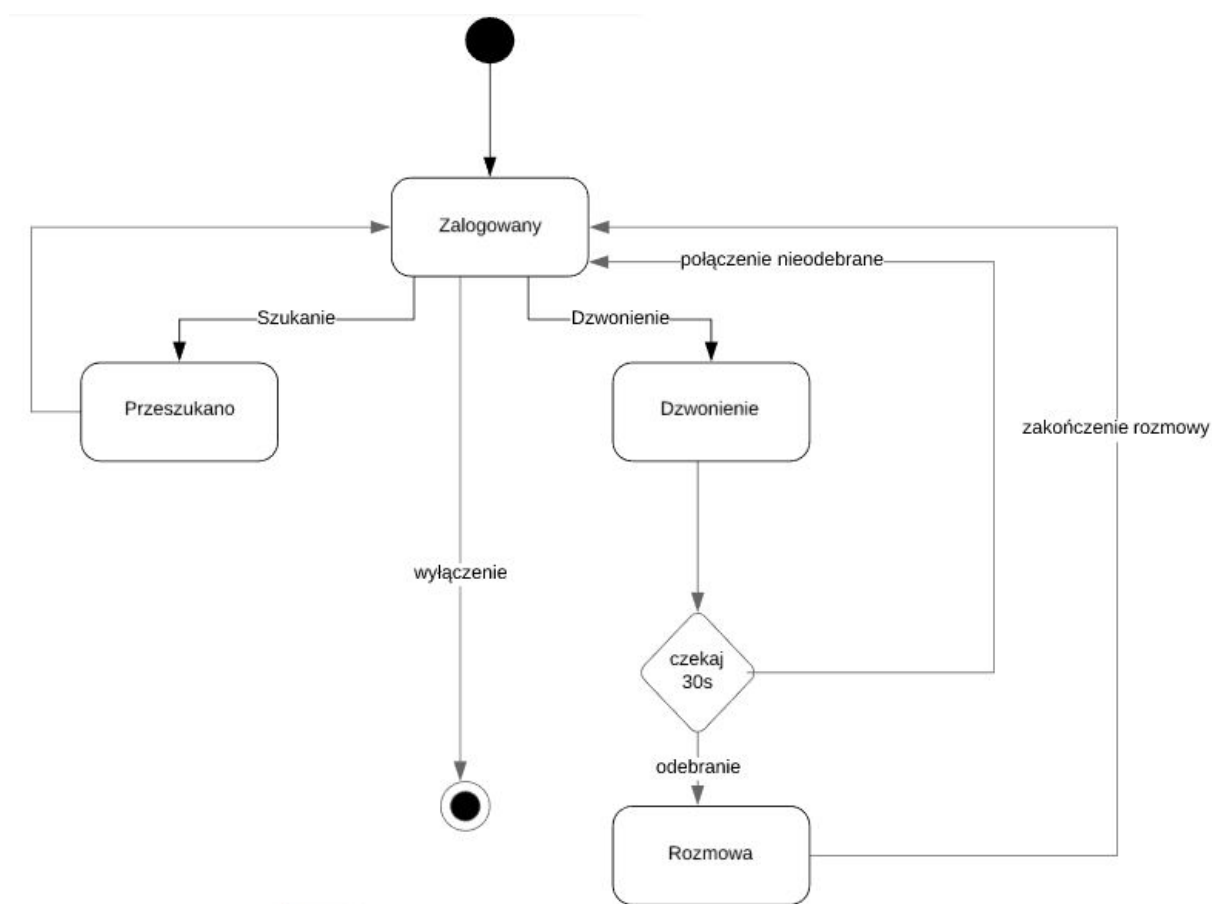
Opisuje interakcję z aplikacją i jej działanie w czasie. Na zdjęciu 7. przedstawiony jest przykładowy diagram sekwencji dla użytkownika z występującym w nim podziałem na aktorów: użytkownika korzystającego z aplikacji lokalnie na swoim komputerze i adresata z którym próbuje nawiązać połączenie



Zdjęcie 7. Diagramu przebiegu

## 10. Diagram stanów




Maszyna stanowa pokazuje przejścia stanów obiektu w danym czasie. Maszyna ta może zmienić stan sama lub spowodować zmianę danego stanu na podstawie czynników zewnętrznych. Na zdjęciu 8. przedstawiony jest przykładowy diagram stanów dla zalogowanego użytkownika.











Zdjęcie 8. Diagramu stanów dla zalogowanego użytkownika

## 11. Diagram klas

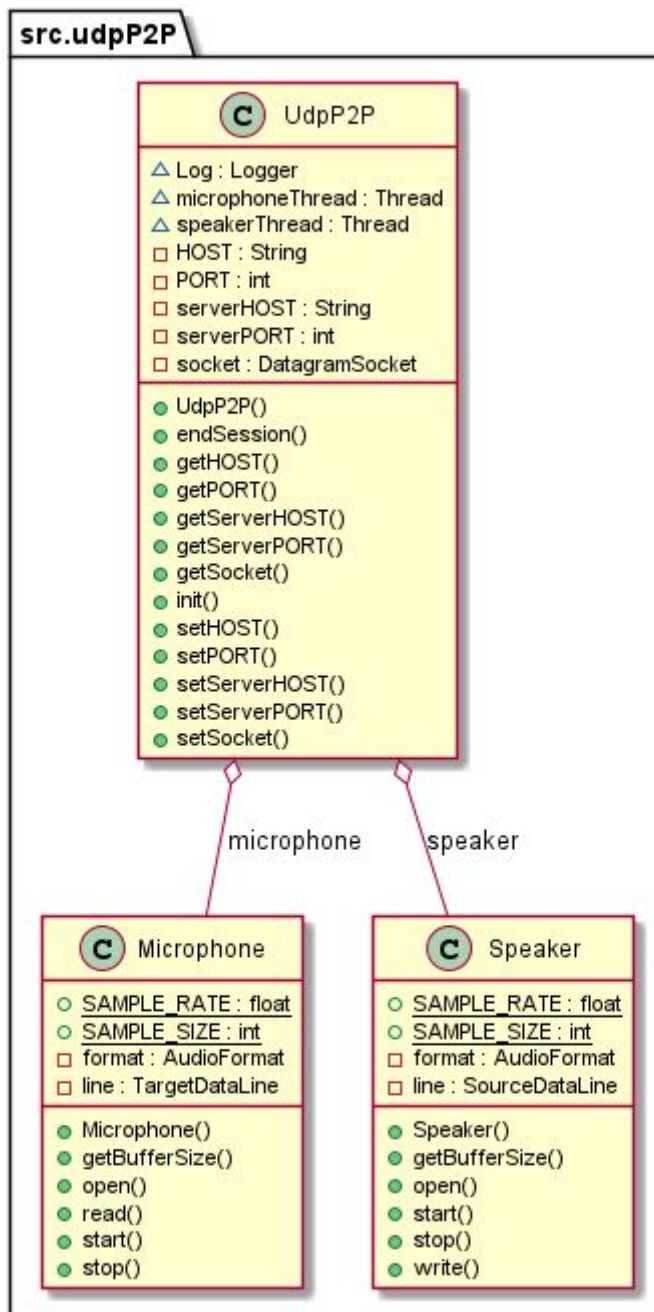
Diagramy klas służą do łatwego zobrazowania powiązań między klasami, ich metodami i zmiennymi. Diagram klas przedstawiony na zdjęciu 10. został wygenerowany przy użyciu pluginu SketchIt! dostępnego w IDE IntelliJ. Legenda objaśniająca symbole narzędzia SketchIt! przedstawiona została poniżej na zdjęciu 9.

| Type        | Symbol | Drawing   |
|-------------|--------|---|
| Extension   | < --   |  |
| Composition | *--    |  |
| Aggregation | o--    |  |

| Character | Icon for field  | Icon for method   | Visibility      |
|-----------|---|---|-----------------|
| -         |    |    | private         |
| #         |    |    | protected       |
| ~         |   |   | package private |
| +         |  |  | public          |

Zdjęcie 9. Symboli diagramu klas w SketchIt!

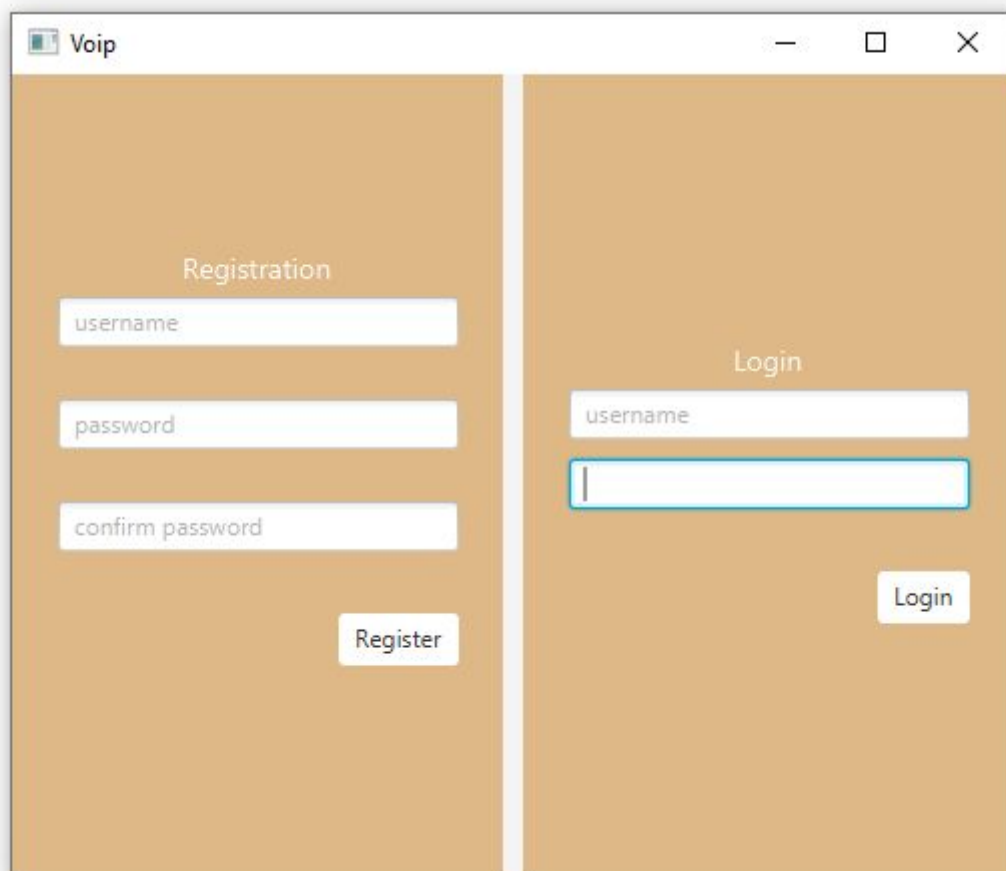
Na zdjęciu 10. przedstawiony został moduł odpowiadający za połączenie P2P UDP. Wewnątrz modułu pomiędzy klasami UdpP2P, Microphone i klasami UdpP2P, Speaker występuje relacja agregacji, czyli silniej wiążąca relacja asocjacji. W relacji agregacji rombem koloru tła z czerwonym obwodem wskazano właściciela, a więc klasę UdpP2P. Obiektami podrzędnymi relacji są klasy Microphone i Speaker związane czasem istnienia z właścicielem.



Zdjęcie 10. Diagramu klas przedstawiającego moduł UDP

## 12. Projekt interfejsu graficznego

Interfejs graficzny został wykonany z użyciem narzędzia SceneBuilder który służy do tworzenia layoutów JavaFX w sposób łatwiejszy niż pisanie ich ręcznie. Panel loginu i rejestracji ukazany na zdjęciu 11. pozwala na założenie lokalnego konta użytkownika lub zalogowanie się na już istniejące konto.

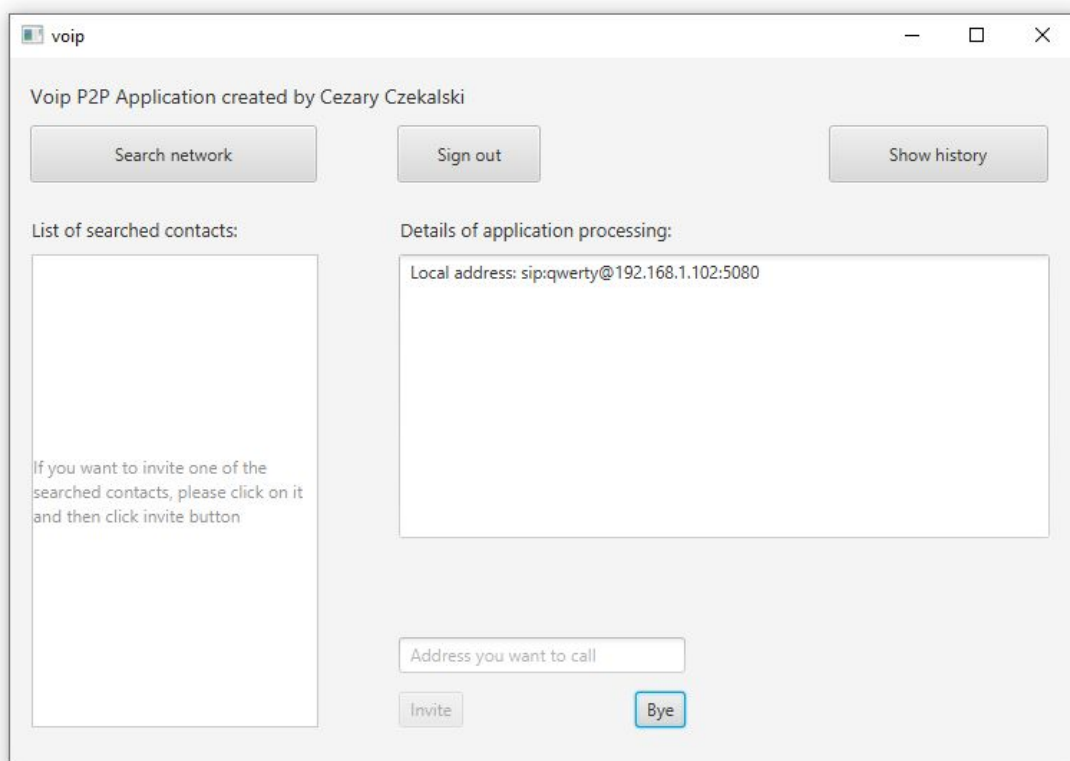


Zdjęcie 11. Panelu loginu i rejestracji

Po zalogowaniu ukaże się panel główny przedstawiony na zdjęciu 12. umożliwiający wyszukanie aktywnych użytkowników w bezprzewodowej sieci lokalnej poprzez naciśnięcie przycisku o nazwie "Search Network".

Próba nawiązania połączenia z użytkownikiem którego adres jest znany możliwa jest poprzez jego wpisanie w polu z podpowiedzią "Address you want to call" i naciśnięcie przycisku o nazwie "Invite". Innym sposobem jest kliknięcie na listę z aktywnymi użytkownikami i naciśnięcie przycisku "Invite".

Innymi usługami udostępnionymi za pomocą panelu głównego są sprawdzenie historii rozmów poprzez przycisk "Show History", kończenie rozmowy przyciskiem "Bye" lub oczekiwania na przychodzące połączenia.



Zdjęcie 12. Panelu głównego

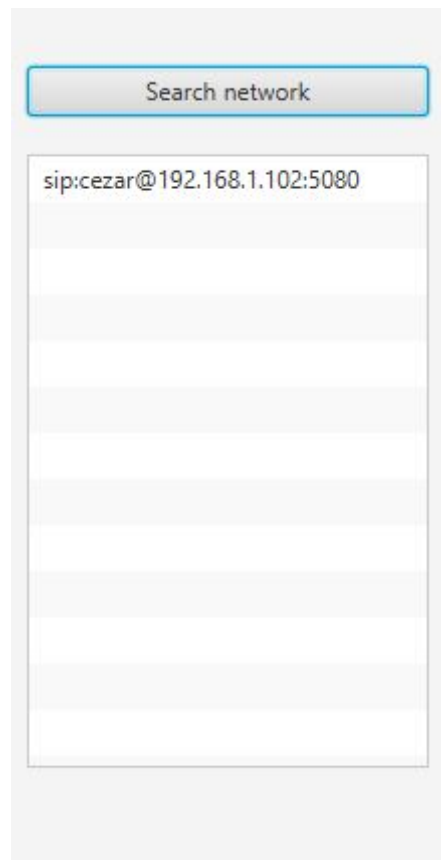
Po naciśnięciu przycisku “Show History” ukaże się panel o nazwie “connection history” przedstawiony na zdjęciu 13 z kolumnami opisanymi poniżej w tabeli 1.

Tabela 1. Przedstawiająca nazwy kolumn i ich znaczenie

| Nazwy kolumn | Znaczenie                |
|--------------|--------------------------|
| #            | ponumerowane krotki      |
| user_id      | id usera w bazie danych  |
| uri_sender   | adres wysyłającego       |
| uri_invited  | adres zaproszonego       |
| begin_date   | data rozpoczęcia rozmowy |
| end_data     | data zakończenia rozmowy |



Wynik wyszukiwania aktywnych użytkowników w bezprzewodowej sieci lokalnej po naciśnięciu przycisku o nazwie "Search Network" ukazany jest w tabeli na zdjęciu 15. Tabela wchodzi w skład głównego panelu przedstawionego na zdjęciu 12.



Zdjęcie 15. Panelu wyszukiwania aktywnych użytkowników

### 13. Najważniejsze metody i fragmenty kodu aplikacji

Najważniejsze metody przesłonięte w JAIN SIP:

```
public void processRequest(RequestEvent requestEvent)
```

która przetwarza prośby UAC lub UAS

```
public void processResponse(ResponseEvent responseEvent)
```

która przetwarza odpowiedzi UAC lub UAS

Z modułu UDP:

```
public void init()
```

całkowicie autorska metoda odpowiadająca za przesyłanie pakietów UDP

### 14. Testy i przebieg sesji

Przebieg dialogu SIP przetestowany przy użyciu programu wireshark przedstawiony na zdjęciu 16. Celowo pominięto przesyłane datagramy UDP podczas połączenia między dwoma hostami aby nie zaciemniać zaobserwowanego przebiegu .



rozмова.pcapng

Plik Edytuj Widok Idź Przechwytyj Analizuj Statystyki Telefonia Bezprzewodowe Narzędzia Pomoc

| No. | Time      | Source       | Destination  | Protocol | Length | Info   |
|-----|-----------|--------------|--------------|----------|--------|--|
| 29  | 8.557393  | 192.168.0.13 | 192.168.0.11 | SIP      | 419    | Request: INVITE sip:cezar@192.168.0.11:5080, in-dialog |
| 30  | 8.561240  | 192.168.0.11 | 192.168.0.13 | SIP      | 378    | Status: 180 Ringing                                    |
| 31  | 8.562040  | 192.168.0.11 | 192.168.0.13 | SIP      | 373    | Status: 200 OK   |
| 32  | 8.883278  | 192.168.0.13 | 192.168.0.11 | SIP      | 407    | Request: ACK sip:192.168.0.11:5080                     |
| 721 | 24.854476 | 192.168.0.13 | 192.168.0.11 | SIP      | 373    | Request: BYE sip:192.168.0.11:5080                     |
| 722 | 24.858343 | 192.168.0.11 | 192.168.0.13 | SIP      | 370    | Status: 200 OK   |

<

> Frame 29: 419 bytes on wire (3352 bits), 419 bytes captured (3352 bits) on interface 0

> Ethernet II, Src: HonHaiPr\_3b:65:16 (0c:60:76:3b:65:16), Dst: IntelCor\_33:c6:94 (9c:4e:36:33:c6:94)

> Internet Protocol Version 4, Src: 192.168.0.13, Dst: 192.168.0.11

> User Datagram Protocol, Src Port: 5082, Dst Port: 5080

> Session Initiation Protocol (INVITE)

> Request-Line: INVITE sip:cezar@192.168.0.11:5080 SIP/2.0

> Message Header

> Call-ID: ac1a01e940e3dbe22250e4fdfe9d04af@192.168.0.13

> [Call-ID: ac1a01e940e3dbe22250e4fdfe9d04af@192.168.0.13]

> CSeq: 1 INVITE

> From: <sip:192.168.0.13:5082>;tag=1123065289

> To: <sip:cezar@192.168.0.11:5080>;tag=-1712289241

> Via: SIP/2.0/UDP 192.168.0.13:5082;branch=z9hG4bK-313531-ab01beea7c02ca61cb301b4071b015c8

> Max-Forwards: 70

> Contact: <sip:192.168.0.13:5082>

> Content-Length: 0

Zdjęcie 16. Przebiegu sesji pomiędzy dwoma komputerami

## 15. Analiza bezpieczeństwa

W rozdziale dotyczącym analizy bezpieczeństwa zostały ukazane mechanizmy ochrony danych, potencjalne problemy dotyczące podsłuchu transmisji głosowej i informacji zapisanych lokalnie na dysku komputera użytkownika korzystającego z aplikacji “Voip P2P”. Z informacji przechowywanych lokalnie na dysku zostały zaszyfrowane adresy użytkowników występujące w tabeli historii połączeń. Do ich szyfrowania został użyty symetryczny szyfr blokowy AES(*Advanced Encryption Standard*) w trybie CBC(*Cipher Block Chaining*). Reszta danych w tabeli nie jest zaszyfrowana, przez co osoba mająca dostęp do danego fragmentu dysku na którym znajduje się instancja bazy danych jest w stanie zobaczyć czasy trwania rozmów i przyporządkować je do konta użytkownika. Osoba ta nie może jednak zobaczyć adresu dzwoniącego i adresata.

Hasło konta użytkownika jest zapisane w bazie danych w formie wyniku działania funkcji skrótu SHA-512, login nie jest szyfrowany. Rozmowa nie jest szyfrowana przez co istnieje możliwość jej podsłuchania przez hosty będące w jednej sieci WLAN. W Perspektywie rozwoju jest jednak zapewnienie jej szyfrowania.

## 16. Podsumowanie

Podział prac:

- Projekt był tworzony wyłącznie przez Cezarego Czekalskiego przez co nie było podziału prac. Sama praca została podzielona na kamienie milowe każdy oznaczający znaczący postęp.

Cele zrealizowane i niezrealizowane:

- Wszystkie wyznaczone cele zostały zrealizowane. Postawione cele można zobaczyć w wymaganiach funkcjonalnych i pozafunkcjonalnych.

Perspektywa rozwoju:

- Stworzenie instalatora który pomoże osobom w łatwiejszym użytkowaniu aplikacji.
- Szyfrowanie transmisji danych i zastosowanie kodeka zmniejszającego rozmiar transmitowanych danych.