Uniwersytet Przyrodniczo – Humanistyczny
w Siedlcach

Kierunek Informatyka

Patrycja Zajączkowska

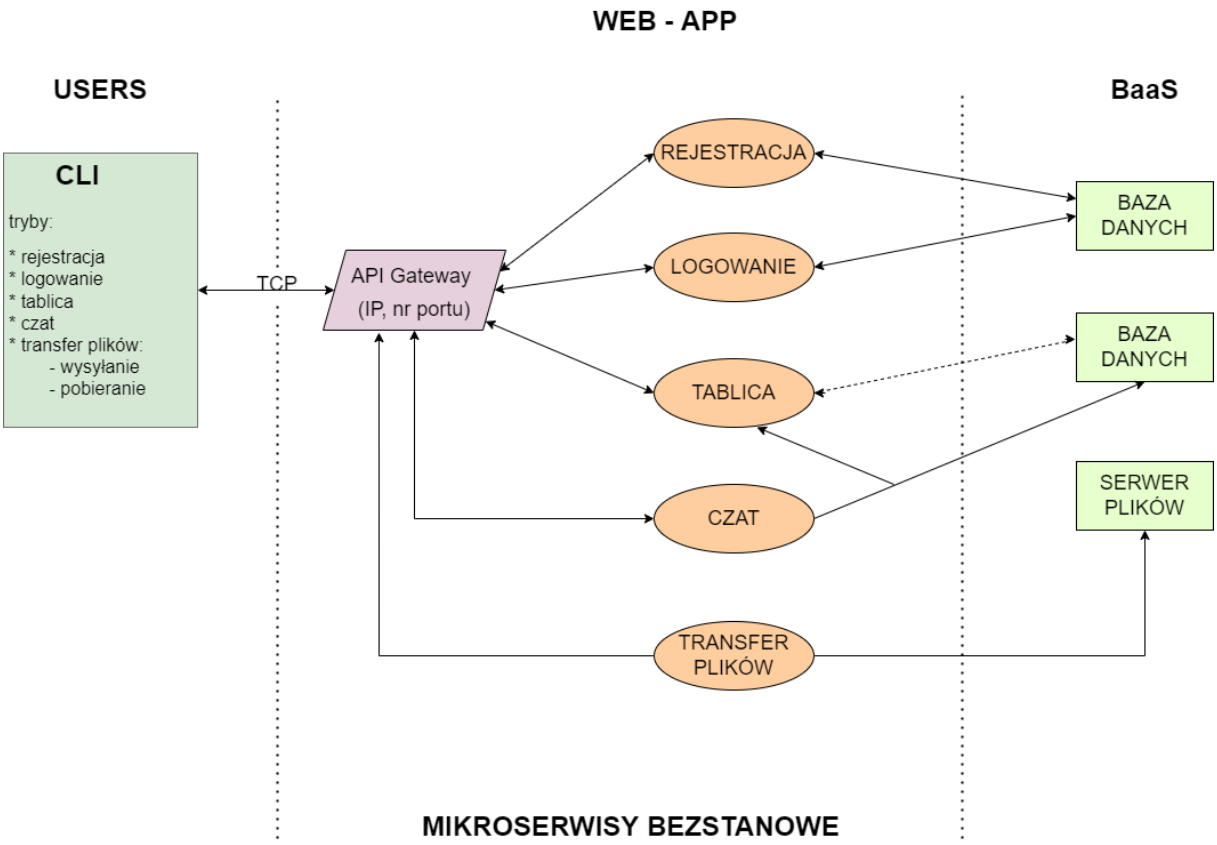
Dokumentacja aplikacji sieciowej w architekturze chmury

Praca wykonana pod kierunkiem
dr hab. prof. ucz. Stanisława Ambroszkiewicza

Siedlce, 2022r.

Spis treści

Schemat aplikacji



Schemat 1. Schemat aplikacji

Omówienie elementów aplikacji

- USERS – wiersz poleceń (command-line interface, CLI) o funkcjonalności (trybach):
    1. Rejestracja nowego użytkownika
    2. Logowanie użytkownika
    3. Dodawanie postów (czat)
    4. Wyświetlanie tablicy postów (czatu)
    5. Transfer plików:
        a) Wysyłanie pliku o podanej ścieżce dostępu na specjalny serwer dyskowy do kartoteki ogólnej.
        b) Pobieranie pliku o podanej nazwie i umieszczanie go w „folderze domowym” klienta na serwerze.

Ponadto, po wpisaniu polecenia *man*, CLI zwróci instrukcję obsługi poleceń, które możemy uruchomić na terminalu.

- API Gateway – uproszczony serwer http, bezstanowy (przekazujący strumienie bajtów tam gdzie trzeba, tj. do mikroserwisów oraz z mikroserwisów do klientów). Utrzymuje połączenia z klientami przy pomocy wątków.

- Mikroserwisy są bezstanowe, przekazują dane (bajty) dalej albo do końcowych usług magazynowych (bazy danych, dyski).

Mikroserwisy:
  - do oglądania tablicy (zamieszczonych postów; bez konieczności logowania)
  - czat (tylko zalogowani użytkownicy): dodawanie nowych wpisów do tablicy
  - rejestracja
  - logowanie
  - transfer plików: upload lub download wskazanych przez użytkownika plików.

- BaaS (Backend as a Service):
  - bazy danych: dane zarejestrowanych użytkowników, baza postów czatowych, baza URL-ów
  - serwer dyskowy z plikami (URL-e w bazie danych)

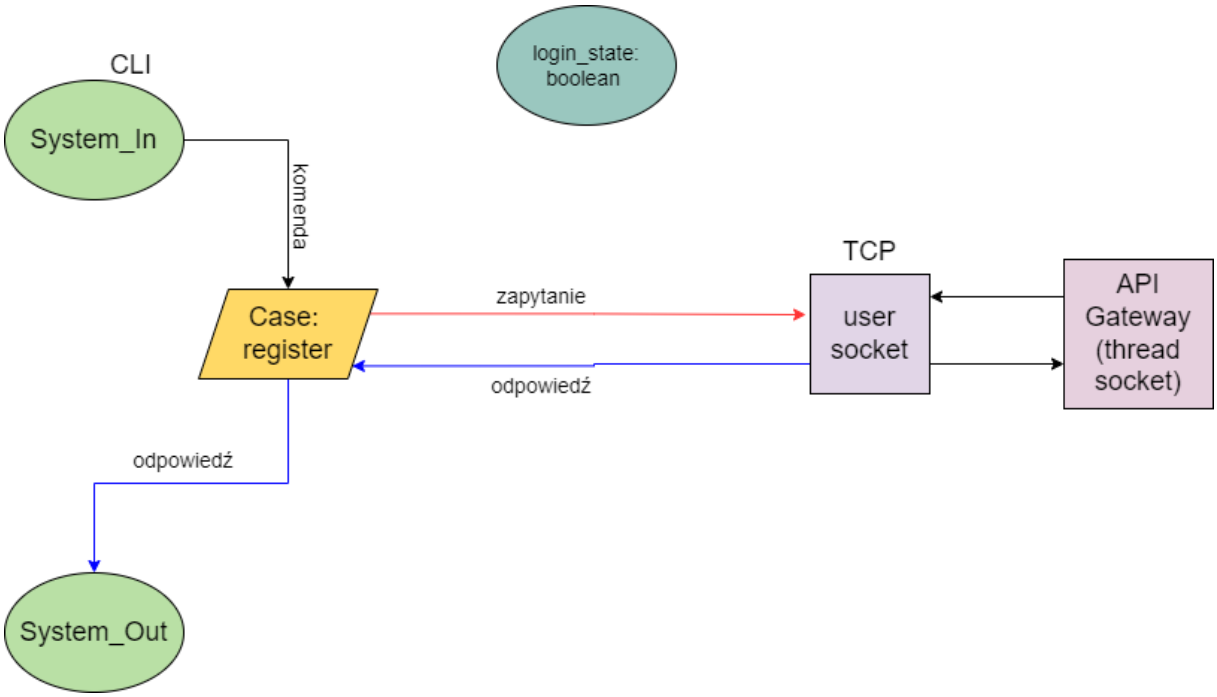
Protokoły

Zapytanie	Odpowiedź
Type: register User_name: … User_password: …	Type: register State: true/false Msg: …
Type: login User_name: … User_password: …	Type: login State: true/false Msg: …
Type: chat User_name: … message: …	Type: chat State: true/false Msg: …
Type: table	Type: table State: true/false Msg: posts
Type: file_transfer **Mode: download** User_name: … File_name: … Offset: …	Type: download File_name: … File_length: … offset: … File_content: … Msg: …
Type: file_transfer **Mode: upload** File_name: … File_length: … offset: … File_content: …	Type: upload State: true/false Msg: …

Tabela 1. Protokoły: zapytania i odpowiedzi

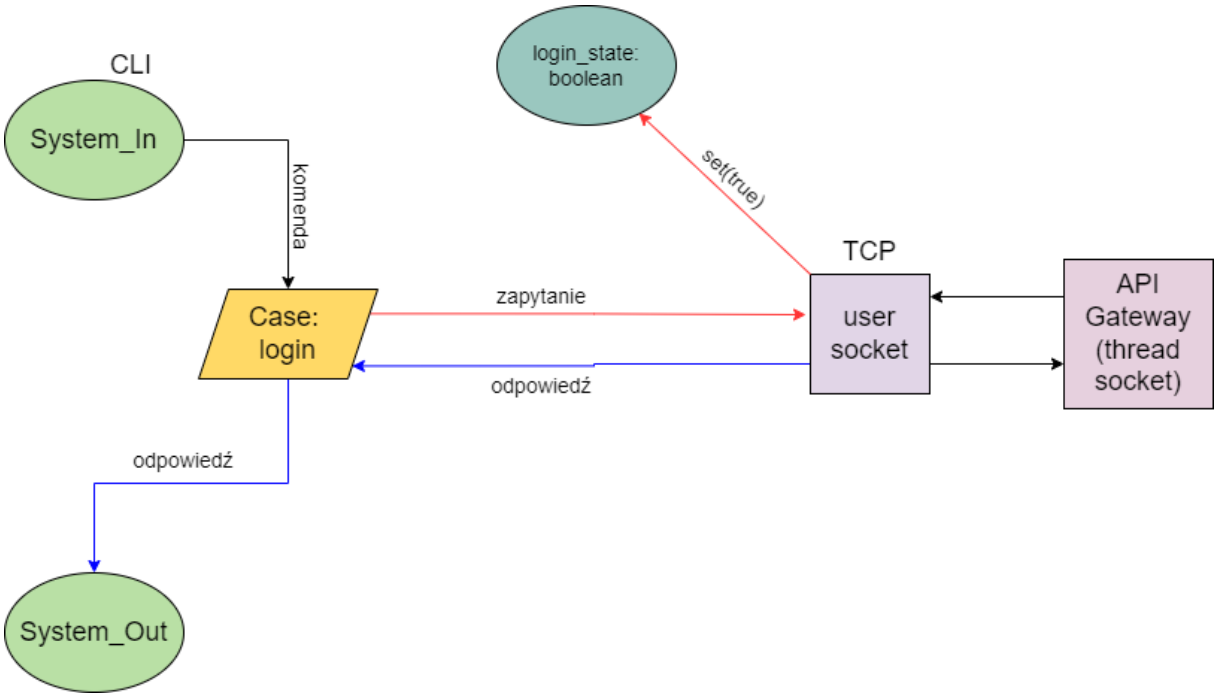
Schematy realizacji zapytań

I. rejestracja



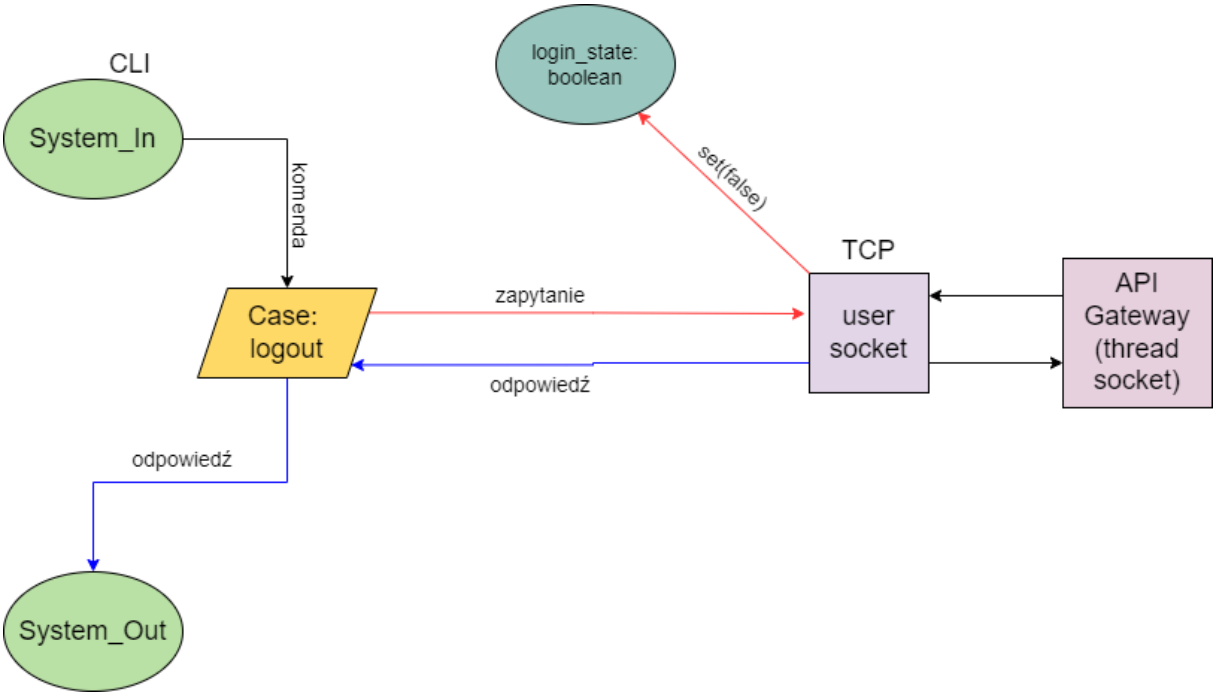
Schemat 2. rejestracja

II. logowanie



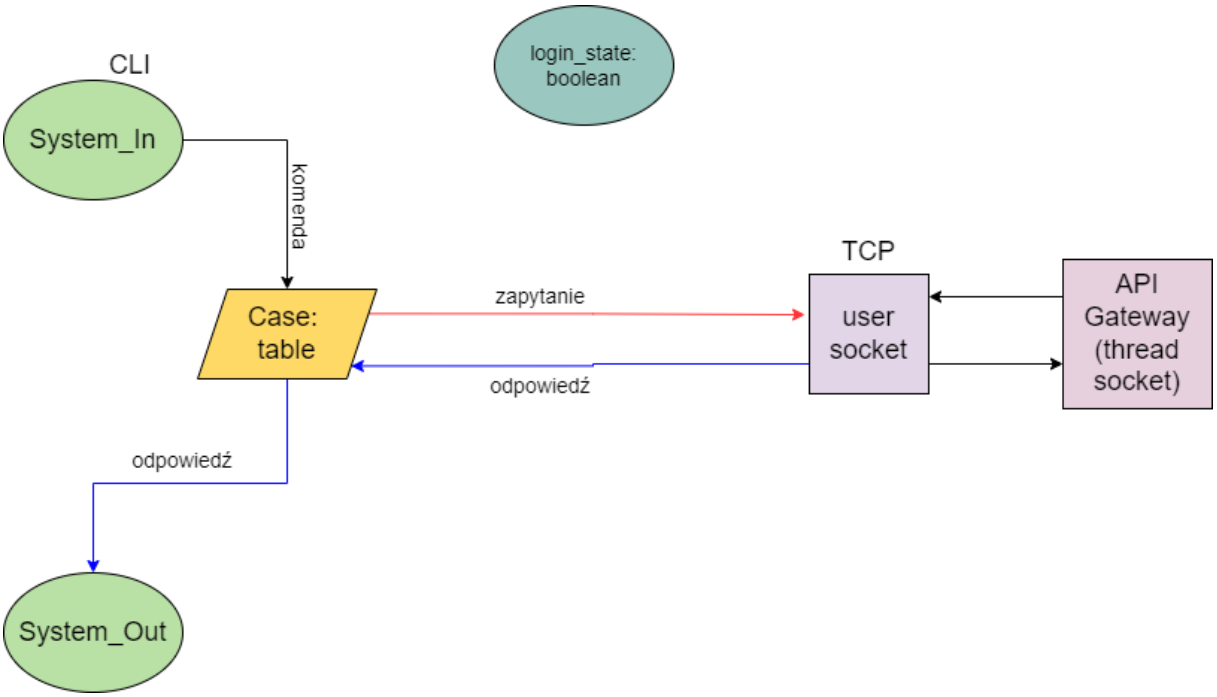
Schemat 3. logowanie

III. wylogowanie



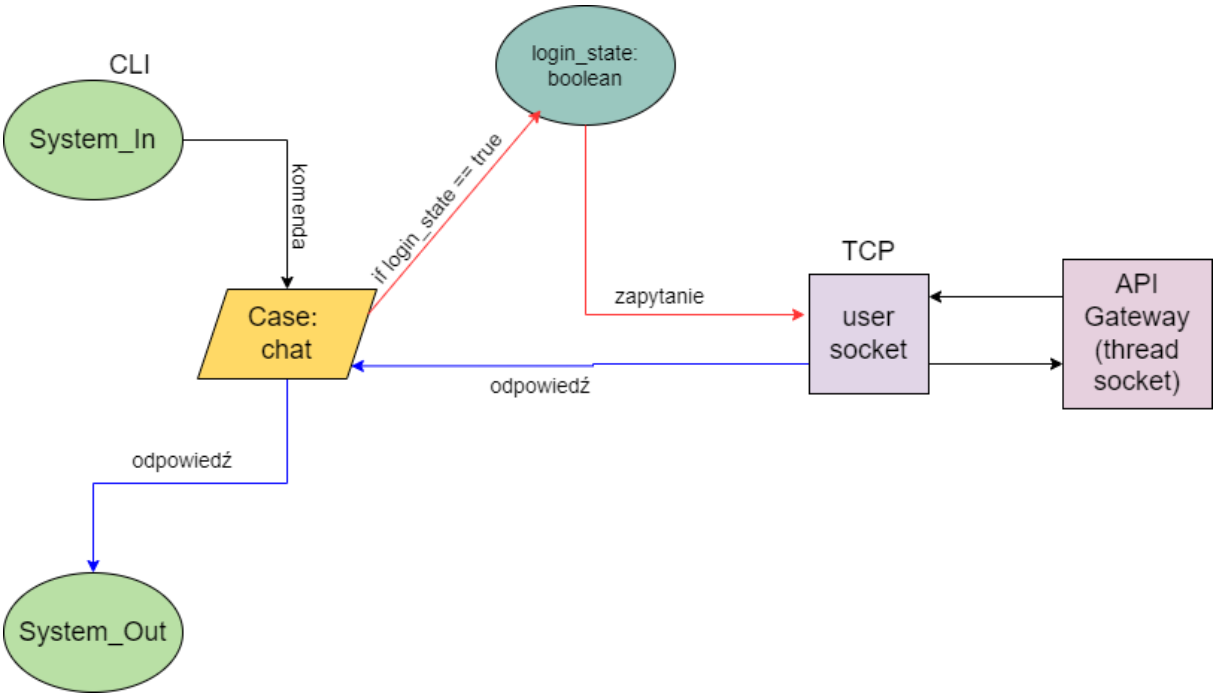
Schemat 4. wylogowanie

IV. wyświetlanie tablicy 10 ostatnich postów



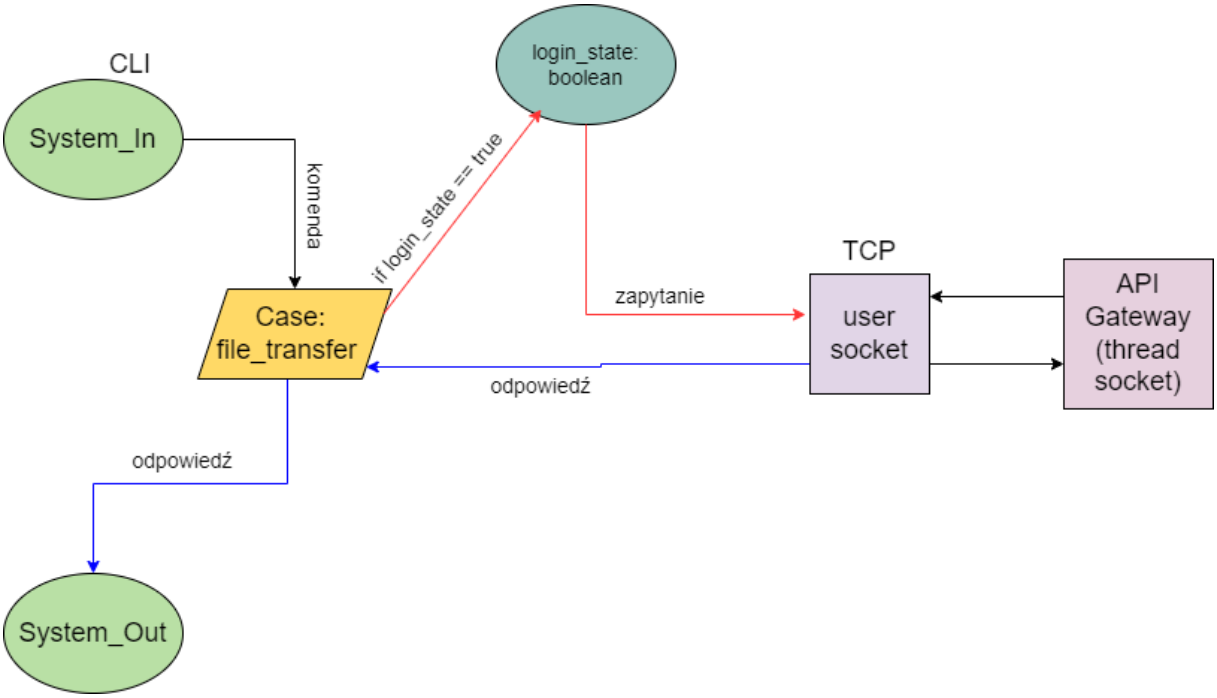
Schemat 5. tablica

V. dodawanie nowego posta



Schemat 6. dodawanie posta

VI. przesyłanie pliku na serwer oraz pobieranie pliku z serwera do katalogu domowego użytkownika

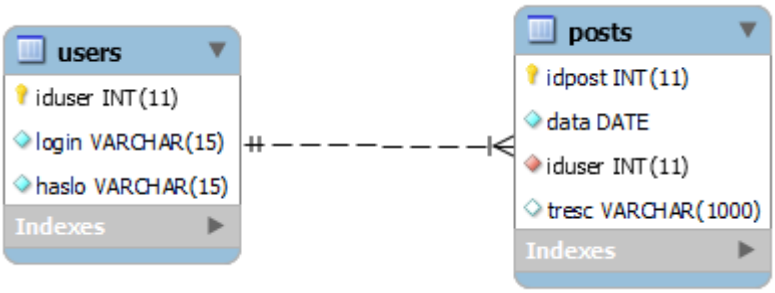


Schemat 7. Przesyłanie i pobieranie pliku

Komunikacja API Gateway – serwer dyskowy

API Gateway tworzy wątek (gniazdo) dla każdego użytkownika. Gdy użytkownik wysyła zapytanie o plik, API przekazuje je dalej przy użyciu protokołu TCP, w odpowiedzi na które serwer dyskowy tworzy nowy wątek (gniazdo) w celu przesłania pliku. Plik jest wysyłany w kawałkach do API Gateway. Po przesłaniu ostatniej porcji pliku, serwer dyskowy zamyka wątek.

Schemat bazy danych



Schemat 8. Baza danych

Uproszczona implementacja projektu: programowanie na gniazdach w języku Java

I. Lista uproszczeń:
   1. Serwer dyskowy – pobrane pliki użytkowników umieszczane są w katalogu użytkownika w folderze *user_files*, a dodane na serwer – w folderze *file_server*.
   2. Baza danych – brak implementacji bazy URL-ów do plików.

II. Rozwiązania programowe:
   1. Protokoły – przesyłane w formie łańcuchów znaków (np. type:login#user_name:patka#user_password:haslo), które najpierw są rozdzielane, a następnie tworzy się z nich słownik w celu szybszego i łatwiejszego dostania się do danych.
   2. Przesył plików pomiędzy klientem, API i mikroserwisem – przesył całej tablicy bajtów przy użyciu metody *readFully* obiektu klas *FileOutputStream* i *FileInputStream*.
   3. Łączność z bazą danych – interfejs JDBC.

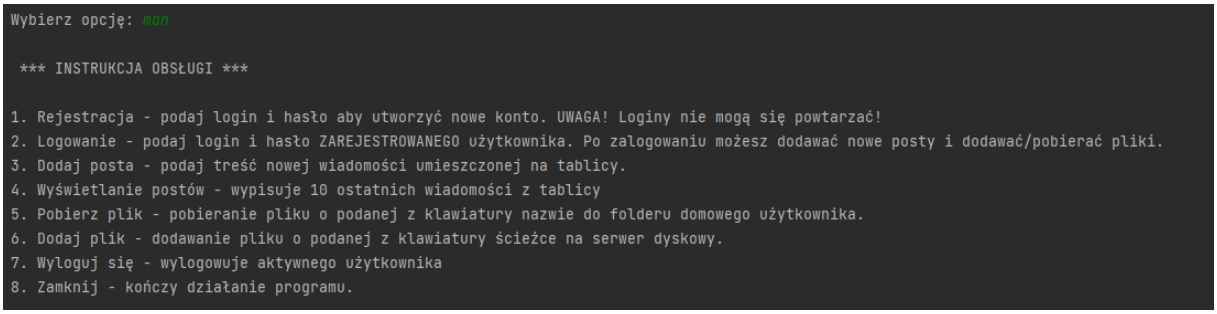
III. Działanie programu:

a. Menu

Java Socket 1. Menu

Java Socket 2. Menu - błąd

Użytkownik podaje z klawiatury numer interesującej go opcji lub wpisuje *man* aby uzyskać informacje o dostępnych w programie funkcjonalnościach. Po każdej akcji użytkownik zostanie poinformowany o jej wyniku (czy opcja zakończyła się powodzeniem czy niepowodzeniem).

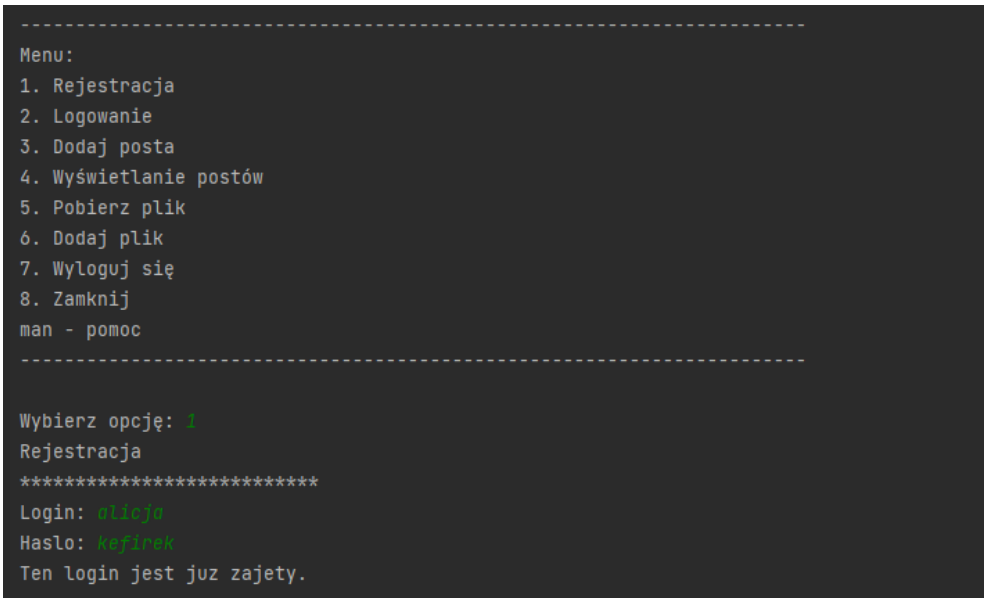


Java Socket 3. Pomoc (man)

b. Rejestracja

Java Socket 4. Rejestracja

Aby zarejestrować nowe konto, użytkownik podaje z klawiatury unikalny login oraz hasło. Jeśli podany login jest już zajęty, na ekranie pojawi się komunikat o tym informujący. Rejestracji może dokonać tylko niezalogowany użytkownik.



Java Socket 5. Rejestracja – błąd 1.

```
--------------------------------------------------------------------------
Menu:
1. Rejestracja
2. Logowanie
3. Dodaj posta
4. Wyświetlanie postów
5. Pobierz plik
6. Dodaj plik
7. Wyloguj się
8. Zamknij
man - pomoc
--------------------------------------------------------------------------

Wybierz opcję: 1
Wyloguj się aby zarejestrować nowe konto.
```

Java Socket 6. Rejestracja - błąd 2.

c. Logowanie

```
--------------------------------------------------------------------------
Menu:
1. Rejestracja
2. Logowanie
3. Dodaj posta
4. Wyświetlanie postów
5. Pobierz plik
6. Dodaj plik
7. Wyloguj się
8. Zamknij
man - pomoc
--------------------------------------------------------------------------

Wybierz opcję: 2
Logowanie
**************************
Login: alicja
Haslo: kefir
Zalogowano.
```

Java Socket 7. Logowanie

Użytkownik loguje się na swoje konto przy użyciu podanego podczas rejestracji loginu i hasła. Gdy któraś z danych logowania jest niepoprawna, użytkownik zostanie o tym poinformowany. Stan logowania jest podtrzymywany do czasu wybrania opcji *Wyloguj się* – w przypadku, gdy zalogowany użytkownik wybierze opcję nr 2, zostanie poinformowany na jakim koncie jest aktualnie zalogowany.

```
--------------------------------------------------------------------------
Menu:
1. Rejestracja
2. Logowanie
3. Dodaj posta
4. Wyświetlanie postów
5. Pobierz plik
6. Dodaj plik
7. Wyloguj się
8. Zamknij
man - pomoc
--------------------------------------------------------------------------

Wybierz opcję: 2
Logowanie
**************************
Login: ala
Haslo: kefir
Niepoprawny login.
```

Java Socket 8. Logowanie – błąd 1.

```
--------------------------------------------------------------------------
Menu:
1. Rejestracja
2. Logowanie
3. Dodaj posta
4. Wyświetlanie postów
5. Pobierz plik
6. Dodaj plik
7. Wyloguj się
8. Zamknij
man - pomoc
--------------------------------------------------------------------------

Wybierz opcję: 2
Logowanie
**************************
Login: alicja
Haslo: kefirek
Niepoprawne haslo.
```

Java Socket 9. Logowanie – błąd 2.

```
------------------------------------------------------------------------
Menu:
1. Rejestracja
2. Logowanie
3. Dodaj posta
4. Wyświetlanie postów
5. Pobierz plik
6. Dodaj plik
7. Wyloguj się
8. Zamknij
man - pomoc
------------------------------------------------------------------------

Wybierz opcję: 2
Jesteś już zalogowany jako alicja
```

Java Socket 10. Logowanie – błąd 3.

d. Dodawanie nowego posta

```
------------------------------------------------------------------------
Menu:
1. Rejestracja
2. Logowanie
3. Dodaj posta
4. Wyświetlanie postów
5. Pobierz plik
6. Dodaj plik
7. Wyloguj się
8. Zamknij
man - pomoc
------------------------------------------------------------------------

Wybierz opcję: 3
Dodawanie posta
**************************
Tresc: Geniusz niech się weźmie do roboty...
Dodano nowego posta.
```

Java Socket 11. Dodaj posta

Zalogowany użytkownik może dodać nową wiadomość (posta) do tablicy. Aby to zrobić, wybiera opcję nr 3 i podaje treść wiadomości. Wszystkie posty są przechowywane w bazie danych.

```
------------------------------------------------------------------------
Menu:
1. Rejestracja
2. Logowanie
3. Dodaj posta
4. Wyświetlanie postów
5. Pobierz plik
6. Dodaj plik
7. Wyloguj się
8. Zamknij
man - pomoc
------------------------------------------------------------------------

Wybierz opcję: 3
Zaloguj się aby dodać posta.
```

Java Socket 12. Dodaj posta - błąd

e. Wyświetlanie tablicy postów

```
------------------------------------------------------------------------
Menu:
1. Rejestracja
2. Logowanie
3. Dodaj posta
4. Wyświetlanie postów
5. Pobierz plik
6. Dodaj plik
7. Wyloguj się
8. Zamknij
man - pomoc
------------------------------------------------------------------------

Wybierz opcję: 4
alicja napisał(a)          2022-05-31
Geniusz niech się weźmie do roboty...
- - - - - - - - - - - - - - - - - - - -

patka napisał(a)           2022-05-29
Sushi jest smaczne!
- - - - - - - - - - - - - - - - - - - -

bananowiec68 napisał(a)            2022-05-29
jabułka som zue
- - - - - - - - - - - - - - - - - - - -

patka napisał(a)           2022-05-29
Dzisiaj jest pochmurna niedziela, pada deszcz...
- - - - - - - - - - - - - - - - - - - -

test napisał(a)            2022-05-28
Hallo Welt!
- - - - - - - - - - - - - - - - - - - -

igor napisał(a)            2022-05-28
witaj świecie!
- - - - - - - - - - - - - - - - - - - -
```

Java Socket 13. Wyświetlanie postów

Wyświetlanie tablicy nie jest zarezerwowane dla zarejestrowanych użytkowników – można to zrobić będąc niezalogowanym. Wyświetlane jest dziesięć ostatnich wiadomości z bazy danych.

f. Pobierz plik

```
------------------------------------------------------------------------
Menu:
1. Rejestracja
2. Logowanie
3. Dodaj posta
4. Wyświetlanie postów
5. Pobierz plik
6. Dodaj plik
7. Wyloguj się
8. Zamknij
man - pomoc
------------------------------------------------------------------------

Wybierz opcję: 5
Podaj nazwę pliku: obrazek.png
Pobrano plik do katalogu użytkownika alicja
```

Java Socket 14. Pobierz plik

Opcja *Pobierz plik* kopiuje plik (o nazwie podanej z klawiatury) z folderu ogólnego do folderu domowego aktualnie zalogowanego użytkownika. W przypadku wybrania nieistniejącego pliku użytkownik zobaczy komunikat o błędzie.

```
------------------------------------------------------------------------
Menu:
1. Rejestracja
2. Logowanie
3. Dodaj posta
4. Wyświetlanie postów
5. Pobierz plik
6. Dodaj plik
7. Wyloguj się
8. Zamknij
man - pomoc
------------------------------------------------------------------------

Wybierz opcję: 5
Zaloguj się aby pobrać plik.
```

Java Socket 15. Pobierz plik - błąd 1.

```
------------------------------------------------------------------------
Menu:
1. Rejestracja
2. Logowanie
3. Dodaj posta
4. Wyświetlanie postów
5. Pobierz plik
6. Dodaj plik
7. Wyloguj się
8. Zamknij
man - pomoc
------------------------------------------------------------------------

Wybierz opcję: 5
Podaj nazwę pliku: sernik.jpg
PLIK o podanej nazwie nie istnieje!
```

Java Socket 16. Pobierz plik - błąd 2.

g. Dodaj plik

```
------------------------------------------------------------------------
Menu:
1. Rejestracja
2. Logowanie
3. Dodaj posta
4. Wyświetlanie postów
5. Pobierz plik
6. Dodaj plik
7. Wyloguj się
8. Zamknij
man - pomoc
------------------------------------------------------------------------

Wybierz opcję: 6
Podaj ścieżkę do pliku: C:\Users\Patka\Documents\#studia\IVsemestr\ts\java_socket\pdf.pdf
Pomyślnie dodano plik na serwer.
------------------------------------------------------------------------
```

Java Socket 17. Dodaj plik

Dodawać pliki na serwer mogą tylko zalogowani użytkownicy. W tym celu program prosi użytkownika o podanie z klawiatury ścieżki do pliku znajdującego się na urządzeniu klienta. Jeśli ścieżka wskazuje na folder lub nieistniejący plik, pojawi się komunikat o błędzie.

```
--------------------------------------------------------------------------
Menu:
1. Rejestracja
2. Logowanie
3. Dodaj posta
4. Wyświetlanie postów
5. Pobierz plik
6. Dodaj plik
7. Wyloguj się
8. Zamknij
man - pomoc
--------------------------------------------------------------------------

Wybierz opcję: 6
Zaloguj się aby dodać plik na serwer.
```

Java Socket 18. Dodaj plik - błąd 1.

```
--------------------------------------------------------------------------
Menu:
1. Rejestracja
2. Logowanie
3. Dodaj posta
4. Wyświetlanie postów
5. Pobierz plik
6. Dodaj plik
7. Wyloguj się
8. Zamknij
man - pomoc
--------------------------------------------------------------------------

Wybierz opcję: 6
Podaj ścieżkę do pliku: C:\Users\Patka\Documents\#studia\IVsemestr\ts\java_socket\sernik.jpg
Plik o podanej ścieżce nie istnieje! Sprawdź czy podana ścieżka jest poprawna.
```

Java Socket 19. Dodaj plik - błąd 2.

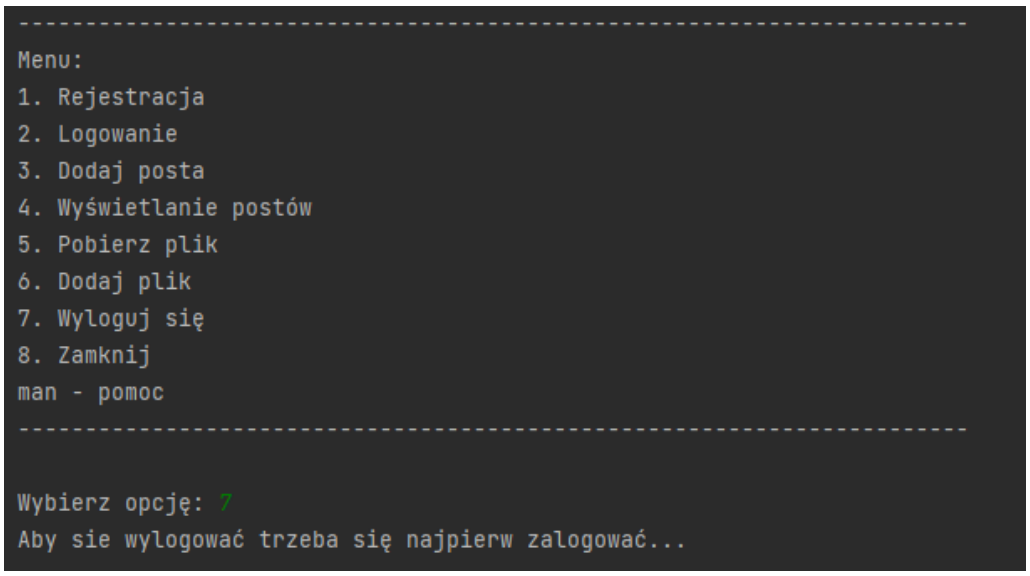
h. Wyloguj się

```
--------------------------------------------------------------------------
Menu:
1. Rejestracja
2. Logowanie
3. Dodaj posta
4. Wyświetlanie postów
5. Pobierz plik
6. Dodaj plik
7. Wyloguj się
8. Zamknij
man - pomoc
--------------------------------------------------------------------------

Wybierz opcję: 7
Wylogowano.
```

Java Socket 20. Wyloguj się

Opcja nr 7 powoduje wylogowanie się (wykasowanie przetrzymywanego loginu) użytkownika. Gdy niezalogowany użytkownik spróbuje się wylogować, zostanie poinformowany o błędzie.



Java Socket 21. Wyloguj się - błąd

i. Zamknij

Aby zakończyć działanie programu klienckiego należy podać z klawiatury cyfrę 8. Opcja ta zamyka aktywne połączenie między klientem a serwerem. Serwer pozostaje aktywny i czeka na dalsze zgłoszenia klientów.

IV. Implementacja

a. Client.java

```java
import java.io.*;
import java.net.*;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.Base64;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class Client {

    public static void main(String[] args) {
        try {
            Scanner scanner = new Scanner(System.in);
            Socket socket = new Socket( host: "localhost",  port: 5056);

            DataInputStream dataInput = new DataInputStream(socket.getInputStream());
            DataOutputStream dataOutput = new DataOutputStream(socket.getOutputStream());
            Base64.Encoder encoder = Base64.getEncoder();
            Base64.Decoder decoder = Base64.getDecoder();

            String protocol;
            Map<String, String> response = new HashMap<~>();

            boolean loggedIn = false;
            String login = null;
            boolean download = false;

            // wymiana informacji między klientem a serwerem
            while (true) {
                if (!download) {
                    System.out.print("\n------------------------------------------------------------------------\n" +
                            "Menu:\n" +
                            "1. Rejestracja \n" +
                            "2. Logowanie \n" +
                            "3. Dodaj posta \n" +
                            "4. Wyświetlanie postów \n" +
                            "5. Pobierz plik \n" +
                            "6. Dodaj plik \n" +
                            "7. Wyloguj się \n" +
                            "8. Zamknij \n" +
                            "man - pomoc" +
                            "\n------------------------------------------------------------------------\n\nWybierz opcję: ");

                    String tosend = scanner.nextLine();

                    if (tosend.equals("8")) {
                        dataOutput.writeUTF( str: "8");
                        socket.close();
                        break;
                    }

                    switch (tosend) {
                        case "1":
                            if (!loggedIn) {
                                protocol = "type:register";
                                System.out.println("Rejestracja\n************************** ");
                                System.out.print("Login: ");
                                protocol += "#user_name:" + scanner.nextLine();
                                System.out.print("Haslo: ");
                                protocol += "#user_password:" + scanner.nextLine();

                                dataOutput.writeUTF(protocol);
                            } else
                                dataOutput.writeUTF( str: "type:writeMessage#msg:Wyloguj się aby zarejestrować nowe konto.");
                            break;

                        case "2":
                            if (!loggedIn) {
                                protocol = "type:login";
                                System.out.println("Logowanie\n************************** ");
                                System.out.print("Login: ");
                                protocol += "#user_name:" + scanner.nextLine();
                                System.out.print("Haslo: ");
                                protocol += "#user_password:" + scanner.nextLine();

                                dataOutput.writeUTF(protocol);
```

```
                } else
                    dataOutput.writeUTF( str: "type:writeMessage#msg:Jesteś już zalogowany jako " + login);

                break;

            case "3":
                if (loggedIn) {
                    protocol = "type:chat";
                    System.out.println("Dodawanie posta\n************************** ");
                    protocol += "#user_name:" + login;
                    System.out.print("Treść: ");
                    protocol += "#message:" + scanner.nextLine();

                    dataOutput.writeUTF(protocol);
                } else
                    dataOutput.writeUTF( str: "type:writeMessage#msg:Zaloguj się aby dodać posta.");
                break;

            case "4":
                dataOutput.writeUTF( str: "type:table");
                break;

            case "5":
                if (loggedIn) {
                    protocol = "type:file_transfer#mode:download";
                    protocol += "#user_name:" + login;
                    System.out.print("Podaj nazwę pliku: ");
                    protocol += "#file_name:" + scanner.nextLine();
                    protocol += "#offset:0";
                    dataOutput.writeUTF(protocol);
                } else
                    dataOutput.writeUTF( str: "type:writeMessage#msg:Zaloguj się aby pobrać plik.");
                break;

            case "6":
                if (loggedIn) {
                    System.out.print("Podaj ścieżkę do pliku: ");
                    String p = scanner.nextLine();
                   try {
                       Path path = Paths.get(p);
                       if (!Files.exists(path))
                           throw new FileNotFoundException();

                       else {
                           if (Files.isDirectory(path))
                               throw new FileNotFoundException();

                           File file = new File(p);
                           FileInputStream fileInputStream = new FileInputStream(file.getAbsolutePath());
                           String fileName = file.getName();

                           String packet;
                           int packetSize = 1024*30;                                        //1024*50 za duże
                           int fileLength = (int) (Math.ceil(file.length()/packetSize) * packetSize);
                           long loadedBytes = 0;
                           long offset = 0;
                           byte[] fileContentBytes = new byte[packetSize];

                           protocol = "type:file_transfer#mode:upload#file_name:" + fileName + "#file_length:" + fileLength + "#offset:-1";
                           dataOutput.writeUTF(protocol);
                           System.out.println("Dodawanie... proszę czekać...");
                           while ((loadedBytes = fileInputStream.read(fileContentBytes)) != -1) {
                               packet = encoder.encodeToString(fileContentBytes);
                               dataOutput.writeUTF( str: "type:file_transfer#mode:upload#file_name:" + fileName + "#file_length:" + fileLength + "#offset:" + offset
                                       + "#file_content:" + packet);
//                               System.out.println("dlugosc: "+fileLength+"\toffset: "+offset);

                               clear(fileContentBytes);
                               offset += loadedBytes;
                           }
                           fileInputStream.close();
                       }
                   } catch (FileNotFoundException e) {
                       dataOutput.writeUTF( str: "type:writeMessage#msg:Plik o podanej ścieżce nie istnieje! Sprawdź czy podana ścieżka jest poprawna.");
                   }
                } else
                    dataOutput.writeUTF( str: "type:writeMessage#msg:Zaloguj się aby dodać plik na serwer.");
```

```
                    break;


                case "man":
                    dataOutput.writeUTF( str: "type:writeMessage#msg:\n *** INSTRUKCJA OBSŁUGI ***\n\n" +
                            "1. Rejestracja - podaj login i hasło aby utworzyć nowe konto. UWAGA! Loginy nie mogą się powtarzać!\n" +
                            "2. Logowanie - podaj login i hasło ZAREJESTROWANEGO użytkownika. Po zalogowaniu możesz dodawać nowe posty i dodawać/pobierać pliki.\n" +
                            "3. Dodaj posta - podaj treść nowej wiadomości umieszczonej na tablicy.\n" +
                            "4. Wyświetlanie postów - wypisuje 10 ostatnich wiadomości z tablicy\n" +
                            "5. Pobierz plik - pobieranie pliku o podanej z klawiatury nazwie do folderu domowego użytkownika.\n" +
                            "6. Dodaj plik - dodawanie pliku o podanej z klawiatury ścieżce na serwer dyskowy.\n" +
                            "7. Wyloguj się - wylogowuje aktywnego użytkownika\n" +
                            "8. Zamknij - kończy działanie programu.");
                    break;

                default:
                    dataOutput.writeUTF( str: "type:writeMessage#msg:Wybrano błędną opcję!");
            }
        }
        // Przetwarzanie i wypisywanie otrzymanej wiadomości

        String received = dataInput.readUTF();
        String[] data = received.split( regex: ":|#");

        for (int i = 0; i < data.length; i += 2)
            response.put(data[i], data[i+1]);

        if (response.get("type").equals("login") && response.get("state").equals("true")) {
                loggedIn = true;
                login = response.get("user_name");
        }


        if (response.get("type").equals("download")){
            Path downloadpath = Paths.get( first: "C:\\Users\\Patka\\Documents\\#studia\\IVsemestr\\ts\\java_socket\\user_files\\"+login+"\\" + response.get("file_name"));
            File downloaded = new File(String.valueOf(downloadpath));
            if (response.get("offset").equals("-1"))
                if (downloaded.exists())
                    downloaded.delete();

            FileOutputStream fileOutputStream = new FileOutputStream(downloaded,  append: true);
            int offset = Integer.parseInt(response.get("offset"));
            int fileLength = Integer.parseInt(response.get("file_length"));
            byte[] fileContentBytes;

//                System.out.println(fileLength+" - > "+offset);

            if (!response.get("offset").equals("-1")) {
                fileContentBytes = decoder.decode(response.get("file_content"));
                fileOutputStream.write(fileContentBytes);
            }

            if (offset == fileLength)
                download = false;
            else {
                download = true;
                protocol = "type:file_transfer#mode:download#user_name:"+response.get("user_name")+"#file_name:"+response.get("file_name")+"#offset:"+offset;
                dataOutput.writeUTF(protocol);
            }

            fileOutputStream.close();
        }

        System.out.print(response.get("msg"));
    }

        scanner.close();
        dataInput.close();
        dataOutput.close();

    } catch (Exception e) {
        e.printStackTrace();
    }
}

@ public static void clear(byte[] table) {
    for (int i = 0; i < table.length; i++)
        table[i] = 0;
}
}
```

Java Socket 22. implementacja - Client.java

b. Server.java (API Gateway)

```java
import java.io.*;
import java.net.*;
import java.util.HashMap;
import java.util.Map;

public class Server {
    public static void main(String[] args) throws IOException {

        ServerSocket serverSocket = new ServerSocket( port: 5056);

        // running infinite loop for getting client request
        while (true) {
            Socket socket = null;

            try {
                // socket object to receive incoming client requests
                socket = serverSocket.accept();

                System.out.println("Nawiazano polaczenie z : " + socket);

                // obtaining input and out streams
                DataInputStream dataInput = new DataInputStream(socket.getInputStream());
                DataOutputStream dataOutput = new DataOutputStream(socket.getOutputStream());

                System.out.println("Tworzenie watku dla nowego klienta");

                // create a new thread object
                Thread thread = new ClientHandler(socket, dataInput, dataOutput);
                thread.start();

            } catch (Exception e) {
                socket.close();
                e.printStackTrace();
            }
        }
    }
}

class ClientHandler extends Thread {
    final DataInputStream dataInput;
    final DataOutputStream dataOutput;
    final Socket clientSocker;

    public ClientHandler(Socket s, DataInputStream dataInput, DataOutputStream dataOutput) {
        this.clientSocker = s;
        this.dataInput = dataInput;
        this.dataOutput = dataOutput;
    }

    @Override
    public void run() {
        String received;

        while (true) {
            try {
                received = dataInput.readUTF();

                if (received.equals("8")) {
                    System.out.println("Client " + this.clientSocker + " sends exit...");
                    System.out.println("Closing this connection.");
                    this.clientSocker.close();
                    System.out.println("Connection closed");
                    break;
                }

                String[] data = received.split( regex: ":|#");
                Map<String, String> query = new HashMap<~>();
                for (int i = 0; i < data.length; i += 2) {
                    query.put(data[i], data[i + 1]);
//                    System.out.println(data[i]+": "+data[i+1]);
                }


                // Odpowiedź na żądania klienta
                switch (query.get("type")) {
```

```
77                  case "register":
78                      Socket registerSocket = new Socket( host: "localhost",  port: 1111);
79                      DataOutputStream registerOutput = new DataOutputStream(registerSocket.getOutputStream());
80                      DataInputStream registerInput = new DataInputStream(registerSocket.getInputStream());
81                      registerOutput.writeUTF(received);
82
83                      received = registerInput.readUTF();
84                      this.dataOutput.writeUTF(received);
85
86                      registerOutput.close();
87                      registerInput.close();
88                      registerSocket.close();
89                      break;
90
91                  case "login":
92                      Socket loginSocket = new Socket( host: "localhost",  port: 1212);
93                      DataOutputStream loginOutput = new DataOutputStream(loginSocket.getOutputStream());
94                      DataInputStream loginInput = new DataInputStream(loginSocket.getInputStream());
95                      loginOutput.writeUTF(received);
96
97                      received = loginInput.readUTF();
98                      this.dataOutput.writeUTF(received);
99                      loginOutput.close();
100                     loginInput.close();
101                     loginSocket.close();
102                     break;
103
104                 case "chat":
105                     Socket chatSocket = new Socket( host: "localhost",  port: 1313);
106                     DataOutputStream chatOutput = new DataOutputStream(chatSocket.getOutputStream());
107                     DataInputStream chatInput = new DataInputStream(chatSocket.getInputStream());
108                     chatOutput.writeUTF(received);
109
110                     received = chatInput.readUTF();
111                     this.dataOutput.writeUTF(received);
112
113                     chatInput.close();
114                     chatOutput.close();
115                     chatSocket.close();
116                     break;
117
118                 case "table":
119                     Socket tableSocket = new Socket( host: "localhost",  port: 1414);
120                     DataInputStream tableInput = new DataInputStream(tableSocket.getInputStream());
121
122                     received = tableInput.readUTF();
123                     this.dataOutput.writeUTF(received);
124
125                     tableInput.close();
126                     tableSocket.close();
127                     break;
128
129                 case "file_transfer":
130 //                      System.out.println(received);
131                     Socket fileSocket = new Socket( host: "localhost",  port: 2526);
132                     DataOutputStream fileOutput = new DataOutputStream(fileSocket.getOutputStream());
133                     DataInputStream fileInput = new DataInputStream(fileSocket.getInputStream());
134 //                      System.out.println(query.get("offset") +" - > "+query.get("file_length"));
135 //                      System.out.println(received);
136                     fileOutput.writeUTF(received);
137
138                     if (query.get("mode").equals("download")){
139                         received = fileInput.readUTF();
140 //                          System.out.println(received);
141                         this.dataOutput.writeUTF(received);
142                     }
143
144                     else if (query.get("mode").equals("upload") && query.get("offset").equals(query.get("file_length"))) {
145                         received = fileInput.readUTF();
146 //                          System.out.println(received);
147                         this.dataOutput.writeUTF(received);
148                     }
149
150                     fileInput.close();
151                     fileOutput.close();
152                     fileSocket.close();
153                     break;
```

```
                    case "writeMessage":
                        dataOutput.writeUTF( str: "type:msg#msg:"+query.get("msg"));
                        break;
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        try {
            this.dataInput.close();
            this.dataOutput.close();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Java Socket 23. implementacja - Server.java (API Gateway)

c. Register.java

```
import java.io.*;
import java.net.*;
import java.sql.*;
import java.util.HashMap;
import java.util.Map;

public class Register {

    public static void main(String[] args) throws IOException, SQLException, ClassNotFoundException {
        ServerSocket registerSocket = new ServerSocket( port: 1111);

        while (true){
            Socket clientSocket = registerSocket.accept();
            DataOutputStream dataOutputStream = new DataOutputStream(clientSocket.getOutputStream());
            DataInputStream dataInputStream = new DataInputStream(clientSocket.getInputStream());

            String received = dataInputStream.readUTF();
            String data[] = received.split( regex: ":|#");
            Map<String, String> query = new HashMap<String, String>();
            for (int i = 0; i < data.length; i += 2)
                query.put(data[i], data[i + 1]);


            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection connection = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/tsy",  user: "root",  password: "");
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery( sql: "select login from users");
            boolean exist = false;

            while (resultSet.next()) {
                if (resultSet.getString( columnLabel: "login").equals(query.get("user_name")))
                    exist = true;
            }

            if (!exist) {
                PreparedStatement pstmt = connection.prepareStatement( sql: "insert into users values (0, ?, ?)");
                pstmt.setString( parameterIndex: 1, query.get("user_name"));
                pstmt.setString( parameterIndex: 2, query.get("user_password"));
                pstmt.executeUpdate();
                new File( pathname: "C:\\Users\\Patka\\Documents\\#studia\\IVsemestr\\ts\\java_socket\\user_files\\"+query.get("user_name")).mkdirs();
                dataOutputStream.writeUTF( str: "type:register#state:true#msg:Zarejestrowano.");
            }
            else
                dataOutputStream.writeUTF( str: "type:register#state:false#msg:Ten login jest juz zajety.");

            connection.close();
        }
    }
}
```

Java Socket 24. implementacja - Register.java

d. Login.java

```java
import java.io.*;
import java.net.*;
import java.sql.*;
import java.util.HashMap;
import java.util.Map;

public class Login {

    public static void main(String[] args) throws IOException, SQLException, ClassNotFoundException {
        ServerSocket loginSocket = new ServerSocket( port: 1212);

        while (true){
            Socket clientSocket = loginSocket.accept();
            DataOutputStream dataOutputStream = new DataOutputStream(clientSocket.getOutputStream());
            DataInputStream dataInputStream = new DataInputStream(clientSocket.getInputStream());

            String received = dataInputStream.readUTF();
            String data[] = received.split( regex: ":|#");
            Map<String, String> query = new HashMap<~>();
            for (int i = 0; i < data.length; i += 2)
                query.put(data[i], data[i + 1]);

            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection connection = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/tsy",  user: "root",  password: "");
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery( sql: "select login, haslo from users");

            while (resultSet.next()){
                if (resultSet.getString( columnLabel: "login").equals(query.get("user_name"))) {
                    if (resultSet.getString( columnLabel: "haslo").equals(query.get("user_password")))
                        dataOutputStream.writeUTF( str: "type:login#state:true#user_name:"+query.get("user_name")+"#msg:Zalogowano.");
                    else
                        dataOutputStream.writeUTF( str: "type:login#state:false#msg:Niepoprawne haslo.");
                }
            }
            dataOutputStream.writeUTF( str: "type:login#state:false#msg:Niepoprawny login.");
            connection.close();
        }
    }
}
```

Java Socket 25. implementacja - Login.java

e. Chat.java

```java
import java.io.*;
import java.net.*;
import java.sql.*;
import java.util.HashMap;
import java.util.Map;

public class Chat {

    public static void main(String[] args) throws IOException, SQLException, ClassNotFoundException {
        ServerSocket chatSocket = new ServerSocket( port: 1313);

        while (true){
            Socket clientSocket = chatSocket.accept();
            DataOutputStream dataOutputStream = new DataOutputStream(clientSocket.getOutputStream());
            DataInputStream dataInputStream = new DataInputStream(clientSocket.getInputStream());

            String received = dataInputStream.readUTF();
            String data[] = received.split( regex: ":|#");
            Map<String, String> query = new HashMap<~>();
            for (int i = 0; i < data.length; i += 2)
                query.put(data[i], data[i + 1]);

            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection connection = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/tsy", user: "root", password: "");
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery( sql: "select login, iduser from users where login ='"+query.get("user_name")+"'");
            int uid = 0;

            while (resultSet.next())
                if (resultSet.getString( columnLabel: "login").equals(query.get("user_name")))
                    uid = resultSet.getInt( columnLabel: "iduser");

            PreparedStatement pstmt = connection.prepareStatement( sql: "insert into posts values (0, curdate(), ?, ?)");
            pstmt.setInt( parameterIndex: 1, uid);
            pstmt.setString( parameterIndex: 2, query.get("message"));
            pstmt.executeUpdate();

            dataOutputStream.writeUTF( str: "type:chat#state:true#msg:Dodano nowego posta.");
            connection.close();
        }
    }
}
```

Java Socket 26. implementacja - Chat.java

f. Table.java

```java
import java.io.*;
import java.net.*;
import java.sql.*;

public class Table {

    public static void main(String[] args) throws IOException, SQLException, ClassNotFoundException {
        ServerSocket loginSocket = new ServerSocket( port: 1414);
        while (true){
            Socket clientSocket = loginSocket.accept();
            DataOutputStream dataOutputStream = new DataOutputStream(clientSocket.getOutputStream());

            String result = "";

            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection connection = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/tsy", user: "root", password: "");
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery( sql: "select p.data, p.tresc, u.login from users u inner join posts p on u.iduser = p.iduser order by p.data desc limit 10");

            while (resultSet.next())
                result += resultSet.getString( columnLabel: "login")+" napisał(a) \t\t\t"+resultSet.getString( columnLabel: "data")+"\n"+resultSet.getString( columnLabel: "tresc")
                        +"\n- - - - - - - - - - - - - - - - - - - -\n\n";

//            System.out.println(result);
            dataOutputStream.writeUTF( str: "type:table#state:true#msg:"+result);
            connection.close();
        }
    }
}
```

Java Socket 27. implementacja - Table.java

g. FileTransfer.java

```java
import java.io.*;
import java.net.*;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.Base64;
import java.util.HashMap;
import java.util.Map;

public class FileTransfer {

    public static void main(String[] args) throws IOException {
        ServerSocket fileSocket = new ServerSocket( port: 2526);

        while (true) {
            Socket clientSocket = fileSocket.accept();
            DataOutputStream dataOutputStream = new DataOutputStream(clientSocket.getOutputStream());
            DataInputStream dataInputStream = new DataInputStream(clientSocket.getInputStream());
            Base64.Decoder decoder = Base64.getDecoder();
            Base64.Encoder encoder = Base64.getEncoder();
            byte[] fileContentBytes;

            String received = dataInputStream.readUTF();
            String data[] = received.split( regex: ":|#");
            Map<String, String> query = new HashMap<~~>();
            for (int i = 0; i < data.length; i += 2)
                query.put(data[i], data[i + 1]);

            if (query.get("mode").equals("download")) {
                try {
                    String p = "C:\\Users\\Patka\\Documents\\#studia\\IVsemestr\\ts\\java_socket\\file_server\\" + query.get("file_name");
                    Path path = Paths.get(p);
                    if (!Files.exists(path))
                        throw new FileNotFoundException();

                    else {
                        if (Files.isDirectory(path))
                            throw new FileNotFoundException();

                        File file = new File(p);
                        FileInputStream fileInputStream = new FileInputStream(file.getAbsolutePath());
                        FileInputStream fileInputStream = new FileInputStream(file.getAbsolutePath());
                        String fileName = file.getName();

                        int packetSize = 1024*30;                                                   //1024*50 za duże
                        String packet;
                        int fileLength = (int)(Math.ceil(file.length()/packetSize)*packetSize);     //ile pakietów trzeba wysłać
                        long loadedBytes = 0;
                        long offset = 0;
                        fileContentBytes = new byte[packetSize];


                        while ((loadedBytes = fileInputStream.read(fileContentBytes)) != -1) {
                            packet = encoder.encodeToString(fileContentBytes);
                            if ((Long.parseLong(query.get("offset"))) == offset) {
                                System.out.println("dlugosc: " + fileLength + "\toffset: " + offset+" "+query.get("offset"));
                                offset += packetSize;
                                if (offset == fileLength)
                                    dataOutputStream.writeUTF( str: "type:download#file_name:" + fileName + "#file_length:" + fileLength + "#offset:" + offset
                                            + "#file_content:" + packet + "#msg:Pobrano plik do katalogu użytkownika "+query.get("user_name"));
                                else
                                    dataOutputStream.writeUTF( str: "type:download#file_name:" + fileName + "#file_length:" + fileLength + "#offset:" + offset
                                            + "#file_content:" + packet + "#msg:...\n");
                                dataOutputStream.flush();
                                break;
                            }
                            clear(fileContentBytes);
                            offset += loadedBytes;
                        }

                        fileInputStream.close();
                    }
                } catch (FileNotFoundException e) {
                    dataOutputStream.writeUTF( str: "type:download#state:false#msg:PLIK o podanej nazwie nie istnieje!");
                }
            } else {
                Path uploadpath = Paths.get( first: "C:\\Users\\Patka\\Documents\\#studia\\IVsemestr\\ts\\java_socket\\file_server\\" + query.get("file_name"));
                File uploaded = new File(String.valueOf(uploadpath));
                if (query.get("offset").equals("-1"))
                    if (uploaded.exists())
                        uploaded.delete();
```

```
81
82                  FileOutputStream fileOutputStream = new FileOutputStream(uploaded,  append: true);
83                  int offset = Integer.parseInt(query.get("offset"));
84                  int fileLength = Integer.parseInt(query.get("file_length"));
85
86                  System.out.println(fileLength+" - > "+offset);
87
88                  if (!query.get("offset").equals("-1")) {
89                      fileContentBytes = decoder.decode(query.get("file_content"));
90                      fileOutputStream.write(fileContentBytes);
91                  }
92
93                  if (offset == fileLength) {
94                      System.out.println("KONIEC");
95                      dataOutputStream.writeUTF( str: "type:upload#state:true#msg:Pomyślnie dodano plik na serwer.");
96                  }
97                  fileOutputStream.close();
98              }
99          }
100     }
101 @   public static void clear(byte[] table) {
102         for (int i = 0; i < table.length; i++)
103             table[i] = 0;
104     }
105 }
106
```

Java Socket 28. implementacja - FileTransfer.java

Spis rysunków