Asynchroniczny Serwer TCP
Inżynieria Oprogramowania

Wykonali:
Mateusz Kuźniak 139092
Artur Jackowiak 140713
Jordan Kondracki 140721

1. Wprowadzenie – opis systemu

Aplikacja serwera asynchronicznego TCP z zaimplementowanym wzorcem TAP umożliwiająca logowanie użytkownika do systemu oraz założenie konta. Użytkownik za pomocą klienta łączy się z serwerem przy użyciu podanego adresu IPv4 oraz portu. Następnie klient loguje się na serwerze za pomocą loginu i hasła. Po zalogowaniu klient ma dostęp do podstawowej wersji chmury – obsługa plików txt. Prosta aplikacja, umożliwiająca dalszy rozwój projektu. Serwer został wyposażony w bazę danych (*SQLite)*, dzięki czemu po wyłączeniu serwera pozostają dane zarejestrowanych użytkowników oraz ich plików. Serwer również nie pozwoli na zarejestrowanie się dwóch użytkowników o tej samej nazwie.

2. Słownik

- IPv4 (adres IP) – adres wykorzystywany do komunikacji (format xxx.xxx.xxx.xxx -> np. 168.154.202.2),
- port - jeden z parametrów gniazda, który umożliwia nawiązanie połączenia,
- puTTy – darmowe oprogramowanie klienckie,
- asynchroniczność – sposób przesyłania danych pozwalający na nieregularne wysyłanie danych. Pozwala obsługiwać więcej niż 1 klienta na raz,
- TCP – niezawodny protokół komunikacyjny stosowany do przesyłania danych między procesami uruchomionymi na różnych maszynach,
- tryb BeginInvoke – umożliwia wykonanie delegata asynchronicznie w wątku, w którym został utworzony uchwyt bazowy formantu,
- baza danych – zbór danych zapisanych zgodnie z określonymi regułami,
- SQLite – system zarządzania bazą danych, obsługujący język SQL,

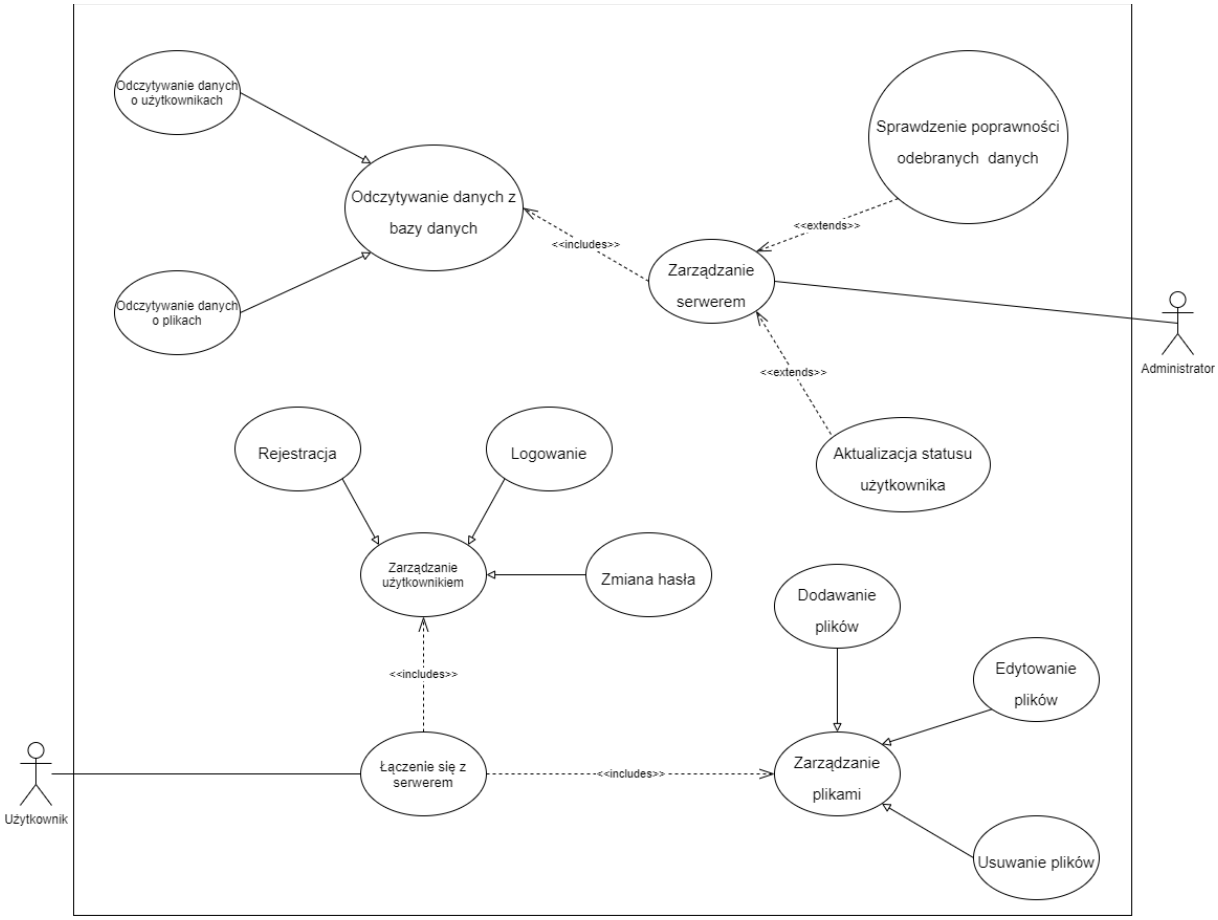
3. Wymagania funkcjonalne

1. serwer utrzymuje połączenie z wieloma klientami,
2. administrator kończy działanie serwera,
3. klient dodaje, edytuje lub usuwa pliki tekstowe,
4. klient loguje się,
5. klient rejestruje się,
6. klient zmienia hasło,
7. serwer sprawdza poprawność hasła,
8. serwer przechowuje pliki tekstowe,
9. serwer łączy się z bazą danych.

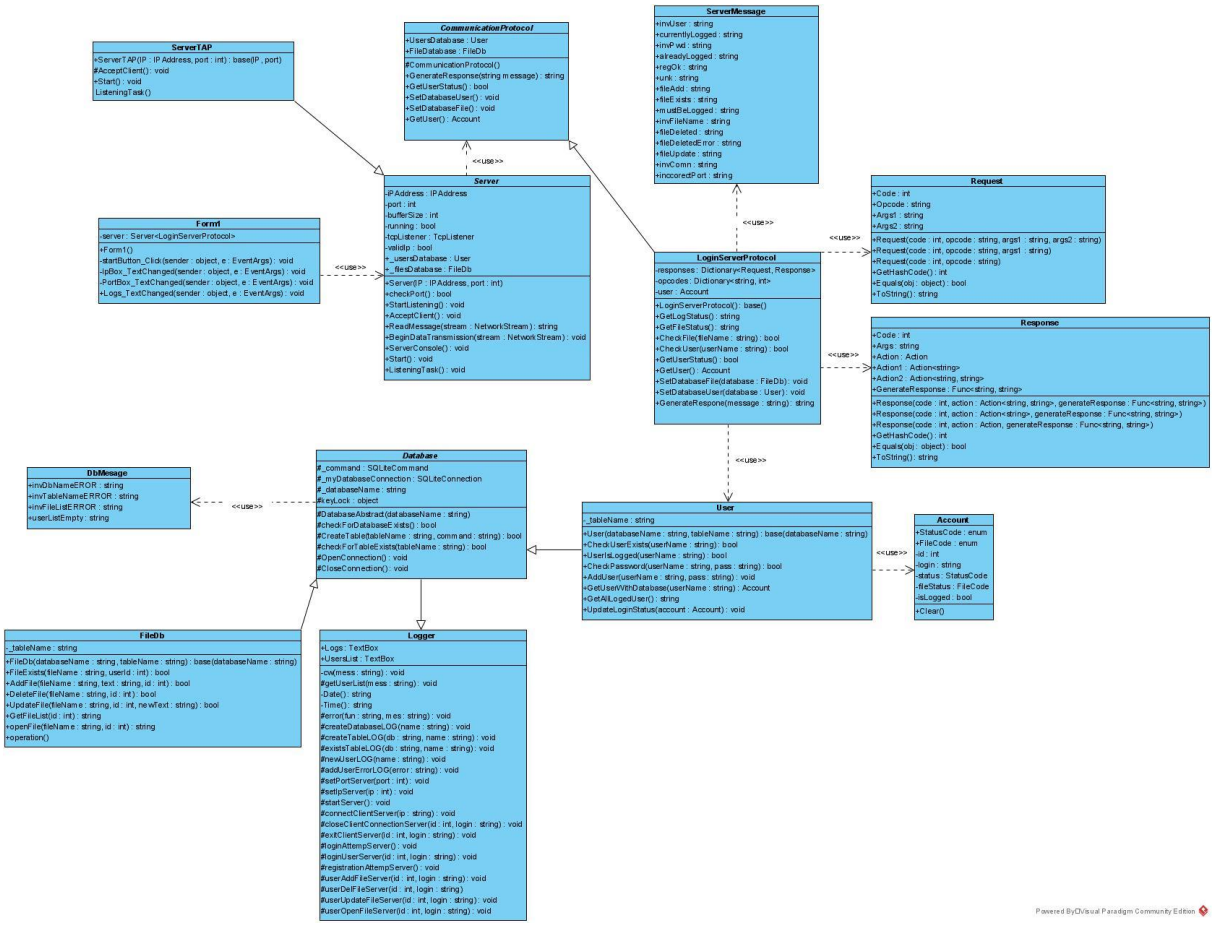
4. Wymagania niefunkcjonalne

1. Język programowania: C# (.NET Framework),
2. środowisko programistycznie: Visual Studio 2017,
3. system operacyjny: Windows 10,
4. Język bazy danych: SQL,
5. Baza danych: SQLite,
6. Interfejs: graficzny,
7. do przetestowania działania komunikacji klient-serwer niezbędne jest dodatkowe oprogramowanie klienckie,
8. serwer utrzymuje połączenie z wieloma klientami jednocześnie
9. serwer po rozłączeniu z klientem jest w stanie nadal obsługiwać pozostałych klientów,
10. po rozłączeniu klienta serwer wyświetla odpowiedni komunikat,
11. po utracie połączenia z klientem serwer wyświetla odpowiedni komunikat,
12. możliwość łatwej rozbudowy projektu,
13. serwer obsługuje wyjątki,
14. serwer zapobiega rejestracji dwóch kont o tej samej nazwie,
15. serwer nie wymaga żadnych dodatkowych działań, prócz uruchomienia.

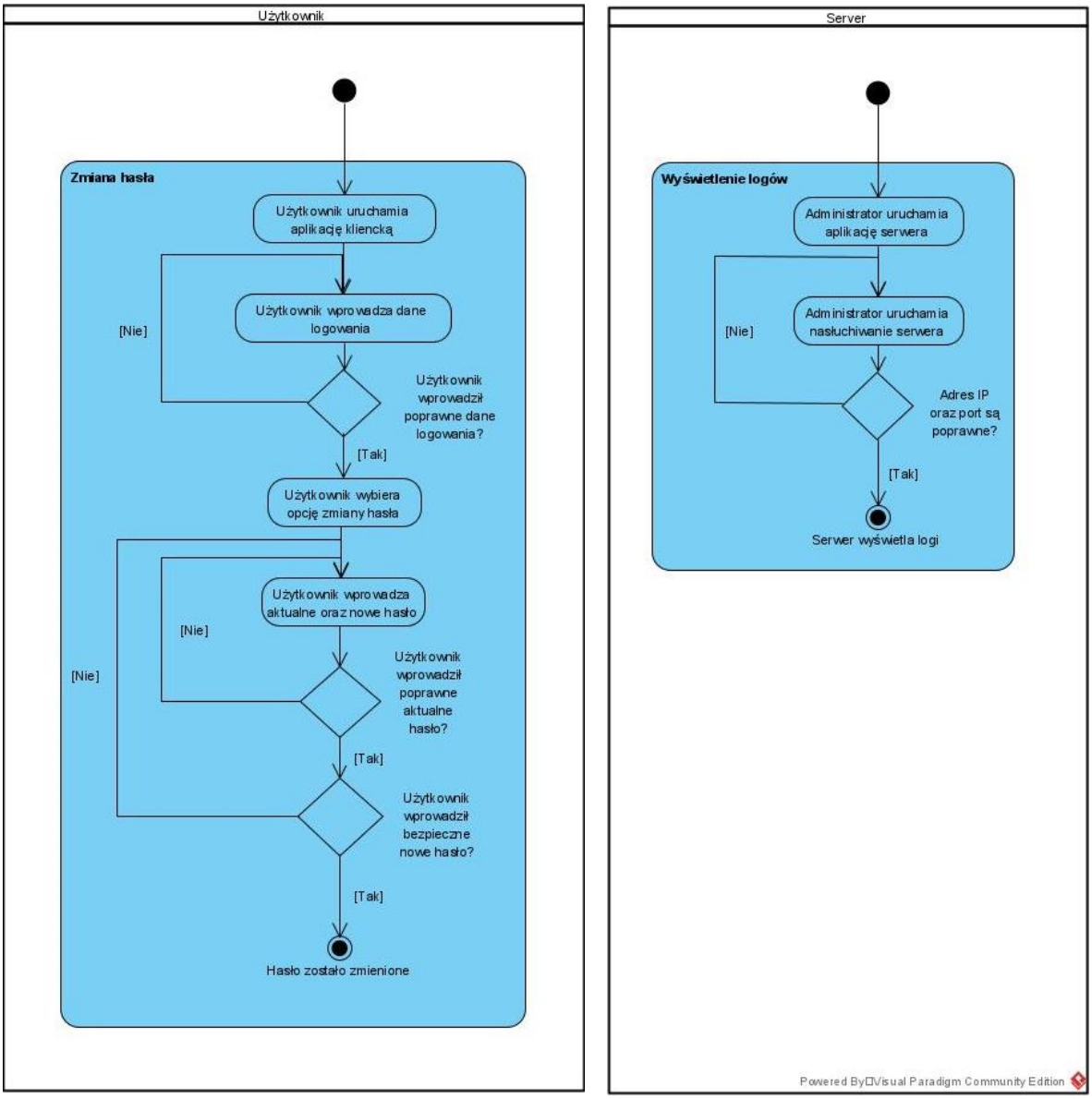
5. Diagram przypadków użycia

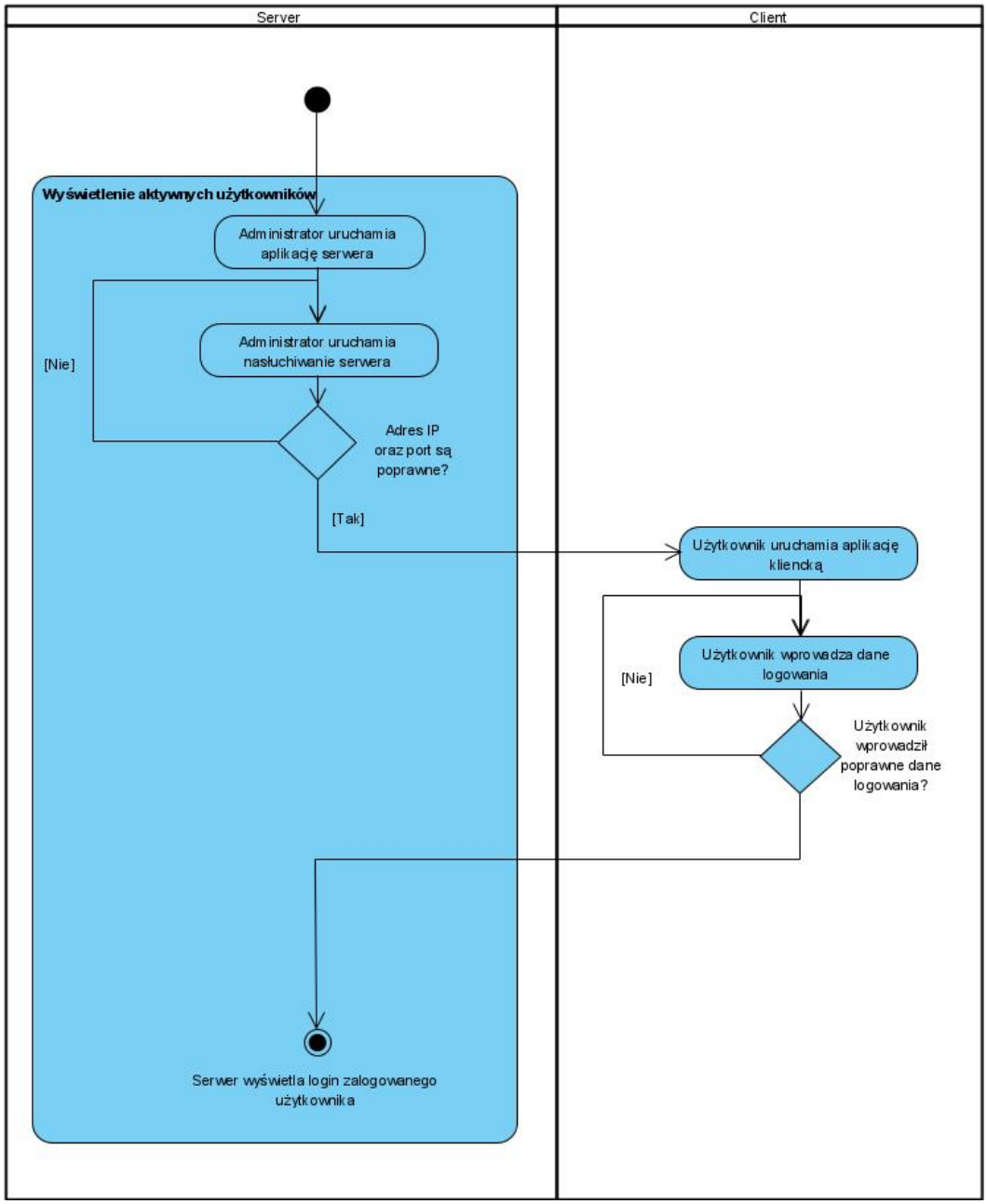


6. Diagram klas



7. Diagramy aktywności




Server
Client
Wyświetlenie aktywnych użytkowników
Administrator uruchamia aplikację serwera
Administrator uruchamia nasłuchiwanie serwera
[Nie]
Adres IP oraz port są poprawne?
[Tak]
Użytkownik uruchamia aplikację kliencką
Użytkownik wprowadza dane logowania
[Nie]
Użytkownik wprowadził poprawne dane logowania?
Serwer wyświetla login zalogowanego użytkownika

Zespół

- Product Owner: Jordan Kondracki
- Scrum Team:
    - Development Team:
        - Mateusz kuźniak
        - Artur Jackowiak
        - Jordan Kondracki
- Scrum Master: Mateusz kuźniak

Linki

https://trello.com/b/4kTUE6L0/sprint-4-backlog

https://github.com/mateuszkuzniak/ServerTCP