Licencjacki projekt programistyczny
IIUWr 2009/2010

Adrian Chudziński, Przemysław Gospodarczyk

e-Talk – komunikator internetowy

Dokumentacja

Wersja	Zmiany	Autorzy	Data
1.0	Pierwsza wersja	Adrian Chudziński Przemysław Gospodarczyk	2010-03-25
2.0	Druga wersja	Adrian Chudziński Przemysław Gospodarczyk	2010-06-27

Spis treści

1. Słownik pojęć

America Online (AOL) – największy amerykański dostawca usług internetowych. Jeden z pierwszych wydawców elektronicznego biuletynu informacyjnego (BBS) w USA, który od 1995 roku jest dostępny również w Internecie. W 1998 roku wykupił firmę Netscape – producenta przeglądarki Netscape Navigator.

API (*Application Programming Interface*) – zbiór konwencji wywoływania funkcji określających sposób dostępu do danej usługi przez system operacyjny oraz lista dostępnych funkcji wraz z opisem znaczenia ich parametrów.

Commodore 64 – model popularnego w latach osiemdziesiątych domowego komputera 8-bitowego firmy Commodore. Oparty na procesorach 6510 lub 8502 (w zależności od wersji), wyposażony w 64 KB pamięci operacyjnej RAM oraz odrębne procesory przetwarzające dane wejściowe i wyjściowe, obraz oraz dźwięk, co odciążało jednostkę centralną i umożliwiało tworzenie wydajnego oprogramowania.

Emotikony (ang. *emoticons*) – zestaw symboli wyrażających emocje, używanych w wiadomościach przesyłanych pocztą elektroniczną, przez komunikator internetowy, w grupach dyskusyjnych i w wiadomościach SMS.

Excite – wyszukiwarka internetowa autorstwa firmy o tej samej nazwie, która została w roku 1999 wykupiona przez amerykańską firmę At Home.

Extensible Messaging and Presence Protocol (dawniej Jabber) – protokół komunikacji oraz powiadamiania o obecności w czasie rzeczywistym oparty na języku XML. Głównym zastosowaniem Extensible Messaging and Presence Protocol jest wymiana wiadomości w komunikatorach internetowych. Serwery XMPP umożliwiają także za pomocą tzw. transportów komunikację z użytkownikami innych protokołów, np. GaduGadu, Tlen.pl, ICQ i MSN Messenger. Protokół jest również stosowany w systemie blogowania Jogger przez XMPP.

Google Talk – komunikator internetowy i usługa VoIP amerykańskiej firmy Google. Google Talk działa od 24 sierpnia 2005 roku. Wygląd programu (kolorystyka, układ tabel, czcionki) jest stylizowany na interfejs graficzny poczty elektronicznej Gmail.

Interfejs graficzny (ang. *graphical interface, GUI*) – dominująca obecnie odmiana interfejsu użytkownika, w którym kontakt z komputerem i sterowanie programami odbywa się za pomocą okien, ikon, przycisków, suwaków i rozmaitych menu. Interfejs graficzny jest obsługiwany za pomocą myszy i sprzężonego z nią kursora; możliwe jest też używanie w interfejsie graficznym operacji klawiszowych (skrótów klawiaturowych).

Jabber – patrz Extensible Messaging and Presence Protocol.

Komunikator internetowy (ang. *instant messenger*) – program komputerowy pozwalający na przesyłanie natychmiastowych komunikatów (ang. *instant messaging*) między dwoma lub więcej komputerami poprzez sieć Internet. Od poczty elektronicznej różni się tym, że oprócz samej wiadomości, przesyłane są także informacje o obecności użytkowników, co zwiększa znacznie szansę na prowadzenie bezpośredniej konwersacji.

MSN (ang. *MicroSoft Network*) – serwis internetowy firmy Microsoft, który oferuje pocztę elektroniczną oraz fora dyskusyjne dotyczące różnych dziedzin i tematów.

PETSCII (ang. *PET Standard Code of Information Interchange*) – rodzaj zbioru znaków ASCII z 1977 roku znany również jako CBM ASCII, używany w 8-bitowych komputerach Commodore 64.

Pidgin (dawniej GAIM) – wielosystemowy komunikator internetowy obsługujący wiele protokołów transmisyjnych. Pidgin jest oprogramowaniem bezpłatnym (o otwartym kodzie), dostępnym na warunkach GNU GPL i został zbudowany przez Marka Spencera dla systemów uniksowych, jednak obecnie jest dostępny także dla systemów Windows, MacOS, SkyOS oraz Qt Extended (dla urządzeń PDA).

Serwis społecznościowy (portal społecznościowy) – rodzaj interaktywnych stron WWW, które są współtworzone przez sieci społeczne osób podzielających wspólne zainteresowania lub chcących poznać zainteresowania innych. Większość portali społecznościowych dostarcza użytkownikom wielu sposobów komunikacji, np. czaty, komunikatory, listy dyskusyjne, blogi i fora dyskusyjne.

Sieć P2P (ang. *peer–to–peer*) – sieć partnerska, architektura sieciowa oparta na równoważności wszystkich jej węzłów. W sieci P2P każdy komputer dysponuje podobnymi możliwościami oraz może inicjować połączenia. Nie ma ustalonej hierarchii ani centralnego serwera. Ten sam komputer może równocześnie pełnić rolę serwera i klienta, czyli pobierać dane z innych komputerów i udostępniać swoje zasoby wszystkim pozostałym komputerom.

Tablica dzielona (ang. *shared whiteboard*) – miejsce na dysku dostępne dla wielu użytkowników jednocześnie. Wersje poszczególnych użytkowników są synchronizowane w czasie zbliżonym do rzeczywistego.

Twitter – darmowy serwis społecznościowy udostępniający usługę mikroblogowania

umożliwiającą użytkownikom wysyłanie oraz odczytywanie tak zwanych tweetów. Tweet to krótka, nie przekraczająca 140 znaków wiadomość tekstowa wyświetlana na stronie użytkownika oraz dostarczana pozostałym użytkownikom, którzy obserwują dany profil. Użytkownicy mogą dodawać krótkie wiadomości do swojego profilu z poziomu strony głównej serwisu, wysyłając wiadomości SMS lub korzystając z zewnętrznych aplikacji.

Ubique – firma produkująca oprogramowanie do błyskawicznej komunikacji. Najbardziej znaną aplikacją Ubique jest Virtual Places Chat, a użyta technika znalazła zastosowanie w aplikacji Lotus Sametime firmy IBM. Obecnie Ubique jest częścią IBM Haifa Labs.

Yahoo – jeden z najpopularniejszych portali internetowych należący do amerykańskiej firmy Yahoo. Oprócz hierarchicznego katalogu kategorii tematycznych zawiera także mechanizm przeszukiwania zasobów Internetu.

Yahoo Messenger – komunikator internetowy firmy Yahoo. Pierwsza wersja komunikatora ukazała się 9 marca 1998 roku pod nazwą Yahoo Pager.

2. Wstęp, Komunikatory jako zjawisko społeczne

Podstawowymi cechami Internetu są globalność i masowość. Istotną cechą Internetu jest masowość jego użytkowników, którzy traktują go nierzadko jako jedyne okno na świat. Użytkownicy, korzystając z Internetu w poszukiwaniu informacji, danych lub szukając kontaktu z innymi ludźmi w świecie wirtualnym nie tylko rozwijają globalną Sieć, ale stają się również jej integralną i nieodzowną częścią. Człowiek jako istota społeczna uzewnętrznia swoją potrzebę komunikacji poprzez pośrednią i bezpośrednią wymianę informacji z innymi ludźmi. Komunikacja sieciowa jest pośrednia i pozbawiona komunikatów niewerbalnych takich jak: ton głosu, mimika, wygląd i zachowanie. Wyżej wymienione aspekty komunikacji sieciowej jednoznacznie wskazują na jej ograniczoność w stosunku do bezpośrednich kontaktów międzyludzkich. Niewerbalna komunikacja może wzmacniać, osłabiać, a nawet negować przekazy werbalne. W związku z tym komunikaty werbalne mają ogromne znaczenie w komunikacji i mogą stać się ważniejsze niż treść wypowiedzi. Internet powoduje, że wzajemne kontakty stają się bardziej powierzchowne i tracą ważną część swojego charakteru.

Komunikacja sieciowa ma jednak swoje zalety, które przez wiele osób są marginalizowane. Najważniejszą cechą komunikacji internetowej jest jej anonimowość. Użytkownicy często fałszują swoje dane osobowe lub nie podają ich w ogóle. Możliwa jest błyskawiczna zmiana tożsamości. Bezpośrednia komunikacja w świecie rzeczywistym jest znacznie

trudniejsza niż internetowa. Ludzie hamowani przez nieśmiałość oraz lęk przed odrzuceniem w dużej mierze rezygnują z tego typu komunikacji. Anonimowość – często złudna – powoduje, że wyżej wymienione czynniki tracą na znaczeniu. Badania japońskich naukowców z Ochanomizu University [1] dotyczące studentów dowodzą, że relacje interpersonalne w życiu wymagają pewnych umiejętności społecznych oraz orientacji na innych.

Komunikatory internetowe stały się bardzo popularną formą komunikacji wśród młodzieży, grupy społecznej, która najbardziej potrzebuje kontaktów z rówieśnikami oraz akceptacji i dowartościowania. Współczesna młodzież jest bardziej spragniona relacji interpersonalnych niż pokolenie urodzone w latach sześćdziesiątych i siedemdziesiątych. Według Stevena Gerali, szefa chicagowskiego *Department of Youth Ministry & Adolescent Studies* [1] współcześni młodzi ludzie chcą intymności, stałości oraz bezpiecznych związków z innymi. Przede wszystkim pragną być doceniani i chronieni. Ważnym czynnikiem staje się anonimowość, która zapewnia młodemu użytkownikowi bezpieczeństwo oraz do pewnego stopnia bezkarność i łatwość w poznawaniu oraz komunikacji z ludźmi z całego świata. W Internecie zawiera się przyjaźnie, przeżywa miłość, zdobywa informacje i nowe hobby.
Nie każdy nastolatek jest wyposażony w wymagany przez komunikację bezpośrednią zestaw cech. Komunikatory internetowe ich nie wymagają. Każdy użytkownik może przedstawić siebie w zupełnie inny, wymarzony przez siebie sposób, koloryzując rzeczywistość. Nastolatkowie utrzymujący takie kontakty zwiększają swoją pewność siebie, która będzie procentować w życiu dorosłym.

Wbrew powszechnym opiniom, z komunikatorów internetowych i portali społecznościowych korzystają jednak coraz częściej ludzie dorośli, których przyciąga darmowość usługi oraz prostota obsługi. W artykule *Rosja też ćwierka*, zamieszczonym w tygodniku *Polityka* [2], autorzy Marek Ostrowski i Adam Szostkiewicz opisują portal społecznościowy **Twitter** jako miejsce wymiany zdań głównie między ludźmi pełnoletnimi.
Za niespodziankę można uznać wielką popularność Twittera zważywszy, że umożliwia on wysyłanie wiadomości o długości do 140 znaków, co powinno stanowić poważne utrudnienie w korespondowaniu. Użytkownicy Twittera nie wykorzystują go jednak do poważnych rozmów, lecz do bezzwłocznego pisania o tym, co ich w danym momencie interesuje i emocjonuje. Ponadto Twitter pozwala na sprawdzenie, co interesuje innych oraz, według psychologa z Uniwersytetu Warszawskiego Jana Zająca, sprzyja rozmowom o niczym w celu podtrzymania relacji towarzyskich z innymi ludźmi.
Według pobieżnej analizy przeprowadzonej przez autorów wspomnianego artykułu, prasa jest ciągle bardziej opiniotwórcza i dostarcza więcej informacji.
Twitter sprzyja również rozwojowi literatury mądrości (ang. *wisdom literature*), czyli pisaniu aforyzmów, czego nie można już uznać za prymitywną formę komunikacji. Wysyłanie krótkich wiadomości nie tylko zaśmieca głowę, ale również potrafi zmobilizować do myślenia.

Komunikatory internetowe pomagają w życiu społecznym osobom niepełnosprawnym, ludziom, którzy z powodu swojego kalectwa posiadają niskie zdanie o sobie oraz obawiają się odrzucenia z powodu swojej odmienności. W Internecie chory człowiek nie obawia się zdemaskowania prawdy o sobie ani o swojej ewentualnej nieatrakcyjności fizycznej spowodowanej kalectwem.

Istnieje wiele teorii na temat wypierania kontaktów w świecie rzeczywistym przez kontakty w Internecie, a wpływ komunikatorów internetowych na życie społeczne człowieka jest oceniany przez naukowców bardzo niejednoznacznie.
Socjolog Sherry Turkle w swojej książce *Life on the Screen* [3] udowodniła, że rozwój komunikatorów internetowych i portali społecznościowych prowadzi najpierw do wyobcowania poszczególnych jednostek, a w konsekwencji do społecznego autyzmu i nadmiernego indywidualizmu internautów. Według autorki internauci stają się leniwi, nie podejmują trudów związanych z utrzymywaniem kontaktów społecznych w świecie rzeczywistym.
Z drugiej strony, według kanadyjskiego socjologa Barry'ego Wellmana uznawanego powszechnie za jednego z najwybitniejszych badaczy współczesnej cywilizacji, w tym także Internetu, komunikacja sieciowa uzupełnia kontakty, nie eliminując jednocześnie podstawowych form komunikacji, np. spotkań bezpośrednich [1]. Według jego badań internaci częściej interesują się polityką i biorą udział w życiu społecznym.

Fakt, że pomimo dużej liczby komunikatorów internetowych na rynku, autorzy projektu tworzą kolejne narzędzie do internetowego plotkowania należy usprawiedliwić ogromnym zainteresowaniem tego typu aplikacjami oraz, co za tym idzie, ogromnymi pieniędzmi za reklamy. Przed napisaniem własnego oprogramowania autorzy przeprowadzili analizę komunikatorów internetowych dostępnych na polskim rynku, aby nie powielać błędów konkurencji i wykorzystać najlepsze pomysły.

3. Historia komunikatorów

Pomysł na komunikator błyskawicznie przesyłający i odbierający wiadomości jest starszy niż sama idea Internetu. Jest związany z wielodostępnymi systemami operacyjnymi z połowy lat sześćdziesiątych. Systemy operacyjne CTSS i Multics były używane do szybkiej i prostej komunikacji pomiędzy użytkownikami pracującymi na tej samej maszynie jednocześnie.

Wraz z pojawieniem się, a następnie rozwojem Internetu protokół wymiany wiadomości został poszerzony o możliwość komunikacyjne globalnej sieci. W pierwszych komunikatorach, wydanych w 1983 roku używano protokołu partnerskiego (**P2P**). Przykładami takich programów były: talk, ytalk oraz ntalk. W niektórych komunikatorach zastosowano architekturę klient-serwer. Przykładami aplikacji typu klient-serwer były: talker (1984) oraz IRC (1988).

Na fali popularności **Bulletin Board System** w latach osiemdziesiątych pojawiło się wiele serwisów komputerowych udostępniających na maszynach miejsca, w których można umieszczać i czytać ogłoszenia, obsługiwać własną skrzynkę pocztową oraz pobierać i przesyłać pliki. Przykładem takiego serwisu jest Freelancin' Roundtable działający przez ponad 3 lata od października 1984 roku do listopada 1987 roku.

W drugiej połowie lat osiemdziesiątych i na początku lat dziewięćdziesiątych internetowy serwis Quantum Link oferował użytkownikom **Commodore 64** przesyłanie wiadomości między połączonymi użytkownikami. Wiadomości nazywane On-Line Messages (w skrócie OLM) pojawiały się na żółtym pasku wraz z informacją o nazwie nadawcy. Quantum Link wykorzystywał znaną z Commodore 64 grafikę tekstową **PETSCII** (zob. słownik). Interfejs graficzny Quantum Link (o ile w ogóle mogła być o nim mowa w tym przypadku) był nieporównywalnie gorszy z możliwościami późniejszych tego typu programów, które działały pod systemami Windows i Unix.
Późniejsza wersja Quantum Link znana jako **America Online** posiadała serwis **AOL Instant Messenger** (w skrócie AIM) do przesyłania wiadomości i zyskała większą popularność niż poprzedniczka.

Pierwsze nowoczesne komunikatory internetowe z interfejsem graficznym, które wyglądem i możliwościami przypominały znane z dzisiejszych czasów zaczęły powstawać od połowy lat dziewięćdziesiątych.
Przykładem takiej aplikacji jest PowWow (1998), który był jednym z pierwszych komunikatorów internetowych działających pod systemem Windows. Program zawierał wiele nietypowych i innowacyjnych dodatków takich jak:

- odtwarzacz plików dźwiękowych w formacie WAV;
- wbudowany syntezator mowy;
- program przysyłania mowy w protokole VoIP;
- program do komunikacji według protokołów POP i SMTP;
- program obsługujący **tablicę dzieloną** (zob. słownik), miejsce na dysku dostępne dla wielu użytkowników jednocześnie.

Innym przykładem jest działający również pod Windows **ICQ** (1996), wyprodukowany przez izraelską firmę Mirabilis. ICQ działa do dzisiaj, a w 1998 roku prawa do ICQ wykupił AOL za 407 milionów dolarów. W serwisie ICQ jest obecnie ponad 100 milionów zarejestrowanych kont, a AOL przyznano prawa do dwóch patentów dotyczących przesyłania informacji przez Internet. Angielski zwrot Instant Messenger, a także skrót IM, w Stanach Zjednoczonych jest własnością AOL i nie może być używany przez inne firmy. Działający pod wszystkimi najpopularniejszymi systemami operacyjnymi komunikator GAIM został z tego powodu przemianowany w 2007 roku na **Pidgin**.

W tym czasie inne firmy, wśród nich **Excite**, **MSN**, **Ubique** i **Yahoo**, pracowały nad swoim własnym oprogramowaniem. Wszystkie aplikacje wyprodukowane przez te firmy miały własne, niezależne protokoły przesyłania wiadomości, co powodowało, że użytkownicy musieli korzystać z wielu komunikatorów, aby móc połączyć się z każdą siecią.

W 1998 roku IBM wprowadził na rynek IBM Lotus Sametime oparte na technologii zastosowanej w komunikatorze firmy Ubique. Oprócz podstawowych, typowych funkcji oprogramowanie IBM oferowało następujące nowości:

- archiwizowanie rozmów;
- możliwość organizowania internetowych wideokonferencji;
- chat;
- możliwość wykorzystania kodeków graficznych i dźwiękowych;
- możliwość łączenia się z innymi sieciami, takimi jak: AOL Instant Messenger, **Yahoo Messenger**, **Google Talk** oraz serwisami opartymi na XMPP;
- otwarty interfejs programowy (API), umożliwiający integrację z innymi tego typu aplikacjami.

W 2000 roku na rynek wprowadzono program **Jabber** z powszechnie dostępnym kodem źródłowym oraz standardowy protokół **Extensible Messaging and Presence Protocol** (w skrócie XMPP). Serwery XMPP potrafiły działać jak bramy (ang. gateway) do innych protokołów wymiany wiadomości, w związku z czym wystarczyła jedna aplikacja typu klient, aby korzystać z wielu sieci. Serwery XMPP obsługiwały wszystkie najpopularniejsze wówczas protokoły.

4. Przegląd najpopularniejszych komunikatorów na polskim rynku

Autorzy projektu wykonali przegląd rynku komunikatorów internetowych w Polsce, aby wybrać najlepsze funkcjonalności dla swojej aplikacji i uniknąć powielania błędów konkurencji. Według ankiety przeprowadzonej przez serwis **idg.pl** [5] najpopularniejszym komunikatorem w Polsce jest Gadu-Gadu, którego używa aż 76% internautów. Drugie miejsce zajął Tlen (niecałe 14%). Pozostałe komunikatory, w tym znane na całym świecie ICQ, cieszą się popularnością jednoprocentową. Poniżej przeanalizowano najpopularniejsze komunikatory internetowe w naszym kraju.

4.1. Gadu-Gadu

Gadu-Gadu jest najpopularniejszym komunikatorem internetowym w Polsce. Pierwsza wersja programu pojawiła się 15 sierpnia 2000 roku i jest to pierwszy polski komunikator internetowy. Premiera spotkała się ze stosunkowo dużym zainteresowaniem – już pierwszego dnia konta założyło ponad 10 tysięcy użytkowników. Od dnia premiery użytkowników przybywa w szybkim tempie. Po roku od premiery Gadu-Gadu miało ćwierć miliona użytkowników. Obecnie Gadu-Gadu posiada ponad 6 milionów zarejestrowanych kont. Każdego dnia, użytkownicy Gadu-Gadu wymieniają między sobą ok. 300 milionów wiadomości.
Na przykładzie programu Gadu-Gadu można z łatwością zauważyć, że najważniejszą cechą komunikatora internetowego jest jego masowość. Duża liczba zarejestrowanych użytkowników przyciąga innych użytkowników, którzy chcą mieć kontakt z jak największą liczbą ludzi (tzw. efekt kuli śnieżnej). Polski rynek komunikatorów internetowych rozwija się wraz z rozwojem Internetu w Polsce.
Drugą istotną cechą, która stawia Gadu-Gadu na 1. miejscu wśród komunikatorów internetowych w Polsce jest łatwość instalacji, konfiguracji i użytkowania. Jeżeli program ma być masowy, to będą go używać również mniej zaawansowani użytkownicy, dla których ta cecha może okazać się najistotniejsza przy wyborze komunikatora.
Ciekawym pomysłem twórców było uruchomienie serwisu społecznościowego **MojaGeneracja.pl**, który stał się serwisem integracyjnym dla użytkowników chcących poznać i wymieniać wiadomości z nieznanymi im dotąd osobami. Ponadto istnieje publiczny katalog użytkowników Gadu-Gadu oraz wyszukiwarka wykorzystująca dane osobowe podawane podczas rejestracji konta. Twórcy Gadu-Gadu na oficjalnej stronie chwalą się rolą, jaką odgrywa ich komunikator w życiu wielu ludzi: „Coraz więcej osób poznaje się dzięki Gadu-Gadu. Zawierane są pierwsze małżeństwa, ale też i pierwsze rozwody.” [4]
Gadu-Gadu posiada jednak pewne wady, które ze względu na wyżej wymienione zalety są marginalizowane przez mniej wymagających użytkowników lub niedostrzegane przez mniej zaawansowanych użytkowników.
Oryginalna wersja programu, bezpośrednio po instalacji zawiera tylko niezbędne funkcjonalności, które w dzisiejszych czasach można uznać za podstawowe. Istnieje jednak możliwość zainstalowania różnego rodzaju nieoficjalnych dodatków, które mogą rozszerzyć możliwości standardowego Gadu-Gadu.
Gadu-Gadu jest aplikacją darmową. Niestety wraz ze wzrostem popularności rośnie również liczba reklam, które program w najnowszej wersji wyświetla praktycznie przy każdej możliwej okazji. Warto również dodać, że z powodu dużej liczby wysyłanych wiadomości, serwer aplikacji nie daje sobie rady z przetwarzaniem dużej liczby zapytań, co powoduje chwilowe przerwy w dostępie do komunikatora.
Pomimo ograniczonej funkcjonalności, irytujących reklam i przejściowych problemów technicznych, Gadu-Gadu niepodzielnie rządzi na polskim rynku komunikatorów internetowych, dystansując również zagraniczną konkurencję.

4.2. Tlen

Tlen jest polskim komunikatorem internetowym serwisu o2.pl. Pierwsza wersja programu pojawiła się w październiku 2001 roku. Obecnie, jak podaje serwis [6] na Tlenie zarejestrowanych jest ponad 1.5 miliona użytkowników.
Program realizuje właściwie wszystkie funkcjonalności znane z Gadu-Gadu oraz zawiera istotne rozszerzenia, których brakuje zwycięzcy rankingu, np.

- połączenia wideo,
- połączenia telefoniczne,
- czaty,
- konferencje.

Ciekawym dodatkiem są proste gry internetowe, w których mogą uczestniczyć użytkownicy oraz tablica dzielona, na której użytkownicy mogą wspólnie rysować. Ponadto, na rynku istnieje wiele nieoficjalnych rozszerzeń wzbogacających funkcjonalności i wygląd komunikatora.
Aby przekonać do siebie użytkowników Gadu-Gadu, których jest więcej, Tlen jest w pełni kompatybilny z siecią Gadu-Gadu obsługując protokół GG. Oznacza to, że można wymieniać wiadomości z użytkownikami Gadu-Gadu, którzy nie mają zainstalowanego komunikatora Tlen. Oczywiście wtedy możliwości komunikacji zostają zredukowane do tego, co oferuje oryginalna wersja Gadu-Gadu. Ponadto istnieją dodatkowe rozszerzenia pozwalające nawiązać tekstową łączność z użytkownikami sieci ICQ i Jabber.
Tlen jest aplikacją darmową, której wadą są reklamy.

4.3. AQQ

Powstały w czerwcu 2007 roku AQQ realizuje najwięcej funkcjonalności ze wszystkich polskich komunikatorów internetowych. Liczne rozszerzenia (oficjalne i nieoficjalne) bogatej w funkcjonalności wersji podstawowej powodują, że program można uznać za medialno–komunikacyjną hybrydę. AQQ oprócz wszystkich podstawowych funkcji komunikatora internetowego oferuje również:

- odtwarzacz audio-wideo,
- organizator,
- program TV,
- tablicę dzieloną,
- monitor pobranych danych dla użytkowników Neostrady,

– monitor wewnętrznych zasobów komputera.

Inną zaletą AQQ są częste aktualizacje i stosunkowo szybko poprawiane błędy.
AQQ oferuje komunikację z następującymi sieciami: Gadu-Gadu, Tlen, Jabber, ICQ oraz Skype.
Największą wadą, która przesądza o mniejszej popularności AQQ, oprócz reklam, jest izolacja użytkowników tej sieci. Żaden z dostępnych komunikatorów internetowych nie obsługuje protokołu AQQ, co oznacza, że użytkownik tego komunikatora nie może wymieniać wiadomości z poziomu swojego konta AQQ z użytkownikami innych sieci. W związku z powyższym w przypadku chęci komunikowania się z użytkownikami innych sieci, użytkownicy AQQ nie mogą całkowicie zrezygnować z kont na innych serwerach i muszą łączyć się z nimi przez AQQ.

5. Planowane funkcje aplikacji

Poniższy opis zawiera funkcje, które powinien realizować e-Talk. Każda funkcja składa się z nazwy, opisu i priorytetu. Priorytet określa, jak ważna jest realizacja danej funkcji. W opisie uwzględniono 3 rodzaje priorytetów: wysoki, średni i niski.

5.1. Rozpoczynanie rozmowy

Priorytet: wysoki.

Użytkownik A rozpoczyna rozmowę z użytkownikiem B. Użytkownik B powinien zostać o tym fakcie poinformowany. Użytkownik B nie musi akceptować rozmowy. Informacja zwrotna o decyzji użytkownika B powinna zostać wysłana do osoby rozpoczynającej rozmowę.

5.2. Prowadzenie rozmowy

Priorytet: wysoki.

Dwóch użytkowników może swobodnie prowadzić rozmowę, wymieniając komunikaty tekstowe.

5.3. Zmiana stanu użytkownika

Priorytet: wysoki.

Każdemu użytkownikowi jest przypisany stan, który może dowolnie zmieniać. W aplikacji E-Talk będą dostępne stany: „dostępny”, „niedostępny”, „zaraz wracam” oraz

„niewidoczny". Stan użytkownika pokazywany jest wszystkim innym użytkownikom. Nie dotyczy to jednak stanu „niewidoczny", użytkownik w tym stanie jest widoczna dla innych użytkowników, podobnie jak użytkownik w stanie „niedostępny".

5.4. Lista kontaktów

Priorytet: wysoki.

Użytkownicy powinni dysponować samodzielnie definiowaną listą kontaktów. Lista może być dowolnie modyfikowana przez użytkownika.

5.5. Zakładanie konta

Priorytet: wysoki.

Osoby nie posiadające konta na serwerze aplikacji mogą przy pierwszym uruchomieniu programu założyć nowe konto. W tym celu muszą wypełnić krótki formularz, podając podstawowe dane osobowe.

5.6. Zmiana danych profilu

Priorytet: wysoki.

Użytkownik może dowolnie zmieniać dane zawarte w swoim profilu.

5.7. Podstawowe funkcje edytora tekstu

Priorytet: wysoki.

Użytkownik powinien mieć możliwość prostego formatowania tekstu swojej wiadomości. Przewiduje się następujące funkcje:

- zmiana rodzaju, wielkości i stylu czcionki;
- zmiana koloru tekstu;
- kopiowanie, wklejanie i wycinanie tekstu.

5.8. Wyszukiwanie użytkowników

Priorytet: średni.

Użytkownik może szukać innych użytkowników, podając fragment ich danych osobowych.

5.9. Obsługa protokołu GG

Priorytet: średni.

W związku z popularnością komunikatora Gadu-Gadu, dobrze by było, aby użytkownik e-Talk miał możliowość tekstowej łączności z użytkownikami tego najpopularniejszego komunikatora w naszym kraju.

5.10. Import oraz eksport listy kontaktów

Priorytet: średni.

Aplikacja E-Talk umożliwia na eksportowanie listy kontaktów do pliku lub serwera oraz import z pliku lub serwera.

5.11. Zapisywanie i przeglądanie archiwum rozmów

Priorytet: średni.

Użytkownik powinien mieć możliwość przeglądania treści swoich poprzednich rozmów.

5.12. Wklejanie obrazów do tekstu

Priorytet: niski.

Użytkownik powinien mieć możliwość dodania do swojego tekstu obrazu pobranego z dysku twardego. Przewiduje się obsługę wszystkich najpopularniejszych formatów graficznych.

5.13. Przesyłanie plików

Priorytet: niski.

Dwóch rozmawiających ze sobą użytkowników powinno mieć możliwość przesyłania między sobą plików.

5.14. Wyświetlanie emotikonów

Priorytet: niski.

Aplikacja E-Talk powinna zamieniać pewne określone ciągi znaków zapisane w tekście wiadomości na emotikony, które wyrażają w sposób symboliczny wyraz twarzy użytkownika.

6. Dokumentacja użytkownika

6.1. Serwer eTalk

6.1.1. Informacje ogólne

Program umożliwia wymianę danych między użytkownikami podłączonymi do sieci Internet. Aplikacja serwer eTalk powinna być uruchomiona na komputerze o dużej mocy obliczeniowej, aby móc przetwarzać wiele zapytań od klientów naraz.
Program umożliwia połączenie z bazą PostgreSQL, co wiąże się z uwierzytelnieniem. Osoba obsługująca serwer powinna być jednocześnie administratorem bazy oraz znać odpowiednią nazwę użytkownika i hasło dostępu do bazy.
Baza PostgreSQL, z której korzysta serwer eTalk może działać na localhost (pętla lokalna). W przypadku, gdy baza jest na innym komputerze należy zmienić ustawienia domyślne serwera, podając odpowiedni adres IP i numer portu.
Po połączeniu z bazą w centralnej części okna głównego aplikacji, serwer udostępnia dane odnośnie zarejestrowanych użytkowników, które są odczytywane na bieżąco z bazy. Dostępne dane można w dowolnym momencie odświeżać, monitorując aktualny stan bazy danych.
Właściwa praca serwera rozpoczyna się po połączeniu z bazą i rozpoczęciu nasłuchiwania na wybranym porcie. Od tej pory serwer powinien poprawnie obsługiwać żądania klienta i poprawnie na nie odpowiadać.

6.1.2. Łączenie się bazą danych PostgreSQL

Okno łączenia się z bazą danych PostgreSQL można wyświetlić na dwa sposoby:

- wybierając przycisk „Ustawiania” w lewym górnym rogu głównego okna aplikacji, a następnie wybierając przycisk „Połącz z bazą” z wyświetlonego menu;
- korzystając ze skrótu klawiszowego Ctrl+L.

W modyfikowalnych polach tekstowych wyświetlonego okna należy podać odpowiednio nazwę użytkownika, hasło dostępu do bazy danych PostgreSQL i zatwierdzić swój wybór przyciskiem „OK". Wybranie przycisku „Anuluj" spowoduje zamknięcie okna i nie dojdzie do nawiązania połączenia z bazą danych PostgreSQL. W przypadku próby podania niepoprawnej nazwy użytkownika lub hasła dostępu do bazy danych, program nie pozwoli na jego zatwierdzenie i nie dojdzie do połączenia z bazą danych.
Lista podstawowych danych zarejestrowanych klientów eTalk wyświetla się w głównym oknie aplikacji. Wybranie przycisku „Odśwież" spowoduje pobranie aktualnej listy podstawowych danych zarejestrowanych użytkowników i wyświetlenie jej. Wybranie z listy zarejestrowanego użytkownika spowoduje wyświetlenie jego dokładnych danych osobowych na liście poniżej.

6.1.3. Włączanie serwera

Warunkiem koniecznym, aby móc włączyć serwer jest wcześniejsze połączenie się z bazą danych PostgreSQL. Okno konfiguracyjne połączenia serwera można wyświetlić na dwa sposoby:

- wybierając przycisk „Ustawiania" w lewym górnym rogu głównego okna aplikacji, a następnie wybierając przycisk „Nasłuchuj" z wyświetlonego menu;
- korzystając ze skrótu klawiszowego Ctrl+N.

W modyfikowalnych polach tekstowych wyświetlonego okna należy podać odpowiednio adres IP oraz numer portu, na którym serwer ma nasłuchiwać i zatwierdzić swój wybór przyciskiem „OK". Istnieje możliwość skorzystania z jednego z dwóch przycisków:

- wybierając przycisk „Moje IP" w pierwszym polu tekstowym pojawia się aktualny adres sieciowy komputera,
- wybierając przycisk „Localhost" w pierwszym polu tekstowym pojawia się adres `127.0.0.1`.

Po włączeniu wielowątkowego serwera, aplikacja jest gotowa obsługiwać żądania klientów. Wybranie przycisku „Anuluj" spowoduje zamknięcie okna i nie dojdzie do nawiązania połączenia.

6.1.4. Wyjście z programu

Użytkownik może wyjść z programu bezpośrednio, zamykając główne okno aplikacji za pomocą przycisku w prawym górnym rogu lub wybierając przycisk „Wyjście" z menu głównego programu.

6.1.5. Uwagi odnośnie kompilacji i uruchamiania programu

Program został napisany w całości w języku C# przy użyciu zestawu narzędzi programistycznych Microsoft Visual Studio 2008. Wykorzystano następujące biblioteki spoza standardu języka:

- `Npgsql` – darmowa biblioteka do obsługi połączenia z bazą danych PostgreSQL. Wszystkie potrzebne do uruchomienia serwera eTalk pliki `.DLL` znajdują się w katalogu `bin` aplikacji;
- `HAKGERSoft` – darmowa biblioteka do obsługi protokołu komunikatora Gadu-Gadu. Wszystkie potrzebne do uruchomienia serwera eTalk pliki `.DLL` znajdują się w katalogu `bin` aplikacji.

Nie gwarantujemy, że przy innej wersji zintegrowanego środowiska programistycznego, a w szczególności innej wersji kompilatora języka C# serwer eTalk będzie działał poprawnie. Aplikacja była testowana pod systemem Windows XP Professional z zainstalowanym dodatkiem Service Pack 2 i jest przeznaczona wyłącznie dla systemów operacyjnych Microsoft Windows.

6.2. Klient eTalk

7. Model konceptualny bazy danych

7.1. Profil klienta

Do tej kategorii należą wszyscy klienci, zarejestrowani w systemie. Użytkownicy powinni mieć dostęp jedynie do swoich kont. Nie powinni mieć dostępu do żadnych danych innych użytkowników systemu. Od klienta nie wymaga się żadnej wcześniejszej wiedzy o systemie. Każdy klient dostanie podręcznik użytkownika zawierający opis wszystkich możliwości systemu. Ważne jest, aby opis był zrozumiały nawet dla bardzo niedoświadczonych użytkowników.

7.2. Administratorzy systemu

Administratorzy, to osoby odpowiedzialne za prawidłowe działanie systemu i zarządzanie bazą danych. Administratorzy powinni zostać gruntownie przeszkoleni tak, aby mogli sprawnie zarzadzać systemem.
Administratorzy mają dostęp do większości danych związanych z działaniem systemu i nieograniczone uprawnienia związane z modyfikowaniem zawartości bazy danych. Informacje o klientach powinni otrzymywać jedynie w sytuacjach, w których jest to niezbędne. Aby dochować tajemnicy korespondencji, nawet administratorzy nie mają dostępu do relacji `wiadomości`.

7.3. Opis encji

7.3.1. Użytkownik

Relacja `użytkownik` przechowuje podstawowe informacje o klientach, które są konieczne do prawidłowego działania systemu uwierzytelniania. Relacja zawiera kolejno następujące atrybuty:

- numer konta, który jest kluczem publicznym relacji `użytkownik` i jest automatycznie generowany przez sekwencję `num_gen` po rejestracji klienta;
- unikatową nazwę użytkownika, która jednoznacznie określa klienta w bazie danych;
- hasło, które użytkownik podaje podczas rejestracji, składające się z co najmniej 6 znaków alfanumerycznych;
- data ostatniego zapytania;
- status, możliwe wartości: „Dostepny", „Niedostepny" i „Niewidoczny".

7.3.2. Dane

Relacja `dane` przechowuje dodatkowe informacje o kliencie, które nie są konieczne do prawidłowego działania systemu. Dane mogą okazać się jednak pomocne przy wyszukiwaniu użytkowników systemu e-Talk. Relacja zawiera kolejno następujące atrybuty:

- numer konta, który jest kluczem publicznym relacji `dane` i jest kluczem obcym z relacji `użytkownik`;
- imię, które jest polem wymaganym do wypełnienia podczas rejestracji;
- nazwisko, które jest polem wymaganym do wypełnienia podczas rejestracji;
- kod pocztowy, który nie jest wymaganym polem do wypełnienia podczas rejestracji. Poprawność tego pola jest sprawdzana na podstawie domeny `kod_type` opartej na prostym wyrażeniu regularnym;
- e-mail, który jest polem wymaganym do wypełnienia podczas rejestracji. Poprawność tego pola jest sprawdzana na podstawie domeny `email_type` opartej na prostym wyrażeniu regularnym;
- data urodzenia, która nie jest wymaganym polem do wypełnienia podczas rejestracji;
- zainteresowania, które użytkownik wypełnia dowolnie, według własnego uznania.

7.3.3. Wiadomości

Relacja `wiadomości` przechowuje informacje o nieodebranych wiadomościach. Dane potrzebne są podczas pobierania z serwera listy nowych wiadomości dla klienta. Relacja zawiera kolejno następujące atrybuty:

- identyfikator wiadomości, który jest kluczem publicznym relacji `wiadomości` i jest automatycznie generowany przez sekwencję `wiad_gen` po wysłaniu przez klienta nowej wiadomości;
- nazwa nadawcy;
- nazwa odbiorcy;
- treść wiadomości.

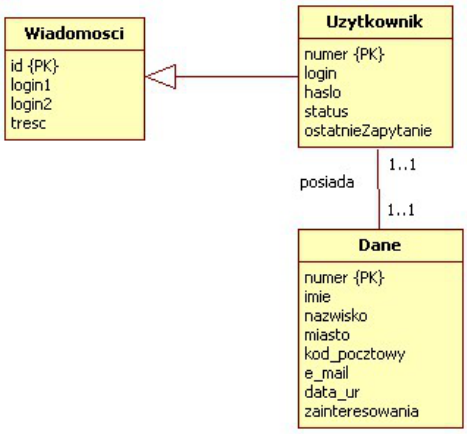
7.4. Dodatkowe elementy bazy danych

W celu przyspieszenia procesu wyszukiwania i sortowania założono indeksy na następujących atrybutach:

- nazwisko w relacji `dane`,
- login w relacji `użytkownik`,
- nazwa odbiorcy w relacji `wiadomości`.

Oprócz powyżej opisanych indeksów, system zarządzania bazą danych założył automatycznie indeksy na kluczach publicznych wszystkich relacji bazy danych.

Rysunek 1: Diagram UML



8. Dokumentacja techniczna

8.1. Serwer eTalk

8.1.1. Opis programu

Kod źródłowy programu znajduje się w następujących plikach:

- `About.cs`,
- `ClientRequest.cs`,
- `GaduGadu.cs`,
- `ListenerWindow.cs`,
- `LoginWindow.cs`,
- `MainForm.cs`,
- `MessageFactory.cs`,
- `QueryMaker.cs`,
- `Serwer.cs`.

Serwer eTalk jest napisany w języku C#, wykorzystuje paradygmat programowania obiektowego i składa się z następujących klas:

- `About`,

- `ClientRequest`,
- `ClientRequestParams`,
- `GaduGadu`,
- `HandleClient`,
- `ListenerWindow`,
- `LoginWindow`,
- `MainForm`,
- `MessageFactory`,
- `Pair`,
- `QueryMaker`,
- `Serwer`.

8.1.2. Opis klasy About

`About` jest publiczną klasą formularza dziedziczącą po bibliotecznej klasie `Form` języka C#. Formularz wyświetla informacje o programie i autorach.

8.1.3. Opis klasy ClientRequest

`ClientRequest` jest publiczną klasą odpowiedzialną za przetwarzanie żądań klienta, które są przesyłane w formacie XML. Format wiadomości powinien być zgodny z protokołem komunikacji opisanym w tym dokumencie.
Metoda `getType` zwraca typ otrzymanej wiadomości.
Metoda `getParams` zwraca główne parametry otrzymanej wiadomości i ewentualnie dodatkowe parametry, jeżeli zostały przesłane. Parametry są przechowywane w obiekcie klasy `ClientRequestParams`.

8.1.4. Opis klasy ClientRequestParams

`ClientRequestParams` stanowi opakowanie dla głównych i ewentualnie dodatkowych parametrów żądania klienta w formacie XML. Zwracanie i ustawianie parametrów jest możliwe dzięki mechanizmowi indekserów (indeksowanie tablicy nazwami parametrów).
Metoda `getExtraData` zwraca dodatkowe parametry, jeżeli zapytanie takie zawiera.

8.1.5. Opis klasy GaduGadu

`GaduGadu` jest publiczną klasą odpowiedzialną za obsługę protokołu komunikatora GaduGadu. Implementując klasę GaduGadu autorzy projektu wykorzystali możliwości darmowej biblioteki HAKGERSoft. Metody `connect` i `disconnect` służą odpowiednio do łączenia się i rozłączania się z serwerem. Operacja łącznia się z serwerem wiąże się z procesem uwierzytelniania klienta.
Metoda `sendMessage` pozwala na wysłanie pod dany numer, danej w parametrze procedury wiadomości. Aby wysłać wiadomość użytkownik musi być uwierzytelniony.
Metoda `gadu_GGMessageReceive` realizuje funkcję nasłuchiwacza wiadomości przychodzących do uwierzytelnionego użytkownika.
Metoda `changeStatus` zmienia status uwierzytelnionego użytkownika na status podany w pierwszym parametrze metody. Zmianie ulega również opis użytkownika. Nowy opis przekazywany jest jako drugi parametr metody.
Metoda `sendImage` wysyła obrazki wraz z wiadomością tekstową do adresata. W trakcie przesyłania obrazek jest pamiętany w strumieniu pamięci (obiekt bibliotecznej klasy `MemoryStream`). Klient powinien określić zawartość wiadomości tekstowej, numer adresata, pozycję obrazka w tekście, oraz sam obrazek przechowywany w obiekcie bibliotecznej klasy `Image` i przekazać te dane jako parametry metody. Metoda `getMsg` zwraca parę (nadawca, treść wiadomości) i jest wywoływana przez metodę nasłuchiwacza `gadu_GGMessageReceive`.

8.1.6. Opis klasy HandleClient

`HandleClient` jest publiczną klasą odpowiedzialną za obsługę żądań klienta eTalk. Metoda `getMessage` wykorzystując obiekt klasy bibliotecznej `NetworkStream` języka C# pobiera żądanie klienta w formacie XML zgodne z opisanym w tym dokumencie protokołem komunikacji.
Wykorzystując obiekty klas `ClientRequest` i `ClientRequestParams` klasa przetwarza żądanie i jego parametry.
Po rozpoznaniu typu zapytania i jego parametrów klasa wykorzystując obiekt `QueryMaker` wykonuje zadaną operację w bazie danych. Po poprawnym wykonaniu zadania, wysyła odpowiedź utworzoną przez obiekt klasy `MessageFactory`.
Metoda `sendMessage` przesyła odpowiedź klientowi wykorzystując obiekty bibliotecznych klas `StreamWriter` i `NetworkStream`.

8.1.7. Opis klasy ListenerWindow

`ListenerWindow` jest publiczną klasą formularza dziedziczącą po klasie bibliotecznej `Form` języka C#. Metoda wyświetla formularz zawierający dwa pole tekstowe przeznaczone na adres IP i numer portu. Podane informacje jednoznacznie określają gniazdo

nasłuchujące serwera eTalk. Po potwierdzeniu wprowadzonych danych program tworzy nowy obiekt klasy `Serwer` i rozpoczyna nasłuchiwanie na określonym porcie pod określonym adresem.

8.1.8. Opis klasy LoginWindow

`LoginWindow` jest publiczną klasą formularza dziedziczącą po klasie bibliotecznej `Form` języka C#. Metoda wyświetla formularz zawierający dwa pole tekstowe przeznaczone na nazwę użytkownika i hasło do bazy danych serwera eTalk. Po potwierdzeniu danych następuje połączenie z bazą danych PostgreSQL. Program tworzy nowy obiekt klasy bibliotecznej `NpgsqlConnection` i nawiązuje połączenie z bazą, której namiary podawane są w parametrze konstruktora.

8.1.9. Opis klasy MainForm

`MainForm` jest publiczną klasą głównego formularza aplikacji serwer eTalk dziedziczącą po klasie bibliotecznej `Form` języka C#. Klasa jest odpowiedzialna za wyświetlanie i konfigurowanie głównego okna aplikacji, jego komponentów oraz obsługę wszystkich zdarzeń z nimi związanych. Menu główne aplikacji jest obiektem klasy `MainMenu` i zawiera przyciski, które są obiektami klasy `MenuItem`. Dolne menu jest obiektem klasy `StatusBar` i zawiera następujące komponenty – obiekty klasy `StatusBarPanel`:

- panel informacji o komponencie, na który użytkownik wskazuje kursorem myszy (obsługa zdarzenia `Select`);
- panel z dniem tygodnia i datą (informacje pobrane z obiektu klasy `DateTime`);
- panel z godziną (informacje pobrane z obiektu klasy `Timer`, odświeżane co sekunda dzięki obsłudze zdarzenia `Tick`).

Klasy `MainForm` użyto również do składowania głównych obiektów serwera eTalk, np. listy połączeń.
W centrum głównego formularza znajduje się lista (obiekt bibliotecznej klasy `ListBox`) wszystkich użytkowników z bazy danych aplikacji. Poniżej opisanej listy znajduje się druga lista zawierająca rozszerzone informacje o użytkowniku wybranym z pierwszej listy (wykorzystano tutaj obsługę zdarzenia `SelectedIndexChanged` dla obiektu klasy `ListBox`). Obie listy można odświeżać.

8.1.10. Opis klasy MessageFactory

`MessageFactory` jest publiczną klasą służącą do wytwarzania odpowiedzi dla klienta. Każda kolejna metoda klasy potrafi zwrócić odpowiedź o ustalonym typie i zadanych parametrach.

Wszystkie odpowiedzi są w formacie XML i są tworzone na podstawie protokołu komunikacyjnego napisanego specjalnie na potrzeby aplikacji eTalk. Dokładny opis protokołu znajduje się w dalszej części tego dokumentu.

8.1.11. Opis klasy Pair

`Pair<T,U>` jest pomocniczą klasą generyczną przechowującą pary elementów.
Pierwszy element jest typu T, drugi element jest typu U. Dostęp do poszczególnych elementów jest możliwy dzięki własnościom `set` i `get` zaimplementowanym w języku C#.

8.1.12. Opis klasy QueryMaker

`QueryMaker` jest główną klasą wykonującą operacje na bazie danych PostgreSQL serwera eTalk.
Do łączenia się z bazą PostgreSQL autorzy projektu wykorzystali darmową bibliotekę `Npgsql` dla języka C#. Klasa `QueryMaker` posiada następujące metody:

- `addClient`, która dodaje nowego klienta do bazy. Informacje o nowym kliencie są zapisane w słowniku (obiekt klasy bibliotecznej `Dictionary<String,String>`). Metoda obsługuje każdą poprawną kombinację danych klienta (pewne pola mogą pozostać niewypełnione, reszta jest obowiązkowa). Metoda przetwarza relacje `użytkownik` oraz `dane` i wykorzystuje polecenia `INSERT` i zapytania `SELECT` w języku SQL. Metoda zwraca 1 jeżeli brakuje obowiązkowych danych, 2 jeżeli dochodzi do błędu połączenia z bazą i 0 jeżeli operacja przebiegnie pomyślnie;

- `changeDate`, zmienia datę i godzinę ostatniej operacji użytkownika zadanego jako parametr metody na obecną. Metoda wykorzystuje obiekt klasy bibliotecznej `DateTime` języka C# i polecenie `INSERT` w języku SQL;

- `changeStatus`, która zmienia stary status uwierzytelnionego klienta na nowy, zadany w drugim parametrze metody. Możliwe statusy to: `Dostepny`, `Niedostepny`, `Niewidoczny` i `Zaraz wracam`. Metoda przetwarza relację `użytkownik` i wykorzystuje polecenie `UPDATE` w języku SQL;

- `deleteClient`, która usuwa z bazy danych informacje o zadanym w parameterze kliencie. Metoda przetwarza relacje `użytkownik` oraz `dane` i wykorzystuje polecenia `DELETE` w języku SQL;

- `getClients`, która zwraca listę wszystkich użytkowników znalezionych w bazie serwera eTalk. Metoda przeszukuje relację `użytkownik` i wykorzystuje proste zapytanie `SELECT` w języku SQL;

- `getClientDetail`, która zwraca dodatkowe dane klienta zadanego przez parametr metody. Metoda przeszukuje relację `dane` i wykorzystuje proste zapytanie `SELECT` w języku SQL;
- `getMessage`, która zwraca wszystkie nieodebrane wiadomości użytkownika o zadanej nazwie w parametrze metody. Metoda wykorzystuje proste zapytanie `SELECT` w języku SQL, a następnie kasuje odebrane wiadomości poleceniem `DELETE` w języku SQL.
- `getStatus`, która przyjmuje jako parametr listę użytkowników i zwraca listę par postaci (użytkownik, status). Metoda wykorzystuje proste zapytanie `SELECT` w języku SQL;
- `login`, która sprawdza, czy podczas uwierzytelniania klient podał poprawną nazwę użytkownika i hasło. Metoda przeszukuje relację `użytkownik` i wykorzystuje proste zapytanie `SELECT` w języku SQL;
- `modifyClient`, która modyfikuje podstawowe i dodatkowe informacje o kliencie. Lista modyfikacji przekazywana jest w postaci słownika jako parametr metody. Metoda przetwarza relacje `użytkownik` oraz `dane` i wykorzystuje polecenia `UPDATE` i zapytania `SELECT` w języku SQL;
- `saveMessage`, która zapisuje wiadomość do bazy danych wykorzystując polecenie `INSERT` w języku SQL. Pierwszym argumentem jest nazwa nadawcy, drugim nazwa odbiorcy, a trzecim treść wysłanej wiadomości.

8.1.13. Opis klasy Serwer

`Serwer` jest publiczną klasą, która tworzy nowy obiekt klasy bibliotecznej `TcpListener` języka C#, nasłuchuje w oczekiwaniu na klientów na gnieździe nasłuchującym i zwraca gniazdo połączone, gdy dojdzie do połączenia. Klasa `Serwer` tworzy nowy obiekt klasy `HandleClient` przekazując jej w parametrze konstruktora, gniazdo połączone – obiekt bibliotecznej klasy `Socket` języka C# oraz numer porządkowy klienta (liczy się kolejność połączeń). Serwer jest aplikacją wielowątkową. Proces macierzysty tworzy wątki do obsługi każdego klienta z osobna. Dzięki zrównoleglonej obsłudze poszczególnych klientów, serwer eTalk działa szybciej. Metoda `start` rozpoczyna pracę serwera, a następnie ją kontroluje i ostatecznie kończy.

8.2. Klient eTalk

8.2.1. Opis programu

Klient eTalk jest programem przeznaczonym dla użytkownika końcowego. Program został napisany, aby umożliwić użytkownikom łatwe korzystanie z serwera eTalk. Program

ukrywa przed użytkownikiem szczegóły protokołu, prezentując w zamian przyjazne okna dialogowe.

8.2.2. Opis okna About

`About` jest publiczną klasą formularza dziedziczącą po klasie bibliotecznej `Form` języka C#. Formularz wyświetla informacje o programie i autorach.

8.2.3. Opis okna Archive

`Archive` jest klasą reprezentującą okno dialogowe, w którym użytkownik może przeglądać wcześniejsze rozmowy. Rozmowy są pogrupowane według rozmówcy.

8.2.4. Opis okna ChangePassword

`ChangePassword` jest klasą reprezentującą okno dialogowe, w którym użytkownik może zmienić podane podczas rejestracji hasło.

8.2.5. Opis okna CreateAccount

`CreateAccount` jest klasą reprezentującą okno dialogowe, w którym użytkownik może podać dane niezbędna do utworzenia nowego hasła.

8.2.6. Opis okna EditContact

`EditContact` jest klasą reprezentującą okno dialogowe, w którym użytkownik może zmieniać dane dotyczące osoby na jego liście kontaktów.

Okno reprezentowane przez klasę `EditContact` może pracować w dwóch trybach:

- tryb `add` uruchamiany jest w przypadku utworzenia okna dialogowego z wykorzystaniem konstruktora `EditContact()`. W tym trybie możliwe jest dodawanie nowych osób do listy kontaktowej,
- tryb `edit` uruchamiany jest w przypadku utworzenia okna dialogowego z wykorzystaniem konstruktora `EditContact(string name, string login)`. W tym trybie możliwe jest modyfikowanie osób znajdujących się na liście kontaktów.

8.2.7. Opis okna MainForm

`MainForm` jest klasą reprezentującą główne okno aplikacji. Klasa posiada wiele dodatkowych metod aktualizujących stan okna:

- `refreschContacts()` – metoda odświeża listę kontaktów po wprowadzeniu zmian,
- `enableMenuItems()` – metoda aktywuje wszystkie przyciski dostępne tylko dla zalogowanego użytkownika,
- `disableMenuItems()` – metoda odwrotna do metody `enableMenuItems()`,
- `updateStatus(string login, int index, string status)` – metoda aktualizuje status użytkownika po pobraniu danych z serwera.

Klasa `MainForm` odpowiada również za cykliczne sprawdzanie statusów użytkowników oraz tego, czy nie ma jakiś oczekujących na pobranie wiadomości.

8.2.8. Opis klasy SelectProfile

`SelectProfile` jest klasą reprezentującą okno dialogowe, w którym użytkownik może podać dane niezbędna do sprawdzenia jego tożsamości.

8.2.9. Opis okna Talk

`Talk` jest klasą reprezentującą okno dialogowe, w którym użytkownik może prowadzić rozmowę.

8.2.10. Opis klasy Talk

`Talk` to klasa kontenerowa przechowująca dane pojedynczej rozmowy.

8.2.11. Opis klasy Archive

`Archive` to klasa udostępniająca metody służące do operowania na archiwum rozmów. Programista może skorzystać z metod:

- `addMessage(int talkId, string login, string senderLogin, string senderName, DateTime time, string message )` – metoda służąca do dodawania wiadomości do archiwum,
- `readTalk(int talkId, string login)` – metoda służąca do czytania zapisu rozmowy,
- `createNewTalk(string login, int talkId, string talker, DateTime time)` – metoda tworząca nowy wpis w archiwum, z informacją o odbytej rozmowie,
- `readTalker(string login)` – metoda czytająca listę rozmówców, z którymi przeprowadziliśmy przynajmniej jedną rozmowę,

- `readTalks(string login, string talker)` – metoda czytająca listę rozmów przeprowadzonych z wybranym rozmówcą,
- `getNextId()` – metoda pobierająca kolejny wolny numer identyfikacyjny rozmowy.

8.2.12. Opis klasy Communicator

`Communicator` to klasa reprezentująca cały mechanizm działania komunikatora eTalk. Klasa została napisana z wykorzystaniem wzorca singleton. Instancję obiektu można pobrać korzystając z metody `getInstance()`. Pozostałe dostępne metody to:

- `login(string login, string password, MainForm form)` – metoda służąca do uwierzytelniania użytkownika,
- `createAccount(string login, string password, string name, string surname, string email, MainForm form)` – metoda służąca do zakładania nowego konta,
- `connectionFail()` – pomocnicza metoda informująca użytkownika o braku połączenia z serwerem eTalk,
- `readMessages()` – metoda odpytująca serwer, czy są jakieś wiadomości oczekujące na wysłanie do aktualnie zalogowanego użytkownika,
- `getLogedUser()` – metoda pobierająca nazwę aktualnie zalogowanego użytkownika,
- `sendMessage(string message, string to)` – metoda wysyłająca wiadomość do innego użytkownika,
- `changePassword(string newPassword)` – metoda umożliwiająca zmianę hasła zalogowanego użytkownika,
- `refreshContactsStatus(MainForm main)` – metoda pytająca serwer eTalk o aktualny status użytkowników znajdujących się na liście kontaktów,
- `changeStatus(string newStatus)` – metoda zmieniająca aktualny status zalogowanego użytkownika.

8.2.13. Opis klasy CommunicatorMessage

`CommunicatorMessage` to klasa kontenerowa reprezentująca pojedynczą wiadomość wysyłaną przez użytkownika.

8.2.14. Opis klasy Connection

`Connection` to klasa odpowiedzialna za nawiązanie połączenia z serwerem eTalk. Klasa jest zaimplementowana zgodnie z wzorcem singleton. Do pobieranie egzemplarza klasy należy użyć metody `getInstance()`. Dodatkowo klasa udostępnia metodę `sendMessage(string message)`, która wysyła zapytanie do serwera eTalk oraz odbiera odpowiedź.

8.2.15. Opis klasy Contact

`Contact` to pomocnicza klasa kontenerowa do przechowywania informacji o jednym użytkowniku znajdującym się na liście kontaktów.

8.2.16. Opis klasy Contacts

`Contacts` to klasa reprezentująca listę kontaktową. Klasa udostępnia metody:

- `loadFromFile(string file)` – metoda czyta listę kontaktów z zewnętrznego pliku,
- `saveToFile(string file)` – metoda zapisująca listę kontaktów do zewnętrznego pliku,
- `addContact(string name, string login)` – metoda umożliwiająca dodawanie osób do listy,
- `removeContact(string login)` – metoda usuwająca kontakt o podanej nazwie użytkownika,
- `updateContact(string name, string login, string oldLogin)` – metoda umożliwiająca zmianę danych osoby znajdującej się na liście kontaktów,
- `findContact(string login)` – metoda wyszukująca osobę o podanej nazwie użytkownika na liście kontaktów,
- `getList()` – metoda pobierająca listę.

8.2.17. Opis klasy Settings

`Settings` jest klasą przechowującą ustawienia programu.

8.2.18. Opis klasy User

`User` jest klasą reprezentującą użytkownika aplikacji. Na klasę składają się metody:

- `logedUser` – zwraca nazwę aktualnie zalogowanego użytkownika,
- `login(string number, string password)` – przeprowadza proces uwierzytelniania użytkownika,
- `createAccount(string number, string password, string name, string surname, string email)` – metoda służąca do zakładania nowego konta.

8.2.19. Opis klasy MessageFactory

`MessageFactory` jest publiczną klasą służącą do wytwarzania żądań dla klienta.
Każda kolejna metoda klasy potrafi zwrócić żądanie o ustalonym typie i zadanych parametrach.
Wszystkie żądania są w formacie XML i są tworzone na podstawie protokołu komunikacyjnego, napisanego specjalnie na potrzeby aplikacji eTalk. Dokładny opis protokołu znajduje się w dalszej części tego dokumentu.

8.2.20. Opis klasy ServerResponse

`SerwerResponse` jest publiczną klasą odpowiedzialną za przetwarzanie odpowiedzi serwera, które są przesyłane w formacie XML. Format wiadomości powinien być zgodny z protokołem komunikacji opisanym w tym dokumencie.
Metoda `getType` zwraca typ otrzymanej wiadomości.
Metoda `getParams` zwraca główne parametry otrzymanej wiadomości i ewentualnie dodatkowe parametry, jeżeli zostały przesłane. Parametry są przechowywane w obiekcie klasy `SerwerResponseParams`.

8.2.21. Opis klasy ServerResponseParams

`SerwerResponseParams` stanowi opakowanie dla głównych i ewentualnie dodatkowych parametrów odpowiedzi klienta w formacie XML. Zwracanie i ustawianie parametrów jest możliwe dzięki mechanizmowi indekserów (indeksowanie tablicy nazwami parametrów). Metoda `getExtraData` zwraca dodatkowe parametry, jeżeli zapytanie takie zawiera.

8.3. Opis protokołu komunikacyjnego aplikacji eTalk

8.3.1. Wstęp

Komunikacja pomiędzy użytkownikami odbywa się przez protokół TCP/IP z wykorzystaniem komunikatów w formacie XML. Każdy komunikat jest zbudowany zgodnie ze wzorem:

$<$ [*request*|*response*] $>$
  $<$ *type* $>$[nazwa akcji]$<$ /*type* $>$
  [parametry dodatkowe]
$<$ /[*request*|*response*] $>$

Część [nazwa akcji] określa cel wysłania komunikatu, a sekcja [dane dodatkowe] zawiera informacje potrzebne do wykonania danej akcji, różne, dla różnych operacji.

8.3.2. Uwierzytelnianie

Klient przesyła na serwer żądanie postaci:

$<$ *request* $>$
  $<$ *type* $>$ *login* $<$ /*type* $>$
  $<$ *params* $>$
    $<$ *param name* = "*number*" *value* = *numer* / $>$
    $<$ *param name* = "*password*" *value* = *haslo* / $>$
  $<$ /*params* $>$
$<$ /*request* $>$

Przesyłane są dwa parametry o nazwach: „number” i „password”, ich wartości to odpowiednio numer użytkownika i hasło.
Serwer przesyła odpowiedź postaci:

$<$ *response* $>$
  $<$ *type* $>$ *login* $<$ /*type* $>$
  $<$ *params* $>$
    $<$ *param name* = "*result*" *value* = [*success*|*fail*] / $>$
  $<$ /*params* $>$
$<$ /*response* $>$

Przesyłany jest jeden parametr o nazwie „result”, jego wartości to napisy „success”

lub „fail” w zależności od tego, czy uwierzytelnie powiodło się lub nie.

8.3.3. Rejestracja

Klient przesyła na serwer żądanie postaci:

$$
\begin{aligned}
&< request > \\
&\quad < type > createAccount < /type > \\
&\quad < params > \\
&\qquad < param\ name = "username"\ value = nazwa\_uzytkownika\ / > \\
&\qquad < param\ name = "password"\ value = haslo\ / > \\
&\qquad < param\ name = "name"\ value = imie\ / > \\
&\qquad < param\ name = "surname"\ value = nazwisko\ / > \\
&\qquad < param\ name = "email"\ value = email\ / > \\
&\quad < /params > \\
&< /request >
\end{aligned}
$$

Przesyłanych jest pięć parametrów o nazwach: „username”, „password”, „name”, „surname”, „email”, ich wartości to odpowiednio numer użytkownika, hasło, imię, nazwisko i adres email.
Serwer przesyła odpowiedź postaci:

$$
\begin{aligned}
&< response > \\
&\quad < type > createAccount < /type > \\
&\quad < params > \\
&\qquad < param\ name = "result"\ value = [success|fail]\ / > \\
&\quad < /params > \\
&< /response >
\end{aligned}
$$

Przesyłany jest jeden parametr o nazwie „result”, jego wartości to napisy „success” lub „fail” w zależności od tego, czy rejestracja powiodła się lub nie.

8.3.4. Zmiana hasła

Klient przesyła na serwer żądanie postaci:

$$
\begin{aligned}
&< request > \\
&\quad < type > changePassword < /type > \\
&\quad < params > \\
&\qquad < param\ name = "username"\ value = nazwa\_uzytkownika\ / >
\end{aligned}
$$

$< param\ name = "newPassword"\ value = nowe\_haslo\ / >$
$< /params >$
$< /request >$

Przesyłane są dwa parametry o nazwach: „username” i „newPassword”, ich wartości to odpowiednio nazwa użytkownika i nowe hasło.
Serwer przesyła odpowiedź bez parametrów postaci:

$< response >$
$< type > changePassword < /type >$
$< /response >$

8.3.5. Zmiana statusu

Klient przesyła na serwer żądanie postaci:

$< request >$
$< type > changeStatus < /type >$
$< params >$
$< param\ name = "username"\ value = nazwa\_uzytkownika\ / >$
$< param\ name = "newStatus"\ value = nowy\_status\ / >$
$< /params >$
$< /request >$

Przesyłane są dwa parametry o nazwach: „username” i „newStatus”, ich wartości to odpowiednio nazwa użytkownika i nowy status.
Serwer przesyła odpowiedź bez parametrów postaci:

$< response >$
$< type > changeStatus < /type >$
$< /response >$

8.3.6. Odświeżanie listy kontaktów

Klient przesyła na serwer żądanie postaci:

$< request >$
$< type > refreshContactsStatus < /type >$
$< params >$

```
        < param name = "users" value = liczba_kontaktow / >
        < param name = "usernameX" value = nazwa_uzytkownika / >
    < /params >
< /request >
```

Przesyłanych jest n+1 parametrów, gdzie n to liczba znajomych w kontaktach, a pierwszy parametr to nazwa użytkownika, który prosi o odświeżenie kontaktów. Lista kontaktów jest reprezentowana przez n kolejnych parametrów.
Serwer przesyła odpowiedź postaci:

```
< response >
    < type > refreshContactsStatus < /type >
    < params >
        < param name = "usernameX" value = obecny_status / >
    < /params >
< /response >
```

Serwer przesyła n parametrów, każdy parametr jako nazwę przyjmuje nazwę użytkownika, a wartością jest obecny status.

8.3.7. Wysyłanie wiadomości

Klient przesyła na serwer żądanie postaci:

```
< request >
    < type > sendMessage < /type >
    < params >
        < param name = "username" value = nazwa_nadawcy / >
        < param name = "message" value = tresc / >
        < param name = "to" value = nazwa_adresata / >
    < /params >
< /request >
```

Przesyłane są trzy parametry o nazwach: „username”, „message” i „to”, ich wartości to odpowiednio nazwa nadawcy, treść wiadomości i nazwa odbiorcy.
Serwer przesyła odpowiedź bez parametrów postaci:

```
< response >
    < type > sendMessage < /type >
```

$< /response >$

8.3.8. Odbieranie wiadomości

Klient przesyła na serwer żądanie postaci:

$< request >$
$< type > getMessages < /type >$
$< params >$
$< param\ name = "username"\ value = nazwa\_uzytkownika\ / >$
$< /params >$
$< /request >$

Przesyłany jest jeden parametr o nazwie „username", a wartością jest nazwa użytkownika, który chce odebrać swoje wiadomości. Serwer przesyła odpowiedź postaci:

$< response >$
$< type > getMessages < /type >$
$< params >$
$< param\ name = "messages"\ value = liczba\_wiadomosci\ / >$
$< param\ name = X\ value = tresc\ extra = nazwa\_nadawcy\ / >$
$< /params >$
$< /response >$

Serwer przesyła n+1 parametrów, gdzie n to liczba wiadomości dla danego użytkownika.
Pierwszy parametr to liczba wiadomości do odebrania. Kolejne n parametrów zawiera informacje o nieodebranych wiadomościach. Nazwa parametru to numer porządkowy wiadomości, wartość zawiera treść wiadomości, a paramter dodatkowy przechowuje nazwę nadawcy.

9. Raport z przeprowadzonych testów

9.1. Test 1.

Opis:
Uruchamiamy program serwera eTalk, łączymy się z bazą. Serwer wyświetla w centralnej części głównego okna zawartość relacji `użytkownik` i `dane`.

Uruchamiamy bazę danych PostgreSQL i wykorzystując proste zapytania `SELECT` języka SQL pobieramy dane z wyżej wymienionych relacji sprawdzając, czy rezultaty obu operacji się zgadzają.
Następnie zmieniamy zawartość obu relacji poprzez dodanie, odjęcie i zmodyfikowanie kilku krotek.
Wybieramy przycisk „Odśwież" i sprawdzamy, czy serwer poprawnie wyświetla w centralnej części głównego okna zmienioną zawartość relacji `użytkownik` i `dane`.
Cel:
Test ma na celu sprawdzenie poprawności komunikacji serwera eTalk z bazą danych PostgreSQL, poprawności wyświetlania pobranych danych i ich odświeżania.
Wynik:
Serwer eTalk prawidłowo pobiera, wyświetla i odświeża dane z bazy PostgreSQL.

9.2. Test 2.

Opis:
Uruchamiamy program serwera eTalk. Przy próbie połączenia z bazą celowo podajemy niepoprawną nazwę użytkownika i hasło. Serwer poprawnie wyświetla komunikat o błędzie i prosi o weryfikację wprowadzonych danych. Po zamianie danych na poprawne następuje połączenie z bazą PostgreSQL. Serwer wyświetla w centralnej części głównego okna zawartość relacji `użytkownik` i `dane`.
Następnie włączamy właściwą część aplikacji serwera eTalk rozpoczynając nasłuchiwanie pod adresem `localhost`, na porcie nr 7777. Staramy się nawiązać i sprawdzić połączenie poleceniem systemowym `telnet`. Powyższy test powtarzamy dwa razy. Za 1. razem wykorzystujemy przyciski w górnym menu aplikacji, a za 2. razem zdefiniowane skróty klawiszowe.
Cel:
Test ma na celu sprawdzenie poprawności komunikacji serwera eTalk z bazą danych PostgreSQL, zabezpieczenie przed podaniem błędnych danych i nawiązywanie połączenia z danym gniazdem.
Wynik:
Serwer eTalk prawidłowo reaguje na błędne dane, poprawkę błędnych danych i nawiązuje połączenie na określonym gnieździe.

9.3. Test 3.

Opis:
Zakładamy, że serwer eTalk został poprawnie uruchomiony, połączony z bazą danych PostgreSQL i nasłuchuje pod zadanym adresem. Uruchamiamy program klienta eTalk. Zakładamy nowe konto zostawiając luki w wypełnionych danych. Klient eTalk informuje o braku wymaganych przy rejestracji danych. Dopisujemy brakujące dane i wysyłamy

je na serwer, który dodaje je do bazy danych i wysyła klientowi potwierdzenie udanej rejestracji.
Cel:
Test ma na celu sprawdzenie poprawności komunikacji serwera eTalk z klientem eTalk przy rejestracji nowego konta i sprawdzenie zabezpieczenia przed brakiem niezbędnych przy rejestracji danych.
Wynik:
Klient eTalk prawidłowo przesyła dane, serwer eTalk prawidłowo dodaje dane do bazy. Rejestracja przebiegła pomyślnie.

9.4. Test 4.

Opis:
Wykonując ten test zakładamy, że wykonaliśmy test 3.
Uwierzytelniamy się podając błędne dane nieistniejącego konta. Klient przesyła podaną nazwę użytkownika i hasło do serwera, który porównuje te dane z wpisami z bazy danych. Odpowiedź serwera jest negatywna i klient eTalk prosi o ponowne podanie danych.
Uwierzytelniamy się podając dane konta założonego w teście 3. Klient przesyła podaną nazwę użytkownika i hasło do serwera, który sprawdza te dane z wpisami z bazy danych. Odpowiedź serwera jest tym razem pozytywna i klient eTalk wyświetla w centralnej części głównego okna aplikacji poprawną listę kontaktów i odpowiadających im statusów.
Cel:
Test ma na celu sprawdzenie poprawności komunikacji serwera eTalk z klientem eTalk przy uwierzytelnianiu użytkownika i sprawdzenie zabezpieczenia przed podaniem błędnych danych. Ponadto test sprawdza, czy wyświetlają się wszystkie kontakty użytkownika i czy statusy tych kontaktów są zgodne z wpisami w bazie danych PostgreSQL serwera eTalk.
Wynik:
Uwierzytelnienie przebiegło pomyślnie, wyświetlona lista kontaktów wraz ze statusami jest poprawna.

9.5. Test 5.

Opis:
Wykonując ten test zakładamy, że wykonaliśmy testy 3. i 4.
Zmieniamy hasło, wylogowujemy się i uwierzytelniamy się ponownie. Za pierwszym razem podajemy stare hasło, które powoduje zgłoszenie błędu przez klienta eTalk i prośbę o zmianę wprowadzonych danych. Podajemy nowe, poprawne hasło. Tym razem proces uwierzytelnienia przebiega poprawnie.

Cel:
Test ma na celu sprawdzenie poprawności zmiany hasła dla danego konta i poprawności operacji wylogowania.
Wynik:
Klient eTalk poprawnie zmienia hasło i poprawnie wylogowuje użytkownika.

9.6. Test 6.

Opis:
Wykonując ten test zakładamy, że wykonaliśmy dwukrotnie testy 3. i 4., tzn. mamy założone 2 konta i uwierzytelniliśmy się 2 razy.
Dodajemy nowy kontakt dla 1. konta, jest nim 2. konto. Analogiczną operację przeprowadzamy dla 2. konta. Kilkakrotnie zmieniamy statusy obu kont sprawdzając, czy obie listy kontaktów wraz ze statusami są poprawnie odświeżane.
Cel:
Test ma na celu sprawdzenie poprawności dodawania nowych kontaktów i automatycznego odświeżania statusów.
Wynik:
Klient eTalk poprawnie dodaje nowe kontakty i automatycznie odświeża statusy.

9.7. Test 7.

Opis:
Wykonując ten test zakładamy, że wykonaliśmy test 6.
Wybieramy 1. konto. Dwukrotnie klikamy lewym przyciskiem myszy na dodany w teście 6. kontakt.
Klient eTalk poprawnie wyświetla okno rozmowy, które jest na początku puste. Wpisujemy treść wiadomości i potwierdzamy przyciskiem „Wyślij". W oknie pojawia się zatwierdzona treść wiadomości.
Jednocześnie 2. konto wyświetla okno rozmowy, a w nim wiadomość i nadawcę. Odpowiadamy wpisują treść 2. wiadomości i zatwierdzając przyciskiem „Wyślij". Wymieniamy około 50 wiadomości pomiędzy dwoma uwierzytelnionymi użytkownikami.
Cel:
Test ma na celu sprawdzenie poprawności przesyłania i automatycznego odbierania wiadomości.
Wynik:
Klient eTalk poprawnie przesyła i odbiera wiadomości w czasie rzeczywistym.

9.8. Test 8.

Opis:
Wykonując ten test zakładamy, że wykonaliśmy test 6.
Wylogowujemy się z 2. konta. Wybieramy 1. konto. Dwukrotnie klikamy lewym przyciskiem myszy na dodany w teście 6. kontakt.
Klient eTalk poprawnie wyświetla okno rozmowy, które jest na początku puste. Wpisujemy treść wiadomości i potwierdzamy przyciskiem „Wyślij". W oknie pojawia się zatwierdzona treść wiadomości.
Wysyłamy około 10 wiadomości do tego samego użytkownika.
Uwierzytelniamy się używając danych z 2. konta. Serwer poprawnie przesyła wszystkie zaległe, nieodebrane wiadomości, we właściwej kolejności.
Cel:
Test ma na celu sprawdzenie poprawności przesyłania i odbierania zaległych wiadomości.
Wynik:
Klient eTalk poprawnie przesyła i odbiera zaległe wiadomości.

10. Przyszły rozwój aplikacji

10.1. Obsługa protokołu Gadu-Gadu

Serwer eTalk posiada klasę `GaduGadu`, która działa na bazie darmowej biblioteki HAKGERSoft. Napisana klasa teoretycznie powinna w pełni wykorzystywać protokół komunikatora Gadu-Gadu i umożliwiać klientowi pełny dostęp do konta na tym serwerze. Niestety w trakcie testowania aplikacji serwera okazało się, że biblioteka jest niestabilna i nie potrafi obsługiwać niektórych serwerów Gadu-Gadu. Problemy wynikają z faktu, że twórcy od 2008 roku (rok premiery) zaprzestali prac nad uaktualnianiem biblioteki. W związku z zawodnością i brakiem alternatywy w postaci innej, nowszej biblioteki obsługującej protokół komunikatora Gadu-Gadu, twórcy komunikatora eTalk zawiesili prace nad obsługą protokołu i usunęli z aplikacji klienta eTalk możliwość obsługi konta Gadu-Gadu. Wraz z pojawieniem się nowej wersji, którejś ze starych bibliotek lub zupełnie nowej, twórcy aplikacji planują wznowienie prac nad obsługą najpopularniejszego protokołu komunikacyjnego w Polsce.

Literatura

[1] http://www.seoteka.pl/a85.php (ostatni dostęp do strony 2010-04-12)

[2] *Polityka*, nr. 11 (2747), 2010-03-13

[3] S.Turkle, *Life on the Screen*, Simon & Schuster, 1997

[4] http://info.gadu-gadu.pl/producent/historia (ostatni dostęp do strony 2010-04-12)

[5] http://www.pcworld.pl/artykuly/41638/Lepsze.niz.Gadu.Gadu.html (ostatni dostęp do strony 2010-04-12)

[6] http://firma.o2.pl/ (ostatni dostęp do strony 2010-04-12)