Dokumentacja wstępna projektu TIN „Rozgłaszacz internetowy”.

Autorzy: Joanna Raczyńśka, Kamil Kacperski, Michał Mudel, Wojciech Zieliński

Prowadzący: dr inż. Piotr Gawkowski

# 1.Wstęp

Niniejszy dokument przedstawia projekt wstępny programu pt. „Rozsyłacz pocztowy” realizowanego w ramach przedmiotu TIN na wydziale EiTI Politechniki Warszawskiej. Głównym założeniem projektu jest stworzenie aplikacji, umożliwiającej proste i szybkie wysyłanie wiadomości – powiadomień do wybranej grupy odbiorców. Każdy klient posiada własne grupy i szablony wiadomości. Na ich podstawie wysyła wiadomości mailowe.

# 2. Słownik dziedziny problemu

2.1 Klient – użytkownik systemu korzystający z części klienckiej aplikacji.

2.2 Administrator – użytkownik systemu zarządzający aplikacją po stronie serwera.

2.3 Grupa – zbiór adresów mailowych, będący podmiotem wysyłania wiadomości.

2.4 Szablon wiadomości – treść wiadomości mogąca posiadać pola wymagające uzupełnienia

2.5 Wiadomość – treść wiadomości z wypełnionymi wszystkimi polami, gotowa do wysłania.

2.6 Adres – adres mailowy który jest przechowywany w grupie.

2.7 Pole szablonu – tekst który musi zostać uzupełniony w szablonie wiadomości przed jej wysłaniem.

2.8 Sesja klienta – połączenie z danym klientem w danym czasie.

2.9 Aplikacja kliencka – aplikacja uruchamiana na komputerze klienta.

2.10 Aplikacja serwer – aplikacja uruchamiana przez administratora na serwerze, obsługująca klientów.

# 3. Wymagania funkcjonalne

3.1 Klient loguję się do systemu.

3.2 Klient tworzy nowy szablon wiadomości

3.3 Klient modyfikuje wcześniej utworzony szablon wiadomości.

3.4 Klient usuwa szablon wiadomości.

3.5 Klient wyświetla nazwy wszystkich szablonów wiadomości.

3.6 Klient wyświetla całą treść wybranego szablonu wiadomości.

3.7 Klient tworzy nową grupę.

3.8 Klient usuwa grupę.

3.9 Klient wyświetla grupę.

3.9 Klient dodaje adres do grupy.

3.10 Klient usuwa adres z grupy.

3.11 Klient wysyła wiadomość.

3.12 Klient wylogowuje się z systemu.

3.13 Administrator wyświetla wszystkie obecnie otwarte sesje klientów.

3.14 Administrator kończy wybraną sesję z klientem.

3.15 Administrator blokuje wybranego klienta.

3.16 Administrator odblokowuje wybranego klienta.

3.17 Administrator tworzy nowego klienta.

3.18 Administrator usuwa wybranego klienta.

# 4. Wymagania niefunkcojnalne

4.1 Aplikacja kliencka będzie uruchamiana na platformach Windows 7, Windows 8, Windows 10.

4.2 Aplikacja serwera będzie uruchamiana na platformach Windows 7, Windows 8, Windows 10, Windows Server 2012.

4.3 Aplikacja kliencka zostanie wykonana w technologi Windows Presentation Foundation, C# i .NET 4.6.

4.4 Aplikacja serwerowa zostanie wykonana za pomocą języka C++ 14.

4.5 Sesja kliencka jest kończona po 5 minutach braku komunikacji.

4.6 Sesja kliencka jest kończona jeśli wysłane przez klienta żądanie jest niezgodne z protokołem ASIA.

4.7 Sesja kliencka jest kończona jeśli wysyła więcej niż 10 żądań na sekundę.

4.8 Wszystkie sesje klienckie są zamykane jeśli do serwera przychodzi więcej niż 10000 żądań na sekundę.

4.9 W przypadkach 4.6 i 4.7 konto klienta jest blokowane.

4.10 Natychmiast po utracie łączności wszystkie wątki składające się na komunikacje klienta z serwerem są zakańczane a gniazda zamykane.

4.11 Jako baza danych zostanie wykorzystany MS SQL Server 2014 Standard Edition with Service Pack 1.

# 5. Szczegółowy opis modułów aplikacji i interfejsów pomiędzy nimi. ( musi być kompletny i jednoznaczny )

# 6. Opis protokołu SMTP

Struktura ramek smtp by się przydała jakaś? Co jest na jakim bajcie i co to oznacza.

Schemat działania połączenia:

Wątek rozsyłający zwany dalej kentem ustanawia połączenie z serwerem SMTP osoby do której wysyłamy wiadomość i czeka aż ten odeśle komunikat 220 READY FOR MAIL. Po odebraniu komunikatu klient odpowiada HELO a serwer odpowiada własnym identyfikatorem. Po ustanowieniu połączenia wątek może wysłać jeden lub więcej wiadomości. Może również w każdym momencie zakończyć połączenie.

Wymiana poczty rozpoczyna się poleceniem MAIL ( definiuje nadawcę ) FROM adres ( pod ten adres wysłane zostaną wszystkie informacje o błędach ) Po pomyślnym wykonaniu plecenia MAIL nadawca wysyła serię poleceń RCPT adres ( identyfikując odbiorcę/ów listu ) i czeka na ich potwierdzenie przez serwer. Następnie nadawca wysyła polecenie DATA określające gotowość do przesłania całego listu. Odbiorca odpowiada komunikatem 354 Start mail input i określa ciąg znaków odpowiadających zakończeniu listu.

Gdy klient kończy wysyłanie listów do danego odbiorcy, może wysłać plecenie TURN, które odwraca połączenie i sprawia że odbiorca przejmuje nad nim kontrolę. Zakończenie połączenia można wykonać wysyłając polecenie QUIT i czekając na odpowiedzą 221 ( zgoda na rozłączenie )

# 7. Opis autoryskiego protokołu o nazwie ASIA (Automated Sending Information to Addresses)

|  |  |
| --- | --- |
| KOD KOMUNIKATU | LICZBA DANYCH |
| KOD DANEJ | DŁUGOŚĆ DANEJ |
| DANA  ..... | |
| ..... | |
| KOD DANEJ | DŁUGOŚĆ DANEJ |
| DANA  ...... | |

3 BAJTY

## 7.1 Typy komunikatów

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nazwa | Inicjator | Odpowiedź | Kod | Dane |
| 7.1.1 | AUTH | Serwer | AUTH | 0x00 | ASYMKEY |
| Serwer wysyła asymetryczny klucz publiczny | | | | |
| 7.1.2 | AUTH | Klient | ACK | 0x00 | SYMKEY, LOGIN, PASSW |
| Klient wysyła zaszyfrowany klucz symetryczny i dane do autoryzacji | | | | |
| 7.1.3 | ACK | Serwer | - | 0x01 | - |
| Serwer potwierdza poprzednie żądanie klienta | | | | |
| 7.1.4 | MSGCREATE | Klient | ACK | 0x02 | MSGTMPLT, MSGNAME |
| Klient tworzy nowy szablon wiadomości | | | | |
| 7.1.5 | MSGMODIFY | Klient | ACK | 0x03 | MSGID, MSGTMPLT, MSGNAME |
| Klient modyfikuje szablon wiadomości | | | | |
| 7.1.6 | MSGDELETE | Klient | ACK | 0x04 | MSGID |
| Klient usuwa szablon wiadomości | | | | |
| 7.1.7 | MSGGETALL | Klient | MSGGETALL | 0x05 | - |
| Klient pobiera wszystkie szablony wiadomosci | | | | |
| 7.1.8 | MSGGETALL | Serwer | - | 0x05 | {MSG} |
| Serwer wysyła wszystkie szablony wiadomosci | | | | |
| 7.1.9 | MSGGETONE | Klient | MSGGETONE | 0x06 | MSGID |
| Klient pobiera wybraną wiadomość | | | | |
| 7.1.10 | MSGGETONE | Serwer | - | 0x06 | MSG |
| Serwer wysyła jedną wiadomość | | | | |
| 7.1.11 | GRPCREATE | Klient | ACK | 0x07 | GRPNAME |
| Klient tworzy grupę | | | | |
| 7.1.12 | GRPDELETE | Klient | ACK | 0x08 | GRPID |
| Klient usuwa grupę | | | | |
| 7.1.13 | GRPGETONE | Klient | GRPGETONE | 0x09 | GRPID |
| Klient pobiera jedną grupę | | | | |
| 7.1.14 | GRPGETONE | Serwer | - | 0x09 | GRP |
| Serwer wysyła jedną grupę | | | | |
| 7.1.15 | GRPGETALL | Klient | GRPGETALL | 0x0A | - |
| Klient pobiera wszystkie grupy | | | | |
| 7.1.16 | GRPGETALL | Serwer | - | 0x0A | {GRP} |
| Serwer wysyła wszystkie grupy | | | | |
| 7.1.17 | GRPADRADD | Klient | ACK | 0x0B | GRPID, ADDRVAL |
| Klient dodaje adres do grupy | | | | |
| 7.1.18 | GRPADRRMV | Klient | ACK | 0x0C | GRPID, ADDRID |
| Klient usuwa adres z grupy | | | | |
| 7.1.19 | SEND | Klient | ACK | 0x0D | GRPID, MSGID, {TMPLT} |
| Klient wysyła wiadomość | | | | |

## 7.2 Typy Danych

# 9. Scenariusze użycia

# 10. Diagram stanów sesji klienta

# 11. Przykłady testowe ( uzasadnienie i nawiązanie do wymagań funkcjonalnych i niefunkcjonalnych)

# 12. Podział pracy pomiędzy członków zespołu ( każdy z członków zespołu ma jasno określone prace do wykonania )

13. Analiza sytuacji krytycznych i propozycje ich rozwiązania

a) chwilowa/trwała utrata łączności między węzłami,

b) błędne pakiety

c) przekroczone limity czasowe

d) scenariusz "wstawania" poszczególnych węzłów systemu

14. Diagram sekwencji dla modułu klienta i serwera

/////STARE

# 2.Szkic rozwiązania

Aplikacja zorganizowana będzie w strukturze klient – serwer. Każdy egzemplarz aplikacji użytkownika będzie klientem, który łączył się będzie z publicznym serwerem. Komunikacja między klientem a serwerem oparta będzie o protokół TCP i nadbudowanym na nim autorskim protokołem służącym do komunikacji w tworzonej aplikacji. Na serwerze przechowywane będą zdefiniowane grupy użytkowników (pogrupowane odpowiednio adresy e-mail) oraz szablony wiadomości. Każdy klient będzie mógł korzystać z danych dostępnych na serwerze, a także będzie miał możliwość ich modyfikacji. Autoryzacja odbywać się będzie na podstawie unikalnego loginu i hasła klienta. W celu zachowania poufności danych, połączenie między klientem a serwerem będzie szyfrowane. Serwer będzie wysyłał wiadomości w oparciu o protokół SMTP. Podstawowy schemat organizacji struktury programu przedstawiono na rysunku.

# 3.Interfejs użytkownika

Użytkownik będzie miał możliwość rozsyłania wiadomości e-mail do zdefiniowanych wcześniej grup odbiorców (na przykład prowadzący mają dostęp do list mailingowych poszczególnych przedmiotów). Główną funkcjonalnością programu będą zdefiniowane wcześniej szablony wiadomości, które umożliwią bardzo szybkie utworzenie standardowych wiadomości (na przykład informacja o spóźnieniu czy odwołaniu spotkania). Użytkownik wybiera w oknie aplikacji wcześniej zdefiniowaną grupę, do której chce wysłać wiadomość, następnie dodaje wymagane do uzupełnienia szczegóły wiadomości (na przykład godzinę na którą chce przełożyć spotkanie) i klika wyślij. W razie potrzeby istnieje również możliwość modyfikacji domyślnego szablonu wiadomości czy też modyfikacja grup odbiorców.

# 4. Szkic implementacji

Implementację projektu można wstępnie podzielić na 3 główne moduły – moduł kilenta, komunikujący się z serwerem, moduł komunikacji między klientem a serwerem oraz moduł rozsyłający wiadomości (moduł serwera).

## 4.1 Moduł Klienta

Aplikacja kliencka zorganizowana zostanie jako aplikacja okienkowa. Głównym założeniem jej interfejsu ma być prostota i szybka intuicyjna obsługa. Umożliwiać ma ona logowanie (uwierzytelnienie użytkownika), zarządzanie grupami odbiorców oraz wysyłanie wiadomości. Podstawową funkcjonalnością będzie wysyłanie wiadomości wcześniej zdefiniowanych jako szablony na serwerze. Wtedy konieczne będzie jedynie uzupełnienie niezbędnych danych (np. informacji o czasie spóźnienia czy godzinie na którą chcemy przełożyć spotkanie), co zdecydowanie przyspieszy proces komunikacji. Dla każdego klienta, po połączeniu się do serwera zostanie utworzony odrębny wątek kliencki, który zarządzał będzie dalszą komunikacją z serwerem.

## 4.2 Moduł komunikacji

Połączenie obsługiwać będzie wiele współpracujących ze sobą wątków. Każdy z nich wykonywał będzie ściśle określone zadanie, niezależnie od innych. Komunikacja między wątkami odbywać się będzie za pomocą potoków (pipe).

* Wątek nasłuchujący na połączenia – jest to wątek serwera, który nasłuchiwał będzie na połączenia od klientów. Po tym jak klient nawiąże z nim połączenie, wątek nasłuchujący tworzył będzie nowy wątek kliencki.
* Wątek kliencki – jest to pierwszy wątek jaki powstaje po podłączeniu się klienta do serwera. Odpowiada on za autoryzację klienta, komunikację z wątkiem dostępu do danych oraz przesyłanie wiadomości jakie chce wysłać klient do kolejki wiadomości wątku rozsyłającego serwera
* Wątek dostępu do danych – będzie on odpowiadał za komunikację z bazą danych przechowującą informacje o grupach i wiadomościach
* Wątek rozsyłający serwera – odpowiadał będzie za rozsyłanie wiadomości pobranych z kolejki wiadomości za pomocą protokołu SMTP
* Wątek zarządzający – będzie nasłuchiwał na sygnały, że konieczne jest już zakończenie pracy. Będzie odpowiedzialny za zakończenie wszystkich wątków



