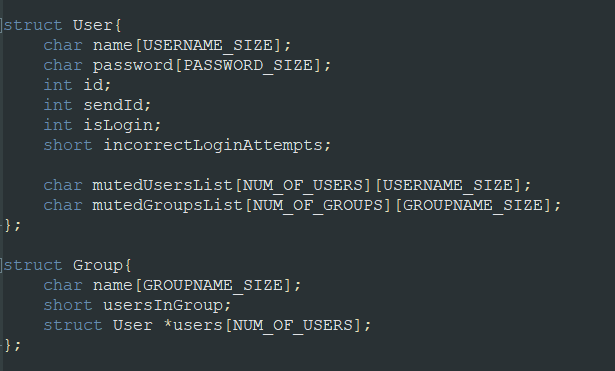
* We wstępie dodać coś w tym stylu:

Serwer przechowuje również PID procesu potomnego użytkownika, który jest odpowiedzialny za odbiór wiadomości od innych użytkowników. Wykorzystanie współbieżności jest uzasadnione tym, że program klienta musi cały czas oczekiwać na przychodzące wiadomości wysłane przez innych użytkowników, a w przypadku zapytań bezpośrednich do serwera robi to tylko w momencie wysłania zapytania.

* W 1.3 Komunikacja:

//z tego co pamiętam to mamy w sprawku napisać o wykorzystywanych strukturach

Serwer przechowuje dane o użytkownikach i grupach w następujących strukturach:



Pola wymagające wyjaśnienia:

* id – identyfikator użytkownika na serwerze (PID), wykorzystywany również do komunikacji serwera z użytkownikiem (adresat -> serwer -> adresat)
* sendId – PID procesu potomnego, wykorzystywany w komunikacji adresat -> serwer -> odbiorca/grupa
* UsersInGroup – aktualna liczba osób w grupie
* Wysyłanie wiadomości:

Do serwera zostaje wysłana informacja z ID użytkownika - adresata, typem zapytania oraz wiadomość w formacie: nazwa\_odbiorcy;treść. Serwer przetwarza otrzymaną wiadomość i gdy nie zostanie wykryty żaden błąd, przekazuje jej treść do odbiorcy - wiadomość ta jest odbierana przez proces potomny odbiorcy. W przypadku niepowodzenia adresat otrzymuje odpowiedni komunikat z treścią błędu, w przeciwnym przypadku otrzymuje potwierdzenie wysłania wiadomości.

Możliwe błędy:

* Próba wysyłania wiadomości do nieistniejącego użytkownika
* Użytkownik nie jest zalogowany
* Adresat jest zablokowany przez odbiorcę
* Próba wysłania wiadomości do samego siebie

Wymagane: long type, int senderID, char message[1024]

* Wysyłanie wiadomości do grupy:

Do serwera zostaje wysłana informacja z ID użytkownika - adresata, typem zapytania oraz wiadomość w formacie: nazwa\_grupy;treść. Serwer przetwarza otrzymaną wiadomość i gdy nie zostanie wykryty żaden błąd, przekazuje jej treść do zalogowanych użytkowników w danej grupie. Wiadomość ta jest odbierana przez procesy potomne odbiorców. W przypadku niepowodzenia adresat otrzymuje odpowiedni komunikat z treścią błędu, w przeciwnym przypadku otrzymuje potwierdzenie wysłania wiadomości.

Możliwe błędy:

* Próba wysyłania wiadomości do nieistniejącej grupy

Wymagane: long type, int senderID, char message[1024]

* Blokowanie użytkownika

Do serwera zostaje wysłana informacja z ID użytkownika - adresata, typem zapytania oraz wiadomość w formacie: nazwa\_użytkownika. Serwer przechowuje informację o zablokowaniu użytkownika przez adresata. W przypadku niepowodzenia zablokowania adresat otrzymuje komunikat o błędzie, w przeciwnym przypadku otrzymuje potwierdzenie zablokowania.

Możliwe błędy:

- Próba zablokowania nieistniejącego użytkownika

- Użytkownik jest już zablokowany

- Próba zablokowania samego siebie

Gdy zablokowany użytkownik podejmie próbę wysłania wiadomości - otrzyma odpowiedni komunikat o błędzie.

Wymagane: long type, int senderID, char message[1024]

* Odblokowanie użytkownika

Do serwera zostaje wysłana informacja z ID użytkownika - adresata, typem zapytania oraz wiadomość w formacie: nazwa\_użytkownika. Serwer usuwa informację o zablokowaniu użytkownika i przywraca możliwość komunikacji zablokowanemu użytkownikowi z adresatem. W przypadku niepowodzenia adresat otrzymuje komunikat o błędzie, w przeciwnym przypadku otrzymuje potwierdzenie odblokowania.

Możliwe błędy:

* Próba odblokowania nieistniejącego użytkownika
* Próba odblokowania użytkownika, który nie był zablokowany
* Próba odblokowania samego siebie

Wymagane: long type, int senderID, char message[1024]

* Blokowanie grupy

Do serwera zostaje wysłana informacja z ID użytkownika - adresata, typem zapytania oraz w wiadomość w formacie: nazwa\_grupy. Serwer przechowuje informacje o zablokowaniu grupy przez adresata. W przypadku niepowodzenia zablokowania adresat otrzymuje komunikat o błędzie, w przeciwnym przypadku otrzymuje potwierdzenie zablokowania.

Możliwe błędy:

- Próba zablokowania nieistniejącej grupy

- Grupa jest już zablokowana

W przypadku wysłania wiadomości do zablokowanej grupy, adresat zostanie zignorowany i wiadomość nie zostanie mu dostarczona.

Wymagane: long type, int senderID, char message[1024]

* Odblokowanie grupy

Do serwera zostaje wysłana informacja z ID użytkownika - adresata, typem zapytania oraz wiadomość w formacie: nazwa\_grupy. Serwer usuwa informację o zablokowaniu grupy przez adresata. W przypadku niepowodzenia adresat otrzymuje komunikat o błędzie, w przeciwnym przypadku otrzymuje potwierdzenie odblokowania.

Możliwe błędy:

- Próba odblokowania nieistniejącej grupy

- Próba odblokowania grupy, która nie była zablokowana

Wymagane: long type, int senderID, char message[1024]