Komunikator II

Cel zajęć: ulepszenie poprzedniej wersji kumunikatora poprzez zaimplementowanie możliwości rozmów grupowych.

Wstęp:
Na zajęciach potrzebny będzie kod z poprzednich lekcji (jest umieszony na stronie). Część serwerowa zostanie zmodyfikowana w taki sposób, aby umożliwiała połączenie się wielu klientów i propagowała wiadomości między nimi. Ta część zajęć będzie realizowana wspólnie, na tablicy, z pomocą prowadzącego. W kodzie klienta wprowadzone zostaną zmiany ułatwiające korzystanie z programu (tzn porawki UX-owe, UX – User Experience) oraz porawiające wygląd szaty graficznej. Komunikacja między klientem a serwerem zostanie ustandaryzowana - wiadomości będą przesyłane zgodnie z określonym wzorcem. Do serializacji wiadomości użyty zostanie moduł *pickle*.

Standard wiadomości:

Do tej pory przeysłaliśmy dane w formie zykłych stringów co miało spore ograniczenia. Od tej pory przesyłanie będą obiekty typu słownikowego (dictionary), które będa musiały posiadać w swojej strukturze wymagane klucze. Takie podejście umożliwi w przyszłości nie tylko przesyłanie wiadomości użytkowników, ale także „sterowanie" grupą np: dołączanie w określonym nickiem, tworzenie prywatnych czatów itp. Opis Standardu:

1. Wiadomoścu mają formę słownika (dictionary).
2. Pola obowiązkowe:
   a) ***COMMAND*** – określa typ wiadomości. Mozliwe opcje:
      - *MESSAGE* – informuje że wiadomość zawieraja wiadomość użytkownika do rozpropagowania
      - ... (inne takie jakie JOIN, NICK – na tych zajęciach niepotrzebne)
3. Pola opcjonalne
   a) ***MESSAGE*** – zawiera treść wiadomości użytkownika – jest to pole obowiązkowe dla wiadomości typ MESSAGE
   b) ***NICK*** – zawiera nick użytkownika, który wysłał wiadomość - jest to pole obowiązkowe dla wiadomości typ MESSAGE
   c) ... (inne, potrzebne w przypadku innych typów wiadomości)

Innymi słowy: Przesyłane wiadomosci muszą zawierać typ. Jesli typ jest określony jako MESSAGE, czyli jest to po prostu wiadomość użytkownika, to wymagane jest, aby zawierała ona także pola MESSAGE i NICK. Jeśli pól tych nie ma, lub są puste, możemy uznać że wiadomość jest niepoprawna – serwer nie będzie jej propagował. Na tych zajęciach będziemy przesyłać tylko wiadomości użytkowników, więc w praktyce każde przesyłane wiadomości będą mieć ustawiony typ na wartość MESSAGE.

TODO:

1. **Modyfikacja kodu serwera (wspólnie z prowadzącym):**

- cyklicznie: otwieranie kolejnych połączeń w ramach gniazda, akceptowanie klientów, nasłuchiwanie na wiadomości oraz propagowanie ich do pozostałych klientów

- wstępna walidacja wiadomości wg standardu

- modyfikacja GUI serwera – poprawki UX-owe i inne

2. **Modyfikacja kodu klienta:**

a) Modyfikacja funkcji *listen*:

- zaminieć dekodowanie wiadomości (*decode*()) na deserializację z wykorzystaniem modułu *pickle*
- dodać walidację wiadomości wg standardu – sprawdzić czy wiadomość ma klucz *COMMAND* ustawiony na *MESSAGE*
- zwrócić do GUI wiadomość wraz z nickiem rozmówcy.

b) Modyfikacja funkcji *send*:

- dodać do wiadomości wymagane standardem pola: *COMMAND*, *MESSAGE*, *NICK*

- usunać białe znaki na końcach wiadomości użytkownika (tak, aby nie wysyłac znaków końca linii i pustych spacji)

- zserializować wiadomość za pomocą modułu *pickle*

3. **Modyfikacja GUI klienta:**

a) podpięcie wysłania wiadomości pod klawisz ENTER:

```
self.parent.bind('<Return>', self.onEnter)
```

b) ustawienie widoku okienka wyświetlającego wiadomości tak, aby zawsze pokazywał ostatnie wiadomości:

```
self.text_output.see(END)
```

c) dodanie scrolla do okienka wyświetlającego wiadomości:

- dodanie elementu typu *Frame* który będzie kontenerem dla tego okienka i dla scrolla
- modyfikacja kodu tak aby wspomniane okienko znajdowało się w kontenerze (odpowiedni parent)
- stworzenie elementu typu ScrollBar

```
scroll = Scrollbar(frame, orient=VERTICAL,
command=self.text_output.yview)
```

- podpięcie scrolla do okienka:

```
self.text_output['yscroll'] = scroll.set
```

- umieszenie scrolla w kontenerze:

```
scroll.pack(side="right", fill="y")
```

-umieszczenie kontenera w oknie

```
frame.pack()
```