Sprawozdanie do projektu:

Sieci komputerowe 2

Temat: Komunikator internetowy typu GG

Data: 08.01.2017r.

Prowadzący: Michał Kalewski

Zajecia: wtorek 9:45

Wykonali:

Kasper Bojaruniec, 127221, I4

Mikołaj Musidłowski, 127251, I4

1. Temat zadania:

Komunikator internetowy typu GG

2. Opis protokołu komunikacyjnego:

Protokół komunikacyjny oparty na kluczach, które odpowiadają nazwie klas odpowiednich typów wiadomości. Każdy komunikat musi posiadać klucz i odpowiadające mu argumenty np.:
Ask_login – posiada argumenty login, password (user -> server)
Ask_login_acc – id, czyPoprawneDane, warning (server -> user)
Send_message – text, id_from, id_to (user -> server -> user)

3. Opis implementacji (w tym krótki opis zawartości plików źródłowych)

a) server:

ValidMessage.h – klasa nadrzędna dla poniższych, opisuje protokuł komunikacyjny, zawiera jego klucz

Add_user.h, Add_user_acc.h, Ask_chat.h, Ask_login.h, Ask_sign_in.h, Ask_sign_in_acc.h, Close_connection.h, Send_message.h – klasy dziedziczące z ValidMessage.h, każda posiada specyficzne dla swojego typu komunikatu pola odpowiadające argumentom danego klucza. Oprócz tego posiadają metody tworzące z objektu json, oraz konwertujeące do obiektu json.

Message.h – klasa zawierająca informacje o treści wiadomości oraz o jej nadawcy (id nadawcy)

Chat.h – klasa przechowująca wektor messege'ów

Chats.h – klasa przechowująca informacje o rozmowach między danymi użytkownikami (pole chats -> przechowuje Chat dla danej pary klientów).

Posiada metody zapisujące, odczytujące czaty z pliku.

Client.h – klasa nadrzędna dla poniższej, posiada pole name, id.

Client_me.h – rozwinięcie klasy Client zwierająca dodatkowo pola

contacts (wektor Client'ów) – zawiera informacje o kontaktach,

login, password, ifLogged.

json.hpp – biblioteka wykorzystywana do parsowania obiektów na stringi, i odwrotnie

header.h –plik nagłówkowy zawierający pozostałe pliki nagłówkowe

MAKEFILE – skrypt kompilacyjnY

server.cpp – plik implementujący logikę servera, opartego na komunikacji przez funkcję select

b) client:

Chat.java, Client.java, Client_me.java, Message.java, Add_user.java, Add_user_acc.java, Ask_chat.java, Ask_login.java, Ask_login_acc.java, Ask_sign_in.java, Ask_sign_in_acc.java, Close_connection.java, Send_message. Java, ValidMessage.java – odpowiedniki plików serwera napisane w języku java.

Communication_handler.java – klasa odpowiedzialna za kolejkowanie komunikatów wychodzących z clienta oraz obsługę komunikatów przychodzących.

Connection_handler.java – klasa odpowiedzialna za połączenie z serwerem oraz wysyłanie i odbieranie kominukatów.

CzatStage.java – klasa dziedzicząca ze Stage, zawiera elementy GUI okna z czatem

Protocol_converter.java – klasa zamieniająca obiekty JSON na ValidMessage

ReadChat.java/WriteChat.java – klasy odpowiednio wczytujące historię czatów i ją aktualizujące.

Refresher.java – klasa odpowiadająca za aktualiziowanie czatu

Variables.java – klasa trzymająca wartości zmiennych globalnych

ProjektSK2 – klasa główna clienta, zawiera wszystkie pozostałe elementy GUI oraz obsługę ruchów użytkownika

4. Sposób kompilacji, uruchomienia i obsługi programów projektu.

W folderze SK_2/server/ otwieramy terminal bash, wpisujemy ./MAKEFILE, klikamy ENTER. Uruchamiamy server przez wpisanie ./server oraz kliknięcie ENTER.

Uruchomić clienta w dowolnym IDE(polecamy NetBeans), w Connection_handler ustawiany IP na aktualne IP serwera, kompilujemy oraz uruchamiamy. Tworzymy konto, dodajemy znajomych(aby dodać znajomych trzeba założyć iwecej kont, kolejne konta dostają kolejne od 1 numery ID), poprzez podanie ich ID, czatujemy.