Міністерство освіти і науки України  
Тернопільський національний економічний університет  
Факультет комп’ютерних інформаційних технологій

кафедра EK

Комплексне практичне індивідуальне завдання

з дисципліни

«Організація комп'ютерних мереж»

на тему:

«Розробка програми обміну інформацією між станціями ЛОМ по технології клієнт - сервер.»

Виконав :

Студент групи ПЗС-32

Ваврух Р.В.

Перевірив:

Сирник О.Й.

Тернопіль 2015

Зміст.

Вступ………………………………………………………..…….…..3

1. Технологія клієнт – сервер……………………………..………..4
2. Огляд протоколів TCP/IP…………………………….………..…6
3. Git………………………………………………………………….8
4. Створення програми чату………………………………..………9

4.1. Сервер……………………………………………..………….9

4.2. Клієнт…………………………………………………….…..11

Висновки…………………………………………………….……….15

Список літератури………………………………………….…….….16

Вступ

Архітектура клієнт-сервер є одним із [архітектурних шаблонів програмного забезпечення](https://uk.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%BD%D1%96_%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F) та є домінуючою концепцією у створенні розподілених мережних [застосунків](https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA) і передбачає взаємодію та обмін даними між ними.

Сучасні тенденції розвитку інформаційних систем у великій мірі зумовлені необхідністю підвищення швидкості доступу кінцевого користувача - фахівця до потрібної інформації. Сьогодні інформаційні системи у фінансових установах засновані на концепції відкритих систем невід'ємною складовою частиною яких є технологія клієнт/ сервер. Слід зазначити, що свого часу централізовану обробку даних змінила обробка даних в режимі розподілу часу на центральному комп'ю­тері. Згодом процедури обробки та доступу до даних почали роз приділятися між робочими станціями, що зв'язані з центральним комп'ю­тером. Тому початкове поняття клієнт/сервер означало використання персональних комп'ютерів клієнтів, об'єднаних локальними чи гло­бальними мережами з центральним комп'ютером або сервером.

1. Технологія клієнт – сервер.

Технологія клієнт - сервер, яка широко застосовується при роботі з базами даних в мережі, відома вже давно і найчастіше застосовувалась у великих організаціях. Сьогодні, з розвитком INTERNET, ця технологія все частіше приваблює погляди розробників програмного забезпечення, оскільки в світі нагромаджено величезну кількість інформації по різноманітних питаннях і найчастіше ця інформація зберігається в базах даних. Технологію клієнт - сервер можна описати наступним алгоритмом:

· клієнт формує і посилає запит до бази даних серверу, вірніше - до програми, яка обробляє запити;

· ця програма проводить маніпуляції з базами даних, що знаходяться на сервері, у відповідності з запитом, формує результат і передає його клієнту;

· клієнт отримує результат, відображає його на дисплеї і чекає подальших дій користувача. Цикл повторюється до того часу, поки користувач не завершить роботу з сервером.

Стандартне програмне забезпечення, що реалізує технологію клієнт – сервер, має хорошу масштабованість (ефективне використання нарощеного апаратного забезпечення), стійкість в роботі, захист від несанкціонованого доступу і потужність при роботі з великими проектами в галузі баз даних.

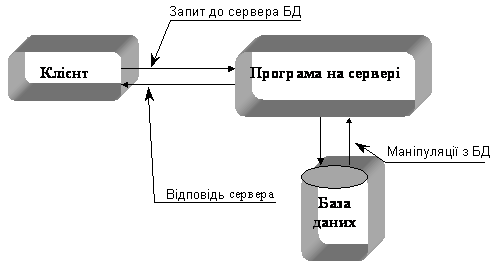


Рис 1.1. Основні складові технології клієнт-сервер

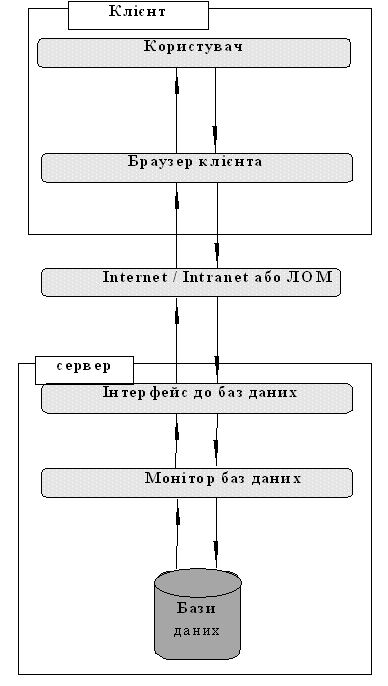


Рис 1.2. Принцип роботи технології клієнт-сервер

На рисунку 1.1 показано, як працює технологія клієнт - сервер в загальному випадку під’єднання до глобальної комп’ютерної мережі. Конкретно все залежить від того, де знаходиться клієнт та сервер, і як клієнт під’єднаний до серверу. Користувач на клієнтському комп’ютері в програмі перегляду заповнює запропоновану форму або вибирає подальшу дію. Браузер (програма пошуку) по натиску однієї з кнопок на формі пересилає дані із заповненої форми або відображає заново отримані в результаті деякої операції. Не важливо, до якої з мереж під’єднаний клієнт. Він навіть може бути віддаленим користувачем і з’єднуватися по модему. Програма приймає дані, перевіряє їх і формує запит до монітора баз даних або отримує від нього результат. Отримавши запит, монітор опрацьовує його і тоді, якщо не сталося помилок обробляє і відправляє потрібні дані програмі. На диску сервера зберігається база даних, що модифікується по запиту клієнта. При такому режимі роботи забезпечується високий рівень безпеки бази даних як від збоїв обладнання і програм, так і від несанкціонованого доступу, висока продуктивність, навантаження на мережу падають, але зростають вимоги до серверу.

1. Огляд протоколів TCP/IP

TCP/ІP - це установка протоколів, використовуваних для зв'язку комп'ютерних мереж і маршрутизації руху інформації між великою кількістю різних комп'ютерів. "TCP" означає "Протокол контролю передачі", а "ІP" означає "[Протокол](http://ua-referat.com/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB)між мережевої взаємодії". Протоколи стандартизовані описаними припустимими форматами, [обробкою](http://ua-referat.com/%D0%9E%D0%B1%D1%80%D0%BE%D0%B1%D0%BA%D0%B0)помилок, передачею повідомлень і [стандартами](http://ua-referat.com/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82)зв'язку. Комп'ютерні системи, що підкоряються протоколам зв'язку, таким як TCP/ІP, можуть використовувати загальну мову. Будь-яка машина глобальної [мережі](http://ua-referat.com/%D0%9C%D0%B5%D1%80%D0%B5%D0%B6%D1%96)може взаємодіяти з будь-який іншої (термін "глобальна мережа" звичайно використовується для назви об'єднання двох чи більше локальних мереж. У результаті виходить [мережа](http://ua-referat.com/%D0%9C%D0%B5%D1%80%D0%B5%D0%B6%D0%B0)з мереж "іnternet"). Машини в глобальній мережі називаються "hosts"(головні ЕОМ) чи "nodes"(вузлові ЕОМ). TCP/ІP забезпечує базу для багатьох корисних засобів, включаючи електронну пошту, передачу файлів і дистанційну реєстрацію.

ІP визначає незв'язану пакетну доставку. Ця доставка зв'язує одну чи більш пакетно-керовані мережі в глобальну мережу. Термін "незв'язану" означає, що одержує і посилає машини не зв'язані собою безпосереднім контуром. Тут індивідуальні пакети даних маршрутизуются через різні машини глобальної [мережі](http://ua-referat.com/%D0%9C%D0%B5%D1%80%D0%B5%D0%B6%D1%96)до локального мережі-одержувачу і машині, що одержує. У такий спосіб повідомлення розбиваються на трохи дейтаграм, що посилаються окремо. Помітьте, що незв'язана пакетна доставка сама по собі ненадійна. Окремі дейтаграми можуть бути отримані чи не отримані і з великою імовірністю можуть бути отримані не в тім порядку, у якому вони були послані. TCP збільшує надійність. Дейтаграма складається з заголовка, [інформації](http://ua-referat.com/%D0%86%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)й області даних. Заголовок використовується для маршрутизації і [процесу](http://ua-referat.com/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81)дейтаграми. Дейтаграма може бути розбита на малі частини в залежності від фізичних можливостей локальної мережі, по якій вона передається.

Коли шлюз посилає дейтаграмму до локальної мережі, що не може розмістити дейтаграму як єдиний пакет, вона повинна бути розбита на частини, що досить малі для передачі по цій мережі. Заголовки фрагментів дейтаграми містять інформацію, необхідну для збору фрагментів у закінчену дейтаграму. Фрагменти необов'язково прибувають один по одному, у якому вони були послані; програмний модуль, що виконує ІP [протокол](http://ua-referat.com/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB) на машині, що одержує, повинний збирати фрагменти у вихідну дейтаграму. Якщо які-небудь фрагменти загублені, повна дейтаграма скидається.

Протокол [контролю](http://ua-referat.com/%D0%9A%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8C)передачі даних (TCP) працює разом з ІP для забезпечення надійної доставки. Він пропонує засоби забезпечення надійності того, що різні дейтаграми, що складають повідомлення, збираються в правильному порядку на приймаючій машині і що деякі пропущені дейтаграми будуть послані знову, поки вони не будуть прийняті правильно. Перша мета TCP – це забезпечення надійності, безпеки і сервісу віртуального контуру зв'язку між парами зв'язаних [процесів](http://ua-referat.com/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81)на рівні ненадійних в середині мережевих пакетів, де можуть трапитися втрати, знищення, дублювання, чи затримка втрата упорядкованості пакетів. Таким чином, забезпечення безпеки, наприклад такий як обмеження доступу користувачів до [відповідного](http://ua-referat.com/%D0%92%D1%96%D0%B4%D0%BF%D0%BE%D0%B2%D1%96%D0%B4%D1%8C)машинам, може бути виконане за допомогою TCP. TCP стосується тільки загальної надійності. Мається кілька розумінь щодо можливості одержання надійного сервісу дейтаграм. Якщо дейтаграма послана через локальну мережу до віддаленої головної машини, то проміжні мережі не гарантують доставку. Крім того, що посилає машина не може знати маршрут передачі дейтаграми. Надійність шляху "джерело-приймач" забезпечується TCP на тлі ненадійності середовища.

Це робить TCP добре пристосованої до широкої розмаїтості додатків багатомашинних зв'язків. Надійність забезпечується за допомогою [контрольної](http://ua-referat.com/%D0%9A%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8C)суми (коди виявлення помилок) послідовних чисел у заголовку TCP, прямого підтвердження одержання даних і повторної передачі непідтверджених даних.

3.1. Git

Git — [розподілена система керування версіями](https://uk.wikipedia.org/wiki/%D0%A0%D0%BE%D0%B7%D0%BF%D0%BE%D0%B4%D1%96%D0%BB%D0%B5%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D1%96%D1%8F%D0%BC%D0%B8) [файлів](https://uk.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB) та спільної роботи. Проект створив [Лінус Торвальдс](https://uk.wikipedia.org/wiki/%D0%9B%D1%96%D0%BD%D1%83%D1%81_%D0%A2%D0%BE%D1%80%D0%B2%D0%B0%D0%BB%D1%8C%D0%B4%D1%81) для управління розробкою [ядра Linux](https://uk.wikipedia.org/wiki/%D0%AF%D0%B4%D1%80%D0%BE_Linux), а сьогодні підтримується Джуніо Хамано ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) Junio C. Hamano). Git є однією з найефективніших, надійних і високопродуктивних [систем керування версіями](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D1%96%D1%8F%D0%BC%D0%B8), що надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні і злитті гілок. Для забезпечення цілісності історії та стійкості до змін заднім числом використовуються криптографічні методи, також можлива прив'язка [цифрових підписів](https://uk.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B8%D0%B9_%D0%BF%D1%96%D0%B4%D0%BF%D0%B8%D1%81) розробників до тегів і комітів.

Прикладами проектів, що використовують Git, є [ядро Linux](https://uk.wikipedia.org/wiki/%D0%AF%D0%B4%D1%80%D0%BE_Linux), [Android](https://uk.wikipedia.org/wiki/Android), [LibreOffice](https://uk.wikipedia.org/wiki/LibreOffice), [Cairo](https://uk.wikipedia.org/wiki/Cairo), [GNU Core Utilities](https://uk.wikipedia.org/w/index.php?title=GNU_Core_Utilities&action=edit&redlink=1)[[2]](https://uk.wikipedia.org/wiki/Git#cite_note-2), Mesa 3D[[3]](https://uk.wikipedia.org/wiki/Git#cite_note-3), [Wine](https://uk.wikipedia.org/wiki/Wine), багато проектів з X.org[[4]](https://uk.wikipedia.org/wiki/Git#cite_note-4), XMMS2[[5]](https://uk.wikipedia.org/wiki/Git#cite_note-5), [GStreamer](https://uk.wikipedia.org/wiki/GStreamer), Debian DragonFly BSD, [Perl](https://uk.wikipedia.org/wiki/Perl), [Eclipse](https://uk.wikipedia.org/wiki/Eclipse), [GNOME](https://uk.wikipedia.org/wiki/GNOME), [KDE](https://uk.wikipedia.org/wiki/KDE), [Qt](https://uk.wikipedia.org/wiki/Qt), [Ruby on Rails](https://uk.wikipedia.org/wiki/Ruby_on_Rails),[PostgreSQL](https://uk.wikipedia.org/wiki/PostgreSQL), [VideoLAN](https://uk.wikipedia.org/w/index.php?title=VideoLAN&action=edit&redlink=1), [PHP](https://uk.wikipedia.org/wiki/PHP), Beryl[[6]](https://uk.wikipedia.org/wiki/Git#cite_note-6), One Laptop Per Child ([OLPC](https://uk.wikipedia.org/wiki/OLPC)), [АБІС](https://uk.wikipedia.org/wiki/%D0%90%D0%91%D0%86%D0%A1) [Koha](https://uk.wikipedia.org/wiki/Koha), GNU LilyPond[[7]](https://uk.wikipedia.org/wiki/Git#cite_note-7) та ELinks[[8]](https://uk.wikipedia.org/wiki/Git#cite_note-8) і деякі дистрибутиви[GNU](https://uk.wikipedia.org/wiki/GNU)/Linux (див. нижче).

Програма є вільною і випущена під ліцензією [GNU GPL](https://uk.wikipedia.org/wiki/GNU_General_Public_License) версії 2.

Система спроектована як набір програм, спеціально розроблених з врахуванням їхнього використання у [скриптах](https://uk.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B2%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0). Це дозволяє зручно створювати спеціалізовані системи управління версіями на базі Git або користувацькі інтерфейси. Наприклад, Cogito[[9]](https://uk.wikipedia.org/wiki/Git#cite_note-9) є саме таким прикладом фронтенда до репозиторіїв Git. А StGit[[10]](https://uk.wikipedia.org/wiki/Git#cite_note-10) використовує Git для управління колекцією [латок](https://uk.wikipedia.org/w/index.php?title=%D0%9B%D0%B0%D1%82%D0%BA%D0%B0&action=edit&redlink=1).

Система має ряд користувацьких інтерфейсів: наприклад, gitk та git-gui розповсюджуються з самим Git.

Віддалений доступ до [репозиторіїв](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D1%96%D0%B9) Git забезпечується git-[демоном](https://uk.wikipedia.org/wiki/%D0%94%D0%B5%D0%BC%D0%BE%D0%BD_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0)), [SSH](https://uk.wikipedia.org/wiki/SSH) або [HTTP](https://uk.wikipedia.org/wiki/HTTP) [сервером](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80). [TCP](https://uk.wikipedia.org/wiki/TCP)-сервіс git-daemon входить у дистрибутив Git і є разом з SSH найпоширенішим і надійним методом доступу. Метод доступу HTTP, незважаючи на ряд обмежень, дуже популярний в контрольованих мережах, тому що дозволяє використання існуючих конфігурацій мережевих фільтрів.

4. Створення чату

Для створення чату використовувалось середовище програмування Embarcadero Delphi XE8.

Embarcadero Delphi XE - це [інтегроване середовище](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D1%81%D0%B5%D1%80%D0%B5%D0%B4%D0%BE%D0%B2%D0%B8%D1%89%D0%B5_%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B8) [швидкої розробки програмного забезпечення](https://uk.wikipedia.org/wiki/%D0%A8%D0%B2%D0%B8%D0%B4%D0%BA%D0%B0_%D1%80%D0%BE%D0%B7%D1%80%D0%BE%D0%B1%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F). Середовище Embarcadero Delphi працює під [Microsoft Windows](https://uk.wikipedia.org/wiki/Microsoft_Windows) і підтримує розробку програмного забезпечення для [операційних систем](https://uk.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) [Microsoft Windows](https://uk.wikipedia.org/wiki/Microsoft_Windows) [x86](https://uk.wikipedia.org/wiki/X86) та [x64](https://uk.wikipedia.org/wiki/X64), [Mac OS](https://uk.wikipedia.org/wiki/Mac_OS) [x86](https://uk.wikipedia.org/wiki/X86), [Apple iOS](https://uk.wikipedia.org/wiki/Apple_iOS) та [Android](https://uk.wikipedia.org/wiki/Android) на [мові програмування Delphi](https://uk.wikipedia.org/wiki/Delphi_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) ([Object Pascal](https://uk.wikipedia.org/wiki/Object_Pascal)). Embarcadero Delphi входить до групи засобів розробки [Embarcadero RAD Studio](https://uk.wikipedia.org/wiki/Embarcadero_RAD_Studio) і, в залежності від набору функціоналу, постачається в редакціях Starter, Professional, Enterprise та Architect.

4.1.Сервер

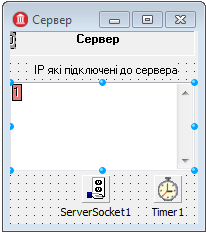


Рис. 4.1. Вікно серверу

Для розробки сервера нам потрібно для початку спроектувати його вигляд, у моєму випадку він буде написаний на Windows Forms. На сервері будуть відображатись тільки IP підключенні до нього. На рисунку 4.1. зображено як виглядає вікно розробленого серверу. Далі приступимо до самого коду лістинг якого представлено нижче:

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Uphorum.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

TStringList \*Users = new TStringList;

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::ServerSocket1ClientRead(TObject \*Sender,

TCustomWinSocket \*Socket)

{

AnsiString S = Socket->ReceiveText();

//Прийом запиту від клієнта

if (S[1] == '#')

//Якщо в запиті першим символом є решітка тоді ми знаємо що це ім’я

{

Users->Add(S.SubString(2,S.Length()));

//Додаємо імя в глобальний масив

for(int i=0; i<ServerSocket1->Socket->ActiveConnections; i++)

ServerSocket1->Socket->Connections[i]->SendText("#"+Users->Text);

//Відправляємо запит з іменами на всі клієнти

StaticText1->Caption =

IntToStr(ServerSocket1->Socket->ActiveConnections)+" клієнтів в мережі";

//Знаходимо кількість клієнтів

return;

}

AnsiString SName=S.SubString(0, S.Pos("#") - 1);

//Вирізаємо з нашого імені решітку і кидаємо у нову переміну

if (AnsiCompareText(SName, "Всем") == 0)

//Розсилка всім або конкретному іменні

for (int i= 0;i<ServerSocket1->Socket->ActiveConnections;i++)

{

ServerSocket1->Socket->Connections[i]->SendText(

S.SubString(SName.Length()+2, S.Length()));

//Відправляємо всім

}

else

{

ServerSocket1->Socket->Connections[Users->IndexOf(SName)-1]->

SendText(S.SubString(SName.Length() + 2, S.Length()));

//Відправляємо вибраному іменні

}}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::ServerSocket1ClientDisconnect(TObject \*Sender,

TCustomWinSocket \*Socket)

{

int i;

for (i= 0;i<ServerSocket1->Socket->ActiveConnections;i++)

if (ServerSocket1->Socket->Connections[i] == Socket)

// Визначення індексу роз’єднаного з’єднання.

break;

Users->Delete(i+1);

// Видалення із списку роз’єднаного з’єднання.

for (i= 0; i < ServerSocket1->Socket->ActiveConnections ;i++)

if(ServerSocket1->Socket->Connections[i] != Socket)

ServerSocket1->Socket->Connections[i]->SendText("#" + Users->Text);

// Розсилка клієнтам виправленого списку імен

StaticText1->Caption =

IntToStr(ServerSocket1->Socket->ActiveConnections -1)

+ " клієнтів в мережі";

// Виправлення кількості підключених клієнтів.

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::ServerSocket1ClientError(TObject \*Sender,

TCustomWinSocket \*Socket, TErrorEvent ErrorEvent, int &ErrorCode)

{

//Ну тут все зрозуміло

switch(ErrorEvent)

{

case eeSend: ShowMessage("Ошибка записи сообщения");

break;

case eeReceive: ShowMessage("Ошибка чтения сообщения");

break;

case eeConnect: ShowMessage("Ошибка открытия соединения");

break;

case eeDisconnect: ShowMessage("Ошибка завершения соединения");

break;

case eeAccept: ShowMessage("Ошибка доступа");

break;

case eeGeneral: ShowMessage("Ошибка соединения");

}

ErrorCode = 0;}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::FormCreate(TObject \*Sender)

{

ServerSocket1->Active=true;

//Включення серверу

Users->Add("Всім");

//Добавлення імя (Всім) в наш масив імен

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::FormDestroy(TObject \*Sender)

{

Users->Free();

//Звільнюємо наш масив з іменами після закриття програми.}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Timer1Timer(TObject \*Sender)

{

String IP\_ALL; int i;

IP\_ALL="";

Memo1->Text="";

//Об’являємо перемінні для того щоб записувати туди адреса

for (i = 0; i <= ServerSocket1->Socket->ActiveConnections -1; i++) {

IP\_ALL=ServerSocket1->Socket->Connections[i]->RemoteAddress;

//Перебираемо усі активні підключення до серверу

Memo1->Lines->Add(IP\_ALL);

//Додаємо усі знайдені підключення у поле Memo.

}}

//---------------------------------------------------------------------------

4.2. Клієнт

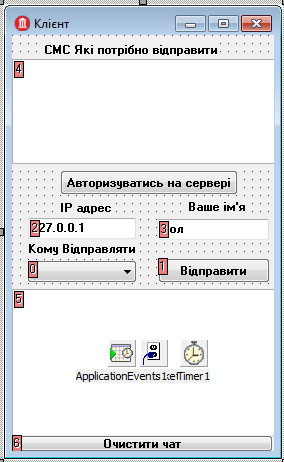


Рис. 4.2. Вікно клієнта

Для розробки клієнта нам потрібно також розробити його вигляд, так як сервер він буде написаний на Windows Forms. У клієнті будуть відображатись тільки імена других учасників підключених до серверу, IP адреса сервера, та повідомлення так як відправленні і отримані від інших користувачів . На рисунку 4.2. зображено як виглядає вікно розробленого клієнту. Далі приступимо до самого коду лістинг якого представлено нижче:

//---------------------------------------------------------------------------

#include <vcl.h>

#include <winuser.h>

#pragma hdrstop

#include "UClient.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button2Click(TObject \*Sender)

{

if (Button2->Down)

//якщо кнопка опущена вниз то виконуємо команди далі (В деяких версіях с++ може не робити)

{

if ((Edit1->Text == "") || (Edit2->Text == ""))

//Провірка чи заповненні поля айпі та ім’я

{

ShowMessage("Не вказано IP чи Ім’я");

Button2->Down = false;

//Піднімаємо кнопку якщо не заповненні поля і видаємо повідомлення про це

return;

}

if ((int(Edit1->Text[1])>=48)&&(int(Edit1->Text[1])<=57))

//Перевіряємо чи це дійсно айпішка

ClientSocket1->Address = Edit1->Text;

//Якщо це айпі то прописуємо його клієнту

else ClientSocket1->Host = Edit1->Text;

//Іначе якщо кількість символів менша чи більша тоді це рахується як Хоче

try

//Пробуємо підключитись до серверу

{

if(!ClientSocket1->Active)

ClientSocket1->Open();

}

catch(...)

//Якщо це виходить тоді видаємо повідомлення про це.

{

ShowMessage("Неможливо встановити з’єднання з сервером");

}

} else ClientSocket1->Close();

//І якщо все взагалі безнадійно тоді закриваємо сокет

}

//--------------------------------------------------------

void \_\_fastcall TForm1::ClientSocket1Connect(TObject \*Sender,

TCustomWinSocket \*Socket)

{

ClientSocket1->Socket->SendText("#"+Edit2->Text);

//Відправляємо серверу наше ім’я

}

//---------------------------------------------------------

void \_\_fastcall TForm1::ClientSocket1Error(TObject \*Sender, TCustomWinSocket \*Socket, TErrorEvent ErrorEvent,int &ErrorCode)

{

//Тут знову все зрозуміло тільки може не робити команда Button2->Down (Повязанно з версіями)

switch(ErrorEvent)

{

case eeSend: ShowMessage("Помилка запису повідомлення"); break;

case eeReceive: ShowMessage("Помилка читання повідомлення"); break;

case eeConnect: ShowMessage("Помилка відкриття зєднання(Сервер не запущений)");

Button2->Down = false; break;

case eeDisconnect: ShowMessage("Помилка закриття з’єднання"); break;

case eeAccept: ShowMessage("Помилка доступа");

Button2->Down = false; break;

case eeGeneral: ShowMessage("Помилка з’єднання");

Button2->Down = false;

}

ErrorCode = 0;

}

//---------------------------------------------------------

void \_\_fastcall TForm1::ClientSocket1Read(TObject \*Sender,

TCustomWinSocket \*Socket)

{

if (!Application->Active)

Timer1->Enabled=true;

//Перевірка для того якщо щось прийде щоб картинка в треї мигала

AnsiString S = Socket->ReceiveText();

//Прийом команди від серверу

if (S[1] == '#')

//Якщо Перша буква в запиті Решітка то це ім’я

{

ComboBox1->Items->Text = S.SubString(2,S.Length()-1 );

ComboBox1->ItemIndex = ComboBox1->Items->Count - 1;

//Добавляємо впіймане ім’я до комбо боксу

return;

}

AnsiString SName = S.SubString(0, S.Pos("#") - 1);

//Вирізаємо з іменні решітку

RichEdit2->Lines->Add(SName + ": " +

S.SubString(SName.Length()+2, S.Length()));

//І вставляємо імя в наш чат разом з повідомленням

}

//---------------------------------------------------------

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{

TDateTime CurrentTime;

CurrentTime = Now();

//Описуєммо переміні для того щоб зібрати данні про теперішній час

if (RichEdit1->Text=="") {

ShowMessage("Веддіть повідомлення");

//Перевірка чи введено повідомлення

} else {

if (!ClientSocket1->Active)

{

//Ще одна перевірка чи є у нас якісь користувачі і чи ми взагалі активні

ShowMessage("Немає активних користувачів");

return;

}

if ((int(Edit1->Text[1])>=48)&&(int(Edit1->Text[1])<=57))

//знову ж перевіряємо чи це дійсьно айпішка

ClientSocket1->Address = Edit1->Text;

else ClientSocket1->Host = Edit1->Text;

ClientSocket1->Socket->SendText(ComboBox1->Text +"#" +

Edit2->Text + "#" +RichEdit1->Text +" (" +CurrentTime +")");

//І відправляєммо на сервер запит з Ім’ям на яке ми хочемо відправити повідомлення, сам текст повідомлення і його дату.

}

RichEdit1->Text="";

//Після відправки очищаємо поле введення повідомлення

}

//---------------------------------------------------------

void \_\_fastcall TForm1::Timer1Timer(TObject \*Sender)

{

FlashWindow(Application->Handle,true);

//Для того щоб коли зверуто програму в трей вона мигала якщо в ній сталася якась дія

}

//---------------------------------------------------------

void \_\_fastcall TForm1::ApplicationEvents1Activate(TObject \*Sender)

{

Timer1->Enabled = false;

//Для того щоб коли звернуто програму в трей вона мигала якщо в ній сталася якась дія

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button3Click(TObject \*Sender)

{

RichEdit2->Text="";

//Очищаємо наш чат

}

//---------------------------------------------------------------------------

Висновок

### Вивчаючи докладніше особливості одного з найпоширеніших протоколів мережі Інтернет - [TCP](https://uk.wikipedia.org/wiki/TCP/IP)-протоколу, походу написання цього комплексного практичного індивідуального завдання, я ознайомився з його особливостями, його організацією роботи і т.д..

У даній комплексному практичному завданні я використовував систему контролю версій Git. Регулярно пушив проект на “<https://github.com/>” .

На мою думку, цей протокол як один із засобів передачі даних через Інтернет, ще довго залишатиметься актуальним для користувача, оскільки зарекомендував себе як такий, що виконує передачу інформації економно у часі, пропонує хороші можливості передачі потрібних даних, витримав випробування часом як один з найстаріших протоколів, зараз у використанні. Тому кожна людина, що користується глибинами інформаційних ресурсів мережі Інтернет, повинна знати про існування [TCP](https://uk.wikipedia.org/wiki/TCP/IP)-протоколу і активно використовувати його як швидкий і надійний спосіб передачі даних.

Використана література

1. <https://uk.wikipedia.org/wiki/Поштовий_сервер>
2. <https://ru.wikipedia.org/wiki/HMailServer>
3. <https://www.hmailserver.com/docs>
4. <https://ru.wikipedia.org/wiki/SMTP>
5. <https://ru.wikipedia.org/wiki/POP3>
6. <https://uk.wikipedia.org/wiki/IMAP>
7. <http://wiki.taltek.info//Поштовий_сервер>