Laborator 10– Utilizarea Socket-urilor în C++ Builder

Tema 10.1

Realizați o aplicație în C++ Builder care să conțină următoarele:

- -O forma care va avea numele „fServer” și care va fi salvată într-un fișier cu numele „AppServer”;
- -O componentă de tip „Button” care să aibă numele „btnExit” și pe care să scrie textul „Exit” iar la apăsarea acestuia să se închidă aplicația;
- -O componentă de tip Memo cu numele „mText”;

Proiectul se va salva în folderul propriu cu numele „Server.bpr.

Considerații teoretice 10.1

Pentru realizarea acestei teme se recomandă citirea laboratorului 1.

Componenta Memo este o componentă care permite afișarea pe ecran au unui text pe mai multe linii (nu doar pe o singură linie ca și componenta Edit) precum și derularea acestuia. În timpul rulării aplicației utilizatorul poate modifica textul scris în componenta Memo. Pentru ca utilizatorul să nu mai poată accesa textul din componenta Memo se poate modifica proprietatea „Enable” (din fereastra Object inspector) în false a acestei componente. Astfel utilizatorul doar va vedea textul, nu va putea să îl și modifice.

Dacă textul afișat depășește dimensiunile componentei se parcurge textul folosind ScrollBar-ul componentei. Pentru a putea utiliza scrollbar-ul acesta trebuie să fie activat în prealabil. Pentru a activa ScrollBar-ul se poate modifica proprietatea ScrollBars în una dintre opțiunile: „ssHorizontal”, „ssVertical” sau „ssBoth”, iar ca efect va afișa scrollbar-ul orizontal, vertical sau ambele.

Pentru adăugarea din cod a unei linii de text într-o componentă de tip Memo se poate folosi metoda Add() a acestei componente din proprietatea „Lines” astfel:

```
mText->Lines->Add(„Sir de caractere care se adaugă in componenta Memo”);
```

Tema 10.2

Să se modifice aplicația anterioară astfel încât să afișeze pe ecran, în componenta Memo, toate mesajele recepționate de pe placa de rețea. Aceasta se va realiza folosind în aplicație o componentă de tip „ServerSocket”. Variabila din aplicație care accesează componenta ServerSocket se va numi „ssServer”.

Considerații teoretice 10.2

În C++ Builder sunt 2 categorii de componente:

- Componente vizuale – sunt acele componente care rămân pe formă și în timpul rulării aplicației. Toate componentele folosite până acum au fost componente vizuale;
- Componente nevizuale – sunt acele componente care ajută la configurarea aplicației în timpul dezvoltării acesteia și dispar de pe formă în momentul rulării acesteia;

Componenta de tip „ServerSocket” se găsește în tabul de componente „Internet” din C++ Builder. Aceasta este o componentă nevizuală care ajută la configurarea programului pentru a avea acces la un Socket de pe placa de rețea. Pentru configurarea acestei componente în *ObjectInspector”* trebuie specificate următoarele caracteristici (atenție la ordinea în care se specifică):

- *Port* – reprezintă adresa de port de pe placa de rețea pe care o va folosi aplicația. Fiecare aplicație care folosește placa de rețea trebuie să aibă un identificator unic (numit port) iar în momentul în care se recepționează ceva de pe rețea se va ști exact pentru care aplicație a venit acel mesaj. Adresa de port este un număr întreg și trebuie să fie o valoare care nu este folosită de celelalte aplicații din windows. Se recomandă folosirea unor valori mai mari de 1000. Pentru aplicația noastră recomand valoarea 2000.
- *Active* – se comută pe *true*. În acest moment se verifică dacă portul specificat este liber și se poate activa serverul. Dacă portul nu este liber această proprietate nu poate fi trecută pe *true*. Nu pot exista pe același calculator 2 servere care ascultă pe același port.

După ce această proprietate este trecută pe *true* componenta *ServerSocket* va monitoriza placa de rețea pe portul specificat și va aștepta să primească sau să trimită mesaje (șiruri de caractere).

Pentru aplicația server a noastră vom activa doar partea de recepție de mesaje, acest lucru se face prin activarea evenimentului „OnClientRead" de la această componentă. Metoda creată pentru acest eveniment se va apela automat în momentul recepției unui șir de caractere de pe placa de rețea pe portul specificat. Pentru citirea acestui șir de caractere se poate apela metoda ReceiveText() astfel:

```
Socket->ReceiveText();
```

unde *Socket* este parametrul primit de metoda generată pentru evenimentului *OnClientRead*() în care se găsește textul recepționat. Pentru fiecare text recepționat sistemul de operare creează câte o instanță de tip *Socket* din componenta *ServerSocket* care se va distruge după tratarea evenimentului. Instanța respectivă sistemul de operare o pune în coada de mesaje a aplicației urmând ca metoda creată pentru evenimentul *OnClientRead* să o preia în parametrul *Socket*. Textul recepționat există doar în această instanță și nu există în componenta generală inclusă în proiect (*ssServer->Socket* în cazul nostru).

În prima fază server-ul va aștepta clienți care să se conecteze la el. Acesta va crea câte o conexiune separată pentru fiecare client nou conectat la el. Astfel el poate determina și identificarea exact clientul care i-a trimis mesajul. Pentru a trimite un text la un client din listă (de exemplu clientul 3), se poate scrie următoarea instrucțiune:

```
ServerSocket->Socket->Connections[3]->SendText("Un Text Oarecare");
```

Indicații 10.2

- ⇨ La rularea aplicației dacă sistemul de operare vă atenționează că o aplicație dorește să monitorizeze un port de pe placa de rețea să permiteți acest lucru.
- ⇨ În acest pas aplicația nu afișează nimic deoarece nu există deocamdată nici un client conectat la ea.
- ⇨ Pentru Embarcadeor componenta de tip Socket nu sunt încărcată implicit, de aceea prima dată trebuie încărcate aceste componente pentru a le putea folosi ulterior în aplicații. Pentru încărcarea aceste componente se execută următorii pași:
    - Din meniu Embarcadero alegeți: "Select Component > Install Packages".
    - Apoi din fereastra de dialog "Install Packages", click Add.
    - In fereastra de dialog care apare mergeți în folderul BIN din Embarcadero (calea de exemplu C:\Program Files (x86)\Embarcadero\Studio\20.0\bin).
    - Alegeți **dclsockets260.bpl** (sau orice altă versiune aveti 270,,,290) și apoi click pe Open.
    - Alegeți OK pentru a instala pachetul cu componentele de tip Socket.
    - Componentele de tip socket se vor vedea in lista Internet din Tool Palette

Tema 10.3

Realizați o nouă aplicație în C++ Builder care să conțină următoarele:

- -O forma care va avea numele „*fClient*” și care va fi salvată într-un fișier cu numele „AppClient”;
- -O componentă de tip „Button” care să aibă numele „*btnExit*” și pe care să scrie textul „Exit” iar la apăsarea acestuia să se închidă aplicația;
- -O componentă de tip Edit cu numele „*mEdit*” și în care să scrie textul „Semigrupa”;
- -O componentă de tip Button cu numele „*btnSend*”și pe care să scrie textul ”Send”;

Proiectul se va salva în folderul propriu cu numele „Client.bpr”.

Considerații teoretice 10.3

Pentru realizarea acestei teme se recomandă citirea laboratorului 1.

Tema 10.4

Modificați aplicația de la tema 10.3 astfel încât aceasta să permită transmiterea de mesaje de tip text pe placa de rețea către un server. Utilizatorul va avea posibilitatea să introducă textul dorit în componenta „Edit” iar la apăsarea butonului „Send” acesta să fie transmis către server.

Considerații teoretice 10.4

Pentru crearea unei aplicații de tip client (aplicație care se conectează pe placa de rețea la un server) se poate folosi componenta „ClientSocket” care se găsește în tabul cu componente „Internet”. Această componentă este tot o componentă nevizuală care permite configurarea aplicației să permită conectarea la un server de pe rețea, printr-un port, și să permită transmiterea și recepționarea de mesaje de tip text prin aceasta.

Pentru a configura această componentă trebuie specificate următoarele caracteristici (atenție la ordinea de specificare):

- *Address* – trebuie specificată adresa IP a serverului. Fiecare calculator legat la internet are o adresă unică numită IP. Pentru a putea să ne conectăm la un server trebuie să cunoaștem IP-ul acestuia. Pentru lucrul pe același calculator (rulăm și aplicația server și aplicația client pe același PC) se poate specifica adresa 127.0.0.1 sau adresa IP a calculatorului respectiv;
- *Port* – se specifică portul pe care se comunică cu serverul. Trebuie specificat exact același port ca și cel de la aplicația server;
- *Active* – se comută pe true;

Pentru transmiterea unui șir de caractere pe placa de rețea se poate folosi metoda *SendText*() din componenta *Socket* a componetei *ClientSocket* astfel:

```
csClient->Socket->SendText(„Text de trimis”);
```

Indicații 10.4

- ⇨ Atenție că această aplicație se va bloca dacă aplicația server nu este pornită în prealabil. Aplicația client va încerca conectarea la server imediat la pornire deoarece clientul este activat din cod, iar dacă nu reușește va da eroare și se va bloca;

Tema 10.5

Modificația aplicația client astfel încât aceasta să nu se mai conecteze la server atunci când pornește ci să permită modificarea/specificarea în timpul rulării a adresei serverului la care dorim să ne

conectăm. Pentru aceasta trebuie să adăugați în aplicație un buton pe care se va afișa textul „Connect" și are numele „*btnConnect"*. La apăsarea acestuia se va crea o nouă formă în care să putem specifica adresa IP a serverului dorit. Din forma nouă putem ieși folosind butonul „*OK"* sau butonul „*Cancel"*. În cazul folosirii butonului OK se va lua în considerare noua adresă IP specificată și se va realiza conectarea la noul server. În cazul părăsirii ferestrei folosind butonul „*Cancel"* se va păstra vechia adresă IP care a fost folosită până în acel moment.

Considerații teoretice 10.5

Pentru ferestre de dialog simple care să permită utilizatorului introducerea unui șir de caractere mediul de dezvoltare C++ Builder pune la dispoziție o funcție numită *InputQuery* astfel încât se evită crearea unei noi forme în aplicație. Sintaxa funcției este următoarea:

```
bool InputQuery(constAnsiString ACaption, const AnsiString APrompt, AnsiString
&Value);
```

- *ACaption* – textul care apare pe fereastra de dialog în partea stângă sus;
- *APrompt* - textul care apare în dreptul căsuței de editare pentru a indica utilizatorului ce trebuie specificat în căsuța de editare;
- *Value* – este parametrul returnat de funcție și conține textul care a fost scris de utilizator în căsuța de editare;
- Funcția *InputQuery* întoarce *true* dacă utilizatorul a ales să apese pe butonul OK și *false* altfel.

Un exemplu de utilizare a acestei funcții ar fi:

```
String text;
if(InputQuery("Fereastra de Exemplu", "Scrie un text", text)){
  ShowMessage("Ati introdus textul: " + text + " si ati apasat OK");
}
else{
  ShowMessage("Ati apasat butonul Cancel");
}
```

ShowMessage() este o altă funcție disponibilă în mediul C++ Builder care permite ușor afișarea de mesaje pe ecran pentru utilizator. Mesajele vor fi afișate într-o fereastră de atenționare cu un singur buton „*OK*". Funcția primește un singur parametru, șirul de caractere care se dorește a fi afișat. Pentru concatenarea mai multor șiruri de caractere se poate folosi operatorul „+".

Indicații 10.5

- ⇨ Atenție în componenta *ClientSocket* înainte de a putea modifica câmpul *Address* proprietatea *Active* trebuie să fie trecută obligatoriu pe *false*, iar după modificarea adresei pentru a funcționa aplicația aceasta trebuie trecută înapoi pe *true*;
- ⇨ La folosirea funcției *InputQuery* trebuie verificat și cazul în care utilizatorul nu introduce nimic în câmpul de editare și iese folosind butonul Ok. În acest caz apelul metodei „*Length*()" pentru variabila de tipul clasei String va întoarce valoarea 0.

Tema 10.6

Modificați aplicația server (de la tema 10.2) astfel încât aceasta să afișeze pe ecran (în componenta Memo) pe lângă mesajul recepționat și adresa IP a clientului care a trimis mesajul.

Considerații teoretice 10.6

În realitate componenta *ServerSocket* creată în aplicația noastră va configura sistemul de operare să monitorizeze portul specificat și de câte ori se recepționează un mesaj pe acel port să îl trimită la aplicația noastră. În sistemul de operare pentru fiecare text recepționat de pe placa de rețea acesta creează câte o instanță de tip *Socket* din componenta *ServerSocket* pe care o pune în coada de mesaje a aplicației. Aceasta se face deoarece sistemul de operare poate recepționa mesaje pe placa de rețea mult mai reped decât le tratează aplicația noastră. Pentru fiecare mesaj din coada de mesaje se va activa evenimentul specificat din aplicația noastră. În evenimentul respectiv există mai multe informații despre cel care a trimis mesajul nu numai mesajul efectiv. De exemplu *Socket->RemoteAddress* conține adresa IP a celui care a trimis mesajul. După tratarea evenimentului obiectul de tip *Socket* se va distruge automat.

În momentul în care o aplicație client se conectează la un server acesta îi creează o conexiune separată prin intermediul căreia el va comunica doar cu acel client. Astfel la server se vor realiza atâtea conexiuni câți clienți sunt conectați la un moment dat la el. De exemplu pentru a trimite un mesaj la toți clienții conectați la un moment dat la server acesta va trebui să execute următorul cod:

```
for(int i=0;i<ssServer->Socket->ActiveConnections;i++)
   ssServer->Socket->Connections[i]->SendText("Mesaj");
```