

Klienckie aplikacje internetowe w C++ Builder

Zadaniem jest stworzenie prostej aplikacji umożliwiającej odbiór poczty przy użyciu protokołu POP3 oraz wysyłanie poczty przy użyciu protokołu SMTP, wykorzystując komponenty Indy.

1. Projektowanie głównego formularza z zakładkami

1. Nowo otwarty projekt zapisać pod wybraną nazwą, natomiast jego okno główne nazwać `F_Glowne` i zapisać pod nazwą `Glowne`.
2. Wybrać odpowiedni styl formularza, tak aby nie można było zmieniać jego wymiarów.
3. Na nowym projekcie, tzn. na jego głównym formularzu należy umieścić komponent odpowiadający *zakładce* (zakładka **Win32**, komponent **PageControl**).
4. Klikając prawym klawiszem myszy na komponencie **PageControl** należy wybrać z menu kontekstowego polecenie *New Page* służący dodawaniu nowej zakładki. Powtórz czynność dodając drugą zakładkę (*Rysunek 1*).



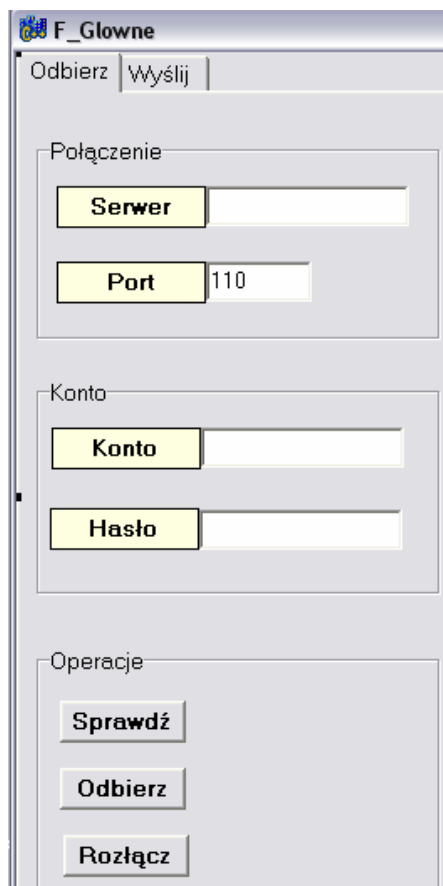
Rysunek1: Projektowanie zakładek.

5. Gdy są już dwie zakładki na kontrolce, zmień właściwości *Captions* obiektu `TabSheet1` na *Odbierz* oraz `TabSheet2` na *Wyślij*. Ustaw właściwość *Align* siatki jako *alClient*, aby wypełniała ona cały arkusz zakładki.

2. Projektowanie formularza umożliwiającego odbiór poczty z serwera.

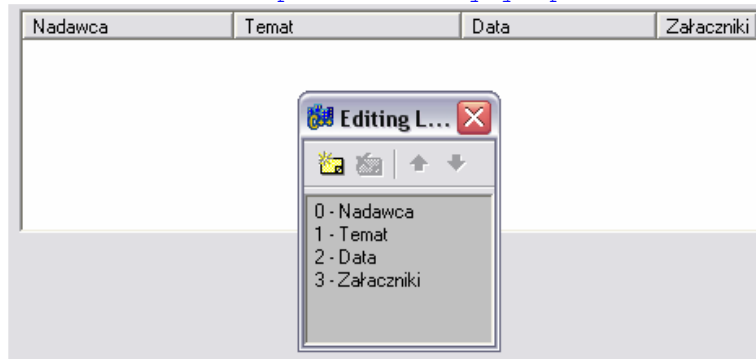
1. Umieść na formularzu następujące komponenty:
 - Trzy pola typu **TEdit** z zakładki **Standard**. Nazwij je odpowiednio *SerwerEdit*, *KontoEdit* oraz *PortEdit*. Wyzeruj właściwości *Text* dwóch pierwszych a dla *PortEdit* wpisz 110, który jest standardowym numerem portu
 - Jedno pole edycyjne typu **TMaskEdit** z zakładki **Additional**. Nazwij je *HasloEdit*. Tu będzie wpisywane hasło potrzebne do zalogowania się na serwer poczty. We właściwość *PasswordChar* wpisz *. Wyczyść właściwość *Text*.

- Pięć etykiet **TLabel** z zakładki **Standard**. Etykiety będą miały za zadanie jedynie opisywać odpowiadające im pola edycyjne. Zmień odpowiednio ich właściwości *Caption* na *Serwer*, *Konto*, *Port*, *Hasło* oraz *Załączniki*.
- Cztery komponenty **TShape** z etykiety **Additional**. Będą to dodatki do etykiet w celu poprawienia efektów wizualnych. Ustaw ich kolor jak na rysunku nr 2.
- Cztery przyciski **TButton** z zakładki **Standard**. Nazwij je odpowiednio *SprawdzBtn*, *PobierzBtn*, *RozlaczBtn* oraz *ZapiszBtn*. Pierwszy będzie wywoływał funkcję, która poinformuje nas, ile jest obecnie wiadomości w skrzynce pocztowej, drugi odbierze je, trzeci będzie powodował odłączenie od serwera a czwarty będzie umożliwiał zapisanie załączników. Zmień ich właściwości *Caption* na *Sprawdź*, *Pobierz*, *Rozłącz* oraz *Zapisz*.
- Trzy komponenty **TGroupBox** z zakładki **Standard**. Będą to dodatki w celu poprawienia efektów wizualnych. Ustaw ich właściwości *Caption* jak na rysunku nr 2.



Rysunek 2: Projektowanie pól edycyjnych

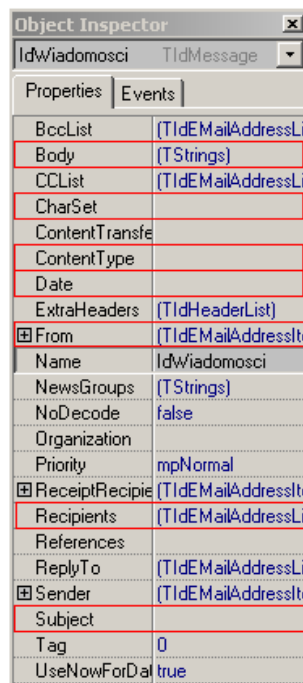
- Komponent **TListView** z zakładki **Win32**, który będzie wyświetlał listę wiadomości ostatnio pobranych z serwera (tytuł, nadawcę, datę oraz ilość załączników). Nazwij go *ListaWiadomosci*. Wybierz właściwość *Columns* i spraw, by lista posiadała cztery kolumny zatytułowane *Nadawca*, *Temat*, *Data* oraz *Załączniki* (Rysunek 3). Ustaw właściwość *ViewStyle* na *VsReport*, a właściwość *ReadOnly* na *true*.



Rysunek 3: Projektowanie kolumn komponentu ListView

- Komponent **TMemo** z zakładki **Standard**. Nazwij go *WiadomoscMemo*. W komponencie tym będzie wyświetlana zawartość aktualnie wybranej wiadomości. Wybierając właściwość *Lines* wyczyść jego pole. Ustaw właściwość *ScrollBars* na *ssBoth* a *ReadOnly* na *true*.
- Komponent **TListBox** z zakładki **Standard**. Nazwij go *ZalacznikListBox*. Tu będą wyświetlane załączniki dołączone do przychodzącej wiadomości.
- Komponent **TStatusBar** z zakładki **Win32**. Nazwij go *StatusBar*. Jest to pasek statusowy. Pasek statusowy zostanie automatycznie ulokowany wzdłuż dolnej krawędzi formularza.
- Komponent **TSaveDialog** z zakładki **Dialogs**. Umożliwi on ustalić nazwę pliku, w który zostanie zapisany załącznik.
- Komponent **TIdMessage** z zakładki **IndyMisc**. Jest to niezbędny komponent. Komponent *IdPOP3* pobierając wiadomość z serwera zwraca obiekt *IdMessage*. Tu zostają zapamiętane poszczególne składniki wiadomości (tytuł, autor, data, treść, załączniki itp.).

Ważniejsze właściwości komponentu **TIdMessage**:



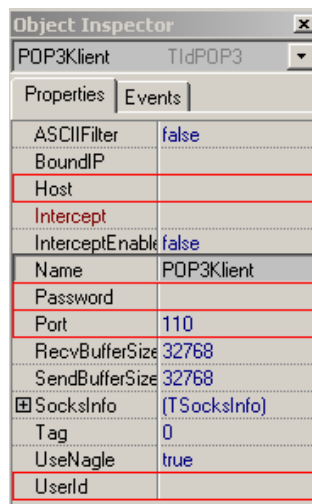
Rysunek 4: Właściwości komponentu TIdMessage

- **TIdMessage->Body** – zawiera treść przesyłanej wiadomości
- **TIdMessage->CharSet** – określa sposób kodowania znaków w wiadomości

www.sprawozdania.yoyo.pl

- **TIdMessage->ContentType** – określa typ wiadomości
- **TIdMessage->Date** – zawiera informacje o dacie wysłania wiadomości
- **TIdMessage->From** – zawiera adres nadawcy wiadomości
- **TIdMessage->Recipients** – zawiera adres odbiorcy wiadomości
- **TIdMessage->Subject** – zawiera temat wiadomości

- **procedure TIdMessage->Clear** – czyści treść i nagłówki wiadomości
- Komponent **TIdPOP3** z zakładki **IndyClients**. Nazwij go *POP3Klient*. Komponent ten realizuje pobieranie poczty elektronicznej w protokole POP3. POP3 służy do przenoszenia poczty elektronicznej z serwera pocztowego na komputer użytkownika. Protokół oparty jest na architekturze klient-serwer, w której pocztę odbiera serwer pocztowy. Serwer przechowuje wiadomość aż do momentu, gdy użytkownik się zaloguje i ją pobierze.
Ważniejsze właściwości i metody komponentu **TIdPOP3**:

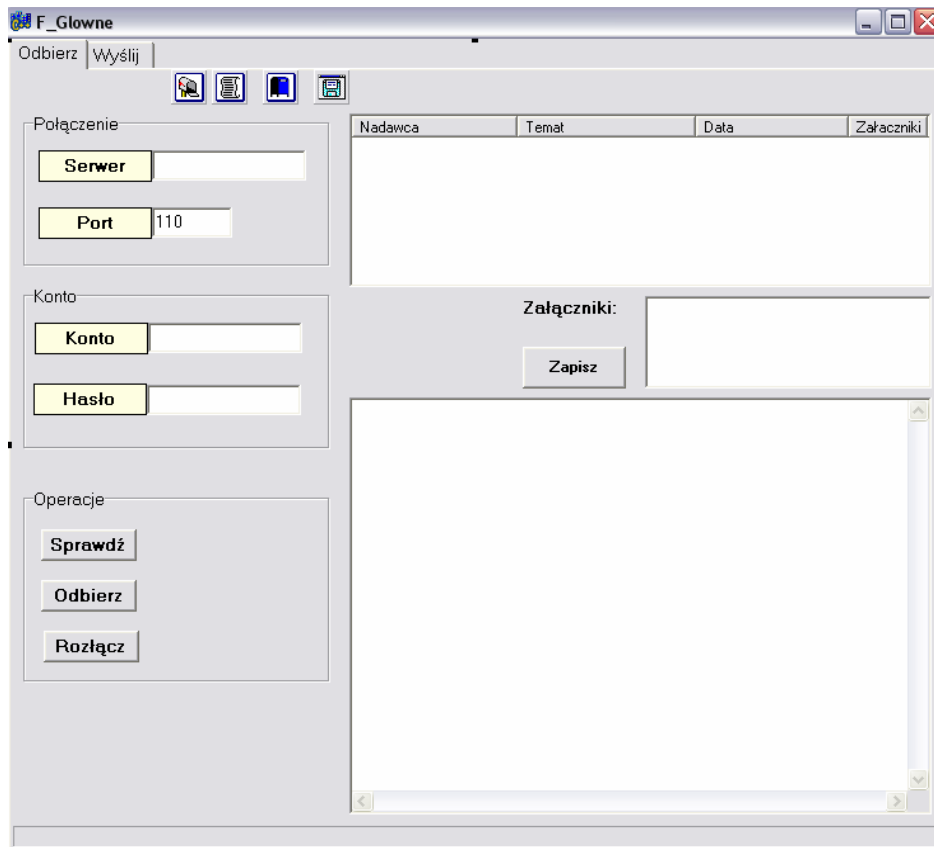


Rysunek 5: Właściwości komponentu TIdPOP3

- **TIdPOP3->Host** - określa adres serwera
- **TIdPOP3->Password** - określa hasło
- **TIdPOP3->Port** - określa port, na którym program będzie łączył się z serwerem
- **TIdPOP3->UserId** – określa login użytkownika.

- **function TIdPOP3->Connected: boolean** - określa czy nawiązano połączenie
- **procedure TIdPOP3->Connect** - próbuje nawiązać połączenie z serwerem
- **function TIdPOP3->CheckMessages: longint** - zwraca liczbę wiadomości w skrzynce
- **procedure TIdPOP3->Disconnect** - kończy połączenie z serwerem.
- **function TIdPOP3->Retrieve(const MsgNum: Integer; AMsg: TIdMessage): Boolean**
– pobiera wiadomość o wskazanym indeksie i umieszcza w zmiennej AMsg

Po wykonaniu powyższej listy czynności należy uruchomić program i sprawdzić jego działanie. Wygląd formularza powinien być zbliżony do formularza zamieszczonego na *Rysunku 6*.



Rysunek 6: Projektowanie formularza umożliwiającego odbiór poczty.

3. Programowanie elementów aplikacji umożliwiającej odbiór poczty z serwera..

1. Jako pierwszą należy napisać funkcję, która będzie ustawiała parametry połączenia:

```
void __fastcall TForm1::ustaw()
{
    POP3Klient->Host = SerwerEdit->Text;
    POP3Klient->Port=StrToInt( PortEdit->Text);
    POP3Klient->UserId = KontoEdit->Text;
    POP3Klient->Password = HasloEdit->Text;
}
```

W funkcji tej realizujemy kolejno operacje:

- ustawienie adresu IP/nazwy serwera
- ustawienie numeru portu
- ustawienie nazwy użytkownika
- ustawienie hasła użytkownika

2. W funkcji obsługi zdarzenia polegającego na naciśnięciu przycisku *SprawdźBtn* wprowadzić następujący kod:

```
void __fastcall TForm1::SprawdzBtnClick(TObject *Sender)
{
    int iloscWiad;
    ustaw();

    if(! POP3Klient->Connected())
    {
        try
        {
            StatusBar->SimpleText="Odbieranie poczty - zestawianie połączenia...";

            POP3Klient->Connect();
        }
    }
}
```

```

        www.sprawozdania.yoyo.pl
        StatusBar->SimpleText="Odbieranie poczty - połączony";
    }
    catch (Exception &e)
    {
        Beep();
        StatusBar->SimpleText="Odbieranie poczty - błąd połączenia";
        ShowMessage("Błąd połączenia z serwerem "+ POP3Klient->Host);
    }
}

if(POP3Klient->Connected())
{
    iloscWiad=POP3Klient->CheckMessages();

    POP3Klient->Disconnect();
    StatusBar->SimpleText="Rozłączony";

    if(iloscWiad)
        ShowMessage("W Twojej skrzynce jest "+IntToStr(iloscWiad)+" wiadomosci");
    else ShowMessage("Brak wiadomosci w skrzynce");

}
}

```

W funkcji tej sprawdzana jest zawartość skrzynki pocztowej. Próbujemy połączyć się z serwerem, pobieramy liczbę wiadomości oczekujących oraz rozłączmy się.

3. W funkcji obsługi zdarzenia polegającego na naciśnięciu przycisku *PobierzBtn* wprowadzić następujący kod:

```

void __fastcall TForm1::PobierzBtnClick(TObject *Sender)
{
    int iloscWiad, iloscZal,zalaczniki;
    TListItem *ListItem;

    IdWiadomosci->Clear();

    ustaw();

    if(! POP3Klient->Connected())
    {
        try
        {
            StatusBar->SimpleText="Odbieranie poczty - zestawianie połączenia...";

            POP3Klient->Connect();
            StatusBar->SimpleText="Odbieranie poczty - połączony";
        }
        catch (Exception &e)
        {
            Beep();
            StatusBar->SimpleText="Odbieranie poczty - błąd połączenia";
            ShowMessage("Błąd połączenia z serwerem "+ POP3Klient->Host);
        }
    }

    if(POP3Klient->Connected())
    {

        iloscWiad=POP3Klient->CheckMessages();
        ListaWiadomosci->Items->Clear();

        for(int i=1; i<=iloscWiad; i++)
        {
            IdWiadomosci = new TIdMessage(ListaWiadomosci);

```

www.sprawozdania.yoyo.pl

```
POP3Klient->Retrieve(i, IdWiadomosci);

iloscZal = IdWiadomosci->MessageParts->Count;

ListItem = ListaWiadomosci->Items->Add();
ListItem->Caption = IdWiadomosci->From->Address;
ListItem->SubItems->Add(IdWiadomosci->Subject);
ListItem->SubItems->Add(IdWiadomosci->Date.DateTimeString());

zalaczniki=0;

for(int j=0; j<iloscZal;j++)
    if(IdWiadomosci->MessageParts->Items[j]->DisplayName=="TIdAttachment")
zalaczniki++;

    ListItem->SubItems->Add(IntToStr(zalaczniki));

    ListItem->Data = IdWiadomosci;
}

POP3Klient->Disconnect();
StatusBar->SimpleText="Rozłączony";
}
}
```

Funkcja ta pobiera wiadomości z serwera i zapamiętuje je.

4. W funkcji obsługi *OnSelectItem* komponentu *ListaWiadomosci* wprowadzić następujący kod:

```
void __fastcall TForm1::ListaWiadomosciSelectItem(TObject *Sender,
    TListItem *Item, bool Selected)
{
    WiadomoscMemo->Lines->Clear();

    if(Selected && Item)
    {
        ZalacznikListBox->Clear();
        IdWiadomosci = (TIdMessage*)Item->Data;

        for(int i=0;i<IdWiadomosci->MessageParts->Count;i++)
        {
            if(IdWiadomosci->MessageParts->Items[i]->DisplayName=="TIdText")
            {
                TIdText *IdText = (TIdText*)IdWiadomosci->MessageParts-
>Items[i];
                WiadomoscMemo->Lines->AddStrings(IdText->Body);
            }

            if(IdWiadomosci->MessageParts->Items[i]-
>DisplayName=="TIdAttachment")
            {
                TIdAttachment *zal = (TIdAttachment*)IdWiadomosci->MessageParts-
>Items[i];
                ZalacznikListBox->Items->Add(zal->FileName);
            }
        }
    }
}
```

Funkcja ta będzie wyświetlała tekst wiadomości.

5. W funkcji obsługi zdarzenia *OnCanClose* komponentu *SaveDialog1* wprowadzić następujący kod:

```
void __fastcall TForm1::SaveDialog1CanClose(TObject *Sender,
    bool &CanClose)
{
    int j=0;

    for(int i=0;i<IdWiadomosci->MessageParts->Count;i++)
    {
        if(IdWiadomosci->MessageParts->Items[i]->DisplayName=="TIdAttachment")
        {
            TIdAttachment *zal = (TIdAttachment*)IdWiadomosci->MessageParts-
>Items[i];
            if(ZalacznikListBox->ItemIndex==j) zal->SaveToFile(SaveDialog1-
>FileName);

            j++;
        }
    }
}
```

6. W funkcji obsługi zdarzenia polegającego na naciśnięciu przycisku *Zapisz* wprowadzić następujący kod:

```
void __fastcall TForm1::ZapiszBtnClick(TObject *Sender)
{
    if(ZalacznikListBox->ItemIndex!=-1)
    {
        SaveDialog1->FileName=ZalacznikListBox->Items->Strings[ZalacznikListBox-
>ItemIndex];
        SaveDialog1->Execute();
    }
}
```

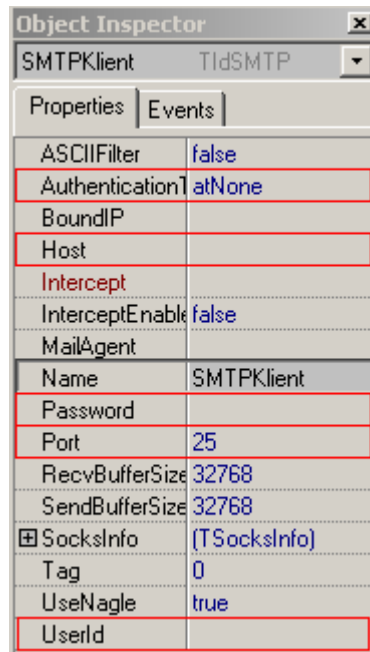
7. W funkcji obsługi zdarzenia polegającego na naciśnięciu przycisku *Rozłącz* wprowadzić następujący kod:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    if(POP3Klient->Connected())
    {
        POP3Klient->Disconnect();
        StatusBar->SimpleText="Rozłączony";
    }
}
```

4. Projektowanie formularza umożliwiającego wysyłanie poczty.

1. Przejdź na zakładkę *Wyślij*.
2. Umieść na formularzu następujące komponenty:
 - Sześć pól typu **TEdit** z zakładki **Standard**. Nazwij je odpowiednio *SerwerEdit2*, *KontoEdit2*, *PortEdit2*, *DoEdit*, *TematEdit* oraz *Nadawca Edit*. Wyzeruj ich właściwości *Text* a dla *PortEdit* wpisz 25, który jest standardowym numerem portu
 - Jedno pole edycyjne typu **TMaskEdit** z zakładki **Additional**. Nazwij je *HasloEdit2*. Tu będziesz wpisywał hasło potrzebne do zalogowania się na serwer poczty. We właściwość *PasswordChar* wpisz *. Wyczyść właściwość *Text*.
 - Osiem etykiet **TLabel** z zakładki **Standard**. Etykiety będą miały za zadanie jedynie opisywać odpowiadające im pola edycyjne. Zmień odpowiednio ich właściwości *Caption* na *Serwer*, *Konto*, *Port*, *Hasło*, *Do*, *Temat*, *Nadawca* oraz *Treśćwiadomości*.
 - Siedem komponentów **TShape** z etykiety **Additional**. Będą to dodatki do etykiet w celu poprawienia efektów wizualnych. Ustaw ich kolor jak na rysunku nr 8.
 - Dwa przyciski **TButton** z zakładki **Standard**. Nazwij je *WyślijBtn* oraz *LadujBtn*. Zmień ich właściwości *Caption* na *Wyślij* oraz *Załącznik*.
 - Pięć komponentów **TGroupBox** z zakładki **Standard**. Będą to dodatki w celu poprawienia efektów wizualnych. Ustaw ich właściwości *Caption* jak na rysunku nr 8.
 - Komponent **TMemo** z zakładki **Standard**. Nazwij go *TrescMemo*. Wybierając właściwość *Lines* wyczyść jego pole. Ustaw właściwość *ScrollBars* na *ssVertical*.
 - Komponent **TListBox** z zakładki **Standard**. Nazwij go *Zalacznik2ListBox*. Tu będą wyświetlane załączniki dołączone do wysyłanej wiadomości.
 - Komponent **TStatusBar** z zakładki **Win32**. Nazwij go *StatusBar2*. Jest to pasek statusowy. Pasek statusowy zostanie automatycznie ulokowany wzdłuż dolnej krawędzi formularza.
 - Komponent **TOpenDialog** z zakładki **Dialogs**. Umożliwi on ustalić nazwę pliku, w który zostanie dodany do wiadomości jako załącznik.
 - Komponent **TIdSMTP** z zakładki **IndyClients**. Nazwij go *SMTPKlient*. Komponent ten realizuje wysyłanie poczty elektronicznej w protokole SMTP. Protokół ten jest głównie używany do przenoszenia poczty elektronicznej.

Ważniejsze właściwości i metody komponentu **TIdSMTP**:



Rysunek 7: Właściwości komponentu TIdSMTP

- **TIdSMTP-> AuthenticationType** - ustawienie trybu autentifikacji
- **TIdSMTP->Host** - określa adres serwera
- **TIdSMTP->Password** - właściwość przechowująca hasło
- **TIdSMTP->Port** - określa port, na którym program będzie łączył się z serwerem
- **TIdSMTP->UserId** - w tej właściwości zawarty jest login użytkownika.

- **function TIdSMTP->Connected: boolean** - określa czy nawiązano połączenie
- **procedure TIdSMTP->Connect** - próbuje nawiązać połączenie z serwerem
- **procedure TIdSMTP->Send(AMsg: TIdMessage)longint** – wysyła wiadomość
- **procedure TIdSMTP->Disconnect** - kończy połączenie z serwerem.

Po wykonaniu powyższej listy czynności należy uruchomić program i sprawdzić jego działanie. Wygląd formularza powinien być zbliżony do formularza zamieszczonego na *Rysunku 8*.

Rysunek 8: Projektowanie formularza umożliwiającego wysyłanie poczty.

5. Programowanie elementów aplikacji umożliwiającej odbiór poczty z serwera..

1. Jako pierwszą należy napisać funkcję, która będzie ustawiała parametry połączenia:

```
void __fastcall TForm1::ustawSMTP()
{
    SMTPKlient->Host = SerwerEdit2->Text;
    SMTPKlient->Port=StrToInt (PortEdit2->Text);
    SMTPKlient->UserId = KontoEdit2->Text;
    SMTPKlient->Password = HasloEdit2->Text;
    SMTPKlient->AuthenticationType = atLogin;
}
```

W funkcji tej realizujemy kolejno operacje:

- ustawienie adresu IP/nazwy serwera
- ustawienie numeru portu
- ustawienie nazwy użytkownika
- ustawienie hasła użytkownika
- ustawienie trybu autentifikacji

2. W funkcji obsługi zdarzenia polegającego na naciśnięciu przycisku *WyślijBtn* wprowadzić następujący kod:

```
void __fastcall TForm1::WyślijBtnClick(TObject *Sender)
{
    IdWiadomosci->Clear();

    ustawSMTP();

    if(! SMTPKlient->Connected())
```

www.sprawozdania.yoyo.pl

```
{
try
{
    StatusBar2->SimpleText="Zestawienie połączenia z serwerem...";

    SMTPKlient->Connect();
    StatusBar2->SimpleText="Połączony...";
}
catch (Exception &e)
{
    Beep();
    ShowMessage("Błąd połączenia z serwerem "+ SMTPKlient->Host);
    StatusBar2->SimpleText="Błąd połączenia z serwerem "+ SMTPKlient->Host;
}
}

if(SMTPKlient->Connected())
{
    IdWiadomosci->From->Text = NadawcaEdit->Text;
    IdWiadomosci->Subject = TematEdit->Text;
    IdWiadomosci->Recipients->EMailAddresses = DoEdit->Text;
    IdWiadomosci->Body->Text = TrescMemo->Lines->Text;

    for(int i=1;i<=Zalacznik2ListBox->Items->Count;i++)
        TIdAttachment(IdWiadomosci->MessageParts, Zalacznik2ListBox->Items->Strings[i-1]);

    IdWiadomosci->ContentType = "text/plain";
    IdWiadomosci->CharSet = "charset=iso-8859-1";

    StatusBar2->SimpleText = "Wysyłanie wiadomości ...";

    try
    {
        SMTPKlient->Send(IdWiadomosci);

        StatusBar2->SimpleText = "Wiadomość wysłana";
        IdWiadomosci->Clear();

        NadawcaEdit->Clear();
        TematEdit->Clear();
        DoEdit->Clear();
        TrescMemo->Clear();
        Zalacznik2ListBox->Clear();

        SMTPKlient->Disconnect();
        StatusBar2->SimpleText="Rozłączony";

        ShowMessage("Wiadomość wysłana");
    }
    catch (Exception &e)
    {
        Beep();
        ShowMessage("Błąd przy wysyłaniu wiadomości");
        StatusBar2->SimpleText = "Błąd przy wysyłaniu wiadomości !!!";
        IdWiadomosci->Clear();
    }
}
}
```

3. W funkcji obsługi zdarzenia polegającego na naciśnięciu przycisku *LadujBtn* wprowadzić następujący kod:

```
void __fastcall TForm1::LadujBtnClick(TObject *Sender)
{
    OpenDialog1->Execute();
    Zalacznik2ListBox->Items->Add(OpenDialog1->FileName);
}
```