

C++ Builder : CRUD

Criado em: 03/03/2021 por Vlademiro

Atualizado em: 08/03/2021

Visão geral e objetivo

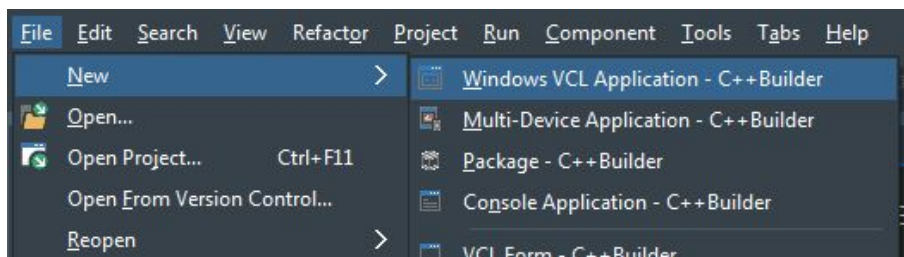
Criar uma tela de CRUD com C++ Builder 10.3.3

Procedimento geral

1. Criação do projeto
2. Abertura de formulário.
3. Montagem da tela de clientes
4. Inserção de labels
5. Data modules
6. Habilitar e desabilitar botões
7. Preparação do Formulário
8. Inclusão
9. Formatação do grid
10. Trazendo dados do banco para o formulário
11. Gravação da alteração
12. Cancelamento
13. Exclusão

Passo à Passo

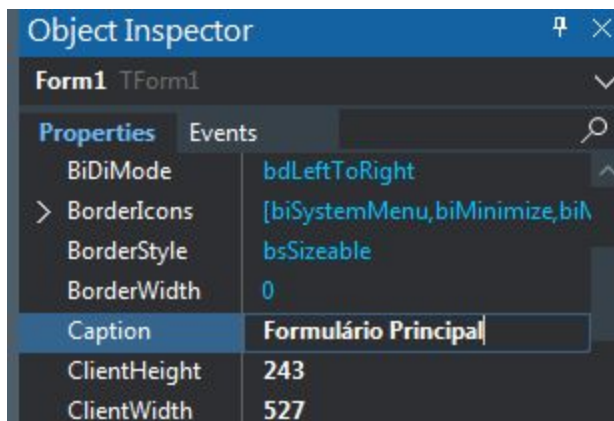
1. Criação do projeto
 - 1.1. Ao abrir o C++ Builder crie um novo projeto VCL (Somente para windows)



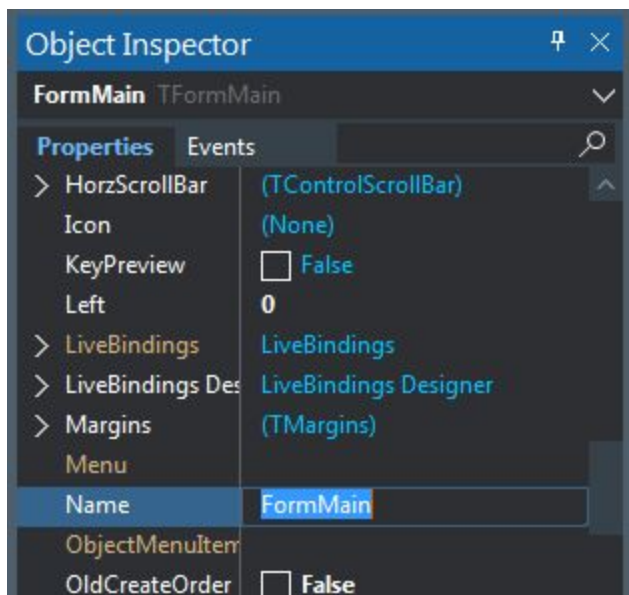
1.2. Atribua as seguintes propriedades ao formulário recém-criado :

Name :	FormMain
Caption :	Formulário principal

Primeiro o caption :



Depois o name :



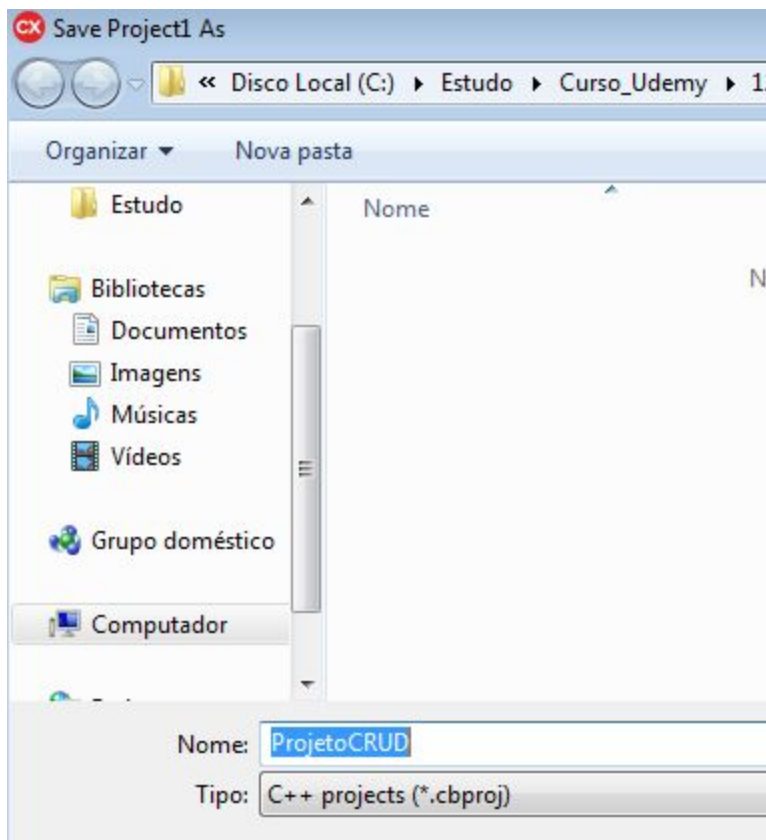
Dica : Quando você digitar o valor de uma propriedade tecle ENTER para gravar o valor. Se você simplesmente “sair” da edição e for para outro local da IDE o valor não será gravado.

1.3. Agora salve o projeto :



Adote a seguinte nomenclatura :

Arquivo cpp	Digite “UnitPrincipal” (vai gerar 3 arquivos, com as respectivas extensões : cpp, h e dfm).
ProjectPCH1	Digite “CRUD”
Para o projeto C++ digite	Digite “ProjetoCRUD” (veja figura abaixo)



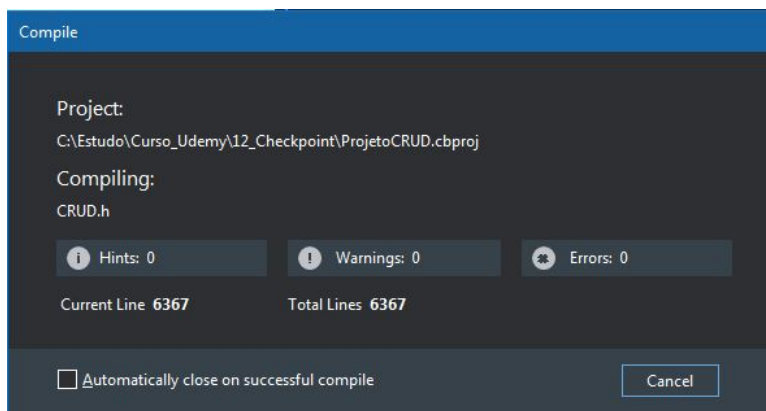
Sua pasta deve ficar assim :

Nome	Data de modificaç...	Tipo	Tamanho
__astcache	03/03/2021 22:21	Pasta de arquivos	
CRUD	03/03/2021 22:16	Header file	1 KB
ProjetoCRUD	03/03/2021 22:21	BCB Project File	54 KB
ProjetoCRUD.cbproj.local	03/03/2021 22:21	Arquivo LOCAL	2 KB
ProjetoCRUD	03/03/2021 22:21	Arquivo CPP	1 KB
UnitPrincipal	03/03/2021 22:17	Arquivo CPP	1 KB
UnitPrincipal.dfm	03/03/2021 22:17	Arquivo DFM	1 KB
UnitPrincipal	03/03/2021 22:17	Header file	1 KB

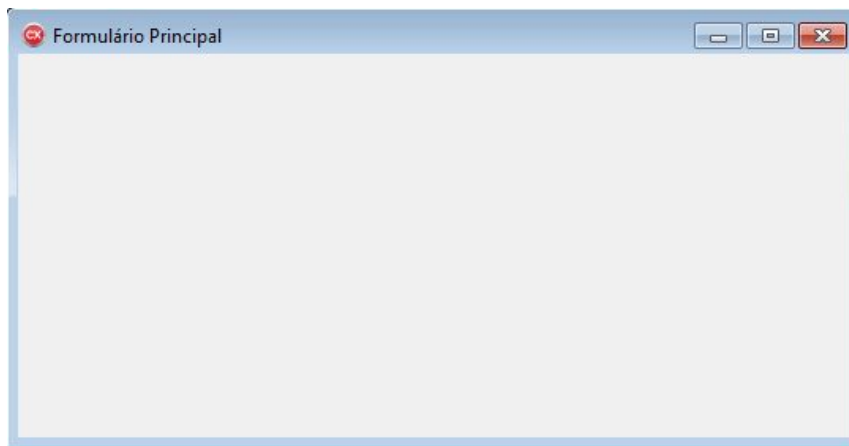
1.4. Para compilar clique na seta verde :



Processo de compilação :



Resultado:



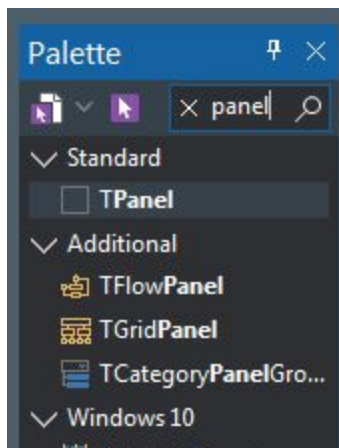
2. Abertura de formulário.

Vamos agora criar um formulário e abri-lo a partir do formulário principal. Primeiramente vamos criar um botão dentro de um painel, ainda na janela principal.

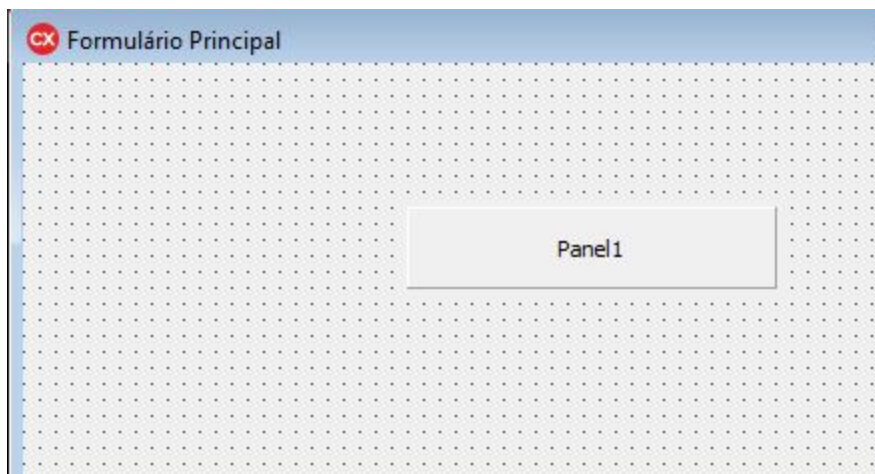
2.1. Primeiro o painel:

No canto direito, janela Palette selecione o componente TPanel contido no grupo de controles Standard.

Dica : use o campo de busca na parte superior da janela e digite "Panel"



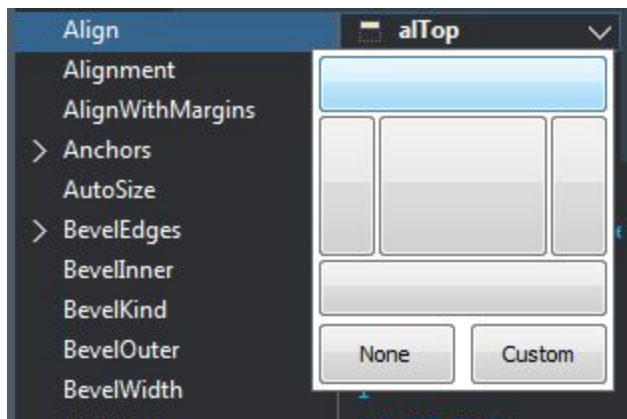
Clique sobre TPanel e o arraste para a janela do seu projeto. A janela do formulário principal ficará assim :



Vamos alinhar o painel no canto superior do formulário. Clique sobre o Panel1, note que a janela “Object Inspector” agora exibe as propriedades do painel.

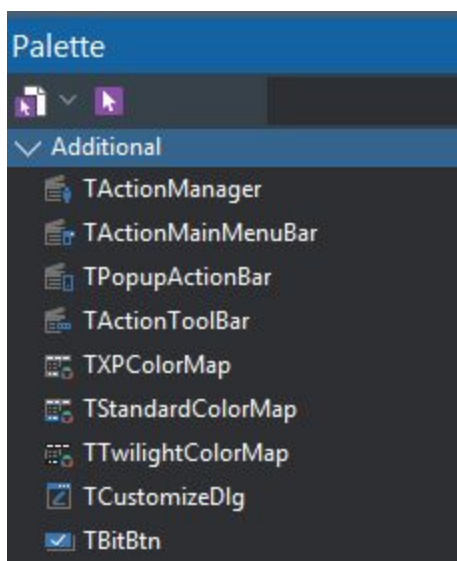
Altere as seguintes propriedades :

Caption	Apague “Panel1”. Deixe em branco
Align	AlTop (se baseie na figura abaixo)
Color	Azul claro



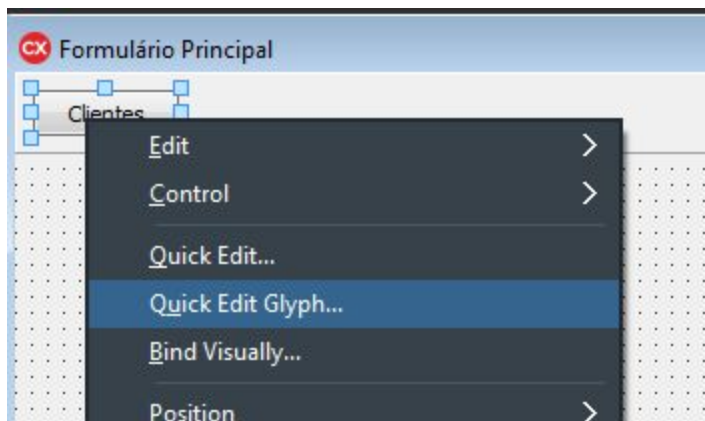
2.2. Agora insira um botão. Esse botão será usado para chamar o futuro formulário de cliente :

O botão será do tipo "TBitBtn" (com imagem) e está no grupo Additional.

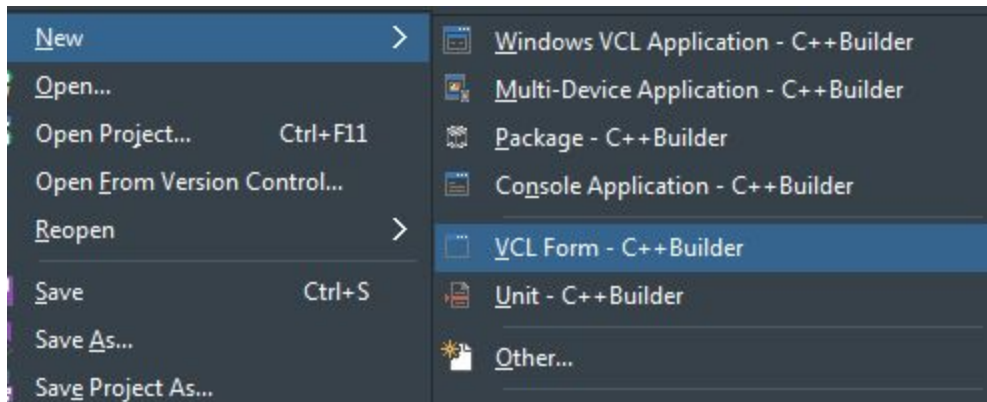


Caption	Clientes
Name	btCliente

Depois de inserir o botão, insira uma imagem nele. Faça isso clicando com o botão direito sobre o botão e selecione "Quick Edit Glyph...".



2.3. Agora vamos criar um novo formulário (nos próximos passos ele será usado para cadastrar os clientes)

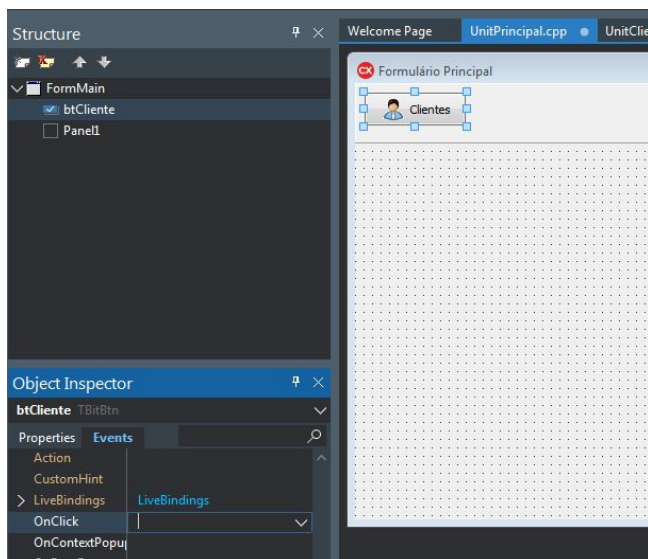


Name	FormCliente
Caption	Manutenção de clientes

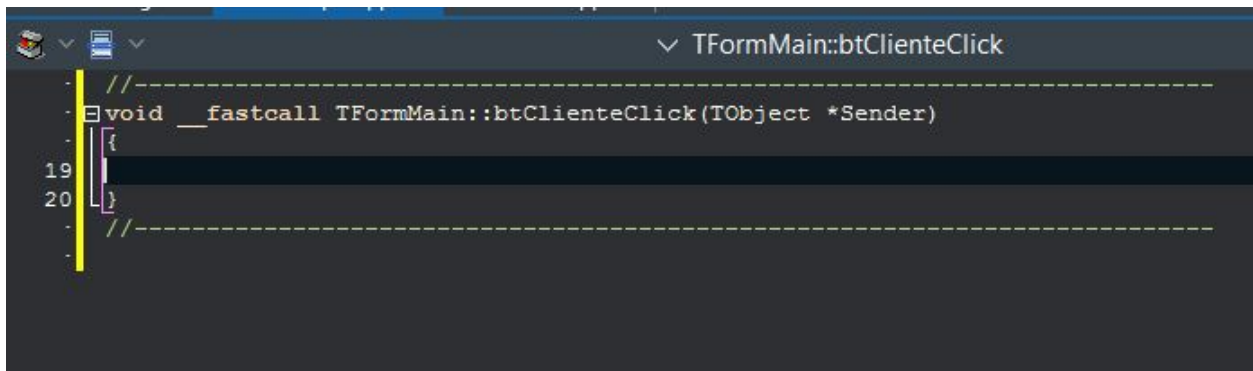
Salve o formulário (Ctrl+S) com o seguinte nome : “UnitCliente”

2.4. Agora vamos criar o seguinte evento : quando o usuário clicar no botão criado no formulário principal, o FormCliente deverá ser aberto.

2.4.1. Clique duas vezes sobre o evento “click”.



2.4.2. Quando você clica duas vezes a ferramenta lhe direciona para a edição do código:



```
-----  
void __fastcall TFormMain::btClienteClick(TObject *Sender)  
{  
19  
20  
}  
-----
```

2.4.3. Digite o seguinte código

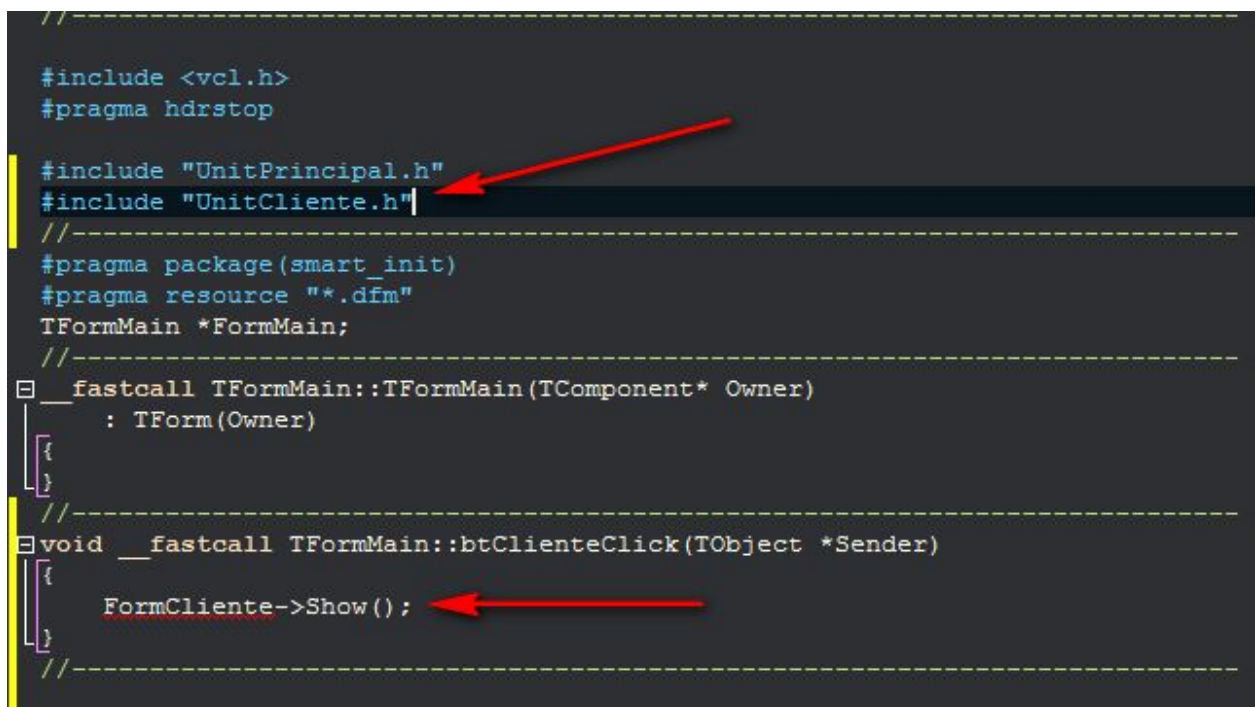
No cabeçalho :

```
#include "UnitCliente.h"
```

No evento :

```
FormCliente->Show();
```

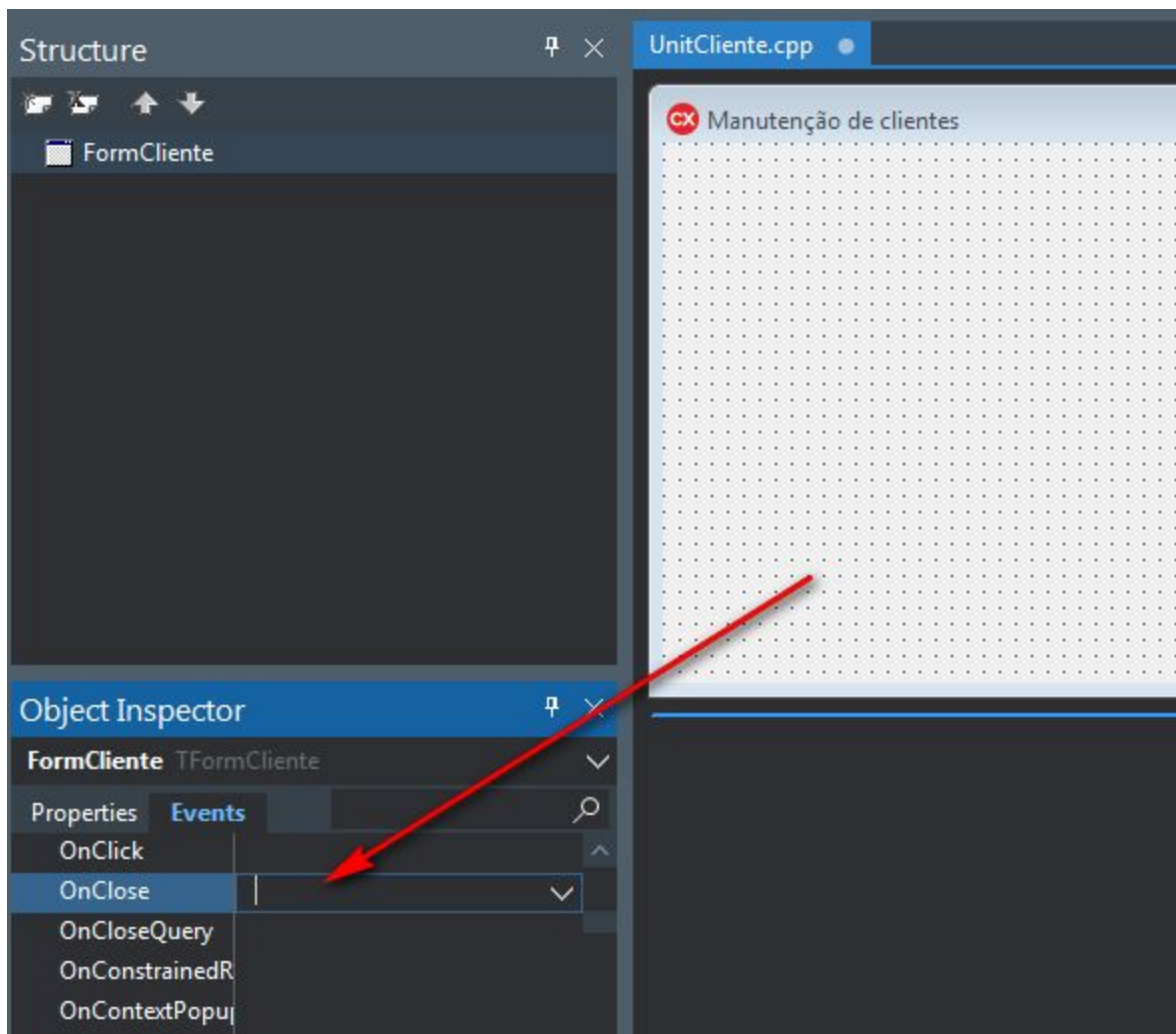
Assim :



```
-----  
#include <vcl.h>  
#pragma hdrstop  
#include "UnitPrincipal.h"  
#include "UnitCliente.h"  
-----  
#pragma package(smart_init)  
#pragma resource "*.dfm"  
TFormMain *FormMain;  
-----  
__fastcall TFormMain::TFormMain(TComponent* Owner)  
: TForm(Owner)  
{  
}  
-----  
void __fastcall TFormMain::btClienteClick(TObject *Sender)  
{  
    FormCliente->Show();  
}  
-----
```

2.4.4. Ainda falta um detalhe : você precisa liberar a memória do formulário de clientes após ele ser fechado. Isso é feito através do evento “onClose”

Dica: caso você esteja no código e quiser alternar para o modo visual (e vice-versa) basta teclar F12.



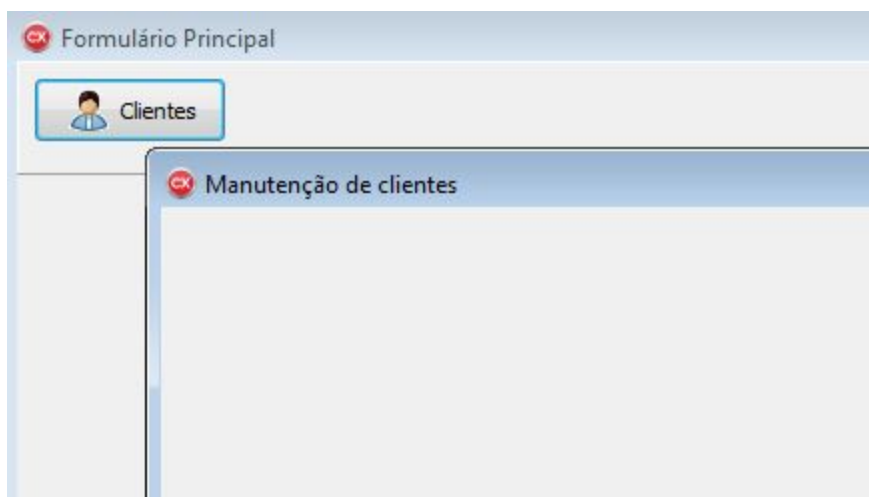
Ao clicar duas vezes sobre o OnClose um código será gerado e você precisa digitar o trecho abaixo para liberar a memória.

```
Action = TCloseAction::caFree;
```

Assim :

```
void __fastcall TFormCliente::FormClose(TObject *Sender, TCloseAction &Action)
{
    Action = TCloseAction::caFree;
}
```

Resultado até agora :



3. Montagem da tela de clientes

Abra o formulário de clientes.

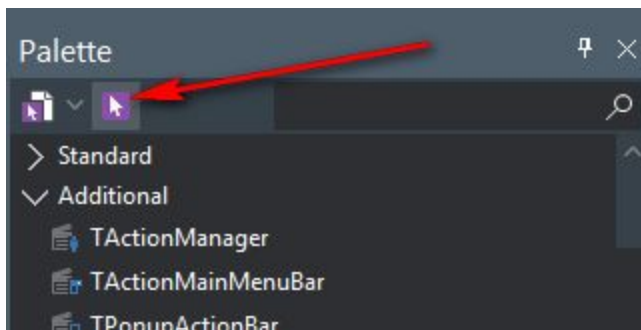
3.1. Insira um Panel

Align	Bottom
Cor	Azul
Caption	Sem caption

3.2. Insira 6 botões (BitBtn) dentro do painel



Dica : Para não ter que ficar clicando repetidas vezes no componente TBitBtn faça assim: pressione a tecla SHIFT enquanto seleciona o primeiro botão. Clique repetidamente sobre o painel e note que ,a cada clique, um novo botão aparece. Para finalizar esse modo de inserção clique no botão representado na figura abaixo :



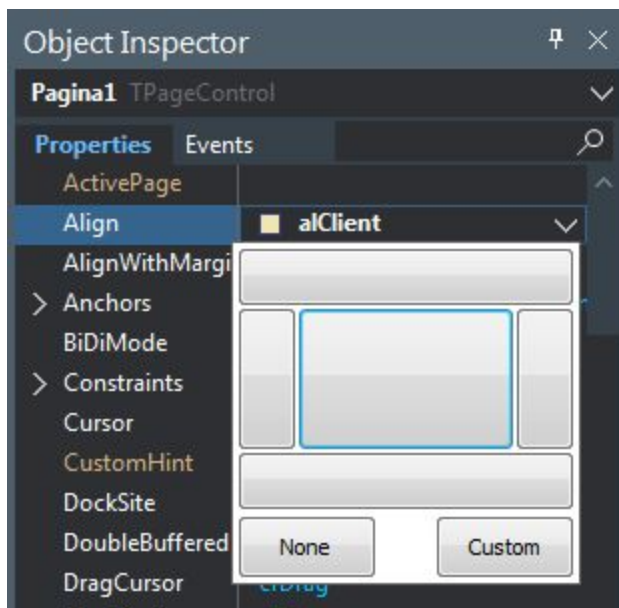
Certifique-se de que as propriedades “Name” e “Caption” de cada botão está conforme o quadro abaixo :

Name	Caption
btIncluir	Incluir
btAlterar	Alterar
btExcluir	Excluir
btCancelar	Cancelar
btGravar	Gravar
btSair	Sair

3.3. Agora iremos inserir **2 abas** no centro do formulário. A ideia é a seguinte: a primeira aba conterá um grid com a lista e a segunda aba conterá o formulário para a edição dos dados.

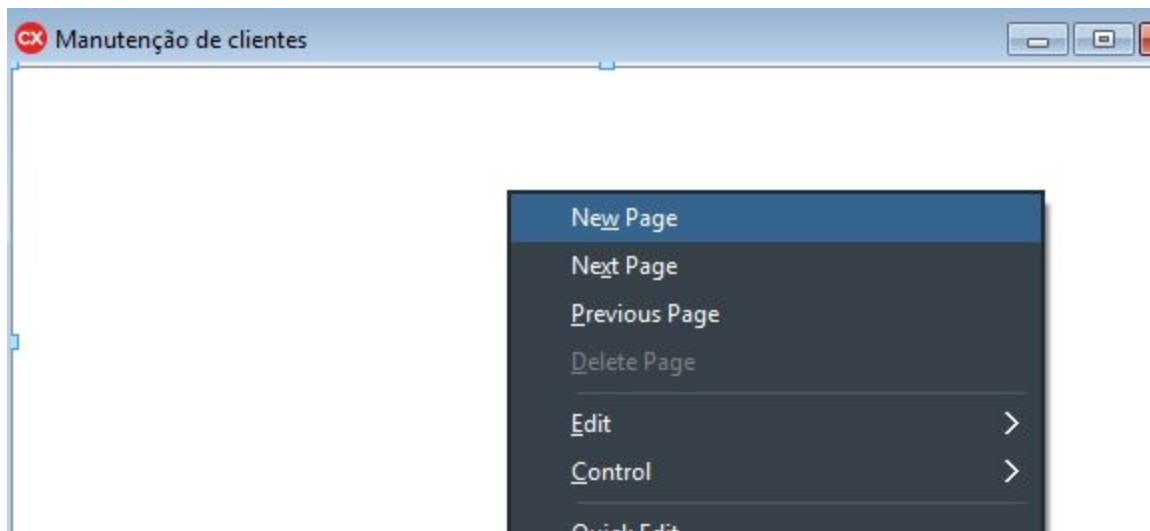
3.3.1. Insira um PageControl (Grupo Win32)

Name	Pagina1
Align	alClient

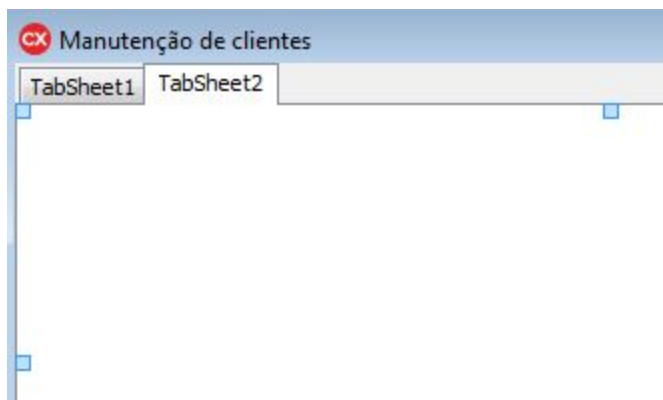


3.3.2 Insira um TabSheet para o grid e outro TabSheet para o formulário (dentro do PageControl)

Para inserir a TabSheet clique com o botão direito dentro do TPageControl e selecione a opção "New Page" do menu.



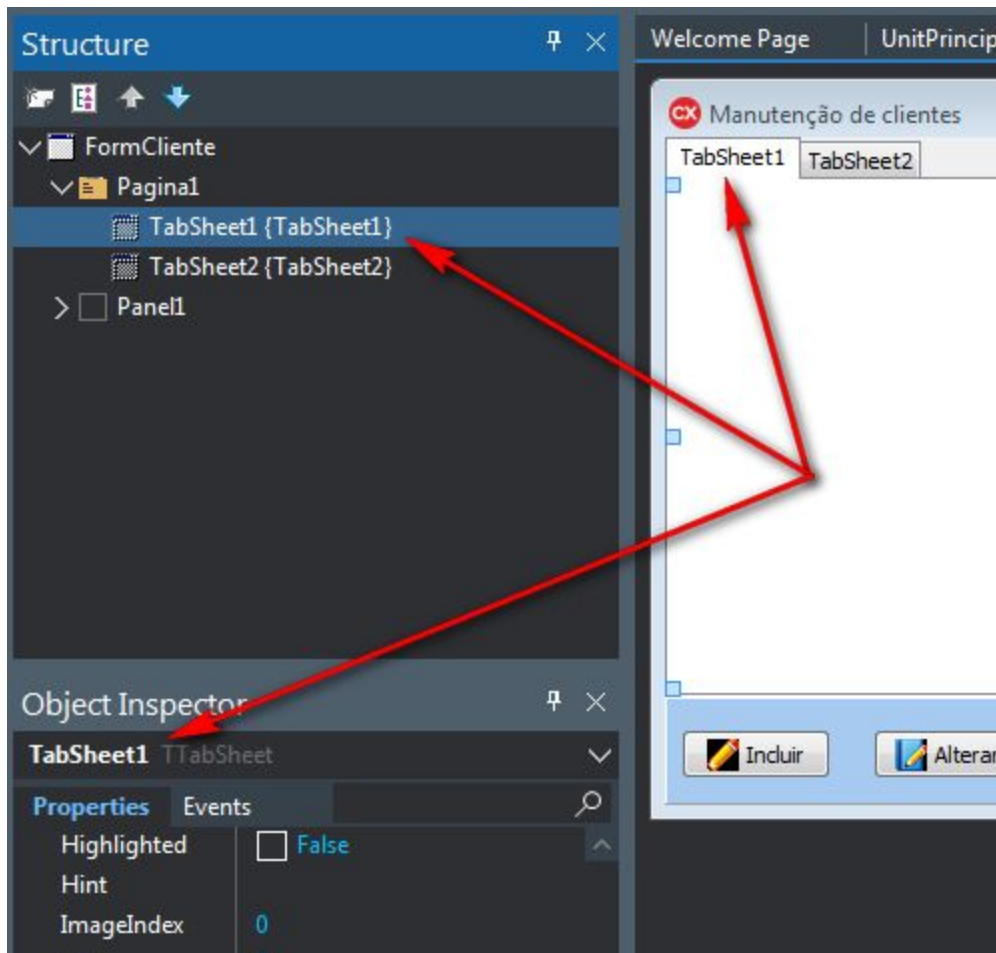
Repita o procedimento para a outra TabSheet.
O seu projeto deverá ficar assim:



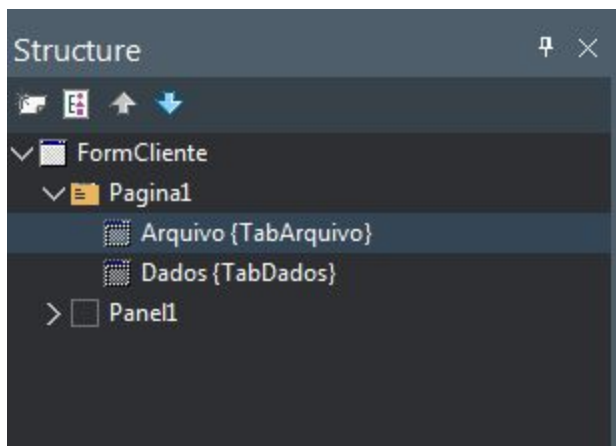
Agora altere as seguintes propriedades

	TabSheet1	TabSheet2
Name	TabArquivo	TabDados
Caption	Arquivo	Dados

Dica : Para facilitar a seleção das abas (TabSheet) note que no painel superior à sua esquerda tem uma árvore contendo os componentes do formulário selecionado. Nele você pode alterar a ordem dos componentes e também selecioná-los para poder alterar as propriedades.



Após a alteração, a sua estrutura ficará assim :

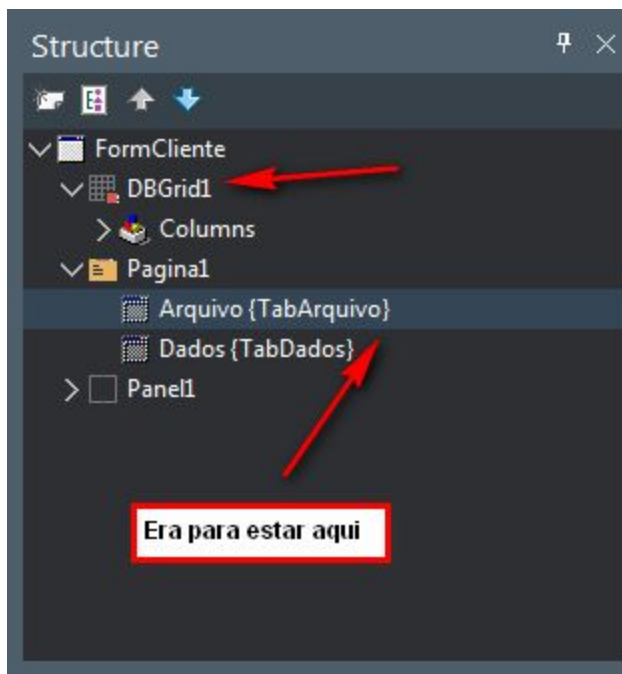


3.3.3 Insira um DBGrid (Componente do grupo “DataControl”) dentro da aba “TabArquivo”. Para facilitar, use o painel “Structure” para selecionar o componente.

Altere as seguintes propriedades do grid :

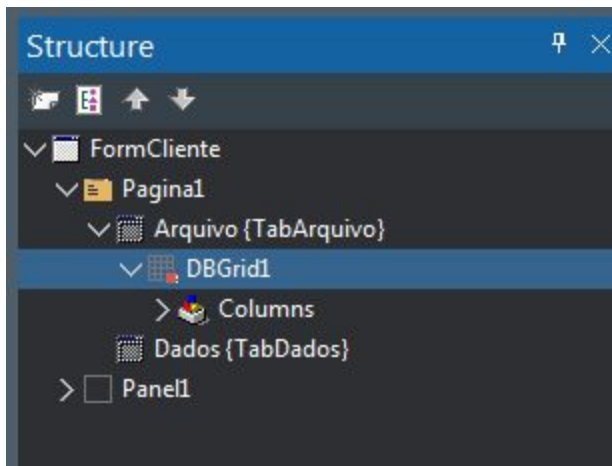
Name	
Align	alClient

Dica : Mesmo você selecionando o local onde o grid será inserido (no nosso caso a aba “TabArquivo”), pode ser que o C++ Builder não coloque o grid no local correto. O painel “Structure” mostrará isso assim :

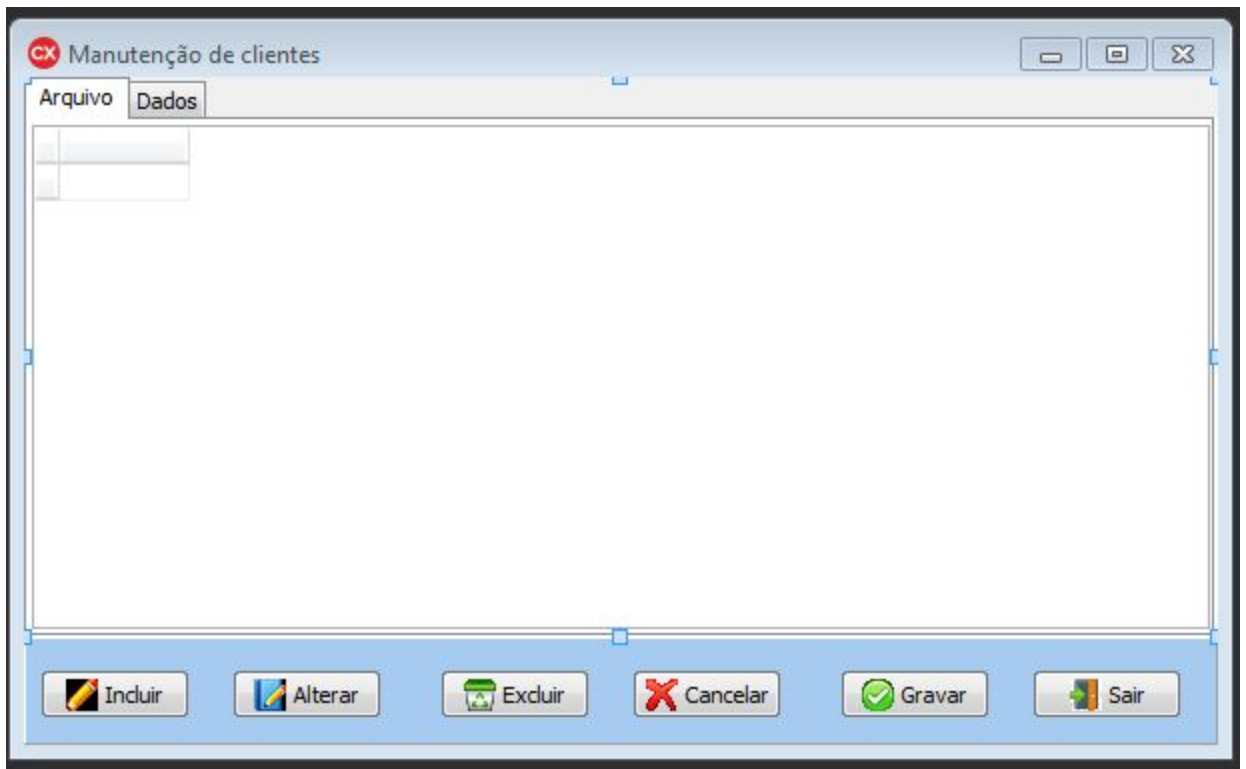


Caso isso aconteça você deverá clicar sobre o DBGrid1 e arrastar ele para dentro da aba “TabArquivo”. Faça isso dentro do painel “Structure”.

Após mover o seu painel “Structure” ficará assim :



Nossa tela está assim :



4. Inserção de labels

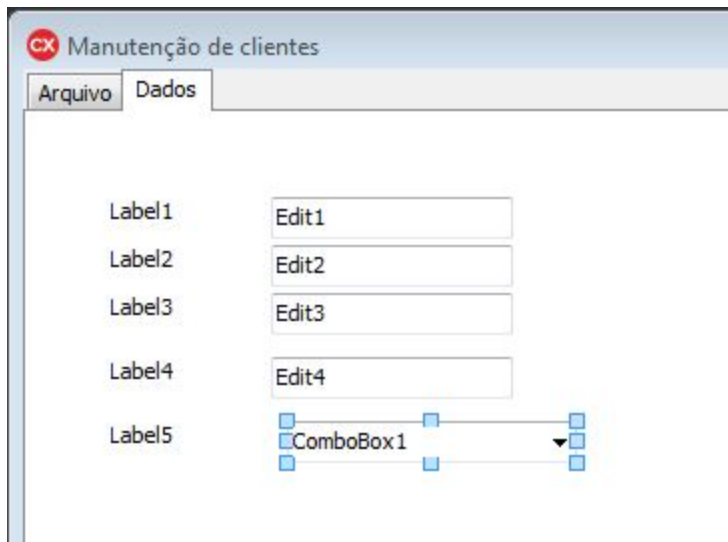
Agora iremos inserir labels e controles na aba “Dados”

4.1. Selecione a aba “TabDados” usando o painel “Structure”.

4.2. Insira 5 labels (painel “Palette”, grupo “Standard”)

4.3. Insira 4 edits e 1 combobox (painel “Palette”, grupo “Standard”)

Sua tela ficará assim :



4.4. Altere a propriedade name dos controles inseridos

EdtID
EdtNome
EdtEndereco
EdtCidade
CBEstado

Note que o prefixo “Edt” refere-se ao edit e o prefixo “CB” refere-se ao combobox.

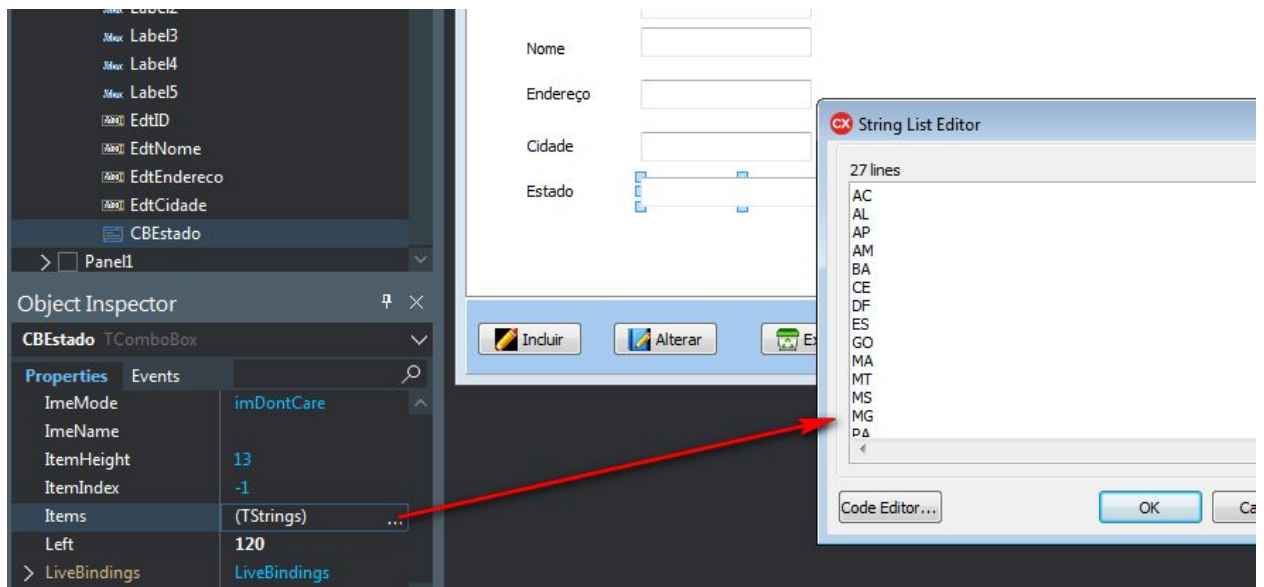
4.5. Exclua o conteúdo da propriedade “text” de todos os controles que você inseriu.

The screenshot shows a Windows Forms application window titled "Manutenção de clientes" with a red "CX" icon. It has a menu bar with "Arquivo" and "Dados". The main area contains five labels, each followed by a text control: Label1 with a text box, Label2 with a text box, Label3 with a text box, Label4 with a text box, and Label5 with a dropdown menu.

4.6. Nos labels, altere a propriedade “Caption” dos labels.

The screenshot shows the same application window, but the labels have been updated with specific field names: "ID", "Nome", "Endereço", "Cidade", and "Estado". Each label is followed by its corresponding text control: a text box for ID, Nome, and Cidade; and a dropdown menu for Endereço and Estado.

4.7. Agora vamos inserir as opções do Combobox com a lista de estados. Para isso selecione o controle CBEstado e, na propriedade “Items”, insira os estados.



4.8. Agora vamos codificar o fechamento do formulário através do evento “Click” do botão “Sair”

Clique sobre o botão “Sair”. Você será direcionado para o código correspondente. Nele insira a função Close();

```
void __fastcall TFormCliente::btSairClick(TObject *Sender)
{
    Close();
}
//-----
```

5. Data modules

Agora iremos configurar o acesso ao banco de dados. Vamos partir do princípio que o banco de dados já está configurado e funcionando. Trata-se de um banco de dados Firebird, com apenas uma tabela.

```
CREATE TABLE CLIENTE
(
  ID integer NOT NULL,
  NOME varchar(50) COLLATE WIN_PTBR,
  ENDERECO varchar(50) COLLATE WIN_PTBR,
  CIDADE varchar(50) COLLATE WIN_PTBR,
  ESTADO varchar(2) COLLATE WIN_PTBR
);
```

O campo ID será alimentado por um generator.

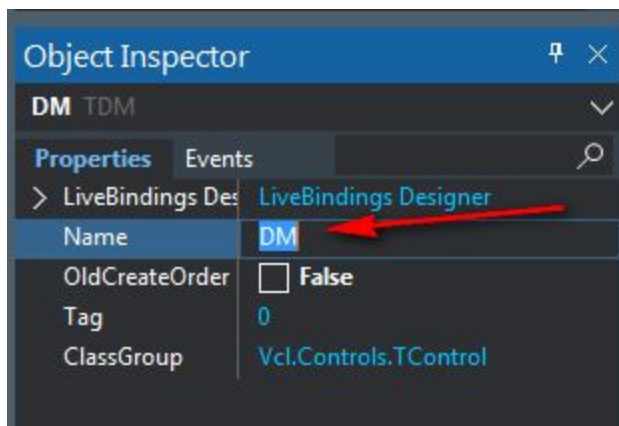
```
CREATE SEQUENCE ID_CLIENTE ;
```

5.1. Primeiro vamos adicionar um Data Module.

New → Other → Database

Salve o datamodule com o nome **UnitDM**

Coloque o nome do Data Module = DM



5.2. Agora vamos adicionar um componente FireDAC chamado TFDConnection que irá gerenciar a conexão ao banco de dados Firebird.

Name	ConexaoDB
DriverName	FB (Selecione da lista)
Params	CharacterSet : csWIN1252 Database : Path para o arquivo FDB DriverID : FB UserName : SYSDBA Password : masterkey

5.3. No grupo FireDAC Links insira um componente FDPhysFBDriverLink e na propriedade **VendorLib** coloque o link para a sua DLL do firebird (fbclient.dll) Esse componente é utilizado automaticamente pelos demais componentes, de modo que não precisa nem ser nomeado. Deixe, portanto o seu nome padrão, que é **FDPhysFBDriverLink1**

5.4. Para testar a conexão vá nas propriedades de ConexaoDB, e em "Connected" marque True.

5.5. Caso nenhum erro tenha ocorrido, vamos agora adicionar as Querys do CRUD. São 4 queries: SELECT, INSERT, UPDATE e DELETE.

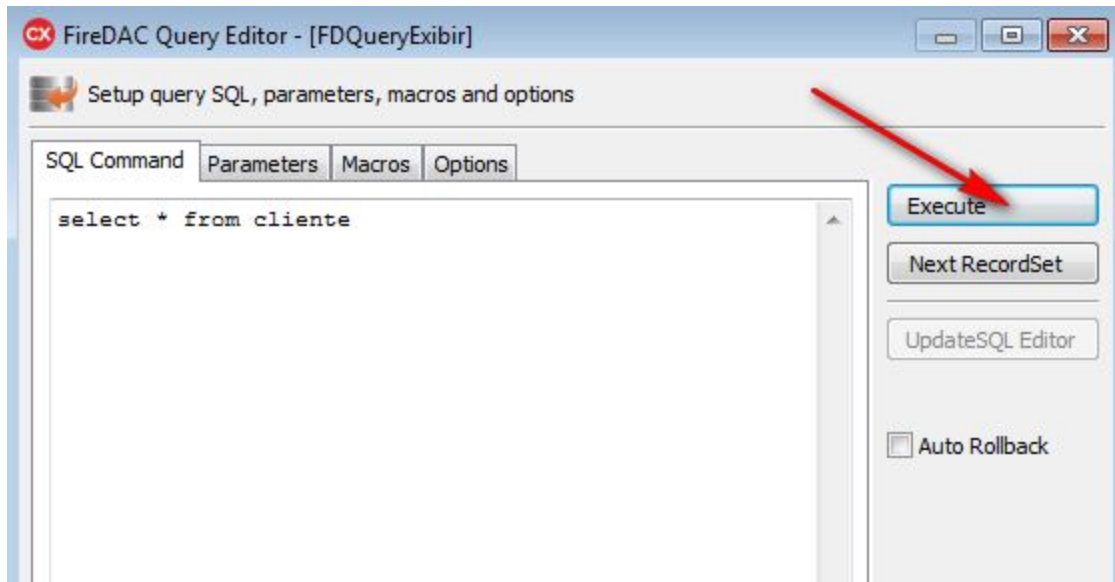
5.5.1. Query SELECT : que irá preencher o DBGrid

5.5.1.1. Insira um componente do tipo FireDAC chamado TFDQuery.

Connection	ConexaoDB
Name	FDQueryExibir
SQL	select * from cliente

Dica : Para inserir o código SQL clique duas vezes sobre o ícone do componente FDQueryExibir.

Dica : Você pode realizar um teste simples para ver se o componente realmente está conseguindo listar os dados. Clique duas vezes sobre o componente FDQueryExibir e depois clique no botão “Execute”.



5.5.1.2. Associado ao componente TFDQuery que você inseriu, insira agora um componente do tipo Data Access chamado TDataSource

Name	DSExibir
DataSet	FDQueryExibir

5.5.2. Query INSERT

O procedimento é semelhante ao da Query Select.

5.5.2.1 Insira um componente do tipo FireDAC chamado TFDQuery.

Connection	ConexaoDB
Name	FDQueryIncluir
SQL	insert into cliente (id, nome, endereco, cidade, estado) values (gen_id(id_cliente,1),:nome,:endereco,:cidade,:estado)

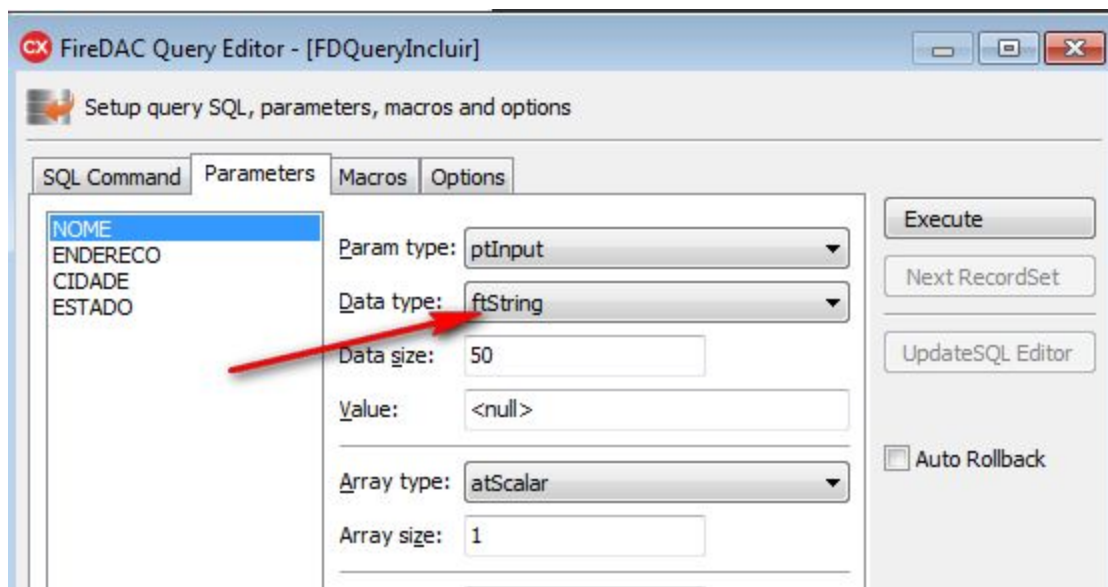
--	--

Note que existem parâmetros adicionais que você deve passar. Esses parâmetros foram usados para receberem os valores digitados pelos usuários. Nesse caso específico, são os seguintes parâmetros :

- :nome
- :endereco
- :cidade
- :estado

Observe bem que esses parâmetros sempre começam com dois pontos (:)

Toda vez que você for criar tais parâmetros o C++ Builder vai listá-los automaticamente na aba “Parameters” da janela “FireDAC Query Editor”, conforme a figura abaixo :



O processo de preenchimento dos parâmetros da Query é feito automaticamente pelo C++ Builder.

Dica : Preste atenção no tipo de dado do parâmetro. O C++ Builder geralmente o identifica como uma string (ftString), mas você pode mudar o tipo de dado caso seja necessário. Por isso é importante você checar essa aba com cuidado.

5.5.2.2. Precisa inserir também um Data Source, mas não muda absolutamente nada. Todos os Data Sources obedecem a mesma regra, mudando apenas o nome (sufixo)

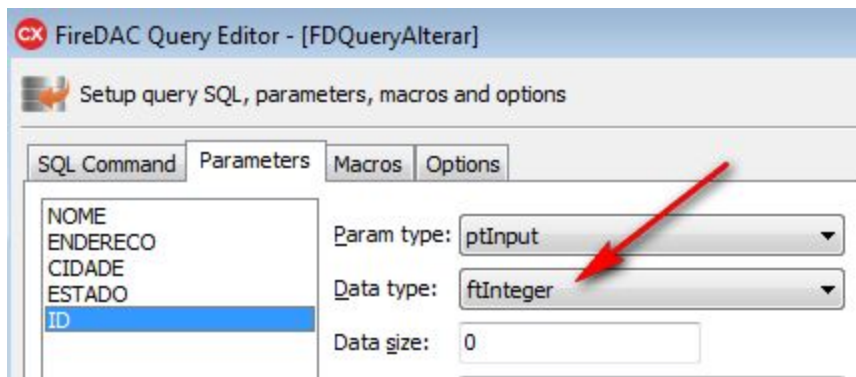
Name	DSIncluir
DataSet	FDQueryIncluir

5.5.3. Query UPDATE

5.5.3.1 Insira um componente do tipo FireDAC chamado TFDQuery.

Connection	ConexaoDB
Name	FDQueryAlterar
SQL	update cliente set nome=:nome,endereco=:endereco,cid ade=:cidade,estado=:estado where id=:id

Observe que a o parâmetro :id foi identificado como string, você deve trocar pelo tipo de dado correto, que é o integer, conforme a figura abaixo :



5.5.3.2. O Data Source

Name	DSAlterar
DataSet	FDQueryAlterar

5.5.4. Query DELETE

5.5.4.1 Insira um componente do tipo FireDAC chamado TFDQuery.

Connection	ConexaoDB
Name	FDQueryExcluir
SQL	delete from cliente where id=:id

5.5.4.2. O Data Source

Name	DSEExcluir
DataSet	FDQueryExcluir

6. Habilitar e desabilitar botões

Antes de programar o código que será ativado pelos eventos dos botões, iremos criar algumas funções-membros auxiliares. Essas funções só serão usadas posteriormente, agora vamos apenas criá-las.

6.1. Hab_Botoes : Essa função irá Habilitar os botões do formulário.

```
void __fastcall TFormCliente::Hab_Botoes()
{
    btIncluir->Enabled = true;
    btAlterar->Enabled = true;
    btExcluir->Enabled = true;
    btCancelar->Enabled = false;
    btGravar->Enabled = false;
    btSair->Enabled = true;
}
```

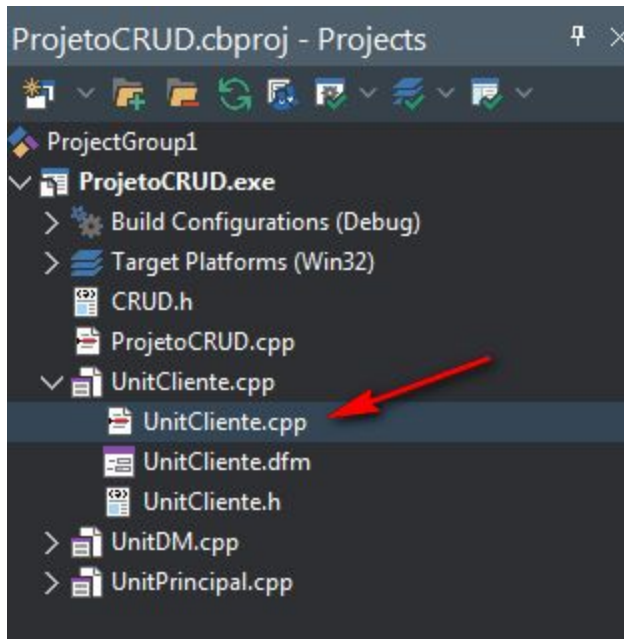
6.2. Desab_Botoes : Essa função irá desabilitar os botões do formulário

```
void __fastcall TFormCliente::Desab_Botoes()
{
    btIncluir->Enabled = false;
    btAlterar->Enabled = false;
    btExcluir->Enabled = false;
    btCancelar->Enabled = true;
    btGravar->Enabled = true;
    btSair->Enabled = false;
}
//-----
```

6.3. Habilitar_Campos: Agora iremos configurar a função que irá Habilitar/Desabilitar os campos do formulário para a edição.

```
void __fastcall TFormCliente::Habilitar_Campos( bool valor )
{
    EdtID->Enabled = valor;
    EdtNome->Enabled = valor;
    EdtEndereco->Enabled = valor;
    EdtCidade->Enabled = valor;
    CBEstado->Enabled = valor;
}
```

Dica : Para adicionar uma função-membro, você deve, primeiramente abrir o arquivo CPP caso ele não esteja aberto.



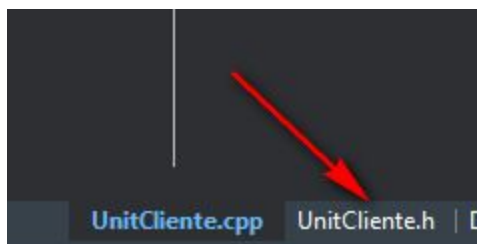
Para exibir o código, caso esteja no modo formulário, tecle F12.

Primeiro digite o corpo da função, sem o código.

```
void __fastcall TFormCliente::Hab_Botoes()  
{  
    //-----  
}
```

Depois, vá para o arquivo UnitCliente.h correspondente e informe o cabeçalho.

Para fazer isso **não use o painel Projects**. O arquivo UnitCliente.h já está aberto, basta selecionar o UnitCliente.h na aba inferior, conforme abaixo :



O cabeçalho é quase igual à função, menos o nome da classe, ou seja, omite o prefixo TFormCliente:

```

class TFormCliente : public TForm
{
__published:      // IDE-managed Components
    TPanel *Panel1;
    TBitBtn *btIncluir;
    TBitBtn *btAlterar;
    TBitBtn *btExcluir;
    TBitBtn *btCancelar;
    TBitBtn *btGravar;
    TBitBtn *btSair;
    TPageControl *Pagina1;
    TTabSheet *TabArquivo;
    TTabSheet *TabDados;
    TDBGrid *DBGrid1;
    TLabel *Label1;
    TLabel *Label2;
    TLabel *Label3;
    TLabel *Label4;
    TLabel *Label5;
    TEdit *EdtID;
    TEdit *EdtNome;
    TEdit *EdtEndereco;
    TEdit *EdtCidade;
    TComboBox *CBEstado;
    void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
    void __fastcall btSairClick(TObject *Sender);
    void __fastcall Hab_Botoes();
private:          // User declarations
public:           // User declarations
    __fastcall TFormCliente(TComponent* Owner);
};

```

Pronto, feito isso você pode voltar lá e adicionar o código:

```

void __fastcall TFormCliente::Hab_Botoes()
{
    btIncluir->Enabled = true;
    btAlterar->Enabled = true;
    btExcluir->Enabled = true;
    btCancelar->Enabled = false;
    btGravar->Enabled = false;
    btSair->Enabled = true;
}

```

7. Preparação do formulário

7.1. Como iremos, nos próximos tópicos, gravar dados no formulário. Essa é a nossa primeira operação que fará uma gravação dos dados no banco de dados. Como é a primeira, você deve fazer uma referência ao Data Module de dentro do FormCliente. Faça isso colocando um include, conforme abaixo :

```
//-----
#include <vcl.h>
#pragma hdrstop

#include "UnitCliente.h"
#include "UnitDM.h"

//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TFormCliente *FormCliente;

//-----
__fastcall TFormCliente::TFormCliente(TComponent* Owner)
: TForm(Owner)
```

7.2. Iremos criar uma variável global chamada situacao para poder diferenciar uma operação da outra. Ela receberá o valor inicial -1, mas os valores válidos serão apenas dois.

situacao igual a 1 para Inclusão

situacao igual a 2 para Alteração

situacao igual a qualquer outro valor não fará nada.

```
UnitCliente.cpp
//-----
#include <vcl.h>
#pragma hdrstop

#include "UnitCliente.h"
#include "UnitDM.h"

//-----
#pragma package(smart_init)
10 #pragma resource "*.dfm"
11 int situacao=-1;

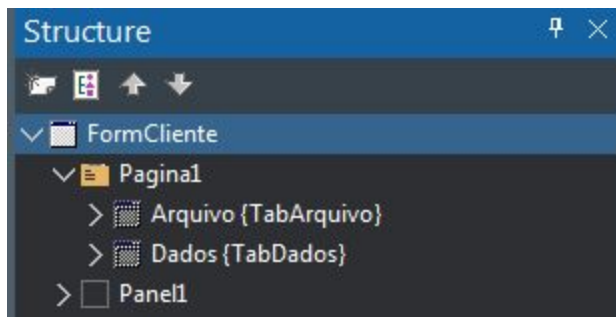
TFormCliente *FormCliente;

//-----
__fastcall TFormCliente::TFormCliente(TComponent* Owner)
```

7.3. Ao abrir o formulário, devemos ir para a aba "Arquivos"

7.3.1. Clique duas vezes no evento "OnCreate" do FormCliente

Dica : Para ter certeza de que é o formulário, use o painel "Structure" para selecionar o mesmo :



7.3.2. Após clicar duas vezes, você será direcionado para a função-membro criada automaticamente.

7.3.3. Repita o procedimento para criar, também, o evento OnActivate para garantir o estado inicial dos botões

Veja os dois código abaixo :

```
void __fastcall TFormCliente::FormCreate(TObject *Sender)
{
    Pagina1->ActivePage = (TabArquivo);
}

//-----

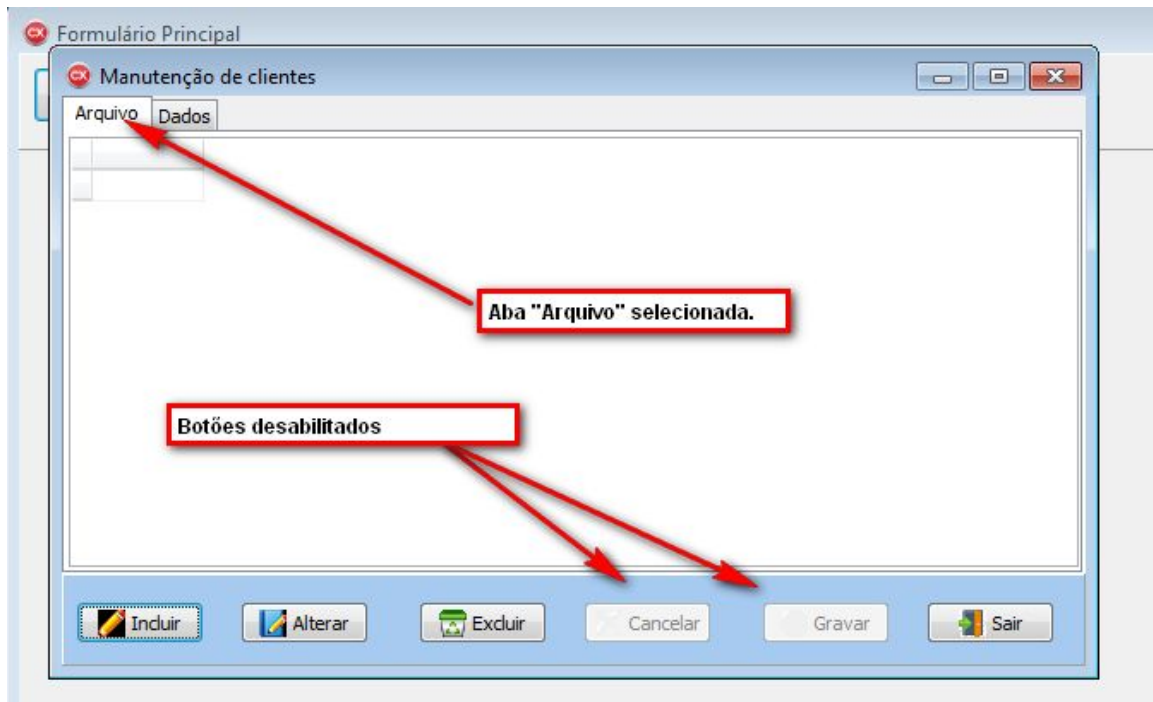
void __fastcall TFormCliente::FormActivate(TObject *Sender)
{
    Hab_Botoes();
}
```

7.3.4. Finalmente, o código para “limpar” os valores do formulário

```
void __fastcall TFormCliente::Limpar_Campos()
{
    EdtID->Text = "";
    EdtNome->Text = "";
    EdtEndereco->Text = "";
    EdtCidade->Text = "";
    CBEstado->ItemIndex = 0;
}
```

Não esqueça de incluir o cabeçalho em UnitCliente.h

Até agora nosso formulário está assim :



8. Inclusão

8.1. O código do botão de incluir :

```
void __fastcall TFormCliente::btIncluirClick(TObject *Sender)
{
    TabArquivo->TabVisible = false;
    TabDados->TabVisible = true;
    Limpar_Campos();
    Pagina1->ActivePage = (TabDados);
    situacao=1;
    Habilitar_Campos(true);
    Desab_Botoes();
    EdtNome->SetFocus();
}
//-----
```

Comentários :

- Esconde a aba de arquivos (o grid)
- Exibe a aba de dados
- Limpa os campos do formulário
- Torna a aba de dados ativa

- e. Selecciona o modo inclusão (situacao=1)
- f. Habilita os campos do formulário
- g. Desabilita os botões relacionados ao grid
- h. Coloca o foco no controle "Nome", para facilitar a edição

8.2. Gravação dos dados

A gravação dos dados é feita através do evento associado ao botão "Gravar".

Esse botão tem uma **dupla função**, tudo vai depender do valor da variável situacao, como já vimos.

```
void __fastcall TFormCliente::btGravarClick(TObject *Sender)
{
    if (Application->MessageBox(L"Confirma a operação ?",
        L"Gravação de registro",
        MB_YESNO + MB_ICONINFORMATION) == IDYES)
    {
        switch ( situacao ) {
        case 1:
            DM->FDQueryIncluir->Close();
            DM->FDQueryIncluir->ParamByName("NOME")->AsString = EdtNome->Text;
            DM->FDQueryIncluir->ParamByName("ENDERECO")->AsString = EdtEndereco->Text;
            DM->FDQueryIncluir->ParamByName("CIDADE")->AsString = EdtCidade->Text;
            DM->FDQueryIncluir->ParamByName("ESTADO")->AsString = CBEstado->Text;
            DM->FDQueryIncluir->ExecSQL();
        default:
            ;
        }

        TabDados->TabVisible=true;
        TabArquivo->TabVisible=true;
        Pagina1->ActivePage=(TabArquivo);
        TabArquivo->SetFocus();
        Hab_Botoes();
        Habilitar_Campos(false);
        situacao=-1;
    }
}
```

O código até agora só contempla a inclusão (situacao == 1)

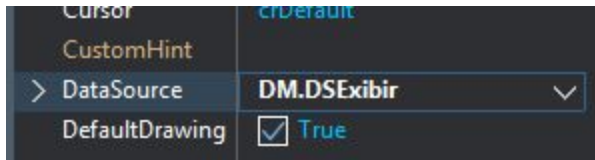
9. Formatação do grid

9.1. Crie o evento OnShow da aba TabArquivo

```
void __fastcall TFormCliente::TabArquivoShow(TObject *Sender)
{
    DM->FDQueryExibir->Close();
    DM->FDQueryExibir->Open();
}
//-----
```

Assim, sempre que ela for exibida o Grid será atualizado

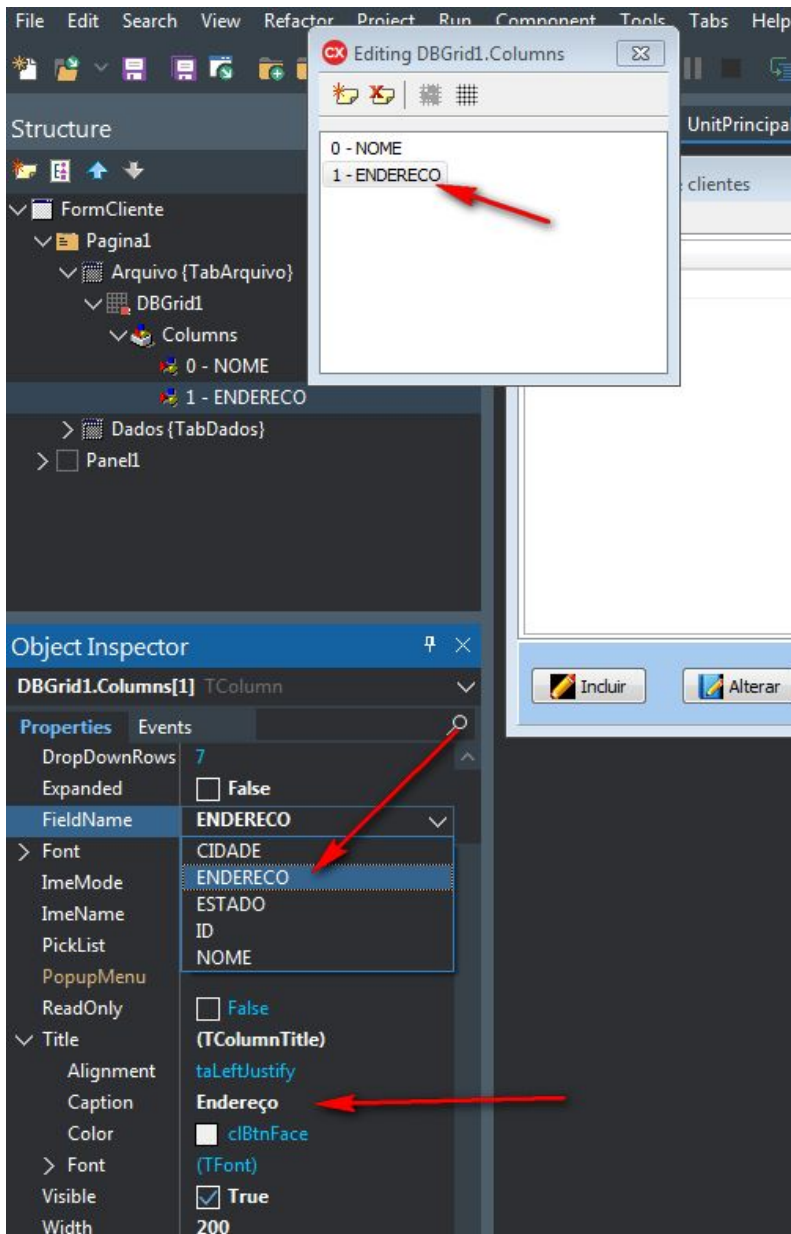
9.2. Em seguida selecione o Datasource do grid com o Datamodule DMExibir.



Compile o programa para testar.

9.3. Agora vamos configurar as colunas:

Selecione o DBGrid1 e acesse a propriedade columns. Clique sobre os 3 pontos para chamar o editor de colunas



Adicione as colunas que desejar.

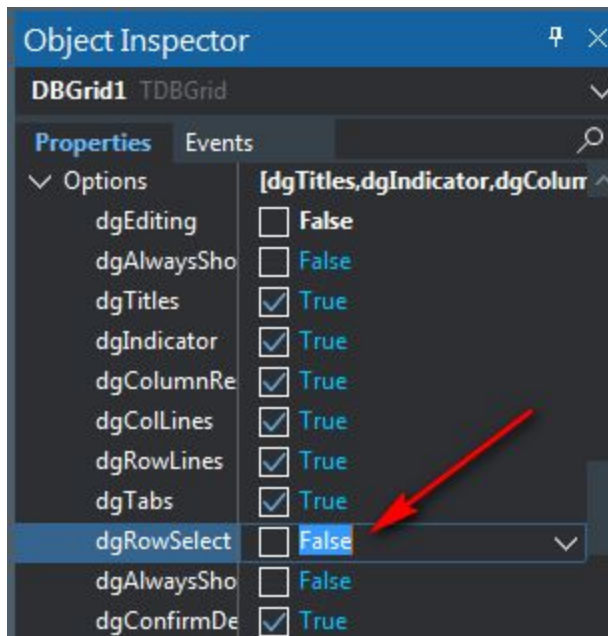
As propriedades que devem ser editadas para cada coluna são:

- a. Fieldname
- b. Title >> Caption (Título da coluna)
- c. Width (Largura da coluna)

Dica : Para facilitar a definição da largura da coluna, você pode selecionar o Grid e posicionar o ponteiro do mouse entre um cabeçalho de coluna e outro, dessa forma o ponteiro muda o formato e você pode arrastar para a direita ou esquerda.



9.4. Bloqueie as células do DBGrid. Isso evita o indesejável “efeito planilha”, que é as células do grid “editáveis”. Faça isso na propriedade “Options”, desmarcando o ítem dgRowSelect



10. Trazendo dados do banco para o formulário

10.1 Criar a função BancoParaInterface

10.2. Adicione um evento ao botão “Alterar”, esse evento será muito semelhante ao evento relacionado ao botão “Incluir”, as diferenças são :

- a. situacao = 2
- b. Não “limpa” os campos do formulário
- c. Ao final, acrescento a função BancoParaInterface

```
void __fastcall TFormCliente::btAlterarClick(TObject *Sender)
{
    TabArquivo->TabVisible = false;
    TabDados->TabVisible = true;
    Pagina1->ActivePage = (TabDados);
    situacao=2;
    Habilitar_Campos(true);
    Desab_Botoes();
    EdtNome->SetFocus();
    BancoParaInterface();
}
```

11. Gravação da alteração

Acrescente na função-membro btGravarClick o código da alteração

```
case 2:
    DM->FDQueryAlterar->Close();
    DM->FDQueryAlterar->ParamByName("NOME")->AsString = EditNome->Text;
    DM->FDQueryAlterar->ParamByName("ENDERECO")->AsString = EditEndereco->Text;
    DM->FDQueryAlterar->ParamByName("CIDADE")->AsString = EditCidade->Text;
    DM->FDQueryAlterar->ParamByName("ESTADO")->AsString = CBEstado->Text;
    DM->FDQueryAlterar->ParamByName("ID")->AsInteger = EditID->Text.ToInt();
    DM->FDQueryAlterar->ExecSQL();
    break;
```

12. Cancelamento

Crie uma função-membro associada ao evento Cancelar

```
void __fastcall TFormCliente::btCancelarClick(TObject *Sender)
{
    TabArquivo->TabVisible = true;
    Pagina1->ActivePage = (TabArquivo);
    TabArquivo->SetFocus();
    Hab_Botoes();
    Habilitar_Campos(false);
}
```

13. Exclusão

```

void __fastcall TFormCliente::btExcluirClick(TObject *Sender)
{
    if (Application->MessageBox(L"Confirma a exclusão ?",
    L"Exclusão de registro",
    MB_YESNO + MB_ICONINFORMATION )==IDYES)
    {
        DM->FDQueryExcluir->Close();
        DM->FDQueryExcluir->ParamByName("ID")->AsInteger = DM->FDQueryExibir->FieldByName("ID")->AsInteger;
        DM->FDQueryExcluir->ExecSQL();

        DM->FDQueryExibir->Close();
        DM->FDQueryExibir->Open();
    }

    TabDados->TabVisible = true;
    TabArquivo->TabVisible = true;

    Paginal->ActivePage = (TabArquivo);
}

```

Ambiente em que foi desenvolvido a lista

1. Windows 7
2. C++ Builder 10.3.3

Fonte

Udemy