



Desenvolvimento Mobile Rápido com Firemonkey

#12

Vander Batista

Da sala de reunião à vida real

Embarcadero Conference 2019

Agradecimentos



DEUS
Minha Família
Diretoria
Superintendência
Colaboradores
Equipe de TI



Desenvolvimento Mobile

Para Desenvolvimento Mobile com Firemonkey, seguimos esta linha:

- Escolha do Layout para Desenvolvimento
- Desenhos e Imagens
- Codificação

DESENVOLVIMENTO MOBILE RÁPIDO

Escolha do Layout da Aplicação

Com base nas escolhas você tem a possibilidade de desenhar, como se desenha em Desktop.

Nós abusamos dos recursos de desenho, com TLayout, TRectangle, TImage e TScrollView.

Estes basicamente, são os componentes à serem utilizados para transformar as aplicações.

Existem outros que no decorrer do desenvolvimento, você vai aplicando e dando robustez ao App, além de melhoria no visual.

O que a RAD Studio nos proporciona

Quem acompanha o Delphi desde as primeiras versões, percebe a evolução da ferramenta e agora com a Embarcadero não é diferente, aliás evolui muito mais rápido.

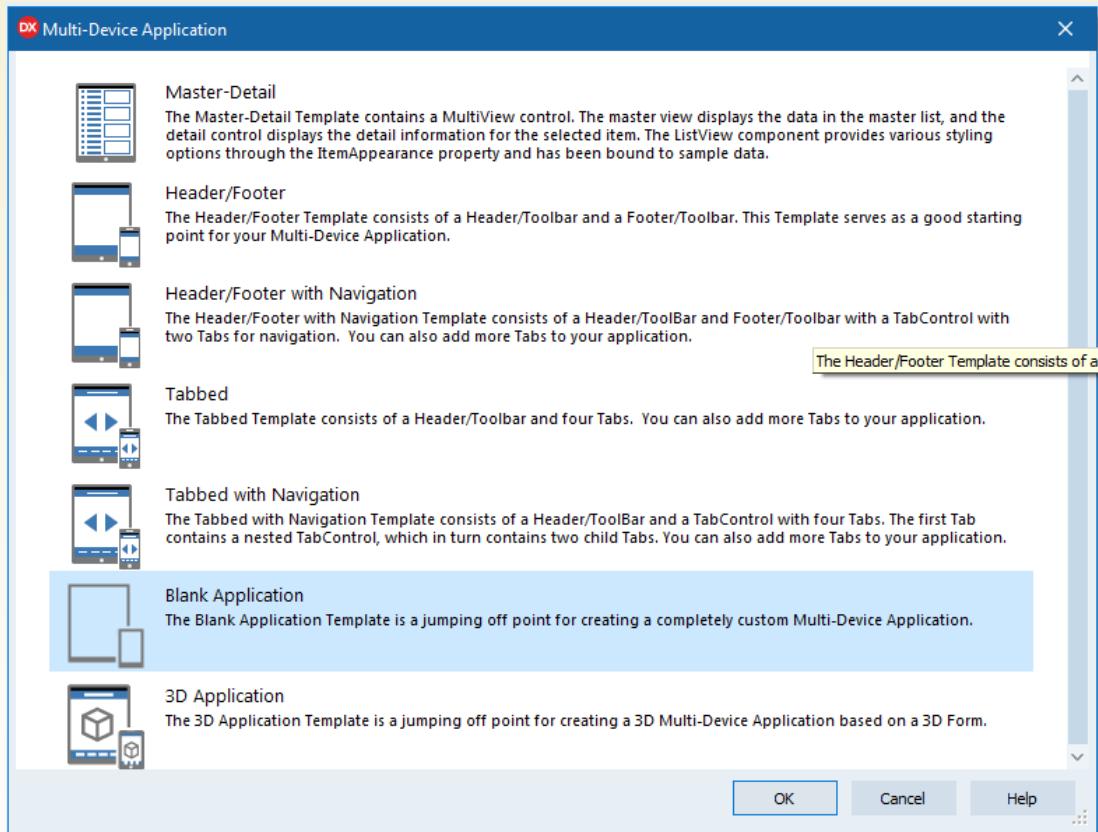
Com a ascendência da plataforma e do mercado de dispositivos móveis, a RAD Studio nos permite a possibilidade de desenvolver Aplicativos robustos, desde as idéias mais simples até as mais avançadas.

Diante deste cenário a pergunta é:



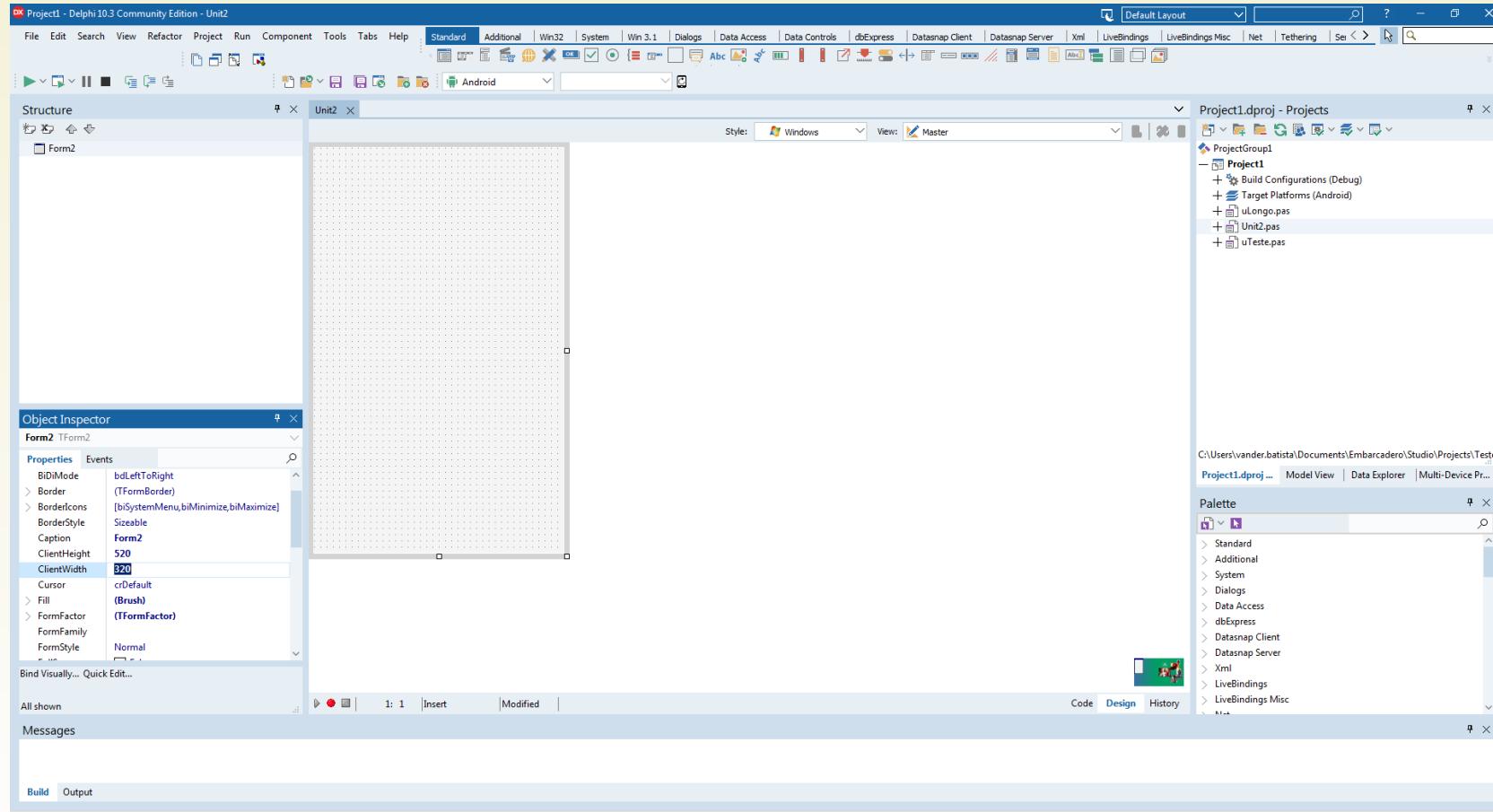
Construção de Layouts Rápidos

Escolha do Layout da Aplicação



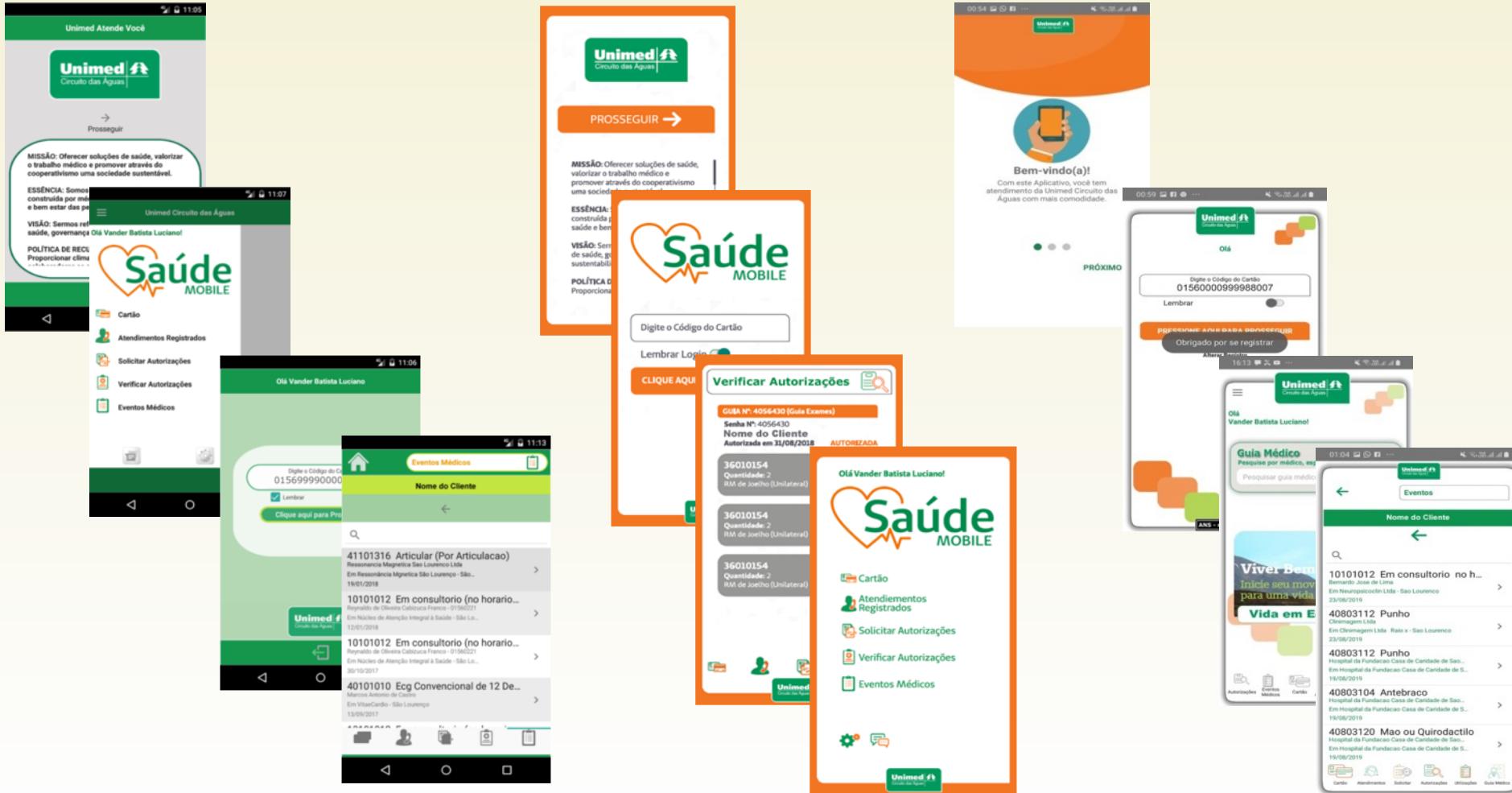
Construção de Layouts Rápidos

Escolha do Layout da Aplicação



Alteração e Manutenção dos Layouts

Evolução de layouts do nosso App



Utilização dos Objetos de Desenho



Basicamente utilizamos na nossa empreitada:

TLayout
TGridPanelLayout
TVertScrollBox
TRectangle
TImage
 TLabel
TGlowEffect
TListView

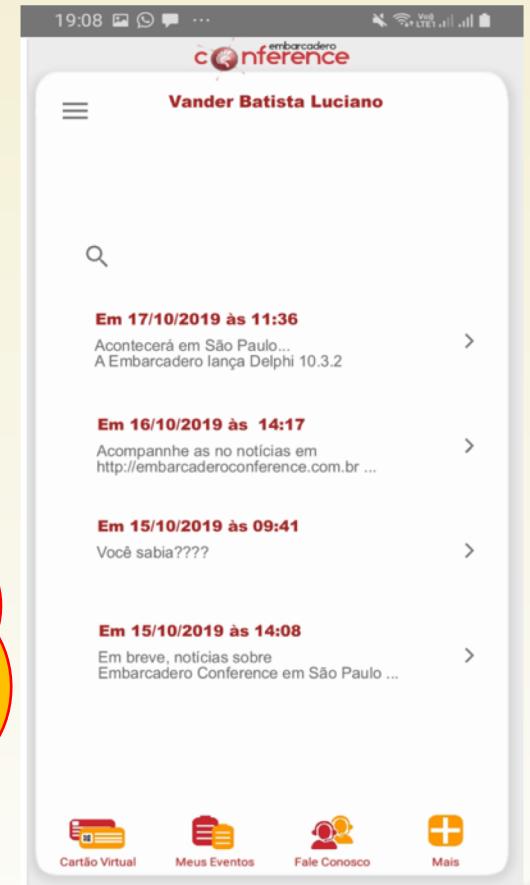
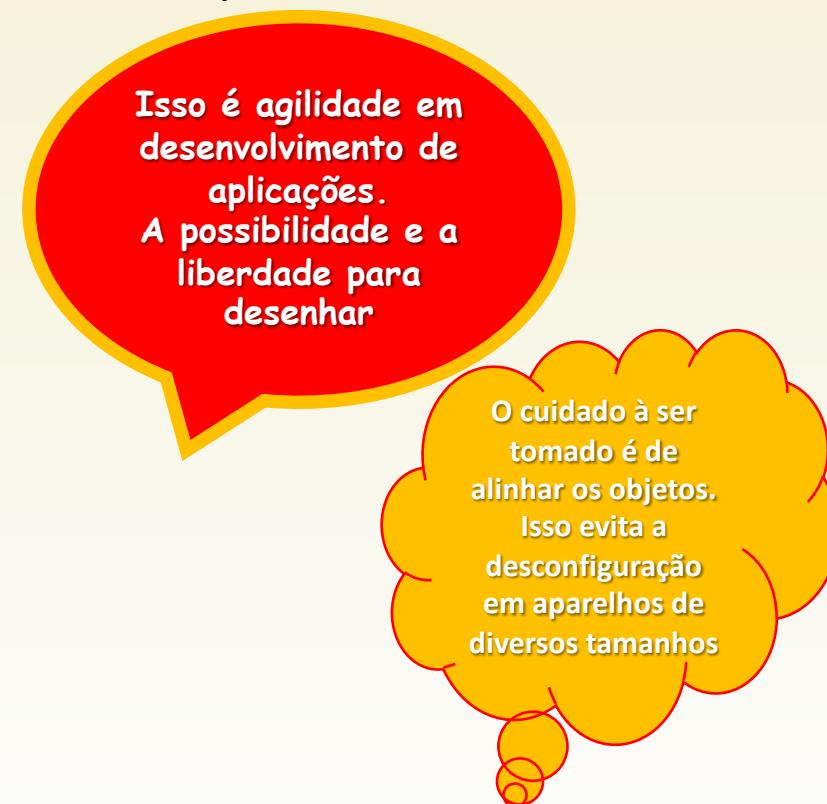
Utilização dos Objetos de Desenho



Um TImage inserido em um TLayout

Um TRectangle

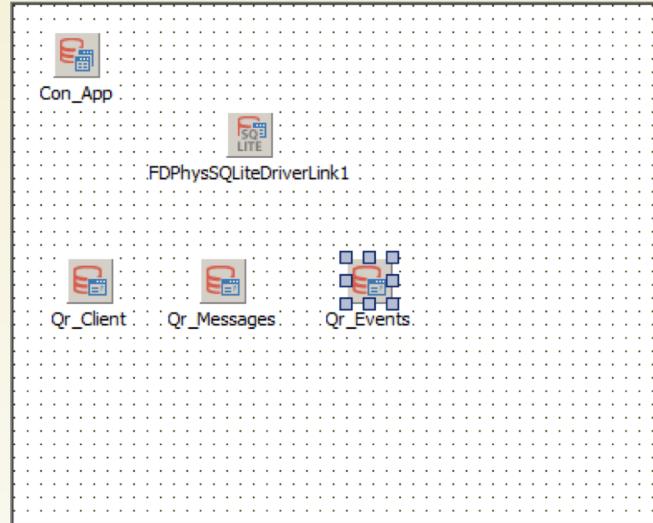
Um TGridLayout com TImage e TLabel



TLabel no Título e nos Rodapés dos Botões

Desenvolvimento do Aplicativo

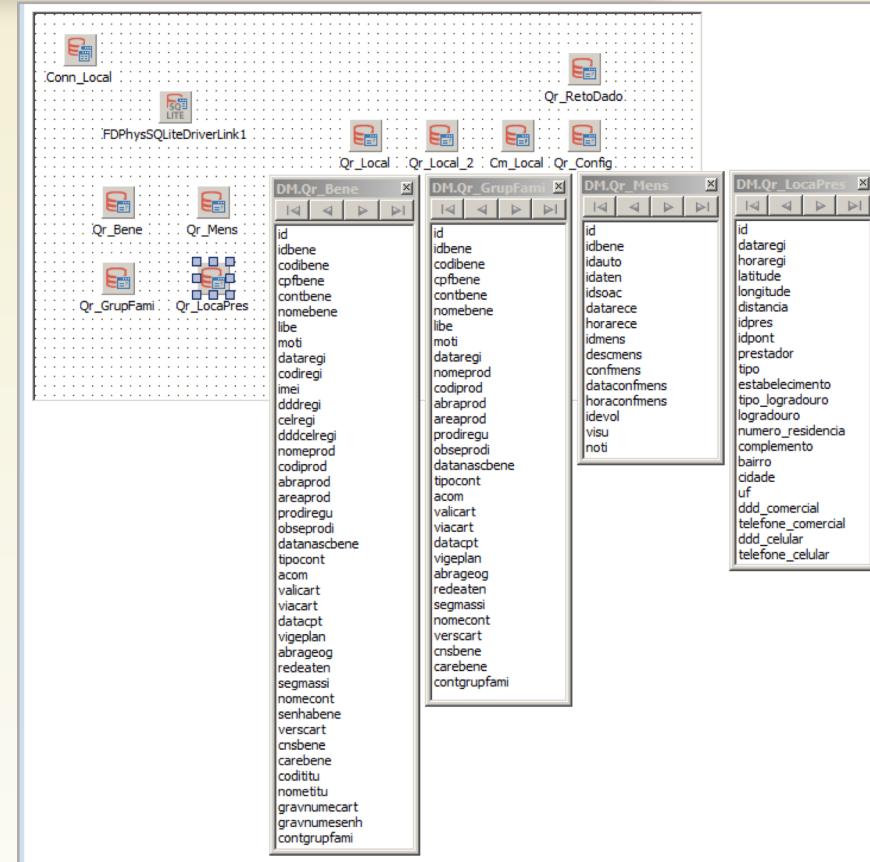
Data Module conectado com SQLite



```
procedure TDM.Conn_LocalBeforeConnect(Sender: TObject);
begin
  {$IF DEFINED(iOS) or DEFINED(ANDROID)}
  Conn_Local.DriverName := 'SQLite';
  Conn_Local.ConnectionDefName := '';
  Conn_Local.Params.Values['Database'] := TPath.Combine(TPath.GetDocumentsPath, 'app.db');
  {$ENDIF}
end;

procedure TDM.DataModuleCreate(Sender: TObject);
begin
  Conn_Local.Connected := True;
end;

procedure TDM.DataModuleDestroy(Sender: TObject);
begin
  Conn_Local.Connected := False;
end.
```



Carrego as informações dos campos nos objetos, quando inicializa o App e utilizo estas informações no fluxo dos processos do Sistema.
Por ser um banco local, isso ganha muito tempo.

Desenvolvimento do Aplicativo

Inicializando o App

```
procedure Tf_BoasVindas.FormActivate(Sender: TObject);
begin

  Rct_Fundo.Visible := False;

  DM.Qr_bene.Open;
  if DM.Qr_Bene.RecordCount > 0 then
  begin
    if (Trim(DM.Qr_BenegravnumecartAsString) = 'S') and (Trim(DM.Qr_BenegravnumesenhAsString) = 'S') then
    begin
      DM.Qr_GrupFami.Close;
      DM.Qr_GrupFami.Open;
      f_Principal.Lbl_Beneficiario.Text := 'Olá ' + DM.Qr_BenenamebeneAsString + '!';
      f_Principal.Show;
      Exit; 
    end
    else
    begin
      f_Login.Show;
      Exit; 
    end;
  end;

  if not f_PrimeiroAcesso then
  begin
    Rct_Fundo.Visible := True;
    f_Registro.Show;
    Exit; 
  end;

end;
```

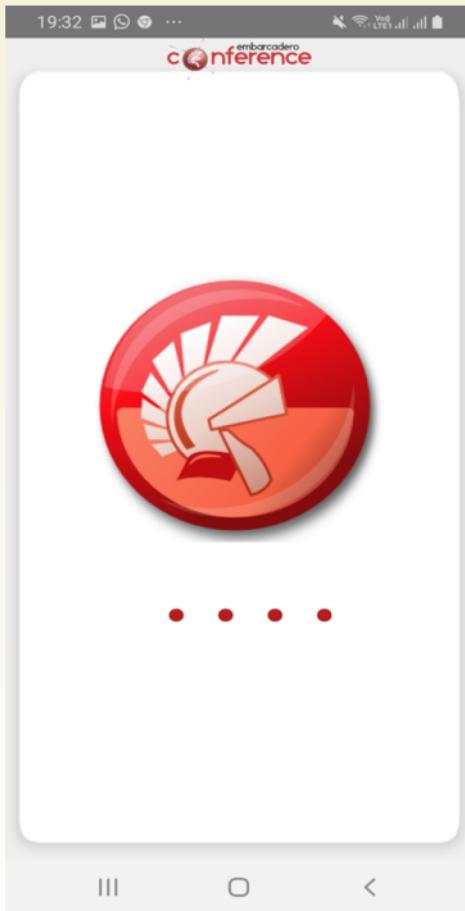
Abro a tabela de registro.
Caso encontre, testo se é
para fazer login ou não,
de acordo com opção pré
marcada pelo usuário.

Se for o primeiro acesso,
abre-se a tela de registro.

“Exit” é chamado, pois é necessário parar a execução do método, senão não funciona.

Desenvolvimento do Aplicativo

Navegando no App

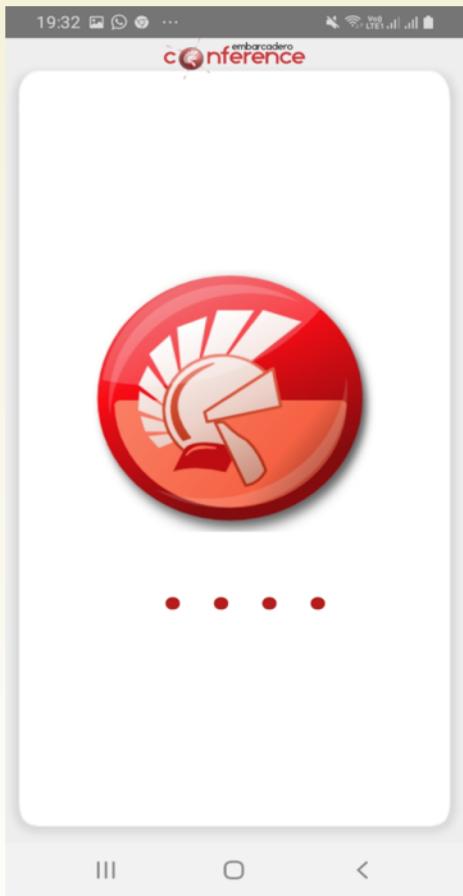


Um SplashScreen personalizado

- ✓ Utilizamos TLayout, TRectangle, TImage, TLabel e um TTimer.
 - ✓ TLayout, TRectangle, TImage, TLabel, são utilizados para desenhar a tela.
 - ✓ TTimer, para controlar o tempo de exposição do Splash.
 - ✓ Neste mesmo tempo, crio alguns formulários.

Desenvolvimento do Aplicativo

Navegando no App



```
procedure Tf_Splash.FormShow(Sender: TObject);
begin

  rct_Background.Next.Visible := False;
  lbl_Wait.Visible := True;

  ProgressBar1.Value := 0;
  ProgressBar1.Max := 30;

  vProcessLoadind := TThread.CreateAnonymousThread(procedure
    var
      I: Integer;
    begin
      lbl_Wait.Text := 'Aguarde ';
      for I := 1 to 30 do
        begin
          lbl_Wait.Text := lbl_Wait.Text + '.';
          ProgressBar1.Value := ProgressBar1.Value + I;
          Sleep(1000);
        end;
      TThread.Synchronize
      (
        TThread.currentThread,
        procedure
        begin
        end
      );
    end;
  vProcessLoadind.FreeOnTerminate := True;
  vProcessLoadind.Start;

end;
```

```
procedure Tf_Splash.tmrCreateFormsTimer(Sender: TObject);
begin

  tmrCreateForms.Enabled := False;

  if not vProcessForm then
  begin

    if not Assigned(DM) then
      Application.CreateForm(TDM, DM);
    if not Assigned(f_Default) then
      Application.CreateForm(Tf_Default, f_Default);
    if not Assigned(f_Default_Register) then
      Application.CreateForm(Tf_Default_Register, f_Default_Register);
    if not Assigned(f_Main) then
      Application.CreateForm(Tf_Main, f_Main);
    if not Assigned(f_VirtualCard) then
      Application.CreateForm(Tf_VirtualCard, f_VirtualCard);
    if not Assigned(f_Login) then
      Application.CreateForm(Tf_Login, f_Login);
    if not Assigned(f_Register) then
      Application.CreateForm(Tf_Register, f_Register);
    if not Assigned(f_ContactUS) then
      Application.CreateForm(Tf_ContactUS, f_ContactUS);

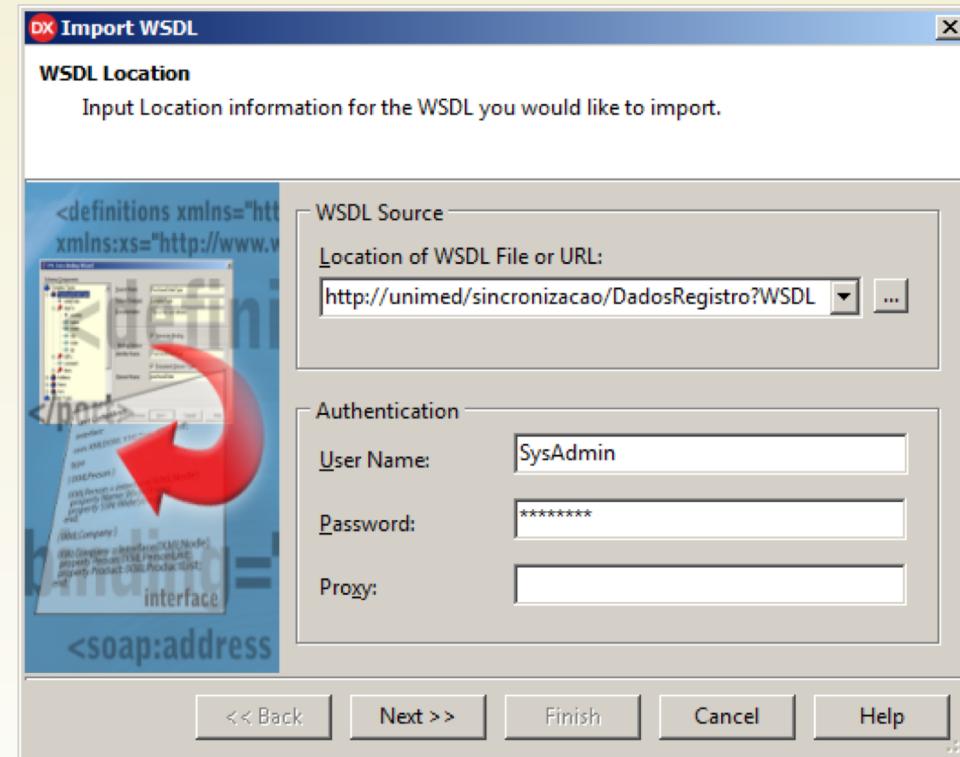
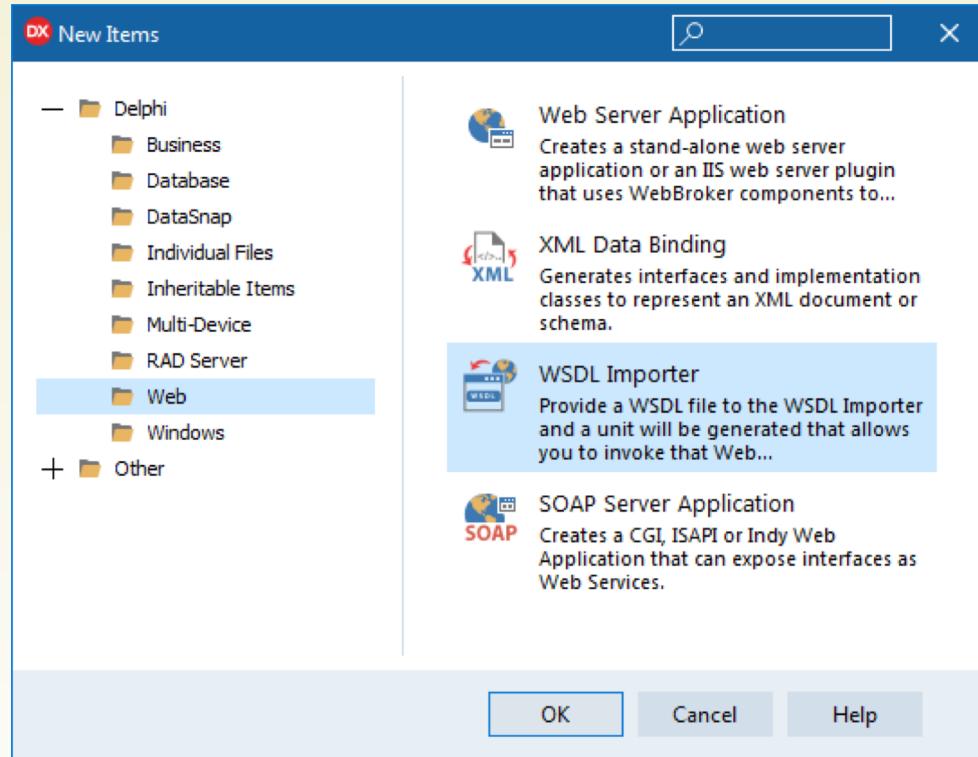
    vProcessForm := True;

    DM.TB_Client.Open;
    if DM.TB_Client.RecordCount > 0 then
    begin
      vProcessLoadind.Terminate;
      f_Main.lbl_Name.Text := 'Olá ' + DM.TB_ClientnameAsString + '!';
      f_Main.Show;
    end
    else
    begin
      vProcessLoadind.Terminate;
      f_Register.Show;
    end;

    f_Main.Show;
    lbl_Wait.Visible := False;
  end;
end;
```

Desenvolvimento do Aplicativo

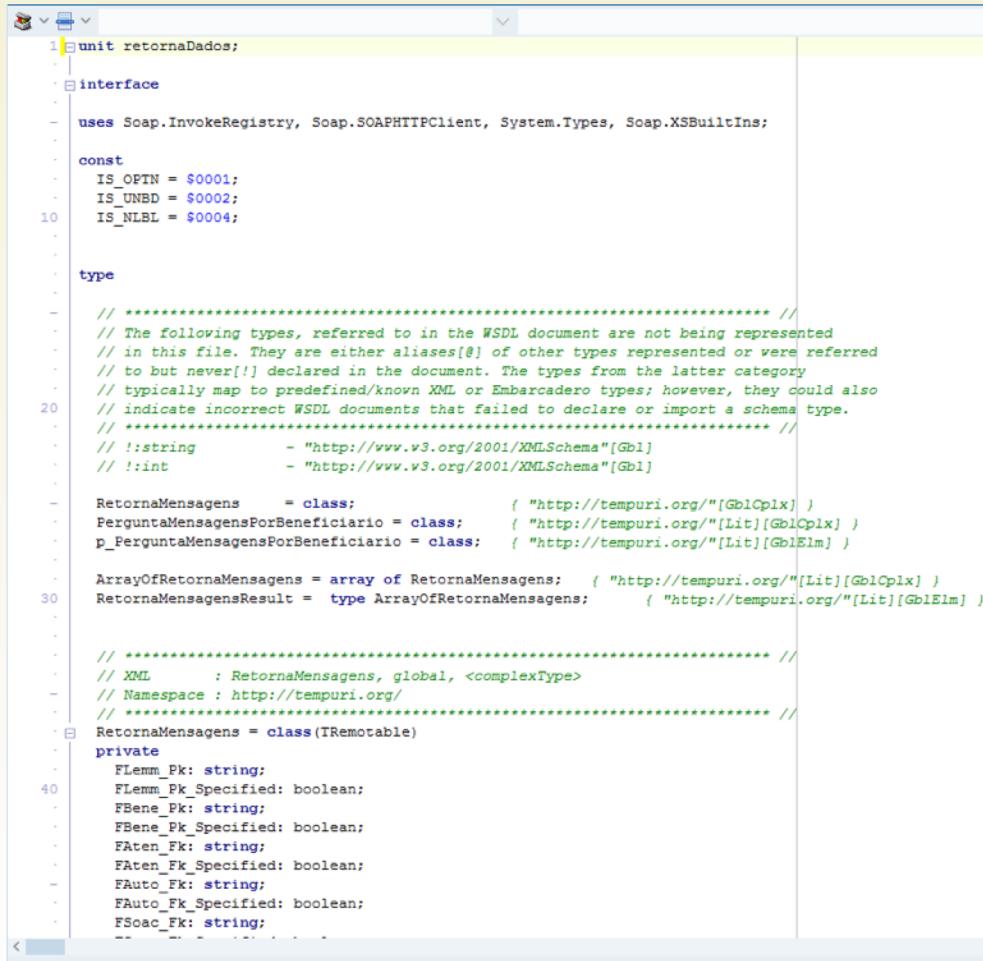
Consumindo Dados da Nuvem – Conectando o App aos WebServices



**Utilizamos WebServices em WSDL.
Delphi possui conexão via RestFull JSON.
Enfim, não ficará limitado em conexão remota.**

Desenvolvimento do Aplicativo

Consumindo Dados da Nuvem – Conectando o App aos WebServices



The screenshot shows a code editor window with Delphi code. The code is for a unit named 'retornaDados'. It includes imports for Soap.InvokeRegistry, Soap.SOAPHTTPClient, System.Types, and Soap.XSBuiltIns. There are constants for IS_OPTN, IS_UNBD, and IS_NLBL. A type section defines several XML schema types, including Gbl, GblCplx, and GblElm. It also defines classes for RetornaMensagens, PerguntaMensagensPorBeneficiario, and p_PerguntaMensagensPorBeneficiario. An array of RetornaMensagens is defined, along with its result type. The code uses annotations like `// XML : RetornaMensagens, global, <complexType>` and `// Namespace : http://tempuri.org/`. A private section contains declarations for various string fields.

```
unit retornaDados;
interface
uses Soap.InvokeRegistry, Soap.SOAPHTTPClient, System.Types, Soap.XSBuiltIns;
const
  IS_OPTN = $0001;
  IS_UNBD = $0002;
  IS_NLBL = $0004;
type
  // *****
  // The following types, referred to in the WSDL document are not being represented
  // in this file. They are either aliases[!] of other types represented or were referred
  // to but never[!] declared in the document. The types from the latter category
  // typically map to predefined/known XML or Embarcadero types; however, they could also
  // indicate incorrect WSDL documents that failed to declare or import a schema type.
  // *****
  // !string      - "http://www.w3.org/2001/XMLSchema"[Gbl]
  // !int         - "http://www.w3.org/2001/XMLSchema"[Gbl]
  RetornaMensagens    = class;           { "http://tempuri.org/[GblCplx] }
  PerguntaMensagensPorBeneficiario = class; { "http://tempuri.org/[Lit][GblCplx] }
  p_PerguntaMensagensPorBeneficiario = class; { "http://tempuri.org/[Lit][GblElm] }

  ArrayOfRetornaMensagens = array of RetornaMensagens; { "http://tempuri.org/[Lit][GblCplx] }
  RetornaMensagensResult = type ArrayOfRetornaMensagens; { "http://tempuri.org/[Lit][GblElm] }

  // *****
  // XML : RetornaMensagens, global, <complexType>
  // Namespace : http://tempuri.org/
  // *****
  RetornaMensagens = class(TRemutable)
private
  FLeimm_Pk: string;
  FLeimm_Pk_Specified: boolean;
  FBene_Pk: string;
  FBene_Pk_Specified: boolean;
  FAten_Fk: string;
  FAten_Fk_Specified: boolean;
  FAuto_Fk: string;
  FAuto_Fk_Specified: boolean;
  FSoac_Fk: string;
  FSoac_Fk_Specified: boolean;
end;
```

- ✓ É criada a classe automaticamente com as propriedades e os métodos
- ✓ Utilização do componente THttpRIO
- ✓ Invocação do método SOAP da Classe para conectar ao WebService



Obrigado



vander.batista@unimedcircuito.coop.br



/vanderbatistaluciano

#12

Da sala de reunião à vida real

Embarcadero Conference 2019