

Nesta seção você encontra artigos intermediários sobre Delphi Win32 e Delphi .NET

PocketPC com Delphi

Desenvolvendo aplicativos móveis – Parte 2



Manoel Campos da Silva Filho

(manoelcampos@gmail.com)

é Analista de Sistemas, Tecnólogo em Processamento de Dados pela Universidade do Tocantins, especializando em Gestão e Consultoria em Telecomunicações, Professor da Escola Técnica Federal de Palmas-TO. Trabalha com desenvolvimento em Delphi, PHP e ASP.NET, programação para Linux em C. Desenvolve em Delphi desde a versão 3.0

Continuando o artigo da edição anterior, veremos agora como criar a aplicação cliente no Pocket PC para consumir o Web Service criado na última edição.

Criando a aplicação

O CF Builder inclui algumas opções no *Repository* do BDS para criação de aplicações para dispositivos móveis. Acesse o menu *File>New>Other*, na opção *Smart Device* escolha *Smart Device Application* para criar um novo projeto (**Figura 1**). Salve a unit principal com o nome “MainForm.pas” e o projeto como “PesquisaOpiniao.dpr”.

Para testarmos a aplicação no emulador acesse o menu *Compact Framework>Run Smart Device Project* no BDS, ou utilize o *Play* existente na barra de ferramentas adicionado pelo CF Builder (**Figura 2**).

Na barra você pode selecionar o emulador que deseja rodar a aplicação. O executável é copiado automaticamente para a

pasta compartilhada que você informou no campo *Shared Folder* nas configurações do emulador.

Assim, após rodar o emulador clicando no *Play* da barra do CF Builder, você deve acessar o *Storage Card*, como mostrado anteriormente, para executar a aplicação.

Lembre-se que se você não fizer um *Soft Reset* no emulador o *Storage Card* não aparecerá. Após o emulador abrir pela primeira vez, você não precisa fechá-lo, assim, o *Storage Card* não desaparece.

Para atualizar a aplicação no emulador, basta clicar no *Play* para que o executável seja copiado novamente para o *Storage Card*.

Acessando o Web Service na aplicação desktop

Para acessar o Web Service a partir da aplicação desktop, precisamos adicionar uma referência à URL do arquivo WSDL que descreve as funções disponíveis no Web Service.

Como vamos trabalhar com dois projetos ao mesmo tempo, e você já está com o projeto do Web Service aberto, para abrir o projeto da aplicação desktop clique com o botão direito em *Project Group 1*, no *Project Manager*, selecione *Add Existing Project* e localize o projeto *PesquisaOpiniao*.

Clique em *Save All* na barra de ferramentas para salvar o grupo de projetos. Salve com o nome de "Projetos Compact Framework". Agora temos os dois projetos no *Project Manager*.

Dê um duplo clique no *PesquisaOpiniao*, a aplicação desktop para Compact Framework. Clique com o botão direito e escolha a opção *Add Web Reference* para adicionar uma referência ao Web Service criado.

Na janela que é aberta você deve informar a URL do Web Service, que se você utilizou os nomes que sugeri, deve ser "http://localhost/PesquisaOpiniaoWS/PesquisaOpiniao.asmx". Porém, como a aplicação cliente não estará rodando

no seu PC e sim no dispositivo móvel, por meio do emulador, você não deve informar *localhost* e sim o endereço IP do seu PC (onde está o servidor web com o WebService hospedado) ou o nome DNS dele.

No meu caso, utilizei o endereço "http://developer/PesquisaOpiniaoWS/PesquisaOpiniao.asmx". Rode o projeto do Web Service que o navegador é aberto indicando a URL, porém, substitua *localhost* pelo IP ou nome DNS do seu PC.

Após informar a URL, pressione ENTER para abrir a página no navegador interno. Clique em *Service Description* para abrir o WSDL (o descritor do Web Service). O valor do campo *Web reference folder name* será usado como prefixo do namespace que será criado para acesso ao Web Service.

Altere o valor do campo para "PesquisaOpiniaoServer". Depois clique em *Add Reference* para adicionar a referência ao projeto. O BDS cria um arquivo PAS

mapeando as funções existentes no Web Service para serem usadas em nossa aplicação.

Agora inclua um *MainMenu* no formulário principal da aplicação desktop. Nele inclua um item "Arquivo" e dentro desse um item "Selecionar dados no Servidor". Para usarmos as funções do Web Service precisamos incluir a unit que foi criada após a referência ao Web Service ser adicionada ao projeto.

Assim, pressione ATL+F11 e selecione a unit *PesquisaOpiniaoServer.PesquisaOpiniao*. Inclua outros componentes no formulário para que esse fique semelhante à **Figura 3**.

Nota: Caso tente testar a aplicação e receba uma mensagem do tipo: *File not found: 'MainForm.frmMain.resources'*, você deve fazer uma cópia do arquivo RESX colocando o mesmo nome da mensagem, sendo esse erro problema no BDS ou *plugin*.

Nota: Não crie um formulário muito grande, pois dependendo do equipamento, o mesmo pode ficar desconfigurado.

Para armazenar os dados na aplicação, utilizaremos o *DataSet*. Como as funções do Web Service, que retornam dados para a aplicação, devolvem *DataSets*,

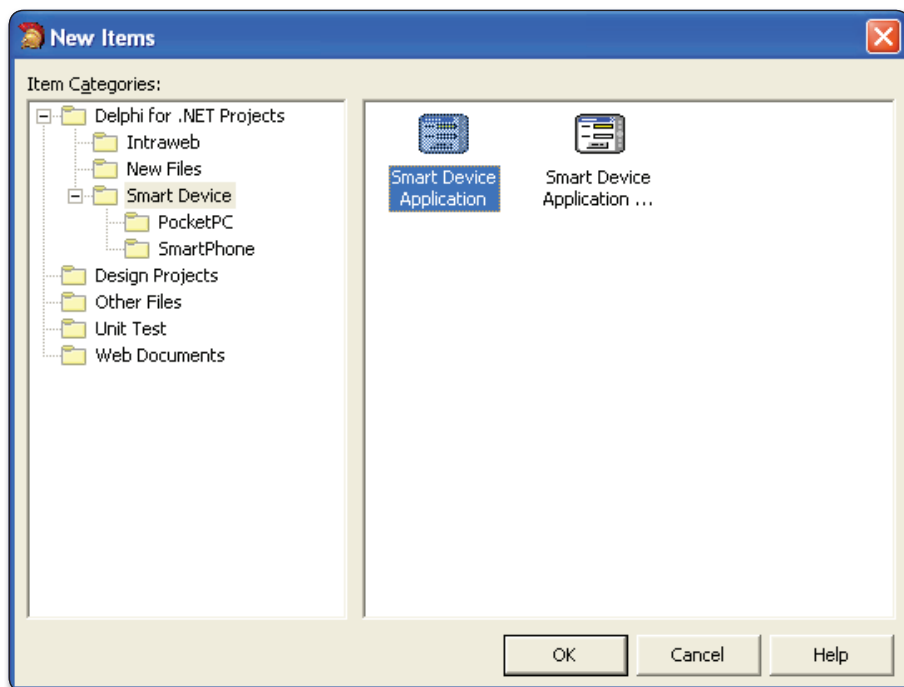


Figura 1. Opções Adicionadas pelo CF Builder no Repository

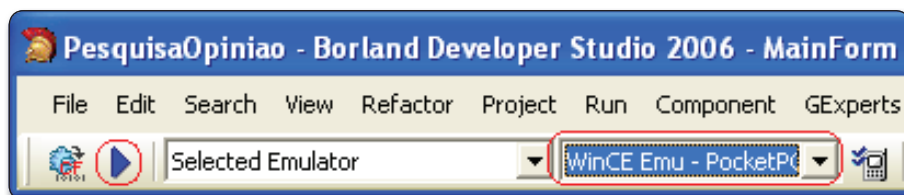


Figura 2. Barra de Ferramentas do CF Builder no BDS

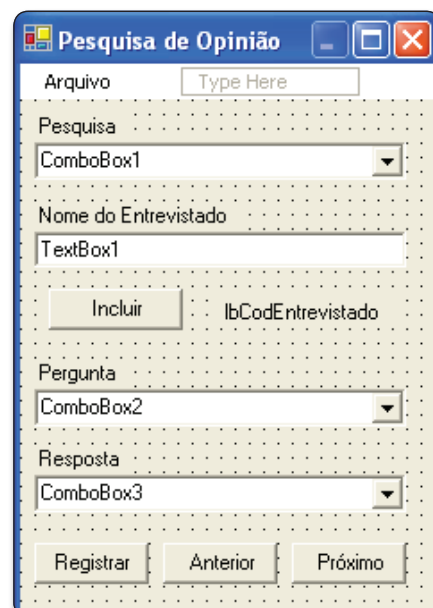


Figura 3. Design do Formulário

utilizaremos um *DataSet* para cada tabela que precisarmos armazenar dados localmente.

Assim, inclua cinco *DataSets* e altere seus nomes, respectivamente, para “ds-Pesquisa”, “dsPergunta”, “dsAlternativa”, “dsEntrevistado” e “dsResposta”. Na **Tabela 1** temos as propriedades que devem ser alteradas para alguns componentes.

Como precisaremos trabalhar com arquivos XML, criaremos uma unit que terá uma classe com funções estáticas (*class function*) que serão acessadas sem a necessidade de instanciar um objeto da classe.

Classes assim servem apenas para agrupar funções e constantes, organizando melhor o código. Crie a unit com o nome

de “ClubeDelphi.PesquisaOpinioXML.pas” e adicione o código da **Listagem 1**.

Nota: O código possui comentários sobre as características de cada função.

A função *XMLCache.ApplicationDir* retorna o caminho onde a aplicação está. Foi usada uma variável de classe *FAppDir* para armazenar, quando a função for executada pela primeira vez, o caminho do executável, para não ter que ficar executando este código repetidamente. Veja o código na **Listagem 2**.

A função *XMLCache.CarregarXML* (**Listagem 3**) verifica se o arquivo informado no parâmetro *NomeArq* existe para poder carregá-lo no *DataSet* passado por referência. Como o arquivo XML estará na pasta da aplicação, e estaremos usando o .NET Compact Framework, necessita que informe o caminho completo do arquivo que se deseja manipular.

Na função *XMLCache.IncluirEntrevistado* (**Listagem 4**), assim como nas outras funções de inclusão, como o registro dos entrevistados é feito localmente, o valor da chave primária é gerado localmente também, de acordo com o valor do último registro. Se o *DataSet* está vazio, o código para o registro a ser adicionado no *DataSet* será 1.

A função *XMLCache.EnviarCacheXML* (**Listagem 5**) envia os dados dos entrevistados, armazenados localmente no dispositivo em arquivo XML, e suas respostas ao Servidor, repassando os dados desses arquivos para as funções do Web Service.

Abra o formulário principal e pressione ALT+F11 para usar a unit recém criada. No evento *Click* do último item de menu criado, vamos solicitar ao usuário que digite o título da pesquisa (ou parte dele) para que seja feito o acesso ao Web Service e retornados os dados da pesquisa.

Como a aplicação estará rodando em um dispositivo móvel, nem sempre teremos conectividade para acessar o Web Service, pois, como é um sistema de pesquisa, o usuário poderá estar em qualquer parte da cidade, sem acesso à internet para fazer a comunicação com o servidor.

Assim, precisaremos carregar os dados necessários para o dispositivo e guardar

Listagem 1. Funções para manipular os arquivos XML da aplicação

```
unit ClubeDelphi.PesquisaOpinioXML;
interface
uses System.Data, System.Reflection, System.IO, System.Windows.Forms;

type
( Classe para agrupar as funções para manipulação dos DataSets )
XMLCache = class
private
( Variável de classe: possui um único valor para todas as instâncias da classe. Ela
é acessada sem a necessidade de instanciar um objeto da classe. Armazenará o caminho
da aplicação )
class var FAppDir: string;
public
( Constantes que definem o nome dos XML usados )
const arqPesquisa = 'pesquisa.xml';
const arqPergunta = 'pergunta.xml';
const arqAlternativa = 'alternativa.xml';
const arqEntrevistado = 'entrevistado.xml';
const arqResposta = 'resposta.xml';
( Descobre a pasta do executável )
class function ApplicationDir: string;
( Inclui um registro de entrevistado no DataSet
passado por parâmetro )
class function IncluirEntrevistado(
var ds: DataSet; Nome: string;
CodPesquisa: Integer): Integer;
( Registra, no DataSet passado por parâmetro, uma resposta de um entrevistado )
class function RegistrarResposta(
var ds: DataSet; CodEntrevistado, CodPergunta, CodAlternativa: Integer): Boolean;
( Carrega um XML, verificando se o mesmo existe, no DataSet passado por parâmetro )
class function CarregarXML(var ds: DataSet; NomeArq: string): Boolean;
( Grava os dados de um DataSet passado por parâmetro, em um arquivo XML )
class procedure GravarXML(ds: DataSet; NomeArq: string);
( Envia os dados dos entrevistados e seus respostas ao servidor )
class procedure EnviarCacheXML(var dsEntrevistado, dsResposta: DataSet);
end;
```

| Componente | Propriedade | Valor | Descrição |
|------------|---------------|-------------------|---|
| ComboBox | Name | cbxPesquisa | |
| | ValueMember | cod_pesquisa | Contém os dados a serem usados como valores para cada um dos itens do componente. A propriedade <i>DataSource</i> será atribuída via programação. |
| | DisplayMember | titulo | Contém os dados que serão exibidos no componente. |
| | DropDownStyle | DropDownList | Não permite que o usuário digite qualquer coisa no componente, só podendo escolher um dos itens existentes nele. |
| TextBox | Name | txtEntrevistado | |
| Label | Name | lbCodEntrevistado | |
| ComboBox | Name | cbxPergunta | |
| | ValueMember | cod_pergunta | |
| | DisplayMember | titulo | |
| | DropDownStyle | DropDownList | |
| ComboBox | Name | cbxResposta | |
| | ValueMember | cod_alternativa | |
| | DisplayMember | descricao | |
| | DropDownStyle | DropDownList | |

Tabela 1. Configuração dos componentes do formulário

localmente em arquivos XML, antes do usuário ir à rua para fazer seu trabalho de pesquisa. Quando o usuário estiver na rua, os dados das entrevistas que fizer devem ser armazenados localmente em arquivos XML, para quando ele chegar a um local que tenha acesso à internet ou à rede local da empresa, transferir os dados ao servidor. Para isso que servirão os *DataSets* adicionados.

Para solicitar ao usuário o título da pesquisa, quando clicar no menu, poderíamos utilizar uma função como *InputBox* do Delphi Win32 (que abre uma janela solicitando um dado do usuário), porém, não temos uma função semelhante em Delphi for .NET.

Dessa forma, criaremos uma classe que implementará um *InputBox*. Crie uma nova unit e salve com o nome de “ClubeDelphi.InputBox.pas”. O código dessa unit é mostrado no **Listagem 6**. Para usar o *InputBox* chame a função de classe *InputBox.Show* passando os devidos parâmetros (a função é chamada a partir da classe e não de uma instância).

A função do *InputBox.Show* é chamar o *InputBox* sem a necessidade de criar um objeto da classe. Para chamá-lo basta fazer *InputBox.Show* passando os parâmetros, sem criar objeto algum, com o código da **Listagem 7**.

Precisamos agora incluir o código para buscar os dados das perguntas e alternativas no servidor. No formulário principal, pressione ALT+F11 para usar a unit recém criada. No evento *Click* do item de menu adicione o código da **Listagem 8**.

Esse código chama as funções existentes no Web Service para retornar os dados que desejamos e em seguida estes dados são armazenados localmente em arquivos XML para a aplicação poder trabalhar *off-line*.

Cadastre algumas pesquisas, compile e execute a aplicação (usando os botões da barra de ferramentas do CF Builder) e clique no menu para ver o resultado. O emulador padrão é aberto.

Acesse o *Storage Card* para executar o programa. Lembre-se de usar o *Soft Reset* caso o emulador não estivesse aberto anteriormente.

Utilizando o Device Emulator

A aplicação, acessando o Web Service

Listagem 2. Retorna o caminho onde a aplicação se encontra

```
class function XMLCache.ApplicationDir: string;
var
  caminho: string;
begin
  if FAppDir = '' then
  begin
    { Pega o caminho completo do Assembly em execução (a aplicação) }
    caminho := Assembly.GetExecutingAssembly.GetName().CodeBase;
    { Extrai somente a pasta onde da aplicação, incluindo uma barra no final }
    Result := System.String.Format('{0}\', Path.GetDirectoryName(caminho), '\');
    { Se a função for chamada a partir de uma aplicação desktop que não seja para Compact
      Framework, o texto file:\ aparecerá no início da string. O if a seguir remove este texto }
    if Result.StartsWith('file:\') then
      Result := Result.Substring(6, result.Length-6);
    FAppDir := Result;
  end
  else Result := FAppDir;
end;
```

Listagem 3. Carrega um arquivo XML passado como parâmetro

```
class function XMLCache.CarregarXML(var ds: DataSet; NomeArq: string): Boolean;
begin
  Result := &File.Exists(ApplicationDir + NomeArq);
  { Se o arquivo existe, carrega-o no DataSet passado por parâmetro }
  if result then
    ds.ReadXml(ApplicationDir + NomeArq);
end;
class procedure XMLCache.GravarXML(ds: DataSet; NomeArq: string);
begin
  { Grava os dados do DataSet no XML, na pasta da aplicação, com o nome passado por parâmetro }
  ds.WriteXml(ApplicationDir + NomeArq);
end;
```

Listagem 4. Incluir um entrevistado

```
class function XMLCache.IncluirEntrevistado(
  var ds: DataSet; Nome: string; CodPesquisa: Integer): Integer;
var
  row: DataRow;
  total, cod: Integer;
begin
  total := ds.Tables[0].Rows.Count;
  if total = 0 then
    cod := 1
  else
    begin
      row := ds.Tables[0].Rows[total-1];
      { Gera um código local que será substituído no servidor quando os dados forem enviados }
      cod := Convert.ToInt32(row['cod_entrevistado'])+1;
    end;
  { Cria uma nova linha (um novo registro) }
  row := ds.Tables[0].NewRow;
  row['cod_entrevistado'] := cod.ToString;
  row['nome'] := Nome;
  row['cod_pesquisa'] := CodPesquisa.ToString;
  { Adiciona a linha no DataSet }
  ds.Tables[0].Rows.Add(row);
  Result := cod;
end;
class function XMLCache.RegistrarResposta(
  var ds: DataSet; CodEntrevistado, CodPergunta, CodAlternativa: Integer): Boolean;
var
  row: DataRow;
  total, cod: Integer;
begin
  total := ds.Tables[0].Rows.Count;
  if total = 0 then
    cod := 1
  else
    begin
      row := ds.Tables[0].Rows[total-1];
      { Gera um código local que será substituído no servidor }
      cod := Convert.ToInt32(row['cod_resposta_entrevistado'])+1;
    end;
  row := ds.Tables[0].NewRow;
  row['cod_resposta_entrevistado'] := cod.ToString;
  row['cod_entrevistado'] := CodEntrevistado.ToString;
  row['cod_pergunta'] := CodPergunta.ToString;
  row['cod_alternativa'] := CodAlternativa.ToString;
  ds.Tables[0].Rows.Add(row);
  Result := True;
end;
```



Listagem 5. Salva os dados em Cache (arquivo XML)

```
uses PesquisaOpiniaoServer.PesquisaOpiniao;
...
class procedure XMLCache.EnviaCacheXML(
  var dsEntrevistado, dsResposta: DataSet);
var
  row: DataRow;
  row2: DataRowView;
  ws: TPesquisaOpiniaoWS;
  i, j, CodEntrevistado: Integer;
begin
  { Instancia a classe do Webservice para acessar suas funções }
  ws := TPesquisaOpiniaoWS.Create;
  try
    { Percorre as linhas do DataSet de entrevistado para enviar
      essas linhas ao servidor para serem incluídas no banco }
    for i := dsEntrevistado.Tables[0].Rows.Count - 1 downto 0 do
    begin
      { Pega a linha atual }
      row := dsEntrevistado.Tables[0].Rows[i];
      { Passa os dados da linha atual do DataSet para a função
        IncluirEntrevistado do Webservice, para incluir estes dados
        no banco, retornando o código gerado pelo Webservice para o
        registro do entrevistado }
      CodEntrevistado := ws.IncluirEntrevistado(row['nome'].
        ToString, Convert.ToInt32(row['cod_pesquisa']));
      { Se retornou maior que zero é porque o registro foi incluído }
      if CodEntrevistado > 0 then
      begin
        { Filtra o DataSet de respostas retornando somente as respostas
          do entrevistado do entrevistado que foi adicionado ao banco
          (utilizando o código existente na linha atual (row) do
            DataSet dsEntrevistado) }
        dsResposta.Tables[0].DefaultView.RowFilter :=
          'cod_entrevistado=' + row['cod_entrevistado'].ToString;
        { Remove o registro do entrevistado do DataSet, pois esse já
          foi enviado ao servidor }
        dsEntrevistado.Tables[0].Rows.RemoveAt(i);
        { Percorre o DataSet dsResposta pegando todas as respostas
          do entrevistado atual }
        for j := dsResposta.Tables[0].DefaultView.Count-1 downto 0 do
        begin
          { Pega a linha atual do DataSet dsResposta }
          row2 := dsResposta.Tables[0].DefaultView.Item[j];
          { Chama a função RegistrarResposta do Webservice, para incluir
            no banco os dados do registro atual do DataSet dsResposta.
            Se a função retornar True, indica que a resposta foi incluída
            no banco(ou atualizada), assim, remove do DataSet, dsResposta o
            registro na posição atual(j) }
          if ws.RegistrarResposta(CodEntrevistado, Convert.
            ToInt32(row2['cod_pergunta']),
            Convert.ToInt32(row2['cod_alternativa'])) then
            dsResposta.Tables[0].DefaultView.Delete(j);
        end;
      end;
    end;
  finally
    { Grava as alterações nos DataSet's em arquivo }
    dsEntrevistado.WriteXml(arqEntrevistado);
    dsResposta.WriteXml(arqResposta);
  end;
end;
end.
```

Listagem 6. Unit ClubeDelphi.InputBox.pas

```
unit ClubeDelphi.InputBox;
interface
uses System.Drawing, System.Windows.Forms, System.ComponentModel;
type
  InputBox = class(System.Windows.Forms.Form)
  private
    TextBox1: System.Windows.Forms.TextBox;
    Label1: System.Windows.Forms.Label;
    btnOK: System.Windows.Forms.Button;
    btnCancel: System.Windows.Forms.Button;
  constructor Create();
  procedure TextBox1_KeyDown(sender: &Object;
    e: System.Windows.Forms.KeyEventArgs);
  procedure InitializeComponent();
  public
    class function Show(Title, Question: string): string;
  end;
implementation
constructor InputBox.Create();
begin
  inherited Create;
  InitializeComponent();
end;
procedure InputBox.TextBox1_KeyDown(sender: &Object;
  e: System.Windows.Forms.KeyEventArgs);
begin
  if e.KeyCode = Keys.Enter then
    Self.DialogResult := System.Windows.Forms.DialogResult.OK;
end;
procedure InputBox.InitializeComponent();
begin
  Self.Label1 := System.Windows.Forms.Label.Create;
  Self.TextBox1 := System.Windows.Forms.TextBox.Create;
  Self.btnOK := System.Windows.Forms.Button.Create;
  Self.btnCancel := System.Windows.Forms.Button.Create;
  { As propriedades e métodos que estão dentro da diretiva $IFDEF
    CF só serão compiladas se não estivermos compilando para o
    Compact Framework }
  {$IFDEF CF}
    Self.SuspendLayout();
  {$ENDIF}
  Self.TextBox1.Location := System.Drawing.Point.Create(16, 16);
  {$IFDEF CF}
    Self.TextBox1.Name := 'textBox1';
    Self.TextBox1.TabIndex := 0;
  {$ENDIF}

  Self.TextBox1.Size := System.Drawing.Size.Create(180, 20);
  Self.TextBox1.Text := '';
  Self.TextBox1.Top := 28;
  Self.TextBox1.Left := 10;
  Include(Self.TextBox1.KeyDown, Self.TextBox1_KeyDown);
  Self.Label1.Text := 'Digite o dado';
  {$IFDEF CF}
    Self.Label1.AutoSize := true;
    Self.Label1.TabIndex := 1;
  {$ELSE}
    Self.Label1.Size.Width := Self.TextBox1.Size.Width;
  {$ENDIF}
  Self.Label1.Top := 10;
  Self.Label1.Left := 10;
  Self.btnOK.Text := 'OK';
  Self.btnOK.DialogResult := System.Windows.Forms.DialogResult.OK;
  {$IFDEF CF}
    Self.btnOK.TabIndex := 2;
    Self.btnOK.NotifyDefault(true);
  {$ENDIF}
  Self.btnOK.Top := 55;
  Self.btnOK.Left := 10;
  Self.btnCancel.Text := 'Cancelar';
  {$IFDEF CF}
    Self.btnCancel.TabIndex := 3;
  {$ENDIF}
  Self.btnCancel.Top := 55;
  Self.btnCancel.Left := 100;
  Self.btnCancel.DialogResult :=
    System.Windows.Forms.DialogResult.Cancel;
  {$IFDEF CF}
    Self.AutoScaleBaseSize := System.Drawing.Size.Create(5, 13);
  {$ENDIF}
  Self.ClientSize := System.Drawing.Size.Create(220, 90);
  Self.ControlBox := false;
  Self.Controls.Add(Self.TextBox1);
  Self.Controls.Add(Self.Label1);
  Self.Controls.Add(Self.btnOK);
  Self.Controls.Add(Self.btnCancel);
  Self.FormBorderStyle :=
    System.Windows.Forms.FormBorderStyle.FixedDialog;
  {$IFDEF CF}
    Self.Name := 'InputBox';
    Self.ResumeLayout(false);
  {$ENDIF}
end;
```



a partir do WindowsCE Emulator não se comportou muito bem. Assim, você pode utilizar o Device Emulator para executar o programa. Como já mencionado, esse não pode ser configurado totalmente usando o CF Builder.

Assim, vamos usar os *links* que são criados na pasta *Microsoft Windows Mobile 5.0 Emulator* no menu *Iniciar* para executá-lo. Na pasta do menu *Iniciar*, indicada anteriormente, existem seis atalhos para imagens de dispositivos. Três terminam com a palavra *Coldboot* e outras três com a palavra *Savestate*.

Os *Coldboot* iniciarão o dispositivo dando um *boot* completo, o que vai demorar bastante. Os *Savestate* iniciam o dispositivo a partir de uma sessão salva anteriormente, o que será bem rápido.

Os atalhos *Savestate* só vão funcionar depois que cada uma das sessões dos atalhos *Coldboot* forem salvas. Ao salvar a sessão no emulador é criado um arquivo DESS, permitindo reiniciar a imagem a partir do mesmo ponto onde ela parou de rodar quando o emulador foi fechado. Para salvar a sessão no emulador, acesse o menu *File* e clique na opção *Save State and Exit*.

Para configurar o *Storage Card* para as imagens, você terá que configurar cada uma das *Coldboot*. Assim, clique com o botão direito em uma e escolha *Propriedades*. Na aba *Atalho*, no campo *Destino*, adicione no final a linha `"/sharedfolder PastaQueDesejarUsarComoStorageCard"`.

Se o caminho da pasta a ser utilizada tiver espaço, coloque todo o caminho "entre aspas". Após configurar o *Storage Card* nos três atalhos *Coldboot*, execute algum deles.

Para abrir o *Storage Card*, clique no menu *Iniciar* do emulador e escolha *File Explorer*. Na parte superior da janela existe um botão onde você pode selecionar *Storage Card* para abrir a pasta compartilhada no seu computador e acessar os arquivos lá.

Configurando a rede no Device Emulator

Com as imagens de Windows Mobile (o Device Emulator só usa essas) o acesso à Internet ou a uma rede local é feito por *Bluetooth*, infravermelho ou via *ActiveSync*, esse último quando se conecta o dispositivo a um PC. Temos também como emular essa conexão do emulador


Listagem 7. Método *Show* do *InputBox*

```
class function InputBox.Show(Title,
    Question: string): string;
var
    box: InputBox;
begin
    box := InputBox.Create;
    box.Text := Title;
    box.Label1.Text := Question;
    if box.ShowDialog = System.Windows.Forms.DialogResult.OK then
        Result := box.TextBox1.Text
    else Result := '';
end;
```

Listagem 8. Menu Selecionar dados no Servidor

```
var
    ws: TPesquisaOpinioWS;
    titulo: string;
    codPesquisa: Integer;
begin
    { Chama o InputBox para solicitar o título da pesquisa ao usuário }
    titulo := InputBox.Show('Selecionar Pesquisa', 'Título da pesquisa');
    if titulo <> '' then
        begin
            { Instancia um objeto da classe do Webservice }
            ws := TPesquisaOpinioWS.Create;
            { Busca os dados da pesquisa no Webservice }
            dsPesquisa := ws.GetPesquisa(titulo);
            if (dsPesquisa.Tables[0].Rows.Count > 0) then
                begin
                    { Liga a tabela do DataSet para que as pesquisas sejam exibidas no ComboBox }
                    cbxPesquisa.DataSource := dsPesquisa.Tables[0];
                    { Guarda o código da pesquisa retornada }
                    codPesquisa := Convert.ToInt32(dsPesquisa.Tables[0].Rows[0]['cod_pesquisa']);
                    { Armazena os dados da pesquisa localmente em um XML. XMLCache é a classe que criamos
                      para agrupar as funções para manipulação dos DataSets }
                    XMLCache.GravarXML(dsPesquisa, XMLCache.arqPesquisa);
                    dsPergunta := ws.GetPerguntas(codPesquisa);
                    XMLCache.GravarXML(dsPergunta, XMLCache.arqPergunta);
                    { Se existe alguma pergunta cadastrada para a pesquisa }
                    if dsPergunta.Tables[0].Rows.Count > 0 then
                        begin
                            dsAlternativa := ws.GetAlternativas(codPesquisa);
                            XMLCache.GravarXML(dsAlternativa, XMLCache.arqAlternativa);
                            { Chama a função GetEntrevistado do Webservice informando um nome de pessoa que não
                              vai existir, para que seja retornado um DataSet vazio, apenas com a estrutura da
                              tabela para que se possa armazenar os dados localmente neste DataSet }
                            dsEntrevistado := ws.GetEntrevistado('nenhum');
                            XMLCache.GravarXML(dsEntrevistado, XMLCache.arqEntrevistado);
                            { A mesma lógica anterior }
                            dsResposta := ws.GetRespostaEntrevistado(-1);
                            XMLCache.GravarXML(dsResposta, XMLCache.arqResposta);
                        end;
                    end
                else
                    MessageBox.Show('Nenhuma pesquisa retornada', 'Informação');
                end;
            end;
```

(o dispositivo que temos) ao PC.

Você precisará que o Device Emulator Manager (encontrado na mesma pasta do menu iniciar indicada anteriormente) e o ActiveSync estejam rodando. O ActiveSync disponibiliza o ícone  no *System Tray*.

Clique com o botão direito nele e escolha a opção *Configurações de conexão*. Na janela que abre, marque a opção *Permitir conexão com um dos seguintes itens* e no campo abaixo, escolha *DMA* para permitir conectar o computador virtualmente ao emulador, sem uma conexão física. Clique em OK.

Após rodar o emulador, no Microsoft Device Emulator deverá aparecer a referência para o dispositivo, se não, clique no *Refresh*. Com o botão direito sobre o link para dispositivo, dentro do Microsoft Device Emulator, escolha a opção *Cradle*, para

simular a conexão do dispositivo ao PC.

O ícone do ActiveSync deve ativar, ficando verde, indicando que está conectando ao dispositivo. Após ele concluir a conexão, você poderá acessar uma rede local ou à internet a partir do dispositivo, sem nenhuma configuração adicional.

Caso o dispositivo não seja conectado pelo ActiveSync automaticamente, clique com o botão direito no ícone do ActiveSync no *System Tray* e escolha a opção *Configurações de conexão*. Na janela que abre, clique em *Conectar*.

Após o emulador estar conectado no ActiveSync, acesse o *Storage Card* e execute a aplicação. Para atualizar o sistema, basta copiar o executável para a pasta compartilhada e executar novamente no emulador, sem precisar fechá-lo.

Listagem 9. SelectedValueChanged do cbxPergunta

```
var
  codPergunta: Integer;
begin
  { Se há alguma pergunta selecionada, mostra no ComboBox de alternativa (cbxResposta),
  somente as referentes à pergunta selecionada }
  if (cbxPergunta.SelectedIndex <> -1) and (cbxPergunta.SelectedValue <> nil) then
  begin
    { Pega o código da pergunta selecionada }
    codPergunta := Convert.ToInt32(cbxPergunta.SelectedValue);
    { Filtra o DataSet de alternativas para exibir apenas as alternativas da resposta selecionada }
    dsAlternativa.Tables[0].DefaultView.RowFilter := 'cod_pergunta=' + codPergunta.ToString;
    { Vincula os registros filtrados ao ComboBox das Alternativas }
    cbxResposta.DataSource := dsAlternativa.Tables[0].DefaultView;
    { Não deixa nenhum item marcado no ComboBox }
    cbxResposta.SelectedIndex := -1;
  end;
end;
```

Listagem 10. Incluir Entrevistado

```
public
  CodEntrevistado: integer;
...
begin
  { Inclui o entrevistado no DataSet local, retornando o código temporário gerado para o
  registro, pois este código é alterado quando o registro é enviado ao banco de dados }
  CodEntrevistado := XMLCache.IncluirEntrevistado(dsEntrevistado, txtEntrevistado.Text,
  Convert.ToInt32(cbxPesquisa.SelectedValue));
  { Exibe o código gerado }
  lblCodEntrevistado.Text := 'Código: ' + CodEntrevistado.ToString;
  { Desvincula o procedimento
  cbxPergunta_SelectedValueChanged do evento SelectedValueChanged do cbxPergunta enquanto os dados
  são vinculados ao cbxPergunta, para evitar que o evento seja disparado no momento errado }
  Exclude(cbxPergunta.SelectedValueChanged, cbxPergunta_SelectedValueChanged);
try
  { Vincula os dados do DataSet dsPergunta ao cbxPergunta para que as perguntas sejam
  exibidas no ComboBox }
  cbxPergunta.DataSource := dsPergunta.Tables[0];
finally
  { Vincula novamente o procedimento cbxPergunta_SelectedValueChanged ao evento
  SelectedValueChanged do cbxPergunta }
  Include(cbxPergunta.SelectedValueChanged, cbxPergunta_SelectedValueChanged);
end;
{ Desmarca a pergunta atualmente selecionada }
cbxPergunta.SelectedIndex := -1;
{ Manda o cursor para o campo }
cbxPergunta.Focus;
```

Listagem 11. Procedimentos para navegar entre as perguntas

```
procedure frmMain.PerguntaAnterior;
begin
  { Volta à pergunta anterior }
  if cbxPergunta.SelectedIndex > 0 then
    cbxPergunta.SelectedIndex := cbxPergunta.SelectedIndex - 1;
  cbxPergunta.Focus;
end;

procedure frmMain.ProximaPergunta;
begin
  { Avança à próxima pergunta }
  if cbxPergunta.SelectedIndex < cbxPergunta.Items.Count - 1 then
    cbxPergunta.SelectedIndex := cbxPergunta.SelectedIndex + 1;
  cbxPergunta.Focus;
end;
```

Listagem 12. Registrar a resposta do entrevistado

```
{ Registra a resposta do entrevistado no DataSet local e passa para a próxima pergunta }
XMLCache.RegistrarResposta(dsResposta, CodEntrevistado, Convert.ToInt32(
  cbxPergunta.SelectedValue), Convert.ToInt32(cbxResposta.SelectedValue));
ProximaPergunta;
```

Listagem 13. Salva os dados dos DataSets em arquivos XML

```
procedure frmMain.SalvarCacheLocal;
begin
  { Se existe alguma tabela no DataSet e existe algum registro nela, salva os dados em arquivo }
  if (dsEntrevistado.Tables.Count > 0) and
    (dsEntrevistado.Tables[0].Rows.Count > 0) then
    XMLCache.GravarXML(dsEntrevistado, XMLCache.arqEntrevistado);
  if (dsResposta.Tables.Count > 0) and (dsResposta.Tables[0].Rows.Count > 0) then
    XMLCache.GravarXML(dsResposta, XMLCache.arqResposta);
end;
```

Continuando o desenvolvimento da aplicação cliente

Depois de ter testado o Device Emulador, vamos continuar no desenvolvimento da aplicação. Quando o usuário selecionar uma pergunta, devemos mostrar no *ComboBox* de alternativas, somente as referentes à pergunta selecionada. Assim, no evento *SelectedValueChanged* do *cbxPergunta* adicione o código da **Listagem 9**.

Agora inclua o código para o *Click* do *Incluir*, que fará a inclusão do entrevistado no *DataSet* local, conforme a **Listagem 10**.

Crie os procedimentos “PerguntaAnterior” e “ProximaPergunta” para permitir ao usuário navegar entre as perguntas disponíveis. Veja o código dos dois procedimentos na **Listagem 11**.

Agora chame os procedimentos nos seus respectivos botões. No evento *Click* do *Registrar* digite o código da **Listagem 12**, para registrar a resposta do entrevistado no *DataSet* local.

Testando a aplicação

Compile a aplicação (usando a barra do CF Builder), copie o executável para a pasta compartilhada e execute-o a partir do *Storage Card* no Device Emulator. Você deverá acessar o menu *Selecionar dados no Servidor*, caso ainda não tenha feito isso, para guardar os dados da pesquisa, as perguntas e alternativas, localmente em arquivos XML no dispositivo móvel.

Digite o nome do entrevistado e clique em *Incluir* para registrá-lo. Em seguida, selecione a pergunta, depois a resposta e clique em *Registrar*. Repita esse último passo até responder todas as perguntas.

Para incluir um novo questionário de um entrevistado, digite o nome do mesmo e clique em *Incluir*, repetindo todo o processo.

Salvando em disco

Vamos agora fazer com que os dados sejam salvos em disco quando o sistema for fechado. Poderíamos fazer isso em outro evento também, como a cada pergunta respondida. Assim, crie a função da **Listagem 13** e depois chame-a no evento *Closed* do *FrmMain*.

Agora que fizemos a rotina para chamar a função para salvar os *DataSets* localmente, precisamos criar o código para carregar esses arquivos quando a aplicação inicializar. Assim, digite o código da **Listagem**

Listagem 14. Evento Load do FrmMain para carregar os arquivos XML

```

[ Se carregou o arquivo de pesquisas, carrega os outros em seguida ]
if XMLCache.CarregarXML(dsPesquisa, XMLCache.arqPesquisa) then
begin
  [ Vincula os dados do DataSet dsPesquisa ao cbxPesquisa para exibir o título da pesquisa
  no ComboBox ]
  cbxPesquisa.DataSource := dsPesquisa.Tables[0];
  [ Se carregou o arquivo de perguntas, então carrega o de alternativas ]
  if XMLCache.CarregarXML(dsPergunta, XMLCache.arqPergunta) then
  XMLCache.CarregarXML(dsAlternativa, XMLCache.arqAlternativa);
  [ Se carregou o arquivo de entrevistados, então carrega o de respostas ]
  if XMLCache.CarregarXML(dsEntrevistado, XMLCache.arqEntrevistado) then
  XMLCache.CarregarXML(dsResposta, XMLCache.arqResposta);
end;

```

14 no evento Load do FrmMain.

Quando o usuário tiver conexão com a internet ou diretamente à rede local da empresa, ele deve enviar os dados ao servidor para serem cadastrados no banco de dados, usando o *EnviarCacheXML* criada anteriormente.

No *FrmMain* inclua um novo item no *MainMenu* com o título (propriedade *Text*) igual a "Enviar dados ao Servidor". No *Click* deste item inclua o código a seguir para chamar a função recém-criada, que enviará os dados armazenados localmente em XML ao servidor:

```
XMLCache.EnviarCacheXML(dsEntrevistado,
dsResposta);
```

Rode a aplicação e faça os testes necessários (**Figura 4**).

Conclusão

Pronto, agora é só fazer os testes. Alguns detalhes para melhoria da interface não foram incluídos, como a habilitação/desabilitação dos botões nos momentos adequados, para evitar problemas caso o usuário não siga a ordem correta, mas isso é bem simples de fazer e fica para vocês poderem melhorar a aplicação.

Um recurso muito interessante seria a inclusão de um *CheckBox* para informar se é para trabalhar off-line. Estando desmarcado, a aplicação já mandaria os dados diretamente para o servidor.

Isso também é fácil de implementar, basta chamar as devidas funções, que já estão implementadas. Outro detalhe, que é bastante importante, é quanto ao controle de transações que não foi aplicado, para garantir, por exemplo, que o registro

do entrevistado com os registros das respostas sejam todos incluídos ou, em caso de erro, nenhum ser incluído.

Também não me preocupei em verificar se o usuário informou valores para os campos, antes de inserir os dados. Assim, fica o homework para vocês fazerem. Neste artigo procurei mostrar a maioria dos detalhes quanto à montagem do ambiente de desenvolvimento, configuração e utilização das ferramentas assim como macetes para permitir que você ingresse no mundo do desenvolvimento de aplicações para dispositivos móveis.

Espero que tenham gostado e até o próximo. ●



Figura 4. Aplicação sendo testada no Pocket PC

ClubeDelphi PLUS

www.devmedia.com.br/clubedelphi/portal.asp

Acesse agora o mesmo o portal do assinante ClubeDelphi e assista a uma série de vídeo aulas de Jefferson Junglaus, sobre o Compact Framework no Delphi.

www.devmedia.com.br/articles/listcomp.asp?txtsearch=Compact+Framework+no+Delphi

Mais Vendidos!



Gerenciando Projetos de Desenvolvimento de Software com PMI, RUP e UML
4ª. Edição

Brasport

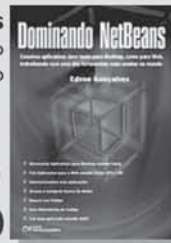
R\$ 75,00

Dominando NetBeans
Construa aplicativos Java tanto em Desktop, como para Web

Ciência Moderna

De: R\$ 76,00

Por: R\$ 72,20



Desenvolvendo Aplicações Web com NetBeans IDE 5.5

Ciência Moderna

R\$ 99,00

Programação Orientada a Objetos Usando Delphi - 4ª edição

Visual Books

De: R\$ 75,00

Por: R\$ 71,25



Frete Grátis
Para todo o Brasil!

Até 6x sem juros no cartão.*
*Parcela mínima de R\$ 20,00

Outras facilidades de pagamento:



Tel: (11) 4062-5152

livrosdeprogramacao.com.br