

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



## Controle on-line de vídeo-locadora - Parte 1

Veja como criar um sistema on-line de controle para uma vídeo-locadora

Junto com o aumento acelerado do uso da internet, a cada dia que passa a necessidade de empresas de todos os ramos e portes em ingressar com seu negócio no mundo virtual tem se tornado uma rotina. Surgem necessidades para as empresas encontrarem maneiras de negociar com o seu cliente à distância, ou seja, através da internet, sejam por meio de grandes portais de informações ou através de simples sistemas rodando na *Web* onde clientes/usuários credenciados podem interagir e realizar reservas, compras, vendas, entre outros.

O objetivo deste artigo é justamente a criação de um sistema *on-line*, onde a situação descrita será aplicada para as atividades diárias de uma vídeo-locadora. Veremos nesta e na próxima edição um passo-a-passo do desenvolvimento do aplicativo, que será dividido em duas formas de interação com usuários: administradores e usuários comuns. Onde toda a parte administrativa de manutenção de

cadastros, reservas e retiradas de vídeos e um espaço para o cliente o qual com uso de seu usuário e senha poderá fazer a consulta de filmes disponíveis e realizar pela *Web* a reserva dos mesmos, além do acesso a um histórico de retiradas já efetuadas, economizando assim o tempo gasto na locadora para a busca do filme ideal.

A aplicação será desenvolvida sob a plataforma .NET Framework 1.1 com uso da ferramenta de desenvolvimento Borland Developer Studio 2006 for ASP.NET e banco de dados Firebird 1.5.

Nesta primeira etapa do artigo veremos a análise e criação da base de dados, criação da área de cadastros de filmes, gêneros, clientes e usuários, manutenção de reservas realizadas por clientes e novas reservas pelo administrador da locadora.

### Antes de começar...

Antes de codificar qualquer coisa em ASP.NET, é importante que você tenha em mente alguns conceitos e tome al-



**Maikel Marcelo Scheid**

([maikelscheid@gmail.com](mailto:maikelscheid@gmail.com))

é técnico em Informática com ênfase em Análise e Programação de Sistemas. Atua na área de Desenvolvimento de Softwares em Delphi para plataforma Win32 e .NET com banco de dados Firebird e MS SQL. É membro da equipe Editorial ClubeDelphi.

guns cuidados importantes, principalmente se está migrando de uma solução Desktop/VCL:

- Não queira fazer na Web as coisas exatamente como você faz em aplicações Windows/Desktop - o uso de *Web Forms* do ASP.NET ajuda, claro, mas saiba que você está rompendo de uma vez duas barreiras: mudança de framework (VCL para .NET FCL) e mudança de paradigma (Windows para Web). Um browser não é um formulário Delphi, infelizmente, então se acostume às limitações do HTML e HTTP;

- Saiba como funciona o modelo de execução do ASP.NET - é importantíssimo lembrar que o ASP.NET (e aplicações *Web* como um todo) são *stateless*: cada requisição é processada como se fosse a primeira. Além disso, o ASP.NET cria e recria o formulário no servidor para cada requisição, o que é bem diferente do modelo "TForm-based". É claro, temos alguns recursos para tratar sessão, cache, *ViewState* etc.

Começaremos então nosso exemplo, partindo pela conexão com o banco de dados.

## Escolhendo o provider para ADO.NET

Temos somente uma opção para acessar o Firebird a partir de aplicações construídas para o .NET Framework: o ADO.NET. Sabendo disso, o próximo passo é escolher um *Provider* para ADO.NET que permita o melhor acesso ao banco de dados.

O .NET Framework 1.1 já é distribuído com quatro *Providers* nativos para ADO.NET:

- SQL Provider* - para acesso ao SQL Server;
- Oracle Provider* - para acesso ao Oracle;
- OleDb Provider* - para acesso a fontes de dados que possuam um driver OleDb;

- ODBC Provider* - para acesso a fontes de dados que possuam um driver ODBC;

Para acessar o Firebird, poderíamos usar qualquer um dos dois últimos *Providers* (ODBC ou OLEDB). No entanto, seria necessário instalar uma camada extra (um driver OLEDB ou ODBC), o que comprometeria o tempo de resposta, que é crítico em soluções *Web*.

Temos ainda mais duas opções:

- BDP - Borland Data Provider* - um *Pro-*

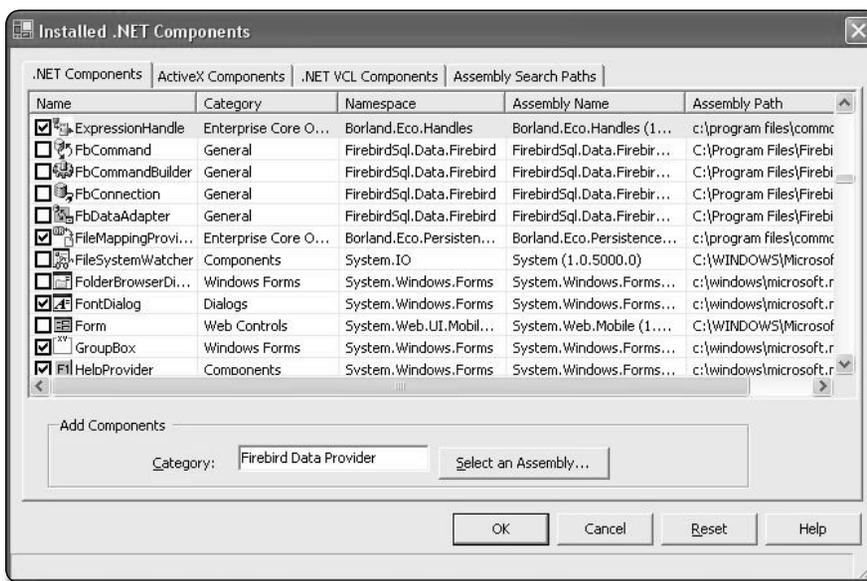


Figura 1. Instalação do provider .NET no Delphi

*vider* para ADO.NET construído pela Borland, distribuído com o Delphi.

*Firebird Data Provider* - *Provider* para ADO.NET específico para o Firebird, desenvolvido e distribuído no mesmo site do banco de dados.

Se você desejar usar o BDP, deverá instalar um *driver* extra para o BDP, já que o Delphi suporta somente o acesso nativo ao InterBase, que como sabemos não é totalmente compatível com o Firebird. Nesse caso, optaremos pelo *Firebird Data Provider*, mais otimizado e específico para acessar o Firebird.

## Download e instalação do Firebird Data Provider

Acesse o endereço [www.firebirdsql.com](http://www.firebirdsql.com), clique no link *Download* e a seguir em *Firebird .NET Data Provider*. Baixe e instale a última versão do *Data Provider for .NET Framework 1.1*, bastando seguir os passos no assistente que será iniciado.

**Nota:** Neste artigo usaremos versão 1.7.1 do *Provider*, as demais versões estão disponíveis para Framework acima do 1.1 usado pelo BDS 2006.

Após a instalação, no Delphi 2006 clique no menu *Component\Installed .NET Components*. Digite "Firebird Data Provider" na opção *Category*, clique no botão *Select an Assembly* (Figura 1) e escolha o arquivo *FirebirdSql.Data.Firebird.dll*, localizado no



Figura 2. Componentes do provider .NET Firebird já instalados

diretório de instalação do *Provider*, por padrão em *C:\Arquivos de programas\FirebirdNETProvider(versão)*. Clique em *Ok* e observe que os novos componentes para acesso ao Firebird estão agora disponíveis no IDE (Figura 2).

## Criando o banco de dados

Para iniciar o projeto, vamos primeiramente criar o banco de dados e as tabelas necessárias para cada tela de cadastro. As tabelas são: *Cientes*, *Usuarios*, *Generos*, *Locacao* e *Videos*. Neste artigo, criaremos o banco de dados utilizando a ferramenta *IBExpert* em sua versão *Standard*. Por isso, acesse o link [www.ibexpert.com](http://www.ibexpert.com) e em seguida entre no item *IBExpert*.

À esquerda do site do fabricante clique

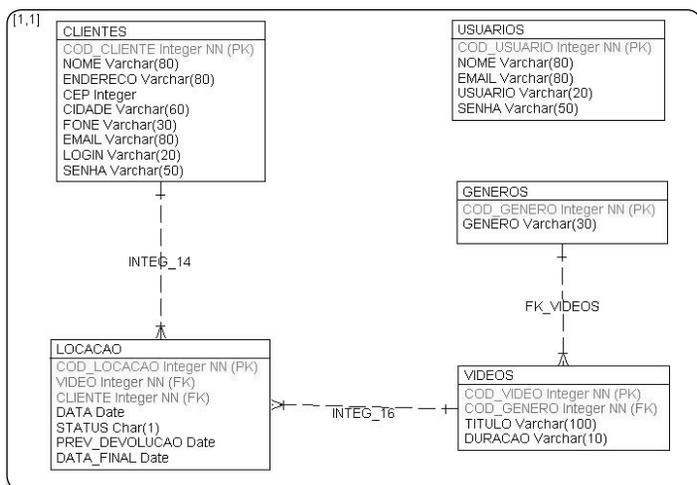


Figura 3. Modelo ER da base de dados

em *Download>Free*. Preencha o formulário de cadastro e aguarde o e-mail com as instruções de *download* da ferramenta.

Nosso banco de dados será como mostrado no modelo ER visualizado na **Figura 3**. Criaremos as tabelas necessárias e seus respectivos índices e chaves estrangeiras conforme o modelo. Para agilizar o processo faremos a criação utilizando o recurso de execução de *scripts* no IBExpert. Abra então a ferramenta e vamos criar o banco de dados usando a opção *Script Executive* presente no menu *Tools*.

Com a tela de *scripts* aberta, digite o código da **Listagem 1**. Nele estão contidos os esquemas para a criação do banco de dados assim como de cada tabela do sistema.

Listagem 1. Script de criação da base de dados

```

SET SQL DIALECT 3;
SET NAMES WIN1252;

SET CLIENTLIB 'C:\Program Files\Firebird\Firebird_1_5\bin\fbclient.dll';

CREATE DATABASE '<Caminho>\Locadora.fdb'
USER 'SYSDBA' PASSWORD 'masterkey'
PAGE_SIZE 4096
DEFAULT CHARACTER SET WIN1252;

CREATE GENERATOR GEN_CLIENTES_ID;
SET GENERATOR GEN_CLIENTES_ID TO 1;
CREATE GENERATOR GEN_GENEROS_ID;
SET GENERATOR GEN_GENEROS_ID TO 6;
CREATE GENERATOR GEN_LOCACAO_ID;
SET GENERATOR GEN_LOCACAO_ID TO 40;
CREATE GENERATOR GEN_USUARIOS_ID;
SET GENERATOR GEN_USUARIOS_ID TO 0;
CREATE GENERATOR GEN_VIDEOS_ID;
SET GENERATOR GEN_VIDEOS_ID TO 7;

CREATE TABLE CLIENTES (
  COD_CLIENTE INTEGER NOT NULL,
  NOME VARCHAR(80) CHARACTER SET NONE,
  ENDEREÇO VARCHAR(80) CHARACTER SET NONE,
  CEP INTEGER,
  CIDADE VARCHAR(60) CHARACTER SET NONE,
  FONE VARCHAR(30) CHARACTER SET NONE,
  EMAIL VARCHAR(80) CHARACTER SET NONE,
  LOGIN VARCHAR(20),
  SENHA VARCHAR(50)
);
CREATE TABLE GENEROS (
  COD_GENERO INTEGER NOT NULL,
  GENERO VARCHAR(30) CHARACTER SET NONE
);
CREATE TABLE LOCACAO (
  COD_LOCACAO INTEGER NOT NULL,
  VIDEO INTEGER NOT NULL,
  CLIENTE INTEGER NOT NULL,
  DATA DATE Default 'TODAY',
  STATUS CHAR(1) CHARACTER SET NONE,
  PREV_DEVOLUCAO DATE Default 'TODAY',
  DATA_FINAL DATE
);
CREATE TABLE USUARIOS (
  COD_USUARIO INTEGER NOT NULL,
  NOME VARCHAR(80),
  EMAIL VARCHAR(80),
  USUARIO VARCHAR(20),
  SENHA VARCHAR(50)
);
CREATE TABLE VIDEOS (
  COD_VIDEO INTEGER NOT NULL,
  COD_GENERO INTEGER NOT NULL,
  TITULO VARCHAR(100),
  DURACAO VARCHAR(10)
);
ALTER TABLE CLIENTES ADD PRIMARY KEY (COD_CLIENTE);
ALTER TABLE GENEROS ADD PRIMARY KEY (COD_GENERO);
ALTER TABLE LOCACAO ADD PRIMARY KEY (COD_LOCACAO);
ALTER TABLE USUARIOS ADD PRIMARY KEY (COD_USUARIO);
ALTER TABLE VIDEOS ADD PRIMARY KEY (COD_VIDEO);
ALTER TABLE LOCACAO ADD FOREIGN KEY (CLIENTE)
  REFERENCES CLIENTES (COD_CLIENTE)
  ON DELETE NO ACTION ON UPDATE NO ACTION;
ALTER TABLE LOCACAO ADD FOREIGN KEY (VIDEO)
  REFERENCES VIDEOS (COD_VIDEO)
  ON DELETE NO ACTION ON UPDATE NO ACTION;
ALTER TABLE VIDEOS ADD CONSTRAINT FK_VIDEOS
  FOREIGN KEY (COD_GENERO) REFERENCES
  GENEROS (COD_GENERO);
SET TERM ^;

CREATE TRIGGER CLIENTES_BI FOR CLIENTES
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  IF (NEW.COD_CLIENTE IS NULL) THEN
    NEW.COD_CLIENTE = GEN_ID(GEN_CLIENTES_ID,1);
END
^
CREATE TRIGGER GENEROS_BI FOR GENEROS
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  IF (NEW.COD_GENERO IS NULL) THEN
    NEW.COD_GENERO = GEN_ID(GEN_GENEROS_ID,1);
END
^
CREATE TRIGGER LOCACAO_BI FOR LOCACAO
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  IF (NEW.COD_LOCACAO IS NULL) THEN
    NEW.COD_LOCACAO = GEN_ID(GEN_LOCACAO_ID,1);
END
^
CREATE TRIGGER USUARIOS_BI FOR USUARIOS
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  IF (NEW.COD_USUARIO IS NULL) THEN
    NEW.COD_USUARIO = GEN_ID(GEN_USUARIOS_ID,1);
END
^
CREATE TRIGGER VIDEOS_BI FOR VIDEOS
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  IF (NEW.COD_VIDEO IS NULL) THEN
    NEW.COD_VIDEO = GEN_ID(GEN_VIDEOS_ID,1);
END
^
SET TERM ; ^
DESCRIBE FIELD STATUS TABLE LOCACAO
N = Normal
R = Reservado
D = Devolvido';

```

Digitado o *script*, basta executá-lo usando o botão *Run Script* ou pressione a tecla F9.

**Nota:** Substitua <Caminho>\Locadora.fdb pelo diretório do banco de dados e seu nome, ex: C:\inetpub\wwwroot\Locadora.fdb.

Após a execução do *script* teremos nosso banco de dados, tabelas, índices, chaves estrangeiras e tudo que é necessário para que nosso *BD* funcione perfeitamente.

### Criando a aplicação

No BDS 2006 selecione o menu *File\New>ASP.NET Web Application – Delphi for .NET*. Na caixa de diálogo que aparece (**Figura 4**), defina o nome do projeto como “VideoLocadora”, no item *Location* selecione o diretório destino onde os arquivos do projeto serão criados e no item *Server* defina o servidor de aplicação para a hospedagem do sistema.

Altere a página inicial, *WebForm1.aspx* para “principal.aspx”, para isso clique com o botão direito do mouse no nome da página no *Project Manager* e selecione *Rename*. Antes de trabalharmos na página criada, iremos definir uma estrutura padrão para o cabeçalho do nosso sistema através de *User Control* na qual mais adiante faremos também o controle de usuários *logados*. Para adicionar o recurso é necessário criar uma página *.ascx* através do menu *File\New>Other>Delphi for .NET Projects>New ASP.NET Files>ASP.NET User Control*. Dessa forma o Delphi fará a criação de uma nova página de edição do conteúdo. Altere no *Project Manager* o nome do *User Control* para “uccontrole.ascx” e na página adicione uma tabela com cerca de 700 *pixels* de largura contendo quatro linhas e uma coluna. Adicione às linhas da tabela o nome do sistema, uma imagem de identificação, um espaço para identificação do usuário *logado*, onde deverá adicionar também um componente *Label* (“lblUsuario”) e mais abaixo uma linha (“<hr>”) para separação do cabeçalho ao conteúdo das páginas. Veja um exemplo de *layout* da **Figura 5**.

Retornando a página a *principal.aspx*, crie uma tabela de quatro linhas e uma

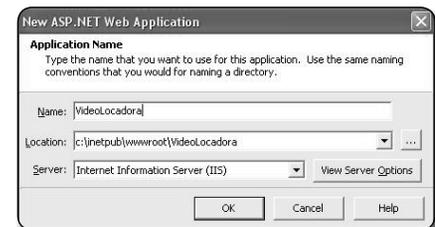
coluna no centro da página com 700 *pixels* de largura. Arraste do *Project Manager* para dentro da primeira linha da tabela o *User Control uccontrole.ascx* que criamos. Já na segunda linha da tabela defina apenas um título para a localização do usuário, descreva como “Área Administrativa”. Na linha três, adicione uma nova tabela para que possamos definir uma estrutura de menus para o sistema, crie uma tabela usando 100% da largura com quatro linhas e três colunas e adicione os seguintes componentes *Button* nas células da tabela: *Gêneros* (“btnGeneros”), *Vídeos* (“btnVideos”), *Clientes* (“btnClientes”), *Usuários* (“btnUsuarios”), *Locações* (“btnLocacao”), *Lista de Reservas* (“btnReservas”) e *Sair* (“btnSair”). Ainda na última linha da primeira tabela criada, adicione uma *tag* “<hr>” e os direitos de criação para sua aplicação. Clique duas vezes no *btnGeneros* e adicione o seguinte código, que ao ser clicado irá direcionar o usuário à devida página de manutenção dos gêneros de vídeos. O método *Redirect* sempre é usado quando necessitamos abrir uma outra página:

```
Response.Redirect('a_generos.aspx');
```

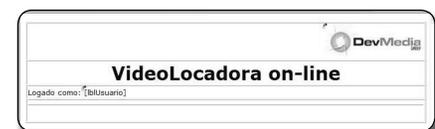
Como já codificamos o botão de manutenção dos gêneros de vídeos, veremos a seguir a criação da página e codificação para o cadastro de novos gêneros e manutenção de registros existentes. Crie uma nova página através do menu *File\New>Other>Delphi for .NET Projects>New ASP.NET Files>ASP.NET Page* e altere seu nome para “a\_generos.aspx”. Adicione ao corpo da página uma tabela centralizada com seis linhas, uma

coluna e 700 *pixels* de largura. Arraste para a primeira linha o *User Control* da aplicação conforme realizado na criação da página principal. Na segunda linha defina o título da página para “Manutenção de Gêneros” e na linha seguinte adicione da *Tool Palette* um componente *DataGrid* (“GridGeneros”).

Faremos uso deste componente para visualizar os gêneros cadastrados no sistema. Mude a aparência do componente acessando a opção *Auto Format*, escolha um formato que lhe agrade e confirme. Em seguida clique novamente com o botão direito do mouse e escolha *Property Builder* (“Editor de Propriedades”). Nessa janela de configurações iremos



**Figura 4.** Criação da aplicação



**Figura 5.** Exemplo de cabeçalho



**Figura 6.** Exemplo de tela de Gêneros

**Listagem 2.** Código do evento Page\_Load da página

```
interface
...
const
  strConexao = 'User=SYSDBA;Password=masterkey;
  Database=<Caminho>LOCADORA.FDB;';
...
procedure TWebForm1.Page_Load(sender: System.Object;
e: System.EventArgs);
var
  Conn : FbConnection;
  DataAdapter : FbDataAdapter;
  Ds : DataSet;
  Comand : FbCommand;
begin
  { Criação dos objetos de conexão }
  Conn := FbConnection.Create;
  DataAdapter := FbDataAdapter.Create;
  Comand := FbCommand.Create;
  { Atribuição da string de conexão e abertura do BD}
  Conn.ConnectionString := strConexao;
  Conn.Open;
  { Atribuição dos atributos de seleção dos dados }
  DataAdapter.SelectCommand := Comand;
  DataAdapter.SelectCommand.Connection := Conn;
  DataAdapter.SelectCommand.CommandText :=
  'SELECT COD_GENERO, GENERO FROM GENEROS';
  { Criação em memória do DataSet auxiliar }
  Ds := DataSet.Create;
  DataAdapter.Fill(Ds, 'Genero');
  try
    GridGeneros.DataSource := Ds;
    GridGeneros.DataBind;
  finally
    Conn.Close;
  end;
end;
```

**Listagem 3.** Código do botão Incluir

```
procedure TWebForm1.btnSalvar_Click(sender:
System.Object; e: System.EventArgs);
var
  Conn : FbConnection;
  DataAdapter : FbDataAdapter;
  Ds : DataSet;
  Comand : FbCommand;
  prGenero : FbParameter;
begin
  { Criação dos objetos de conexão }
  Conn := FbConnection.Create;
  DataAdapter := FbDataAdapter.Create;
  Comand := FbCommand.Create;
  { Atribuição da string de conexão e abertura do BD}
  Conn.ConnectionString := strConexao;
  Conn.Open;
  { Atribuição dos atributos de seleção dos dados }
  DataAdapter.InsertCommand := Comand;
  DataAdapter.InsertCommand.Connection := Conn;
  DataAdapter.InsertCommand.CommandText :=
  'INSERT INTO GENEROS (GENERO) VALUES (?)';
  prGenero := FbParameter.Create;
  prGenero.ParameterName := 'GENERO';
  DataAdapter.InsertCommand.Parameters.Add(prGenero);
  DataAdapter.InsertCommand.Parameters[0].Value :=
  txtGenero.Text.ToUpper;

  if DataAdapter.InsertCommand.ExecuteNonQuery > 0 then
  begin
    { Atribuição dos atributos de seleção dos dados }
    DataAdapter.SelectCommand := Comand;
    DataAdapter.SelectCommand.Connection := Conn;
    DataAdapter.SelectCommand.CommandText :=
    'SELECT COD_GENERO, GENERO FROM GENEROS';
    { Criação em memória do DataSet auxiliar }
    Ds := DataSet.Create;
    DataAdapter.Fill(Ds, 'Genero');
    try
      GridGeneros.DataSource := Ds;
      GridGeneros.DataBind;
      btnIncluir.Enabled := True;
      btnAlterar.Enabled := False;
      txtGenero.Text := '';
    finally
      Conn.Close;
    end;
  end;
end;
```

selecionar a categoria *Columns* e realizar as algumas alterações.

Desmarque, no topo da janela, o item *Create columns automatically at run time*. Em seguida clique no item *Bound Column* em *Available columns* e envie-o para *Selected columns*. Agora selecione-a e atribua o valor COD\_GENERO na propriedade *Data Field*, onde receberemos o valor do campo na tabela *Generos* através do *result set* da *Select* que faremos. Desmarque também a opção *Visible* da coluna.

Agora insira uma nova coluna, ou seja, clique novamente em *Bound Column* e envie-o para a área *Selected columns*. Na propriedade *Header text* ("Texto de cabeçalho"), digite "Gênero" e em *Data Field* o valor GENERO que é justamente o campo de nossa tabela *Generos* assim como no passo anterior.

Por fim inclua uma nova coluna de botões, chamada de *Button Column*. Expanda o *Button Column* e selecione a opção *Select*. Envie-o para a direita e modifique a propriedade *Header text* e *Text* para "Alterar" e clique em *Ok* para confirmar nossas alterações.

---

**Nota:** Para alterar a largura das colunas, basta entrar novamente no *Property Builder* e selecionar o item *Format*. Nele podemos definir atributos para várias características do componente *Data Grid*. Note que temos a opção *Columns*. Expanda este item e clique na última coluna ("Alterar"). A direita digite 50 no campo *Width* ("largura") e confirme.

---

Na linha abaixo do *GridGeneros* digite o texto "Gênero:" e arraste um componente *TextBox*("txtGenero"). Na linha seguinte inclua três *Buttons*, sendo "btnIncluir", "btnAlterar" e "btnCancelar". O *btnAlterar* ficará com a propriedade *Enabled* desativada, ou seja, *False*. Por fim inclua um novo *Button*("btnVoltar") e em seu evento *OnClick* faça a chamada ao método *Redirect* conforme a seguir, para que possamos voltar a página principal (**Figura 6**):

```
Response.Redirect('a_principal.aspx');
```

## Conectando o sistema ao banco de dados

Neste artigo iremos utilizar os *providers* do Firebird .NET Provider, porém faremos

todo o processo de conexão e atualização dos controles em tempo de execução. A primeira coisa que faremos é codificar o evento *Page\_Load* da página, portanto pressione *F12* para visualizar a página de códigos e localize o evento citado. Digite o código da **Listagem 2** e vamos as explicações para entendermos melhor.

A primeira coisa que podemos ver na **Listagem 2** é que criamos uma constante na área *Interface* da página informando a *string* de conexão com o banco. Esse valor será usado mais adiante para que o componente *fbConnection* acesse o *BD*. Já no evento de carregamento da página, *Page\_Load*, criamos quatro componentes em *run time* que são:

- *Conn*: Um *fbConnection* para conexão direta com o *BD*;
- *DataAdapter*: Componente para receber as instruções *SQL*;
- *Ds*: *DataSet* que recebe o *result set* das instruções *SQL* executadas;
- *Comand*: Componente *fbCommand* para execução de comandos;

Logo de início criamos todos os componentes e atribuímos a *string* de conexão com o banco ao *Conn*, componente de conexão. Em seguida atribuímos a conexão ao objeto interno do *DataAdapter* chamado *SelectCommand*. É ele que receberá a instrução *SQL* de seleção dos dados em sua propriedade *CommandText*.

Em seguida criamos um *DataSet*("Ds") e chamamos o método *Fill* do *DataAdapter* passando a ele a variável *Ds*("DataSet") e o nome da tabela ("Genero"). Feito isso modificamos a propriedade *DataSource* do *GridGeneros* informando qual seu *DataSet*, nesse caso a variável *Ds*. E para preencher o *grid* usamos o método *DataBind*.

O uso deste método melhora exponencialmente a performance da página, carregando-a muito mais rápido do que se usássemos os componentes diretamente na página. Vamos agora fazer a codificação dos botões *Alterar* e *Incluir*. Começaremos pelo botão *Incluir* clicando duas vezes nele e digitando o código da **Listagem 3**.

Perceba que utilizamos a mesma metodologia do evento *Page\_Load*, ou seja, criamos os componentes todos em tempo de execução. A diferença aqui é que ao invés de usarmos o componente

#### Listagem 4. Código do botão Alterar

```
procedure TWebForm1.btnAlterar_Click(sender:
  System.Object; e: System.EventArgs);
var
  Conn : FbConnection;
  DataAdapter : FbDataAdapter;
  Ds : DataSet;
  Comand : FbCommand;
  prCod_Genero : FbParameter;
  prGenero : FbParameter;
begin
  Conn := FbConnection.Create;
  DataAdapter := FbDataAdapter.Create;
  Comand := FbCommand.Create;
  Conn.ConnectionString := strConexao;
  Conn.Open;
  DataAdapter.UpdateCommand := Comand;
  DataAdapter.UpdateCommand.Connection := Conn;
  DataAdapter.UpdateCommand.CommandText :=
    'UPDATE GENEROS SET GENERO = ? WHERE (COD_GENERO = ?)';
  prCod_Genero := FbParameter.Create;
  prCod_Genero.ParameterName := 'COD_GENERO';
  prGenero := FbParameter.Create;
  prGenero.ParameterName := 'GENERO';
  DataAdapter.UpdateCommand.Parameters.
    Add(prCod_Genero);
  DataAdapter.UpdateCommand.Parameters.
    Add(prGenero);
  DataAdapter.UpdateCommand.Parameters[0].Value :=
    txtGenero.Text.ToUpper;
  DataAdapter.UpdateCommand.Parameters[1].Value :=
    GridGeneros.Items.Item[GridGeneros.SelectedItem.
    ItemIndex].Cells[0].Text;

  if DataAdapter.UpdateCommand.ExecuteNonQuery > 0 then
  begin
    DataAdapter.SelectCommand := Comand;
    DataAdapter.SelectCommand.Connection := Conn;
    DataAdapter.SelectCommand.CommandText :=
      'SELECT COD_GENERO, GENERO FROM GENEROS';
    Ds := DataSet.Create;
    DataAdapter.Fill(Ds, 'Genero');
    try
      GridGeneros.DataSource := Ds;
      GridGeneros.DataBind;
      btnIncluir.Enabled := True;
      btnAlterar.Enabled := False;
      txtGenero.Text := '';
    finally
      Conn.Close;
    end;
  end;
end;
```

#### Listagem 5. Código de Update

```
...
DataAdapter.UpdateCommand.CommandText :=
  'UPDATE GENEROS SET GENERO = ? WHERE (COD_GENERO = ?)';
prCod_Genero := FbParameter.Create;
prCod_Genero.ParameterName := 'COD_GENERO';
prGenero := FbParameter.Create;
prGenero.ParameterName := 'GENERO';
DataAdapter.UpdateCommand.Parameters.Add(prCod_Genero);
DataAdapter.UpdateCommand.Parameters.Add(prGenero);
DataAdapter.UpdateCommand.Parameters[0].Value := txtGenero.Text.ToUpper;
DataAdapter.UpdateCommand.Parameters[1].Value :=
  GridGeneros.Items.Item[GridGeneros.SelectedItem.ItemIndex].Cells[0].Text;
```



Listagem 6. Código do evento Page\_Load de a\_videos.aspx

```

procedure TWebForm1.Page_Load(sender: System.Object;
e: System.EventArgs);
var
  Conn : FbConnection;
  DataAdapter : FbDataAdapter;
  Ds : DataSet;
  Comand : FbCommand;
begin
  Conn := FbConnection.Create;
  DataAdapter := FbDataAdapter.Create;
  Comand := FbCommand.Create;
  Conn.ConnectionString := strConexao;
  Conn.Open;
  DataAdapter.SelectCommand := Comand;
  DataAdapter.SelectCommand.Connection := Conn;
  DataAdapter.SelectCommand.CommandText :=
  'SELECT VIDEOS.COD_VIDEO, VIDEOS.TITULO, ' +
  'GENEROS.GENERO,VIDEOS.DURACAO,GENEROS.COD_GENERO'+
  ' FROM GENEROS' +
  ' INNER JOIN VIDEOS ON (GENEROS.COD_GENERO = ' +
  'VIDEOS.COD_GENERO)';
  Ds := DataSet.Create;
  DataAdapter.Fill(Ds, 'Genero');
  try
    GridVideos.DataSource := Ds;
    GridVideos.DataBind;
    { Atualiza o componente DropDownList }
    DataAdapter.SelectCommand := Comand;
    DataAdapter.SelectCommand.Connection := Conn;
    DataAdapter.SelectCommand.CommandText :=
    'SELECT * FROM GENEROS';
    DataAdapter.Fill(Ds, 'Generos');
    with ddlGeneros do
      begin
        Items.Clear;
        DataTextField := 'GENERO';
        DataValueField := 'GENERO';
        DataSource := Ds;
        ddlGeneros.DataBind;
      end;
    finally
      Conn.Close;
    end;
  end;
end;

```

interno *SelectCommand*, estamos usando *InsertCommand*. Passamos para ele a instrução *SQL* de inclusão e criamos um parâmetro para que possa receber o nome do Gênero que será incluído. Veja as linhas a seguir:

```

prGenero := FbParameter.Create;
prGenero.ParameterName := 'GENERO';
DataAdapter.InsertCommand.Parameters.
Add(prGenero);
DataAdapter.InsertCommand.Parameters[0].
Value := txtGenero.Text.ToUpper;

```

Declaramos e criamos a variável *prGenero* que é do tipo *FbParameter*. Em seguida incluímos o parâmetro e o valor no *InsertCommand*. Tudo isso em função de nossa instrução de inclusão, veja:

```
INSERT INTO GENEROS (GENERO) VALUES (?)
```

Note o sinal de interrogação ao final de instrução. Para cada valor que precisamos receber em uma instrução *SQL* em ASP.NET incluímos o sinal. Veremos isso também no botão *Alterar*.

Voltando ao exemplo, chamamos o método *ExecuteNonQuery* e verificamos

se o mesmo é maior que 0 (“zero”) o que significa que a *Select* foi executada com sucesso. Para preenchermos novamente o *GridGenero* com os dados da tabela, repetimos o código do evento *Page\_Load*, habilitamos e desabilitamos componentes conforme a necessidade.

Para realizar a alteração (“Update”) de gêneros, faremos algo muito parecido com o código do botão *Incluir*. Veja a **Listagem 4** e implemente-a no evento *OnClick* do botão em questão. O código é muito semelhante, a diferença entre ambas, fiz questão de incluir em uma listagem separada, observe a **Listagem 5** que contém o trecho que comentarei.

Perceba que novamente mudamos o tipo de componente de comando usado no *DataAdapter*. Agora usamos *UpdateCommand* que executa comandos de *Update* no banco de dados. Como podemos perceber, o componente *DataAdapter* implementa quatro controles de acesso a dados. Sim, são quatro! Aqui estamos usando apenas *SelectCommand*,

*InsertCommand* e *UpdateCommand*, porém ainda temos a opção de usar o *DeleteCommand*, para comandos de exclusão.

Na explicação do botão *Incluir*, vimos como usar parâmetros no ASP.NET. Se observarmos com mais calma o código da **Listagem 4**, veremos que estamos usando dois parâmetros: *COD\_GENERO* e *GENERO*. Veja:

```
UPDATE GENEROS SET GENERO = ? WHERE (COD_
GENERO = ?)
```

Para usar esses parâmetros precisamos criar duas variáveis e inicializá-las, assim como podemos ver no trecho adiante.

```

prCod_Genero := FbParameter.Create;
prCod_Genero.ParameterName := 'COD_GENERO';
prGenero := FbParameter.Create;
prGenero.ParameterName := 'GENERO';

```

Por fim enviamos essas variáveis ao *UpdateCommand* e executamos. O restante do código faz o *reload* dos dados no *DataGrid*.

---

**Nota:** Para que possamos criar os componentes da paleta *Firebird Data Provider* em tempo de execução, é necessário declarar o *namespace FirebirdSql.Data.Firebird* ao *Uses* da página.

---

## Criando a tela de manutenção de vídeos

Finalizada a construção, codificação e testes da página de *Gêneros* veremos agora a criação da página de manutenção de vídeos que envolve a utilização de valores de outra tabela do banco de dados, a tabela *Videos*.

O desenvolvimento da página também se dá de forma bastante simples e utilizaremos mecanismos semelhantes ao processo anterior. Após a criação desta página, baseado nestes dois exemplos, faça você mesmo a criação das páginas de clientes e usuários do sistema.

Crie uma nova página no menu *File | New > Other > Delphi for .NET Projects > New ASP.NET Files > ASP.NET Page* e altere seu nome para “a\_generos.aspx”. Insira no corpo da página uma tabela de seis linhas e uma coluna com 700 *pixels* de largura onde iremos adicionar os componentes para o controle da nossa página. Na primeira linha da tabela arraste *uccontrole.ascx*. Escreva

na próxima linha o texto “Manutenção de Vídeos” para identificar a página e na linha seguinte adicione um componente *DataGrid* (“gridVideos”) e acesse no *Object Inspector* o item *Property Builder* posicionando-se na guia *Columns* onde deverão ser observadas algumas configurações.

Desmarque, no topo da janela, o item *Create columns automatically at run time*. Em seguida clique no item *Bound Column* em *Available columns* e envie-o *Selected columns*. Agora selecione-a e atribua o valor *TITULO* nas propriedades *Data Field* e *Header text*, onde receberemos o valor do campo na tabela *Videos* através do *result set* da *Select* que faremos. Lembre-se que a propriedade *Header text* é na verdade o texto que aparecerá para o usuário final de sua aplicação.

Agora insira uma nova coluna, ou seja, clique novamente em *Bound Column* e envie-o para a área *Selected columns*. Na propriedade *Header text* digite “Gênero” e em *Data Field* o valor *GENERO*. Repita os passos anteriores para o campo *DURACAO*, *COD\_VIDEO* e *COD\_GENERO*, porém esses dois últimos deverão ter a propriedade *Visible* desativada.

Por fim faremos a inclusão de uma coluna de botões, chamada de *Button Column*. Expanda o *Button Column* e selecione a opção *Select*. Envie-o para a direita e modifique a propriedade *Header text* e *Text* para “Alterar” e clique em *Ok* para confirmar todas as alterações.

Após isso, insira uma nova tabela com três linhas e duas colunas com 100% de largura, na linha logo abaixo. Digite o texto “Vídeo: ” na primeira célula e na coluna ao lado arraste um componente *TextBox* (“txtVideo”). Agora digite o texto “Gênero: ” na célula seguinte e arraste um componente *DropDownList* (“ddlGeneros”). Por fim, na última linha, utilize o texto “Duração: ” e adicione outro controle *TextBox* (“txtDuracao”).

Nas próximas linhas configure como na página de manutenção de gêneros. No código do botão *btnVoltar*, evento *OnClick*, digite o código a seguir:

```
Response.Redirect('a_principal.aspx');
```

Usaremos os mesmos métodos de conexão e listagem de dados do nosso *BD* usados na tela de *Gêneros*, ou seja, criaremos todos os objetos em *run time*.

Listagem 7. Código do botão Alterar

```
procedure TWebForm1.btnAlterar_Click(sender:
  System.Object; e: System.EventArgs);
var
  Conn : FbConnection;
  DataAdapter : FbDataAdapter;
  Ds : DataSet;
  Comand : FbCommand;
  prCod_Genero : FbParameter;
  prTitulo : FbParameter;
  prDuracao : FbParameter;
  prCod_Video : FbParameter;
begin
  Item := ddlGeneros.SelectedValue.ToString;
  { Criação dos objetos de conexão }
  Conn := FbConnection.Create;
  DataAdapter := FbDataAdapter.Create;
  Comand := FbCommand.Create;
  Conn.ConnectionString := strConexao;
  Conn.Open;
  DataAdapter.UpdateCommand := Comand;
  DataAdapter.UpdateCommand.Connection := Conn;
  DataAdapter.UpdateCommand.CommandText :=
    'UPDATE VIDEOS SET COD_GENERO =?, TITULO =?, '+
    'DURACAO =? WHERE (COD_VIDEO =?)';

  prCod_Genero := FbParameter.Create;
  prCod_Genero.FbDbType := FbDbType(9);
  prCod_Genero.ParameterName := 'COD_GENERO';
  DataAdapter.UpdateCommand.Parameters.Add(
    prCod_Genero);
  DataAdapter.UpdateCommand.Parameters[0].Value :=
    ddlGeneros.SelectedValue.ToString;

  prTitulo := FbParameter.Create;
  prTitulo.ParameterName := 'TITULO';
  DataAdapter.UpdateCommand.Parameters.Add(prTitulo);
  DataAdapter.UpdateCommand.Parameters[1].Value :=
    txtVideo.Text.ToUpper;

  prDuracao := FbParameter.Create;
  prDuracao.ParameterName := 'DURACAO';
  DataAdapter.UpdateCommand.Parameters.Add(prDuracao);
  DataAdapter.UpdateCommand.Parameters[2].Value :=
    txtDuracao.Text;

  prCod_Video := FbParameter.Create;
  prCod_Video.FbDbType := FbDbType(9);
  prCod_Video.ParameterName := 'COD_VIDEO';
  DataAdapter.UpdateCommand.Parameters.Add(
    prCod_Video);
  DataAdapter.UpdateCommand.Parameters[3].Value :=
    gridVideos.Items.Item[gridVideos.SelectedItem
    .ItemIndex].Cells[4].Text.ToString;

  if DataAdapter.UpdateCommand.ExecuteNonQuery > 0 then
  begin
    { Atribuição dos atributos de seleção dos dados }
    DataAdapter.SelectCommand := Comand;
    DataAdapter.SelectCommand.Connection := Conn;
    DataAdapter.SelectCommand.CommandText :=
      'SELECT VIDEOS.COD_VIDEO, VIDEOS.TITULO, '+
      'GENEROS.GENERO, VIDEOS.DURACAO, '+
      'GENEROS.COD_GENERO FROM GENEROS '+
      ' INNER JOIN VIDEOS ON '+
      ' (GENEROS.COD_GENERO = VIDEOS.COD_GENERO)';
    { Criação em memória do DataSet auxiliar }
    Ds := DataSet.Create;
    DataAdapter.Fill(Ds, 'Genero');
    try
      GridVideos.DataSource := Ds;
      GridVideos.DataBind;
      btnIncluir.Enabled := True;
      btnAlterar.Enabled := False;
      txtVideo.Text := '';
      txtDuracao.Text := '';
    finally
      Conn.Close;
    end;
  end;
end;
```



#### Listagem 8. Código do botão Incluir da tela de Vídeos

```
procedure TWebForm1.btnIncluir_Click(sender:
  System.Object; e: System.EventArgs);
var
  Conn : FbConnection;
  DataAdapter : FbDataAdapter;
  Ds : DataSet;
  Comand : FbCommand;
  prCod_Genero : FbParameter;
  prTitulo : FbParameter;
  prDuracao : FbParameter;
begin
  { Criação dos objetos de conexão }
  Conn := FbConnection.Create;
  DataAdapter := FbDataAdapter.Create;
  Ds := DataSet;
  Comand := FbCommand.Create;
  { Atribuição da string de conexão e abertura do BD }
  Conn.ConnectionString := strConexao;
  Conn.Open;
  { Atribuição dos atributos de seleção dos dados }
  DataAdapter.InsertCommand := Comand;
  DataAdapter.InsertCommand.Connection := Conn;
  DataAdapter.InsertCommand.CommandText :=
    'INSERT INTO VIDEOS (COD_GENERO, TITULO, DURACAO)
    VALUES (?, ?, ?)';

  prCod_Genero := FbParameter.Create;
  prCod_Genero.FbDbType := FbDbType(9);
  prCod_Genero.ParameterName := 'COD_GENERO';
  DataAdapter.InsertCommand.Parameters.Add(
    prCod_Genero);
  DataAdapter.InsertCommand.Parameters[0].Value :=
    ddlGeneros.SelectedValue.ToString;

  prTitulo := FbParameter.Create;
  prTitulo.ParameterName := 'TITULO';
  DataAdapter.InsertCommand.Parameters.Add(
    prTitulo);
  DataAdapter.InsertCommand.Parameters[1].Value :=
    txtVideo.Text.ToUpper;

  prDuracao := FbParameter.Create;
  prDuracao.ParameterName := 'DURACAO';
  DataAdapter.InsertCommand.Parameters.Add(prDuracao);
  DataAdapter.InsertCommand.Parameters[2].Value :=
    txtDuracao.Text;

  if DataAdapter.InsertCommand.ExecuteNonQuery > 0 then
  begin
    { Atribuição dos atributos de seleção dos dados }
    DataAdapter.SelectCommand := Comand;
    DataAdapter.SelectCommand.Connection := Conn;
    DataAdapter.SelectCommand.CommandText :=
      'SELECT VIDEOS.COD_VIDEO, VIDEOS.TITULO, ' +
      'GENEROS.GENERO, ' +
      'VIDEOS.DURACAO, GENEROS.COD_GENERO FROM ' +
      'GENEROS ' +
      ' INNER JOIN VIDEOS ON (GENEROS.COD_GENERO' +
      ' = VIDEOS.COD_GENERO)';
    { Criação em memória do DataSet auxiliar }
    Ds := DataSet.Create;
    DataAdapter.Fill(Ds, 'Genero');
    try
      GridVideos.DataSource := Ds;
      GridVideos.DataBind;
      btnIncluir.Enabled := True;
      btnAlterar.Enabled := False;
      txtVideo.Text := '';
      txtDuracao.Text := '';
    finally
      Conn.Close;
    end;
  end;
end;
```

Digite o código da **Listagem 6** ao evento *Page\_Load* da página. As únicas diferenças aqui são o nome do componente *DataGrid*, que será modificado para *GriVideos* e instrução *SQL* que estamos usando para trazer os dados. Precisamos listar todos os vídeos presentes na locadora e seus respectivos gêneros. Para isso usamos um *JOIN* entre as tabelas *Videos* e *Generos*. Para facilitar nossa programação, volte na página *a\_generos.aspx* e copie o código completo do evento *Page\_Load*. Após isso cole-o no *Page\_Load* de nossa página. Altere a instrução *SQL* como mostrado a seguir:

```
DataAdapter.SelectCommand.CommandText :=
  'SELECT VIDEOS.COD_VIDEO,
  VIDEOS.TITULO, GENEROS.GENERO, ' +
  'VIDEOS.DURACAO, GENEROS.COD_GENERO FROM
  GENEROS ' +
  ' INNER JOIN VIDEOS ON (
  GENEROS.COD_GENERO = VIDEOS.COD_GENERO)';
```

Em seguida modifique, no código-fonte, o nome do componente *DataGrid* de *GridGeneros* para *GridVideos*. Só há mais um detalhe que devemos nos lembrar. Inserimos um *DropDownList*("ddlGeneros") onde listaremos todos os gêneros da tabela para que possamos usá-lo na tela de manutenção de vídeos. Repare que entre o *try..finally* inclui um trecho de código que faz uma nova *Select* no banco, traz todos os gêneros e em seguida preenche a propriedade *Items* do *ddlGeneros*. O código completo do evento você pode conferir da **Listagem 6** como dito anteriormente.

A alteração de vídeos é simples, embora com uma quantidade de código extensa. A **Listagem 7** mostra como fazer a alteração do item selecionado. Basicamente o código é bem semelhante ao de inclusão. Assim como fizemos no botão de alteração de gêneros, também devemos fazer em vídeos, ou seja, criamos todos os componentes em *run time*, modificamos a instrução *SQL* para fazer o *Update* dos campos e trocamos o componente de comando no *DataAdapter* de *InsertCommand* para *UpdateCommand*. Também somos obrigados a criar um novo parâmetro, com o valor do campo *COD\_VIDEO* para que nossa instrução *SQL* saiba em qual registro do banco as alterações serão aplicadas. Veja nossa instrução a seguir:



```
UPDATE VIDEOS SET COD_GENERO=?, TITULO=?,
DURACAO=? WHERE (COD_VIDEO=?)
```

Por fim é necessário fazer com que os dados do vídeo selecionado sejam enviados aos controles de tela. Para isso digite o código a seguir no evento *ItemCommand* do *GridVideos*. Perceba que apenas atribuímos os valores das células selecionadas aos controles de tela. Depois habilitamos e desabilitamos os componentes necessários.

```
txtVideo.Text := e.Item.Cells.Item[0].Text;
txtDuracao.Text := e.Item.Cells.Item[2].Text;
ddlGeneros.SelectedValue := e.Item.Cells.Item[5].Text;
btnAlterar.Enabled := True;
btnIncluir.Enabled := False;
```

Faremos agora a última parte de desenvolvimento da página de manutenção de vídeos, a codificação do botão *Incluir*. O esquema é o mesmo do botão *Alterar*, por isso podemos copiar e colar o código do evento *OnClick* dele e colocar no botão *Incluir*. Depois faremos algumas modificações para adaptar o código às nossas necessidades de inclusão. Devemos trocar o componente *UpdateCommand* por *InsertCommand* e incluir a instrução *SQL* para inserção (**Listagem 8**).

Realizada toda a codificação do processo de inclusão e alteração da página de manutenção de vídeos, volte para a página *principal.aspx* e adicione ao evento *OnClick* do botão *btnVideos* o seguinte código:

```
Response.Redirect('a_videos.aspx');
```

Execute sua aplicação e faça a simulação de cadastro de novos vídeos e também da alteração do cadastro de vídeos já existentes. A partir da criação da página de gêneros e vídeos, baseando-se nestes exemplos crie uma página de cadastro e manutenção para usuários que serão os administradores do sistema e também



Figura 7. Sistema em execução

uma página para cadastros e manutenções de registros de clientes, levando em consideração que cada cliente poderá ter um usuário e senha para fazer suas próprias reservas *on-line*. Veja na **Figura 7** o sistema em execução.

## Conclusão

Vimos nesta primeira parte do artigo a criação e configuração da conexão com a base de dados. Também a criação das páginas de cadastro para gêneros e vídeos,

utilizando valores de outras tabelas.

Faça sua própria implementação de chaves de criptografia se preferir e construa funções e procedimentos para validação de registros cadastros e não permitindo redundâncias. Acompanhe no próximo artigo a criação da página de reservas locais, controle de usuários e clientes *logados*, reservas *on-line* e posterior efetivação da retirada do vídeo reservado. Grande abraço e até o próximo artigo. ●

