

Capítulo 12

Intraweb

A grande revolução, e por que não dizer, a agradável surpresa presente na nova versão do Delphi: Intraweb. Desenvolvido pela empresa Atozed Software, e rapidamente *adotado* pela Borland numa distribuição especial (existe uma versão mais poderosa), vem ganhando simpatizantes em todo o mundo. Diversas empresas estão desenvolvendo componentes para trabalhar em conjunto com o Intraweb, dando ainda mais poder a esta incrível tecnologia.

Desenvolver aplicações com o Intraweb é bastante agradável e prático e certamente vai encorajar excelentes desenvolvedores Delphi, que antes *torciam o nariz* para a tecnologia WebBroker, devido às dificuldades comuns encontradas no desenvolvimento de aplicações para Internet, como o aprendizado de HTML, JavaScript, alguma ferramenta aliada, como o DreamWeaver, entre outros.

Desenvolvedores Delphi, criem coragem para entrar no mundo Internet com o Intraweb. Digo isso a todos que chegaram nesse capítulo, passando pelos anteriores, ou até mesmo a quem abriu o livro aqui, neste tópico.

Algumas curiosidades




























A Borland vem trabalhando com extremo profissionalismo, apoiando diversas empresas parceiras em projetos de tecnologia para as suas ferramentas. Só para ter uma idéia disso tudo, incorporou na versão 7 diversas tecnologias, como o próprio Intraweb, Rave Report (fantástico gerador de relatórios, e forte substituto do QuickReport), RxLib agora vem no CD 2, Indy Components, já visto no capítulo 9, entre outras tecnologias. Se a tecnologia é boa, pode ter certeza de que será adotada nas próximas versões, e isso vem crescendo a cada dia. Sorte nossa!

Antes de colocar a mão-na-massa, vou apresentar seus principais componentes:








Os componentes estão divididos em quatro seções: *IWStandard*, *IWData*, *IWClientSide* e *IWControl*.







Componentes IWStandard

COMPONENTE	DESCRIÇÃO/USO
 TIWApplet	Para inserir uma Applet Java em sua página, oferecendo a configuração dos controles através do Object Inspector.
 TIWButton	Botão de formulário, com funções diversas, inclusive Submit.
 TIWCheckBox	Controles de CheckBox para formulário.
 TIWComboBox	ComboBox para seleção, com opções de preenchimento de lista.
 TIWEdit	Campo para edição com diversas propriedades de controle, como tamanho, se é requerido, tipo password, entre outras.
 TIWHRule	Linha horizontal.
 TIWImage	Para exibição de imagens estáticas ou dinâmicas. Transforma automaticamente a maioria dos formatos para JPEG.
 TIWImageFile	Utilizado para exibir imagens estáticas, armazenadas num arquivo.
 TIWLabel	Semelhante ao Label do Delphi.
 TIWLink	Cria um hyperlink, controlado através do evento OnClick.
 TIWList	Utilizado para apresentar listas de marcação ou numeradas.
 TIWListBox	Lista de opções que podem ser selecionadas. Possui um forte controle de propriedades.
 TIWMemo	Área de edição (assim como o Delphi), com múltiplas linhas.
 TIWMenu	Componente fantástico, que cria um menu a partir de um componente TMainMenu do Delphi. Muito bom mesmo. Agora suas aplicações Web terão um aspecto mais profissional.
 TIWRadioGroup	Conjunto de opções ao usuário, semelhante ao RadioGroup do Delphi.
 TIWTable	Amigos, fiquei impressionado com o poder deste componente. Semelhante ao StringGrid, com opção individual de controle de células. Muito bom mesmo.






COMPONENTE	DESCRIÇÃO/USO
 TIWTemplateProcessorHTML	Este componente permite associar um template HTML ao nosso formulário.
 TIWText	Apresenta textos com múltiplas linhas, sem suporte à edição.
 TIWTimer	Cria um Timer no lado servidor para disparar eventos.
 TIWTreeView	Apresenta um TreeView com diversos controles. Muito bom.
 TIWURL	Hyperlinks externos.
 TIWFile	Este componente quebra um galho incrível. Quem já tentou fazer rotinas de upload com a tecnologia WebBroker sabe do que estou falando. Era muito difícil, mas agora ficou muito fácil com este componente.
 TIWGrid	Grid sem vínculo com banco de dados.
 TIWRectangle	Apresenta figuras retangulares, com diversas opções de preenchimento.
 TIWRegion	Amigos, fiquei de boca aberta com os recursos deste componente. Tem a função de um container, como o Panel do Delphi. Fantástico!

Componentes IWData








COMPONENTE	DESCRIÇÃO/USO
 TIWDBCheckbox	CheckBox Data Ware, com diversas opções de configuração.
 TIWDBCombobox	ComboBox Data Ware, com incríveis propriedades, como tipo do cursor, ordenação, controle de scripts, entre outros.
 TIWDBEdit	DBEdit Data Ware, com todos os eventos do HTML, além de propriedades controladas pelo JavaScript.
 TIWDBGrid	DBGrid muito bacana e funcional, com diversos controles e eventos.
 TIWDBImage	DBImage Data Ware.
 TIWDBLabel	DBLabel Data Ware.
 TIWDBListbox	DBListBox com diversos controles e propriedades.

COMPONENTE	DESCRIÇÃO/USO
 TIWDBLookupCombobox	Fantástico componente semelhante ao DBLookupComboBox do Delphi. Quem já desenvolveu aplicações padrão WebBroker sabe o enorme trabalho que dá para fazer um LookupComboBox.
 TIWDBLookupListbox	Outra maravilha, semelhante ao DBLookupListBox do Delphi.
 TIWDBMemo	DBMemo com diversos controles e propriedades.
 TIWDBNavigator	Amigos, este componente é incrível, tem todos os controles do DBNavigator do Delphi e possui controle de mensagens de diversos níveis. Você poderá configurar a própria mensagem de confirmação para exclusão. Muito bom. Nota 10.
 TIWDBText	Semelhante ao DBText.
 TIWDBFile	Utilizado para associar um campo do tipo stream com a função de uploads de arquivo.

Componentes IWClientSide

COMPONENTE	DESCRIÇÃO/USO
 TIWCSLabel	Apresentação de campos de uma tabela do lado cliente (veja controles IWControl).
 TIWCSNavigator	Controle de Navegação dos dados de uma tabela do lado Cliente. Imagine você acessando uma tabela pré-configurada no lado cliente, e navegando à vontade.
 TIWDynamicChart	Gráfico com diversas opções, baseados em dados do lado cliente – embora seja possível alimentar com os dados do lado Servidor, pois não existe vínculo com DataWare, e sim com um IWClientDataSet (IW Control).
 TIWDynamicChartLegend	Legendas do gráfico, para ser utilizado em conjunto com o TIWDynamicChart.
 TIWDynGrid	Grid ligado ao IWClientDataSet (IW Control).

Componentes IWControl

COMPONENTE	DESCRIÇÃO/USO
 TIWClientDataSet	Controle DataSet para armazenar informações e disponibilizar aos controles IWClientSide.
 TIWClientDataSetDBLink	Controle DataSet no padrão DataWare, onde você deverá ligar diretamente a um DataSource. Isso é muito legal, pois o DataSource poderá coletar dados de diversas origens.
 TIWLayoutMgrForm	Controle de layout dos objetos de um formulário.
 TIWLayoutMgrHTML	Permite a edição do formulário no padrão HTML. Quer dar o seu toque pessoal? Então este é o componente certo.
 TIWModuleController	Habilita o controle do servidor para o padrão PageMode do Intraweb.
 TIWPageProducer	Com este componente você poderá utilizar toda a tecnologia Intraweb com WebSnap, e também aproveitar a tecnologia dos Page Producers do WebBroker.
 TIWStandAloneServer	Controle da aparência e outras propriedades da aplicação.

Antes de Começar

Este tópico é muito importante, pois precisamos conhecer alguns conceitos do Intraweb.

Na versão distribuída com o Delphi 7, o Intraweb coloca automaticamente o nome das *units iniciais bem como do projeto*. Isso pode dificultar um pouco no começo, mas em nosso caso iremos criar uma estrutura de diretórios com o intuito de organizar nossos exercícios com o Intraweb, e evitar conflitos.

Crie a seguinte estrutura de diretórios (*diagrama 12.1*)

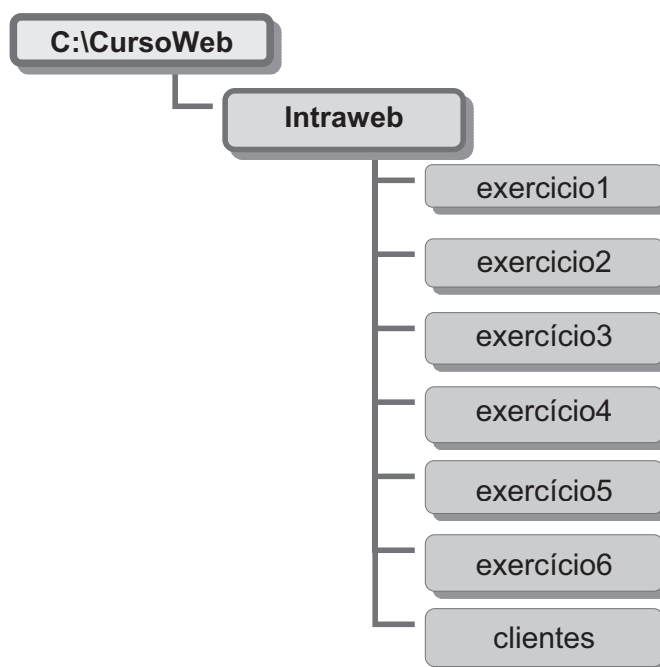
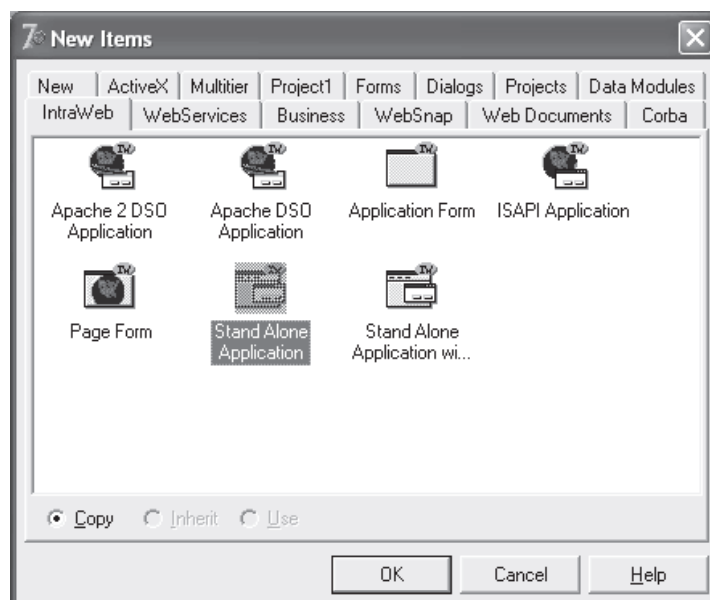


Diagrama 12.1

Primeiro Exemplo (controles Standard)

Vamos desenvolver nossa primeira aplicação, utilizando alguns componentes da seção *IWStandard*. Vamos escolher o padrão *Stand Alone* para facilitar a compreensão e o desenvolvimento.

Através das opções *File/New/Other...*, selecione a seção *Intraweb* e escolha o modelo *Stand Alone Application* (figura 12.1).

*Figura 12.1 Escolha do tipo da Aplicação*

Em seguida, selecione o diretório de nossa primeira aplicação, em nosso caso (c:\cursoweb\intraweb\exercicio1).

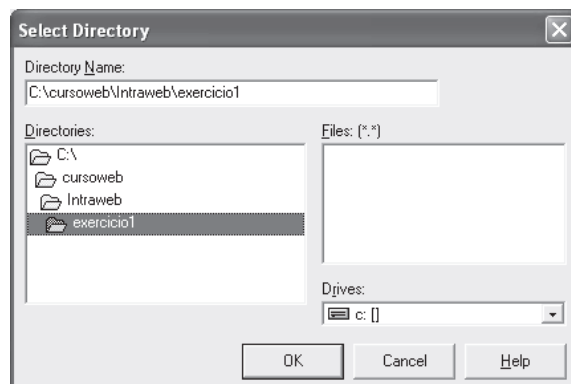


Figura 12.2 Seleção do diretório

Perceba que o Intraweb criou um módulo principal, com alguns controles (figuras 12.3 e 12.4).

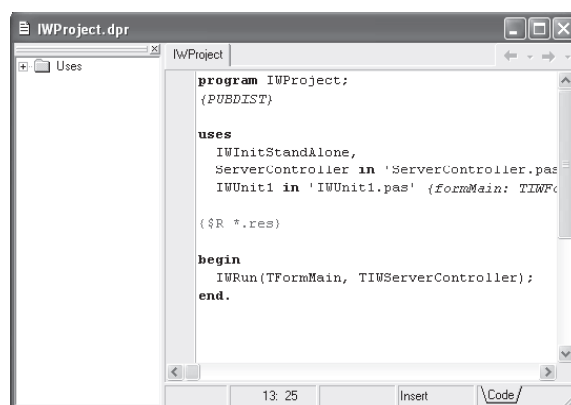


Figura 12.3 Módulo do Projeto

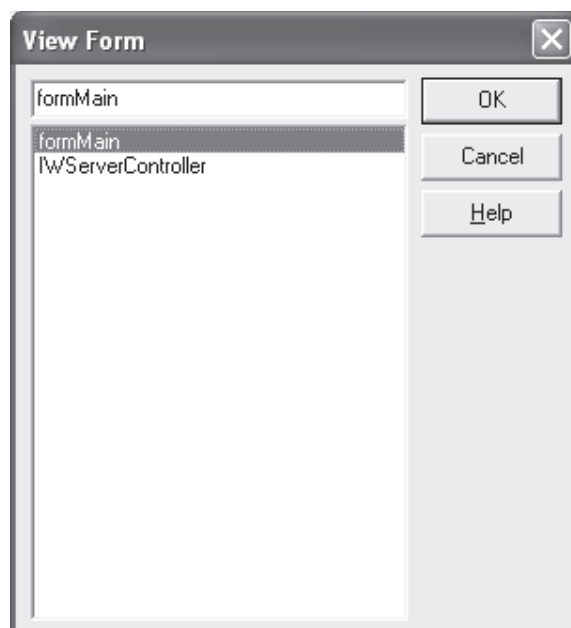


Figura 12.4 Formulários padrão da aplicação Stand Alone

O Intraweb cria a seguinte estrutura de arquivos para o tipo de aplicação *Stand Alone*:

- **IWProject.DPR** Projeto
- **IWUnit1.PAS** Unit principal, contendo o formulário principal da aplicação
- **ServerController.PAS** Controle principal da aplicação, com sessões de usuário, tipo de acesso, protocolos de segurança (SSL), browsers, entre outros.

Agora com o foco no formulário (selecione o formulário da unit IWUnit1), insira um componente do tipo *IWLabel* e altere sua propriedade *caption* para *Exercício 1*. A *figura 12.5* ilustra o formulário principal com o objeto *IWLabel1*.

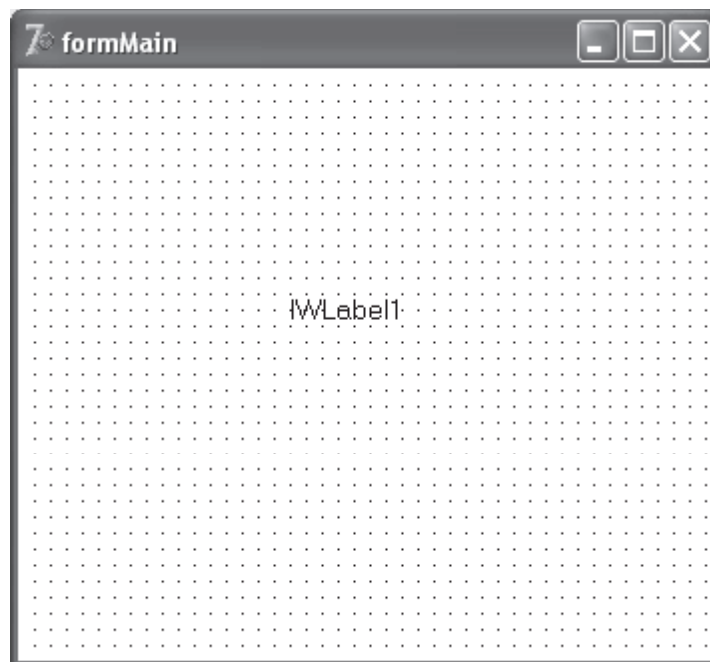


Figura 12.5 Formulário principal

Agora, meus amigos, para quem estava com saudades do famoso *F9* para executar e depurar as aplicações, o Intraweb retorna “aos bons tempos” e habilita esta opção. Aperte *F9* para executar nosso primeiro exercício. Deverá aparecer o servidor de testes do Intraweb, como ilustra a *figura 12.6*.

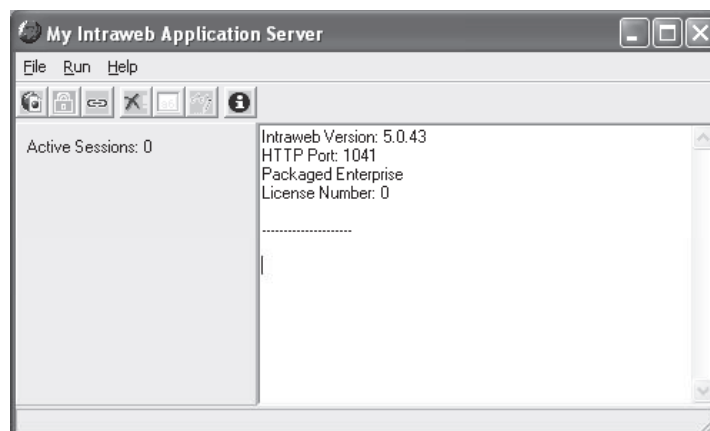


Figura 12.6 Servidor de Aplicações Intraweb

Repare que não existe nenhuma sessão ativa (Active Sessions). Para executar nossa aplicação, você poderá teclar novamente *F9*, ou pressionar o primeiro *speed button*, ou então acessar o menu *File/Run..* A *figura 12.7* ilustra o resultado do nosso primeiro exercício.



Figura 12.7 Resultado do primeiro exercício

Tente pressionar *F9* algumas vezes no *Intraweb Server Application*, e repare que são criadas novas sessões (figura 12.8).

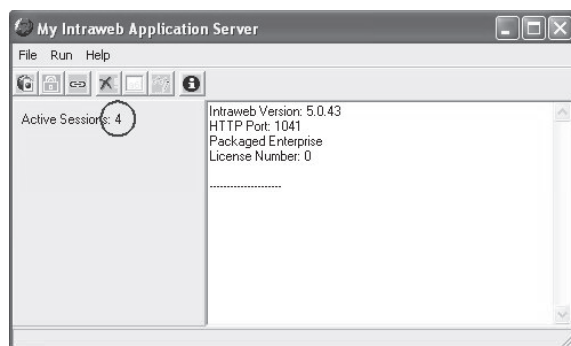


Figura 12.8 Sessões ativas

O mais interessante de tudo isso, é que o Intraweb possui um forte controle de sessões. Repare na figura 12.9 que o Intraweb utiliza um número extenso, definindo a sessão do usuário. Este controle é bastante robusto e facilita muito a “vida” do desenvolvedor no controle de sessões.



Figura 12.9 Controle de sessões

Encerre o servidor de aplicações, e retorne ao Delphi.

Vejamos agora, todo o código criado pelo Intraweb em nossa primeira aplicação.

Listagem 12.1 Unit IWUnit1.

```
unit IWUnit1;
{PUBDIST}

interface

uses
  IWAppForm, IWApplication, IWTypes, Classes, Controls, IWControl,
  IWCompLabel;

type
```

```

TformMain = class(TIWebAppForm)
    IWLabel1: TIWebLabel;
public
end;

implementation
{$R *.dfm}

uses
    ServerController;

end.

```

Basicamente define a classe *TformMain* com o objeto *IWLabel1*.

Para analisar mais profundamente nosso primeiro exercício, clique com o botão direito do mouse no formulário principal e selecione a opção *Preview*. A *figura 12.10* ilustra o *Preview* do *Intraweb*.

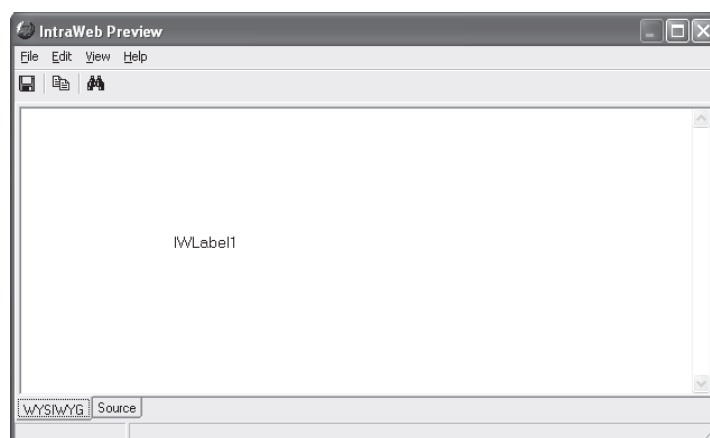


Figura 12.10 Preview do IntraWeb

É muito interessante essa versatilidade do *Intraweb*. Agora, selecione a seção *Source* logo abaixo do *Preview* e repare o código *HTML* gerado, inclusive com *CSS* (*figura 12.11*).

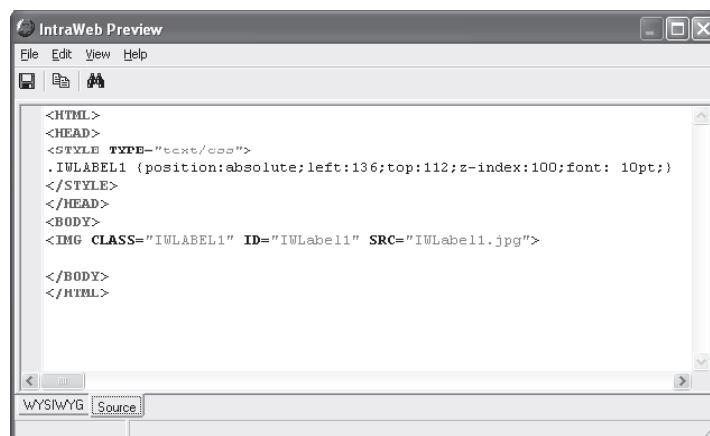


Figura 12.11 Código HTML

Como o código é dinâmico, o modelo gerado e visto na *figura 12.11* é apropriado para o modelo de servidor *IntraWeb Server Application*, no modo *Preview*. Veja a versão HTML gerada fora do *Preview*, mas ainda no servidor *Intraweb*.

Listagem 12.2 HTML

```

<html><head>
<style type="text/css">
.IWLABEL1CSS {position:absolute;left:136;top:112;z-index:100;font: 10pt;}

</style>
<script language="Javascript1.2">
function FormDefaultSubmit()
{return false;}
function Validate() {return true;}
var GURLBase="";
var GAppID="08A197007D9506F23B52E240";
history.go(1);
var IWLABEL1IWCL;

function InitIWCLObjects() {
IWLABEL1IWCL = new CreateIWCLObject(IWCLForm, "IWLABEL1", "IWLABEL1IWCL");
if (IWLABEL1IWCL.Item != null) {
    IWLABEL1IWCL.SetAlign(alNone);
    IWLABEL1IWCL.SetAnchors(new CreateAnchors(true, false, true, false));
}
}

Body_OnResize();

}
function Initialize() {
InitSubmitter();
StaticInit();
if (document.body.leftMargin < 0 && document.body.topMargin < 0) {
    document.body.leftMargin = 0;
    document.body.topMargin = 0;
}
InitRects(349, 296);
InitIWCLObjects();

}
</script>

<meta name="GENERATOR" content="IW5.0.43 Serial 0">
<script language=Javascript src="/js/IWCommon.js_5.0.43"></script>

<script language=Javascript src="/js/IWCL.js_5.0.43"></script>

<script language=Javascript src="/js/IWCSDData.js_5.0.43"></script>

<script language=Javascript src="/js/IWExplorer.js_5.0.43"></script>

</head>
<body onload="Initialize()" onblur="GSubmitting = false;" onresize="return
Body_OnResize();"><form onsubmit="return FormDefaultSubmit();" name="SubmitForm"
action="/EXEC/1/08A197007D9506F23B52E240" method="POST">
    <input type="HIDDEN" name="IW_Action">
    <input type="HIDDEN" name="IW_ActionParam">
</form>

```

```
<span id="IWLABEL1" class="IWLABEL1CSS">IWLabel1</span>
</body>
</html>
```

Complicado, não acham? Podem ficar tranquilos, pois dificilmente teremos que dar manutenção no código HTML. A idéia é justamente essa, utilizar o RAD (Rapid Application Development) do Intraweb para criar e dar manutenção às nossas aplicações.

Só para finalizar a parte de códigos deste tópico, vamos analisar a *unit ServerController*.

Listagem 12.3 Unit ServerController

```
unit ServerController;
{PUBDIST}

interface

uses
  SysUtils, Classes, IWServerControllerBase,
  // For OnNewSession Event
  IWApplication, IWAppForm;

type
  TIWServerController = class(TIWServerControllerBase)
    procedure IWServerControllerBaseNewSession(ASession: TIWApplication;
      var VMainForm: TIWAppForm);
  private
  public
  end;

  // This is a class which you can add variables to that are specific to the user.
  Add variables
  // to this class instead of creating global variables. This object can references
  by using:
  //   UserSession
  // So if a variable named UserName of type string is added, it can be referenced by
  using:
  //   UserSession.UserName
  // Such variables are similar to globals in a normal application, however these
  variables are
  // specific to each user.
  //
  // See the IntraWeb Manual for more details.
  TUserSession = class
  public
  end;

// Procs
function UserSession: TUserSession;

implementation
{$R *.dfm}

uses
  IWInit;
```

```
function UserSession: TUserSession;
begin
  Result := TUserSession(RWebApplication.Data);
end;

procedure TIWServerController.IWServerControllerBaseNewSession(
  ASession: TIWApplication; var VMainForm: TIWAppForm);
begin
  ASession.Data := TUserSession.Create;
end;

end.
```

Amigos, analisando o código da *unit ServerController*, podemos perceber a flexibilidade no controle de sessões por usuário. É possível criar variáveis similares às globais, mas com instâncias por sessão, ou seja, cada usuário terá a sua própria variável. Eu explico melhor. Vamos imaginar uma variável que totaliza o valor da compra, ou até mesmo uma matriz, contendo diversas informações sobre a compra atual, e poder acessar em nível de usuário, ou melhor, por sessão, cada variável. Veja o cenário:

Usuário **1**
Sessão 08A197007D9506F23B52E240
Valor 300

Usuário **2**
Sessão E4949A0062C7BBC23C52E240
Valor 180

Usuário **3**
Sessão D0679B0010B757C63C52E240
Valor 180

Repare que cada sessão possui uma identificação diferente. Imagine quando o usuário solicitar a finalização da compra, onde é necessário apresentar o resumo e o valor total. Com o IntraWeb é extremamente simples; basta acessar o conteúdo das variáveis definidas no *ServerController*. Mas como ele faz isso? Uma mistura de cookies com persistência.

Com isso concluímos nosso primeiro exercício e conhecemos alguns conceitos do IntraWeb.

Segundo Exemplo (controles Standard – 2ª. parte)

Vamos aproveitar o embalo do primeiro exemplo e conhecer alguns componentes do *IW Standard*.

Através das opções *File/New/Other...*, selecione a seção IntraWeb e escolha o modelo *Stand Alone Application* (figura 12.12).

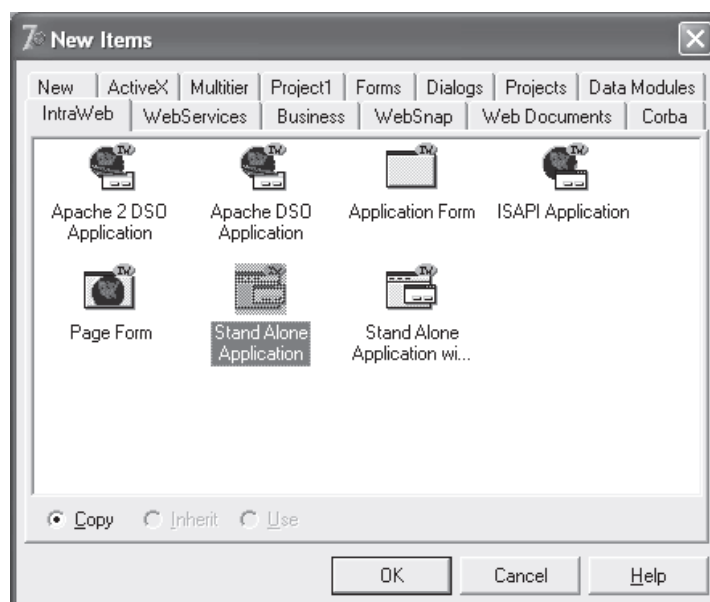






Figura 12.12 Escolha do tipo da Aplicação

Em seguida, selecione o diretório de nossa segunda aplicação, em nosso caso (c:\cursoweb\intraweb\exercicio2). Grave toda a aplicação, e insira os componentes que seguem:

OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLabel1	Name	IWLabel1
	Caption	Segundo Exemplo
	Left	120
	Top	16

OBJETO		
	TIWEdit	
Objeto	Propriedade	Valor
edNome	Name	edNome
	Left	72
	MaxLength	30
	Top	56
	Width	225

OBJETO		
	TIWButton	
Objeto	Propriedade	Valor
btConfirma	Name	btConfirma
	Caption	Confirma
	Left	136
	Top	104

OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
lbMensagem	Name	lbMensagem
	Caption	deixe em branco
	Left	24
	Top	152
	Visible	False

A figura 12.13 ilustra nosso formulário.



Figura 12.13 Formulário segundo exemplo

Agora vamos codificar um pouco. Selecione o objeto *btConfirma* (botão) e insira o código que segue, no evento *OnClick*.

```
lbMensagem.Caption:='Seja bem vindo(a) '+edNome.Text;
lbMensagem.Visible:=True;
```

O mais interessante de tudo isso, é que estamos colocando código *Object Pascal* puro, ou melhor, *Delphi Language* como é chamado agora.

Execute a aplicação, e seguindo o mesmo procedimento do primeiro exemplo, no *Intraweb Server Application*, pressione novamente a tecla *F9*. A *figura 12.14* ilustra o resultado do nosso segundo exemplo.

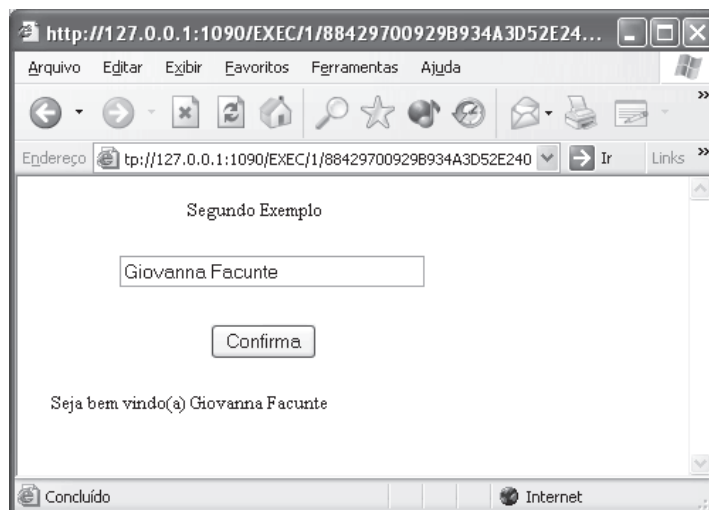


Figura 12.14 Resultado segundo exemplo.

Muito fácil, não é?

Listagem 12.4 Unit *IWUnit1* (segundo exemplo)

```
unit IWUnit1;
{PUBDIST}

interface

uses
  IWAppForm, IWApplication, IWTypes, IWCompEdit, Classes, Controls,
  IWControl, IWCompLabel, IWCompButton;

type
  TformMain = class(TIWAppForm)
    IWLabel1: TIWLabel;
    edNome: TIWEdit;
    btConfirma: TIWButton;
    lbMensagem: TIWLabel;
    procedure btConfirmaClick(Sender: TObject);
  public
  end;

implementation
{$R *.dfm}

uses
  ServerController;

procedure TformMain.btConfirmaClick(Sender: TObject);
begin
  lbMensagem.Caption:='Seja bem vindo(a) '+edNome.Text;
  lbMensagem.Visible:=true;
end;

end.
```


Terceiro Exemplo (controles Standard – 3ª. parte)

Agora que sabemos que o IntraWeb trabalha com *Delphi Language* pura, vamos abusar um pouco da programação.

Através das opções *File/New/Other...*, selecione a seção IntraWeb e escolha o modelo *Stand Alone Application* (figura 12.15).

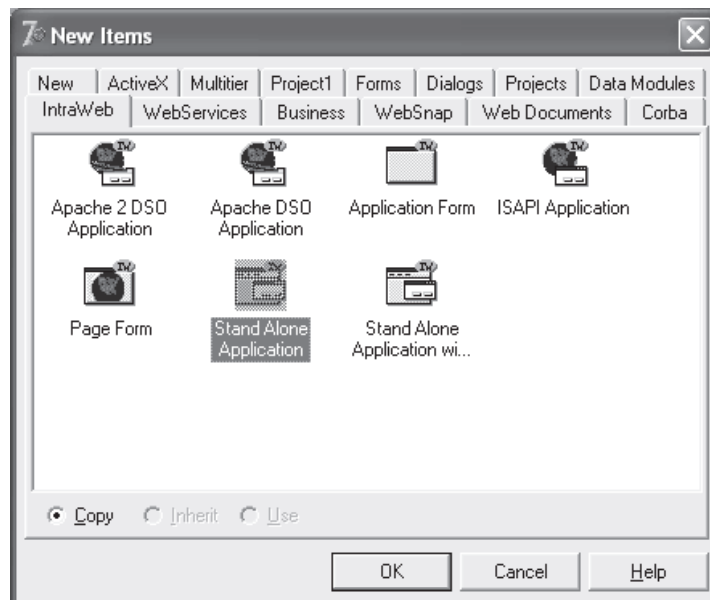






Figura 12.15 Escolha do tipo da Aplicação


Em seguida, selecione o diretório de nossa segunda aplicação, em nosso caso (c:\cursoweb\intraweb\exercicio3). Insira a **unit SysUtils**, na cláusula *uses* da aplicação, para que possamos fazer algumas operações. Grave toda a aplicação, e insira os componentes que seguem:


OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLabel1	Name	IWLabel1
	Caption	Terceiro Exemplo
	Left	112
	Top	16


OBJETO		
	TIWHRule	
Objeto	Propriedade	Valor
IWHRule1	Left	56
	Top	40
	Width	225


OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLabel2	Name	IWLabel2
	Caption	Insira dois números e selecione a operação
	Left	56
	Top	56


OBJETO		
	TIWEdit	
Objeto	Propriedade	Valor
edN1	Name	edN1
	Left	56
	Text	(deixe em branco)
	Top	80


OBJETO		
	TIWEdit	
Objeto	Propriedade	Valor
edN2	Name	edN2
	Left	56
	Text	(deixe em branco)
	Top	112


OBJETO		
	TIWHRule	
Objeto	Propriedade	Valor
IWHRule1	Left	56
	Top	144
	Width	73

OBJETO		
	TIWEdit	
Objeto	Propriedade	Valor
edTotal	Name	edTotal
	Left	56
	Text	(deixe em branco)
	Top	152

OBJETO		
	TIWButton	
Objeto	Propriedade	Valor
btAdicao	Name	btAdicao
	Caption	+
	Left	144
importante >>>	Tag	1
	Width	25

OBJETO		
	TIWButton	
Objeto	Propriedade	Valor
btSubtracao	Name	btSubtracao
	Caption	-
	Left	176
importante >>>	Tag	2
	Width	25

OBJETO		
	TIWButton	
Objeto	Propriedade	Valor
btMultiplicacao	Name	btMultiplicacao
	Caption	x
	Left	208
importante >>>	Tag	3
	Width	25

OBJETO		
	TIWButton	
Objeto	Propriedade	Valor
btDivisao	Name	btDivisao
	Caption	:
	Left	240
importante >>>	Tag	4
	Width	25

A figura 12.16 ilustra nosso formulário.

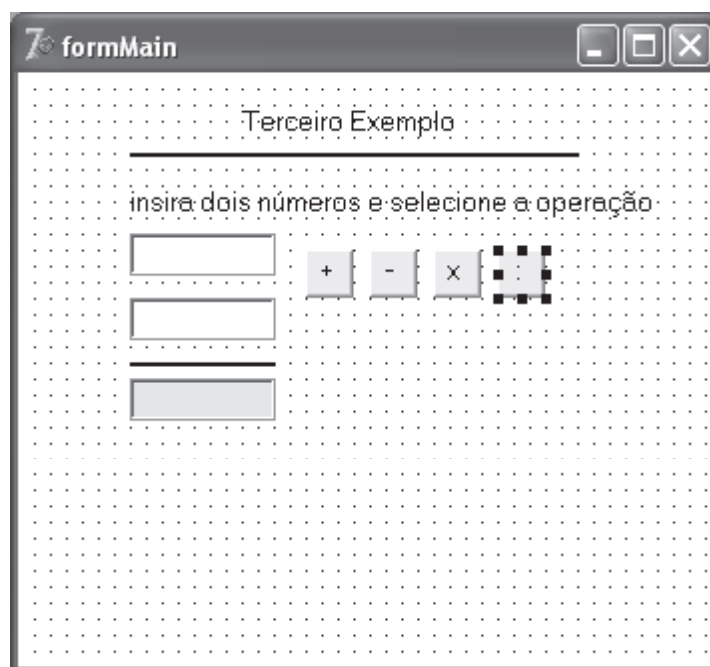


Figura 12.16 Formulário terceiro exemplo

Agora vamos codificar o botão *btAdicao*. No evento *OnClick* do botão, insira o código que segue:

```
var
  n1,n2,total:single;
  operacao:integer;

begin
  try
    n1:=StrtoFloat(edN1.Text);
    n2:=StrtoFloat(edN2.Text);
    total:=0;
    operacao:=(Sender as TIWButton).Tag;
    case operacao of
      1:total:=n1+n2;
      2:total:=n1-n2;
      3:total:=n1*n2;
      4:total:=n1/n2;
    end;
  end;
```

```

edTotal.Text:=FloatToStr(total);
except
  on EZeroDivide do WebApplication.ShowMessage('Divisão por zero !');
  on EOverflow do WebApplication.ShowMessage('Aconteceu Overflow !');
  //
end;

```

Antes de analisar o código, selecione os demais botões e associe o código do evento *OnClick*, como ilustra a *figura 12.17*.

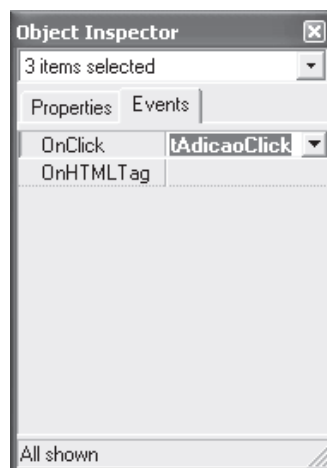


Figura 12.17 Associação do evento *OnClick* aos demais botões.

Vamos analisar o código.

```

var
  n1,n2,total:single;
  operacao:integer;

```

Neste primeiro bloco estamos declarando variáveis para execução da operação matemática (*n1*, *n2* e *total*), bem como uma variável auxiliar (operação) com o intuito de associar o botão que está chamando o evento. Em cada botão, definimos valores diferentes para a propriedade *TAG*, que será facilmente assimilada por nossa aplicação.

```

try
  n1:=StrtoFloat(edN1.Text);
  n2:=StrtoFloat(edN2.Text);
  total:=0;
  operacao:=(Sender as TIWButton).Tag;

```

Em seguida, iniciamos um bloco protegido (*try*) e convertemos o conteúdo dos campos de edição *edN1* e *edN2*, para o tipo *float*. Neste mesmo bloco estamos iniciando a variável *total* e atribuindo à variável *operacao*, o valor contido na propriedade *TAG* de um objeto do tipo *TIWButton*, que em nosso caso, são representadas pelos botões *btAdicao*, *btSubtracao*, *btMultiplicacao* e *btDivisao*.

```

case operacao of
  1:total:=n1+n2;
  2:total:=n1-n2;
  3:total:=n1*n2;
  4:total:=n1/n2;
end;

```

Neste bloco, fazemos as devidas operações de acordo com o botão pressionado.

```

edTotal.Text:=FloatToStr(total);

```

Em seguida, atribuímos ao objeto *edTotal* o resultado da operação.

```
except
    on EZeroDivide do WebApplication.ShowMessage('Divisão por zero !');
    on EOverflow do WebApplication.ShowMessage('Aconteceu Overflow !');
    //
end;
```

Por fim, tratamos duas exceções: *EZeroDivide* (divisão por zero), e *EOverflow* (Overflow na operação). Repare que em caso de erro, estamos apresentando mensagens ao usuário. É muito parecido com o famoso *Application.ShowMessage*. Neste caso, utilizamos o *WebApplication.ShowMessage*. As figuras 12.18, 12.19 e 12.20 ilustram nossa aplicação em tempo de execução.

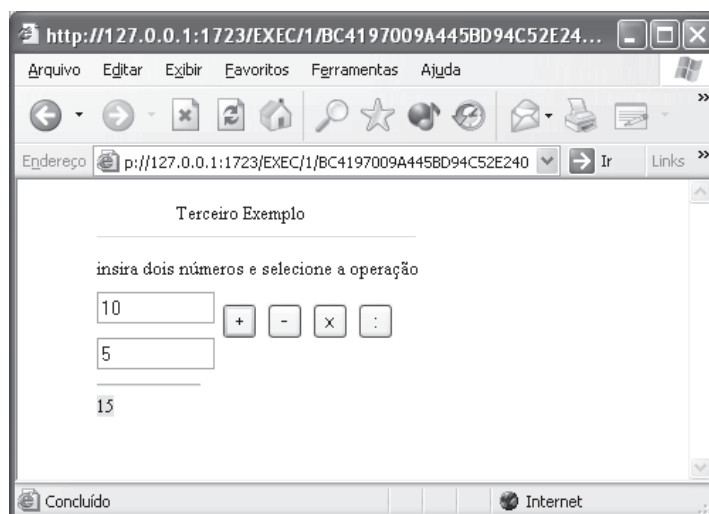


Figura 12.18 Operação de adição

A figura 12.18 ilustra o sucesso da operação de adição.

Em seguida temos nossa primeira mensagem de erro. A figura 12.19 ilustra um erro de divisão por zero.

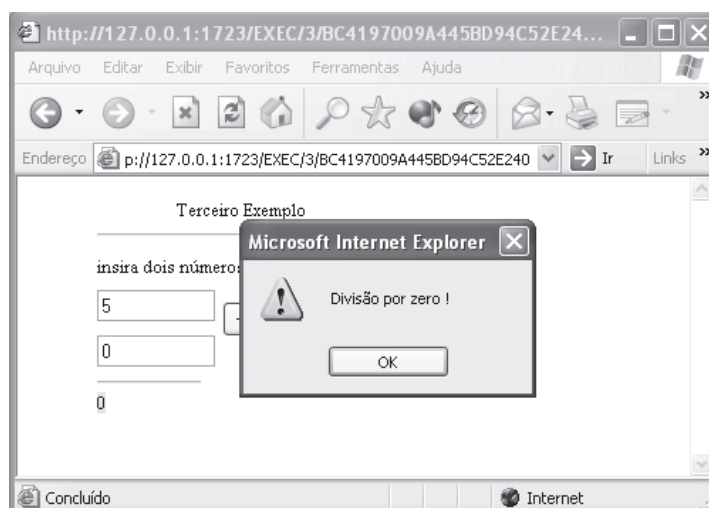


Figura 12.19 Divisão por zero

Em seguida temos um erro de *Overflow* (figura 12.20).

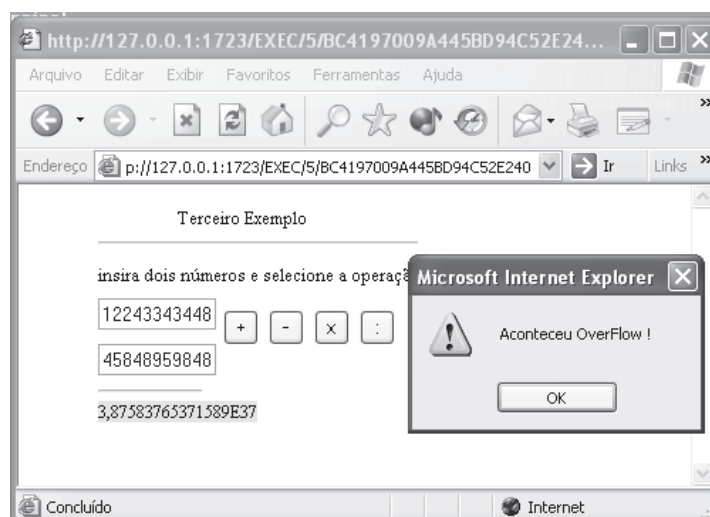


Figura 12.20 Overflow

Listagem 12.5 Unit IWUnit1.pas (exemplo 3)

```

unit IWUnit1;
{ PUBDIST }

interface

uses
  IWAppForm, IWApplication, IWTypes, IWHTMLControls, Classes, Controls,
  IWControl, IWCompLabel, IWCompEdit, IWCompButton, SysUtils;

type
  TFormMain = class(TIWAppForm)
    IWLabel1: TIWLabel;
    IWHRule1: TIWHRule;
    edN1: TIWEdit;
    edN2: TIWEdit;
    btAdicao: TIWButton;
    btSubtracao: TIWButton;
    btMultiplicacao: TIWButton;
    btDivisao: TIWButton;
    IWLabel2: TIWLabel;
    edTotal: TIWEdit;
    IWHRule2: TIWHRule;
    procedure btAdicaoClick(Sender: TObject);
  public
  end;

implementation
{$R *.dfm}

uses
  ServerController;

procedure TFormMain.btAdicaoClick(Sender: TObject);
var
  n1, n2, total: single;
  operacao: integer;

```

```

begin
  try
    n1:=StrtoFloat(edN1.Text);
    n2:=StrtoFloat(edN2.Text);
    total:=0;
    operacao:=(Sender as TIWButton).Tag;
    case operacao of
      1:total:=n1+n2;
      2:total:=n1-n2;
      3:total:=n1*n2;
      4:total:=n1/n2;
    end;
    edTotal.Text:=FloatToStr(total);
  except
    on EZeroDivide do WebApplication.ShowMessage('Divisão por zero !');
    on EOverflow do WebApplication.ShowMessage('Aconteceu Overflow !');
    //
  end;
end;
end.

```

Quarto Exemplo (controles Standard – 4ª. parte)

No quarto exemplo, vamos desenvolver uma rotina para *uploads* de arquivos. Como disse anteriormente, este processo era muito trabalhoso com o Web-Broker e agora ficou muito mais fácil. Com pouquíssimas linhas de código, na realidade apenas uma linha faz o serviço, mas em nosso caso faremos um tratamento de erro. Vamos desenvolver tal rotina.

Através das opções *File/New/Other...*, selecione a seção *Intraweb* e escolha o modelo *Stand Alone Application* (figura 12.21).

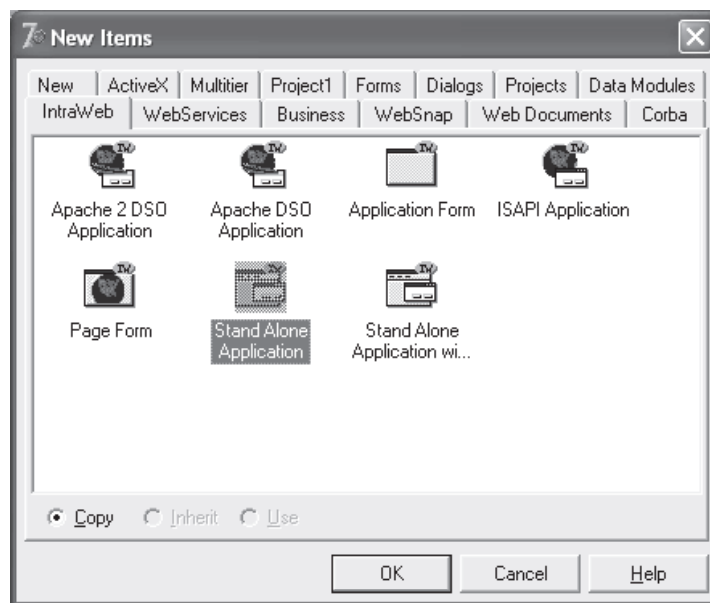






Figura 12.21 Escolha do tipo da Aplicação


Em seguida, selecione o diretório de nossa segunda aplicação, em nosso caso (c:\cursoweb\intraweb\exercicio4). Grave toda a aplicação, e insira os componentes que seguem:

OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLabel1	Name	IWLabel1
	Caption	Quarto Exemplo
	Left	112
	Top	16

OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLabel2	Name	IWLabel2
	Caption	Selecione o arquivo para Upload
	Left	32
	Top	48

OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IbMensagem	Name	IbMensagem
	Caption	Mensagem
	Left	32
	Top	168
	Font.Color	clRed

OBJETO		
	TIWFile1	
Objeto	Propriedade	Valor
IWFile1	Name	IWFile1
	Left	32
	Top	72
	Width	200

OBJETO		
	TIWButton1	
Objeto	Propriedade	Valor
IWButton1	Name	IWButton1
	Caption	UpLoad
	Left	32
	Top	104
	Width	75

A figura 12.22 ilustra o nosso formulário.

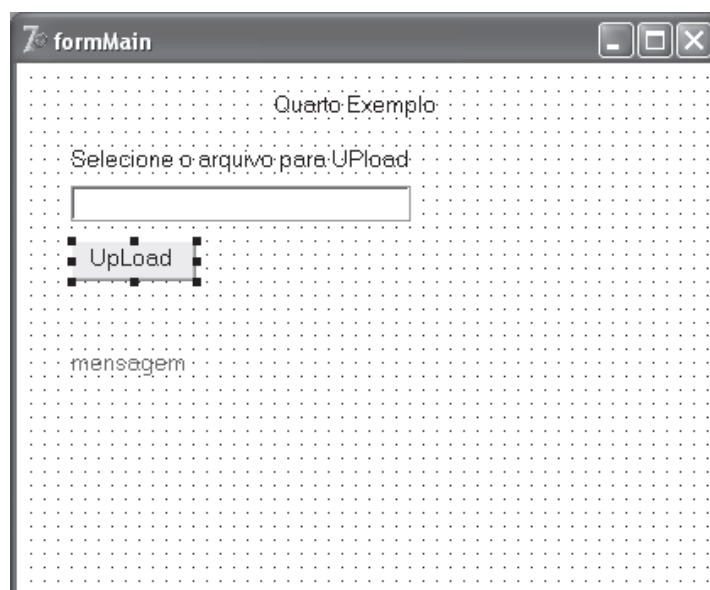


Figura 12.22 Formulário para UpLoad

Agora vamos codificar a rotina de *UpLoad*. No evento *OnClick* do objeto *IWButton1*, coloque o código que segue:

```
try
  IWFile1.SaveToFile(ExtractFilePath(ParamStr(0)) + IWFile1.FileName);
  lbMensagem.Caption:='O arquivo'+IWFile1.FileName+' foi gravado no diretório
'+#13#10+ ExtractFilePath(ParamStr(0))+' com êxito';
  lbMensagem.Visible := True;
except
  lbMensagem.Caption:='Houve um problema com a rotina de UPLOAD';
  lbMensagem.Visible := True;
end;
```

Amigos, a linha em negrito (**IWFile1.SaveToFile...**), é a única linha necessária para fazer o *UpLoad* do arquivo. Em nosso caso estamos fazendo uma rotina de tratamento de erros e exibindo a mensagem do resultado da operação. Caso seja bem sucedida, apresenta o nome do arquivo e em qual diretório foi gravado no servidor. Em caso de erro, apresenta apenas uma mensagem : “**Houve um problema...**”.

A figura 12.23 ilustra nossa aplicação sendo executada.

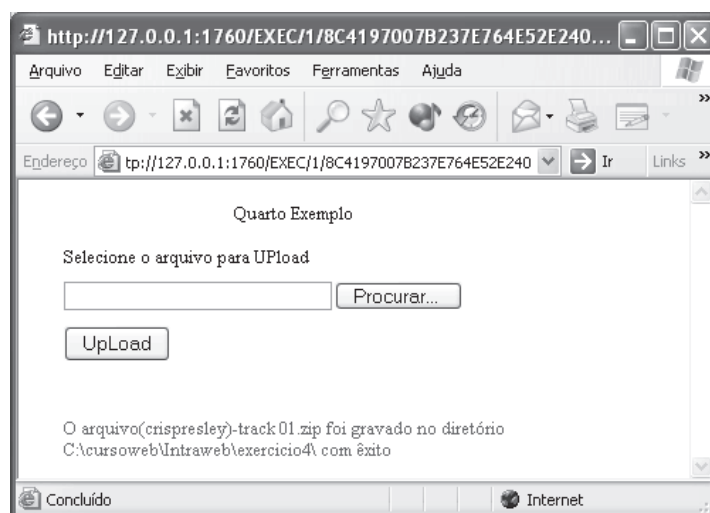


Figura 12.23 Arquivo sendo gravado no diretório da aplicação.

É muito fácil, não é?

Dica



Você poderá configurar o diretório que receberá os arquivos. Normalmente numa Intranet, existe um servidor de arquivos, onde sua aplicação poderá gravar as informações provenientes da rotina de *UpLoad*. Veja o exemplo:

```
IWFile1.SaveToFile('F:\arquivos\' + IWFile1.FileName);
```

Listagem 12.6 IWUnit1.pas (Quarto Exemplo)

```
unit IWUnit1;
{PUBDIST}

interface

uses
  IWAppForm, IWApplication, IWTypes, Classes, Controls, IWControl,
  IWCompLabel, IWCompEdit, SysUtils, IWCompButton;

type
  TFormMain = class(TIWAppForm)
    IWLabel1: TIWLabel;
    IWFile1: TIWFile;
    IWLabel2: TIWLabel;
    lbMensagem: TIWLabel;
    IWButton1: TIWButton;
    procedure IWButton1Click(Sender: TObject);
  public
  end;

implementation
{$R *.dfm}
```

```

uses
  ServerController;

procedure TFormMain.IWButton1Click(Sender: TObject);
begin
  try
    IWFile1.SaveToFile(ExtractFilePath(ParamStr(0)) + IWFile1.FileName);
    lbMensagem.Caption:='O arquivo'+IWFile1.FileName+' foi gravado no diretório
'+#13#10+ ExtractFilePath(ParamStr(0))+' com êxito';
    lbMensagem.Visible := True;
  except
    lbMensagem.Caption:='Houve um problema com a rotina de UPLOAD';
    lbMensagem.Visible := True;
  end;
end;
end.

```

Quinto Exemplo (controles Standard – 5ª. parte)

No quinto exemplo, vamos aprender a trabalhar com múltiplos formulários. O conceito é bem parecido com as aplicações *desktop*, com criação e destruição de formulários dinamicamente. Através das opções *File/New/Other...*, selecione a seção *Intraweb* e escolha o modelo *Stand Alone Application* (figura 12.24).

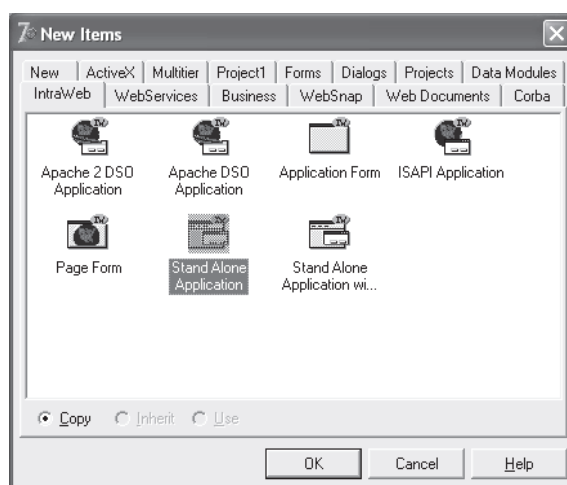






Figura 12.24 Escolha do tipo da Aplicação

Em seguida, selecione o diretório de nossa segunda aplicação, em nosso caso (c:\cursoweb\intraweb\exercicio5). Grave toda a aplicação, e insira os componentes que seguem:

OBJETO		
		
TIWLabel		
Objeto	Propriedade	Valor
IWLabel1	Name	IWLabel1
	Caption	Quinto Exemplo
	Left	112
	Top	16

OBJETO		
	TIWButton	
Objeto	Propriedade	Valor
IWButton1	Name	IWButton1
	Caption	Formulário 2
	Left	120
	Top	56
	Width	115

OBJETO		
	TIWButton	
Objeto	Propriedade	Valor
IWButton2	Name	IWButton2
	Caption	Formulário 3
	Left	225
	Top	124
	Width	115

OBJETO		
	TIWEdit1	
Objeto	Propriedade	Valor
edNome	Name	edNome
	Left	40
	Text	(deixe em branco)
	Top	126
	Width	175

Agora vamos criar um segundo formulário. Através das opções *File/New/Other...*, selecione a seção *Intraweb* e escolha o modelo *Application Form* (figura 12.25).

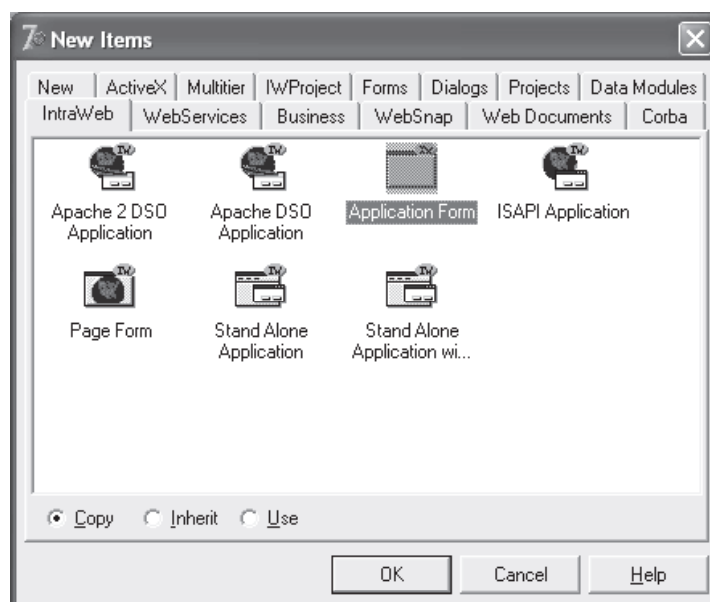




Figura 12.25 Criando um novo formulário

Altere a propriedade *Name* do formulário para *Form2*. Grave a *unit* com o nome *un_formulário2*. Insira os componentes que seguem no *Form2*.

OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLabel1	Name	IWLabel1
	Caption	Formulário 2
	Left	136
	Top	36

OBJETO		
	TIWButton	
Objeto	Propriedade	Valor
IWButton1	Name	IWButton1
	Caption	Fecha
	Left	136
	Top	96
	Width	75

Vamos codificar este formulário. No evento *OnClick* do objeto *IWButton1*, insira o seguinte código

```
Hide;
```

No código que acabamos de inserir, estamos “*escondendo*” o formulário, e retornando à origem. Para entender o funcionamento, vamos codificar o nosso formulário principal.

Insira a *unit un_formulario2* na cláusula *uses* do formulário principal. Agora vamos codificar o evento *OnClick* do objeto *IWButton1* do formulário principal.

```
var
Form2:TForm2;
begin
  Form2 := TForm2.Create(WebApplication);
  Form2.Show;
end;
```

Estamos fazendo uma operação bastante simples. Primeiro declaramos uma variável do tipo *TForm2* (classe herdada da *TForm2* que está contida na *unit un_formulario2*).

```
var
Form2:TForm2;
```

Em seguida instanciamos o objeto.

```
Form2 := TForm2.Create(WebApplication);
```

E por fim, apresentamos o objeto.

```
Form2.Show;
```

As figuras 12.26 e 12.27 ilustram nossos dois primeiros formulários.

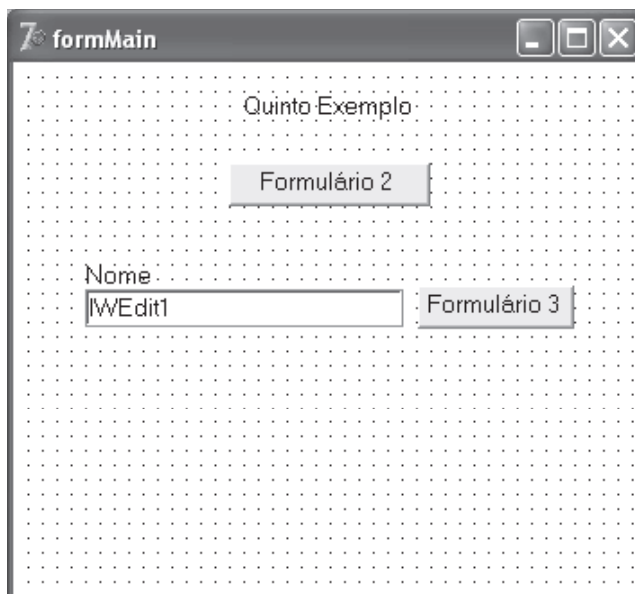


Figura 12.26 Formulário principal.

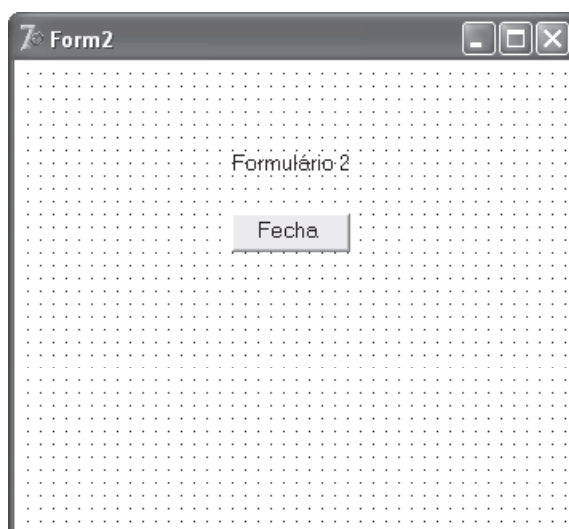


Figura 12.27 Formulário 2

Vamos dar uma olhadinha no resultado desta primeira parte de nossa aplicação. Execute a aplicação e clique no botão que representa a chamada do primeiro formulário. As figuras 12.28 e 12.29 ilustram o resultado da primeira parte de nossa aplicação.

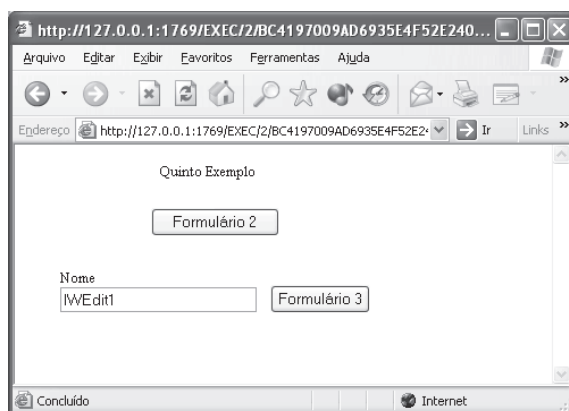


Figura 12.28 Formulário principal em ação

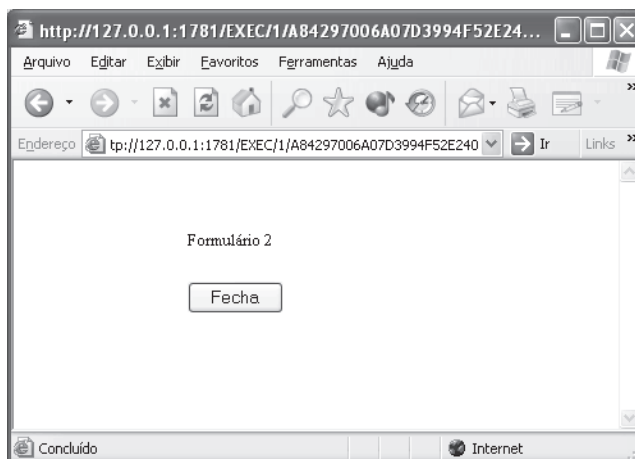


Figura 12.29 Formulário 2 chamado através do formulário principal

Agora vamos criar a segunda parte de nossa aplicação, fazendo uma interatividade entre os formulários. Através das opções *File/New/Other...*, selecione a seção *Intraweb* e escolha o modelo *Application Form* (figura 12.30).

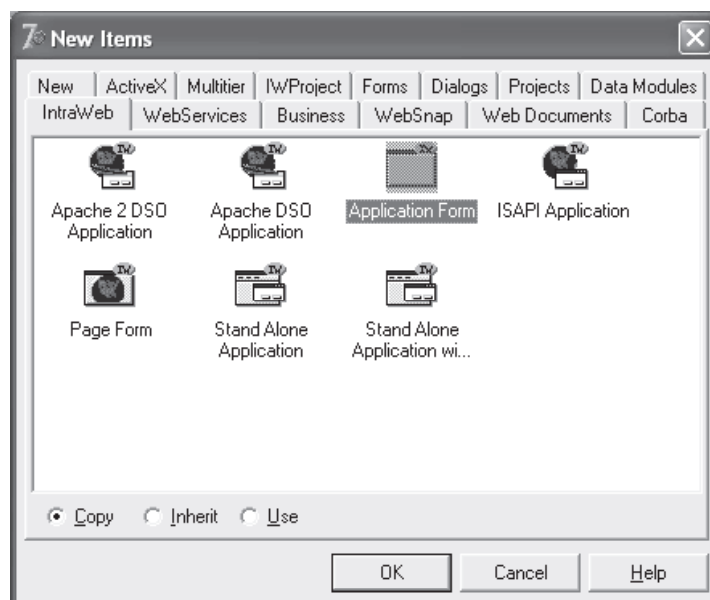





Figura 12.30 Criando um novo formulário

Altere a propriedade *Name* do formulário para *Form3*. Grave a *unit* com o nome *un_formulário3*. Insira os componentes que seguem no *Form3*.

OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLabel1	Name	IWLabel1
	Caption	Formulário 3
	Left	136
	Top	36

OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IbMensagem	Name	IbMensagem
	Caption	Mensagem
	Left	32
	Top	72

OBJETO		
	TIWButton	
Objeto	Propriedade	Valor
IWButton1	Name	IWButton1
	Caption	Fecha
	Left	128
	Top	120
	Width	75

Vamos codificar este formulário. No evento *OnClick* do objeto *IWButton1*, insira o seguinte código

```
Hide;
```

Voltando ao formulário principal, coloque a *unit un_formulario3* na cláusula *uses*. Vamos codificar o botão *IWButton2* do formulário principal. Coloque o código que segue no evento *OnClick* do botão.

```
var
Form3:TForm3;
begin
  Form3 := TForm3.Create(WebApplication);
  Form3.lbMensagem.Caption:=edNome.Text+', seja bem-vindo(a)';
  Form3.Show;
end;
```

Assim como na primeira fase da aplicação, declaramos uma variável do tipo *TForm3* (classe herdada da *TForm3* que está contida na *unit un_formulario3*).

```
var
Form3:TForm3;
```

Em seguida instanciamos o objeto.

```
Form3 := TForm3.Create(WebApplication);
```

E agora, que temos o controle total do objeto, estamos alterando a propriedade *Caption* do objeto *lbMensagem*.

```
Form3.lbMensagem.Caption:=edNome.Text+', seja bem-vindo(a)';
```

E por fim, apresentamos o objeto.

```
Form3.Show;
```

As figuras 12.31 e 12.32 ilustram o resultado da segunda fase de nossa aplicação.

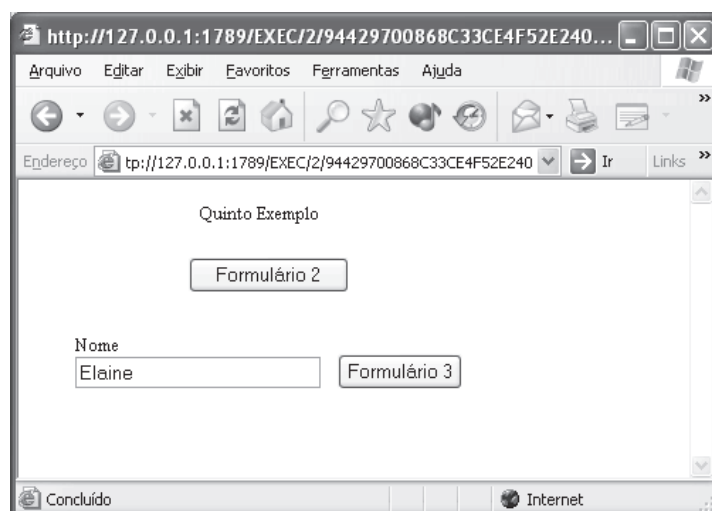


Figura 12.31 Formulário principal fazendo a chamada ao formulário 3

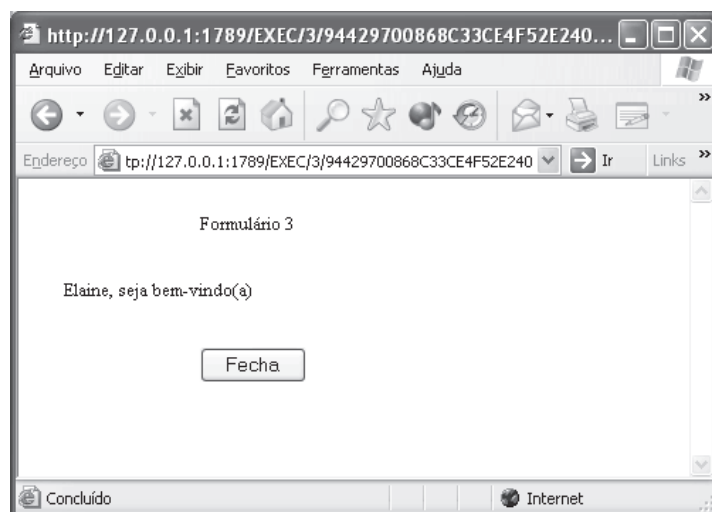


Figura 12.32 Formulário 3 em ação

Este exemplo, embora bastante simples, demonstra a interatividade entre formulários. Bastante comum em aplicações Internet, tenho certeza que utilizarão muito este conceito.

Listagem 12.7 *IWUnit1.pas (Quinto Exemplo, Form Principal)*

```
unit IWUnit1;
{PUBDIST}

interface

uses
  IWAppForm, IWApplication, IWTypes, IWCompEdit, IWCompLabel, Classes,
  Controls, IWControl, IWCompButton;

type
  TFormMain = class(TIWAppForm)
    IWButton1: TIWButton;
```

```

    IWLabel1: TIWLabel;
    edNome: TIWEdit;
    IWLabel2: TIWLabel;
    IWButton2: TIWButton;
    procedure IWButton1Click(Sender: TObject);
    procedure IWButton2Click(Sender: TObject);
  public
  end;

implementation
{$R *.dfm}

uses
  ServerController, un_formulario2, un_formulario3;

procedure TFormMain.IWButton1Click(Sender: TObject);
var
  Form2:TForm2;
begin
  Form2 := TForm2.Create(WebApplication);
  Form2.Show;
end;

procedure TFormMain.IWButton2Click(Sender: TObject);
var
  Form3:TForm3;
begin
  Form3 := TForm3.Create(WebApplication);
  Form3.lbMensagem.Caption:=edNome.Text+', seja bem-vindo(a)';
  Form3.Show;

end;

end.

```

Listagem 12.8 un_formulario2.pás

```

unit un_formulario2;
{PUBDIST}

interface

uses
  IWAppForm, IWApplication, IWTypes, IWCompButton, Classes, Controls,
  IWControl, IWCompLabel;

type
  TForm2 = class(TIWAppForm)
    IWLabel1: TIWLabel;
    IWButton1: TIWButton;
    procedure IWButton1Click(Sender: TObject);
  public
  end;

implementation
{$R *.dfm}

uses

```

```
ServerController;  
  
procedure TForm2.IWButton1Click(Sender: TObject);  
begin  
    Hide;  
end;  
  
end.
```

Listagem 12.9 un_formulario3.pás

```
unit un_formulario3;  
{PUBDIST}  
  
interface  
  
uses  
    IWAppForm, IWApplication, IWTypes, IWCompButton, Classes, Controls,  
    IWControl, IWCompLabel;  
  
type  
    TForm3 = class(TIWAppForm)  
        IWLabel1: TIWLabel;  
        lbMensagem: TIWLabel;  
        IWButton1: TIWButton;  
        procedure IWButton1Click(Sender: TObject);  
    public  
    end;  
  
implementation  
{$R *.dfm}  
  
uses  
    ServerController;  
  
procedure TForm3.IWButton1Click(Sender: TObject);  
begin  
    Hide;  
end;  
  
end.
```

Sexto Exemplo (primeiro com banco de dados)

Amigos, agora chegou a hora boa: trabalhar com banco de dados e Internet. Já desenvolvemos nossas aplicações com banco de dados para Internet com outras tecnologias (Web-Broker e Websnap) e agora teremos o prazer de utilizar o Intraweb.

Através das opções *File/New/Other...*, selecione a seção *Intraweb* e escolha o modelo *Stand Alone Application with Data Module* (figura 12.33).

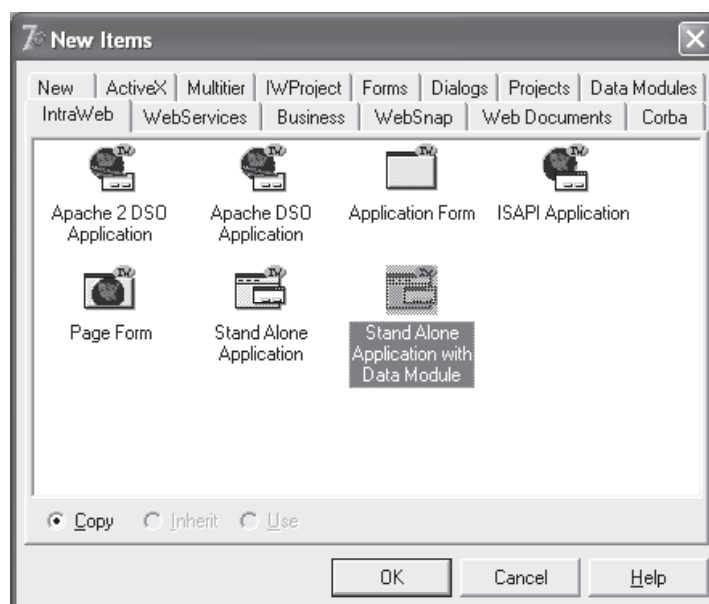


Figura 12.33 Iniciando a aplicação

A única diferença na opção *with Data Module*, é justamente a criação automática de um *Data Module*, bem como suas referências. Insira um objeto do tipo *TSQLConnection*, e através do duplo-clique, já na tela de configuração, aponte para a nossa conexão Clientes, criada anteriormente. Vamos relembrar os atributos da conexão.

PROPRIEDADE	VALOR
CommitRetain	False
Database	<i>localhost:c:\cursoweb\clientes.gdb</i>
Password	a famosa masterkey
UserName	o famoso SYSDBA
Name	ConexaoBD

Altere também a propriedade *LoginPrompt* para false. Nunca esqueça de fazer esta alteração, pois numa aplicação servidora, não existe a possibilidade do usuário interagir no login do banco de dados.

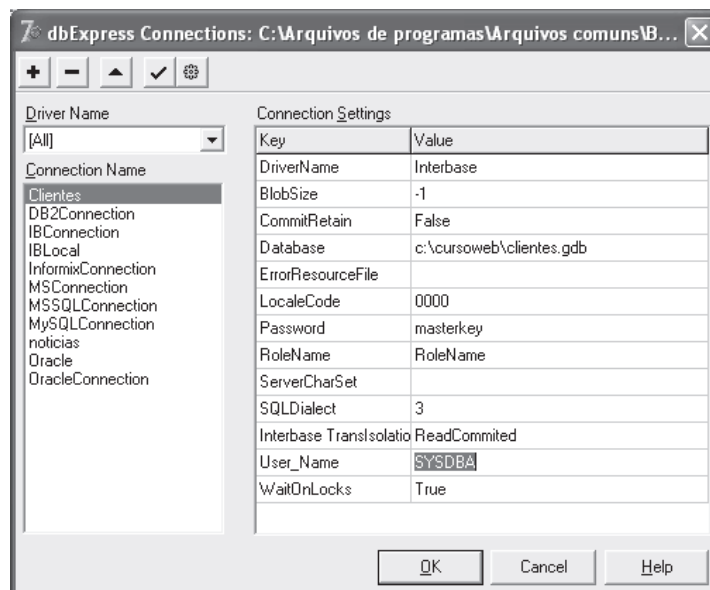




Figura 10.34 Configuração da Conexão


Agora vamos inserir o objeto para manipular nossa tabela de clientes. Insira um objeto do tipo *TSQLDataSet* e altere as seguintes propriedades:

PROPRIEDADE	VALOR
SQLConnection	ConexaoBD
CommandText	select * from TBCLIENTE
Active	True

Ótimo, agora vamos para o formulário principal. Insira a *unit DataModuleUnit* na cláusula *uses*, e grave a aplicação. Insira os componentes que seguem.

OBJETO		
	TIWLabel	
	Objeto	
IWLabel1	Propriedade	Valor
	Name	IWLabel1
	Caption	Sexto Exemplo
	Left	128
	Top	208

OBJETO		
	TDataSource [DataAccess]	
Objeto	Propriedade	Valor
DataSource1	Name	DataSource1
	DataSet	DataModule1.SQLDataSet1

OBJETO		
	TIWDBGrid	
Objeto	Propriedade	Valor
IWDBGrid1	Name	IWDBGRID1
	Align	alTop
	DataSource	DataSource1

A figura 12.35 ilustra nosso formulário principal.

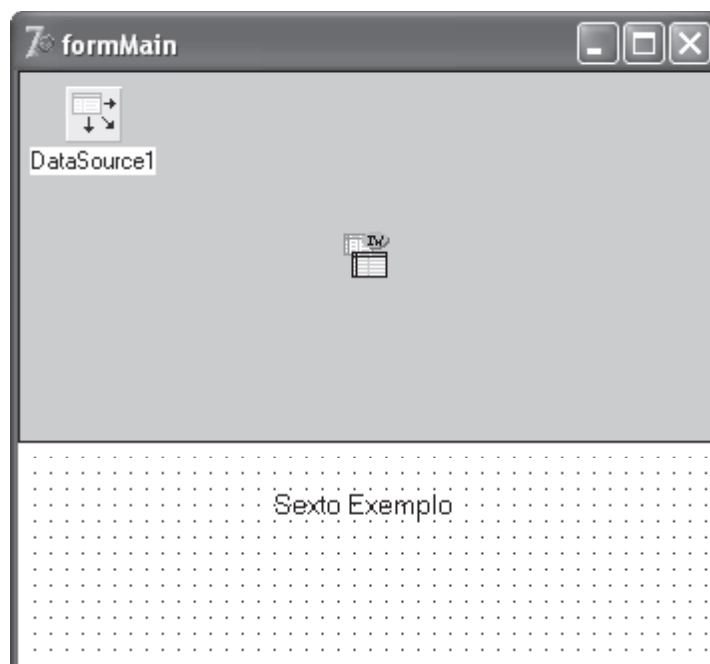


Figura 12.35 Formulário principal

Não colocamos nenhum código nesta aplicação. Assim como em aplicações *desktop*, apenas associamos os objetos. Vamos executar a aplicação e ver o resultado (figura 12.36).



Figura 12.36 Sexto exemplo em execução

Demonstrei neste exemplo uma forma simples de disponibilizar informações de banco de dados na Internet. O grande momento deste exemplo foi justamente a simplicidade e a semelhança com o método tradicional de desenvolvimento.

No próximo tópico vamos desenvolver uma aplicação completa, a mesma que desenvolvemos no Capítulo 7.

Listagem 12.10 *IWUnit1.pas (Form principal)*

```
unit IWUnit1;
{PUBDIST}

interface

uses
  IWAppForm, IWApplication, IWTypes, Classes, Controls, IWControl, IWGrids,
  IWDBGrids, DB, IWDBStdCtrls, IWClientSideDatasetBase,
  IWClientSideDatasetDBLink, IWCompLabel;

type
  TFormMain = class(TIWAppForm)
    IWDBGrid1: TIWDBGrid;
    DataSource1: TDataSource;
    IWLabel1: TIWLabel;
  public
  end;

implementation
{$R *.dfm}

uses
  ServerController, DataModuleUnit;

end.
```

Listagem 12.11 *unit DataModuleUnit.pas*

```
unit DataModuleUnit;
```

```

interface
uses
  {$IFDEF Linux}QForms, {$ELSE}Forms, {$ENDIF}
  SysUtils, Classes, DBXpress, FMTBcd, DB, SqlExpr;

type
  TDataModule1 = class(TDataModule)
    SQLConnection1: TSQLConnection;
    SQLDataSet1: TSQLDataSet;
  private
  public
  end;

// Procs
function DataModule1: TDataModule1;

implementation
{$R *.dfm}

uses
  IWInit,
  ServerController;

// Since we are threaded we cannot use global variables to store form / datamodule
// references
// so we store them in WebApplication.Data and we could reference that each time, but
// by creating
// a function like this our other code looks "normal" almost as if its referencing a
// global.
// This function is not necessary but it makes the code in the main form which
// references this
// datamodule a lot neater.
// Without this function ever time we would reference this datamodule we would use:
//   TDataModule1(WebApplication.Data).Datamodule.<method / component>
// By creating this procedure it becomes:
//   TDataModule1.<method / component>
// Which is just like normal Delphi code.
function DataModule1: TDataModule1;
begin
  Result := TUserSession(RWebApplication.Data).Datamodule1;
end;

end.

```

Cadastro de Clientes

Agora iremos desenvolver uma aplicação parecida com a do Capítulo 7, onde faremos um cadastro de clientes com as principais operações.

Através das opções *File/New/Other...*, selecione a seção *Intraweb* e escolha o modelo *Stand Alone Application with Data Module* (figura 12.37).

Definições iniciais

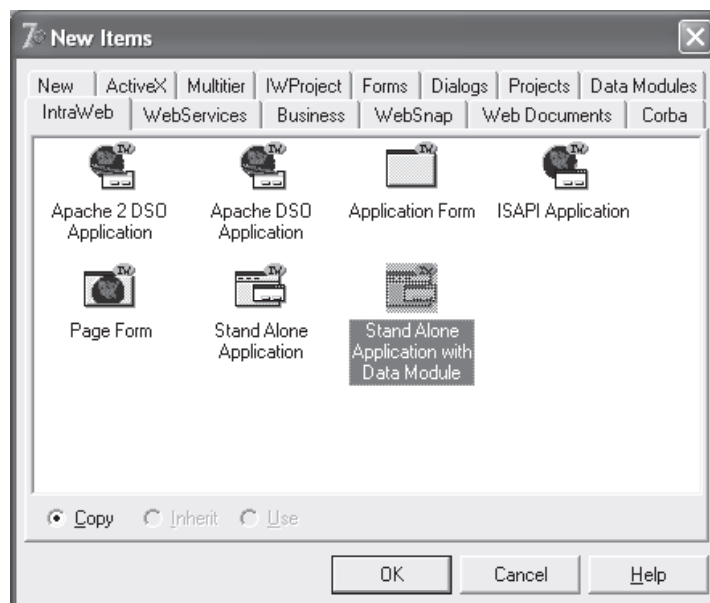


Figura 12.37 Iniciando a aplicação

Selecione o *DataModule* e insira um objeto do tipo *TSQLConnection*. Através do duplo-clique, já na tela de configuração, aponte para a nossa conexão Clientes, criada anteriormente. Vamos relembrar os atributos da conexão.

PROPRIEDADE	VALOR
CommitRetain	False
Database	<i>localhost:c:\cursoweb\clientes.gdb</i>
Password	a famosa masterkey
UserName	o famoso SYSDBA
Name	ConexaoBD

Altere também a propriedade *LoginPrompt* para false. Nunca esqueça de fazer esta alteração, pois numa aplicação servidora, não existe a possibilidade do usuário interagir no login do banco de dados.

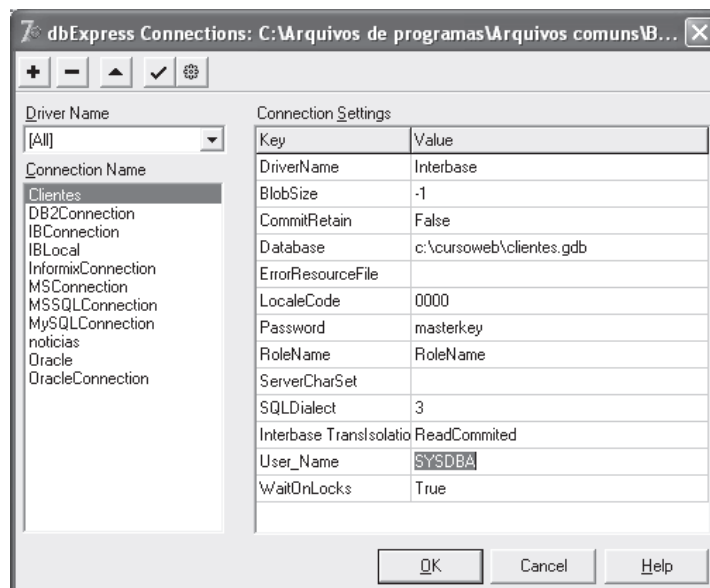



Figura 10.38 Configuração da Conexão

Agora vamos inserir o objeto para manipular nossa tabela de clientes. Insira um objeto do tipo *TSQLDataSet*, e altere as seguintes propriedades:

PROPRIEDADE	VALOR
SQLConnection	ConexaoBD
CommandText	select * from TBCLIENTE
Active	True

Objeto de Inclusão


Insira um objeto do tipo *TSQLQuery* e altere as propriedades que seguem.

OBJETO		
	TSQLQuery	
Objeto	Propriedade	Valor
SQLInclui	Name	SQLInclui
	SQLConnection	ConexaoBD
	SQL	INSERT INTO TBCLIENTE VALUES(0,:prazao, :pendereco,:pcidade, :pestado,:pcep,:pemail)

Na propriedade *PARAMS* do *SQLInclui* altere os tipos dos parâmetros para *String*.

Objeto de Alteração


Insira um objeto do tipo *TSQLQuery* e altere as propriedades que seguem.

OBJETO		
	TSQLQuery	
Objeto	Propriedade	Valor
SQLAlterar	Name	SQLAlterar
	SQLConnection	BancoDados
	SQL	UPDATE TBCLIENTE SET RAZAO_SOCIAL=:prazao, ENDERECO=:pendereco, CIDADE=:pcidade, ESTADO=:pestado, CEP=:pcep, EMAIL=:pemail WHERE COD_CLIENTE=:pcodigo

Na propriedade *PARAMS* do *SQLAlterar* altere os tipos dos parâmetros para *String*, com exceção do parâmetro *PCODIGO*, que deve ser *Integer*.

Objeto de Exclusão

Insira um objeto do tipo *TSQLQuery* e altere as propriedades que seguem.

OBJETO		
	TSQLQuery	
Objeto	Propriedade	Valor
SQLExclui	Name	SQLExclui
	SQLConnection	BancoDados
	SQL	DELETE FROM TBCLIENTE WHERE COD_CLIENTE=:pcodigo

Na propriedade *PARAMS* do *SQLExclui* altere o tipo do parâmetro para *Integer*.

Criando o Formulário Principal

No formulário principal (*FormMain*), adicione um componente do tipo *TMainMenu* (que utilizamos habitualmente), e configure as opções como segue:

Arquivo	Sobre
Inclusão Clientes	Informações...
Manutenção/Consulta	
Finaliza	

A *figura 12.39* ilustra o nosso menu principal.

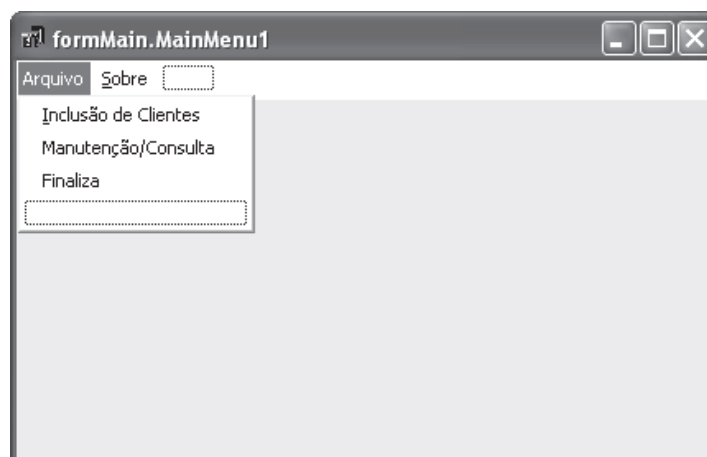




Figura 12.39 Menu Principal

Insira um objeto do tipo *TIWMenu* (seção *IW Standard*) e altere a propriedade *Attached Menu* para *MainMenu1*. Na realidade estamos vinculando nosso objeto *MainMenu1* com o *IWMenu1* do *Intraweb*.

Configure a propriedade *BackColor* do objeto *FormMain* para *\$00DDFFFF*. Amigos, isto é apenas uma sugestão de cor.

Insira os componentes que seguem no *FormMain*.

OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLabel1	Name	IWLabel1
	Caption	Cadastro de Clientes
	Left	136
	Top	80

OBJETO		
	TIWImage	
Objeto	Propriedade	Valor
IWImage1	Name	IWImage1
	Left	168
	Picture	insira uma figura qualquer, exemplo: logo.jpg
	Top	120

A figura 12.40 ilustra o nosso formulário principal.



Figura 12.40 Formulário principal

Agora iremos codificar duas opções do menu principal: *Finaliza* e *Informações/Sobre*.

Selecione a opção *Finaliza* no objeto *MainMenu1* e insira o código que segue no evento *OnClick*.

```
WebApplication.Terminate('Até logo !');
```

Neste evento estamos finalizando a aplicação e apresentando a mensagem “Até Logo!”, ao usuário.

Agora selecione a opção *Informações/Sobre* e insira o código que segue.

```
WebApplication.ShowMessage('Cadastro de Clientes' + #13#10+ 'Versão 1.0',smAlert);
```

Neste evento apresentamos uma janela de diálogo ao usuário, com as informações da aplicação. Bastante simples, não?

Criando o Formulário de Inclusão

Através das opções *File/New/Other...*, selecione a seção *Intraweb* e escolha o modelo *Application Form* (figura 12.41).

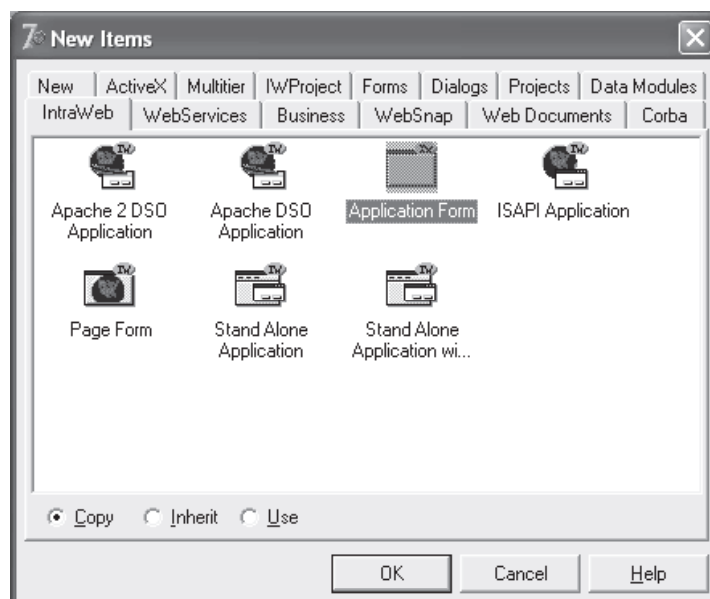






Figura 12.41 Criando um novo formulário


Altere a propriedade *Name* do formulário para *FmInclusao* e *BackColor* para *\$00DDFFFF*. Grave a *unit* com o nome *un_inclusao*. Insira os componentes que seguem no *FmInclusao*.


OBJETO		
	TIWRectangle	
Objeto	Propriedade	Valor
IWRectangle1	Name	IWRectangle1
	Align	alTop
	Color	clNavy
	Font.Color	clWhite
	Font.Size	16
	Font.Style.fsBold	True
	Text	Cadastro de Clientes (Inclusão)


OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLabel1	Name	IWLabel1
	Caption	Razão Social
	Left	24
	Top	88


OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLabel2	Name	IWLabel2
	Caption	Endereço
	Left	24
	Top	120


OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLabel3	Name	IWLabel3
	Caption	Cidade
	Left	24
	Top	152


OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLabel4	Name	IWLabel4
	Caption	UF
	Left	328
	Top	152


OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLLabel5	Name	IWLLabel5
	Caption	CEP
	Left	448
	Top	152


OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLLabel6	Name	IWLLabel6
	Caption	e-Mail
	Left	24
	Top	184


OBJETO		
	TIWEdit	
Objeto	Propriedade	Valor
edRazao	Name	edRazao
	BgColor	clYellow
	Height	21
	Left	128
	Text	deixar em branco
	Top	88
	Width	450


OBJETO		
	TIWEdit	
Objeto	Propriedade	Valor
edEndereco	Name	edEndereco
	BgColor	clYellow
	Height	21
	Left	128
	Text	deixar em branco
	Top	120
	Width	450


OBJETO		
	TIWEdit	
Objeto	Propriedade	Valor
edCidade	Name	edCidade
	BgColor	clYellow
	Height	21
	Left	128
	Text	deixar em branco
	Top	152
	Width	185

OBJETO		
	TIWComboBox	
Objeto	Propriedade	Valor
edUF	Name	edUF
	Height	21
	Left	360
	Items	Insira a sigla de todos os estados brasileiros
	Sorted	True
	Top	152
	Width	65

OBJETO		
	TIWEdit	
Objeto	Propriedade	Valor
edCEP	Name	edCEP
	BgColor	clYellow
	Height	21
	Left	488
	Text	deixar em branco
	Top	152
	Width	90

OBJETO		
	TIWEdit	
Objeto	Propriedade	Valor
edEmail	Name	edEmail
	BgColor	clYellow
	Height	21
	Left	128
	Text	deixar em branco
	Top	184
	Width	450

OBJETO		
	TIWButton1	
Objeto	Propriedade	Valor
btConfirma	Name	btConfirma
	Caption	Confirma
	Height	25
	Left	128
	Top	232
	Width	75

OBJETO		
	TIWButton1	
Objeto	Propriedade	Valor
btDesiste	Name	btDesiste
	Caption	Desiste
	Height	25
	Left	216
	Top	232
	Width	75

Agora vamos codificar o formulário. Insira a *unit DataModuleUnit* na cláusula *uses* do formulário.

```
uses
  ServerController, DatamoduleUnit;
```

No evento *OnClick* do objeto *btDesiste* insira o código que segue:

```
Hide;
```

Para que possamos retornar ao formulário de origem, utilizamos o método *Hide* do formulário. Agora vamos codificar o evento *OnClick* do objeto *btConfirma*.

```

try

  {Inclui Cliente}
  with DataModule1.SQLInclui do
  begin
    ParamByName('prazao').Value:=edRazao.Text;
    ParamByName('pendereco').Value:=edEndereco.Text;
    ParamByName('pcidade').Value:=edCidade.Text;
    ParamByName('pestado').Value:=edUf.Items[edUf.ItemIndex];
    ParamByName('pcep').Value:=edCep.Text;
    ParamByName('pemail').Value:=edEmail.Text;
    ExecSQL;
  end;

  WebApplication.ShowMessage('Cliente incluído com sucesso',smSameWindow);

except
  WebApplication.ShowMessage('Houve um problema na inclusão do cliente',
smSameWindow);
end;
Hide;

```

Vamos analisar o código. Nesta primeira parte, iniciamos um bloco protegido.

```
try
```

Em seguida, estamos atribuindo parâmetros ao objeto *SQLInclui* do *DataModule1*.

```

  {Inclui Cliente}
  with DataModule1.SQLInclui do
  begin
    ParamByName('prazao').Value:=edRazao.Text;
    ParamByName('pendereco').Value:=edEndereco.Text;
    ParamByName('pcidade').Value:=edCidade.Text;
    ParamByName('pestado').Value:=edUf.Items[edUf.ItemIndex];
    ParamByName('pcep').Value:=edCep.Text;
    ParamByName('pemail').Value:=edEmail.Text;

```

E, finalmente, executando a operação e apresentando a mensagem de *sucesso na operação*.

```

ExecSQL;

WebApplication.ShowMessage('Cliente incluído com sucesso',smSameWindow);

```

Em caso de erro, estamos criando o bloco *except*.

```

except
  WebApplication.ShowMessage('Houve um problema na inclusão do cliente',
smSameWindow);
end;

```

E apresentando a mensagem do problema. A *figura 12.42* ilustra o nosso formulário de inclusão de clientes.

Figura 12.42 Formulário inclusão de clientes

Criando o Formulário de Alteração

Através das opções *File/New/Other...*, selecione a seção *Intraweb* e escolha o modelo *Application Form* (figura 12.43).

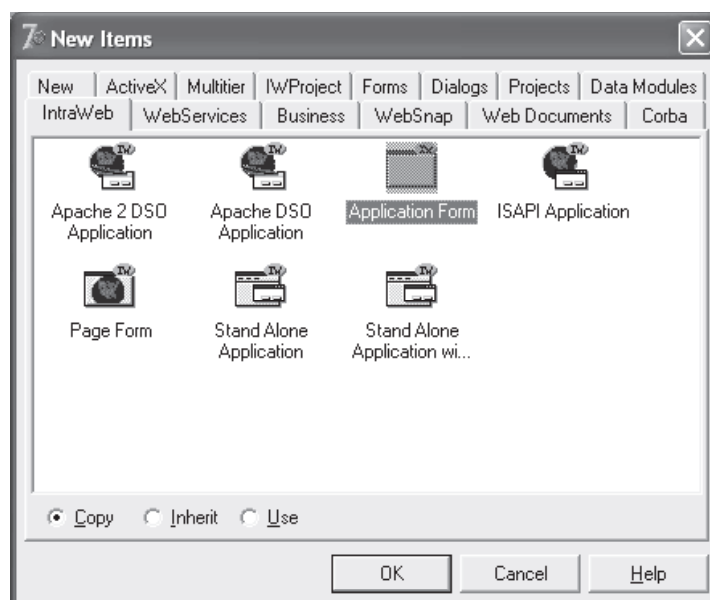






Figura 12.43 Criando um novo formulário


Altere a propriedade *Name* do formulário para *FmAlterar* e *BackColor* para *\$00DDFFFF*. Grave a *unit* com o nome *un_alteracao*. Insira os componentes que seguem no *FmAlterar*.


OBJETO		
	TIWRectangle	
Objeto	Propriedade	Valor
IWRectangle1	Name	IWRectangle1
	Align	alTop
	Color	clNavy
	Font.Color	clWhite
	Font.Size	16
	Font.Style.fsBold	True
	Text	Cadastro de Clientes (Alteração)


OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLabel1	Name	IWLabel1
	Caption	Razão Social
	Left	24
	Top	88


OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLabel2	Name	IWLabel2
	Caption	Endereço
	Left	24
	Top	120


OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLabel3	Name	IWLabel3
	Caption	Cidade
	Left	24
	Top	152


OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLLabel4	Name	IWLLabel4
	Caption	UF
	Left	328
	Top	152


OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLLabel5	Name	IWLLabel5
	Caption	CEP
	Left	448
	Top	152


OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLLabel6	Name	IWLLabel6
	Caption	e-Mail
	Left	24
	Top	184


OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLLabel7	Name	IWLLabel7
	Caption	Código Cliente
	Left	24
	Top	56


OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IbCodigo	Name	IbCodigo
	Caption	código
	Left	144
	Top	56


OBJETO		
	TIWEdit	
Objeto	Propriedade	Valor
edRazao	Name	edRazao
	BgColor	clYellow
	Height	21
	Left	128
	Text	deixar em branco
	Top	88
	Width	450


OBJETO		
	TIWEdit	
Objeto	Propriedade	Valor
edEndereco	Name	edEndereco
	BgColor	clYellow
	Height	21
	Left	128
	Text	deixar em branco
	Top	120
	Width	450


OBJETO		
	TIWEdit	
Objeto	Propriedade	Valor
edCidade	Name	edCidade
	BgColor	clYellow
	Height	21
	Left	128
	Text	deixar em branco
	Top	152
	Width	185

OBJETO		
	TIWComboBox	
Objeto	Propriedade	Valor
edUF	Name	edUF
	Height	21
	Left	360
	Items	Insira a sigla de todos os estados brasileiros
	Sorted	True
	Top	152
	Width	65

OBJETO		
	TIWEdit	
Objeto	Propriedade	Valor
edCEP	Name	edCEP
	BgColor	clYellow
	Height	21
	Left	488
	Text	deixar em branco
	Top	152
	Width	90

OBJETO		
	TIWEdit	
Objeto	Propriedade	Valor
edEmail	Name	edEmail
	BgColor	clYellow
	Height	21
	Left	128
	Text	deixar em branco
	Top	184
	Width	450

OBJETO		
	TIWButton1	
Objeto	Propriedade	Valor
btConfirma	Name	btConfirma
	Caption	Confirma
	Height	25
	Left	128
	Top	232
	Width	75

OBJETO		
	TIWButton1	
Objeto	Propriedade	Valor
btDesiste	Name	btDesiste
	Caption	Desiste
	Height	25
	Left	216
	Top	232
	Width	75

Agora vamos codificar o formulário. Insira a *unit DataModuleUnit* na cláusula *uses* do formulário.

```
uses
  ServerController, DatamoduleUnit;
```

No evento *OnClick* do objeto *btDesiste* insira o código que segue:

```
Hide;
```

Para que possamos retornar ao formulário de origem, utilizamos o método *Hide* do formulário.

Agora vamos codificar o evento *OnClick* do objeto *btConfirma*.

```
try
  {Altera Cliente}
  with DataModule1.SQLAltera do
  begin
    ParamByName('pcodigo').Value:=StrToInt(lbCodigo.Text);
    ParamByName('prazao').Value:=edRazao.Text;
    ParamByName('pendereco').Value:=edEndereco.Text;
    ParamByName('pcidade').Value:=edCidade.Text;
    ParamByName('pestado').Value:=edUf.Items[edUf.ItemIndex];
    ParamByName('pcep').Value:=edCep.Text;
    ParamByName('pemail').Value:=edEmail.Text;
    ExecSQL;
  end;

  WebApplication.ShowMessage('Cliente alterado com sucesso',smSameWindow);
```

```

except
    WebApplication.ShowMessage('Houve um problema na alteração do cliente',
smSameWindow);
end;
Hide;

```

Vamos analisar o código. Nesta primeira parte, iniciamos um bloco protegido.

```

try

```

Em seguida, estamos atribuindo parâmetros ao objeto *SQLAlterar* do *DataModule1*.

```

{Inclui Cliente}
with DataModule1.SQLAlterar do
begin
    ParamByName('pcodigo').Value:=StrToInt(lbCodigo.Text);
    ParamByName('prazao').Value:=edRazao.Text;
    ParamByName('pendereco').Value:=edEndereco.Text;
    ParamByName('pcidade').Value:=edCidade.Text;
    ParamByName('pestado').Value:=edUf.Items[edUf.ItemIndex];
    ParamByName('pcep').Value:=edCep.Text;
    ParamByName('pemail').Value:=edEmail.Text;

```

E finalmente executando a operação e apresentando a mensagem de *sucesso na operação*.

```

ExecSQL;

WebApplication.ShowMessage('Cliente incluído com sucesso',smSameWindow);

```

Em caso de erro, estamos criando o bloco *except*.

```


except
    WebApplication.ShowMessage('Houve um problema na inclusão do cliente',
smSameWindow);
end;

```


E apresentando a mensagem do problema.

Embora o Intraweb forneça objeto com conexão direta a *DataSets*, estamos utilizando os objetos convencionais, sem nenhum vínculo com *DataSet*. Quando trabalhamos com Internet, é complicado disponibilizar objeto de conexão direta com *DataSets*, justamente porque não sabemos o que poderá ocorrer com a conexão. Sugiro adotar este modelo de desenvolvimento pela segurança.

Então, para que nossos campos sejam preenchidos automaticamente com as informações do banco de dados, vamos utilizar o *TDataSource* e *TSQLDataSet*.

OBJETO		
	TSQLDataSet	
Objeto	Propriedade	Valor
SQLDataSet1	Name	SQLDataSet1
	SQLConnection	DataModule1.ConexaoBD
	CommandText	select * from TBCLIENTE WHERE COD_CLIENTE=:pCodigo
	CommandType	Query

Defina o parâmetro *pCodigo* como *Integer*.

OBJETO		
	TDataSource	
Objeto	Propriedade	Valor
DataSource1	Name	DataSource
	DataSet	SQLDataSet1

Pronto, agora concluímos o nosso formulário de alteração de clientes. A *figura 12.44* ilustra o nosso formulário.




Figura 12.44 Formulário alteração de clientes

Criando o Formulário de Manutenção

Através das opções *File/New/Other...*, selecione a seção *Intraweb* e escolha o modelo *Application Form* (*figura 12.45*).

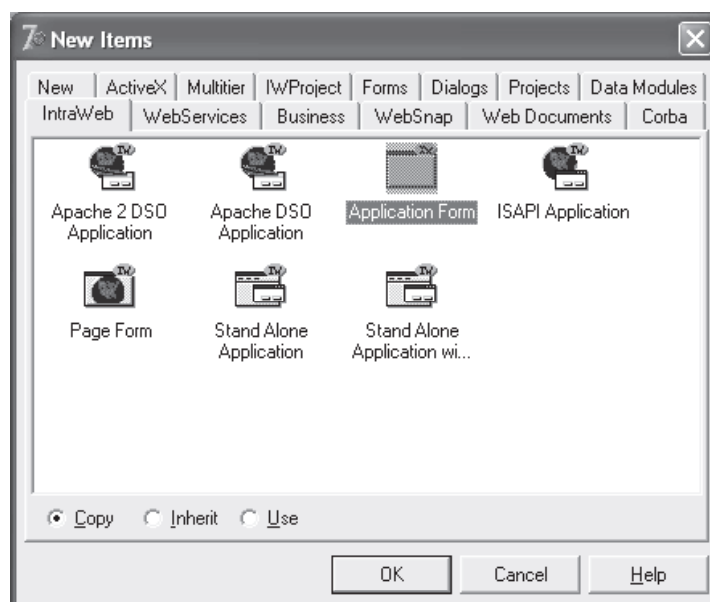




Figura 12.45 Criando um novo formulário


Altere a propriedade *Name* do formulário para *FmManutencao* e *BackColor* para \$00DDFFFF. . Grave a *unit* com o nome *un_manutencao*.


Insira os componentes que seguem no *FmManutencao*..


OBJETO		
	TIWRectangle	
Objeto	Propriedade	Valor
IWRectangle1	Name	IWRectangle1
	Align	alTop
	Color	clNavy
	Font.Color	clWhite
	Font.Size	16
	Font.Style.fsBold	True
	Text	Cadastro de Clientes (Manutenção)


OBJETO		
	TIWRegion1	
Objeto	Propriedade	Valor
IWRegion1	Name	IWRegion1
	Align	alTop
	Color	clWhite
	Height	56

Com o foco no objeto *IWRegion1*, insira os componentes que seguem.


OBJETO		
	TIWLabel	
Objeto	Propriedade	Valor
IWLabel1	Name	IWLabel1
	Caption	Razão Social
	Left	16
	Top	16


OBJETO		
	TIWEdit	
Objeto	Propriedade	Valor
edPesquisa	Name	edPesquisa
	BgColor	clYellow
	Height	21
	Left	112
	Text	deixar em branco
	Top	16
	Width	175

OBJETO		
	TIWButton	
Objeto	Propriedade	Valor
btPesquisa	Name	btPesquisa
	Caption	Confirma
	Height	25
	Left	296
	Top	16
	Width	75

OBJETO		
	TIWButton	
Objeto	Propriedade	Valor
btVolta	Name	btVolta
	Caption	Menu Principal
	Height	25
	Left	384
	Top	16
	Width	122


Em nosso formulário iremos utilizar objetos de acesso a dados. Insira um objeto do tipo *TDataSource* e outro do tipo *TSQLQuery* (*dbExpress*), configurando as propriedades que seguem.

OBJETO		
	TSQLQuery	
Objeto	Propriedade	Valor
SQLPesquisa	Name	SQLPesquisa
	SQLConnection	DataModule1.ConexaoBD
	SQL	select * from TBCLIENTE

OBJETO		
	TDataSource	
Objeto	Propriedade	Valor
DS1	Name	DS1
	DataSet	SQLPesquisa

Repare que estamos colocando o comando *SELECT ** na propriedade *SQL* do objeto *SQLPesquisa*. Fazemos isso apenas para nos auxiliar na montagem do Grid.

Com o foco no formulário, insira um objeto do tipo *TIWDBGrid* e altere as propriedades que seguem.

OBJETO		
	TIWDBGrid	
Objeto	Propriedade	Valor
IWDbGrid1	Name	IWDbGrid1
	DataSource	DS1
	Rollover	True
	RollOverColor	\$00E1FFE1

Através do duplo-clique na propriedade *Columns*, insira os campos conforme a figura 12.46.

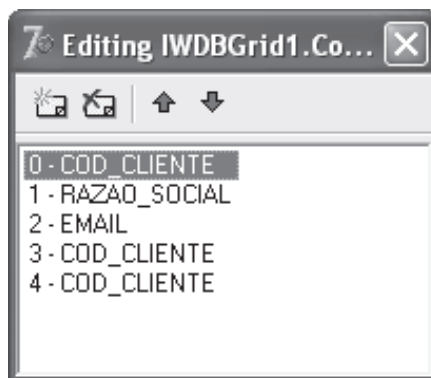


Figura 12.46 Campos do grid.

Curiosidade

Repare que estamos repetindo o campo *COD_CLIENTE* diversas vezes. Fazemos isto para substituir os métodos de chamada de operações, devido a um BUG no IntraWeb. Até a versão 5.51 (que vem junto com o D7), o BUG de vínculo com controles (Control) ainda não estava corrigido. Existe uma maneira de associar um objeto, como por exemplo, um botão, a um evento do Grid. Iríamos fazer isso para disparar as rotinas de alteração e exclusão, mas infelizmente, devido ao BUG, estamos substituindo por outro método.

Vamos configurar as colunas inseridas.

Coluna [0] Cod_Cliente		
Objeto	Propriedade	Valor
IWDbGrid1>Columns[0]		
	DataField	Cod_Cliente
	LinkField	Cod_Cliente
	Title.Text	Código

Coluna [1] Razao_Social		
Objeto	Propriedade	Valor
IWDbGrid1>Columns[1]		
	DataField	Razao_Social
	Title.Text	Razão Social

Coluna [2] EMAIL		
Objeto	Propriedade	Valor
IWDbGrid1>Columns[2]		
	DataField	email
	Title.Text	e-Mail

Coluna [3] Cod_Cliente		
Objeto	Propriedade	Valor
IWDbGrid1>Columns[3]		
	DataField	Cod_Cliente
	LinkField	Cod_Cliente
	Title.Text	Alterar

Coluna [4] Cod_Cliente		
Objeto	Propriedade	Valor
IWDbGrid1>Columns[4]		
	DataField	Cod_Cliente
	LinkField	Cod_Cliente
	Title.Text	Excluir

Agora vamos codificar o nosso formulário. Na cláusula *uses* insira as seguintes *units*.

```
uses
  ServerController, DataModuleUnit, un_alteracao;
```

Em seguida devemos criar uma *procedure* pública para auxiliar no posicionamento do registro, de maneira que possamos atribuir o conteúdo do banco de dados, aos campos do formulário *fmAlterar*. Defina a *procedure LOCALIZA*, como segue.

```
public
  procedure Localiza(Cliente: integer);
```

E na seção *implementation*, insira o código a seguir.

```
procedure Tfmmanutencao.Localiza(Cliente:integer);
begin
  SQLPesquisa.Locate('cod_cliente', Cliente, []);
end;
```

Neste código estamos apenas posicionando o ponteiro do registro de acordo com a seleção do usuário. Agora vamos codificar o objeto *btPesquisa*. Insira o código que segue no evento *OnClick*.

```
SQLPesquisa.SQL.Clear;
SQLPesquisa.SQL.Add('SELECT      *      FROM      TBCLIENTE      WHERE      RAZAO_SOCIAL      LIKE
'+'''+'%'+edPesquisa.Text+'%'+''');
SQLPesquisa.Open;
```

Neste código estamos preparando uma consulta SQL, baseada na informação do usuário, extraída do objeto *edPesquisa*. As informações são apresentadas no *Grid*, devido ao vínculo com o objeto. Coloque o código a seguir, no evento *OnClick* do objeto *btVolta*.

```
Hide;
```

Agora vamos codificar a operação de *alteração* do *Grid*.

Através da propriedade *Columns* do *IWDBGrid1*, selecione a coluna 3 (**Alterar**) e insira o código que segue no evento *OnClick*.

```
var
fmAlterar:TfmAlterar;
begin
  fmAlterar:=TFmAlterar.Create(WebApplication);
  {Atribui o código a Query}
  with fmAlterar do
    begin
      FmAlterar.SQLDataSet1.Params[0].Value:=StrToInt(AValue);
      fmAlterar.SQLDataSet1.Open;

      edRazao.Text:=SQLDataSet1RAZAO_SOCIAL.AsString;
      edEndereco.Text:=SQLDataSet1ENDERECO.AsString;
      edCidade.Text:=SQLDataSet1CIDADE.AsString;
      edCEP.Text:=SQLDataSet1CEP.AsString;
      edUF.ItemIndex:=edUF.Items.IndexOf(QLDataSet1ESTADO.AsString);
      edEmail.Text:=SQLDataSet1EMAIL.AsString;

      {Atribui os valores}
      fmAlterar.lbCodigo.Caption:=AValue;
      fmAlterar.Show;

    end;
```

Vamos analisar o código. No bloco a seguir, estamos definindo um objeto do tipo *TfmAlterar*, para que possamos manipular o objeto.

```
var
```

```
fmAlterar:TfmAlterar;
```

No bloco que segue, instanciamos o objeto *FmAlterar*.

```
begin
  fmAlterar:=TFmAlterar.Create(WebApplication);
```

Em seguida atribuímos o código do cliente, extraído da variável *AValue* (*constante da procedure*), ao objeto *SQLDataSet1*.

```
with fmAlterar do
begin
  FmAlterar.SQLDataSet1.Params[0].Value:=StrToInt(AValue);
  fmAlterar.SQLDataSet1.Open;
```

O bloco a seguir atribui as informações do *SQLDataSet1* aos campos do objeto *fmAlterar*, apresentando o formulário de alteração de clientes.

```
edRazao.Text:=SQLDataSet1RAZAO_SOCIAL.AsString;
edEndereco.Text:=SQLDataSet1ENDERECO.AsString;
edCidade.Text:=SQLDataSet1CIDADE.AsString;
edCEP.Text:=SQLDataSet1CEP.AsString;
edUF.ItemIndex:=edUF.Items.IndexOf(QLDataSet1ESTADO.AsString);
edEmail.Text:=SQLDataSet1EMAIL.AsString;

{Atribui os valores}
fmAlterar.lbCodigo.Caption:=AValue;
fmAlterar.Show;
```

Agora vamos codificar a operação de *exclusão* do *Grid*.

Através da propriedade *Columns* do *IWDBGrid1*, selecione a coluna 4 (**Excluir**) e insira o código que segue no evento *OnClick*.

```
try
  with DataModule1.SQLExclui do
  begin
    ParamByName('pcodigo').Value:=StrToInt(AValue);
    ExecSQL;
  end;
  WebApplication.ShowMessage('Cliente Excluido com sucesso',smSameWindow);

except
  WebApplication.ShowMessage('Houve um problema na exclusao do cliente',
smSameWindow);
end;
```

Vamos analisar o código. No código que segue estamos criando um bloco protegido; atribuindo um parâmetro para o objeto *SQLExclui* e finalmente apresentando uma mensagem ao usuário, em caso de sucesso da operação.

```
try
  with DataModule1.SQLExclui do
  begin
    ParamByName('pcodigo').Value:=StrToInt(AValue);
    ExecSQL;
  end;
  WebApplication.ShowMessage('Cliente Excluido com sucesso',smSameWindow);
```

No bloco a seguir estamos tratando a exceção, apresentando para o usuário uma mensagem de alerta.

```
except
    WebApplication.ShowMessage('Houve um problema na exclusao do cliente',
smSameWindow);
end;
```

Grave a *unit un_manutencao* e selecione o formulário principal. A *figura 12.47* ilustra o nosso formulário de manutenção.



Figura 12.47 Formulário de manutenção

Vamos codificar o formulário principal, para concluir a aplicação.

Na cláusula *uses* do *formulário principal (formMain)*, insira as *units* que seguem.

```
uses
    ServerController, un_Inclusao, un_manutencao;
```

Agora vamos codificar a opção *Inclusão de Clientes* do *Menu Principal*. No evento *OnClick* da opção, insira o código a seguir.

```
var
    fmInclusao:TFmInclusao;
begin
    fmInclusao:=TFmInclusao.Create(WebApplication);
    fmInclusao.Show;
end;
```

Neste código estamos instanciando o objeto *FmInclusao* e apresentando ao usuário. E agora, vamos codificar a opção *Manutenção/Consulta*. Coloque o código que segue no evento *OnClick* da opção.

```
var
    fmManutencao:TFmManutencao;
begin
    fmManutencao:=TFmManutencao.Create(WebApplication);
    fmManutencao.Show;
end;
```

Com isso concluímos o nosso projeto de *Cadastro de Clientes*. As *figuras 12.48 a 12.53* ilustram nossa aplicação em tempo de execução.

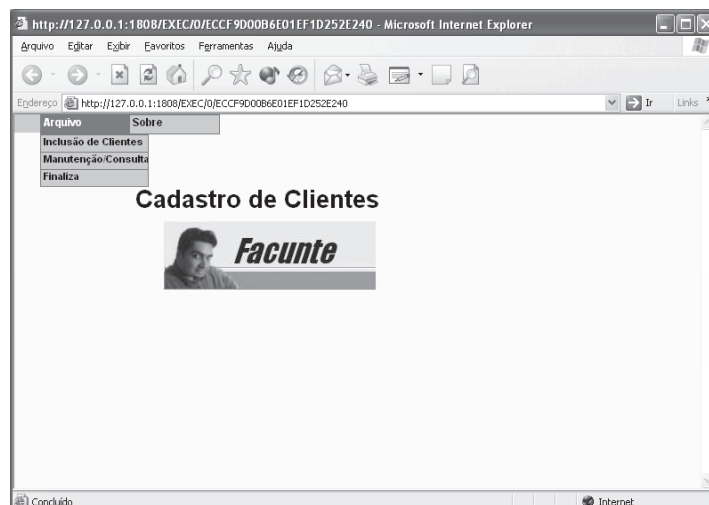


Figura 12.48 Menu principal

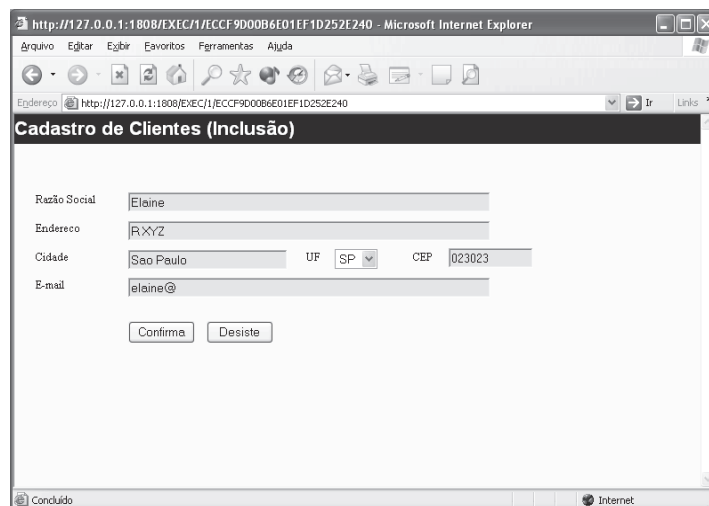


Figura 12.49 Formulário de inclusão



Figura 12.50 Informações sobre a aplicação

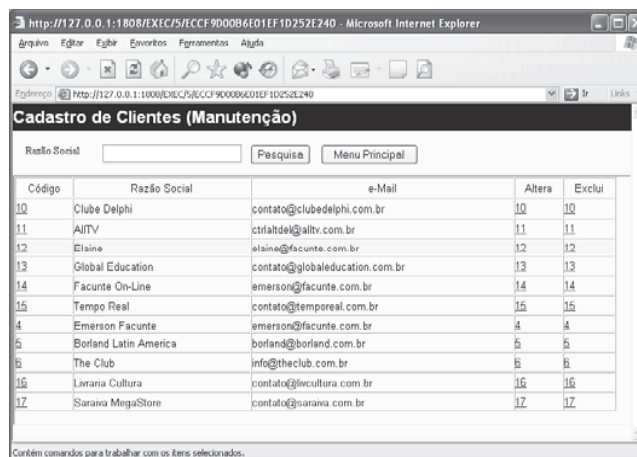


Figura 12.51 Manutenção

Repare que na tela de manutenção existe um *rollover* passando entre os registros. Para que você possa fazer qualquer tipo de operação, clique no código do cliente, correspondente à operação (altera ou exclui).

Você poderá digitar apenas uma parte da razão social do cliente. Por exemplo: digitando Facunte, são apresentados todos os registros que contêm a palavra Facunte, independente da posição.

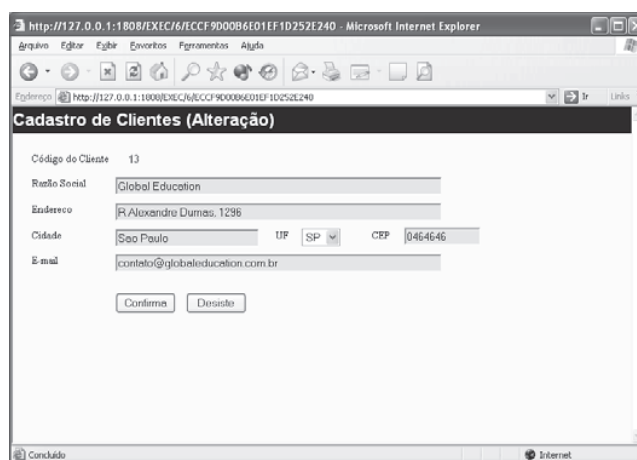


Figura 12.52 Alteração de clientes

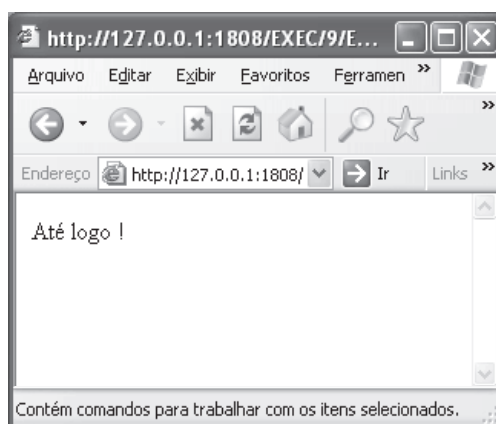


Figura 12.53 Finalizando a aplicação

Listagem 12.12 IWUnit1.pas (FormMain)

```

unit IWUnit1;
{PUBDIST}

interface

uses
  IWAppForm, IWApplication, IWTypes, Menus, Classes, Controls, IWControl,
  IWCompMenu, jpeg, IWExtCtrls, IWCompLabel;

type
  TFormMain = class(TIWAppForm)
    MainMenu1: TMainMenu;
    Arquivol: TMenuItem;
    InclusodeClientes1: TMenuItem;
    Manutenol: TMenuItem;
    Finaliza1: TMenuItem;
    IWMenu1: TIWMenu;
    Sobrel: TMenuItem;
    Informaessobreaaplicaol: TMenuItem;
    IWImage1: TIWImage;
    IWLabel1: TIWLabel;
    procedure InclusodeClientes1Click(Sender: TObject);
    procedure Finaliza1Click(Sender: TObject);
    procedure InformaessobreaaplicaolClick(Sender: TObject);
    procedure ManutenolClick(Sender: TObject);
  public
  end;

implementation
{$R *.dfm}

uses
  ServerController, Un_Inclusao, un_manutencao;

procedure TFormMain.InclusodeClientes1Click(Sender: TObject);
var
  fmInclusao:TFmInclusao;
begin
  fmInclusao:=TFmInclusao.Create(WebApplication);
  fmInclusao.Show;
end;

procedure TFormMain.Finaliza1Click(Sender: TObject);
begin
  WebApplication.Terminate('Até logo !');
end;

procedure TFormMain.InformaessobreaaplicaolClick(Sender: TObject);
begin
  WebApplication.ShowMessage('Cadastro de Clientes'+#13#10+'Versão 1.0',smAlert);
end;

procedure TFormMain.ManutenolClick(Sender: TObject);
var
  fmManutencao:TFmManutencao;

```

```

begin
    fmManutencao:=TFMManutencao.Create(WebApplication);
    fmManutencao.Show;

end;

end.

```

Listagem 12.13 un_inclusao.pas (FmInclusao)

```

unit un_inclusao;
{PUBDIST}

interface

uses
    IWAppForm, IWApplication, IWTypes, IWCompLabel, Classes, Controls,
    IWControl, IWCompEdit, IWCompListbox, IWCompButton, IWCompRectangle;

type
    TfmInclusao = class(TIWAppForm)
        edRazao: TIWEdit;
        IWLabel2: TIWLabel;
        edEndereco: TIWEdit;
        IWLabel3: TIWLabel;
        edCidade: TIWEdit;
        IWLabel4: TIWLabel;
        IWLabel5: TIWLabel;
        edUF: TIWComboBox;
        IWLabel6: TIWLabel;
        edEmail: TIWEdit;
        btConfirma: TIWButton;
        btDesiste: TIWButton;
        IWRectangle1: TIWRectangle;
        edCep: TIWEdit;
        IWLabel1: TIWLabel;
        procedure btDesisteClick(Sender: TObject);
        procedure btConfirmaClick(Sender: TObject);
    public
    end;

implementation
{$R *.dfm}

uses
    ServerController, DatamoduleUnit;

procedure TfmInclusao.btDesisteClick(Sender: TObject);
begin
    Hide;
end;

procedure TfmInclusao.btConfirmaClick(Sender: TObject);
begin
    try
        {Inclui Cliente}
    end;
end;

```



```

with DataModule1.SQLInclui do
begin
    ParamByName('prazao').Value:=edRazao.Text;
    ParamByName('pendereco').Value:=edEndereco.Text;
    ParamByName('pcidade').Value:=edCidade.Text;
    ParamByName('pestado').Value:=edUf.Items[edUf.ItemIndex];
    ParamByName('pcep').Value:=edCep.Text;
    ParamByName('pemail').Value:=edEmail.Text;
    ExecSQL;
end;

WebApplication.ShowMessage('Cliente incluído com sucesso',smSameWindow);

except
    WebApplication.ShowMessage('Houve um problema na inclusão do cliente',
smSameWindow);
end;
Hide;
end;

end.

```

Listagem 12.14 un_alteracao (FmAlterar)

```

unit un_alteracao;
{PUBDIST}

interface

uses
    IAppForm, IWApplication, IWTypes, IWCompRectangle, IWCompButton,
    IWCompListbox, IWCompLabel, Classes, Controls, IWControl, IWCompEdit,
    FMTBcd, DB, SqlExpr, IWDBStdCtrls, DBClient, SimpleDS, SysUtils;

type
    TfmAlterar = class(TIWAppForm)
        IWButton1: TIWButton;
        IWButton2: TIWButton;
        IWRectangle1: TIWRectangle;
        IWLabel7: TIWLabel;
        lbCodigo: TIWLabel;
        DataSource1: TDataSource;
        SQLDataSet1: TSQLDataSet;
        edRazao: TIWEdit;
        IWLabel2: TIWLabel;
        edEndereco: TIWEdit;
        IWLabel3: TIWLabel;
        edCidade: TIWEdit;
        IWLabel4: TIWLabel;
        IWLabel5: TIWLabel;
        edUF: TIWComboBox;
        IWLabel6: TIWLabel;
        edEmail: TIWEdit;
        edCep: TIWEdit;
        IWLabel1: TIWLabel;
        SQLDataSet1COD_CLIENTE: TIntegerField;
        SQLDataSet1RAZAO_SOCIAL: TStringField;
    end;

```

```

    SQLDataSet1ENDERECO: TStringField;
    SQLDataSet1CIDADE: TStringField;
    SQLDataSet1ESTADO: TStringField;
    SQLDataSet1CEP: TStringField;
    SQLDataSet1EMAIL: TStringField;
    procedure IWButton2Click(Sender: TObject);
    procedure IWButton1Click(Sender: TObject);
public
end;

implementation
{$R *.dfm}

uses
    ServerController, DataModuleUnit;

procedure TfmAlterar.IWButton2Click(Sender: TObject);
begin
    Hide;
end;

procedure TfmAlterar.IWButton1Click(Sender: TObject);
begin
    try
        {Alterar Cliente}
        with DataModule1.SQLAlterar do
            begin
                ParamByName('pcodigo').Value:=StrToInt(lbCodigo.Text);
                ParamByName('prazao').Value:=edRazao.Text;
                ParamByName('pendereco').Value:=edEndereco.Text;
                ParamByName('pcidade').Value:=edCidade.Text;
                ParamByName('pestado').Value:=edUf.Items[edUf.ItemIndex];
                ParamByName('pcep').Value:=edCep.Text;
                ParamByName('pemail').Value:=edEmail.Text;
                ExecSQL;
            end;

            WebApplication.ShowMessage('Cliente Alterado com sucesso',smSameWindow);

        except
            WebApplication.ShowMessage('Houve um problema na alteracao do cliente',
smSameWindow);
        end;
        Hide;
    end;
end.

```

Listagem 12.15 un_manutencao (FmManutencao)

```

unit un_manutencao;

{PUBDIST}

interface

```

```

uses
  IWAppForm, IWApplication, IWTypes, Classes, Controls, IWControl,
  IWCompRectangle, Forms, IWContainer, IWRegion, FMTBcd, DB, SqlExpr,
  IWGrids, IWDBGrids, IWCompLabel, IWCompButton, IWCompEdit, SysUtils,
  IWHTMLControls;

type
  TfmManutencao = class(TIWAppForm)
    IWRectangle1: TIWRectangle;
    IWRegion1: TIWRegion;
    IWDBGrid1: TIWDBGrid;
    SQLPesquisa: TSQLQuery;
    Dsl: TDataSource;
    edPesquisa: TIWEdit;
    btPesquisa: TIWButton;
    IWLabel1: TIWLabel;
    btVolta: TIWButton;
    procedure btPesquisaClick(Sender: TObject);
    procedure btVoltaClick(Sender: TObject);
    procedure IWDBGrid1Columns3Click(ASender: TObject;
      const AValue: String);
    procedure IWDBGrid1Columns4Click(ASender: TObject;
      const AValue: String);
  public
    procedure Localiza(Cliente: integer);
  end;

implementation
{$R *.dfm}

uses
  ServerController, DataModuleUnit, un_alteracao;

procedure Tfmmanutencao.Localiza(Cliente:integer);
begin
  SQLPesquisa.Locate('cod_cliente', Cliente, []);
end;

procedure TfmManutencao.btPesquisaClick(Sender: TObject);
begin
  SQLPesquisa.SQL.Clear;
  SQLPesquisa.SQL.Add('SELECT * FROM TBCLIENTE WHERE RAZAO_SOCIAL LIKE
'+''''+edPesquisa.Text+'%'+''');
  SQLPesquisa.Open;
end;

procedure TfmManutencao.btVoltaClick(Sender: TObject);
begin
  Hide;
end;

procedure TfmManutencao.IWDBGrid1Columns3Click(ASender: TObject;
  const AValue: String);
var
  fmAlterar:TfmAlterar;
begin

```

```

fmAlterar:=TFmAlterar.Create(WebApplication);
{ Atribui o código a Query }
with fmAlterar do
begin
    fmAlterar.SQLDataSet1.Params[0].Value:=StrToInt(AValue);
    fmAlterar.SQLDataSet1.Open;
    edRazao.Text:=SQLDataSet1RAZAO_SOCIAL.AsString;
    edEndereco.Text:=SQLDataSet1ENDERECO.AsString;
    edCidade.Text:=SQLDataSet1CIDADE.AsString;
    edCEP.Text:=SQLDataSet1CEP.AsString;
    edUF.ItemIndex:=edUF.Items.IndexOf(SQLDataSet1ESTADO.AsString);
    edEmail.Text:=SQLDataSet1EMAIL.AsString;

    { Atribui os valores }
    fmAlterar.lbCodigo.Caption:=AValue;
    fmAlterar.Show;

end;
end;

procedure TfmManutencao.IWDBGrid1Columns4Click(ASender: TObject;
const AValue: String);
begin
    try
        with DataModule1.SQLExclui do
        begin
            ParamByName('pcodigo').Value:=StrToInt(AValue);
            ExecSQL;
        end;
        WebApplication.ShowMessage('Cliente Excluido com sucesso',smSameWindow);

    except
        WebApplication.ShowMessage('Houve um problema na exclusao do cliente',
smSameWindow);
    end;

end;

end.

```

Listagem 12.16 DataModuleUnit.pas

```

unit DatamoduleUnit;

interface

uses
    {$IFDEF Linux}QForms, {$ELSE}Forms, {$ENDIF}
    SysUtils, Classes, DBXpress, DB, SqlExpr, FMTBcd;

type
    TDataModule1 = class(TDataModule)
        ConexaoBD: TSQLConnection;
        SQLInclui: TSQLQuery;
        SQLAlterar: TSQLQuery;
        SQLExclui: TSQLQuery;
    private

```

```
public
end;

// Procs
function DataModule1: TDataModule1;

implementation
{$R *.dfm}

uses
    IWInit,
    ServerController;

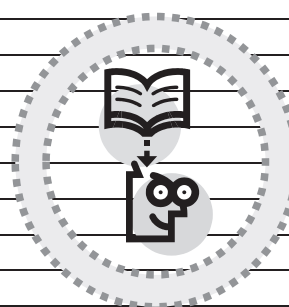
// Since we are threaded we cannot use global variables to store form / datamodule
// references
// so we store them in WebApplication.Data and we could reference that each time, but
// by creating
// a function like this our other code looks "normal" almost as if its referencing a
// global.
// This function is not necessary but it makes the code in the main form which
// references this
// datamodule a lot neater.
// Without this function ever time we would reference this datamodule we would use:
//   TDataModule1(WebApplication.Data).Datamodule.<method / component>
// By creating this procedure it becomes:
//   TDataModule1.<method / component>
// Which is just like normal Delphi code.
function DataModule1: TDataModule1;
begin
    Result := TUserSession(RWebApplication.Data).Datamodule1;
end;

end.
```

Anotações de Dúvidas



Preciso Revisar



Anotações Gerais

