Introdução

Neste artigo, será mostrado como desenvolver uma aplicação CGI para Web usando o Lazarus.

Uma aplicação CGI feita em Lazarus, é uma aplicação console (sem interface gráfica), que recebe requisições de um navegador web e devolve respostas (neste caso arquivos html).

Diferente dos scripts PHP, que são interpretados por um "engine", uma aplicação CGI feito em Lazarus é um arquivo compilado (binário) que só precisa dele mesmo e de um servidor Web configurado, corretamente, para funcionar.

Neste caso a aplicação escolhida para ser demonstrada é de um site que dá suporte a um sistema de Gestão Comercial (Programa de controle de estoque, financeiro, etc...), o qual possuirá uma área de Downloads, uma FAQ (Perguntas frequentes) e uma Área para notificação de erros ocorridos no sistema em questão.

Por se tratar de uma aplicação de complexidade intermediaria não há necessidade de detalhar a instalação dos requisitos.

Requisitos

Será preciso:

- Lazarus + FPC com os componentes Weblaz(FPWeb) e Zeos instalados;
- SGDB Postgresql;
- Servidor Apache;
- Conhecimento Básico em HTML.

Preparação

Crie um arquivo "html" com o link apontado para aplicação que se construir-se-a, nomeie-o como "index.html" e salve-o no localhost configurado no Apache.

```
<html>
<title>Suporte de Gestão Comercial</title>
  <body>
    <br><br><br><br><br><br>
    <center><a href="http://localhost/cgi-bin/suporte.cgi/monta_login">Suporte</a></center>
  </body>
</html>"
```

Banco de Dados

Crie um banco de dados e nele insira as tabelas com o campos necessários ao sistema. Pressupondo-se que haja conhecimento de como trabalhar com Postgresql, será descrito apenas o dicionário de dados. Pode-se usar qualquer outro SGDB, mas antes verifique e existência de componentes com suporte a ele no Lazarus.

- Acessos (Tabela de usuários autorizados a acessarem o sistema);
    - Id – serial (chave primária);
    - nome - character varying (20);
    - senha - character varying (8);

- faqs_cat (Tabela com as categorias das perguntas);
    - id – serial (chave primária);
    - categoria - character varying (30);
- faqs (Tabela com as perguntas e respostas);
    - id – serial (chave primária);
    - pergunta - character varying (5000);
    - resposta - character varying (5000);
    - categoria – character varying (30);
    - postar – boolean default false (campo que indica a aplicação se a pergunta está respondida);
- downloads (Tabela com os links para downloads);
    - id – serial (chave primária);
    - caption – character varyng (50);
    - link – character varyng (200);
- sessoes (Tabela com o registro das sessões autenticadas);
    - id - serial - (chave primária);
    - id_sessao - character varyng (100);
    - id_acesso - integer (chave estrangeira da tabela acessos);
- notificacoes (Tabela que registrará notificações de erros no sistema de Gestão);
    - id - serial - (chave primária);
    - id_acesso - integer (chave estrangeira da tabela acessos);
    - descricao_erro - character varying (5000);
    - data_hora - timestamp without time zone;
    - resolvido - boolean default false (Campo indicador de solução do erro);

Desenvolvimento

Para começar, abra o Lazarus e crie um novo projeto, na tela de seleção do tipo do projeto, selecione **“Aplicação CGI”,** se esta opção não estiver disponível, é porque o Weblaz (FPWeb) não foi instalado.

Crie uma pasta e salve o projeto com o nome “suporte”. Como sugestão a “unit” pode ser salva com o nome de “u_principal”.

Quando se cria uma aplicação CGI, não é criado um “TForm” como em uma aplicação normal, pois é criado um TFPWebModule que é descendente do DataModule.

Altere o nome do FPWebModule para WM. Uma vez que o sistema terá possuirá restrições de acesso, será necessário fazer um controle de sessões, então altere a propriedade “CreateSession” para “True”.

Clique no menu “projeto” / “Opções do Projeto”e na primeira aba coloque como título da aplicação o nome “Suporte”. Na opção configurações de saída, no campo “Nome do arquivo alvo”, coloque “suporte.cgi”.

No desenvolver da aplicação surgirão várias requisições de tipos diferentes, e, estas, aguardarão respostas de diferentes tipos, por isso será necessário trabalhar com o conceito de “actions” (ações).

Sendo assim, para cada possível requisição do navegador, será desenvolvida uma “action” especifica que será chamada pelo seu nome (prop. “name”) na URL.

Para criarmos as “actions”, selecione o WM e no Inspetor de Objetos, clique no botão da propriedade “Actions”. Um editor abrir-se-a. Clique no botão “Adicionar” do editor e altere a propriedade “nome” conforme a relação a seguir.

Abaixo, serão listadas todas as “actions” que deverão ser criadas e tratadas para que a aplicação tenha as funcionalidades desejadas, assim, nesse ponto do

desenvolvimento da aplicação poderá criá-las de uma só vez.

- **monta_login** – Cria a tela de login;
- **autentica_login** – Faz a verificação do usuário e senha no banco e registra a sessão no servidor;
- **logout** – Finaliza a aplicação e encerra a sessão;
- **monta_menu** – Cria a tela principal (Menu);
- **faqs –** Cria a tela com todas as perguntas e respostas do banco de dados;
- **faqs_filtra –** Cria a tela com as perguntas e respostas da categoria selecionada;
- **faqs_postar –** Insere uma pergunta no banco de dados;
- **monta_downloads –** Monta a tela com os links para download dos arquivos listado no banco de dados;
- **monta_notificacao –** Monta uma tela para envio de notificações de erros ocorridos no sistema de Gestão Comercial;
- **notifica_erro –** Insere uma notificação de erro ocorrido no sistema de Gestão Comercial no banco de dados.

Agora pode-se iniciar o tratamento das “actions”.

A primeira “action”, a “monta_login”, terá a função de exibir uma tela de acesso, com os campos Usuário, Senha e um botão para confirmar que terá um link apontando para a próxima “action” a “autentica_login”. Sendo assim pode-se dizer que sua resposta é estática, pois sempre que for requisitada, a “action” vai responder com um arquivo html idêntico. Desta forma pode-se usar a sua propriedade “Contents” que é um “TStringList” através do inspetor de objetos para tratá-la. Clique no botão da propriedade “Contents”, e um editor se abrirá. Coloque:

```
<html>
<title>Acesso ao Suporte</title>
<body bgcolor="#BEBEBE">
<form name="form1" method="post" action="http://localhost/cgi-bin/suporte.cgi/autentica_login">
  <br><br><br><br><br>
  <p align="center">
  <table width="30%" border="0">
     <tr>
       <td width="23%" height="20"><div align="right"><font face="Arial” size="2"><b>Usu&aacute;rio:</b></font></div>
     </td>
           <td width="77%" height="20"><p><font face="Arial" size="2">    <input type="text"
name="usuario" size="28" maxlength="20"></font></p>
     </td>
     </tr>
     <tr>
     <td width="23%" height="20"><div align="right"><font face="Arial" size="2"><b>Senha:</b></font></div>
     </td>
          <td width="77%" height="20"><p><font face="Arial" size="2">    <input type="password"
name="senha" size="28" maxlength="8"></font></p>
     </td>
      </tr>
   </table>
  </p>
   <p align="center">
    <input type="submit" name="Submit"  value="Acessar">
    <input type="reset" value="Limpar"
   </p>
 </form>
</body>
</html>
```

Antes de começar o tratamento da segunda “action”, escreveremos uma “procedure” genérica, pois quando for necessário dar qualquer mensagem ao

usuário, seja de erro, sucesso ou qualquer outra, tratando-se de uma aplicação sem interface gráfica, não terá à disposição o “showmessage” ou qualquer outra tela de diálogo.

Esta “procedure” precisará de uma variável global que se chamará “mensagem”. Declare-a assim:

```
var
 FPWebModule1: TFPWebModule1;
 mensagem : String
implementation

{$R *.lfm}

{ TFPWebModule1 }
```

Logo abaixo de { TFPWebModule1 }, escreva a “procedure”:

```
procedure TFPWebModule1.s_mensagem(mensagem,onclick:string);
begin
  mensagem  := '<html>'+
               '  <head>'+
               '    <title>Suporte - Mensagem</title>'+
               '  </head>'+
               '  <body bgcolor="#BEBEBE">'+
               '    <br><br><br><br><br><br><br><br><br><br><br><br>'+
               '      <form name="form1" method="post" action="'+onclick+'">'+
               '        <center>'+mensagem+'</center>'+
               '        <br><br>'+
               '        <center><input type="submit" name="voltar"  value="Voltar"></center>'+
               '      </form>'+
               '  </body>'+
               '</html>';
end;
```

Feito isso, pode-se partir para próxima “action”, a “autentica_login”. Nesta “action”, a aplicação receberá o nome e a senha que foram digitados pelo usuário no navegador, através do parâmetro “ARequest.ContentFields.Values”. Será necessário verificar a existência do nome de usuário no banco de dados e fazer a comparação da senha. Tudo esteja correto, gravará o id da “session” auto-criada, no banco de dados e dará acesso ao sistema.

Caso do nome de usuário digitado não seja encontrado, ou da senha digitada não conferir com a senha do banco de dados, uma mensagem de erro será exibida e nela, um botão com a função de retornar para a tela de login (para isso usaremos nossa “procedure” genérica “s_mensagem”).

A gravação da “session” tem a função de não permitir que usuários burlem o login, digitando uma URL com qualquer “action” posterior a “autentica_login”. Como é apenas um exemplo, será usada criptografia de senhas, porém é altamente recomendável que se use e dando preferência para as randômicas.

Note que de acordo com os resultados da pesquisa no banco, daremos uma resposta diferente será dada, então, pode-se dizer que esta tem uma resposta dinâmica, e portanto deverá ser tratada por código. Além disso será nessário uma conexão com o banco de dados.

Para conectar em nosso banco de dados, foi escolhida a biblioteca Zeos, usando a versão do SVN 7.0.0.

Coloque e configure os seguintes componentes na aplicação:

TZConnection : Faz a conexão com o banco
Name = Conexao
Conecte o componente com o banco criado

TZReadOnlyQuery: Responsável por efetuar consultas no banco de dados.

Name = QrConsulta
Connection = Conexao

TZReadOnlyQuery: Responsável por efetuar consultas no banco de dados.
Name = QrConsulta2
Connection = Conexao

TZQuery: Executa SQLs de inclusão e edição no Banco de Dados.
Name = QrExec
Connection = Conexao

Criou-se a conexão com o banco, duas “queries” para consultas e uma para execução de SQL (insert e update).

Agora já pode-se partir para o código.

No inspetor de objetos, na aba eventos, clique no botão do evento “OnRequest”, da “action” “autentica_login” e coloque o código que segue.

```
procedure TFPWebModule1.TFPWebActions1Request(Sender: TObject;
  ARequest: TRequest; AResponse: TResponse; var Handled: Boolean);
begin
  if ARequest.ContentFields.Values['usuario'] = '' then
    begin
      s_mensagem('&Eacute; obrigat&oacute;rio informar o usu&aacute;rio','javascript:history.go(-1)');
      AResponse.Content := mensagem;
      Exit;
    end;
  QrConsulta.Close;

QrConsulta.SQL.Text := 'SELECT id, senha FROM acessos WHERE nome
= '+QuotedStr(ARequest.ContentFields.Values['usuario']);
  QrConsulta.Open;
  if QrConsulta.RecordCount = 0 then
    begin //Nenhum usuário com o nome informado
      s_mensagem('Usu&aacute;rio n&atilde;o encontrado','javascript:history.go(-1)');
      AResponse.Content := mensagem;
      exit;
    end
  else
    begin //Usuário encontrado
      if not (QrConsulta.FieldByName('senha').Value = ARequest.ContentFields.Values['senha']) then
        begin //Senha incorreta
          s_mensagem('Senha incorreta','javascript:history.go(-1)');
          AResponse.Content := mensagem;
        end
      else
        begin //Senha confere
          //Limpa Sessões do usuário
          QrExec.Close;
          QrExec.SQL.Text := 'DELETE FROM sessoes WHERE id_acesso = '+QuotedStr(QrConsulta.FieldByName('id').Text);
          QrExec.ExecSQL;
          //Abrir Sessao
          QrExec.Close;
                QrExec.SQL.Text := 'INSERT INTO sessoes (id_sessao,id_acesso) VALUES ('+QuotedStr(Session.SessionID)
+','+QuotedStr(QrConsulta.FieldByName('id').Text)+')';
          QrExec.ExecSQL;
          AResponse.Contents.Clear;
          AResponse.Contents.Add('<html>');
          AResponse.Contents.Add('<title> Suporte - Menu Principal</title>');
          AResponse.Contents.Add('<body bgcolor="#BEBEBE">');
          AResponse.Contents.Add('<br><br><br><br>');
          AResponse.Contents.Add('<center><a href="http://localhost/cgi-bin/suporte.cgi/faqs">FAQs</a></center>');
          AResponse.Contents.Add('<br>');
              AResponse.Contents.Add('<center><a  href="http://localhost/cgi-bin/suporte.cgi/monta_downloads">Downloads</a></
center>');
          AResponse.Contents.Add('<br>');

AResponse.Contents.Add('<center><a  href="http://localhost/cgi-bin/suporte.cgi/monta_notificacao">Notifica&ccedil;&atilde;o de
Erros</a></center>');
          AResponse.Contents.Add('<br>');
```

```
        AResponse.Contents.Add('<center><a href="http://localhost/cgi-bin/suporte.cgi/logout">Sair do Sistema de Suporte</a></
center>');
       AResponse.Contents.Add('</body>');
       AResponse.Contents.Add('</html>');
     end;
  end;
end;
```

A próxima “action” a ser tratada será a “logout”, que será responsável pelo encerramento da aplicação. Nela, além de direcionar para página inicial (index.html), finalizará a “session”, removendo-a do banco de dados. Então, no evento “OnRequest” colocaremos o código abaixo.

```
procedure TFPWebModule1.TFPWebActions2Request(Sender: TObject;
  ARequest: TRequest; AResponse: TResponse; var Handled: Boolean);
var Usuario, Empresa : String;
begin
  //Deleta a sessão do banco
  QrExec.Close;
  QrExec.SQL.Text := 'DELETE FROM sessoes WHERE (id_sessao = '+QuotedStr(Session.SessionID)+')';
  QrExec.ExecSQL;
  //mata a sessão local
  Session.Terminate;
  //Exibe uma tela informativa
  AResponse.Content :=       '<Content-type: text/html\n\n>'+
                        '<html>'+
                        '  <head>'+
                        '    <title>Sess&atilde;o Finalizada</title>'+
                        '  </head>'+
                        '  <body bgcolor="#BEBEBE" text="#000000">'+
                        '    <br> <br><br><br><br><br><br><br><br><br><br>'+
                        '    <center>'+
                        '      <p><b>Sess&atilde;o Finalizada !</b></p>'+
                        '      <br><br><br>'+
                        '      <b><A HREF="http://localhost/index.html"> Home </A></b>'+
                        '    </center>'+
                        '  </body>'+
                        '</html>';
end;
```

Até aqui, foi feito o tratamento das “actions” responsáveis pelo acesso da aplicação. Agora implementar-se-a as funcionalidades da aplicação, mas antes de permitir a execução de qualquer “action” funcional, deve-se verificar no banco de dados se a “session” do “client” (usuário), encontra-se registrada (isto garante que o usuário efetuou o “login”).

Como esta verificação ocorrerá em todas “actions” que contém as funcionalidades da aplicação, crie uma função com este fim, para que não tenha que digitá-la em todas as “actions” que será necessária. O seu nome será “check_session”.

```
function TFPWebModule1.check_session():Boolean;
begin
  QrConsulta.SQL.Clear;
  QrConsulta.SQL.Text := 'SELECT id, id_acesso FROM sessoes WHERE id_sessao = '+QuotedStr(Session.SessionID);
  QrConsulta.Open;
  if QrConsulta.RecordCount > 0 then
    Result := True
  else
    Result := False;
end;
```

Criada nossa função de verificação, comece implementar as funcionalidades do sistema. Inicie pela “action” “monta_menu”, esta terá a função de montar o menu principal (o mesmo que é montado na “autentica_login”) quando retornar de alguma outra tela da aplicação.

```
procedure TFPWebModule1.TFPWebActions3Request(Sender: TObject;
  ARequest: TRequest; AResponse: TResponse; var Handled: Boolean);
begin
  if check_session()= true then
```

```
   begin
     AResponse.Contents.Clear;
     AResponse.Contents.Add('<html>');
     AResponse.Contents.Add('<title> Suporte - Menu Principal</title>');
     AResponse.Contents.Add('<body bgcolor="#BEBEBE">');
     AResponse.Contents.Add('<br><br><br><br>');
     AResponse.Contents.Add('<center><a href="http://localhost/cgi-bin/suporte.cgi/faqs">FAQs</a></center>');
     AResponse.Contents.Add('<br>');
              AResponse.Contents.Add('<center><a   href="http://localhost/cgi-bin/suporte.cgi/monta_downloads">Downloads</a></
center>');
     AResponse.Contents.Add('<br>');
      AResponse.Contents.Add('<center><a href="http://localhost/cgi-bin/suporte.cgi/monta_notificacao">Notifica&ccedil;&atilde;o
de Erros</a></center>');
     AResponse.Contents.Add('<br>');
        AResponse.Contents.Add('<center><a  href="http://localhost/cgi-bin/suporte.cgi/logout">Sair  do  Sistema  de  Suporte</a></
center>');
     AResponse.Contents.Add('</body>');
     AResponse.Contents.Add('</html>');
   end
 else
   begin
     AResponse.Contents.Clear;
     AResponse.Contents.Add('<html>');
     AResponse.Contents.Add('<title> Sess&atilde;o Expirada</title>');
     AResponse.Contents.Add('<body bgcolor="#BEBEBE">');
     AResponse.Contents.Add('<br><br><br><br>');
     AResponse.Contents.Add('<center><b>Esta Sess&atilde;o Expirou, efetue o acesso novamente !</b></center>');
     AResponse.Contents.Add('<br><br>');
       AResponse.Contents.Add('<center><b><A  HREF="http://localhost/cgi-bin/suporte.cgi/monta_login"> Efetuar  Login  </A></
b></center>');
     AResponse.Contents.Add('</body');
     AResponse.Contents.Add('</html');
   end;
end;
```

A próxima “action” a ser tratada será a “monta_downloads”, que montará uma tela contendo os links para downloads cadastrados no banco de dados. Mas antes vamos criar dentro do “localhost” uma pasta chamada “downloads”, dentro dela coloque alguns arquivos e, através de algum software de manutenção de bancos de dados PostgreSQL (eu usei o PGAdmin III) popule a tabela “downloads” com o caminho dos arquivos e o caption a ser exibido na tela de Downloads

No exemplo abaixo foi colocado alguns documentos compactados no formato “rar”

1. texto.rar;
2. planilha.rar;
3. imagem.rar;

A tabela de downloads ficou assim:

ID	Caption	Link
1	Texto	http://localhost/downloads/texto.rar
2	Planilha	http://localhost/downloads/planilha.rar
3	Imagem	"http://localhost/downloads/imagem.rar"

No evento “OnRequest” da “action” “monta_downloads”, coloque o código a seguir:

```
procedure TFPWebModule1.TFPWebActions7Request(Sender: TObject;
  ARequest: TRequest; AResponse: TResponse; var Handled: Boolean);
begin
  if check_session()= true Then
    begin
      AResponse.Contents.Clear;
      AResponse.Contents.Add('<html>');
      AResponse.Contents.Add('<title> Suporte - Downloads</title>');
      AResponse.Contents.Add('<body bgcolor="#BEBEBE">');
```

```
    AResponse.Contents.Add('<br><br><br>');
    AResponse.Contents.Add('<center><b>Links para Download</b></center>');
    AResponse.Contents.Add('<br>');

    QrConsulta.Close;
    QrConsulta.SQL.Text := 'SELECT * FROM downloads ORDER BY caption';
    QrConsulta.Open;
    QrConsulta.First;
    while not QrConsulta.EOF do
     begin

AResponse.Contents.Add('<p><center><a
href="'+QrConsulta.FieldByName('link').Text+'">'+QrConsulta.FieldByName('caption').Text+'</center></p>');
       QrConsulta.Next;
      end;

         AResponse.Contents.Add('<p><center><a  href="http://localhost/cgi-bin/suporte.cgi/monta_menu">Voltar  para  o  Menu</
center></p>');
    AResponse.Contents.Add('</body>');
    AResponse.Contents.Add('</html>');
  end
 else
  begin
    s_mensagem('Esta Sess&atilde;o Expirou, efetue o acesso novamente !','monta_login');
    AResponse.Content := mensagem;
  end;
end;
```

Prossega com o tratamento da “action” “monta_notificacao”, que terá a função de exibir uma tela com uma caixa de texto, para que o usuário escreva uma breve descrição do erro encontrado no software de Gestão Comercial.

Assim como a “action” “monta_login”, a “monta_notificação” dará uma resposta estática, isto é, responderá sempre da mesma forma, com o mesmo html, e sendo assim, pode ser tratada através da propriedade “Contents”.

Diante disso poderá surgir a seguinte dúvida:

Antes de exibir a tela de notificação, não devería ser verificado no banco de dados, se existe registro da sessão do usuário, garantindo que o mesmo passou pelo processo de login ?

Sim, mas há de se concordar que se algum espertinho digitar esta URL no navegador (da “action” “monta_notificacao”), a única coisa que acontecerá, é a tela de notificação ser exibida, mas não conseguirá notificar nada, pois a “action” que grava a notificação no banco de dados fará essa verificação previamente.

Retomando o desenvolvimento da aplicação, tratate a propriedade “Contents” da “action” “monta_notificacao” com o código html abaixo:

```
<html>
<title>Suporte - Notifica&ccedil;&atilde;o de Erro</title>
<body bgcolor="#BEBEBE">
 <form name="form1" method="post" action="http://localhost/cgi-bin/suporte.cgi/notificar_erro">
            <table width="100" align="center">
              <tr>
     <td>
      <br>
                        <b>Descreva sobre o erro encontrado:</b>
                        </td>
            </tr>
              <td>
      <textarea rows="6" cols="68" name="texto"> </textarea>
     </td>
    </tr>
            <tr align="right">
     <td>
      <p><input type="submit" name="Submit"  value="Enviar"></p>
                      </td>
            </tr>
            <tr>
              <td>
```

```
                <p><center><a href="http://localhost/cgi-bin/suporte.cgi/monta_menu">Voltar para o Menu</center></p>
                    </td>
                </tr>
              </table>
  </form>
</body>
</html>
```

A seguir trate a “action” “notificar_erro”, que terá a função de gravar a descrição do erro, o usuário, data e hora da notificação, no banco de dados. No evento “OnRequest” coloque o código abaixo:

```
procedure TFPWebModule1.TFPWebActions9Request(Sender: TObject;
  ARequest: TRequest; AResponse: TResponse; var Handled: Boolean);
var idusuario : Integer;
begin
  if check_session()= true then
    begin
      idusuario :=  QrConsulta.FieldByName('id').Value;
      QrExec.Close;
          QrExec.SQL.Text := 'INSERT INTO notificacoes (id_acesso,descricao_erro,data_hora) VALUES (:idacesso, :descricao,
NOW())';
      QrExec.Params[0].DataType := ftInteger;
      QrExec.Params[1].DataType := ftString;
      QrExec.Params[0].Value := idusuario;
      QrExec.Params[1].Value := ARequest.ContentFields.Values['texto'];
      QrExec.ExecSQL;
      s_mensagem('Notifica&ccedil;&atilde;o enviada com sucesso','monta_menu');
      AResponse.Content := mensagem;
    end
  else
    begin
      s_mensagem('Esta Sess&atilde;o Expirou, efetue o acesso novamente !','monta_login');
      AResponse.Content := mensagem;
    end;
end;
```

Note na “procedure” acima, que como criou-se os parâmetros do SQL em tempo de execução, para que se possa definir seu tipo, deve se acrescentar a unit “db” na clausula “uses” da u_principal, pois senão o Lazarus não reconhecerá seus tipos.

Trate o evento “OnRequest” da “action” “faqs_postar”, que gravará uma pergunta no banco de dados, para que a equipe do suporte a responda e libere sua publicação.

```
procedure TFPWebModule1.TFPWebActions6Request(Sender: TObject;
  ARequest: TRequest; AResponse: TResponse; var Handled: Boolean);
begin
  if check_session()= true then
    begin
      QrExec.Close;
      QrExec.SQL.Text := 'INSERT INTO faqs (pergunta) VALUES (:pergunta)';
      QrExec.Params[0].DataType := ftString;
      QrExec.Params[0].Value := ARequest.ContentFields.Values['pergunta'];
      QrExec.ExecSQL;
      s_mensagem('Pergunta enviada com sucesso','monta_menu');
      AResponse.Content := mensagem;
    end
  else
    begin
      s_mensagem('Esta Sess&atilde;o Expirou, efetue o acesso novamente !','monta_login');
      AResponse.Content := mensagem;
    end;
end;
```

Antes de prosseguir com o tratamento das “actions” restantes, sugere-se que populemos as tabelas “faqs_cat”, com os valores abaixo:

ID	Categoria
1	Configurações

2	Cadastros
3	Consultas

Por fim, trate as “actions” “faqs” e “faqs_filtra”. Ficaram por último, pois em ambas o código de criação do html é idêntico, a única diferença, fica por conta da consulta SQL que será efetuada. Para que não se escrevam códigos redundantes, crie uma “procedure” independente, que fará a montagem do html. Nas “procedures” de tratamento dos eventos “OnRequest” das duas “actions”, somente faça as consultas SQL e chame a “procedure” independente para montar o html.

Antes de escrever a “procedure”, declare a variavel global **FAQs** para transportar o html da “procedure” independente para, a “procedure” do evendo “OnRequest”.

```
var
 FPWebModule1: TFPWebModule1;
 mensagem, FAQs : String
implementation

{$R *.lfm}

{ TFPWebModule1 }
```

Feito isso, escreva a procedure independente, a qual será chamada de “monta_faqs” e a coloque logo abaixo da “function check_session” da seguinte forma:

```
procedure TFPWebModule1.monta_faq;
var cont : Integer;
   temp : TStringList;
begin
temp := TStringList.Create;
temp.Add('<html>');
temp.Add('  <head>');
temp.Add('    <title>FAQs</title>');
temp.Add('     <STYLE TYPE="text/css">');
temp.Add('       textarea');
temp.Add('       {');
temp.Add('        font-family: Arial, Helvetica, sans-serif;');
temp.Add('        font-size: 9pt;');
temp.Add('        color:#000066;');
temp.Add('        border-right:1px solid #696969;');
temp.Add('      border-left:1px solid #696969;');
temp.Add('      border-top:1px solid #696969;');
temp.Add('      border-bottom:1px solid #696969;');
temp.Add('       }');
temp.Add('  select');
temp.Add('       {');
temp.Add('        font-family: Arial, Helvetica, sans-serif;');
temp.Add('        font-size: 9pt;');
temp.Add('        color:#000066;');
temp.Add('        border-right:1px solid #000066;');
temp.Add('        border-left:1px solid #000066;');
temp.Add('        border-top:1px solid #000066;');
temp.Add('        border-bottom:1px solid #000066;');
temp.Add('       }');
temp.Add('     </STYLE>');
temp.Add('  </head>');
temp.Add('  <body bgcolor="#BEBEBE">');
temp.Add('    <table width="100%" border="1" bgcolor="#BEBEBE" bordercolor="#696969">');
temp.Add('      <tr>');
temp.Add('        <td  width="65%" align="center">');
temp.Add('          <br>');
temp.Add('      <form name="form1" method="post" action="faqs_filtra">');
temp.Add('            <b> Exibir apenas da categoria: </b>');
temp.Add('              <select size="1" border="1" name="categoria">');
temp.Add('                <option value="Todas">Todas</option>');

//Looping com as categorias
QrConsulta.First;
```

```
while not QrConsulta.EOF do
  begin
temp.Add('               <option
value="'+QrConsulta.FieldByName('categoria').Value+'">'+QrConsulta.FieldByName('categoria').Value+'</option>');
QrConsulta.Next;
  end;

temp.Add('             </select>');
temp.Add('             <input type="submit" name="SubmitCat"  value="Filtrar">');
temp.Add('          </form>');
temp.Add('        </td>');
temp.Add('        <td width="35%" align="center">');
temp.Add('          <b><A HREF="monta_menu"> Retornar ao Menu </A></b>');
temp.Add('        </td>');
temp.Add('      </tr>');
temp.Add('    </table>');
temp.Add('    <table align="center" width="100%" border="0" bgcolor="#696969">');
temp.Add('      <tr>');
temp.Add('        <td width="100%"align="center">');
temp.Add('          <font color="#FFFFFF" face="Verdana"><b> FAQs - Perguntas Freq&uuml;entes </b></font>');
temp.Add('        </td>');
temp.Add('      </tr>');
temp.Add('    </table>');
temp.Add('    <table width="100%" bgcolor="#FFFFFF" border="1" bordercolor="#696969">');
temp.Add('      <tr>');
temp.Add('        <td>');
temp.Add('          <br>');

QrConsulta2.First;
cont := 1;
while not QrConsulta2.EOF do
  begin
    if cont > 1 then
Temp.Add('          <hr>');
Temp.Add('          <p><b> '+QrConsulta2.FieldByName('pergunta').Value+'</p></b>');
Temp.Add('          '+QrConsulta2.FieldByName('resposta').Value);
    QrConsulta2.Next;
    inc(cont);
  end;

temp.Add('            <br>');
temp.Add('        </td>');
temp.Add('      </tr>');
temp.Add('    </table>');
temp.Add('    <table width="100%" bgcolor="#BEBEBE" border="1" bordercolor="#696969">');
temp.Add('      <tr>');
temp.Add('        <td>');
temp.Add('          <form name="form1" method="post" action="faqs_postar">');
temp.Add('            <table width="100" align="center">');
temp.Add('              <tr>');
temp.Add('                <td>');
temp.Add('                  <br>');
temp.Add('                  <b>Envie-nos sua d&uacute;vida:</b>');
temp.Add('                </td>');
temp.Add('              </tr>');
temp.Add('              <tr>');
temp.Add('                <td>');
temp.Add('                  <textarea rows="3" cols="90" name="pergunta"> </textarea>');
temp.Add('                </td>');
temp.Add('              </tr>');
temp.Add('              <tr align="right">');
temp.Add('                <td>');
temp.Add('                  <p><input type="submit" name="Submit"  value="Enviar"></p>');
temp.Add('                </td>');
temp.Add('              </tr>');
temp.Add('            </table>');
temp.Add('          </form>');
temp.Add('        </td>');
temp.Add('      </tr>');
temp.Add('    </table>');
temp.Add('  </body>');
temp.Add('</html>');
FAQS := Temp.Text;
end;
```

Com a “procedure” independente criada, só restou fazer o tratamento do evento “OnRequest” das “actions” “faqs” e “faqs_ filtra” que, em suas implementações

conterão apenas a verificação de sessão, a execução das consultas SQL, o chamado da “monta_faqs” e a emissão da resposta (html).

Prossiga com a codificação dos eventos como será mostrado a seguir.

Para o tratamento da “action” “faqs”:

```
procedure TFPWebModule1.TFPWebActions4Request(Sender: TObject;
  ARequest: TRequest; AResponse: TResponse; var Handled: Boolean);
begin
  if check_session() = true Then
    begin
      //Sessão OK
      QrConsulta.Close;
      QrConsulta.SQL.Text := 'SELECT id,categoria FROM faqs_cat ORDER BY categoria';
      QrConsulta.Open;
      QrConsulta2.Close;
      QrConsulta2.SQL.Text := 'SELECT * FROM faqs WHERE postar = True';
      QrConsulta2.Open;
      monta_faq;
      AResponse.Content := FAQs;
    end
  else
     begin
      s_mensagem('Esta Sess&atilde;o Expirou, efetue o acesso novamente !','monta_login');
      AResponse.Content := mensagem;
    end;
end;
```

E para a “action” “faqs_filtra”:

```
procedure TFPWebModule1.TFPWebActions5Request(Sender: TObject;
  ARequest: TRequest; AResponse: TResponse; var Handled: Boolean);
begin
  if check_session() = true Then
    begin
      //Sessão OK
      QrConsulta.Close;
      QrConsulta.SQL.Text := 'SELECT id,categoria FROM faqs_cat ORDER BY categoria';
      QrConsulta.Open;
      QrConsulta2.Close;

QrConsulta2.SQL.Text := 'SELECT * FROM faqs WHERE postar = True AND categoria
= '+QuotedStr(ARequest.ContentFields.Values['categoria']);
      QrConsulta2.Open;
      monta_faq;
      AResponse.Content := FAQs;
    end
  else
     begin
      s_mensagem('Esta Sess&atilde;o Expirou, efetue o acesso novamente !','monta_login');
      AResponse.Content := mensagem;
    end;
end;
```

Pronto !!!

Talvés voces esteja perguntando:

Ficaram faltando implementações no sistema, como por exemplo, uma tela onde serão respondidas as FAQs, onde serão lidas as notificações, onde serão disponibilizados os downloads ?

É de se concordar, foi feita apenas a parte dos usuários do sistema de gestão comercial. A parte usada pelo pessoal do suporte será assunto para nosso próximo artigo”.

O Código fonte desta aplicação pode ser baixado "aqui"

Conclusão

Com base neste artigo, podemos ver a grande flexibilidade que o FW FPWeb nos dá, e a variedade de aplicações que poderemos desenvolver utilizando-se destes recursos.

A aplicação desenvolvida aqui, não conta com implementação de aperfeiçoamentos visuais, foi feito o básico, mas para isso podemos utilizar imagens, animações, CSS, JavaScript e acredito que até Flex. Mas deixo esta parte para a criatividade de cada um, tenho certeza que deste artigo nascerão várias aplicações CGI com maravilhosos recursos visuais e funcionais.

Caso alguem queira conhecer o projeto que estou desenvolvendo, pode ser acessado através do link: http://www.yieldbetter.com.br , àrea restrita, usuário: *teste*, senha: *111*.

Rafael Elias