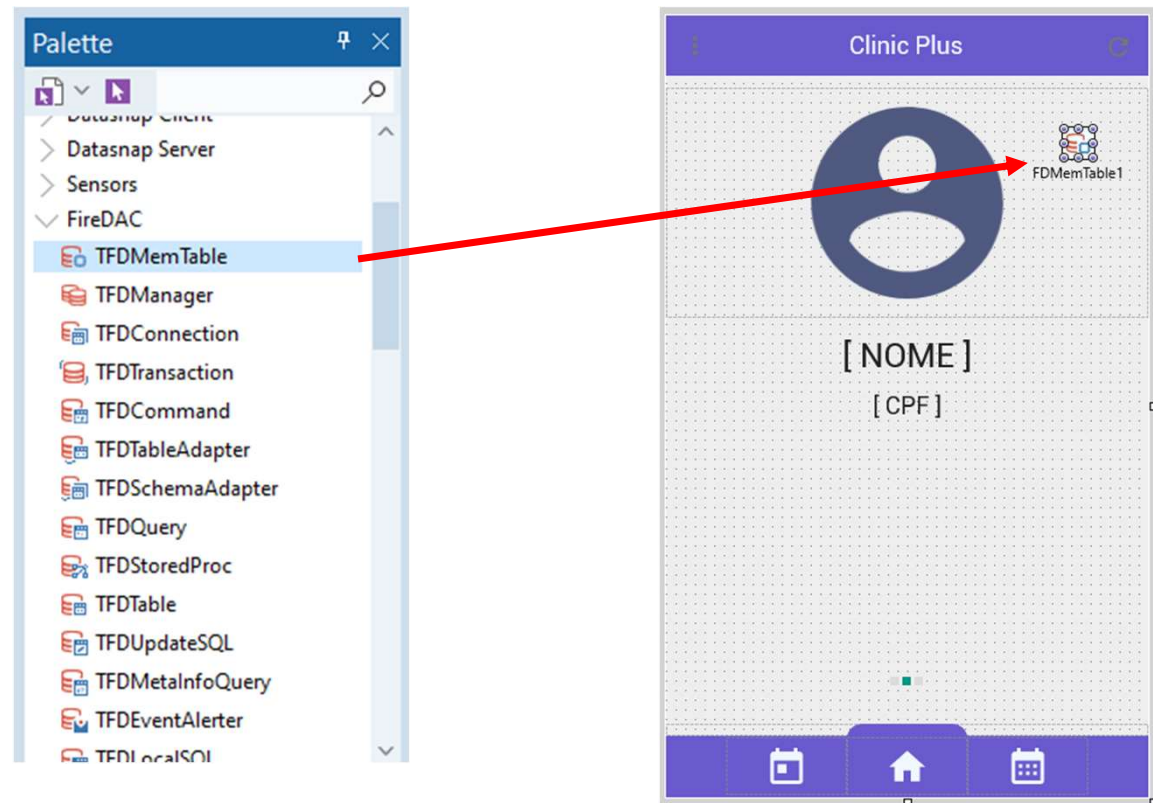


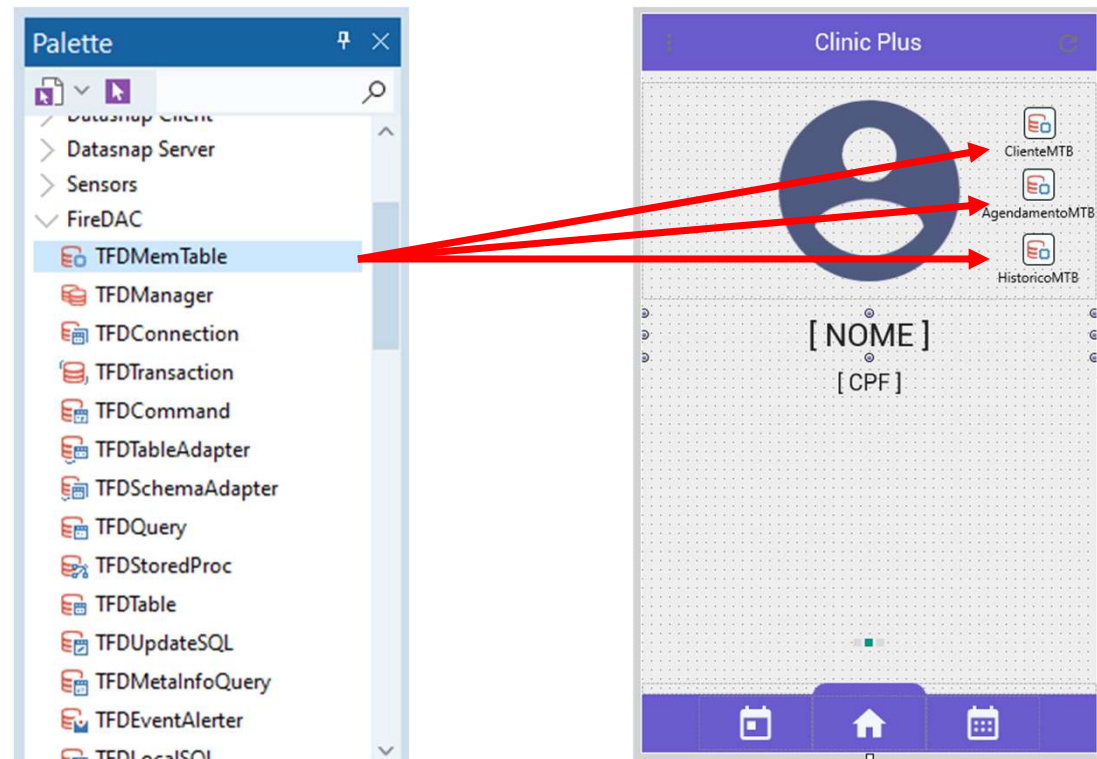
Dataset – FDMemTable – Parte1

- Insira um TFDMemTable para armazenar temporariamente os dados recebidos do Back-end.



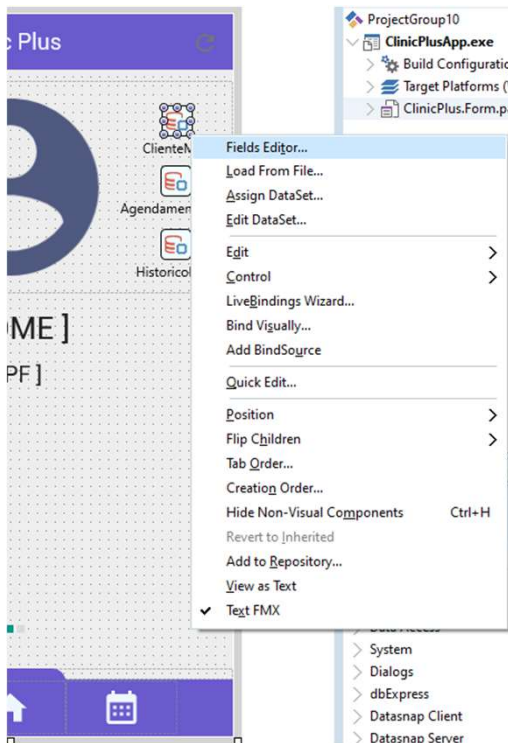
Dataset - FDMemTable – Parte2

- Insira um TFDMemTable para armazenar temporariamente os dados recebidos do Back-end.

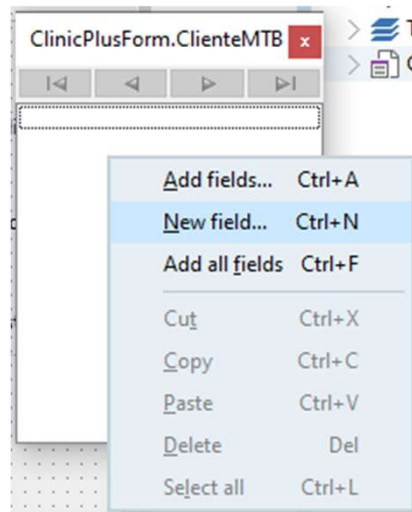


Dataset - FDMemTable – Parte3

Clique com o botão direito sobre o FDMemTable e abra o **Fields Editor**

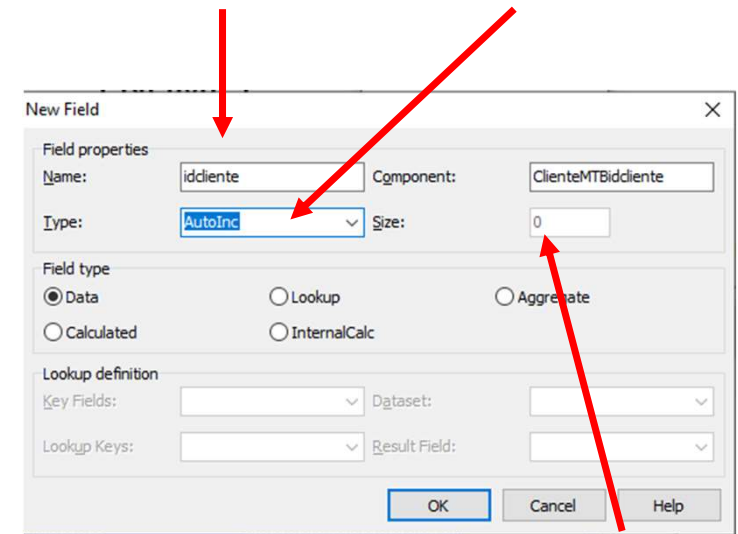


Clique com o botão direito sobre o **Fields Editor** e crie um novo campo.



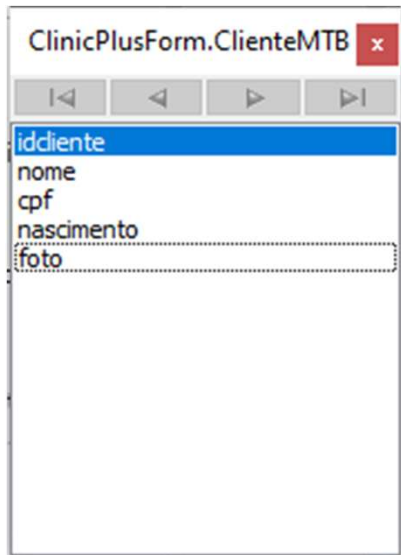
Nome do campo.
Precisa ser igual ao que aparece no JSON

Tipo do campo.
Precisa ser igual ao que está no banco de dados



Quando for um campo de texto, especifica a quantidade de caracteres.

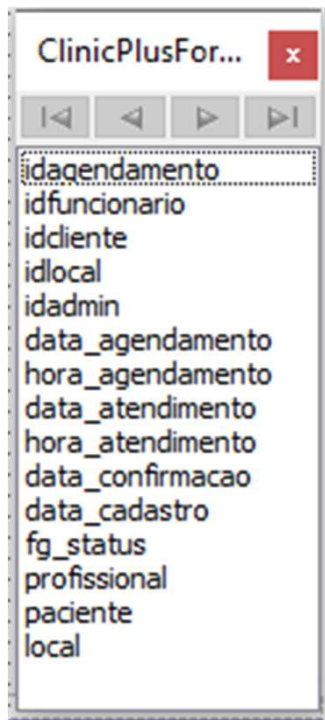
ClienteMTB – Lista de campos



The screenshot shows a window titled "ClinicPlusForm.ClienteMTB" with a close button. Below the title bar are four navigation buttons: a first button with a vertical bar and left arrow, and three buttons with left and right arrows. Below these is a list of fields: "idcliente" (highlighted in blue), "nome", "cpf", "nascimento", and "foto" (which has a dashed border). The "foto" field is followed by a large empty rectangular area.

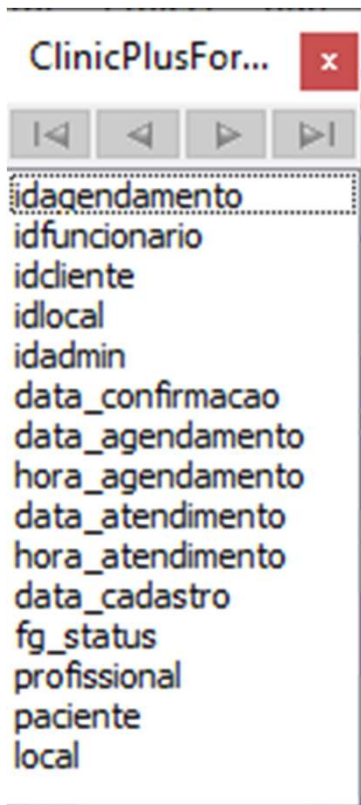
- **idcliente => AutoIncField**
 - ProviderFlags = [pflnWhere, pflnKey]
 - ReadOnly = True
- **Nome => String**
 - Required = True
 - Size = 100
- **Cpf => String**
 - Required = True
 - Size = 14
- **Nascimento => DateTime**
- **Foto => TBlobField**

AgendamentoMTB – Lista de campos



- **idagendamento => AutoIncField**
 - ProviderFlags = [pflnUpdate, pflnWhere, pflnKey]
 - ClientAutoIncrement = False
 - IdentityInsert = True
- **idfuncionario => Integer**
 - Required = True
- **idcliente => Integer**
 - Required = True
- **idlocal => Integer**
 - Required = True
- **idadmin => Integer**
 - Required = True
- **data_agendamento => Date**
 - Required = True
- **hora_agendamento => Time**
 - Required = True
- **data_atendimento => Date**
- **hora_atendimento => Time**
- **data_confirmacao => DateTime**
- **data_cadastro => DateTime**
- **fg_status => String**
 - Size = 1
- **profissional => String**
 - Size = 100
- **paciente => String**
 - Size = 100
- **local => String**
 - Size = 50

HistoricoMTB – Lista de campos

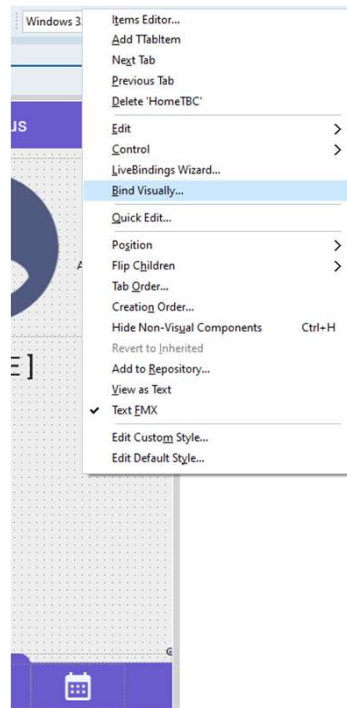


- **idagendamento => AutoIncField**
 - ProviderFlags = [pfInWhere, pfInKey]
 - ReadOnly = True
- **idfuncionario => Integer**
 - Required = True
- **idcliente => Integer**
 - Required = True
- **idlocal => Integer**
 - Required = True
- **idadmin => Integer**
 - Required = True
- **data_confirmacao => DateTime**
- **data_agendamento => Date**
 - Required = True
- **hora_agendamento => Time**
 - Required = True
- **data_atendimento => Date**
- **hora_atendimento => Time**
- **data_cadastro => DateTime**
- **fg_status => String**
 - Size = 1
- **profissional => String**
 - Size = 100
- **paciente => String**
 - Size = 100
- **local => String**
 - Size = 50

LiveBindings Designer

O LiveBindings Designer permite associar (“ligar”) componentes e suas propriedades, uns aos outros.

Clique com o botão direito sobre o formulário e escolha a opção “Bind Visually”



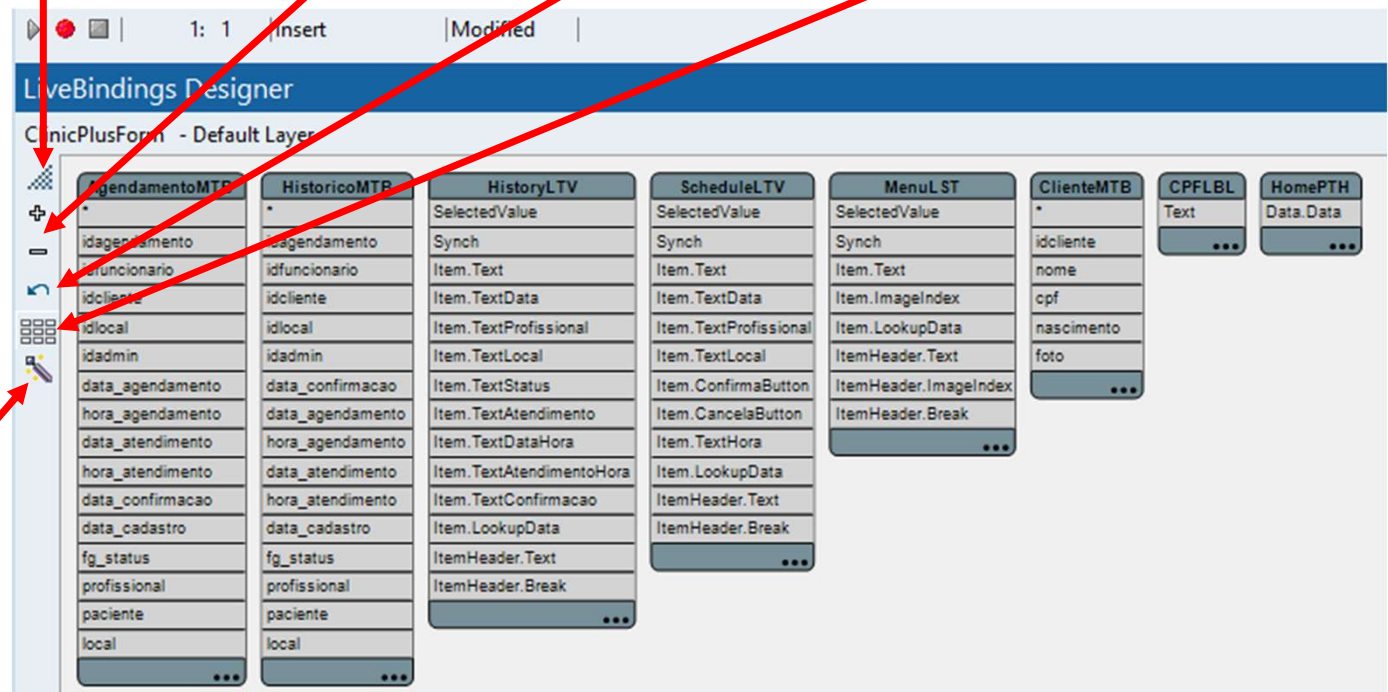
Ajusta o zoom para caber todos

Aumenta/Diminui o Zoom

Zoom para 100%

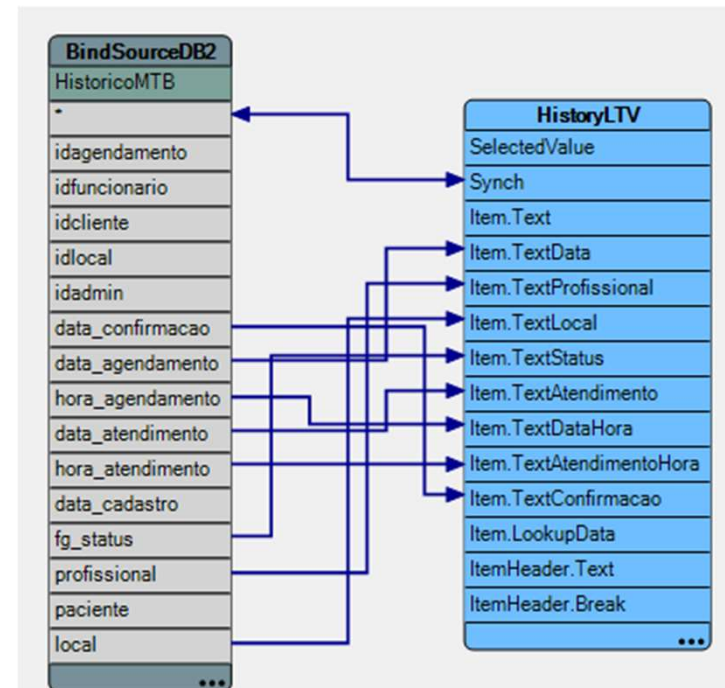
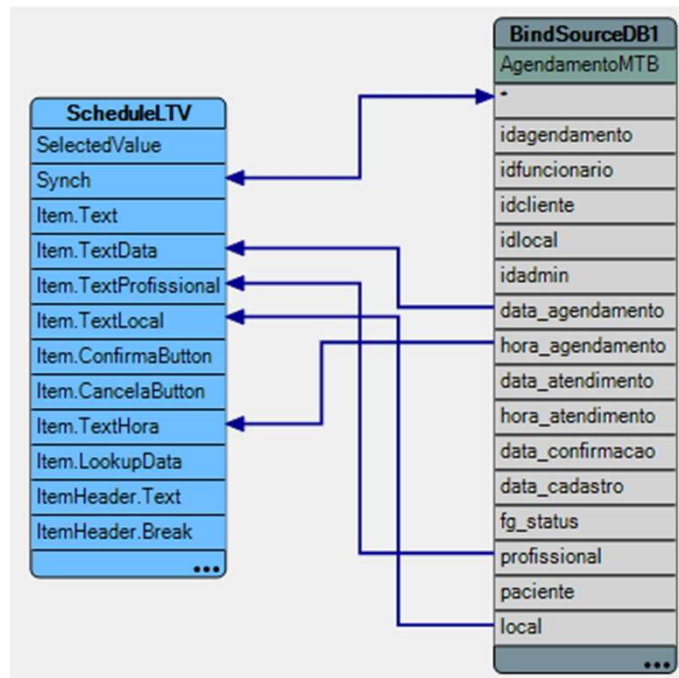
Organizar elementos

Wizard



LiveBindings Designer

Realize as associações, conforme abaixo



ScheduleLYT - OnClick

```
- procedure TClinicPlusForm.ScheduleLYTClick(Sender: TObject);
306 begin
    // Animação do retângulo de navegação
    TAnimator
    .AnimateFloat( AnimerCT, // Componente do formulário a ser animado
310                'position.x', // Propriedade do componente a ser animado
                ScheduleLYT.Position.X, // Valor da prop. ao final da animação
                0.5, // Duração da animação
                TAnimationType.Out, // tipo da animação (entrada/saida ou ambos)
                TInterpolationType.Bounce // tipo da interpolação da animação
    );
    // Animação do Tabcontrol
    ContentTBC
    .SetActiveTabWithTransitionAsync(
        ScheduleTBC, // Aba que será exibida
320        TTabTransition.Slide, // transição estilo escorrega
        TTabTransitionDirection.Reversed, // Animação da esquerda para direita (reversa)
        nil // ponteiro para execução de função ao terminar transição.
    );
end;
```

HistoryLYT - OnClick

```
199 procedure TclinicPlusForm.HistoryLYTClick(Sender: TObject);
200 begin
    . // Animação do retângulo de navegação
    . TAnimator
    . .AnimateFloat( AnimerCT, // Componente do formulário a ser animado
    . . 'position.x', // Propriedade do componente a ser animado
    . . HistoryLYT.Position.X, // Valor da prop. ao final da animação
    . . 0.5, // Duração da animação
    . . TAnimationType.Out, // Tipo da animação (entrada/saída ou ambos)
    . . TInterpolationType.Bounce // Tipo da interpolação da animação
    . );
210 . // Animação do Tabcontrol
    . ContentTBC
    . .SetActiveTabWithTransitionAsync(
    . . HistoryTBC, // Aba que será exibida
    . . TTabTransition.Slide, // Transição estilo escorrega
    . . TTabTransitionDirection.Normal, // Animação da direita para esquerda (normal)
    . . nil // ponteiro para execução de função ao terminar transição.
    . );
    . end;
```

HomeLYT - OnClick

```
220 procedure TClinicPlusForm.HomeLYTClick(Sender: TObject);
.   var
.     TabDirection: TTabTransitionDirection;
.   begin
.     // APENAS PARA O BOTÃO DO CENTRO SERÁ NECESSARIO
.     // AJUSTAR O SENTIDO DA ANIMAÇÃO DE ACORDO COM
.     // A PAGINA ATUAL (direita p/esquerda ou
.     // da esquerda p/ direita)
.     if ContentTBC.ActiveTab.Index > HomeTBC.Index then
.       TabDirection := TTabTransitionDirection.Reversed
230   else
.     TabDirection := TTabTransitionDirection.normal;
.
.     // animação do retângulo de animação
.     TAnimator
.       .AnimateFloat( AnimeRCT, // Componente do formulario a ser animado
.         'position.x', // Propriedade do componente a ser animado
.         HomeLYT.Position.X, // Valor da prop. ao final da animação
.         0.5, // Duração da animação
.         TAnimationType.Out, // Tipo da animação (entrada/saida ou ambos)
240       TInterpolationType.Bounce // Tipo da interpolação da animação
.       );
.     // animação do tabcontrol
.     ContentTBC
.       .SetActiveTabWithTransitionAsync(
.         HomeTBC, // Aba que será exibida
.         TTabTransition.Slide, // Transição estilo escorrega
.         TabDirection, // Animação normal ou reversa, depende do IF acima
.         nil // Ponteiro para execução de função ao terminar transição
.       );
250 end;
```

Declaração de Bibliotecas, Constantes e Variáveis

```
const
    EnderecoServidor = 'http://192.168.0.110:9000/';
    // EnderecoServidor = 'http://localhost:9000/';
var
    ClinicPlusForm: TClinicPlusForm;
    UserID: Integer; // usado para facilitar os testes.
-

implementation

uses FMX.Ani, RESTRequest4D, DataSet.Serialize, System.Threading;

{$R *.fmx}
```

Declaração de Bibliotecas

Declaramos em baixo do **implementation**, mas também poderíamos ter declarado no **uses** do **interface**

Necessário para
TRequest.New...

Necessário para
Itask,
TTask.Run,
TThread.Synchronize
Etc.

```
implementation
```

```
uses FMX.Ani, RESTRequest4D, DataSet.Serialize, System.threading;
```

```
{ $R *.fmx }
```

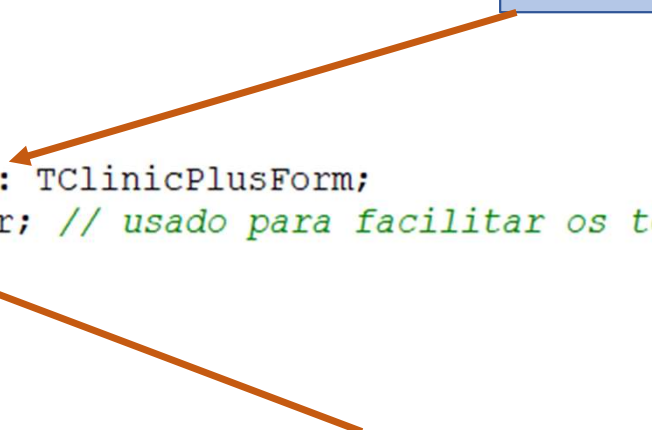
Necessário para
TAnimator.AnimateFloat

Necessário para
DataSet.ToJSONObject(),
AgendamentoMTB.ToJSONObject(),
TRequest.New.DataSetAdapter(HistoricoMTB),
Etc.

Declaração Variáveis

```
·  
120  var  
·    ClinicPlusForm: TClinicPlusForm;  
·    UserID: Integer; // usado para facilitar os testes.  
·  
·  
·  
·  implementation
```

Variável do formulário declarada pelo Delphi



ID do usuário, usado para testes de **requisições sem token**

Declaração Constantes

Contém o endereço do servidor Back-end necessário para requisição Rest

```
116  const
      // EnderecoServidor = 'http://192.168.0.110:9000/';
      EnderecoServidor = 'http://localhost:9000/';
      var
120  ClinicPlusForm: TClinicPlusForm;
```

Usamos localhost para testes no **Windows** onde Back-end e Front-end rodam na mesma máquina.
Usamos o endereço de IP (192.168.0.X) para testes com Wifi, onde **Back-end roda no computador Windows** e o **Front-end roda no celular Android**.

GetAgendamentoAtivo

Declare a função na seção publica da classe

```
private
{ Private declarations }
public
{ Public declarations }
procedure GetAgendamentoAtivo(const User: Integer); // requisição de agendamentos
```

Ctrl+Shift+C para o Delphi criar a implementação da função

```
procedure TclinicPlusForm.GetAgendamentoAtivo(const User: Integer);
begin
end;
```

GetAgendamentoAtivo

Programa a requisição ao servidor Back-end.

Considerando **EnderecoServidor=http://localhost:9000/**, a rota acessada fica:
http://localhost:9000/agendamento

```
procedure TClinicPlusForm.GetAgendamentoAtivo(const User: Integer);  
begin  
    TRequest.New.BaseURL(EnderecoServidor+'agendamento') // URL da API  
        .AddParam('fg_status','a') // QueryParam - Filtra apenas status Ativo  
        .AddParam('idcliente',User.ToString) // QueryParam - Filtra apenas pertencente ao usuario  
        .Accept('application/json') // tipo de dados da resposta que esperamos  
        .DataSetAdapter(AgendamentoMTB) // Conversão de JSON para DATASET  
        .Get; // Verbo da requisição  
end;
```

GetHistorico

Declare a função na seção publica da classe

```
private
{ Private declarations }
public
{ Public declarations }
procedure GetAgendamentoAtivo(const User: Integer); // requisição de agendamentos
procedure GetHistorico(const User: Integer); // requisição de historico com todos agendamentos
```

Ctrl+Shift+C para o Delphi criar a implementação da função

```
procedure TclinicPlusForm.GetHistorico(const User: Integer);
begin
end;
```

GetHistorico

Programa a requisição ao servidor Back-end.

Considerando **EnderecoServidor=http://localhost:9000/**, a rota acessada fica:
http://localhost:9000/agendamento

```
procedure TClinicPlusForm.GetHistorico(const User: Integer);  
begin  
    TRequest.New.BaseURL(EnderecoServidor+'agendamento') // URL da API  
    .AddParam('idcliente',User.ToString) // QueryParam - apenas pertencente ao usuario  
    .Accept('application/json') // tipo de dados da resposta que esperamos  
    .DataSetAdapter(HistoricoMTB) // Conversão de JSON para DATASET  
    .Get; // Verbo da requisição  
end;
```

Percebeu a diferença do **GetHistorico** em relação ao **GetAgendamentoAtivo**?

GetCliente e LoadCliente

Declare a função na seção publica da classe

```
private
{ Private declarations }
public
{ Public declarations }
procedure GetAgendamentoAtivo(const User: Integer); // requisição de agendamentos
procedure GetHistorico(const User: Integer); // requisição de historico com todos agendamentos
110 procedure GetCliente(const ID: Integer); // requisição de nome, cpf e foto do usuario.
procedure LoadCliente(const ID: Integer); // carrega nome, cpf e foto do usuario
```

Ctrl+Shift+C para o Delphi criar a implementação da função

```
procedure TclinicPlusForm.GetCliente(const ID: Integer);
begin
end;

procedure TclinicPlusForm.LoadCliente(const ID: Integer);
begin
end;
```

GetCliente

Programa a requisição ao servidor Back-end.

Considerando **EnderecoServidor=http://localhost:9000/**, a rota acessada fica:
http://localhost:9000/cliente

```
procedure TClinicPlusForm.GetCliente(const ID: Integer);  
begin  
    TRequest.New.BaseURL(EnderecoServidor+'cliente') // URL da API  
    .ResourceSuffix(ID.ToString) // diciona /1 na url  
    .Accept('application/json') // tipo de dados da resposta que esperamos  
    .DataSetAdapter(ClienteMTB) // Conversão de JSON para DATASET  
    .Get; // Verbo da requisição  
end;
```


LoadCliente

Não fizemos **Binds** para os componentes de **nome, CPF e Foto** do usuário.
Então fazemos seu carregamento na boa e velha moda antiga, **com programação**.

```
procedure TClinicPlusForm.LoadCliente(const ID: Integer);
var
  FotoStream: TMemoryStream;
  BrushBmp: TBrushBitmap;
begin
  GetCliente(ID); // requisição no Backend (API)

  // usar synchronize apenas com a certeza de que LoadCliente será chamando dentro de
  // uma thread diferente da thread principal.
  TThread.Synchronize(TThread.CurrentThread, procedure
  begin
    NameLBL.Text := ClienteMTBnome.AsString; // grava nome no formulario
    CPFLLBL.Text := ClienteMTBCPF.AsString; // grava CPF no formulario

    FotoStream := TMemoryStream.Create; // Cria stream para ler foto
    BrushBmp := TBrushBitmap.Create; // Cria Brush para desenhar foto no TCircle

    try
      ClienteMTBFoto.SaveToStream(FotoStream); // Lê a foto do campo
      BrushBmp.Bitmap.LoadFromStream(FotoStream); // Desenha a foto no brush
      BrushBmp.WrapMode := TWrapMode.TileStretch; // Ajusta imagem ao tamanho do componente
      Circle1.Fill.Bitmap.Assign(BrushBmp); // Desenha imagem no componente.
    finally
      // libera variáveis temporárias utilizadas no processo de exibir a foto.
      FotoStream.Free;
      BrushBmp.Free;
    end;
  end);
end;
```


ChangeSchedule

Declare a função na seção publica da classe

```
private
{ Private declarations }
public
{ Public declarations }
procedure GetAgendamentoAtivo(const User: Integer); // requisição de agendamentos
procedure GetHistorico(const User: Integer); // requisição de historico com todos agendamentos
110 procedure GetCliente(const ID: Integer); // requisição de nome, cpf e foto do usuario.
procedure LoadCliente(const ID: Integer); // carrega nome, cpf e foto do usuario

procedure ChangeSchedule(const AID: Integer; JSON: TJSONObject);
```

Ctrl+Shift+C para o Delphi criar a implementação da função

```
procedure TclinicPlusForm.ChangeSchedule(const AID: Integer; JSON: TJSONObject);
begin
end;
```

ChangeSchedule

Programa a requisição ao servidor Back-end.

Considerando **EnderecoServidor=http://localhost:9000/**, a rota acessada fica:
http://localhost:9000/cliente

```
procedure TClinicPlusForm.ChangeSchedule(const AID: Integer; JSON: TJSONObject);  
begin  
    TRequest.New.BaseURL(EnderecoServidor+'agendamento') // URL da API  
    .ResourceSuffix(AID.ToString) // ID do registro  
    .AddBody(JSON, False) // JSON com registro atualizado  
    .Accept('application/json') // tipo de dados da resposta que esperamos  
    .Put; // Verbo da requisição  
end;
```

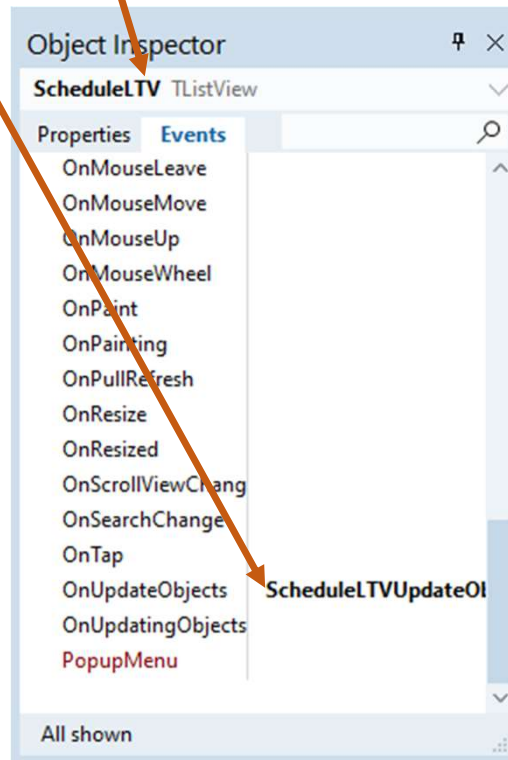
Percebeu a diferença do **ChangeSchedule** em relação ao **GetHistorico** e **GetAgendamentoAtivo**?

AtualizarBTN - OnClick

```
procedure TClinicPlusForm.AtualizarBTNClick(Sender: TObject);  
begin  
    // Atualiza registros em Thread separada  
    TTask.Run(procedure  
    begin  
        GetAgendamentoAtivo(UserID);  
        GetHistorico(UserID);  
    end);  
end;
```

ScheduleLTV - OnUpdateObjects

Crie o evento **OnUpdateObjects**.



FMX.ListView.TListView.OnUpdateObjects

Up to Parent: TListView

Delphi

```
property OnUpdateObjects: TAppearanceListView.TItemEvent read FOnUpdateObjects write FOnUpdateObjects;
```

C++

```
__property OnUpdateObjects;
```

Properties

Type	Visibility	Source
event	published	FMX.ListView.pa FMX.ListView.hp

Segundo a documentação, o evento é disparado imediatamente após o componente **ListView** ser **atualizado**.

Description

FMX.ListView.TListView.OnUpdateObjects inherits from FMX.ListView.TAppearanceListView.OnUpdateObjects. All content below this line refers to FMX.ListView.TAppearanceListView.OnUpdateObjects.

Occurs immediately after the list view component is updated.

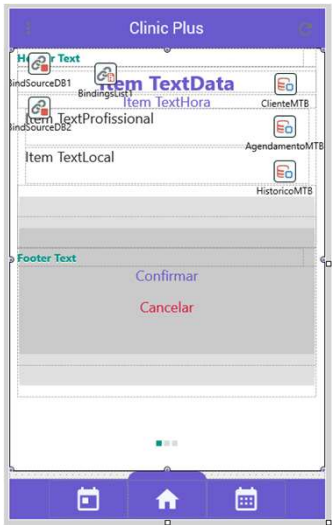
Write an **OnUpdateObjects** event handler to provide additional functionality after updating the list view component.

OnUpdateObjects is an event of type **TItemEvent**.

See Also

- [FMX.ListView.TListViewBase.TListItemEvent](#)
- [FMX.ListView.TAppearanceListView.OnUpdatingObjects](#)

ScheduleLTV - OnUpdateObjects



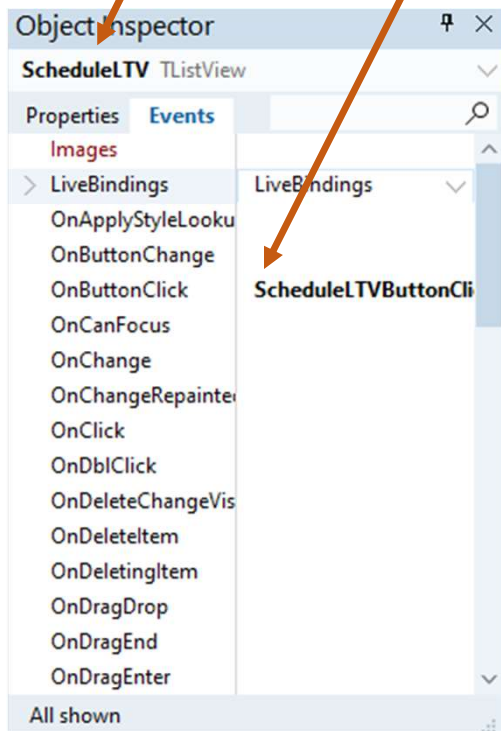
Fazemos a correção dos tamanhos dos botões dos itens do ListView.

```
procedure TClinicPlusForm.ScheduleLTVUpdateObjects(const Sender: TObject;  
  const AItem: TListViewItem);  
begin  
  // Ajusta altura dos botões no item do listview  
  AItem.Objects.DrawableByName('ConfirmaButton').Height := 37;  
  AItem.Objects.DrawableByName('CancelaButton').Height := 37;  
end;
```



ScheduleLTV - OnButtonClick

Crie o evento **OnButtonClick**.



FMX.ListView.TListView.OnButtonClick

Up to Parent: TListView

Delphi

```
property OnButtonClick: TItemControlEvent read FOnButtonClick  
write FOnButtonClick;
```

C++

```
__property OnButtonClick:
```

Properties

Type	Visibility	Source
event	published	FMX.ListView.TListView
		FMX.ListView.hpp

Description

FMX.ListView.TListView.OnButtonClick inherits from FMX.ListView.TAppearanceListView.OnButtonClick. All content below this line refers to FMX.ListView.TAppearanceListView.OnButtonClick.

Occurs immediately when you click on a **text button** or a **glyph button** inside a list view item.

Write an **OnButtonClick** event handler to provide additional functionality when clicking on a **text button** or a **glyph button** inside a list view item.

See Also

- [FMX.ListView.Appearances.TItemControlEvent](#)
- [FMX.ListView.TAppearanceListView.OnButtonClick](#)

Segundo a documentação, o evento é disparado imediatamente após você clicar em um **Text Button** ou **Glyph Button** dentro de um **ListView**.

ScheduleLTV - OnButtonClick

```
· |procedure TClinicPlusForm.ScheduleLTVButtonClick(const Sender: TObject;
· | const AItem: TListItem; const AObject: TListItemSimpleControl);
· |
· | var
· |     JSON: TJSONObject;
· | begin
· |     if AObject.Name.ToLower = 'confirmabutton' then
· |     begin
· |         AgendamentoMTB.Edit; // muda dataset para modo de edição
· |         AgendamentoMTBfg_status.AsString := 'C'; // altera valor do status
· |         AgendamentoMTBdata_confirmacao.Value := Now; // altera para data/hora atual
· |         AgendamentoMTB.Post; // salva dados no dataset local
· |         JSON := AgendamentoMTB.ToJSONObject(); // converte registro para JSON
· |         ChangeSchedule(AgendamentoMTBidagendamento.Value, JSON); // envia mudanças para back-end
· |         JSON.Free; // libera memoria
· |     end;
· |
· |     if AObject.Name.ToLower = 'cancelabutton' then
· |     begin
· |         AgendamentoMTB.Edit; // muda dataset para modo de edição
· |         AgendamentoMTBfg_status.AsString := 'I'; // altera valor do status
· |         AgendamentoMTBdata_confirmacao.Value := Now; // altera para data/hora atual
· |         AgendamentoMTB.Post; // salva dados no dataset local
· |         JSON := AgendamentoMTB.ToJSONObject(); // converte registro para JSON
· |         ChangeSchedule(AgendamentoMTBidagendamento.Value, JSON); // envia mudanças para back-end
· |         JSON.Free; // libera memoria
· |     end;
· |
· |     // Atualiza os registros em thread separadas.
· |     TTask.Run(procedure
· |     begin
· |         Sleep(50);
· |         AgendamentoMTB.EmptyDataSet; // limpa dataset
· |         HistoricoMTB.EmptyDataSet; // limpa dataset
· |         GetAgendamentoAtivo(UserID); // carrega dados atualizados do back-end
· |         GetHistorico(UserID); // carrega dados atualizados do back-end
· |     end);
· |
· | end;
```