

Clinic Plus

App Android/Windows sem Token

Técnicas Avançadas de Programação Mobile e WEB

Projeto: Desenvolvimento em 3 camadas com Restful **com** autenticação

Prof.: Maicon Pires

Etec João B. de Lima Figueiredo (009) / Fatec Mococa (120)

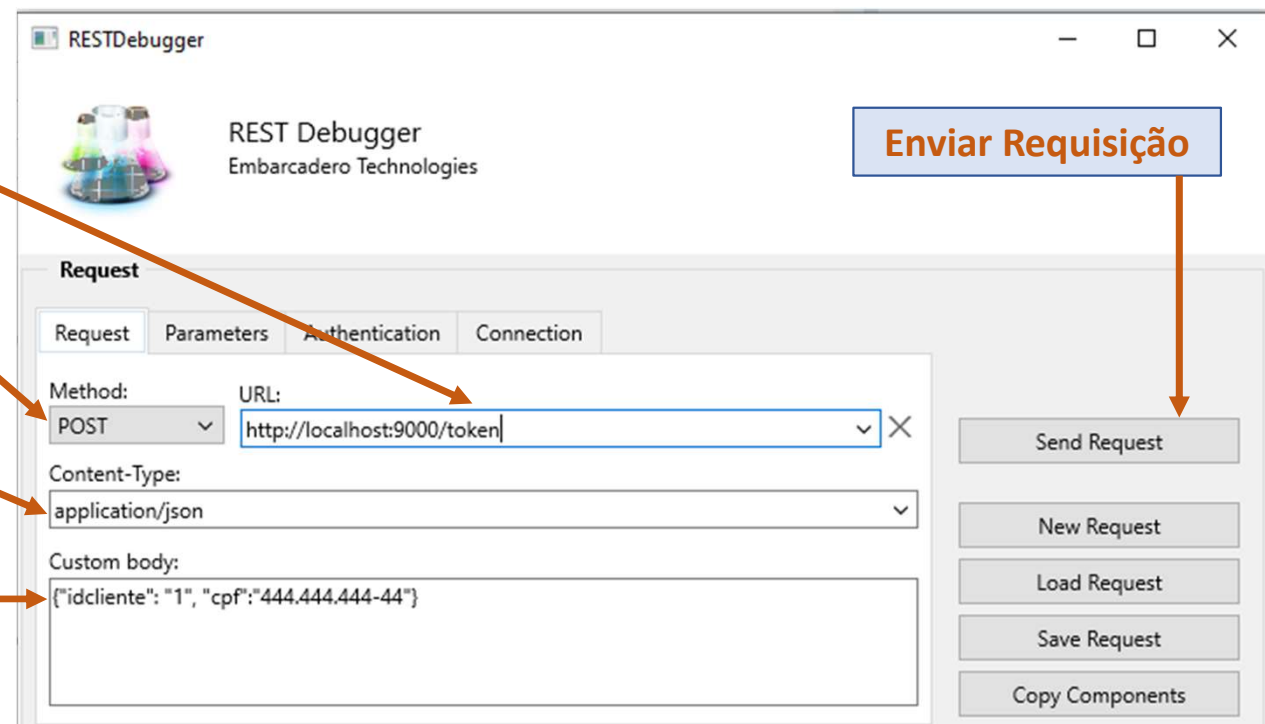
Teste com Rest Debugger

Primeiro precisamos requisitar um novo Token para o servidor Back-end. O token por padrão tem validade de 24 horas, podendo ser reutilizado em milhares de requisições subsequentes.

Para requisitar um novo token, faça a requisição **POST** na rota **token**

Defina o **Content-Type** para **application/json**

No corpo da requisição, informe o objeto json com **ID** e **CPF** do cliente



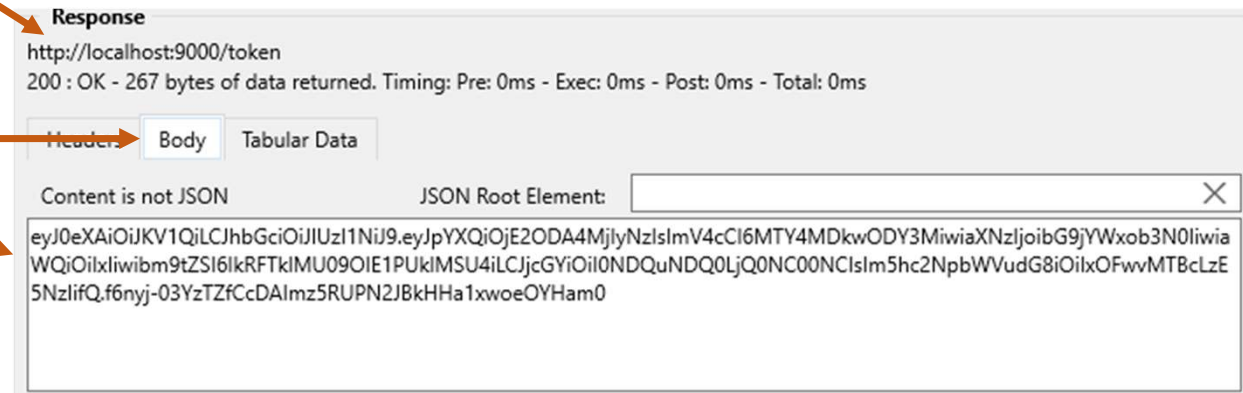
Teste com Rest Debugger

Primeiro precisamos requisitar um novo Token para o servidor Back-end. O token por padrão tem validade de 24 horas, podendo ser reutilizado em milhares de requisições subsequentes.

Dados da requisição

Token de resposta vem no corpo da resposta

Copie o token para usar como autenticação nas demais requisições, veja nos slides a seguir.



Teste com Rest Debugger

Se o ID e CPF do cliente não bater com o que está cadastrado no BD, uma mensagem de erro será retornada.

ID está correto, mas o CPF está errado

Mensagem de erro na criação do token

Request

Request Parameters Authentication Connection

Method: POST URL: http://localhost:9000/token

Content-Type: application/json

Custom body: {"idcliente": "1", "cpf": "444.444.444-00"}

Send Request New Request Load Request Save Request Copy Components

Response

http://localhost:9000/token

500 : Internal Server Error - 121 bytes of data returned. Timing: Pre: 0ms - Exec: 31ms - Post: 0ms - Total: 31ms

Headers Body Tabular Data

Content is valid JSON JSON Root Element:

```
{
  "error": "Falha ao autenticar o usuário: Usuário não encontrado ou informações inválidas."
}
```

Teste com Rest Debugger

Requisição de rota privada **sem fornecimento do token.**

Todas as rotas, com exceção da rota token, passam a ser privadas

Status e mensagem quando token não é fornecido

The screenshot displays the Rest Client interface. The 'Request' tab is active, showing a GET method and the URL 'http://localhost:9000/cliente'. The 'Response' tab is also visible, showing a 401 Unauthorized status with the message 'Token not found'. The 'Body' sub-tab is selected, showing the response content.

Request

Request Parameters Authentication Connection

Method: GET URL: http://localhost:9000/cliente

Content-Type:

Custom body:

Send Request New Request Load Request Save Request Copy Components

Response

http://localhost:9000/cliente

401 : Unauthorized - 15 bytes of data returned. Timing: Pre: 0ms - Exec: 0ms - Post: 0ms - Total: 0ms

Headers Body Tabular Data

Content is not JSON JSON Root Element:

Token not found

Teste com Rest Debugger

Requisição de rota privada com fornecimento do token.

[illegible]

Sucesso na requisição de rota privada com um token válido

Teste do Token no Front-end

Declare a constante Token e concatene um token valido copiado de uma requisição com **Rest Debugger**

```
120 // Concatenado, porque delphi não suporta literal maior que 255 caracteres
    Token = 'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9'+
    '.eyJpYXQiOiE2ODA4MjIyNzIsImV4cCI6MTY4MDkwODY3MiwiYWxzIjoibG9jYWxo'+
    'b3N0IiwiaWQiOiIxIiwibm9tZSI6IHRFTklMU090IE1PUklMSU4iLCJjcGYiOiI0N'+
    'DQuNDQ0LjQ0NC00NCIsIm5hc2NpbWVudG8iOiIxOFwvMTBcLzE5NzIifQ.f6nyj-0'+
    '3YzTZfCcDAImz5RUPN2JBkHHalxwoeOYHam0';
```

Concatene o valor do Token em duas ou mais linhas,
pois o Delphi tem o limite de 255 caracteres por tipos
literais.

Não concatenar o token, irá causar um erro de sintaxe.

Messages

[dcc32 Error] ClinicPlus.Form.pas(121): E2056 String literals may have at most 255 elements
[dcc32 Error] ClinicPlus.Form.pas(140): E2250 There is no overloaded version of 'Run' that can
[dcc32 Error] ClinicPlus.Form.pas(170): E2250 There is no overloaded version of 'Run' that can

Corrigindo as Declarações das Funções

Antes usávamos a variável UserID para identificar o usuário, agora o Token fará isso. Então comente a variável.

```
. var  
. ClinicPlusForm: TClinicPlusForm;  
129 // UserID: Integer; // usado para facilitar os testes.
```

Muitos erros irão acontecer após comentar UserID, precisamos corrigir todas as funções que utilizavam o UserID para usar o Token em seu lugar.

```
- public  
. { Public declarations }  
. procedure GetAgendamentoAtivo(const AToken: String; // requisição  
. procedure GetHistorico(const AToken: String; // requisição de his  
.   
110 procedure GetCliente(const AToken: String; // requisição de nome  
. procedure LoadCliente(const AToken: String; // carrega nome, cpf  
.   
113 procedure ChangeSchedule(const AID: Integer; JSON: TJSONObject);  
. end;  
-   
. const
```

Correção da Implementação de GetAgendamentoAtivo

Adicione o Header de autorização na requisição com o valor do Token.

```
procedure TClínicPlusForm.GetAgendamentoAtivo(const AToken: String);  
begin  
180 TRequest.New.BaseURL(EnderecoServidor+'agendamento') // URL da API  
    .AddHeader('Authorization', AToken, [poDoNotEncode])  
    .AddParam('fg_status','a') // QueryParam - Filtra apenas status .  
    // .AddParam('idcliente',User.ToString) // QueryParam - Filtra ap  
    .Accept('application/json') // tipo de dados da resposta que es  
    .DataSetAdapter(AgendamentoMTB) // Conversão de JSON para DATAS  
    .Get; // Verbo da requisição  
end;
```

Comente o query param que usávamos para filtrar registros do usuário. Isso agora é feito pelo Back-end baseado no Token.

Correção da Implementação de **GetHistorico**

Adicione o Header de autorização na requisição com o valor do Token.

```
procedure TclinicPlusForm.GetHistorico(const AToken: String);  
begin  
    TRequest.New.BaseURL(EnderecoServidor+'agendamento') // URL  
    .AddHeader('Authorization', AToken, [poDoNotEncode])  
    // .AddParam('idcliente', User.ToString) // QueryParam - apenas  
    .Accept('application/json') // tipo de dados da resposta que  
    .DataSetAdapter(HistoricoMTB) // Conversão de JSON para DAT.  
    .Get; // Verbo da requisição  
end;
```

Comente o query param que usávamos para filtrar registros do usuário. Isso agora é feito pelo Back-end baseado no Token.

Correção da Implementação de GetCliente

Adicione o Header de autorização na requisição com o valor do Token.

```
procedure TClinicPlusForm.GetCliente(const AToken: String);  
190 begin  
191     FRequest.New.BaseURL(EnderecoServidor+'cliente') // URL da  
        .AddHeader('Authorization', AToken, [poDoNotEncode])  
        // .ResourceSuffix(ID.ToString) // diciona /1 na url  
        .Accept('application/json') // tipo de dados da resposta  
        .DataSetAdapter(ClienteMTB) // Conversão de JSON para DA  
        .Get; // Verbo da requisição  
end;
```

Comente o query param que usávamos para filtrar registros do usuário. Isso agora é feito pelo Back-end baseado no Token.

Correção da Implementação de LoadCliente

Comente a chamada da função GetCliente que era feita com parâmetro ID (numero inteiro)

Adicione uma nova chamada de GetCliente com parâmetro AToken (String)

```
procedure TClinicPlusForm.LoadCliente(const AToken: String);
var
  FotoStream: TMemoryStream;
  BrushBmp: TBrushBitmap;
begin
  // GetCliente(ID); // requisição no Backend (API)
  268 GetCliente(AToken); // requisição no Backend (API)
  // usar synchronize apenas com a certeza de que LoadCliente
  // uma thread diferente da thread principal.
  TThread.Synchronize(TThread.CurrentThread, procedure
  begin
    NameLBL.Text := ClienteMTBnome.AsString; // grava nome no
    CPFLBL.Text := ClienteMTBCPF.AsString; // grava CPF no

    FotoStream := TMemoryStream.Create; // Cria stream para l
    BrushBmp := TBrushBitmap.Create; // Cria Brush para desen

    280 try
      ClienteMTBFoto.SaveToStream(FotoStream); // Lê a foto d
      BrushBmp.Bitmap.LoadFromStream(FotoStream); // Desenha
      BrushBmp.WrapMode := TWrapMode.TileStretch; // Ajusta i
      Circle1.Fill.Bitmap.Assign(BrushBmp); // Desenha imagem
    finally
      // libera variáveis temporárias utilizadas no processo de
      FotoStream.Free;
      BrushBmp.Free;
    end;
  290 end);
end;
```

Correção do evento ClinicPlusForm OnCreate

```
· procedure TClinicPlusForm.FormCreate(Sender: TObject);  
· begin  
·   // configura DatasetSerialize para manter todos os nomes dos campos em minusculas  
·   // data_confirmacao ao invés de dataConfirmacao  
160   TDataSetSerializeConfig.GetInstance.CaseNameDefinition := TCaseNameDefinition.cndLower;  
·  
·   // Ajusta posicionamento do retangulo animado  
·   AnimeRCT.Position.X := HomeLYT.Position.X;  
·  
·   // grava temporariamente nosso usuário de teste,  
·   // até criarmos a parte que identifica o usuario no APF  
·   // UserID := 1;  
·  
·   TTask.Run (procedure  
170   begin  
·     LoadCliente(Token);  
·     GetAgendamentoAtivo(Token);  
·     GetHistorico(Token);  
·   end);  
· end;
```

Substitua UserID pelo Token nos
parâmetros das funções **LoadCliente**,
GetAgendamentoAtivo e **GetHistorico**

Correção do evento ScheduleLTV OnButtonClick

```
procedure TclinicPlusForm.ScheduleLTVButtonClick(const Sender: TObject;
3  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/hor
        AgendamentoMTB.Post; // salva dados no dataset local
        JSON := AgendamentoMTB.ToJSONObject(); // converte registro para JSO
        ChangeSchedule(AgendamentoMTBidagendamento.Value, JSON); // envia mu
        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/hor
        AgendamentoMTB.Post; // salva dados no dataset local
        JSON := AgendamentoMTB.ToJSONObject(); // converte registro para JSO
        ChangeSchedule(AgendamentoMTBidagendamento.Value, JSON); // envia mu
        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(Token); // carrega dados atualizados do back-end
        GetHistorico(Token); // carrega dados atualizados do back-end
    end);
end;
```

Substitua UserID pelo Token nos
parâmetros das funções
GetAgendamentoAtivo e **GetHistorico**

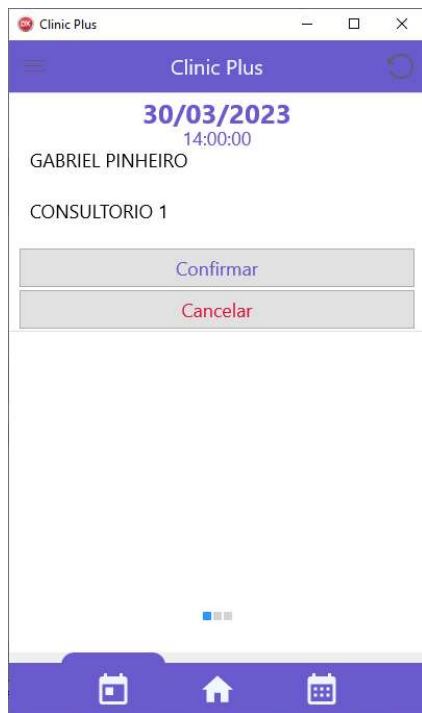
Correção do evento AtualizarBTN OnClick

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

Substitua UserID pelo Token nos
parâmetros das funções
GetAgendamentoAtivo e **GetHistorico**

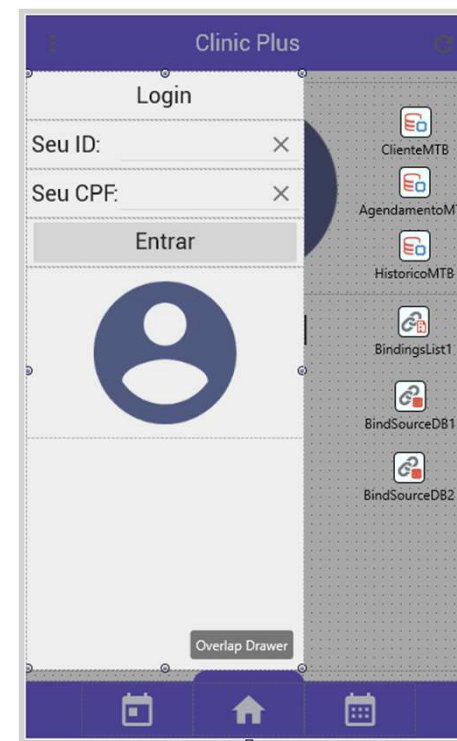
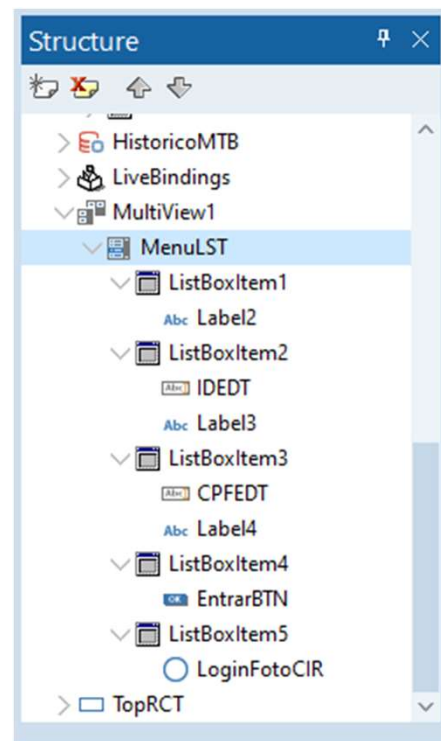
Teste a aplicação

Teste a aplicação, deve continuar funcionando exatamente igual antes, mas dessa vez com a autenticação por token identificando o usuário.



Multiview e Aquisição do Token pelo Front-end

Agora que testamos o front-end usando o token que copiamos do Rest Debbuger, vamos implementar a aquisição do Token pelo nosso front-end, similar a um sistema de login.



Multiview e Aquisição do Token pelo Front-end



- ListBoxItem1 => TListBoxItem
- Size.Width = 250
- Size.Height = 44
- Label2 => TLabel
 - Align = Center
 - AutoSize = True
 - Size.Width = 46
 - Size.Height = 22
 - TextSettings.Font.Size = 20
 - TextSettings.WordWrap = False
 - Text = Login

Multiview e Aquisição do Token pelo Front-end



- ListBoxItem2 => TListBoxItem
- Size.Width = 250
- Size.Height = 44
- Label3 => TLabel
 - Align = Left
 - Margins.Left = 5
 - Size.Width = 80
 - Size.Height = 44
 - TextSettings.Font.Size = 20
 - Text = Seu ID:
- IDEDT => TEdit
 - Align = Client
 - StyleLookup = clearingeditstyle
 - TextSettings.Font.Size = 18
 - Margins.Top = 5
 - Margins.Right = 5

Multiview e Aquisição do Token pelo Front-end



- ListBoxItem3 => TListBoxItem
- Size.Width = 250
- Size.Height = 44
- Label4 => TLabel
 - Align = Left
 - Margins.Left = 5
 - Size.Width = 80
 - Size.Height = 44
 - TextSettings.Font.Size = 20
 - Text = Seu CPF:
- CPFEDT => TEdit
 - Align = Client
 - StyleLookup = clearingeditstyle
 - TextSettings.Font.Size = 18
 - Margins.Top = 5
 - Margins.Right = 5

Multiview e Aquisição do Token pelo Front-end



- ListBoxItem4 => TListBoxItem
- Size.Width = 250
- Size.Height = 44
- EntrarBTN: TButton
 - Align = Client
 - Margins.Left = 5
 - Margins.Top = 2
 - Margins.Right = 5
 - Margins.Bottom = 2
 - Text = Entrar
 - TextSettings.Font.Size = 20

Multiview e Aquisição do Token pelo Front-end



- ListBoxItem5 => TListBoxItem
- Size.Width = 250
- Size.Height = 161
- LoginFotoCIR: TCircle
 - Align = HorzCenter
 - Fill.Bitmap.WrapMode = TileStretch
 - Fill.Kind = Bitmap
 - Size.Width = 286
 - Size.Height = 161
 - Stroke.Kind = None

Função LOGIN

```
public
{ Public declarations }
procedure GetAgendamentoAtivo(const AToken: String); // requisiçã
procedure GetHistorico(const AToken: String); // requisição de hi

procedure GetCliente(const AToken: String); // requisição de nom
procedure LoadCliente(const AToken: String); // carrega nome, cpf

procedure ChangeSchedule(const AID: Integer; JSON: TJSONObject);
procedure Login(const ID, CPF: String);
end;
```

Declare a função **Login** na seção publica da classe.

Função LOGIN

Ctrl+Shift+C
para fazer a
implementação
da função

```
procedure TclinicPlusForm.Login(const ID, CPF: String);
var
  JSON: TJSONObject;
  Resposta: IResponse;
begin
  JSON := TJSONObject.Create;
  try
    // cria json com informações para requisição
    JSON.AddPair('idcliente', ID);
    JSON.AddPair('cpf', cpf);
    try
      Resposta := TRequest.New
        .BaseURL(EnderecoServidor+'token') // URL da API
        .AddBody(JSON, False) // JSON com registro atualizado
        .Accept('application/json') // tipo de dados da resposta que esperamos
        .Post; // Verbo da requisição

      if Resposta.StatusCode = 200 then begin // token recebido com sucesso
        // ShowMessage('Sucesso: '+Resposta.Content);
        Token := 'Bearer '+Resposta.Content; // Concatena 'Bearer ' com token recebido.
      end
      else begin // falha ao pegar o token
        ShowMessage('Falha na autenticação: '+Resposta.Content); // falha de id ou cpf inválido.
      end;
    except
      On E:Exception do begin
        ShowMessage('Falha no login: '+E.Message); // falhas de rede, etc.
      end;
    end;
  finally
    JSON.Free; // libera memoria da variável
  end;
end;
```

Altere o evento ClinicPlusForm OnCreate

Precisamos mudar a declaração de Token, que antes era uma constante para uma variável.

Comentamos as requisições iniciais no Back-end. Elas devem ser realizadas após o login do usuário, quando tivermos o token em mão.

Direcionamos o usuário para realizar o login

```
procedure TClinicPlusForm.FormCreate(Sender: TObject);
begin
    // configura DatasetSerialize para manter todos os nomes c
    // data_confirmacao ao invés de dataConfirmacao
    TDataSetSerializeConfig.GetInstance.CaseNameDefinition :
    .
    // Ajusta posicionamento do retângulo animado
    AnimeRCT.Position.X := HomeLYT.Position.X;
    .
    // grava temporariamente nosso usuário de teste,
    // até criarmos a parte que identifica o usuário no APP
    // UserID := 1;
    .
    // TTask.Run(procedure
    // begin
    //     LoadCliente(Token);
    //     GetAgendamentoAtivo(Token);
    //     GetHistorico(Token);
    // end);
    .
    MultiView1.ShowMaster; // Abre multiview para login
    IDEDT.SetFocus; // joga foco do teclado em IDEDT
    .
end;
```

Implemente o evento EntrarBTN OnClick

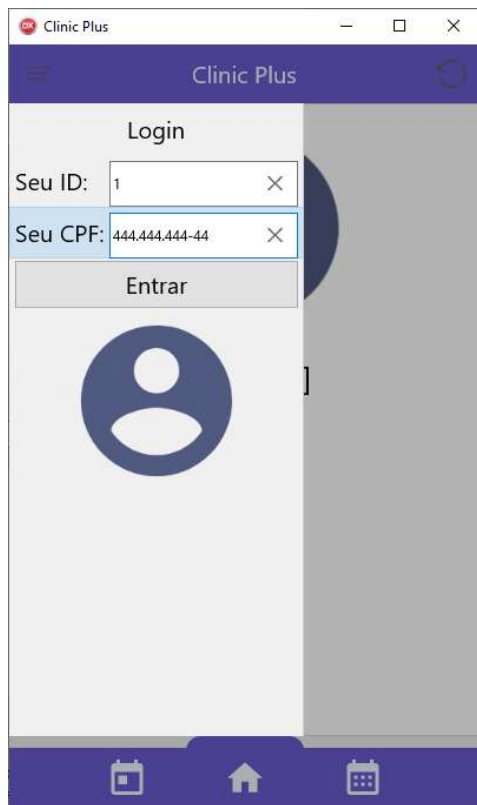
```
procedure TClinicPlusForm.EntrarBTNClick(Sender: TObject);
begin
    // validação das informações
    if (IDEDT.Text = '') or (CPFEDT.Text = '') then
        raise Exception.Create('Informe o ID e o CPF'); // falha se vazio

    // executamos as requisições em thread separada
    TTask.Run(procedure
    begin
        Login(IDEDT.Text, CPFEDT.Text); // faz login primeiro

        LoadCliente(Token); // carrega info do usuário
        GetAgendamentoAtivo(Token); // carrega agendamentos ativos
        GetHistorico(Token); // carrega historico de agendamentos

        TThread.Synchronize(TThread.Current, procedure begin
            // copia foto do usuario
            LoginFotoCIR.Fill.Bitmap.Assign( Circle1.Fill.Bitmap );
            // Esconde o painel
            MultiView1.HideMaster;
        end);
    end);
end;
```

Teste Front-end com Token – **Usuario 1**



Clinic Plus

Login

Seu ID: 1

Seu CPF: 444.444.444-44

Entrar

Profile icon placeholder

Bottom navigation bar: Home, Calendar, Search



Clinic Plus

30/03/2023
14:00:00

GABRIEL PINHEIRO

CONSULTORIO 1

Confirmar

Cancelar

Bottom navigation bar: Home, Calendar, Search



Clinic Plus

Agendamento: 30/03/2023
14:00:00
Confirmação:
Atendimento:

Profissional: GABRIEL PINHEIRO

Local: CONSULTORIO 1

Status: A

Agendamento: 21/03/2023
15:30:00
Confirmação: 18/03/2023 19:02:00
Atendimento:

Profissional: LUIS RAMALHO

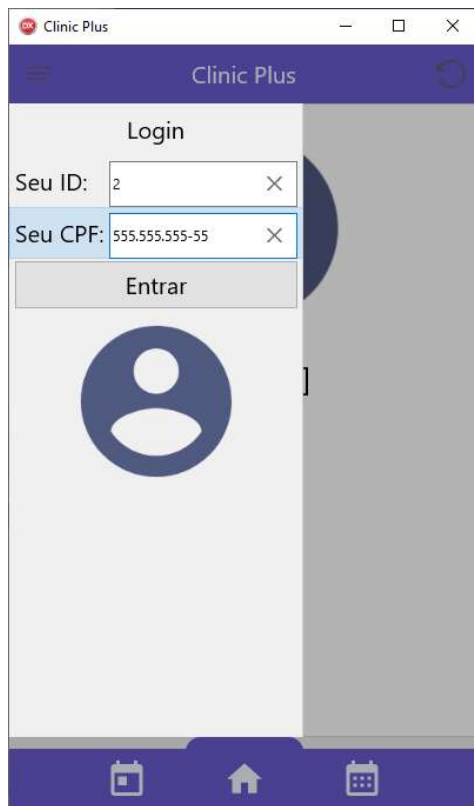
Local: CONSULTORIO 1

Status: C

Agendamento: 19/03/2023

Bottom navigation bar: Home, Calendar, Search

Teste Front-end com Token – **Usuario 2**



Clinic Plus

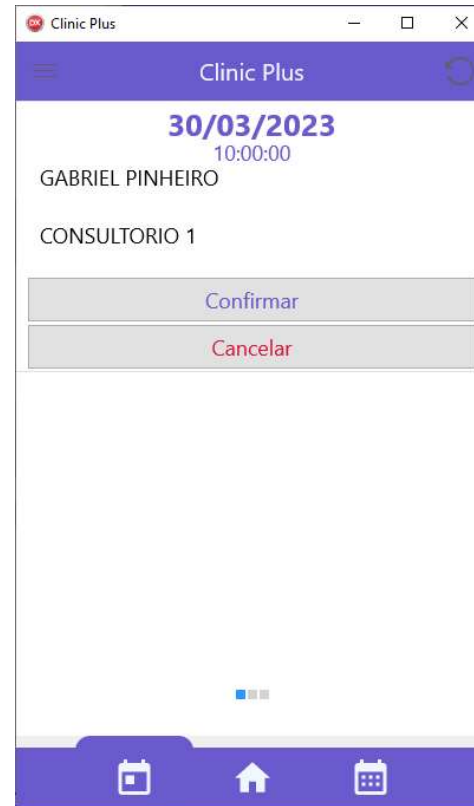
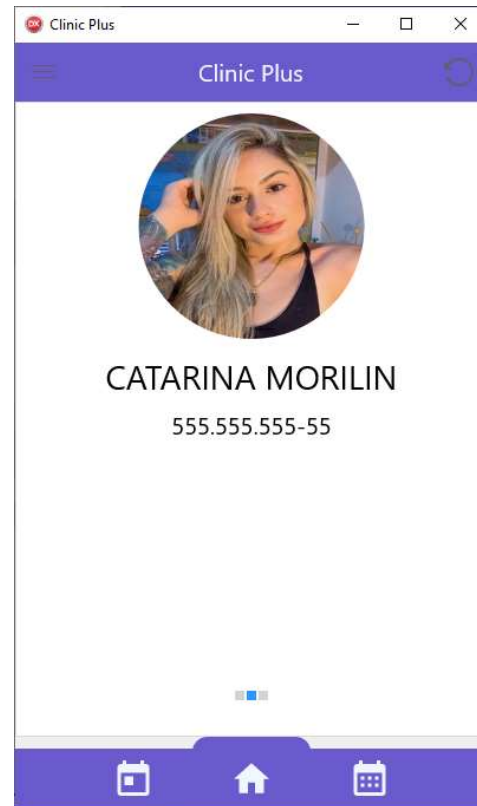
Login

Seu ID: 2

Seu CPF: 555.555.555-55

Entrar

Bottom navigation: calendar, home, calendar



Clinic Plus

30/03/2023
10:00:00

GABRIEL PINHEIRO

CONSULTORIO 1

Confirmar

Cancelar

Bottom navigation: calendar, home, calendar



Clinic Plus

Agendamento: 30/03/2023
10:00:00

Confirmação:

Atendimento:

Profissional: GABRIEL PINHEIRO

Local: CONSULTORIO 1

Status: A

Bottom navigation: calendar, home, calendar