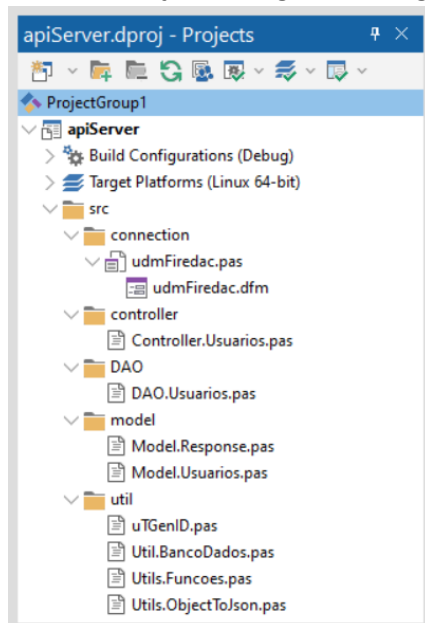
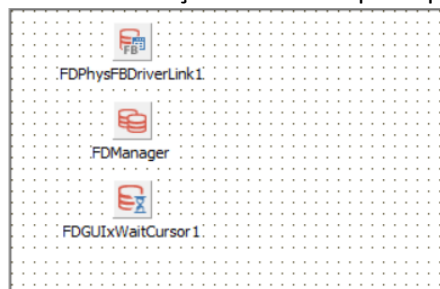


## MicroUsuarios

1. O microserviço está organizado seguindo as boas práticas de desenvolvimento de software.



2. Na pasta “connection”, temos o datamodule “udmFiredac” com os componentes de conexão e na unit está a codificação necessária para que microserviço se conecte a um banco de dados Firebird.



```
procedure TdmFiredac.DataModuleCreate(Sender: TObject);
begin
    FDManager.Active := false;
    FDPhysFBDriverLink1.vendorLib := '';
    {$IFDEF MSWINDOWS}
    FDPhysFBDriverLink1.vendorLib := ExtractFilePath(ParamStr(0)) +
        'fbClient.dll';
    {$ENDIF}
end;

procedure TdmFiredac.FDManagerBeforeStartup(Sender: TObject);
var
    cnxDef: IFDStanConnectionDef;
begin
    FDManager.connectionDefs.Clear;
    cnxDef := FDManager.connectionDefs.addConnectionDef;
    cnxDef.Name := FIREDAC_CONNECTION_DEF_NAME;
    cnxDef.Params.DriverID := 'FB';
    cnxDef.Params.UserName := 'sysdba';
    cnxDef.Params.Password := 'masterkey';
    cnxDef.Params.Database :=
        'C:\Desenvolvimento\Projetos\Avance\Micro4Delphi\MicroUsuarios\API\banco\Usuarios.fdb';
    cnxDef.Params.pooled := false;
    cnxDef.Params.Add('Protocol=TCPIP');
    cnxDef.Params.Add('Server=192.168.0.183');
    cnxDef.Params.Add('Port=3050');
    cnxDef.Params.Add('CharacterSet=WIN1252');
    cnxDef.Apply;
end;

function TdmFiredac.getConnectionDefName: string;
begin
    result := FIREDAC_CONNECTION_DEF_NAME;
end;
```

3. Na pasta “controller”, temos a unit “Controller.Usuarios” responsável por controlar o acesso a camada “DAO” que faz as solicitações e alterações no banco de dados. Nessa unit também estão os comandos do Swagger para a criação automática da documentação da API da Usuários.

```
type
[SwagPath('Usuarios', 'Usuarios')]

TControllerUsuarios = class
private
    FRequest: THorseRequest;
    FResponse: THorseResponse;
    function getBody: TModelUsuarios;
public
    //Comandos do Swagger para documentação
    [SwagParamBody('body', TModelUsuarios)]

    //Métodos Post
    [SwagPOST('', 'Post', true)]
    [SwagResponse(200, TModelUsuarios, 'Success')]
    [SwagResponse(400, TModelResponse, 'Bad Request')]

    //Procedure post que chama o método setUsuarios da camada DAOUsuarios
    procedure post;

    //Métodos Get
    [SwagGET('', 'Get', true)]
    [SwagResponse(200, TModelUsuarios, 'Success')]
    [SwagResponse(400, TModelResponse, 'Bad Request')]

    //Procedure get que chama o método getUsuarios da camada DAOUsuarios
    procedure get;

    constructor Create(Req: THorseRequest; Res: THorseResponse);
end;

procedure TControllerUsuarios.post;
var
    Usuarios: TModelUsuarios;
    LRetorno: TModelResponse;
    DAOUsuarios: TDAOUsuarios;
begin
    Usuarios := getBody;
    DAOUsuarios := TDAOUsuarios.Create;

    try
        try
            // Realiza a chamada do método setUsuarios da camada DAOUsuarios
            FResponse.Status(200).Send<TJSONObject>
                (DAOUsuarios.setUsuarios(Usuarios));
        except
            on E: Exception do
                begin
                    LRetorno := TModelResponse.Create;
                    LRetorno.Status := 400;
                    LRetorno.mensagem := E.Message;

                    // Retorna o Json como objeto para a camada cliente
                    FResponse.Status(400).Send<TJSONObject>
                        (TJSON.ObjectToJsonObject(LRetorno, [joIgnoreEmptyArrays,
                            joIgnoreEmptyStrings]));
                end;
            end;
        finally
            if Usuarios <> nil then
                begin
                    Usuarios.free;
                    Usuarios := nil;
                end;

                if DAOUsuarios <> nil then
                    begin
                        DAOUsuarios.free;
                        DAOUsuarios := nil;
                    end;
                end;
            end;
        end;
end;
```

```

procedure TControllerUsuarios.get;
var
  LRetorno: TModelResponse;
  DAOUsuarios: TDAOUsuarios;
begin
  DAOUsuarios := TDAOUsuarios.Create;

  try
    try
      //Realiza a chamada do método getUsers da camada DAOUsuarios
      FResponse.Status(200).Send<TJSONArray>(DAOUsuarios.getUsuarios);
    except
      on E: Exception do
        begin
          LRetorno := TModelResponse.Create;
          LRetorno.Status := 400;
          LRetorno.mensagem := E.Message;

          //Retorna o Json como objeto para a camada cliente
          FResponse.Status(400).Send<TJSONObject>
            (TJSON.ObjectToJsonObject(LRetorno, [joIgnoreEmptyArrays,
              joIgnoreEmptyStrings]));
        end;
      end;
    finally
      if DAOUsuarios <> nil then
        begin
          DAOUsuarios.free;
          DAOUsuarios := nil;
        end;
      end;
    end;
  end;
end;

```

4. Na pasta “DAO”, temos a unit “DAO.Usuarios” responsável por realizar os comandos SQL para consulta e inclusão de usuários no banco de dados. Para consulta de usuários é utilizado o método “getUsuarios” e para inclusão é utilizado o método “postUsuarios”.

```

function TDAOUsuarios.getUsuarios: TJSONArray;
var
  Usuarios: TModelUsuarios;
  UsuariosList: TArray<TObject>;
begin
  result := nil;
  UsuariosList := TArray<TObject>.Create(nil);
  FDQuery := TUtilBancoDados.getFDQuery;

  try
    // Comando SQL para consulta
    FDQuery.SQL.Clear;
    FDQuery.SQL.Add('SELECT CODIGO, NOME');
    FDQuery.SQL.Add('FROM USUARIOS');
    FDQuery.SQL.Add('WHERE CODIGO > 0');
    FDQuery.open;

    SetLength(UsuariosList, FDQuery.RecordCount);

    while not FDQuery.Eof do
      begin
        //Grava o retorno nas propriedades do Model.Usuarios
        Usuarios := TModelUsuarios.Create;
        Usuarios.codigo := FDQuery.FieldName('CODIGO').asInteger;
        Usuarios.nome := FDQuery.FieldName('NOME').AsString;

        //Adiciona os dados na lista de usuários
        UsuariosList[FDQuery.recno - 1] := Usuarios;
        FDQuery.next;
      end;

      //Retorna a lista em formato Json para o controlador
      if Length(UsuariosList) > 0 then
        result := getJSONArray(UsuariosList);
    finally
      closeQuery;

      if UsuariosList <> nil then
        UsuariosList := nil;
      end;
    end;
  end;
end;

```

```

function TDAOUsuarios.postUsuarios(const Usuarios: TModelUsuarios)
: TModelResponse;
begin
    result := TModelResponse.Create;
    result.status := 0;
    result.mensagem := '';

    FDQuery := TUtilBancoDados.getFDQuery;

    try
        // Comando SQL para consulta
        FDQuery.SQL.Clear;
        FDQuery.SQL.Add('SELECT CODIGO, NOME');
        FDQuery.SQL.Add('FROM USUARIOS');
        FDQuery.SQL.Add('WHERE CODIGO > 0');
        FDQuery.open;

        try
            // Comando para inclusão
            FDQuery.Append;
            if Usuarios.codigo = 0 then
                FDQuery.FieldByName('CODIGO').asInteger :=
                    TGenID.getGenId('GEN_USUARIOS_ID')
            else
                FDQuery.FieldByName('CODIGO').asInteger := Usuarios.codigo;
                FDQuery.FieldByName('NOME').AsString := Usuarios.nome;
            FDQuery.Post;
        finally
            closeQuery;
        end;
    finally
        result.status := 200;
        result.mensagem := 'Dado inserido com sucesso';
    end;
end;

```

5. Na pasta “model”, temos a unit “Model.Usuarios” responsável por instanciar as propriedades da tabela usuários e a unit “Model.Response” responsável por instanciar as propriedades de retorno das solicitações da camada Cliente.

```

unit Model.Usuarios;

interface

type
    TModelUsuarios = class
    private
        Fcodigo: integer;
        Fnome: string;
    published
        property codigo: integer read Fcodigo write Fcodigo;
        property nome: string read Fnome write Fnome;
    end;

implementation

end.

unit Model.Response;

interface

type
    TModelResponse = class
    private
        Fstatus: integer;
        Fmensagem: string;
    published
        property status: integer read Fstatus write Fstatus;
        property mensagem: string read Fmensagem write Fmensagem;
    end;

implementation

end.

```

6. Na pasta “util”, temos as units auxiliares do projeto:
  - a. Unit “uTGenID”, responsável por buscar o generator da tabela USUARIOS no banco dados;
  - b. Unit “Util.BancoDados”, responsável por criar o datamodule “udmFiredac” e os objetos de conexão com o banco de dados, sendo eles o “FFDConnection” e o “FDQuery”;
  - c. Unit “Utils.Funcoes”, responsável por alocar todas as funções auxiliares do projeto;
  - d. Unit “Utils.ObjectToJson”, responsável por converter objetos em JSON.