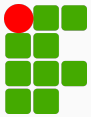
Lazarus

07 - Aplicações Avançadas com Banco de Dados no Lazarus

Marcos Roberto Ribeiro



Instituto Federal Minas Gerais - Campus Bambuí

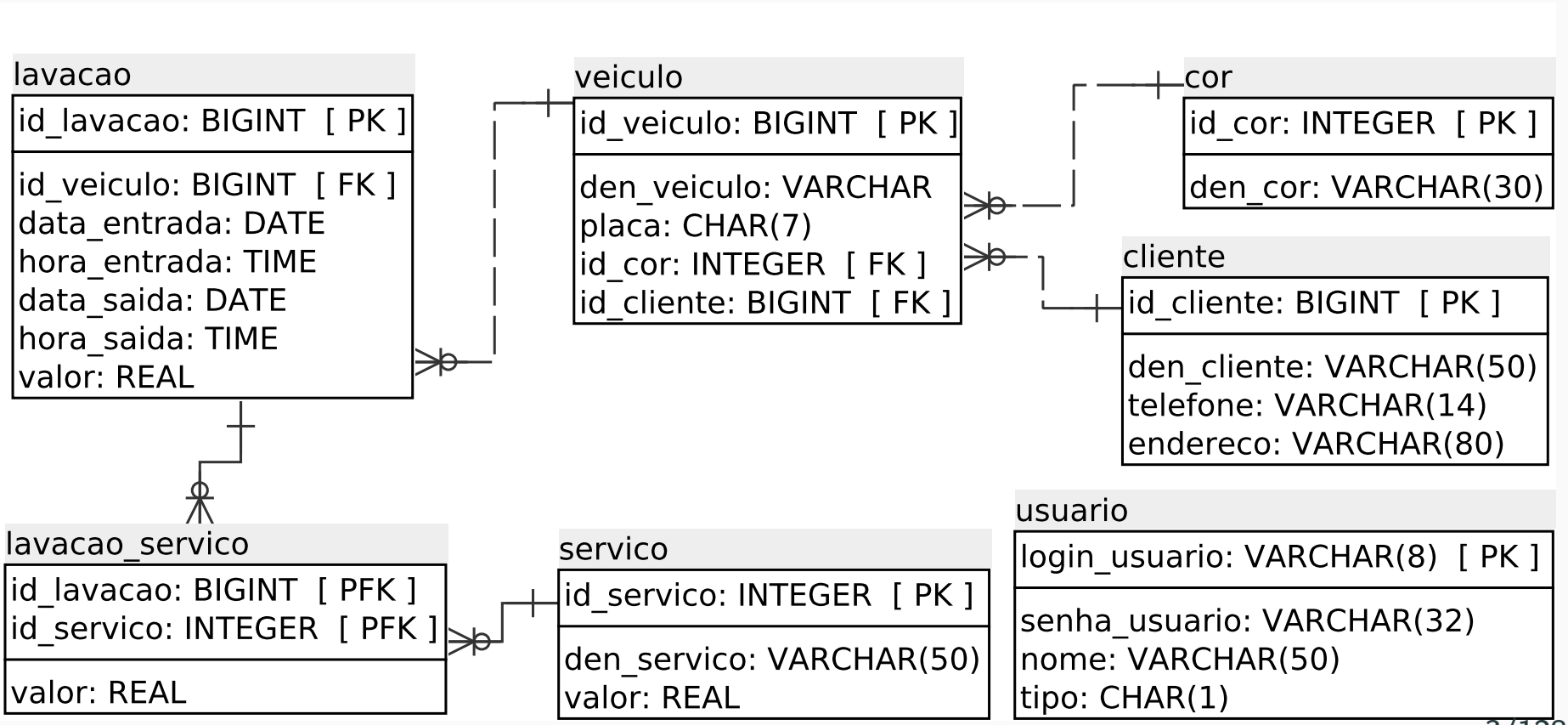
2018

Introdução

- Apesar do Lazarus possuir ferramentas para agilizar o desenvolvimento de aplicações com acesso a banco de dados, é interessante adotar algumas técnicas para a obtenção de aplicações com bom desempenho;
- Nesta aula vamos aprender algumas destas técnicas para controlar a lógica de execução do sistema pelos módulos de dados, evitar manter dados em memória e trabalhar com mestre-detalhes mais avançados.

Banco de Dados

- A demonstração das técnicas avançadas será feita com o desenvolvimento de um sistema como exemplo;
- O sistema fará o controle de um lava-rápido de veículos e o banco de dados será o seguinte:



Modularização

- Em primeiro lugar, vamos dividir o sistema usando vários módulos de dados, são eles:

  ConexaoData Módulo de dados para conexão com banco de dados;

  CadastroData Módulo de dados para controlar o cadastro de cores, clientes, veículos e serviços;

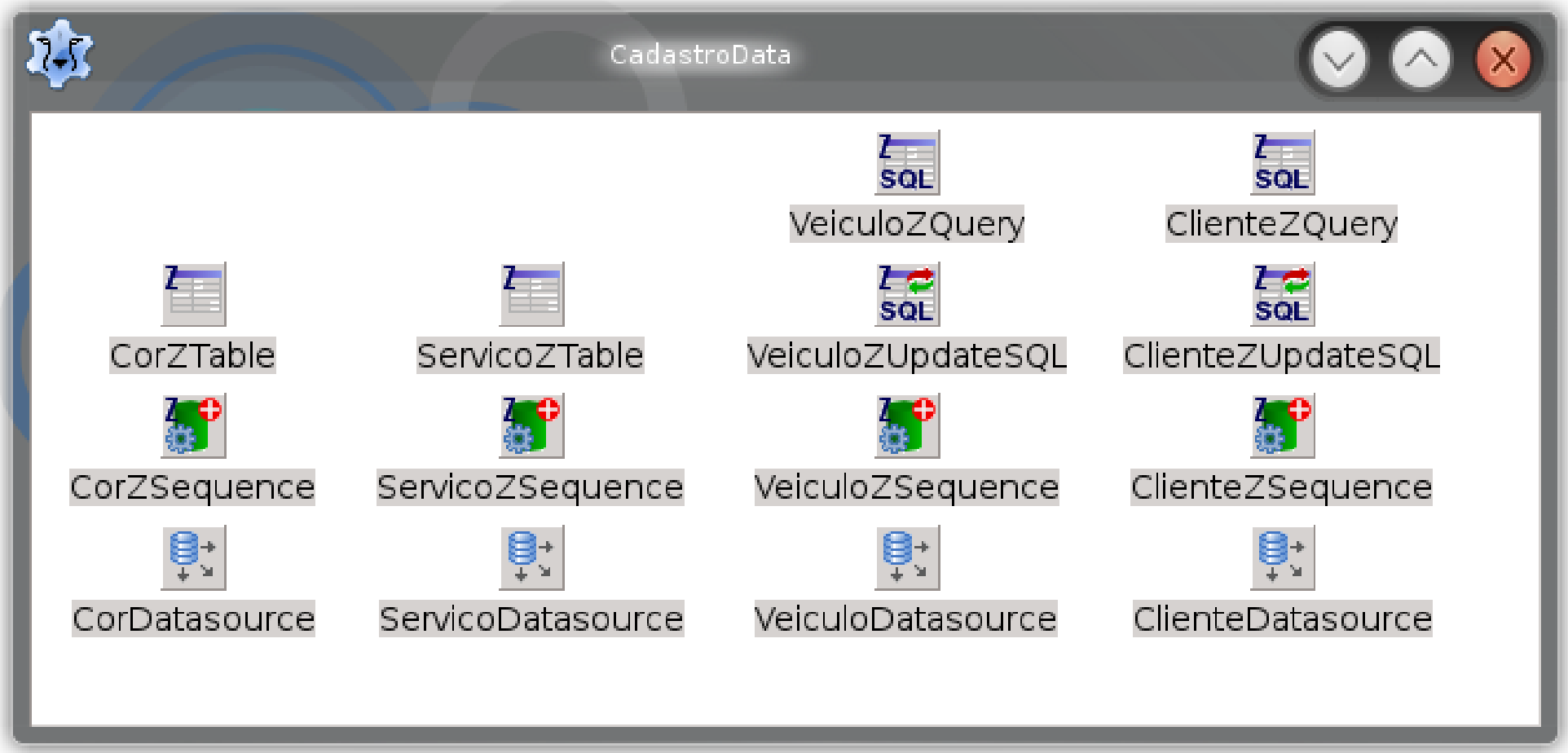
  LavacaoData Módulo de dados para controlar a lavação de carros;

  UsuarioData Módulo de dados para controle de usuários.

ConexaoData (unit ConexaoDM)

```
object ConexaoData: TConexaoData
  object PrincipalZConnection: TZConnection
    HostName = 'localhost'
    Database = 'lavarapido'
    User = 'postgres'
    Password = 'postgres'
    Protocol = 'postgresql'
    Connected = True
  end
end
```

CadastroData (unit CadastroDM) I



CadastroData (unit CadastroDM) II

```
object CadastroData: TCadastroData
  object CorZTable: TZTable
    Connection = ConexaoData.PrincipalZConnection
    TableName = 'cor'
    Sequence = CorZSequence
    SequenceField = 'id_cor'
  end
  object CorZSequence: TZSequence
    Connection = ConexaoData.PrincipalZConnection
    SequenceName = 'cor_id_cor_seq'
  end
  object CorDatasource: TDatasource
    AutoEdit = False
    DataSet = CorZTable
  end
  object ServicoZTable: TZTable
    Connection = ConexaoData.PrincipalZConnection
```

CadastroData (unit CadastroDM) III

```
  TableName = 'servico'
  Sequence = ServicoZSequence
  SequenceField = 'id_servico'
end
object ServicoZSequence: TZSequence
  Connection = ConexaoData.PrincipalZConnection
  SequenceName = 'servico_id_servico_seq'
end
object ServicoDatasource: TDatasource
  AutoEdit = False
  DataSet = ServicoZTable
end
object VeiculoZQuery: TZQuery
  Connection = ConexaoData.PrincipalZConnection
  UpdateObject = VeiculoZUpdateSQL
  SQL.Strings = (
    'SELECT v.*, c.den_cliente'
```

CadastroData (unit CadastroDM) IV

```
    'FROM veiculo AS v, Cliente AS c'
    'WHERE v.id_cliente = c.id_cliente'
    'LIMIT 0;'
  )
  Sequence = VeiculoZSequence
  SequenceField = 'id_veiculo'
end
object VeiculoZUpdateSQL: TZUpdateSQL
  DeleteSQL.Strings = (
    'DELETE FROM veiculo'
    'WHERE veiculo.id_veiculo = :OLD_id_veiculo'
  )
  InsertSQL.Strings = (
    'INSERT INTO veiculo
    '  (id_veiculo, den_veiculo, placa, id_cor, id_cliente)'
    'VALUES'
    '  (:id_veiculo, :den_veiculo, :placa, :id_cor, :id_cliente)'
```

CadastroData (unit CadastroDM) V

```
  )
  ModifySQL.Strings = (
    'UPDATE veiculo SET'
    '  id_veiculo = :id_veiculo,'
    '  den_veiculo = :den_veiculo,'
    '  placa = :placa,'
    '  id_cor = :id_cor,'
    '  id_cliente = :id_cliente'
    'WHERE veiculo.id_veiculo = :OLD_id_veiculo'
  )
end
object VeiculoZSequence: TZSequence
  Connection = ConexaoData.PrincipalZConnection
  SequenceName = 'veiculo_id_veiculo_seq'
end
object VeiculoDatasource: TDatasource
  AutoEdit = False
```

CadastroData (unit CadastroDM) VI

```
  DataSet = VeiculoZQuery
end
object ClienteZQuery: TZQuery
  Connection = ConexaoData.PrincipalZConnection
  UpdateObject = ClienteZUpdateSQL
  SQL.Strings = (
    'SELECT * FROM cliente LIMIT 0'
  )
  Sequence = ClienteZSequence
  SequenceField = 'id_cliente'
end
object ClienteZUpdateSQL: TZUpdateSQL
  DeleteSQL.Strings = (
    'DELETE FROM cliente'
    'WHERE cliente.id_cliente = :OLD_id_cliente'
  )
  InsertSQL.Strings = (
```

CadastroData (unit CadastroDM) VII

```
    'INSERT INTO cliente'
    '  (id_cliente, den_cliente, telefone, endereco)'
    'VALUES'
    '  (:id_cliente, :den_cliente, :telefone, :endereco)'
  )
  ModifySQL.Strings = (
    'UPDATE cliente SET'
    '  id_cliente = :id_cliente,'
    '  den_cliente = :den_cliente,'
    '  telefone = :telefone,'
    '  endereco = :endereco'
    'WHERE cliente.id_cliente = :OLD_id_cliente'
  )
end
object ClienteZSequence: TZSequence
  Connection = ConexaoData.PrincipalZConnection
  SequenceName = 'cliente_id_cliente_seq'
```

CadastroData (unit CadastroDM) VIII

```
  end
  object ClienteDatasource: TDatasource
    AutoEdit = False
    DataSet = ClienteZQuery
  end
end
```

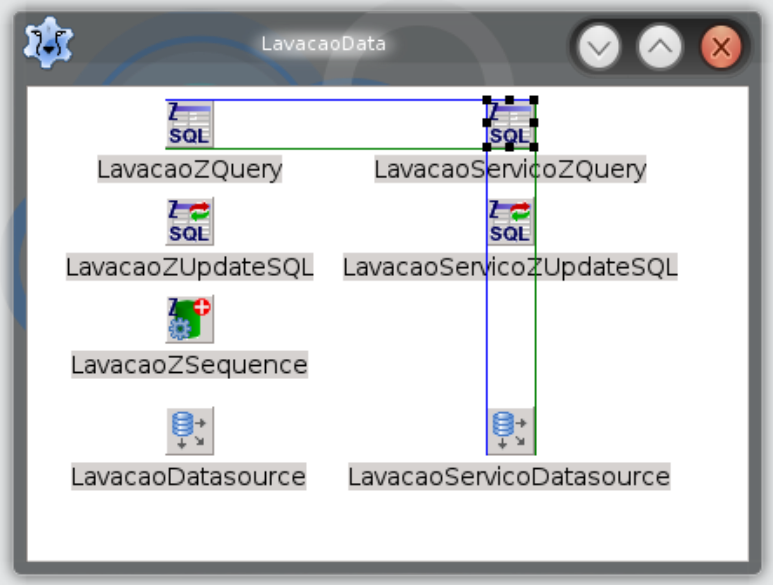
ZTables e ZQueries

- Os objetos do banco de dados (sequências e tabelas) podem ou não ser precedidos de *public*;
- Por que usamos o componente *ZTable* para as tabelas *cor* e *serviço* e o componente *ZQuery* para as tabelas *veiculo* e *cliente*?
- Por que os **SELECT** dos *ZQueries* possuem a cláusula *LIMIT 0*?
- O que o *VeiculoZQuery* tem de diferente? Por que esta diferença?

ZTables e ZQueries

- Os objetos do banco de dados (sequências e tabelas) podem ou não ser precedidos de *public*;
- Por que usamos o componente *ZTable* para as tabelas *cor* e *serviço* e o componente *ZQuery* para as tabelas *veiculo* e *cliente*?
- Por que os **SELECT** dos *ZQueries* possuem a cláusula *LIMIT 0*?
- Uma consulta com esta cláusula permite que o *Dataset* seja aberto se retornar dados, de forma que o usuário pode incluir novos dados.
- O que o *VeiculoZQuery* tem de diferente? Por que esta diferença?
- O SQL busca os veículos já com a denominação do cliente, isto é feito para exibir tal informação nos formulários.

LavacaoData (unit LavacaoDM) I



LavacaoData (unit LavacaoDM) II

```
object LavacaoData: TLavacaoData
  object LavacaoZQuery: TZQuery
    Connection = ConexaoData.PrincipalZConnection
    UpdateObject = LavacaoZUpdateSQL
    SQL.Strings = (
      'SELECT l.*,'
      '       v.placa,'
      '       v.den_veiculo,'
      '       c.id_cliente,'
      '       c.den_cliente'
      'FROM lavacao AS l,'
      '     veiculo as v,'
      '     cliente as C'
      'WHERE l.id_veiculo = v.id_veiculo'
      '  AND v.id_cliente = c.id_cliente'
      'LIMIT 0;'
    )
```

LavacaoData (unit LavacaoDM) III

```
  Sequence = LavacaoZSequence
  SequenceField = 'id_lavacao'
end
object LavacaoServicoZQuery: TZQuery
  Connection = ConexaoData.PrincipalZConnection
  UpdateObject = LavacaoServicoZUpdateSQL
  SQL.Strings = (
    'SELECT ls.*, s.den_servico'
    'FROM lavacao_servico AS ls,'
    '             servico AS s'
    'WHERE ls.id_servico = s.id_servico'
  )
  MasterFields = 'id_lavacao'
  MasterSource = LavacaoDatasource
  LinkedFields = 'id_lavacao'
end
object LavacaoZUpdateSQL: TZUpdateSQL
```

LavacaoData (unit LavacaoDM) IV

```
DeleteSQL.Strings = (
  'DELETE FROM lavacao'
  'WHERE'
  '  lavacao.id_lavacao = :OLD_id_lavacao'
)
InsertSQL.Strings = (
  'INSERT INTO lavacao'
  '  (id_lavacao, id_veiculo, data_entrada, hora_entrada, data_sa
  '   valor)'
  'VALUES'
  '  (:id_lavacao, :id_veiculo, :data_entrada, :hora_entrada, :da
  '   :hora_saida, :valor)'
)
ModifySQL.Strings = (
  'UPDATE lavacao SET'
  '  id_lavacao = :id_lavacao,'
  '  id_veiculo = :id_veiculo,'
```

LavacaoData (unit LavacaoDM) V

```
    '  data_entrada = :data_entrada,'
    '  hora_entrada = :hora_entrada,'
    '  data_saida = :data_saida,'
    '  hora_saida = :hora_saida,'
    '  valor = :valor'
    'WHERE'
    '  lavacao.id_lavacao = :OLD_id_lavacao'
  )
end
object LavacaoZSequence: TZSequence
  Connection = ConexaoData.PrincipalZConnection
  SequenceName = 'lavacao_id_lavacao_seq'
end
object LavacaoDatasource: TDatasource
  AutoEdit = False
  DataSet = LavacaoZQuery
end
```

LavacaoData (unit LavacaoDM) VI

```
object LavacaoServicoDatasource: TDatasource
  AutoEdit = False
  DataSet = LavacaoServicoZQuery
end
object LavacaoServicoZUpdateSQL: TZUpdateSQL
  DeleteSQL.Strings = (
    'DELETE FROM lavacao_servico'
    'WHERE'
    '  lavacao_servico.id_lavacao = :OLD_id_lavacao AND'
    '  lavacao_servico.id_servico = :OLD_id_servico'
  )
  InsertSQL.Strings = (
    'INSERT INTO lavacao_servico'
    '  (id_lavacao, id_servico, valor)'
    'VALUES'
    '  (:id_lavacao, :id_servico, :valor)'
  )
```

LavacaoData (unit LavacaoDM) VII

```
    ModifySQL.Strings = (
      'UPDATE lavacao_servico SET'
      '  id_lavacao = :id_lavacao,'
      '  id_servico = :id_servico,'
      '  valor = :valor'
      'WHERE'
      '  lavacao_servico.id_lavacao = :OLD_id_lavacao AND'
      '  lavacao_servico.id_servico = :OLD_id_servico'
    )
  end
end
```

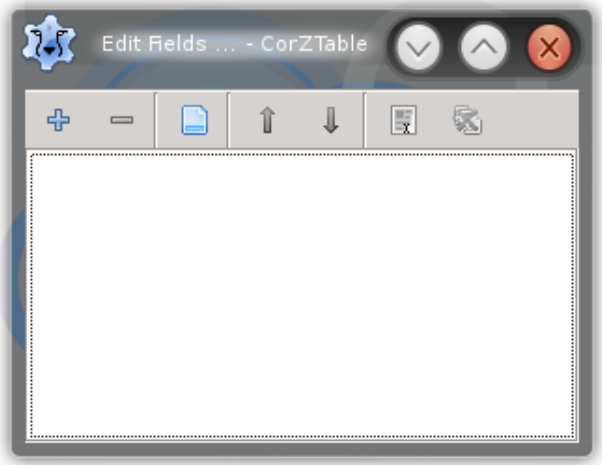
- O que os componentes *VeiculoZQuery*, *LavacaoZQuery* e *LavacaoServicoZQuery* têm em comum?

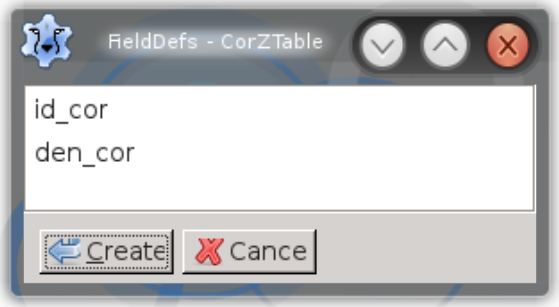
UsuarioData (unit UsuarioDM) I

```
object UsuarioData: TUsuarioData
  object UsuarioZTable: TZTable
    Connection = ConexaoData.PrincipalZConnection
    TableName = 'usuario'
  end
  object UsuarioDatasource: TDatasource
    AutoEdit = False
    DataSet = UsuarioZTable
  end
end
```

Editando Campos

- Depois que um *DataSet* está configurado, podemos editar os campos do mesmo para personalizá-los, ou até mesmo criarmos novos campos;[1]
- Para cada *DataSet* podemos editar os campos através do menu *popup* *Edit Fields ...*;
- A janela exibida permite a inclusão dos campos existentes no banco de dados por meio do botão +.





- O botão *Create* cria os campos para edição.

[1] Estes novos campos não precisam existir no banco de dados.

Editando Campos do Sistema Exemplo

- Como exercício vamos criar os campos de todos os *DataSets* de nosso sistema exemplo;
- Observe que os *DataSets* do projeto **não devem estar ativos**;[2]
- Na criação dos campos é interessante modificar a propriedade *DisplayLabel* que é o rótulo para o campo.

[2] A ativação será feita via código posteriormente.

Formulários de Cadastros

- Na criação dos formulários de cadastros vamos considerar o tipo de *DataSet* envolvido;
- Os formulários ligados a *ZQueries* serão construídos com vários componentes (*DBEdits*, *DBLookupComBoxes*, e outros);
- Os formulários ligados a *ZTables* (exceto o de usuários) serão construídos com *DBGrids*.

ServicoCadForm (unit ServicoCadFM) I


```
object ServicoCadForm: TServicoCadForm
  Caption = 'Cadastro de Serviços'
  Position = poScreenCenter
  object Panel1: TPanel
    Align = alTop
    BevelOuter = bvNone
    object SairBitBtn: TBitBtn
      Caption = '&Sair'
```

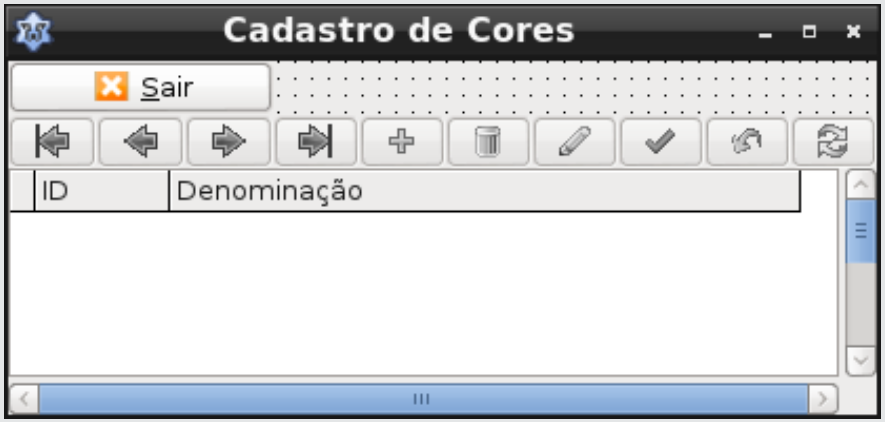
ServicoCadForm (unit ServicoCadFM) II

```
    Kind = bkClose
  end
  object ClienteDBNavigator: TDBNavigator
    Align = alBottom
    DataSource = CadastroData.ServicoDatasource
  end
end
object ServicoDBGrid: TDBGrid
  Align = alClient
  Columns = <
    item
      Title.Caption = 'ID'
      FieldName = 'id_servico'
      ReadOnly = True
    end
    item
      Title.Caption = 'Denominação'
```

ServicoCadForm (unit ServicoCadFM) III

```
      Width = 300
      FieldName = 'den_servico'
    end
    item
      Alignment = taRightJustify
      Title.Caption = 'Valor'
      FieldName = 'valor'
    end>
  DataSource = CadastroData.ServicoDatasource
end
end
```

CorCadForm (unit CorCadFM) I




```
object CorCadForm: TCorCadForm
  Caption = 'Cadastro de Cores'
  Position = poScreenCenter
  object Panel1: TPanel
    Align = alTop
    BevelOuter = bvNone
    object SairBitBtn: TBitBtn
```

CorCadForm (unit CorCadFM) II

```
    Caption = '&Sair'
    Kind = bkClose
  end
  object CorDBNavigator: TDBNavigator
    Align = alBottom
    DataSource = CadastroData.CorDatasource
  end
end
object CorDBGrid: TDBGrid
  Align = alClient
  Columns = <
    item
      Title.Caption = 'ID'
      FieldName = 'id_cor'
    end
    item
      Title.Caption = 'Denominação'
```

CorCadForm (unit CorCadFM) III

```
        Width = 300
        FieldName = 'den_cor'
      end>
    DataSource = CadastroData.CorDatasource
  end
end
```

ClienteCadForm (unit ClienteCadFM) I


```
object ClienteCadForm: TClienteCadForm
  Caption = 'Cadastro de Clientes'
  Position = poScreenCenter
  object Panel1: TPanel
    Align = alTop
    BevelOuter = bvNone
```

ClienteCadForm (unit ClienteCadFM) II

```
  object SairBitBtn: TBitBtn
    Caption = '&Sair'
    Kind = bkClose
  end
  object BuscarBitBtn: TBitBtn
    Caption = '&Buscar'
  end
  object ClienteDBNavigator: TDBNavigator
    Align = alBottom
    DataSource = CadastroData.ClienteDatasource
  end
end
object Label1: TLabel
  Caption = '&ID'
  FocusControl = IDClienteDBEdit
end
object Label2: TLabel
```

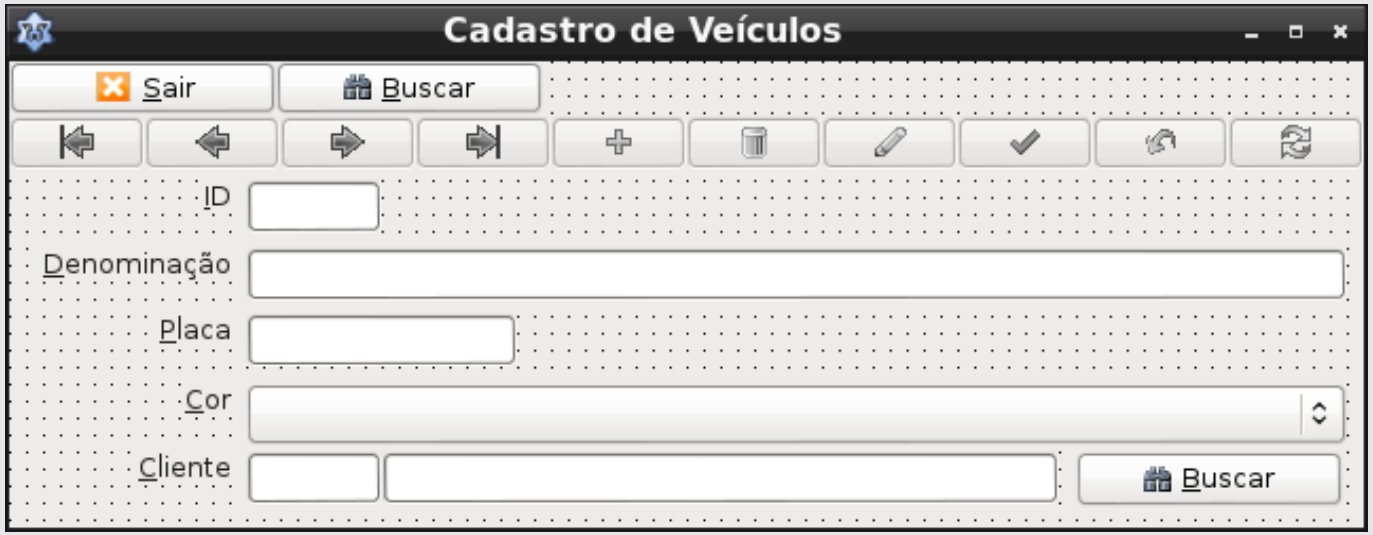
ClienteCadForm (unit ClienteCadFM) III

```
  Caption = '&Denominação'
  FocusControl = DenClienteDBEdit
end
object Label3: TLabel
  Caption = '&Telefone'
  FocusControl = TelefoneDBEdit
end
object Label4: TLabel
  Caption = '&Endereço'
  FocusControl = EnderecoDBEdit
end
object IDClienteDBEdit: TDBEdit
  DataField = 'id_cliente'
  DataSource = CadastroData.ClienteDatasource
  ReadOnly = True
end
object DenClienteDBEdit: TDBEdit
```

ClienteCadForm (unit ClienteCadFM) IV

```
    DataField = 'den_cliente'
    DataSource = CadastroData.ClienteDatasource
  end
  object TelefoneDBEdit: TDBEdit
    DataField = 'telefone'
    DataSource = CadastroData.ClienteDatasource
  end
  object EnderecoDBEdit: TDBEdit
    DataField = 'endereco'
    DataSource = CadastroData.ClienteDatasource
  end
end
```

VeiculoCadForm (unit VeiculoCadFM) I



VeiculoCadForm (unit VeiculoCadFM) II

```
object VeiculoCadForm: TVeiculoCadForm
  Caption = 'Cadastro de Veículos'
  Position = poScreenCenter
  object Panel1: TPanel
    Align = alTop
    BevelOuter = bvNone
    object SairBitBtn: TBitBtn
      Caption = '&Sair'
      Kind = bkClose
    end
    object BuscarBitBtn: TBitBtn
      Caption = '&Buscar'
    end
    object VeiculoDBNavigator: TDBNavigator
      Align = alBottom
      DataSource = CadastroData.VeiculoDatasource
    end
```

VeiculoCadForm (unit VeiculoCadFM) III

```
end
object Label1: TLabel
  Caption = '&ID'
  FocusControl = IDVeiculoDBEdit
end
object Label2: TLabel
  Caption = '&Denominação'
  FocusControl = DenVeiculoDBEdit
end
object Label3: TLabel
  Caption = '&Placa'
  FocusControl = PlacaDBEdit
end
object Label4: TLabel
  Caption = '&Cor'
  FocusControl = CorDBLookup
end
```

VeiculoCadForm (unit VeiculoCadFM) IV

```
object Label5: TLabel
  Caption = '&Cliente'
  FocusControl = IDClienteDBEdit
end
object IDVeiculoDBEdit: TDBEdit
  DataField = 'id_veiculo'
  DataSource = CadastroData.VeiculoDatasource
  ReadOnly = True
end
object DenVeiculoDBEdit: TDBEdit
  DataField = 'den_veiculo'
  DataSource = CadastroData.VeiculoDatasource
end
object PlacaDBEdit: TDBEdit
  DataField = 'placa'
  DataSource = CadastroData.VeiculoDatasource
end
```

VeiculoCadForm (unit VeiculoCadFM) V

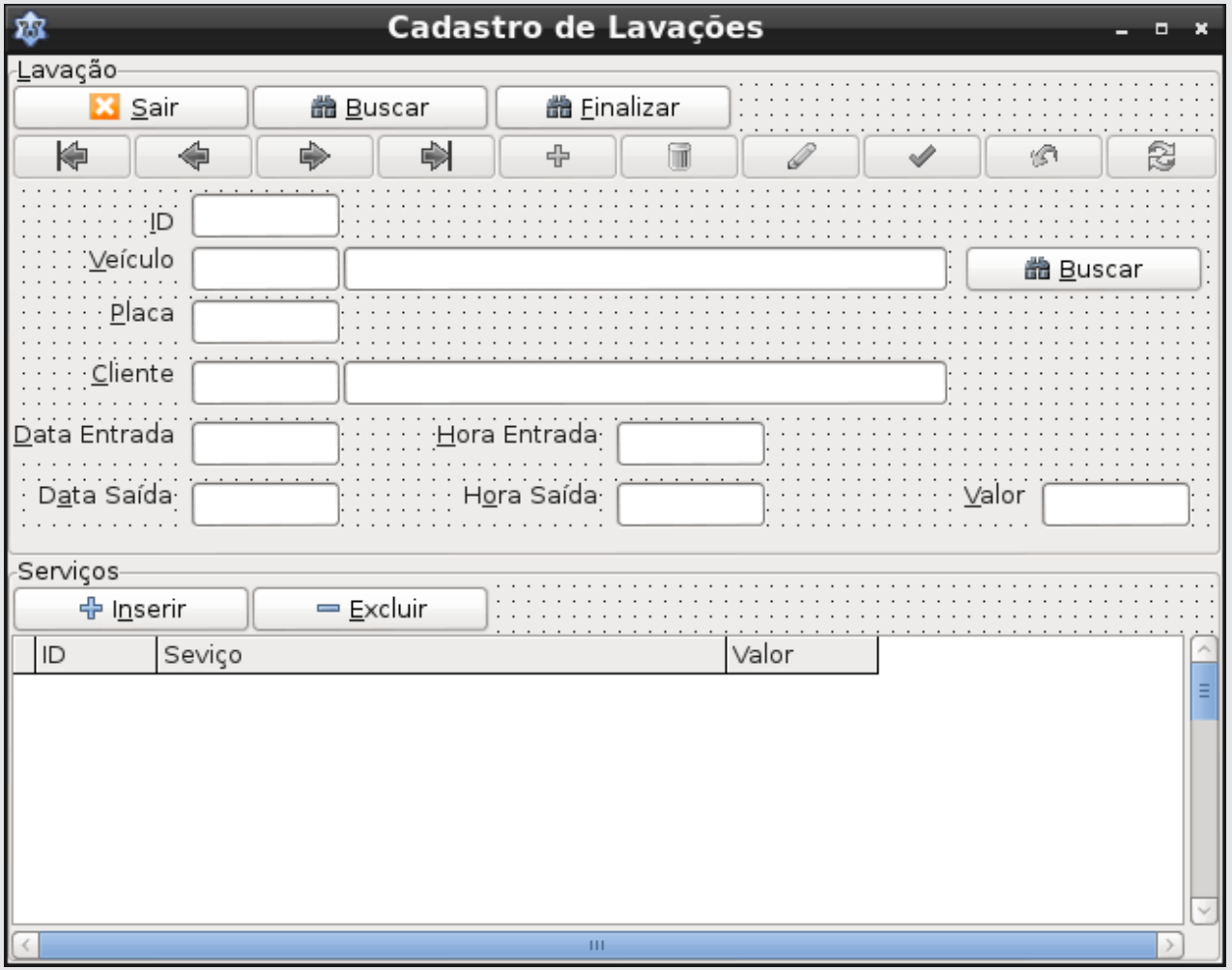
```
object CorDBLookup: TDBLookupComboBox
  DataField = 'id_cor'
  DataSource = CadastroData.VeiculoDatasource
  KeyField = 'id_cor'
  ListField = 'den_cor'
  ListSource = CadastroData.CorDatasource
  Style = csDropDownList
end
object IDClienteDBEdit: TDBEdit
  DataField = 'id_cliente'
  DataSource = CadastroData.VeiculoDatasource
end
object DenClienteDBEdit: TDBEdit
  DataField = 'den_cliente'
  DataSource = CadastroData.VeiculoDatasource
  ReadOnly = True
end
```

VeiculoCadForm (unit VeiculoCadFM) VI

```
  object BuscarClienteBitBtn: TBitBtn
    Caption = '&Buscar'
  end
end
```

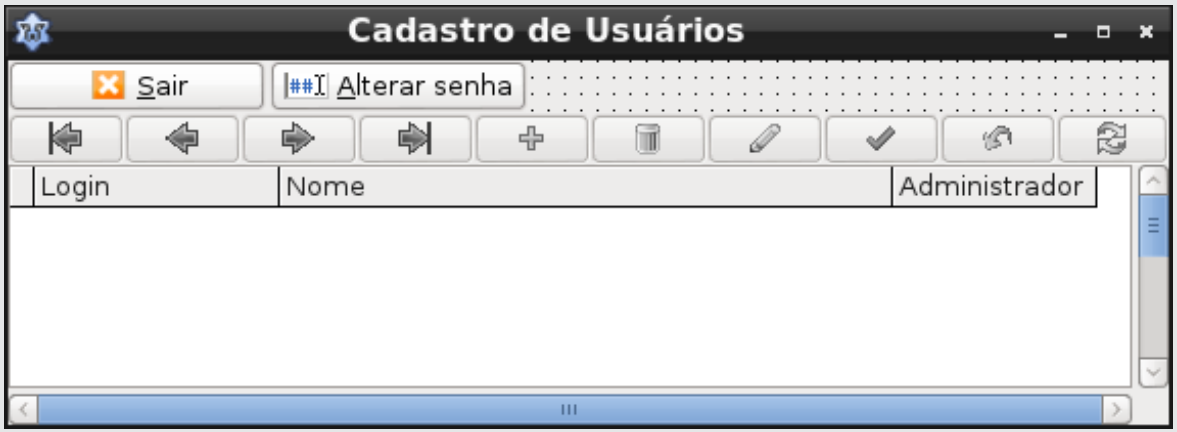
LavacaoCadForm (unit LavacaoCadFM) I

LavacaoCadForm (unit LavacaoCadFM) II




```
object LavacaoCadForm: TLavacaoCadForm
  Caption = 'Cadastro de Lavações'
  Position = poScreenCenter
```

UsuarioCadForm (unit usuarioCadFM) I




```
object UsuarioCadForm: TUsuarioCadForm
  Caption = 'Cadastro de Usuários'
  Position = poScreenCenter
  object Panel1: TPanel
    Align = alTop
    BevelOuter = bvNone
    object SairBitBtn: TBitBtn
```

UsuarioCadForm (unit usuarioCadFM) II

```
    Caption = '&Sair'
    Kind = bkClose
  end
  object ClienteDBNavigator: TDBNavigator
    Align = alBottom
    BevelOuter = bvNone
    DataSource = UsuarioData.UsuarioDatasource
  end
  object AlterarSenhaBitBtn: TBitBtn
    Caption = '&Alterar senha'
  end
end
object DBGrid1: TDBGrid
  Align = alClient
  Columns = <
    item
      Title.Caption = 'Login'
```

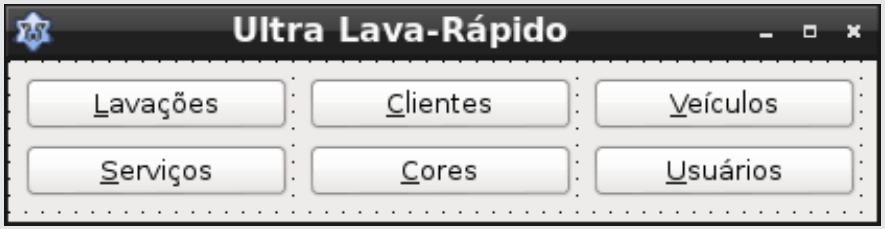
UsuarioCadForm (unit usuarioCadFM) III

```
      Width = 120
      FieldName = 'login_usuario'
    end
    item
      Title.Caption = 'Nome'
      Width = 300
      FieldName = 'nome'
    end
    item
      ButtonStyle = cbsCheckboxColumn
      Title.Caption = 'Administrador'
      Width = 100
      ValueChecked = 'A'
      ValueUnchecked = 'U'
      FieldName = 'tipo'
    end>
  DataSource = UsuarioData.UsuarioDatasource
```

UsuarioCadForm (unit usuarioCadFM) IV

```
  end
end
```

PrincipalForm (unit PrincipalFM)



Programação Inicial

- Ainda não foi implementada nenhuma linha de código em nosso sistema;
- Primeiramente vamos acrescentar as *units* dos módulos de dados na seção *implementation* das *units* de formulários necessárias:

  CadastroDM ClienteCadFM, CorCadFM, ServicoCadFM, CadastroDM;

  LavacaoDM LavacaoCadFM;

  UsuarioDM UsuarioCadFM.

- Através da opção .... configure o projeto para os módulos de dados serem criados primeiro, ordem sugerida:
  1. ConexaoData
  2. CadastroData
  3. LavacaoData
  4. UsuarioData
  5. PrincipalForm
  6. ServicoCadForm
  7. CorCadForm
  8. ClienteCadForm
  9. VeiculoCadForm
  10. LavacaoCadForm
  11. UsuarioCadForm

Controle de Exibição dos Formulários I

- O controle de exibição dos formulários será feito por meio dos módulos de dados, para isto acrescente as *units* dos módulos de dados à *unit PrincipalFM*.
- Para a exibição do *LavacaoCadForm* crie o seguinte código no *LavacaoDM*:

```
unit LavacaoDM;
...
  TLavacaoData = class(TDataModule)
    ...
    procedure LavacaoCadastroShow();
  private
...
implementation
```

Controle de Exibição dos Formulários II

```
uses LavacaoCadFM;

procedure TLavacaoData.LavacaoCadastroShow;
begin
  LavacaoCadForm.ShowModal();
end;
...
```

- Para cada módulo de dados devem ser criados os procedimentos de exibição de seus formulários.

Código do Formulário Principal

```
unit PrincipalFM;
...
  TPrincipalForm = class(TForm)
    ...
    procedure LavacoesBitBtnClick(Sender: TObject);
  private
...
implementation

uses CadastroDM, LavacaoDM, UsuarioDM;
...
procedure TPrincipalForm.LavacoesBitBtnClick(Sender: TObject);
begin
  LavacaoData.LavacaoCadastroShow();
end;
...
```

O código dos demais botões são similares.

Inicialização do Sistema I

- A inicialização do sistema será feita via código, portanto vamos mudar a propriedade *Connected* para *False* do componente *ConexaoData.PrincipalZConnection*;
- Os código de inicialização serão adicionados nos eventos *Create* dos módulos de dados como se segue:[3]

ConexaoDM

```
procedure TConexaoData.DataModuleCreate(Sender: TObject);
begin
  PrincipalZConnection.Connect();
end;
```

Inicialização do Sistema II

CadastroDM

```
procedure TCadastroData.DataModuleCreate(Sender: TObject);
begin
  CorZTable.Open();
  ServicoZTable.Open();
  VeiculoZQuery.Open();
  ClienteZQuery.Open();
end;
```

Inicialização do Sistema III

LavacaoDM

```
procedure TLavacaoData.DataModuleCreate(Sender: TObject);
begin
  LavacaoZQuery.Open();
  LavacaoServicoZQuery.Open();
end;
```

Inicialização do Sistema IV

UsuarioDM

```
procedure TUsuarioData.DataModuleCreate(Sender: TObject);
begin
  UsuarioZTable.Open();
end;
```

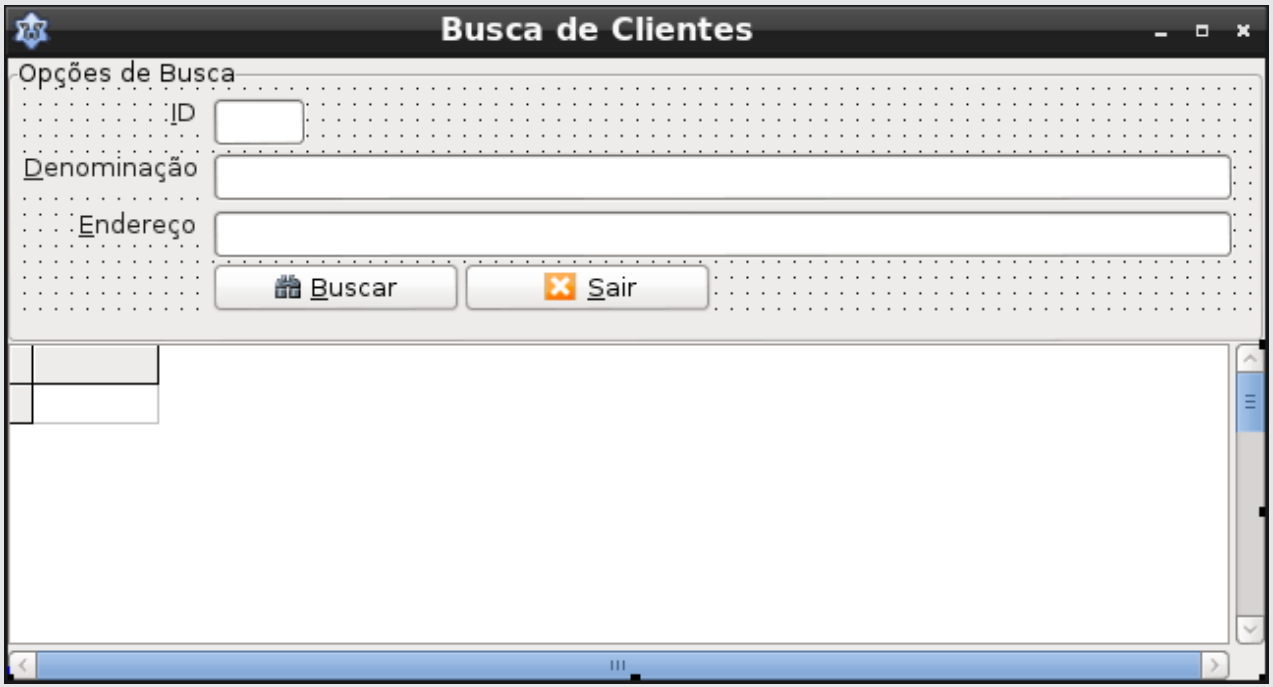
[3]Não basta inserir o código, é preciso atribuir o evento.

Execução do Sistema

- Agora nosso sistema pode ser executado, contudo iremos perceber que os formulários vinculados a *ZQueries* permitirão apenas a inclusão de novos dados e não exibirão os dados já cadastrados;
- Isto acontece por que incluímos a cláusula **LIMIT 0** em nossas consultas;
- Não podemos retirar tal cláusula porque os *ZQueries* funcionariam como *ZTables*, então precisaremos criar formulários de busca para resolver este problema.[4]

[4] Todos os dados da consulta são mantidos na memória.

ClienteBuscaForm (unit ClienteBuscaFM) I



ClienteBuscaForm (unit ClienteBuscaFM) II

```
object ClienteBuscaForm: TClienteBuscaForm
  Caption = 'Busca de Clientes'
  Position = poScreenCenter
  object GroupBox1: TGroupBox
    Align = alTop
    Caption = 'Opções de Busca'
    object Label1: TLabel
      Caption = '&ID'
      FocusControl = IDEdit
    end
    object Label2: TLabel
      Caption = '&Denominação'
      FocusControl = DenominacaoEdit
    end
    object Label3: TLabel
      Caption = '&Endereço'
      FocusControl = EnderecoEdit
    end
```

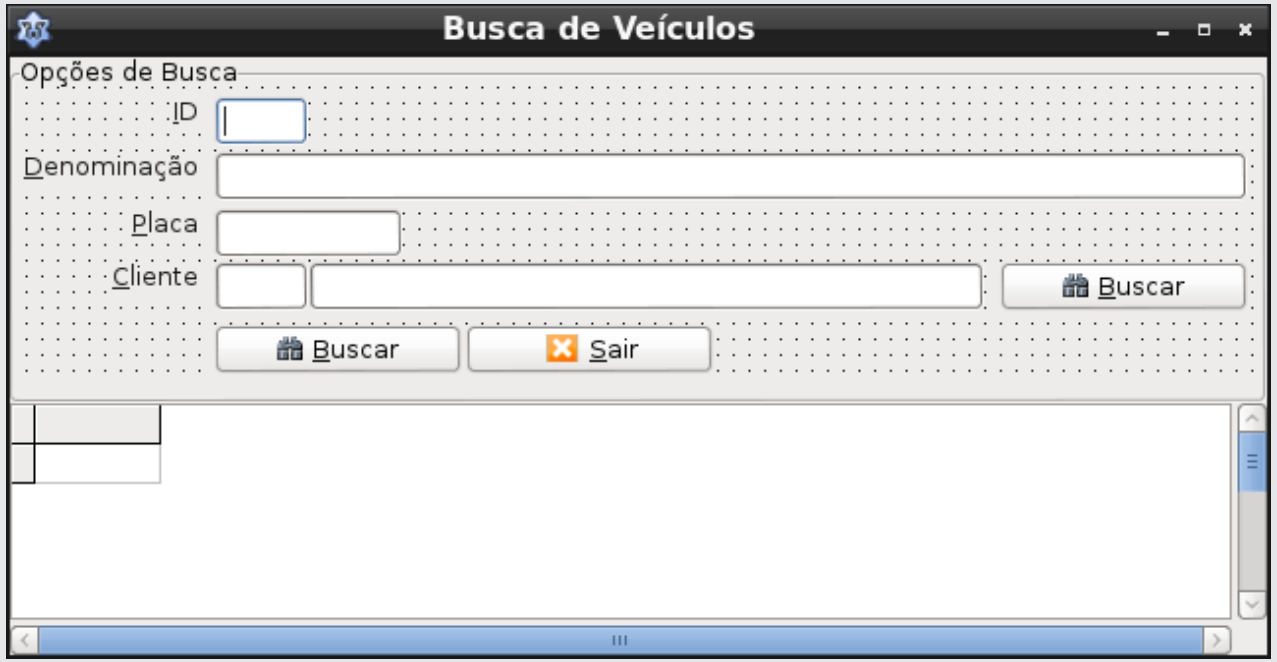
ClienteBuscaForm (unit ClienteBuscaFM) III

```
  object IDClienteEdit: TEdit
  end
  object DenClienteEdit: TEdit
  end
  object EnderecoEdit: TEdit
  end
  object SairBitBtn: TBitBtn
    Caption = '&Sair'
    Kind = bkClose
  end
  object BuscarBitBtn: TBitBtn
    Caption = '&Buscar'
  end
end
object ClienteDBGrid: TDBGrid
  Align = alClient
  DataSource = CadastroData.ClienteDatasource
```

ClienteBuscaForm (unit ClienteBuscaFM) IV

```
  end
end
```

VeiculoBuscaForm (unit VeiculoBuscaFM) I



VeiculoBuscaForm (unit VeiculoBuscaFM) II

```
object VeiculoBuscaForm: TVeiculoBuscaForm
  Caption = 'Busca de Veículos'
  object GroupBox1: TGroupBox
    Align = alTop
    Caption = 'Opções de Busca'
    object Label1: TLabel
      Caption = '&ID'
      FocusControl = IDVeiculoEdit
    end
    object Label2: TLabel
      Caption = '&Denominação'
      FocusControl = DenVeiculoEdit
    end
    object Label3: TLabel
      Caption = '&Placa'
      FocusControl = PlacaEdit
    end
```

VeiculoBuscaForm (unit VeiculoBuscaFM) III

```
object Label4: TLabel
  Caption = '&Cliente'
  FocusControl = IDClienteEdit
end
object IDVeiculoEdit: TEdit
end
object DenVeiculoEdit: TEdit
end
object PlacaEdit: TEdit
end
object IDClienteEdit: TEdit
end
object DenClienteEdit: TEdit
end
object BuscarBitBtn: TBitBtn
  Caption = '&Buscar'
end
```

VeiculoBuscaForm (unit VeiculoBuscaFM) IV

```
    object SairBitBtn: TBitBtn
      Caption = '&Sair'
      Kind = bkClose
    end
    object BuscarClienteBitBtn: TBitBtn
      Caption = '&Buscar'
    end
  end
  object VeiculoDBGrid: TDBGrid
    Align = alClient
    DataSource = CadastroData.VeiculoDatasource
  end
end
```

LavacaoBuscaForm (unit LavacaoBuscaFM) I

LavacaoBuscaForm (unit LavacaoBuscaFM) II

```
object LavacaoBuscaForm: TLavacaoBuscaForm
  Caption = 'Busca de Lavações'
  object GroupBox1: TGroupBox
    Caption = 'Opções de Busca'
    object Label1: TLabel
      Caption = '&ID'
      FocusControl = IDLavacaoEdit
    end
    object Label2: TLabel
      Caption = '&Veículo'
      FocusControl = IDVeiculoEdit
    end
    object Label3: TLabel
      Caption = '&Placa'
      FocusControl = PlacaEdit
    end
    object IDVeiculoEdit: TEdit
```

LavacaoBuscaForm (unit LavacaoBuscaFM) III

```
end
object DenVeiculoEdit: TEdit
end
object PlacaEdit: TEdit
end
object SairBitBtn: TBitBtn
  Caption = '&Sair'
  Kind = bkClose
end
object BuscarBitBtn: TBitBtn
  Caption = '&Buscar'
end
object IDClienteEdit: TEdit
end
object DenClienteEdit: TEdit
end
object Label4: TLabel
```

LavacaoBuscaForm (unit LavacaoBuscaFM) IV

```
  Caption = '&Cliente'
  FocusControl = IDClienteEdit
end
object BuscarClienteBitBtn: TBitBtn
  Caption = '&Buscar'
end
object IDLavacaoEdit: TEdit
end
object DataEntradaEdit: TEdit
end
object DataSaidaEdit: TEdit
end
object Label5: TLabel
  Caption = 'Data &Entrada'
  FocusControl = DataEntradaEdit
end
object Label7: TLabel
```

LavacaoBuscaForm (unit LavacaoBuscaFM) V

```
      Caption = '&Data Saída'
      FocusControl = DataSaidaEdit
    end
    object BuscarVeiculoBitBtn: TBitBtn
      Caption = '&Buscar'
    end
  end
  object VeiculoDBGrid: TDBGrid
    Align = alClient
    DataSource = CadastroData.LavacaoDatasource
  end
end
```

Codificação das Buscas

- Existem duas situações de busca:
  Busca do cadastro atual Neste caso, após o usuário fazer a busca, o *DataSet* deve ficar aberto para que os dados fiquem disponíveis no cadastro. Assim o usuário poderá alterar ou excluir um dado existente;
  Busca de chave estrangeira O dado que o usuário selecionou deve ser retornado para o cadastro atual e o *DataSet* pode ser fechado. Desta maneira, o dado buscado pode ser usado para preencher alguns campos do cadastro atual.
- Além disto a chamada dos formulários de busca será feita por sub-rotinas nos módulos de dados, assim como foi feito para os cadastros.

Busca de Cliente

- Pensando na busca de chave estrangeira, devemos prover uma maneira de os dados do cliente buscado serem retornados. Para isto vamos criar o tipo de registro para armazenar os dados de retorno na *unit CadastroDM*:[5]

```
...
type
  // Dados de Cliente
  TCliente = record
    id_cliente: string;
    den_cliente: string;
    telefone: string;
    endereco: string;
  end;
...
```

[5] Os dados do registro são todos **String** para facilitar a cópia para *Edits* e outros componentes

Função ClienteBuscaShow() I

```
unit CadastroDM;
...
TCadastroData = class(TDataModule)
    ...
    function ClienteBuscaShow(FechaDataset: Boolean): TCliente;
  private
...
implementation
...
function TCadastroData.ClienteBuscaShow(FechaDataset: boolean):
 ↪  TCliente;
var
  // Dados a serem retornados
  Cliente: TCliente;
begin
  // Exibe o formulário para o usuário personalizar a busca
```

Função ClienteBuscaShow() II

```
  ClienteBuscaForm.ShowModal();
  // Verifica se a busca encontrou algum dado
  if (ClienteZQuery.Active) and (ClienteZQuery.RecordCount > 0)
↪   then begin
    // Pega os dados atuais do Dataset para retornar
    Cliente.id_cliente :=
↪   ClienteZQuery.FieldByName('id_cliente').AsString;
    Cliente.den_cliente :=
↪   ClienteZQuery.FieldByName('den_cliente').AsString;
    Cliente.endereco :=
↪   ClienteZQuery.FieldByName('endereco').AsString;
    Cliente.telefone :=
↪   ClienteZQuery.FieldByName('telefone').AsString;
  end else begin
    // Se não há dados atribui strings vazias para retornar
    Cliente.id_cliente := '';
    Cliente.den_cliente := '';
```

Função ClienteBuscaShow() III

```
    Cliente.endereco := '';
    Cliente.telefone := '';
  end;
  // Verifica se o dataset deve ser fechado
  if (FechaDataset) then begin
    // Fecha o dataset
    ClienteZQuery.Close();
  end;
  // Retorna os dados buscados
  ClienteBuscaShow := Cliente;
end;
```

Código para a unit ClienteBuscaFM I

```
...
uses CadastroDM;
...
procedure TClienteBuscaForm.BuscarBitBtnClick(Sender: TObject);
begin
  // Usando CadastroData.ClienteZQuery para os próximos comandos
  with  CadastroData.ClienteZQuery do begin
    // Fecha o dataset
    Close();
    // Limpa o SQL do DataSet
    SQL.Clear();
    // Insere nova consulta
    SQL.Add('SELECT * FROM cliente WHERE 1=1');
    // Parâmetros para id_cliente
    if (IDClienteEdit.Text <> '') then begin
      SQL.Add('AND id_cliente = :id_cliente');
```

Código para a unit ClienteBuscaFM II

```
      ParamByName('id_cliente').AsString := IDClienteEdit.Text;
    end;
    // Parâmetros para den_cliente
    if (DenClienteEdit.Text <> '') then begin
      SQL.Add('AND den_cliente LIKE ' + QuotedStr('%' +
↪   DenClienteEdit.Text + '%'));
    end;
    // Parâmetros para endereco
    if (EnderecoEdit.Text <> '') then begin
      SQL.Add('AND endereco LIKE ' + QuotedStr('%' +
↪   EnderecoEdit.Text + '%'));
    end;
    // Abre o DataSet (executa a consulta)
    Open();
  end;
end;
```

Código para a unit ClienteCadFM

```
procedure TClienteCadForm.BuscarBitBtnClick(Sender: TObject);
begin
  // Chama a busca e não fecha Dataset
  CadastroData.ClienteBuscaShow(False);
end;
```

Código para a unit CadastroDM I

```
unit CadastroDM;
...
type
  ...
  // Dados de Veículo
  TVeiculo = record
    id_veiculo: string;
    den_veiculo: string;
    placa: string;
    id_cor: string;
    id_cliente: string;
    den_cliente: string;
  end;
  ...
  TCadastroData = class(TDataModule)
    ...
```

Código para a unit CadastroDM II

```
    function VeiculoBuscaShow(FechaDataset: boolean): TVeiculo;
  private
...
implementation
...
function TCadastroData.VeiculoBuscaShow(FechaDataset: boolean):
 ↪  TVeiculo;
var
  // Dados a serem retornados
  Veiculo: TVeiculo;
begin
  // Exibe o formulário para o usuário personalizar a busca
  VeiculoBuscaForm.ShowModal();
  // Verifica se a busca encontrou algum dado
  if (VeiculoZQuery.Active) and (VeiculoZQuery.RecordCount > 0)
 ↪  then begin
    // Pega os dados atuais do Dataset para retornar
```

Código para a unit CadastroDM III

```
    Veiculo.id_veiculo :=
↪   VeiculoZQuery.FieldByName('id_veiculo').AsString;
    Veiculo.den_veiculo :=
↪   VeiculoZQuery.FieldByName('den_veiculo').AsString;
    Veiculo.placa := VeiculoZQuery.FieldByName('placa').AsString;
    Veiculo.id_cor :=
↪   VeiculoZQuery.FieldByName('id_cor').AsString;
    Veiculo.id_cliente :=
↪   VeiculoZQuery.FieldByName('id_cliente').AsString;
    Veiculo.den_cliente :=
↪   VeiculoZQuery.FieldByName('den_cliente').AsString;
  end else begin
    // Se não há dados atribui strings vazias para retornar
    Veiculo.id_veiculo := '';
    Veiculo.den_veiculo := '';
    Veiculo.placa := '';
    Veiculo.id_cor := '';
```

Código para a unit CadastroDM IV

```
    Veiculo.id_cliente := '';
    Veiculo.den_cliente := '';
  end;
  // Verifica se o dataset deve ser fechado
  if (FechaDataset) then begin
    // Fecha o dataset
    VeiculoZQuery.Close();
  end;
  // Retorna os dados buscados
  VeiculoBuscaShow := Veiculo;
end;
```

Código para a unit VeiculoBuscaFM I

```
unit VeiculoBuscaFM;
...
implementation
...
uses CadastroDM;
...
procedure TVeiculoBuscaForm.BuscarBitBtnClick(Sender:
 ↪  TObject);
begin
  // Usando CadastroData.ClienteZQuery para os próximos
 ↪  comandos
  with  CadastroData.VeiculoZQuery do begin
    // Fecha o dataset
    Close();
```

Código para a unit VeiculoBuscaFM II

```
    // Limpa o SQL do DataSet
    SQL.Clear();
    // Insere nova consulta
    SQL.Add('SELECT v.*, c.den_cliente');
    SQL.Add('FROM veiculo AS v, Cliente AS c');
    SQL.Add('WHERE v.id_cliente = c.id_cliente');
    // Verifica se o usuário informar algum parâmetro para
↪   busca
    // Parâmetros para id_veiculo
    if (IDVeiculoEdit.Text <> '') then begin
      SQL.Add('AND id_veiculo = :id_veiculo');
      ParamByName('id_veiculo').AsString :=
↪   IDVeiculoEdit.Text;
    end;
    // Parâmetros para den_veiculo
```

Código para a unit VeiculoBuscaFM III

```
    if (DenVeiculoEdit.Text <> '') then begin
      SQL.Add('AND den_cliente LIKE ' + QuotedStr('%' +
↪   DenVeiculoEdit.Text + '%'));
    end;
    // Parâmetros para placa
    if (PlacaEdit.Text <> '') then begin
      SQL.Add('AND placa = :placa');
      ParamByName('placa').AsString := PlacaEdit.Text;
    end;
    // Parâmetros para id_cliente
    if (IDClienteEdit.Text <> '') then begin
      SQL.Add('AND id_cliente = :id_cliente');
      ParamByName('id_cliente').AsString :=
↪   IDClienteEdit.Text;
    end;
```

Código para a unit VeiculoBuscaFM IV

```
    // Parâmetros para den_cliente
    if (DenClienteEdit.Text <> '') then begin
      SQL.Add('AND den_cliente LIKE ' + QuotedStr('%' +
↪   DenClienteEdit.Text + '%'));
    end;
    // Abre o DataSet (executa a consulta)
    Open();
  end;
end;
...
procedure TVeiculoBuscaForm.BuscarClienteBitBtnClick(Sender:
↪   TObject);
var
  // Dados de cliente
  Cliente: TCliente;
```

Código para a unit VeiculoBuscaFM V

```
begin
  // Busca de cliente (fechando o dataset)
  Cliente := CadastroData.ClienteBuscaShow(True);
  // Atribui dados buscados aos respectivos Edits
  IDClienteEdit.Text := Cliente.id_cliente;
  DenClienteEdit.Text := Cliente.den_cliente;
end;
...
```

Código para a unit VeiculoCadFM I

```
unit VeiculoCadFM;
...
implementation

uses CadastroDM, DB;
...
procedure TVeiculoCadForm.BuscarBitBtnClick(Sender: TObject);
begin
  // Busca de veículo (sem fechar dataset)
  CadastroData.VeiculoBuscaShow(False);
end;
...
procedure TVeiculoCadForm.BuscarClienteBitBtnClick(Sender:
↪  TObject);
var
  // Dados de cliente
```

Código para a unit VeiculoCadFM II

```
  Cliente: TCliente;
begin
  // Busca de cliente (fechando dataset)
  Cliente := CadastroData.ClienteBuscaShow(True);
  // Usando CadastroData.VeiculoZQuery para os próximos comandos
  with CadastroData.VeiculoZQuery do begin
    // Se o dataset está ativo e em modo de escrita
    if (Active) and (State in dsWriteModes) then begin
      FieldByName('id_cliente').Value := cliente.id_cliente;
      FieldByName('den_cliente').Value := cliente.den_cliente;
    end;
  end;
end;
...
```

Código para a unit LavacaoDM I

```
unit LavacaoDM;
...
type
  // Dados de Lavacao
  TLavacao = record
    id_lavacao: string;
    data_entrada: string;
    hora_entrada: string;
    data_saida: string;
    hora_saida: string;
    valor: string;
    id_veiculo: string;
    den_veiculo: string;
    placa: string;
    id_cliente: string;
    den_cliente: string;
```

Código para a unit LavacaoDM II

```
  end;
...
implementation

uses LavacaoCadFM, LavacaoBuscaFM;
...
function TLavacaoData.LavacaoBuscaShow(FechaDataSet: boolean):
 ↪  TLavacao;
var
  // Dados a serem retornados
  Lavacao: TLavacao;
begin
  // Exibe o formulário para o usuário personalizar a busca
  LavacaoBuscaForm.ShowModal();
  // Verifica se a busca encontrou algum dado
  if (LavacaoZQuery.Active) and (LavacaoZQuery.RecordCount > 0)
 ↪  then begin
```

Código para a unit LavacaoDM III

```
    // Pega os dados atuais do Dataset para retornar
    Lavacao.id_lavacao :=
↪   LavacaoZQuery.FieldByName('id_lavacao').AsString;
    Lavacao.data_entrada :=
↪   LavacaoZQuery.FieldByName('data_entrada').AsString;
    Lavacao.hora_entrada :=
↪   LavacaoZQuery.FieldByName('hora_entrada').AsString;
    Lavacao.data_saida :=
↪   LavacaoZQuery.FieldByName('data_saida').AsString;
    Lavacao.hora_saida :=
↪   LavacaoZQuery.FieldByName('hora_saida').AsString;
    Lavacao.valor := LavacaoZQuery.FieldByName('valor').AsString;
    Lavacao.id_veiculo :=
↪   LavacaoZQuery.FieldByName('id_veiculo').AsString;
    Lavacao.den_veiculo :=
↪   LavacaoZQuery.FieldByName('den_veiculo').AsString;
    Lavacao.placa := LavacaoZQuery.FieldByName('placa').AsString;
```

Código para a unit LavacaoDM IV

```
    Lavacao.id_cliente :=
↪   LavacaoZQuery.FieldByName('id_cliente').AsString;
    Lavacao.den_cliente :=
↪   LavacaoZQuery.FieldByName('den_cliente').AsString;
  end else begin
    // Se não há dados atribui strings vazias para retornar
    Lavacao.id_lavacao := '';
    Lavacao.data_entrada := '';
    Lavacao.hora_entrada := '';
    Lavacao.data_saida := '';
    Lavacao.hora_saida := '';
    Lavacao.valor := '';
    Lavacao.id_veiculo := '';
    Lavacao.den_veiculo := '';
    Lavacao.placa := '';
    Lavacao.id_cliente := '';
    Lavacao.den_cliente := '';
```

Código para a unit LavacaoDM V

```
  end;
  // Verifica se o dataset deve ser fechado
  if (FechaDataset) then begin
    // Fecha o dataset
    LavacaoZQuery.Close();
  end;
  // Retorna os dados buscados
  LavacaoBuscaShow := Lavacao;
end;
...
```

Código para a unit LavacaoBuscaFM I

```
unit LavacaoBuscaFM;
...
implementation

uses LavacaoDM, CadastroDM;
...
procedure TLavacaoBuscaForm.BuscarBitBtnClick(Sender:
 ↪  TObject);
begin
  // Usando CadastroData.ClienteZQuery para os próximos
 ↪  comandos
  with  LavacaoData.LavacaoZQuery do begin
    // Fecha o dataset
    Close();
```

Código para a unit LavacaoBuscaFM II

```
    // Limpa o SQL do DataSet
    SQL.Clear();
    // Insere nova consulta
    SQL.Add('SELECT l.*, v.placa, v.den_veiculo,
↪   c.id_cliente, c.den_cliente');
    SQL.Add('FROM lavacao AS l, veiculo as v, cliente as C');
    SQL.Add('WHERE l.id_veiculo = v.id_veiculo');
    SQL.Add('AND v.id_cliente = c.id_cliente');
    // Verifica se o usuário informar algum parâmetro para
↪   busca
    // Parâmetros para id_lavacao
    if (IDLavacaoEdit.Text <> '') then begin
      SQL.Add('AND id_lavacao = :id_lavacao');
      ParamByName('id_lavacao').AsString :=
↪   IDLavacaoEdit.Text;
```

Código para a unit LavacaoBuscaFM III

```
    end;
    // Parâmetros para id_veiculo
    if (IDVeiculoEdit.Text <> '') then begin
      SQL.Add('AND id_veiculo = :id_veiculo');
      ParamByName('id_veiculo').AsString :=
↪   IDVeiculoEdit.Text;
    end;
    // Parâmetros para den_veiculo
    if (DenVeiculoEdit.Text <> '') then begin
      SQL.Add('AND den_cliente LIKE ' + QuotedStr('%' +
↪   DenVeiculoEdit.Text + '%'));
    end;
    // Parâmetros para placa
    if (PlacaEdit.Text <> '') then begin
      SQL.Add('AND placa = :placa');
```

Código para a unit LavacaoBuscaFM IV

```
    ParamByName('placa').AsString := PlacaEdit.Text;
  end;
  // Parâmetros para id_cliente
  if (IDClienteEdit.Text <> '') then begin
    SQL.Add('AND id_cliente = :id_cliente');
    ParamByName('id_cliente').AsString :=
↪   IDClienteEdit.Text;
  end;
  // Parâmetros para den_cliente
  if (DenClienteEdit.Text <> '') then begin
    SQL.Add('AND den_cliente LIKE ' + QuotedStr('%' +
↪   DenClienteEdit.Text + '%'));
  end;
  // Parâmetros para data_entrada
  if (DataEntradaEdit.Text <> '') then begin
```

Código para a unit LavacaoBuscaFM V

```
      SQL.Add('AND data_entrada = :data_entrada');
      ParamByName('data_entrada').AsString :=
↪   DataEntradaEdit.Text;
    end;
    // Parâmetros para data_saida
    if (DataSaidaEdit.Text <> '') then begin
      SQL.Add('AND data_saida = :data_saida');
      ParamByName('data_saida').AsString :=
↪   DataSaidaEdit.Text;
    end;
    // Abre o DataSet (executa a consulta)
    Open();
  end;
end;
```

Código para a unit LavacaoBuscaFM VI

```
procedure TLavacaoBuscaForm.BuscarClienteBitBtnClick(Sender:
 ↪  TObject);
var
  // Dados de cliente
  Veiculo: TVeiculo;
begin
  // Busca de cliente (fechando o dataset)
  Veiculo := CadastroData.VeiculoBuscaShow(True);
  // Atribui dados buscados aos respectivos Edits
  IDVeiculoEdit.Text := Veiculo.id_veiculo;
  DenVeiculoEdit.Text := Veiculo.den_veiculo;
end;

procedure TLavacaoBuscaForm.BuscarVeiculoBitBtnClick(Sender:
 ↪  TObject);
```

Código para a unit LavacaoBuscaFM VII

```
var
  // Dados de cliente
  Cliente: TCliente;
begin
  // Busca de cliente (fechando o dataset)
  Cliente := CadastroData.ClienteBuscaShow(True);
  // Atribui dados buscados aos respectivos Edits
  IDClienteEdit.Text := Cliente.id_cliente;
  DenClienteEdit.Text := Cliente.den_cliente;
end;
...
```

Código para a unit LavacaoCadFM I

```
unit LavacaoCadFM;
...
implementation
...
procedure TLavacaoCadForm.BuscarBitBtnClick(Sender: TObject);
begin
  LavacaoData.LavacaoBuscaShow(False);
end;

procedure TLavacaoCadForm.BuscarVeiculoBitBtnClick(Sender:
↪  TObject);
var
  // Dados de veiculo
  Veiculo: TVeiculo;
```

Código para a unit LavacaoCadFM II

```
begin
  // Busca de veiculo (fechando dataset)
  Veiculo := CadastroData.VeiculoBuscaShow(True);
  // Usando CadastroData.VeiculoZQuery para os próximos
↪   comandos
  with CadastroData.VeiculoZQuery do begin
    // Se o dataset está ativo e em modo de escrita
    if (Active) and (State in dsWriteModes) then begin
      FieldByName('id_veiculo').Value := Veiculo.id_veiculo;
      FieldByName('den_ceiculo').Value :=
↪   Veiculo.den_veiculo;
      FieldByName('placa').Value := Veiculo.placa;
      FieldByName('id_cliente').Value := Veiculo.id_cliente;
      FieldByName('den_cliente').Value :=
↪   Veiculo.den_cliente;
```

Código para a unit LavacaoCadFM III

```
    end;
  end;
end;
...
```

Finalizando o Cadastro de Lavações

- No cadastro de lavações ainda existem as seguintes pendências:
  Finalizar Lavação Fechar a lavação inserindo a data e hora de saída;
  Inserir Serviço Exibir a lista de serviços para o usuário escolher e inseri-lo na lavação;
  Excluir Serviço Excluir o serviço selecionado da lavação.
- Na inserção e exclusão de serviços é preciso atualizar o valor total da lavação.

Inserir Serviço

unit CadastroDM

```
unit CadastroDM;
...
interface
...
  // Dados de Serviço
  TServico = record
    id_servico: string;
    den_servico: string;
    valor: string;
  end;
...
```

AtualizaZQuery

- No módulo de dados *LavacaoData*, vamos acrescentar um *ZQuery* com as seguintes propriedades:

  `Name = AtualizaZQuery`

  `DataSource = ConexaoData.PrincipalZConnection`
- Este componente será necessário nos próximos códigos que vamos escrever.

Código para a unit LavacaoDM I

```
unit LavacaoDM;
...
  TLavacaoData = class(TDataModule)
    ...
    procedure LavacaoServicoInsere();
    procedure LavacaoAtualizaValor();
...
implementation

uses ConexaoDM, LavacaoCadFM, LavacaoBuscaFM, CadastroDM;
...
procedure TLavacaoData.LavacaoServicoInsere();
var
  Servico: TServico;
begin
  // Exibe o formulário para o usuário selecionar
```

Código para a unit LavacaoDM II

```
CadastroData.ServicoCadastroShow();
// Usando CadastroData.ServicoZTable para os próximos comandos
with CadastroData.ServicoZTable do begin
  // Pega os dados do registro atual do DataSet
  Servico.id_servico := FieldByName('id_servico').AsString;
  Servico.den_servico := FieldByName('den_servico').AsString;
  Servico.valor := FieldByName('valor').AsString;
end;
// Usando LavacaoServicoZQuery para os próximos comandos
with LavacaoServicoZQuery do begin
  // Insere novo registro
  Insert();
  // Insere dados buscados
  FieldByName('id_servico').Value := Servico.id_servico;
  FieldByName('den_servico').Value := Servico.den_servico;
  FieldByName('valor').Value := Servico.valor;
  // Grava os dados
```

Código para a unit LavacaoDM III

```
    Post();
  end;
  // Atualiza o total da lavacão
  LavacaoAtualizaValor();
end;

procedure TLavacaoData.LavacaoAtualizaValor();
var
  id: integer;
  Valor: real;
begin
  // Obtém o ID da lavação atual
  id := LavacaoZQuery.FieldByName('id_lavacao').AsInteger;
  // Usando AtualizaZQuery para os próximos comandos
  with AtualizaZQuery do begin
    // Fecha o DataSet
    Close();
```

Código para a unit LavacaoDM IV

```
    // Limpa o SQL
    SQL.Clear();
    // Consulta para calcular o valor da lavação
    SQL.Add('SELECT SUM(valor) AS total');
    SQL.Add('FROM lavacao_servico');
    SQL.Add('WHERE id_lavacao = :id_lavacao');
    ParamByName('id_lavacao').Value := id;
    // Abre o DataSet (executa consulta)
    Open();
    // Obtém o valor
    Valor := FieldByName('total').AsFloat;
    // Fecha o DataSet
    Close();
  end;
  // Usando LavacaoZQuery para os próximos comandos
  with LavacaoZQuery do begin
    // Coloca o DataSet em modo de edição
```

Código para a unit LavacaoDM V

```
    Edit();
    // Modifica o valor da lavação
    FieldByName('valor').Value := Valor;
    // Grava alterações
    Post();
  end;
end;
...
```

Excluir Serviço

```
unit LavacaoDM;
...
  TLavacaoData = class(TDataModule)
    ...
    procedure LavacaoServicoRemove();
...
implementation
...
procedure TLavacaoData.LavacaoServicoRemove();
begin
  // Apaga o registro selecionado
  LavacaoServicoZQuery.Delete();
  // Atualiza o valor total
  LavacaoAtualizaValor();
end;
...
```

Finalizar Lavação I

```
unit LavacaoDM;
...
  TLavacaoData = class(TDataModule)
    ...
    procedure LavacaoFinaliza();
...
implementation
...
procedure TLavacaoData.LavacaoFinaliza();
var
  id: integer;
begin
  // Usando LavacaoZQuery para os próximos comandos
  with LavacaoZQuery do begin
```

Finalizar Lavação II

```
    // Coloca o DataSet em modo de edição
    Edit();
    // Modifica dados
    FieldByName('data_saida').AsDateTime := Date();
    FieldByName('hora_saida').AsString :=  TimeToStr(Time());
    // Grava alterações
    Post();
  end;
end;
```

Código para a unit LavacaoCadForm

- Após a codificação do módulo de dados basta chamar as sub-rotinas apropriadas nos botões:
  Inserir Serviço *LavacaoData.LavacaoServicoInsere()*
  Excluir Serviço *LavacaoData.LavacaoServicoRemove()*
  Finalizar Lavação *LavacaoData.LavacaoFinaliza()*

Cadastro de Usuários I

- Para o cadastro de usuários precisamos fazer uma rotina de alterar senha.
- *unit UsuarioDM*:

```
unit UsuarioDM;
...
implementation

uses UsuarioCadFM, Dialogs, md5;
...
procedure TUsuarioData.UsuarioAlterarSenha;
var
  Senha, SenhaConfirma: string;
begin
  // Pega a senha e a confirmação com o usuario
  if (InputQuery('Alteração de Senha', 'Informe a nova senha',
 ↪   Senha)) and
    (InputQuery('Alteração de Senha', 'Repita a senha',
 ↪   SenhaConfirma)) then begin
```

Cadastro de Usuários II

```
    // Verifica se as senha confirmam
    if (Senha <> SenhaConfirma) then begin
      ShowMessage('A confirmação de senha falhou');
    end else begin
      // Criptografa a senha com MD5
      Senha := MD5Print(MD5String(Senha));
      // Grava as alterações
      with UsuarioZTable do begin
        Edit();
        FieldByName('senha_usuario').Value := Senha;
        Post();
      end;
    end;
  end;
end;
...
```

- *unit UsuarioCadFM*:

Cadastro de Usuários III

```
unit UsuarioCadFM;
...
  TUsuarioCadForm = class(TForm)
    ...
    procedure AlterarSenhaBitBtnClick(Sender: TObject);
...
implementation

uses UsuarioDM;
...
procedure TUsuarioCadForm.AlterarSenhaBitBtnClick(Sender: TObject);
begin
  UsuarioData.UsuarioAlterarSenha();
end;
```

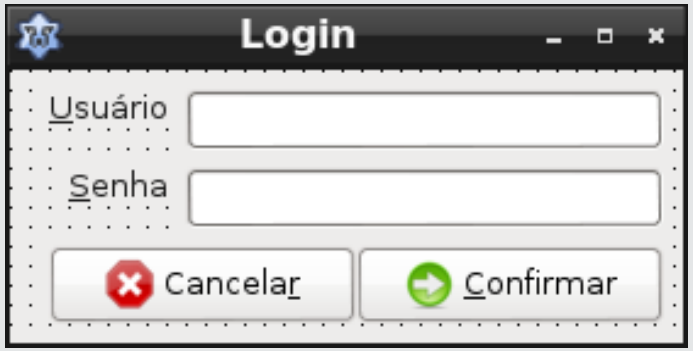
Controle de Login

- A última etapa do sistema será desenvolver um controle de login, permitindo o acesso ao sistema somente com um usuário e senha;
- No módulo de dados *UsuarioData*, vamos acrescentar um *ZQuery* com as seguintes propriedades:

  `Name = LoginZQuery`

  `DataSource = ConexaoData.PrincipalZConnection`
- Também vamos criam um formulário de login que deve ser o primeiro do projeto.

Formulário LoginForm I




```
object LoginForm: TLoginForm
  BorderStyle = bsSingle
  Caption = 'Login'
  Position = poScreenCenter
  object Label1: TLabel
    Caption = '&Usuário'
    FocusControl = UsuarioEdit
  end
```

Formulário LoginForm II

```
object UsuarioEdit: TEdit
end
object Label2: TLabel
  Caption = '&Senha'
  FocusControl = SenhaEdit
end
object SenhaEdit: TEdit
  PasswordChar = '*'
end
object OKBitBtn: TBitBtn
  Kind = bkOK
  Caption = '&Confirmar'
end
object CancelarBitBtn: TBitBtn
  Kind = bkCancel
  Caption = 'Cancela&r'
```

Formulário LoginForm III

```
  end
end
```

Código para a Unit UsuarioDM I

```
unit UsuarioDM;
...
  TUsuarioData = class(TDataModule)
    ...
    procedure Login(usuario, senha: string);
  ...
implementation

uses UsuarioCadFM, Dialogs, md5, PrincipalFM, LoginFM;
...
procedure TUsuarioData.Login(Usuario, Senha: string);
var
  TipoUsuario: string;
begin
  // Criptografa a senha com MD5
  Senha := MD5Print(MD5String(Senha));
```

Código para a Unit UsuarioDM II

```
// Grava as alterações
with LoginZQuery do begin
  Close();
  SQL.Clear;
  // Busca pelo usuário e senha recebidos
  SQL.Add('SELECT * FROM usuario');
  SQL.Add('WHERE login_usuario = :usuario');
  SQL.Add('  AND senha_usuario = :senha');
  ParamByName('usuario').AsString := Usuario;
  ParamByName('senha').AsString := Senha;
  Open();
  // Verifica se o usuario foi encontrado
  if (RecordCount > 0) then begin
    // Obtém o tipo de usuário
    TipoUsuario := FieldByName('tipo').AsString;
    // Verifica se o usuário é administrador
    if (TipoUsuario = 'A') then begin
```

Código para a Unit UsuarioDM III

```
        // Somente o administrador tem acesso ao cadastro de
↪   usuários
        PrincipalForm.UsuariosBitBtn.Enabled := True;
      end else begin
        PrincipalForm.UsuariosBitBtn.Enabled := False;
      end;
      // Abre Form principal
      PrincipalForm.Show();
      LoginForm.Hide();
    end else begin
      ShowMessage('Usuário ou senha inválidos!');
    end;
  end;
end;
...
```

Códigos para LoginForm e PrincipalForm

LoginForm

```
procedure TLoginForm.OKBitBtnClick(Sender: TObject);
begin
  UsuarioData.Login(UsuarioEdit.Text, SenhaEdit.Text);
end;
```

PrincipalForm

```
procedure TPrincipalForm.FormClose(Sender: TObject; var
↪  CloseAction: TCloseAction);
begin
  Application.Terminate();
end;
```

Referências

- Lazarus Database Tutorial. Lazarus Wiki. Disponível em `http://wiki.lazarus.freepascal.org/Lazarus_Database_Tutorial`;
- Zeos Quick Start Guide. Disponível em `http://zeos.firmos.at/kb.php?mode=article&k=6`.