Projeto GetCep - 1.0

Projeto para consulta de endereço com base em CEP fornecido.

Autor	Versão	Modificação
Ronaldo Filho	1.0	Criação da documentação

1. Descrição

O projeto trata da criação de aplicação para consumo de APIs responsáveis por fornecer dados de endereço com base em CEP. O mesmo consumirá as APIs e retornará ao usuário padrão de dados para consumo.

2. Padrões (Server/Client)

2.1. Estrutura/organização de pastas

O projeto conta com o layout de pastas definido da seguinte forma:

- Controllers:
  - Responsável pelo armazenamento dos arquivos gerados para controle de acesso a dados, tudo que necessite acessar dados, seja em APIs externas ou em outros meios, deverá ser colocado nessa pasta;
- Models:
  - Pasta responsável por armazenar os arquivos para preenchimento e tratamento dos dados trazidos pelo controller, os mesmos também deverá ser usados para devolver os dados requisitante;
- Util:
  - Pasta responsável por armazenar arquivos usados para armazenagem de tipos, processos e funções a serem usados no tratamento dos dados;
- View (Apenas Client):
  - Pasta responsável por guardar todos os arquivos destinados a apresentação dos dados ao usuário.

2.2. Nomenclatura

2.2.1. Variáveis e constantes

Todas as variáveis contam com a especificidade do tipo e local de declaração, dessa forma tomaremos o modelo a seguir:

- Para variáveis globais será usada a letra **F** início da declaração da mesma Ex.:
  - FsVariavel;

- para a declaração de constantes, usaremos o indicador **_k** ao início da declaração das mesmas Ex.:
    - _ksConstante;

Da mesma forma usaremos os tipos das mesmas como segundo indicador:

- S - para string;
- N - para inteiros, reais;
- D - para datas;
- B - para boolean.

Para as variáveis instanciadas de objetos usaremos o indicador **O**.

- Exemplos:
    - FsVariavel: string - Variável global do tipo string;
    - sVariavel: string - Variável local do tipo string;
    - AsParametro: string - Parametro do tipo string;
    - _ksConstante = “” - Constante do tipo string;

Assim teremos maior rapidez na identificação das variáveis, sabendo suas características apenas em ler seus nomes.

2.2.2. Métodos

As procedures e functions deverão expressar a sua ação em seus nomes, Ex.:

- Métodos para retorno de dados:
    - GetValor;
    - ReturnValor;
- Métodos para tratamento de dados:
    - ModificaDado;
    - TrataDado;
- Métodos para validação:
    - ValidaDado;
    - TestaDado;

2.2.3. Espaços

Ao iniciar um bloco de código oriundo do início de escopo de método/classe, ou de estrutura lógica/repetição, deverá realizado o espaçamento duplo:

```
procedure TCustomEdit.Change;
begin
  inherited Changed;
  if Assigned(FOnChange) then FOnChange(Self);
end;
```

Caso o trecho contenha estruturas de repetição/lógicas:

```
procedure TForm1.edtCepChange(Sender: TObject);
var
  bError: Boolean;
begin
  if (Length(edtCep.Text) = 8) then
  begin
    MontDataOnForm(TCepController.New
                                  .SetCep(edtcep.Text)
                                  .SearchCep(bError)
                                  .GetModel);
    if bError then
    begin
      MessageDlg('CEP não localizado', TMsgDlgType.mtError,
mbOKCancel,0);
      bError := false;
    end;
  end
end;
```