RMSRetail®

MANUAL DO USUÁRIO

CONCEITOS E FUNCIONALIDADES

MOBILE - MANUAL DO DESENVOLVEDOR

RMS

Uma empresa TOTVS

A RMS Software é a maior fornecedora nacional de software de gestão corporativa para o mercado de comércio e varejo.

Este documento contém informações conceituais, técnicas e telas do produto que são confidenciais, podendo ser utilizadas somente pelos clientes RMS no projeto de utilização do RMS/Retail.

RMS Software S.A. - Uma empresa TOTVS.
11 2699-0008 – www.rms.com.br

ÍNDICE

Mobile – Manual do Desenvolvedor

Conceito

- ⇨ Maior legibilidade de código.
- ⇨ Maior performance de execução.
- ⇨ Maior produtividade.
- ⇨ Menor tempo de manutenção.
- ⇨ Menor ocorrências de problemas conhecidos.

O padrão de codificação então, sob a premissa acima, foi desenvolvido dentro das qualificações exigidas para uma codificação. Foram observados os tipos de codificação que ocasionavam maiores problemas de inteligibilidade, de visualização, de eficiência no desenvolvimento e de manutenção.

O RMSMobile é o módulo do RMS que engloba as aplicações destinadas a equipamentos portáteis com Windows (Coletores de dados).

Descrição do processo

Ambiente de Desenvolvimento

O ambiente de desenvolvimento RMSMobile deve possuir os seguintes softwares e suas respectivas configurações:

- ⇨ Microsoft Visual Studio
- ⇨ Microsoft Data Access (ADO) 2.8
- ⇨ Oracle Client (módulo de acesso ao banco de dados)
- ⇨ PL/SQL Developer (execução de tarefas no BD, criação de procedures, etc...)
- ⇨ Microsoft Framework 1.1 (utilizado pelo web service)
- ⇨ Ferramentas do OpenNetCF (utilizado pelas aplicações Móbile)
- ⇨ IIS 5.0 (ou superior) (utilizado pelo Web Service)
- ⇨ ComponentOne (ferramentas específicas para desenvolvimento)

Estes são os softwares principais, projetos especiais podem possuir outros não listados acima.

O ambiente de desenvolvimento do RMSMobile se divide em 3 partes: Interface (VB.NET), Web Service (C#) e Atualizações de registros em base de dados (PL SQL-Oracle).

A Interface Mobile não possui comunicação direta com o banco de dados. Para existir a interação dos dados é utilizado um intermediário: Web Service.

O Web Service é responsável pela integração do RMSMobile com o Banco de Dados. Nele se encontra toda a seleção dos dados necessários para utilização do sistema. O RMS identifica as atividades realizadas na Aplicação, conforme as atualizações no Banco de dados.

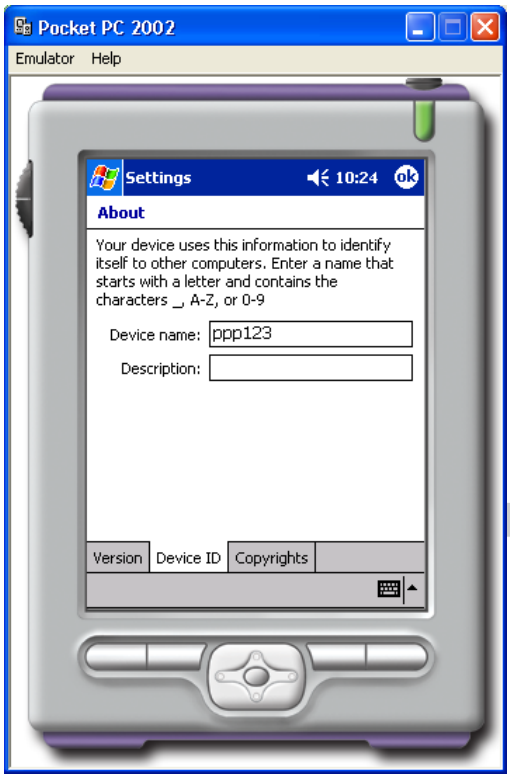
Quaisquer atualizações em banco de dados são feitas através de Packages, ou seja, são nelas que estão contidas todas partes lógicas de atualizações em base de dados.

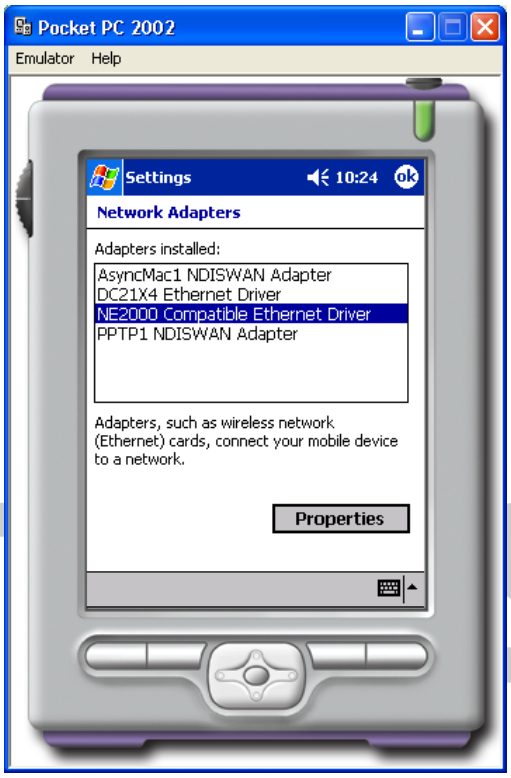
Configuração do Emulador do Visual Studio

Configurações de rede

O emulador funciona como uma estação independente. É necessário especificar um IP fixo sendo que o mesmo deve ser liberado na rede para acessar o servidor de WebService (geralmente a própria máquina de desenvolvimento).

As propriedades a serem alteradas são: nome, ip, e modo de conexão.





Componentes Utilizados

Para que o Software funcione no equipamento, são necessárias as instalações de dois componentes:

⇨ OpenNetCF
⇨ NetCF

Estes componentes são fornecidos de acordo com o modelo e o Windows instalado no equipamento. Durante o desenvolvimento de aplicações, caso o emulador utilizado não possua estes componentes instalados, o Visual Studio os instala automaticamente.

Projetos Genéricos

RMSFunções

Responsável pelas principais funções utilizadas na maior parte das aplicações:

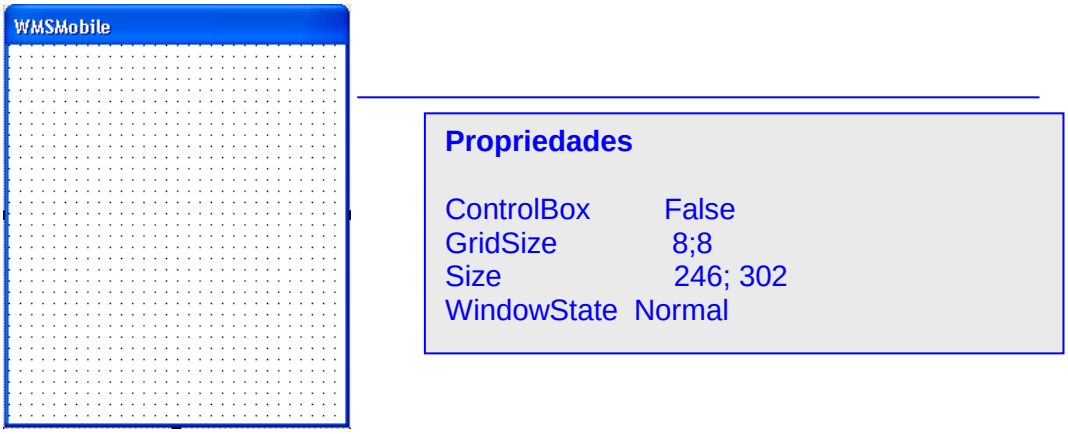
⇨ Identificação e formatação do código de barras lido pelo Equipamento.
⇨ Formatação de Datas.
⇨ Processo de tradução.
⇨ Tratamento das mensagens de Erro.

Arquivo de Configuração (*.config)

Cada projeto contém um arquivo de configuração. Este arquivo deverá conter a propriedade “Build Action = Content”. Esta propriedade identifica que o arquivo não deve ser compilado dentro da aplicação.

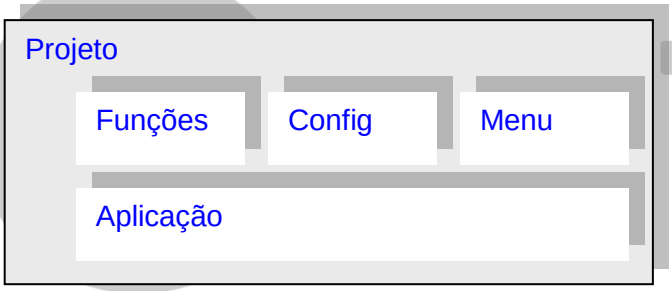
Desenho de interfaces

As interfaces devem seguir o mesmo padrão de desenho, permitindo ao usuário uma identificação visual com o sistema.



Padrões de Codificação

Os tópicos abaixo abrangem os pontos principais.



Estrutura da Interface Mobile

As aplicações em Mobile serão divididas em duas partes: Formulário de Inicialização e Interface. Esta divisão permite maior reaproveitamento de código, uma vez que possibilita o compartilhamento das instruções das Funções Genéricas.

Abaixo está o modo de implementação e as considerações que devem ser feitas para cada parte.

- Formulário de inicialização (Forms)
  É um formulário dentro do projeto responsável pela integração com as funções genéricas e com o WebService.
  Cada programa deve ter seu próprio formulário, neste caso utiliza-se o que já existe e cria-se um componente apenas com as necessidades específicas do programa sendo desenvolvido.
  Cada projeto deve ter um formulário que contenha as funções de acesso. Neste caso o módulo de funções deve ser referenciado no mesmo.

- Interface
  Cada aplicação poderá conter mais de uma interface, e mais de uma chamada ao WebService, sendo que cada interface ativada e cada chamada ao WebService deverão ser acionadas através do formulário de inicialização (Forms).

Arquivo de Configuração do Programa

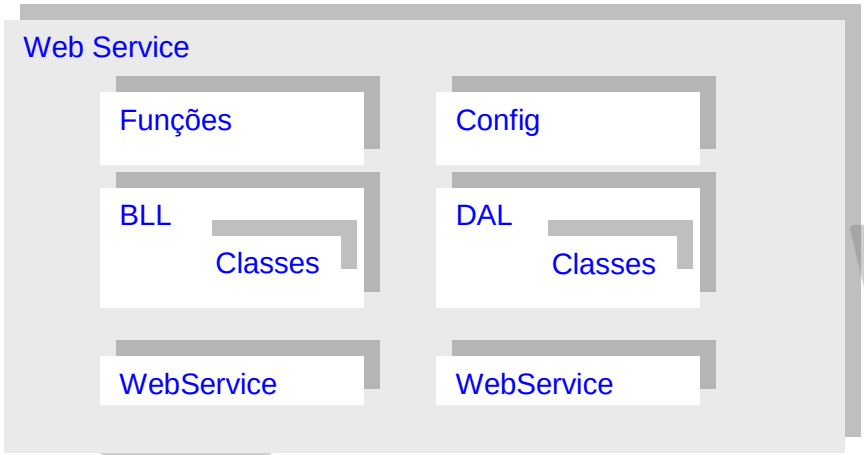
Responsável pelas informações de integração da Interface com o Web Service e com o RMS.

- ⇨ Endereço do WebService;
- ⇨ Contexto;
- ⇨ IP do servidor do serviço RMS;
- ⇨ Idioma utilizado;
- ⇨ Identificação do Equipamento;

Estrutura do Web Service

Atualmente o Web Service englobam 2 dois projetos (WMSMobile e StoreMobile), e se divide em 3 camadas:

- ⇨ Páginas para chamadas de Aplicações externas.
- ⇨ BLL – Classes que processa informações sem chamadas a base de dados.
- ⇨ DAL – Classes responsáveis pelas chamadas ao banco de dados.



Funções (common.cs)

Esta classe possui as funções diversas utilizadas no projeto.

- ⇨ Montagem da string de conexão com o banco de dados.
- ⇨ Gerenciamento de criptografia de conexão.
- ⇨ Funções de recuperação de data, endereços e parâmetros.
- ⇨ Funções para registros de Log.
- ⇨ Controle de Versão.

Arquivo de Configuração do WebService

Responsável pelas informações de integração do WebService com a Base de Dados.

- ⇨ String de Conexão com a base de dados;
- ⇨ Idioma para tradução;
- ⇨ Parâmetros para registro de Log;
- ⇨ Caminho de identificação do arquivo de Login.

Estrutura do código fonte

Manter o código de uma rotina bem estruturado e limpo, facilita a leitura, entendimento e manutenção dos códigos, além de permitir melhor gerenciamento dos erros que possam ocorrer. Existem algumas técnicas que devem ser adotadas para se obter um bom resultado.

Indentação

Manter o código corretamente indentado, dá ao código melhor visualização de cada bloco de comandos, permitindo melhor entendimento das decisões (*Ifs*) e *Loops*. Para isso utilizar a tecla <TAB> (equivalente a 4 posições) para cada nível.

With... End With

Esta estrutura, além de dar maior clareza, aumenta a velocidade de execução do código.

Tratamento de Erro

Ao se implementar uma rotina de tratamento de erros bem estruturada, tem-se a garantia que a aplicação não cairá, que o usuário saberá o que está acontecendo e que nenhum componente desnecessário ficará preso na memória. Vejamos os seguintes passos:

ATENÇÃO: Colocar na primeira linha de código, após aos comentários e antes das declarações de variáveis, a instrução:

Try..... Catch (ex As Exception) ☞ Utilizar "**ex**" como descrição do erro Ocorrido.

Declaração de Variáveis

Declarar as variáveis que se vai utilizar nos programas é uma boa prática, pois possibilita identificar qual o seu escopo de visibilidade e a sua função dentro do programa.

As variáveis devem ser declaradas no **início** do módulo (*General Declarations*) ou rotina em que serão empregadas, nunca no meio do código. No caso de variáveis públicas, devem ser declaradas na classe pública (Forms) para simplificar a localização.

Todas as variáveis não utilizadas devem ser removidas dos programas, para facilitar o entendimento do código e diminuir a utilização de memória.

Utilizar nomes claros para as variáveis, nomes que identifiquem a sua função dentro do código. Quando isto não for possível, incluir comentários identificando a utilização que ela terá.

Para variáveis utilizadas em loops utilizar preferencialmente a letra "i", ou pode-se utilizar nomes como *i,x,y,z,* etc., normalmente utilizadas em loops, desde que estejam devidamente comentadas.

O padrão de nomenclatura serve para facilitar a identificação do tipo e/ou do escopo de algum elemento, permitindo melhor leitura e compreensão do código.

A Microsoft fornece, na documentação de suas ferramentas, alguns padrões de nomenclatura que podemos usar.
Normalmente utiliza-se uma sigla de três letras no início do nome para identificar o seu tipo e para identificar o escopo utiliza-se uma letra antes do tipo. Logo abaixo estão as abreviações para cada tipo de elemento e escopo.

Prefixo de escopo de declaração da variável

g	**Variável global,**
m	**Variável local, definida dentro do form (private)**

Prefixo para o tipo de dados da variável

bln	**Boolean**
dec	**Decimal**
dtm	**Date (Time)**
int	**Integer**
arr	**Array**
str	**String**
dst	**DataSet**
drw	**DataRow**
lst	**ArrayList**

Nomenclatura de Componentes

A nomenclatura dos componentes deve seguir o mesmo padrão das variáveis, abaixo estão os prefixos para os componentes mais utilizados.

Para os componentes não listados, deve-se criar um prefixo com dois ou três caracteres que o identifique unicamente. Pode-se utilizar mais letras se for necessário

No caso de componentes de terceiros, utilizar a abreviação empregada pelo fabricante. Caso esta abreviação entre em conflito com alguma já existente, deve-se adicionar algumas letras para melhor distinção.

Sempre que houver a criação de uma nova abreviação, o fato deve ser informado ao departamento de O&M para correção deste manual.

Componentes de Aplicação

frm	**Formulários (Forms).**
cs	**Módulos Classe.**

Componentes de Tela

lbl	**Label**
lst	**List box**
mnu	**Menu**
chk	**Check box**
cbo	**Combo box, drop-down list box**
btn	**button**
dst	**DataSet**
dtw	**DataRow**
txt	**Text box**
pct	**PictureBox**

Componentes de Web Service

ws	**WebService**

Declaração de Rotinas

Uma rotina pode ser uma sub rotina ou uma função.

Não é necessário especificar o escopo e o tipo de dados de retorno da função, a preocupação deve ser com a clareza do nome empregado, sempre considerando que a primeira palavra deve ser um verbo, como por exemplo *FormatarData*. Caso o nome seja muito extenso (mais de 30 caracteres) pode-se utilizar abreviações, porém, estas abreviações também devem ser padronizadas, não se pode dizer que a abreviação de *Formatar* em uma função é *Fmt* e na outra *Frmt* por exemplo. Todas as rotinas que tenham comportamento semelhante, devem possuir no mínimo, um prefixo igual. Isto serve para ajudar a identificar a funcionalidade da rotina no código onde ela é chamada.

Comentários

A utilidade dos comentários é facilitar o entendimento de códigos complexo. Pode ser utilizado de duas formas:

Bloco de identificação de módulos, rotinas e funções.
Identifica o propósito do código, parâmetros de entrada, parâmetros de saída, objetos disponíveis, etc.

EXEMPLO:

```
'****************************************************************
' Propósito : Identifica a funcionalidade do código
' Autor     : Programador que criou a rotina
' Criação   : Data de criação
' Entrada   : Identifica o tipo de dados e propósito dos parâmetros
' Retorno   : Identifica os valores retornados pela rotina
'****************************************************************
```

Podem-se incluir, também, trechos de código para facilitar o entendimento da utilização.

- Comentário em Linha:
  Comentário para o entendimento de uma ou várias linhas de código dentro da rotina.

  Ex.: Me.carregaForms = True 'Identifica que os formulários estão sendo carregados.

Manutenção de datas e valores

Para que os programas funcionem corretamente, devemos levar em consideração os parâmetros de configuração regional do sistema operacional da estação.

Outro fator importante é o formato dos campos data das tabelas mais antigas. Estes campos estão formatados com seis e sete posições. Os campos com seis posições estão no formato DDMMYY e são, normalmente, apenas atributos das tabelas. Os campos com sete posições estão no formato YYYMMDD e são usados para composição de chaves e índices.

O formato mais utilizado atualmente é 1YYYYMMDD. Para facilitar a conversão basta passar a data para a função *"GetRMSDate"* presente no Web Service, ou a função `formataDataRMS` presente no programa WMSMobile e no StoreMobile.

Consistência na entrada de dados

Na implementação da consistência devemos utilizar o evento *LostFocus* dos campos Texts.

No momento de confirmação da tela deve-se chamar todos os procedimentos de validação existes para o campo.

Trabalhando com o Menu (Menu WMSMobile e Menu StoreMobile)

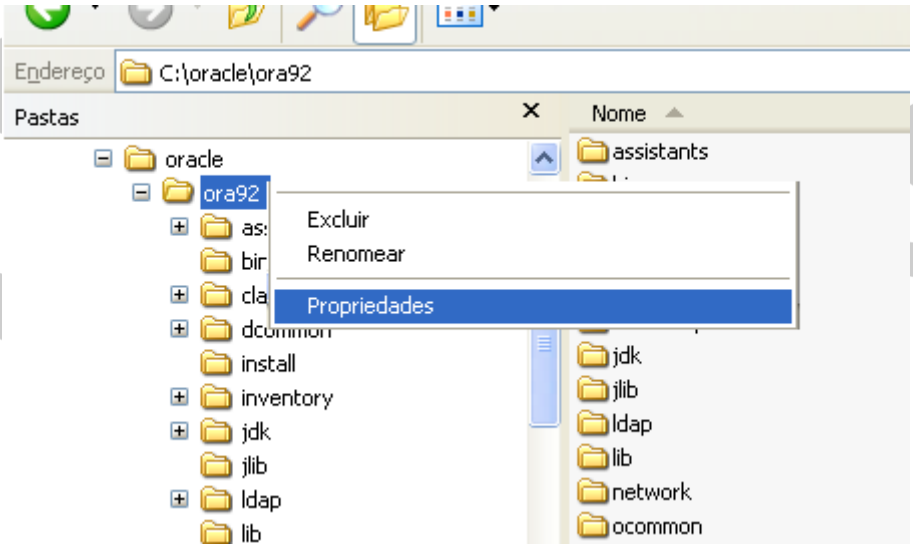
O Menu é o módulo que controla o ambiente de execução do sistema. Toda e qualquer aplicação contida no sistema deve ser iniciado a partir dele.

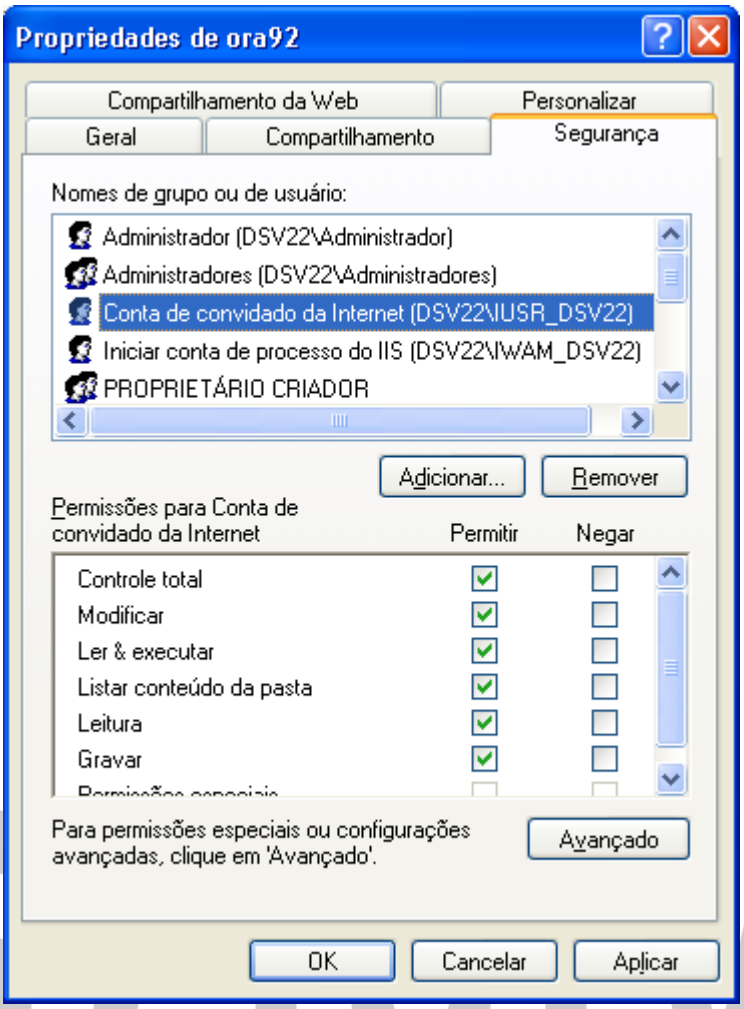
Acesso a dados

A aplicação do WebService utiliza o Oracle Client para acessar o banco de dados.

Atualmente trabalhamos com o WebService utilizando o usuário anônimo. Geralmente este usuário anônimo é IUSR_nome_do_computador.

Para que o WebService se comunique com a base de dados, é necessário que a pasta do Oracle tenha permissão para o usuário anônimo registrado na segurança do diretório do WebService.





Estrutura das funções de acesso

Todas as aplicações do RMSMobile possuem a comunicação com o banco de dados através do Web Service.

No WebService encontramos apenas seleção de registros e chamadas a procedures do banco de dados, ou seja, não deve ser colocado rotinas de Atualizações, Inclusões e Exclusões de registros. Estas rotinas devem ser administradas através de packages.

Seleção de registros

A obtenção das tabelas é muito simples, basta utilizar a função *"common.ExecuteDataSet(SQL)"*. Após ter referenciado o namespace do projeto:

```
using RMS.RMSMobile.wsRMSMobile;
```

Programa Padrão RMS

Premissas

⇨ Navegação
O usuário poderá navegar por todo o programa (campos) e caso seja informado algum dado para algum campo o mesmo deve ser validado, caso não informe e o campo seja obrigatório, deve ser solicitado a informação no momento de gravação (F4);

⇨ Tratamento de Permissão de Acesso (VITUTRAN – Programa: RMSMOBIL)
Verificação se a tecla está liberada

⇨ Código Fonte Limpo
Definir variáveis no início da rotina;
Variáveis com nome lógico;

Programa Exemplo

No exemplo abaixo, vamos demonstrar a padronização em um programa que está em produção.

Programa CAD_LFIS – Cadastro de Localização Física, onde os 3 primeiros caracteres indicam o módulo (Cadastro = CAD), o restante identifica o tipo, sempre com 4 caracteres.

Este programa tem como objetivo cadastrar as gôndolas da Loja.

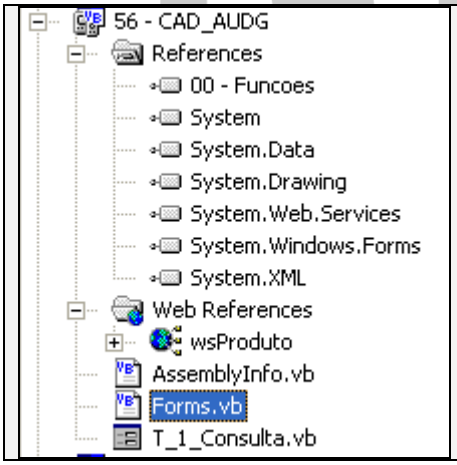
Estrutura da Tabela – AA3CLOFI.

```
LOF_LOJA          NUMBER(8) Y
LOF_GONDOLA       NUMBER(3) Y
LOF_COD_ITEM      NUMBER(3) Y
```

Onde LOF_LOJA é a identificação da loja, o `LOF_GONDOLA` é o número da gôndola e o `LOF_COD_ITEM` é o código do item.

Estrutura do Programa

O programa deve possuir a seguinte estrutura:



Referência: Possui a referência dos objetos utilizados na aplicação. Nele deveremos adicionar referência às Funções, por exemplo.

Web References: Nesta propriedade devemos referenciar o Web Service utilizado na aplicação.

Forms: Classe pública de inicialização do projeto.

Telas da Aplicação: Cada aplicação poderá ter mais de uma tela. O nome das mesmas poderá ser aplicado na sequência em que ocorre o acesso durante a navegação.

Ex: T_X_YYYYYY

Onde X é o número da tela e YYYYYY nome da tela.

Classe: Forms

O programa principal (WMSMobile) inicia o programa a partir desta classe, o conteúdo altera de acordo com a aplicação.

Exemplo:

```
Imports System.Web.Services
Imports RMSMobile_Funcoes.Forms          'Importa a referencia a Funções
Imports RMS.RMSMobile.CAD.AUDG.wsProduto'Importa ao WebService utilizado na
Aplicação
```

```
Public Class Forms
    Inherits RMSMobile_Funcoes.Forms    'Importa as instruções do Funções para
dentro
                                        da classe
    Shared instance As Forms

    Dim f_T_1_Exemplo As T_1_Exemplo         'declarar as telas contidas na
aplicação
    .....                                   'Caso necessário declarar também
                                             variáveis globais
    Public _wsProduto As wsProduto.wsProduto 'declarar a variável de acesso ao
Web
                                             Service

    'procedimento público para acesso interno das funções
    Public Shared Function getInstance() As Forms
           getInstance = RMS.RMSMobile.CAD.AUDG.Forms.getInstance(Nothing,
           Nothing, 0, Nothing)
    End Function

    'procedimento público para inicializar a aplicação
   Public Shared Function getInstance(ByVal form_back As
   System.Windows.Forms.Form, ByVal SystemUser As String, ByVal Store As
   Decimal, ByVal comprador As String) As Forms
        If (instance Is Nothing And form_back Is Nothing) Then
           MessageBox.Show("ERRO NO PROGRAMA (instance==null &&
           form_back==null) chamando a classe RMS.RMSMobile.CAD.AUDG.Forms")
            getInstance = Nothing

            Exit Function
        End If
        If (instance Is Nothing) Then
            instance = New RMS.RMSMobile.CAD.AUDG.Forms
            instance.Forms(SystemUser, Store, comprador)
        End If
        If Not (instance.iniciado) Then
            instance.inicio(form_back)
        End If
        getInstance = instance
    End Function

    'ao carregar o classe valida primeiramente se existem os objetos utilizados
na
     aplicação
    Private Shadows Sub Forms(ByVal SystemUser As String, ByVal Store As
Decimal,
    ByVal comprador As String)
        MyBase.setObjetos("'AA3CITEM','AA3CCEAN','AA2CTABE'")
        MyBase.Forms(SystemUser, Store, comprador)
    End Sub

    'procedimento para carregar as telas e a chamada ao WebService da aplicação
    Protected Overrides Function iniciaForms() As Boolean
        iniciaForms = True           'variável auxiliar no carregamento do
módulo

       'carregar e traduzir as telas da aplicação
        Me.f_T_1_Exemplo = MyBase.TraduzirForm(New T_1_Exemplo)
```

```
        'carregar o WebService utilizado
        _wsProduto = New wsProduto.wsProduto
        _wsProduto.Url = Me.wsPath + "wsProduto.asmx"


        'Carregar a tela principal
        Me.f_T_1_Exemplo.Show()
    End Function

    'Descarregar as telas ao encerrar a aplicação
    Protected Overrides Sub fimForms()
        ' libera recursos de todos os Forms do módulo
        Me.f_T_1_Exemplo.Dispose()
        instance = Nothing
    End Sub
End Class
```

DataSet e Cursores

Para trabalhar com tabelas nas Aplicações é utilizado o objeto DataSet.

Sempre que é feita uma chamada ao WebService, alteramos o cursor da aplicação para o modo em espera.

Exemplo:

```
      Dim ds As DataSet

       Cursor.Current = Cursors.WaitCursor
       ds = Forms.getInstance()._wsProduto.detProdutos(codigo, Store)
       Cursor.Current = Cursors.Default
```

Para utilizar os objetos DataSet e Cursores é necessário importar a classe das mesmas:

```
Imports System.Data
imports System.Windows.Forms
```

Mensagens

Premissas

Utilizar rotina de tradução.

Exemplo:

```
      MsgBox(Forms.Traduzir("Produto inválido"))
```

Tradução

Um ponto muito importante no Sistema RMSMobile é a tradução de telas e mensagens.

A tradução dos componentes do form é automática pelo *RMSMobile_Funcoes*, por isso deve-se durante o desenvolvimento colocar no arquivo config do Programa o seguinte parâmetro:

```
<add key = "enviaPalavras" value = "S" />
```

Para que no banco de dados de tradução seja carregada a palavra a ser traduzida.

Em funções na rotina “Traduzir”, identifica se o parâmetro “enviaPalavras” está com o valor “S”, e chama o processo de gravação da palavra no WebService.

No WebService a gravação da palavra no banco de dados utiliza o parâmetro do Web.Config:

```
<add key = "connTrad" value = "Provider=MSDAORA.1;User ID=RMS_TRADUCAO;Data Source=DESENV;Password=RMS_TRADUCAO" />
```

Caso seja necessário concatenar uma mensagem e utilizar a função Traduzir nas strings.

Controle de Fontes

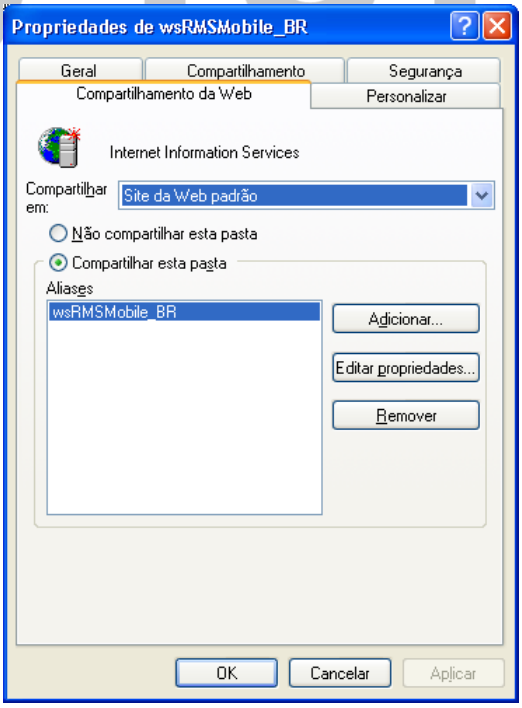
Sempre solicitar uma vez antes de alocar um fonte no RMSCatal, ou de preferência deletar o destino antes de solicitar e substituir o arquivo para que seja alterado.

Para efetuar um desenvolvimento do WebService é necessário que o mesmo esteja instalado na máquina de desenvolvimento.

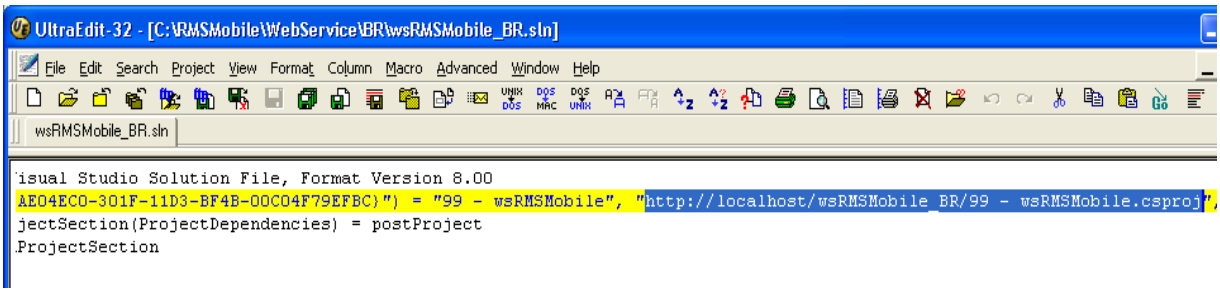
Ao instalar o os códigos fontes em uma máquina de desenvolvimento, é interessante a consulta também no manual de instalação dos programas.

Web Service

Ao baixar um arquivo com extensão .SLN de um projeto WebService para implementar na máquina, verificar se a pasta do projeto está com o Compartilhamento Web definido:



Abrir o arquivo em modo texto e averiguar se o endereço do projeto está correto:



Projetos RMSMobile

Ao baixar um projeto do RMSMobile deve-se baixar todos os módulos do mesmo, pois o menu do programa guarda todas as referências apontando diretamente para o projeto.