

Essa sub-rotina é um código único e inteiro, particione essa sub-rotina em três sub-rotinas separadas:

```
Sub subrotina1()
    'Altera a cor de fundo das células selecionadas
    Range("A1").CurrentRegion.Select
    Selection.Interior.Color = vbYellow
End Sub

Sub subrotina2()
    'Aplica bordas a planilha
    Dim i As Byte
    For i = 7 To 12
        With Selection.Borders(i)
            .Weight = xlThin
            .LineStyle = xlContinuous
            .Color = vbRed
        End With
    Next
End Sub

Sub subrotina3()
    'Formata as células da planilha
    With Range("A1:C1")
        .Font.Color = vbRed
        .Merge
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlCenter
    End With

    For linha = 2 To 7
        Cells(linha, 2) = Format(Cells(linha, 2), "R$ #,###.00")
    Next
    Range("A1:C7").EntireColumn.AutoFit
End Sub
```

Observe que cada sub-rotina agora possui um código individual, vamos uni-las novamente através da instrução “Call” que executa uma “chamada” ao código fora da sub-rotina.

```
Sub subrotina1()
    'Altera a cor de fundo das células selecionadas
    Range("A1").CurrentRegion.Select
    Selection.Interior.Color = vbYellow

    Call subrotina2 'Chama a sub-rotina2

    Call subrotina3 'Chama a sub-rotina3
End Sub
```

Nesse caso, embora não ocorra à presença do código das outras sub-rotinas diretamente, elas estão associadas por meio de **referências**, que não altera a maneira como o código é executado mas permite **distribuir** o uso do código de maneira inteligente, a esse conceito chamamos de “**modularização**”.

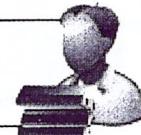
A vantagem desse conceito é que o programador cria o código uma única vez e a chama em qualquer ponto do código.





VBA

MICROSOFT EXCEL 2013 COM VBA

Apostila do Aluno

Passando Valores a Sub-Rotinas (ByVal):

Outro recurso interessante é a **passagem de valores** a uma sub-rotina onde além de chamar uma sub-rotina é passado diretamente um valor durante sua execução, veja um exemplo abaixo:

```
Sub selecionar()
    Range("A1").CurrentRegion.Select

    'Chama a sub-rotina formatar e passa
    'um valor de parâmetro
    Call formatar(4)
End Sub

Sub formatar(ByVal cor As Integer)
    Selection.Interior.ColorIndex = cor
End Sub
```

Observe que ao chamar a função formatar foi passado um **valor de parâmetro**, esse valor informa internamente a instrução ColorIndex um valor que altera a cor de fundo das células selecionadas, nesse caso o valor é obrigatório, caso contrário ocorrerá um erro.

O parâmetro é um tipo de variável declarada onde apresenta os mesmos aspectos de uma variável convencional.

Passando Valores opcionais a Sub-Rotinas (Optional):

Você poderá refinar a passagem de valores a sub-rotina usando o atributo **Optional** que torna a passagem de parâmetro opcional:

```
Sub selecionar()
    Range("A1").CurrentRegion.Select
    Call formatar
End Sub

Sub formatar(Optional cor As Integer)
    If cor > 0 Then
        Selection.Interior.ColorIndex = cor
    Else
        Selection.Interior.ColorIndex = 6
    End If
End Sub
```

Caso você não passe nenhum valor para a sub-rotina será executada a condição falsa do teste lógico e um preenchimento padrão de fundo será utilizado.





VBA

MICROSOFT EXCEL 2013 COM VBA

Apostila do Aluno

Construindo funções (Métodos)

Uma função apresenta um uso semelhante à sub-rotina, existindo algumas diferenças usuais, enquanto a sub-rotina executa um grupo distinto de instruções quando chamada, uma função executa um grupo de instruções de maneira integrada a uma sub-rotina apresentando retorno de valores após o processamento a sub-rotina a qual está integrada.

Funções são também mencionadas pelo termo **métodos**, sub-rotinas também, a diferença que na programação, funções são **métodos com retorno** e sub-rotinas são **métodos sem retorno ou procedimentos**.

Criando Funções:

No caso de uma função (método) ela apresenta o aspecto de retornar ao processamento valores que poderão ser utilizados dentro de uma sub-rotina. A função inicia sua escrita utilizando a instrução **Function** no lugar de **Sub** e **End Function** no lugar de **End Sub**.

Veja abaixo um exemplo de função:

```
Function somar()  
    Dim a, b As Integer  
    a = Range("A1")  
    b = Range("B1")  
    somar = a + b  
End Function
```

A função embora parecida com uma sub-rotina, ela apresenta um **retorno de valor** que é devido pela **própria função**.

Exemplo do uso de uma função:

```
Sub calcular()  
    'Exibe na célula o resultado da função  
    Range("C1") = somar  
End Sub
```

No caso de uso da função é semelhante à **atribuição de valores**, que quando existe um valor, repassa-o diretamente a uma variável ou instrução na sub-rotina.

Ao contrário de uma sub-rotina, não será chamada via instrução **Call**, mas é atribuída como parte integrante de um código.

Passando Valores a Funções (ByVal):

Assim como a sub-rotina, podemos atribuir valores a função quando usada para receber valores externos a serem processados.

Exemplo de uma função usando passagem de valores:

```
Function calculaDesconto(ByVal valor As Single, ByVal desconto As Integer) As Single  
    calculaDesconto = valor - valor * (desconto / 100)  
End Function
```

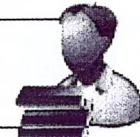
Repare que na função, assim como na sub-rotina, apresenta também variáveis declaradas para o parâmetro, a diferença da função para a sub-rotina é que o resultado do processamento é **atribuído a própria função**.





VBA

MICROSOFT EXCEL 2013 COM VBA

Apostila do Aluno

Outro detalhe a ser reparado, será a declaração de retorno da função, fora do parâmetro.

Exemplo da utilização da função:

```
Sub exexutaFuncao()
    Dim a As Single, b As Integer, resultado As Single
    a = 100
    b = 5
    resultado = calculaDesconto(a, b)

End Sub

Function calculaDesconto(ByVal valor As Single, ByVal desconto As Integer) As Single
    calculaDesconto = valor - valor * (desconto / 100)
End Function
```

Usando Referências públicas e privadas (Heranças de Objetos)

Na programação VBA existe o princípio da orientação objeto que permite criar e herdar objetos dentro do código. Mas o que significa isso?

Por exemplo, se você criar a sub-rotina abaixo:

```
Sub tabuada()
    Dim linha As Byte

    For linha = 1 To 10
        resultado = linha * 5
        Range("A" & linha) = linha
        Range("B" & linha) = "X"
        Range("C" & linha) = 5
        Range("D" & linha) = "="
        Range("E" & linha) = resultado
    Next
End Sub
```

Essa sub-rotina, embora não pareça, é uma sub-rotina pública, ou seja, qualquer código do VBA poderá enxergar e executar essa sub-rotina. Muitas vezes o código é acompanhado da declaração de acesso “Public”:

```
Public Sub tabuada()
    Dim linha As Byte

    For linha = 1 To 10
        resultado = linha * 5
        Range("A" & linha) = linha
        Range("B" & linha) = "X"
        Range("C" & linha) = 5
        Range("D" & linha) = "="
        Range("E" & linha) = resultado
    Next
End Sub
```

Se o código se encontrar dentro de um módulo a sub-rotina será executada normalmente via uso da instrução “Call”.

Caso a sub-rotina se encontre dentro de um objeto, por exemplo, associamos ao nome da sub-rotina o nome da planilha. Exemplo abaixo:

Call Plan1.Tabuada

Quando essa associação ocorre afirmamos que a sub-rotina é uma “Herança” do objeto **Worksheet** “Plan1”.



ADV – Curso de Informática

Telefone: (21) 2210-1180

Av. Treze de Maio, 23 – 8º andar – Centro – Rio de Janeiro - RJ

<http://www.cursoadv.com.br/>

Página 73



Variáveis também podem ser declaradas para uso público:

Public valor As Integer

Nos casos que não haja necessidade desse tipo de acesso tornaremos nosso código restrito a partir da declaração de acesso "Private". Veja o exemplo abaixo:

```
Private Sub tabuada()
    Dim linha As Byte

    For linha = 1 To 10
        resultado = linha * 5
        Range("A" & linha) = linha
        Range("B" & linha) = "X"
        Range("C" & linha) = 5
        Range("D" & linha) = "="
        Range("E" & linha) = resultado
    Next
End Sub
```

A declaração de acesso em um código também diz respeito à **segurança**, se um código usa métodos privados isso significa que somente o código poderá manipular seus próprios métodos usuário não poderão usá-los diretamente.

CAPÍTULO 8 – USANDO INTERFACES

Aprendemos que com o código VBA criamos programas eficientes e que através de códigos bem construídos podem nos proporcionar ótimos resultados.

O problema que o código aqui estudado é muito pouco amigável, ou seja não foi apresentada ainda uma forma de interação prática com o usuário.

Uma interface nada mais é que um meio visual usado por um programa, existem dois tipos de interfaces: as **janelas modais** e os **formulários**.

Janelas Modais

A janela modal é um tipo de janela utilizada por programas para exibir algum tipo de mensagem ou permitir alguma entrada de dados. Existem dois tipos de janelas modais usadas pelo VBA: a InputBox e MsgBox.

Uma Inputbox apresenta um meio de um usuário realizar entrada de valores para o código e a MsgBox é utilizada para exibir mensagens de aviso ao usuário.

Abaixo um exemplo de código usando uma Inputbox e uma MsgBox:

```
Sub idade()
    Dim ano, idade As Integer

    ano = InputBox("Digite o ano que você nasceu:")
    idade = Year(Date) - CInt(ano)

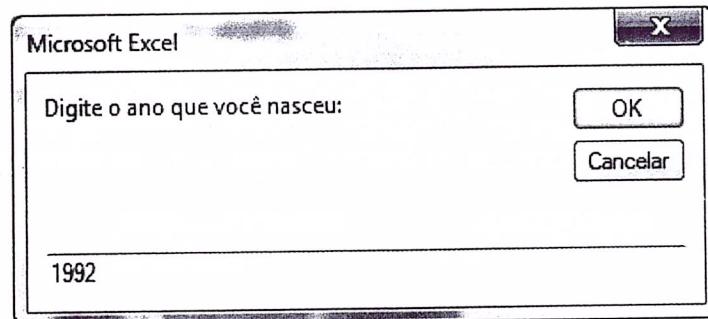
    MsgBox "Você tem " & idade & " anos!"

End Sub
```

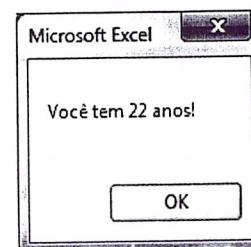




Será exibido inicialmente um caixa de entrada perguntando sobre o ano de nascimento:



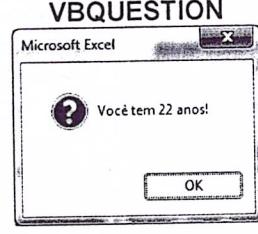
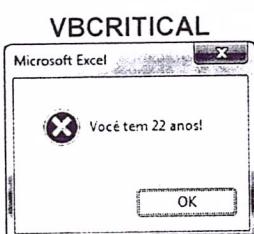
Uma mensagem será exibida logo em seguida exibindo a idade calculada:



Na prática é simples e útil utilizar janelas modais, no caso de MsgBox podemos alterar bastante sua configuração. Podemos incluir ícones de alerta diferenciados atribuindo constantes visuais **vbCritical** (crítica), **vbExclamation** (indagação), **vbInformation** (informação) e **vbQuestion** (pergunta):

MsgBox "Você tem " & idade & " anos!", vbCritical
MsgBox "Você tem " & idade & " anos!", vbExclamation
MsgBox "Você tem " & idade & " anos!", vbInformation
MsgBox "Você tem " & idade & " anos!", vbQuestion

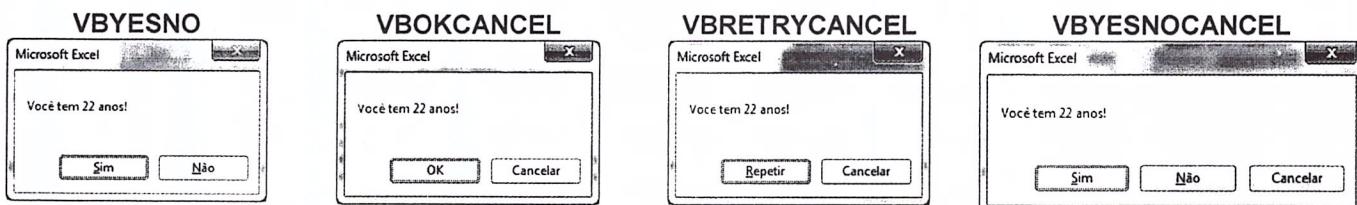
Resultados esperados:





É possível ainda atribuir **botões confirmação** a MsgBox esses botões alteram a forma de confirmação da MsgBox. Abaixo é exibido um trecho de código com alternativas a botões de confirmação:

```
MsgBox "Você tem " & idade & " anos!", vbYesNo  
MsgBox "Você tem " & idade & " anos!", vbOKCancel  
MsgBox "Você tem " & idade & " anos!", vbRetryCancel  
MsgBox "Você tem " & idade & " anos!", vbYesNoCancel
```



Esses são apenas alguns botões de confirmação, existem um grupo maior de opções que o VBA poderá informar pela própria ajuda.

Podemos criar MsgBox mais sofisticados combinando diversos elementos visuais, veja o código abaixo:

```
Sub suaIdade()  
    Dim ano, idade As Integer  
  
    ano = InputBox("Digite o ano que você nasceu:")  
  
    idade = Year(Date) - CInt(ano)  
  
    resultado = MsgBox("Você tem " & idade & " anos?", vbInformation + vbYesNo, "Confirmação")  
  
    If resultado = vbYes Then  
  
        MsgBox "Parabéns, você é feliz!", vbInformation  
    Else  
        MsgBox "Que pena! aproveite mais a vida", vbExclamation  
  
        If MsgBox("Repetir a pergunta?", vbQuestion + vbOKCancel, "Pergunta:") = vbOK Then  
            Call suaIdade  
        Else  
            Exit Sub  
        End If  
    End If  
End Sub
```

Esse código é capaz de avaliar a idade e testar a confirmação executadas em cada MsgBox.

Formulários

O uso de formulários vai muito além de simples janelas de mensagem, no caso de janelas modais a finalidade é simplesmente de avisar ao usuário sobre determina ação do código.



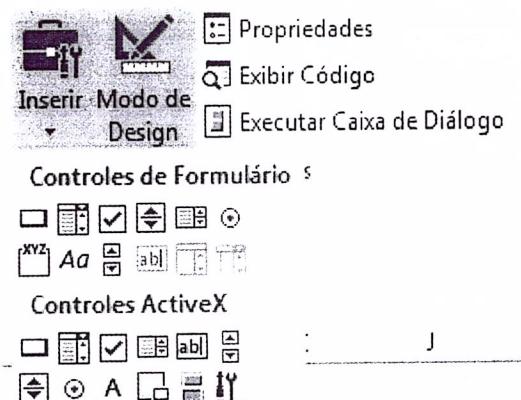


VBA

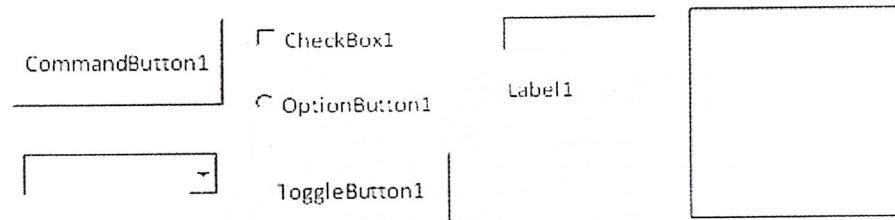
MICROSOFT EXCEL 2013 COM VBA

Apostila do Aluno

Já os formulários (Useforms) possuem elementos visuais e de interação maiores que as janelas modais, os formulários fazem parte dos chamados componentes ActiveX, esses componentes podem ser utilizados de duas formas. Diretamente na planilha, sendo inseridos e acessados na faixa de opções "desenvolvimento" em forma de componentes avulsos:



Exemplo de componentes ActiveX:



Geralmente criamos no VBE um formulário e inserimos os componentes necessários a sua manipulação:

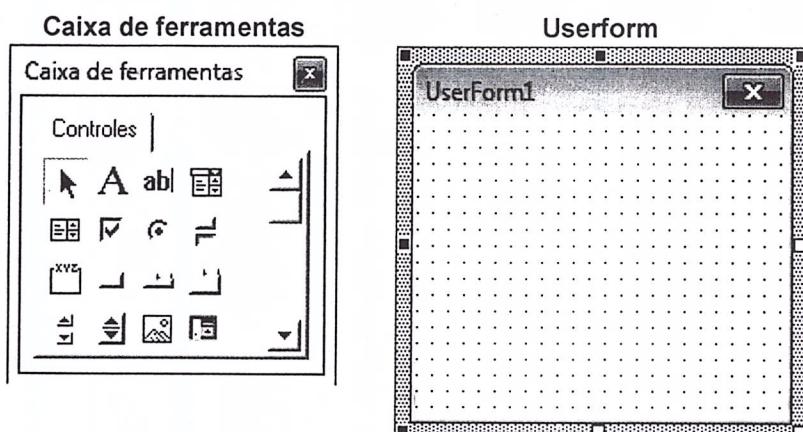
Inserindo um formulário (Useform):



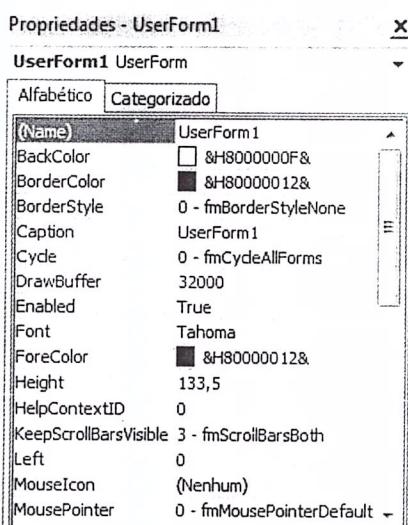


Componentes de um Useform

Para começar a criar o Userform precisamos das janelas abaixo:

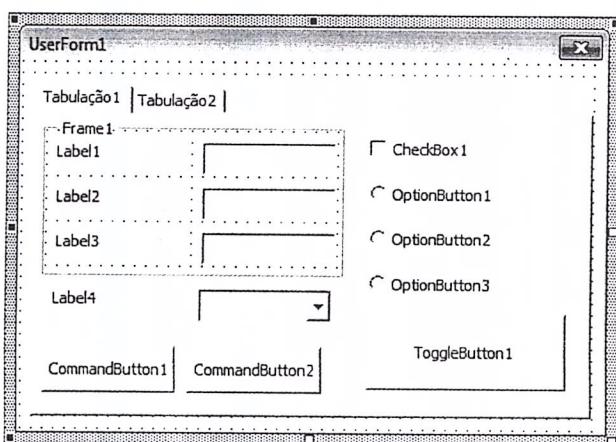


Cada componente dispõe de uma Janela de Propriedades:



Nessa listagem existem diversas configurações de controle associadas aos elementos do formulário que poderão ser manuseadas diretamente ou via código.

No formulário cada componente deve ser “desenhado” dentro da área do formulário:





Abaixo é apresentada uma lista rápida dos componentes existentes na caixa de ferramentas:

- | | |
|---|-------------------------------------|
| → Seletor de Objetos (Seleção de componentes) | ↔ Botão de ativação (Toggle Button) |
| A Rótulo (label) | ↔ Quadro (Frame) |
| abl Caixa de texto (Textbox) | └ Botão de comando (Command Button) |
| Caixa de combinação (Combobox) | REF Botão de referência (RefEdit) |
| Caixa de listagem (Listbox) | MPG Multi-Página (Multipage) |
| Barra de rolagem (Scrollbar) | Tabstrip |
| Botão de rotação (Spin Button) | IMG Imagem (Image) |
| Caixa de verificação (Checkbox) | OPC Botão de opção (Option Button) |

Criando um Formulário

Para ajudar a fixar o conceito de formulários criamos um formulário passo-a-passo.

✓ Configurando um formulário

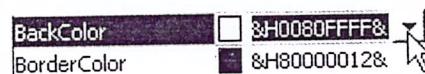
Crie um novo formulário configurando em seguida algumas propriedades abaixo para o formulário:

- Propriedade **Name**: FrmRegVendas
- Propriedade **Caption**: Registro de Vendas
- Propriedade **Height**: 450
- Propriedade **Width**: 485
- Propriedade **BackColor**: &H0080FFFF&

Isso é feito inserindo valores diretamente em cada propriedade:

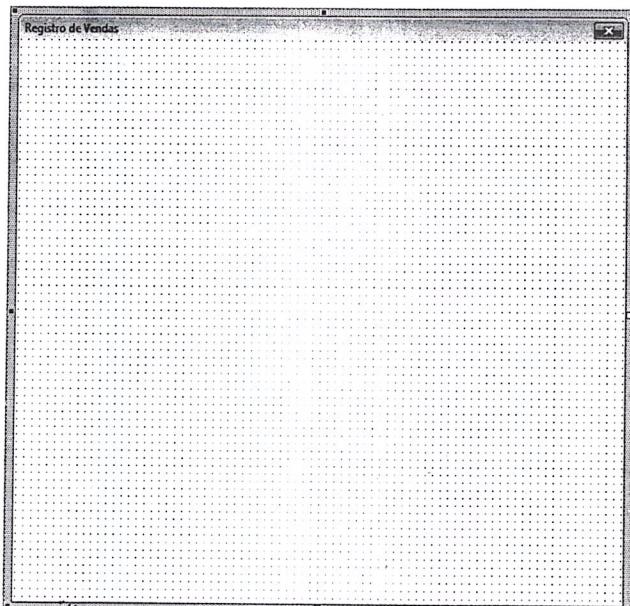


Ou clicando nos controles laterais:





Aparência do formulário FrmRegVendas



✓ Inserindo componentes no formulário

A alma do formulário são os componentes, são eles que criam a forma ideal de comunicação do usuário com os dados, esse conceito também é chamado "Interface". Abordaremos nesse tópico a inserção dos componentes em detalhes.

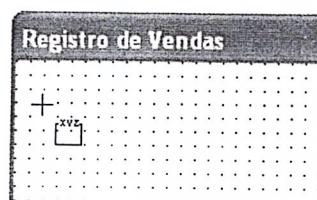
A) Inserindo Quadros (Frames):

Os quadros são componentes visuais que tem como característica destacar as principais áreas do formulário.

Na caixa de ferramentas o componente Quadro está marcado em destaque:

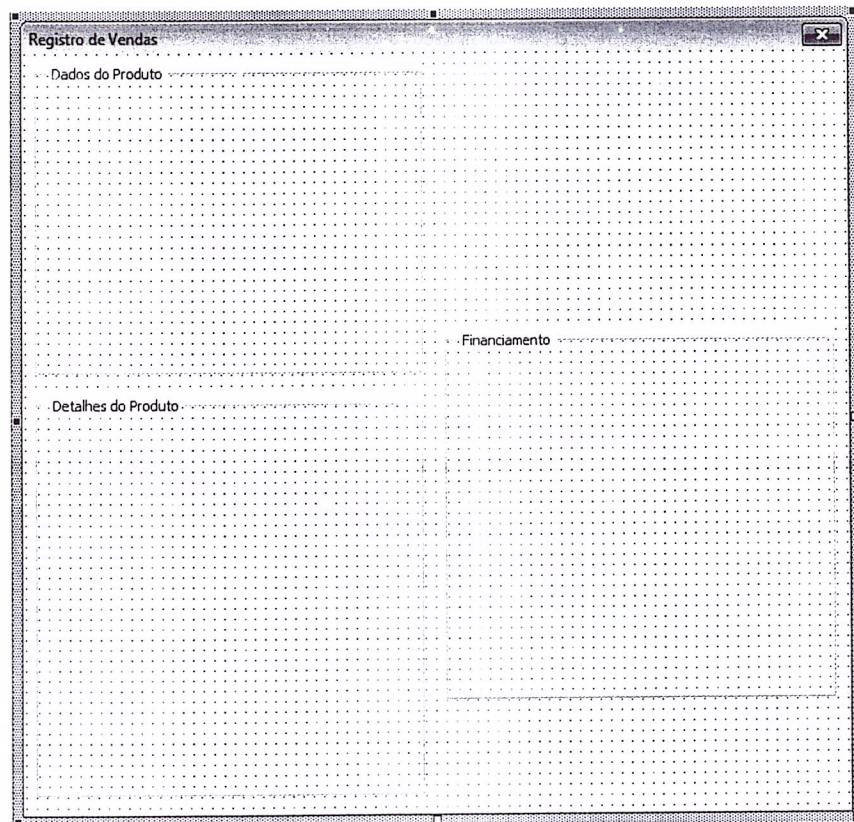


Para inserir os quadros selecione o componente na caixa de ferramentas e clique e arraste o mouse dentro da área do formulário:





Insira o total de 3 quadros dentro do formulário conforme abaixo:



Altere as propriedades Name e Caption de cada Quadro inserido:

Frame 1:

Propriedade Name: FrProduto
Propriedade Caption: Dados do Produto

Frame 3:

Propriedade Name: FrFinanc
Propriedade Caption: Financiamento

Frame 2:

Propriedade Name: FrDetProduto
Propriedade Caption: Detalhes do Produto

B) Inserindo Botão de Comando (Comand Button):

Os botões de comando já são conhecidos nossos, iremos inserir botões no formulário selecionando esse componente na caixa de ferramentas, esse componente se encontra em destaque:

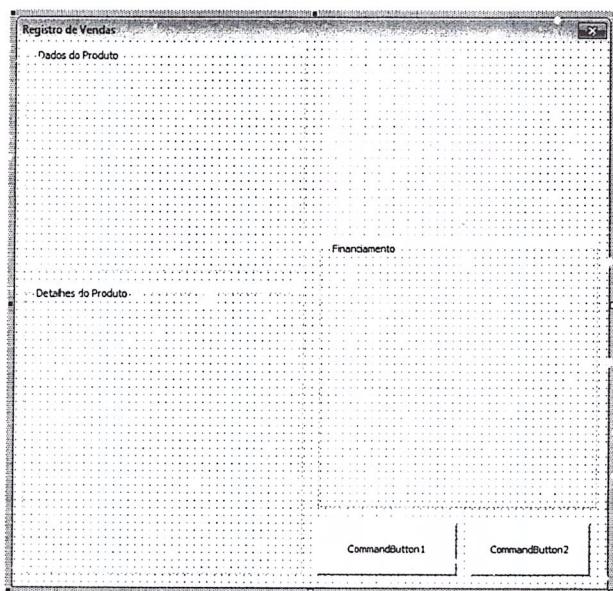


**VBA**

MICROSOFT EXCEL 2013 COM VBA

Apostila do Aluno

Insira os botões no local do formulário conforme a ilustração:



Altere as propriedades Name e Caption de cada Botão inserido:

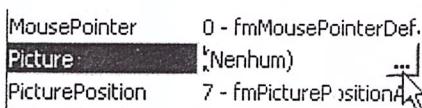
CommandButton1:

Propriedade **Name**: BtCadastrar
Propriedade **Caption**: Cadastrar

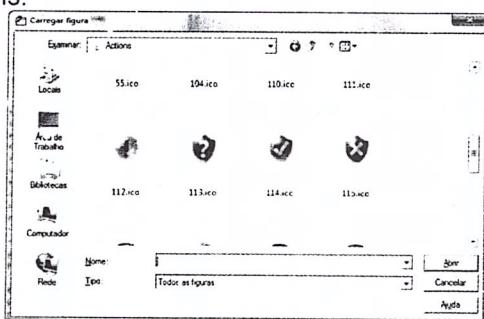
Command Button2:

Propriedade **Name**: BtCancelar
Propriedade **Caption**: Cancelar

Outra propriedade interessante para se usar com botões é a Picture:

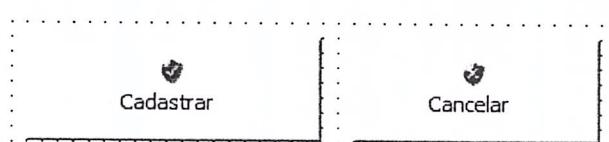


Essa propriedade permite anexar uma imagem ao botão, para isso clique no botão de três pontos, na lateral e selecione o local onde está as imagens.



O ideal que essas imagens sejam do tipo ico (arquivos de ícones) que se obtém um ótimo resultado.

Abaixo são exibidos os botões com as imagens inseridas:





Nota: Ao nomear um componente do formulário usamos siglas como Fr, Frm Bt, Lb para facilitar a identificação do objeto no código.

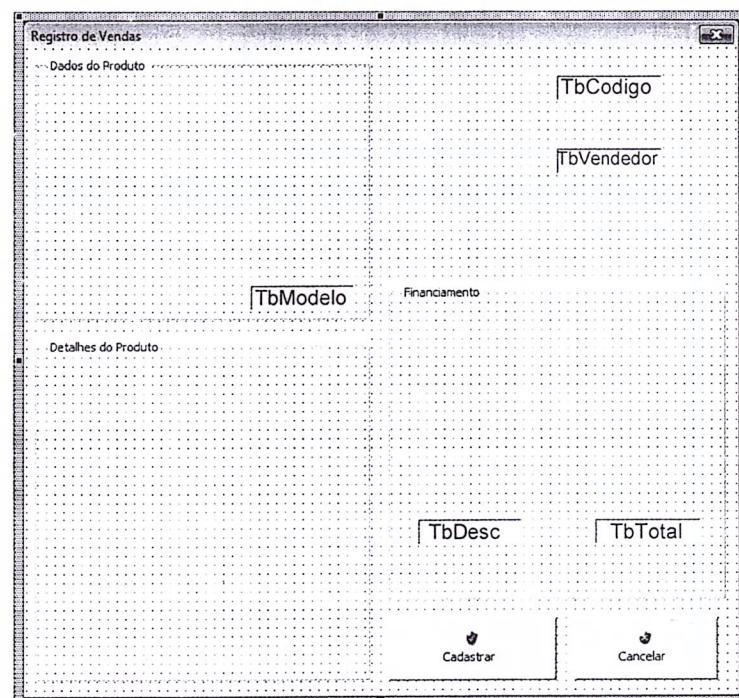
C) Inserindo Caixa de Texto (TextBox):

A caixa de texto é um componente utilizado tanto para entrada de dados, como para saída de dados.

Abaixo na caixa de ferramentas é localizado o componente TextBox.



Insira o componente TextBox conforme o exemplo abaixo:



Altere as propriedade Name de cada TextBox inserido:

TextBox1:

Propriedade Name: TbModelo

TextBox2:

Propriedade Name: TbCodigo

TextBox3:

Propriedade Name: TbVendedor

TextBox4:

Propriedade Name: TbDesc

TextBox5:

Propriedade Name: TbTotal





D) Inserindo Caixa de Listagem (ListBox):

A caixa de listagem, também conhecida como **ListBox**, esse componente exibe uma listagem visível para a seleção do usuário.

Exemplo de uma ListBox e de uma ComboBox:

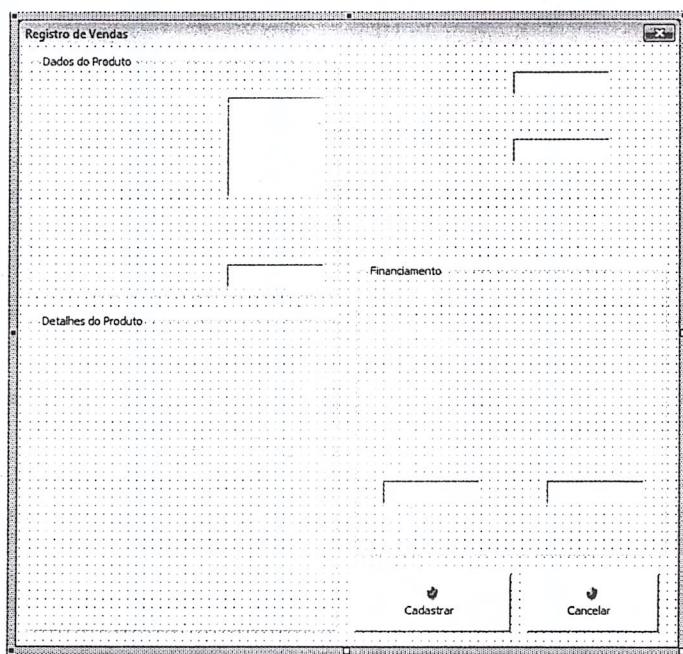
Caixa de listagem (ListBox)



Abaixo na caixa de ferramentas é localizado o componente **ListBox**.



Insira o componente **ListBox** conforme o exemplo abaixo:



Altere as propriedade **Name** de cada **ListBox** inserido:

ListBox1:

Propriedade Name: LbMarca



VBA

MICROSOFT EXCEL 2013 COM VBA

Apostila do Aluno

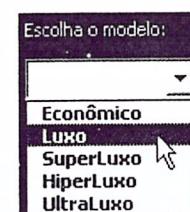
E) Inserindo Caixa de Combinação (ComboBox):

A caixa de listagem, também conhecida como **ListBox**, esse componente exibe uma listagem visível para a seleção do usuário.

A caixa de combinação ou **ComboBox** tem a princípio a mesma função da caixa de listagem (ListBox), ambas exibe uma lista de itens que podem ser selecionados, a diferença é que na ComboBox parece, a princípio, com um campo texto mais que em sua lateral existe geralmente um controle que quando clicado exibe uma lista de opções.

Exemplo de uma ListBox e de uma ComboBox:

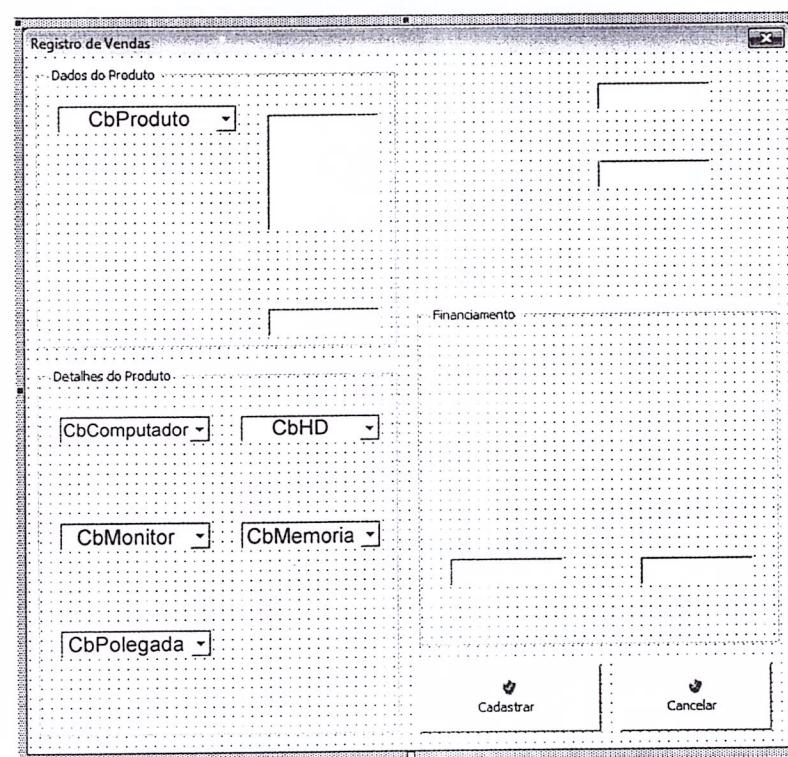
Caixa de Combinação (ComboBox)



Abaixo na caixa de ferramentas é localizado o componente **ComboBox**.



Insira o componente ComboBox conforme o exemplo abaixo:



ADV – Curso de Informática Telefone: (21) 2210-1180

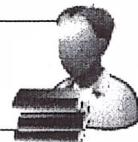
Av. Treze de Maio, 23 – 8º andar – Centro – Rio de Janeiro - RJ

<http://www.cursoadv.com.br/>

Página 85

**VBA**

MICROSOFT EXCEL 2013 COM VBA

Apostila do Aluno

Altere a propriedade Name de cada ListBox inserido:

ComboBox1:

Propriedade Name: CbProduto

ComboBox2:

Propriedade Name: CbComputador

ComboBox3:

Propriedade Name: CbMonitor

ComboBox4:

Propriedade Name: CbPolegada

ComboBox5:

Propriedade Name: CbHd

ComboBox5:

Propriedade Name: CbMemoria

F) Inserindo de checagem (CheckBox):

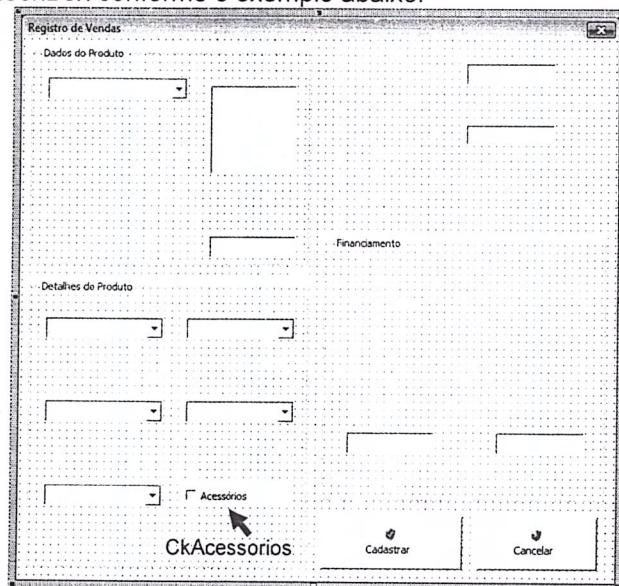
Esse componente pode assumir dois estados, um quando clicado (True) e o outro quando não clicado (False).
Exemplo de um componente Checkbox:



Abaixo na caixa de ferramentas é localizado o componente CheckBox.



Insira o componente CheckBox conforme o exemplo abaixo:



Altere as propriedades Name e Caption do componente Checkbox inserido:

CheckBox1:

Propriedade Name: CkAcessorios

Propriedade Caption: Acessórios





G) Inserindo Botões de Opção (OptionButton)

Esse componente é muito conhecido pelo efeito “múltipla escolha”, ele permite a seleção dinâmica de uma única opção de um grupo de opções, gerando um valor associado à opção escolhida.

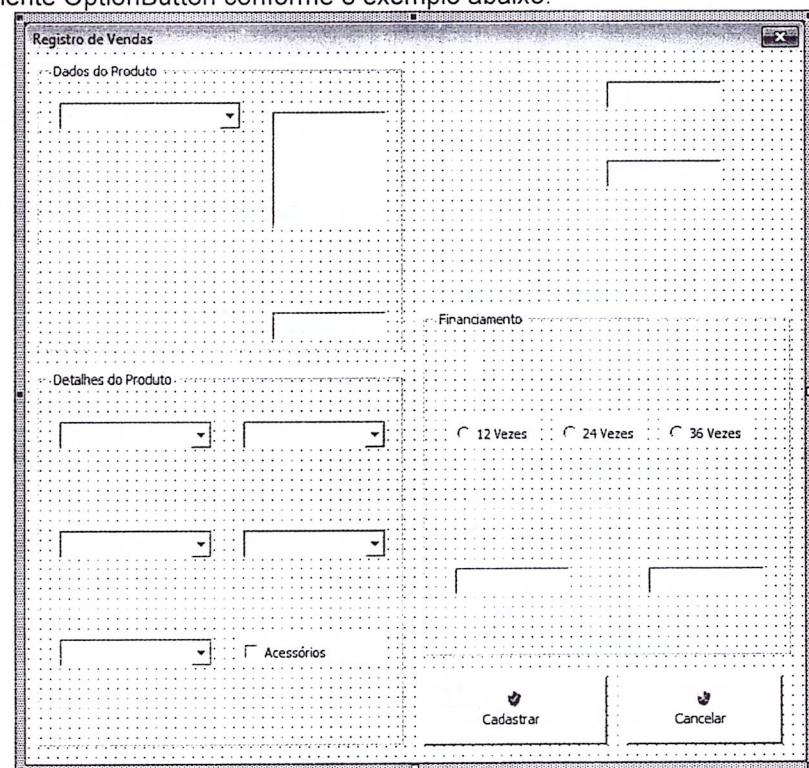
Exemplo de um componente OptionButton:



Abaixo na caixa de ferramentas é localizado o componente OptionButton.



Insira o componente OptionButton conforme o exemplo abaixo:



Altere as propriedades Name e Caption dos componentes OptionButton inseridos:

OptionButton1:

Propriedade Name: Opt12x

Propriedade Caption: 12 Vezes

Propriedade Enable: False

OptionButton2:

Propriedade Name: Opt24x

Propriedade Caption: 24 Vezes

Propriedade Enable: False

OptionButton3:

Propriedade Name: Opt36x

Propriedade Caption: 36 Vezes

Propriedade Enable: False





H) Inserindo Rótulos (Label)

Esse componente geralmente apresenta a função de "identificador" da maioria dos componentes dentro do formulário, em alguns casos poderá ser usado como exibição de mensagens.

Abaixo na caixa de ferramentas é localizado o componente Label.



Insira o componente Label conforme o exemplo abaixo:

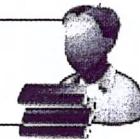




VBA

MICROSOFT EXCEL 2013 COM VBA

Apostila do Aluno



Altere as propriedades **Name** e **Text** dos componentes Label inseridos:

Label1:

Propriedade **Caption**: Produto

Label9:

Propriedade **Caption**: Memória RAM

Label2:

Propriedade **Caption**: Imagem do Produto

Label10:

Propriedade **Caption**: Código de Venda

Label3:

Propriedade **Caption**: Marca

Label11:

Propriedade **Caption**: Vendedor

Label4:

Propriedade **Caption**: Modelo do Produto

Label12:

Propriedade **Caption**: Data da Venda

Label5:

Propriedade **Caption**: Computador

Label13:

Propriedade **Text**: LbVenda

Label6:

Propriedade **Caption**: Monitor

Propriedade **BorderStyle**: 1 - FmBorderStyleSingle

Propriedade **TextAlign**: 2 – Fm TextAlignCenter

Propriedade **Font**: Negrito – Tamanho: 12

Label7:

Propriedade **Caption**: Polegadas

Label14:

Propriedade **Caption**: Desconto

Label8:

Propriedade **Caption**: Disco Rígido

Label15:

Propriedade **Caption**: Total



ADV – Curso de Informática

Telefone: (21) 2210-1180

Av. Treze de Maio, 23 – 8º andar – Centro – Rio de Janeiro - RJ

<http://www.cursoadv.com.br/>

Página 89



I) Inserindo Imagens (Image)

Esse componente tem a função de exibir dados visuais no formulário, como por exemplo, imagens de produtos, clientes, etc.

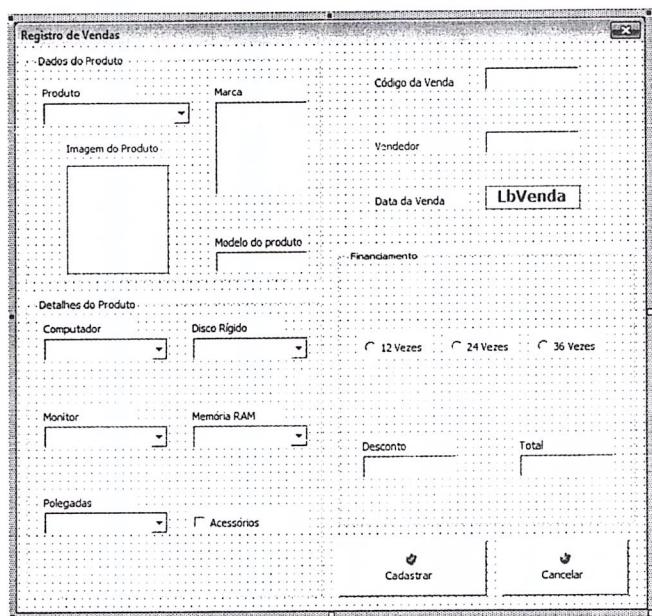
Exemplo de um componente Image:



Abaixo na caixa de ferramentas é localizado o componente Image.



Insira o componente Image conforme o exemplo abaixo:



Altere as propriedades Name e Text do componente Image inserido:

Image1:

Propriedade Name: ImgProduto

Propriedade Enable: False (Desabilita o componente)

Propriedade PictureSizeMode: 1 - fmPictureSizeModeStretch (ajusta o tamanho da image)





J) Inserindo um Botão de Ativação (ToggleButton)

Esse botão se diferencia do botão tradicional por assumir estados visuais de ativação: Ativado (true) e Desativado (False), indicando se está ativo ou não.

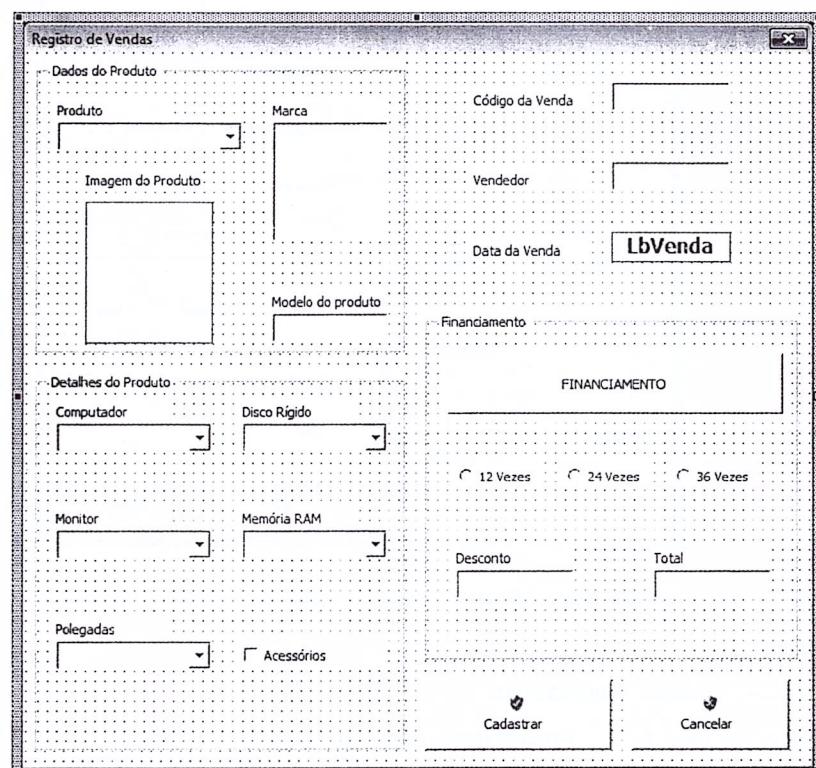
Exemplo de um componente Image:



Abaixo na caixa de ferramentas é localizado o componente ToggleButton.



Insira o componente ToggleButton conforme o exemplo abaixo:



Altere as propriedades Name e Caption do componente ToggleButton inserido:

ToggleButton1:

Propriedade Name: TgbFinanciamento

Propriedade Caption: FINANCIAMENTO





K) Inserindo Barra de Rolagem (ScrollBar)

A barra de rolagem é um componente que gera uma sequência de faixa de valores alteráveis através dos botões laterais.

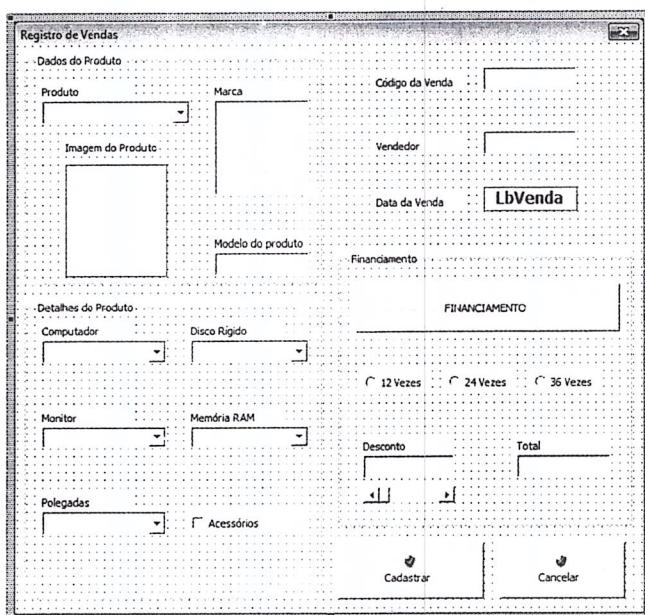
Exemplo de um componente ScrollBar:



Abaixo na caixa de ferramentas é localizado o componente ScrollBar.



Insira o componente ScrollBar conforme o exemplo abaixo:



Altere as propriedades Name, Min e Max do componente ScrollBar inserido:

ScrollBar1:

Propriedade Name: SbDesconto

Propriedade Min: 0 Propriedade Max: 30

Nota: Nessa etapa do formulário, embora demorada, é necessário para definir o "layout" de nosso formulário.

Embora não tenhamos utilizado todos os componentes do formulário, utilizamos a maioria dos componentes mais comuns, com o tempo e dedicação você aprenderá a mexer com todos componentes disponíveis.





Configurando componentes do Formulário

Após a inicialização do formulário, onde configuramos de certa forma alguns componentes do formulário; vamos configurar outros componentes para que funcionem de maneira adequada.

✓ Carregando as Imagens do Produto Selecionado

Vamos criar uma configuração para o formulário, que quando for escolhido na caixa de combinação **CbProduto** um determinado produto será exibido no componente **ImgProduto** a imagem do produto.

Insira o código abaixo dentro do módulo do objeto CbProduto:

```
Private Sub CbProduto_Change()
    'Carrega Imagens
    If CbProduto.Text = "" Then

        'Se não houver produto selecionado não exibe nada
        ImgProduto.Picture = LoadPicture(none)

    Else

        'Caso tenha um produto selecionado
        'Obtem o caminho de onde estão as imagens
        caminho = ThisWorkbook.Path
        ImgProduto.Picture = LoadPicture(caminho & "\Imagens\" &
        CbProduto & ".jpg")
    End If

End Sub
```

Nota: A propriedade **Picture** configura a imagem que será exibida e a função **LoadPicture** indica o caminho aonde se encontra o arquivo de imagem.

✓ Habilitando o Financiamento do Produto

Repare que inicialmente não é possível selecionar o tipo de financiamento pois os botões de opção estão desabilitados, vamos habilitá-los quando for clicado o botão de ativação.

Insira o código abaixo dentro do módulo do objeto TgbFinanciamento:

```
Private Sub TgbFinanciamento_Click()

    'Verifica o botão TgbFinanciamento estiver ativo
    If TgbFinanciamento = True Then

        'Será ativado os botões de opção
        Opt12x.Enabled = True
        Opt24x.Enabled = True
        Opt36x.Enabled = True

    Else

        'Caso contrário manterá os botões de opção desativado
        Opt12x.Enabled = False
        Opt24x.Enabled = False
        Opt36x.Enabled = False

    End If
End Sub
```





Abra seu formulário e clique no botão financiamento para testar se os botões de opção se tornam habilitados:

Botões de Opção Desabilitados

| | |
|---------------|--|
| FINANCIAMENTO | |
|---------------|--|

12 Vezes 24 Vezes 36 Vezes

Botões de Opção Habilitados

| | |
|---------------|--|
| FINANCIAMENTO | |
|---------------|--|

12 Vezes 24 Vezes 36 Vezes

✓ Configurando o Preenchimento do Nome do Vendedor e do Produto

Vamos configurar a caixa de texto TbVendedor para permite escrita somente em caixa alta.

Insira o código abaixo dentro do módulo do objeto TbVendedor:

```
Private Sub TbVendedor_Change()
    TbVendedor = UCase(TbVendedor)
End Sub
```

Vamos realizar uma configuração semelhante para o objeto TbNome, insira o código baixo dentro do módulo do objeto:

```
Private Sub TbModelo_Change()
    TbModelo = UCase(TbModelo)
End Sub
```

Abra seu formulário e clique e teste se as caixas de texto estão realmente escrevendo em maiúscula:

Caixa de texto TbVendedor

Vendedor

Caixa de texto TbModelo

Modelo do produto

✓ Configurando a Barra de Rolagem SbDesconto

A barra de rolagem SbDesconto irá gerar um valor percentual para preencher a caixa de texto SbDesc informando o percentual de desconto selecionado.

Insira o código abaixo no módulo do objeto SbDesconto:

```
Private Sub SbDesconto_Change()

    'Atribuindo o valor da barra de rolagem a caixa de texto
    TbDesc = SbDesconto

    'Convertendo o formato da caixa de texto para percentual
    TbDesc = Format(TbDesc / 100, "0%")

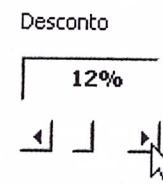
    'Alinhando o texto da caixa de texto ao centro
    TbDesc.TextAlign = fmTextAlignCenter

    'Tornando o conteúdo da caixa de texto negrito
    TbDesc.Font.Bold = True
End Sub
```





Abra seu formulário e clique nos botões da barra de rolagem SbDesconto para verificar se realmente está gerando o percentual de desconto dentro da caixa de texto TbDesc:



✓ Calculando o Valor da Venda

Perceba que até agora configuramos nosso formulário, ele já permite entrada e seleção de informações, mas ainda não está gerando o valor da compra.

Vamos a partir de agora criar uma rotina chamada **calculapreco** que conforme a escolha do usuário irá alterar o resultado do valor da compra.

1º) Insira a chamada da sub-rotina **calculapreco** nos seguintes módulos:

- Módulo da caixa de combinação CbComputador:

```
Private Sub CbComputador_Change()  
    Call calculapreco  
End Sub
```

- Módulo da caixa de combinação CbPolegada:

```
Private Sub CbPolegada_Change()  
    Call calculapreco  
End Sub
```

- Módulo da caixa de combinação CbHD:

```
Private Sub CbHD_Change()  
    Call calculapreco  
End Sub
```

- Módulo da caixa de combinação CbMemoria:

```
Private Sub CbMemoria_Change()  
    Call calculapreco  
End Sub
```

- Módulo da caixa de combinação CkAccesorios:

```
Private Sub CkAccesorios_Click()  
    Call calculapreco  
End Sub
```





- Módulo da caixa de combinação SbDesconto:

```
Private Sub SbDesconto_Change()  
  
    'Atribuindo o valor da barra de rolagem a caixa de texto  
    TbDesc = SbDesconto  
  
    'Convertendo o formato da caixa de texto para percentual  
    TbDesc = Format(TbDesc / 100, "0%")  
  
    'Alinhando o texto da caixa de texto ao centro  
    TbDesc.TextAlign = fmTextAlignCenter  
  
    'Tornando o conteúdo da caixa de texto negrito  
    TbDesc.Font.Bold = True  
  
    Call calculapreco  
End Sub
```

2º) Abra a área de módulos do formulário **FrmRegVendas** e crie a sub-rotina **calculapreco**, o código é mostrado abaixo:

```
Private Sub calculapreco()  
    Dim valorcomputador, valormonitor, valorHD, valormemoria As Currency  
    Dim valorpodegada, valoracessorios, valordesconto, valortotal As Currency  
  
    'PREÇO DO COMPUTADOR  
    If CbComputador.Text = "" Then  
        Exit Sub  
    Else  
        Computador = CbComputador.Text  
        Sheets("Dados").Select  
        Cells.Find(Computador).Activate  
        linha = ActiveCell.Row  
        coluna = ActiveCell.Column  
        valorcomputador = Cells(linha, coluna + 1)  
    End If
```

3º) Continuando o código anterior:

```
'PREÇO DO HD  
If CbHD.Text = "" Then  
    Exit Sub  
Else  
    HD = CbHD.Value  
    Sheets("Dados").Select  
    Cells.Find(HD).Activate  
    linha = ActiveCell.Row  
    coluna = ActiveCell.Column  
    valorHD = Cells(linha, coluna + 1)  
End If  
  
'PREÇO DA MEMÓRIA  
If CbMemoria.Text = "" Then  
    Exit Sub  
Else  
    memoria = CbMemoria.Value  
    Sheets("Dados").Select  
    Cells.Find(memoria).Activate  
    linha = ActiveCell.Row  
    coluna = ActiveCell.Column  
    valormemoria = Cells(linha, coluna + 1)  
End If
```





```
'PREÇO DO MONITOR
If CbPolegada.Text = "" Then
    Exit Sub
Else
    monitor = CbPolegada.Value
    Sheets("Dados").Select
    Cells.Find(monitor).Activate
    linha = ActiveCell.Row
    coluna = ActiveCell.Column
    valormonitor = Cells(linha, coluna + 2)
End If

'PREÇO DOS ACESSÓRIOS
If CkAcessorios.Value = True Then
    Sheets("Dados").Select
    valoracessorios = Range("e12").Value
Else
    valoracessorios = 0
End If

'PREÇO DO COMPUTADOR
valortotal = valorHD + valorcomputador + valormemoria + valormonitor + valoracessorios

'DESCONTO
If TbDesc = "" Then
    desconto = 0
Else
    tamanho = Len(TbDesc)
    desconto = Left(TbDesc, tamanho - 1)

    'Converte texto para valor single
    desconto = CSng(desconto)

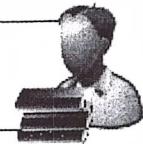
    'Obtem o percentual de desconto
    desconto = desconto / 100
End If

'Calcula o desconto sobre o valor total
valortotal = valortotal * (1 - desconto)

'converte o valortotal para formato moeda
TbTotal.Value = Format(valortotal, "currency")

'Formata o valor da caixa de texto
TbTotal.TextAlign = fm.TextAlignCenter
TbTotal.Font.Bold = True
End Sub
```





4º) Teste selecionar agora informações nos campos do formulário e veja se consegue resultado semelhante ao mostrado abaixo:

| | | | |
|---------------------|--|--|---|
| Detalhes do Produto | | FINANCIAMENTO | |
| Computador | Disco Rígido | | |
| Dual Core | 500 Gb | | |
| Monitor | Memória RAM | | |
| LED | 8Gb | | |
| Polegadas | <input checked="" type="checkbox"/> Acessórios | | |
| 32 pol. | | Desconto | Total |
| | | 2% | R\$ 2.881,20 |
| | | <input type="button" value=""/> | <input type="button" value=""/> |
| | | <input type="button" value="Cadastrar"/> | <input type="button" value="Cancelar"/> |

✓ Verificando Campos Vazios

Começaremos agora a preparar o formulário para realizar verificações de preenchimento dos campos pois devemos evitar que fique algum dados sem preencher antes de registrar alguma venda.

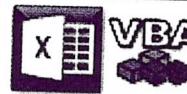
Isso será feito inserindo um código de verificação dentro do módulo do botão **Bt_Cadastrar**.

Insira o código abaixo para verificação do preenchimento dos campos:

```
Private Sub Bt_Cadastrar_Click()
    'Verifica se o campo CbProduto está vazio
    If CbProduto.Text = "" Then
        MsgBox "Selecione o Produto", vbCritical, "Aviso"
        CbProduto.SetFocus
        Exit Sub
    End If
```

(O código continua na próxima página)





(Continuação do código anterior)

```
'Verifica se o campo LbMarca esta vazio
If LbMarca.Text = "" Then
    MsgBox "Selecione a Marca", vbCritical, "Aviso"
    LbMarca.SetFocus
    Exit Sub
End If

'Verifica se o campo TbModelo esta vazio
If TbModelo.Text = "" Then
    MsgBox "Insira o Modelo", vbCritical, "Aviso"
    TbModelo.SetFocus
    Exit Sub
End If

'Verifica se o campo TbVendedor esta vazio
If TbVendedor.Text = "" Then
    MsgBox "Insira o nome do Vendedor", vbCritical, "Aviso"
    TbVendedor.SetFocus
    Exit Sub
End If

'Verifica se o campo CbComputador esta vazio
If CbComputador.Text = "" Then
    MsgBox "Selecione o Computador", vbCritical, "Aviso"
    CbComputador.SetFocus
    Exit Sub
End If

'Verifica se o campo CbMonitor esta vazio
If CbMonitor.Text = "" Then
    MsgBox "Selecione o tipo de Monitor", vbCritical, "Aviso"
    CbMonitor.SetFocus
    Exit Sub
End If

'Verifica se o campo CbPolegada esta vazio
If CbPolegada.Text = "" Then
    MsgBox "Selecione a Polegada do Monitor", vbCritical, "Aviso"
    CbPolegada.SetFocus
    Exit Sub
End If

'Verifica se o campo CbHD esta vazio
If CbHD.Text = "" Then
    MsgBox "Selecione a capacidade do Disco Rígido", vbCritical, "Aviso"
    CbHD.SetFocus
    Exit Sub
End If
```

(Continuação do código na próxima página)

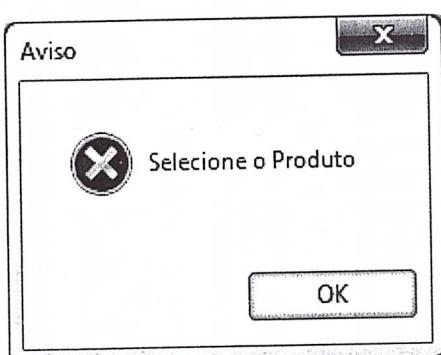




Digite o restante do código abaixo:

```
'Verifica se o campo CbMemoria esta vazio
If CbMemoria.Text = "" Then
    MsgBox "Selecione a capacidade da Memoria RAM", vbCritical, "Aviso"
    CbMemoria.SetFocus
    Exit Sub
End If
End Sub
```

Teste o código clicando no botão Confirmar, enquanto houver campos a serem preenchidos no formulário será exibida uma mensagem semelhante à mostrada abaixo:



✓ Preenchendo um Relatório a partir dos dados de um Formulário

Até agora fizemos apenas procedimentos de verificação simples em nosso formulário, a partir de agora daremos maior funcionalidade a utilização do formulário, iremos inserir registros cadastrados no formulário na planilha Cadastro.

Dentro do módulo de código do botão Bt_Cadastrar, clique após o último If inserido no código:

```
'Verifica se o campo CbMemoria esta vazio
If CbMemoria.Text = "" Then
    MsgBox "Selecione a capacidade da Memoria RAM", vbCritical, "Aviso"
    CbMemoria.SetFocus
    Exit Sub
End If
|
End Sub
```





Insira o código abaixo para gerar o cadastro a partir do formulário:

```
'Gerar código do cadastro
Sheets("Cadastro").Select
Range("A5").Select

If Range("A5") = "" Then 'se A5 está vazio, o banco estará vazio
    Cells(5, 1) = 1 'Registra a primeira venda sendo 1
    linha = ActiveCell.Row
Else
    Cells(4, 1).Select
    Selection.End(xlDown).Select
    linha = ActiveCell.Row
    valor = Cells(linha, 1).Value
    Cells(linha + 1, 1) = valor + 1 'Registra o número da próxima compra
    linha = ActiveCell.Row + 1
End If

'Registra a Data da Venda
Cells(linha, 2) = Date

'Registra o Nome do Vendedor
Cells(linha, 3) = TbVendedor.Text

'Registra o Nome do Produto
Cells(linha, 4) = CbProduto.Text

'Registra a Marca do Produto
Cells(linha, 5) = LbMarca.Text

'Registra o Modelo do Produto
Cells(linha, 6) = TbModelo.Text

'Registra o Tamanho da Memória RAM
Cells(linha, 7) = CbMemoria.Text

'Registra o Tamanho do Disco Rígido
Cells(linha, 8) = CbHD.Text

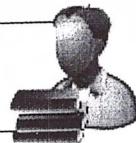
'Registra o Tamanho do Monitor
Cells(linha, 9) = CbPolegada.Text

'Registra o Tipo do Monitor
Cells(linha, 10) = CbMonitor.Text

'Registra se há Acessórios
If CkAcessorios = True Then
    Cells(linha, 11) = "OK"
Else
    Cells(linha, 11) = "NC"
End If
```

(Continue o código na próxima página)





Digite o restante do código abaixo:

```
'Registra se a Forma de Pagamento
If TgbFinanciamento = False Then
    Cells(linha, 12) = "À Vista"
Else
    If Opt12x = True Then
        Cells(linha, 12) = "12 Vezes"
    ElseIf Opt24x = True Then
        Cells(linha, 12) = "24 Vezes"
    Else
        Cells(linha, 12) = "36 Vezes"
    End If
End If

'Registra o Valor Total da Compra
Cells(linha, 13) = TbTotal.Text
End Sub
```

Tente inserir dados no formulário, clique no botão cadastrar e veja se obteve o resultado abaixo:

| | A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|--------|---------------|----------|----------|---------|--------|---------|--------|---------|------|------------|------------|----------------|
| 1 | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | |
| 4 | Código | Data da Venda | Vendedor | Produto | Marca | Modelo | Memória | HD | Monitor | Tipo | Acessórios | Financiado | Valor da Venda |
| 5 | 1 | 7/6/2010 | ADRIANO | NoteBook | Itautec | W-7655 | 4Gb | 320 Gb | 15 pol. | LCD | OK | 12 Vezes | R\$ 1.904,00 |

✓ Limpando Dados de um Formulário

Embora nosso formulário já funcione e permita cadastro ainda é mantido o cadastro nos campos, e isso não é interessante. Vamos inserir um código para que realize essa tarefa de limpeza.

Após o final do código que digitamos no módulo do botão Bt_Cadastrar clique no local indicado abaixo:

```
'Registra o Valor Total da Compra
Cells(linha, 13) = TbTotal.Text
```

```
End Sub
```



Insira o código abaixo para limpar todos os campos do formulário:

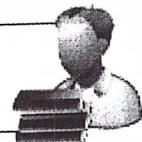
```
'Limpando Dados dos Campos do Formulário
CbProduto = ""
LbMarca = ""
TbModelo = ""
TbVendedor = ""
CbComputador = ""
CbMonitor = ""
CbPolegada = ""
CbHD = ""
CbMemoria = ""
CkAcessorios = False
TgbFinanciamento = False
Opt12x.Enabled = False
Opt24x.Enabled = False
Opt36x.Enabled = False
SbDesconto.Value = 0
TbDesc = ""
TbTotal = ""
```

Abra o formulário, cadastre alguns dados e clique no botão Cadastrar e veja se os campos ficarão totalmente limpos:

Registro de Vendas

| | | |
|--|--------------------------|---|
| Dados do Produto | | Código da Venda |
| Produto | Marca | <input type="text"/> |
| Imagen do Produto | | Vendedor |
| <input type="image"/> | | <input type="text"/> |
| Modelo do produto | | Data da Venda |
| <input type="text"/> | | <input type="text" value="<data>"/> |
| Detalhes do Produto | | |
| Computador | Disco Rígido | FINANCIAMENTO |
| <input type="text"/> | <input type="text"/> | <input type="checkbox"/> 12 Vezes <input type="checkbox"/> 24 Vezes <input type="checkbox"/> 36 Vezes |
| Monitor | Memória RAM | Desconto |
| <input type="text"/> | <input type="text"/> | <input type="text"/> |
| Polegadas | Acessórios | Total |
| <input type="text"/> | <input type="checkbox"/> | <input type="text"/> |
| <input type="button" value="Cadastrar"/> | | <input type="button" value="Cancelar"/> |





✓ Gerar Código de Cadastro no Formulário

Nosso formulário já é capaz de cadastrar agora dados de todo o registro de vendas realizado, falta agora ser indicado qual o código do novo registro a cadastrar.

Crie o código abaixo na área de módulo do código do funcionário e crie a sub-rotina do código abaixo:

```
Private Sub geracodigo()
    If Sheets("Cadastro").Range("A5") = "" Then
        codigo = 1
    Else
        linha = 5
        codigo = 1
        Do While Sheets("Cadastro").Range("A" & linha) <> ""
            codigo = codigo + 1
            linha = linha + 1
        Loop
    End If

    TbCodigo.Value = codigo
    TbCodigo.TextAlign = fmTextAlignCenter
    TbCodigo.Font.Bold = True
End Sub
```

Em seguida, vamos criar a chamada da sub-rotina em dois módulos, ao final do módulo do botão **Bt_Cancelar_Click()** e ao final do módulo da sub-rotina **Userform_Initialize()**.

Final do módulo Bt_Cadastrar_Click():

```
SbDesconto.Value = 0
TbDesc = ""
TbTotal = ""

' Inseri o código do cadastro no formulário
Call geracodigo
End Sub
```

Final do módulo Userfom_Inialize():

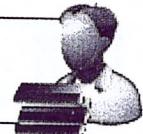
```
' Exibindo a Data Atual no rótulo LbData
LbData.Caption = Date

' Inseri o código do cadastro no formulário
Call geracodigo

End Sub
```

Se tudo der certo, toda vez que o formulário for aberto será exibido o código do próximo registro a ser cadastrado.





Observe o campo **Código da Venda** com o número do código de registro a cadastrar:

✓ Encerrando um Formulário

Para finalizar nosso formulário só falta uma ação simples que ainda não programamos, o de cancelar a janela do formulário.

Na verdade o código é bem simples, abra o módulo de código do botão **Bt_Cancelar** e insira o código abaixo:

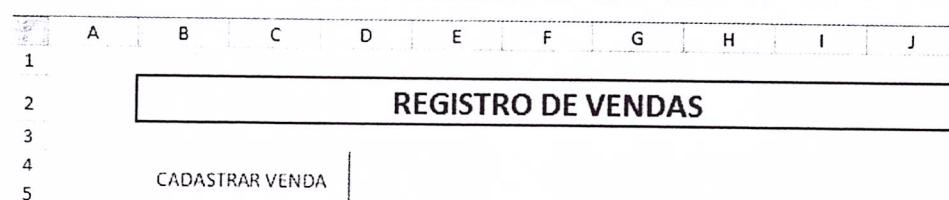
```
Private Sub Bt_Cancelar_Click()
    Unload FrmRegVendas
    Sheets("Principal").Activate
End Sub
```

Execute o formulário e clique no botão **Cancelar** que o formulário será prontamente fechado.

✓ Carregando um Userform:

O UserForm funciona de maneira semelhante a uma sub-rotina, precisa ser chamado para que carregue na tela, faça então o seguinte:

1º) Insira um botão de comando ActiveX diretamente na planilha:





2º) Abra o código do módulo do botão **Cadastrar Venda** e insira o código abaixo:

```
Private Sub CommandButton1_Click()  
  
    'Carrega o formulário registro de vendas  
    FrmRegVendas.Show  
  
End Sub
```

3º) Clique no botão **Cadastrar Venda** e verifique se o formulário abriu, caso sim, o código funcionou.

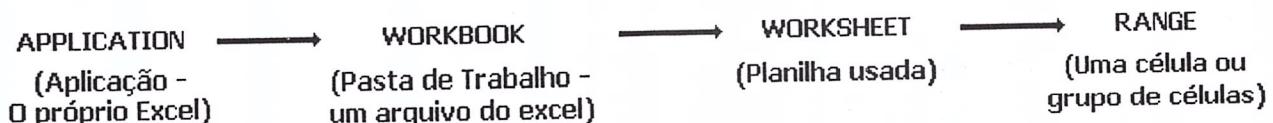
Nota: A propriedade **Show** exibe o formulário, algumas propriedades que configuramos manualmente também podem ser configuradas através do código.

CAPÍTULO 9 – MANIPULANDO OBJETOS E EVENTOS

9.1. Visão Geral de Objetos

Durante seu estudo com o VBA criamos muitos códigos que envolveram simples processamento, mas também alguns códigos que manipularam objetos.

Lembrando que no capítulo 1 desta apostila foi comentando que o VBA é um tipo linguagem orientada a objetos, ou seja, todos os componentes do Excel são considerados como "objetos".



Os Objetos aparecem nessa ilustração que foi apresentada inicialmente representando a hierarquia estrutural que a orientação a objetos seguida pelo VBA sendo Application o objeto de mais alto nível e o range o objeto de mais baixo nível.

Segundo o estudo realizado, os códigos abaixo têm o seguinte significado:

| Objeto | Significado |
|---------------------------------|---|
| RANGE("A1") ou Cells(1,1) | Diz respeito a uma célula na planilha |
| SHEETS("PRINCIPAL").RANGE("A1") | Identifica inicialmente a planilha em que uma célula é pertencente. |

Ou seja, quando fazemos referências a objetos o objeto de maior relevância vem sempre à esquerda dos objetos de menor relevância.

Outro conceito abordado até agora foram às propriedades, cada objeto possui um grupo de propriedades específicas.





As **Propriedades** dizem respeito as características relacionadas a cada objeto. Veja os exemplos abaixo:

| Propriedade | Significado |
|----------------------------------|--|
| RANGE("A1").VALUE = "CASA" | Valor aplicado ao objeto range. |
| RANGE("A1").FONT.COLOR = VBGREEN | Altera a propriedade de cor do objeto Range. |

As **Funções** representam capacidades associadas a determinado objeto. Exemplos de funções associadas a objetos:

| Função | Significado |
|---------------------|---------------------------------------|
| MsgBox UCASE(Dados) | Converte caracteres em maiúscula |
| MsgBox Now | Exibe a data e hora atual do sistema. |

Repare que nem todas as funções aparecem associadas a objetos, algumas são nativas, ou seja, o VBA reconhece e prontamente executa e outras devem ter uma associação a um determinado objeto.

Os **Eventos** estão associados à forma como as sub-rotinas são executadas pelo VBA, veja os exemplos abaixo:

| Eventos | Significado |
|-----------------------------------|--|
| PRIVATE SUB USERFORM_INITIALIZE() | Executa os códigos quando o formulário é aberto. |
| PRIVATE SUB BOTAO_CLICK() | Executa os códigos quando o botão é clicado. |

9.2. Objeto Application

Como já foi abordado antes o objeto Application se encontra no ponto mais alto da hierarquia do modelo orientado a objetos no VBA e representa o aplicativo Excel como um todo. Contém todas as pastas de trabalho (Workbooks) abertas, e cada pasta de trabalho possuidora de suas próprias planilhas (Worksheets) e cada planilha contendo suas células (Ranges).

Quando se utiliza orientação a objetos, por exemplo, todas as propriedades pertencentes ao objeto, mesmo esse sendo inferior na hierarquia não poderá ser acessado diretamente sem passar pelo objeto.

Conhecendo alguns Métodos e Propriedades do Objeto Application

✓ Propriedades do Objeto Application

Eles alteram ou acessam propriedades do objeto Application.

- **Caption:** Essa propriedade permite alterar a legenda da barra de título do aplicativo.

