Excel Avançado

Trabalhando com Macros e VBA

Trilha Conhecimento em Rede

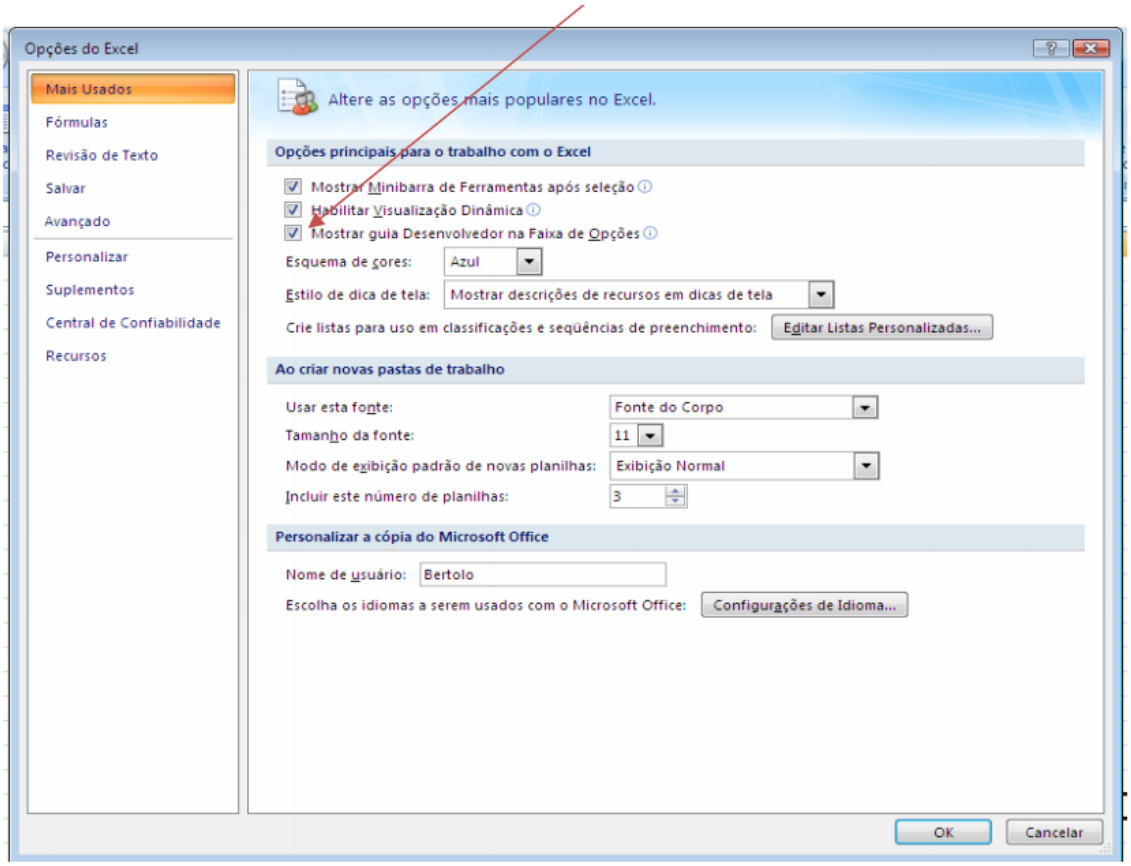
GOVERNO DO ESTADO
DO ESPÍRITO SANTO

1. Habilitar Guia Desenvolvedor

Para trabalhar com Macros ou VBA deve ser habilitado a Guia *Desenvolvedor*.

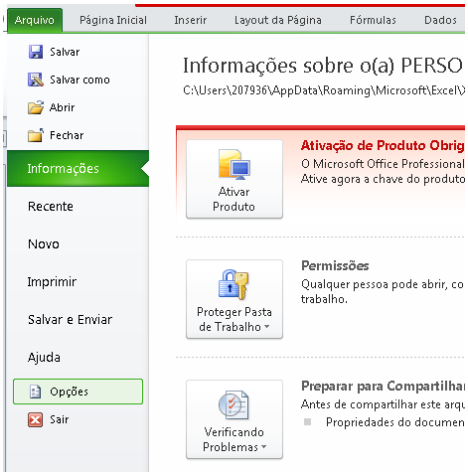
1.1. Office 2007

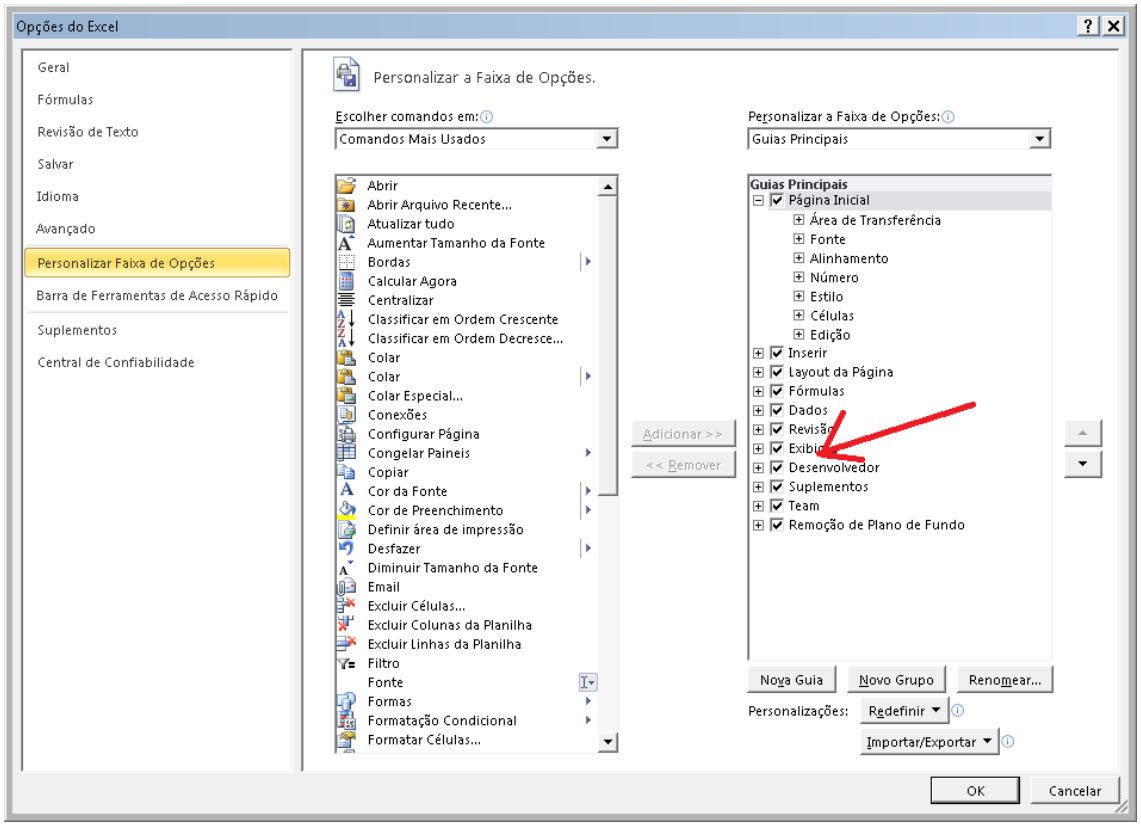
- Para tanto vá ao botão do Office 
- Clique em Opções do Excel
- E marque a caixa de verificação abaixo:



1.2. Office 2010 e posteriores

- Ir na Guia *Arquivos*
- Ir em Opções (conforme figura ao lado)
- Ir em *Personalizar Faixa de Opções*
- Marcar a Guia *Desenvolvedor* conforme figura abaixo





2. Macro

Se houver tarefas executadas de forma recorrente no Microsoft Excel, você poderá gravar uma macro que automatize essas tarefas. Macro é uma ação ou um conjunto de ações que podem ser executadas quantas vezes você desejar. Ao criar uma macro, você está gravando cliques do mouse e pressionamentos de tecla. Depois de criar uma macro, você pode editá-la para fazer pequenas alterações na maneira como ela funciona.

Suponha que todo mês você cria um relatório para seu gerente de contabilidade e deseja formatar os nomes dos clientes com pagamento atrasado em vermelho e aplicar a formatação de negrito. Você pode criar e executar uma macro que aplique rapidamente essas alterações de formatação nas células selecionadas.

2.1. Antes de gravar uma macro

As ferramentas Macros e VBA podem ser encontradas na guia Desenvolvedor, que fica oculta por padrão, portanto, a primeira etapa é habilitá-la (Ver tópico 1).

2.2. Gravar a Macro

- No grupo **Código** da guia **Desenvolvedor**, clique em **Gravar Macro**;
- Opcionalmente, insira um nome para a macro na caixa **Nome da macro**, insira uma tecla de atalho na caixa **Tecla de atalho** e uma descrição na caixa **Descrição** e clique em **OK** para começar a gravação;

- Execute as ações que deseja automatizar, como por exemplo, inserindo um texto clichê ou preenchendo uma coluna de dados;
- Na guia **Desenvolvedor**, clique em **Interromper Gravação**;

2.3. Examine melhor a macro

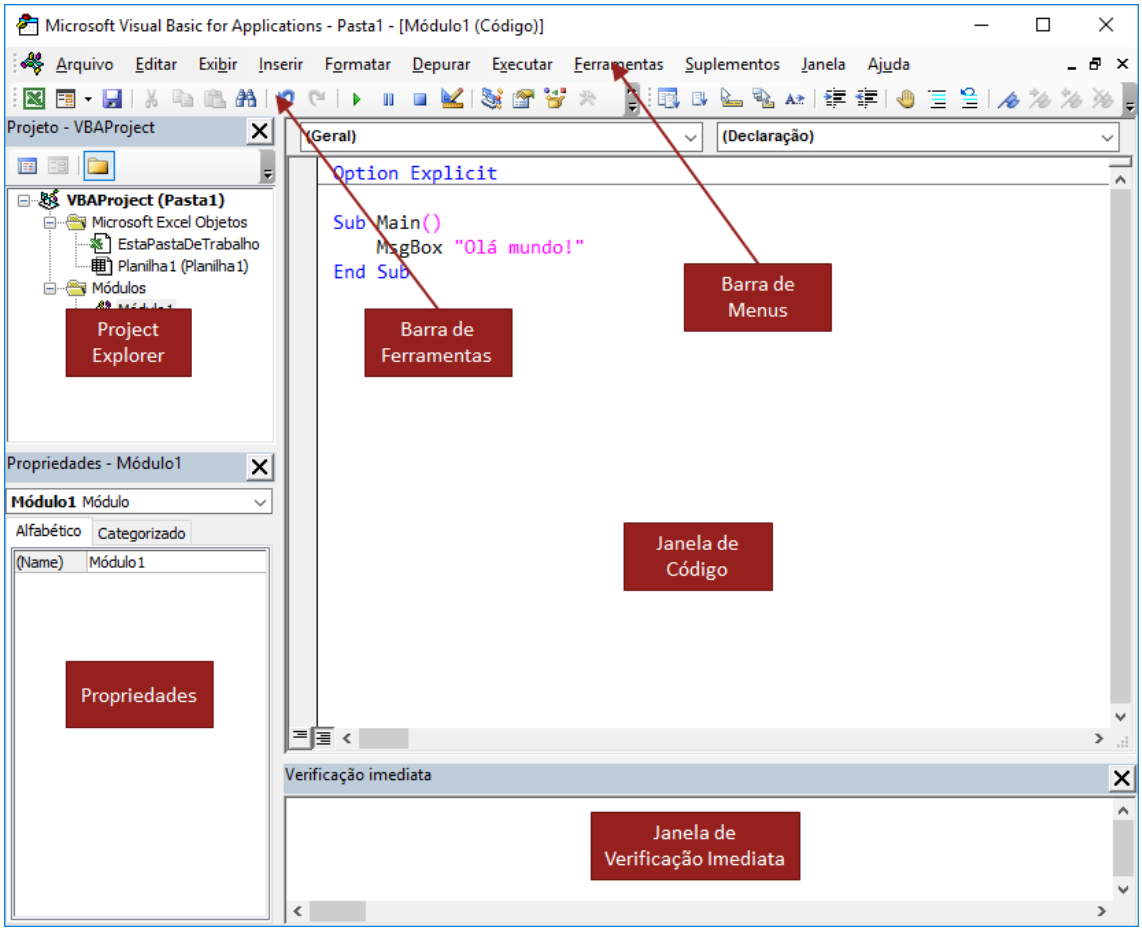
Você pode aprender um pouco sobre a linguagem de programação Visual Basic editando uma macro.

Para editar uma macro, no grupo **Código** da guia **Desenvolvedor**, clique em **Macros**, selecione o nome da macro e clique em **Editar**. Isso inicia o VBE - Editor do Visual Basic. Outra forma de acessá-lo é o atalho ALT + F11.

Veja como as ações que você gravou aparecem codificadas. Alguns códigos provavelmente serão claros para você, enquanto outros não.

Teste o código, feche o Editor do Visual Basic e execute a macro novamente. Desta vez, veja se algo diferente acontece!

A figura abaixo exibi o VBE - ambiente onde fica organizado os códigos VBA.

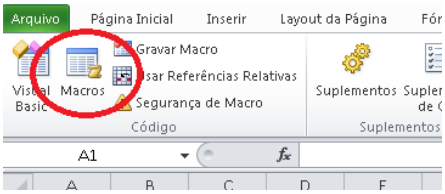


2.4. Executar a Macro

Há várias maneiras de executar uma macro:

2.4.1.Lista de Macros

Através da guia **Desenvolvedor** a opção **Macros** abre uma lista de Macros gravadas permitindo a sua execução.



2.4.2.Tecla de Atalho

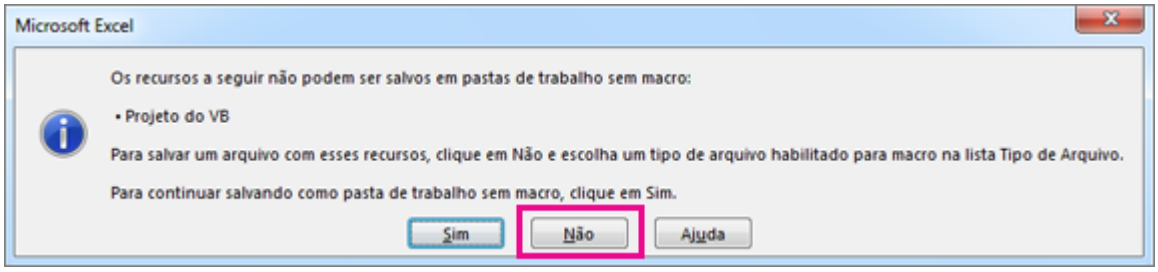
Ao gravar a macro é permitido acrescentar uma tecla de atalho;

2.4.3.Associando a algum objeto do Excel como por exemplo um botão
2.4.4.Associando a algum evento do Excel como por exemplo abrir a planilha
2.4.5.Diretamente no editor de códigos (VBE) através do item **Executar Macro** (F5)

3. Salvar uma Macro

3.1. Salvar uma macro com a pasta de trabalho atual

Se você precisar da macro somente na pasta de trabalho onde ela foi criada, clique em **Salvar** ou **Salvar como**, como faria normalmente. Mas o processo para salvar uma pasta de trabalho com macros é um pouco diferente porque ela precisa estar em um formato de arquivo especial "habilitado para macro". Então, quando você tentar salvá-la, o Excel oferece duas opções:

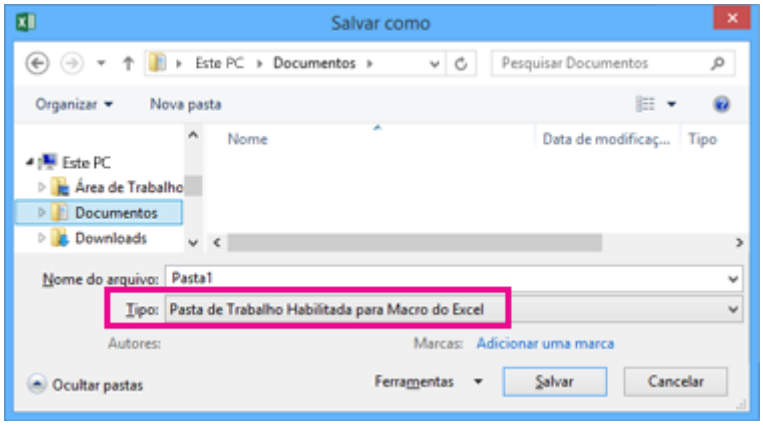


- Salvar como uma pasta de trabalho habilitada para macro (tipo de arquivo *.xlsm) clicando em **Não**.
- Salvar como uma pasta de trabalho sem macro clicando em **Sim**.

Para salvar como uma pasta de trabalho habilitada para macro:

a. Clique em **Não**.

b. Na caixa Salvar como , sob a caixa de listagem **Salvar como tipo**, escolha **Pasta de Trabalho Habilitada para Macro do Excel (*.xlsm).**



c. Clique em **Salvar.**

3.2. Crie e salve a macro na sua pasta de trabalho Pessoal

Para deixar suas macros disponíveis sempre que abrir o Excel, crie-as em uma pasta de trabalho chamada Pessoal.xlsb. Essa é uma pasta de trabalho oculta, armazenada no seu computador, que é aberta sempre que você inicia o Excel.

4. Tipos de Macros

4.1. Macros Sub

Se você deseja que o Excel VBA execute algumas ações, você pode usar uma sub. Coloque uma sub em um módulo (no Editor do Visual Basic, clique em Inserir, módulo).

```
Sub Area(x As Double, y As Double)

MsgBox x * y

End Sub
```

As Macros Sub podem ser geradas pela **Gravação de Macros.**

4.2. Macros Function

Se você deseja que o Excel VBA execute uma tarefa que retorna um resultado, você pode usar uma função Coloque uma função em um módulo (no Editor do Visual Basic, clique em Inserir, módulo). Por exemplo, a função com o nome Area.

```
Function Area(x As Double, y As Double) As Double

Area = x * y

End Function
```

As funções podem ser chamadas pela planilha semelhante de como chamamos a função SOMA na planilha.

5. Princípios da Programação

Na **Gravação de Macros** é gerada uma Macro contendo todos os passos que usuário efetuar no arquivo Excel. Esses passos na verdade são comandos que ficam armazenados na planilha. Esses comandos são códigos da linguagem de programação *Visual Basic for Application* – VBA. Abaixo segue alguns conceitos aplicados a programação.

5.1. Comentários

Um comentário pode ser usado para documentar o código.

Um comentário é iniciado com um apóstrofo (‘). O VBA ignora qualquer texto que siga um apóstrofo em uma linha de código. Você pode usar uma linha inteira para o seu comentário ou inserir o seu comentário ao final de uma linha de código.

5.2. Variável

Uma variável é apenas um local de armazenagem, nomeado, na memória do seu computador.

```
Dim YourName As String
Dim AmountDue As Double
Dim RowNumber As Long
```

5.3. Constantes

O valor de uma variável pode mudar (e normalmente muda) enquanto o seu procedimento está em execução. Por isso é que ela é chamada de variável. Às vezes, você precisa fazer referência a um valor ou string que nunca muda. Nesse caso, você precisa de uma constante – um elemento nomeado cujo valor não muda.

```
Const NumQuarters As Integer = 4
Const Rate = .0725, Period = 12
```

5.4. Atribuição

Uma atribuição é quando alteramos o valor de uma variável. Ex.

```
x = 1
x = x + 1
x = (y * 2) / (z * 2)
HouseCost = 375000
```

5.5. Estruturas Condicionais

A Estrutura Condicional possibilita a escolha de um grupo de ações e estruturas a serem executadas quando determinadas condições são ou não satisfeitas.

```
If Time < 0.5 Then
    MsgBox "Bom dia"
Else
    MsgBox "Boa tarde"
End If
```

5.6. Estruturas de Repetição

É uma estrutura que permite executar mais de uma vez o mesmo comando ou conjunto de comandos, de acordo com uma condição ou com um contador.

```
  Dim StartVal As Long
  Dim NumToFill As Long
  Dim CellCount As Long
  StartVal = InputBox("Insira o valor inicial: ")
  NumToFil = InputBox("Quantas células? ")
  For CellCount = 1 To NumToFill
      ActiveCell.Offset(CellCount - 1, 0) = _
         StartVal + CellCount - 1
Next CellCount
```

```
Do While ActiveCell.Value <> Empty
    ActiveCell.Value = ActiveCell.Value * 2
    ActiveCell.Offset(1, 0).Select
Loop
```

6. Objetos

´Você já usou um pouco o Excel, mas provavelmente nunca pensou nele como um objeto. Quanto mais você trabalha com VBA, mais vê o Excel nesses termos. Você entenderá que o Excel é um objeto e que ele contém outros objetos. Esses objetos, por sua vez, contém ainda mais objetos. Em outras palavras, a programação VBA envolve trabalhar com uma hierarquia de objetos.

No alto desta hierarquia está o objeto Application (Aplicativo) — neste caso, o próprio Excel (a mãe de todos os objetos).
Principais objetos:

- Application (Próprio Excel)
- Window (Janela)
- Workbook (pasta de trabalho)
- WorksheetFunction (Funções das planilhas)
- Worksheet (Planilha)
- Range (Intervalo de Células)

Se você quer trabalhar com o objeto Application, é fácil: comece digitando Application.
Todos os outros objetos no modelo objeto do Excel estão sob o objeto Application.Você obtém esses objetos descendo na hierarquia e conectando cada objeto à sua maneira, com o operador ponto (.). Para ter o objeto Workbook chamado “Pasta1.xlsx”.comece com o objeto Application e navegue para o objeto da coleção Workbooks.
Para navegar além de uma planilha específica, adicione um operador ponto e acesse o objeto da coleção Worksheets.
Se quiser obter o valor da célula A1 na primeira planilha da pasta de trabalho chamada Pasta1.xlsx, é preciso passar para o nível do objeto Range. Ex.
Application.Workbooks(“Book1.xlsx”).Worksheets(1).Range(“A1”).Value
Ao fazer referência a um objeto Range dessa forma, isso é chamado de referência totalmente qualificada. Você informou ao Excel exatamente qual intervalo deseja, em qual planilha e pasta de trabalho, e não deixou qualquer coisa à imaginação.
Caso simplifique o comando como o código abaixo:
Range(“A1”).Value
A referencia será para a planilha ativa.

7. Formulário

Um formulário, seja impresso ou online, é um documento projetado com uma estrutura e um
formato padrão que facilitam a captura, a organização e a edição de informações. Formulários online contêm os mesmos recursos de formulários impressos e também possuem controles.

7.1. Tipos de Formulários

No Excel podemos trabalhar com os seguintes tipos de formulários:

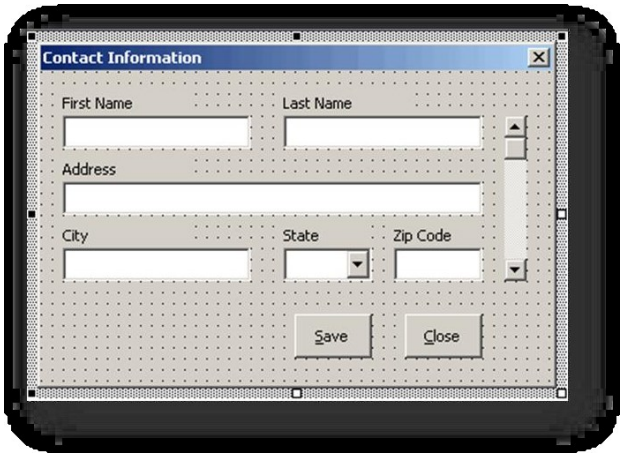
- **Formulário de dados:** Utilizado para inserir/exibir uma linha

completa de informações em um intervalo ou uma tabela. Este tipo de formulário torna a entrada/alteração de dados mais fácil do que aquela realizada diretamente na planilha de dados.

- **Planilha com controles de formulário e ActiveX:** São controles e objetos armazenados em planilhas que podem ser vinculados às células e permite a inserção de dados por meio de botões e caixas de texto.
- **Formulários do Visual Basic (VBA):** São formulários criados a partir do editor do Visual Basic (VBE), com campos conectados à base de dados por meio de código.

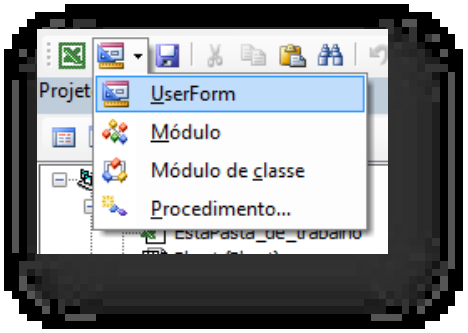
7.2. Formulários de Usuários (UserForms)

Combinam as capacidades da Inputbox e da MsgBox para criar uma maneira mais eficiente de interagir com o usuário.

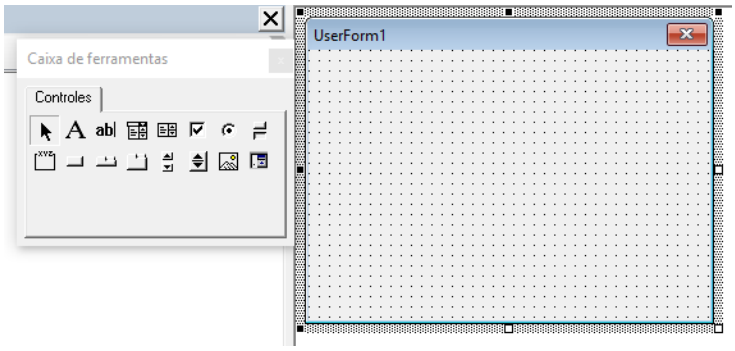


7.3. Criando um UserForm

Para criar um UserForm dentro do VBE clicamos no menu “Inserir > UserForm” ou no botão e, em seguida clicar em “UserForm”.



Uma janela de edição de seu UserForm será aberta, juntamente com a caixa de ferramentas para inclusão dos objetos de formulários.



7.4. Chamando e ocultando um UserForm

Para que um código VBA faça uso de um UserForm é necessário estabelecer os seguintes comandos:

```
<nome_formulario>.Show
```
Mostra o formulário para o usuário

```
Load <nome_formulario>
```
Chama o formulário sem exibição (ativa)

```
<nome_formulario>.Hide
```
Oculta um formulário para o usuário (permanece ativo)

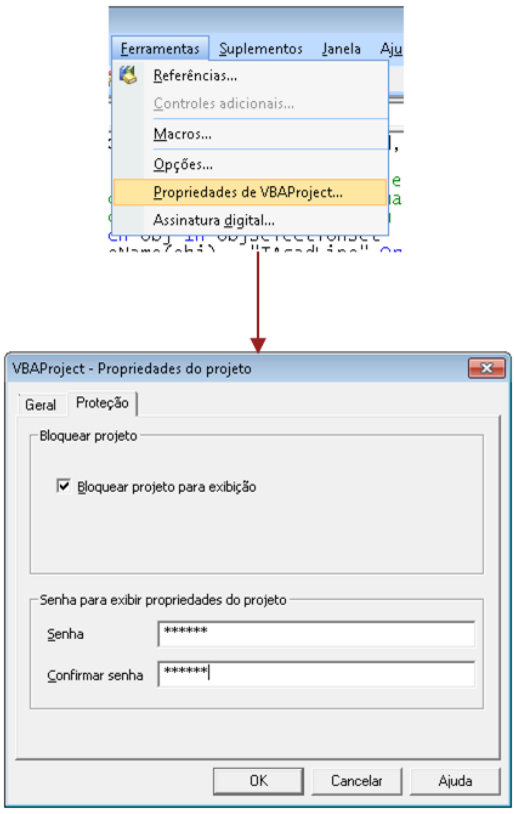
```
Unload Me
```
Descarrega o formulário da memória (torna inacessível)

Para descarregar o formulário da memória utilizamos um comando "*Me*" dentro do código VBA. "*Me*" é uma palavra-chave que pode se usada para se referir ao userform em si. Pode ser usada no código de qualquer controle para se referir a si mesmo.

Todo o controle dos campos de formulário é realizado pela linguagem VBA, portanto, para que se realize a interação com o usuário e o fluxo dos dados precisamos criar nossos códigos referenciando cada um dos objetos utilizados no formulário pelo seu atributo ***Name.*** Dessa forma, recomenda-se que sejam criados nomes que facilitem a identificação dos campos, aumentando assim a legibilidade do código VBA

8. Proteger um Projeto

É possível proteger seu projeto para que outras pessoas não vejam a estrutura de módulos e código. Você pode fazer isso para proteger sua propriedade intelectual ou simplesmente para que nenhum usuário altere o código de seu programa e você possa garantir sua funcionalidade. Para fazer isso, vá na guia Ferramentas >> controle **Propriedades de <nome do seu projeto>**. Na nova janela, selecione a aba **Proteção**, habilite a caixa de seleção **Bloquear projeto para exibição**, preencha o campo de senha, redigite a senha na caixa de texto abaixo e, em seguida, clique em **OK**:



9. Dicas

9.1. Alguns Atalhos

ALT + F8 -> Testar/Executar Macro
ALT + F11 -> VBE (Visual Basic Editor)
F5 -> No VBE, Executa Código
F2 -> No VBE, Abre o Pesquisador de Objeto
CTRL + Space -> Nos códigos abre o *autocoplete*

9.2. Algumas formas de usar o objeto Range

Range(“A1:C5”)
Range(“K9”)

Range(“PriceList”)

Worksheets(“Sheet1”).Range(“A1:C5”)

Workbooks("Budget.xlsl”).Worksheets(“Sheet1”).Range(“A1:C5”)

Range(“3:3”)

Range(“D:D”)

Range(“A1:B8,D9:G16”)

Worksheets(“Sheet2”).Cells(2, 3) -> Linha e Coluna ... Propriedade

Range(Cells(1, 1), Cells(10, 10)) -> Esta expressão refere-se a uma faixa de célula 100, que se estende da célula A1 (linha 1, coluna 1) à célula J10 (linha 10, coluna 10)

Range(“A1:J10”).Value = 99

Range(Cells(1, 1), Cells(10, 10)).Value = 99

Range(“A1”).Offset(1, 2)

Como a propriedade Cells, a propriedade Offset toma dois argumentos.

O primeiro argumento representa o número de linhas a deslocar; o

segundo representa o número de colunas a deslocar.

Range(“C2”).Offset(-1, -2)

9.3. Algumas Propriedades úteis do objeto Range

-> Value

-> Text (valor formatado na célula se tiver uma formula o resultado da formula)

-> Count

-> Column

-> Row

Não confunda as propriedades Column e Row com as propriedades

Columns e Rows (discutidas anteriormente neste capítulo). As propriedades Column e Row retornam um único valor. As propriedades

Columns e Rows retornam um objeto Range. Que diferença um “s” faz.

-> Address

-> HasFormula (True/False/Null(faixa de valores e parte seja de formulas))

Dim FormulaTest As Variant

FormulaTest = Range(“A1:A2”).HasFormula

If TypeName(FormulaTest) = “Null” Then MsgBox “Mixed!”

-> Font

-> Interior

-> Formula

Range(“A13”).Formula = “=SUM(A1:A12)”

-> NumberFormat

Columns(“A:A”).NumberFormat = “0.00%”

Métodos Úteis:

-> Select Use o método Select para selecionar uma faixa de células.

Antes de selecionar uma faixa, geralmente é uma boa ideia usar uma declaração adicional para garantir que a planilha certa esteja ativa. Por exemplo, se Sheet1 contém a faixa que você deseja selecionar, use as seguintes declarações para selecionar a faixa:

```
Sheets("Sheet1").Activate
Range("A1:C12").Select
```

9.4. Performance

Na execução da Macro desligar a atualização da tela:

```
Application.ScreenUpdating = False

... códigos ....

Application.ScreenUpdating = True
```

10. Referências

https://docs.microsoft.com/pt-br/office/vba/library-reference/concepts/getting-started-with-vba-in-office