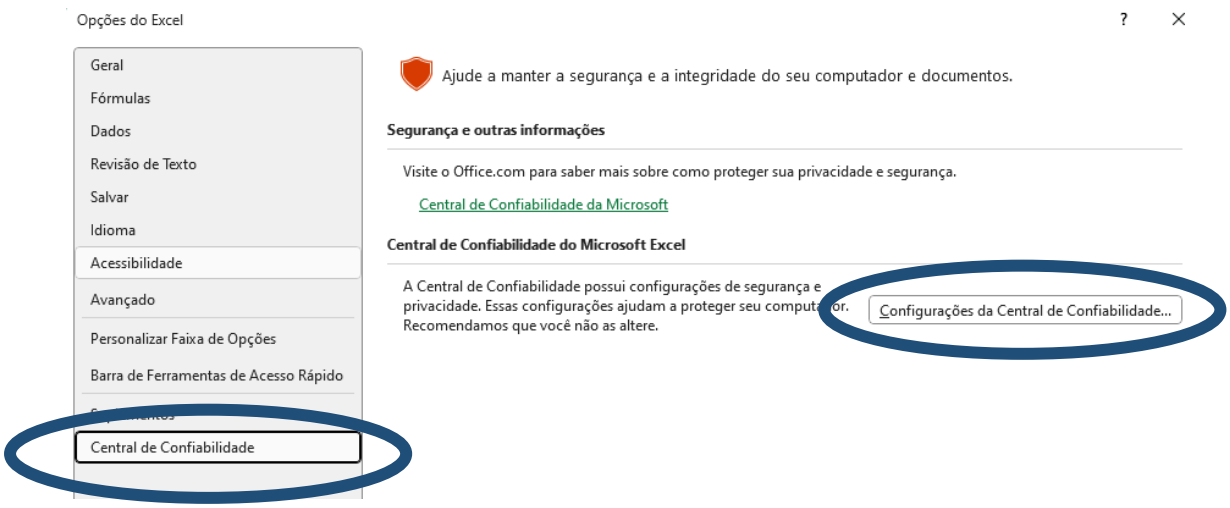
Introdução ao VBA

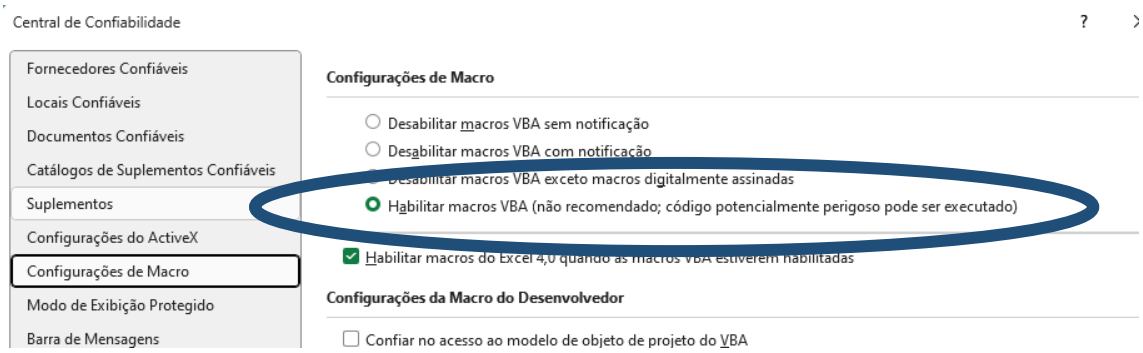
Habilitando macros no seu computador:

a. As macros devem ser gravadas com a extensão XLSM (e não xlsx, que é a padrão). Se não gravar como xlsm, você vai perder as macros gravadas!
b. O STJ não deixa rodar macros nas máquinas do Tribunal. A única forma é gravando as planilhas no OneDrive. Habilite seu OneDrive, faça uma nova pasta e grave seus arquivos lá.

c. O Excel vem por padrão com segurança contra macros.
    i. Selecione a guia Arquivo e escolha Opções.
    ii. Selecione Central de Confiança e escolha Configurações da Central de Confiança.



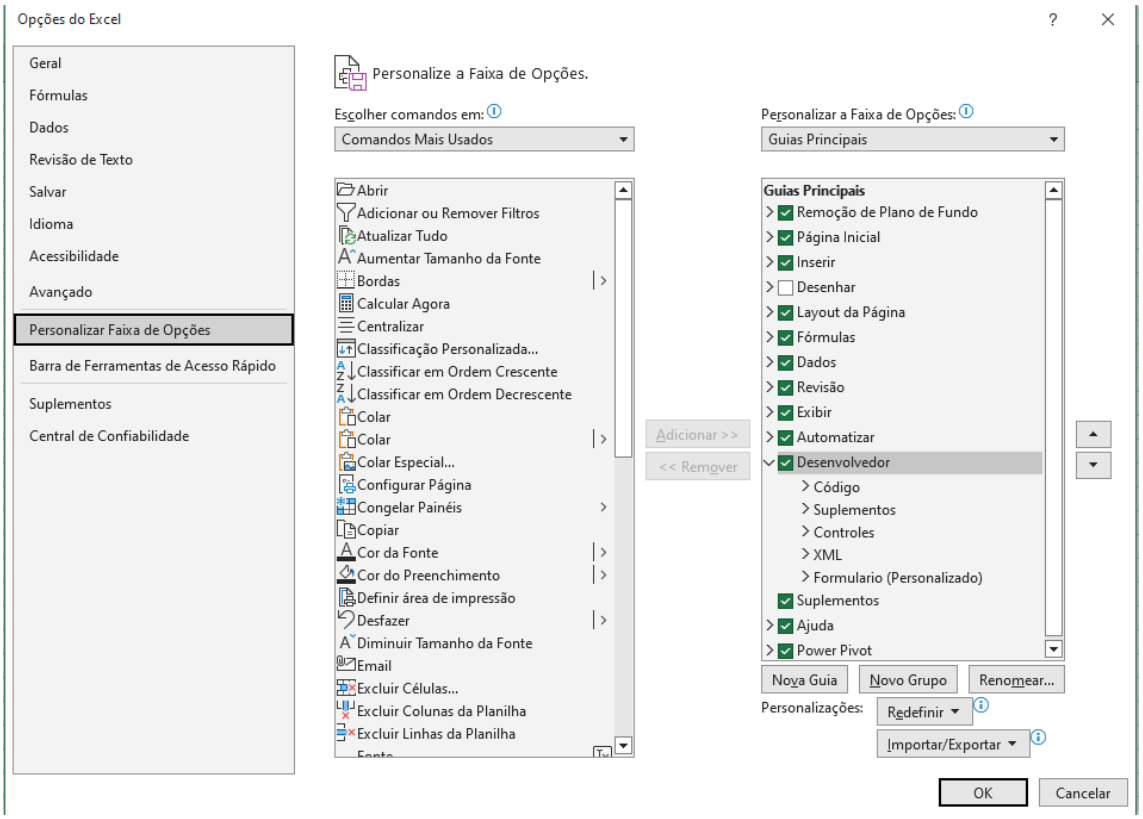
iii. Na Central de Confiança, selecione Configurações de Macro. ...

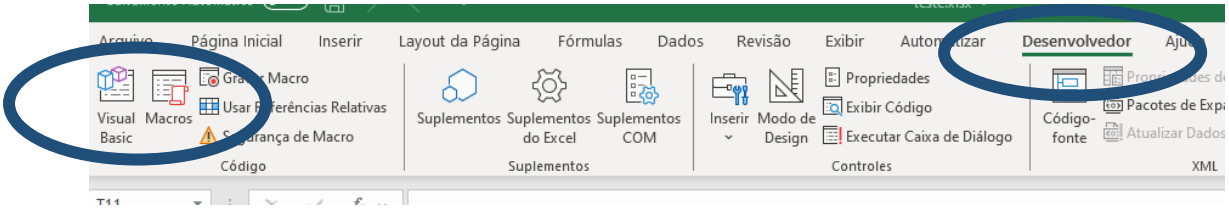


iv. Selecione OK e abra novamente o arquivo.

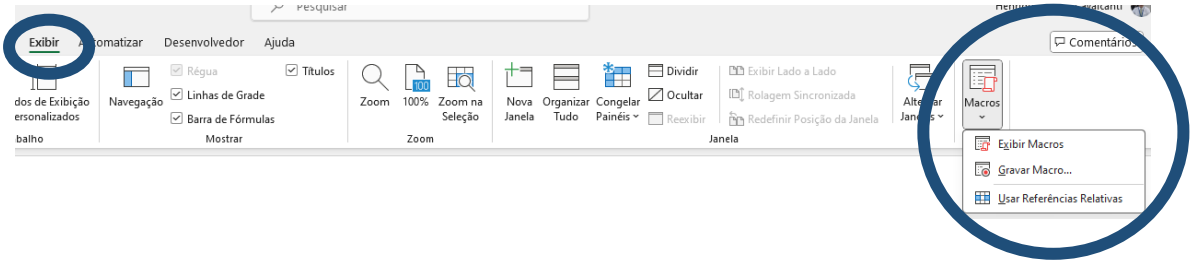
Fundamentos de Macros – Aba desenvolvedor

2. Arquivo -> Opções:
   Personalizar Faixa de opções -> Desenvolvedor



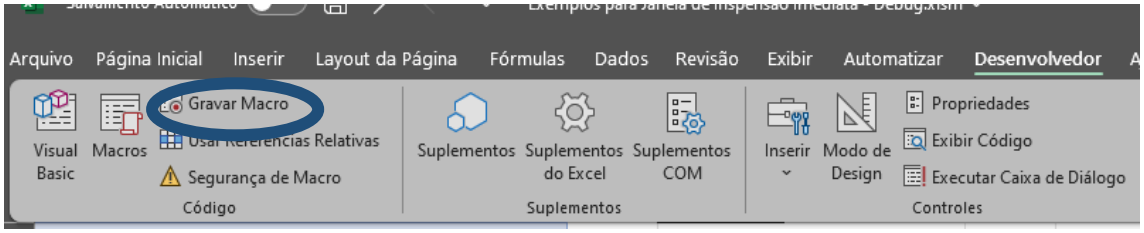


3. Se não quiser usar a aba de Desenvolvedor, podemos encontrar os principais ícones para desenvolvimento em Exibir -> Macros



Gravar Primeira Macro

1. Na guia **Desenvolvedor**, clique em **Gravar Macro**.



2. Dê um nome à sua macro. Nomes de macro devem começar com uma letra e não podem conter espaços. Você pode também adicionar uma tecla de atalho e escolher onde a macro será armazenada (na pasta de trabalho atual ou em outra localização).
3. Clique em **OK** para começar a gravação.
4. Execute as ações que deseja automatizar. Tudo o que fizer, como digitar texto em células, formatar planilhas ou inserir fórmulas, será gravado.
5. Quando terminar, volte à guia **Desenvolvedor** e clique em **Parar Gravação**.

Executando Macros Gravadas

Depois de gravar uma macro, você pode executá-la sempre que precisar repetir a sequência de ações que gravou.

Para executar uma macro:

1. Na guia **Desenvolvedor**, clique em **Macros**.
2. Na janela que aparece, selecione a macro que deseja executar.
3. Clique em **Executar**.

Dica: Se você atribuiu uma tecla de atalho à sua macro durante a gravação, pode simplesmente pressionar essa tecla para executar a macro.

Editando Macros no Editor VBA

As macros gravadas podem ser editadas para ajustá-las ou expandir suas funcionalidades. O código gerado pela gravação é um bom ponto de partida, mas muitas vezes você precisará ajustá-lo para otimizar a execução ou para adicionar funcionalidades adicionais.

Como editar uma macro:

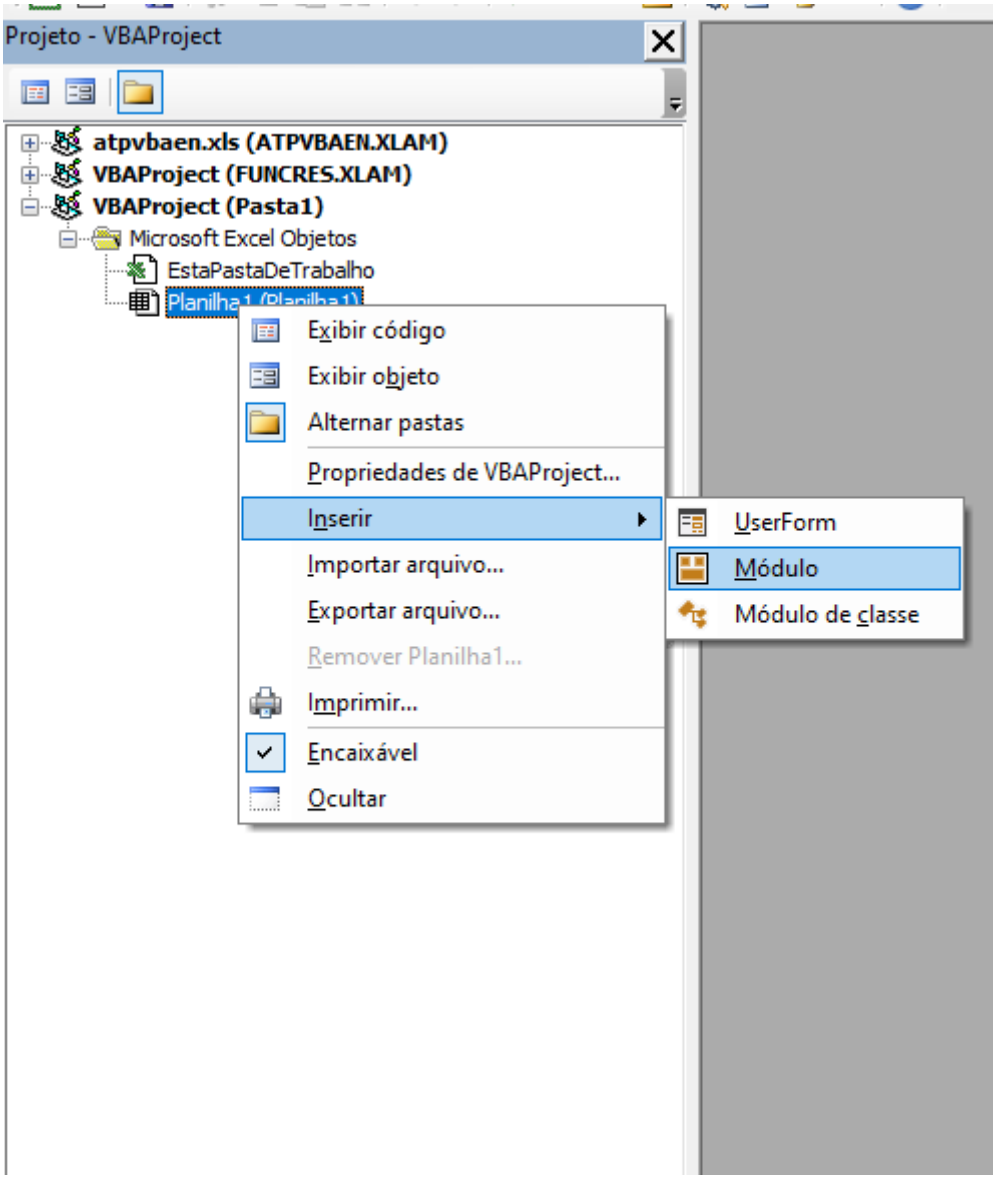
1. Abra o Editor VBA pressionando Alt + F11.
2. No Editor VBA, localize a macro que deseja editar no **Explorador de Projeto**.
3. Dê um duplo clique no módulo onde a macro está armazenada para abrir o código.
4. Edite o código conforme necessário. Por exemplo, você pode alterar valores, adicionar loops, ou incluir condições para tornar a macro mais flexível.

Visão Geral do Editor VBA

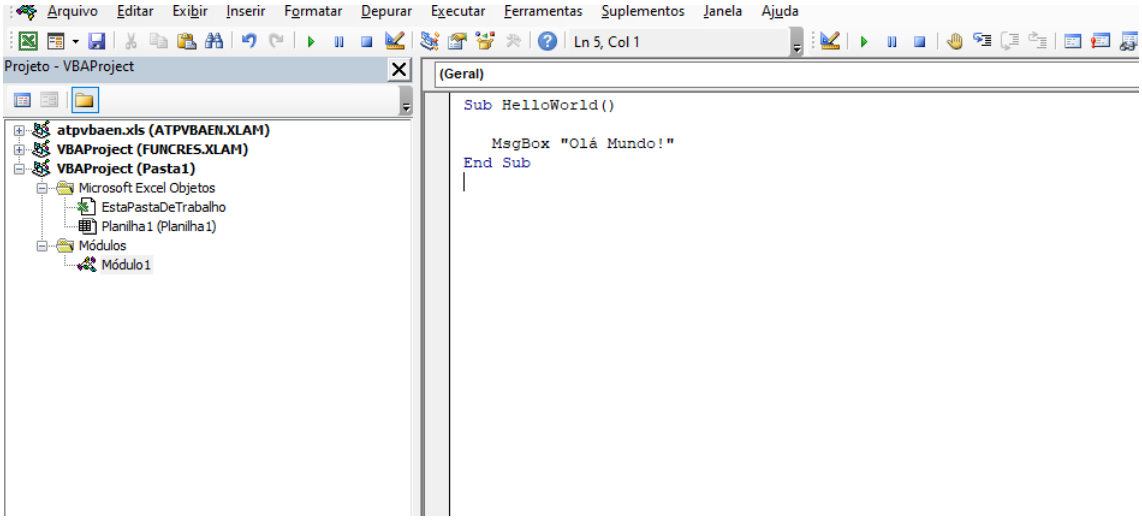
O Editor VBA é o ambiente onde o código VBA é escrito e gerenciado. Ele pode ser acessado diretamente pela guia Desenvolvedor ou pressionando Alt + F11 no Excel.

Componentes principais do Editor VBA:

- **Janela de Projeto:** Mostra a estrutura dos seus projetos VBA, incluindo módulos, formulários e objetos de planilhas.



- **Janela de Código:** Área onde você escreve e edita o código VBA.



Diferença entre Sub e Function

Em VBA, a estrutura básica do código é composta por **Subrotinas** (Sub) e **Funções** (Function). Ambas são blocos de código que realizam tarefas específicas, mas há diferenças fundamentais entre elas.

- **Subrotinas (Sub):** São usadas para realizar ações, como modificar planilhas ou manipular dados. Elas não retornam valores, e o código dentro de uma subrotina é executado diretamente quando ela é chamada.- esses são comentários (não rodam nada – sempre que tiver em vazio e depois um apóstrofe
    - **Sintaxe:**

```
Sub NomeDaSubrotina()

    'Código da Subrotina

End Sub
```

- **Funções (Function):** São usadas para realizar cálculos ou operações que retornam um valor. Ao contrário das subrotinas, as funções podem ser usadas diretamente em fórmulas do Excel, como qualquer outra função embutida.
    - **Sintaxe:**

```
 Function Dobro(Numero As Double) As Double

    Dobro = Numero * 2

End Function
```

Criando e Utilizando Subrotinas

Subrotinas são o alicerce das macros no VBA. Elas permitem automatizar tarefas repetitivas e organizar o código em blocos lógicos.

Exemplo de Subrotina:

```
Sub OlaMundo()

    MsgBox "Olá! Bem-vindo ao VBA!"

End Sub
```

Essa subrotina exibe uma caixa de mensagem com uma saudação. Você pode executar essa subrotina chamando-a pelo seu nome, como OlaMundo.

Funções Personalizadas no Excel

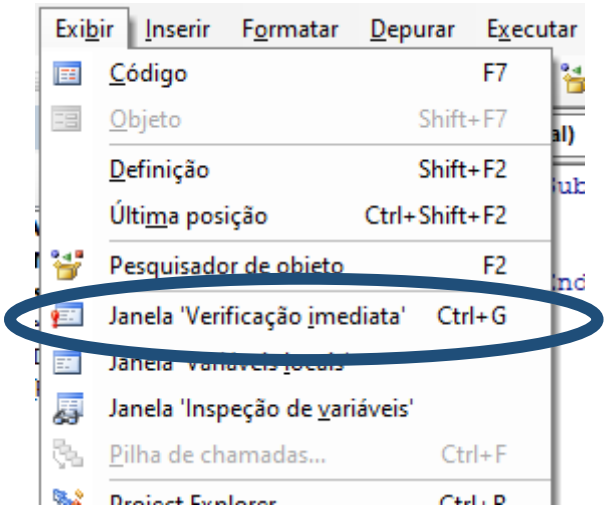
As funções personalizadas permitem expandir as capacidades do Excel, criando cálculos que não são cobertos pelas funções padrão.

```
Function Dobro(Numero As Double) As Double

    Dobro = Numero * 2

End Function
```

Essa função recebe um número como argumento e retorna o dobro desse valor. Você pode usar essa função em uma célula do Excel como qualquer outra função, por exemplo, =Dobro(A1).

```
Function Desconto(Preco As Double, Percentual As Double) As Double

    Desconto = Preco - (Preco * Percentual / 100)

End Function
```

Inspeção de Variáveis e Verificação Imediata



A **Inspeção de Variáveis** no VBA é uma técnica essencial para o desenvolvimento e depuração de macros. Ela permite ao desenvolvedor monitorar o valor de uma ou mais variáveis enquanto o código está sendo executado, facilitando a identificação de erros e a compreensão do comportamento do código.

Como funciona:

- Durante a execução do código, você pode pausar a execução em um determinado ponto (usando um breakpoint, por exemplo) e inspecionar as variáveis.
- A janela de **Inspeção de Variáveis** no Editor VBA exibe uma lista de todas as variáveis ativas, mostrando seus valores atuais e permitindo que você observe como esses valores mudam à medida que o código avança.
- Para inspecionar uma variável específica, clique com o botão direito sobre a variável no código e selecione "Adicionar à Inspeção". A variável será adicionada à janela de inspeção e seu valor será atualizado em tempo real.

Por que usar:

- **Depuração:** Identifique rapidamente valores inesperados ou incorretos nas variáveis, o que pode ajudar a localizar a causa de erros no código.
- **Monitoramento:** Acompanhe o comportamento de variáveis ao longo da execução de loops ou outras estruturas de controle para garantir que estejam funcionando conforme o esperado.

A **Verificação Imediata** é uma ferramenta poderosa no Editor VBA que permite ao desenvolvedor executar comandos VBA, testar partes do código, e verificar valores de variáveis instantaneamente, sem precisar rodar a macro completa. A verificação é feita através da **Janela de Imediato**.

Como funciona:

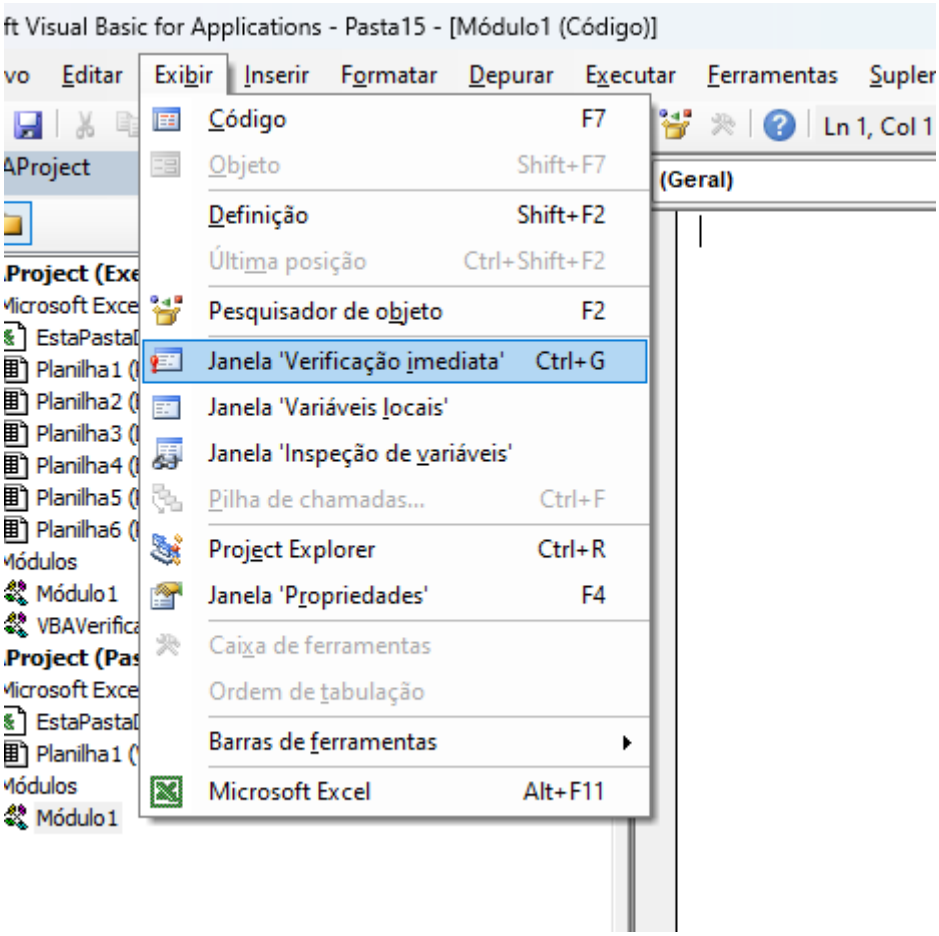
- A **Janela de Imediato** pode ser aberta no Editor VBA pressionando Ctrl + G ou acessando-a pelo menu Exibir > Imediato.

- Nessa janela, você pode digitar comandos VBA ou expressões para serem executadas imediatamente.
- É possível avaliar o valor de uma variável ou executar uma linha de código diretamente, o que é extremamente útil para testar hipóteses e verificar rapidamente o comportamento do código.
- Por exemplo, você pode digitar ? nomeVariavel na Janela de Imediato para ver o valor atual de nomeVariavel.

Por que usar:

- **Teste Rápido:** Teste fragmentos de código e veja o resultado instantaneamente sem precisar executar a macro inteira.
- **Verificação de Valores:** Verifique o valor de uma variável ou o resultado de uma função durante o processo de depuração.
- **Exploração de Código:** Execute e teste pequenas partes do código para entender como ele funciona ou para verificar o efeito de modificações.

1. Para treinar vamos abrir o arquivo da Verificação Imediata e gravá-lo em uma pasta do One Drive. Isso é muito importante, pois no STJ é bloqueado o uso de arquivos xlsm e apenas conseguiremos essa liberação se gravarmos o arquivo na nuvem: Exemplos para Janela de Verificação Imediata - Debug.xlsm
2. Abra o Editor do VBA e abra a janela de Verificação Imediata



3. Vamos testar os seguintes códigos:

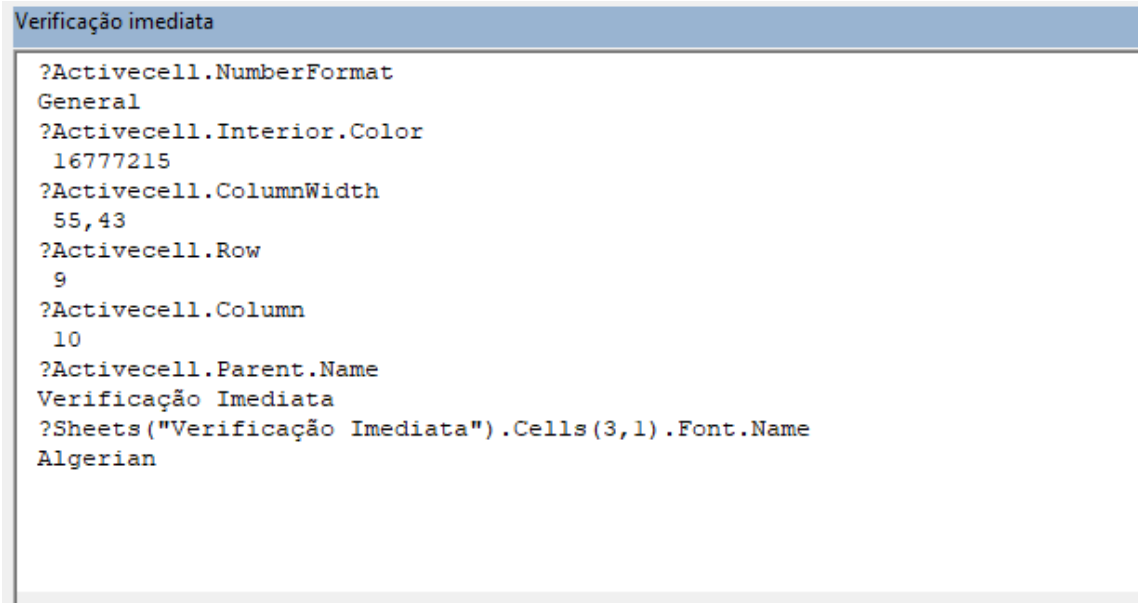
   Primeiro coloque Dados para testarmos a partir da célula A1

Célula A1
Minha Célula A2
R$ 32.131,00
1

   Agora vamos colar esses códigos na janela de verificação imediata e apertar enter para cada um deles.

Exemplos
?Range("A2").Value
?Activecell.NumberFormat
?Activecell.Interior.Color
?Activecell.ColumnWidth
?Activecell.Row
?Activecell.Column
?Activecell.Parent.Name
?Sheets("Verificação Imediata").Cells(1,3).Font.Name

4. Resultado Esperado:



5. Agora vamos testar alguns códigos que executaram uma ação na página:

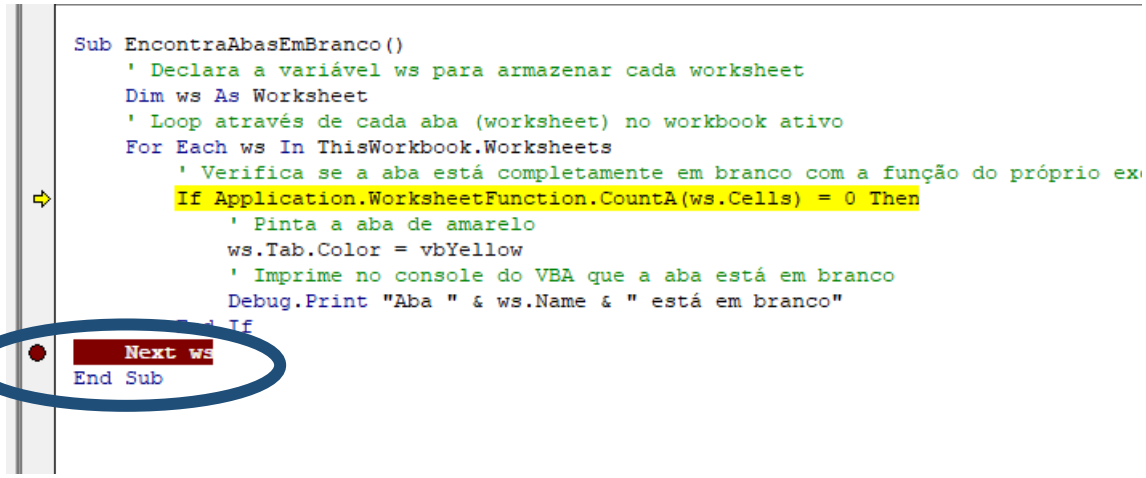
Exemplos de Códigos Executáveis
Selection.HorizontalAlignment = xlCenterAcrossSelection
Worksheets("Verificação Imediata").Visible = xlVeryHidden
Worksheets("Verificação Imediata").Visible = true
Columns("A:D").columnwidth = 5
ActiveSheet.Tab.Color = vbYellow
ActiveSheet.Tab.Color = xlColorIndexNone

6. Vamos criar algumas abas em branco para testar um código onde encontraremos todas as abas vazias. Vamos usar o Debug.Print para mostrar elas na janela de Verificação Imediata. Depois vamos pintá-las de amarelo:

```
Sub EncontraAbasEmBranco()

    ' Declara a variável ws para armazenar cada worksheet

    Dim ws As Worksheet

    ' Loop através de cada aba (worksheet) no workbook ativo

    For Each ws In ThisWorkbook.Worksheets

        ' Verifica se a aba está completamente em branco com a função do
próprio excel que conta células não vazias.

        If Application.WorksheetFunction.CountA(ws.Cells) = 0 Then

            ' Pinta a aba de amarelo
```

```
            ws.Tab.Color = vbYellow

            ' Imprime no console do VBA que a aba está em branco

            Debug.Print "Aba " & ws.Name & " está em branco"

        End If

    Next ws

End Sub
```

7. Vamos rodar esse código passo a passo para entendermos o que está acontecendo. Para isso vamos usar uma pausa no código.
   Clique na parte esquerda do código e marque uma pausa que estará indicada com o sinal vermelho



8. Para navegar pelo código utilizamos o **F5** para irmos até o próximo ponto de parada ou o **F8** para irmos para a próxima linha

Interação com o Usuário no VBA - MsgBox e InputBox.

1. Exibindo Mensagens com MsgBox:

```
Sub ExibeMensagem()

     MsgBox "Processo concluído com sucesso!", vbInformation,
"Confirmação"

End Sub
```

2. Capturando Entradas com InputBox

```
Sub CapturaEntrada()

    Dim nomeUsuario As String

    nomeUsuario = InputBox("Por favor, insira seu nome:", "Entrada de
Nome", "Usuário")

    MsgBox "Bem-vindo, " & nomeUsuario & "!", vbInformation, "Saudação"

End Sub
```

3. Interação Condicional com o Usuário

```
Sub ConfirmarAcao()

    Dim resposta As Integer

    resposta = MsgBox("Deseja continuar?", vbYesNo + vbQuestion,
"Confirmação")

    If resposta = vbYes Then

        MsgBox "Você escolheu continuar!", vbInformation, "Continuação"

        ' Adicione o código que deve ser executado em caso positivo

    Else

        MsgBox "Operação cancelada!", vbExclamation, "Cancelado"

        ' Adicione o código que deve ser executado em caso negativo

    End If

End Sub
```

Loops

1. For...Next

   For contador = valorInicial To valorFinal [incremento]
      ' Código a ser repetido
   Next contador

   contador: Variável usada para contar as iterações.
   valorInicial: O valor inicial do contador.
   valorFinal: O valor final do contador.

incremento (opcional): Valor pelo qual o contador será incrementado a cada iteração. Se omitido, o incremento padrão é 1.

```
Sub ExemploFor()

    Dim i As Integer

    For i = 1 To 5

        Cells(i, 1).Value = "Linha " & i

    Next i

End Sub
```

Se eu quiser o dobro da linha na coluna ao lado basta colocar o comando

```
 Cells(i, 2).Value = i * 2
```

2. For Each...Next
   O loop For Each...Next é usado para percorrer todos os itens em uma **coleção**, como todas as células em um intervalo ou todas as planilhas em um workbook.

   For Each item **In** coleção
   ' Código a ser repetido
   Next item

```
Sub ExemploForEach()

    Dim ws As Worksheet

    For Each ws In ThisWorkbook.Worksheets

        MsgBox "Aba: " & ws.Name

    Next ws

End Sub
```

3. Do While...Loop
   É útil quando você não sabe o número exato de iterações necessárias, mas sabe a condição que deve ser satisfeita para continuar o loop.

   Do While condição
   ' Código a ser repetido
   Loop.

```
Sub ExemploDoWhile()
```

```
‘Este exemplo preenche a coluna B com "Preenchido" enquanto a coluna A
‘contém valores, parando quando encontra uma célula vazia.

    Dim i As Integer

    i = 1

    Do While Cells(i, 1).Value <> ""

        Cells(i, 2).Value = "Preenchido"

        i = i + 1

    Loop

End Sub
```

4. If...Then...Else

```
If condição Then
   ' Código executado se a condição for verdadeira
ElseIf outraCondição Then
   ' Código executado se a outra condição for verdadeira
Else
   ' Código executado se nenhuma condição for verdadeira
End If
```

```
Sub ExemploIfThenElse()

    Dim valor As Integer

    valor = Range("A1").Value

    If valor > 10 Then

        MsgBox "O valor é maior que 10"

    ElseIf valor = 10 Then

        MsgBox "O valor é igual a 10"

    Else

        MsgBox "O valor é menor que 10"

    End If

End Sub
```

5. Combinação de Estruturas: For com If

```
If condição Then
    ' Código executado se a condição for verdadeira
ElseIf outraCondição Then
    ' Código executado se a outra condição for verdadeira
Else
    ' Código executado se nenhuma condição for verdadeira
End If
```

Esse exemplo abaixo irá preencher a coluna B com "Maior que 5" ou "Menor ou igual a 5", dependendo do valor na coluna A, para as primeiras 10 linhas.

Dessa forma, preencha valores na coluna A até a linha 10.

```
Sub ExemploForComIf()

    Dim i As Integer

    For i = 1 To 10

        If Cells(i, 1).Value > 5 Then

            Cells(i, 2).Value = "Maior que 5"

        Else

            Cells(i, 2).Value = "Menor ou igual a 5"

        End If

    Next i

End Sub
```

6. **Exercício: Encontrando Valores Específicos**
   Crie uma macro que percorre uma coluna e destaca todas as células do `Range("A1:A20")` que contêm um valor maior que 100. Use um loop For Each e uma estrutura If.

```
Sub DestacaValores()

    Dim cell As Range

    For Each cell In Range("A1:A20")

        If cell.Value > 100 Then

            cell.Interior.Color = vbYellow
```

```
        End If

    Next cell

End Sub
```

Criando Relatórios Simples

1. Para melhor desenvolvermos as macros abaixo, vamos popular de dados uma aba com dados fictícios. Criei duas macros para isso. A primeira é uma função que verificará se determinada planilha existe. Se ela existir, ativa a aba, senão ela cria. Grave as duas macros e rode a segunda (GerarDadosFicticios):

```
Sub DeletarECriarPlanilha(nomeAba As String)

    Dim ws As Worksheet

    Dim wsDados As Worksheet

    Dim planilhaExiste As Boolean

    ' Inicializa a variável que verifica se a planilha já existe

    planilhaExiste = False

    ' Loop para verificar se a planilha já existe

    For Each ws In ThisWorkbook.Sheets

        If ws.Name = nomeAba Then

            planilhaExiste = True

            Set wsDados = ws

            Exit For

        End If

    Next ws



    ' Se a planilha já existe, exclui e cria novamente; se não, cria a
nova planilha

    If planilhaExiste Then
```

```
        wsDados.Delete

    End If

        ' Se a planilha não existir, cria uma nova
    Set wsDados2 = ThisWorkbook.Sheets.Add
    wsDados2.Name = nomeAba

End Sub

Sub GerarDadosFicticios()
    Dim i As Integer
    Dim dataInicial As Date
    Dim valorVenda As Double
    Dim statusVenda As String
    Dim ws As Worksheet
    ' Cria uma nova planilha para os dados fictícios
    Call DeletarECriarPlanilha("Dados Fictícios")
    Set ws = Sheets("Dados Fictícios")
    ' Adiciona os cabeçalhos das colunas
    ws.Cells(1, 1).Value = "Data da Venda"
    ws.Cells(1, 2).Value = "Valor da Venda (R$)"
    ws.Cells(1, 3).Value = "Status"
    ' Define a data inicial para os exemplos
    dataInicial = Date - 60  ' Começa 60 dias atrás
```

```
    ' Gera 200 linhas de dados fictícios

    For i = 2 To 201

        ' Data da venda: dias consecutivos a partir de dataInicial

        ws.Cells(i, 1).Value = dataInicial +
WorksheetFunction.RandBetween(0, 59)

        ' Valor da venda: valor aleatório entre R$ 100,00 e R$ 5.000,00

        valorVenda = WorksheetFunction.RandBetween(1, 5000)

        ws.Cells(i, 2).Value = valorVenda

        ' Status da venda: "Concluído", "Cancelado" ou "Pendente"
aleatoriamente

        Select Case WorksheetFunction.RandBetween(1, 3)

            Case 1

                statusVenda = "Concluído"

            Case 2

                statusVenda = "Cancelado"

            Case 3

                statusVenda = "Pendente"

        End Select

        ws.Cells(i, 3).Value = statusVenda

    Next i

    ' Ajusta a largura das colunas para melhor visualização

    ws.Columns("A:C").AutoFit

‘Cria uma tabela de nome TabFicticio

    ws.ListObjects.Add(xlSrcRange, Range("$A$1:$C$201"), , xlYes).Name = _

        "TabFicticio"

    MsgBox "Dados fictícios gerados com sucesso na planilha 'Dados
Fictícios'.", vbInformation, "Concluído"

End Sub
```

2. Combinando Filtragem, Ordenação e Remoção

```
Sub GerarConsolidacaoSimples()

    Dim wsDados As Worksheet

    Dim wsRelatorio As Worksheet

    Dim ws As Worksheet

    Dim relatorioExiste As Boolean

    Dim ultimaLinha As Long

    Dim totalVendas As Double

    Dim totalConcluido As Double

    Dim totalCancelado As Double


    ' Define a planilha de dados

    Set wsDados = Sheets("Dados Fictícios")


    ' Verifica se a planilha "Relatório Simples" já existe

    relatorioExiste = False

    For Each ws In ThisWorkbook.Sheets

        If ws.Name = "Relatório Simples" Then

            relatorioExiste = True

            Set wsRelatorio = ws

            Exit For

        End If

    Next ws


    ' Se a planilha já existe, exclui e cria novamente; se não, cria a
nova planilha

    If relatorioExiste Then
```

```
        Application.DisplayAlerts = False

        wsRelatorio.Delete

        Application.DisplayAlerts = True

    End If



    Set wsRelatorio = ThisWorkbook.Sheets.Add

    wsRelatorio.Name = "Relatório Simples"

    ‘ Posso trocar por:

     ‘ Call DeletarECriarPlanilha(“Relatório Simples”)

    ' Adiciona cabeçalhos ao relatório

    wsRelatorio.Cells(1, 1).Value = "Resumo de Vendas"

    wsRelatorio.Cells(2, 1).Value = "Total de Vendas"

    wsRelatorio.Cells(3, 1).Value = "Total Concluído"

    wsRelatorio.Cells(4, 1).Value = "Total Cancelado"



    ' Encontra a última linha de dados na planilha de vendas

    ultimaLinha = wsDados.Cells(wsDados.Rows.Count, 1).End(xlUp).Row



    ' Calcula os totais

    totalVendas = Application.WorksheetFunction.Sum(wsDados.Range("B2:B" &
ultimaLinha))

    totalConcluido =
Application.WorksheetFunction.SumIf(wsDados.Range("C2:C" & ultimaLinha),
"Concluído", wsDados.Range("B2:B" & ultimaLinha))

    totalCancelado =
Application.WorksheetFunction.SumIf(wsDados.Range("C2:C" & ultimaLinha),
"Cancelado", wsDados.Range("B2:B" & ultimaLinha))



    ' Preenche o relatório com os totais
```

```
    wsRelatorio.Cells(2, 2).Value = totalVendas

    wsRelatorio.Cells(3, 2).Value = totalConcluido

    wsRelatorio.Cells(4, 2).Value = totalCancelado



    ' Ajusta a largura das colunas

    wsRelatorio.Columns("A:B").AutoFit



    MsgBox "Relatório Simples gerado com sucesso na planilha 'Relatório
Simples'.", vbInformation, "Concluído"

End Sub
```

1. Planilha Dinâmica

```
Sub GerarRelatorioTabelaDinamica()

    Dim wsDados As Worksheet

    Dim wsRelatorio As Worksheet

    Dim ptCache As PivotCache

    Dim ptTabela As PivotTable

    Dim rngDados As Range



    ' Define a planilha de dados e cria uma nova planilha para o relatório

    Set wsDados = Sheets("Dados Fictícios")

    Set wsRelatorio = ThisWorkbook.Sheets.Add

    wsRelatorio.Name = "Relatório Tabela Dinâmica"



    ' Define o intervalo de dados

    Set rngDados = wsDados.Range("A1").CurrentRegion
```

```
    ' Cria o cache da Tabela Dinâmica

    Set ptCache = ThisWorkbook.PivotCaches.Create(xlDatabase, rngDados)



    ' Cria a Tabela Dinâmica na nova planilha

    Set ptTabela = ptCache.CreatePivotTable(wsRelatorio.Cells(1, 1),
"TabelaDinâmica")



    ' Configura a Tabela Dinâmica

    With ptTabela

        ' Configura os campos de linhas e colunas

        .PivotFields("Status").Orientation = xlRowField

        .PivotFields("Data da Venda").Orientation = xlRowField



        ' Adiciona o campo "Valor da Venda (R$)" na área de valores

        With .PivotFields("Valor da Venda (R$)")

            .Orientation = xlDataField

            .Function = xlSum

            .NumberFormat = "#,##0.00"

        End With

    End With



    MsgBox "Relatório com Tabela Dinâmica gerado com sucesso na planilha
'Relatório Tabela Dinâmica'.", vbInformation, "Concluído"

End Sub
```

Automatizando Word a Partir do Excel

1. Você pode usar o VBA no Excel para automatizar a criação de documentos no Word. Por exemplo, você pode gerar relatórios em Word a partir de dados contidos em uma planilha do Excel.
   Vamos gerar um arquivo chamado `RelatórioVendas.docx`. Ele vai conter uma lista com os dados da tabela Dados Fictícios:

```
Sub GerarRelatorioWord()

    Dim wordApp As Object

    Dim wordDoc As Object

    Dim wsDados As Worksheet

    Dim ultimaLinha As Long

    Dim i As Integer

    Dim relatorioTexto As String


    ' Define a planilha de dados

    Set wsDados = Sheets("Dados Fictícios")


    ' Encontra a última linha de dados na planilha

    ultimaLinha = wsDados.Cells(wsDados.Rows.Count, 1).End(xlUp).Row


    ' Cria uma nova instância do Word

    Set wordApp = CreateObject("Word.Application")

    wordApp.Visible = True ' Define o Word como visível para o usuário


    ' Cria um novo documento no Word

    Set wordDoc = wordApp.Documents.Add


    ' Gera o relatório com base nos dados do Excel

    For i = 2 To ultimaLinha
```

```
        relatorioTexto = relatorioTexto & "Data da Venda: " &
wsDados.Cells(i, 1).Value & vbCrLf

        relatorioTexto = relatorioTexto & "Valor da Venda: R$ " &
Format(wsDados.Cells(i, 2).Value, "#,##0.00") & vbCrLf

        relatorioTexto = relatorioTexto & "Status: " & wsDados.Cells(i,
3).Value & vbCrLf

        relatorioTexto = relatorioTexto & String(40, "-") & vbCrLf

    Next i



    ' Insere o texto no documento Word

    wordDoc.Content.Text = "Relatório de Vendas" & vbCrLf & vbCrLf &
relatorioTexto



    ' Salva e fecha o documento Word

    wordDoc.SaveAs ThisWorkbook.Path & "\RelatórioVendas.docx"

    MsgBox "Relatório gerado no Word e salvo como
'RelatórioVendas.docx'.", vbInformation, "Concluído"

End Sub
```

Explicação do Código:

- **Word.Application:** Cria uma instância do Word e a torna visível para o usuário.
- **Document.Add:** Cria um novo documento.
- **Loop de Dados:** Itera sobre os dados do Excel e os formata como texto para inserir no documento Word.
- **Salvar Documento:** O documento é salvo no mesmo diretório do arquivo Excel.

2. Automatizando Emails no Outlook

```
Sub EnviarEmailComOutlook()

    Dim outlookApp As Object

    Dim outlookMail As Object

    Dim wsDados As Worksheet

    Dim ultimaLinha As Long
```

```
    Dim i As Integer

    Dim corpoEmail As String



    ' Define a planilha de dados

    Set wsDados = Sheets("Dados Fictícios")



    ' Encontra a última linha de dados na planilha

    ultimaLinha = wsDados.Cells(wsDados.Rows.Count, 1).End(xlUp).Row



    ' Cria uma nova instância do Outlook

    Set outlookApp = CreateObject("Outlook.Application")

    Set outlookMail = outlookApp.CreateItem(0) ' Cria um novo email



    ' Gera o corpo do email com base nos dados do Excel

    corpoEmail = "Relatório de Vendas" & vbCrLf & vbCrLf

    For i = 2 To ultimaLinha

        corpoEmail = corpoEmail & "Data da Venda: " & wsDados.Cells(i,
1).Value & vbCrLf

        corpoEmail = corpoEmail & "Valor da Venda: R$ " &
Format(wsDados.Cells(i, 2).Value, "#,##0.00") & vbCrLf

        corpoEmail = corpoEmail & "Status: " & wsDados.Cells(i, 3).Value &
vbCrLf

        corpoEmail = corpoEmail & String(40, "-") & vbCrLf

    Next i



    ' Configura o email

    With outlookMail

        .To = "destinatario@exemplo.com" ' Altere para o endereço de email
desejado
```

```
        .Subject = "Relatório de Vendas"
        .Body = corpoEmail
        .Display ' Exibe o email para revisão antes de enviar
    End With

    ' Envia o email
    ' .Send ' Descomente esta linha para enviar o email automaticamente

    MsgBox "Email gerado com sucesso.", vbInformation, "Concluído"
End Sub
```

Explicação do Código:

- **Outlook.Application:** Cria uma instância do Outlook e configura um novo email.
- **Corpo do Email:** Formata os dados do Excel e insere como o corpo do email.
- **Envio do Email:** O email é exibido para revisão e pode ser enviado automaticamente.

3. Automatizando PowerPoint

```
Sub CriarApresentacaoPowerPoint()
    Dim pptApp As Object
    Dim pptPres As Object
    Dim pptSlide As Object
    Dim wsDados As Worksheet
    Dim ultimaLinha As Long
    Dim i As Integer

    ' Define a planilha de dados
    Set wsDados = Sheets("Dados Fictícios")
```

```
    ' Encontra a última linha de dados na planilha
    ultimaLinha = wsDados.Cells(wsDados.Rows.Count, 1).End(xlUp).Row

    ' Cria uma nova instância do PowerPoint
    Set pptApp = CreateObject("PowerPoint.Application")
    pptApp.Visible = True ' Torna o PowerPoint visível

    ' Cria uma nova apresentação
    Set pptPres = pptApp.Presentations.Add

    ' Adiciona um slide de título
    Set pptSlide = pptPres.Slides.Add(1, 1) ' 1 = Layout de Título
    pptSlide.Shapes(1).TextFrame.TextRange.Text = "Relatório de Vendas"
    pptSlide.Shapes(2).TextFrame.TextRange.Text = "Gerado a partir do
Excel"

    ' Adiciona slides com os dados
    For i = 2 To ultimaLinha
        Set pptSlide = pptPres.Slides.Add(i, 2) ' 2 = Layout de Título e
Conteúdo
        pptSlide.Shapes(1).TextFrame.TextRange.Text = "Venda " & i - 1
        pptSlide.Shapes(2).TextFrame.TextRange.Text = "Data: " &
wsDados.Cells(i, 1).Value & vbCrLf & _
                                                      "Valor: R$ " &
Format(wsDados.Cells(i, 2).Value, "#,##0.00") & vbCrLf & _
                                                      "Status: " &
wsDados.Cells(i, 3).Value
    Next i
```

```
    MsgBox "Apresentação criada com sucesso no PowerPoint.",
vbInformation, "Concluído"

End Sub
```

Explicação do Código:

- **PowerPoint.Application:** Cria uma instância do PowerPoint e uma nova apresentação.
- **Slides:** O código adiciona slides, configurando o título e o conteúdo com base nos dados do Excel.

Proteção e Segurança no VBA

1. Protegendo Planilhas e Pastas de Trabalho

```
Sub ProtegerPlanilha()

    Dim ws As Worksheet

    Set ws = ThisWorkbook.Sheets("Dados Fictícios")



    ' Protege a planilha com uma senha

    ws.Protect Password:="senhaSegura", AllowFiltering:=True,
AllowSorting:=True

    MsgBox "A planilha 'Dados Fictícios' foi protegida com sucesso.",
vbInformation, "Proteção Aplicada"

End Sub
```

Explicação:

- **ws.Protect:** Protege a planilha específica com uma senha.
- **AllowFiltering e AllowSorting:** Especifica que o usuário ainda pode filtrar e ordenar os dados, mesmo com a proteção ativada.

```
Sub DesprotegerPlanilha()

    Dim ws As Worksheet

    Set ws = ThisWorkbook.Sheets("Dados Fictícios")



    ' Desprotege a planilha
```

```
    ws.Unprotect Password:="senhaSegura"

    MsgBox "A planilha 'Dados Fictícios' foi desprotegida.",
vbInformation, "Proteção Removida"

End Sub
```

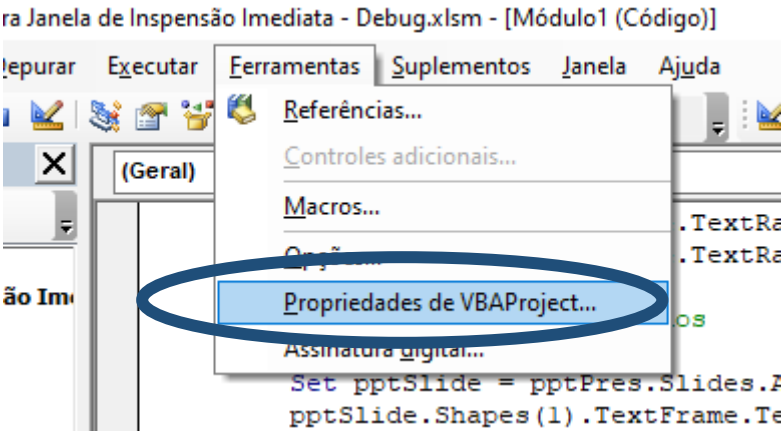
```
Sub ProtegerPastaDeTrabalho()

    ' Protege toda a pasta de trabalho

    ThisWorkbook.Protect Password:="senhaSegura"

    MsgBox "A pasta de trabalho foi protegida com sucesso.",
vbInformation, "Proteção Aplicada"

End Sub
```

Explicação:

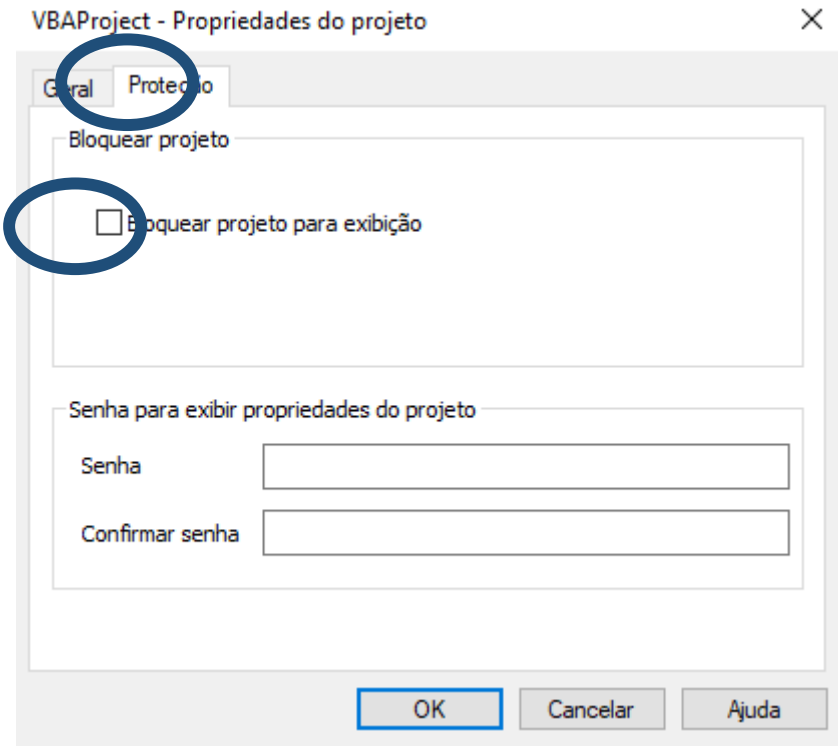
- **ThisWorkbook.Protect:** Protege a pasta de trabalho inteira com uma senha, impedindo alterações na estrutura (como adicionar ou excluir planilhas).

```
Sub DesprotegerPastaDeTrabalho()

    ' Desprotege toda a pasta de trabalho

    ThisWorkbook.Unprotect Password:="senhaSegura"

    MsgBox "A pasta de trabalho foi desprotegida.", vbInformation,
"Proteção Removida"

End Sub
```

2. Proteger Código VBA com Senha
   1.1. Abra o Editor VBA (Alt + F11).
   1.2. No menu, clique em Ferramentas > Propriedades do VBAProject.



1.3. Na aba Proteção, marque a opção Bloquear projeto para exibição.



1.4. Digite e confirme a senha.
1.5. Clique em OK e feche o Editor VBA.

3. Ocultando Planilhas com VBA

```
Sub OcultarPlanilha()

    Dim ws As Worksheet

    Set ws = ThisWorkbook.Sheets("Dados Fictícios")

    ' Oculta a planilha de forma que ela não possa ser desocultada via
interface do Excel

    ws.Visible = xlSheetVeryHidden
```

```
    MsgBox "A planilha 'Dados Fictícios' foi ocultada.", vbInformation,
"Planilha Ocultada"

End Sub
```

Explicação:

- **xlSheetVeryHidden:** Torna a planilha invisível e inacessível pelo menu de interface do Excel, diferente do **xlSheetHidden**, que permite ao usuário reexibi-la.

```
Sub ExibirPlanilhaOculta()

    Dim ws As Worksheet

    Set ws = ThisWorkbook.Sheets("Dados Fictícios")

    ' Exibe novamente a planilha que estava oculta

    ws.Visible = xlSheetVisible

    MsgBox "A planilha 'Dados Fictícios' foi exibida novamente.",
vbInformation, "Planilha Exibida"

End Sub
```

Depuração e Tratamento de Erros

1. Tratamento de Erros com **On Error**
   1.1. On Error Resume Next : Ignora o erro e continua a execução do código na próxima linha

```
Sub ExemploResumeNext()

    On Error Resume Next

    ' Tenta ativar uma planilha que pode não existir

    Sheets("Inexistente").Activate

    MsgBox "Continua mesmo com erro"

End Sub
```

   1.2. On Error GoTo: Redireciona a execução para uma rotina específica de tratamento de erros.

```
Sub ExemploGoTo()
```

```
    On Error GoTo TratarErro

    ' Tenta ativar uma planilha que pode não existir

    Sheets("Inexistente").Activate

    MsgBox "Isso não será exibido se houver um erro"

    Exit Sub


TratarErro:

    MsgBox "Erro encontrado: " & Err.Description

End Sub
```

1.3. On Error GoTo 0: Desativa o tratamento de erros, fazendo com que o VBA retorne ao modo padrão de tratamento (interrompe a execução em caso de erro).

```
Sub ExemploGoToZero()

    On Error Resume Next

    ' Código com potencial de erro

    Sheets("Inexistente").Activate

    On Error GoTo 0

    MsgBox "A partir daqui, erros serão tratados normalmente."

End Sub
```

1.4. Verificando Erros com Err: O objeto Err fornece informações sobre o erro que ocorreu. Ele pode ser usado para identificar o tipo de erro e tratar especificamente esses casos.

```
Sub ExemploErr()

    On Error GoTo TratarErro

    ' Código com potencial de erro

    Sheets("Inexistente").Activate

    Exit Sub
```

```
TratarErro:

    If Err.Number = 9 Then

        MsgBox "Erro: A planilha não existe. Código do erro: " &
Err.Number

    Else

        MsgBox "Erro desconhecido: " & Err.Description

    End If

End Sub
```

Propriedades Úteis do Objeto Err:

- **Err.Number**: Retorna o número do erro.
- **Err.Description**: Retorna uma descrição do erro.
- **Err.Source**: Identifica o nome do objeto que gerou o erro.
- **Err.Clear**: Limpa o objeto Err após o tratamento.

1.5. Para treinar, vamos fazer um exemplo de somar as células A1 de quatro abas. Três delas existentes e uma inexistente. Coloque valores na célula para as existentes e rode o código:

```
Sub SomaPlanilhasComErroTratado()

    Dim ws As Worksheet

    Dim total As Double

    Dim planilhas As Variant

    Dim i As Integer



    ' Lista de nomes das abas para somar valores

    planilhas = Array("Planilha1", "Planilha2", "PlanInexistente",
“Planilha3”)
```

```
    On Error GoTo TratarErro

    For i = LBound(planilhas) To UBound(planilhas)

        Set ws = Sheets(planilhas(i))

        total = total + ws.Range("A1").Value

    Next i



    MsgBox "A soma total é: " & total

    Exit Sub



TratarErro:

    MsgBox "Erro ao acessar a planilha: " & planilhas(i) & " - " &
Err.Description

    Resume Next

End Sub
```

<u>OBSERVAÇÃO</u>: **LBound** e **UBound**

- **LBound(planilhas):**
  - **Significado:** LBound (Lower Bound) retorna o menor índice do array planilhas, ou seja, o índice inicial.
  - **No VBA:** Arrays geralmente começam no índice 0. Portanto, LBound(planilhas) retorna 0.
- **UBound(planilhas):**
  - **Significado:** UBound (Upper Bound) retorna o maior índice do array planilhas, ou seja, o índice final.
  - **No VBA:** Para o array planilhas que contém 4 elementos (0 a 3), UBound(planilhas) retornará 3.

Iteração do Loop:

O loop For i = LBound(planilhas) To UBound(planilhas) faz o seguinte:

- **Itera sobre cada elemento do array planilhas:**
  - **Primeira Iteração (i = 0):** O código acessa planilhas(0), que é "Planilha1".
  - **Segunda Iteração (i = 1):** O código acessa planilhas(1), que é " Planilha2".
  - **Terceira Iteração (i = 2):** O código acessa planilhas(2), que é "PlanInexistente".
  - **Quarta Iteração (i = 3):** O código acessa planilhas(3), que é " Planilha3".

    Em cada iteração, o loop tenta definir a variável ws para a planilha correspondente, e em seguida soma o valor na célula A1 dessa planilha ao total.

Resumindo a Função do Código:

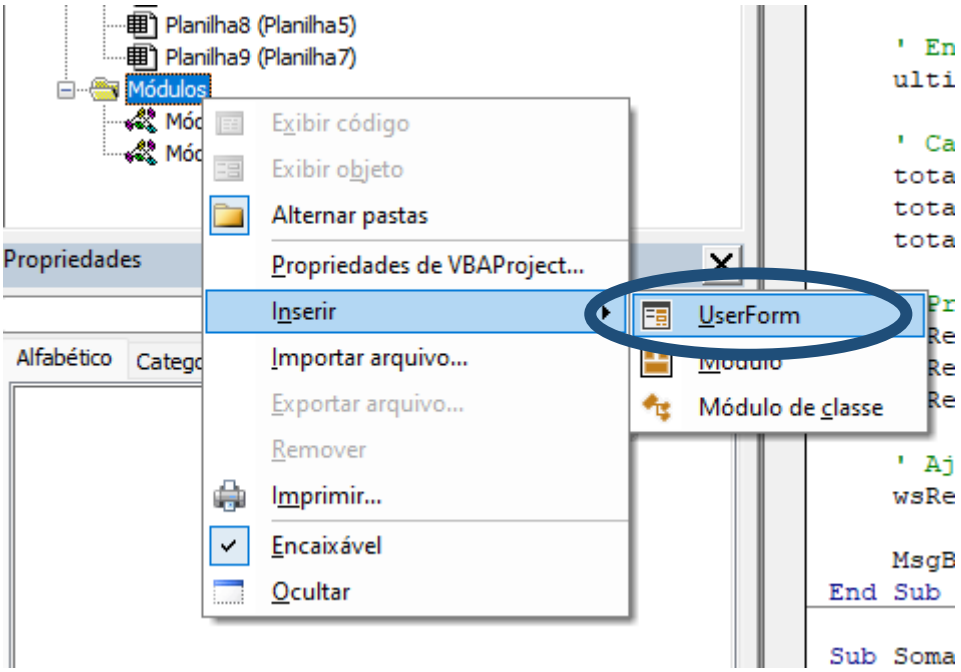
- **Iteração Completa:** O loop percorre cada planilha listada no array planilhas, da primeira (LBound(planilhas)) até a última (UBound(planilhas)).
- **Acesso aos Elementos:** Durante cada iteração, i é o índice atual que aponta para um elemento específico no array planilhas. Esse índice é utilizado para acessar o nome da planilha (planilhas(i)) e realizar operações sobre ela.
- **Flexibilidade:** Usar LBound e UBound permite que o código seja flexível. Se você alterar o número de elementos no array planilhas, o loop continuará funcionando corretamente, iterando através de todos os elementos do array, sem necessidade de alterar manualmente os valores de início e fim do loop.

Esse é um uso comum de LBound e UBound no VBA para percorrer todos os elementos de um array de maneira segura e eficiente, garantindo que todas as entradas do array sejam processadas.

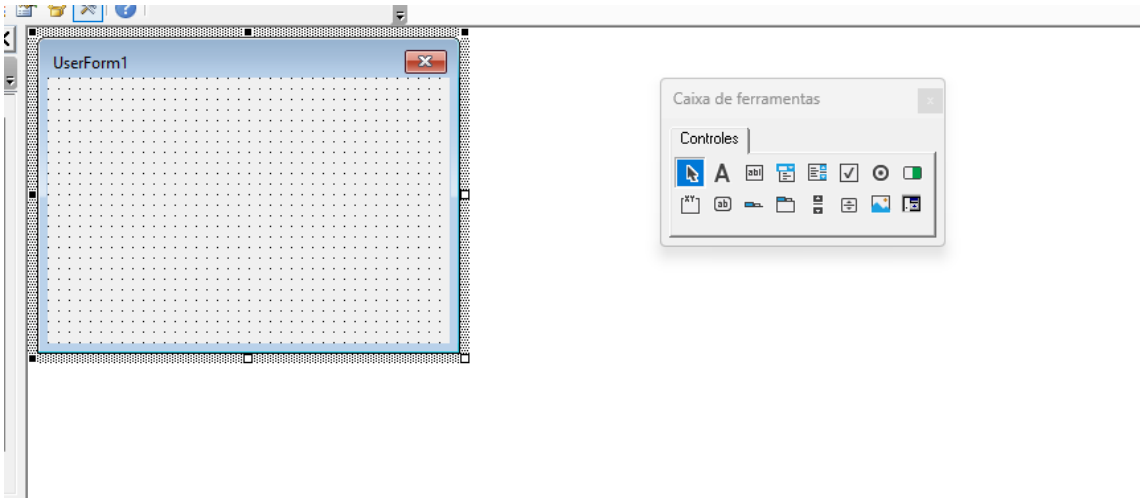
UserForms: Criando Interfaces de Usuário no VBA

1. Um **UserForm** é uma janela que pode conter vários controles, como caixas de texto, botões, caixas de listagem, entre outros. Você pode usar esses controles para capturar entradas do usuário, exibir informações ou permitir que o usuário interaja com os dados de maneira mais estruturada.
2. Como Criar um UserForm:

- No Editor VBA, clique em Inserir > UserForm.

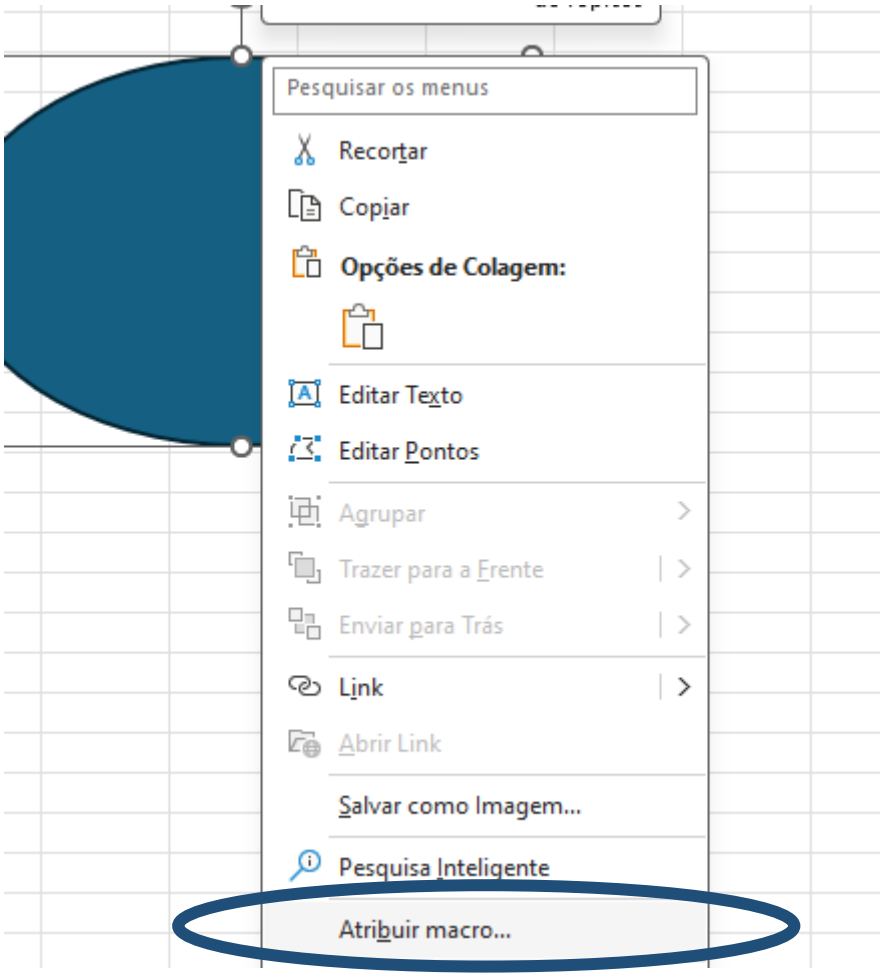


- Um novo UserForm será adicionado ao seu projeto, e você verá uma caixa de ferramentas com controles que podem ser adicionados ao formulário.

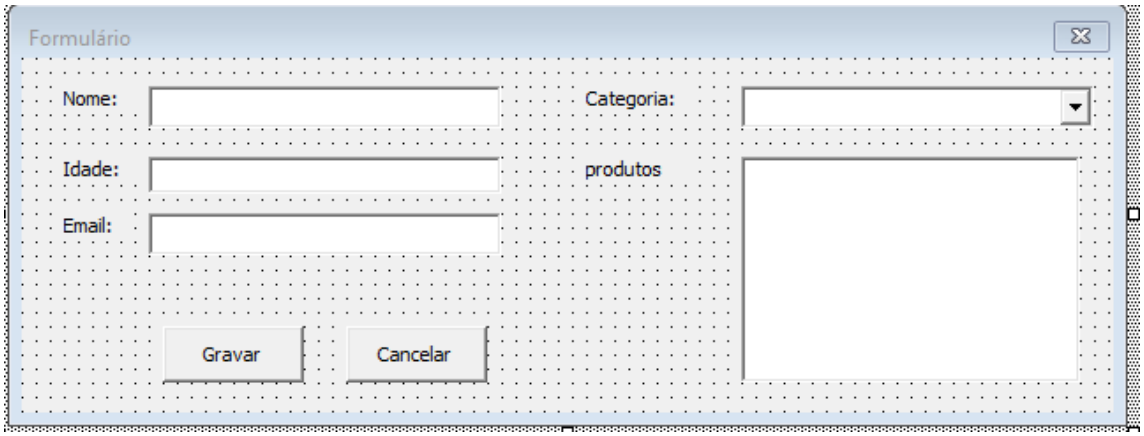


3. Em algum módulo, grave o código de abrir o UserForm1 criado. Para acionar, vamos inserir uma figura e atribuir a macro de abrir a ela:

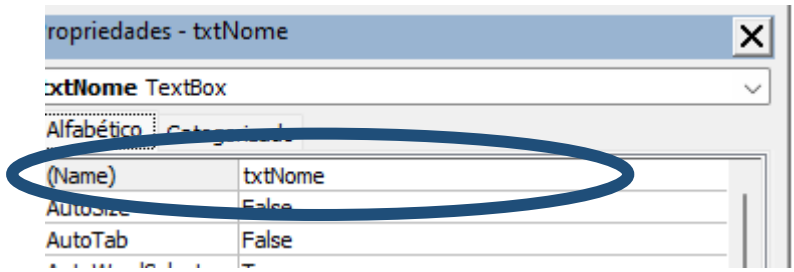
```
Sub MostrarUserForm()

    ' Exibe o UserForm chamado UserForm1

    UserForm1.Show

End Sub
```



4. Se precisar clicar no botão para alterá-lo, faça com o botão ALT.
5. Controles Básicos em **UserForms**
    - **Caixa de Texto (TextBox):** Permite ao usuário digitar texto ou números.
    - **Botão (CommandButton):** Executa uma ação quando clicado.
    - **Caixa de Listagem (ListBox):** Exibe uma lista de itens que o usuário pode selecionar.
    - **Caixa de Combinação (ComboBox):** Combina uma caixa de texto com uma lista suspensa.
    - **Rótulo (Label):** Exibe texto estático no UserForm.
    - **Caixa de Seleção (CheckBox):** Permite ao usuário fazer uma escolha binária (Sim ou Não).
    - **Botão de Opção (OptionButton):** Permite escolher uma opção em um grupo.
6. Vamos tentar montar um formulário parecido com o abaixo:



7. Serão Rótulos para os nomes. Serão caixas de texto para os 3 primeiros campos. Vamos mudar os nomes para: txtNome, txtIdade e txtEmail. Para a categoria vamos criar um caixa de combinação de nome cboCategoria. Depois a caixa de listagem de nome lstProdutos



8. Para os botões vamos criar um de gravar de nome btnGravar e outro de btnCancelar
9. Clique duas vezes sobre o formulário para abrir o código dele.
10. Vamos criar um código para popular as listas de categorias e produtos:

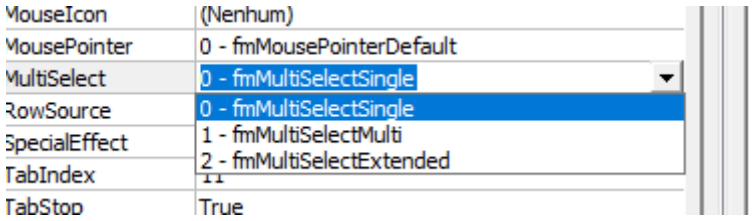
```
Private Sub UserForm_Initialize()

    ' Adiciona categorias à ComboBox

    Me.cboCategoria.AddItem "Eletrônicos"

    Me.cboCategoria.AddItem "Móveis"

    Me.cboCategoria.AddItem "Roupas"

End Sub
```

Perceba que tem um gancho para sempre que a Categoria mudar ele limpará a lista e a recriará de acordo com a escolha

```
Private Sub cboCategoria_Change()

    ' Limpa a ListBox antes de adicionar novos itens
```

```
    Me.lstProdutos.Clear


    ' Adiciona produtos à ListBox com base na seleção da categoria
    Select Case Me.cboCategoria.Value
        Case "Eletrônicos"
            Me.lstProdutos.AddItem "Televisão"
            Me.lstProdutos.AddItem "Smartphone"
            Me.lstProdutos.AddItem "Laptop"
        Case "Móveis"
            Me.lstProdutos.AddItem "Sofá"
            Me.lstProdutos.AddItem "Cama"
            Me.lstProdutos.AddItem "Mesa"
        Case "Roupas"
            Me.lstProdutos.AddItem "Camiseta"
            Me.lstProdutos.AddItem "Calça"
            Me.lstProdutos.AddItem "Jaqueta"
    End Select
End Sub
```

11. Se deseja escolher mais de um produto, mude essa propriedade:



Resumo das Diferenças:

- **fmMultiSelectSingle (0):** Apenas um item pode ser selecionado de cada vez.
- **fmMultiSelectMulti (1):** Vários itens podem ser selecionados
- **fmMultiSelectExtended (2):** Combina a funcionalidade de fmMultiSelectMulti, permitindo tanto seleções individuais (com Ctrl) quanto intervalos (com Shift).

12. Para o botão de cancelar vai esse código:

```
Private Sub btnCancelar_Click()

    ' Fecha o UserForm

    Unload Me

End Sub
```

13. Para salvar, vamos fazer uma validação dos campos inseridos e gravaremos na aba chamada Planilha2 o resultado. Para isso, crie uma tabela chamada TabUsuario na Planilha2 dessa forma:

Nome	Idade	Email	Categoria	Produtos
Henrique	42	hmuniz@stj.jus.br	Roupas	Camiseta

14. Agora, clique duas vezes no UserForm e coloque o código abaixo:

```
Private Sub btnGravar_Click()

    Dim ws As Worksheet

    Dim ultimaLinha As Long

    Dim i As Integer

    Dim produtosSelecionados As String



    ' Validação dos campos

    If Me.txtNome.Value = "" Then

        MsgBox "Por favor, preencha o campo de nome.", vbExclamation,
"Erro de Validação"

        Exit Sub

    End If



    If Not IsNumeric(Me.txtIdade.Value) Or Me.txtIdade.Value <= 0 Then
```

```
        MsgBox "Por favor, insira uma idade válida.", vbExclamation, "Erro
de Validação"

        Exit Sub

    End If



    If InStr(1, Me.txtEmail.Value, "@") = 0 Then

        MsgBox "Por favor, insira um e-mail válido.", vbExclamation, "Erro
de Validação"

        Exit Sub

    End If



    ' Exibe mensagem de boas-vindas

    MsgBox "Bem-vindo, " & Me.txtNome.Value & "! Sua idade é " &
Me.txtIdade.Value & " e seu e-mail é " & Me.txtEmail.Value, vbInformation,
"Registro Completo"



 '_____________________________ TERMINA
VALIDAÇÃO_____________________________________



    ' Define a Planilha2 como o local onde os dados serão salvos

    Set ws = ThisWorkbook.Sheets("Planilha2")



    ' Encontra a última linha com dados na tabela TabUsuario

    ultimaLinha = ws.ListObjects("TabUsuario").ListRows.Count +
ws.ListObjects("TabUsuario").Range.Row



    ' Concatena os valores selecionados na ListBox, separados por vírgula

    For i = 0 To lstProdutos.ListCount - 1

        If lstProdutos.Selected(i) Then

            If produtosSelecionados = "" Then
```

```
                produtosSelecionados = lstProdutos.List(i)

            Else

                produtosSelecionados = produtosSelecionados & ", " &
lstProdutos.List(i)

            End If

        End If

    Next i



    ' Salva os valores na tabela TabUsuario

    With ws.ListObjects("TabUsuario")

        .ListRows.Add

        .ListRows(.ListRows.Count).Range(1, 1).Value = Me.txtNome.Value

        .ListRows(.ListRows.Count).Range(1, 2).Value = Me.txtIdade.Value

        .ListRows(.ListRows.Count).Range(1, 3).Value = Me.txtEmail.Value

        .ListRows(.ListRows.Count).Range(1, 4).Value =
Me.cboCategoria.Value

        .ListRows(.ListRows.Count).Range(1, 5).Value =
produtosSelecionados

    End With



    ' Exibe uma mensagem de confirmação

    MsgBox "Dados gravados com sucesso!", vbInformation, "Confirmação"



    ' Limpa o formulário para novas entradas

    Me.txtNome.Value = ""

    Me.txtIdade.Value = ""

    Me.txtEmail.Value = ""

    Me.cboCategoria.Value = ""
```

```
    Me.lstProdutos.Clear

End Sub
```

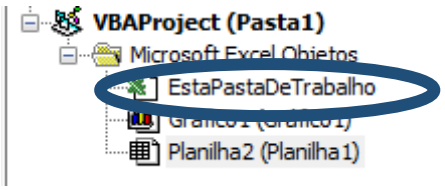
15. Para o botão cancelar, use esse código de fechar o UserForm:

```
Private Sub btnCancelar_Click()

    ' Fecha o UserForm

    Unload Me

End Sub
```

Executando Macros com Base em Eventos

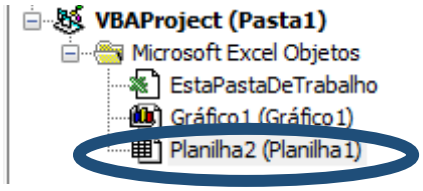
1. Tipos Comuns de Eventos:
   - **Eventos de Pasta de Trabalho:** Como abrir, fechar, salvar a pasta de trabalho.
   - **Eventos de Planilha:** Como mudanças em células, inserção de linhas/colunas, seleção de células.
   - **Eventos de Aplicativo:** Como novas pastas de trabalho sendo abertas, fechamento do Excel.

2. Macros Executadas na Abertura de uma Pasta de Trabalho:
   - Clique duas vezes sobre a EstaPastaDeTrabalho no Editor do VBA (Alt+F11)




```
Private Sub Workbook_Open()

    MsgBox "Bem-vindo ao Excel! Este aviso é gerado automaticamente ao
abrir a pasta de trabalho."

End Sub
```

3. Executando Macros Quando Células São Alteradas
    - Clique duas vezes sobre a Planilha na qual você deseja capturar o evendo no Editor do VBA (Alt+F11)




```
Private Sub Worksheet_Change(ByVal Target As Range)

    If Not Intersect(Target, Me.Range("A1")) Is Nothing Then

        MsgBox "Você alterou o conteúdo da célula A1."

    End If

End Sub
```

4. Executando Macros Quando Células São Alteradas

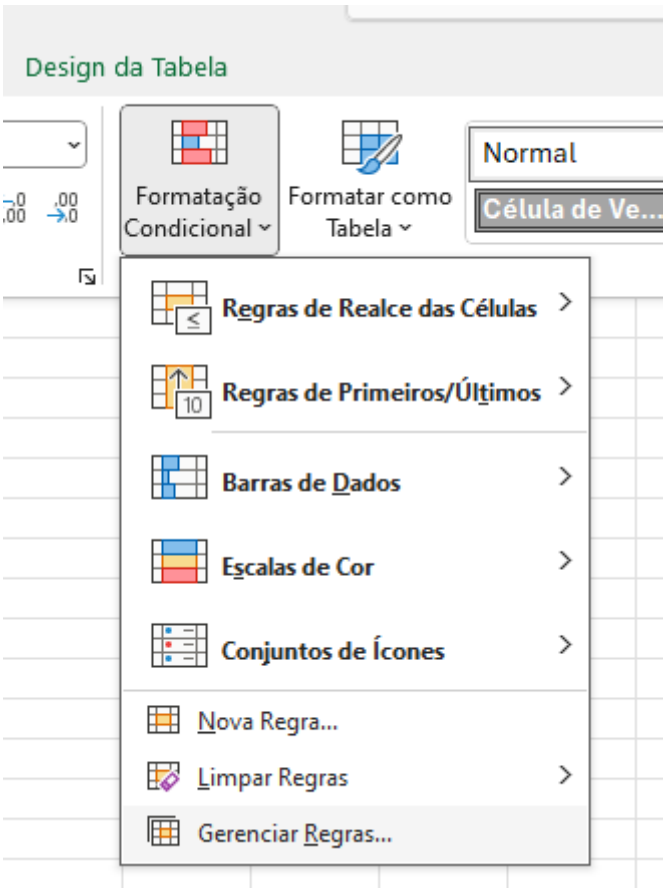
```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)

    If Not Intersect(Target, Me.Range("A1:A10")) Is Nothing Then

        MsgBox "Você selecionou uma célula na faixa A1:A10."

    End If

End Sub
```

5. Executando Macros no Fechamento de uma Pasta de Trabalho

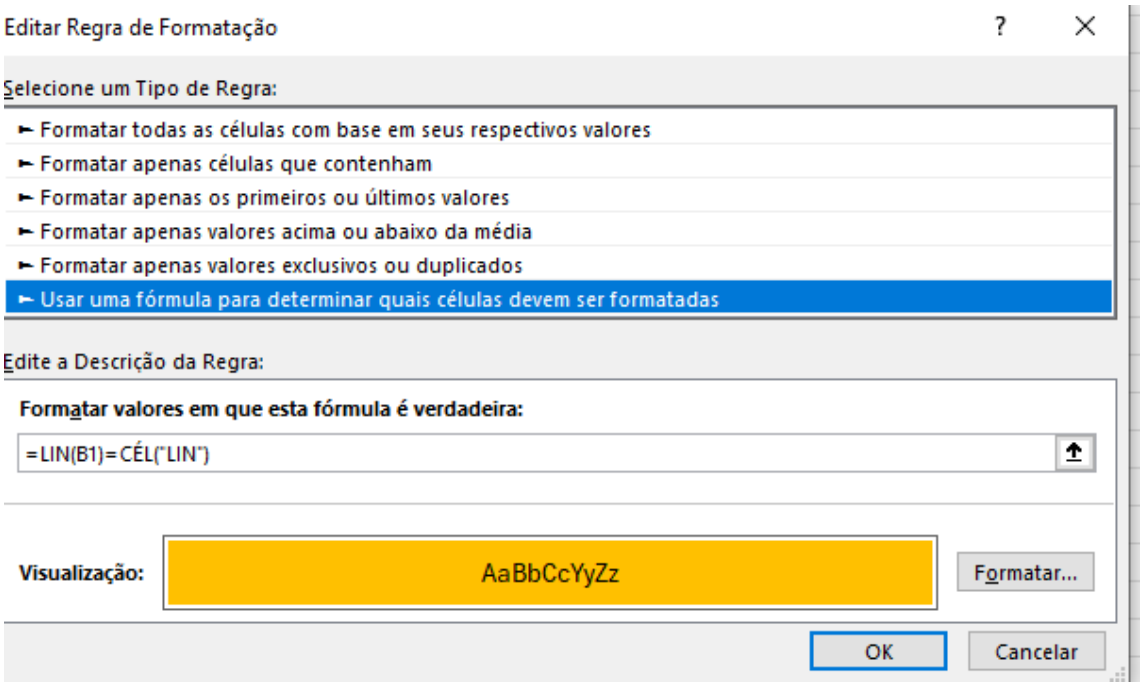
```
Private Sub Workbook_BeforeClose(Cancel As Boolean)

    If MsgBox("Você realmente deseja fechar a pasta de trabalho?",
vbYesNo) = vbNo Then

        Cancel = True ' Cancela o fechamento da pasta de trabalho

    End If

End Sub
```

6. Destacando uma linha:

- Vá na Formatação Condicional para adicionar uma condição para identificar a linha que está clicada:

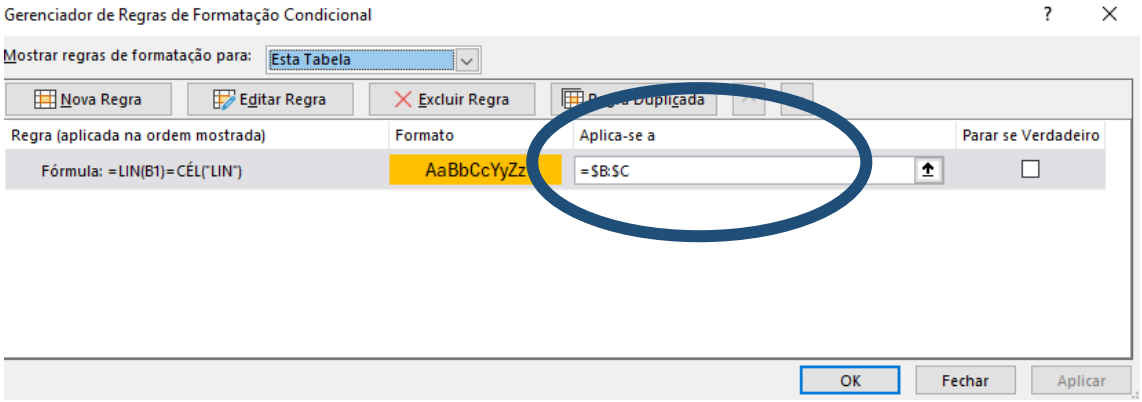


- Formate para a cor de destaque e coloque a fórmula abaixo:

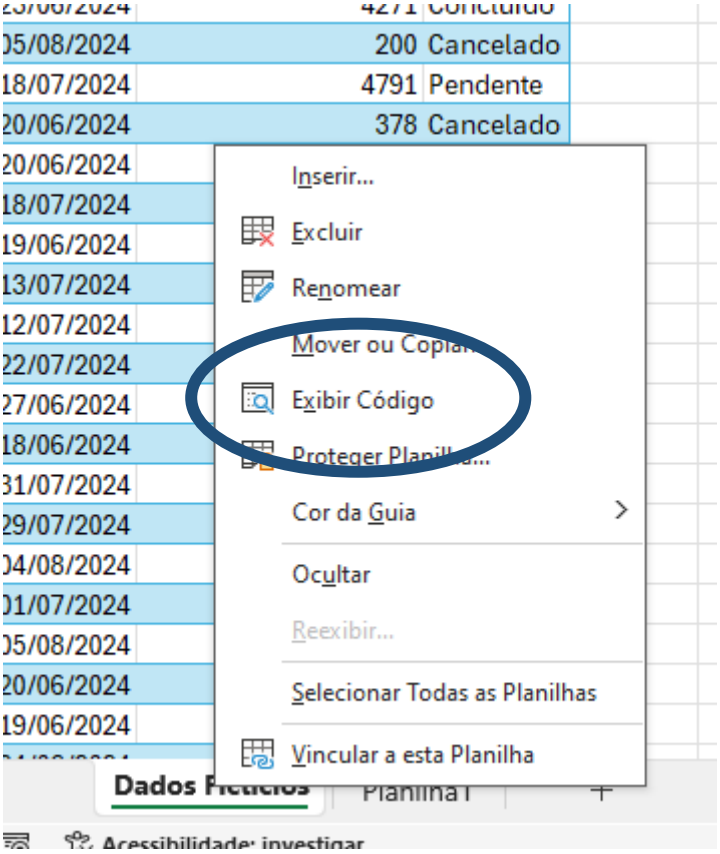


```
=LIN(B1)=CÉL("LIN")
```

- Agora mude a aplicação para onde você deseja destacar:



- Esse cálculo não é feito automaticamente. Para isso, vamos colocar um código de cálculo toda vez que houver uma nova seleção da nossa aba:




```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)

    Calculate
```

```
End Sub
```