

VBA

Estudo dirigido de Macro e VBA

Sergio Pedro R Oliveira

24 março 2024

SUMÁRIO

1	OBJETIVO	1
2	MACROS	2
2.1	Diferenças entre Macros e VBA	2
2.2	Habilitar opção desenvolvedor	2
2.3	Gravar Macro	3
2.4	Acessando Macros	7
2.5	Salvando Arquivo com Macro	9
2.6	Noção básica do script das Macros	10
2.7	Simplificando Macros	12
2.7.1	Teoria de simplificação de Macros	12
2.7.2	Técnicas para simplificar Macros	13
2.8	Referência Relativa (ou Macro Relativa)	15
2.8.1	Referência Relativa	15
2.8.2	Referência Absoluta	15
2.8.3	Usando referências relativa e absoluta	15
2.8.4	Simplificando VBA de referência relativa e absoluta	16
2.9	Diferença entre Macro Absoluta e Macro Relativa - Uso de Offset	20
2.9.1	Diferença entre Macro Absoluta e Relativa	20
2.9.2	Verificação imediata	20
2.9.3	Offset	21
2.9.4	Depuração	21
2.10	Meu primeiro programa VBA	22
2.10.1	Caixa de Diálogo	22
3	DESENVOLVENDO PRIMEIRA FERRAMENTA (BÁSICO)	24
3.1	Sistema de cadastro de clientes	24
3.1.1	Apagar linhas de grade, barra de fórmulas e títulos	24
3.1.2	Criar tabela	26
3.1.3	Proteger/Bloquear/Trancar tabela	27
3.1.4	Atribuir Macro num botão	29
3.2	Sistema de cadastro de clientes - VBA	30
3.2.1	Inserir Controles	30
3.2.2	Omitir Ações da Macro - função Application.ScreenUpdating	32
3.2.3	Cadastrar última linha de clientes	34

3.2.4	Teste de formulários - Preenchimento coletivo de células	34
3.2.5	Depurar código VBA da Macro	35
3.2.6	Movimentar cursor usando última célula como referência - <code>ActiveCell.Offset(0,0).Select</code>	35
3.2.7	Selecionar diversos campos com VBA	35

LISTA DE FIGURAS

1	Habilitar opção de Desenvolvedor.	2
2	Primeira opção para gravar macro.	3
3	Segunda opção para gravar macro.	3
4	Terceira opção para gravar macro.	3
5	Tela inicial de gravação de macros.	4
6	Encerrar gravação de Macro, primeira forma.	5
7	Encerrar gravação de Macros, segunda forma.	5
8	Acessando a tela de Macros, opção 1 (aba “Desenvolvedor” > opção “Macros”).	7
9	Acessando a tela de Macros, opção 2 (aba “Exibir” > opção “Macros”).	7
10	Tela de Macros, opção <i>Editar</i>	8
11	Tela de Visual Basic, para editar Macro.	8
12	Salvando arquivo Excel com Macros.	9
13	Script original da Macro.	12
14	Script simplificado da Macro.	12
15	Opção de “Usar Referências Relativas”.	15
16	Tela de Verificação Imediata.	20
17	Executar código VBA.	21
18	Depuração de código atrás de linha com erro.	21
19	Caixa de diálogo.	22
20	Programa VBA para criação de caixa de diálogo.	23
21	Exemplo de tabela com linhas de grade, barra de fórmula e títulos.	25
22	Exemplo de tabela sem linhas de grade, barra de fórmula e títulos.	25
23	Exemplo de tabela.	26
24	Proteger/bloquear planilha.	28
25	Opções para proteção/bloquear planilha.	28
26	Inserir controles.	30
27	Ferramentas de controle.	31
28	Desliga a atualização de tela na edição da Macro. <code>Application.ScreenUpdating = False</code> .	32
29	Religa a atualização de tela na edição da Macro. <code>Application.ScreenUpdating = True</code> .	33
30	Preencher vários campos ao mesmo tempo.	34

LISTA DE TABELAS

1 OBJETIVO

Estudo dirigido Macro e **VBA** Excel.

2 MACROS

2.1 Diferenças entre Macros e VBA

- Macros:
 - Script para automatização de tarefas.
 - Gravar sequências de ações e transformar em um script ou comandos (macros).
 - Não necessita de conhecimento de programação.
- VBA (*Visual Basic for Application*):
 - Linguagem de programação de script do Excel.

2.2 Habilitar opção desenvolvedor

Excel > Arquivo > Opções > Personalizar faixa de opções > (marcar opção) Desenvolvedor

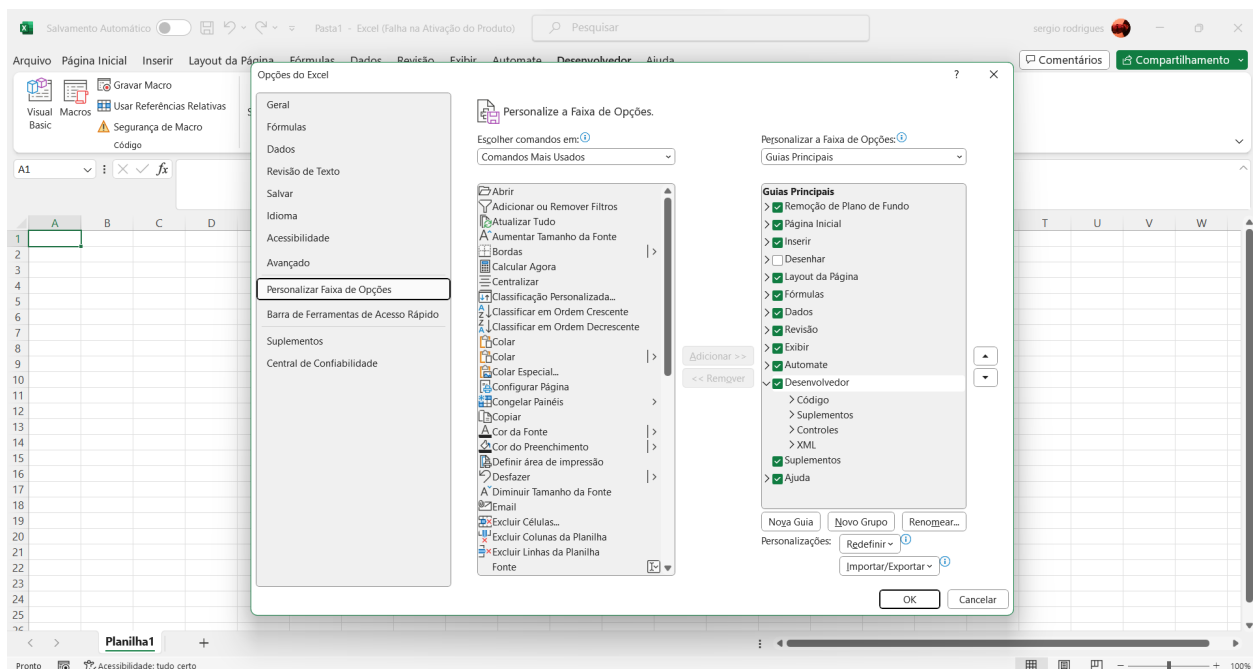


Figure 1: Habilitar opção de Desenvolvedor.

2.3 Gravar Macro

- Três opções para abrir uma gravação de uma Macro:

1. Primeira opção:

Na aba Desenvolvedor, opção “Gravar macro”.

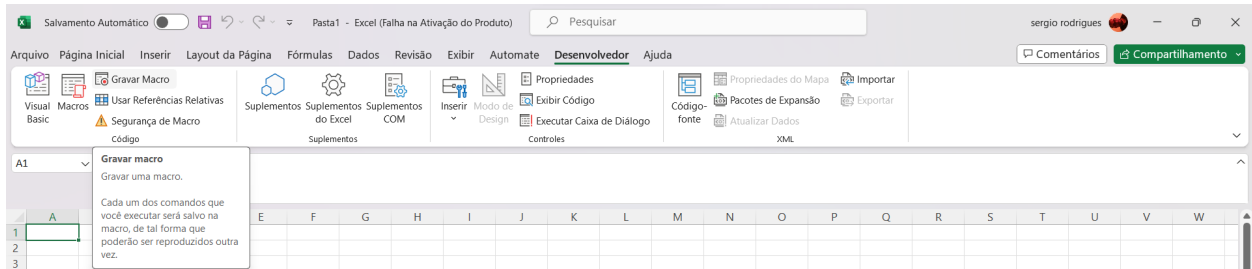


Figure 2: Primeira opção para gravar macro.

2. Segunda opção:

Na aba Exibir, dentro da opção “Macros”, tem a opção “Gravar macro”.

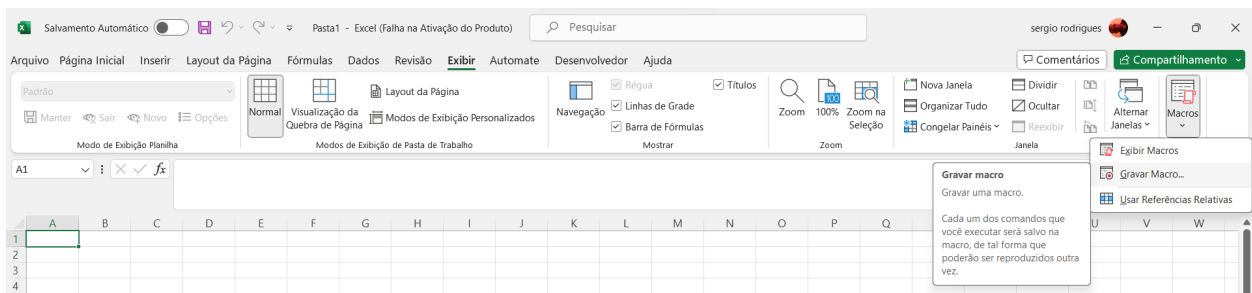


Figure 3: Segunda opção para gravar macro.

3. Terceira opção:

No rodapé esquerdo tem um ícone para gravar macro.

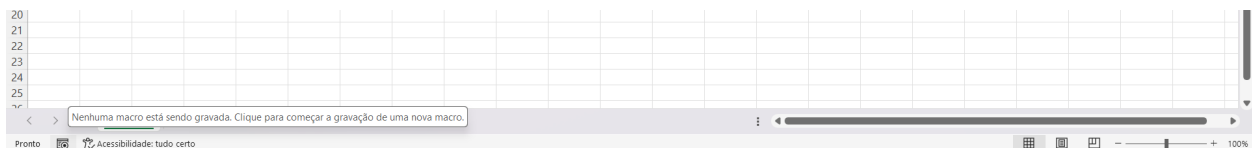


Figure 4: Terceira opção para gravar macro.

- Tela inicial de gravação de Macro:
 - *Nome da macro:*
 - * Inserir o nome da Macro.
 - * Não aceita espaço, logo, ou colar o nome da macro tudo junto (“PrimeiraMacro”), ou usar o underline (“Primeira_Macro”).
 - *Tecla de atalho:*
 - * Podemos colocar um atalho para chamar a Macro.
 - * Caso o atalho já esteja em uso, o Excel sugere outro atalho, mas não deixa sobreescrever.
 - *Armazenar macro em:*
 - * *Esta pasta de trabalho*
A Macro funcionará apenas para este arquivo.
 - * *Nova pasta de trabalho*
Cria a Macro em outra pastas.
 - * *Pasta de trabalho pessoal de macros*
É criado uma macro global que serve para todos as planilhas.
 - *Descrição:*
Podemos inserir um texto que descreve o que a macro faz.

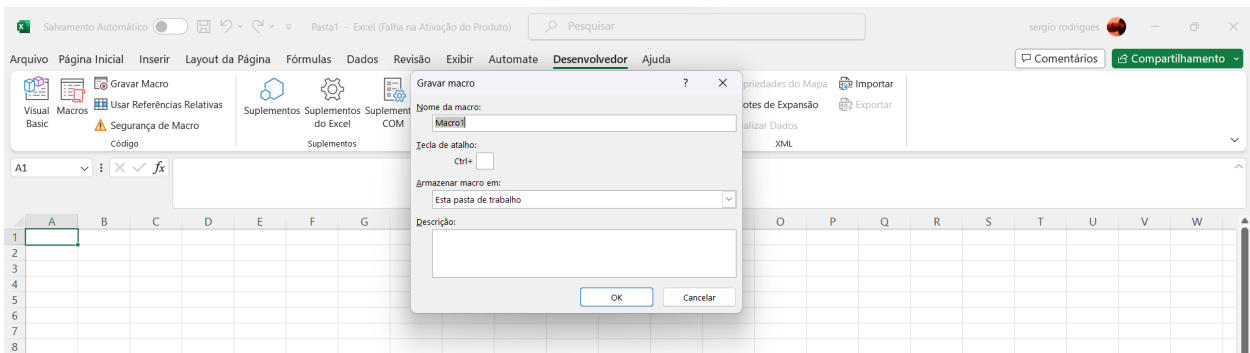


Figure 5: Tela inicial de gravação de macros.

- Gravando Macro:
 - Ao iniciar uma gravação de uma macro, o programa não pega a movimentação do mouse e nem o tempo.
 - O programa pega apenas as tarefas aplicadas.
- Encerrando gravação de uma Macro:

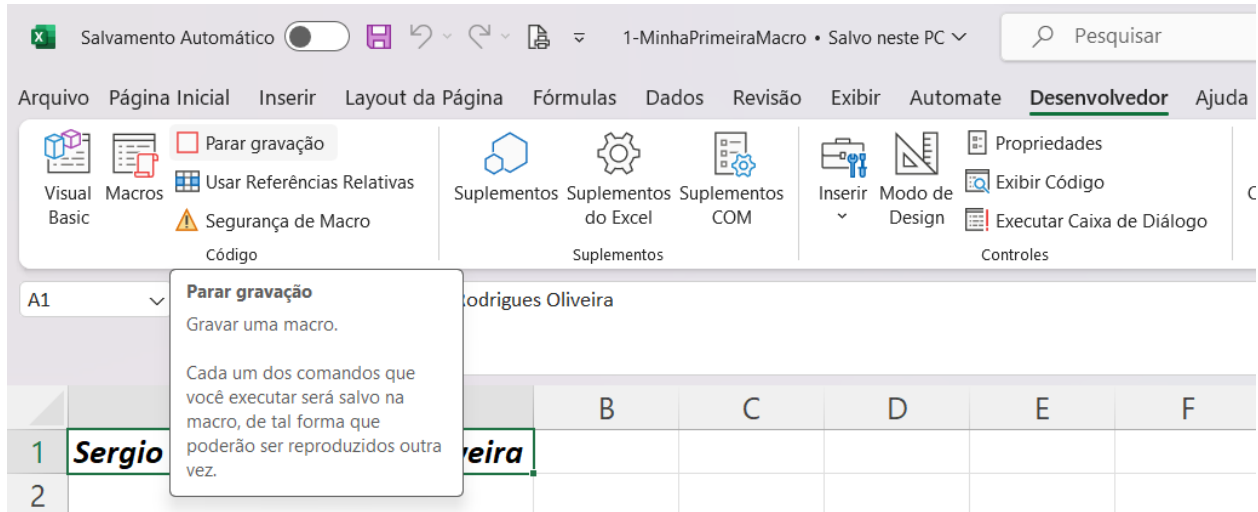


Figure 6: Encerrar gravação de Macro, primeira forma.

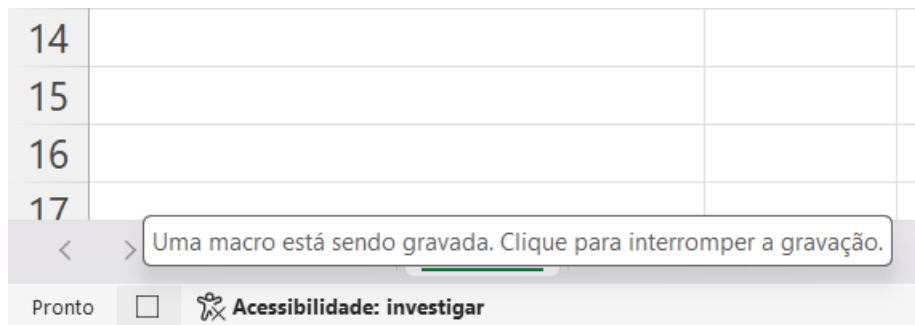
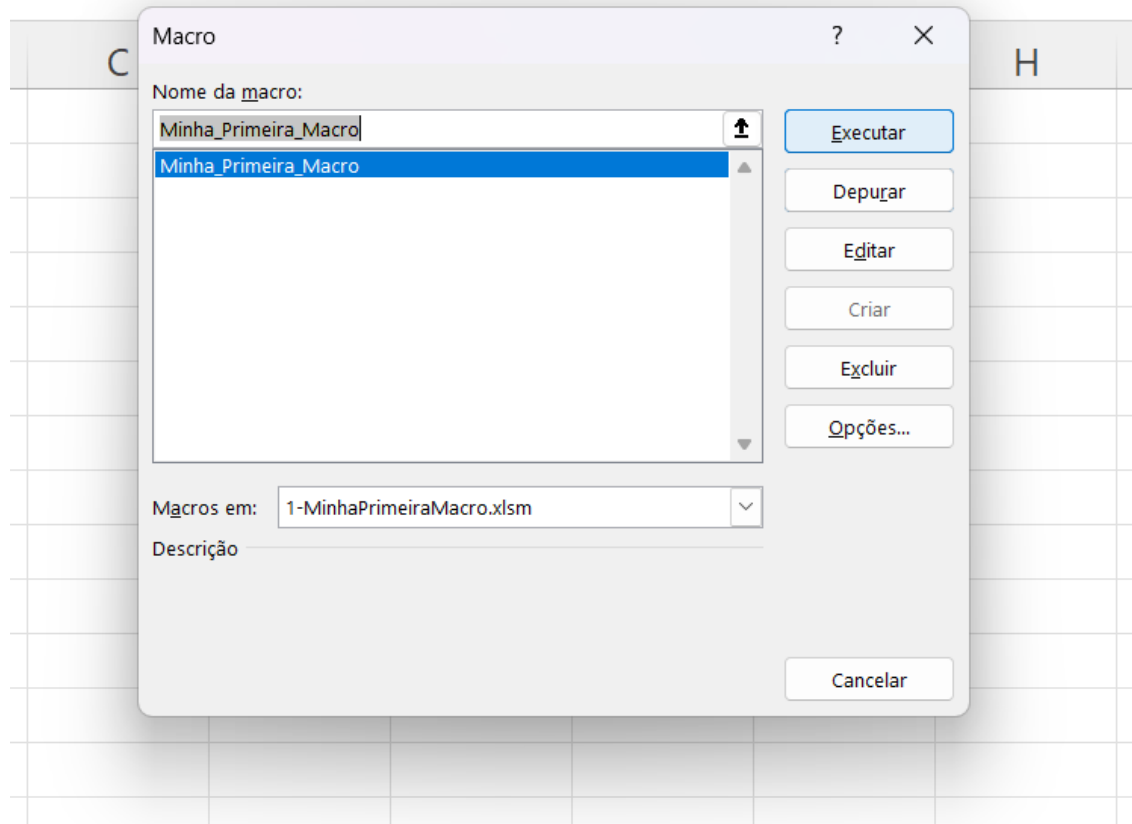


Figure 7: Encerrar gravação de Macros, segunda forma.

- Executando uma Macro:
 - Usando o atalho de teclado sugerido.
Caso tenha esquecido qual o atalho do teclado para determinada Macro, na tela de edição da Macro (tela de *Visual Basic*), no script da Macro, fica comentado o atalho do teclado.
 - Na aba “*Desenvolvedor*”, na opção “*Macros*”. Na tela de “*Macro*”, selecionar a Macro desejada e apertar a opção “*Executar*”.



2.4 Acessando Macros

- Para acessar as Macros criadas basta ir na aba “Desenvolvedor”, ou na aba “Exibir”, na opção “Macros”.

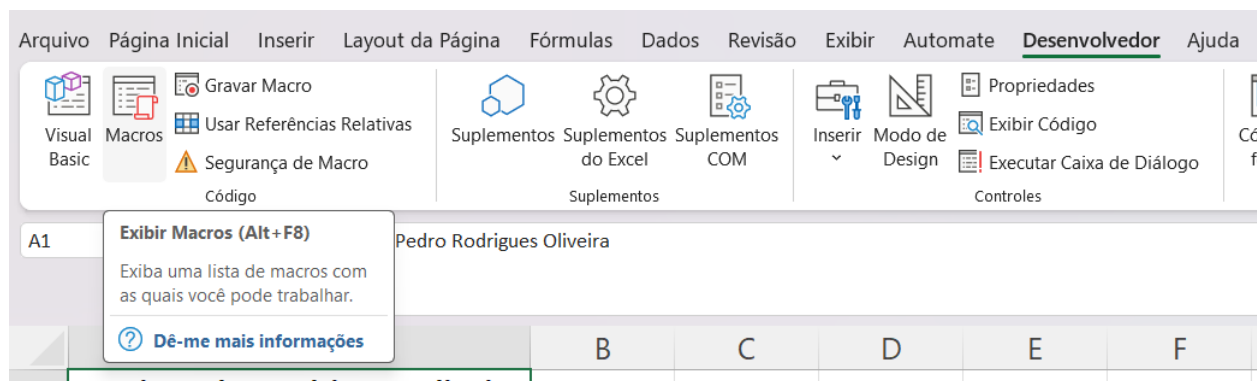


Figure 8: Acessando a tela de Macros, opção 1 (aba “Desenvolvedor” > opção “Macros”).

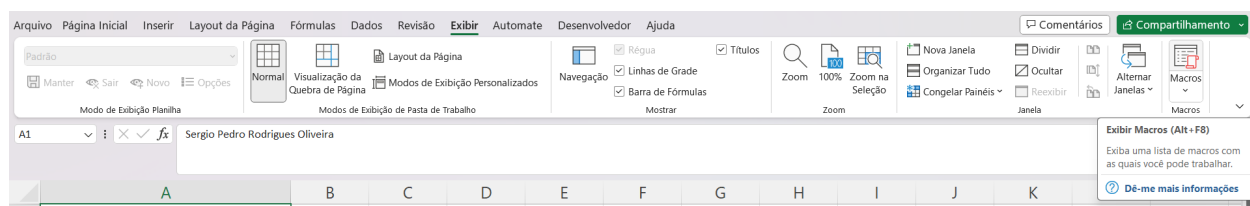


Figure 9: Acessando a tela de Macros, opção 2 (aba “Exibir” > opção “Macros”).

- Para acessar o script gerado pela Macro e poder editá-lo, na tela de “*Macros*”, selecionamos a Macro desejada e procuramos a opção “*Editar*”. Assim somos enviados para a tela do “*Visual Basic*”.

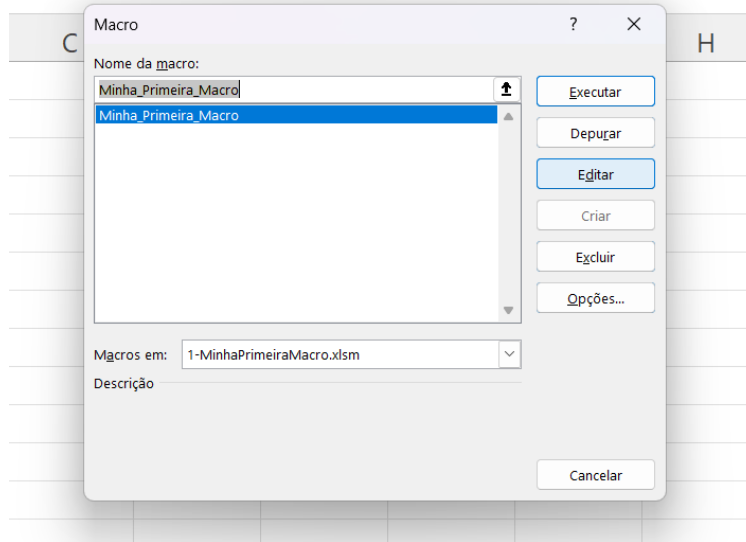


Figure 10: Tela de Macros, opção *Editar*.

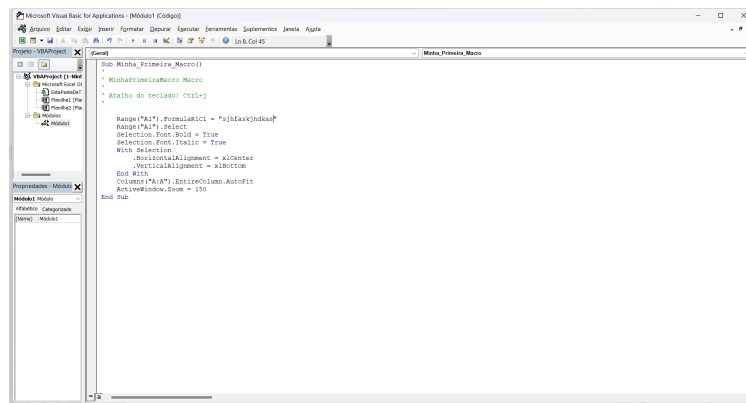


Figure 11: Tela de Visual Basic, para editar Macro.

2.5 Salvando Arquivo com Macro

- O Excel por default não salva as Macros criadas em um arquivo Excel.
- Quando o arquivo é salvo e fechado as Macros são apagadas.
- Para salvar um arquivo com Macros é necessario:
 - ir em “*salvar como*”.
 - Definir o local de salvamento.
 - Definir um nome para o arquivo.
 - No tipo de arquivo, selecionar a opção:
“*Pasta de Trabalho Habilitada para Macro do Excel (*.xlsm)*”

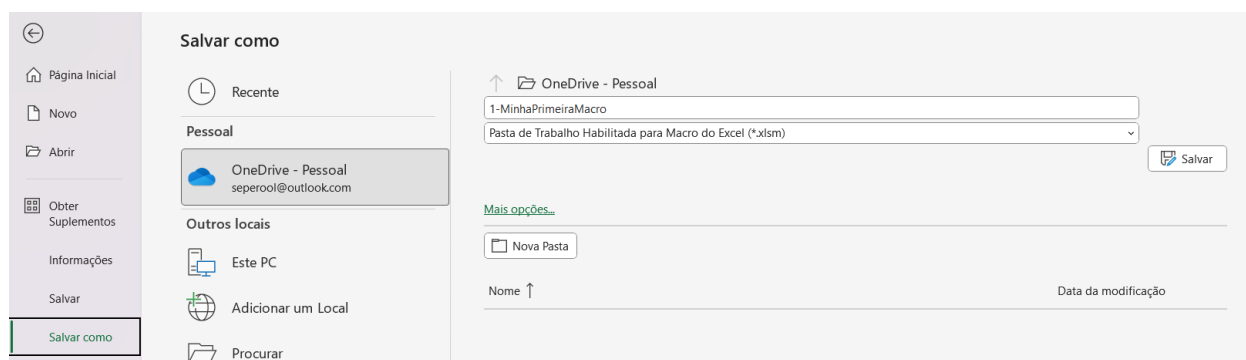


Figure 12: Salvando arquivo Excel com Macros.

2.6 Noção básica do script das Macros

- Adicionar comentários:
 - Para adicionar comentários ao script usamos uma aspas simples, tudo que vier depois na linha é um comentário.
 - Como boas práticas normalmente adicionamos:
 - * O nome da função (Macro). (Automático)
 - * Atalho no teclado para executar a Macro. (Automático)
 - * Data de criação.
 - * Quem criou a Macro e o seu contato.
 - * Breve descrição da Macro.
- Seleção:
 - Selecionar célula:
`Range("A1").Select`
 - Selecionar coluna:
`Columns("A:A").Select`
 - Selecionar linhas:
`Rows("1:3").Select`
- Zoom:
Determina o valor do zoom da janela do Excel.
`ActiveWindow.Zoom = 100`
- Inserir fórmula numa célula ativa (selecionada):
 - Texto:
`ActiveCell.FormulaR1C1 = "Texto"`
 - Fórmula:
`ActiveCell.FormulaR1C1 = "=SUM(4,5)"`

- Formatação básica:

- Itálico
`Selection.Font.Italic = True | False`
- Negrito
`Selection.Font.Bold = True | False`
- Alinhamentos:
Dentro das células selecionadas pelo comando `with`:

```
With Selection
    .Bloco de programação
End With

– Horizontal
    .HorizontalAlignment = xlRight | xlCenter | xlLeft

– Vertical
    .VerticalAlignment = xlTop | xlCenter | xlBottom

– Auto-ajuste coluna/linha (Automático)
    Columns("A:A").EntireColumn.AutoFit
    Rows("1:1").EntireColumn.AutoFit
```

- Função (Macro):

- VBA é uma linguagem indentada.
- A função Macro no VBA do Excel é uma função `Sub()`.
- Como o nome da Macro é o nome que encontramos na função `Sub()`, podemos mudar o nome da Macro alterando o nome na função `Sub()`.

Exemplo:

```
Mudando nome da Macro,
Sub MinhaPrimeiraMacro()
para,
Sub Minha_Primeira_Macro()
```

- Sintaxe:

```
Sub nome_da_macro(argumento_s_1, argumento_s_2, ...)
    Bloco de programação
End Sub
```


2.7 Simplificando Macros

2.7.1 Teoria de simplificação de Macros

- Podemos simplificar Macros ao abrir o script e cortar, ou juntar, trechos que são desnecessários, ou redundantes. Deixando a Macro mais objetiva.
- Exemplo de script simplificado:

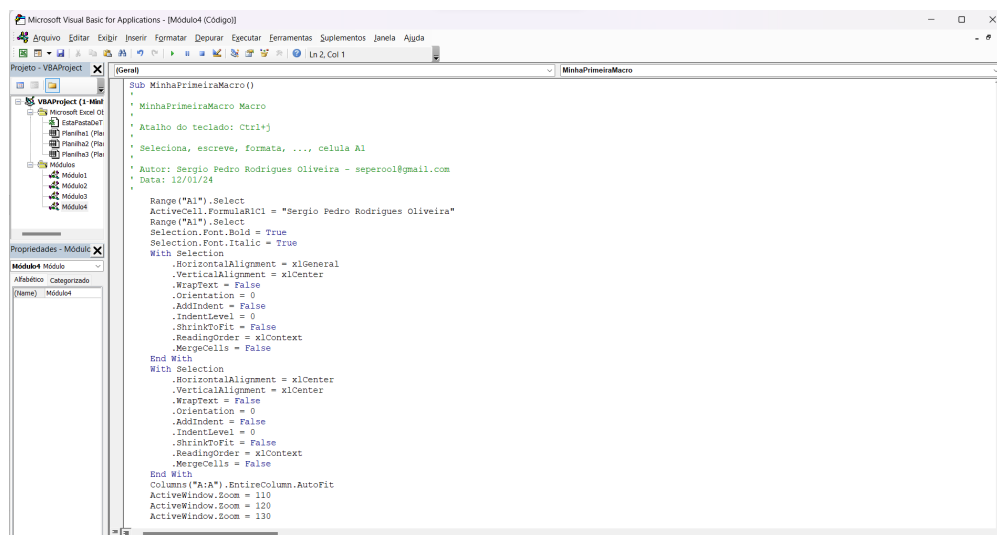


Figure 13: Script original da Macro.

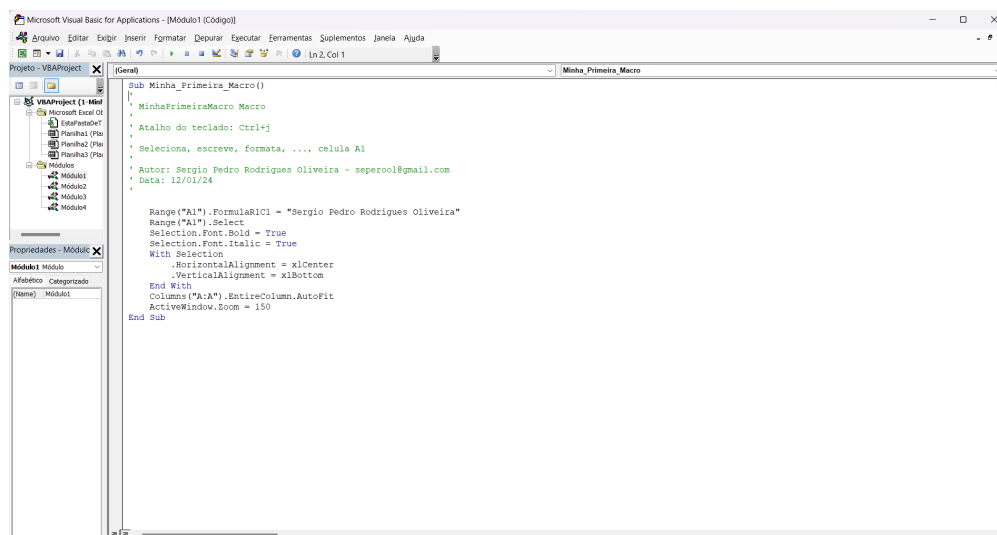


Figure 14: Script simplificado da Macro.

2.7.2 Técnicas para simplificar Macros

- Juntar seleção de célula e inserção de fórmula:
Ao invés de selecionar uma célula e inserir fórmula separadamente.

```
Range("A1").Select  
ActiveCell.FormulaR1C1 = "Sergio Pedro Rodrigues Oliveira"
```

Podemos fazer as duas coisas de uma só vez.

```
Range("A1").FormulaR1C1 = "Sergio Pedro Rodrigues Oliveira"
```

- Eliminando comandos desnecessários da formatação básica:
Existem determinados comandos que são gerados automaticamente quando desejamos fazer formatação do alinhamento do texto numa célula, dentro do comando `with selection`, nem todos esses comandos são necessários.

```
With Selection  
    .HorizontalAlignment = xlGeneral  
    .VerticalAlignment = xlCenter  
    .WrapText = False  
    .Orientation = 0  
    .AddIndent = False  
    .IndentLevel = 0  
    .ShrinkToFit = False  
    .ReadingOrder = xlContext  
    .MergeCells = False
```

End With

```
With Selection  
    .HorizontalAlignment = xlCenter  
    .VerticalAlignment = xlCenter  
    .WrapText = False  
    .Orientation = 0  
    .AddIndent = False  
    .IndentLevel = 0  
    .ShrinkToFit = False  
    .ReadingOrder = xlContext  
    .MergeCells = False
```

End With

Logo, podemos omiti-los para simplificar o script.

Também podemos juntar os comandos de formatação de alinhamento da mesma célula dentro de um mesmo comando `with selection`.

```
With Selection  
    .HorizontalAlignment = xlCenter  
    .VerticalAlignment = xlBottom  
End With
```

- Zoom simplificado:
Ao invés de passar de valor de zoom por valor de zoom.

```
ActiveWindow.Zoom = 110  
ActiveWindow.Zoom = 120  
ActiveWindow.Zoom = 130  
ActiveWindow.Zoom = 140  
ActiveWindow.Zoom = 150
```

Podemos ir direto no valor de zoom desejado, eliminando assim as transições.

```
ActiveWindow.Zoom = 150
```

2.8 Referência Relativa (ou Macro Relativa)

2.8.1 Referência Relativa

- Quando ativado a opção “Usar referência relativa”, na aba “Desenvolvedor”, aplica a Macro onde o cursor esta ativo, ou usa o curso como referência.
- Ex.: Se você gravar na célula A1 uma Macro que move o cursor para A3, com a opção “Usar referência relativa” ativada, a execução da Macro resultante na célula J6 moverá o cursor para J8.

2.8.2 Referência Absoluta

- A forma tradicional com a opção “Usar referência relativa” desativada, é chamado de referência absoluta, onde a Macro apenas repete os comandos nas células | colunas | linhas selecionadas, previamente estabelecidas.
- Ex.: Se você gravar na célula A1 uma Macro que move o cursor para A3, com a opção “Usar referência relativa” desativada, a execução da Macro resultante na célula J6 moverá o cursor para A3.

2.8.3 Usando referências relativa e absoluta

- Podemos usar as duas opções de referência durante a gravação de uma Macro, basta ativar e desativar a opção “Usar referência relativa” durante o processo de gravação.

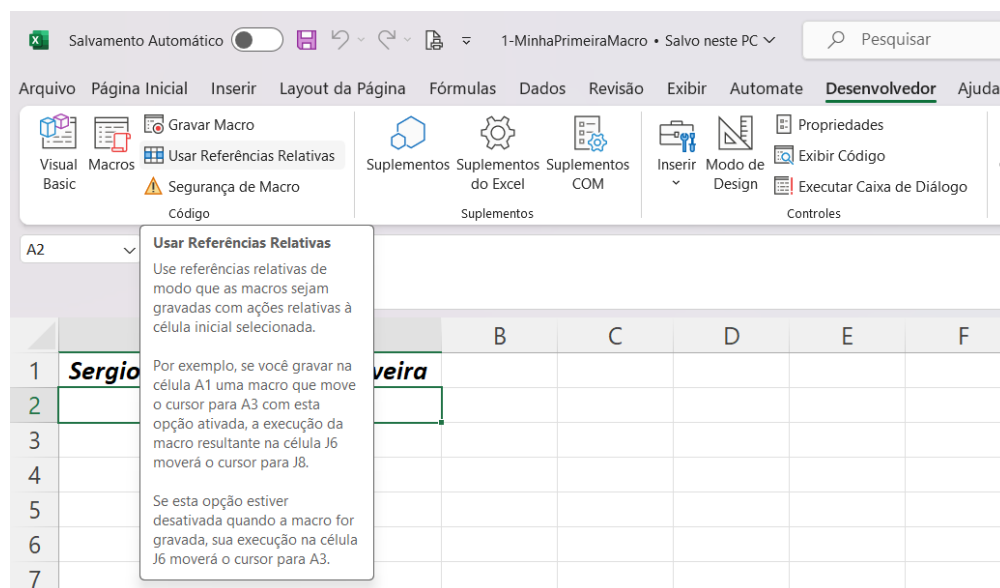


Figure 15: Opção de “Usar Referências Relativas”.

2.8.4 Simplificando VBA de referência relativa e absoluta

- Ao criar uma Macro usando referência relativa ou absoluta o código tem as seguintes características:

- Referência Relativa:

- * Selecionar linha de referencia (linha inteira):
`ActiveCell.Rows("1:1").EntireRow.Select`
- * Selecionar coluna de referencia (coluna inteira):
`ActiveCell.Columns("A:A").EntireColumn.Select`
- * Deletar seleção:
`Selection.Delete`
- * Mover linhas|colunas após apagar uma linha|coluna:
 - Linha:
`Selection.Delete Shift:=xlUp`
 - Coluna:
`Selection.Delete Shift:=xlToLeft`

- Referência Absoluta:

- * Selecionar linha:
`Rows("1:1").Select`
- * Selecionar coluna:
`Columns("A:A").Select`
- * Deletar seleção:
`Selection.Delete`
- * Mover linhas|colunas após apagar uma linha|coluna:
 - Linha:
`Selection.Delete Shift:=xlUp`
 - Coluna:
`Selection.Delete Shift:=xlToLeft`

- Podemos simplificar o código:

– Referência Relativa:

- * Simplificando seleção de linha de referência:

- Gerado automaticamente:
`ActiveCell.Rows("1:1").EntireRow.Select`
- Simplificando:
`ActiveCell.EntireRow.Select`

- * Simplificando seleção de coluna de referência:

- Gerado automaticamente:
`ActiveCell.Columns("A:A").EntireRow.Select`
- Simplificando:
`ActiveCell.EntireColumn.Select`

- * Deletando determinada seleção de linha|coluna:

- Linha, programação gerada automaticamente:

```
ActiveCell.Rows("1:1").EntireRow.Select
Selection.Delete Shift:=xlUp
```

- Linha simplificando:
`ActiveCell.EntireRow.Delete`

- Coluna, programação gerada automaticamente:

```
ActiveCell.Columns("A:A").EntireColumn.Select
Selection.Delete Shift:=xlToLeft
```

- Coluna simplificando:
`ActiveCell.EntireColumn.Delete`

- * Mover linhas|colunas após apagar uma linha|coluna:
Não é preciso colocar nada, o programa faz automaticamente.
Podemos simplesmente omitir:`Shift:=xlUp` | `Shift:=xlToLeft`

– Referência Absoluta:

- * Deletando determinada seleção de linha|coluna:

- Linha gerada automaticamente:

```
Rows("1:1").Select
Selection.Delete Shift:=xlUp
```

- Linha Simplificado:
`Rows("1:1").Delete`

- Coluna gerada automaticamente:

```
Columns("A:A").Select  
Selection.Delete Shift:=xlToLeft
```

- Coluna simplificado:
`Columns("A:A").Delete`

- * Mover linhas|colunas após apagar uma linha|coluna:
Não é preciso colocar nada, o programa faz automaticamente.
Podemos simplesmente omitir:`Shift:=xlUp` | `Shift:=xlToLeft`

- Exemplo de Simplificação de código.

Código Original:

```
Sub ApagarLinhaRelativamente()  
,  
' ApagarLinhaRelativamente Macro  
,  
,  
,  
    ActiveCell.Rows("1:1").EntireRow.Select  
    Selection.Delete Shift:=xlUp  
    Rows("10:10").Select  
    Selection.Delete Shift:=xlUp  
End Sub
```

Código Simplificado:

```
Sub Apagar_Linha_Relativamente()  
,  
' Apagar_Linha_Relativamente Macro  
' Macro Simplificada  
,  
,  
    ActiveCell.EntireRow.Delete  
    Rows("10:10").Delete  
End Sub
```


2.9 Diferença entre Macro Absoluta e Macro Relativa - Uso de Offset

2.9.1 Diferença entre Macro Absoluta e Relativa

- Quando marcamos, ou desmarcamos, a opção “Usar Referências Relativas”, o código VBA da Macro é alterado.
- Diferença é o uso do `Offset`, que cria uma relação relativa (referência relativa) no código VBA gerado.
- Diferença entre códigos:
 - Macro Absoluta:
`Range("A1").Select`
Determina o célula selecionada.
 - Macro Relativa:
`ActiveCell.Offset(-4, -3).Range("A1").Select`
A partir da célula selecionada como referência, anda determinadas coordenadas de células.

2.9.2 Verificação imediata

- Abre uma janela de testes rápidos “verificação imediata” para testar códigos VBA.
- Atalho para abrir a tela no editor VBA:
`Ctrl + G`
- Exemplo de verificação imediata:
`ActiveCell.Offset(-4, -3).Select`
Move a célula selecionada em determinada coordenada (-4 linha,-3 coluna).

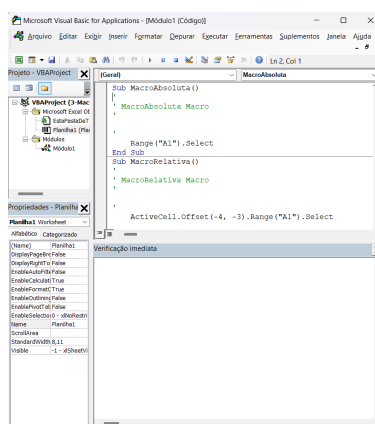


Figure 16: Tela de Verificação Imediata.

2.9.3 Offset

- O comando `Offset` movimenta o cursor da célula ativa, em determinada coordenada (linha, coluna).
`ActiveCell.Offset(-4, -3).Select`
- Coordenadas negativas movem o cursor para cima e para esquerda.
`ActiveCell.Offset(-4, -3).Select`
- Coordenadas positivas movem o cursor para baixo e para direita.
`ActiveCell.Offset(1, 2).Select`

2.9.4 Depuração

- Para executar uma Macro através da janela de VBA, podemos usar o `F5`, ou o botão continuar na barra superior da janela (seta de execução). Basta deixar o cursor dentro do código que deverá ser executado.

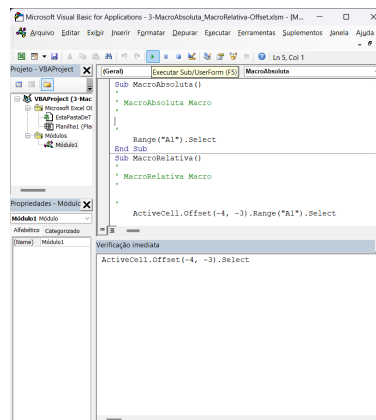


Figure 17: Executar código VBA.

- Caso ocorra algum erro, podemos pedir para depurar o código atrás da linha que ocasionou/desencadeou o erro. A linha do erro é destacada.

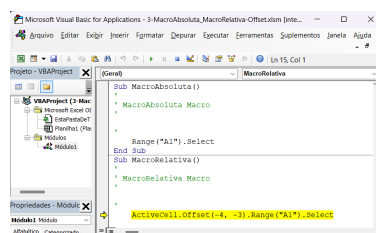


Figure 18: Depuração de código atrás de linha com erro.

2.10 Meu primeiro programa VBA

2.10.1 Caixa de Diálogo

- No VBA, para exibir uma caixa de diálogo usamos a função `MsgBox`.
- Outros argumentos que podemos adicionar a caixa de diálogo são:
 - Botão OK
Para fechar a caixa de diálogo.
`MsgBox "Mensagem", vbOKOnly`
 - Título a caixa de diálogo
Podemos dar um título para a caixa de diálogo.
`MsgBox "Mensagem", vbOKOnly, "Título"`

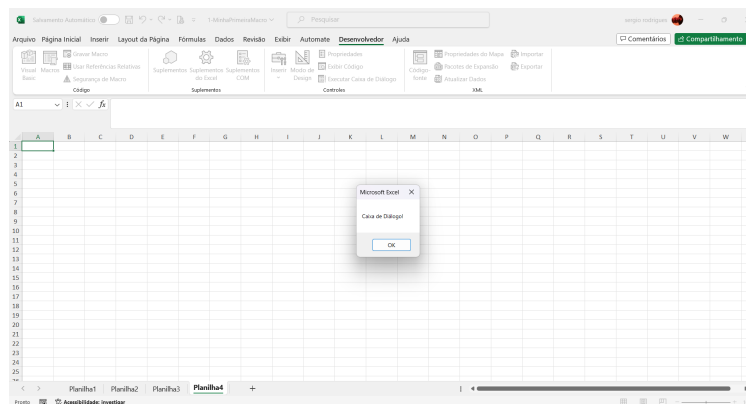


Figure 19: Caixa de diálogo.

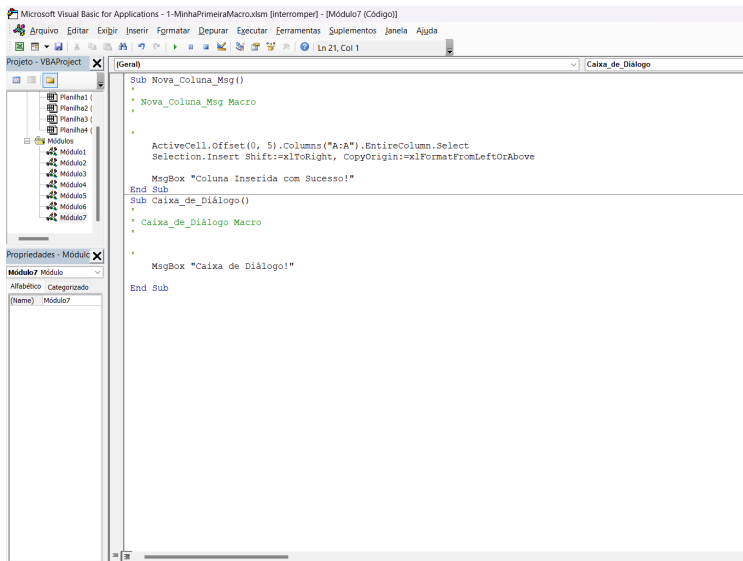


Figure 20: Programa VBA para criação de caixa de diálogo.

- Ex.:

```

Sub nome_Macro()
    MsgBox "Caixa de diálogo."
End Sub

```

3 DESENVOLVENDO PRIMEIRA FERRAMENTA (BÁSICO)

3.1 Sistema de cadastro de clientes

3.1.1 Apagar linhas de grade, barra de fórmulas e títulos

- Podemos para deixar uma aba com aspecto de programa/sistema, remover/omitir informações da tabela.
- Entre as partes que podemos omitir temos:
 - Linhas de grade.
Grid da tabela que demarca as células.
 - Barra de fórmulas.
 - Títulos.
São as coordenadas do tabela, tanto da parte superior (letras), quanto a parte lateral esquerda (números).
- Para apagar/omitir esses aspectos da tabela:
 - ir na aba `Exibir > Mostrar`;
 - Marcar ou desmarcar as opções:
 - * Linhas de grade.
 - * Barra de fórmulas.
 - * Títulos.

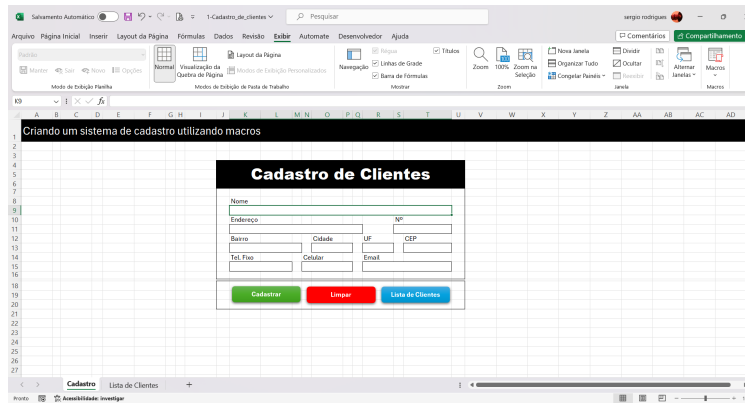


Figure 21: Exemplo de tabela com linhas de grade, barra de fórmula e títulos.

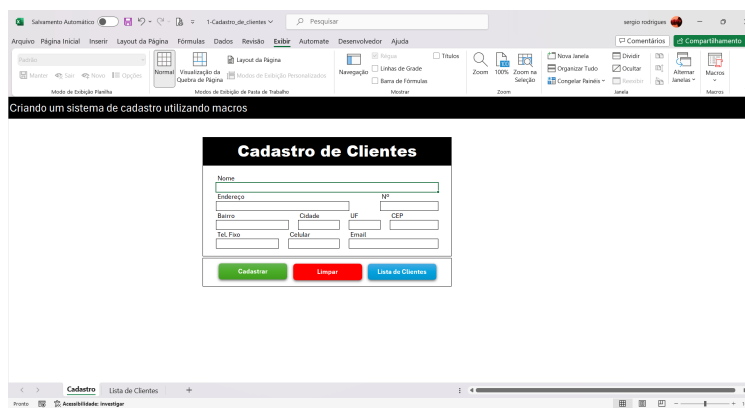


Figure 22: Exemplo de tabela sem linhas de grade, barra de fórmula e títulos.

3.1.2 Criar tabela

- A tabela organiza e analisa os dados.
- A tabela facilita:
 - Classificação;
 - Filtragem;
 - Formatação dos dados.
- Para criar uma tabela:
 - Basta selecionar os dados;
 - Ir em:
`inserir > Tabelas > Tabela`
 - Definir se tem ou não cabeçalho;

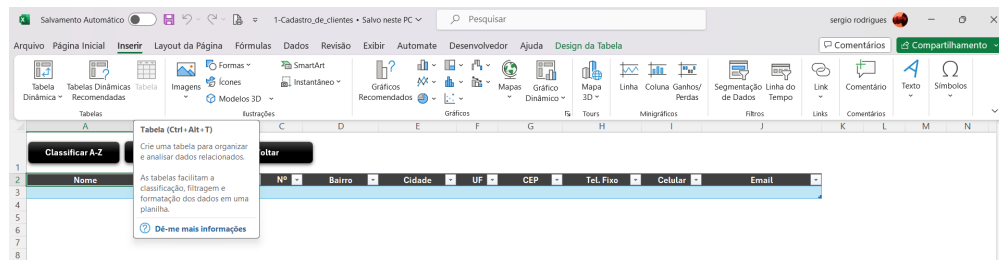


Figure 23: Exemplo de tabela.

3.1.3 Proteger/Bloquear/Trancar tabela

- É possível para proteger a tabela, ou evitar/restringir que o usuário da planilha edite algum campo não autorizado. Podemos trancar/bloquear a planilha.
- Para bloquear ou desbloquear a planilha:
 - Bloquear planilha:
`Aba Revisão > Proteger planilha`
 - Desbloquear planilha:
`Aba Revisão > Desproteger planilha`
- Trancar a planilha limita o usuário a ações predeterminadas:
 - Selecionar células bloqueadas
 - Selecionar células desbloqueadas
 - Formatar células
 - Formatar Colunas
 - Formatar linhas
 - Inserir colunas
 - Inserir linhas
 - Inserir hiperlinks
 - Excluir colunas
 - Excluir linhas
 - Classificar
 - Usar filtro automático
 - Utilizar tabela dinâmica e gráfico dinâmico
 - Editar objetos
 - Editar cenários
- Para destrancar a planilha é necessário senha.
- Útil para criação de formulários.
- Podemos deixar apenas determinados campos/células destrancadas para o usuário preencher. Basta selecionar tudo e desselecionar os campos que queremos deixar destrancados/desprotegidos.

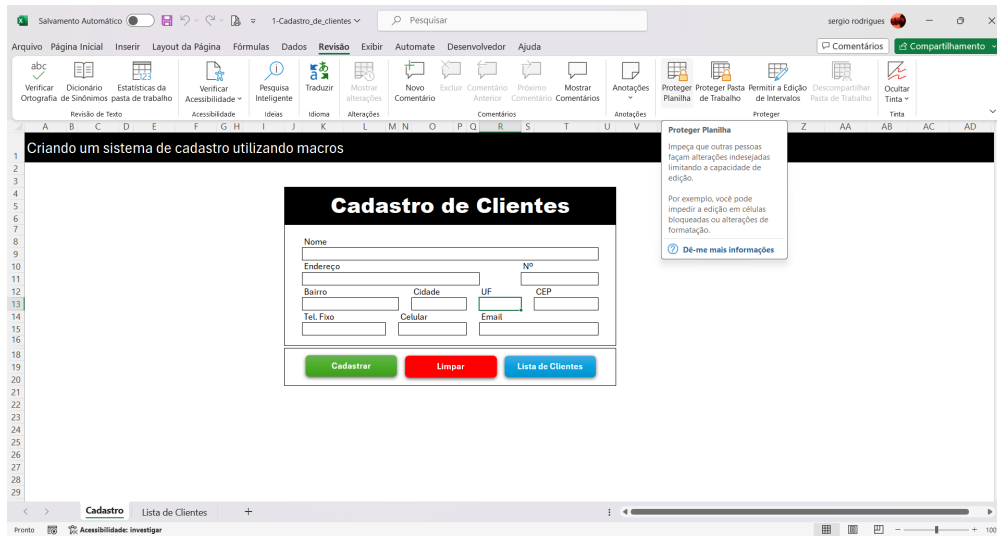


Figure 24: Proteger/bloquear planilha.

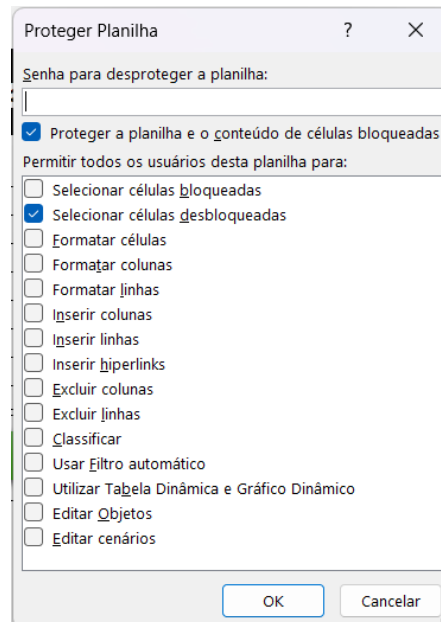


Figure 25: Opções para proteção/bloquear planilha.

3.1.4 Atribuir Macro num botão

- Podemos atribuir uma determinada Macro a um botão.
- Exemplos de utilidade de atribuição de Macro a um botão:
 - Limpar campos;
 - Cadastrar informações;
Inserir informações de uma aba cadastral para uma tabela apropriada.
 - Direcionar/levar o usuário para outra tabela;
- Passos para atribuir Macro num botão:
 1. Criar um botão;
 - Caminho:
`Inserir > Ilustrações > Formas`
Desenhar botão no formato desejado.
 - Na aba **Formatar** podemos alterar as especificações visual do botão.
 - Para inserir texto no botão, clicamos dentro do botão com o botão direito do mouse e vamos na opção `Editar texto`.
 2. Criar Macro;
Gravar uma Macro com as ações que desejamos que o botão execute.
 3. Atribuir Macro no botão.
 - Clicar com o botão direito do mouse sobre o botão e ir na opção `Atribuir macro`.
 - Selecionamos a Macro existente e confirmamos com `ok`.

3.2 Sistema de cadastro de clientes - VBA

- Podemos usar ferramentas mais sofisticadas de VBA e Macros para construir formulários.
- Algumas deles são:
 - **Inserir controles**
Ferramentas preexistentes no excel para ações com Macros.
 - Atraves da programação em **visual basic**
Serve para omitir ações executadas pelas macros.
Isso evita a sensação de bug no sistema.

3.2.1 Inserir Controles

- São ferramentas preexistentes no excel para ações com Macros.
Ex.: Botões preparados para receber Macro.
- Caminho:
Desenvolvedor > Controles > Inserir

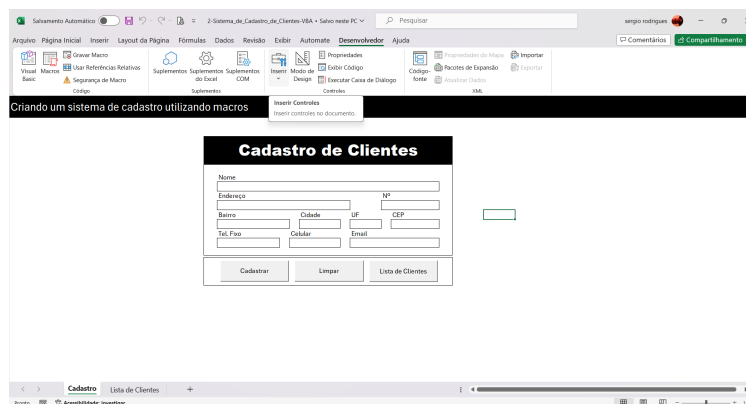


Figure 26: Inserir controles.

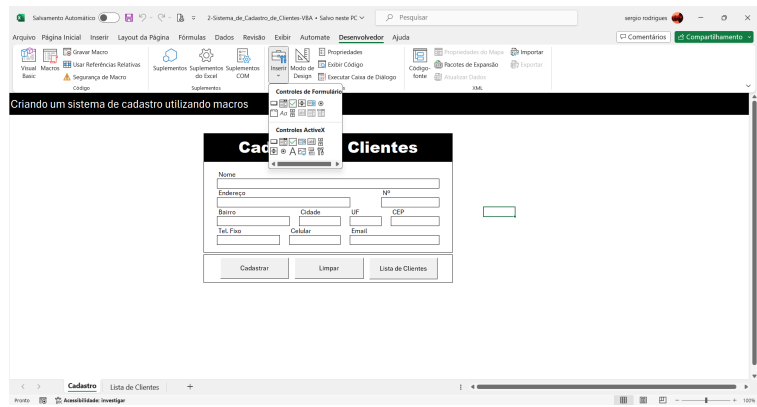


Figure 27: Ferramentas de controle.

3.2.2 Omitir Ações da Macro - função Application.ScreenUpdating

- Podemos omitir as ações executadas pela Macro, evitar que o sistema do excel pisque paracendo que esta bugado.
- Para omitir as ações usamos a função Application.ScreenUpdating dentro da edição VBA da Macro.
- A função Application.ScreenUpdating:
 - Application.ScreenUpdating = False
Desliga o que for atualizado na tela.
Não mostra as ações executadas.
Podemos colocar ao inicio do código da Macro para omitir as ações da Macro.
 - Application.ScreenUpdating = True
Religa ou ativa atualização na tela.
Mostra as ações executadas.
Por default o sistema sempre entende como ligado.
Caso desligado, ao final do programa é necessario ligar essa opção, ou seja, é precisamos inserir ao final do código, se no inicio do código a ação foi dado como desligado.

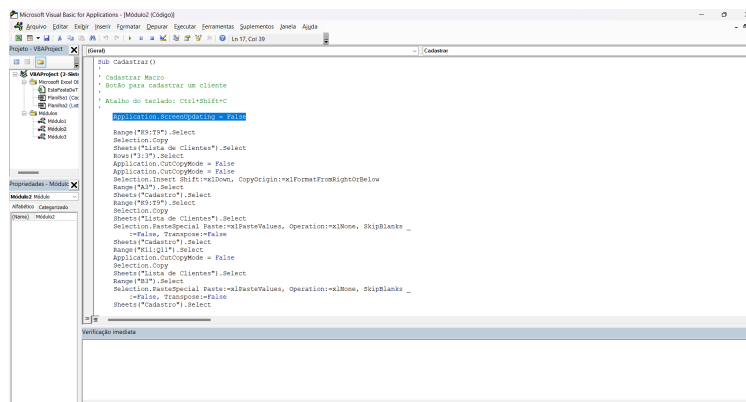


Figure 28: Desliga a atualização de tela na edição da Macro. Application.ScreenUpdating = False

3.2.3 Cadastrar última linha de clientes

3.2.4 Teste de formulários - Preenchimento coletivo de células

- Para testar formulários podemos preencher vários campos ao mesmo tempo.
- Passo a passo:
 1. Selecionar todos os campos que deseja preencher.
Selecionar células com **Ctrl**
 2. Pressione **F2** para editar uma das células selecionadas.
 3. **Ctrl + Enter** para replicar em o valor em todas as células selecionadas.

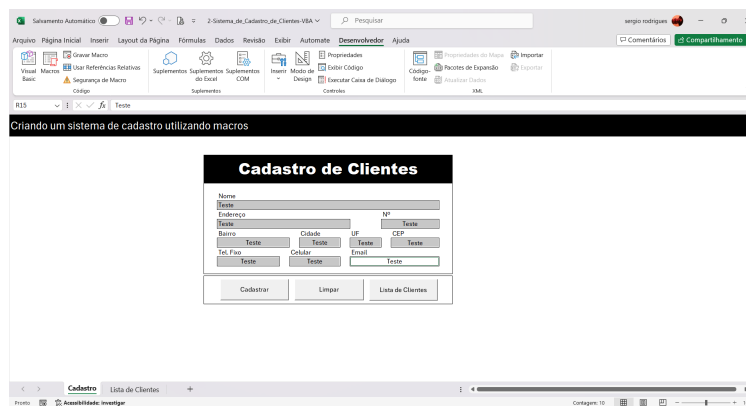


Figure 30: Preencher vários campos ao mesmo tempo.

3.2.5 Depurar código VBA da Macro

3.2.6 Movimentar cursor usando última célula como referência - `ActiveCell.Offset(0,0).Select`

3.2.7 Selecionar diversos campos com VBA

- É possível selecionar várias células usando a função `Range()` (no VBA):
`Range("K9:T9,K11:Q11,S11:T11,K13:M13,O13:P13,R13,T13,K15:L15,N15:P15,R15:T15").Select`