

# APOSTILA

VBA Impressão

versão 2.0



# O FORMATO

que vai fazer você relembrar tudinho



## Módulo X - Tema A - Assunto A

Qual **módulo**  
está sendo  
explicado

Qual **tema** está  
sendo explicado

Qual **assunto**  
está sendo  
explicado

### Print Screen

da aba que está sendo explicada

Página

**Explicação**  
Do tema trabalhado

# O CONTEÚDO

que você precisa para mandar bem



# Conteúdo

## Módulo 1 Introdução

01	Introdução .....	1
02	Qual é o nível preciso ter para aprender VBA .....	2
03	Exemplos de aplicações práticas: Exemplo 1 .....	3
04	Exemplos de aplicações práticas: Exemplo 2 .....	4
05	Exemplos de aplicações práticas: Exemplo 3 .....	5
06	Habilitando a guia Desenvolvedor .....	6
07	Formas de criar um código em VBA .....	9
08	Gravando a primeira macro (Parte 1) .....	11
09	Gravando a primeira macro (Parte 2) .....	18
10	Gravação de macro: Exemplo 1 (Parte 1) .....	24
11	Gravação de macro: Exemplo 1 (Parte 1) .....	32
12	Salvando o arquivo como habilitado para macro .....	42

## Módulo 2 Estrutura básica e objetos do VBA

13	Introdução .....	44
14	Objeto Range: Escrevendo em uma célula .....	47
15	Objeto Range: Escrevendo em várias células .....	50
16	Objeto Cells: Escrevendo em uma célula .....	52
17	Formas de executar uma macro .....	53
18	Registrando informações .....	55

## Módulo 2 Estrutura básica e objetos do VBA

19	Variáveis .....	57
20	Como declarar variáveis no VBA .....	60
21	Declarando variáveis no VBA .....	62
22	Exercícios variáveis .....	64
23	Exercícios variáveis (Parte 2) .....	66
24	Exercícios variáveis (Parte 3) .....	70
25	Debugando a macro .....	71
26	Janela de variáveis .....	75
27	Fórmulas do Excel no VBA .....	77
28	Usando as fórmulas do Excel no VBA .....	79

## Módulo 3 Estruturas avançadas do VBA

29	Introdução .....	81
30	Estrutura If: Explicação .....	82
31	Exemplo If .....	83
32	Estrutura If ... Elseif: Explicação .....	85
33	Exemplo If ... Elseif .....	86
34	Exemplo If ... Elseif (Parte 2) .....	88
35	If com várias condições (And) .....	89
36	If com várias condições (Or) .....	91

# Conteúdo

## Módulo 3 Estruturas avançadas do VBA

37	Do Until .....	92
38	Do Until: Exercício (Parte 1) .....	94
39	Do Until: Exercício (Parte 2) .....	97
40	Do Until: Exercício (Parte 3) .....	99
41	Do While .....	100
42	Do While: Exercício (Parte 1) .....	102
43	Do While: Exercício (Parte 2) .....	104
44	Do While: Exercício (Parte 3) .....	106
45	Do While: Exercício (Parte 4) .....	108
46	Combinação de estruturas (Parte 1) .....	109
47	Combinação de estruturas (Parte 2) .....	111
48	For .....	113
49	For: Exemplo .....	115
50	For: Exercício .....	116
51	For: Descobrindo a última linha da tabela .....	118
52	For: Exercício 2 .....	123
53	For: Descobrindo a última coluna da tabela .....	125
54	For: Exercício 3 .....	126
55	Compilação: Excluindo linhas da tabela .....	129
56	Desafio Compilação de Códigos: Explicação .....	136
57	Desafio Compilação de Códigos: Parte 1 .....	137

## Módulo 3 Estruturas avançadas do VBA

58	Desafio Compilação de Códigos: Parte 2 .....	141
59	Desafio Compilação Grifes: Explicação .....	142
60	Desafio Compilação Grifes: Parte 1 .....	144
61	Desafio Compilação Grifes: Parte 2 .....	147
62	Desafio Compilação Grifes: Parte 3 .....	153
63	Desafio Compilação Grifes: Parte 4 .....	156
64	Desafio Compilação Grifes: Parte 5 .....	161
65	Exemplo estrutura If e Select Case .....	165
66	Estrutura Select Case: Explicação .....	168
67	Exemplo Select Case .....	170

## Módulo 4 Trabalhando com objetos

68	Introdução .....	171
69	Set: Definindo células como variáveis .....	176
70	O que o Set muda na prática .....	177
71	Usando o Set com abas .....	181
72	Estrutura For Each: Explicação .....	184
73	For Each: Percorrendo células .....	187
74	For Each: Percorrendo abas .....	188
75	Compilação For Each 1: Explicação .....	190

# Conteúdo

## Módulo 4 Trabalhando com objetos

76	Compilação For Each 1: Resolução .....	191
77	Compilação For Each 2: Explicação .....	197
78	Compilação For Each 2: Resolução (Parte 1) .....	199
79	Compilação For Each 2: Resolução (Parte 2) .....	201
80	Compilação For Each 2: Resolução (Parte 3) .....	204
81	Compilação For Each: Padronizando Gráficos .....	205

## Módulo 5 Fórmulas de texto e data no VBA

82	Introdução .....	211
83	Fórmula Left .....	212
84	Fórmula Right .....	214
85	Fórmula Mid .....	216
86	Combinação de fórmulas de texto (Left + InStr) .....	218
87	Combinação de fórmulas de texto (Mid + InStr) .....	221
88	Arrumar textos e concatenar no VBA (Trim e &) .....	222
89	Fórmulas de data e hora .....	224
90	Fórmulas Date, Time e Now .....	225

## Módulo 6 Boxes

91	Introdução .....	226
92	Criando uma MsgBox simples .....	229
93	Criando uma MsgBox com botões .....	231
94	Personalizando a MsgBox .....	233
95	Exercício MsgBox .....	235
96	InputBox .....	237
97	Criando uma InputBox .....	239
98	Exercício InputBox: Explicação .....	241
99	Exercício InputBox: Solução .....	242
100	Exercício MsgBox + InputBox: Explicação .....	243
101	Exercício MsgBox + InputBox: Solução .....	244
102	Exercício Cadastro Funcionários: Explicação .....	245
103	Exercício Cadastro Funcionários: Solução .....	246

## Módulo 7 Exercícios

104	Introdução .....	248
105	Desafio Compilação Concessionárias: Explicação .....	249
106	Desafio Compilação Concessionárias: Parte 1 .....	251
107	Desafio Compilação Concessionárias: Parte 2 .....	252
108	Desafio Compilação Concessionárias: Parte 3 .....	253

# Conteúdo

## Módulo 7 Exercícios

109	Desafio Compilação Concessionárias: Parte 4 .....	255
110	Desafio Compilação Concessionárias: Parte 5 .....	256
111	Desafio Compilação Concessionárias: Ajustes finais .....	258
112	Desafio Compilação Concessionárias: Correções .....	260
113	Desafio Compilação Plataformas: Explicação .....	263
114	Desafio Compilação Plataformas: Parte 1 .....	265
115	Desafio Compilação Plataformas: Parte 2 .....	266
116	Desafio Compilação Plataformas: Parte 3 .....	267
117	Desafio Compilação Plataformas: Parte 4 .....	268
118	Desafio Compilação Plataformas: Otimizando o código ...	269
119	Otimizando o código: Application.ScreenUpdating .....	270
120	Otimizando o código: Application.Calculation .....	271
121	Desafio Compilação Plataformas: Otimizando o código ...	272

## Módulo 8 Tratamento de Erros

122	Introdução .....	274
123	On Error Resume Next: Exemplo 1 .....	276
124	On Error Resume Next: Exemplo 2 .....	280
125	On Error GoTo: Exemplo .....	283

## Módulo 9 Function

126	Introdução .....	286
127	Criando uma soma simples .....	289
128	Exemplo MeuConcatenar .....	291
129	Exemplo Salário Funcionários .....	294
130	Usando Function dentro de Sub .....	297
131	Executando a Function em várias abas .....	299

## Módulo 10 Trabalhando com vários arquivos

132	Hierarquia do VBA .....	301
133	Exercício arquivos .....	306
134	Abrindo arquivos pelo VBA .....	307
135	Exercício arquivos .....	309
136	Registrando informações em outro arquivo .....	310
137	Declarando um Workbook com o Set .....	313
138	Exercício Estoque .....	314
139	Resolução Exercício Estoque: Parte 1 .....	315
140	Resolução Exercício Estoque: Parte 2 .....	316
141	Resolução Exercício Estoque: Parte 3 .....	317
142	Resolução Exercício Estoque: Parte 4 .....	318
143	Percorrendo arquivos da pasta .....	319

# Conteúdo

## Módulo 10 Trabalhando com vários arquivos

144	Percorrendo arquivos da pasta: Parte 1 .....	320
145	Percorrendo arquivos da pasta: Parte 2 .....	321
146	Percorrendo arquivos da pasta: Parte 3 .....	322
147	Percorrendo arquivos da pasta: Parte 4 .....	323
148	Percorrendo arquivos da pasta: Parte 5 .....	324
149	Percorrendo arquivos da pasta: Parte 6 .....	325
150	Percorrendo arquivos da pasta: Parte 7 .....	327

## Módulo 11 Eventos no VBA

151	Introdução .....	328
152	Com criar eventos no VBA (Parte 1) .....	329
153	Com criar eventos no VBA (Parte 2) .....	335
154	Evento Activate: Atualizando Tabela Dinâmica .....	337
155	Evento Change: Folha de Ponto (Explicação) .....	341
156	Evento Change: Entendendo o Target .....	343
157	Evento Change: Folha de Ponto (Resolução Pt. 1) .....	344
158	Evento Change: Folha de Ponto (Resolução Pt. 2) .....	345
159	Evento Change: Folha de Ponto (Resolução Pt. 3) .....	346
160	Evento Change: Folha de Ponto (Resolução Pt. 4) .....	348
161	Evento Change: Desafio Projetos (Explicação) .....	349

## Módulo 11 Eventos no VBA

162	Evento Change: Desafio Projetos (Resolução Pt. 1) .....	350
163	Evento Change: Desafio Projetos (Resolução Pt. 2) .....	351
164	Evento Change: Desafio Projetos (Resolução Pt. 3) .....	352
165	Evento Change: Desafio Projetos (Resolução Pt. 4) .....	353
166	Evento Change: Desafio Projetos (Resolução Pt. 5) .....	354
167	Evento Change: Desafio Projetos (Resolução Pt. 6) .....	356
168	Evento Change: Desafio Projetos (Resolução Pt. 7) .....	358
169	Evento Change: Formatação de Gráfico (Explicação) .....	359
170	Evento Change: Formatação de Gráfico (Res. Pt. 1) .....	360
171	Evento Change: Formatação de Gráfico (Res. Pt. 2) .....	364
172	Eventos de Arquivo (Explicação) .....	367
173	Eventos de Arquivo: Exemplo Dashboard (Parte 1) .....	368
174	Eventos de Arquivo: Exemplo Dashboard (Parte 2) .....	369

# Conteúdo

## Módulo 12 UserForms

175	Introdução .....	372
176	Criando um UserForm no VBA .....	373
177	Rótulo e Caixa de Texto (Parte 1) .....	380
178	Rótulo e Caixa de Texto (Parte 2) .....	382
179	Botão de Opção .....	383
180	Botões de Comando e Imagem .....	384
181	Caixa de Combinação .....	385
182	Preenchendo a Caixa de Combinação (Parte 1) .....	388
183	Preenchendo a Caixa de Combinação (Parte 2) .....	390
184	Formatando a Base de Dados (Parte 1) .....	395
185	Formatando a Base de Dados (Parte 2) .....	397
186	Botão Cancelar .....	398
187	Excluir Funcionário (Parte 1) .....	400
188	Excluir Funcionário (Parte 2) .....	404
189	Excluir Funcionário (Parte 3) .....	406
190	Registro de Vendas (Explicação) .....	408
191	Construindo o Layout do Formulário .....	412
192	Inicializando o Formulário .....	420
193	Preenchendo Caixa de Listagem e Caixa de Combinação .....	423
194	Exibindo o valor do desconto .....	427

## Módulo 12 UserForms

195	Calculando o Preço Final da Moto (Explicação) .....	430
196	Calculando o Preço Final da Moto (Parte 1) .....	431
197	Calculando o Preço Final da Moto (Parte 2) .....	432
198	Calculando o Preço Final da Moto (Parte 3) .....	433
199	Calculando o Preço Final da Moto (Parte 4) .....	434
200	Calculando o Preço Final da Moto (Parte 5) .....	435
201	Calculando o Preço Final da Moto (Parte 6) .....	436
202	Registrando a Venda da Moto (Parte 1) .....	437
203	Registrando a Venda da Moto (Parte 2) .....	440
204	Registrando a Venda da Moto (Parte 3) .....	441
205	Registrando a Venda da Moto (Parte 4) .....	442
206	Registrando a Venda da Moto (Parte 5) .....	443
207	Registrando a Venda da Moto (Parte 6) .....	446
208	Registrando a Venda da Moto (Parte 7) .....	447
209	Registrando a Venda da Moto (Parte 8) .....	448
210	Botão Cancelar .....	449

# Conteúdo

## Módulo 13 Integração com Word

211	Abrindo um Documento em Word do Zero .....	452
212	Registrando Informações no Word .....	454
213	Manipulando os Textos .....	455
214	Escrevendo Várias Informações .....	456
215	Registrando Atividades dos Funcionários (Parte 1) .....	458
216	Registrando Atividades dos Funcionários (Parte 2) .....	461
217	Salvando o Arquivo .....	462
218	Apagando arquivo existente e salvando por cima .....	463
219	Geração de Contratos (Explicação) .....	464
220	Configurações Iniciais .....	467
221	Geração Automática de Contratos (Parte 1) .....	468
222	Geração Automática de Contratos (Parte 2) .....	471
223	Geração Automática de Contratos (Parte 3) .....	473
224	Geração Automática de Contratos (Parte 4) .....	474
225	Geração Automática de Contratos (Parte 5) .....	475
226	Geração Automática de Contratos (Parte 6) .....	477
227	Criação de Relatório Automático (Explicação) .....	478
228	Criação de Relatório Automático (Parte 1) .....	485
229	Criação de Relatório Automático (Parte 2) .....	488
230	Criação de Relatório Automático (Parte 3) .....	490

## Módulo 13 Integração com Word

231	Criação de Relatório Automático (Parte 4) .....	492
232	Criação de Relatório Automático (Parte 5) .....	495
233	Criação de Relatório Automático (Parte 6) .....	499
234	Criação de Relatório Automático (Parte 7) .....	501
235	Criação de Relatório Automático (Parte 8) .....	503
236	Criação de Relatório Automático (Parte 9) .....	511
237	Criação de Relatório Automático (Parte 10) .....	514
238	Criação de Relatório Automático (Parte 11) .....	523
239	Criação de Relatório Automático (Parte 12) .....	527
240	Criação de Relatório Automático (Parte 13) .....	531
241	Criação de Relatório Automático (Parte 14) .....	533

# Conteúdo

## Módulo 14 Integração com Outlook

242 Criando um E-mail no Outlook pelo VBA .....	536
243 Preenchendo e enviando o primeiro e-mail .....	540
244 Criando Assinatura Padrão no Outlook .....	542
245 Enviando e-mail com assinatura .....	548
246 Enviando vários e-mails .....	550
247 Colocando anexo no e-mail .....	553
248 HTML básico para formatação de E-mail (Parte 1) .....	556
249 HTML básico para formatação de E-mail (Parte 2) .....	558
250 HTML básico para formatação de E-mail (Parte 3) .....	559
251 Organizando o e-mail em parágrafos .....	560
252 Formatando o corpo do e-mail .....	562
253 Outros tipos de formatação de fonte .....	564

## Módulo 15 Integração com PowerPoint

254 Ferramenta 1: Dashboard no PPT .....	569
255 Criando a apresentação e o botão de controle .....	572
256 Preenchendo a Caixa de Combinação por VBA .....	576
257 Usar VBA para mudar Excel e PPT .....	578
258 Erros comuns e atualizando .....	581
259 Ferramenta 2: Criar apresentações com gráficos .....	582
260 Criar PPT e colar gráfico .....	585
261 Adicionar slide ou criar novo PPT .....	586
262 Redimensionar e ajeitar criação de gráfico .....	587
263 Ferramenta 3: Gerando Relatório Automático .....	589
264 Criar lógica de variáveis e Function GerarPDF .....	592
265 Abrir PPT existente .....	593
266 Editar texto no PPT pelo VBA .....	594
267 Salvar como PDF e Formatar Números .....	597
268 Fechar PPT forçadamente e finalizando .....	599
269 Considerações Finais .....	601

# BONS ESTUDOS

Dúvidas ou Feedbacks nos procure

# Módulo 1

# Introdução

O Visual Basic for Applications, ou simplesmente **VBA**, é uma linguagem de programação que pode ser usada no Microsoft Excel para dar mais opções de controle e edição de uma planilha. Basicamente, o VBA permite a criação de macros e a automatização de processos dentro das planilhas e tabelas utilizadas. Através destes códigos, é possível controlar qualquer ação executada dentro do Excel, permitindo total capacidade de manipular qualquer aspecto do programa em favor da criação de planilhas muito mais otimizadas.



Por meio de macros é possível ter o controle de comandos que vão desde a edição e formatação de células, controle das ferramentas da guia de menus de forma mais automática, criação de lógicas de programação com estruturas de repetição. Não só isso, também é possível realizar integração com outros programas da Microsoft, tais como: PowerPoint, Word e Outlook.

O principal benefício de uma macro é o ganho de tempo que ela proporciona, pois ela substitui todo o trabalho manual caracterizado pela execução de tarefas repetitivas e demoradas - que podem durar horas ou até dias - por códigos que executam estas mesmas tarefas em poucos minutos ou até mesmo em alguns segundos.

O VBA é uma linguagem de programação que não requer conhecimento prévio em Excel. É claro que certa afinidade com o programa irá facilitar o entendimento da lógica em si, mas será visto mais a frente que é possível iniciar o curso do completo zero, passando pelo básico, intermediário e indo até o avançado. Conforme avançamos, fatalmente vamos entendendo na prática diferentes conceitos do Excel em si.

Então pode ficar tranquilo que o conhecimento em Excel não é um pré-requisito, então você se sentirá totalmente capaz de aplicar os códigos conforme avançamos no conteúdo.



## 1. Planilha de controle de produtos com alerta de falta de estoque

O exemplo ao lado é um controle de produtos de uma empresa, levando em consideração seus diferentes aspectos, tais como marca/grife, dentre outros. Uma empresa precisa ter capacidade de controlar de forma ágil todos os seus produtos. Para isso é necessário criar diferentes tabelas resumo, frequentemente atualizadas, possibilitando a rápida tomada de decisão. Se fosse feito de forma manual, a princípio, este trabalho seria bem demorado pois haveria a necessidade de analisar milhares de produtos e separar cada um de acordo com as suas características, o que poderia levar horas e até dias.

Com VBA, conseguimos criar este controle facilmente, em segundos, e ainda podemos marcar, de forma automática, todos os produtos que estiverem com um estoque abaixo de um estoque mínimo aceitável, sendo capaz até de indicar em vermelho os produtos cujo estoque deve ser verificado.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Grife	Reserva													
2	Status	Coleção Antiga													
3	Estoque Mínimo	3500													
4															
5															
6															
7															
8															
9															
10															
11															
12															
13															

**Compilar Grifes**

Cor	Produto 1	Produto 2	Produto 3	Produto 4	Produto 5	Produto 6	Produto 7	Produto 8	Produto 9	Produto 10	Estoque
AMARELO											
BRANCO											
AZUL											
ROSA											
VERDE ESMERALDA											
VERMELHO											

A	B	C	D	E	F	G	H	I
1	Código	Grupo	Cor	Grife	Coleção	Estoque	Status	Grifes
2	02.020.235	Bota	ROSA	Osklen	13	498	Coleção Antiga	Reserva
3	02.020.242	Camiseta	AZUL	Hashtag	14	1540	Nova Coleção	Hashtag
4	02.020.242	Chinelo	BRANCO	Hashtag	02	546	Coleção Antiga	Osklen
5	02.020.250	Chinelo	AMARELO	Hashtag	03	451	Coleção Antiga	
6	02.030.232	Chinelo	BRANCO	Osklen	08	1941	Nova Coleção	
7	02.030.236	Sandália	BRANCO	Osklen	02	4000	Nova Coleção	
8	02.030.240	Bota	VERMELHO	Reserva	15	1778	Coleção Antiga	
9	02.030.245	Sandália	AMARELO	Reserva	28	37	Coleção Antiga	
10	02.030.249	Boné	BRANCO	Osklen	04	51	Coleção Antiga	
11	02.040.236	Sandália	VERMELHO	Reserva	08	1063	Coleção Antiga	

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Grife	Reserva												
2	Status	Coleção Antiga												
3	Estoque Mínimo	3500												
4														
5														
6														
7														
8														
9														
10														
11														
12														

**Compilar Grifes**

Cor	Produto 1	Produto 2	Produto 3	Produto 4	Produto 5	Produto 6	Produto 7	Produto 8	Produto 9	Produto 10	Estoque
AMARELO	02.030.245	02.060.230	02.090.243	02.110.233	02.150.237	04.070.250	04.130.230	06.040.237			5803
BRANCO	02.050.230	03.050.247	03.120.234	05.090.232	06.030.238	06.060.250					6094
AZUL	05.100.242	06.140.239									1272
ROSA	02.080.246	02.150.244									513
VERDE ESMERALDA	04.020.236	06.030.239									3461
VERMELHO	02.030.240	02.040.236	02.100.236	03.050.231							4773

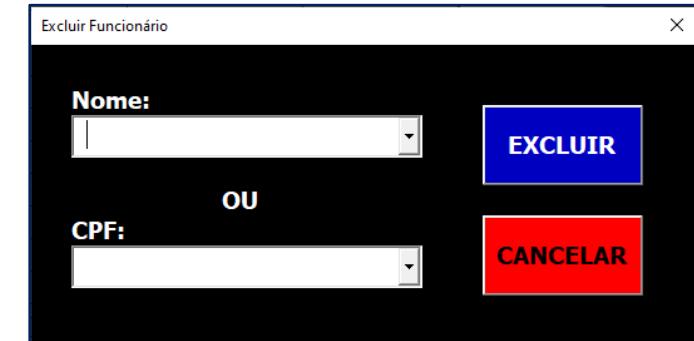
## 2. Cadastro de novos funcionários e exclusão de um funcionário

Outra tarefa comum é o cadastro ou exclusão de dados em uma planilha. Um recurso muito útil nestas situações é a criação de Userforms, que são formulários que permitem uma interação mais prática e visual entre a planilha e o usuário.

Ao lado temos um exemplo de cadastro de um novo funcionário da empresa ou exclusão de um novo funcionário. Cada botão abre uma caixinha na tela que pode ser preenchida facilmente por meio de botões e caixas de texto.

A ideia do Userform é tornar a interação muito mais intuitiva ao usuário, sem que ele precise necessariamente entender alguma coisa sobre Excel para utilizar a sua planilha. Esta mesma lógica pode ser utilizada em diferentes situações, como uma planilha de cadastro de vendas, ou de cadastro de clientes e serviços prestados, e por ai vai.

	A	B	C	D	E	F	G	H
1	Nome	Sexo	Área	CPF	Salário			
2	Adriane de Carvalho Gomes	Feminino	RH	082.442.274-27	R\$ 4.650,00			
3	Elvis de Freitas Derossi	Masculino	RH	104.030.731-00	R\$ 12.320,00			
4	Gabrielle Andrade da Silva	Feminino	Administrativo	081.419.765-67	R\$ 12.320,00			
5	Gustavo de Vasconcelos	Masculino	Administrativo	137.532.186-15	R\$ 16.300,00			
6	Isabella Ronfini	Feminino	RH	133.655.149-67	R\$ 12.320,00			
7	Izabelle Anderson	Feminino	Marketing	154.896.314-20	R\$ 15.000,00			
8	João Lira	Masculino	RH	152.896.317-24	R\$ 5.000,00			
9	João Martins	Masculino	Administrativo	187.542.124-52	R\$ 12.000,00			
10	Juan Fernandes	Masculino	RH	107.000.473-49	R\$ 16.300,00			
11	Júlio Fraga	Masculino	Compras	133.351.567-51	R\$ 7.200,00			
12	Leticia Mesquita	Feminino	Administrativo	136.725.629-63	R\$ 9.450,00			



## 3. Envio automático de e-mails pelo VBA

Como comentado anteriormente, é possível fazer integrações do VBA com outros programas além do Excel.

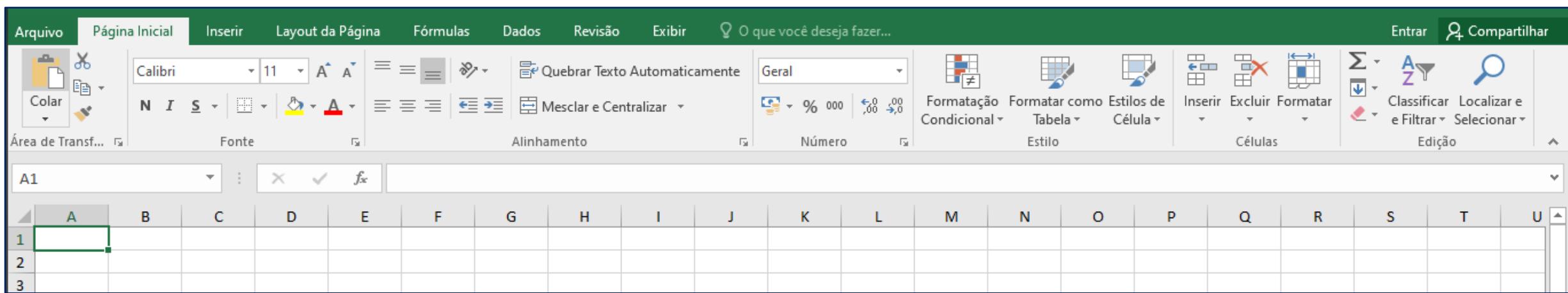
Ao lado, temos uma planilha que permite o envio automático de e-mails para diferentes destinatários, uma tarefa que para muitas pessoas faz parte do cotidiano e costuma ser algo bem exaustivo e demorado. Com o VBA, é possível automatizar este processo e com apenas um clique enviar uma série de e-mails de forma rápida e fácil.

Espero que com estes exemplos você tenha uma ideia de como o VBA poderá te ajudar no dia a dia. Todas essas ferramentas, bem como muitas outras, serão ensinadas no curso do completo zero, então fique tranquilo pois não vai demorar até você conseguir construir códigos e planilhas muito mais avançadas que essas. Seja bem vindo e tenha um ótimo curso!

	A	B	C	D	E	F
1	E-mail	Vendedor	Corpo			
2	<a href="mailto:joaoprira@poli.ufrj.br">joaoprira@poli.ufrj.br</a>	João Paulo	segue em anexo o relatório de vendas com o seu desempenho neste ano.			
3	<a href="mailto:sergio@hashtagtreinamentos.com">sergio@hashtagtreinamentos.com</a>	Sergio Tranjan	segue em anexo o relatório de vendas com o seu desempenho neste ano.			
4	<a href="mailto:jessica.hollander@uol.com.br">jessica.hollander@uol.com.br</a>	Jessica Hollander	segue em anexo o relatório de vendas com o seu desempenho neste ano.			
5	<a href="mailto:d.amorim.santos96@gmail.com">d.amorim.santos96@gmail.com</a>	Diego Amorim	segue em anexo o relatório de vendas com o seu desempenho neste ano.			
6	<a href="mailto:luiza.franca@gmail.com">luiza.franca@gmail.com</a>	Luiza França	segue em anexo o relatório de vendas com o seu desempenho neste ano.			
7						
8						

Enviar Relatórios

Finalmente, vamos dar início. Quando abrimos um arquivo Excel em branco, esta é a visualização inicial. Na parte superior, temos a lista de guias na Barra de Menus, que contém basicamente as guias: Arquivo, Página Inicial, Inserir, Layout da Página, Fórmulas, Dados, Revisão e Exibir. Para trabalhar com as macros, o primeiro passo é habilitar uma nova guia no Excel, a guia **Desenvolvedor**. Nela vamos encontrar todos os comandos que vão nos permitir trabalhar com as macros dentro do Excel.



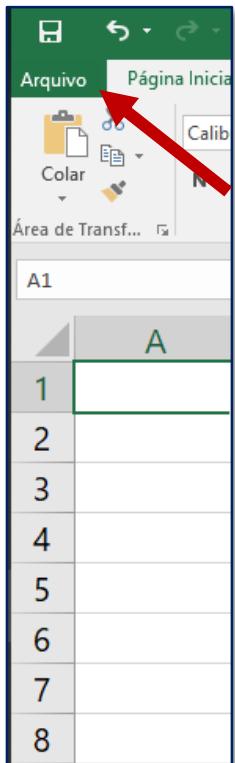
# Módulo 1 – Introdução – Habilitando a guia Desenvolvedor

7

Para habilitar esta nova guia, você pode seguir o passo a passo das imagens abaixo.

1

Clicar na guia **Arquivo**.



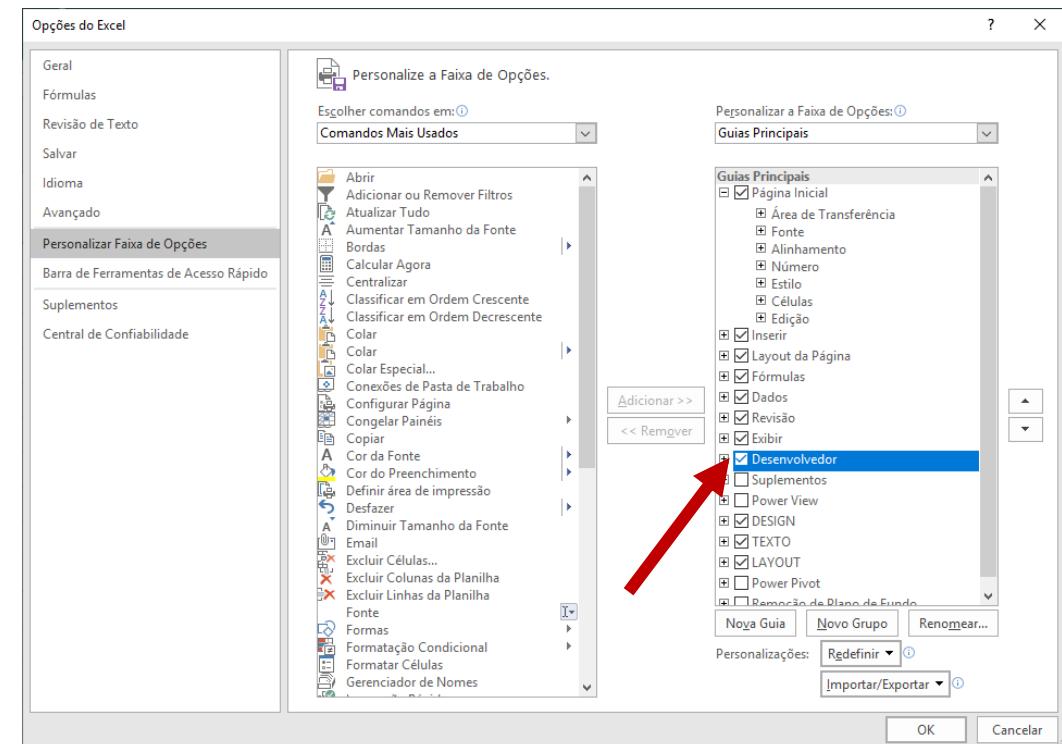
2

Clicar em **Opções**.



3

Marcar a guia **Desenvolvedor**.

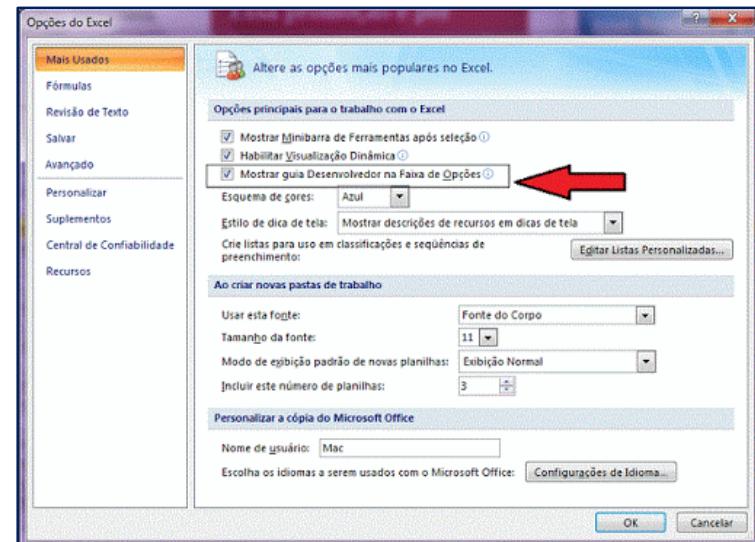


Ao terminar, deverá aparecer para você esta nova guia. Basicamente o que temos nela são as opções mostradas na imagem abaixo, que serão detalhadas mais a frente.



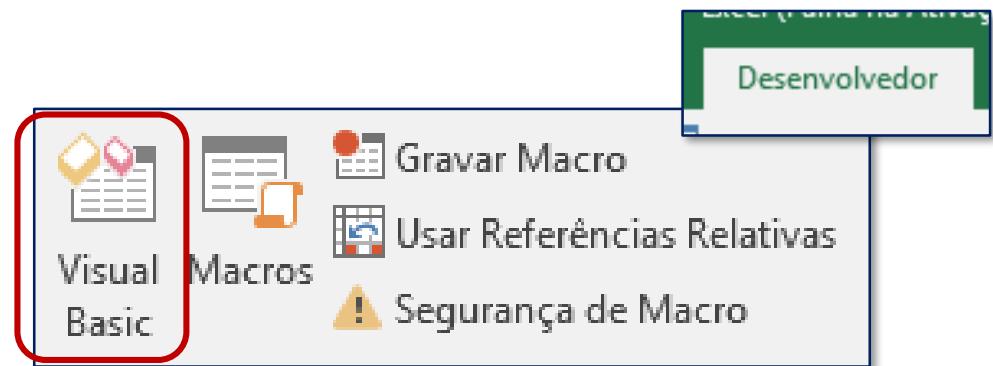
## ATENÇÃO !

Caso a sua versão do Excel seja 2010 ou anterior, você vai encontrar a opção de ativar a guia Desenvolvedor na guia Arquivo → Opções do Excel → Mais Usados!



Antes de começar, devemos entender que existem **duas formas** de criar códigos no VBA.

A primeira, mais avançada, é **criar os códigos diretamente no ambiente do VBA, do zero**. Isso requer experiência e um conhecimento mais avançado, além de uma certa boa memória da nossa parte para lembrar as estruturas lógicas de programação.



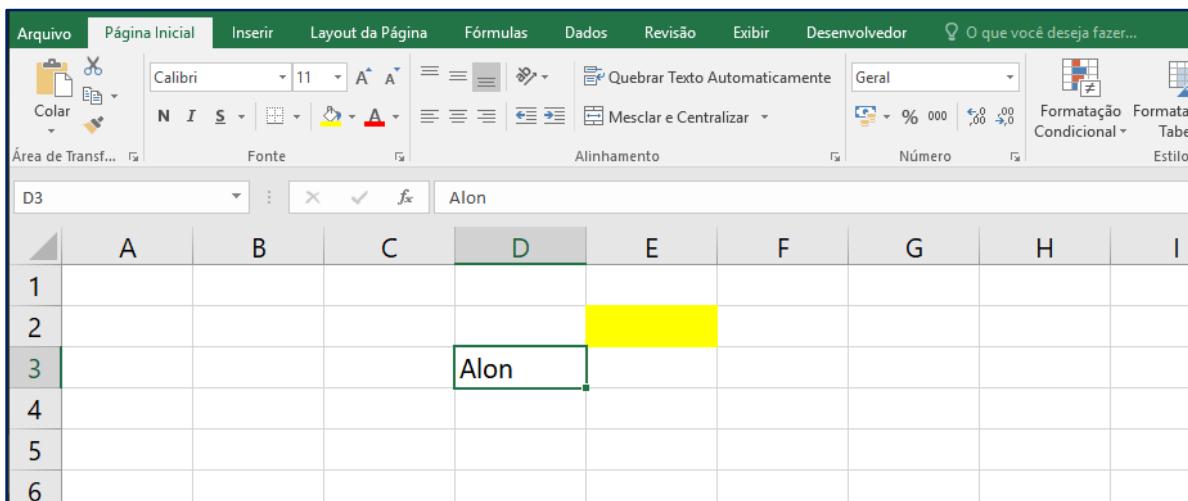
Para entrar no ambiente do VBA clicamos na opção **Visual Basic**.

The image shows the Microsoft Visual Basic for Applications (VBA) editor. A code module named 'primeira\_macro' is open, containing the following VBA code:

```
Sub primeira_macro()
    'O código abaixo escreve um texto na célula A1
    Range("A1").Value = "Essa é a minha primeira macro"
End Sub
```

Porém, é muito difícil lembrar de todos os códigos de cabeça. Além disso, como é a nossa primeira vez com o VBA, devemos pensar em uma outra maneira mais fácil para começar. Nesse caso, temos a **segunda opção** para criação de códigos, **que é por meio da Gravação de Macros**.

O que temos que entender nesse ponto é que qualquer ação que executamos no Excel (editar e formatar células, criar fórmulas, formatação condicional, etc.) possui um código VBA por trás. E para termos acesso a esses códigos, podemos simplesmente usar a gravação de macro.



**Exemplo:** Imagine que a gente queira saber qual é o código para pintar de amarelo a célula E2, ou então o código para escrever “Alon” na célula D3. Quando usamos o Excel para fazer isso, não precisamos nos preocupar com nenhum código por trás.

Porém, quando começamos a criar códigos em VBA, queremos saber como executar estas mesmas ações que executamos no Excel, só que utilizando uma lógica de programação dentro do VBA, visando a automatização das tarefas do dia a dia.

# Módulo 1 – Introdução – Gravando a primeira macro (Parte 1)

11

The screenshot shows a Microsoft Excel interface. The ribbon menu is visible at the top with tabs like Arquivo, Página Inicial, Inserir, Layout da Página, Fórmulas, Dados, Revisão, Exibir, Desenvolvedor, and O que você deseja fazer... The task pane on the right contains the following VBA code:

```
Sub Macro1()
    Range("C10").Select
    ActiveCell.Value = "Nome"
    Range("C11").Select
    ActiveCell.Value = "Idade"
    Range("C12").Select
    ActiveCell.Value = "Nascimento"
End Sub
```

The spreadsheet area shows rows 1 through 14. Cells C10, C11, and C12 are highlighted in orange. The formula bar shows "C10". The task pane also contains a note in bold black text:

**Grave uma macro que escreva seu nome na célula C10, a sua idade na célula C11 e a sua data de nascimento na célula C12. Vincule essa macro a um botão**

		Nome	
		Idade	
		Nascimento	

Vimos anteriormente que existem duas formas de criar códigos no VBA.

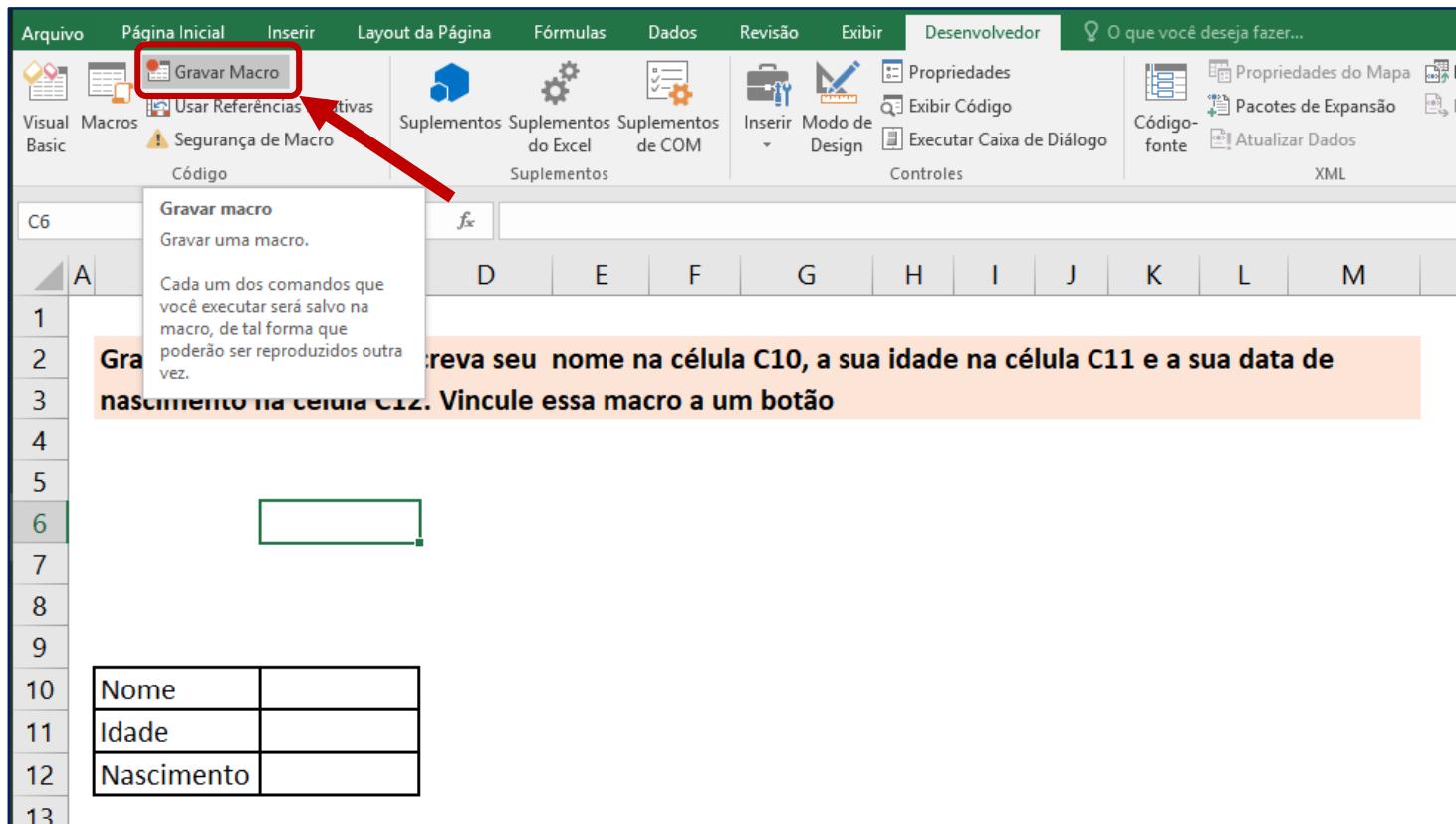
A primeira seria programando diretamente no ambiente VBA (que veremos com calma mais a frente) e a segunda é por meio da **Gravação de Macros**, que veremos bem passo a passo a partir de agora.

No exemplo ao lado queremos gravar uma macro para visualizar o código por trás de comandos simples: por exemplo, edição de células.

Vamos escrever o nome na célula C10, a idade na célula C11 e a data de nascimento na célula C12. Em seguida, vamos ver qual é o código por trás destes comandos. **Não faça ainda, antes precisamos começar a gravar a macro!**

# Módulo 1 – Introdução – Gravando a primeira macro (Parte 1)

12



The screenshot shows the Microsoft Excel ribbon with the 'Desenvolvedor' tab selected. In the 'Suplementos' group, the 'Gravar Macro' button is highlighted with a red box and a red arrow pointing to it. The Excel interface includes a status bar at the bottom with the text: 'Crieva seu nome na célula C10, a sua idade na célula C11 e a sua data de nascimento na célula C12. Vincule essa macro a um botão'.

Nome	
Idade	
Nascimento	

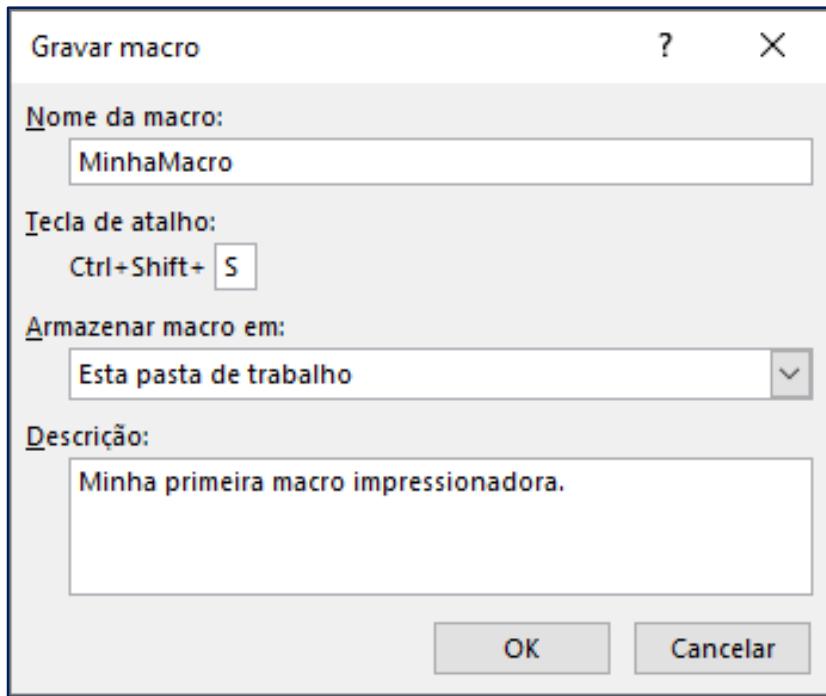
Aqui vale destacar que absolutamente todas as ações executadas serão gravadas no VBA. É como se o Excel escrevesse em um bloco de notas todo o passo a passo de cada ação executada, só que em forma de código de programação.

Por isso, para ficarmos com exatamente o mesmo código, o primeiro passo será selecionar a célula C6.

Isso não significa que toda a macro deverá começar assim. Isso é apenas para que o meu código fique exatamente igual ao seu.

Enfim, para começar gravando uma macro, primeiro clicamos na guia **Desenvolvedor**.

Em seguida, é só clicar em **Gravar Macro**.



Ao clicar em **Gravar Macro**, a janela ao lado será mostrada. Começamos preenchendo o “Nome da macro”. Esse nome não pode conter espaços, caracteres especiais (asteriscos, operadores como ‘+’, ‘=’, parênteses, etc.) e nem começar com números.

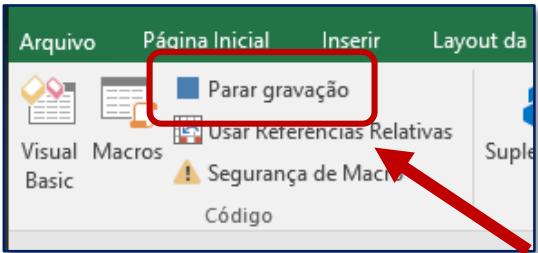
No campo “Tecla de atalho” podemos utilizar um atalho para executar automaticamente a nossa macro, como por exemplo, Ctrl + Shift + S.

Por fim, no campo “Descrição”, podemos deixar algum comentário ou alguma observação sobre aquela macro. Estes comentários não vão influenciar em nada durante a execução da macro, como veremos mais a frente.

Em seguida, para confirmar, basta clicar em ‘Ok’.

## Se liga!

As macros no VBA nada mais são do que uma sequência de ações que são executadas toda vez que o usuário quiser. Para executar essas ações, podemos utilizar um atalho. O cuidado que devemos tomar é que o Excel já possui uma infinidade de atalhos. Por conta disso, para não correr o risco de sobrepor algum atalho já existente, podemos usar as letras em maiúscula, usando a combinação SHIFT + S, por exemplo. Assim é muito mais difícil sobrepor algum atalho que comece com CTRL + SHIFT pois existem poucos casos desse tipo. De qualquer forma, veremos mais a frente que a melhor opção para executar uma macro é criando botões.



Repare que agora o botão de Gravar Macro virou o botão **Parar gravação**.

A partir de agora, tudo o que a gente fizer será gravado através de um código em VBA.

Nome	Alon
Idade	24
Nascimento	19/04/1995

Iniciamos então a nossa macro e escrevemos nome, idade e data de nascimento. Finalizamos clicando no botão **Parar gravação**. Feito isso, voltaremos a ter o botão **Gravar macro**.

Para executar esta macro, você pode simplesmente selecionar as células C10, C11 e C12, deletar usando a tecla DELETE e em seguida usar o atalho CTRL + SHIFT + S que atribuímos à macro. Feito isso, o Excel irá escrever exatamente os valores digitados.

# Módulo 1 – Introdução – Gravando a primeira macro (Parte 1)

15

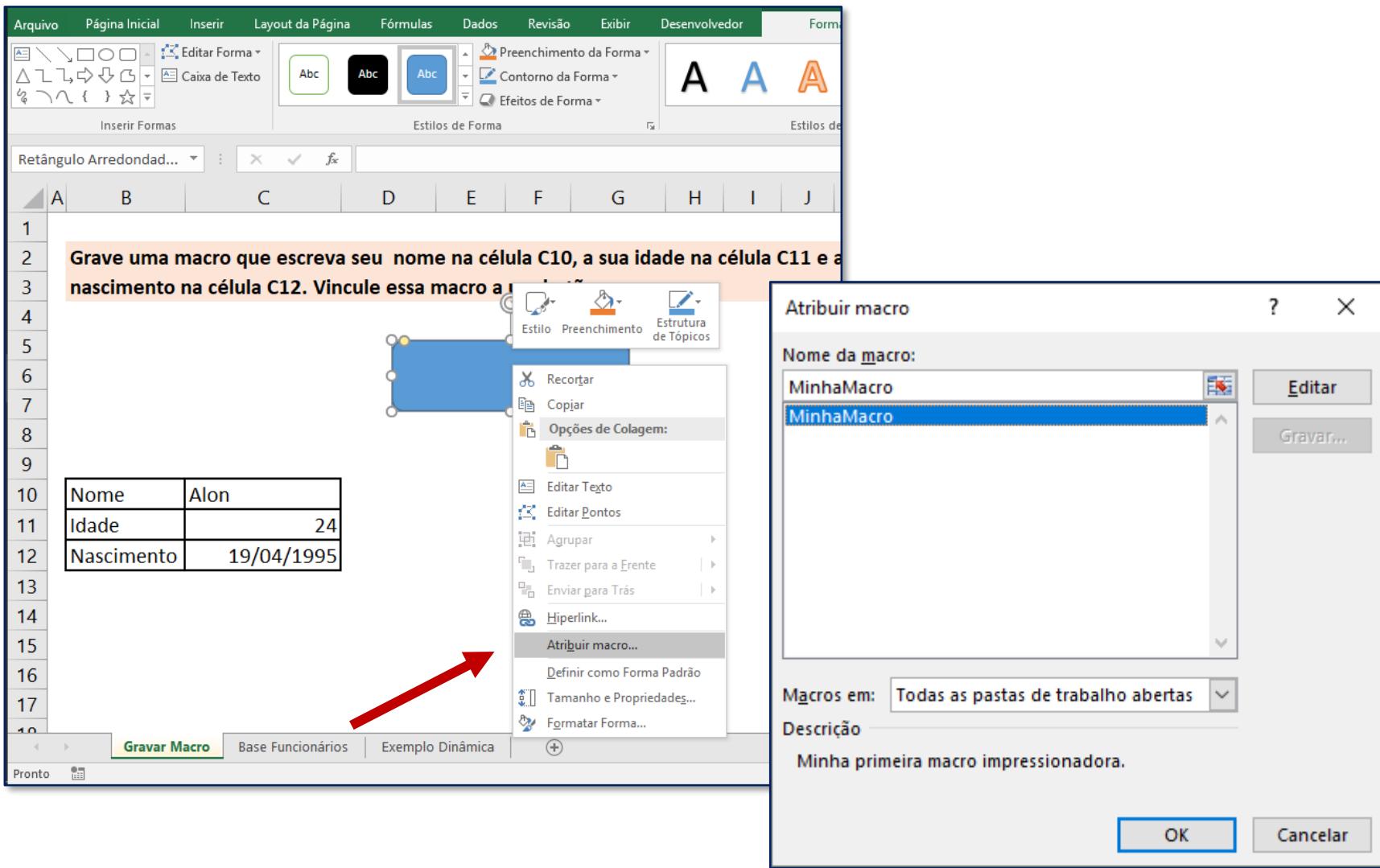
The screenshot shows a Microsoft Excel interface. The ribbon at the top has tabs: Arquivo, Página Inicial, Inserir, Layout da Página, Fórmulas, Dados, Revisão, Exibir, Desenvolvedor, and O que você deseja fazer... A red arrow points to the 'Inserir' tab. The main area shows a table with columns A, B, C and rows 1 to 17. Row 1 contains the header: 'Grave uma macro que escreva sua idade na célula C11 e a sua data de nascimento na célula C12. Vira'. Rows 10, 11, and 12 contain data: Nome (Alon), Idade (24), and Nascimento (19/04/1995). A context menu for drawing shapes is open over the table, with 'Retângulo Arredondado' selected. The status bar at the bottom shows 'Gravar Macro'.

Mais interessante do que usar um atalho para executar a macro é clicar em um botão, como mencionado anteriormente.

Para inserir um botão, podemos criar uma forma pela guia **Inserir** para associar à nossa macro.

# Módulo 1 – Introdução – Gravando a primeira macro (Parte 1)

16



Após escolher a forma desejada, basta desenhar na planilha com o mouse.

Feito isso, clique na forma com o botão direito e depois vá na opção **Atribuir macro...**

Na janela que abrir, selecione a macro criada: *MinhaMacro*.

# Módulo 1 – Introdução – Gravando a primeira macro (Parte 1)

17

The screenshot shows a Microsoft Excel interface with the ribbon menu at the top. A task pane on the left contains the following text:

Grave uma macro que escreva seu nome na célula C10, a sua idade na célula C11 e a sua data de nascimento na célula C12. Vincule essa macro a um botão.

Below the task pane, there is a blue rectangular button placeholder and a green rectangular button placeholder. In the bottom-left corner of the worksheet area, there is a small green rectangular button.

A table is present in the worksheet area:

Nome	Alon
Idade	24
Nascimento	19/04/1995

Para executar a macro agora basta apagar novamente os textos escritos nas células C10, C11 e C12 e depois clicar no botão.

# Módulo 1 – Introdução – Gravando a primeira macro (Parte 2)

18

The screenshot shows a Microsoft Excel interface. At the top, the ribbon is visible with the 'Desenvolvedor' tab selected. A red box highlights the 'Macros' icon in the 'Visualizar Basic' group. In the center, a 'Macro' dialog box is open, listing a single macro named 'MinhaMacro'. Below the dialog box, a table contains the following data:

Nome	Alon
Idade	24
Nascimento	19/04/1995

At the bottom of the screen, a taskbar shows tabs for 'Gravar Macro', 'Base Funcionários', and 'Exemplo Dinâmica'. The 'Gravar Macro' tab is currently active.

Entendemos como gravar uma macro. Agora temos que entender qual é o código que o VBA gravou para saber o passo a passo do que fizemos durante a gravação de macro e saber como escrever em cada célula sempre que a gente executar a macro.

Para ver o código criado, vamos na guia **Desenvolvedor** → **Macros** → **Selecionar MinhaMacro** → **Clicar em Editar**.

# Módulo 1 – Introdução – Gravando a primeira macro (Parte 2)

19

The screenshot shows the Microsoft Visual Basic for Applications (VBA) editor window. The title bar reads "Microsoft Visual Basic for Applications - Gravação de Macro.xlsx - [Módulo1 (Código)]". The menu bar includes Arquivo, Editar, Exibir, Inserir, Formatar, Depurar, Executar, Ferramentas, Suplementos, Janela, and Ajuda. The toolbar has various icons for file operations. The Project Explorer on the left shows "VBAProject (Gravação de Macro)" with "Microsoft Excel Objetos" expanded, listing "EstaPasta\_de\_trabalho", "Plan8 (Exemplo Dinâmica)", "Planilha1 (Base Funcionários)", and "Sheet13 (Gravar Macro)". A "Módulos" folder contains "Módulo1". The Properties window on the left is titled "Propriedades - Módulo1" and shows "(Name) Módulo1". The main code editor window is titled "MinhaMacro" and contains the following VBA code:

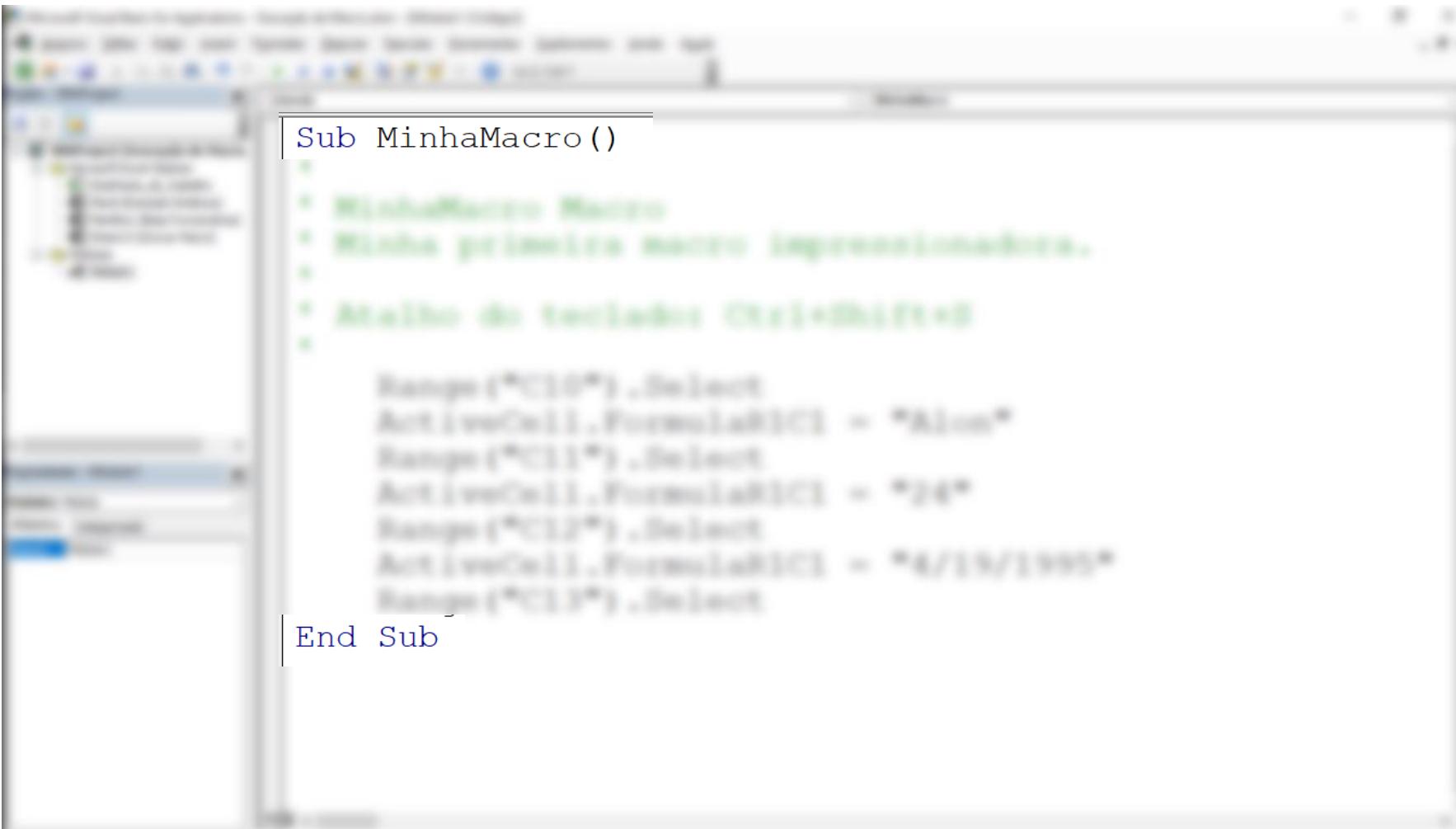
```
Sub MinhaMacro()
    ' MinhaMacro Macro
    ' Minha primeira macro impressionadora.

    ' Atalho do teclado: Ctrl+Shift+S

    Range("C10").Select
    ActiveCell.FormulaR1C1 = "Alon"
    Range("C11").Select
    ActiveCell.FormulaR1C1 = "24"
    Range("C12").Select
    ActiveCell.FormulaR1C1 = "4/19/1995"
    Range("C13").Select
End Sub
```

O código criado pelo VBA para executar o passo a passo de escrever nas células está mostrado ao lado.

Vamos agora entender o que significa cada parte deste código:

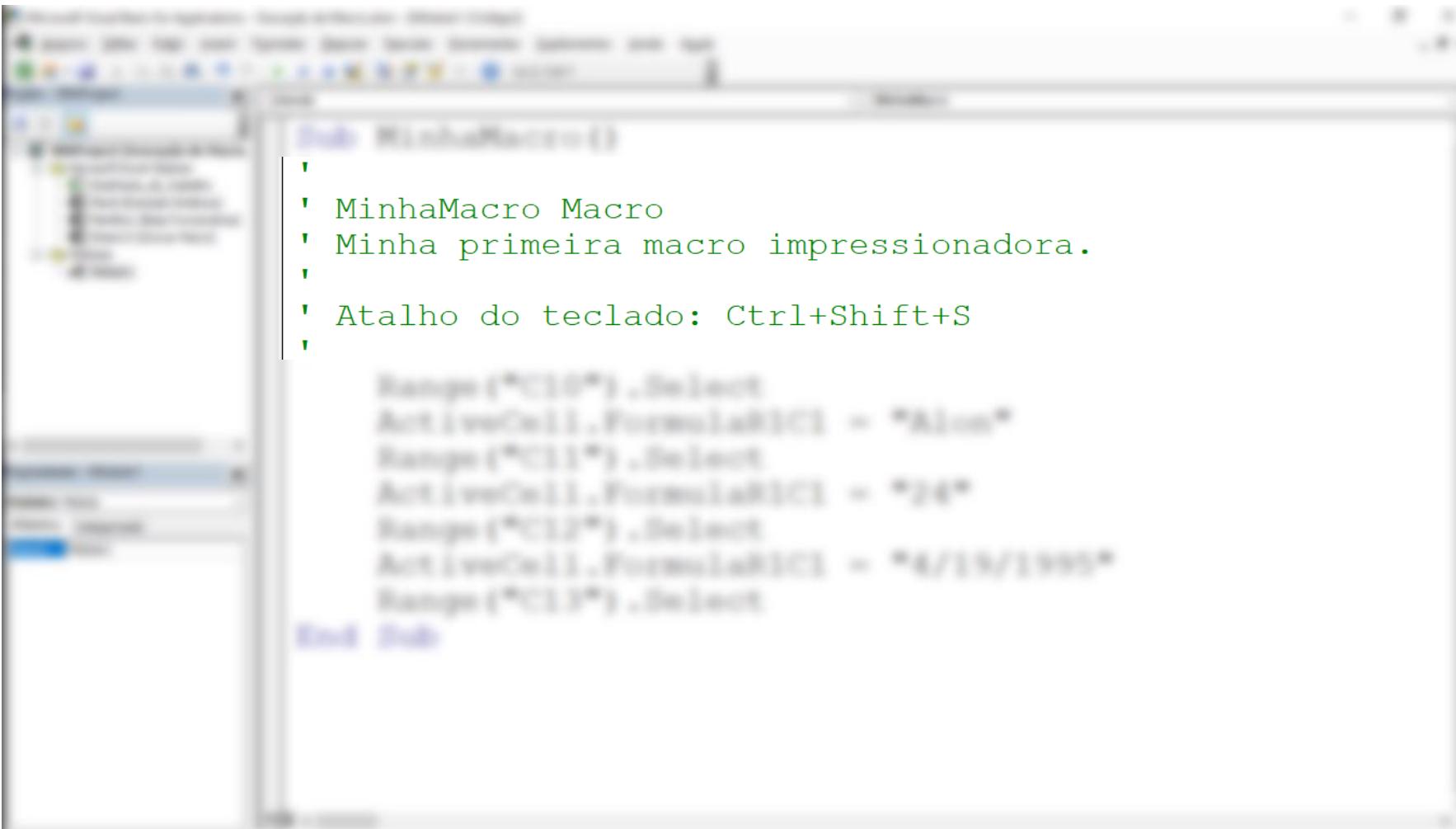


```
Sub MinhaMacro()
    ' MinhaMacro Macro
    ' Minha primeira Macro. Segue o comando.
    ' Atalho de teclado: Ctrl+Shift+M
    Range("C100").Select
    ActiveCell.FormulaR1C1 = "Bom"
    Range("C111").Select
    ActiveCell.FormulaR1C1 = "Legal"
    Range("C122").Select
    ActiveCell.FormulaR1C1 = "6/19/1999"
    Range("C133").Select
End Sub
```

Todo código VBA começa com **Sub**, de sub-rotina, e termina com **End Sub**.

Uma sub-rotina nada mais é do que uma sequência de comandos que serão executados sempre que solicitado.

Seguido de **Sub**, temos sempre o nome da macro, seguido também por uma abertura e fechamento de parênteses.



The screenshot shows the Microsoft Word ribbon at the top. Below it, the VBA editor window is open, displaying the recorded macro. The code is as follows:

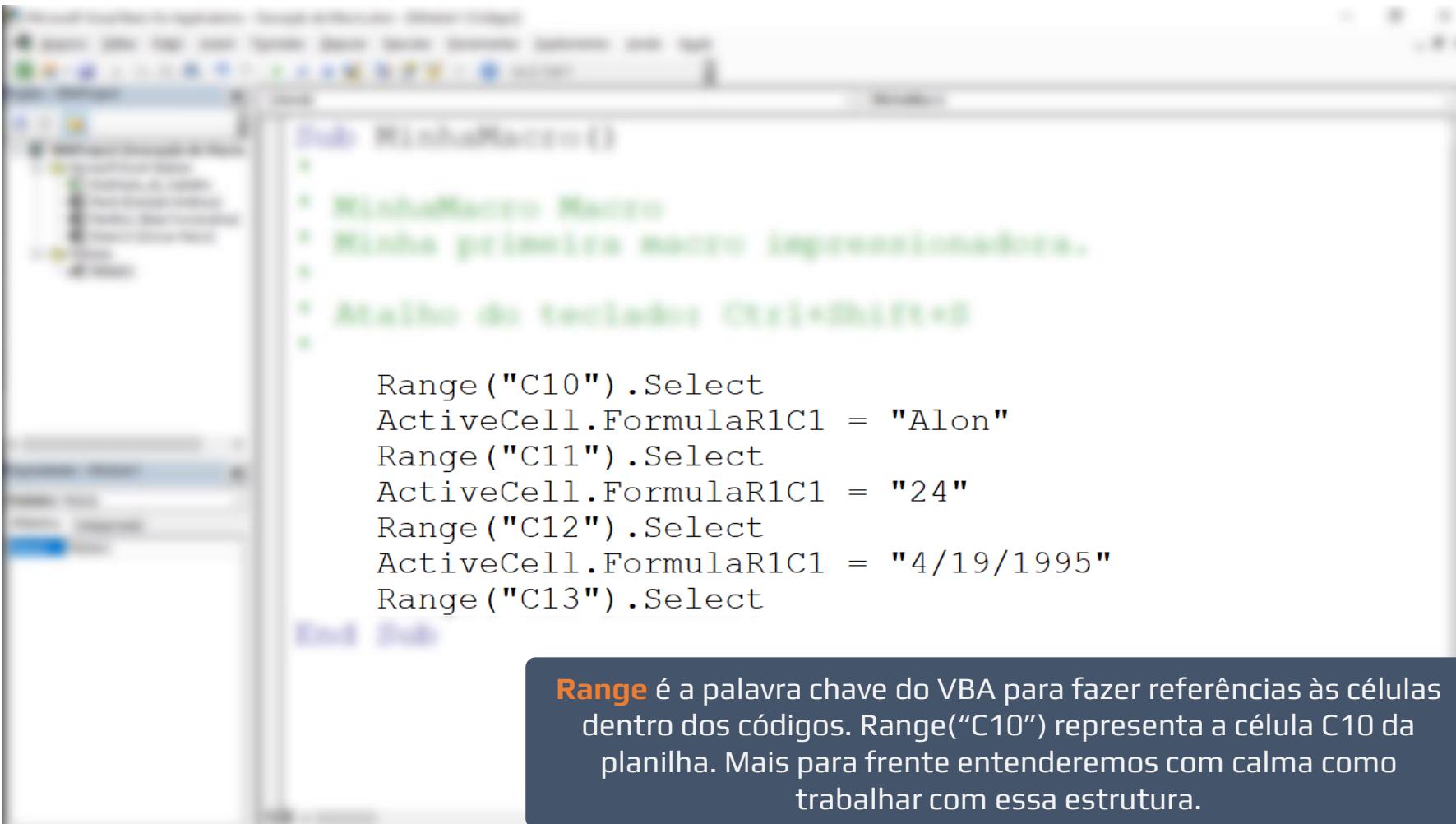
```
' MinhaMacro Macro
' Minha primeira macro impressionadora.
' Atalho do teclado: Ctrl+Shift+S

Range("C107").Select
ActiveCell.FormulaR1C1 = "Bom"
Range("C111").Select
ActiveCell.FormulaR1C1 = "Legal"
Range("C112").Select
ActiveCell.FormulaR1C1 = "Legal"
Range("C113").Select
End Sub
```

Logo abaixo da palavra Sub, em verde, temos os comentários do código. Aqui é onde ele escreve aquele texto da Descrição no momento em que clicamos em Gravar Macro.

Um comentário no VBA não influencia em nada no código, serve apenas para documentar o que foi feito.

Sempre que quiser inserir comentários no seu código, é só iniciar a linha com um apóstrofo (aspas simples).



```
Sub Mikaela()
    Range("C10").Select
    ActiveCell.FormulaR1C1 = "Alon"
    Range("C11").Select
    ActiveCell.FormulaR1C1 = "24"
    Range("C12").Select
    ActiveCell.FormulaR1C1 = "4/19/1995"
    Range("C13").Select
End Sub
```

**Range** é a palavra chave do VBA para fazer referências às células dentro dos códigos. Range("C10") representa a célula C10 da planilha. Mais para frente entenderemos com calma como trabalhar com essa estrutura.

Logo após os comentários é onde de fato começa o nosso código.

Repare como ele de certa forma é intuitivo. Lembra que a primeira coisa que fizemos **após iniciar a gravação** foi selecionar a célula C10 para escrever o nome?

Então, a primeira linha é *Range("C10").Select* para selecionar a célula C10. Já para escrever, o comando é *ActiveCell.FormulaR1C1 = "Alon"*. E assim vai.

# Módulo 1 – Introdução – Gravando a primeira macro (Parte 2)

23

The screenshot shows the Microsoft Visual Basic for Applications (VBA) editor window. The title bar reads "Microsoft Visual Basic for Applications - Gravação de Macro.xlsx [módulo1 (Código)]". The menu bar includes Arquivo, Editar, Exibir, Inserir, Formatar, Depurar, Executar, Ferramentas, Suplementos, Janela, and Ajuda. A red arrow points to the "Executar" button in the toolbar. The Project Explorer on the left shows "VBAProject (Gravação de Macro)" with "Módulo1" selected. The code editor contains the following VBA code:

```
Sub MinhaMacro()
    ' MinhaMacro Macro
    ' Minha primeira macro impressionadora.
    ' Atalho do teclado: Ctrl+Shift+S

    Range("D10").Select
    ActiveCell.FormulaR1C1 = "Alon"
    Range("D11").Select
    ActiveCell.FormulaR1C1 = "24"
    Range("D12").Select
    ActiveCell.FormulaR1C1 = "4/19/1995"
    Range("D13").Select
End Sub
```

The status bar at the bottom says "Na planilha...". Below the status bar is a small table:

10	Nome	Alon	Alon
11	Idade	24	24
12	Nascimento	19/04/1995	#####

The cell containing "19/04/1995" has a green border, indicating it is selected.

Importante ter em mente também que esse código é totalmente editável. Se quiséssemos escrever as mesmas informações na coluna D em vez da coluna C, bastaria fazer a troca ao lado.

E se quisermos executar a macro dentro do próprio ambiente VBA, podemos clicar no botão de play (▶) ou usar o atalho F5 (ou Fn + F5).

*Obs: caso apareça essas # na sua planilha, é só aumentar o tamanho da coluna D para que seja possível visualizar o conteúdo inteiro da célula.*

The screenshot shows a Microsoft Excel spreadsheet titled "Base Funcionários". The table contains 25 rows of data with columns: Nome, Sexo, Área, CPF, and Salário. The "Dados" tab is selected in the ribbon. The status bar at the bottom shows "Gravar Macro" and "Base Funcionários".

	A	B	C	D	E	F	G	H	I	J	K
1	Nome	Sexo	Área	CPF	Salário						
2	Juan Fernandes	Masculino	RH	107.000.473-49	R\$ 16.300,00						
3	Marcela de Freitas	Feminino	Financeiro	140.745.317-54	R\$ 4.650,00						
4	Renan Freire	Masculino	Marketing	101.981.446-34	R\$ 7.200,00						
5	Adriane de Carvalho Gome	Feminino	RH	082.442.274-27	R\$ 4.650,00						
6	Yuri Araujo Senna	Masculino	RH	144.402.781-94	R\$ 12.320,00						
7	Leticia Mesquita	Feminino	Administrativo	136.725.629-63	R\$ 9.450,00						
8	Rafael Machado Cardoso	Masculino	Logística	151.782.237-21	R\$ 16.300,00						
9	Beatriz Leite Júnior	Feminino	Administrativo	134.819.231-78	R\$ 4.650,00						
10	Victor Ruzza de Carvalho	Masculino	Logística	097.867.609-77	R\$ 1.600,00						
11	Gustavo de Vasconcelos	Masculino	Administrativo	137.532.186-15	R\$ 16.300,00						
12	Marianna Pestana	Feminino	Marketing	110.857.529-80	R\$ 3.700,00						
13	Matheus de Souza	Masculino	Marketing	136.163.401-29	R\$ 7.200,00						
14	Carolina Codeceira	Feminino	Marketing	085.347.936-87	R\$ 8.100,00						
15	Gabrielle Andrade da Silva	Feminino	Administrativo	081.419.765-67	R\$ 12.320,00						
16	Daniela Muller Braga	Feminino	Logística	127.900.156-18	R\$ 2.000,00						
17	Isabella Ronfini	Feminino	RH	133.655.149-67	R\$ 12.320,00						
18	Andressa Rotsztejn	Feminino	Marketing	066.779.932-62	R\$ 8.100,00						
19	Elvis de Freitas Derossi	Masculino	RH	104.030.731-00	R\$ 12.320,00						
20	Pedro Costa	Masculino	Logística	127.511.539-73	R\$ 16.300,00						
21	Bruna dos Santos Gomes	Feminino	Administrativo	073.033.309-74	R\$ 4.650,00						
22	Thaís Ribeiro	Feminino	Financeiro	121.464.801-28	R\$ 16.300,00						
23	Júlio Fraga	Masculino	Compras	133.351.567-51	R\$ 7.200,00						
24											
25											

Ainda no mesmo arquivo, na aba **Base Funcionários**, nossa próxima aplicação é um pouquinho mais avançada. Imagina que queremos, de forma fácil, ordenar os nomes da tabela de funcionários ao lado em ordem alfabética.

Poderíamos simplesmente selecionar a tabela, ir até a guia Dados, procurar pela ferramenta **Classificar** e ordenar pela coluna de Nome. Apesar de ser exatamente isso que vamos fazer, para tornar a ação mais automática, vamos criar um botão bem prático onde a pessoa possa clicar e ordenar os nomes, sem que ela precise ter que procurar (ou até mesmo saber) a opção de ordenar.

The screenshot shows a Microsoft Excel interface with the 'Desenvolvedor' tab selected in the ribbon. A red arrow points to the 'Gravar Macro' button in the 'Suplementos' group. A 'Gravar macro' dialog box is open, prompting for macro name ('OrdemAlfabetica'), keyboard shortcut ('Ctrl+'), and save location ('Esta pasta de trabalho'). The background shows a table of employee data.

	A	B	C	D	E	F	G	H	I	J	K
1	Nome	Sexo	Área	CPF	Salário						
2	Juan Fernandes	Masculino	RH	107.000.473-49	R\$ 16.300,00						
3	Marcela de Freitas	Feminino	Financeiro	140.745.317-54	R\$ 4.650,00						
4	Renan Freire	Masculino	Marketing	101.981.446-34	R\$ 7.200,00						
5	Adriane de Carvalho Gomes	Feminino	RH	082.442.274-27	R\$ 4.650,00						
6	Yuri Araujo Senna	Masculino	RH	144.402.781-94	R\$ 12.320,00						
7	Leticia Mesquita	Feminino	Administrativo	136.725.629-63	R\$ 9.450,00						
8	Rafael Machado Cardoso	Masculino	Logística	151.782.237-21	R\$ 16.300,00						
9	Beatriz Leite Júnior	Feminino	Administrativo	134.819.231-78	R\$ 4.650,00						
10	Victor Ruzza de Carvalho	Masculino	Logística	097.867.609-77	R\$ 1.600,00						
11	Gustavo de Vasconcelos	Masculino	Administrativo	137.532.186-15	R\$ 16.300,00						
12	Marianna Pestana	Feminino	Marketing	110.857.529-80	R\$ 3.700,00						
13	Matheus de Souza	Masculino	Marketing	136.163.401-29	R\$ 7.200,00						
14	Carolina Codeceira	Feminino	Marketing	085.347.936-87	R\$ 8.100,00						
15	Gabrielle Andrade da Silva	Feminino	Administrativo	081.419.765-67	R\$ 12.320,00						
16	Daniela Muller Braga	Feminino	Logística	127.900.156-18	R\$ 2.000,00						
17	Isabella Ronfini	Feminino	RH	133.655.149-67	R\$ 12.320,00						
18	Andressa Rotsztejn	Feminino	Marketing	066.779.932-62	R\$ 8.100,00						
19	Elvis de Freitas Derossi	Masculino	RH	104.030.731-00	R\$ 12.320,00						
20	Pedro Costa	Masculino	Logística	127.511.539-73	R\$ 16.300,00						
21	Bruna dos Santos Gomes	Feminino	Administrativo	073.033.309-74	R\$ 4.650,00						
22	Thaís Ribeiro	Feminino	Financeiro	121.464.801-28	R\$ 16.300,00						
23	Júlio Fraga	Masculino	Compras	133.351.567-51	R\$ 7.200,00						
24											
25											

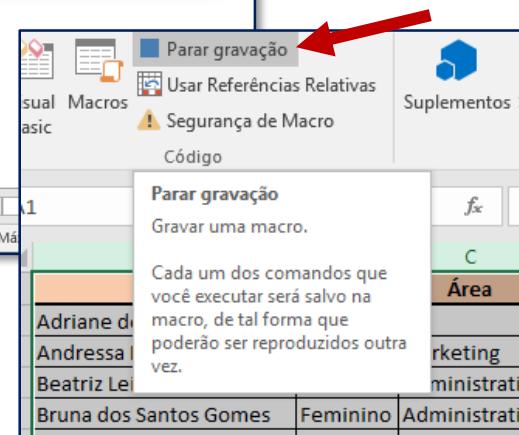
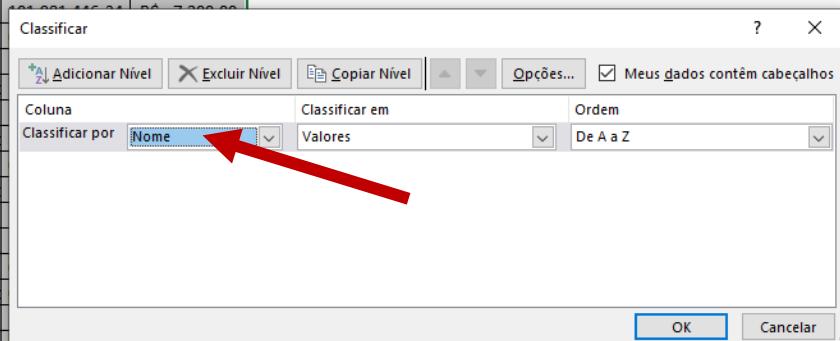
Gravar Macro      Base Funcionários      Exemplo Dinâmica      +

Para gravar essa nova macro, realizamos o mesmo procedimento: [Guia Desenvolvedor](#) → [Gravar Macro](#).

Pode chamar a macro de *OrdemAlfabetica* e clicar em Ok.

Dessa vez não vamos usar atalhos pois vamos criar um botão para executar a macro.

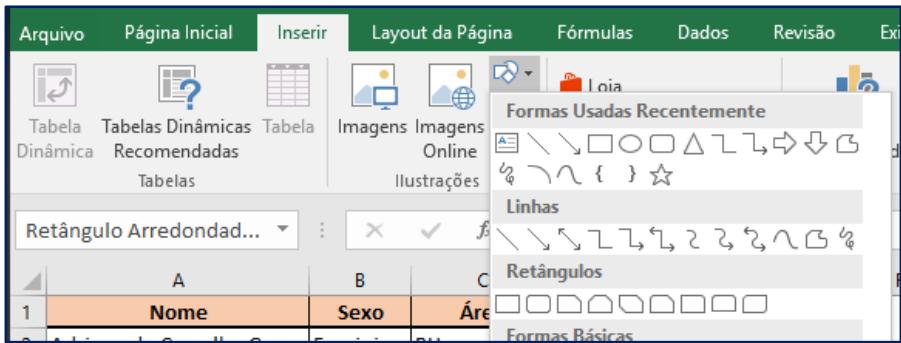
A screenshot of Microsoft Excel showing the ribbon with the 'Dados' (Data) tab selected. A red box highlights the 'Classificar' (Sort) button in the 'Classificar e Filtrar' (Sort & Filter) group. Below the ribbon, a table of employee data is displayed, with rows 1 through 23 visible. The columns include Nome (Name), Sexo (Gender), Área (Area), CPF (CPF), and Salário (Salary).



Iniciada a gravação, siga o seguinte passo a passo:

- 1 - Selecionar as colunas A até E.
- 2 - Clicar na guia Dados
- 3 - Clicar na opção Classificar.
- 4 - Na janela que abrir, configure para **Classificar por Nome**.

Feito isso, clique em **Parar gravação**.



Feito isso, vamos criar um botão seguindo o mesmo passo a passo já visto anteriormente (página 15).

A	B	C	D	E
Nome	Sexo	Área	CPF	Salário
Adriane de Carvalho Gomes	Feminino	RH	082.442.274-27	R\$ 4.650,00
Andressa Rotsztein	Feminino	Marketing	066.779.932-62	R\$ 8.100,00
Beatriz Leite Júnior	Feminino	Administrativo	134.819.231-78	R\$ 4.650,00
Bruna dos Santos Gomes	Feminino	Administrativo	073.033.309-74	R\$ 4.650,00
Carolina Codeceira	Feminino	Marketing	085.347.936-87	R\$ 8.100,00
Daniela Muller Braga	Feminino	Logística	127.900.156-18	R\$ 2.000,00
Elvis de Freitas Derossi	Masculino	RH	104.030.731-00	R\$ 12.320,00
Gabrielle Andrade da Silva	Feminino	Administrativo	081.419.765-67	R\$ 12.320,00
Gustavo de Vasconcelos	Masculino	Administrativo	137.532.186-15	R\$ 16.300,00
Isabella Ronfini	Feminino	RH	133.655.149-67	R\$ 12.320,00
Juan Fernandes	Masculino	RH	107.000.473-49	R\$ 16.300,00
Júlio Fraga	Masculino	Compras	133.351.567-51	R\$ 7.200,00
Leticia Mesquita	Feminino	Administrativo	136.725.629-63	R\$ 9.450,00
Marcela de Freitas	Feminino	Financeiro	140.745.317-54	R\$ 4.650,00
Marianna Pestana	Feminino	Marketing	110.857.529-80	R\$ 3.700,00
Matheus de Souza	Masculino	Marketing	136.163.401-29	R\$ 7.200,00
Pedro Costa	Masculino	Logística	127.511.539-73	R\$ 16.300,00
Rafael Machado Cardoso	Masculino	Logística	151.782.237-21	R\$ 16.300,00
Renan Freire	Masculino	Marketing	101.981.446-34	R\$ 7.200,00
Thais Ribeiro	Feminino	Financeiro	121.464.801-28	R\$ 16.300,00
Victor Ruzza de Carvalho	Masculino	Logística	097.867.609-77	R\$ 1.600,00
Yuri Araujo Senna	Masculino	RH	144.402.781-94	R\$ 12.320,00

A screenshot of Microsoft Excel showing a table of employee data in rows 1 to 8. The columns are labeled A through J, and the rows are labeled 1 through 8. The first row contains column headers: Nome, Sexo, Área, CPF, and Salário. The data includes names like Adriane de Carvalho Gomes, Andressa Rotsztejn, Beatriz Leite Júnior, etc., with their respective genders, areas, CPF numbers, and salaries. To the right of the table, there is a blue rounded rectangle button with the text "Ordenar Nomes". The "Fonte" (Font) section of the ribbon's "Formatar" tab is highlighted with a red box.

Antes de atribuir a macro, vale uma pequena pausa para mostrar como deixar a forma inserida com mais cara de botão.

**Primeiro**, selecione a forma e comece a digitar o texto do botão: Ordenar Nomes. Você pode formatar tanto o tamanho quanto o alinhamento do texto na guia Página Inicial, na parte marcada na figura.

**Segundo**, com a forma ainda selecionada, você pode ir na guia **Formatar** → Efeitos da Forma → Predefinição. Aqui temos alguns modelos de botão que podemos utilizar. O mais usado é o Predefinição 2.

**Por fim**, o botão ficará assim:

A screenshot of the "Formatar" tab in Excel. The "Efeitos da Forma" (Format Shape) section is open, showing various styles for shapes. The "Predefinição" (Predefined Style) dropdown menu is selected, displaying a grid of style options. The "Predefinição 2" style is highlighted. On the left, a list of other styles is visible, including "D", "CPF", "Sombra", "Reflexo", "Brilho", "Bordas Suaves", "Bisel", and "Rotação 3D".



The screenshot shows a Microsoft Excel interface with a data table in the background. A context menu is open over a black rounded rectangle button labeled "Ordenar Nomes". A red arrow points to the "Atribuir Macro" option in the menu, which has opened a dialog box titled "Atribuir macro". The "Nome da macro:" dropdown contains three entries: "OrdemAlfabetica", "MinhaMacro", and "OrdemAlfabetica" (which is highlighted with a blue selection bar). The "OK" button at the bottom left of the dialog is also highlighted with a red arrow.

	A	B	C	D	E
1	Nome	Sexo	Área	CPF	Salário
2	Adriane de Carvalho Gome	Feminino	RH	082.442.274-27	R\$ 4.650,00
3	Andressa Rotsztejn	Feminino	Marketing	066.779.932-62	R\$ 8.100,00
4	Beatriz Leite Júnior	Feminino	Administrativo	134.819.231-78	R\$ 4.650,00
5	Bruna dos Santos Gomes	Feminino	Administrativo	073.033.309-74	R\$ 4.650,00
6	Carolina Codeceira	Feminino	Marketing	085.347.936-87	R\$ 8.100,00
7	Daniela Muller Braga	Feminino	Logística	127.900.156-18	R\$ 2.000,00
8	Elvis de Freitas Derossi	Masculino	RH	104.030.731-00	R\$ 12.320,00
9	Gabrielle Andrade da Silva	Feminino	Administrativo	081.419.765-67	R\$ 12.320,00
10	Gustavo de Vasconcelos	Masculino	Administrativo	137.532.186-15	R\$ 16.300,00
11	Isabella Ronfini	Feminino	RH	133.655.149-67	R\$ 12.320,00
12	Juan Fernandes	Masculino	RH	107.000.473-49	R\$ 16.300,00
13	Júlio Fraga	Masculino	Compras	133.351.567-51	R\$ 7.200,00
14	Leticia Mesquita	Feminino	Administrativo	136.725.629-63	R\$ 9.450,00
15	Marcela de Freitas	Feminino	Financeiro	140.745.317-54	R\$ 4.650,00
16	Marianna Pestana	Feminino	Marketing	110.857.529-80	R\$ 3.700,00
17	Matheus de Souza	Masculino	Marketing	136.163.401-29	R\$ 7.200,00
18	Pedro Costa	Masculino	Logística	127.511.539-73	R\$ 16.300,00
19	Rafael Machado Cardoso	Masculino	Logística	151.782.237-21	R\$ 16.300,00
20	Renan Freire	Masculino	Marketing	101.981.446-34	R\$ 7.200,00
21	Thaís Ribeiro	Feminino	Financeiro	121.464.801-28	R\$ 16.300,00
22	Victor Ruzza de Carvalho	Masculino	Logística	097.867.609-77	R\$ 1.600,00
23	Yuri Araujo Senna	Masculino	RH	144.402.781-94	R\$ 12.320,00
24					
25					

Gravar Macro    Base Funcionários    Exemplo Dinâmica    +

Pronto

Feito isso, fazemos o procedimento de atribuir a macro ao botão.

Para isso, basta clicar na forma com o botão direito, ir na opção Atribuir Macro e depois selecionar a macro criada, no caso, *OrdemAlfabetica*.

# Módulo 1 – Introdução – Gravação de Macro - Exemplo de Aplicação (Parte 1)

30

The screenshot shows a Microsoft Excel spreadsheet titled "Base Funcionários". The table contains columns for Nome, Sexo, Área, CPF, and Salário. A context menu is open at row 9, specifically at the cell containing "Gabrielle Andrade da Silva". The menu items visible are Recortar, Copiar, Opções de Colagem:, Colar Especial..., Inserir (which is highlighted with a red box), Excluir, Limpar conteúdo, Formatar células..., Altura da Linha..., Ocultar, and Re-exibir. A black button labeled "Ordenar Nomes" is overlaid on the menu. The status bar at the bottom shows "Gravar Macro" and "Base Funcionários".

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Nome	Sexo	Área	CPF	Salário												
2	Adriane de Carvalho Gome	Feminino	RH	082.442.274-27	R\$ 4.650,00												
3	Andressa Rotsztejn	Feminino	Marketing	066.779.932-62	R\$ 8.100,00												
4	Beatriz Leite Júnior	Feminino	Administrativ	134.819.231-78	R\$ 4.650,00												
5	Bruna dos Santos Gomes	Feminino	Administrativ	073.033.309-74	R\$ 4.650,00												
6	Calibri	11	A	% 000	R\$ 8.100,00												
7	N	I	A	Marketing	085.347.936-87	R\$ 8.100,00											
8				Logística	127.900.156-18	R\$ 2.000,00											
9	Gabrielle Andrade da Silv	Feminino	Administrativ	081.419.765-67	R\$ 12.320,00												
10				Masculino	Administrativ	137.532.186-15	R\$ 16.300,00										
11				Feminino	RH	133.655.149-67	R\$ 12.320,00										
12				Masculino	RH	107.000.473-49	R\$ 16.300,00										
13				Masculino	Compras	133.351.567-51	R\$ 7.200,00										
14				Feminino	Administrativ	136.725.629-63	R\$ 9.450,00										
15				Feminino	Financeiro	140.745.317-54	R\$ 4.650,00										
16				Feminino	Marketing	110.857.529-80	R\$ 3.700,00										
17				Masculino	Marketing	136.163.401-29	R\$ 7.200,00										
18				Masculino	Logística	127.511.539-73	R\$ 16.300,00										
19				Masculino	Logística	151.782.237-21	R\$ 16.300,00										
20				Masculino	Marketing	101.981.446-34	R\$ 7.200,00										
21				Feminino	Financeiro	121.464.801-28	R\$ 16.300,00										
22				Masculino	Logística	097.867.609-77	R\$ 1.600,00										
23				Masculino	RH	144.402.781-94	R\$ 12.320,00										
24																	
25																	

Para ver que a macro está funcionando corretamente, vamos adicionar um novo nome no meio da tabela.

Para isso, clique com o botão direito em cima do número 9 que identifica a linha e depois selecione a opção Inserir.

A	B	C	D	E	F	G	H	I	J
1	Nome	Sexo	Área	CPF	Salário				
2	Adriane de Carvalho Gome	Feminino	RH	082.442.274-27	R\$ 4.650,00				
3	Andressa Rotsztejn	Feminino	Marketing	066.779.932-62	R\$ 8.100,00				
4	Beatriz Leite Júnior	Feminino	Administrativo	134.819.231-78	R\$ 4.650,00				
5	Bruna dos Santos Gomes	Feminino	Administrativo	073.033.309-74	R\$ 4.650,00				
6	Carolina Codeceira	Feminino	Marketing	085.347.936-87	R\$ 8.100,00				
7	Daniela Muller Braga	Feminino	Logística	127.900.156-18	R\$ 2.000,00				
8	Elvis de Freitas Derossi	Masculino	RH	104.030.731-00	R\$ 12.320,00				
9	Alon Pinheiro	Masculino	RH	140.542.154-85	R\$ 5.000,00				
10	Gabrielle Andrade da Silva	Feminino	Administrativo	081.419.765-67	R\$ 12.320,00				
11	Gustavo de Vasconcelos	Masculino	Administrativo	137.532.186-15	R\$ 16.300,00				
12	Isabella Ronfini	Feminino	RH	133.655.149-67	R\$ 12.320,00				
13	Juan Fernandes	Masculino	RH	107.000.473-49	R\$ 16.300,00				
14	Júlio Fraga	Masculino	Compras	133.351.567-51	R\$ 7.200,00				
15	Letícia Mesquita	Feminino	Administrativo	136.725.629-63	R\$ 9.450,00				
16	Marcela de Freitas	Feminino	Financeiro	140.745.317-54	R\$ 4.650,00				
17	Marianna Pestana	Feminino	Marketing	110.857.529-80	R\$ 3.700,00				
18	Matheus de Souza	Masculino	Marketing	136.163.401-29	R\$ 7.200,00				
19	Pedro Costa	Masculino	Logística	127.511.539-73	R\$ 16.300,00				
20	Rafael Machado Cardoso	Masculino	Logística	151.782.237-21	R\$ 16.300,00				
21	Renan Freire	Masculino	Marketing	101.981.446-34	R\$ 7.200,00				
22	Thais Ribeiro	Feminino	Financeiro	121.464.801-28	R\$ 16.300,00				
23	Victor Ruzza de Carvalho	Masculino	Logística	097.867.609-77	R\$ 1.600,00				
24	Yuri Araujo Senna	Masculino	RH	144.402.781-94	R\$ 12.320,00				

Ordenar Nomes

Você pode inserir qualquer informação. No exemplo, inseri o meu nome.

Depois, é só clicar no botão Ordenar Nomes que a tabela será ordenada automaticamente.

A	B	C	D	E	F	G	H	I	J
1	Nome	Sexo	Área	CPF	Salário				
2	Adriane de Carvalho Gome	Feminino	RH	082.442.274-27	R\$ 4.650,00				
3	Alon Pinheiro	Masculino	RH	140.542.154-85	R\$ 5.000,00				
4	Andressa Rotsztejn	Feminino	Marketing	066.779.932-62	R\$ 8.100,00				
5	Beatriz Leite Júnior	Feminino	Administrativo	134.819.231-78	R\$ 4.650,00				
6	Bruna dos Santos Gomes	Feminino	Administrativo	073.033.309-74	R\$ 4.650,00				
7	Carolina Codeceira	Feminino	Marketing	085.347.936-87	R\$ 8.100,00				
8	Daniela Muller Braga	Feminino	Logística	127.900.156-18	R\$ 2.000,00				
9	Elvis de Freitas Derossi	Masculino	RH	104.030.731-00	R\$ 12.320,00				
10	Gabrielle Andrade da Silva	Feminino	Administrativo	081.419.765-67	R\$ 12.320,00				
11	Gustavo de Vasconcelos	Masculino	Administrativo	137.532.186-15	R\$ 16.300,00				
12	Isabella Ronfini	Feminino	RH	133.655.149-67	R\$ 12.320,00				

Ordenar Nomes

Agora, a nossa planilha está muito mais intuitiva para quem for usar. Ninguém precisará saber Excel para organizar a tabela na ordem correta, basta clicar no botão criado que isso será feito automaticamente, de maneira muito mais fácil.

The screenshot shows a Microsoft Excel interface with a large table of products on the left and a smaller dynamic summary table on the right.

**Large Product Table (Left):**

	A	B	C	D	E	F	G	H	I	J	K	L
1	Descrição	Marca	Faturado	Perda	Fábrica	Origem						
2	Produto 1	Marca A	R\$ 20.822,00	R\$ 1.500,00	Fábrica SP	Importado						
3	Produto 2	Marca B	R\$ 75.789,00	R\$ 16.848,00	Fábrica RJ	Nacional						
4	Produto 3	Marca A	R\$ 20.173,00	R\$ 3.557,00	Fábrica SP	Importado						
5	Produto 4	Marca C	R\$ 29.707,00	R\$ 3.300,00	Fábrica RJ	Nacional						
6	Produto 5	Marca C	R\$ 32.568,00	R\$ 17.271,00	Fábrica RJ	Nacional						
7	Produto 6	Marca A	R\$ 32.488,00	R\$ 17.691,00	Fábrica SP	Nacional						
8	Produto 7	Marca A	R\$ 22.528,00	R\$ 181,00	Fábrica SP	Importado						
9	Produto 8	Marca C	R\$ 71.753,00	R\$ 18.067,00	Fábrica RJ	Nacional						
10	Produto 9	Marca C	R\$ 54.213,00	R\$ 45.986,00	Fábrica SP	Nacional						
11	Produto 10	Marca A	R\$ 55.171,00	R\$ 46.029,00	Fábrica RJ	Nacional						
12	Produto 11	Marca C	R\$ 54.213,00	R\$ 47.060,00	Fábrica SP	Nacional						
13	Produto 12	Marca B	R\$ 66.952,00	R\$ 31.412,00	Fábrica SP	Nacional						
14	Produto 13	Marca A	R\$ 65.536,00	R\$ 25.875,00	Fábrica SP	Nacional						
15	Produto 14	Marca C	R\$ 55.473,00	R\$ 32.267,00	Fábrica SP	Nacional						
16	Produto 15	Marca C	R\$ 53.158,00	R\$ 43.025,00	Fábrica SP	Importado						
17	Produto 16	Marca A	R\$ 73.278,00	R\$ 53.866,00	Fábrica RJ	Nacional						
18	Produto 17	Marca C	R\$ 20.412,00	R\$ 9.120,00	Fábrica RJ	Nacional						
19	Produto 18	Marca C	R\$ 88.358,00	R\$ 166,00	Fábrica SP	Nacional						
20	Produto 19	Marca A	R\$ 70.381,00	R\$ 40.897,00	Fábrica RJ	Importado						
21	Produto 20	Marca B	R\$ 62.129,00	R\$ 46.566,00	Fábrica RJ	Importado						
22	Produto 21	Marca B	R\$ 76.050,00	R\$ 57.852,00	Fábrica RJ	Nacional						

**Dynamic Summary Table (Right):**

Marcas	Soma de Faturado
Marca A	R\$ 1.680.867,00
Marca B	R\$ 2.017.270,00
Marca C	R\$ 8.126.204,00
<b>Total Geral</b>	<b>R\$ 11.824.341,00</b>

Um outro exemplo bastante comum que a gente costuma usar na gravação de macro é **atualizar uma tabela dinâmica**.

A tabela dinâmica é basicamente um resumo das informações de uma tabela maior. No arquivo ao lado, temos uma tabela bem grande com várias informações de produtos, e imediatamente ao lado temos uma tabela dinâmica, como o resumo do total de faturado para cada uma das 3 marcas.

# Módulo 1 – Introdução – Gravação de Macro - Exemplo de Aplicação (Parte 2)

33

The screenshot shows a Microsoft Excel interface with a dynamic table and its summary.

**Table Data:**

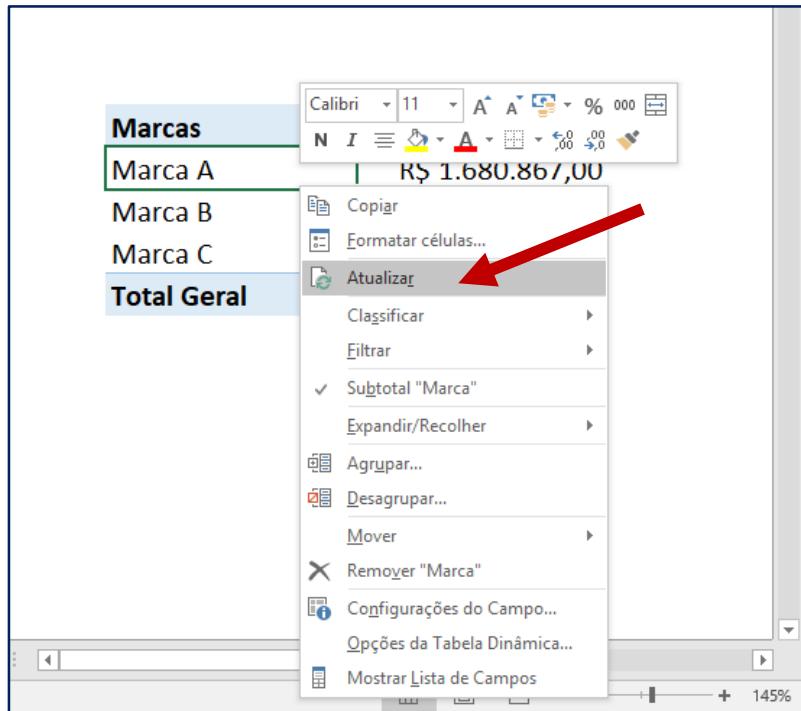
	A	B	C	D	E	F	G	H	I	J
1	Descrição	Marca	Faturado	Perda	Fábrica	Origem				
2	Produto 1	Marca A	R\$ 1.000.000,00	R\$ 1.500,00	Fábrica SP	Importado				
3	Produto 2	Marca B	R\$ 75.789,00	R\$ 16.848,00	Fábrica RJ	Nacional				
4	Produto 3	Marca A	R\$ 20.173,00	R\$ 3.557,00	Fábrica SP	Importado				
5	Produto 4	Marca C	R\$ 29.707,00	R\$ 3.300,00	Fábrica RJ	Nacional				
6	Produto 5	Marca C	R\$ 32.568,00	R\$ 17.271,00	Fábrica RJ	Nacional				
7	Produto 6	Marca A	R\$ 32.488,00	R\$ 17.691,00	Fábrica SP	Nacional				
8	Produto 7	Marca A	R\$ 22.528,00	R\$ 181,00	Fábrica SP	Importado				
9	Produto 8	Marca C	R\$ 71.753,00	R\$ 18.067,00	Fábrica RJ	Nacional				
10	Produto 9	Marca C	R\$ 54.213,00	R\$ 45.986,00	Fábrica SP	Nacional				
11	Produto 10	Marca A	R\$ 55.171,00	R\$ 46.029,00	Fábrica RJ	Nacional				
12	Produto 11	Marca C	R\$ 54.213,00	R\$ 47.060,00	Fábrica SP	Nacional				
13	Produto 12	Marca B	R\$ 66.952,00	R\$ 31.412,00	Fábrica SP	Nacional				
14	Produto 13	Marca A	R\$ 65.536,00	R\$ 25.875,00	Fábrica SP	Nacional				
15	Produto 14	Marca C	R\$ 55.473,00	R\$ 32.267,00	Fábrica SP	Nacional				
16	Produto 15	Marca C	R\$ 53.158,00	R\$ 43.025,00	Fábrica SP	Importado				
17	Produto 16	Marca A	R\$ 73.278,00	R\$ 53.866,00	Fábrica RJ	Nacional				
18	Produto 17	Marca C	R\$ 20.412,00	R\$ 9.120,00	Fábrica RJ	Nacional				

**Summary (Soma de Faturado):**

Marcas	Soma de Faturado
Marca A	R\$ 1.680.867,00
Marca B	R\$ 2.017.270,00
Marca C	R\$ 8.126.204,00
<b>Total Geral</b>	<b>R\$ 11.824.341,00</b>

Vamos ao problema: se alterarmos qualquer valor dentro da tabela (por exemplo, o valor da célula C2 para R\$ 1.000.000,00), a **tabela dinâmica não se atualiza automaticamente**.

Isso é um problema! Pois alguém que tenha acesso a essa tabela dinâmica pode acabar tomando alguma decisão baseado em um valor errado caso ela não saiba que a tabela dinâmica não se atualiza automaticamente ou caso ela esqueça de atualizá-la.



Marcas	Soma de Faturado
Marca A	R\$ 2.660.045,00
Marca B	R\$ 2.017.270,00
Marca C	R\$ 8.126.204,00
<b>Total Geral</b>	<b>R\$ 12.803.519,00</b>

A princípio, para atualizar a tabela dinâmica, basta clicar nela com o botão direito e em seguida selecionar a opção **Atualizar**.

Para evitar qualquer problema com essa questão de atualização da tabela dinâmica, o ideal seria gravar uma macro que atualiza a tabela e em seguida atribuir a um botão. Isso tornaria a utilização da tabela dinâmica muito mais fácil e segura.

# Módulo 1 – Introdução – Gravação de Macro - Exemplo de Aplicação (Parte 2)

35

The screenshot shows an Excel spreadsheet with the following data:

	A	B	C	D	E	F	G	H	I	J
1	Descrição	Marca	Faturado	Perda	Fábrica	Origem				
2	Produto 1	Marca A	R\$ 1.000.000,00	R\$ 1.500,00						
3	Produto 2	Marca B	R\$ 75.789,00	R\$ 16.848,00						
4	Produto 3	Marca A	R\$ 20.173,00	R\$ 3.557,00						
5	Produto 4	Marca C	R\$ 29.707,00	R\$ 3.300,00						
6	Produto 5	Marca C	R\$ 32.568,00	R\$ 17.271,00						
7	Produto 6	Marca A	R\$ 32.488,00	R\$ 17.691,00						
8	Produto 7	Marca A	R\$ 22.528,00	R\$ 181,00						
9	Produto 8	Marca C	R\$ 71.753,00	R\$ 18.067,00						
10	Produto 9	Marca C	R\$ 54.213,00	R\$ 45.986,00						
11	Produto 10	Marca A	R\$ 55.171,00	R\$ 46.029,00						
12	Produto 11	Marca C	R\$ 54.213,00	R\$ 47.060,00						
13	Produto 12	Marca B	R\$ 66.952,00	R\$ 31.412,00	Fábrica SP	Nacional				
14	Produto 13	Marca A	R\$ 65.536,00	R\$ 25.875,00	Fábrica SP	Nacional				
15	Produto 14	Marca C	R\$ 55.473,00	R\$ 32.267,00	Fábrica SP	Nacional				
16	Produto 15	Marca C	R\$ 53.158,00	R\$ 43.025,00	Fábrica SP	Importado				
17	Produto 16	Marca A	R\$ 73.278,00	R\$ 53.866,00	Fábrica RJ	Nacional				
18	Produto 17	Marca C	R\$ 20.412,00	R\$ 9.120,00	Fábrica RJ	Nacional				

Below the table, a summary table shows the total sales by brand:

Marcas	Soma de Faturado
Marca A	R\$ 2.660.045,00
Marca B	R\$ 2.017.270,00
Marca C	R\$ 8.126.204,00
Total Geral	R\$ 12.803.519,00

Assim, o primeiro passo é gravar uma macro que atualiza a nossa tabela dinâmica.

Para isso, basta ir na guia Desenvolvedor -> Gravar Macro. O nome da nossa macro pode ser *AtualizaTabela*. Em seguida, é só clicar em 'Ok'.

# Módulo 1 – Introdução – Gravação de Macro - Exemplo de Aplicação (Parte 2)

36

The screenshot shows a Microsoft Excel spreadsheet with a dynamic table named 'Marca' (Product) containing 18 rows of data. The columns are labeled: Descrição, Marca, Faturado, Perda, Fábrica, and Origem. A context menu is open over the table, with the 'Atualizar' (Update) option highlighted by a red arrow. The menu also includes options like Copy, Format Cells, and Subtotal.

1	A	B	C	D	E	F	G	H	I	J
1	Descrição	Marca	Faturado	Perda	Fábrica	Origem				
2	Produto 1	Marca A	R\$ 1.000.000,00	R\$ 1.500,00	Fábrica SP	Importado				
3	Produto 2	Marca B	R\$ 75.789,00	R\$ 16.848,00	Fábrica RJ	Nacional				
4	Produto 3	Marca A	R\$ 20.173,00	R\$ 3.557,00	Fábrica SP	Importado				
5	Produto 4	Marca C	R\$ 29.707,00	R\$ 3.300,00	Fábrica RJ	Nacional				
6	Produto 5	Marca C	R\$ 32.568,00	R\$ 17.271,00	Fábrica RJ	Nacional				
7	Produto 6	Marca A	R\$ 32.488,00	R\$ 17.691,00	Fábrica SP	Nacional				
8	Produto 7	Marca A	R\$ 22.528,00	R\$ 181,00	Fábrica SP	Importado				
9	Produto 8	Marca C	R\$ 71.753,00	R\$ 18.067,00	Fábrica RJ	Nacional				
10	Produto 9	Marca C	R\$ 54.213,00	R\$ 45.986,00	Fábrica SP	Nacional				
11	Produto 10	Marca A	R\$ 55.171,00	R\$ 46.029,00	Fábrica RJ	Nacional				
12	Produto 11	Marca C	R\$ 54.213,00	R\$ 47.060,00	Fábrica SP	Nacional				
13	Produto 12	Marca B	R\$ 66.952,00	R\$ 31.412,00	Fábrica SP	Nacional				
14	Produto 13	Marca A	R\$ 65.536,00	R\$ 25.875,00	Fábrica SP	Nacional				
15	Produto 14	Marca C	R\$ 55.473,00	R\$ 32.267,00	Fábrica SP	Nacional				
16	Produto 15	Marca C	R\$ 53.158,00	R\$ 43.025,00	Fábrica SP	Importado				
17	Produto 16	Marca A	R\$ 73.278,00	R\$ 53.866,00	Fábrica RJ	Nacional				
18	Produto 17	Marca C	R\$ 20.412,00	R\$ 9.120,00	Fábrica RJ	Nacional				

Iniciada a gravação, vamos simplesmente clicar com o botão direito na tabela dinâmica e depois em **Atualizar**.

The screenshot shows a Microsoft Excel window with the ribbon menu open. The 'Desenvolvedor' tab is selected, indicating that a macro is currently being recorded. A context menu is open over a cell in the first row of a table, with the option 'Parar gravação' highlighted.

**Table Data:**

	Descrição	aturado	Perda	Fábrica	Origem
1	Produto 1	\$ 1.000.000,00	R\$ 1.500,00	Fábrica SP	Importado
2	Produto 2	\$ 75.789,00	R\$ 16.848,00	Fábrica RJ	Nacional
3	Produto 3	Marca A	R\$ 20.173,00	R\$ 3.557,00	Fábrica SP
4	Produto 4	Marca C	R\$ 29.707,00	R\$ 3.300,00	Fábrica RJ
5	Produto 5	Marca C	R\$ 32.568,00	R\$ 17.271,00	Fábrica RJ
6	Produto 6	Marca A	R\$ 32.488,00	R\$ 17.691,00	Fábrica SP
7	Produto 7	Marca A	R\$ 22.528,00	R\$ 181,00	Fábrica SP
8	Produto 8	Marca C	R\$ 71.753,00	R\$ 18.067,00	Fábrica RJ
9	Produto 9	Marca C	R\$ 54.213,00	R\$ 45.986,00	Fábrica SP
10	Produto 10	Marca A	R\$ 55.171,00	R\$ 46.029,00	Fábrica RJ
11	Produto 11	Marca C	R\$ 54.213,00	R\$ 47.060,00	Fábrica SP
12	Produto 12	Marca B	R\$ 66.952,00	R\$ 31.412,00	Fábrica SP
13	Produto 13	Marca A	R\$ 65.536,00	R\$ 25.875,00	Fábrica SP
14	Produto 14	Marca C	R\$ 55.473,00	R\$ 32.267,00	Fábrica SP
15	Produto 15	Marca C	R\$ 53.158,00	R\$ 43.025,00	Fábrica SP
16	Produto 16	Marca A	R\$ 73.278,00	R\$ 53.866,00	Fábrica RJ
17	Produto 17	Marca C	R\$ 20.412,00	R\$ 9.120,00	Fábrica RJ
18	Produto 18				Nacional

**Summary Table:**

Marcas	Soma de Faturado
Marca A	R\$ 2.660.045,00
Marca B	R\$ 2.017.270,00
Marca C	R\$ 8.126.204,00
Total Geral	R\$ 12.803.519,00

Por fim, é só clicar em **Parar gravação**.

# Módulo 1 – Introdução – Gravação de Macro - Exemplo de Aplicação (Parte 2)

38

The screenshot shows a Microsoft Excel spreadsheet with a dynamic table. The table has columns labeled 'Descrição', 'Marca', 'Faturado', 'Perda', 'Fábrica', and 'Origem'. The 'Faturado' column contains values like R\$ 1.000.000,00 and R\$ 75.789,00. The 'Perda' column contains values like R\$ 1.500,00 and R\$ 16.848,00. The 'Fábrica' column contains values like Fábrica SP and Fábrica RJ. The 'Origem' column contains values like Importado and Nacional. A blue rounded rectangle shape is overlaid on the table, containing the text 'Atualizar Tabela'. A context menu is open for this shape, with 'Atribuir macro...' highlighted. The ribbon menu at the top includes tabs like Arquivo, Página Inicial, Inserir, Layout da Página, Fórmulas, Dados, Revisão, Exibir, Desenvolvedor, Formatar, Entrar, and Compartilhar.

	A	B	C	D	E	F
1	Descrição	Marca	Faturado	Perda	Fábrica	Origem
2	Produto 1	Marca A	R\$ 1.000.000,00	R\$ 1.500,00	Fábrica SP	Importado
3	Produto 2	Marca B	R\$ 75.789,00	R\$ 16.848,00	Fábrica RJ	Nacional
4	Produto 3	Marca A	R\$ 20.173,00	R\$ 3.557,00	Fábrica SP	Importado
5	Produto 4	Marca C	R\$ 29.707,00	R\$ 3.300,00	Fábrica RJ	Nacional
6	Produto 5	Marca C	R\$ 32.568,00	R\$ 17.271,00	Fábrica RJ	Nacional
7	Produto 6	Marca A	R\$ 32.488,00	R\$ 17.691,00	Fábrica SP	Nacional
8	Produto 7	Marca A	R\$ 22.528,00	R\$ 181,00	Fábrica SP	Importado
9	Produto 8	Marca C	R\$ 71.753,00	R\$ 18.067,00	Fábrica RJ	Nacional
10	Produto 9	Marca C	R\$ 54.213,00	R\$ 45.986,00	Fábrica SP	Nacional
11	Produto 10	Marca A	R\$ 55.171,00	R\$ 46.029,00	Fábrica RJ	Nacional
12	Produto 11	Marca C	R\$ 54.213,00	R\$ 47.060,00	Fábrica SP	Nacional
13	Produto 12	Marca B	R\$ 66.952,00	R\$ 31.412,00	Fábrica SP	Nacional
14	Produto 13	Marca A	R\$ 65.536,00	R\$ 25.875,00	Fábrica SP	Nacional
15	Produto 14	Marca C	R\$ 55.473,00	R\$ 32.267,00	Fábrica SP	Nacional
16	Produto 15	Marca C	R\$ 53.158,00	R\$ 43.025,00	Fábrica SP	Importado
17	Produto 16	Marca A	R\$ 73.278,00	R\$ 53.866,00	Fábrica RJ	Nacional
18	Produto 17	Marca C	R\$ 20.412,00	R\$ 9.120,00	Fábrica RJ	Nacional
19	Produto 18	Marca C	R\$ 88.358,00	R\$ 166,00	Fábrica SP	Nacional
20	Produto 19	Marca A	R\$ 70.381,00	R\$ 40.897,00	Fábrica RJ	Importado

Seguindo o que fizemos até agora, vamos criar um botão para atribuir a nossa macro. É só clicar na guia Inserir → Ilustrações.

Você também pode configurar a forma assim como vimos na página 15.

Para fechar, é só clicar na forma com o botão direito e em seguida em Atribuir macro.

# Módulo 1 – Introdução – Gravação de Macro - Exemplo de Aplicação (Parte 2)

39

A screenshot of Microsoft Excel showing the 'Atribuir macro' (Assign Macro) dialog box. The dialog box is centered over a table. The 'Nome da macro:' dropdown shows 'AtualizarTabela' selected. The 'Macros em:' dropdown shows 'Todas as pastas de trabalho abertas'. The 'Descrição' field contains 'Exemplo Dinâmica'. The 'OK' button is highlighted with a blue border.

	A	B	C	D	E	F	G	H	I	J	K
1	Descrição	Marca	Faturado	Perda	Fábrica	Origem					
2	Produto 1	Marca A	R\$ 1.000.000,00	R\$ 1.500,00	Fábrica SP	Importado					
3	Produto 2	Marca B	R\$ 75.789,00	R\$ 16.848,00							
4	Produto 3	Marca A	R\$ 20.173,00	R\$ 3.557,00							
5	Produto 4	Marca C	R\$ 29.707,00	R\$ 3.300,00							
6	Produto 5	Marca C	R\$ 32.568,00	R\$ 17.271,00							
7	Produto 6	Marca A	R\$ 32.488,00	R\$ 17.691,00							
8	Produto 7	Marca A	R\$ 22.528,00	R\$ 181,00							
9	Produto 8	Marca C	R\$ 71.753,00	R\$ 18.067,00							
10	Produto 9	Marca C	R\$ 54.213,00	R\$ 45.986,00							
11	Produto 10	Marca A	R\$ 55.171,00	R\$ 46.029,00							
12	Produto 11	Marca C	R\$ 54.213,00	R\$ 47.060,00							
13	Produto 12	Marca B	R\$ 66.952,00	R\$ 31.412,00							
14	Produto 13	Marca A	R\$ 65.536,00	R\$ 25.875,00							
15	Produto 14	Marca C	R\$ 55.473,00	R\$ 32.267,00							
16	Produto 15	Marca C	R\$ 53.158,00	R\$ 43.025,00							
17	Produto 16	Marca A	R\$ 73.278,00	R\$ 53.866,00							
18	Produto 17	Marca C	R\$ 20.412,00	R\$ 9.120,00							
19	Produto 18	Marca C	R\$ 88.358,00	R\$ 166,00							
20	Produto 19	Marca A	R\$ 70.381,00	R\$ 40.897,00							

E selecionar a macro *AtualizaTabela*.

# Módulo 1 – Introdução – Gravação de Macro - Exemplo de Aplicação (Parte 2)

40

The screenshot shows a Microsoft Excel interface with a table of product data and a summary box.

**Table Data:**

	A	B	C	D	E	F
1	Descrição	Marca	Faturado	Perda	Fábrica	Origem
2	Produto 1	Marca A	R\$ 1.000.000,00	R\$ 1.500,00	Fábrica SP	Importado
3	Produto 2	Marca B	R\$ 10.000.000,00	R\$ 16.848,00	Fábrica RJ	Nacional
4	Produto 3	Marca A	R\$ 20.173,00	R\$ 3.557,00	Fábrica SP	Importado
5	Produto 4	Marca C	R\$ 29.707,00	R\$ 3.300,00	Fábrica RJ	Nacional
6	Produto 5	Marca C	R\$ 32.568,00	R\$ 17.271,00	Fábrica RJ	Nacional
7	Produto 6	Marca A	R\$ 32.488,00	R\$ 17.691,00	Fábrica SP	Nacional
8	Produto 7	Marca A	R\$ 22.528,00	R\$ 181,00	Fábrica SP	Importado
9	Produto 8	Marca C	R\$ 71.753,00	R\$ 18.067,00	Fábrica RJ	Nacional
10	Produto 9	Marca C	R\$ 54.213,00	R\$ 45.986,00	Fábrica SP	Nacional
11	Produto 10	Marca A	R\$ 55.171,00	R\$ 46.029,00	Fábrica RJ	Nacional
12	Produto 11	Marca C	R\$ 54.213,00	R\$ 47.060,00	Fábrica SP	Nacional
13	Produto 12	Marca B	R\$ 66.952,00	R\$ 31.412,00	Fábrica SP	Nacional
14	Produto 13	Marca A	R\$ 65.536,00	R\$ 25.875,00	Fábrica SP	Nacional
15	Produto 14	Marca C	R\$ 55.473,00	R\$ 32.267,00	Fábrica SP	Nacional
16	Produto 15	Marca C	R\$ 53.158,00	R\$ 43.025,00	Fábrica SP	Importado
17	Produto 16	Marca A	R\$ 73.278,00	R\$ 53.866,00	Fábrica RJ	Nacional
18	Produto 17	Marca C	R\$ 20.412,00	R\$ 9.120,00	Fábrica RJ	Nacional
19	Produto 18	Marca C	R\$ 88.358,00	R\$ 166,00	Fábrica SP	Nacional
20	Produto 19	Marca A	R\$ 70.381,00	R\$ 40.897,00	Fábrica RJ	Importado

**Summary Box (Initial State):**

Marcas	Soma de Faturado
Marca A	R\$ 2.660.045,00
Marca B	R\$ 2.017.270,00
Marca C	R\$ 8.126.204,00
<b>Total Geral</b>	<b>R\$ 12.803.519,00</b>

**Summary Box (Updated State):**

Marcas	Soma de Faturado
Marca A	R\$ 2.660.045,00
<b>Marca B</b>	<b>R\$ 11.941.481,00</b>
Marca C	R\$ 8.126.204,00
<b>Total Geral</b>	<b>R\$ 22.727.730,00</b>

An orange arrow points from the initial summary box to the updated one, indicating the result of the update.

Para ver a macro funcionando, podemos simplesmente alterar outro valor da tabela (na célula C3 para R\$ 10.000.000,00, por exemplo) e depois clicar no botão Atualizar Tabela.

Como pode ser visto, os valores para a marca B foram atualizados.

The screenshot shows a Microsoft Excel interface with a dynamic table and a summary bar chart.

**Table Data:**

	A	B	C	D	E	F
1	Descrição	Marca	Faturado	Perda	Fábrica	Origem
2	Produto 1	Marca A	R\$ 1.000.000,00	R\$ 1.500,00	Fábrica SP	Importado
3	Produto 2	Marca B	R\$ 10.000.000,00	R\$ 16.848,00	Fábrica RJ	Nacional
4	Produto 3	Marca A	R\$ 20.173,00	R\$ 3.557,00	Fábrica SP	Importado
5	Produto 4	Marca D	R\$ 29.707,00	R\$ 3.300,00	Fábrica RJ	Nacional
6	Produto 5	Marca C	R\$ 32.568,00	R\$ 17.271,00	Fábrica RJ	Nacional
7	Produto 6	Marca A	R\$ 32.488,00	R\$ 17.691,00	Fábrica SP	Nacional
8	Produto 7	Marca A	R\$ 22.528,00	R\$ 181,00	Fábrica SP	Importado
9	Produto 8	Marca C	R\$ 71.753,00	R\$ 18.067,00	Fábrica RJ	Nacional
10	Produto 9	Marca C	R\$ 54.213,00	R\$ 45.986,00	Fábrica SP	Nacional
11	Produto 10	Marca A	R\$ 55.171,00	R\$ 46.029,00	Fábrica RJ	Nacional
12	Produto 11	Marca C	R\$ 54.213,00	R\$ 47.060,00	Fábrica SP	Nacional
13	Produto 12	Marca B	R\$ 66.952,00	R\$ 31.412,00	Fábrica SP	Nacional
14	Produto 13	Marca A	R\$ 65.536,00	R\$ 25.875,00	Fábrica SP	Nacional
15	Produto 14	Marca C	R\$ 55.473,00	R\$ 32.267,00	Fábrica SP	Nacional
16	Produto 15	Marca C	R\$ 53.158,00	R\$ 43.025,00	Fábrica SP	Importado
17	Produto 16	Marca A	R\$ 73.278,00	R\$ 53.866,00	Fábrica RJ	Nacional
18	Produto 17	Marca C	R\$ 20.412,00	R\$ 9.120,00	Fábrica RJ	Nacional
19	Produto 18	Marca C	R\$ 88.358,00	R\$ 166,00	Fábrica SP	Nacional
20	Produto 19	Marca A	R\$ 70.381,00	R\$ 40.897,00	Fábrica RJ	Importado

**Summary Bar Chart:**

Atualizar Tabela

Marcas	Soma de Faturado
Marca A	R\$ 2.660.045,00
Marca B	R\$ 11.941.481,00
Marca C	R\$ 8.126.204,00
<b>Total Geral</b>	<b>R\$ 22.727.730,00</b>

**Final State:**

Marcas	Soma de Faturado
Marca A	R\$ 2.660.045,00
<b>Marca B</b>	<b>R\$ 11.941.481,00</b>
Marca C	R\$ 8.096.497,00
Marca D	R\$ 29.707,00
<b>Total Geral</b>	<b>R\$ 22.727.730,00</b>

Agora temos uma macro totalmente automatizada. Se adicionarmos uma marca D na célula B5, por exemplo, ao clicar no botão de Atualizar Tabela essa nova marca será inserida na nossa tabela dinâmica, como podemos ver na imagem ao lado.

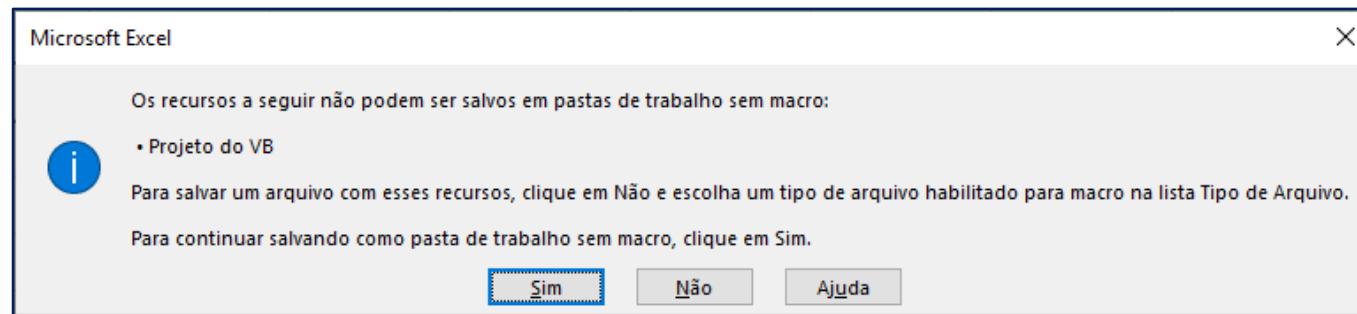


### ATENÇÃO !

Um ponto importante quando estamos trabalhando com macros é a forma como devemos salvar o arquivo. Os arquivos Excel que não contêm macros possuem a extensão `*.xlsx`.

Nome	Data de modificaç...	Tipo	Tamanho
exemplo.xlsx	20/02/2020 22:53	Planilha do Microsoft Excel	8 KB

Porém, quando tentamos salvar um arquivo existente com alguma macro criada recebemos o aviso abaixo.



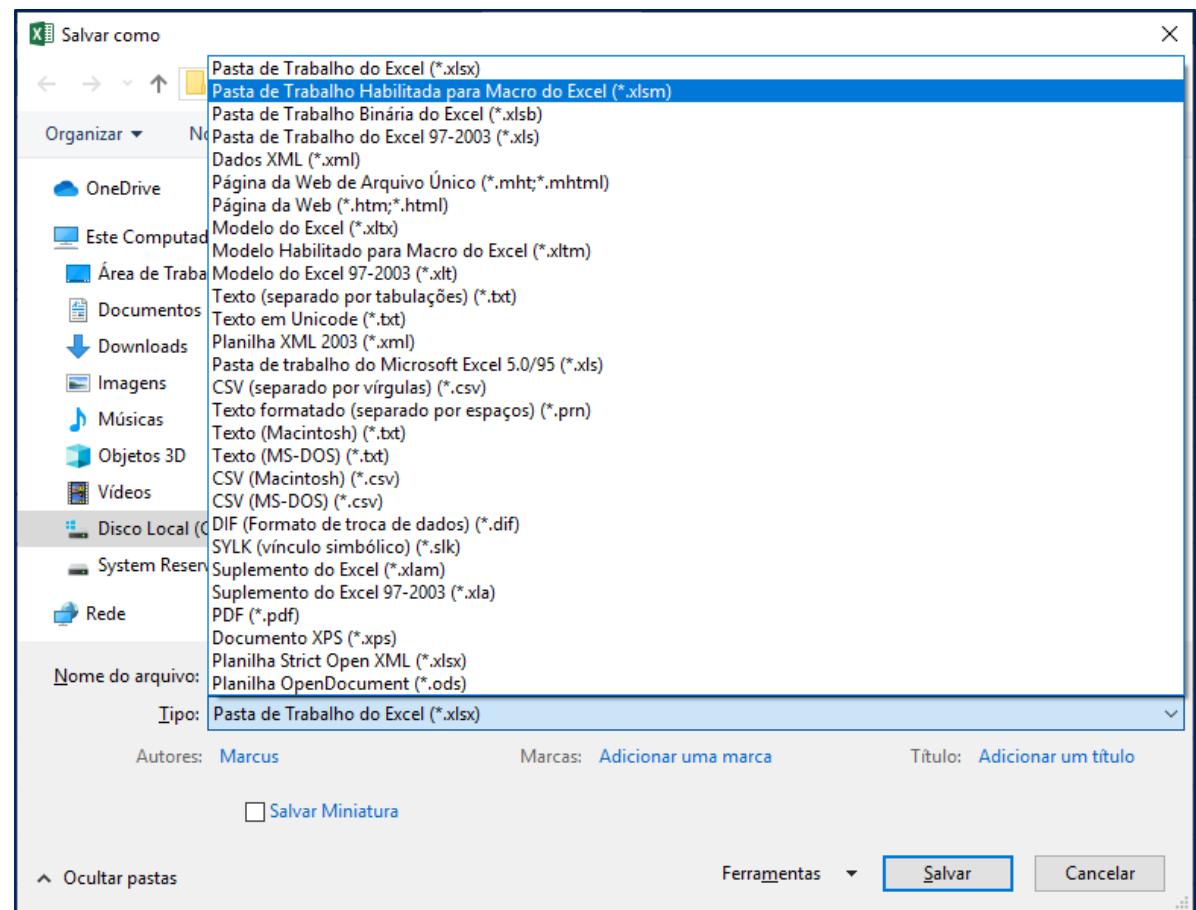
Basicamente essa é a forma de o Excel avisar que o arquivo precisa ser salvo em uma extensão diferente. **Aqui não podemos clicar em Sim pois o arquivo será salvo sem as macros e todo o trabalho será perdido.** O correto é clicar em 'Não' e escolher a extensão correta para esse tipo de arquivo.

A maneira correta de salvar um arquivo com macros é usando a opção **Pasta de Trabalho Habilitada para Macro do Excel**, cuja extensão é a **\*.xlsm**.

Uma boa prática ao se trabalhar com macros é salvar o arquivo sempre antes de rodar uma macro, pois se cometermos algum erro de programação em nosso código corremos o risco de entrar em *loops* (repetições) infinitas, fazendo com que o Excel trave e feche inesperadamente. Veremos mais para frente as causas desses problemas e como se prevenir de qualquer problema durante a execução das macros.

O atalho para salvar uma planilha é o CTRL + B (em português, ou CTRL + S em inglês) ou simplesmente clicar no símbolo de disquete, no canto superior esquerdo da tela. 

Outro atalho é o de **Salvar Como**, que salva uma cópia do seu arquivo atual. O atalho para o *Salvar Como* é o F12 (ou Fn + F12). Os procedimentos para salvamento de arquivo com macros é exatamente o explicado no início do texto.



Módulo 2

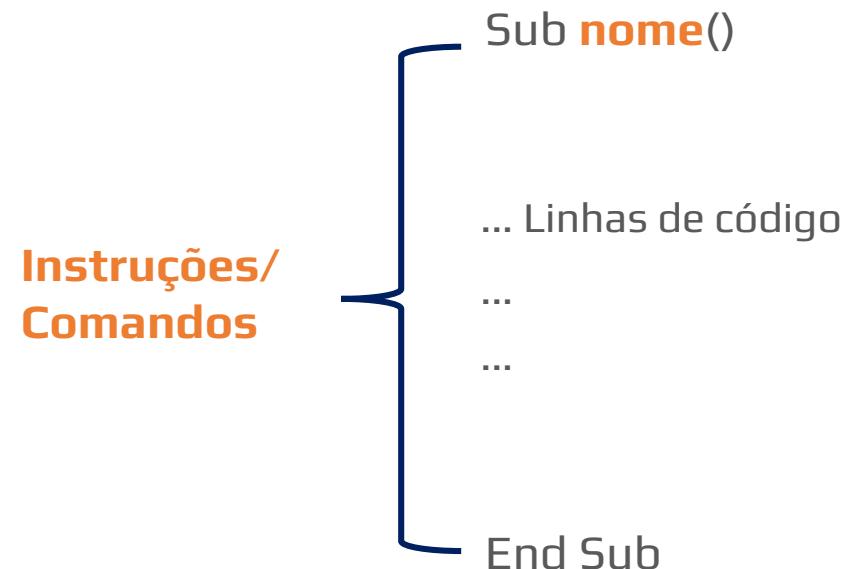
# Estrutura básica e objetos do VBA

Vamos agora começar a entender com calma como é a estrutura básica dos códigos em VBA. Até agora nos preocupamos apenas com exemplos de gravação de macros, que já nos dão os códigos prontos. O que precisamos agora é entender do zero como iniciamos códigos do zero.

Como já vimos anteriormente, todo código em VBA se inicia por uma palavra chave chamada **Sub**, de Sub-rotina, e se encerra com a palavra chave **End Sub**. Uma sub-rotina nada mais é que uma sequência de ações que será executada toda vez que o usuário quiser. Esta sequência de ações (instruções/comandos) é inserida sempre entre essas duas palavras chave.

Em geral, vamos sempre executar comandos do tipo: selecionar células, escrever em células, copiar uma célula, selecionar uma aba, e por ai vai.

Mas quais seriam esses comandos? Vamos entender.



### Instruções/ Comandos

O VBA pode ser considerado uma linguagem de programação orientada a objetos. O que isso significa? Significa que todas as nossas ações serão feitas em cima de objetos do Excel, que em resumo podem ser:

- Células (*Cells / Range*)
- Abas (*Sheets*)
- Arquivos (*Workbooks*)
- Gráficos (*ChartObjects*)

Existem também variações destes objetos:

- *ActiveCell* → Célula Ativa
- *Selection* → Objeto selecionado
- *ActiveSheet* → Aba ativa (*Workbooks*)
- *ActiveWorkbook* → Arquivo ativo

Então, quando quisermos executar qualquer comando sobre uma célula, vamos escrever o código *Cells* ou *Range* (ou *ActiveCell*) para nos referirmos a qual célula queremos executar aquele comando. O box ao lado mostra alguns exemplos do que podemos fazer com os objetos no VBA.

No VBA, sempre vamos ter que especificar qual é o **objeto** que estamos mexendo, seguido de qual **ação** ou **propriedade** vamos querer utilizar para aquele objeto.

- **Ações:**

- *Range("A1").Select* → Seleciona a célula A1
- *Sheets("Base de Dados").Activate* → Ativa a Aba 'Base de Dados'
- *Cells(2, 1).Copy* → Copia a célula A2 (linha 2, coluna 1)
- *Selection.Copy* → Copia as células selecionadas
- *ActiveCell.ClearContents* → Apaga o valor da célula ativa.

- **Propriedades:**

- *Range("A1").Value = 22* → Escreve 22 na célula A1
- *Cells(3, 3).Interior.Color = vbYellow* → Pinta a célula C3 (linha 3, coluna 3) de amarelo
- *Sheets("Base de Dados").Name = "Cadastro Funcionários"* → Renomeia a aba 'Base de Dados' para 'Cadastro Funcionários'
- *Selection.Value = "Hashtag"* → Escreve 'Hashtag' nas células selecionadas
- *ActiveCell.Value = "Alon"* → Escreve 'Alon' na célula ativa

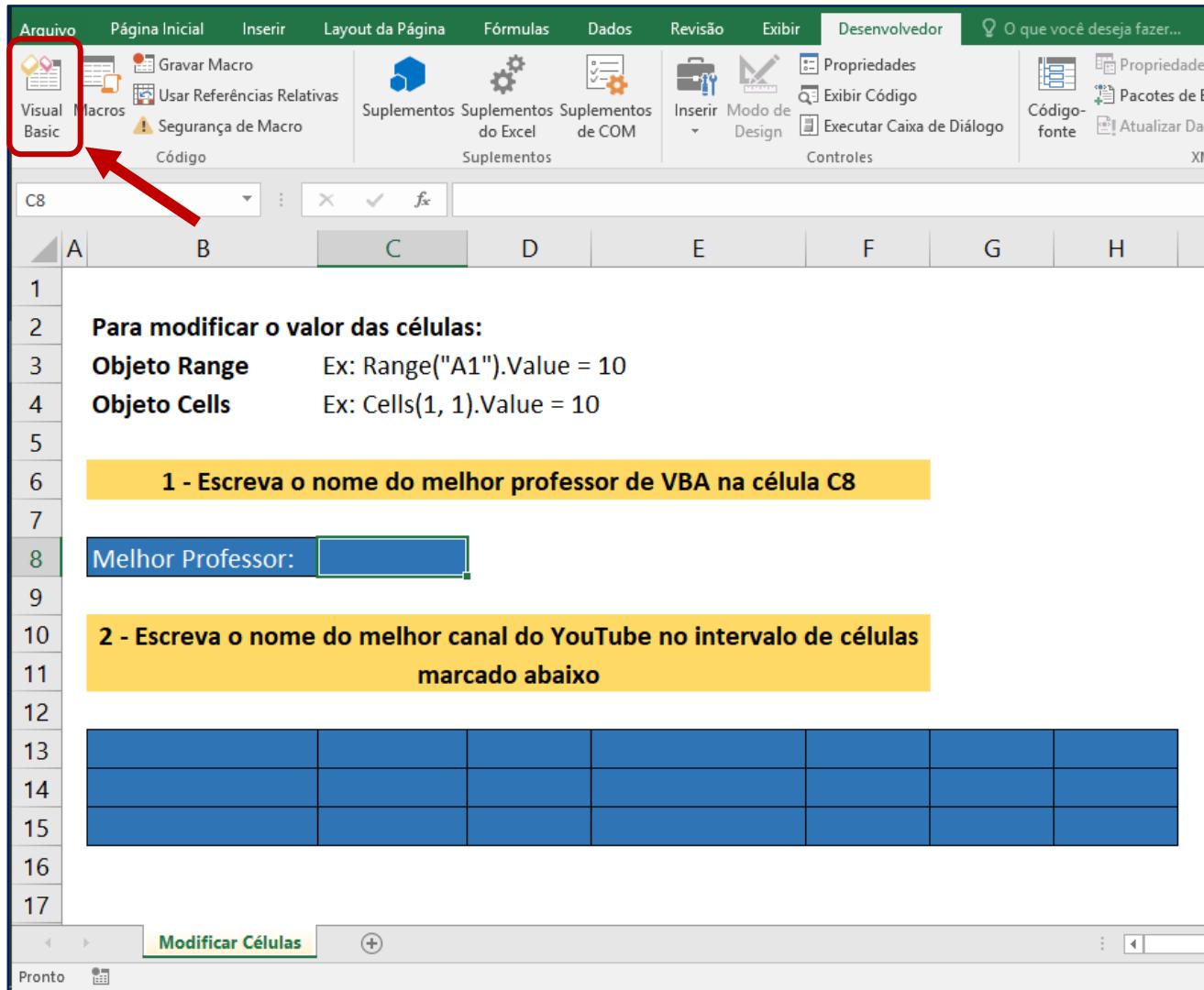
Ao lado temos um exemplo completo de um código simples, chamado **minha\_macro**. O código começa selecionando a célula A1. Em seguida, na célula ativa (célula A1) escrevemos o texto “Alon”. Na linha seguinte temos um comentário, que como já explicamos, sempre será iniciado por uma aspa simples e não influencia em nada o código, sendo ignorado durante a execução da macro.

Em seguida, a aba “Base de Dados” é ativada. Nesta aba ativa é escrito o nome “Diego” na célula C3 (linha 3, coluna 3) e, por fim, o fundo da célula C3 é modificado para amarelo.

Repare que o VBA é uma linguagem em inglês, então todos os comandos e instruções serão escritas em inglês. Mas não se preocupe, será mais fácil do que você imagina.

### Instruções/ Comandos

```
Sub minha_macro()
    Range("A1").Select
    ActiveCell.Value = "Alon"
    'comentários
    Sheets("Base de Dados").Activate
    Cells(3, 3).Value = "Diego"
    Cells(3, 3).Interior.Color = vbYellow
End Sub
```



Araivo Página Inicial Inserir Layout da Página Fórmulas Dados Revisão Exibir Desenvolvedor O que você deseja fazer...

Gravar Macro Usar Referências Relativas Suplementos do Excel Suplementos de COM Inserir Modo de Design Executar Caixa de Diálogo Controles Código-fonte Pacotes de Excel Atualizar Dados XM

Visual Basic Macros Segurança de Macro Suplementos Códigos

C8 A B C D E F G H

Para modificar o valor das células:

Objeto Range Ex: Range("A1").Value = 10  
Objeto Cells Ex: Cells(1, 1).Value = 10

1 - Escreva o nome do melhor professor de VBA na célula C8

Melhor Professor: [C8]

2 - Escreva o nome do melhor canal do YouTube no intervalo de células marcado abaixo

[C13:C18]

Modificar Células

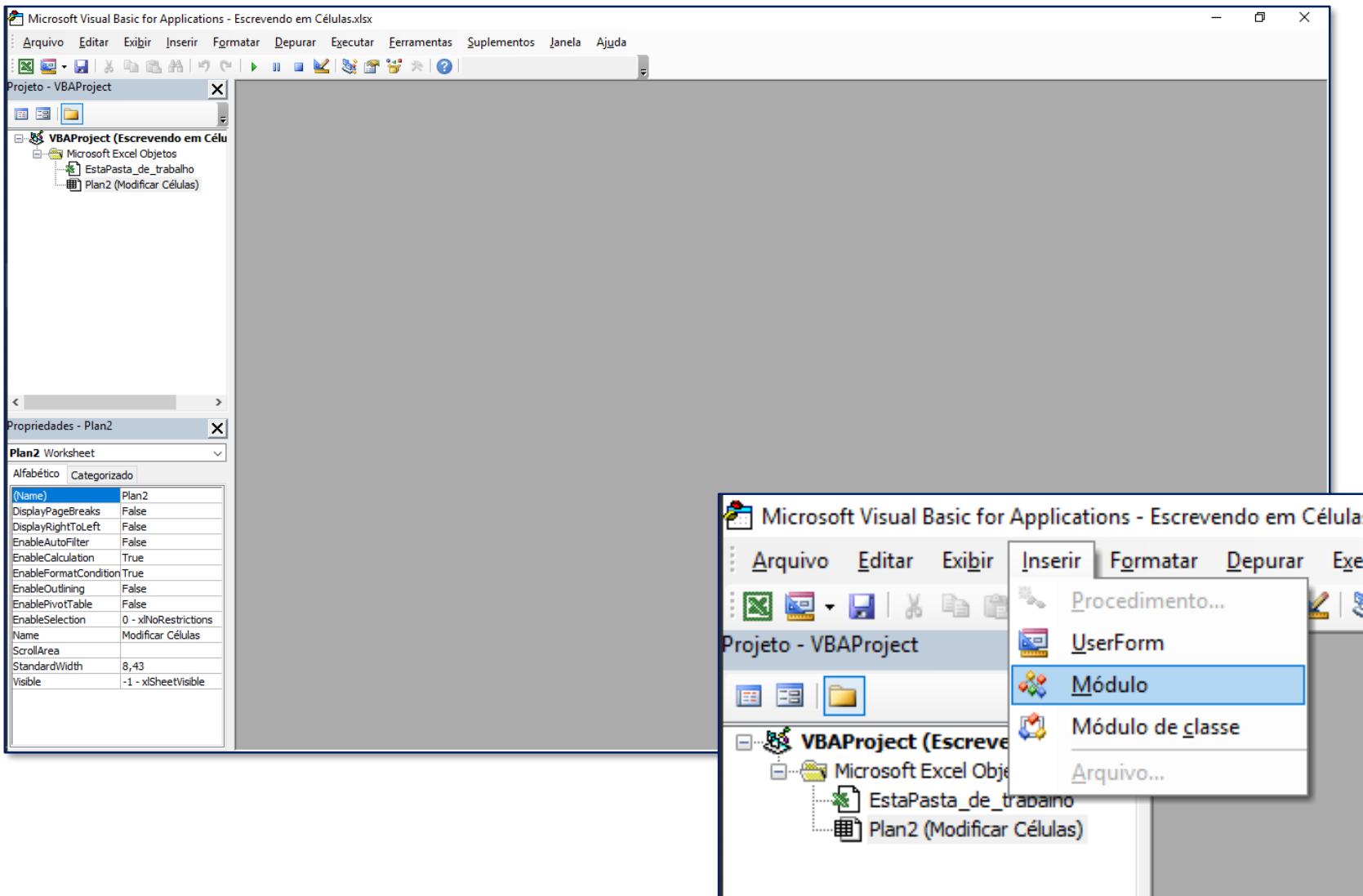
Como vimos anteriormente, existem **duas formas** de escrever em células por meio do VBA: utilizando o comando *Cells* ou utilizando o comando *Range*.

Quando utilizamos o comando *Range*, devemos informar o nome da célula entre aspas dentro do comando, seguindo a identificação de letras (para as colunas) e números (para linhas). Assim, se quisermos escrever o valor 10 na célula A1 usando o *Range*, simplesmente fazemos:

*Range("A1").Value = 10*

O *.Value* significa que queremos modificar a propriedade valor da célula A1, fazendo com que ela receba o valor 10.

Vamos então escrever esse código no VBA. Para isso, podemos abrir o ambiente VBA clicando no botão Visual Basic ou utilizando o atalho ALT + F11.



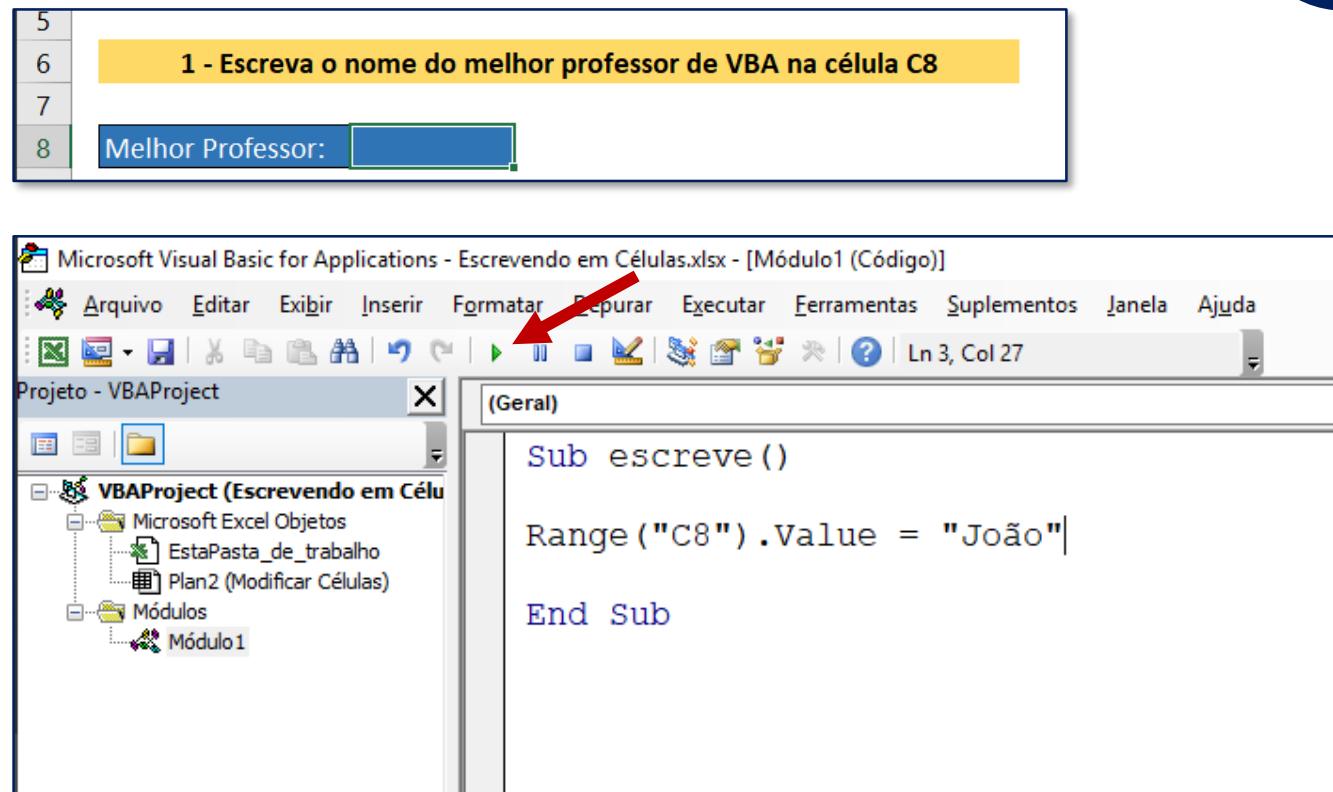
Este é o nosso ambiente do VBA. Nele podemos ver, a princípio, duas janelas do lado esquerdo: uma de Projeto, onde ele mostra todas as abas do nosso arquivo (neste caso, há apenas a aba “Modificar Células”) e abaixo temos uma outra janela chamada Propriedades, onde é possível visualizar algumas propriedades referentes à nossa aba, por exemplo. Mais para frente iremos detalhar com mais calma estas duas janelas.

Por enquanto, o que nos interessa é criar uma nova página onde possamos criar os nossos códigos. Para isso, clicamos na guia **Inserir** → **Módulo**.

Voltando ao nosso primeiro exemplo, queremos escrever na célula C8 o nome do melhor professor de VBA. Para isso, vamos começar usando o comando *Range*.

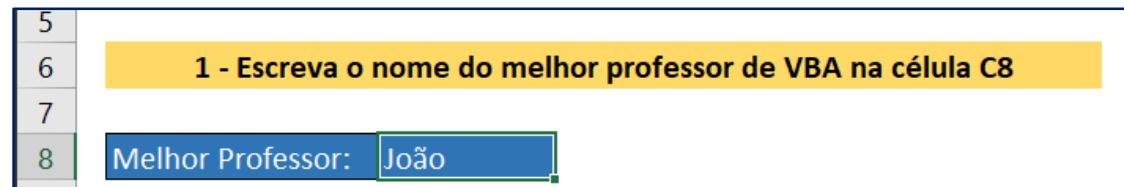
O código está escrito na imagem ao lado. Ao finalizar, para executar a macro, clicamos no botão de play mostrado na imagem, ou simplesmente utilizamos o atalho F5 (ou Fn + F5).

Para voltar para o arquivo Excel e visualizar o resultado, você pode simplesmente minimizar a janela do VBA ou usar o atalho ALT + TAB para mudar de janela.



1 - Escreva o nome do melhor professor de VBA na célula C8

```
Sub escreve()
    Range("C8").Value = "João"
End Sub
```



5	
6	
7	
8	Melhor Professor: João

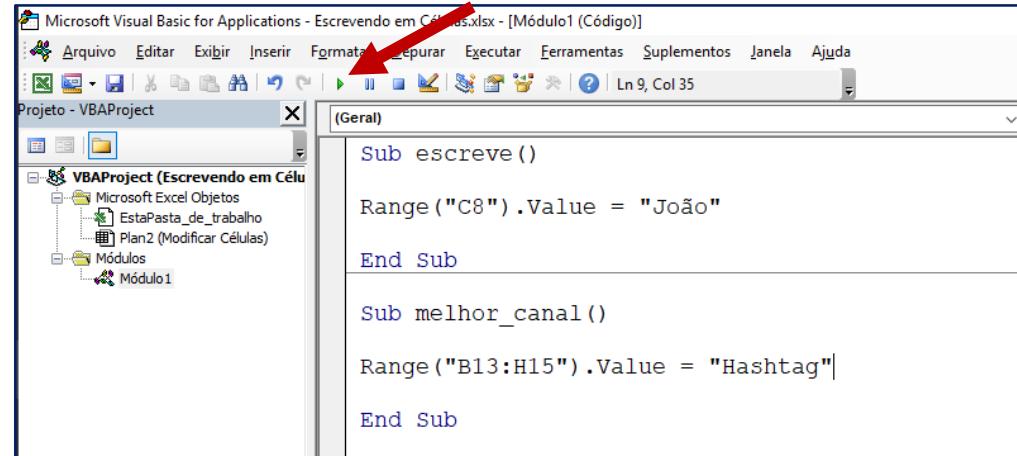
Também podemos escrever em várias células de uma vez. No próximo exercício, queremos escrever o nome do melhor canal do YouTube nas células B13 até H15.

Para isso, usamos o seguinte comando:

*Range("B13:H15").Value = "Hashtag"*

Em seguida, é só clicar novamente no play para executar a macro ou então usar o atalho F5 (ou Fn + F5)

A	B	C	D	E	F	G	H
9							
10							
11	2 - Escreva o nome do melhor canal do YouTube no intervalo de células marcado abaixo						
12							
13							
14							
15							
16							



```
Sub escreve()
    Range ("C8").Value = "João"
End Sub

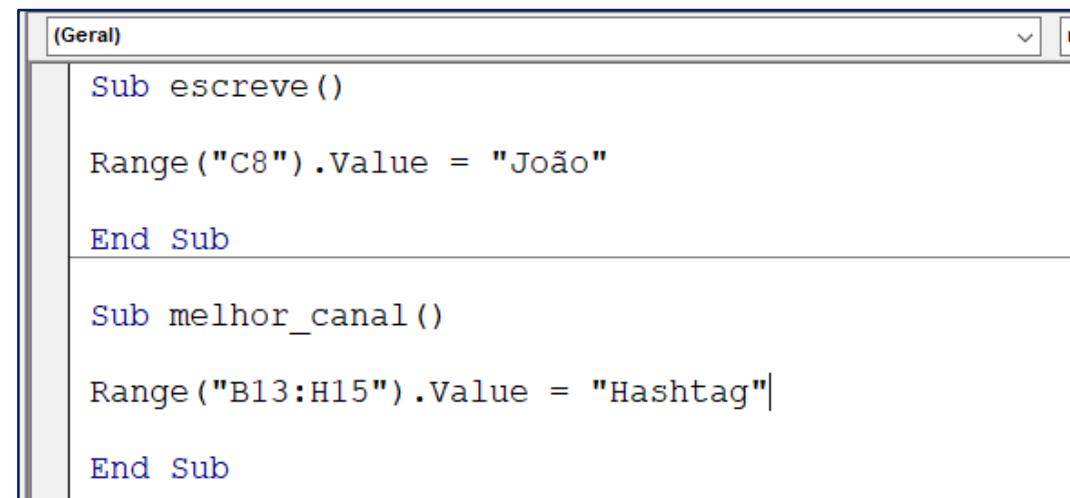
Sub melhor_canal()
    Range ("B13:H15").Value = "Hashtag"
End Sub
```

A	B	C	D	E	F	G	H
9							
10							
11	2 - Escreva o nome do melhor canal do YouTube no intervalo de células marcado abaixo						
12							
13	Hashtag	Hashtag	Hashtag	Hashtag	Hashtag	Hashtag	Hashtag
14	Hashtag	Hashtag	Hashtag	Hashtag	Hashtag	Hashtag	Hashtag
15	Hashtag	Hashtag	Hashtag	Hashtag	Hashtag	Hashtag	Hashtag
16							

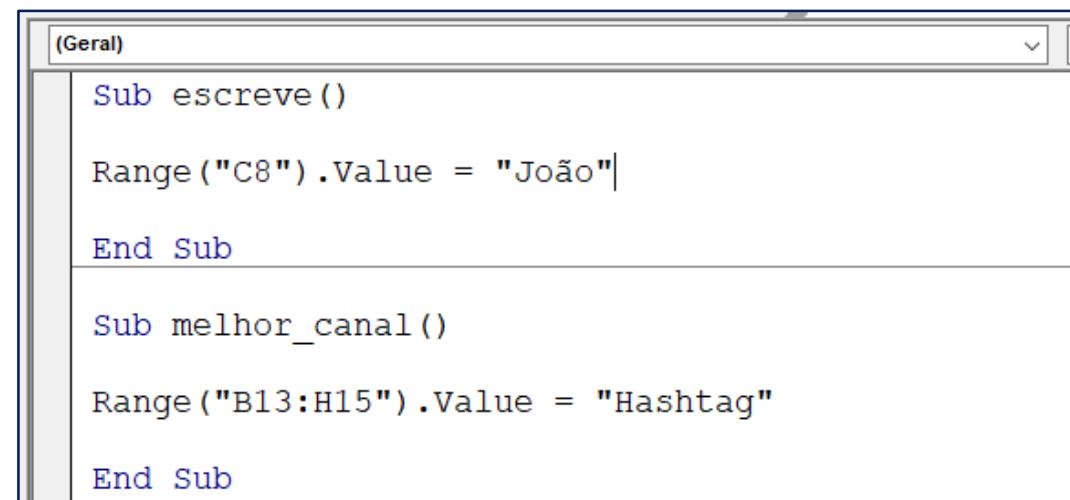
Um detalhe muito importante agora é que temos mais de uma macro no mesmo código. Isso não significa que todas elas serão executadas de uma vez. Apenas uma será executada. Qual?

A macro executada vai depender de onde o cursor do mouse está posicionado. Na primeira imagem, o cursor do mouse está posicionado na macro “melhor\_canal”, então se executarmos o código, apenas esta macro será executada.

No caso da imagem de baixo, é o contrário, como o cursor está posicionado dentro do código “escreve”, então ao apertar o botão de play, esta será a macro executada.



```
(Geral)  
Sub escreve()  
    Range ("C8") .Value = "João"  
End Sub  
  
Sub melhor_canal()  
    Range ("B13:H15") .Value = "Hashtag"  
End Sub
```



```
(Geral)  
Sub escreve()  
    Range ("C8") .Value = "João"  
End Sub  
  
Sub melhor_canal()  
    Range ("B13:H15") .Value = "Hashtag"  
End Sub
```

Além do *Range* também podemos usar o comando *Cells*. Esta é apenas outra forma de escrever em uma célula. A estrutura do *Cells* é a seguinte:

*Cells(núm\_linha, núm\_coluna)*

Assim, diferente do *Range*, devemos especificar o número tanto da linha quanto da coluna para especificar a célula. Como queremos refazer o primeiro exercício de escrever na célula C8 só que agora usando o *Cells*, devemos fazer:

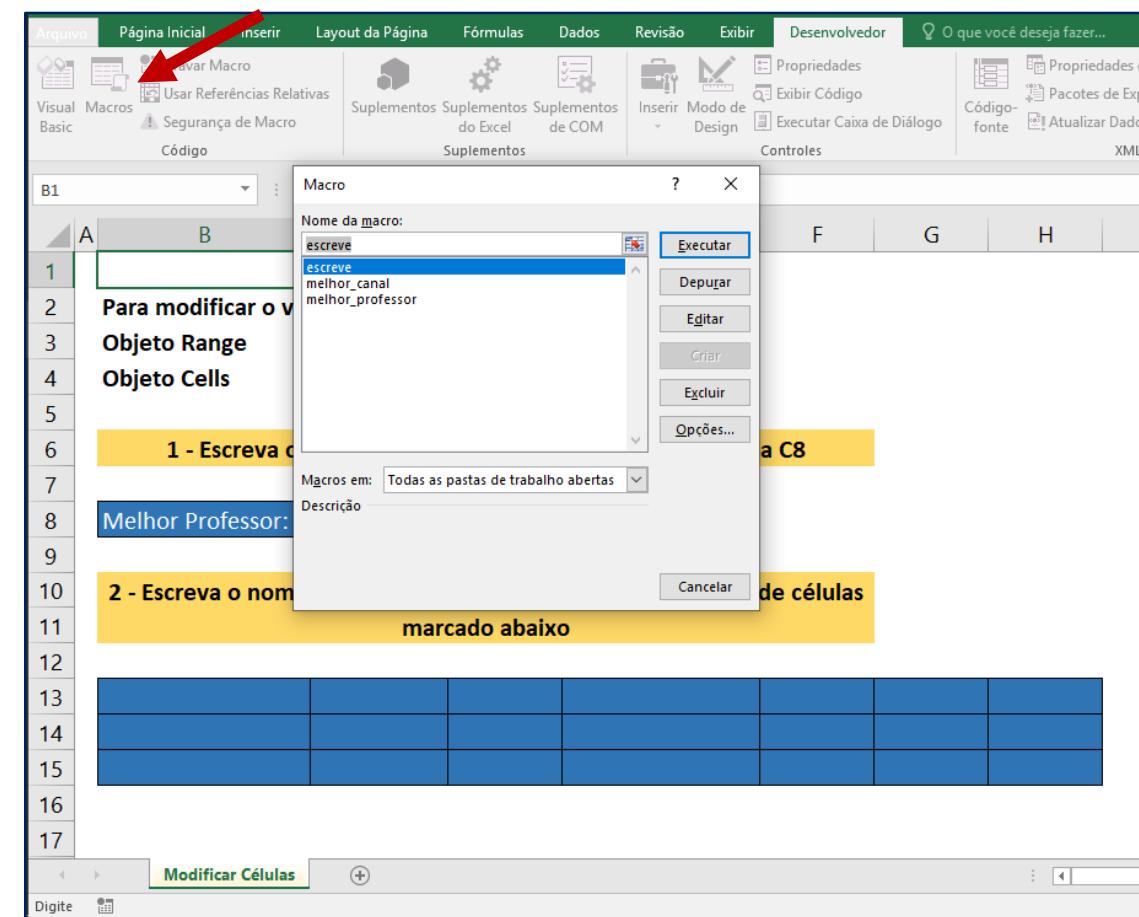
*Cells(8, 3).Value = "João Paulo"*

Mais para frente veremos o porquê de existirem duas formas de fazer uma mesma coisa, e entender quando é melhor usar um e quando é melhor usar o outro.

```
Sub melhor_professor()
    Cells(8, 3).Value = "João Paulo"
End Sub
```

Basicamente temos **3 maneiras** de executar os nossos códigos do VBA. Na aula anterior vimos a primeira, que é a opção de clicar no botão de play (ou F5) dentro do próprio ambiente VBA. Essa seria a mais trabalhosa, dado que precisaríamos entrar no ambiente VBA toda vez que quiséssemos executar aquela macro. Porém, é a maneira como a gente utiliza quando queremos executar a macro enquanto estamos criando os códigos.

A segunda maneira, mostrada na imagem ao lado, é ir na guia **Desenvolvedor** → **Macros**. Uma janela será aberta com todas as macros criadas e assim você pode escolher qual macro deseja executar. Basta selecionar a macro na lista e clicar no botão **Executar**.



A terceira maneira já vimos na página 15, que é criar botões para executar as macros. Esta é a forma mais prática de se fazer. Então vamos fazer isso, criar um botão para executar a macro do primeiro exercício (escreve ou melhor\_professor) e outro botão para executar a macro do segundo exercício (melhor\_canal).

O passo a passo será exatamente o mesmo da página 15.

Se você quisesse utilizar um único botão para executar as duas macros, você poderia simplesmente juntar os dois códigos da macro “escreve” e da macro “melhor\_canal” em um só, e associar essa nova macro “completa” a um único botão:

```
Sub completa()
    Range("C8").Value = "João"
    Range("B13:H15").Value = "Hashtag"
End Sub
```

A	B	C	D	E	F	G	H	I
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13	Hashtag							
14	Hashtag							
15	Hashtag							
16								



A ideia é que você sempre irá colocar em uma mesma Sub todos os comandos que você quiser executar de uma única vez. Não é necessário dividir cada comando em uma Sub diferente quando você pode juntar todos em uma Sub só!

A	B	C	D
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			

1 - Crie uma rotina que registre uma nova venda na tabela abaixo. Deverá ser registrado o produto Guaraná Natural, com preço unitário de R\$1,50 e com 10.000 unidades vendidas.

Em seguida, adapte seu código para registrar o mesmo produto em uma nova linha.

Produtos	Preço Unitário	Quantidade Vendida
Coca Cola	R\$5,00	1500
Joelho de Queijo e Presunto	R\$6,50	200
Coxinha	R\$4,00	500

O próximo passo agora é registrar uma nova venda na linha 11 da nossa tabela à esquerda, como pedido no exercício. Para criarmos o nosso código, primeiro utilizamos o atalho ALT + F11 (ou Guia **Desenvolvedor** → **Visual Basic**) e abrimos a janela do Visual Basic. Não esqueça de criar um novo Módulo para começarmos a criar os nossos códigos.



Para começar a macro, escrevemos Sub e em seguida o nome da macro: registro. Para escrever nas células B11, C11 e D11 vamos utilizar o comando *Cells*. Repare que apenas textos devem ser colocados entre aspas. Números não precisam. Além disso, como o VBA é uma linguagem em inglês, o separador decimal deve ser o ponto, e não a vírgula.

Por fim, é só clicar no botão ➤ para executar a macro (ou usar o atalho F5).

```
Sub registro()
    Cells(11, 2).Value = "Guaraná Natural"
    Cells(11, 3).Value = 1.5
    Cells(11, 4).Value = 10000
End Sub
```

A	B	C	D
1			
2			
3			
4			
5			
6			
7	<b>Produtos</b>	<b>Preço Unitário</b>	<b>Quantidade Vendida</b>
8	Coca Cola	R\$5,00	1500
9	Joelho de Queijo e Presunto	R\$6,50	200
10	Coxinha	R\$4,00	500
11	Guaraná Natural	R\$ 1,50	10000
12			

Após rodar a macro, o nosso registro foi realizado. A seguir vamos ver como podemos otimizar este código para facilitar a criação de novos registros de vendas.

A	B	C	D
1			
2			
3			
4			
5			
6			
7	<b>Produtos</b>	<b>Preço Unitário</b>	<b>Quantidade Vendida</b>
8	Coca Cola	R\$5,00	1500
9	Joelho de Queijo e Presunto	R\$6,50	200
10	Coxinha	R\$4,00	500
11	Guaraná Natural	R\$ 1,50	10000
12			

1 - Crie uma rotina que registre uma nova venda na tabela abaixo. Deverá ser registrado o produto Guaraná Natural, com preço unitário de R\$1,50 e com 10.000 unidades vendidas.

Em seguida, adapte seu código para registrar o mesmo produto em uma nova linha.

O próximo exercício seria adaptar o código para torná-lo mais automático, para que, sempre que formos adicionar uma nova venda, tenhamos o processo mais otimizado possível.

A princípio, se quisermos adicionar uma nova venda na linha 12, bastaria mudar o número da linha do comando *Cells* em cada linha de código, substituindo o 11 pelo 12.

```
Sub registro()
    Cells(11, 2).Value = "Guaraná Natural"
    Cells(11, 3).Value = 1.5
    Cells(11, 4).Value = 10000
End Sub
```



```
Sub registro()
    Cells(12, 2).Value = "Guaraná Natural"
    Cells(12, 3).Value = 1.5
    Cells(12, 4).Value = 10000
End Sub
```

Porém, esse processo é de certa forma cansativo. Por mais simples que pareça, ter que mudar o número da linha de 11 para 12 em todo o código requer um tempo desnecessário. Com apenas 3 linhas de código, não é tão doloroso, mas na prática, nossos códigos vão ter dezenas e até centenas de linhas de código, e ter que mudar várias linhas de código para cadastrar uma nova venda seria um processo bem exaustivo.

Muito melhor que isso seria a gente criar uma variável chamada **linha**, por exemplo, que recebe um valor fixo (no caso, o valor da linha da nova venda), e usar essa variável dentro do comando *Cells*. Assim, sempre que quisermos mudar o número da linha para adicionar uma nova venda, a gente simplesmente mudaria uma única vez o valor da variável **linha**, o que ia atualizar automaticamente o nosso código sem que precisássemos alterar 3 vezes o número da linha da nova venda. Com a variável, fazemos essa alteração uma única vez.

As imagens ao lado mostram qual é a ideia de se criar essa nova variável.

```
Sub registro()
    linha = 11
    Cells(11, 2).Value = "Guaraná Natural"
    Cells(11, 3).Value = 1.5
    Cells(11, 4).Value = 10000
End Sub
```



```
Sub registro()
    linha = 11
    Cells(linha, 2).Value = "Guaraná Natural"
    Cells(linha, 3).Value = 1.5
    Cells(linha, 4).Value = 10000
End Sub
```



Uma variável é um nome que criamos no nosso código para armazenar algum valor específico que poderá ser utilizado diversas vezes ao longo da macro.

Repare que agora fica muito mais fácil de cadastrar uma nova venda.

Como atribuímos um valor à variável **linha**, toda vez que escrevermos esta variável no código o VBA vai entender que deve na verdade pegar o valor atribuído à variável **linha**, que na primeira imagem é igual a 12, e na de baixo é igual a 13.

Uma variável pode receber qualquer nome. Em geral, escolhemos nomes intuitivos para as nossas variáveis para que fique fácil de entender que valor está sendo armazenado. Evite escolher nomes como 'x', 'a', 'valor', pois são nomes genéricos e podem causar confusão futuramente.

```
Sub registro()
    linha = 12
    
    Cells(linha, 2).Value = "Guaraná Natural"
    Cells(linha, 3).Value = 1.5
    Cells(linha, 4).Value = 10000
    
End Sub
```

```
Sub registro()
    linha = 13
    
    Cells(linha, 2).Value = "Guaraná Natural"
    Cells(linha, 3).Value = 1.5
    Cells(linha, 4).Value = 10000
    
End Sub
```



A	B	C	D
1			
2			
3			
4			
5			
6			
7			
8	Produtos	Preço Unitário	Quantidade Vendida
9	Coca Cola	R\$5,00	1500
10	Joelho de Queijo e Presunto	R\$6,50	200
11	Coxinha	R\$4,00	500
12	Guaraná Natural	R\$ 1,50	10000
13	Guaraná Natural	R\$ 1,50	10000
14			

A	B	C	D
1			
2			
3			
4			
5			
6			
7			
8	Produtos	Preço Unitário	Quantidade Vendida
9	Coca Cola	R\$5,00	1500
10	Joelho de Queijo e Presunto	R\$6,50	200
11	Coxinha	R\$4,00	500
12	Guaraná Natural	R\$ 1,50	10000
13	Guaraná Natural	R\$ 1,50	10000
14			

Já introduzimos o conceito de variável na aula anterior. Agora, vamos ver qual é a maneira correta de declarar variáveis nos nossos códigos.

- **Declarando uma Variável no VBA**

**Dim nome\_da\_variável As tipo\_da\_variável**

“Dimensionar”

Exemplos:

- linha
- coluna
- produto
- numero1
- numero2
- numero\_1
- pizza
- joao\_paulo



Sempre começamos declarando uma variável com a palavra **Dim**, seguido do nome da variável. Este nome não pode ter espaços, nem caracteres especiais.

Uma variável sempre vai ter um tipo: pode ser um número inteiro, decimal, texto, data, célula, etc. É uma excelente prática de programação declarar o tipo correto das variáveis.

Exemplos:

- *Integer* → Número inteiro (pequeno, até 32.000)
- *Long* → Número inteiro (grande, até 2 bilhões)
- *Double* → Número decimal
- *String* → Texto
- *Date* → Data
- *Range* → Célula/Intervalo

Para exemplificar, na imagem ao lado declaramos 3 variáveis: **linha**, **produto** e **valor\_produto**. Cada uma de um tipo diferente: *Integer*, *String* e *Double*, respectivamente.

A declaração de variáveis não é obrigatória, e provavelmente em algum momento você criará os seus códigos sem declarar o tipo de uma variável sequer e ele vai funcionar perfeitamente.

Porém, existe uma questão muito importante na declaração de variáveis: **uma variável no VBA ocupa um certo espaço de memória**. Caso a variável não seja declarada, o computador terá que armazenar um espaço muito maior para aquela variável pois ele não sabe o tamanho que aquela variável pode ter (um *Integer* ocupa um espaço diferente de um *Double*). Se nenhuma das variáveis for declarada, durante a execução do código o computador precisará de muito mais memória, levando muito mais tempo para executar a macro. Em códigos pequenos, como os que criamos nesta parte inicial, esse tempo não é percebido. Porém, mais para frente, quando trabalharmos com códigos mais complexos, é possível notar a diferença. Um código que leva 10 segundos para rodar, com todas as variáveis corretamente declaradas, pode levar até 7 minutos para rodar caso nenhuma variável tenha sido declarada.

- **Declarando uma Variável no VBA**

Ex:

**Dim linha as Integer**

linha = 10

**Dim produto as String**

produto = "Guaraná Natural"

**Dim valor\_produto as Double**

valor\_produto = 2.50

Se quisermos alterar o código criado para o registro de vendas, declarando a nossa variável, basta incluir no início do código a linha indicada ao lado. Ao clicar em executar, o código será executado normalmente, da mesma forma que havia sido anteriormente. Porém, como dissemos anteriormente, declarar variáveis é uma maneira de otimizar o código e tornar a execução mais rápida, pois estaremos utilizando de forma inteligente o espaço na memória que o computador dedica para armazenar as variáveis.

Outro ponto importante é que declarar as variáveis torna o código mais organizado. Veremos mais para frente que, quando tivermos muitas variáveis, a organização do código vai fazer toda a diferença para o seu entendimento.

```
Sub registro()
    Dim linha As Integer|-----^
    linha = 13
    Cells(linha, 2).Value = "Guaraná Natural"
    Cells(linha, 3).Value = 1.5
    Cells(linha, 4).Value = 10000
End Sub
```

A	B	C	D
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			

1 - Crie uma rotina que registre uma nova venda na tabela abaixo. Deverá ser registrado o produto Guaraná Natural, com preço unitário de R\$1,50 e com 10.000 unidades vendidas.

Em seguida, adapte seu código para registrar o mesmo produto em uma nova linha.

Produtos	Preço Unitário	Quantidade Vendida
Coca Cola	R\$5,00	1500
Joelho de Queijo e Presunto	R\$6,50	200
Coxinha	R\$4,00	500
Guaraná Natural	R\$ 1,50	10000
Guaraná Natural	R\$ 1,50	10000
Guaraná Natural	1,5	10000

Poderíamos também criar uma variável chamada **produto** que recebe o valor do produto a ser registrado. Essa alteração é mostrada no código ao lado.

Mais a frente veremos como tornar esse registro ainda mais automático, por exemplo, fazendo com que o valor da variável **linha** se adapte de acordo com o tamanho da nossa tabela e sempre registre a nova venda na linha correta, sem que a gente precise mudar esse valor na mão.

```
Sub registro()
    Dim linha As Integer
    Dim produto As String
    produto = "Guaraná Natural"
    linha = 13
    Cells(linha, 2).Value = produto
    Cells(linha, 3).Value = 1.5
    Cells(linha, 4).Value = 10000
End Sub
```

1 - Crie uma rotina que registre uma nova venda na tabela abaixo. Deverá ser registrado o produto Guaraná Natural, com preço unitário de R\$1,50 e com 10.000 unidades vendidas.

Em seguida, adapte seu código para registrar o mesmo produto em uma nova linha.

A	B	C	D
1			
2			
3			
4			
5			
6			
7			
8	Produtos	Preço Unitário	Quantidade Vendida
9	Coca Cola	R\$5,00	1500
10	Joelho de Queijo e Presunto	R\$6,50	200
11	Coxinha	R\$4,00	500
12	Guaraná Natural	R\$ 1,50	10000
13	Guaraná Natural	R\$ 1,50	10000
14	Guaraná Natural	1,5	10000

A	B	C	D	E	F
2	<b>Faturamento</b>	R\$ 15.000,00			
3	<b>Imposto Sobre Faturamento</b>	R\$ 3.000,00			
4	<b>Custo sobre Produto Vendido</b>	R\$ 5.250,00			
5	<b>Despesas Operacionais</b>	R\$ 1.200,00			
6	<b>Despesas Financeiras</b>	R\$ 350,00			
7	<b>Outras Despesas</b>	R\$ 497,00			
8					
9	<b>Lucro</b>				
10	<b>Margem</b>				
11					
12					

Variáveis Resultado Operacional

No próximo exercício vamos usar o conceito de variáveis para calcular o Lucro e a Margem de Lucro de acordo com os valores da tabela ao lado.

Como faturamento, temos o valor da célula C2. Já para os custos (despesas) temos os valores de C3 até C7. Isso quer dizer que, o nosso lucro será igual a:

$$= C2 - (C3 + C4 + C5 + C6 + C7)$$

Para resolver esse exercício, iremos criar 4 variáveis: **faturamento**, **custos**, **lucro** e **margem**.

```
Sub calcula_faturamento()
    Dim faturamento As Double
    Dim custos As Double
    Dim lucro As Double
    Dim margem As Double

    faturamento = Cells(2, 3).Value
    custos = Cells(3, 3).Value + Cells(4, 3).Value + Cells(5, 3).Value + Cells(6, 3).Value + Cells(7, 3).Value
    lucro = faturamento - custos
    margem = lucro / faturamento

    Cells(9, 3).Value = lucro
    Cells(10, 3).Value = margem
End Sub
```

A	B	C	D	E	F
2	Faturamento	R\$ 15.000,00			
3	Imposto Sobre Faturamento	R\$ 3.000,00			
4	Custo sobre Produto Vendido	R\$ 5.250,00			
5	Despesas Operacionais	R\$ 1.200,00			
6	Despesas Financeiras	R\$ 350,00			
7	Outras Despesas	R\$ 497,00			
9	Lucro	4703			
10	Margem	0,313533333			
11					
12					



Como exercício,  
atribua a macro  
criada ao botão  
**Calcular Margem.**

O código completo é mostrado na imagem ao lado. Alguns comentários importantes:

1 - O trecho do código circulado em vermelho significa que as 4 variáveis recebem os valores das células da planilha. Já o trecho de código em verde é o contrário: dessa vez, são as células que recebem o valor das variáveis **lucro** e **margem**.

2 - É muito melhor fazer com que as variáveis recebam os valores das células, pois estas podem mudar. Se fizéssemos que faturamento = 15000, por exemplo, nosso código ficaria restrito a esse valor fixo.

Por fim, lembre-se de executar a macro com o atalho F5 (ou ➤).

A	B	C	D	E	F
2	<b>Faturamento</b>	R\$ 15.000,00			
3	<b>Imposto Sobre Faturamento</b>	R\$ 3.000,00			
4	<b>Custo sobre Produto Vendido</b>	R\$ 5.250,00			
5	<b>Despesas Operacionais</b>	R\$ 1.200,00			
6	<b>Despesas Financeiras</b>	R\$ 350,00			
7	<b>Outras Despesas</b>	R\$ 497,00			
8					
9	<b>Lucro</b>	4703			
10	<b>Margem</b>	0,313533333			
11					
12					

Variáveis Resultado Operacional

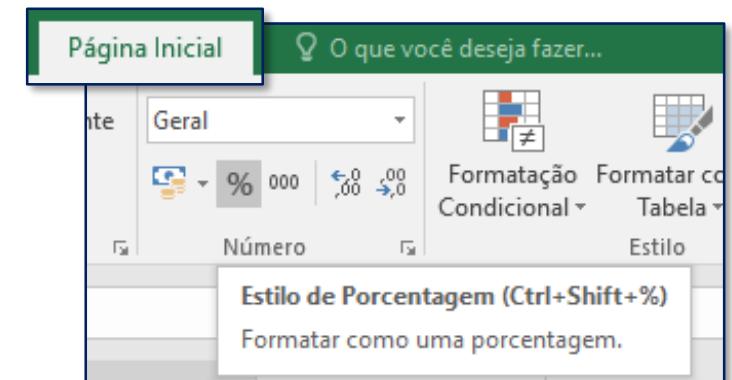
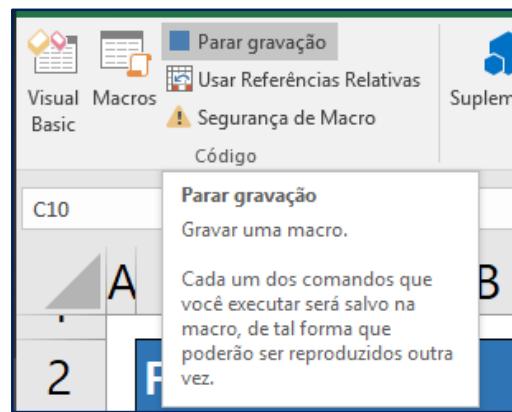
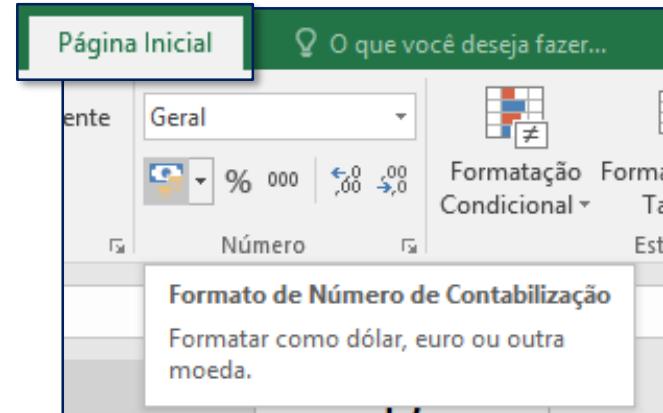
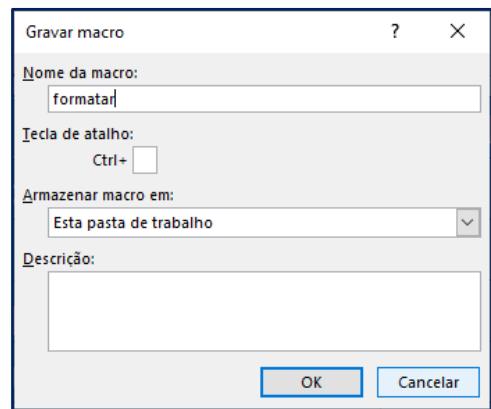
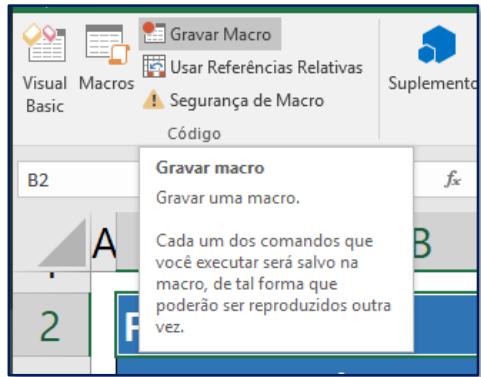
Se você for uma pessoa perfeccionista, provavelmente deve ter notado que o resultado nas células C9 e C10 está desformatado. Ou seja, o lucro não está com formato de moeda e a margem não está no formato de porcentagem.

O que vamos fazer agora é descobrir qual é o código que devemos utilizar para formatar essas células corretamente. Para isso, podemos usar a gravação de macro para acessar o código por trás dessa ação.

O passo a passo para isso está mostrado na página a seguir.

## Módulo 2 – Estrutura e Objetos do VBA – Exercício Variáveis (Parte 2)

67



```
Sub formatar()
    '
    ' formatar Macro
    '
    Range("C9").Select
    Selection.Style = "Currency"
    Range("C10").Select
    Selection.Style = "Percent"
End Sub
```

```
Sub calcula_faturamento()
    Dim faturamento As Double
    Dim custos As Double
    Dim lucro As Double
    Dim margem As Double

    faturamento = Cells(2, 3).Value
    custos = Cells(3, 3).Value + Cells(4, 3).Value + Cells(5, 3).Value + Cells(6, 3).Value
    lucro = faturamento - custos
    margem = lucro / faturamento

    Cells(9, 3).Value = lucro
    Cells(10, 3).Value = margem

    Range("C9").Select
    Selection.Style = "Currency"
    Range("C10").Select
    Selection.Style = "Percent"
End Sub
```

A nova macro foi gravada no Módulo 2. Para visualizar o código, é só dar um duplo clique nele.

Repare na estrutura do código: começamos selecionando a célula C9 e em seguida formatamos para Contábil. Em seguida, selecionamos C10 e formatamos para porcentagem. Pelo código, vemos que a propriedade que mudou a formatação das células foi a propriedade *.Style*.

Este código podemos copiar e colar para o código que estávamos criando no Módulo 1.

```
Sub calcula_faturamento()

Dim faturamento As Double
Dim custos As Double
Dim lucro As Double
Dim margem As Double

faturamento = Cells(2, 3).Value
custos = Cells(3, 3).Value + Cells(4, 3).Value + Cells(5, 3).Value + Cells(6, 3).Value + Cells(7, 3).Value
lucro = faturamento - custos
margem = lucro / faturamento

Cells(9, 3).Value = lucro
Cells(10, 3).Value = margem

Range("C9").Style = "Currency"
Range("C10").Style = "Percent"

End Sub
```



Para fechar, poderíamos ser ainda mais perfeccionistas. O código anterior de formatação possuía duas linhas extras com o comando *Select* de forma desnecessária.

Poderíamos simplesmente ter usado a propriedade *.Style* sem a necessidade de selecionar previamente as células, tornando o código mais limpo e resumido.

```
(Geral) calcula_faturamento

Sub registro()
    Sheets("Variáveis").Activate
    Dim linha As Integer
    Dim produto As String
    produto = "Guaraná Natural"
    linha = 11
    Cells(linha, 2).Value = produto
    Cells(linha, 3).Value = 1.5
    Cells(linha, 4).Value = 10000
End Sub

Sub calcula_faturamento()
    Sheets("Resultado Operacional").Activate
    Dim faturamento As Double
    Dim custos As Double
    Dim lucro As Double
    Dim margem As Double
    faturamento = Cells(2, 3).Value
    custos = Cells(3, 3).Value + Cells(4, 3).Value + Cells(5, 3).Value + Cells(6, 3).Value + Cells(7, 3).Value
    lucro = faturamento - custos
    margem = lucro / faturamento
    Cells(9, 3).Value = lucro
    Cells(10, 3).Value = margem
    Range("C9").Style = "Currency"
    Range("C10").Style = "Percent"
End Sub
```

O nome da aba dentro do comando *Sheets* deve ser escrito exatamente igual ao nome da aba na planilha, incluindo espaços, acentuação, etc.

Um detalhe importante é que, quando criamos códigos em arquivos que possuem mais de uma aba, o código sempre será executado na aba selecionada naquele momento. Isso pode ser um problema, pois criamos um código para registrar uma venda (macro **registro**), mas se estivermos com a aba “Resultado Operacional” selecionada não fará nenhum sentido a macro ser executada nesta aba.

O que podemos fazer para garantir que o código está sendo executado na aba certa é utilizar o comando *Sheets.Activate* para selecionar a aba correta antes de executar de fato a macro.

Imagine agora que rodamos a nossa macro **calcula\_faturamento**, porém não há nenhum valor escrito nas células da coluna C. O seguinte aviso de erro será mostrado.

Para entendermos exatamente o que aconteceu, clicamos em Depurar. Isso vai fazer com que sejamos levados exatamente ao ponto em que ocorreu o erro.

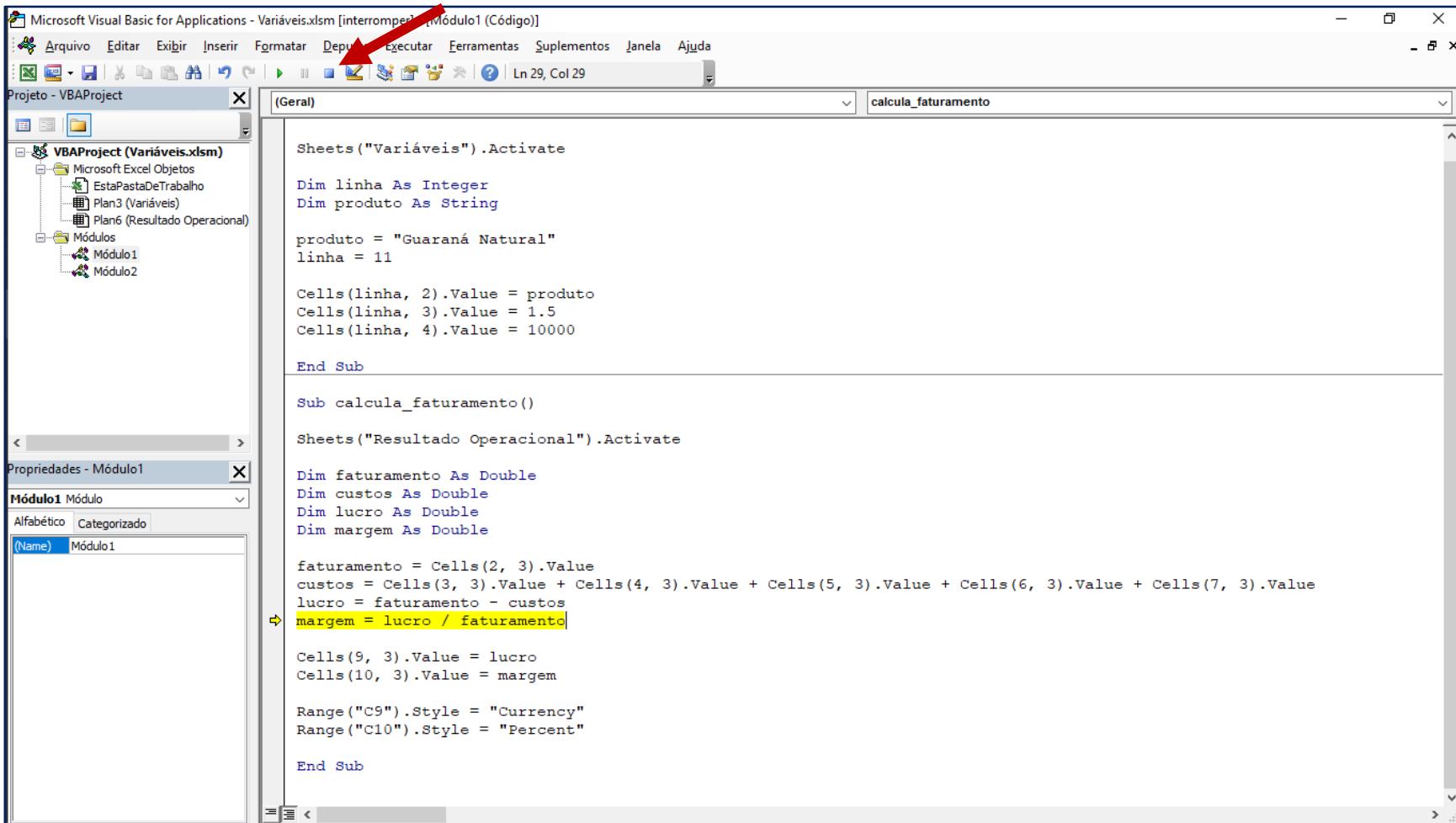
Vemos que a linha marcada foi a linha de cálculo da margem. Mas por que será que houve um erro nesta linha?

The screenshot shows a Microsoft Excel spreadsheet with a VBA error dialog box overlaid. The spreadsheet has columns A through F and rows 2 through 11. Row 2 contains 'Faturamento' in cell B2. Row 3 contains 'Imposto Sobre Faturamento' in cell B3. Row 4 contains 'Custo sobre Produto Vendido' in cell B4. Row 5 contains 'Despesas Operacionais' in cell B5. Row 6 contains 'Despesas Financeiras' in cell B6. Row 7 contains 'Outras Despesas' in cell B7. Row 9 contains 'Lucro' in cell B9. Row 10 contains 'Margem' in cell B10. The VBA error dialog box is centered, displaying the message 'Microsoft Visual Basic', 'Erro em tempo de execução '6':', and 'Estouro'. It has buttons for 'Continuar', 'Fim', 'Depurar', and 'Ajuda'. The 'Depurar' button is highlighted. To the right of the dialog box, the VBA code for the 'calcula\_faturamento' subroutine is shown:

```
Sub calcula_faturamento()
    Sheets("Resultado Operacional").Activate
    Dim faturamento As Double
    Dim custos As Double
    Dim lucro As Double
    Dim margem As Double
    faturamento = Cells(2, 3).Value
    custos = Cells(3, 3).Value + Cells(4, 3).Value
    lucro = faturamento - custos
    margem = lucro / faturamento
    Cells(9, 3).Value = lucro
    Cells(10, 3).Value = margem
    Range("C9").Style = "Currency"
    Range("C10").Style = "Percent"
End Sub
```

## Módulo 2 – Estrutura e Objetos do VBA – Debugando a macro

72



The screenshot shows the Microsoft Visual Basic for Applications (VBA) editor interface. The title bar reads "Microsoft Visual Basic for Applications - Variáveis.xlsxm [interromper] [Módulo1 (Código)]". The menu bar includes "Arquivo", "Editar", "Exibir", "Inserir", "Formatar", "Depurar" (with a red arrow pointing to it), "Executar", "Ferramentas", "Suplementos", "Janela", and "Ajuda". The status bar at the bottom left says "Ln 29, Col 29". The left pane shows the "Projeto - VBAProject" tree with "Variáveis.xlsxm" expanded, showing "Microsoft Excel Objetos", "Plan3 (Variáveis)", "Plan6 (Resultado Operacional)", and "Módulos" with "Módulo1" selected. The main code editor window contains the following VBA code:

```
Sheets("Variáveis").Activate

Dim linha As Integer
Dim produto As String

produto = "Guaraná Natural"
linha = 11

Cells(linha, 2).Value = produto
Cells(linha, 3).Value = 1.5
Cells(linha, 4).Value = 10000

End Sub

Sub calcula_faturamento()

Sheets("Resultado Operacional").Activate

Dim faturamento As Double
Dim custos As Double
Dim lucro As Double
Dim margem As Double

faturamento = Cells(2, 3).Value
custos = Cells(3, 3).Value + Cells(4, 3).Value + Cells(5, 3).Value + Cells(6, 3).Value + Cells(7, 3).Value
lucro = faturamento - custos
margem = lucro / faturamento

Cells(9, 3).Value = lucro
Cells(10, 3).Value = margem

Range("C9").Style = "Currency"
Range("C10").Style = "Percent"

End Sub
```

Antes de dar continuidade, vamos rodar a macro linha por linha para entender o que pode ter acontecido (apesar de saber que você já tem alguma ideia).

Primeiro, clicamos no botão de stop para encerrar a execução da macro. Sempre que o seu código fica com alguma linha marcada de amarelo é porque ele está em execução, então antes de continuar apenas aperta na tecla stop, indicada na imagem ao lado.

The screenshot shows the Microsoft Visual Basic for Applications (VBA) environment. The title bar reads "Microsoft Visual Basic for Applications - Variáveis.xlsxm [interromper] - [Módulo1 (Código)]". The menu bar includes Arquivo, Editar, Exibir, Inserir, Formatar, Depurar, Executar, Ferramentas, Suplementos, Janela, and Ajuda. The status bar at the bottom left says "Ln 29, Col 1". The Project Explorer on the left shows "VBAProject (Variáveis.xlsxm)" with "Microsoft Excel Objetos", "EstaPastaDeTrabalho", "Plan3 (Variáveis)", "Plan6 (Resultado Operacional)", and "Módulos" containing "Módulo1" and "Módulo2". The Properties window on the right is set to "Módulo1 Módulo" and shows "(Name) Módulo1". The code editor window contains the following VBA code:

```
Sheets("Variáveis").Activate
Dim linha As Integer
Dim produto As String
produto = "Guaraná Natural"
linha = 11
Cells(linha, 2).Value = produto
Cells(linha, 3).Value = 1.5
Cells(linha, 4).Value = 10000
End Sub

Sub calcula_faturamento()
Sheets("Resultado Operacional").Activate
Dim faturamento As Double
Dim custos As Double
Dim lucro As Double
Dim margem As Double
faturamento = Cells(2, 3).Value
faturamento = 0 ' Lines 3, 4, 5, 6, 7, 8
lucro = faturamento - custos
margem = lucro / faturamento
Cells(9, 3).Value = lucro
Cells(10, 3).Value = margem
Range("C9").Style = "Currency"
Range("C10").Style = "Percent"
End Sub
```

A red exclamation mark icon is positioned above the code editor. A callout box with a dark blue background and white text points to the line "margem = lucro / faturamento" in the "calcula\_faturamento" subroutine. The text in the callout box reads:

A divisão por zero é um erro bem comum de ocorrer no VBA. Apesar de ser fácil identificar neste código o erro, em códigos maiores este se torna um trabalho mais complicado, e a possibilidade de depurar o código é a melhor forma de encontrar erros e falhas no código.

Para começar a debugar o código você vai usar o atalho F8 (ou Fn + F8). Cada vez que você clica neste atalho, ele executa uma linha do seu código. Você também pode passar o mouse em cima de cada variável para ver o valor armazenado nela.

Se apertarmos a tecla F8 até a linha ao lado ficar marcada de amarelo, vemos que a variável **faturamento** continua sem nenhum valor. Isso é óbvio, pois apagamos todos os valores. E ai o erro ocorre no cálculo da margem porque, ao dividir o lucro por zero ( dado que o faturamento é igual a zero), temos como resultado infinito, que é um erro no VBA.

## Módulo 2 – Estrutura e Objetos do VBA – Debugando a macro

74

The screenshot shows a Microsoft Excel interface with a worksheet titled "Resultado Operacional". The worksheet contains the following data:

	A	B	C
2	Faturamento	R\$ 15.000,00	
3	Imposto Sobre Faturamento	R\$ 3.000,00	
4	Custo sobre Produto Vendido	R\$ 5.250,00	
5	Despesas Operacionais	R\$ 1.200,00	
6	Despesas Financeiras	R\$ 350,00	
7	Outras Despesas	R\$ 497,00	
9	Lucro		
10	Margem		

In the VBA Editor, the code for the "calcula\_faturamento" subroutine is displayed:

```
Sheets("Variáveis").Activate
Dim linha As Integer
Dim produto As String
produto = "Guaraná Natural"
linha = 11

Cells(linha, 2).Value = produto
Cells(linha, 3).Value = 1.5
Cells(linha, 4).Value = 10000

End Sub

Sub calcula_faturamento()
    Sheets("Resultado Operacional").Activate
    Dim faturamento As Double
    Dim custos As Double
    Dim lucro As Double
    Dim margem As Double

    faturamento = Cells(2, 3).Value
    faturamento = 15000
    faturamento = 15000 * Cells(3, 3).Value + Cells(4, 3).Value
    lucro = faturamento - custos
    margem = lucro / faturamento

    Cells(9, 3).Value = lucro
    Cells(10, 3).Value = margem

    Range("C9").Style = "Currency"
    Range("C10").Style = "Percent"
End Sub
```



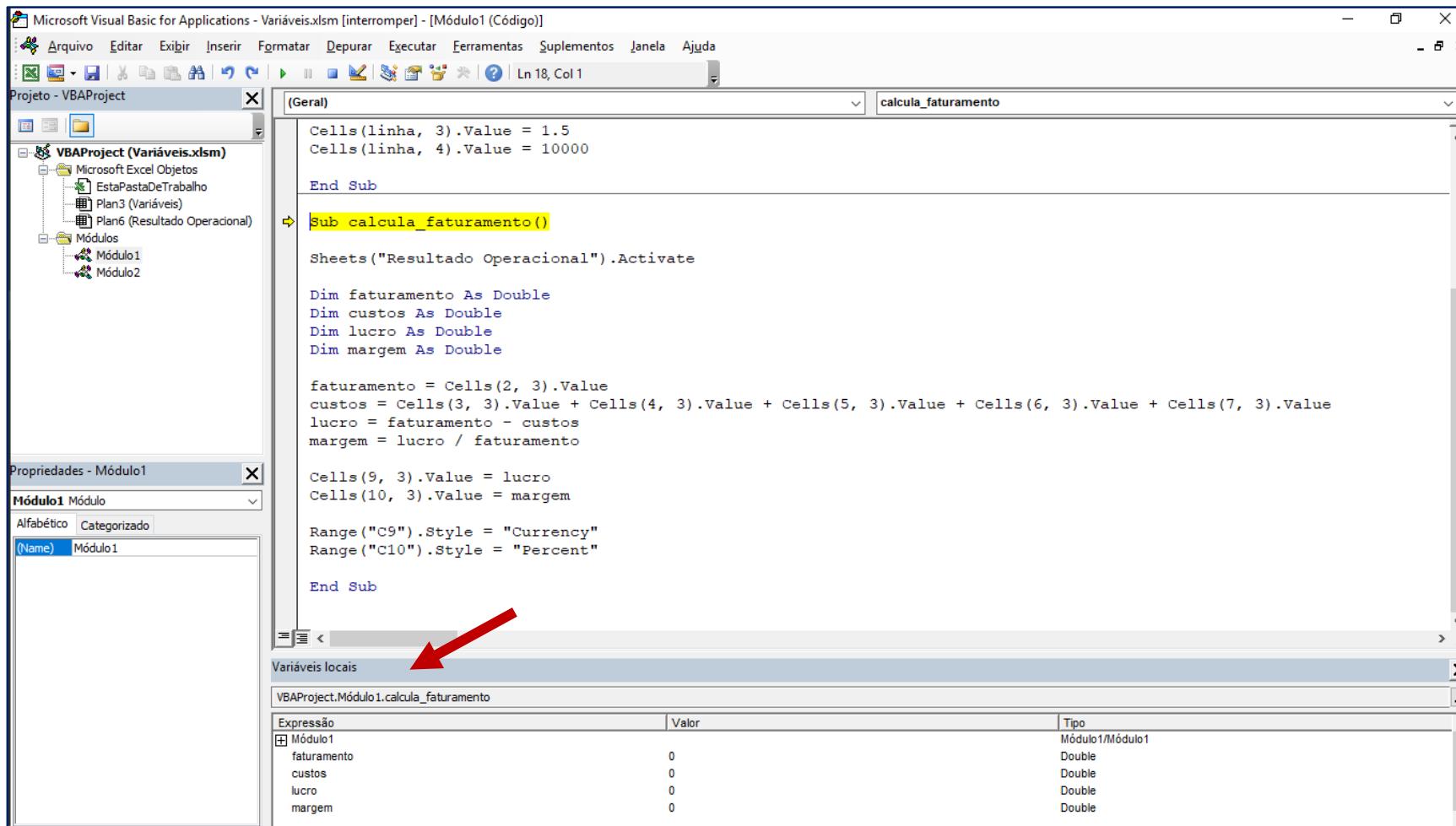
Para conseguir dividir a tela da maneira que é mostrada no print acima, utilize o atalho **WINDOWS** + **Seta para direita/esquerda**. A tecla **WINDOWS** é a janelinha do menu do Windows.

Se voltarmos com os valores da planilha, ao debugar o código para mais uma vez visualizar o valor de cada variável, percebemos que agora, na variável **faturamento**, temos armazenado o valor 15000.

Como exercício, você pode passar o mouse em cima de cada variável para visualizar o valor armazenado em cada uma delas (custos, lucro e margem).

Lembrando que para parar a depuração é só apertar em e para executar o código de uma vez, clicar em .

## Módulo 2 – Estrutura e Objetos do VBA – Janela de Variáveis



The screenshot shows the Microsoft Visual Basic for Applications (VBA) environment. The code editor window displays a module named 'Módulo1' containing a subroutine 'calcula\_faturamento'. The code calculates revenue, costs, profit, and margin based on cell values from row 3 to 10. The 'Properties' window on the left shows 'Módulo1' selected. A red arrow points to the 'Locals' tab in the bottom-left corner of the VBA interface, which lists the local variables 'faturamento', 'custos', 'lucro', and 'margem' all with a value of 0.

```

Cells(linha, 3).Value = 1.5
Cells(linha, 4).Value = 10000

End Sub

Sub calcula_faturamento()
    Sheets("Resultado Operacional").Activate
    Dim faturamento As Double
    Dim custos As Double
    Dim lucro As Double
    Dim margem As Double

    faturamento = Cells(2, 3).Value
    custos = Cells(3, 3).Value + Cells(4, 3).Value + Cells(5, 3).Value + Cells(6, 3).Value + Cells(7, 3).Value
    lucro = faturamento - custos
    margem = lucro / faturamento

    Cells(9, 3).Value = lucro
    Cells(10, 3).Value = margem

    Range("C9").Style = "Currency"
    Range("C10").Style = "Percent"

End Sub

```

Expressão	Valor	Tipo
Módulo1		Módulo1/Módulo1
faturamento	0	Double
custos	0	Double
lucro	0	Double
margem	0	Double

Outra forma de visualizar os valores das variáveis do código é utilizando a Janela de Variáveis. Para exibi-la, você vai no menu **Exibir → Janela 'Variáveis locais'**.

Em seguida, comece a depurar o código com o atalho F8. No canto inferior do ambiente VBA, todas as variáveis serão mostradas. Como ainda estamos na primeira linha de código todas as variáveis estão zeradas.

## Módulo 2 – Estrutura e Objetos do VBA – Janela de Variáveis

76

The screenshot shows the Microsoft Visual Basic for Applications (VBA) environment. The code editor displays a module named 'Módulo1' containing a subroutine 'calcula\_faturamento'. The subroutine calculates faturamento, custos, lucro, and margem based on values from cells 2 to 7 in row 3. It then sets the styles for cells C9 and C10. The 'Variáveis locais' (Local Variables) window at the bottom shows the current values of the variables:

Expressão	Valor	Tipo
Módulo1.faturamento	15000	Double
Módulo1.custos	10297	Double
Módulo1.lucro	4703	Double
Módulo1.margem	0,3135333333333333	Double

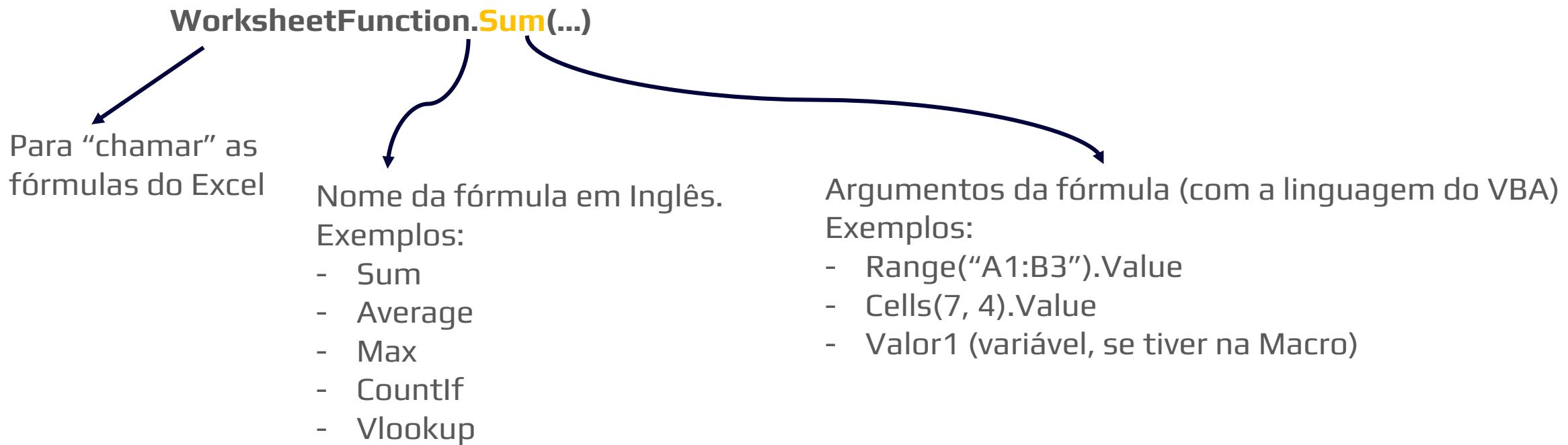
Ao terminar de depurar o código, quando chegamos na última linha, todas as variáveis possuem os valores armazenados.

Esta é uma outra forma até mais elegante de conseguir acompanhar o valor de cada variável conforme você executa a macro.

Como dica final, procure criar essa boa prática de depurar o seu código para entender o passo a passo de cada linha, especialmente neste início de aprendizado. Você verá que isso ajudará muito a entender a lógica por trás dos códigos em VBA.

A partir de agora vamos aprender a como utilizar fórmulas do Excel no VBA. Basicamente, para que o código entenda que você quer acessar uma fórmula do Excel, você precisa utilizar o comando WorksheetFunction, seguido da fórmula do Excel que você quer utilizar, **esta em inglês**. A estrutura abaixo resume a ideia geral:

- **Usando Fórmulas do Excel no VBA**



Ao lado temos dois exemplos de aplicação das fórmulas do Excel dentro do VBA.

A primeira faz uma soma dos valores do intervalo de A2 até A5 e escreve o resultado na célula A1, enquanto o segundo exemplo faz uma média dos valores das células B7 até C10 e armazena o resultado na variável **media**.

- **Usando Fórmulas do Excel no VBA**

Ex:

```
Range("A1").Value = WorksheetFunction.Sum(Range("A2:A5").Value)
```

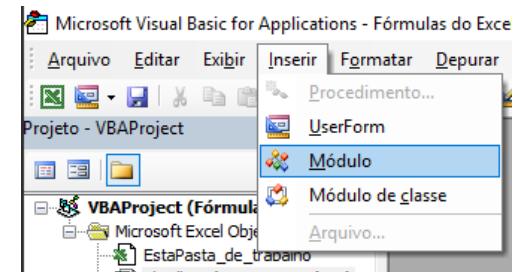
```
media = WorksheetFunction.Average(Range("B7:C10").Value)
```

The screenshot shows a Microsoft Excel interface with a table titled 'Funcionário' containing 19 rows of data. The columns are labeled A through G, with headers: Funcionário, Data de Contratação, Cargo, and Salário. The data includes various employees like Adriana de Azevedo, Adriana Mosqueira Rodrigues Lopes, and Alice Monteiro de Oliveira, along with their hire dates, roles, and salaries.

**Questions:**

1. Qual é o gasto com salário da empresa?
2. Qual é a média dos salários?
3. Quantos estagiários existem na empresa?

Na planilha ao lado vamos fazer 3 exercícios: calcular a soma de gastos com salários, média de salários e quantidade de estagiários na empresa.



Para isso, dentro do ambiente VBA, vamos criar um novo módulo para criar os nossos cálculos.

```
Sub formulas_excel()

'Primeiro devemos identificar em quais células queremos escrever o resultado.
'Ou seja, são as células F3, F6 e F9 que vão receber os resultados das fórmulas.

Range("F3").Value = WorksheetFunction.Sum(Range("D:D"))
Range("F6").Value = WorksheetFunction.Average(Range("D:D"))

'Para o último exercício, devemos usar o Countif (CONT.SE)

Range("F9").Value = WorksheetFunction.CountIf(Range("C:C"), "Estagiário")

'O Countif vai contar os valores de uma coluna, seguindo um critério.
'No caso acima, ele vai contar, na coluna C, quantas vezes aparece "Estagiário".|

End Sub
```

A	B	C	D	E	F
1 Funcionário	Data de Contratação	Cargo	Salário		
2 Adriana de Azevedo	14/11/2015	Diretor	R\$12.320	1. Qual é o gasto com salário da empresa?	6320230
3 Adriana Mosqueira Rodrigues Lopes	14/11/2015	Estagiário	R\$1.600	2. Qual é a média dos salários?	5329,030354
4 Adriana Passos	14/11/2015	Diretor	R\$12.320	3. Quantos estagiários existem na empresa?	132
5 Adriane de Carvalho Gomes	14/11/2015	Analista Senior	R\$4.650		
6 Adriano De Biase	14/11/2015	Analista Júnior	R\$2.850		
7 Adrielle Vieira	16/01/2016	Coordenador	R\$7.200		
8 Alberto Almeida da Silveira	16/01/2016	Analista Júnior	R\$2.850		
9 Alessandra Pascoto da Silveira	16/01/2016	Analista Júnior	R\$2.850		
10 Alex Fernandes	16/01/2016	Analista Pleno	R\$3.700		

Na planilha ao lado vamos fazer 3 exercícios: calcular a soma de gastos com salários, média de salários e quantidade de estagiários na empresa.

Os resultados finais estão mostrados na imagem ao lado. Em seguida, basta executar a macro para visualizar os resultados na planilha.

Como exercício extra, tente seguir o processo da página 66 para fazer com que os valores sejam escritos com a formatação correta em cada uma das células.

Módulo 3

# Estruturas avançadas do VBA

A partir de agora vamos começar a estudar estruturas mais avançadas do VBA e que são extremamente importantes para a construção da maior parte dos códigos que utilizamos no mercado. A estas estruturas damos o nome de **Estruturas de Controle**. As Estruturas de Controle podem ser divididas em: **Estruturas de Seleção** e **Estruturas de Repetição**. Abaixo, temos um resumo destas estruturas, que veremos com detalhes a partir da próxima seção.

### ESTRUTURAS DE CONTROLE NO VBA

#### ESTRUTURAS DE SELEÇÃO

- If ... Then
- If ... Then ... Else
- If ... Then ... Elseif ... Else
- Select Case ... Else

#### ESTRUTURAS DE REPETIÇÃO

- Do Until
- Do While
- For ... Next
- For Each ... Next

As **Estruturas de Seleção** avaliam se uma condição é verdadeira ou falsa e em seguida especificam um ou mais códigos a serem executados, dependendo do resultado desta condição.

As **Estruturas de Repetição** permitem a execução de um bloco de instruções repetidamente. As instruções podem ser repetidas até que uma condição seja Verdadeira ou Falsa (Do Until/Do While) ou repetidas em um número específico de vezes (For/For Each).

A primeira estrutura que veremos é o If, que está resumido na imagem abaixo. Repare que esta estrutura (assim como todas as outras mostradas anteriormente) precisa necessariamente terminar com uma instrução de Fim, no caso, **End If**.

- **Estrutura If no VBA**

```
If condição Then
```



Comandos que vão ser  
executados caso a condição  
seja verdadeira

```
Else
```

Comandos que vão ser  
executados caso a condição  
seja falsa

```
End If
```

- **Exemplo**

```
If Range("A2").Value >= 7 Then
```

	A	B
1	Nota	Situação
2	7,5	Aprovado

```
Range("B2").Value = "Aprovado"
```

```
Else
```

```
Range("B2").Value = "Reprovado"
```

```
End If
```



A **condição** será sempre a comparação entre dois valores.

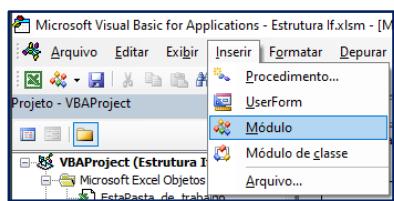
Ex:

- Range("A1").Value > 10
- Cells(1, 1).Value = "RJ"
- Cells(1, 1).Value <> "SP"
- media >=7

Vendedor	Venda	Bônus
João Paulo	R\$ 80.000,00	

Calcular Situação

Situação 1 | Situação 2 | Estados | If & And | If & Or



```
Sub calcula_bonus()

If Cells(3, 3).Value >= 100000 Then

    Range("D3").Value = 0.13 * Range("C3").Value

Else

    Range("D3").Value = 0

End If

End Sub
```



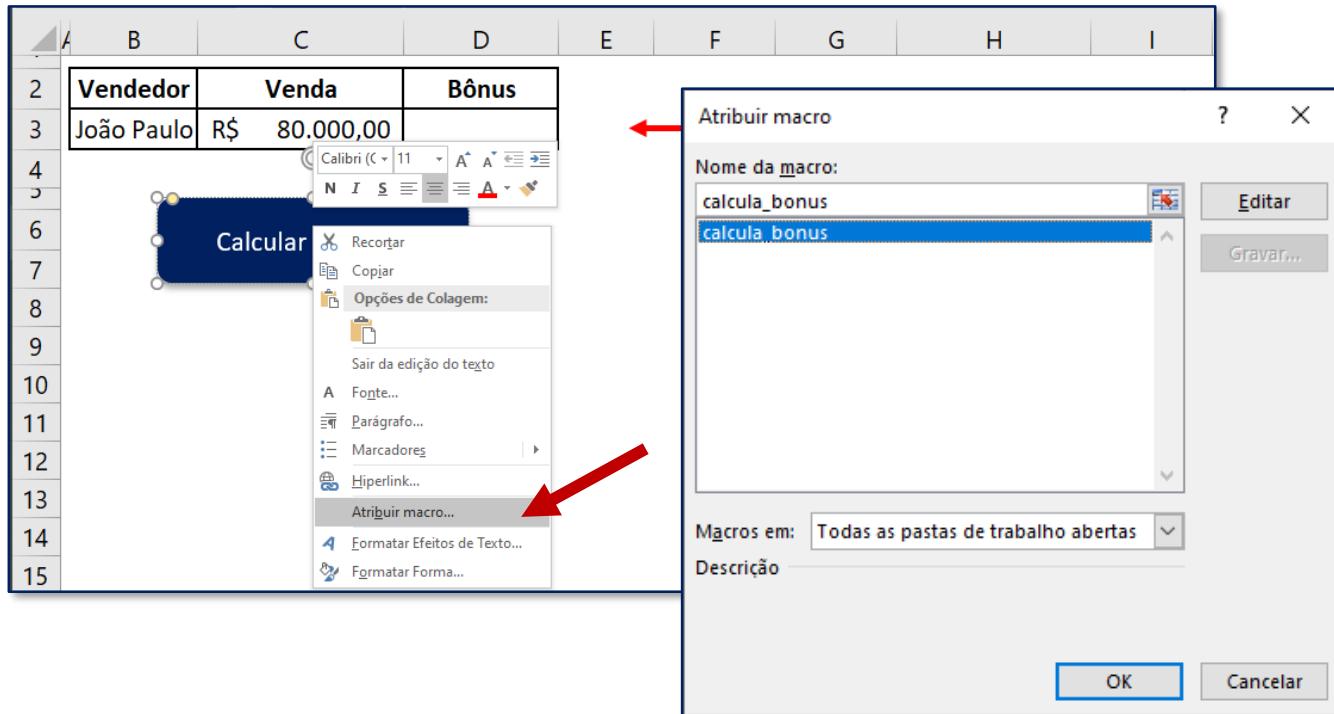
Toda estrutura no VBA possui um início e um fim. Portanto, ao iniciar uma estrutura no VBA, não esqueça de fechar logo em seguida para não esquecer. **No caso do If, é necessário ao final um End If!**

Na planilha ao lado queremos calcular o bônus do vendedor João Paulo. Sabemos que o bônus está condicionado ao fato da venda ter sido **maior ou igual a R\$ 100.000,00**.

Assim, precisamos testar se a venda do vendedor (célula C3) é maior ou igual ao valor de 100000. Se for, o vendedor ganha 13% de bônus em cima das suas vendas, caso contrário, não ganha nada.

Para criar o nosso código, entramos no ambiente VBA com o atalho ALT + F11 e inserimos um novo módulo na guia **Inserir**.

A solução é mostrada ao lado.



Para executar o código, podemos 1) clicar no botão de play (ou usar o atalho F5);



... ou 2) associar a macro ao botão **Calcular Situação**, como já fizemos anteriormente em outros exercícios.

Vimos anteriormente que com a estrutura If... Else só tivemos duas situações possíveis: “Aprovado” ou “Reprovado”. Porém, existem casos em que temos mais de duas situações possíveis. Imagina que quiséssemos acrescentar a situação “Prova Final”. A nova estrutura é mostrada na imagem abaixo:

- **Estrutura If no VBA**

If **condição** Then



Comandos que vão ser executados caso a **primeira** condição seja verdadeira

Elseif **condição** Then



Comandos que vão ser executados caso a **segunda** condição seja verdadeira (e a primeira seja falsa)

Else



End If

- **Exemplo**

If Range("A2").Value >= 7 Then

A	B
Nota	Situação
1	
2	Aprovado

Range("B2").Value = "Aprovado"

Elseif Range("A2").Value >= 5 Then

Range("B2").Value = "Prova Final"

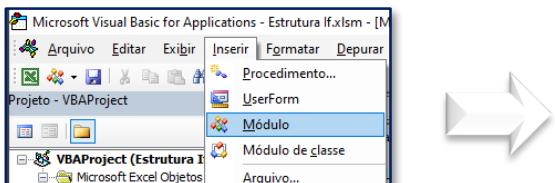
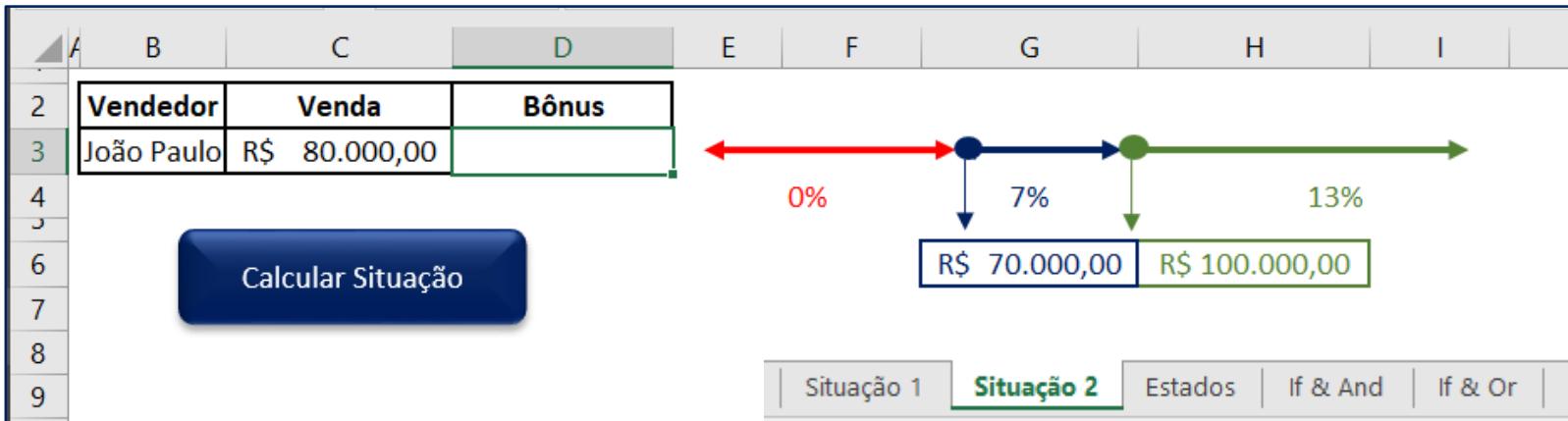
Else

Range("B2").Value = "Reprovado"

End If

## Módulo 3 – Estrutura e Objetos do VBA – Exemplo If ... Elseif

86



```
Sub calcula_bonus2()

    If Cells(3, 3).Value >= 100000 Then

        Range("D3").Value = 0.13 * Range("C3").Value

    ElseIf Cells(3, 3).Value >= 70000 Then

        Range("D3").Value = 0.07 * Range("C3").Value

    Else

        Range("D3").Value = 0

    End If

End Sub
```

No próximo exemplo temos 3 possibilidades de bônus: 13%, 7% e 0%. Neste caso, um simples If .. Else não será suficiente, então usaremos a estrutura If ... Elseif ... Else. Aqui, uma venda **maior ou igual a R\$ 100.000,00** recebe um bônus de 13%. Uma venda entre R\$ 70.000,00 e R\$ 100.000,00 recebe um bônus de 7%, caso contrário, não recebe nada.

Para criar o nosso código, entramos mais uma vez no ambiente VBA com o atalho ALT + F11 e inserimos um novo módulo na guia **Inserir**.

A nova solução é mostrada ao lado.

```
Sub calcula_bonus2 ()  
  
    If Cells(3, 3).Value >= 100000 Then  
  
        Range("D3").Value = 0.13 * Range("C3").Value  
  
    ElseIf Cells(3, 3).Value >= 70000 Then  
  
        Range("D3").Value = 0.07 * Range("C3").Value  
  
    Else  
  
        Range("D3").Value = 0  
  
    End If  
  
End Sub
```

Dois detalhes importantes:

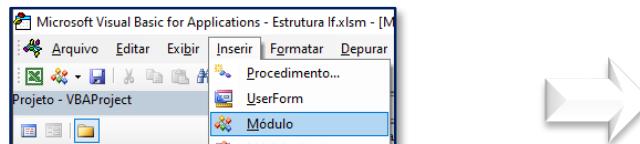
1 - Para que o código fique mais fácil de ler, cada comando que inserimos dentro de uma estrutura de controle deixamos um recuo para mostrar quais comandos estão dentro de quais Ifs/Elseif. Fazemos isso usando a tecla TAB do teclado. Esta organização se chama **indentação** e será uma boa prática a partir de agora nos nossos códigos.

2 - No Elseif não precisamos testar se a venda está entre 70000 e 100000. Se a venda chegou ai, significa que a venda automaticamente é menor que 100000, sendo necessário testar apenas se é maior que 70000.

Vendedor	Estado	Salário
Sérgio Tranjan	RJ	

Calcular Salário

Situação 1 | Situação 2 | **Estados** | If & And | If & Or



```
Sub salario()

    If Cells(3, 3).Value = "RJ" Then
        Cells(3, 4).Value = 7000
    ElseIf Cells(3, 3).Value = "SP" Then
        Cells(3, 4).Value = 5500
    ElseIf Cells(3, 3).Value = "RS" Then
        Cells(3, 4).Value = 5000
    Else
        Cells(3, 4).Value = 4000
    End If
End Sub
```

Mais um exemplo. Dessa vez, queremos calcular o salário do vendedor de acordo com o estado onde ele trabalha. Agora temos 4 possibilidades. Vamos precisar mais uma vez de uma estrutura If ... Elseif, com vários Elseif.

Sim, é possível colocar mais de um Elseif dentro do If. Quantos colocar vai depender da sua necessidade.

A solução para essa nova macro está mostrada ao lado.

A	B	C	D	E	F	G	H	I
2	Vendedor	Venda	Pontualidade	Bônus				
3	Diego Amorim	R\$ 55.000,00	65%		Venda Mínima	Pontualidade Mínima	Bônus	
4					R\$ 50.000,00	75%	15%	
5								
6								
7								
8								

```
Sub calcula_bonus3()
    If Cells(3, 3).Value >= 50000 And Cells(3, 4).Value >= 0.75 Then
        Cells(3, 5).Value = 0.15 * Cells(3, 3).Value
    Else
        Cells(3, 5).Value = 0
    End If
End Sub
```

No próximo exemplo queremos calcular o bônus do vendedor só que desta vez dois critérios devem ser atendidos ao mesmo tempo. Ou seja, não basta que a venda seja maior que a venda mínima, mas a pontualidade também deve ser maior que a pontualidade mínima.

Para avaliar duas condições de uma vez, vamos precisar utilizar a estrutura And.

Na imagem ao lado, vemos que é possível, por meio do And, juntar em um único If dois testes lógicos ao mesmo tempo.

A screenshot of an Excel spreadsheet. The table has four columns: Vendedor, Venda, Pontualidade, and Bônus. The row for Diego Amorim shows a value of R\$ 55.000,00 for Venda and 65% for Pontualidade. The Bônus column contains a formula that results in 0%. A blue button labeled "Calcular Bônus" is positioned below the table.

A screenshot of the Excel ribbon showing the "Fonte" tab selected. The font dropdown shows "Moeda" selected, which is highlighted in grey. The table and button are identical to the one in the previous screenshot.

Por fim, você pode associar esta nova macro ao botão na planilha para executar o código.

É possível que a formatação da célula E3 fique esquisita (como porcentagem). Para mudar, basta ir na guia **Página Inicial** e mudar a formatação para moeda.

Você pode testar outros valores de Venda e Pontualidade para rodar a macro para diferentes casos.

Vendedor	Venda	Desempenho	Bônus
João Martins	R\$ 75.000,00	8,5	

Venda Mínima	Desempenho Mínimo	Bônus
R\$ 80.000,00	8,0	15%

Calcular Bônus

Situação 1 | Situação 2 | Estados | If & And | **If & Or**

```
Sub calcula_bonus4()

If Cells(3, 3).Value >= 80000 Or Cells(3, 4).Value >= 8 Then

    Cells(3, 5).Value = 0.15 * Cells(3, 3).Value

Else

    Cells(3, 5).Value = 0

End If

End Sub
```

**Apesar da venda ter sido menor que R\$ 80.000,00, o desempenho foi maior que 8, então o João ganha bônus.**

Vendedor	Venda	Desempenho	Bônus
João Martins	R\$ 75.000,00	8,5	R\$ 11.250,00

Nosso último exemplo com a estrutura If é mostrado ao lado. Aqui, mais uma vez, queremos calcular o bônus do vendedor, que irá depender da venda e do desempenho.

Porém, a lógica é um pouquinho diferente do exercício anterior. Aqui, tanto faz se o vendedor atinge um ou outro. Se ele vendeu acima de R\$ 80.000,00 mas teve um desempenho menor que 8 não tem problema. O importante é atingir um dos dois critérios. Atingiu um, ganha bônus. Se atingir os dois, melhor ainda. Mas pelo menos um deve ser atendido.

Para essa lógica usaremos a estrutura Or, mostrada na imagem ao lado.

Vamos agora entrar nas estruturas de repetição. As estruturas de repetição têm como objetivo executar comandos repetidas vezes. Ou seja, sempre que você quiser executar um comando para várias linhas de uma tabela, ou ajeitar vários gráficos de uma vez, ou mexer em várias abas/arquivos de uma vez, são essas estruturas que usaremos.

Vamos começar com a estrutura **Do Until**. Assim como o If, a estrutura também deve ter um início e um fim. Ela começa com uma palavra chave (**Do Until**) e termina com uma palavra chave (que aqui, será o **Loop**). Do Until significa literalmente “executar até que” uma condição seja satisfeita.

- **Estrutura Do Until**

**Do Until** **condição**



**Loop**

**Comandos que vão ser executados várias vezes até a condição ser atendida**

Abaixo temos um exemplo de aplicação da estrutura Do Until. Imagine que queremos, na coluna D, realizar o cálculo de estoque mínimo, que será 70% o valor da meta de cada produto. Precisaremos executar repetidas vezes a multiplicação do valor da meta de vendas por 70% (no VBA será 0.7), começando da linha 3. Para isso, usamos a estrutura Do Until.

- **Exemplo Do Until**

linha = 3

Do Until linha>10

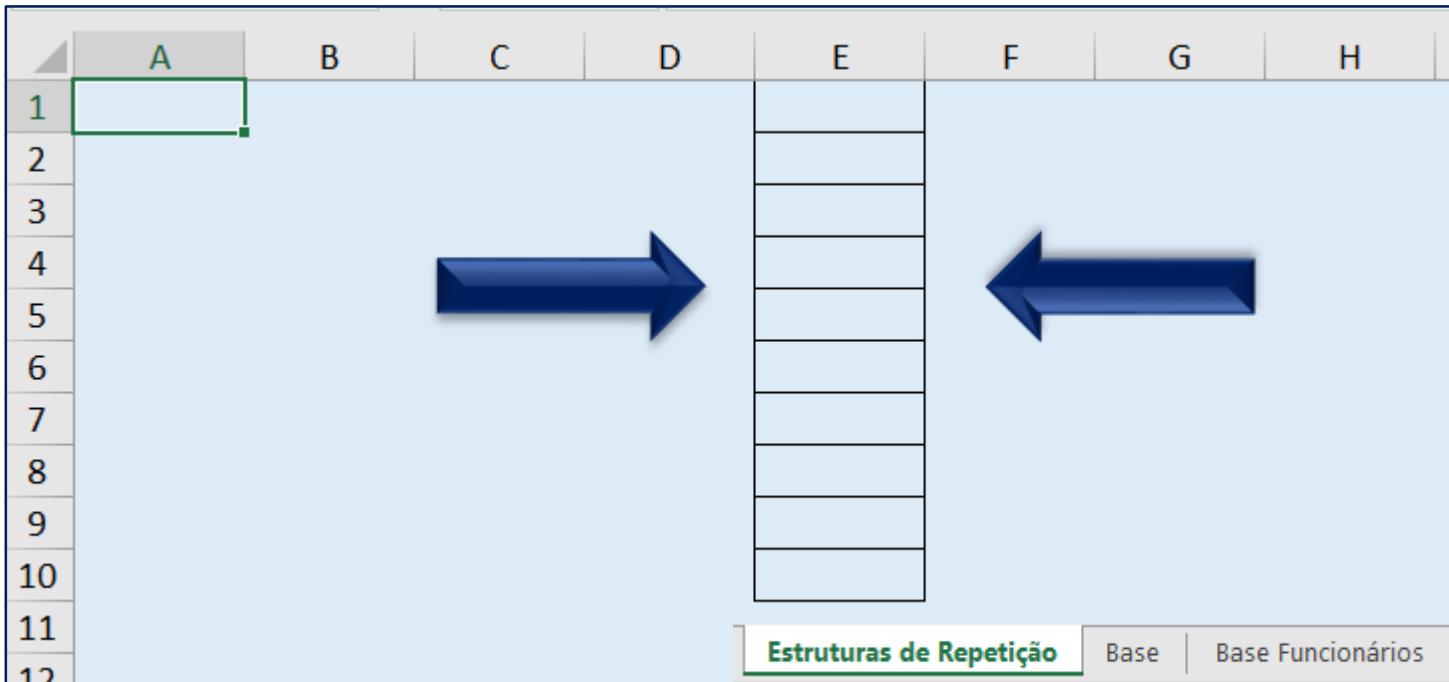
Cells(linha, 4).Value = Cells(linha, 3).Value \* 0.7

linha = linha + 1

Loop

A	B	C	D
2	Produto	Meta de Vendas	Estoque Mínimo (70%)
3	iPhone	80	
4	Galaxy	80	
5	Apple Watch	20	
6	Notebook Asus	30	
7	Macbook	40	
8	Smart TV	100	
9	Notebook Dell	60	
10	Tablet	70	

A leitura do código é a seguinte: como ele deve começar a executar o cálculo a partir da linha 3, começamos com uma variável **linha = 3**. Em seguida vem a nossa estrutura de repetição. A condição será **linha>10** porque o código deve ser executado até que a linha seja maior que 10 (ou seja, assim que linha = 11). A cada Loop, atualizamos o valor da variável **linha** com o trecho **linha = linha + 1**. Assim, no segundo loop, linha = 4. Como na nossa condição linha ainda não é maior que 10, então a multiplicação será feita e a variável linha seguirá sendo atualizada. Isso até que linha seja atualizada para 11. Nesse momento, a condição é atendida e assim o Do Until não será mais executado.



Vamos agora praticar.

Na aba ao lado queremos escrever a palavra “VBA” em cada uma das células do intervalo que vai de E1 até E10. Poderíamos fazer isso utilizando a estrutura:

```
Range("E1:E10").Value = "VBA"
```

Ou então escrevendo 10 vezes o texto “VBA” dentro do nosso código, variando apenas o número da linha da célula onde queremos escrever em cada vez.

Porém, a ideia é que a gente faça esta mesma ação de escrever nas células só que utilizando a estrutura Do Until, que irá repetidas vezes escrever o texto “VBA” até que esteja finalizado.

```
Sub repeticao()

' (1) Criamos uma variável linha que irá armazenar o número da
' linha onde queremos escrever o texto "VBA" a cada repetição.

Dim linha As Integer

' (2) Como queremos começar escrevendo a partir da linha 1 da
' planilha, começamos fazendo: linha = 1.

linha = 1

' (3) A estrutura Do Until linha = 11 significa que uma sequência
' de comandos será executada até que a variável linha seja
' igual a 10. E ai quando: linha = 11, significa que o código
' chegou ao fim e deverá sair do Loop.

Do Until linha = 11

    Cells(linha, 5).Value = "VBA"

    ' (4) Muito importante aqui é atualizar o valor da variável
    ' linha a cada repetição, através da estrutura: linha = linha + 1.

    linha = linha + 1

Loop

End Sub
```



Sempre que iniciar uma estrutura no VBA, não esqueça de fechar essa estrutura. No caso do **Until**, lembre de fechar a estrutura com a palavra **Loop**.

Para começar o nosso código, vamos até o ambiente do VBA utilizando o atalho ALT + F11 (ou ALT + Fn + F11). Para iniciar o nosso código, inserimos um novo módulo na guia **Inserir → Módulo**.

O nome da nossa Sub pode ser “repeticao”. Lembrando que sempre devemos evitar utilizar acentos ou caracteres especiais nos nomes das macros.

O código final está mostrado ao lado, com comentários em verde. Um detalhe importante sobre o **Do Until** é que ele é uma estrutura de repetição que será executada até que uma condição seja atendida. Quando esta condição for atendida, ele simplesmente encerra o *loop*. Como queremos escrever até a linha 10, a condição Do Until linha = 10 faria com que, na linha 10, a estrutura de repetição se encerrasse, pois a condição seria verdadeira. Por isso a nossa condição deve ser linha = 11, pois neste momento, a condição será verdadeira e o código sairá do *loop*, escrevendo pelo menos até a linha 10.

	A	B	C	D	E	F	G	H
1				VBA				
2				VBA				
3				VBA				
4				VBA				
5				VBA				
6				VBA				
7				VBA				
8				VBA				
9				VBA				
10				VBA				
11				VBA				
12				VBA				

O resultado final após executar a macro está mostrado na imagem ao lado. Lembrando que, para executar a macro, basta usar o atalho F5.

Como exercício, execute a macro linha por linha utilizando o comando F8 (ou Fn + F8) assim como fizemos na página 71 e veja o comportamento da macro durante cada Loop. Lembre também de passar o mouse em cima da variável **linha** a cada loop para verificar o valor armazenado e veja que a estrutura se encerra quando linha = 11.

	B	C	D	E	F
2	Produto	Meta de Vendas	Estoque de Segurança (70%)		
3	iPhone	80			
4	Galaxy	80			
5	Apple Watch	20			
6	Notebook Asus	30			
7	Macbook	40			
8	Smart TV	100			
9	Notebook Dell	60			
10	Tablet	70			
11					
12					

Estruturas de Repetição

**Base**

Base Funcionários

No próximo exemplo queremos calcular o estoque de segurança dos produtos, que deve ser feita da linha 3 até a linha 10.

Nesse momento vale uma pausa, pois foi dito da linha 3 **até a linha 10**. Porém, quando você chegar na solução, o código estará **Do Until linha = 11**.

Isso realmente pode parecer confuso, pois se traduzirmos ao pé da letra, o intuitivo a pensar seria Do Until linha = 10. Porém, essa estrutura de repetição será executada **até o momento que a condição for verdadeira**, e no momento que isso acontecer, o Do Until sai do loop. Se fizermos Do Until linha = 10, no momento em que a linha for 10, ele não executará o comando para essa linha, e sairá do loop.

Por isso, **falamos que queremos executar até a linha 10, porém escrevemos no código Do Until linha = 11**.

```
Sub vendas()

' (1) Declarar variáveis não é obrigatório, porém é uma excelente
' prática de programação.

Dim linha As Integer

' (2) Lembrando que queremos começar a escrever a partir da linha 3,
' então devemos começar o código especificando a linha de início.

linha = 3

Do Until linha = 11

    ' (3) A célula na linha = 3, depois 4, depois 5 ... na coluna D
    ' (coluna de número 4) vai receber o valor da célula na linha = 3,
    ' depois 4, depois 5 ... na coluna C (valor de vendas) multiplicado
    ' pelo % de segurança.

    Cells(linha, 4).Value = Cells(linha, 3).Value * 0.7

    ' (4) Não podemos esquecer de atualizar a variável linha a cada loop.

    linha = linha + 1

Loop

End Sub
```

O código final está mostrado ao lado e o resultado na planilha está mostrado abaixo. Lembrando que, para executar a macro, basta utilizar o atalho F5.

	B	C	D
2	Produto	Meta de Vendas	Estoque de Segurança (70%)
3	iPhone	80	R\$ 56,00
4	Galaxy	80	R\$ 56,00
5	Apple Watch	20	R\$ 14,00
6	Notebook Asus	30	R\$ 21,00
7	Macbook	40	R\$ 28,00
8	Smart TV	100	R\$ 70,00
9	Notebook Dell	60	R\$ 42,00
10	Tablet	70	R\$ 49,00
11			

```
Sub vendas()

    '(1) Declarar variáveis não é obrigatório, porém é uma excelente
    'prática de programação.

    Dim linha As Integer

    '(2) Lembrando que queremos começar a escrever a partir da linha 3,
    'então devemos começar o código especificando a linha de início.

    linha = 3

    Do Until Cells(linha, 2).Value = ""

        '(3) A célula na linha = 3, depois 4, depois 5 ... na coluna D
        '(coluna de número 4) vai receber o valor da célula na linha = 3,
        'depois 4, depois 5 ... na coluna C (valor de vendas) multiplicado
        'pelo % de segurança.

        Cells(linha, 4).Value = Cells(linha, 3).Value * 0.7

        '(4) Não podemos esquecer de atualizar a variável linha a cada loop.

        linha = linha + 1

    Loop

End Sub
```



A macro anterior funciona perfeitamente para a nossa tabela no tamanho que ela possui. Porém, se cadastrarmos um novo produto na linha 11, o nosso código não será executado em todas as linhas, parando sempre na linha 10.

Assim, melhor que utilizar o teste linha = 11 seria usar um teste em que o loop fosse executado até o momento em que ele encontrar uma célula vazia, por exemplo, na coluna B. Nesse momento, significa que não temos mais dados na tabela e que o código deverá ser encerrado. Assim, podemos mudar o teste linha = 11 por Cells(linha, 2).Value = "".

Este abrir e fechar aspas é a mesma coisa que vazio. Então, agora, a cada loop, será testado se a célula da coluna B é igual a vazio. Quando for, o loop se encerra e a macro é finalizada, tornando agora o nosso código automático para qualquer quantidade de linhas.

Outra estrutura de repetição, além da Do Until, é a estrutura **Do While**. Diferente da primeira, onde os comandos são executados **até** que uma condição seja atendida, no Do While a sequência de comandos será executado **enquanto** uma condição for atendida.

Assim, quando queríamos escrever um código que seria executado da linha 1 até a linha 10, com o Do Until fizemos: **Do Until linha = 11**. Com o Do While, faremos, por exemplo, **Do While linha < 11**.

- **Estrutura Do While**

**Do While condição**



**Loop**

**Comandos que vão ser executados várias vezes  
enquanto a condição for atendida**

Abaixo temos o mesmo exemplo do estoque mínimo para aplicação da estrutura Do While. A estrutura do código ficaria assim:

- **Exemplo Do While**

linha = 3

Do While linha<11

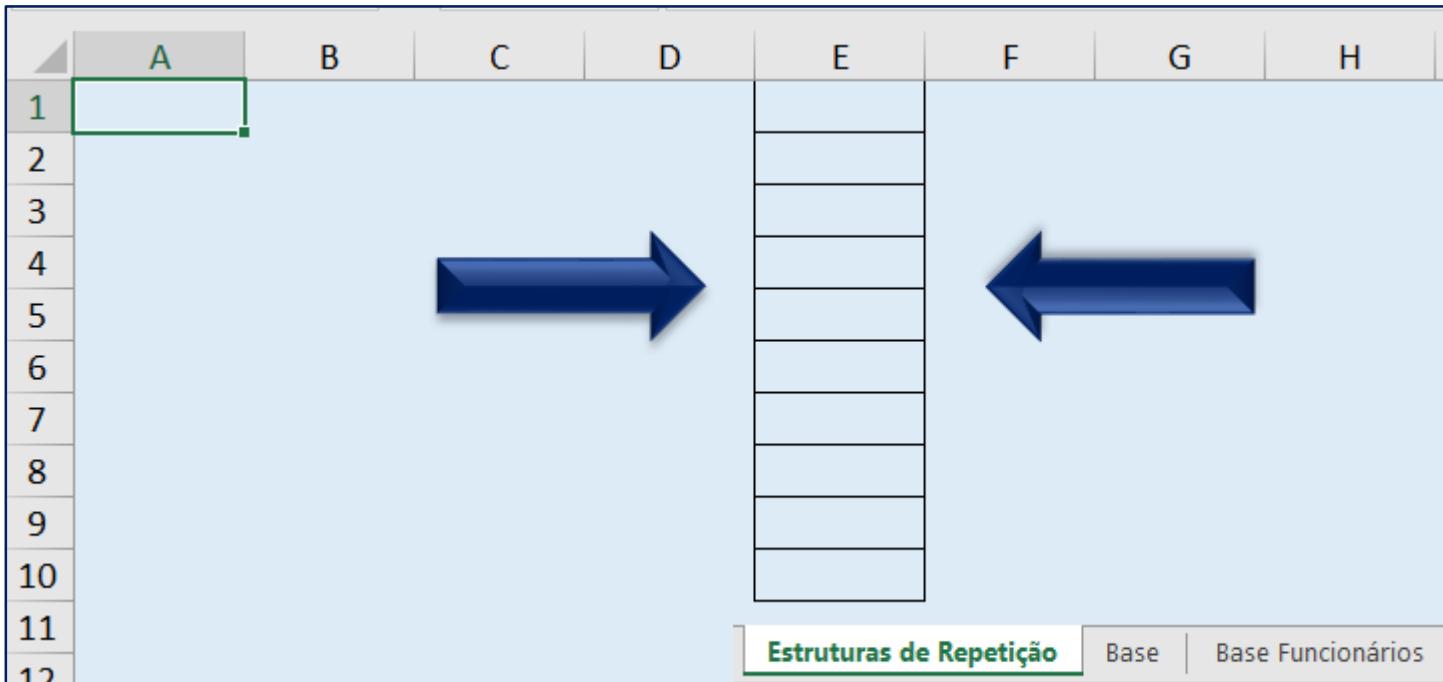
Cells(linha, 4).Value = Cells(linha, 3).Value \* 0.7

linha = linha + 1

Loop

A	B	C	D
2	Produto	Meta de Vendas	Estoque Mínimo (70%)
3	iPhone	80	
4	Galaxy	80	
5	Apple Watch	20	
6	Notebook Asus	30	
7	Macbook	40	
8	Smart TV	100	
9	Notebook Dell	60	
10	Tablet	70	

A leitura do código é a seguinte: como ele deve começar a executar o cálculo a partir da linha 3, começamos com uma variável **linha = 3**. Em seguida vem a nossa estrutura de repetição. A condição será **linha<11** porque o código deve ser executado enquanto a linha for menor que 11 (ou, enquanto linha menor ou igual a 10). A cada Loop, atualizamos o valor da variável **linha** com o trecho **linha = linha + 1**. Assim, no segundo loop, linha = 4. Como na nossa condição linha ainda é menor que 11, então a multiplicação será feita e a variável linha seguirá sendo atualizada. Isso até que linha seja atualizada para 11. Nesse momento, a condição **não** é atendida e assim o Do While não será mais executado.



Vamos praticar em cima da mesma aba “Estrutura de Repetição”.

Queremos novamente escrever o texto “VBA” em cada uma das células do intervalo de E1 até E10.

Para abrir o ambiente do VBA e criar o código, lembre-se de usar o atalho ALT + F11 (ou ALT + Fn + F11).

Caso queira criar um novo módulo, lembre-se também que você pode inserir um na guia **Inserir** no ambiente VBA.

```
Sub repeticao2()

'(1) Criamos uma variável linha que irá armazenar o número da
'linha onde queremos escrever o texto "VBA" a cada repetição.

Dim linha As Integer

'(2) Como queremos começar escrevendo a partir da linha 1 da
'planilha, começamos fazendo: linha = 1.

linha = 1

'(3) A estrutura Do While linha < 11 significa que uma sequência
'de comandos será executada enquanto a variável linha for
'menor que 11 (ou menor ou igual a 10). E ai quando: linha = 11,
'o teste do Do While será falso e o Loop se encerra.

Do While linha < 11

    Cells(linha, 5).Value = "VBA"

    '(4) Muito importante aqui é atualizar o valor da variável
    'linha a cada repetição, através da estrutura: linha = linha + 1.

    linha = linha + 1

Loop

End Sub
```



Sempre que iniciar uma estrutura no VBA, não esqueça de fechar essa estrutura. No caso do **While**, lembre de fechar a estrutura com a palavra **Loop**.

A estrutura Do While é muito parecida com a estrutura Do Until, mudando apenas a condição a ser avaliada, no caso:

Do While linha < 11

Em seguida, execute o código usando o atalho F5. Você obterá o seguinte resultado na sua planilha:

E
VBA

	B	C	D	E
2	Funcionário	Data de Contratação	Tempo de Empresa	
3	Antonio Manhães	22/07/2014		
4	Bianca Paz	04/04/2018		
5	Ana Silva	07/07/2014		
6	Natalia Marinho	30/12/2014		
7	Leonardo Ferreira	23/09/2010		
8	Vitória Santos	29/01/2014		
9	Fernanda Ferreira	06/07/2011		
10	Raissa Negrelli	25/04/2018		
11	João Aires	05/11/2015		
12	Caio Caldas	15/03/2010		
13	Tiago Pereira	27/08/2015		
14	Julia Penteado	21/05/2017		
15	Bernardo Botelho	15/04/2010		
16	Victor Ferreira	03/09/2014		
17	Thays Castro	18/06/2014		
18	Ruan Lopes	09/08/2011		
19				

Estruturas de Repetição

Base

**Base Funcionários**

Nosso próximo exemplo será feito na Base Funcionários. O objetivo aqui é calcular o tempo de empresa de cada funcionário.

Em cada linha da coluna D, o tempo de empresa, em anos, será calculado pela fórmula:

**(HOJE - DATA DE CONTRATAÇÃO)/365**

Como queremos fazer esse cálculo em cada célula da coluna D, vamos usar a estrutura Do While.

```
Sub tempodeempresa()

' (1) Declaramos uma variável idade para armazenar o tempo de empresa
'do funcionário. A seguir vamos entender a vantagem disso.

Dim linha As Integer
Dim idade As Double

' (2) Lembrando que queremos começar a escrever a partir da linha 3,
' então devemos começar o código especificando a linha de início.
linha = 3

Do While linha < 19

    ' (3) Cada célula na coluna D vai fazer a subtração da data de hoje
    ' menos a data de contratação. Para usar a data de hoje em uma fórmula
    ' usamos a estrutura: Date.

    ' Além disso, para organizar o código, armazenamos o resultado da conta
    ' em uma variável chamada idade.

    idade = (Date - Cells(linha, 3).Value) / 365

    ' Como a conta acima retorna um valor quebrado, para ter o tempo certo de
    ' empresa devemos arredondar o valor para baixo. Para isso usamos a função
    ' do Excel chamada ROUNDDOWN (ARREDONDAR.PARA.BAIXO do Excel), com zero
    ' casas decimais.

    Cells(linha, 4).Value = WorksheetFunction.RoundDown(idade, 0)

    ' (4) Não podemos esquecer de atualizar a variável linha a cada loop.

    linha = linha + 1

Loop

End Sub
```

O código ao lado mostra a solução final.

Observe que usamos uma função chamada **RoundDown** para arredondar corretamente os tempos de empresa calculados em anos. Como a divisão por 365 dias resulta em números com casas decimais, precisamos arredonda-los. Porém, alguém que tenha 8,6 anos de empresa, pelo arredondamento comum do Excel, irá para 9 anos. Só que 8,6 anos na verdade são 8 anos de empresa (sem considerar os meses, apenas os anos) então devemos arredondar os valores para baixo. Por isso o uso da função RoundDown.

	B	C	D	E
2	Funcionário	Data de Contratação	Tempo de Empresa	
3	Antonio Manhães	22/07/2014	5,0	
4	Bianca Paz	04/04/2018	1,0	
5	Ana Silva	07/07/2014	5,0	
6	Natalia Marinho	30/12/2014	5,0	
7	Leonardo Ferreira	23/09/2010	9,0	
8	Vitória Santos	29/01/2014	6,0	
9	Fernanda Ferreira	06/07/2011	8,0	
10	Raissa Negrelli	25/04/2018	1,0	
11	João Aires	05/11/2015	4,0	
12	Caio Caldas	15/03/2010	9,0	
13	Tiago Pereira	27/08/2015	4,0	
14	Julia Penteado	21/05/2017	2,0	
15	Bernardo Botelho	15/04/2010	9,0	
16	Victor Ferreira	03/09/2014	5,0	
17	Thays Castro	18/06/2014	5,0	
18	Ruan Lopes	09/08/2011	8,0	
19	Estruturas de Repetição   Base   <b>Base Funcionários</b>			
20				

Neste código podemos fazer uma automatização parecida com a que fizemos para o Do Until na página 99. Como podemos ter o cadastro de novos funcionários, a ideia é que a macro seja executada sempre até o final da tabela, e não apenas enquanto a linha for menor que 19, pois a nossa tabela fatalmente vai aumentar em algum momento.

```
Sub tempodeempresa2()

'(1) Declaramos uma variável idade para armazenar o tempo de empresa
'do funcionário. A seguir vamos entender a vantagem disso.

Dim linha As Integer
Dim idade As Double

'(2) Lembrando que queremos começar a escrever a partir da linha 3,
'então devemos começar o código especificando a linha de início.
linha = 3

Do While Cells(linha, 2).Value <> ""

'(3) Cada célula na coluna D vai fazer a subtração da data de hoje
'menos a data de contratação. Para usar a data de hoje em uma fórmula
'usamos a estrutura: Date.

'Além disso, para organizar o código, armazenamos o resultado da conta
'em uma variável chamada idade.

idade = (Date - Cells(linha, 3).Value) / 365

'Como a conta acima retorna um valor quebrado, para ter o tempo certo de
'empresa devemos arredondar o valor para baixo. Para isso usamos a função
'do Excel chamada ROUNDDOWN (ARREDONDAR.PARA.BAIXO do Excel), com zero
'casas decimais.

Cells(linha, 4).Value = WorksheetFunction.RoundDown(idade, 0)

'(4) Não podemos esquecer de atualizar a variável linha a cada loop.

linha = linha + 1

Loop

End Sub
```

Seguindo uma lógica parecida com a que já vimos para o Do Until, queremos executar a macro enquanto os valores de uma determinada coluna (por exemplo, da coluna B) sejam diferentes de vazio. O sinal de diferente é feito com um sinal de menor seguido por um sinal de maior.

A imagem ao lado indica a modificação que deverá ser feita para automatizar o nosso código.

Você deve ter percebido que em geral utilizamos a estrutura Cells para escrever nas células, em vez da estrutura Range. A justificativa para isso é até intuitiva: em estruturas de repetição, é muito mais fácil lidar com a variável linha dentro da estrutura Cells do que dentro da estrutura Range porque, no Range, o nome das células é escrito como texto.

Isso não significa que não conseguimos trabalhar com o Range nesses casos. Inclusive, uma maneira equivalente de resolver está mostrada ao lado. No lugar de Cells, poderíamos usar a estrutura:

```
Range("E" & linha).Value = "VBA"
```

Onde a variável linha é concatenada com um e comercial (&) à letra referente à coluna. Assim, quando linha for igual a 5, por exemplo, o código acima será entendido como:

```
Range("E5").Value = "VBA"
```

Qual dos dois usar vai depender do que você preferir. Ambos chegam no mesmo resultado.

	A	B	C	D	E	F	G	H
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								

```
Sub repeticao2()
    Dim linha As Integer
    linha = 1

    Do While linha < 11

        'Cells(linha, 5).Value = "VBA"
        Range("E" & linha).Value = "VBA"

        linha = linha + 1
    Loop
End Sub
```

A	B	C	D	E	F	G
2	<b>Crie uma macro que calcule o bônus de cada funcionário na tabela abaixo segundo a regra:</b>					
3	<b>Faixa de Atingimento da Meta</b>	<b>Bônus</b>				
4	0 a 90%	R\$	-			
5	> 90% a 99%	R\$	500,00			
6	100% ou maior	R\$	3.000,00			
7						
8						
9	<b>Nome do Funcionário:</b>	<b>% Atingimento da Meta</b>	<b>Bônus</b>			
10	Fabio	70%				
11	Diogo	30%				
12	Álvaro	99%				
13	Sofia	65%				
14	Camila	127%				
15						
16						

**Combinação 1**    **Combinação 2**

Neste e no próximo exercício vamos praticar a combinação de 2 estruturas: uma lógica (If) e outra de repetição.

Para este primeiro exercício, vamos utilizar como estrutura de repetição o Do Until.

A ideia aqui é a seguinte: para cada funcionário da tabela ao lado vamos calcular o bônus. Porém, o bônus está condicionado a uma faixa de atingimento da Meta. Se o % de atingimento for de 0 a 90%, então o funcionário não recebe bônus. Se o % de atingimento estiver entre 90 e 99%, então o bônus será de R\$ 500. E se o % de atingimento for maior que 100%, então o bônus será de R\$ 3.000.

```
Sub combinacao()
    Dim linha As Integer
    linha = 10
    Do Until Cells(linha, 2).Value = ""
        If Cells(linha, 3).Value >= 1 Then
            Cells(linha, 4).Value = 3000
        ElseIf Cells(linha, 3).Value > 0.9 Then
            Cells(linha, 4).Value = 500
        Else
            Cells(linha, 4).Value = 0
        End If
        linha = linha + 1
    Loop
End Sub
```



Como queremos combinar duas estruturas, temos que pensar como será a ordem delas no nosso código. Quem vem primeiro? A estrutura de repetição ou a estrutura lógica do If.

A resposta certa é: o Do Until vem primeiro. Isso porque, como queremos executar uma mesma ação (testar o % de meta do funcionário) diversas vezes, então a nossa estrutura principal é o Do Until. Assim, começando da linha 10, queremos executar o teste If até que o loop encontre uma célula vazia. Assim que essa célula vazia for encontrada, significa que chegamos no final da tabela e o código deve ser finalizado.

A solução está mostrada no código ao lado.

Cuidado para não esquecer de fechar as estruturas Do Until e If! Lembre-se que toda estrutura no VBA possui um início e um fim. Se você esquecer do End If ou do Loop o seu código retornará um erro!

A	B	C	D	E	F	G
2	<b>Crie uma macro que preencha os prefixos segundo a tabela abaixo:</b>					
3	<b>Estado</b>	<b>Prefixo</b>				
4	RJ	21				
5	SP	11				
6	MG	31				
7	Outros	Desconhecido				
8						
9	<b>Filiais Hash&amp;Co</b>	<b>Telefone:</b>	<b>Estado:</b>	<b>Prefixo:</b>		
10	Filial Origem	98484-7485	RJ			
11	Filial 1ª Expansão	99384-6280	SP			
12	Filial ABC	99100-7746	MG			
13	Filial EFG	98485-0098	RS			
14	Filial Barra	97163-8457	RJ			
15						
16						
17						

[Combinação 1](#) [Combinação 2](#)

Neste segundo exercício queremos escrever o prefixo correto na coluna E de acordo com o estado da filial. Para isso, temos uma tabela auxiliar no começo da planilha que resume os prefixos de acordo com o estado.

Como teremos que executar essa ação de escrever repetidas vezes na coluna E o prefixo dos estados usaremos uma estrutura de repetição. Dessa vez, será o **Do While**.

```
Sub combinacao2()
    Dim linha As Integer
    linha = 10
    Do While Cells(linha, 2).Value <> ""
        If Cells(linha, 4).Value = "RJ" Then
            Cells(linha, 5).Value = 21
        ElseIf Cells(linha, 4).Value = "SP" Then
            Cells(linha, 5).Value = 11
        ElseIf Cells(linha, 4).Value = "MG" Then
            Cells(linha, 5).Value = 31
        Else
            Cells(linha, 5).Value = "Desconhecido"
        End If
        linha = linha + 1
    Loop
End Sub
```



Lembre sempre de organizar o seu código dando espaços a cada início de uma nova estrutura utilizando o atalho TAB. Caso tenha esquecido como fazer para indentar o código, volte na página 87.

Vamos criar uma macro chamada combinacao2.

A ideia aqui é bem parecida com o exercício anterior. Declaramos uma variável linha, pois ela irá mudar a cada loop. Em seguida, como estrutura de repetição, utilizamos o **Do While**. Aqui é importante lembrar da lógica: **o Do While irá executar uma ação enquanto uma condição for verdadeira**.

Neste caso, para deixar o nosso código inteligente para que ele sempre execute até o final da tabela, usamos a estrutura Do While Cells(linha, 2).Value <> "", ou seja, faça enquanto a célula da coluna B seja diferente de vazio.

Mas faça o que?

Os comandos que serão executados são os testes lógicos para verificar qual é o estado em questão e, dependendo do estado, iremos escrever o seu respectivo prefixo.

Por fim, não podemos esquecer de atualizar a variável linha logo após as estruturas do If. Para executar a macro, basta usar o atalho F5.

Agora vamos falar de uma das estruturas de repetição mais utilizadas no VBA: a estrutura **For**. Esta estrutura tem exatamente o mesmo objetivo do While ou Do Until: executar uma série de comandos repetidas vezes. Só que com algumas vantagens que veremos na página seguinte.

O For possui a seguinte estrutura:

- **Estrutura For**

**For variável = início To fim**



**Comandos que vão ser executados várias vezes até chegar no valor final da variável**

**Next**

Como dito anteriormente, o For possui algumas vantagens em relação a outras estruturas de repetição. Para entender essas diferenças, vejamos aquele mesmo exemplo de estoque mínimo, só que agora utilizando a estrutura For:

- **Exemplo For**

For **linha = 3 To 10**

**Cells(linha, 4).Value = Cells(linha, 3).Value \* 0.7**

**Next**

A	B	C	D
2	Produto	Meta de Vendas	Estoque Mínimo (70%)
3	iPhone	80	
4	Galaxy	80	
5	Apple Watch	20	
6	Notebook Asus	30	
7	Macbook	40	
8	Smart TV	100	
9	Notebook Dell	60	
10	Tablet	70	

A primeira vantagem desta estrutura é que podemos dar um valor inicial e final para a variável **linha** diretamente na estrutura For, com: For **linha = 3 To 10**. Isso não era possível com as outras estruturas. A primeira linha completa do código podemos ler como: Para a linha igual a 3 até 10... E ai é executado o comando referente ao cálculo do estoque mínimo. Outra vantagem é que não precisamos atualizar o valor da **linha** com o trecho de código: **linha = linha + 1**. O comando **Next** ao final se encarrega de incrementar automaticamente o valor de linha a cada loop. Assim, não há o risco de esquecermos de atualizar o valor da variável linha e entrar em um loop infinito. Só não podemos esquecer de fechar essa estrutura com o comando **Next**, lembrando que todas as estruturas possuem um início e um fim.

A	B	C	D	E	F	G	H
1				VBA			
2				VBA			
3				VBA			
4				VBA			
5				VBA			
6				VBA			
7				VBA			
8				VBA			
9				VBA			
10				VBA			
11				VBA			
12				VBA			

Estruturas de Repetição    Notas    Base Funcionários    Black Friday    Compilação

```
Sub nova_estrutura()
    Dim linha As Integer
    For linha = 1 To 10
        Cells(linha, 5).Value = "VBA"
    Next
End Sub
```

Como exemplo de utilização do For, vamos refazer o nosso código de escrever em cada uma das células da coluna E, na aba Estruturas de Repetição.

O código final é mostrado na imagem ao lado. Repare que a estrutura é até mais simples do que com as estruturas Do While e Do Until.

Aqui, não precisamos do comando linha = linha + 1. O comando Next já se encarrega de atualizar a variável linha assim que for passar para o próximo loop.

Como exercício, execute a macro linha por linha utilizando o comando F8 (ou Fn + F8) assim como fizemos na página 71 e veja o comportamento da macro durante cada Loop. Lembre também de passar o mouse em cima da variável **linha** a cada loop para verificar que o valor armazenado na variável se atualiza automaticamente.

	B	C	D	E	F	G	H
2	Aluno	Nota	Situação		Média	Situação	
3	João Paulo Martins	7,8			< 5,0	Reprovado	
4	Eduardo Julianelli	8,6			5,0 <= Média < 7,0	Prova Final	
5	Fred Araújo	2,6			>= 7,0	Aprovado	
6	Carolina Cotta	6,5					
7	Marcus Cavalcanti	4,5					
8	Arthur Barreto	8,9					
9	Alon Pinheiro	2,1					
10	João Lira	1,3					
11	Raíssa Rocha	6,8					
12							
13							

Estruturas de Repetição    Notas    Base Funcionários    Black Friday    Compilação

No próximo exercício vamos combinar duas estruturas (For e If) para calcular a situação de cada um dos alunos de acordo com as suas respectivas notas.

Lembrando a ideia de que a estrutura principal é o For, pois queremos fazer uma determinada ação (um teste com a função If) para cada linha da tabela.

Reforçando a ideia de sempre fechar uma estrutura assim que a abrimos, comece o código como na imagem ao lado. Inicie o For e antes de qualquer comando já feche com um Next. E dentro, inicie o If e já feche com um End If. Com essa boa prática você evitará erros futuros por esquecimento do Next e do End If em códigos que fiquem relativamente mais complexos.

```
Sub situacao_alunos()
    Dim linha As Integer
    For linha = 3 To 11
        If Cells(linha, 3).Value >= 7 Then
            Cells(linha, 4).Value = "Aprovado"
        ElseIf Cells(linha, 3).Value >= 5 Then
            Cells(linha, 4).Value = "Prova Final"
        Else
            Cells(linha, 4).Value = "Reprovado"
        End If
    Next
End Sub
```

```
Sub situacao_alunos()
    Dim linha As Integer
    For linha = 3 To 11
        If Cells(linha, 3).Value >= 7 Then
            |
        End If
    Next
End Sub
```

Completando o código, ele ficará como na imagem ao lado. Temos um loop que irá executar uma série de comandos desde a linha 3 até a linha 11. Estes comandos que serão executados a cada loop basicamente são os testes lógicos para verificar a nota daquele aluno e escrever a sua situação na coluna D.

	B	C	D	E	F	G	H
2	Aluno	Nota	Situação		Média	Situação	
3	João Paulo Martins	7,8	Aprovado		< 5,0	Reprovado	
4	Eduardo Julianelli	8,6	Aprovado		5,0 <= Média < 7,0	Prova Final	
5	Fred Araújo	2,6	Reprovado		>= 7,0	Aprovado	
6	Carolina Cotta	6,5	Prova Final				
7	Marcus Cavalcanti	4,5	Reprovado				
8	Arthur Barreto	8,9	Aprovado				
9	Alon Pinheiro	2,1	Reprovado				
10	João Lira	1,3	Reprovado				
11	Raíssa Rocha	6,8	Prova Final				

Estruturas de Repetição    Notas    Base Funcionários    Black Friday    Compilação

A macro anterior funciona perfeitamente para a nossa tabela no tamanho que ela possui. Porém, se cadastrarmos um novo aluno na linha 12, o nosso código não será executado em todas as linhas, parando sempre na linha 11.

Seria isso um Déjà vu?

Então, na página 99 fizemos uma otimização no nosso código Do Until para fazer com que ele funcionasse até o final da tabela. Usamos até aquela estrutura de Do Until celula = "", ou seja, fazer até que a célula seja vazia. Aqui no For não temos um comando similar, pois em sua estrutura não podemos fazer um teste lógico. Então, temos que descobrir de fato qual é o número da última linha preenchida da tabela e fazer algo do tipo:

**For linha = 3 To ultima\_linha**



	B	C	D
2	Aluno	Nota	Situação
3	João Paulo Martins	7,8	Aprovado
4	Eduardo Julianelli	8,6	Aprovado
5	Fred Araújo	2,6	Reprovado
6	Carolina Cotta	6,5	Prova Final
7	Marcus Cavalcanti	4,5	Reprovado
8	Arthur Barreto	8,9	Aprovado
9	Alon Pinheiro	2,1	Reprovado
10	João Lira	1,3	Reprovado
11	Raíssa Rocha	6,8	Prova Final
12			



	B	C	D
2	Aluno	Nota	Situação
3	João Paulo Martins	7,8	Aprovado
4	Eduardo Julianelli	8,6	Aprovado
5	Fred Araújo	2,6	Reprovado
6	Carolina Cotta	6,5	Prova Final
7	Marcus Cavalcanti	4,5	Reprovado
8	Arthur Barreto	8,9	Aprovado
9	Alon Pinheiro	2,1	Reprovado
10	João Lira	1,3	Reprovado
11	Raíssa Rocha	6,8	Prova Final
12			

Antes de entender como faremos isso no VBA, vamos entender como descobrimos a última célula preenchida usando um atalho do Excel.

Se selecionarmos a célula B2 e usarmos o atalho CTRL + Seta para baixo, a célula selecionada será a B11, que é a ultima preenchida na coluna B. Ou seja, o que esse atalho faz é percorrer um intervalo de células preenchidas e parar na última célula preenchida.

Saber a última célula preenchida na coluna B, mais especificamente a linha dessa última célula preenchida, é exatamente o que precisamos para automatizar a estrutura For.

```
Sub situacao_alunos()

Dim linha As Integer

ult_linha = Range("B2").End(xlDown)  
For linha = 3 To 11
    If Cells(linha, 3).Value <= 7 Then
        Cells(linha, 4).Value = "Aprovado"
    End If
Next linha
```

```
Sub situacao_alunos()

Dim linha As Integer

ult_linha = Range("B2").End(xlDown).Row
For linha = 3 To 11
    If Cells(linha, 3).Value >= 7 Then
```

O código que fará o papel do CTRL + Seta para baixo está mostrado ao lado. Começamos criando uma variável chamada ult\_linha. Você pode declará-la ou não, fica a seu critério. Recomendamos que declare, mas por simplicidade, vamos passar direto nisso.

Em seguida, para dar o CTRL + Seta para baixo a partir da célula B2, começamos escrevendo o comando Range("B2").End ... Este comando permite que a gente percorra um intervalo de células em qualquer direção. No nosso caso, queremos percorrer para baixo, então usamos a opção "xlDown".

Para descobrir a linha da última célula preenchida, acrescentamos o comando .Row.

```
Sub situacao_alunos()
    Dim linha As Integer
    ult_linha = Range("B2").End(xlDown).Row
    For linha = 3 To ult_linha|
        If Cells(linha, 3).Value >= 7 Then
            Cells(linha, 4).Value = "Aprovado"
        ElseIf Cells(linha, 3).Value >= 5 Then
            Cells(linha, 4).Value = "Prova Final"
        Else
            Cells(linha, 4).Value = "Reprovado"
        End If
    Next
End Sub
```

Criada a variável **ult\_linha**, substituímos o valor 11 na linha de código:

For linha = 3 To **11**

Por:

For linha = 3 To **ult\_linha**

Pronto! Agora nosso código está 100% automatizado. Quando rodamos a macro utilizando o atalho de depuração F8, após passar a linha de código referente a **ult\_linha**, temos o seguinte:

```
Sub situacao_alunos()
    Dim linha As Integer
    ult_linha = Range("B2").End(xlDown).Row
    ult_linha = 11
    For linha = 3 To ult_linha
```

	B	C	D	E	F	G	H
2	Aluno	Nota	Situação		Média	Situação	
3	João Paulo Martins	7,8	Aprovado		< 5,0	Reprovado	
4	Eduardo Julianelli	8,6	Aprovado		5,0 <= Média < 7,0	Prova Final	
5	Fred Araújo	2,6	Reprovado		>= 7,0	Aprovado	
6	Carolina Cotta	6,5	Prova Final				
7	Marcus Cavalcanti	4,5	Reprovado				
8	Arthur Barreto	8,9	Aprovado				
9	Alon Pinheiro	2,1	Reprovado				
10	João Lira	1,3	Reprovado				
11	Raíssa Rocha	6,8	Prova Final				
12							
13							

Agora o nosso código irá funcionar para qualquer quantidade de alunos, incluindo qualquer novo aluno adicionado.

Este comando `.End(xlDown)` (assim como seus variantes, `xlUp`, `xlToLeft` e `xlToLeft`) serão muito utilizados nos nossos códigos mais para frente, então é muito importante que este comando fique 100% claro para você.

O próximo exercício é muito parecido com o anterior, com a diferença que agora, em vez de fazer um loop para preencher cada linha da tabela, vamos executar um loop para percorrer cada coluna da tabela e escrever o respectivo bônus.

A lógica é a mesma, vamos usar um For como estrutura principal do nosso código, e dentro desse For, um If para testar as vendas de cada vendedor e registrar o seu respectivo bônus.

	B	C	D	E	F	G	H	I
2	<b>Funcionário</b>	Antonio Manhães	Bianca Paz	Ana Silva	Natalia Marinho	Leonardo Ferreira	Vitória Santos	Fernanda Ferreira
3	<b>Vendas</b>	R\$ 3.470,00	R\$ 7.240,00	R\$ 5.060,00	R\$ 1.520,00	R\$ 3.270,00	R\$ 2.780,00	R\$ 2.010,00
4	<b>Bônus</b>							
5								
6	<b>Vendas</b>	<b>Bônus</b>						
7	< R\$2.500	0%						
8	R\$2.000 <= Média < R\$5.000	10%						
9	≥ R\$5.000	30%						
10								
11								

Estruturas de Repetição

Notas

**Base Funcionários**

Black Friday

Compilação

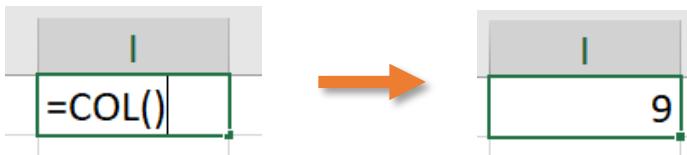
```

Sub bonus()
    Dim coluna As Integer
    For coluna = 3 To 9
        If Cells(3, coluna).Value < 2000 Then
            Cells(4, coluna).Value = 0
        ElseIf Cells(3, coluna).Value < 5000 Then
            Cells(4, coluna).Value = 0.1
        Else
            Cells(4, coluna).Value = 0.3
        End If
    Next
End Sub

```

O resultado final é mostrado ao lado.

O código final é mostrado ao lado. Repare que desta vez o que irá variar no nosso código não é mais a linha. Esta é fixa, pois para todos os funcionários, queremos escrever o seu bônus sempre na linha 4. O que varia aqui na verdade é o número da coluna, que vai de 3 (coluna C) até 9 (coluna I). Se você quiser saber facilmente o número de uma coluna, você pode usar a fórmula COL no Excel:



Por fim, basta fazer os Ifs para testar o valor de vendas de cada vendedor. Repare que dessa vez começamos pelo menor valor (2000) e fomos aumentando. Tanto faz por qual ponta você decide começar, desde que mantenha essa ordem (do menor para o maior ou do maior para o menor).

B	C	D	E	F	G	H	I
Funcionário	Antonio Manhães	Bianca Paz	Ana Silva	Natalia Marinho	Leonardo Ferreira	Vitória Santos	Fernanda Ferreira
Vendas	R\$ 3.470,00	R\$ 7.240,00	R\$ 5.060,00	R\$ 1.520,00	R\$ 3.270,00	R\$ 2.780,00	R\$ 2.010,00
Bônus	10%	30%	30%	0%	10%	10%	10%
Vendas							
< R\$2.500	0%						
R\$2.000 <= Média < R\$5.000	10%						
>= R\$5.000	30%						

```
Sub bonus()

    Dim coluna As Integer

    ult_coluna = Range("B2").End(xlToRight).Column

    For coluna = 3 To ult_coluna

        If Cells(3, coluna).Value < 2000 Then

            Cells(4, coluna).Value = 0

        ElseIf Cells(3, coluna).Value < 5000 Then

            Cells(4, coluna).Value = 0.1

        Else

            Cells(4, coluna).Value = 0.3

        End If

    Next

End Sub
```

Se quisermos automatizar para que a macro seja executada sempre até a última coluna preenchida, podemos usar uma estrutura bem parecida com a utilizada na página 118. Só que aqui, em vez de descobrir a última linha preenchida usando o atalho CTRL + Seta para baixo, nós vamos usar o comando equivalente ao CTRL + Seta para a direita, a partir da célula B2, para encontrar o número da última coluna preenchida, que está indicado na imagem ao lado.

Lembrando que, ao criar uma variável chamada ult\_coluna, você deve atualizar a linha:

For coluna = 3 To 9

Para:

For linha = 3 To ult\_coluna

O próximo exercício é um desafio. O enunciado está mostrado abaixo. O objetivo é o seguinte: queremos saber se, de acordo com as vendas dos vendedores, eles serão promovidos ou não. Caso não sejam, eles ainda terão uma segunda chance. E a lógica é a seguinte:

- Se a média de vendas dos bimestres 1 e 2 for maior ou igual a R\$ 6.000,00, então o vendedor é “Promovido”.
- Se não, vamos substituir as vendas do pior bimestre e calcular quanto ele deve vender na Black Friday para conseguir uma média de R\$ 6.000,00. Ou seja, a venda a ser feita deverá ser = 12000 - venda do melhor bimestre.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
2	Crie uma macro para analisar a promoção dos vendedores seguindo a regra:															
3	O vendedor é promovido se sua média de venda entre os dois bimestres é maior do que 6.000 mensal.															
4	A média de venda dos bimestres é a média entre Bimestre 1 e Bimestre 2. Se essa média for maior do que 6.000, na coluna E deve aparecer "Promovido".															
5	Se a média de vendas dos bimestres for menor do que R\$6.000, então suas vendas da BlackFriday ainda podem garantir sua promoção, substituindo as vendas do pior bimestre.															
6	Se a média entre as vendas da BlackFriday e o melhor bimestre do vendedor for maior do que 6.000 (mesma coisa se a soma das vendas da blackfriday com o melhor bimestre ser $\geq 12.000$ ), então ele é promovido															
7	Seu objetivo, nesse caso, é colocar quanto que o vendedor precisa vender na BlackFriday para ser promovido (Dica: é 12.000 - o melhor bimestre de vendas, caso ele já não tenha sido promovido)															
8																
9																
10	Nome do Vendedor	Bimestre 1	Bimestre 2	Quanto precisa vender na Black Friday												
11	Fabio	R\$ 6.000,00	R\$ 4.500,00													
12	Diogo	R\$ 9.000,00	R\$ 8.000,00													
13	Álvaro	R\$ 5.000,00	R\$ 10.000,00													
14	Sofia	R\$ 1.000,00	R\$ 5.500,00													
15	Camila	R\$ 9.000,00	R\$ 2.500,00													
16																

**Exemplo:** Diogo e Álvaro foram os únicos que tiveram média acima de 6000, portanto, são os únicos promovidos. Agora o Fábio, por exemplo, vendeu 6000 no primeiro bimestre e 4500 no segundo bimestre. Assim, descartamos a sua pior venda (4500) e para conseguir uma média de 6000 ele vai precisar vender 6000 novamente para conseguir a média necessária. Já a Sofia, precisaria vender 6500 (pois a sua melhor venda foi 5500) e a Camila precisaria vender 3000 (pois sua melhor venda foi 9000).

```
Sub black_friday()
    For linha = 11 To 15
        bim1 = Cells(linha, 3).Value
        bim2 = Cells(linha, 4).Value
        media = (bim1 + bim2) / 2
    Next
End Sub
```

Como esse exercício é um pouco mais complicado, vamos resolver bem passo a passo. A primeira coisa que devemos pensar é a seguinte: queremos executar um comando repetidas vezes? Sim, pois queremos saber a situação de cada vendedor, e para isso vamos usar uma estrutura de repetição, que pode ser a estrutura For. Feito isso, o primeiro passo é calcular a média de vendas de cada vendedor. Para isso, precisamos de 2 variáveis para armazenar os valores de vendas do bimestre 1 (bim1) e bimestre 2 (bim2) e uma para calcular a média dos dois bimestres (media). Com isso, temos o suficiente para testar se a média do vendedor é maior ou igual a meta de R\$ 6.000.

```
Sub black_friday()
    For linha = 11 To 15
        bim1 = Cells(linha, 3).Value
        bim2 = Cells(linha, 4).Value
        media = (bim1 + bim2) / 2
        If media >= 6000 Then
            Cells(linha, 5).Value = "Promovido"
        Else
            '...
        End If
    Next
End Sub
```

Para fazer esse teste, precisamos de uma estrutura If. A situação em que o vendedor está promovido é a mais fácil. Basta testar se a média é maior que 6000 e se for escrever na coluna E o texto “Promovido”.

A dificuldade começa a partir do Else...

```
Sub black_friday()

    For linha = 11 To 15

        bim1 = Cells(linha, 3).Value
        bim2 = Cells(linha, 4).Value

        media = (bim1 + bim2) / 2

        If media >= 6000 Then

            Cells(linha, 5).Value = "Promovido"

        ElseIf bim1 >= bim2 Then

            Cells(linha, 5).Value = 12000 - bim1

        Else

            Cells(linha, 5).Value = 12000 - bim2

        End If

    Next

End Sub
```

Acrescentamos então a situação Else. Assim, se a média não foi maior ou igual a 6000, devemos testar se as vendas do bim1 são maiores que as vendas do bim2. Se forem, significa que bim1 foi o melhor bimestre. E para saber quanto falta para vender na Black Friday, basta fazer  $12000 - \text{bim1}$ .

Caso contrário, sabemos que bim2 é o melhor bimestre e para descobrir quanto falta para vender na Black Friday basta fazer  $12000 - \text{bim2}$ .

O código final é mostrado ao lado e o resultado após executar a macro está mostrado abaixo:

Nome do Vendedor	Bimestre 1	Bimestre 2	Quanto precisa vender na Black Friday
Fabio	R\$ 6.000,00	R\$ 4.500,00	R\$ 6.000,00
Diogo	R\$ 9.000,00	R\$ 8.000,00	Promovido
Álvaro	R\$ 5.000,00	R\$ 10.000,00	Promovido
Sofia	R\$ 1.000,00	R\$ 5.500,00	R\$ 6.500,00
Camila	R\$ 9.000,00	R\$ 2.500,00	R\$ 3.000,00

	A	B	C	D	E	F
1	Funcionário	Data de Contratação	Idade	Cargo	Salário	Funcionário Antigo/Atual
2	Adriana Passos	14/11/2015	40	Gerente 2	R\$12.320	Atual
3	Adriane Chagas	14/11/2015	48	Estagiário 2	R\$2.000	Atual
4	Adriane Pinheiro	14/11/2015	61	Coordenador 2	R\$8.100	Atual
5	Adrielle Penna Forte	16/01/2016	46	Estagiário 2	R\$2.000	Antigo
6	Alberto Almeida da Silveira	16/01/2016	31	Analista Júnior	R\$2.850	Atual
7	Alex Fernandes	16/01/2016	40	Analista Pleno	R\$3.700	Antigo
8	Alexandre Correa Rodriguez	16/01/2016	47	Analista Pleno	R\$3.700	Atual
9	Alexandre Falcão do Amaral	16/01/2016	58	Analista Júnior	R\$2.850	Antigo
10	Alexsandro da Silva Costa	16/01/2016	55	Coordenador 1	R\$7.200	Atual
11	Alexsandro Pereira da Costa	16/01/2016	51	Estagiário 1	R\$1.600	Atual
12	Alfredo Stockler	16/01/2016	23	Coordenador 2	R\$8.100	Antigo
13	Alice Silva Alves	16/01/2016	53	Analista Senior	R\$4.650	Atual
14	Aline de Andrade	27/02/2016	23	Coordenador 2	R\$8.100	Atual
15	Aline de Oliveira e Mello	27/02/2016	61	Gerente 2	R\$12.320	Antigo
16	Aline Garcia Lima	27/02/2016	22	Gerente 2	R\$12.320	Antigo
17	Aline Morais	27/02/2016	61	Coordenador 1	R\$7.200	Antigo
18	Aline Sá	27/02/2016	34	Analista Júnior	R\$2.850	Atual
19	Allan de Lira	27/02/2016	30	Analista Júnior	R\$2.850	Atual
20	Allan Souza Santos Candido	27/02/2016	55	Analista Júnior	R\$2.850	Antigo
21	Alon Bernardes de Almeida	27/02/2016	36	Analista Senior	R\$4.650	Atual
22	Alon Fiorenza Duque Pinho	27/02/2016	59	Gerente 1	R\$9.450	Atual
23	Alon Paes de Sousa Guedes	27/02/2016	59	Jovem Aprendiz	R\$800	Atual
24	Alon Palmeira	27/02/2016	42	Jovem Aprendiz	R\$800	Atual
25	Alvaro Ferreira	27/02/2016	24	Coordenador 2	R\$8.100	Atual

No último exercício deste arquivo vamos fazer um código muito comum do dia a dia que é o de excluir linhas de acordo com algum critério. Por exemplo, queremos excluir, da tabela ao lado, todos os funcionários antigos.

Pensando na lógica por trás, queremos criar uma macro que vai checar cada linha da tabela e verificar, na coluna F, se o funcionário é antigo ou atual. E se for antigo, deve excluir toda a linha.

Vamos começar criando o nosso código no ambiente VBA. Para chegar até lá, basta usar o atalho ALT + F11 (ou ALT + Fn + F11).

```
Sub compila()
    ult_linha = Range("A1").End(xlDown).Row
    For linha = 2 To ult_linha
        If Cells(linha, 6).Value = "Antigo" Then
            Rows(linha & ":" & linha).Delete
        End If
    Next
End Sub
```

O código final é relativamente simples e está mostrado ao lado. O único detalhe que temos que entender é a maneira como o VBA entende quando estamos nos referindo a uma linha inteira no Excel. Neste caso, por exemplo, se quisermos selecionar toda a linha 2, o código para isso é:

Rows("2:2").Select

E se quiséssemos excluí-la:

Rows("2:2").Delete

Porém, no nosso código, não podemos fixar uma determinada linha pois as linhas vão variar de acordo com o loop do For. Mas não tem problema, poderíamos simplesmente fazer:

Rows(linha & ":" & linha).Delete

Ou seja, como o "2:2" dentro do comando Rows é um texto, para chegar nesse resultado usando a variável linha teríamos que concatenar essa variável com o texto ":". Assim, a cada loop, a linha se atualiza e estaremos analisando uma linha diferente.

	A	B	C	D	E	F
1	Funcionário	Data de Contratação	Idade	Cargo	Salário	Funcionário Antigo/Atual
2	Adriana Passos	14/11/2015	40	Gerente 2	R\$12.320	Atual
3	Adriane Chagas	14/11/2015	48	Estagiário 2	R\$2.000	Atual
4	Adriane Pinheiro	14/11/2015	61	Coordenador 2	R\$8.100	Atual
5	Alberto Almeida da Silveira	16/01/2016	31	Analista Júnior	R\$2.850	Atual
6	Alexandre Correa Rodriguez	16/01/2016	47	Analista Pleno	R\$3.700	Atual
7	Alexsandro da Silva Costa	16/01/2016	55	Coordenador 1	R\$7.200	Atual
8	Alexsandro Pereira da Costa	16/01/2016	51	Estagiário 1	R\$1.600	Atual
9	Alice Silva Alves	16/01/2016	53	Analista Senior	R\$4.650	Atual
10	Aline de Andrade	27/02/2016	23	Coordenador 2	R\$8.100	Atual
11	Aline Garcia Lima	27/02/2016	22	Gerente 2	R\$12.320	Antigo
12	Aline Sá	27/02/2016	34	Analista Júnior	R\$2.850	Atual
13	Allan de Lira	27/02/2016	30	Analista Júnior	R\$2.850	Atual
14	Alon Bernardes de Almeida	27/02/2016	36	Analista Senior	R\$4.650	Atual
15	Alon Fiorenza Duque Pinho	27/02/2016	59	Gerente 1	R\$9.450	Atual
16	Alon Paes de Sousa Guedes	27/02/2016	59	Jovem Aprendiz	R\$800	Atual
17	Alon Palmeira	27/02/2016	42	Jovem Aprendiz	R\$800	Atual
18	Alvaro Ferreira	27/02/2016	24	Coordenador 2	R\$8.100	Atual
19	Amanda Andrade Gontijo	02/04/2016	43	Jovem Aprendiz	R\$800	Antigo
20	Amanda Botelho	02/04/2016	32	Gerente 1	R\$9.450	Atual
21	Amanda Bruno	02/04/2016	46	Gerente 3	R\$16.300	Atual
22	Amanda de Carvalho	02/04/2016	36	Coordenador 2	R\$8.100	Atual
23	Amanda de Souza Roberto	02/04/2016	34	Coordenador 1	R\$7.200	Atual
24	Amanda Figueiredo Peçanha	02/04/2016	60	Coordenador 1	R\$7.200	Atual
25	Amanda Procaci	02/04/2016	48	Estagiário 1	R\$1.600	Atual

Após executar o nosso código com o atalho F5, voltamos na planilha para ver o resultado.

Porém, vemos que nem todos os funcionários antigos foram excluídos. Isso quer dizer que a nossa macro está com algum problema que teremos que entender e corrigir.

	A	B	C	D	E	F	G
1	Funcionário	Data de Contratação	Idade	Cargo	Salário	Funcionário Antigo/Atual	
2	Adriana Passos	14/11/2015	40	Gerente 2	R\$12.320	Atual	
3	Adriane Chagas	14/11/2015	48	Estagiário 2	R\$2.000	Atual	
4	Adriane Pinheiro	14/11/2015	61	Coordenador 2	R\$8.100	Atual	
5	Adrielle Penna Forte	16/01/2016	46	Estagiário 2	R\$2.000	Antigo	
6	Alberto Almeida de Oliveira	15/01/2016	21	Analista 1	R\$2.000	Atual	

	A	B	C	D	E	F	G
1	Funcionário	Data de Contratação	Idade	Cargo	Salário	Funcionário Antigo/Atual	
2	Adriana Passos	14/11/2015	40	Gerente 2	R\$12.320	Atual	
3	Adriane Chagas	14/11/2015	48	Estagiário 2	R\$2.000	Atual	
4	Adriane Pinheiro	14/11/2015	61	Coordenador 2	R\$8.100	Atual	
5	Adrielle Penna Forte	16/01/2016	46	Estagiário 2	R\$2.000	Antigo	
6	Alberto Almeida de Oliveira	15/01/2016	21	Analista 1	R\$2.000	Atual	

	A	B	C	D	E	F	G
1	Funcionário	Data de Contratação	Idade	Cargo	Salário	Funcionário Antigo/Atual	
2	Adriana Passos	14/11/2015	40	Gerente 2	R\$12.320	Atual	
3	Adriane Chagas	14/11/2015	48	Estagiário 2	R\$2.000	Atual	
4	Adriane Pinheiro	14/11/2015	61	Coordenador 2	R\$8.100	Atual	
5	Adrielle Penna Forte	16/01/2016	46	Estagiário 2	R\$2.000	Antigo	

Vamos tentar entender porque tivemos problema na hora de excluir as linhas. O principal ponto aqui é o seguinte. A estrutura For faz com que o loop seja feito para cada linha da tabela. Assim que executamos os comandos dentro do For, a variável linha é atualizada (devido ao Next) e a próxima linha é analisada. Isso vai sendo feito linha a linha.

```
For linha = 2 To ult_linha
```

13	Alice Silva Alves	16/01/2016	53	Analista Senior	R\$4.650	Atual	
14	Aline de Andrade	27/02/2016	23	Coordenador 2	R\$8.100	Atual	
15	Aline de Oliveira e Mello	27/02/2016	61	Gerente 2	R\$12.320	Antigo	
16	Aline Garcia Lima	27/02/2016	22	Gerente 2	R\$12.320	Antigo	
17	Aline Morais	27/02/2016	61	Coordenador 1	R\$7.200	Antigo	
18	Aline Sá	27/02/2016	34	Analista Júnior	R\$2.850	Atual	

Assim, quando ele chega em uma linha de um funcionário antigo (por exemplo, Aline de Oliveira, linha = 15), o If será executado e a linha será excluída:

```
If Cells(linha, 6).Value = "Antigo" Then
    Rows(linha & ":" & linha).Delete
End If
```

13	Alice Silva Alves	16/01/2016	53	Analista Senior	R\$4.650	Atual	
14	Aline de Andrade	27/02/2016	23	Coordenador 2	R\$8.100	Atual	
15	Aline Garcia Lima	27/02/2016	22	Gerente 2	R\$12.320	Antigo	
16	Aline Morais	27/02/2016	61	Coordenador 1	R\$7.200	Antigo	
17	Aline Sá	27/02/2016	34	Analista Júnior	R\$2.850	Atual	
18	Allan de Lira	27/02/2016	30	Analista Júnior	R\$2.850	Atual	

Porém, quando ele exclui a linha daquele funcionário antigo, a linha de baixo é deslocada para cima antes do loop atualizar a variável linha. E ai quando o código executa o comando **Next**, a Aline Garcia, que subiu para a linha 15 por conta da Aline de Oliveira ter sido excluída, não é verificada pois a variável linha passa a ser igual a 16 antes de checar a linha da Aline.

13	Alice Silva Alves	16/01/2016	53	Analista Senior	R\$4.650	Atual	
14	Aline de Andrade	27/02/2016	23	Coordenador 2	R\$8.100	Atual	
15	Aline Garcia Lima	27/02/2016	22	Gerente 2	R\$12.320	Antigo	
16	Aline Morais	27/02/2016	61	Coordenador 1	R\$7.200	Antigo	
17	Aline Sá	27/02/2016	34	Analista Júnior	R\$2.850	Atual	
18	Allan de Lira	27/02/2016	30	Analista Júnior	R\$2.850	Atual	

```
Sub compila()
    ult_linha = Range("A1").End(xlDown).Row

    For linha = 2 To ult_linha
        If Cells(linha, 6).Value = "Antigo" Then
            Rows(linha & ":" & linha).Delete
            linha = linha - 1
        End If
    Next
End Sub
```

Para contornar isso, dentro do If, acrescentamos o código:

linha = linha - 1

Isso fará com que, no momento que uma linha seja excluída, este comando subtraia 1 da variável linha atual (no exemplo da página anterior, linha = 15 - 1 = 14. Assim, quando esta variável linha subtraída de 1 chegar no comando Next, no próximo loop, linha passa a ser 15 (em vez de 16) e consegue checar a linha que antes havia sido pulada.

Ao executar este código (atalho F5) as linhas serão corretamente excluídas.

## Módulo 3 – Estrutura e Objetos do VBA – Compilação: Excluindo linhas da tabela

135

	A	B	C	D	E	F
1	Funcionário	Data de Contratação	Idade	Cargo	Salário	Funcionário Antigo/Atual
2	Adriana Passos	14/11/2015	40	Gerente 2	R\$12.320	Atual
3	Adriane Chagas	14/11/2015	48	Estagiário 2	R\$2.000	Atual
4	Adriane Pinheiro	14/11/2015	61	Coordenador 2	R\$8.100	Atual
5	Alberto Almeida da Silveira	16/01/2016	31	Analista Júnior	R\$2.850	Atual
6	Alexandre Correa Rodriguez	16/01/2016	47	Analista Pleno	R\$3.700	Atual
7	Alexsandro da Silva Costa	16/01/2016	55	Coordenador 1	R\$7.200	Atual
8	Alexsandro Pereira da Costa	16/01/2016	51	Estagiário 1	R\$1.600	Atual
9	Alice Silva Alves	16/01/2016	53	Analista Senior	R\$4.650	Atual
10	Aline de Andrade	27/02/2016	23	Coordenador 2	R\$8.100	Atual
11	Aline Sá	27/02/2016	34	Analista Júnior	R\$2.850	Atual
12	Allan de Lira	27/02/2016	30	Analista Júnior	R\$2.850	Atual
13	Alon Bernardes de Almeida	27/02/2016	36	Analista Senior	R\$4.650	Atual
14	Alon Fiorenza Duque Pinho	27/02/2016	59	Gerente 1	R\$9.450	Atual
15	Alon Paes de Sousa Guedes	27/02/2016	59	Jovem Aprendiz	R\$800	Atual
16	Alon Palmeira	27/02/2016	42	Jovem Aprendiz	R\$800	Atual
17	Alvaro Ferreira	27/02/2016	24	Coordenador 2	R\$8.100	Atual
18	Amanda Botelho	02/04/2016	32	Gerente 1	R\$9.450	Atual
19	Amanda Bruno	02/04/2016	46	Gerente 3	R\$16.300	Atual
20	Amanda de Carvalho	02/04/2016	36	Coordenador 2	R\$8.100	Atual
21	Amanda de Souza Roberto	02/04/2016	34	Coordenador 1	R\$7.200	Atual
22	Amanda Figueiredo Peçanha	02/04/2016	60	Coordenador 1	R\$7.200	Atual
23	Amanda Procaci	02/04/2016	48	Estagiário 1	R\$1.600	Atual
24	Ana Alverne Leite	04/04/2016	50	Analista Júnior	R\$2.850	Atual
25	Ana Bandeira de Mello de Albuquerque Campos	04/04/2016	26	Gerente 3	R\$16.300	Atual

E o resultado final é mostrado na imagem ao lado.

	A	B	C	D	E	F	G	H	I	
1	SKU	Dia	Mês	Ano	Tipo	Marca	Custo de compra	Valor de venda	Lucro Bruto	
2	SKU 511	01/01/13	1	2013	Camisa	Reserva	R\$ 119,00	R\$ 252,28	R\$ 133,28	
3	SKU 199	02/01/13	1	2013	Camisa	Reserva	R\$ 117,00	R\$ 248,04	R\$ 131,04	
4	SKU 849	01/01/13	1	2013	Casaco	Hashtag	R\$ 32,00	R\$ 41,60	R\$ 9,60	
5	SKU 988	04/01/13	1	2013	Camiseta	Osklen	R\$ 44,00	R\$ 62,48	R\$ 18,48	
6	SKU 802	01/01/13	1	2013	Sandália	Reserva	R\$ 103,00	R\$ 218,36	R\$ 115,36	
7	SKU 63	06/01/13	1	2013	Boné	Hashtag	R\$ 64,00	R\$ 83,20	R\$ 19,20	
8	SKU 263	07/01/13	1	2013	Camiseta	Hashtag	R\$ 23,00	R\$ 29,90	R\$ 6,90	
9	SKU 207	08/01/13	1	2013	Meia	Osklen	R\$ 63,00	R\$ 89,46	R\$ 26,46	
10	SKU 830	09/01/13	1	2013	Boné	Osklen	R\$ 87,00	R\$ 123,54	R\$ 36,54	
11	SKU 755	10/01/13	1	2013	Casaco	Reserva	R\$ 108,00	R\$ 228,96	R\$ 120,96	
12	SKU 248	11/01/13	1	2013	Meia	Hashtag	R\$ 36,00	R\$ 46,80	R\$ 10,80	
13	SKU 205	12/01/13	1	2013	Bota	Osklen	R\$ 98,00	R\$ 139,16	R\$ 41,16	
14	SKU 592	13/01/13	1	2013	Camisa	Reserva	R\$ 109,00	R\$ 231,08	R\$ 122,08	
15	SKU 222	14/01/13	1	2013	Chinelo	Osklen	R\$ 95,00	R\$ 134,90	R\$ 39,90	
16	SKU 637	14/01/13	1	2013	Boné	Osklen	R\$ 60,00	R\$ 85,20	R\$ 25,20	
17	SKU 924	14/01/13	1	2013	Casaco	Reserva	R\$ 87,00	R\$ 184,44	R\$ 97,44	
18	SKU 215	14/01/13	1	2013	Casaco	Osklen	R\$ 90,00	R\$ 127,80	R\$ 37,80	
19	SKU 67	14/01/13	1	2013	Camisa	Hashtag	R\$ 21,00	R\$ 27,30	R\$ 6,30	
20	SKU 546	14/01/13	1	2013	Bota	Osklen	R\$ 67,00	R\$ 95,14	R\$ 28,14	
21	SKU 849	14/01/13	1	2013	Meia	Reserva	R\$ 83,00	R\$ 175,96	R\$ 92,96	
22	SKU 246	14/01/13	1	2013	Boné	Osklen	R\$ 75,00	R\$ 106,50	R\$ 31,50	
23	SKU 75	14/01/13	1	2013	Meia	Osklen	R\$ 58,00	R\$ 82,36	R\$ 24,36	
24	SKU 968	14/01/13	1	2013	Meia	Reserva	R\$ 104,00	R\$ 220,48	R\$ 116,48	
25	SKU 93	14/01/13	1	2013	Camisa	Hashtag	R\$ 39,00	R\$ 50,70	R\$ 11,70	

Base

Compilação



Vamos fazer agora um desafio. Na próxima planilha temos uma aba “Base” com uma tabela contendo várias linhas, cada uma com a informação de venda de um determinado SKU (SKU nada mais é que um código para um produto).

O nosso objetivo final é construir uma macro que seja capaz de, na aba “Compilação”, atualizar todos os SKUs de acordo com a Marca e Ano selecionados nas células A2 e B2, respectivamente, toda vez que a gente clicar no botão de compilar.

A	B	C	D	E	F	G	H
1	Marca	Ano	SKU				
2	Reserva	2013					
3							
4							
5							
6							
7							
8							

Compilar SKUs

```
Sub compilacao()
    Sheets("Compilação").Activate
    marca = Cells(2, 1).Value
    ano = Cells(2, 2).Value
    Sheets("Base").Activate
    ult_linha = Range("A1").End(xlDown).Row
    For linha = 2 To ult_linha
        |
    Next
End Sub
```

Vamos abrir o ambiente VBA (ALT + F11) e começar o nosso código.

Toda vez que construímos um código, devemos pensar exatamente no passo a passo que queremos executar. Em primeiro lugar é importante lembrar que o compilado deve ser feito de acordo com ano e marca selecionados. Como vamos precisar dessas duas informações no nosso código, o primeiro passo é armazenar essas duas informações em 2 variáveis: ano e marca. Só que, antes de armazenar esses valores, para o código saber que estamos pegando uma informação da aba “Compilação”, na primeira linha de código usamos o comando [Sheets\("Compilação"\).Activate](#), para ativar essa aba.

Em seguida, para começar a percorrer cada linha da tabela da aba “Base”, selecionamos essa aba com o comando [Sheets\("Base"\).Activate](#).

Neste ponto, devemos ter em mente que precisamos testar, para cada linha da tabela, se ano e marca correspondem aos valores selecionados na aba “Compilação”. Por isso, precisamos de uma estrutura de repetição. Vamos usar o For.

O For deve ser executado desde a primeira linha contendo informação (linha 2) até a última linha da tabela, que descobrimos através da variável [ult\\_linha](#).

```
Sub compilacao()
    Sheets("Compilação").Activate
    marca = Cells(2, 1).Value
    ano = Cells(2, 2).Value
    Sheets("Base").Activate
    ult_linha = Range("A1").End(xlDown).Row
    For linha = 2 To ult_linha
        If Cells(linha, 4).Value = ano And Cells(linha, 6).Value = marca Then
            codigo = Cells(linha, 1).Value
            Sheets("Compilação").Activate
        End If
    Next
End Sub
```

Dentro da estrutura de repetição For vamos precisar utilizar uma estrutura para testar se ano = ano selecionado e também se marca = marca selecionada. Para isso usamos um If com um comando And. Isto porque devemos testar 2 condições: tanto o ano (que está na coluna 4) quanto a marca (que está na coluna 6).

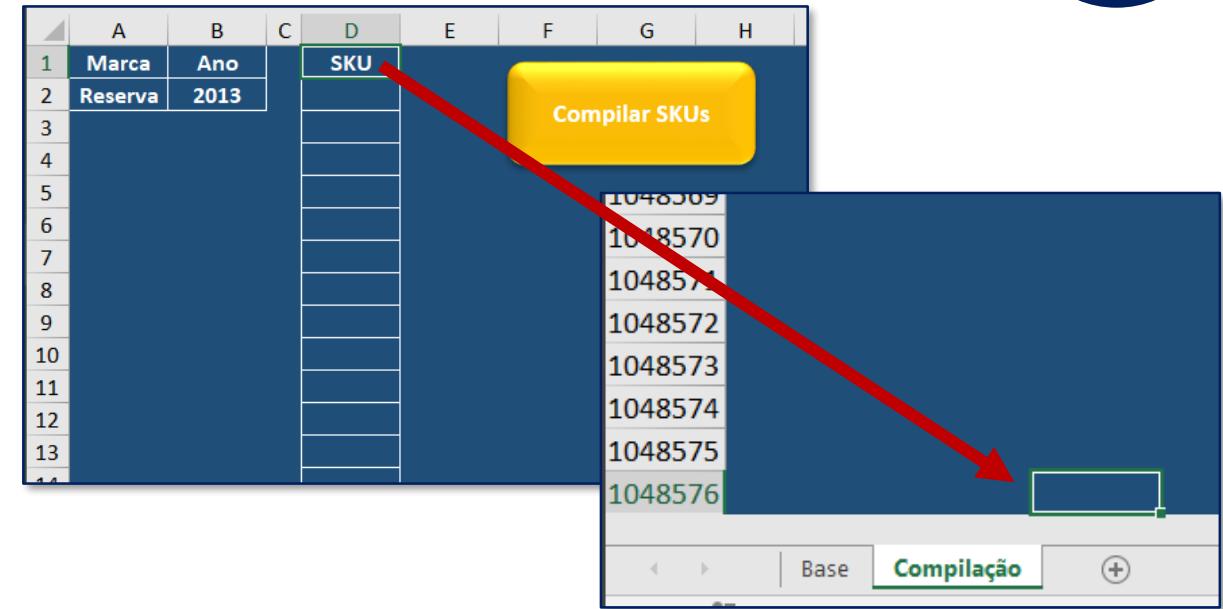
Feito isso, se ano e marca correspondem aos valores selecionados, vamos armazenar a informação de SKU em uma variável chamada **codigo**.

Em seguida, precisamos escrever este código na aba “Compilação”. Portanto, devemos utilizar o comando **Sheets("Compilação").Activate** para selecionar a aba antes de escrever o código na coluna D.

Continuando o nosso código, devemos agora registrar o SKU na coluna D. Porém, antes de mais nada precisamos descobrir qual é a última célula preenchida na coluna D da aba “Compilação”.

O problema é que aqui, se usarmos o comando de descobrir a última linha: Range("D1").End(xlDown).Row, como não tem nada escrito ainda na coluna D abaixo da linha 1, este comando irá selecionar a última linha do Excel (1.048.576, vide imagem ao lado), pois ele nunca vai encontrar um valor preenchido nessa coluna e vai até o final dela (se você está com dúvida em relação a esse comando, volte na página 118 onde explicamos sobre ele).

Para evitar esse problema, em vez de dar um CTRL + Seta para baixo a partir da célula D1, podemos dar um CTRL + Seta para cima a partir de uma célula bem lá embaixo na coluna D, por exemplo, a célula D1000. Assim, ele vai subir até o começo da planilha e de fato selecionar a última célula preenchida naquela coluna.



Porém, não queremos escrever na última linha preenchida, e sim exatamente uma linha abaixo desta. Assim, usamos o seguinte comando:

Range("D1000").End(xlUp).Row + 1

**Com xlUp porque queremos dar um CTRL + Seta para cima a partir de D1000.**

```
Sub compilacao()
    Sheets("Compilação").Activate
    marca = Cells(2, 1).Value
    ano = Cells(2, 2).Value
    Sheets("Base").Activate
    ult_linha = Range("A1").End(xlDown).Row
    For linha = 2 To ult_linha
        If Cells(linha, 4).Value = ano And Cells(linha, 6).Value = marca Then
            codigo = Cells(linha, 1).Value
            Sheets("Compilação").Activate
            linha_registro = Range("D1000").End(xlUp).Row + 1
            Cells(linha_registro, 4).Value = codigo
            Sheets("Base").Activate
        End If
    Next
End Sub
```

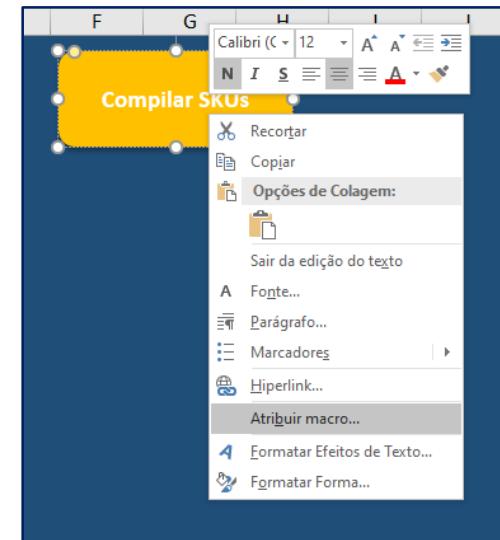


**Não esquecer de voltar para a aba “Base” antes de sair do If, pois precisamos continuar percorrendo as linhas da tabela da aba “Base”!**

Para registrar a linha seguinte à última linha preenchida, criamos a variável **linha\_registro** e em seguida escrevemos na coluna D o código armazenado na variável **codigo**.

Feito isso, não podemos esquecer de voltar para a aba “Base” antes de passar para a próxima linha da tabela da aba “Base”.

O código final é mostrado ao lado. Se você quiser, você pode atribuir a macro “compilacao” ao botão “Compilar”.



```
Sub compilacao()
    Sheets("Compilação").Activate
    Range("D2:D100000").ClearContents ←
    marca = Cells(2, 1).Value
    ano = Cells(2, 2).Value
    Sheets("Base").Activate
    ult_linha = Range("A1").End(xlDown).Row
    For linha = 2 To ult_linha
        If Cells(linha, 4).Value = ano And Cells(linha, 6).Value = marca Then
            codigo = Cells(linha, 1).Value
            Sheets("Compilação").Activate
            linha_registro = Range("D1000").End(xlUp).Row + 1
            Cells(linha_registro, 4).Value = codigo
            Sheets("Base").Activate
        End If
    Next
    Sheets("Compilação").Activate| ←
End Sub
```

Para facilitar, apagamos desde a linha 2 da coluna D (pois não podemos apagar a coluna toda por conta do título) até uma célula bem lá embaixo na planilha.

Antes de finalizar o código, podemos fazer mais duas coisas para deixar a macro 100% otimizada.

O código até a página anterior sempre terminava selecionando a aba “Base” pois é o último comando de seleção de aba realizado. Só que na verdade queremos terminar a macro sempre com a aba “Compilação” selecionada, pois é nela que temos o resumo do código. Assim, o que devemos fazer imediatamente antes de finalizar o código é selecionar a aba “Compilação” através do comando: **Sheets("Compilação").Activate**.

Em segundo lugar, sempre que a gente rodar a macro para um novo ano/marca, devemos apagar os valores das células da coluna D para escrever os novos valores. Para apagar os SKUs anteriormente escritos antes de escrever os novos, logo após o comando de selecionar a aba “Compilação” no início do código, usamos o comando **.ClearContents** para apagar os valores antes de executar a macro.

O próximo exercício pode ser considerado um dos mais desafiadores até agora. A ideia é, mais uma vez, juntar os principais conceitos vistos até aqui para criar uma ferramenta bem avançada. As duas abas principais da nossa planilha estão mostradas abaixo.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1 Grife	Reserva														
2 Status	Coleção Antiga														
3 Estoque Mínimo	3500														
4															
5															
6															
7															
8															
9															
10															
11															
12															
13															

**Compilar Grifes**

Cor	Produto 1	Produto 2	Produto 3	Produto 4	Produto 5	Produto 6	Produto 7	Produto 8	Produto 9	Produto 10	Estoque
AMARELO											
BRANCO											
AZUL											
ROSA											
VERDE ESMERALDA											
VERMELHO											

Enunciado | Base | **Produtos**

A	B	C	D	E	F	G	H	I
1	Código	Grupo	Cor	Grife	Coleção	Estoque	Status	Grifes
2	02.020.235	Bota	ROSA	Osklen	13	498	Coleção Antiga	Reserva
3	02.020.242	Camiseta	AZUL	Hashtag	14	1540	Nova Coleção	Hashtag
4	02.020.242	Chinelo	BRANCO	Hashtag	02	546	Coleção Antiga	Osklen
5	02.020.250	Chinelo	AMARELO	Hashtag	03	451	Coleção Antiga	
6	02.030.232	Chinelo	BRANCO	Osklen	08	1941	Nova Coleção	
7	02.030.236	Sandália	BRANCO	Osklen	02	4000	Nova Coleção	
8	02.030.240	Bota	VERMELHO	Reserva	15	1778	Coleção Antiga	
9	02.030.245	Sandália	AMARELO	Reserva	28	37	Coleção Antiga	
10	02.030.249	Boné	BRANCO	Osklen	04			
11	02.040.236	Sandália	VERMELHO	Reserva	08			

Enunciado | **Base** | Produtos

Temos basicamente 3 objetivos a cumprir na aba “Produtos”.

- 1- Após o usuário selecionar a Grife e o Status desejado, a macro deverá analisar todas as linhas da tabela “Base” e todos os códigos correspondentes à Grife e Status selecionados na aba “Produtos” devem ser registrados na tabela resumo.
- 2- Além disso, na coluna O, para cada Cor, deve ser registrado o estoque total de cada produto, de acordo com a coluna F de Estoque da aba “Base”.
- 3- Por fim, caso o estoque de uma determinada cor esteja abaixo do estoque mínimo estipulado na célula B3 da aba “Produtos”, toda a linha para aquela cor deve ser pintada de vermelho.

O resultado final é mostrado abaixo. Todos os códigos dos produtos foram registrados, o total de Estoque também foi registrado na coluna O e, por fim, todos aqueles produtos com estoque abaixo do estoque mínimo (no exemplo, 3500) ficaram com a cor da linha em vermelho.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Grife	Reserva												
2	Status	Coleção Antiga												
3	Estoque Mínimo	3500												
4														
5														
6	Cor	Produto 1	Produto 2	Produto 3	Produto 4	Produto 5	Produto 6	Produto 7	Produto 8	Produto 9	Produto 10	Estoque		
7	AMARELO	02.030.245	02.060.230	02.090.243	02.110.233	02.150.237	04.070.250	04.130.230	06.040.237					5803
8	BRANCO	02.050.230	03.050.247	03.120.234	05.090.232	06.030.238	06.060.250							6094
9	AZUL	05.100.242	06.140.239											1272
10	ROSA	02.080.246	02.150.244											513
11	VERDE ESMERALDA	04.020.236	06.030.239											3461
12	VERMELHO	02.030.240	02.040.236	02.100.236	03.050.231									4773



O código com a solução final completa é mostrado nas páginas 163 e 164. Até lá, toda a sua construção será detalhada passo a passo, mas caso você queira pular até lá (ou consultar durante a construção ao longo da explicação) é só avançar as páginas.

```

Sub compilacao()
    Sheets("Produtos").Activate
    grife = Range("B1").Value
    Status = Range("B2").Value
    cor = Cells(6, 4).Value
    Sheets("Base").Activate
    For linha = 2 To 149
        If Cells(linha, 3).Value = cor And Cells(linha, 4).Value = grife And Cells(linha, 7).Value = Status Then
            codigo = Cells(linha, 1).Value
            Sheets("Produtos").Activate
            coluna = Range("O6").End(xlToLeft).Column + 1
            Cells(6, coluna).Value = codigo
            Sheets("Base").Activate
        End If
    Next
End Sub

```

A	B	C	D	E	F	G	H	I
1	Código	Grupo	Cor	Grife	Coleção	Estoque	Status	Grifes
2	02.020.235	Bota	ROSA	Osklen	13	498	Coleção Antiga	Reserva
3	02.020.242	Camiseta	AZUL	Hashtag	14	1540	Nova Coleção	Hashtag
4	02.020.242	Chinelo	BRANCO	Hashtag	02	546	Coleção Antiga	Osklen
5	02.020.250	Chinelo	AMARELO	Hashtag	03	451	Coleção Antiga	
6	02.030.232	Chinelo	BRANCO	Osklen	08	1941	Nova Coleção	
7	02.030.236	Sandália	BRANCO	Osklen	02	4000	Nova Coleção	
8	02.030.240	Bota	VERMELHO	Reserva	15	1778	Coleção Antiga	
9	02.030.245	Sandália	AMARELO	Reserva	28	37	Coleção Antiga	
10	02.030.249	Boné	BRANCO	Osklen	04			
11	02.040.236	Sandália	VERMELHO	Reserva	08			

Enunciado

Base

Produtos

Para facilitar o entendimento, vamos pensar no básico: como conseguiríamos preencher os códigos apenas para a linha 6, referente à cor AMARELO?

O código final é mostrado ao lado, e a lógica é muito parecida com o exercício anterior. Primeiro começamos armazenando os valores de grife, status e cor em 3 variáveis. A variável cor será fixa na célula D6.

Em seguida, selecionamos a aba “Base” e percorremos cada linha da tabela, que começa na linha 2 e vai até a linha 149.

Será necessário testar os valores das colunas C (cor), D (Grife) e G (status). Por isso a estrutura If junto do And.

Sempre que todas essas condições forem atendidas, armazenamos o código na variável código.

```

Sub compilacao()
    Sheets("Produtos").Activate
    grife = Range("B1").Value
    Status = Range("B2").Value
    cor = Cells(6, 4).Value
    Sheets("Base").Activate
    For linha = 2 To 149
        If Cells(linha, 3).Value = cor And Cells(linha, 4).Value = grife And Cells(linha, 7).Value = Status Then
            codigo = Cells(linha, 1).Value
            Sheets("Produtos").Activate
            coluna = Range("O6").End(xlToLeft).Column + 1
            Cells(6, coluna).Value = codigo
            Sheets("Base").Activate
        End If
    Next
End Sub

```

Este comando irá preencher um novo código sempre na última coluna vazia.



Após armazenar o código, devemos voltar para a aba “Produtos” e em seguida registrar o código na coluna certa. Quando o primeiro código for preenchido, ele será preenchido na coluna E. Depois, na F. Depois, na G, e por ai vai. Para preencher sempre na coluna certa (a primeira que estiver vazia logo após a última coluna preenchida), usamos a mesma lógica do comando .End a partir da célula O6, dando um Ctrl + Seta para direita. E ai, para preencher na coluna seguinte, somamos 1 à variável **coluna**.

Em seguida, voltamos para a aba “Base” para continuar o nosso Loop para testar a próxima linha.

Ao executar a macro, o resultado apenas para a linha 6 será o da imagem ao lado. Não esqueça de atribuir esta macro ao botão de “Compilar”.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1 Grife	Reserva													
2 Status	Coleção Antiga													
3 Estoque Mínimo	1200													
4														
5														
6														
7														
8														
9														
10														
11														
12														

```
Sub compilacao()
    Sheets("Produtos").Activate
    grife = Range("B1").Value
    Status = Range("B2").Value
    cor = Cells(6, 4).Value
    Sheets("Base").Activate
    For linha = 2 To 149
        If Cells(linha, 3).Value = cor And Cells(linha, 4).Value = grife And Cells(linha, 7).Value = Status Then
            codigo = Cells(linha, 1).Value
            Sheets("Produtos").Activate
            coluna = Range("O6").End(xlToLeft).Column + 1
            Cells(6, coluna).Value = codigo
            Sheets("Base").Activate
        End If
    Next
End Sub
```

Para que você possa entender 100% a ideia desta macro, a sugestão é que você execute linha por linha utilizando o comando F8 de depuração, como já explicamos anteriormente (página 71).

Assim, você pode ver, passo a passo, como o código está funcionando e entender o que cada comando está fazendo.

```
Sub compilacao()
    Sheets("Produtos").Activate
    grife = Range("B1").Value
    Status = Range("B2").Value
    For linha_registro = 6 To 11 1
        cor = Cells(linha_registro, 4).Value 2
        Sheets("Base").Activate
        For linha = 2 To 149
            If Cells(linha, 3).Value = cor And Cells(linha, 4).Value = grife And Cells(linha, 7).Value = Status Then
                codigo = Cells(linha, 1).Value
                Sheets("Produtos").Activate
                coluna = Range("O" & linha_registro).End(xlToLeft).Column + 1 3
                Cells(linha_registro, coluna).Value = codigo
                Sheets("Base").Activate
            End If
        Next
        Sheets("Produtos").Activate 4
    Next
End Sub
```

Dando mais um passo em direção à solução final, vamos tornar a macro automática para que seja possível preencher os códigos não só para a cor AMARELO da linha 6 mas para todas as cores, da linha 6 até a linha 11.

Para isso, será necessário criar uma nova estrutura de repetição For para, antes de percorrer cada linha da tabela da aba “Base”, percorrer cada linha da tabela resumo da aba “Produtos”.

O código final é mostrado ao lado, e a explicação das 4 modificações criadas está nas páginas seguintes.

```
Sub compilacao()

Sheets("Produtos").Activate

grife = Range("B1").Value
Status = Range("B2").Value

For linha_registro = 6 To 11
    1

    cod = Cells(linha_registro, 4).Value
    Sheets("Base").Activate

    For linha = 2 To 149
        If Cells(linha, 3).Value = cod And Cells(linha, 4).Value = grife And
            Cells(linha, 5).Value = Status Then
            codigo = Cells(linha, 1).Value
            Sheets("Produtos").Activate
            celula = Range("D" & (linha_registro).ToString() & "E" & linha + 1)
            Cells(linha_registro, celula).Value = codigo
            Sheets("Base").Activate
        End If
    Next
Next
```

**! Não esqueça de fechar a nova estrutura For com o Next!**

Em primeiro lugar, criamos um novo loop principal que irá percorrer cada linha da tabela de resumo dos códigos na aba “Produtos”.

Como já criamos uma variável linha, para não ter conflito, criamos uma nova chamada **linha\_registro**, que vai percorrer desde a linha 6 (cor AMARELO) até linha 11 (cor VERMELHO).

Não esqueça de fechar o Next deste for ao final do código criado na parte 1.



**Não esqueça de fechar a nova estrutura For com o Next!**

```
Sub compilar()
    Dim cor As String
    Dim cel As Range
    Dim grife As Range
    Dim linha_registro As Integer
    Dim linha As Integer
    Dim coluna As Integer

    linha_registro = 6 'linha
    cor = Cells(linha_registro, 4).Value

    cel = Range("B6:D6").EntireRow
    cel.Font.Color = cor

    linha = 6 'linha
    coluna = 3 'coluna

    Do While cel(1, coluna).Value = cor And cel(1, coluna + 1).Value = grife And
        cel(1, coluna + 2).Value = cel(1, coluna).Value
        cel(1, coluna).Font.Color = cor
        cel(1, coluna + 1).Font.Color = grife
        cel(1, coluna + 2).Font.Color = cel(1, coluna).Font.Color
        cel = cel.Offset(1, 0)
    Loop
End Sub
```

2

Em seguida, não podemos mais usar o código abaixo para a variável cor, pois ela não é mais fixa na célula D6.

```
cor = Cells(6, 4).Value
```

O que fazemos agora é tornar automático o valor da linha que antes era 6. Para isso, é só trocar o 6 pela variável **linha\_registro** que será responsável pelo novo loop na tabela de registro.

```
cor = Cells(linha_registro, 4).Value
```

```

Sub compilar()
Sheets("Resumo").Activate
grife = Range("B1").Value
fimres = Range("B2").Value

For linha_registro = 6 To 15
    res = celula(linha_registro, 8).Value
    Sheets("Resumo").Activate
    For linha = 3 To 149
        If celula(linha, 3).Value = res And celula(linha, 4).Value = grife And
            celula(linha, 5).Value = celula(linha, 3).Value Then
            celula = Range("O" & linha_registro).End(xlToLeft).Column + 1
            celula(linha, 11).Value = celula(linha, 10).Value * celula(linha, 11)
    Next
    Sheets("Resumo").Activate
    End If
Next
End Sub

```

Outra parte que deve ser adaptada é a de encontrar a última coluna preenchida. Como agora vamos percorrer da linha 6 até a linha 11, não podemos mais ter a antiga variável coluna com o Range("O6") fixo:

```
coluna = Range("O6").End(xlToLeft).Column + 1
```

Devemos atualizar a linha 6 dentro do comando Range. Para isso, concatenamos o texto “0” referente à coluna com a variável **linha\_registro**:

(3)

```
coluna = Range("O" & linha_registro).End(xlToLeft).Column + 1
```

```

Sheets("Resumo").Activate

grife = Range("B1").Value
tintura = Range("B2").Value

For Linha_preguntas = 6 To 15
    var = celle(Linha_preguntas, 4).Value
    Sheets("Base").Activate

    For Linha = 2 To 149
        If celle(Linha, 3).Value = var And celle(Linha, 4).Value = grife And
            celle(Linha, 13).Value = 0 Then
            celle(Linha, 13).Value = var
            celle(Linha_preguntas, 13).Value = var
            celle(Linha_preguntas, 14).Value = 1
        End If
    Next Linha
    Sheets("Base").Activate
Next Linha_preguntas
End Sub

```

Por fim, não podemos esquecer de selecionar a aba “Produtos” antes de voltar para o Loop principal, pois devemos selecionar a próxima cor que está na coluna D da aba “Produtos”.

Agora que passamos por todas as modificações, você pode voltar para a página 147 caso queira visualizar o código completo novamente.

Você já pode rodar a macro que agora ela irá executar todos os comandos para todas as cores da tabela de registro, escrevendo todos os códigos de cada uma das cores da coluna D.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1 Grife	Hashtag														
2 Status	Nova Coleção														
3 Estoque Mínimo	1200														
4															
5															
6															
7															
8															
9															
10															
11															
12															
13															
14															
15															
16															

**Compilar Grifes**

Cor	Produto 1	Produto 2	Produto 3	Produto 4	Produto 5	Produto 6	Produto 7	Produto 8	Produto 9	Produto 10	Estoque
AMARELO	02.070.235	02.140.243	03.070.244	03.130.230	05.050.247	07.030.237					
BRANCO	03.060.250	03.140.239	06.020.249	07.030.236							
AZUL	02.020.242	02.070.230	03.120.234	04.030.234	04.090.230	05.120.233	06.130.238				
ROSA	02.050.231	04.080.247	06.060.240	06.100.249							
VERDE ESMERALDA	03.140.240	04.060.250	04.150.236	05.020.239	06.150.232						
VERMELHO	02.050.246	03.070.242	03.110.241	05.030.239	05.150.235	05.150.236	06.070.240	07.030.239			

 Até o presente momento, como ainda não fizemos esta automatização você deverá sempre apagar os códigos da tabela antes de executar a macro novamente, se não ele irá preenchendo novos códigos repetidamente. Então se o resultado final nesta nova rodada não estiver igualzinho a esse print, é porque você não apagou os códigos anteriores antes de executar a macro. Mais para frente veremos como automatizar isso para não precisar apagar manualmente os códigos toda vez!

```
Sub compilacao()
    Sheets("Produtos").Activate
    grife = Range("B1").Value
    Status = Range("B2").Value
    For linha_registro = 6 To 11
        estoque = 0 ←
        cor = Cells(linha_registro, 4).Value
        Sheets("Base").Activate
        For linha = 2 To 149
```

Agora vamos incluir o cálculo do estoque, a ser preenchido na coluna O da aba “Produtos”.

Primeiramente, devemos declarar a nossa variável **estoque**.

Como ela deve ser atualizada a cada linha da tabela de registro, começamos declarando logo que começa o Loop principal e atribuindo o valor zero. Isso porque ao longo do segundo Loop (que irá percorrer a aba “Base”) esta variável irá somar o total de estoque para a cor que estiver sendo analisada no momento, no começo, linha\_registro = 6 (cor AMARELO). Só que, assim que a gente passar para a linha\_registro = 7 (cor BRANCO), esta variável estoque deverá ser zerada para somar o estoque da nova cor.

```
Sub compilacao()
    Sheets("Produtos").Activate
    grife = Range("B1").Value
    Status = Range("B2").Value
    For linha_registro = 6 To 11
        estoque = 0
        cor = Cells(linha_registro, 4).Value
        Sheets("Base").Activate
        For linha = 2 To 149
            If Cells(linha, 3).Value = cor And Cells(linha, 4).Value =
                codigo = Cells(linha, 1).Value
                estoque = estoque + Cells(linha, 6).Value| ←
            Sheets("Produtos").Activate
        End If
    Next linha
    ' Aqui vai o resultado
End Sub
```

Começamos a somar o valor de estoque na linha de código indicada ao lado. Lembra da estrutura linha = linha + 1 que utilizamos nas estruturas Do Until e Do While para atualizar a variável linha a cada loop?

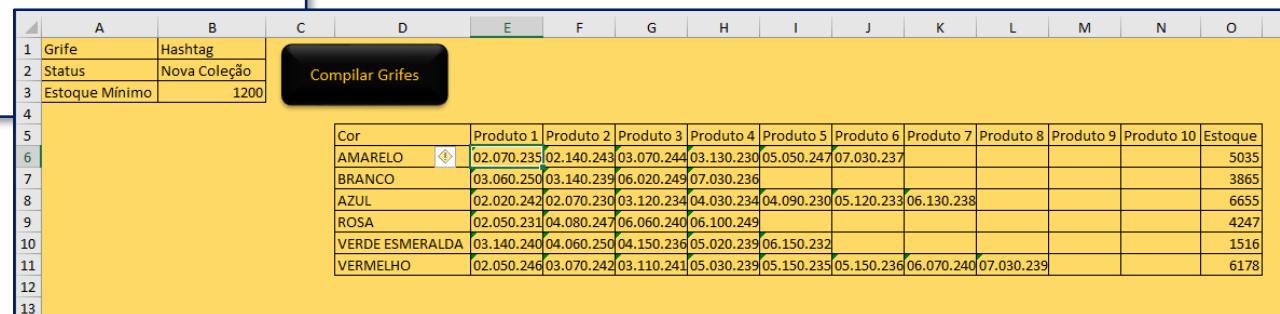
A ideia aqui é parecida. A variável **estoque** começa com o valor igual a zero e a cada loop na tabela da aba “Base” é somado o valor de estoque que está na coluna F desta aba.

Assim, após percorrer todas as linhas da tabela, desde a linha 2 até a linha 149, a variável estoque irá acumular o valor total, de acordo com a cor, grife e status que é testada por meio da estrutura If + And.

```

Sub compilacao()
    Sheets("Produtos").Activate
    grife = Range("B1").Value
    Status = Range("B2").Value
    For linha_registro = 6 To 11
        estoque = 0
        cor = Cells(linha_registro, 4).Value
        Sheets("Base").Activate
        For linha = 2 To 149
            If Cells(linha, 3).Value = cor And Cells(linha, 4).Value = grife And Cells(linha, 6).Value >= Status Then
                codigo = Cells(linha, 1).Value
                estoque = estoque + Cells(linha, 6).Value
            End If
        Next
        Cells(linha_registro, 15).Value = estoque 'registra na coluna O
    Sheets("Base").Activate
End Sub

```



The screenshot shows a Microsoft Excel spreadsheet with two tabs: 'Produtos' and 'Base'. The 'Produtos' tab contains three rows of data: 'Grife' (AMARELO), 'Hashtag' (Nova Coleção), and 'Status' (1200). The 'Base' tab contains a large table with columns for 'Cor' (Color) and various product codes ('Produto 1' through 'Produto 10') along with an 'Estoque' (Stock) column. The 'Estoque' column shows values such as 5035, 3865, 6655, 4247, 1516, and 6178. A button labeled 'Compilar Grifes' is visible on the ribbon.

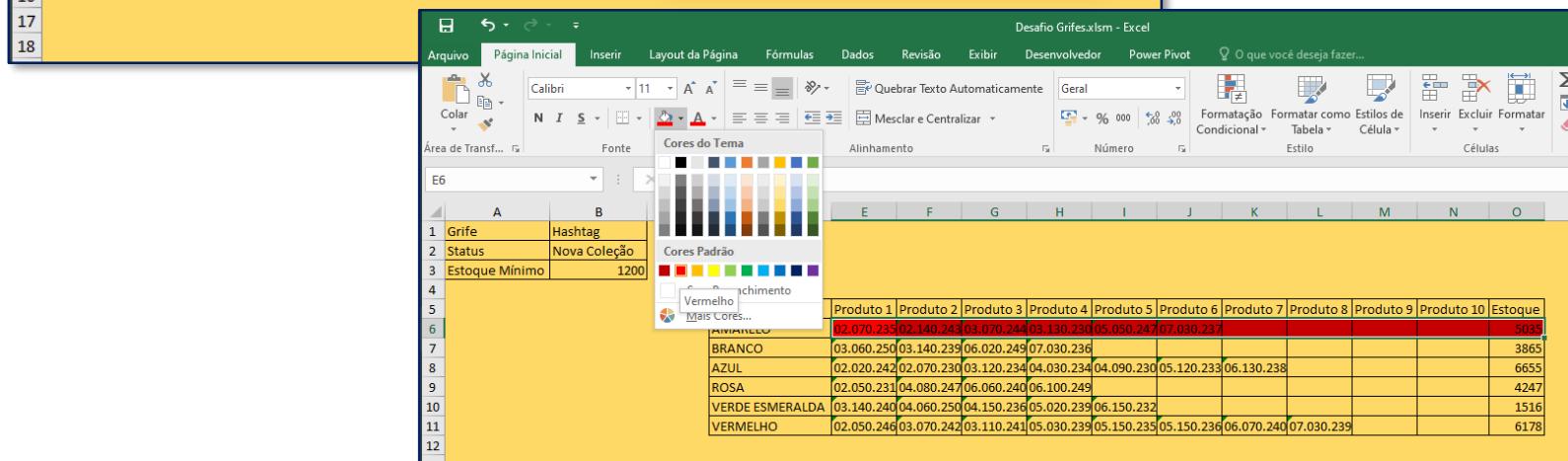
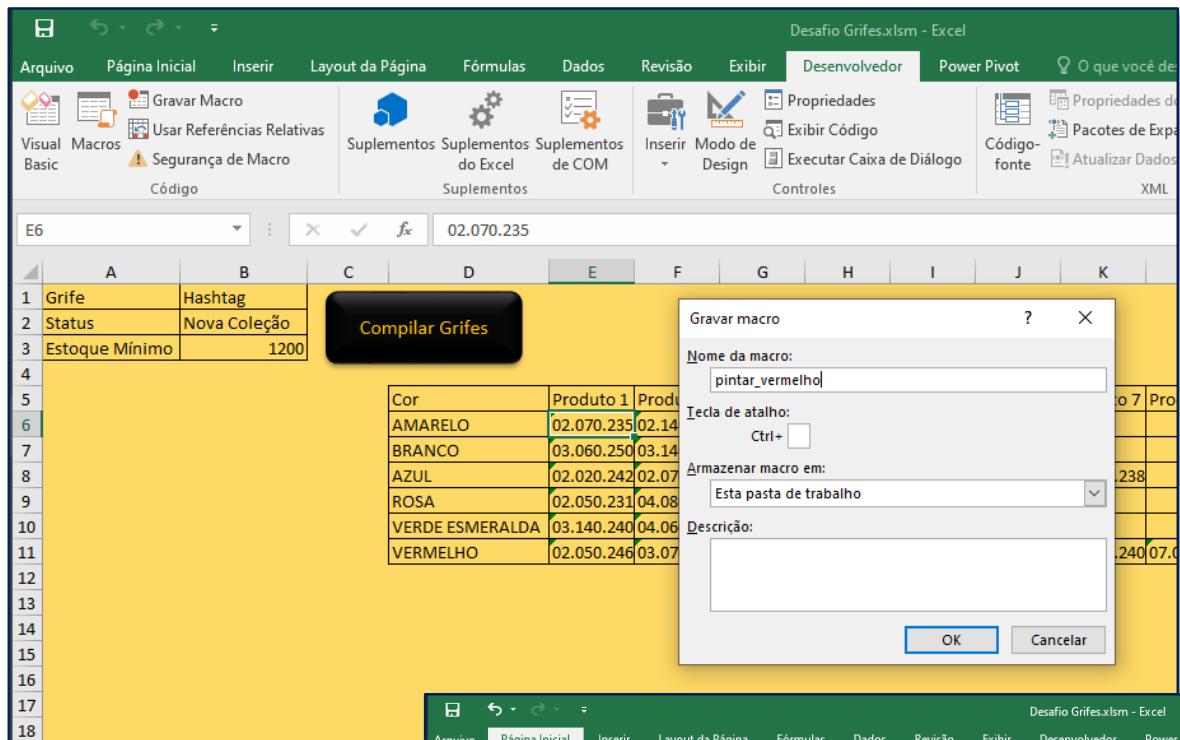
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Grife	Hashtag													
2	Status	Nova Coleção													
3	Estoque Mínimo	1200													
4															
5															
6															
7															
8															
9															
10															
11															
12															
13															

Em seguida, essa variável estoque deve ser escrita em cada linha da coluna O na aba “Produtos”. O código completo até o momento é mostrado na imagem ao lado.

Agora, ao rodar a macro, já conseguimos incluir o cálculo do estoque na coluna O da tabela.

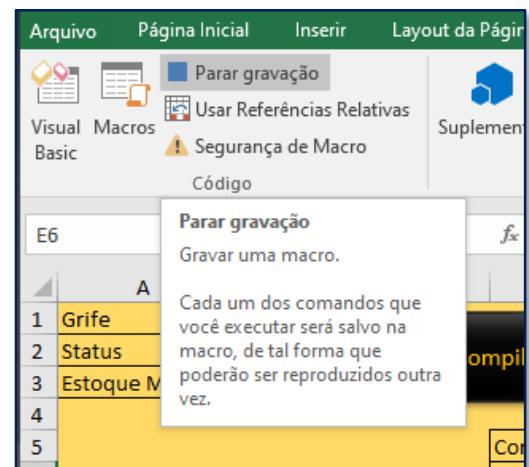
# Módulo 3 – Estrutura e Objetos do VBA – Desafio Compilação Grifes (Parte 4)

156



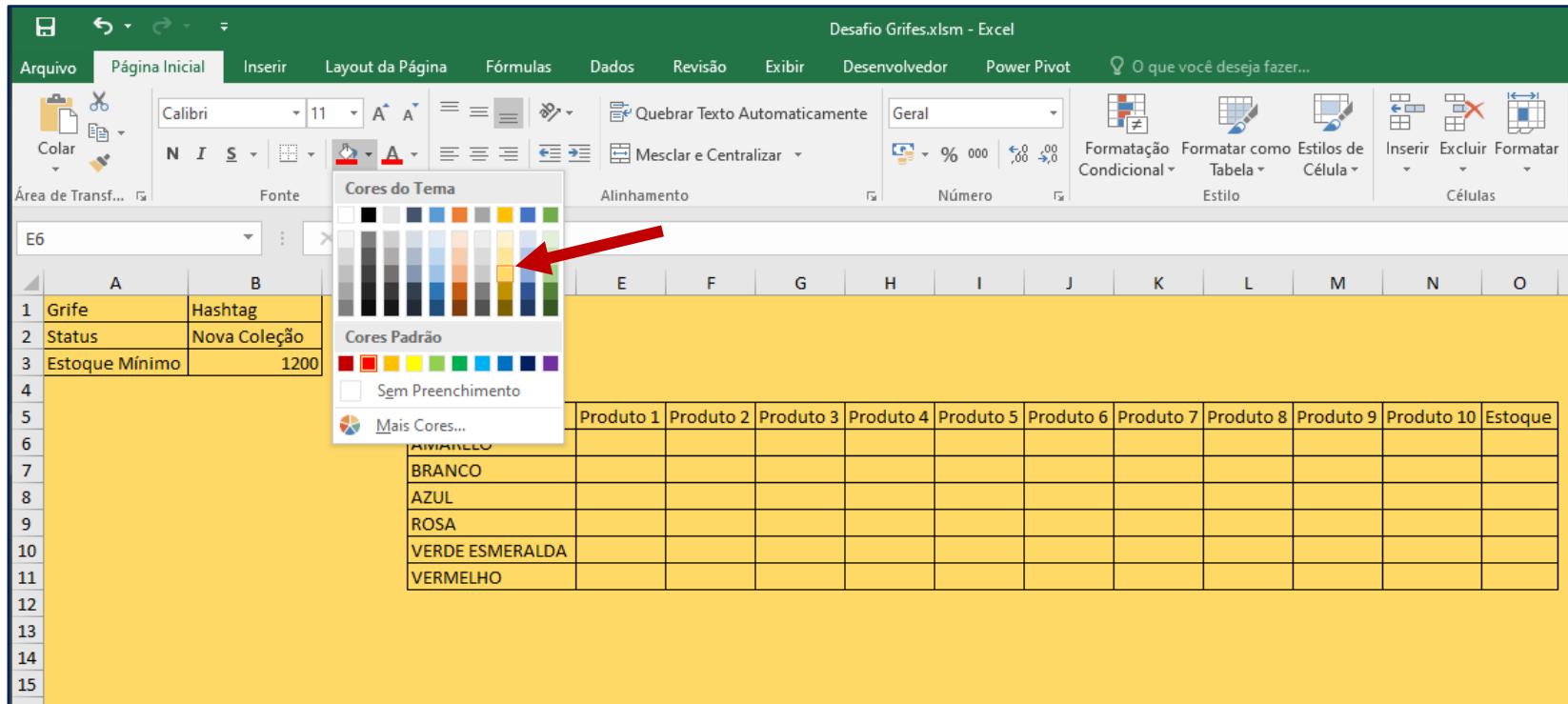
O próximo passo agora é descobrir como pintar de vermelho as linhas com estoque abaixo do estoque mínimo da célula B3. Para isso, vamos gravar uma macro chamada **pintar\_vermelho** simplesmente pintando a primeira linha da tabela de vermelho, como é indicado nas figuras ao lado.

Feito isso, não esqueça de clicar no botão de Parar Gravação na guia **Desenvolvedor**.



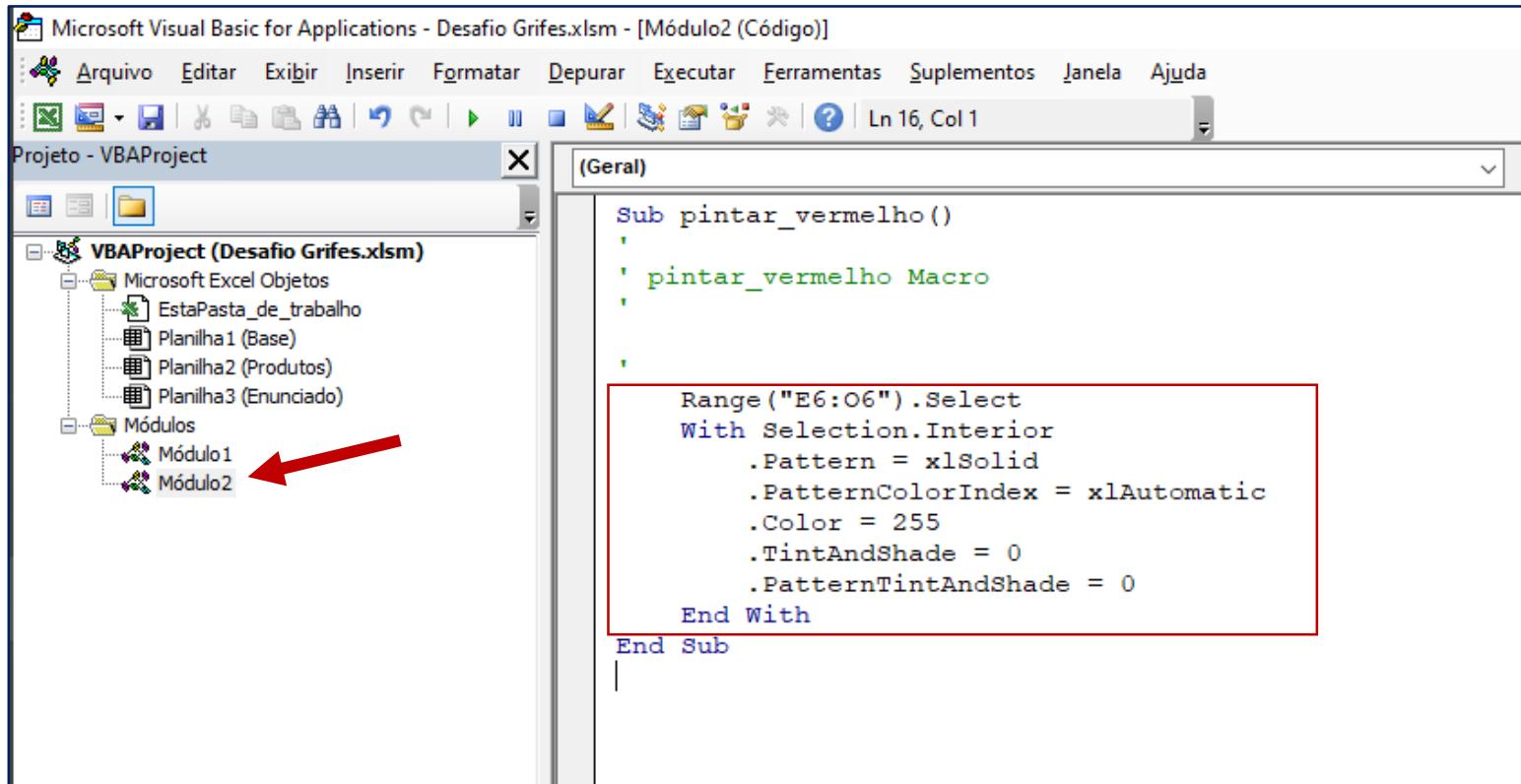
## Módulo 3 – Estrutura e Objetos do VBA – Desafio Compilação Grifes (Parte 4)

157



The screenshot shows a Microsoft Excel window titled "Desafio Grifes.xlsxm - Excel". The ribbon is visible at the top with tabs like Arquivo, Página Inicial, Inserir, etc. A color palette is overlaid on the screen, centered over a table. The palette has sections for "Cores do Tema" (with a yellow square highlighted by a red arrow), "Cores Padrão" (with a yellow square), and "Mais Cores..." (listing AMARELO, BRANCO, AZUL, ROSA, VERDE ESMERALDA, and VERMELHO). The table below the palette has rows for "Grife", "Status", and "Estoque Mínimo" with values "Hashtag", "Nova Coleção", and "1200" respectively. The rest of the table is empty with columns labeled "Produto 1" through "Produto 10" and "Estoque".

Em seguida, volte a pintar as células na cor da planilha para voltar ao estado anterior.



No ambiente VBA, clique no Módulo2 e copie o código criado para pintar as células de vermelho.

Não se preocupe com a estrutura With que aparece, este é um código padrão que só precisamos copiar e colar no nosso código que está no Módulo1.

```
Next  
Sheets("Produtos").Activate  
Next  
estoque_minimo = Range("B3").Value  
For linha2 = 6 To 11  
    If Cells(linha2, 15).Value < estoque_minimo Then  
        Range("E6:O6").Select  
        With Selection.Interior  
            .Pattern = xlSolid  
            .PatternColorIndex = xlAutomatic  
            .Color = 255  
            .TintAndShade = 0  
            .PatternTintAndShade = 0  
        End With  
    End If  
Next  
End Sub
```

Para testar o valor do estoque e pintar a linha inteira de acordo com o estoque mínimo, vamos criar uma nova estrutura de repetição, ao final do nosso código principal.

Primeiro declaramos uma variável chamada estoque\_mínimo, que recebe o valor da célula B3.

Em seguida, para a linha 6 da tabela até a linha 11, vamos verificar se o estoque total é menor que o estoque mínimo. Repare que criamos uma variável linha2 porque já tínhamos criado uma variável chamada linha anteriormente.

O código para pintar a linha de vermelho deverá ser colado dentro do teste If, pois se estoque < estoque\_mínimo então a linha deverá ser pintada de vermelho.

```

        Cells(linha_registro, coluna).Value = codigo
        Cells(linha_registro, 15).Value = estoque 'registra na
        Sheets("Base").Activate
    End If
Next
Sheets("Produtos").Activate
Next
estoque_minimo = Range("B3").Value
For linha2 = 6 To 11
    If Cells(linha2, 15).Value < estoque_minimo Then
        Range("E" & linha2 & ":O" & linha2).Select
        With Selection.Interior
            .Pattern = xlSolid
            .PatternColorIndex = xlAutomatic
            .Color = 255
            .TintAndShade = 0
            .PatternTintAndShade = 0
        End With
    End If
Next
End Sub

```

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1 Grife	Reserva													
2 Status	Coleção Antiga													
3 Estoque Mínimo	1500													
4														
5	Cor	Produto 1	Produto 2	Produto 3	Produto 4	Produto 5	Produto 6	Produto 7	Produto 8	Produto 9	Produto 10	Estoque		
6	AMARELO	06.040.237	02.140.243	03.070.244	03.130.230	05.050.247	07.030.237	02.070.235	02.140.243	03.070.244	03.130.230	5803		
7	BRANCO	06.060.250	03.140.239	06.020.249	07.030.236	03.060.250	03.140.239	06.020.249	07.030.236	02.050.230	03.050.247	6094		
8	AZUL	06.140.239	02.070.230	03.120.234	04.030.234	04.090.230	05.120.233	06.130.238	02.020.242	02.070.230	03.120.234	1272		
9	ROSA	02.050.231	04.080.247	06.060.240	06.100.249	02.050.231	04.080.247	06.060.240	06.100.249	02.080.246	02.150.244	513		
10	VERDE ESMERALDA	06.030.239	04.060.250	04.150.236	05.020.239	06.150.232	03.140.240	04.060.250	04.150.236	05.020.239	06.150.232	3461		
11	VERMELHO	03.050.231	03.070.242	03.110.241	05.030.239	05.150.235	05.150.236	06.070.240	07.030.239	02.050.246	03.070.242	4773		
12														
13														

Devemos tomar cuidado porque o código do jeito que está pinta apenas as células da linha 6.

`Range ("E6:O6").Select`

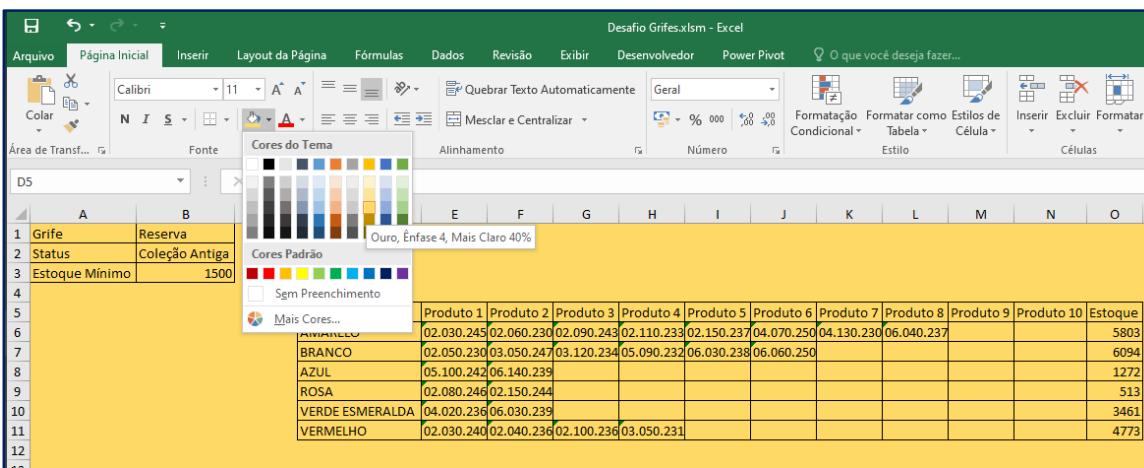
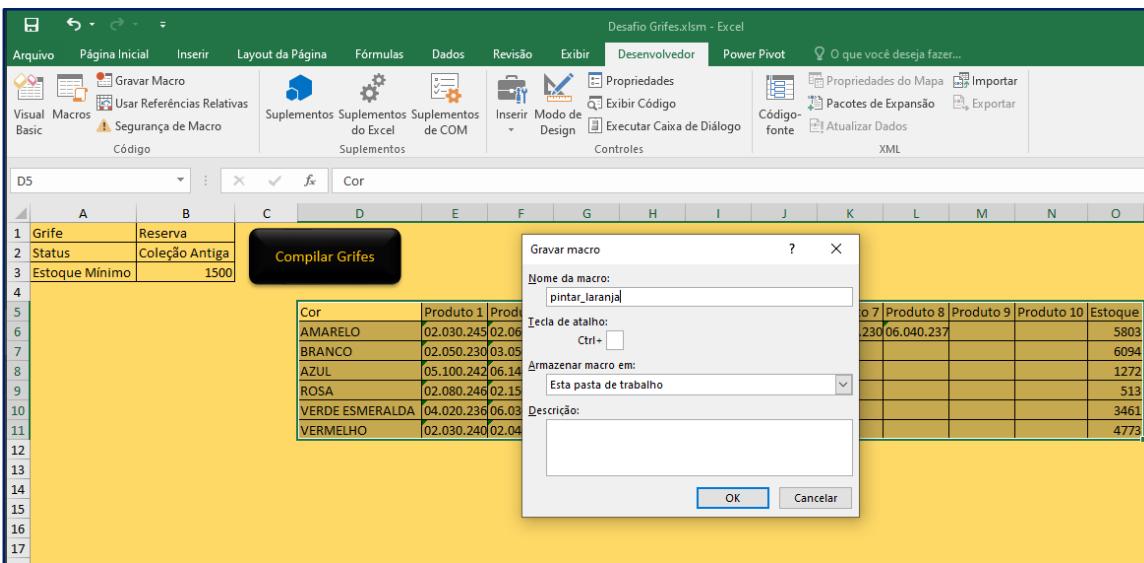
Para tornar isso automático de acordo com a variável linha2 do loop  
For devemos substituir o texto 6 pela variável linha2:

`Range ("E" & linha2 & ":O" & linha2).Select`

Com essa alteração, ao executarmos o código, as linhas com estoque abaixo do estoque mínimo serão pintadas corretamente.

# Módulo 3 – Estrutura e Objetos do VBA – Desafio Compilação Grifes (Parte 5)

161



Estamos quase finalizando o nosso código, porém ainda temos que resolver 2 problemas.

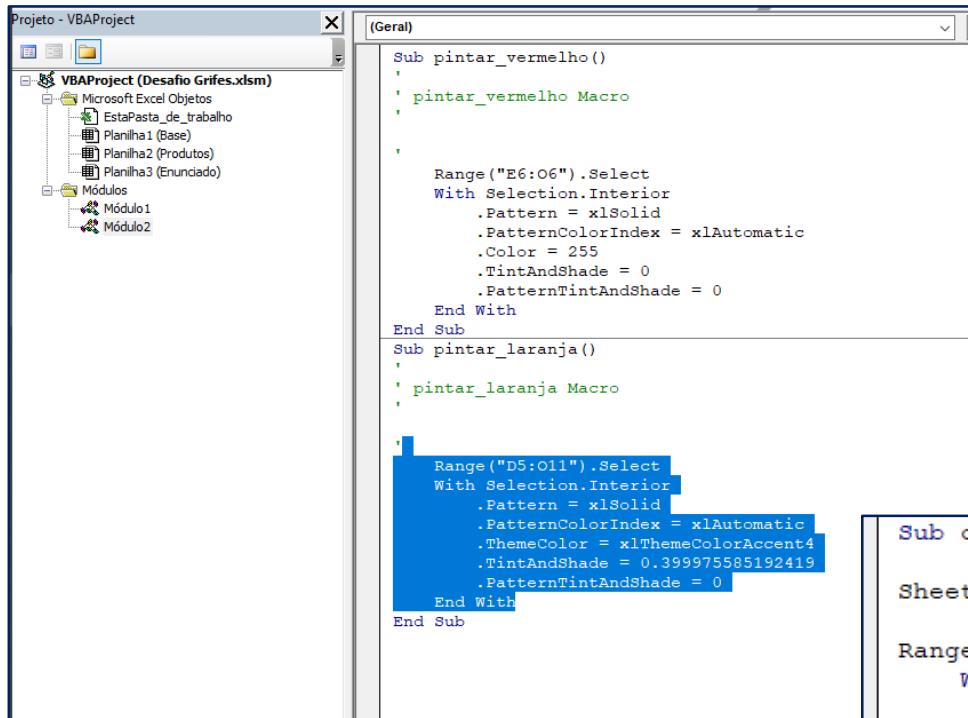
**Primeiro**, a macro do jeito que está, pinta as células corretamente, mas caso a gente mude a Grife na célula B1, o Status na célula B2 e o Estoque Mínimo na célula B3, a linha pintada continuará pintada de vermelho.

Então, os próximos passos agora são:

- 1) gravar uma macro chamada “pintar\_laranja”;
- 2) selecionar toda a tabela de D6 até O11.
- 3) Pintar as células selecionadas com a cor laranja original.

Lembre-se de parar a gravação após finalizar esse passo a passo anterior.

**Segundo**, antes de executar a macro, temos que apagar os códigos já preenchidos. O ideal é que a gente torne esse processo automático, limpando sempre os valores da tabela antes de executar o código.



```
Sub compilacao()

    Sheets("Produtos").Activate

    Range("D5:O11").Select
    With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .ThemeColor = xlThemeColorAccent4
        .TintAndShade = 0.399975585192419
        .PatternTintAndShade = 0
    End With

    Range("E6:O11").ClearContents

    grife = Range("B1").Value
    Status = Range("B2").Value

    For linha_registro = 6 To 11
```

Para copiar o código de pintar a tabela de laranja é só selecionar o Módulo2, copiar o código e colar logo no início da nossa macro no Módulo1.

Além disso, usamos o comando Range("E6:O11").ClearContents para limpar todo o conteúdo desse intervalo para adicionar os novos códigos.

```
Sub compilacao()
    Sheets("Produtos").Activate
    Range("D5:O11").Select
    With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .ThemeColor = xlThemeColorAccent4
        .TintAndShade = 0.399975585192419
        .PatternTintAndShade = 0
    End With
    Range("E6:O11").ClearContents

    grife = Range("B1").Value
    Status = Range("B2").Value

    For linha_registro = 6 To 11
        estoque = 0
        cor = Cells(linha_registro, 4).Value
        Sheets("Base").Activate
        For linha = 2 To 149
            If Cells(linha, 3).Value = cor And Cells(linha, 4).Value = grife And Cells(linha, 7).Value = Status Then
                codigo = Cells(linha, 1).Value
                estoque = estoque + Cells(linha, 6).Value
            End If
        Next linha
        Sheets("Produtos").Activate
        coluna = Range("O" & linha_registro).End(xlToLeft).Column + 1
        Range("O" & coluna).Value = estoque
    Next linha_registro
End Sub
```

O código final é mostrado em 2 prints  
nesta e na próxima página.

```
    coluna = Range("O" & linha_registro).End(xlToLeft).Column + 1

    Cells(linha_registro, coluna).Value = codigo

    Cells(linha_registro, 15).Value = estoque 'registra na coluna O

    Sheets("Base").Activate

    End If

    Next

Sheets("Produtos").Activate

Next

estoque_minimo = Range("B3").Value

For linha2 = 6 To 11

    If Cells(linha2, 15).Value < estoque_minimo Then

        Range("E" & linha2 & ":O" & linha2).Select

        With Selection.Interior
            .Pattern = xlSolid
            .PatternColorIndex = xlAutomatic
            .Color = 255
            .TintAndShade = 0
            .PatternTintAndShade = 0
        End With

    End If

Next

End Sub
```

A	B	C	D	E	F	G	H	I	J	K	L	M
2	Crie uma macro que preencha os gastos em BRL segundo a tabela abaixo:											
3	Moeda	Conversão p/ BRL										
4	USD	4,08										
5	EUR	4,52										
6	PEN	1,21										
7	BOB	0,59										
8												
9												
10												
11												
12												
13												
14												
15												

Gasto	Data	Gasto	Moeda	Total BRL
Máquinas	02/05/2016	2.000,00	USD	
Alimentação	25/05/2016	1.000,00	PEN	
Mão de Obra	28/06/2016	1.200,00	USD	
Estoque	15/07/2016	3.000,00	EUR	
Mão de Obra	14/08/2016	10.000,00	BOB	
Alimentação	21/08/2016	3.000,00	EUR	
Estoque	26/08/2016	6.000,00	EUR	
Mão de Obra	15/09/2016	2.000,00	USD	
Máquinas	25/09/2016	2.000,00	PEN	
Alimentação	13/10/2016	9.000,00	BOB	
Mão de Obra	22/10/2016	9.000,00	EUR	
Estoque	14/11/2016	4.000,00	PEN	

No próximo exemplo vamos ver uma nova estrutura chamada Select Case. Assim como o If, esta também é uma estrutura de seleção, que permite retornar valores de acordo com alguma condição.

Vamos aplicar este conceito na planilha ao lado. Temos uma tabela com os gastos de uma empresa em diferentes moedas, pois se trata de uma empresa que importa produtos de diferentes países. E na tabela da esquerda temos a conversão de cada moeda para o real. O que queremos é calcular na coluna L o total gasto levando em consideração o câmbio de cada país.

```
Sub moedas()
    For linha = 3 To 14
        If Cells(linha, 11).Value = "USD" Then
            Cells(linha, 12).Value = Cells(linha, 10).Value * Cells(4, 3).Value
        ElseIf Cells(linha, 11).Value = "EUR" Then
            Cells(linha, 12).Value = Cells(linha, 10).Value * Cells(5, 3).Value
        ElseIf Cells(linha, 11).Value = "PEN" Then
            Cells(linha, 12).Value = Cells(linha, 10).Value * Cells(6, 3).Value
        Else
            Cells(linha, 12).Value = Cells(linha, 10).Value * Cells(7, 3).Value
        End If
    Next
End Sub
```

A solução que já sabemos fazer é mostrada no código ao lado. Como queremos executar um mesmo comando repetidas vezes na tabela (multiplicar o gasto pelo câmbio) então nossa estrutura principal é um For, que executará uma série de comandos da linha 3 até a linha 14.

Essa série de comandos são as diversas estruturas If para testar cada valor da coluna K da tabela e, de acordo com a sigla da moeda (USD, EUR, PEN, BOB) queremos escolher o câmbio certo.

	B	C	D	E	F	G	H	I	J	K	L
2	Crie uma macro que preencha os gastos em BRL segundo a tabela abaixo:										
3	Moeda	Conversão p/ BRL									
4	USD	4,08									
5	EUR	4,52									
6	PEN	1,21									
7	BOB	0,59									
8											
9											
10											
11											
12											
13											
14											
15											

Gasto	Data	Gasto	Moeda	Total BRL
Máquinas	02/05/2016	2.000,00	USD	R\$ 8.160,00
Alimentação	25/05/2016	1.000,00	PEN	R\$ 1.210,00
Mão de Obra	28/06/2016	1.200,00	USD	R\$ 4.896,00
Estoque	15/07/2016	3.000,00	EUR	R\$ 13.560,00
Mão de Obra	14/08/2016	10.000,00	BOB	R\$ 5.900,00
Alimentação	21/08/2016	3.000,00	EUR	R\$ 13.560,00
Estoque	26/08/2016	6.000,00	EUR	R\$ 27.120,00
Mão de Obra	15/09/2016	2.000,00	USD	R\$ 8.160,00
Máquinas	25/09/2016	2.000,00	PEN	R\$ 2.420,00
Alimentação	13/10/2016	9.000,00	BOB	R\$ 5.310,00
Mão de Obra	22/10/2016	9.000,00	EUR	R\$ 40.680,00
Estoque	14/11/2016	4.000,00	PEN	R\$ 4.840,00

Para executar a macro é só usar o atalho F5. O resultado está mostrado ao lado.

A partir da próxima página vamos entender como é a estrutura Select Case e perceber que este código poderia ser simplificado com uma estrutura mais resumida e sem a necessidade de repetir diversas vezes praticamente o mesmo teste lógico em cada If/Elseif.

Em resumo, a estrutura Select Case permite que você analise algum valor e, de acordo com esse valor, ele executa algum caso específico. Podemos traduzí-lo ao pé da letra como: Selecione o caso X. E ai no momento em que ele encontrar esse caso X nas opções dadas dentro da estrutura ele executa os comandos para aquele caso.

- **Estrutura Select Case no VBA**

Select Case **valor\_analisado**

Este **valor\_analisado** pode ser uma **célula**, uma **variável**, um **número** ou um **texto**. A ideia nessa estrutura é que, se o **valor\_analisado** for igual ao **caso1**, então serão executados os comandos do **caso1**. Se for igual ao **caso2**, então serão executados os comandos no **caso2**, se não, serão executados os comandos no **case Else**. Podemos ter diversos casos.

Case **caso1**



Comandos que vão ser executados caso o **valor\_analisado** seja igual ao **caso1**

Case **caso2**



Comandos que vão ser executados caso o **valor\_analisado** seja igual ao **caso2**

Case **Else**



Comandos que vão ser executados caso o **valor\_analisado** não seja igual a nenhum dos casos anteriores

End Select

Como exemplo, voltamos à situação de calcular a situação do aluno. Assim, o valor a ser testado é a nota na célula B2. Assim, com o Select Caso, damos as possíveis situações em que esse valor analisado pode cair: Case  $\geq 7$ , Case  $\geq 5$  e Case Else, ou seja, o caso contrário. Como no exemplo a nota é igual a 7,5, então ele cairá no primeiro Case e retornará o resultado “Aprovado”. Esta é uma estrutura mais resumida que o If e pode ser mais interessante em situações onde precisamos fazer diversos testes.

- **Estrutura Select Case no VBA**

```
nota = Cells(2, 1).Value
```

```
Select Case nota
```

```
Case >=7
```

```
    Cells(2, 2).Value = "Aprovado"
```

```
Case >=5
```

```
    Cells(2, 2).Value = "Prova Final"
```

```
Case Else
```

```
    Cells(2, 2).Value = "Reprovado"
```

```
End Select
```

	A	B
1	Nota	Situação
2	7,5	

```
Sub moedas_selec_case()
    For linha = 3 To 14
        Select Case Cells(linha, 11).Value
        Case "USD"
            Cells(linha, 12).Value = Cells(linha, 10).Value * Cells(4, 3).Value
        Case "EUR"
            Cells(linha, 12).Value = Cells(linha, 10).Value * Cells(5, 3).Value
        Case "PEN"
            Cells(linha, 12).Value = Cells(linha, 10).Value * Cells(6, 3).Value
        Case Else
            Cells(linha, 12).Value = Cells(linha, 10).Value * Cells(7, 3).Value
        End Select
    Next
End Sub
```

A solução do exercício anterior utilizando a estrutura Select Case está mostrada ao lado. Note que a estrutura é um pouco mais simples pois não precisamos repetir o comando **Cells(linha, 11).Value** diversas vezes pois o Select Case pede que este valor analisado seja escrito apenas uma vez, e cada situação possível é simplesmente iniciada pela palavra Case.

Se você executar o código ao lado, ele resulta no mesmo resultado obtido anteriormente. Portanto, ele é uma alternativa ao If. De qualquer forma, o If é uma estrutura muito mais utilizada nestes casos, mas saber o funcionamento do Select Case é muito importante, principalmente caso você tenha que lidar com planilhas de outras pessoas que podem preferir esta estrutura e a utilizem em seus códigos.

Módulo 4

# Trabalhando com Objetos

A	B	C	D	E	F	G	H	I	J
1	Vendedor	Vendas	Unidade	Venda Mínima					
2	Diego	R\$ 4.100,00	Rio de Janeiro						
3	Alon	R\$ 6.900,00	Belo Horizonte						
4	João	R\$ 8.100,00	São Paulo						
5	Lira	R\$ 6.000,00	Belo Horizonte						
6	Sergio	R\$ 8.400,00	São Paulo						
7	Amanda	R\$ 8.000,00	Rio de Janeiro						
8	Karina	R\$ 3.500,00	Fortaleza						
9	Luiza	R\$ 6.000,00	Fortaleza						
10	Jessica	R\$ 7.500,00	São Paulo						
11									
12									
13									
14									
15									

- 1 Registre na tabela a venda mínima:  
R\$5.000
- 2 Coloque os valores em Negrito, Itálico e Sublinhados
- 3 Formate a Venda Mínima como Moeda

Vendas

Exemplo Aba Verde

Neste módulo 4 o nosso objetivo é trabalhar com diferentes objetos no VBA.

Na página 45, quando falamos que o VBA poderia ser considerada uma linguagem de programação orientada a objetos, justificamos com o fato de que nos códigos sempre executamos uma ação em cima de um objeto: uma célula, uma aba, um gráfico, um arquivo, etc.

Agora a ideia é a gente ver como trabalhar com estes objetos de uma maneira que não vimos até agora.

A	B	C	D	E	F	G	H	I
1	Vendedor	Vendas	Unidade	Venda Mínima				
2	Diego	R\$ 4.100,00	Rio de Janeiro	5.000				
3	Alon	R\$ 6.900,00	Belo Horizonte					
4	João	R\$ 8.100,00	São Paulo					
5	Lira	R\$ 6.000,00	Belo Horizonte					
6	Sergio	R\$ 8.400,00	São Paulo					
7	Amanda	R\$ 8.000,00	Rio de Janeiro					
8	Karina	R\$ 3.500,00	Fortaleza					
9	Luiza	R\$ 6.000,00	Fortaleza					
10	Jessica	R\$ 7.500,00	São Paulo					
11								
12								
13								
14								

- 1 Registre na tabela a venda mínima:  
R\$5.000
- 2 Coloque os valores em Negrito, Itálico e Sublinhados
- 3 Formate a Venda Mínima como Moeda

Para começar, vamos registrar, na célula D2, o valor de venda mínima igual a 5000.

Em seguida, antes de começar a gravar a macro, selecione a célula A1.

Agora sim, vamos gravar uma macro chamada **formatar** para:

- 1- Selecionar a célula D2.
- 2- Colocar o valor da célula em negrito.
- 3- Colocar o valor da célula em itálico.
- 4- Colocar o valor da célula sublinhado
- 5- Formatar a célula D2 para moeda.

## Módulo 4 – Trabalhando com Objetos – Introdução

173

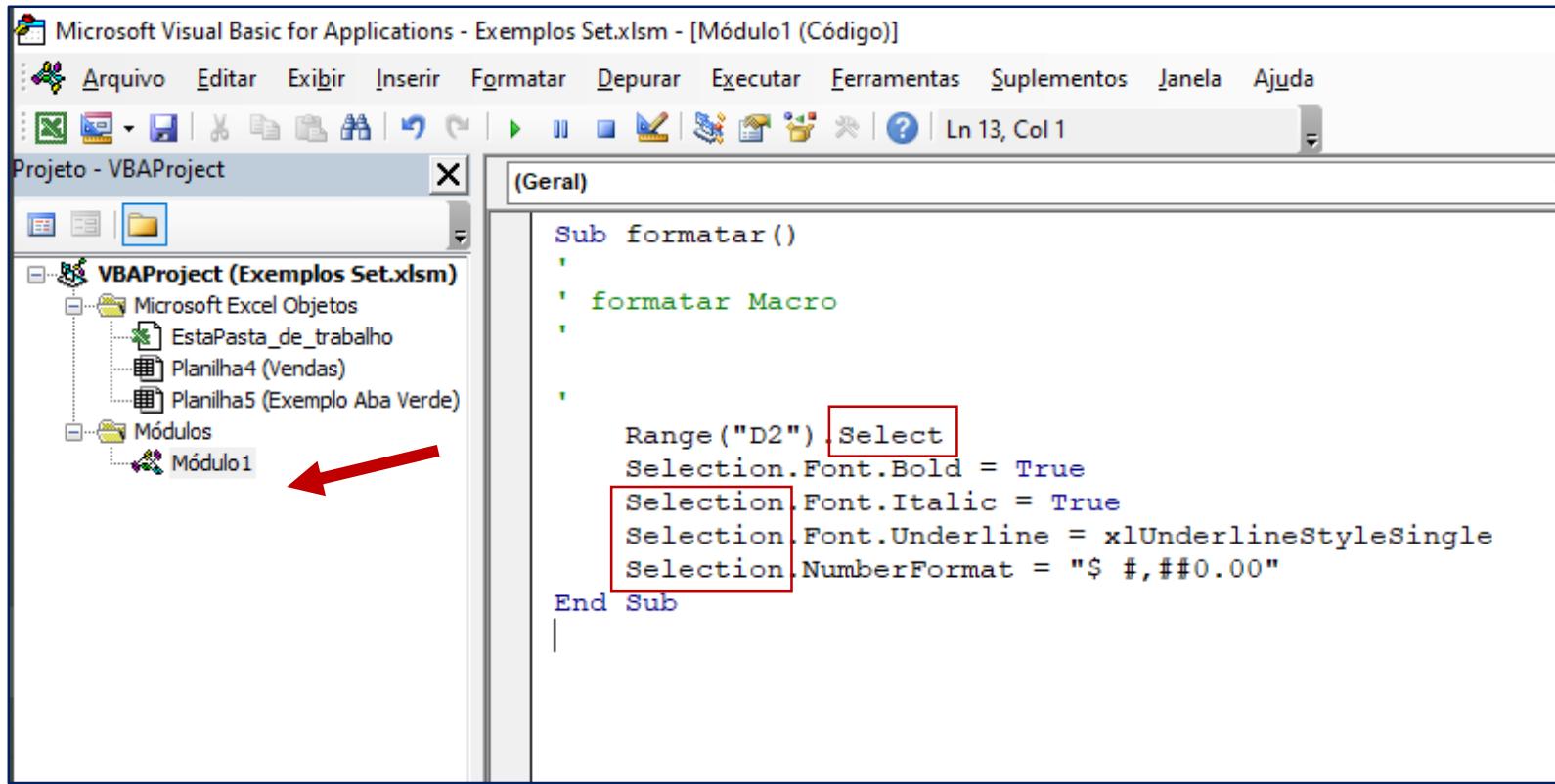
A screenshot of Microsoft Excel showing the ribbon with the 'Desenvolvedor' tab selected. A 'Gravar macro' dialog box is open, prompting the user to name the macro ('formatar'), assign a keyboard shortcut (Ctrl+Shift+F), and specify where to store the macro (current workbook). The background shows a table with columns Vendedor, Vendas, Unidade, and Venda Mínima.

Após realizar as formatações, não esqueça de Parar a Gravação na guia Desenvolvedor.

A screenshot of Microsoft Excel showing the ribbon with the 'Desenvolvedor' tab selected. The 'Número' tab in the ribbon is highlighted with a red arrow. The formula bar shows 'D2'. The font toolbar has bold, italic, and underline buttons highlighted with a red box. The number format dropdown shows 'Moeda' highlighted with a red box. The background shows a table with columns Vendedor, Vendas, Unidade, and Venda Mínima, where the last cell contains 'R\$ 5.000,00'.

1

Registre na tabela a venda mínima  
R\$5.000



The screenshot shows the Microsoft VBA editor interface. The title bar reads "Microsoft Visual Basic for Applications - Exemplos Set.xlsxm - [Módulo1 (Código)]". The menu bar includes Arquivo, Editar, Exibir, Inserir, Formatar, Depurar, Executar, Ferramentas, Suplementos, Janela, and Ajuda. The toolbar has various icons for file operations and macros. The Project Explorer on the left shows "VBAProject (Exemplos Set.xlsxm)" with "Microsoft Excel Objetos" and "Módulos" sections, where "Módulo1" is selected. The code editor window displays the following VBA code:

```
Sub formatar()
    '
    ' formatar Macro
    '

    Range("D2").Select
    Selection.Font.Bold = True
    Selection.Font.Italic = True
    Selection.Font.Underline = xlUnderlineStyleSingle
    Selection.NumberFormat = "$ #,##0.00"
End Sub
```

```
'Range("D2").Select
Range("D2").Font.Bold = True
Range("D2").Font.Italic = True
Range("D2").Font.Underline = xlUnderlineStyleSingle
Range("D2").NumberFormat = "$ #,##0.00"
```

No ambiente VBA, visualizamos o código no Módulo1. Agora o que vamos fazer é utilizar estes comandos para escrever uma macro que registre no intervalo de D2 até D10 o valor 5000, com as mesmas formatações que gravamos.

Aqui vale a gente observar que no código, a célula D2 é selecionada. Em seguida, uma vez que esta célula está selecionada, o VBA trata ela como um objeto Selection, ou seja, uma seleção. A partir daí, ele altera as propriedades fonte e formato do número dessa seleção. Uma outra maneira de escrever este código é mostrada ao lado. Ou seja, podemos substituir o objeto Selection pelo objeto Range, dá no mesmo.

```
Sub venda_minima()
    Range("D2:D10").Value = 5000
    Range("D2").Select
        Selection.Font.Bold = True
        Selection.Font.Italic = True
        Selection.Font.Underline = xlUnderlineStyleSingle
        Selection.NumberFormat = "$ #,##0.00"
End Sub
```

Agora criamos uma macro chamada **venda\_mínima** e começamos escrevendo a estrutura `Range("D2:D10").Value = 5000` para escrever em todo o intervalo. Abaixo colamos a estrutura da macro gravada, ainda com o objeto `Selection`.

Para formatar todo o intervalo de células (e não só D2), aqui também poderíamos fazer a substituição do objeto `Selection` pelo objeto `Range`. A linha referente ao `Range("D2").Select` pode ser apagada.

```
Sub venda_minima()
    Range("D2:D10").Value = 5000
    Range("D2:D10").Font.Bold = True
    Range("D2:D10").Font.Italic = True
    Range("D2:D10").Font.Underline = xlUnderlineStyleSingle
    Range("D2:D10").NumberFormat = "$ #,##0.00"
End Sub
```

Se você executar o código ao lado, verá que todas as células do intervalo D2 até D10 serão formatadas. Porém, podemos observar que este código não está otimizado. Repare que tivemos que escrever o trecho `Range("D2:D10")` diversas vezes.

Assim como já vimos sobre declaração de variáveis, na próxima página veremos como poderíamos declarar uma variável que recebe um intervalo, em vez de apenas um valor. Assim, se o intervalo variasse, só precisaríamos mudar na variável uma única vez.

```
Sub venda_minima()
    Dim intervalo As Range
    Set intervalo = Range("D2:D10")
    intervalo.Value = 5000
    intervalo.Font.Bold = True
    intervalo.Font.Italic = True
    intervalo.Font.Underline = xlUnderlineStyleSingle
    intervalo.NumberFormat = "$ #,##0.00"
End Sub
```

Declarar uma variável **intervalo** e atribuir a ela o valor do Range permite que a gente escreve essa instrução Range("D2:D10") uma única vez. Assim, sempre que precisarmos fazer referência a essas células, basta utilizar a variável **intervalo**. Além disso, sempre que for necessário alterar as células desse Range, também fazemos uma única vez na variável **intervalo**.

O código ao lado mostra a maneira como declaramos uma variável chamada **intervalo** que recebe um intervalo de células. Diferente do que vimos até agora, a nossa variável não vai armazenar um Integer, ou String, ou Double. Essa variável vai receber um **intervalo de células**. Dessa forma, o seu tipo será na verdade um **Range**.

Porém, na hora de atribuir um intervalo à nossa variável, teremos que usar a instrução **Set**. Esta instrução é **necessária na declaração de toda a variável que receber um objeto do Excel**. Já comentamos que objetos no Excel são essencialmente: células, abas, arquivos e gráficos. Veremos mais para frente que o tipo destas variáveis são, respectivamente: Range, Worksheet, Workbook e ChartObject. Assim, se você tiver qualquer dúvida com relação a quando usar o Set, será apenas em casos que precisarmos declarar este tipo de variável. Variáveis do tipo Integer, Double, Long, String, Boolean, Date, não precisam dessa estrutura.

Veremos mais para frente diversos exemplos para entender corretamente quando o Set será necessário.

```
Sub cadastro_vendedor()
    Dim celula As Range
    Set celula = Range("A10")
    celula = "Julia"
End Sub
```

Para entender qual a diferença do Set na prática, vamos fazer um exemplo bem simples. Imagina que nós queremos escrever na célula A10, por meio do VBA, um nome diferente, por exemplo, João. A macro que faria isso é mostrada ao lado. Após executá-la, teremos como resultado o nome João na célula A10.



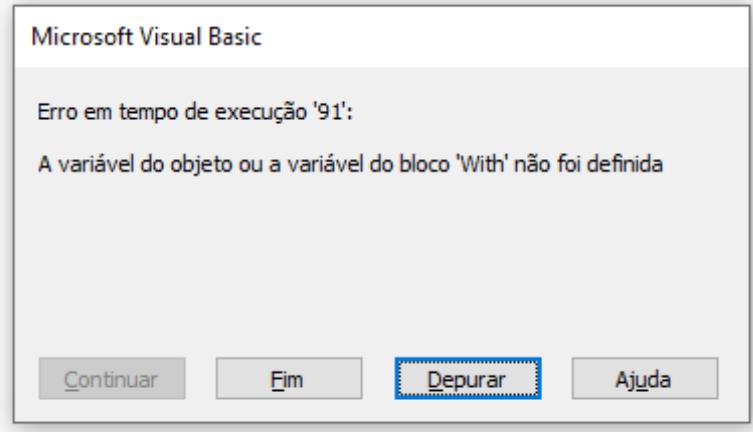
	A	B	C	D
1	Vendedor	Vendas	Unidade	Venda Mínima
2	Diego	R\$ 4.100,00	Rio de Janeiro	<b>R\$ 5.000,00</b>
3	Alon	R\$ 6.900,00	Belo Horizonte	<b>R\$ 5.000,00</b>
4	João	R\$ 8.100,00	São Paulo	<b>R\$ 5.000,00</b>
5	Lira	R\$ 6.000,00	Belo Horizonte	<b>R\$ 5.000,00</b>
6	Sergio	R\$ 8.400,00	São Paulo	<b>R\$ 5.000,00</b>
7	Amanda	R\$ 8.000,00	Rio de Janeiro	<b>R\$ 5.000,00</b>
8	Karina	R\$ 3.500,00	Fortaleza	<b>R\$ 5.000,00</b>
9	Luiza	R\$ 6.000,00	Fortaleza	<b>R\$ 5.000,00</b>
10	Jessica	R\$ 7.500,00	São Paulo	<b>R\$ 5.000,00</b>

	A	B	C	D
1	Vendedor	Vendas	Unidade	Venda Mínima
2	Diego	R\$ 4.100,00	Rio de Janeiro	<b>R\$ 5.000,00</b>
3	Alon	R\$ 6.900,00	Belo Horizonte	<b>R\$ 5.000,00</b>
4	João	R\$ 8.100,00	São Paulo	<b>R\$ 5.000,00</b>
5	Lira	R\$ 6.000,00	Belo Horizonte	<b>R\$ 5.000,00</b>
6	Sergio	R\$ 8.400,00	São Paulo	<b>R\$ 5.000,00</b>
7	Amanda	R\$ 8.000,00	Rio de Janeiro	<b>R\$ 5.000,00</b>
8	Karina	R\$ 3.500,00	Fortaleza	<b>R\$ 5.000,00</b>
9	Luiza	R\$ 6.000,00	Fortaleza	<b>R\$ 5.000,00</b>
10	Julia	R\$ 7.500,00	São Paulo	<b>R\$ 5.000,00</b>

Mas o que aconteceria se não colocássemos o Set na hora de declarar a variável **celula**?

```
Sub cadastro_vendedor()
    Dim celula As Range
    celula = Range("A10")
    celula = "Julia"
End Sub
```



```
Sub cadastro_vendedor()
    Dim celula As Range
    celula = Range("A10") // Line 3 is highlighted with a yellow background
    celula = "Julia"
End Sub
```

O código irá retornar o erro ao lado. Acontece que, como não definimos para o Excel que a variável **celula** se trata de uma célula, e não do valor dela, ele tenta armazenar o valor dessa célula (a propriedade **.Value** do **Range** é padrão caso não escrevamos nada).

Só que, no início do código, definimos que **celula** é do tipo **Range**, e não do tipo **String**, por exemplo.

Ao clicar em depurar, a linha **celula = Range("A10")** é sublinhada. Bom, então se o erro é que a variável **celula** deveria então ser do tipo **String**, então vamos declará-la assim e ver no que dá. Além disso, vamos mudar agora o nome para “**Pedro**” e ver se o código funcionará como esperado.

```
Sub cadastro_vendedor()  
  
    Dim celula As String  
  
    celula = Range("A10")  
  
    celula = "Pedro"  
  
End Sub
```

Ao executar o código, nenhum erro ocorre. Porém, o nome “Pedro” não é escrito na célula.



	A	B	C	D
1	Vendedor	Vendas	Unidade	Venda Mínima
2	Diego	R\$ 4.100,00	Rio de Janeiro	<u>R\$ 5.000,00</u>
3	Alon	R\$ 6.900,00	Belo Horizonte	<u>R\$ 5.000,00</u>
4	João	R\$ 8.100,00	São Paulo	<u>R\$ 5.000,00</u>
5	Lira	R\$ 6.000,00	Belo Horizonte	<u>R\$ 5.000,00</u>
6	Sergio	R\$ 8.400,00	São Paulo	<u>R\$ 5.000,00</u>
7	Amanda	R\$ 8.000,00	Rio de Janeiro	<u>R\$ 5.000,00</u>
8	Karina	R\$ 3.500,00	Fortaleza	<u>R\$ 5.000,00</u>
9	Luiza	R\$ 6.000,00	Fortaleza	<u>R\$ 5.000,00</u>
10	Julia	R\$ 7.500,00	São Paulo	<u>R\$ 5.000,00</u>



	A	B	C	D
1	Vendedor	Vendas	Unidade	Venda Mínima
2	Diego	R\$ 4.100,00	Rio de Janeiro	<u>R\$ 5.000,00</u>
3	Alon	R\$ 6.900,00	Belo Horizonte	<u>R\$ 5.000,00</u>
4	João	R\$ 8.100,00	São Paulo	<u>R\$ 5.000,00</u>
5	Lira	R\$ 6.000,00	Belo Horizonte	<u>R\$ 5.000,00</u>
6	Sergio	R\$ 8.400,00	São Paulo	<u>R\$ 5.000,00</u>
7	Amanda	R\$ 8.000,00	Rio de Janeiro	<u>R\$ 5.000,00</u>
8	Karina	R\$ 3.500,00	Fortaleza	<u>R\$ 5.000,00</u>
9	Luiza	R\$ 6.000,00	Fortaleza	<u>R\$ 5.000,00</u>
10	Pedro	R\$ 7.500,00	São Paulo	<u>R\$ 5.000,00</u>

Isso porque agora, o que está acontecendo é o seguinte:

1. A variável celula é declarada como String.
2. A variável celula recebe o valor de Range("A10"), ou seja, recebe “Julia”.
3. A variável celula agora recebe o nome “Pedro”. Porém, não escreve em nenhum lugar o resultado.

Portanto, podemos ver que o Set faz toda a diferença para o código entender corretamente que estamos trabalhando com um range, e não com uma variável de um outro tipo comum (String, Integer, Double, etc.)



### ATENÇÃO !

Neste ponto, pode ser que você tenha se perguntado porque não colocamos a propriedade `.Value` ao declarar as variáveis do tipo `Range`:

```
Set celula = Range("A10")
```

```
Set intervalo = Range("D2:D10")
```

Ou ainda: você colocou o `.Value` e houve algum erro. A explicação é a seguinte: quando estamos trabalhando com uma variável do tipo `Range`, não podemos especificar que esta variável vai receber o valor da célula. Se fizéssemos `Set celula = Range("A10").Value` estariámos dizendo que uma variável do tipo célula (`Range`) deveria receber, na verdade, um valor, e não a célula em si, o que geraria um conflito para o VBA. Existe uma diferença chave entre uma variável que recebe um valor e uma variável que recebe um `Range`.

Você deve lembrar que, com um `Range`, podemos fazer diversas coisas, como selecionar (`Range("A1").Select`), mudar a fonte para negrito (`Range("A1").Font.Bold = True`), limpar o conteúdo (`Range("A1").ClearContents`), inclusive modificar o seu valor através da propriedade `.Value` (`Range("A1").Value = 10`). Quando declaramos uma variável do tipo `Range`, o nosso objetivo é o mesmo: poder modificar todas essas propriedades, só que dessa vez, através de variáveis declaradas (`celula` e `intervalo`). Se fizermos `Set celula = Range("A10").Value`, estaremos limitando a variável a receber apenas o valor da célula, e perderemos o acesso a todas as propriedades que esta célula possui (`.Select`, `.Activate`, `.ClearContents`, etc.).

Por isso, sempre que você declarar uma variável do tipo `Range` com o `Set`, você nunca poderá colocar o `.Value` ao final, pois o código não funcionará corretamente.

A	B	C	D	E	F	G	H	I	J
1	Vendedor	Vendas	Unidade	Venda Mínima					
2	Diego	R\$ 4.100,00	Rio de Janeiro						
3	Alon	R\$ 6.900,00	Belo Horizonte						
4	João	R\$ 8.100,00	São Paulo						
5	Lira	R\$ 6.000,00	Belo Horizonte						
6	Sergio	R\$ 8.400,00	São Paulo						
7	Amanda	R\$ 8.000,00	Rio de Janeiro						
8	Karina	R\$ 3.500,00	Fortaleza						
9	Luiza	R\$ 6.000,00	Fortaleza						
10	Jessica	R\$ 7.500,00	São Paulo						
11									
12									
13									
14									
15									

1 Registre na tabela a venda mínima:  
R\$5.000

2 Coloque os valores em Negrito, Itálico e Sublinhados

3 Formate a Venda Mínima como Moeda

Vendas Exemplo Aba Verde

A	B	C	
1	Funcionário	Cargo	Salário
2	Adriana de Azevedo	Gerente 2	R\$12.320
3	Adriana Mosqueira Rodrigues Lopes	Estagiário 1	R\$1.600
4	Adriana Passos	Gerente 2	R\$12.320
5	Adriana Torres Carneiro	Estagiário 2	R\$2.000
6	Adriane Chagas	Estagiário 2	

Vendas Exemplo Aba Verde

Além de utilizar variáveis que podem receber um Range, também podemos utilizar variáveis que recebem planilhas.

Para exemplificar, imagine o seguinte: neste nosso arquivo, temos duas abas: “Vendas” e “Exemplo Aba Verde”.

Imagine que queremos:

1. Na aba “Vendas”, escrever o valor 5000 em todas as células do intervalo D2 até D10.
2. Na mesma macro, modificar as informações do primeiro funcionário, no caso da Adriana de Azevedo, na aba “Exemplo Aba Verde”.
3. Selecionar a aba “Vendas” ao final do código.

```
Sub registra_valores()
    Sheets("Vendas").Activate
    Range("D2:D10").Value = 5000
    Sheets("Exemplo Aba Verde").Activate
    Range("A2").Value = "Pedro Bastos"      'Nome
    Range("B2").Value = "Estagiário 2"     'Cargo
    Range("C2").Value = 2100                'Salário
    Sheets("Vendas").Activate
End Sub
```

Então, teríamos que fazer algo parecido com o código ao lado. O que não é muito prático e otimizado é ter que escrever várias vezes o comando Sheets(nome da aba). O ideal seria declarar duas variáveis, uma para armazenar cada aba, e sempre que necessário, utilizar essa variável no código para facilitar as ações que quisermos executar nestas abas.

```
Sub registra_valores()
    Dim aba_vendas As Worksheet
    Dim aba_verde As Worksheet

    Set aba_vendas = Sheets("Vendas")
    Set aba_verde = Sheets("Exemplo Aba Verde")

    aba_vendas.Activate
    Range("D2:D10").Value = 5000

    aba_verde.Activate

    Range("A2").Value = "Pedro Bastos"      'Nome
    Range("B2").Value = "Estagiário 2"      'Cargo
    Range("C2").Value = 2100                 'Salário

    aba_vendas.Activate
End Sub
```

O código modificado é mostrado ao lado. Abas são declaradas como sendo do tipo **Worksheet**.

Esta forma de declarar abas como variáveis é muito útil quando trabalhamos com arquivos que possuem muitas planilhas. Além disso, caso o nome de uma aba mude, será muito mais fácil mudar no código uma única vez na sua respectiva variável do que ficar procurando todas as vezes que escrevemos o nome da aba no código e ter que mudar um por um.

Achou que não ia ter mais estrutura de repetição? Achou errado. Faltou a gente falar da estrutura **For Each**. Esta é uma estrutura de repetição parecida com o For, com uma pequena diferença: em vez dela ser executada de uma determinada linha inicial até uma final, por exemplo, o **For Each** vai executar uma série de comandos para cada objeto de um conjunto de objetos. Ou seja, agora conseguiremos executar uma série de comandos em várias células de uma vez, ou várias abas de uma vez, ou vários arquivos de uma vez. Por isso deixamos para falar sobre ela no módulo 4, que é o momento mais oportuno, dado que estamos trabalhando com objetos.

A estrutura **For Each** está resumida abaixo:

- **Estrutura For Each no VBA**

**For Each variável in conjunto**

Comandos que vão ser executados pra cada objeto do conjunto (cada célula, ou cada aba, etc)

**Conjuntos de objetos. Exemplo:**

1. Células | **Range**
2. Abas | **Sheets**
3. Arquivos | **Workbooks**
4. Gráficos | **ChartObjects**

Um primeiro exemplo simples é a estrutura do For Each no caso de a gente querer executar um determinado comando repetidas vezes, no intervalo que vai de A1 até A10. O comando em questão é escrever o texto “VBA” em cada uma das células do intervalo. A estrutura do código é mostrada abaixo.

Podemos ler a primeira linha do código como sendo, literalmente: **Para Cada célula no intervalo D1 até D10, a célula recebe o texto “VBA”.**

- **Estrutura For Each no VBA: Escrevendo em cada célula de um intervalo**

For Each **celula** in Range("A1:A10")

**celula.Value = "VBA"**

Variável do tipo **Range**  
Declaração:  
Dim **celula** as **Range**

Vai escrever “VBA” nas células A1 até A10. Primeiro na A1, depois na A2, depois na A3, e assim vai

Next

De certa forma, o exemplo anterior já conseguiríamos fazer com o For. Mas o que não conseguimos com o que sabemos até agora é percorrer várias abas de um arquivo e executar uma ação em cada uma delas. Com o For Each isso é possível.

No código exemplo abaixo, podemos ler novamente a primeira linha do código como sendo: **Para Cada aba no intervalo de abas (Sheets), ative a aba e em seguida escreva na célula A1 o texto “VBA”.**

- Estrutura For Each no VBA: Escrevendo em cada aba de um arquivo

For Each aba in Sheets

aba.Activate

Cells(1, 1).Value = "VBA"

Next

Variável do tipo Worksheet  
Declaração:  
Dim aba as Worksheet

Vai escrever VBA na célula A1 de todas as abas abertas.  
Primeiro vai ativar a 1<sup>a</sup> aba e escrever, depois vai ativar a 2<sup>a</sup> aba e escrever, e assim por diante

```
Sub percorre_celulas()
    Dim celula As Range
    For Each celula In Range("G1:G10")
        celula.Value = "VBA"
    Next
End Sub
```

Vamos começar com um exemplo simples: escrever, em cada uma das células do intervalo G1 até G10, na aba “Exemplo, o texto “VBA”.

	E	F	G	H	I
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					

O código final é mostrado na imagem à esquerda. É importante você entender que a variável **celula** irá percorrer cada uma das células do intervalo especificado. Então, a cada Next, a variável **celula** se atualiza e recebe uma nova célula do intervalo Range("G1:G10"), de forma sequencial.

	A	B	C
1	Funcionário		
2			
3			
4			

O próximo exercício vai ser escrever o nome do funcionário que está utilizando o arquivo em cada uma das abas. Para isso, usaremos agora a estrutura de repetição **For Each** para percorrer cada uma das abas do arquivo e, na célula A2 de cada uma das abas, escrever o nome do funcionário.

```
Sub percorre_abas()
    Dim aba As Worksheet
    For Each aba In Sheets
        'Não basta usar o For Each para percorrer todas as planilhas.
        'A cada loop devemos selecionar a aba antes de escrever na
        'célula A2, se não o VBA não saberá em que aba ele deve escrever.
        aba.Activate
        Range("A2").Value = "João Paulo"
    Next
End Sub
```

O código final é mostrado na imagem à esquerda. Em primeiro lugar, repare que usamos o comando **For Each aba in Sheets**. **O Sheets é um comando que permite o VBA entender todo o conjunto de abas de todos os arquivos que estiverem abertos**, assim, a cada Loop, a variável **aba** vai se atualizar de acordo com a próxima aba do conjunto Sheets, de todos os arquivos abertos. Isto é bem ruim porque, se tivermos outros arquivos abertos mas que não possuem relação com o nosso arquivo atual, ele irá preencher o nome do funcionário em todas as abas de todos os arquivos abertos.

```
Sub percorre_abas()
    Dim aba As Worksheet
    For Each aba In ThisWorkbook.Sheets
        'Não basta usar o For Each para percorrer todas as planilhas.
        'A cada loop devemos selecionar a aba antes de escrever na
        'célula A2, se não o VBA não saberá em que aba ele deve escrever.

        aba.Activate
        Range("A2").Value = "João Paulo"
    Next
End Sub
```

Resolvemos esse problema facilmente usando o comando:

### ThisWorkbook.Sheets

Este **ThisWorkbook** faz com que o VBA entenda que queremos executar uma ação apenas para Este Arquivo (fazendo uma tradução literal). Assim, esta linha de código fará com que apenas as abas (Sheets) do arquivo atual (ThisWorkbook) sejam modificadas durante o For Each.

Outro ponto importante é que, antes de escrever na célula, é necessário ativar a aba para que o VBA escreva corretamente o texto em cada aba.

Após executar este código, cada célula A2 das abas do arquivo terão o nome “João Paulo” escrito nelas.

	A	B	C	D	E	F	G
1	Descrição	Unidades,Litros,Perda,Origem					
2	SKU 1,	785,981,157,Nacional					
3	SKU 2,	558,698,112,Importado					
4	SKU 3,	851,1064,170,Nacional					
5	SKU 4,	587,734,117,Nacional					
6	SKU 5,	396,495,79,Importado					
7	SKU 6,	113,141,23,Nacional					
8	SKU 7,	364,455,73,Nacional					
9	SKU 8,	1000,1250,200,Nacional					
10	SKU 9,	607,759,121,Nacional					
11	SKU 10,	135,169,27,Importado					
12	SKU 11,	463,579,93,Nacional					
13	SKU 12,	853,1066,171,Nacional					
14	SKU 13,	629,786,126,Importado					
15	SKU 14,	649,811,130,Importado					
16	SKU 15,	791,989,158,Importado					
17	SKU 16,	153,191,31,Nacional					
18	SKU 17,	593,741,119,Nacional					
19	SKU 18,	564,705,113,Nacional					
20	SKU 19,	207,259,41,Nacional					
21	SKU 20,	530,663,106,Importado					
22	SKU 21,	316,395,63,Nacional					
23	SKU 22,	882,1103,176,Nacional					
24	SKU 23,	237,296,47,Nacional					
25	SKU 24,	116,145,23,Importado					

No próximo exercício vamos corrigir um problema bem comum. No arquivo em questão, se você verificar cada aba, vai perceber que todas as tabelas estão desformatadas, ou seja, todas as informações estão concentradas na primeira coluna, e cada informação, nessa mesma coluna, está separada por vírgula.

Isso costuma acontecer quando extraímos informações de outros sistemas, e quando abrimos o arquivo no Excel, ele não vem no formato de tabela.

Portanto, o que vamos fazer é aprender como corrigir esse problema e, além disso, replicar essa correção para todas as abas do arquivo, tornando esse ajuste das tabelas algo automático.

The screenshot shows a Microsoft Excel interface with the 'Desenvolvedor' tab selected in the ribbon. A 'Gravar macro' dialog box is open, prompting the user to name the macro ('Nome da macro:'), assign a keyboard shortcut ('Tecla de atalho:'), and choose a location to store the macro ('Armazenar macro em:'). The macro name is set to 'texto\_para\_colunas'. The Excel spreadsheet below the ribbon contains 17 rows of data, each with a unique SKU number, its description ('Unidades,Litros,Perda,Origem'), and other details like 'Nacional' or 'Importado'.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Descrição	Unidades,Litros,Perda,Origem										
2	SKU 1,785,981,157,Nacional											
3	SKU 2,558,698,112,Importado											
4	SKU 3,851,1064,170,Nacional											
5	SKU 4,587,734,117,Nacional											
6	SKU 5,396,495,79,Importado											
7	SKU 6,113,141,23,Nacional											
8	SKU 7,364,455,73,Nacional											
9	SKU 8,1000,1250,200,Nacional											
10	SKU 9,607,759,121,Nacional											
11	SKU 10,135,169,27,Importado											
12	SKU 11,463,579,93,Nacional											
13	SKU 12,853,1066,171,Nacional											
14	SKU 13,629,786,126,Importado											
15	SKU 14,649,811,130,Importado											
16	SKU 15,791,989,158,Importado											
17	SKU 16,153,191,31,Nacional											

Para criar esse código, primeiro precisamos aprender sobre a ferramenta **Texto para Colunas** do Excel. Como de qualquer forma vamos precisar descobrir o código por trás, vamos começar gravando uma macro. Certifique-se de que você não está com a coluna A selecionada, pois vamos selecioná-la apenas após o início da gravação da macro.

Então, selecione por exemplo a célula A1, e agora sim vá na guia Desenvolvedor e em seguida clique em Gravar Macro.

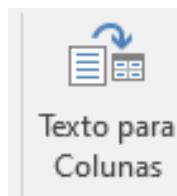
## Módulo 4 – Trabalhando com Objetos – Compilação For Each 1 (Resolução)

192

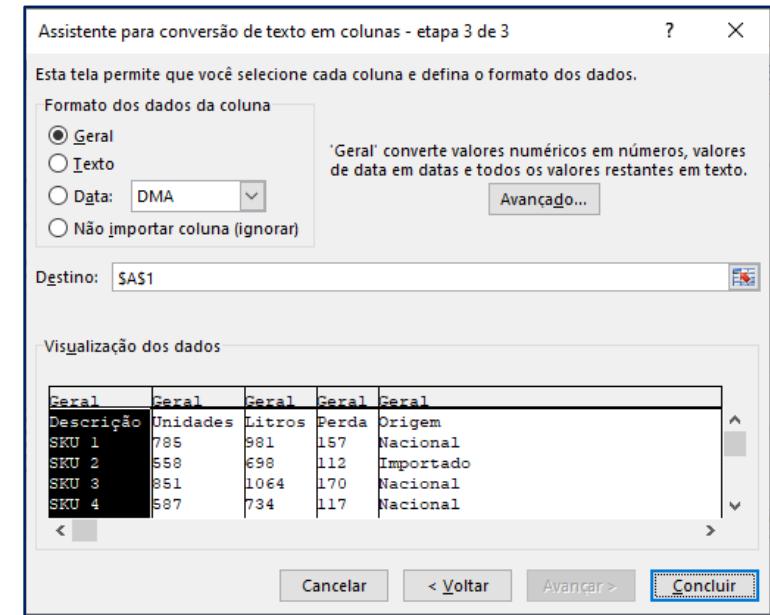
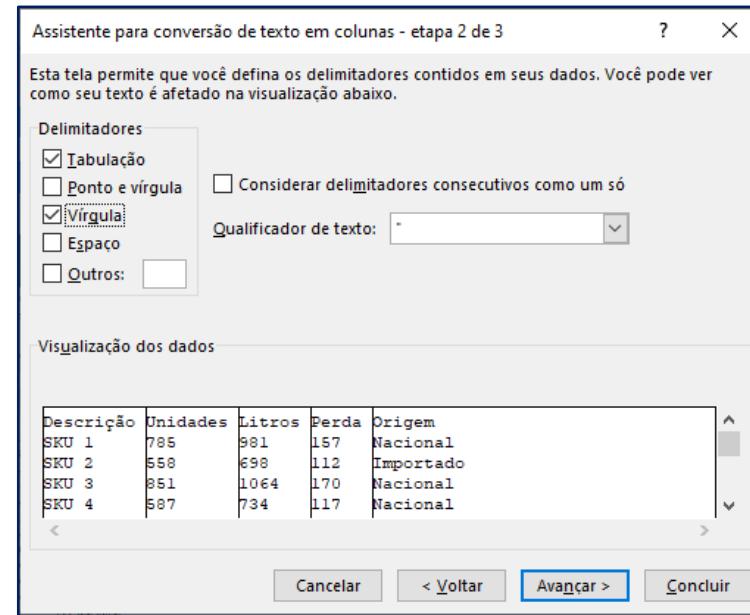
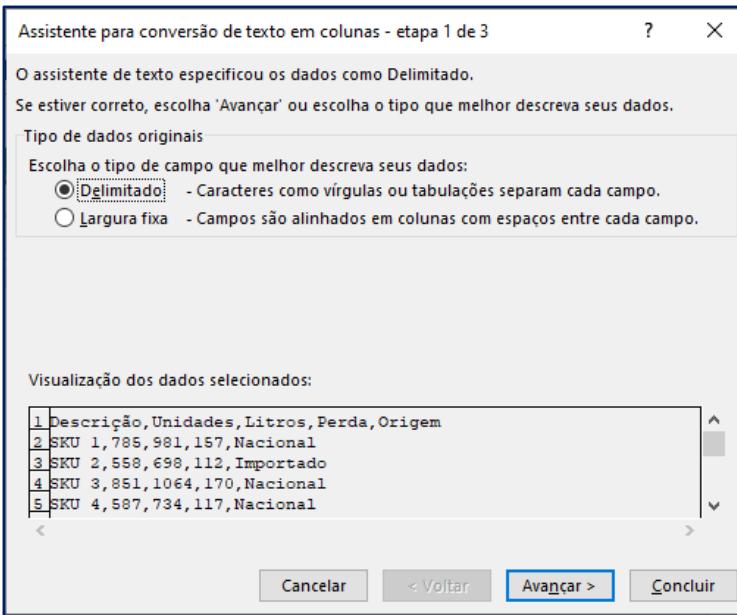
The screenshot shows an Excel spreadsheet titled "Exemplos For Each - Compilação 1.xlsxm - Excel". The "Dados" tab is active in the ribbon. A red arrow points to the "Texto para Colunas" icon in the "Classificar e Filtrar" group. The main window displays a list of data from row 1 to 25, with column A selected. A dialog box titled "Assistente para conversão de texto em colunas - etapa 1 de 3" is open, showing the configuration for importing delimited text. The "Delimitado" option is selected under "Tipo de dados originais". The "Visualização dos dados selecionados" section shows the first five rows of the data. Buttons at the bottom of the dialog are "Cancelar", "< Voltar", "Avançar >" (highlighted in blue), and "Concluir".

Iniciada a gravação de macro, execute o seguinte passo a passo:

1. Selecione a coluna A clicando na letra A referente à coluna.
2. Selecione a guia Dados.
3. Clique na opção **Texto para Colunas**.



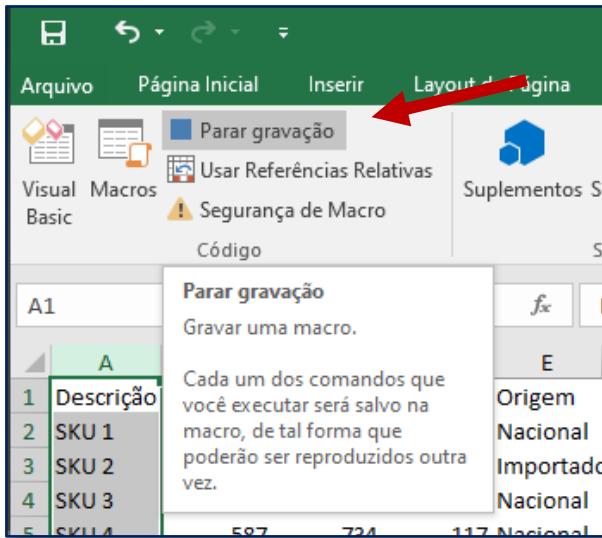
A janela ao lado irá aparecer pra você.



**Na primeira janela** que abriu, vamos escolher a opção “Delimitado”. Isso porque as nossas informações estão separadas (delimitadas) por vírgulas, então queremos separar uma nova coluna a cada vez que ele encontrar uma delimitação, no caso, a vírgula. Clique em **Avançar**.

**Na segunda janela**, vamos escolher o delimitador, que neste caso, é a vírgula. Se no seu caso fosse um outro delimitador que não aparece na lista de opções, você poderia marcar a caixa “Outros” e digitar ao lado qual seria esse delimitador. Clique em **Avançar**.

**Na terceira e última janela**, vemos que os dados foram divididos corretamente em 5 colunas. Para finalizar, clique em **Concluir**.



Por fim, não esqueça de clicar no botão de **Parar gravação**.

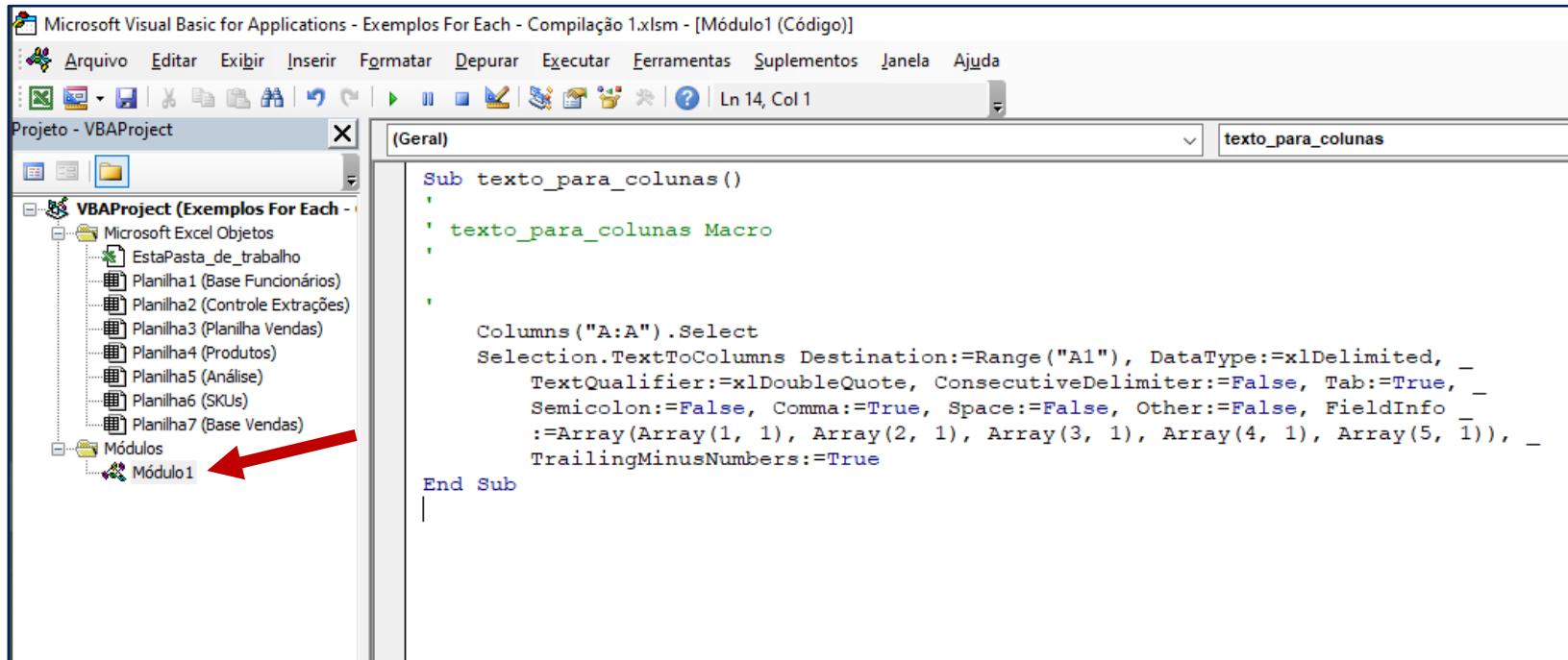
Além disso, não esqueça de desfazer a ação que acabamos de fazer, usando o atalho CTRL + Z. Isso porque queremos executar esse comando por meio da macro que vamos adaptar mais para frente.

Portanto, sua tabela deve voltar para a estrutura inicial, mostrada ao lado.

A	B	C	D	E	F	G	H	I
1	Descrição	Unidades,Litros,Perda,Origem						
2	SKU 1	785,981,157,Nacional						
3	SKU 2	2,558,698,112,Importado						
4	SKU 3	851,1064,170,Nacional						
5	SKU 4	587,734,117,Nacional						
6	SKU 5	396,495,79,Importado						
7	SKU 6	113,141,23,Nacional						
8	SKU 7	364,455,73,Nacional						
9	SKU 8	1000,1250,200,Nacional						
10	SKU 9	607,759,121,Nacional						
11	SKU 10	136,169,27,Importado						

## Módulo 4 – Trabalhando com Objetos – Compilação For Each 1 (Resolução)

195



The screenshot shows the Microsoft Visual Basic for Applications (VBA) interface. The title bar reads "Microsoft Visual Basic for Applications - Exemplos For Each - Compilação 1.xlsxm - [Módulo1 (Código)]". The menu bar includes Arquivo, Editar, Exibir, Inserir, Formatar, Depurar, Executar, Ferramentas, Suplementos, Janela, and Ajuda. The toolbar has various icons for file operations. The Project Explorer on the left shows the VBAProject (Exemplos For Each) with several worksheets like EstaPasta\_de\_trabalho, Planilha1, etc., and a Módulos folder containing "Módulo1". A red arrow points to the "Módulo1" entry. The code editor window displays the following VBA code:

```
Sub texto_para_colunas()
    ' texto_para_colunas Macro
    ' Columns("A:A").Select
    Selection.TextToColumns Destination:=Range("A1"), DataType:=xlDelimited,
        TextQualifier:=xlDoubleQuote, ConsecutiveDelimiter:=False, Tab:=True, _
        Semicolon:=False, Comma:=True, Space:=False, Other:=False, FieldInfo _:=Array(Array(1, 1), Array(2, 1), Array(3, 1), Array(4, 1), Array(5, 1)), _
        TrailingMinusNumbers:=True
End Sub
```

Para visualizar o código gravado, vamos até o ambiente VBA e damos um duplo clique em Módulo1.

```
Sub compila_abas()
    For Each aba In ThisWorkbook.Sheets
        aba.Activate
        Columns("A:A").Select
        Selection.TextToColumns Destination:=Range("A1"), DataType:=xlDelimited, _
            TextQualifier:=xlDoubleQuote, ConsecutiveDelimiter:=False, Tab:=True, _
            Semicolon:=False, Comma:=True, Space:=False, Other:=False, FieldInfo _
            :=Array(Array(1, 1), Array(2, 1), Array(3, 1), Array(4, 1), Array(5, 1)), _
            TrailingMinusNumbers:=True
    Next
End Sub
```

O código modificado está mostrado ao lado. Tudo o que fizemos foi criar uma nova Sub chamada `compila_abas`, e nela, colocamos o código do Texto para Colunas dentro de um For Each que percorre cada uma das abas.

Além disso, não podemos esquecer de ativar cada aba antes de executar os comandos do Texto para Colunas.

Ao executar esta macro, teremos todas as colunas divididas corretamente em todas as abas do nosso arquivo.

	A	B	C	D	E	F	G	H	I
1	Nome	Telefone	Área	Salário	Idade	Tempo de Empresa		Quantidade de Funcionários	0
2								Industrial	0
3								Administrativo	0
4								Comercial	0
5								Logística	0
6								Diretoria	0
7								Parcerias	0
8									
9									
10								Média de Salários	R\$ -
11								Total Folha Salarial	R\$ -
12								Tempo Médio de Empresa	0
13								Idade Média	0
14									
15									
16									
17									
18									
19									
20									
21									
22									
23									
24									
25									

Compilar Funcionários

No nosso próximo exercício de compilação, vamos criar um código muito útil para juntar dados de diferentes abas e agrupar em uma aba resumo.

No exemplo ao lado, temos várias abas com informações dos funcionários de cada aba, e temos uma aba resumo chamada “Resumo Funcionários” onde queremos juntar todos os funcionários de todas as áreas, que estão divididas ao longo das abas. Nesta aba também será possível fazer algumas análises como:

1. Quantidade de funcionários por área.
2. Média de salários.
3. Total de gastos com salário.
4. Tempo médio de empresa.
5. Idade média dos funcionários.

	A	B	C	D	E	F	G	H	I
1	Nome	Telefone	Área	Salário	Idade	Tempo de Empresa		Quantidade de Funcionários	0
2							Industrial	0	
3							Administrativo	0	
4							Comercial	0	
5							Logística	0	
6							Diretoria	0	
7							Parcerias	0	
8									
9									
10							Média de Salários	R\$ -	
11							Total Folha Salarial	R\$ -	
12							Tempo Médio de Empresa	0	
13							Idade Média	0	
14									
15									
16									
17									
18									
19									
20									
21									
22									
23									
24									
25									

Compilar Funcionários



Caso queira pular para a solução, o código com a solução final completa é mostrado na página 204.

Para resolver o desafio de compilação serão necessárias basicamente duas estruturas:

Primeiro, o nosso conhecido For Each, para percorrer cada uma das abas, copiar as informações dessa aba e colar na aba “Resumo Funcionários”. Porém, não queremos copiar informações de todas as abas do arquivo, somente a partir da aba “Resumo Funcionários”. Por isso, para verificar se estamos em uma aba onde devemos copiar os dados (ou seja, na aba diferente de “Resumo Funcionários”) devemos copiar as informações.

Se você quiser, pode tentar resolver este exercício antes de começarmos a solução.

```
Sub compila_funcionarios()
    For Each aba In ThisWorkbook.Sheets
        If aba.Name <> "Resumo Funcionarios" Then
            |
        End If
    Next
End Sub
```

No ambiente VBA, começamos a escrever o nosso código. Devemos entender que, como queremos executar uma determinada ação (copiar valores de uma aba e colar na “Resumo Funcionarios”) repetidas vezes, precisamos começar o nosso código com uma estrutura de repetição, no caso, um For Each.

Além disso, temos que pensar em quais abas queremos executar aquela ação de copiar valores. No caso, queremos em todas, exceto na aba “Resumo Funcionarios”, onde na verdade precisamos colar aquelas informações.

Assim, utilizamos uma estrutura If para testar o nome de cada aba e só vamos executar o código de copiar as informações se o nome da aba for diferente de “Resumo Funcionarios”.

```
Sub compila_funcionarios()

For Each aba In ThisWorkbook.Sheets

    If aba.Name <> "Resumo Funcionarios" Then

        aba.Activate

        ult_linha = Range("A1").End(xlDown).Row

        Range("A2:F" & ult_linha).Copy

    End If

Next

End Sub
```

Prosseguindo com o nosso código, temos que pensar qual é o passo a passo que deve ser feito a cada loop do For Each. Considerando que a aba em questão é de fato uma aba de onde queremos copiar as informações, devemos, entes de mais nada, ativar aquela aba para ai sim copiar os dados.

Além disso, para copiar sempre até a última linha da tabela, devemos usar aquela nossa variável **ult\_linha** para descobrir a última linha preenchida da tabela (se estiver em dúvida quanto a estrutura End(xlDown), retorne para a explicação na página 118).

Feito isso, concatenamos essa variável com o texto “A2:F” para considerar todas as células do intervalo da célula A2 até a última célula preenchida na coluna F, que é até onde a nossa tabela vai.

Para copiar o intervalo em questão, usamos a propriedade .Copy.

```
Sub compila_funcionarios()
    For Each aba In ThisWorkbook.Sheets
        If aba.Name <> "Resumo Funcionarios" Then
            aba.Activate
            ult_linha = Range("A1").End(xlDown).Row
            Range("A2:F" & ult_linha).Copy
            Sheets("Resumo Funcionarios").Activate
        End If
    Next
End Sub
```

Após copiar o intervalo, o que devemos fazer? Se queremos colar na aba “Resumo Funcionarios”, então precisamos, antes de mais nada, selecionar esta aba com o comando .Activate (ou .Select, tanto faz).

```
Sub compila_funcionarios()

For Each aba In ThisWorkbook.Sheets

    If aba.Name <> "Resumo Funcionarios" Then

        aba.Activate

        ult_linha = Range("A1").End(xlDown).Row

        Range("A2:F" & ult_linha).Copy

        Sheets("Resumo Funcionarios").Activate

        linha_registro = Range("A100000").End(xlUp).Row + 1

        Range("A" & linha_registro).PasteSpecial|
```

End If

Next

End Sub

Após ativar a aba “Registro Funcionarios”, precisamos encontrar qual é a última linha preenchida para então colar os dados logo abaixo desta linha. Não podemos usar o comando End(xlDown) a partir da célula A1 pois, inicialmente, esta aba vai estar sem nenhum dado a partir da linha 2. Assim, se dermos um CTRL + Seta para baixo o Excel nos levará para a última linha da planilha, o que não é o que queremos.

Assim, para contornar este problema, em vez de usar o comando xlDown a partir da célula A1, fazemos o contrário: vamos usar o comando xlUp a partir de uma célula bem lá embaixo na nossa planilha (por exemplo, A100000), para encontrar a última linha preenchida.

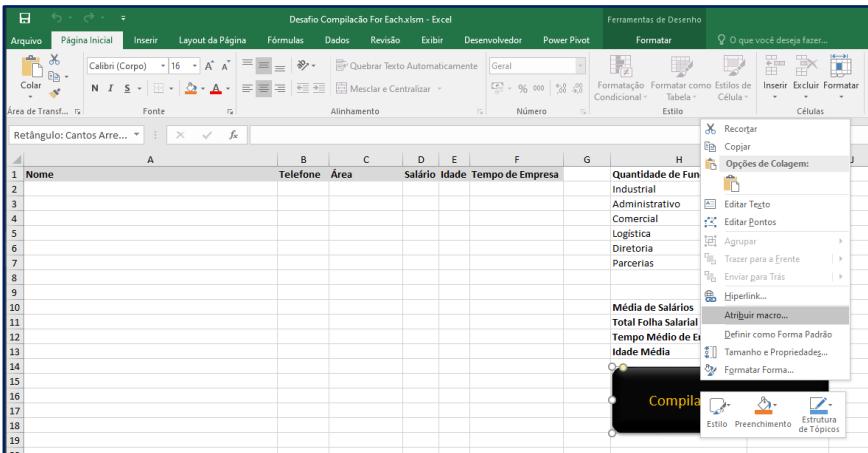
Este foi o mesmo comando utilizado e explicado no exercício da página 139. Então, se você não está lembrado ou ficou com alguma dúvida, volte nesta página para entender a lógica.

Após encontrar a última linha preenchida, não queremos colar nela e sim na linha logo abaixo. Por isso o **+1**.

Por fim, queremos colar a partir da célula na coluna A, na linha referente à variável linha\_registro. Para isso, usamos a propriedade PasteSpecial (não existe apenas Paste).

## Módulo 4 – Trabalhando com Objetos – Compilação For Each 2 (Parte 2)

203



Por fim, você pode atribuir esta macro ao botão da planilha (botão direito → atribuir macro) e executá-la. Você verá que ela funcionará corretamente.

Porém, ao final da execução, repare que estamos na linha 992. Isso porque ele realizou o processo de colar na aba várias vezes, até colar o último intervalo referente à aba de Parcerias. O ideal seria que, ao final da execução, voltássemos para o topo da tabela.

A	B	C	D	E	F
974 Vivian Martins	975651272	Logística	2971	41	19
975 Mateus Pereira	931489485	Logística	1844	41	14
976 Luiz Tiradentes	917130026	Logística	1005	36	18
977 Pedro Costa Garcia da Silva	916853407	Logística	1947	34	10
978 Lucas de Albuquerque Maranhão	989658935	Logística	3068	50	12
979 Matheus Soares	957088473	Logística	3053	36	12
980 Nathalie Oliveira	996403695	Logística	3087	48	13
981 Maria Jorge	946069975	Logística	1942	45	12
982 Paulo Miehrig	997230237	Logística	2652	47	11
983 Breno Moreira	965953086	Logística	1332	39	16
984 Amanda Florim	923083419	Logística	3429	44	20
985 Diogo Guimarães	936409155	Logística	3170	43	19
986 Ana de Souza	936865238	Diretoria	45976	48	16
987 Rafael Gondales Garcia	901557493	Diretoria	45227	36	20
988 Vivian Barcellos	925060709	Diretoria	22598	38	13
989 Giovana Lobo	967447157	Diretoria	35729	36	14
990 Felipe Frossard	913928625	Diretoria	41424	35	11
991 Daniella da Silva	968421594	Diretoria	18033	35	10
992 Naiara Short Santa Cecilia	926230166	Parcerias	3353	42	10
993 Letícia Tavares	969379359	Parcerias	4223	41	14
994 Maria Coutinho Beltrao	967447573	Parcerias	4999	50	11
995 Cinthia Bezerra	921595516	Parcerias	1996	37	15
996 Erick Cozendey	998241170	Parcerias	4819	41	18
997 Ana Souto do Vabo	908850917	Parcerias	2819	43	11
998 Camila Soares	949761735	Parcerias	4768	31	11

```
Sub compila_funcionarios()
    Range("A2:F100000").ClearContents 
    For Each aba In ThisWorkbook.Sheets
        If aba.Name <> "Resumo Funcionarios" Then
            aba.Activate
            ult_linha = Range("A1").End(xlDown).Row
            Range("A2:F" & ult_linha).Copy
            Sheets("Resumo Funcionarios").Activate
            linha_registro = Range("A100000").End(xlUp).Row + 1
            Range("A" & linha_registro).PasteSpecial
        End If
    Next
    Range("A1").Select 
End Sub
```

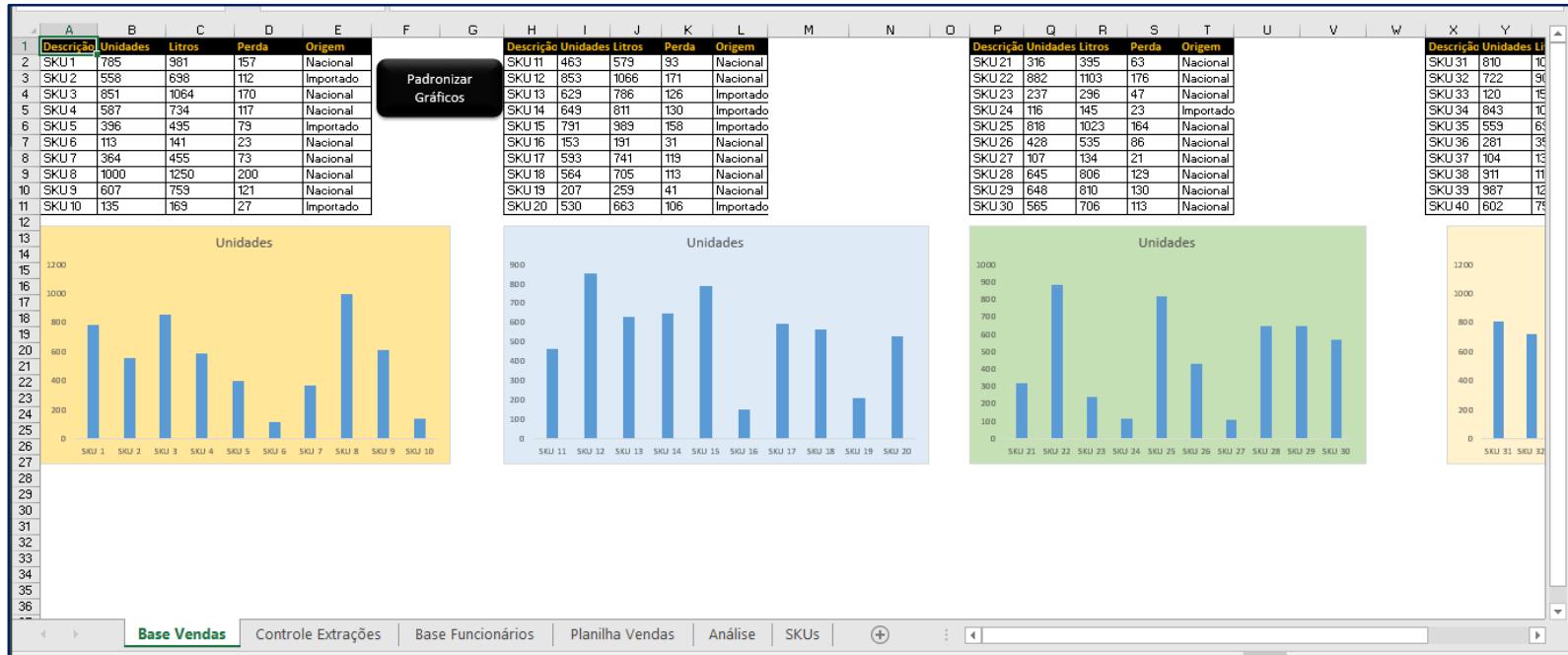
Para fazer com que o código mostre o topo da tabela, basta inserir, ao final do código, o comando:

Range("A1").Select

Além disso, é importante que, antes de executar a nossa macro de compilação, a gente exclua todos os valores anteriores que possam estar escritos na aba “Base Funcionarios”, para que seja possível sempre adicionar os dados atualizados.

Para isso, no começo do código, limpamos o conteúdo das células por meio da propriedade .ClearContents. Para facilitar essa exclusão de valores, podemos limpar o intervalo que vai da célula A2 até uma célula bem lá embaixo da planilha, por exemplo, a F100000.

Com estes dois ajustes, deixamos o nosso código 100% ainda mais automático.



Caso queira pular para a solução, o código com a solução final completa é mostrado na página 210.

No próximo exercício, vamos executar uma sequência de ações que vão percorrer não só todas as abas do arquivo, mas também todos os gráficos de cada aba.

A ideia aqui é a seguinte: temos vários relatórios ao longo das abas com diferentes gráficos. Porém, estes gráficos não seguem um padrão de formatação. Isso para apresentar em um relatório é ruim, pois a empresa deve seguir um padrão de formatação por questões de organização.

Assim, com a macro que faremos a seguir, será possível configurar a formatação de todos os gráficos ao mesmo tempo, padronizando configurações como cor, tamanho, etc.

The diagram illustrates the workflow for recording a macro to standardize charts in Excel:

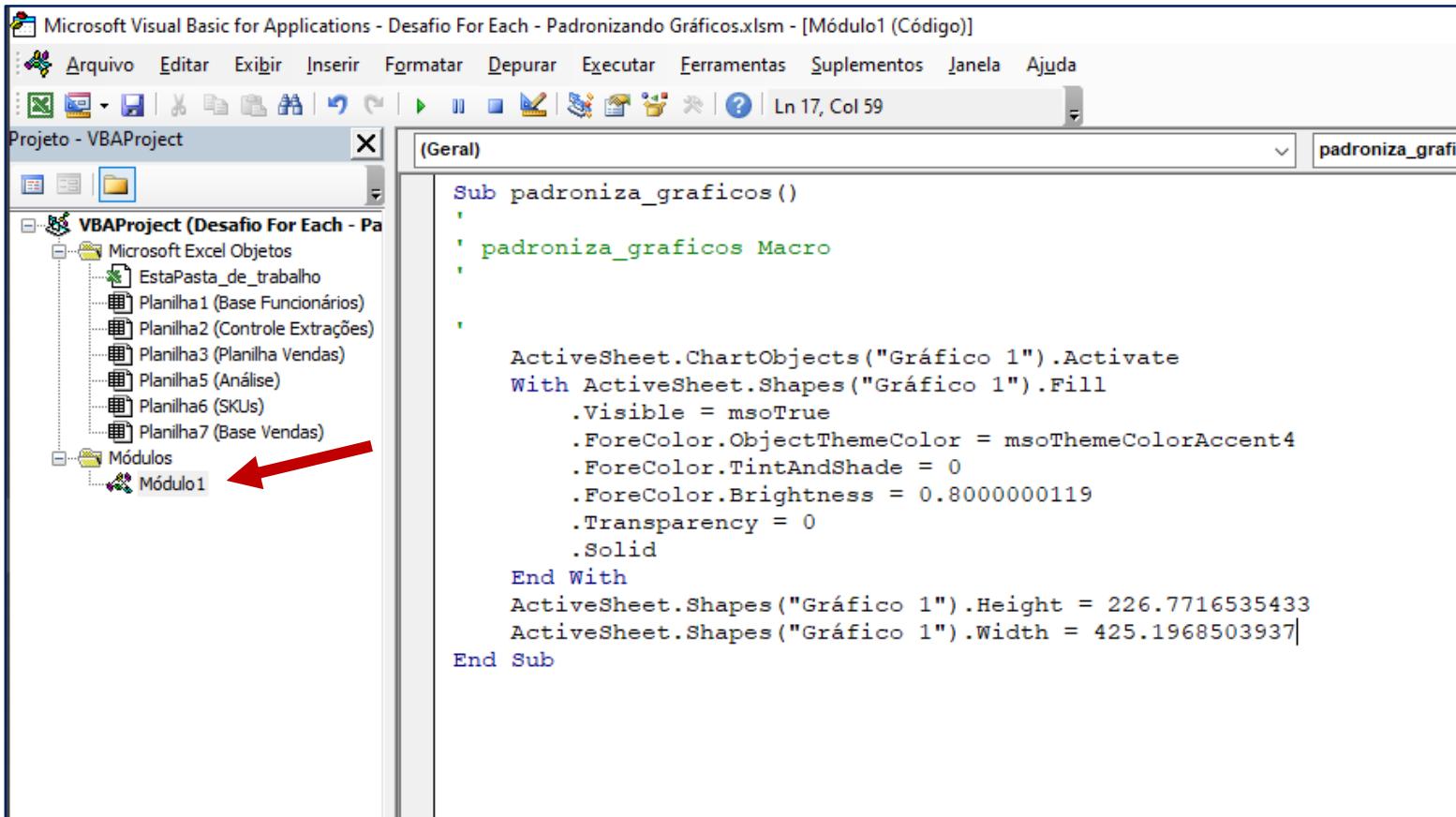
- Macro Recording:** The process begins with the "Gravar Macro" (Record Macro) button in the "Visual Basic" ribbon tab.
- Macro Definition:** A "Gravar macro" dialog box appears, prompting for a macro name ("padroniza\_graficos"), a keyboard shortcut ("Ctrl+ "), and a save location ("Esta pasta de trabalho").
- Chart Selection:** A bar chart titled "Gráfico 1" is selected on a worksheet. The chart displays data from a table with columns: Descrição, Unidades, Litros, and Perda. The chart has a blue color scheme.
- Format Tab:** The "Format" tab of the ribbon is active, showing the "Design" tab selected. The "Tamanho" (Size) group is highlighted, showing width set to 8 cm and height set to 15 cm.
- Color Change:** The "Format" tab remains active, and the "Cor" (Color) palette is used to change the chart's fill color to orange.
- Macro Stopping:** The "Format" tab remains active, and the "Visual Basic" ribbon tab is shown again, with the "Parar gravação" (Stop Recording) button highlighted.

A large yellow arrow points from the initial macro recording step through the chart selection and format changes to the final stopping of the macro.

**A sequência de etapas é mostrada. Após iniciar a gravação de macro, vamos:**

1. Mudar a cor do gráfico.
2. Alterar a altura para 8 cm e a largura para 15 cm.

**Não esqueça de Parar a gravação.**



The screenshot shows the Microsoft Visual Basic for Applications (VBA) editor interface. The title bar reads "Microsoft Visual Basic for Applications - Desafio For Each - Padronizando Gráficos.xlsx - [Módulo1 (Código)]". The menu bar includes Arquivo, Editar, Exibir, Inserir, Formatar, Depurar, Executar, Ferramentas, Suplementos, Janela, and Ajuda. The toolbar has various icons for file operations. The Project Explorer on the left shows a VBAProject named "Desafio For Each - Pa" containing "Microsoft Excel Objetos" (with "EstaPasta\_de\_trabalho" expanded) and "Módulos" (with "Módulo1" selected). The code editor window displays the following VBA code:

```
Sub padroniza_graficos()
    ' padroniza_graficos Macro

    ActiveSheet.ChartObjects("Gráfico 1").Activate
    With ActiveSheet.Shapes("Gráfico 1").Fill
        .Visible = msoTrue
        .ForeColor.ObjectThemeColor = msoThemeColorAccent4
        .ForeColor.TintAndShade = 0
        .ForeColor.Brightness = 0.8
        .Transparency = 0
        .Solid
    End With
    ActiveSheet.Shapes("Gráfico 1").Height = 226.7716535433
    ActiveSheet.Shapes("Gráfico 1").Width = 425.1968503937
End Sub
```

No módulo 1 temos o nosso código. A ideia é, a partir dele, construir a nossa estrutura de repetição para utilizá-lo diversas vezes para formatar cada um dos gráficos das nossas abas.

Dessa vez, apenas por simplificação, vamos escrever nosso código em cima do próprio `padroniza_graficos`, para você ver que também é possível alterar na própria macro gravada.

```
Sub padroniza_graficos()
'
' padroniza_graficos Macro
'

For Each aba In ThisWorkbook.Sheets
    aba.Activate

    ActiveSheet.ChartObjects("Gráfico 1").Activate
    With ActiveSheet.Shapes("Gráfico 1").Fill
        .Visible = msoTrue
        .ForeColor.ObjectThemeColor = msoThemeColorAccent4
        .ForeColor.TintAndShade = 0
        .ForeColor.Brightness = 0.8000000119
        .Transparency = 0
        .Solid
    End With
    ActiveSheet.Shapes("Gráfico 1").Height = 226.7716535433
    ActiveSheet.Shapes("Gráfico 1").Width = 425.1968503937

Next
End Sub
```

Do jeito que está, apenas o “Gráfico 1” será formatado.



Primeiro, começamos inserindo a estrutura de repetição For Each para percorrer cada uma das abas e executar a sequência de código referente a formatação do gráfico. Lembrando que, não basta percorrer as abas, precisamos ativá-las antes de executar os comandos. Por isso o **aba.Activate**.

Porém, apenas esta estrutura de repetição não é suficiente, pois o que ela vai fazer é simplesmente alterar o gráfico 1 de cada aba, e não todos os gráficos de cada aba, que é o que queremos.

Portanto, além da estrutura de repetição para percorrer cada uma das abas, também será necessária uma estrutura de repetição para percorrer cada um dos gráficos.

```
Sub padroniza_graficos()
    ' padroniza_graficos Macro
    '
    '
    For Each aba In ThisWorkbook.Sheets
        aba.Activate
        For Each grafico In ActiveSheet.ChartObjects
            ActiveSheet.ChartObjects(grafico.Name).Activate
            With ActiveSheet.Shapes(grafico.Name).Fill
                .Visible = msoTrue
                .ForeColor.ObjectThemeColor = msoThemeColorAccent4
                .ForeColor.TintAndShade = 0
                .ForeColor.Brightness = 0.8000000119
                .Transparency = 0
                .Solid
            End With
            ActiveSheet.Shapes(grafico.Name).Height = 226.7716535433
            ActiveSheet.Shapes(grafico.Name).Width = 425.1968503937
        Next
    Next
End Sub
```

Para percorrer cada gráfico daquela aba, precisamos de uma outra estrutura For Each. Dessa vez, uma variável **grafico** irá percorrer cada um dos gráficos em ActiveSheet.ChartObjects.

Além disso, não podemos deixar o nome fixo “Gráfico 1” aparecendo em cada uma das linhas onde ele aparece. Para tornar automático para considerar cada um dos gráficos daquela aba, substituímos “Gráfico 1” por **grafico.Name**.

( "Gráfico 1" ) → ( **grafico.Name** )

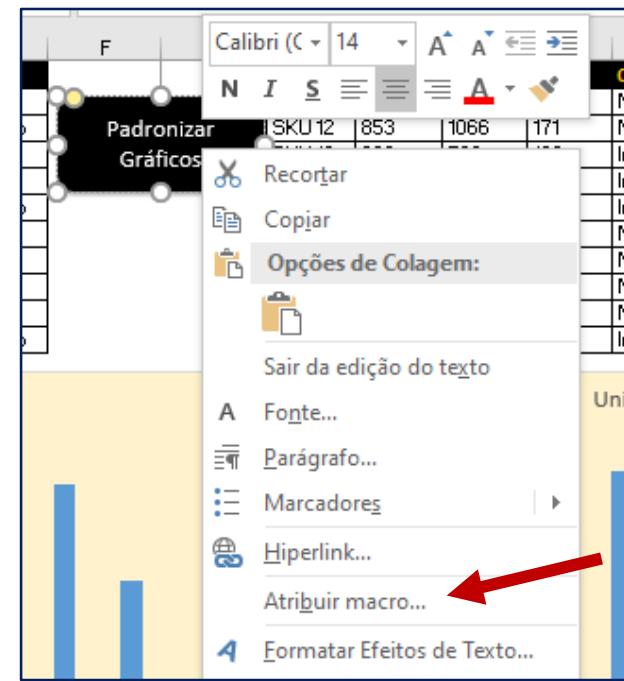
```
Sub padroniza_graficos()
    ' padroniza_graficos Macro
    '
    '
    For Each aba In ThisWorkbook.Sheets
        aba.Activate

        For Each grafico In ActiveSheet.ChartObjects

            ActiveSheet.ChartObjects(grafico.Name).Activate
            With ActiveSheet.Shapes(grafico.Name).Fill
                .Visible = msoTrue
                .ForeColor.ObjectThemeColor = msoThemeColorAccent4
                .ForeColor.TintAndShade = 0
                .ForeColor.Brightness = 0.8000000119
                .Transparency = 0
                .Solid
            End With
            ActiveSheet.Shapes(grafico.Name).Height = 226.7716535433
            ActiveSheet.Shapes(grafico.Name).Width = 425.1968503937

        Next
    Next
End Sub
```

O resultado final está mostrado no print ao lado. O código já está pronto para ser executado. Para isso, você pode atribuir esta macro ao botão da planilha e executar para ver o resultado final em cada uma dos gráficos de todas as abas do arquivo,



## Módulo 5

# Fórmulas de Texto e Data no VBA

Fórmulas de texto são fórmulas utilizadas para fazer tratamentos de texto. Assim como no Excel, também temos fórmulas no VBA que tratam de textos de uma maneira muito semelhante. Em geral, utilizamos fórmulas de textos para fazer correções ou manipulações em textos que não vem na estrutura correta par se trabalhar no Excel.

A imagem abaixo resume as principais funções de texto, com as respectivas aplicações e objetivos.

Principais Funções de Texto

Função:	Exemplo:	Resultado:	Descrição:
Operador &	"Nome" & ":" & " Pedro"	Nome: Pedro	concatenação
InStr	InStr ("carro","a")	2	retorna a posição de um conjunto de caracteres
Left	Left ("carro",2)	ca	retorna uma string a partir do corte pela esquerda de acordo com o nº de caracteres
Right	Right ("carro",2)	ro	retorna uma string a partir do corte pela direita de acordo com o nº de caracteres
Mid	Mid ("carro",2)	arro	corta a string a partir da esquerda, incluindo o caractere indicado
Len	Len ("carro")	5	retorna o número de caracteres de uma string
UCase	Ucase ("carro")	CARRO	convertem os caracteres para letras maiúsculas
LCase	LCase ("CARRO")	carro	convertem os caracteres para letras minúsculas

	B	C	D
2	Produto	Valor de Venda	Código
3	PA303001 - Guaraná Copo	R\$ 808,00	
4	PA323002 - Água Copo	R\$ 1.971,00	
5	PA334001 - Mate Limão Garrafa	R\$ 1.489,00	
6	PA314003 - Guaraná com Açaí Garrafa	R\$ 1.901,00	
7	PA307869 - Coca Cola 2L	R\$ 1.270,00	
8	PA348372 - Coca Cola Lata	R\$ 1.962,00	
9	PA384739 - Suco de Laranja Caixa	R\$ 2.012,00	
10	PA378274 - Coca Cola 1L	R\$ 879,00	
11	PA384736 - Mate Copo	R\$ 719,00	
12			
13			

Fórmulas de Texto & Data no VBA    **Fórmulas de Texto - Parte 1**    Fórmulas de Texto - Parte 2

Na primeira aplicação de fórmulas de texto, vamos utilizar a fórmula **Left**.

O objetivo desta fórmula é retornar uma quantidade de caracteres, a partir da **esquerda** de um texto. No caso, queremos, na coluna D, retornar apenas o código de cada um dos produtos na coluna B.

```
Sub produtos()
Cells(3, 4).Value = Left(Cells(3, 2).Value, 8)
End Sub
```



Produto	Valor de Venda	Código
PA303001 - Guaraná Copo	R\$ 808,00	PA303001
PA323002 - Água Copo	R\$ 1.971,00	

O código que irá retornar apenas os 8 primeiros caracteres mais à esquerda referentes ao código do produto está mostrado na imagem ao lado. A princípio, criamos uma estrutura que funciona apenas para a primeira linha da tabela.

```
Sub produtos()
For linha = 3 To 11
    Cells(linha, 4).Value = Left(Cells(linha, 2).Value, 8)
Next
End Sub
```



Produto	Valor de Venda	Código
PA303001 - Guaraná Copo	R\$ 808,00	PA303001
PA323002 - Água Copo	R\$ 1.971,00	PA323002
PA334001 - Mate Limão Garrafa	R\$ 1.489,00	PA334001
PA314003 - Guaraná com Açaí Garrafa	R\$ 1.901,00	PA314003
PA307869 - Coca Cola 2L	R\$ 1.270,00	PA307869
PA348372 - Coca Cola Lata	R\$ 1.962,00	PA348372
PA384739 - Suco de Laranja Caixa	R\$ 2.012,00	PA384739
PA378274 - Coca Cola 1L	R\$ 879,00	PA378274
PA384736 - Mate Copo	R\$ 719,00	PA384736

Para repetir o código acima para cada uma das linhas da tabela, utilizamos a estrutura For e incluímos a variável **linha** para atualizar a fórmula para cada linha no código da imagem anterior.



**Funções de Texto, Data e Hora NÃO precisam do comando WorksheetFuncion.**

	B	C	D	E
2	Cliente	Valor de Contrato	Estado	
3	João Martins - RJ	R\$ 59.000,00		
4	Hecio Gomes - RS	R\$ 37.000,00		
5	João Lira - SP	R\$ 63.000,00		
6	Diego Amorim - RJ	R\$ 51.000,00		
7	Allan Centurione - AM	R\$ 57.000,00		
8	Giovanna Menaged - BA	R\$ 48.000,00		
9	Izabelle Carvalho - GO	R\$ 74.000,00		
10	Julia Campos - DF	R\$ 83.000,00		
11	Luiza França - SP	R\$ 34.000,00		

Fórmulas de Texto & Data no VBA | Fórmulas de Texto - Parte 1 | **Fórmulas de Texto - Parte 2**

Na próxima aplicação vamos utilizar a fórmula **Right**.

O objetivo desta fórmula é retornar uma quantidade de caracteres, a partir da **direita** de um texto. Neste exercício, queremos retornar apenas a sigla do estado de cada cliente, no caso, sempre os últimos dois caracteres mais à direita.

```
Sub clientes()

For linha = 3 To 11

    Cells(linha, 4).Value = Right(Cells(linha, 2).Value, 2)

Next

End Sub
```



Cliente	Valor de Contrato	Estado
João Martins - RJ	R\$ 59.000,00	RJ
Hecio Gomes - RS	R\$ 37.000,00	RS
João Lira - SP	R\$ 63.000,00	SP
Diego Amorim - RJ	R\$ 51.000,00	RJ
Allan Centurione - AM	R\$ 57.000,00	AM
Giovanna Menaged - BA	R\$ 48.000,00	BA
Izabelle Carvalho - GO	R\$ 74.000,00	GO
Julia Campos - DF	R\$ 83.000,00	DF
Luiza França - SP	R\$ 34.000,00	SP

O código que irá retornar apenas os 2 primeiros caracteres mais à direita referentes à sigla do estado está mostrado na imagem ao lado.

Para repetir o código acima para cada uma das linhas da tabela, utilizamos a estrutura For e incluímos a variável **linha** para atualizar a fórmula para cada linha no código da imagem anterior.

	B	C	D
2	Produto	Valor de Venda	Descrição
3	PA303001 - Guaraná Copo	R\$ 808,00	
4	PA323002 - Água Copo	R\$ 1.971,00	
5	PA334001 - Mate Limão Garrafa	R\$ 1.489,00	
6	PA314003 - Guaraná com Açaí Garrafa	R\$ 1.901,00	
7	PA307869 - Coca Cola 2L	R\$ 1.270,00	
8	PA348372 - Coca Cola Lata	R\$ 1.962,00	
9	PA384739 - Suco de Laranja Caixa	R\$ 2.012,00	
10	PA378274 - Coca Cola 1L	R\$ 879,00	
11	PA384736 - Mate Copo	R\$ 719,00	
12			

Até aqui vimos fórmulas de texto que retornam sempre os caracteres mais à esquerda ou mais à direita de um texto.

Porém, ainda não saberíamos como extrair um texto que começa no meio da palavra.

No exemplo ao lado, queremos extrair apenas a descrição dos Produtos, sem o código. Ou seja, para o primeiro, queremos apenas o texto “Guaraná Copo”, para o segundo, apenas “Água Copo”, e por ai vai.

Neste caso, para extrair um texto que começa no meio da palavra, usamos a fórmula **Mid**.

```
Sub compila_descricoes()  
  
    For linha = 3 To 11  
  
        Cells(linha, 4).Value = Mid(Cells(linha, 2).Value, 12, 100)  
  
    Next|  
  
End Sub
```



Produto	Valor de Venda	Descrição
PA303001 - Guaraná Copo	R\$ 808,00	Guaraná Copo
PA323002 - Água Copo	R\$ 1.971,00	Água Copo
PA334001 - Mate Limão Garrafa	R\$ 1.489,00	Mate Limão Garrafa
PA314003 - Guaraná com Açaí Garrafa	R\$ 1.901,00	Guaraná com Açaí Garrafa
PA307869 - Coca Cola 2L	R\$ 1.270,00	Coca Cola 2L
PA348372 - Coca Cola Lata	R\$ 1.962,00	Coca Cola Lata
PA384739 - Suco de Laranja Caixa	R\$ 2.012,00	Suco de Laranja Caixa
PA378274 - Coca Cola 1L	R\$ 879,00	Coca Cola 1L
PA384736 - Mate Copo	R\$ 719,00	Mate Copo

O código que irá retornar a descrição está mostrado ao lado. Aqui, o padrão que vemos é que, todos os nomes dos produtos começam a partir do caractere 12. O que a fórmula Mid faz é permitir que a gente decida a partir de que caractere começa a extrair o texto. No caso do Left, é como se usássemos o Mid começando no caractere 1.

Em seguida, o terceiro argumento é a quantidade de caracteres que queremos a partir da posição inicial 12. Como queremos tudo até o final do texto, podemos colocar um valor bem alto porque assim não precisamos nos preocupar em contar a quantidade de caracteres de cada nome, dado que de qualquer forma queremos tudo até o final. Fique tranquilo que isso não irá adicionar espaços ou algo do tipo. Isso apenas vai fazer com que o Mid pegue todos os caracteres até o final.

	B	C	D	E	F
2	Funcionário	Faturamento Anual	Descrição (Em Letra Maiúscula)	Código	
3	P20 - Lira Batom	R\$ 279.000,00			
4	P450 - Sombras Ivo's	R\$ 292.000,00			
5	P273 - Maybelline	R\$ 423.000,00			
6	P5260 - Martins Shampoo	R\$ 494.000,00			
7	P27362 - Santo's Lotions	R\$ 332.000,00			
8	P6 - Sabom Paes Leme	R\$ 448.000,00			

Fórmulas de Texto - Parte 2

Fórmulas de Texto - Parte 3

**Fórmulas de Texto - Parte 4**

Fórmula: ...

Vamos dificultar um pouco agora. Na coluna E, queremos escrever apenas o código da coluna B.

Quando trabalhamos com fórmulas de texto, sempre devemos identificar um padrão:

1. No exercício Left, os códigos sempre começavam da esquerda e tinham 8 caracteres.
2. No exercício Right, as siglas dos estados estavam sempre nos últimos 2 caracteres da palavra.
3. No exercício Mid, a descrição sempre começava a partir da posição 12.

Com estes padrões identificados, foi fácil aplicar as fórmulas de texto.

	B	C	D	E
2	Funcionário	Faturamento Anual	Descrição (Em Letra Maiúscula)	Código
3	P20 - Lira Batom	R\$ 279.000,00		
4	P450 - Sombras Ivo's	R\$ 292.000,00		
5	P273 - Maybelline	R\$ 423.000,00		
6	P5260 - Martins Shampoo	R\$ 494.000,00		
7	P27362 - Santo's Lotions	R\$ 332.000,00		
8	P6 - Sabom Paes Leme	R\$ 448.000,00		
9				
10				
11	<a href="#">Fórmulas de Texto - Parte 2</a>	<a href="#">Fórmulas de Texto - Parte 3</a>	<a href="#">Fórmulas de Texto - Parte 4</a>	Fórmula: ... <span style="color: green;">+</span>
12				

Porém, nem sempre estes padrões são tão fáceis de identificar.

No exercício em questão, os códigos não seguem um padrão de tamanho: o primeiro código começa nos 3 primeiros caracteres, o segundo começa nos 4 primeiros caracteres, e assim vai.

Apesar de parecer uma situação difícil de resolver, aqui o padrão é um pouco mais sutil. Todas as palavras tem um “-” que separa o texto referente ao código do texto referente à descrição. Melhor ainda: o código sempre vai até 2 caracteres antes do “-”. Isso significa que encontrar a posição do “-” vai nos ajudar e muito a resolver esse problema.

```
Sub compila_produtos()
    For linha = 3 To 8
        posicao_traco = InStr(Cells(linha, 2).Value, "-")
        Cells(linha, 5).Value = Left(Cells(linha, 2).Value, posicao_traco - 2)
    Next
End Sub
```



Funcionário	Faturamento Anual	Descrição (Em Letra Maiúscula)	Código
P20 - Lira Batom	R\$ 279.000,00		P20
P450 - Sombras Ivo's	R\$ 292.000,00		P450
P273 - Maybelline	R\$ 423.000,00		P273
P5260 - Martins Shampoo	R\$ 494.000,00		P5260
P27362 - Santo's Lotions	R\$ 332.000,00		P27362
P6 - Sabom Paes Leme	R\$ 448.000,00		P6

A fórmula que retorna a posição do “-” é o **InStr**. Ela pede basicamente dois argumentos:

1. A célula que queremos localizar.
2. O caractere a ser procurado, neste caso, o “-”.

Feito isso, vamos utilizar a fórmula **Left** para retornar os caracteres mais à esquerda da célula. De acordo com o padrão que identificamos, o código de qualquer produto, independente do seu tamanho, sempre termina na posição do traço menos dois caracteres. Isso foi exatamente o que fizemos no nosso código, e o resultado é mostrado ao lado.

```
Sub compila_produtos()
    For linha = 3 To 8
        posicao_traco = InStr(Cells(linha, 2).Value, "-")
        Cells(linha, 4).Value = UCase(Mid(Cells(linha, 2).Value, posicao_traco + 2, 100))
    Next
End Sub
```



Funcionário	Faturamento Anual	Descrição (Em Letra Maiúscula)	Código
P20 - Lira Batom	R\$ 279.000,00	LIRA BATOM	P20
P450 - Sombras Ivo's	R\$ 292.000,00	SOMBRA'S IVO'S	P450
P273 - Maybelline	R\$ 423.000,00	MAYBELLINE	P273
P5260 - Martins Shampoo	R\$ 494.000,00	MARTINS SHAMPOO	P5260
P27362 - Santo's Lotions	R\$ 332.000,00	SANTO'S LOTIONS	P27362
P6 - Sabom Paes Leme	R\$ 448.000,00	SABOM PAES LEME	P6

Já para a descrição, vamos usar a fórmula Mid com o InStr, seguindo a mesma lógica usada anteriormente. Neste caso, o padrão para os nomes das descrições, que começam no meio do texto, é que elas começam sempre 2 caracteres depois da posição do traço.

Além disso, queremos colocar a descrição em letra maiúscula. Para isso, vamos usar o comando Ucase (lembmando que estas fórmulas de texto estão explicadas e exemplificadas na página 211).

O código final e o resultado na planilha são mostrados na imagem ao lado.

	B	C	D	E
2	Nome	Sobrenome	Nome Completo	
3	Alon José	Pinheiro		
4	Luiza Gomes	França		
5	João Paulo	Martins		
6	João Paulo	Lira		
7	Diego Santos	Amorim		
8	Sergio Lima	Tranjan		

[Fórmulas de Texto - Parte 3](#) [Fórmulas de Texto - Parte 4](#) [Fórmulas de Texto e Data](#)

No próximo exercício vamos fazer uma correção bem comum no Excel, que é a de eliminar os espaços a mais que estão aparecendo nas células de nomes da coluna B e C. Para isso, vamos usar a fórmula **Trim**.

Além disso, queremos juntar Nome e Sobrenome para escrever o nome completo de cada pessoa na coluna D.

```
Sub compila_produtos()
    For linha = 3 To 8
        nome = WorksheetFunction.Trim(Cells(linha, 2).Value)
        sobrenome = WorksheetFunction.Trim(Cells(linha, 3).Value)
        Cells(linha, 4).Value = nome & " " & sobrenome
    Next
End Sub
```



Nome	Sobrenome	Nome Completo
Alon José	Pinheiro	Alon José Pinheiro
Luiza Gomes	França	Luiza Gomes França
João Paulo	Martins	João Paulo Martins
João Paulo	Lira	João Paulo Lira
Diego Santos	Amorim	Diego Santos Amorim
Sergio Lima	Tranjan	Sergio Lima Tranjan

O código final está mostrado ao lado.

A fórmula Trim é a única fórmula de texto que precisa do comando WorksheetFunction para ser acessada.

Finalizamos concatenando a variável nome com a variável sobrenome por meio do e-comercial (&). Além disso, para os dois textos (nome e sobrenome) não ficarem colados, inserimos um espaço “ ” entre os dois.

O resultado final está mostrado ao lado.

Assim como no Excel, também temos fórmulas de data no VBA. Em essência, estas fórmulas tem como objetivo retornar o ano de uma data, o mês de uma data, a data de hoje, o horário atual, e por ai vai.

A tabela abaixo resume as principais fórmulas de data e hora que existem no VBA, com respectivos exemplos de utilização.

**Principais Funções de Data e Hora**

Função	Utilidade	Exemplo de Uso
Now()	Data e Hora Atuais	Cells(1, 1).Value = Now()
Date()	Data de hoje	Cells(1, 1).Value = Date()
Day()	Dia de uma data	Cells(1, 1).Value = Day(Date)
Month()	Mês de uma data	numero_mes = Month(Date)
Year()	Ano de uma data	Cells(1, 1).Value = Year(Date)
Hour()	Hora de um horário	qtde_horas = Hour(Now)
Minute()	Minutos de um horário	qtde_minutos = Minute(Now)
Second()	Segundos de um horário	qtde_segundos = Second(Now)

```
Sub datas()  
  
Cells(3, 6).Value = Date 'retorna a data atual  
  
Cells(3, 8).Value = Time 'retorna a hora atual em formato americano  
  
Cells(6, 6).Value = Now 'retorna data e hora atual  
  
End Sub
```

Vamos finalizar esta nossa aba preenchendo as informações de data e hora solicitadas. O código ao lado resume as 3 fórmulas de data e hora que utilizamos para esta situação. Repare que estas fórmulas não precisam de argumentos, elas são na verdade palavras chave que o VBA entende como funções de data e hora.



	B	C	D	E	F	G	H
2	Nome	Sobrenome	Nome Completo		Data da última Compilação	Horário da última Compilação	
3	Alon José	Pinheiro	Alon José Pinheiro		04/03/2020	5:48:55 PM	
4	Luiza Gomes	França	Luiza Gomes França				
5	João Paulo	Martins	João Paulo Martins				
6	João Paulo	Lira	João Paulo Lira				
7	Diego Santos	Amorim	Diego Santos Amorim				
8	Sergio Lima	Tranjan	Sergio Lima Tranjan				
9							
10							

Módulo 6

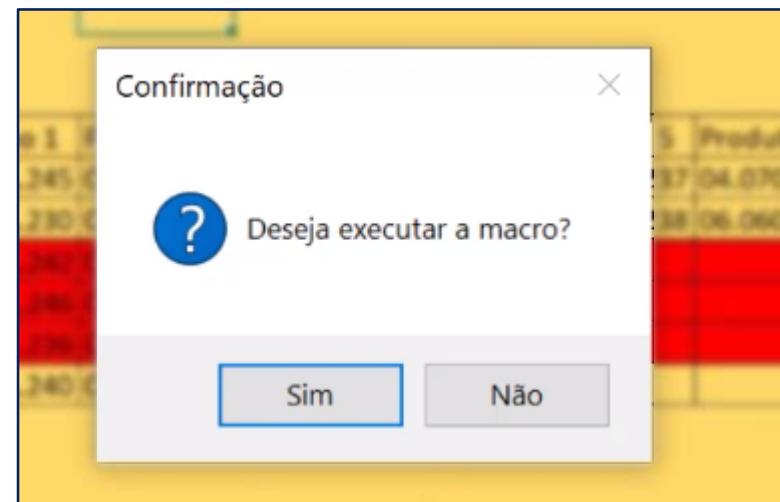
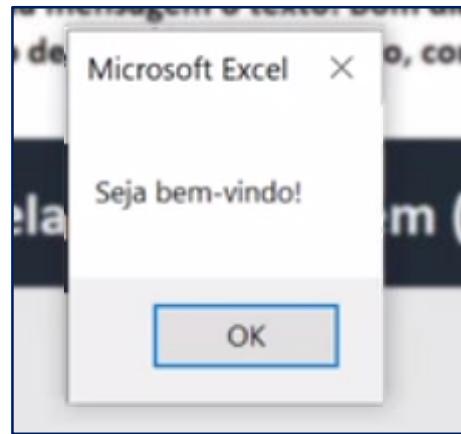
**Boxes:**

**Criando interações com  
o usuário**

Boxes no VBA são caixas de mensagem que surgem com algum objetivo específico. Um box pode ser uma caixa de mensagem para simplesmente emitir um aviso ao usuário, ou então uma mensagem que é executada com o objetivo de interagir com o usuário e, de acordo com a resposta, executar ou não uma determinada ação.

Abaixo temos dois exemplos de boxes: à esquerda, uma caixa de mensagem que envia o texto “Seja bem-vindo”.

Já à direita, um box que surge para perguntar ao usuário se o mesmo deseja executar uma determinada macro ou não.



A imagem abaixo resume a ideia por trás de um box.

Janela de Mensagem (MsgBox)	
Parâmetros:	<b>Finalidade:</b> Interface
Sintaxe:	<b>MsgBox</b>
	Exibir uma informação para o usuário
	<b>MsgBox texto, botões, título, helpfile, contexto</b>
	<b>Variável = (MsgBox texto, botões, título, helpfile, contexto)</b>
	<b>texto</b> - texto obrigatório com a mensagem a ser apresentada ao usuário
	<b>botões</b> - um número inteiro que indica quantos e quais botões irão aparecer na janela de mensagem
	<b>título</b> - título que aparece na caixa de diálogo (opcional)
	<b>helpfile e contexto</b> - dizem respeito à criação de um texto de ajuda associado a esta MsgBox

E a tabela abaixo resume quais botões podem ser exibidos no box e quais os valores de saída. Veremos passo a passo como utilizar estes comandos. Como sugestão, retorne a esta página após finalizar o módulo de boxes para entender melhor o significado destas opções.

Botões		
Valor:	Constante:	Descrição:
0	vbOkOnly	Exibe somente o botão OK
1	vbOkCancel	Exibe os botões OK e Cancelar
2	vbAbortRetryIgnore	Exibe os botões Abortar, Repetir e Ignorar
3	vbYesNoCancel	Exibe os botões Sim, Não e Cancelar
4	vbYesNo	Exibe os botões Sim e Não
5	vbRetryCancel	Exibe os botões Repetir e Cancelar
16	vbCritical	Exibe o ícone de mensagem crítica
32	vbQuestion	Exibe o ícone consulta de aviso
48	vbExclamation	Exibe o ícone mensagem de aviso
64	vbInformation	Exibe o ícone mensagem de informação
0	vbDefaultButton1	O primeiro botão é o padrão
256	vbDefaultButton2	O segundo botão é o padrão
512	vbDefaultButton3	O terceiro botão é o padrão
768	vbDefaultButton4	O quarto botão é o padrão

Valores de Saída		
Valor:	Constante:	Descrição:
1	vbOk	OK
2	vbCancel	Cancelar
3	vbAbort	Abortar
4	vbRetry	Repetir
5	vbIgnore	Ignorar
6	vbYes	Sim
7	vbNo	Não

The screenshot shows a portion of an Excel spreadsheet with the following content:

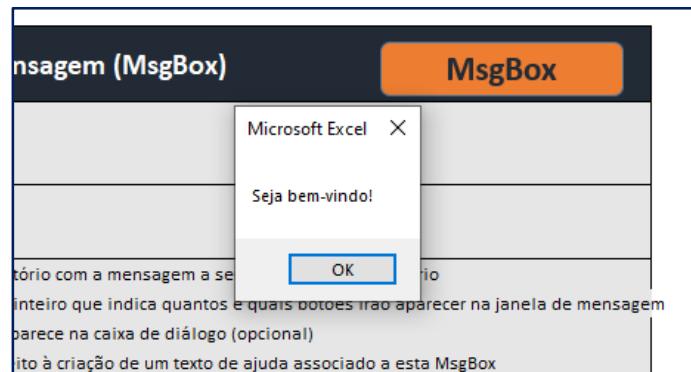
	B	C	D	E
2	1. Crie uma macro que exiba em uma mensagem para a pessoa que estiver usando a planilha com o texto: "Seja bem-vindo!"			
3	2. Crie uma macro que exiba uma mensagem com o nome do usuário na mensagem o texto: Bom dia, (nome)!			
4	3. Crie uma macro que exiba uma mensagem perguntando se o usuário deseja executar a macro, com botões de 'Sim' ou 'Não'.			
5	4. Personalize a MsgBox			
7	Finalidade: Interface:	Janela de Mensagem (MsgBox)	MsgBox	
8	Exibir uma informação para o usuário	<b>MsgBox</b>	Exercício MsgBox	Exercício Boxes

A ribbon tab labeled "MsgBox" is selected. Below the ribbon, there is a help section with the following text:

Finalidade: Interface:  
Exibir uma informação para o usuário  
MsgBox Exercício MsgBox InputBox Exercício Boxes Exercício Confirmar

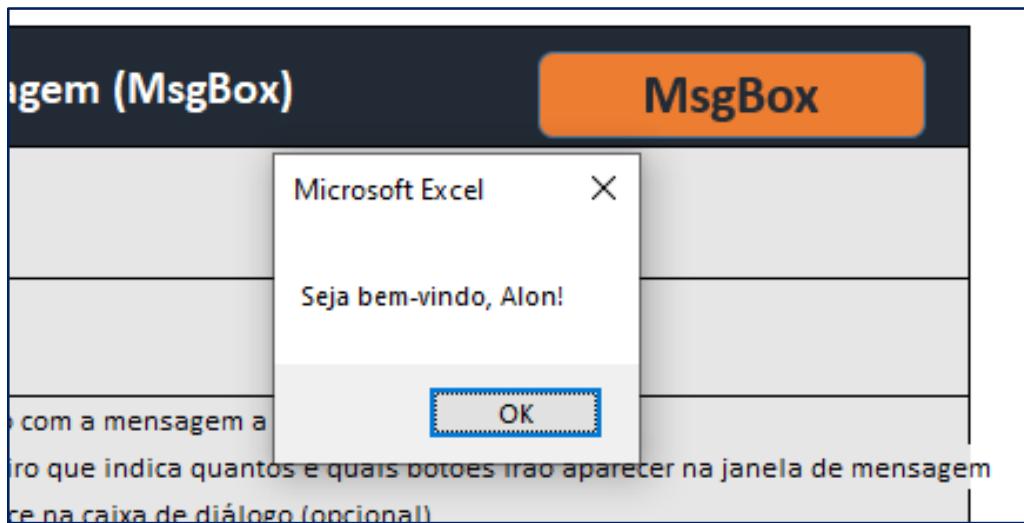
```
Sub mensagem()
    MsgBox ("Seja bem-vindo!")
End Sub
```

O código é bem simples e está mostrado ao lado. Você pode atribuir esta macro ao botão laranja “MsgBox” da aba e ao executar você vai receber a seguinte mensagem:

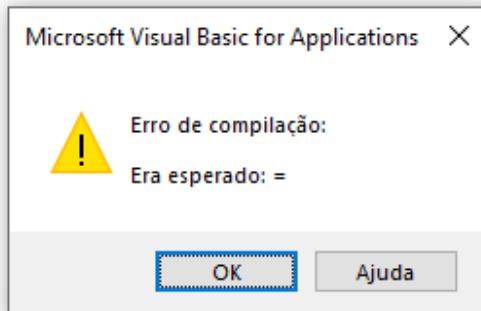


```
Sub mensagem()  
  
    MsgBox ("Seja bem-vindo, " & Application.UserName & "!")  
  
End Sub
```

Uma acréscimo que podemos fazer à MsgBox anterior é inserir o nome do usuário automaticamente através do comando Application.UserName, que irá escrever o nome do usuário do computador.



```
Sub mensagem()
    MsgBox ("Deseja executar a macro?", vbYesNo)
End Sub
```



No próximo exercício, vamos perguntar ao usuário se ele deseja executar a macro, além disso, vamos dar a ele as opções de botões “Sim” ou “Não” (vbYesNo).

Agora, como esperamos uma resposta do usuário (diferente do exercício anterior em que apenas foi enviada uma mensagem “Seja bem-vindo”), o VBA vai precisar armazenar esta resposta em algum lugar. Por isso, se tentarmos apertar o Enter para continuar a macro, ele vai retornar um erro porque não armazenamos a resposta em nenhum lugar.

Para corrigir isso, simplesmente atribuímos o resultado dessa MsgBox a uma variável.

```
Sub mensagem()
    resposta = MsgBox("Deseja executar a macro?", vbYesNo)
|
End Sub
```

```
Sub mensagem()

    resposta = MsgBox("Deseja executar a macro?", vbYesNo)

    If resposta = vbYes Then

        MsgBox ("Macro executada com sucesso!")

    Else

        MsgBox ("Macro cancelada.")

    End If

End Sub
```

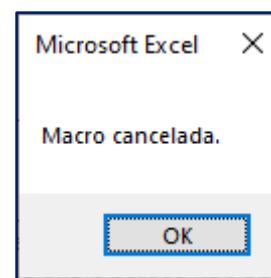
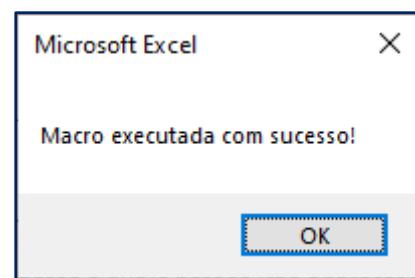
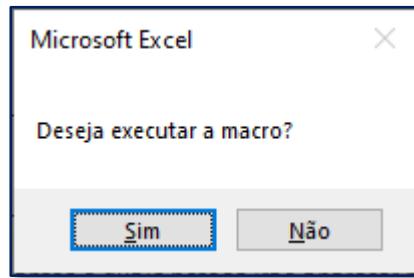
Assim, agora que esperamos uma resposta do usuário, podemos começar a pensar em utilizar estruturas que executam uma determinada ação condicionada à resposta do usuário.

Neste exemplo, vamos criar uma macro que exibe a mensagem “Macro executada com sucesso!” caso a resposta do usuário para a pergunta “Deseja executar a macro?” seja igual a sim, e que exibe a mensagem “Macro cancelada.” caso a resposta seja igual a não.

Para saber se a resposta foi igual a sim, testamos se resposta = vbYes (ou vbNo, caso a negativa viesse primeiro).

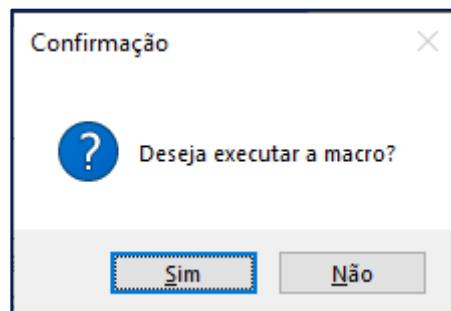
Com este código simples acabamos de criar uma estrutura extremamente importante no VBA, que permite uma interação entre código e usuário.

Até agora, nossas MsgBox ainda estão visualmente pobres". Não temos nenhum ícone nelas e também o título está como o padrão Microsoft Excel.



Os ícones e botões que podemos acrescentar à MsgBox estão resumidos na página 228. Além disso, podemos personalizar o título. A MsgBox tem vários argumentos, que podem ser preenchidos conforme o exemplo abaixo. O mais comum é ir até a opção [Title]. O resultado para essa personalização é mostrado logo abaixo.

```
resposta = MsgBox ("Deseja executar a macro?", vbYesNo + vbQuestion, "Confirmação")  
MsgBox(Prompt, [Buttons As VbMsgBoxStyle = vbOKOnly], [Title], [HelpFile], [Context]) As VbMsgBoxResult
```



```
Sub mensagem()

    resposta = MsgBox("Deseja executar a macro?", vbYesNo + vbQuestion, "Confirmação")

    If resposta = vbYes Then

        resposta2 = MsgBox("Macro executada com sucesso!", vbInformation, "Executado")

    Else

        resposta3 = MsgBox("Macro cancelada.", vbCritical, "Cancelamento")

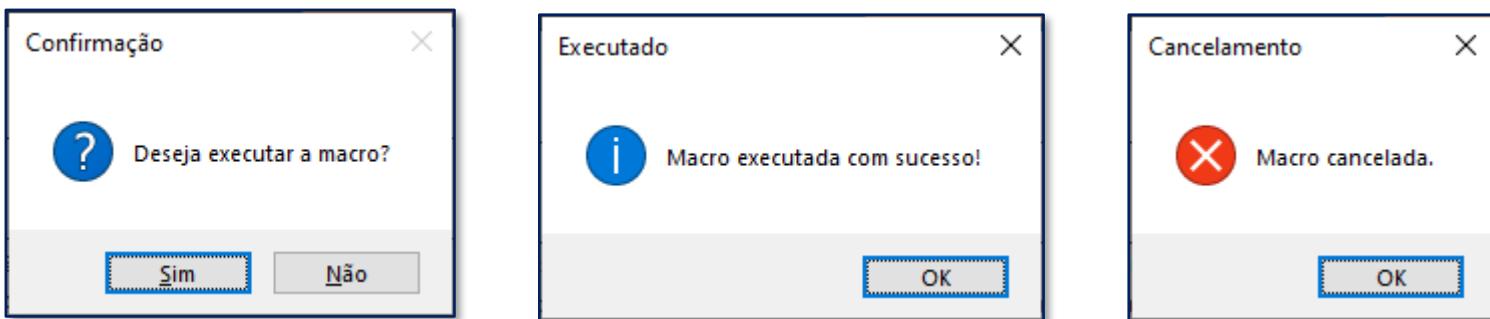
    End If

End Sub
```

A adaptação completa é mostrada na imagem ao lado.

Repare que tivemos que criar duas variáveis (resposta2 e resposta3) para conseguir rodar o código sem erro. Isso é devido a exatamente aquele problema de que, ao colocar um botão na MsgBox, ele obrigatoriamente precisa ser armazenado em uma variável.

O resultado final é mostrado ao lado.



The screenshot shows a Microsoft Excel spreadsheet with the following details:

- Header Row:** A, B, C, D, E, F, G, H, I, J, K, L
- Row 1:** Contains a green border around cell A1.
- Row 2:** Contains a yellow background box with the following text:

Crie uma macro que registre o status do estoque na coluna E. Se o estoque Atual for maior do que o Estoque Mínimo, exiba registre na célula E7 o Status: "Produto OK". Caso contrário, o status deve ser registrado como "Produto em Falta".
- Row 3:** Contains the text: O usuário deve confirmar se deseja executar a macro ou não.
- Row 6:** Contains a table with columns: Produto, Estoque Atual, Estoque Mínimo, and Status. The table has one row with data: Guaraná, 75, 60, and an empty cell for Status.
- Row 7:** Contains a black button labeled "Verificar Status".
- Row 11:** Contains a ribbon bar with tabs: MsgBox (selected), Exercício MsgBox, InputBox, Exercício Boxes, and Exercício Confirmar.

O próximo exercício está explicado na imagem ao lado.

Você pode tentar fazer antes de ver a solução.

```
Sub estoque()

    resposta = MsgBox("Deseja executar a macro?", vbYesNo + vbQuestion, "Confirmação")

    If resposta = vbYes Then

        If Range("C7").Value >= Range("D7").Value Then

            Range("E7").Value = "Produto OK"

        Else

            Range("E7").Value = "Estoque insuficiente"

        End If

    Else

        resposta3 = MsgBox("Macro cancelada.", vbCritical, "Cancelamento")

    End If

End Sub
```

A	B	C	D	E	F	G
1						
2						
3						
4						
5						
6						
7						
8						

Crie uma macro que registre o status do estoque na coluna E. Se o estoque Atual (Coluna D) for maior ou igual ao estoque Mínimo (Coluna C), registre na célula E7 o Status: "Produto OK". Caso contrário, o status deve ser "Estoque insuficiente". O usuário deve confirmar se deseja executar a macro.

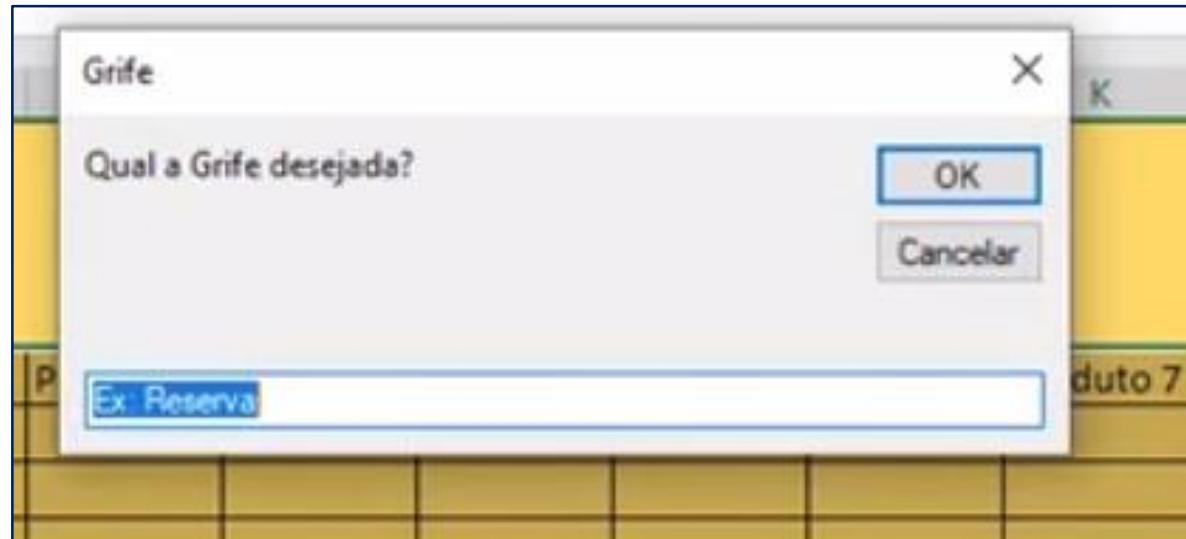
Produto	Estoque Atual	Estoque Mínimo	Status
Guaraná	74	60	Produto OK

Verificar Status

A solução está mostrada ao lado. A lógica é simples, primeiro precisamos perguntar ao usuário se ele deseja executar a macro Sim ou Não.

Se a resposta for Sim, precisamos verificar o valor na célula C7 (estoque atual) é maior ou igual a estoque mínimo (célula D7). Se for, registramos na célula E7 o status “Produto Ok”, caso contrário, registramos “Estoque insuficiente”.

A MsgBox é uma caixa de mensagem que permite ao usuário interagir com o código apenas clicando em botões com opções do tipo “sim”, “não”, “ok”, “cancelar”. Existe uma outra caixa de mensagem que é a InputBox. Essa caixa de mensagem permite ao usuário digitar informações que serão lidas pelo código durante a execução da macro.



A tabela abaixo resume o funcionamento desta caixa de mensagem, que veremos em detalhes a seguir, com exemplos.

Parâmetros: Sintaxe: Finalidade: Interface:	Janela de Entrada (InputBox)	InputBox
	Permitir o recebimento, por parte de uma macro, de uma informação	
	<b>InputBox</b> <b>texto, título, default, xpos, ypos, helpfile, contexto</b> <b>Variável = (InputBox texto, título, default, xpos, ypos, helpfile, contexto)</b>	
	<b>texto</b> - texto obrigatório com a mensagem a ser apresentada ao usuário <b>título</b> - título que aparece na caixa de diálogo (opcional) <b>default</b> - valor default que aparece no campo de entrada de dados <b>xpos e ypos</b> - posição x e y do canto superior esquerdo da janela, em pixels <b>helpfile e contexto</b> - dizem respeito à criação de um texto de ajuda associado a esta InputBox	

A screenshot of Microsoft Excel. The task pane on the left contains the following VBA code:

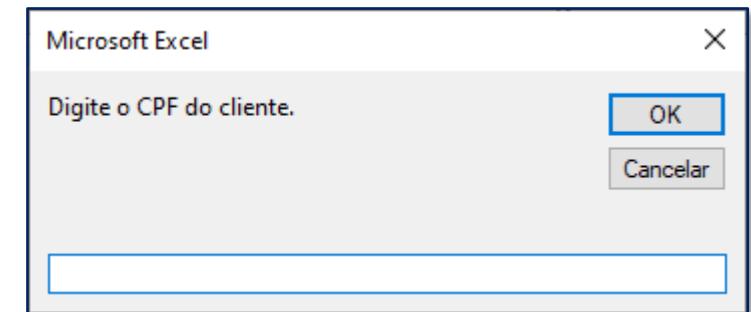
```
Sub cadastro_cliente()
    resposta = InputBox("Digite o CPF do cliente.")
    Cells(15, 3).Value = resposta
End Sub
```

The ribbon at the top has tabs: MsgBox, Exercício MsgBox, **InputBox**, Exercício Boxes, and Exercício Confirmar. The 'InputBox' tab is selected.

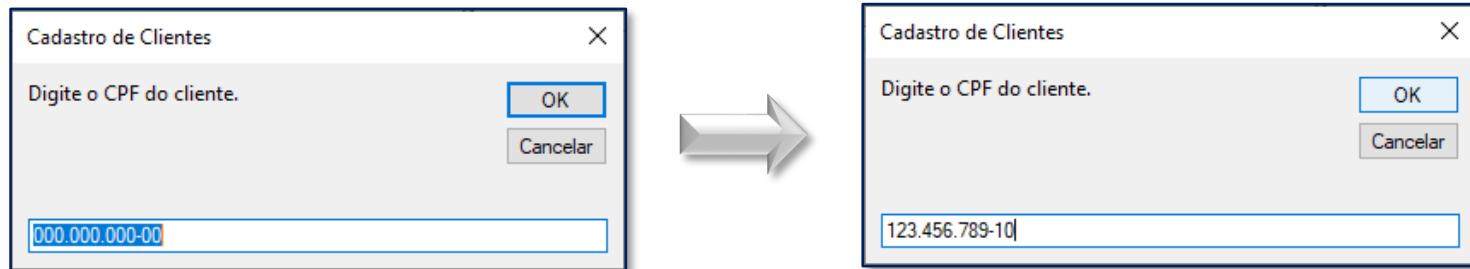
O exercício ao lado pede para criar uma InputBox para inserir o CPF do funcionário na célula C15.

A estrutura do código que executa esta ação é mostrada também na imagem ao lado.

```
Sub cadastro_cliente()
    resposta = InputBox("Digite o CPF do cliente.")
    Cells(15, 3).Value = resposta
End Sub
```



```
Sub cadastro_cliente()
    resposta = InputBox("Digite o CPF do cliente.", "Cadastro de Clientes", "000.000.000-00")
    Cells(15, 3).Value = resposta
End Sub
```



	A	B	C
13	Crie uma macro que cadastre o CPF do cliente na célula C15. O CPF deve ser inserido por meio de InputBox		
14			
15	CPF:	123.456.789-10	
16		InputBox	
17		Exercício MsgBox	Exercício MsgBox

Podemos ainda fazer algumas melhorias na nossa inputbox:

1. **[Title]** : Título da inputbox.
2. **[Default]** : Texto de exemplo para auxiliar o usuário em como preencher o campo.

	A	B	C	D	E	F	G	H	I
1									
2	<b>Crie uma macro que cadastre um novo produto na tabela abaixo.</b>								
3	<b>1. O produto, o preço unitário e a quantidade em estoque devem ser pedidas por meio de InputBox</b>								
4	<b>2. Adapte a Macro para funcionar em qualquer linha da tabela</b>								
5									
6	<b>Produto</b>	<b>Preço Unitário</b>	<b>Quantidade em Estoque</b>						
7	Guaraná	R\$ 4,40							
8	Coca	R\$ 1,60							
9	Coxinha	R\$ 4,30							
10	Kibe	R\$ 4,00							
11	Pastel	R\$ 2,70							
12	Água de Coca	R\$ 2,00							
13									
14									
15									
16									
17									

**Cadastrar Novo Produto**

Tente agora praticar com o exercício ao lado. O enunciado é mostrado ao lado.

Dica: Você deverá criar 3 InputBox diferentes e também uma variável para armazenar a posição da próxima linha a ser preenchida na tabela, independente da quantidade de linhas.

Mão à obra!

```
Sub cadastro_produtos()

ult_linha = Range("B6").End(xlDown).Row + 1
'ult_linha = Range("B10000").End(xlUp).Row + 1 'outra opção para ult_linha

produto = InputBox("Digite o nome do produto", "Produto")
preco = InputBox("Digite o preço do produto", "Preço")
quantidade = InputBox("Digite a quantidade em estoque", "Quantidade em estoque")

Cells(ult_linha, 2).Value = produto
Cells(ult_linha, 3).Value = preco
Cells(ult_linha, 4).Value = quantidade

End Sub
```

Bom, espero que tenha tentado fazer. O resultado final está mostrado ao lado.

Último Carro Vendido

A	B	C	D	E	
1					
2	Último Carro Vendido				
3	Data	Carro	Preço	Opcionais	
4					
5					
6	<b>Cadastrar Venda</b>				
7					
8					
9					
10	Crie uma macro para cadastrar a venda de um carro. Essa macro deve:				
11	1. Confirmar com o usuário se deseja cadastrar a venda (MsgBox)				
12	2. Perguntar por meio de InputBox: Qual o carro vendido, qual o preço e quais os opcionais (3 inputbox ao todo)				
13	3. Cadastrar as informações nas células destacadas				
14	4. Exibir uma mensagem confirmando o cadastro, informando o nome do carro cadastrado				
15					
16	Obs: Você pode pensar em opcionais como: banco de couro, kit multimídia, tapete original, vidros e travas elétricos				
17					

Pronto

MsgBox | Exercício MsgBox | InputBox | Exercício Boxes | **Exercício Confirmar** | +

O próximo exercício vai juntar os conceitos de MsgBox e InputBox.

O enunciado é mostrado ao lado. Antes de ver a resposta final na próxima aba, tente construir o código sozinho. Todo o passo a passo do exercício está descrito na imagem ao lado.

```
Sub carros()

    resposta = MsgBox("Deseja executar a macro?", vbYesNo)

    If resposta = vbYes Then

        Cells(4, 2).Value = Date
        Cells(4, 3).Value = InputBox("Digite o nome do carro")
        Cells(4, 4).Value = InputBox("Digite o preço")
        Cells(4, 5).Value = InputBox("Digite os opcionais")|  
End If

End Sub
```

A solução final está mostrada ao lado.

Repare que, desta vez, escrevemos o valor da InputBox diretamente na célula (em vez de criar uma variável antes).

Se você criou variáveis, perfeito, está certo da mesma forma!

	A	B	C	D	E	F
1	Nome	Área	Salário			
2	Adriane de Carvalho Gomes	RH	R\$ 4.650,00			
3	Andressa Rotsztejn	Marketing	R\$ 8.100,00			
4	Beatriz Leite Júnior	Administrativo	R\$ 4.650,00			
5	Bruna dos Santos Gomes	Administrativo	R\$ 4.650,00			
6	Carolina Codeceira	Marketing	R\$ 8.100,00			
7	Daniela Muller Braga	Logística	R\$ 2.000,00			
8	Elvis de Freitas Derossi	RH	R\$ 12.320,00			
9	Gabrielle Andrade da Silva	Administrativo	R\$ 12.320,00			
10	Gustavo de Vasconcelos	Administrativo	R\$ 16.300,00			
11	Leticia Mesquita	Administrativo	R\$ 9.450,00			
12	Marcela de Freitas	Financeiro	R\$ 4.650,00			
13	Marianna Pestana	Marketing	R\$ 3.700,00			
14	Matheus de Souza	Marketing	R\$ 7.200,00			
15	Pedro Costa	Logística	R\$ 16.300,00			
16	Rafael Machado Cardoso	Logística	R\$ 16.300,00			
17	Renan Freire	Marketing	R\$ 7.200,00			
18	Thaís Ribeiro	Financeiro	R\$ 16.300,00			
19	Victor Ruzza de Carvalho	Logística	R\$ 1.600,00			

**Novo Funcionário**

Pronto

O próximo exercício é um desafio.

Desafio não porque seja difícil, e sim porque é necessário um ponto de atenção.

O objetivo é criar uma macro que, por meio de 3 InputBox, cadastre um novo funcionário, sua área e seu salário, sempre no final da tabela. Além disso, será necessário que, ao final da macro, a tabela seja ordenada por ordem alfabética. Nós já vimos como criar uma macro classificando em ordem alfabética (página 24). Você terá que gravar uma macro para descobrir o código e em seguida colar no seu código principal.

Obs: Utilize seu senso crítico para observar a macro gravada e ver se você terá que fazer alguma alteração para evitar qualquer problema.

Mãos à obra!

```
ActiveWorkbook.Worksheets("Cadastro").Sort.SortFields.Clear  
ActiveWorkbook.Worksheets("Cadastro").Sort.SortFields.Add Key:=Range("A2:A20") _  
    , SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal  
With ActiveWorkbook.Worksheets("Cadastro").Sort  
    .SetRange Range("A1:C20")  
    .Header = xlYes  
    .MatchCase = False  
    .Orientation = xlTopToBottom  
    .SortMethod = xlPinYin  
    .Apply  
End With
```

Vamos para a solução.

Quando você realizou a gravação de macro para classificar a tabela, possivelmente seus intervalos apareceram como na imagem ao lado:

Range("A2:A20") e  
Range("A1:C20")

Você não precisa entender a fundo esse código para saber que ele só vai funcionar até a linha 20 da tabela.

Portanto, você deverá fazer uma pequena adaptação no código, e a solução final está mostrada na página seguinte.

```
Sub cadastro_funcionarios()
    ult_linha = Range("A1").End(xlDown).Row + 1
    Cells(ult_linha, 1).Value = InputBox("Digite o nome do funcionário", "Funcionário")
    Cells(ult_linha, 2).Value = InputBox("Digite a área do funcionário", "Área")
    Cells(ult_linha, 3).Value = InputBox("Digite o salário do funcionário", "Salário")

    ActiveWorkbook.Worksheets("Cadastro").Sort.SortFields.Clear
    ActiveWorkbook.Worksheets("Cadastro").Sort.SortFields.Add Key:=Range("A:A")
        , SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
    With ActiveWorkbook.Worksheets("Cadastro").Sort
        .SetRange Range("A:C")
        .Header = xlYes
        .MatchCase = False
        .Orientation = xlTopToBottom
        .SortMethod = xlPinYin
        .Apply
    End With

    Range("A1").Select
End Sub
```

A melhor maneira de corrigir este problema é considerando todo o intervalo da coluna, sem especificar as linhas. Assim, independente do tamanho da tabela, ele sempre irá classificar a coluna inteira.

Este exemplo mostra que, mesmo gravando uma macro, é importante ter algum senso crítico para analisar se não será necessário fazer alguma modificação, o mais simples que seja, para que a macro gravada funcione para qualquer situação.

Reforçando, se você ficou com dúvida em como gravar uma macro para classificar a sua tabela de A à Z, recomendamos que volte até a página 24 pois este exercício já foi feito.

## Módulo 7

# Exercícios: Compilação e Otimização de Macros

Neste módulo faremos vários desafios de compilações avançadas utilizando as estruturas aprendidas até o momento do curso. Antes de ver a resolução final dos exercícios, é importante que você tente fazer sozinho para testar os seus conhecimentos adquiridos ao longo do curso.

Pensando que você poderá ter dúvidas ao longo da construção do código, mas não necessariamente vai querer ir direto para a solução final, as resoluções serão divididas em partes e as páginas indicadas no momento da explicação do exercício. Assim, caso você tenha alguma dúvida em algum ponto específico do código, mas não queira necessariamente pular para a solução final, você pode avançar até a página indicada e obter uma pequena “colinha” para prosseguir o código sozinho a partir dali.

A estrutura será:



### SOLUÇÃO

**Parte 1** - Descrição da parte 1 ([página XXX](#))

**Parte 2** - Descrição da parte 2 ([página XXX](#))

**Parte 3** - Descrição da parte 3 ([página XXX](#))

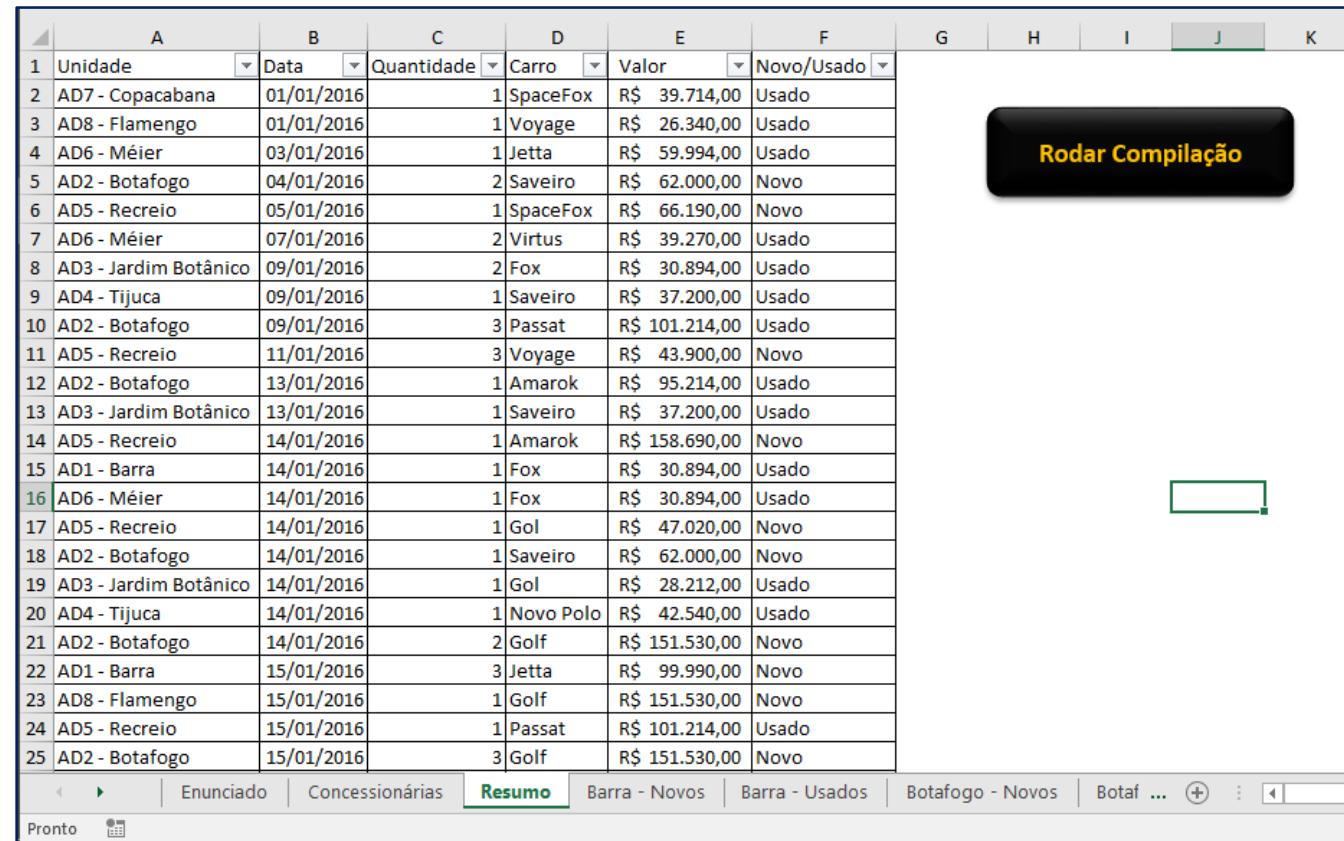
...

**Parte N** - Descrição da parte final ([página XXX](#))

No primeiro exercício vamos partir de uma base de dados com todos os registros de vendas, de uma determinada empresa, de diferentes carros conforme a coluna D. Além disso, cada carro vendido pode ter sido um carro usado ou um carro novo.

O objetivo final do código distribuir os diferentes carros de acordo com a aba correspondente à Unidade (da coluna a). Em resumo, devemos ter uma macro que irá:

1. Perguntar ao usuário se ele deseja executar a macro.
2. Exibir um InputBox perguntando se a compilação será feita para carros novos ou carros usados.
3. Filtrar a tabela da aba resumo corretamente.
4. Copiar as linhas da tabela filtrada na aba “Resumo” e colar nas abas correspondentes.



The screenshot shows a Microsoft Excel spreadsheet with a table of car sales data. The table has columns: A (Unidade), B (Data), C (Quantidade), D (Carro), E (Valor), and F (Novo/Usado). The data includes various units like AD7-Copacabana, dates from 01/01/2016 to 15/01/2016, quantities ranging from 1 to 3, car models like SpaceFox, Voyage, Jetta, Saveiro, etc., prices in R\$ (e.g., 39.714,00 to 151.530,00), and status as Novo or Usado. A black button labeled "Rodar Compilação" is visible on the ribbon bar. The ribbon bar also shows tabs for "Resumo", "Barra - Novos", "Barra - Usados", "Botafogo - Novos", and "Botafogo - Usados".

A	B	C	D	E	F
1 Unidade	Data	Quantidade	Carro	Valor	Novo/Usado
2 AD7 - Copacabana	01/01/2016	1	SpaceFox	R\$ 39.714,00	Usado
3 AD8 - Flamengo	01/01/2016	1	Voyage	R\$ 26.340,00	Usado
4 AD6 - Méier	03/01/2016	1	Jetta	R\$ 59.994,00	Usado
5 AD2 - Botafogo	04/01/2016	2	Saveiro	R\$ 62.000,00	Novo
6 AD5 - Recreio	05/01/2016	1	SpaceFox	R\$ 66.190,00	Novo
7 AD6 - Méier	07/01/2016	2	Virtus	R\$ 39.270,00	Usado
8 AD3 - Jardim Botânico	09/01/2016	2	Fox	R\$ 30.894,00	Usado
9 AD4 - Tijuca	09/01/2016	1	Saveiro	R\$ 37.200,00	Usado
10 AD2 - Botafogo	09/01/2016	3	Passat	R\$ 101.214,00	Usado
11 AD5 - Recreio	11/01/2016	3	Voyage	R\$ 43.900,00	Novo
12 AD2 - Botafogo	13/01/2016	1	Amarok	R\$ 95.214,00	Usado
13 AD3 - Jardim Botânico	13/01/2016	1	Saveiro	R\$ 37.200,00	Usado
14 AD5 - Recreio	14/01/2016	1	Amarok	R\$ 158.690,00	Novo
15 AD1 - Barra	14/01/2016	1	Fox	R\$ 30.894,00	Usado
16 AD6 - Méier	14/01/2016	1	Fox	R\$ 30.894,00	Usado
17 AD5 - Recreio	14/01/2016	1	Gol	R\$ 47.020,00	Novo
18 AD2 - Botafogo	14/01/2016	1	Saveiro	R\$ 62.000,00	Novo
19 AD3 - Jardim Botânico	14/01/2016	1	Gol	R\$ 28.212,00	Usado
20 AD4 - Tijuca	14/01/2016	1	Novo Polo	R\$ 42.540,00	Usado
21 AD2 - Botafogo	14/01/2016	2	Golf	R\$ 151.530,00	Novo
22 AD1 - Barra	15/01/2016	3	Jetta	R\$ 99.990,00	Novo
23 AD8 - Flamengo	15/01/2016	1	Golf	R\$ 151.530,00	Novo
24 AD5 - Recreio	15/01/2016	1	Passat	R\$ 101.214,00	Usado
25 AD2 - Botafogo	15/01/2016	3	Golf	R\$ 151.530,00	Novo

Uma explicação mais detalhada do exercício é mostrada abaixo. Mão à obra!

Sua macro deve olhar para todas as Concessionárias existentes na aba 'Concessionárias'. Todas as vendas da história da empresa estão descritas na aba 'Resumo'.

O objetivo é que a macro filtre as vendas de cada concessionária, e copie e cole na aba específica dessa concessionária, dependendo se o usuário deseja compilar as concessionárias de Carros Novos ou usados.

Antes de começar a executar a macro, o usuário deve confirmar se realmente deseja executar, por meio de uma MsgBox, e deve também informar se deseja compilar as vendas dos carros novos ou usados, por meio de uma InputBox.

Exemplo: Se o usuário digitar 'Usados' na Inputbox, sua macro deverá filtrar somente os carros usados, e copiar e colar as vendas de cada concessionária apenas nas abas das concessionárias de carros usados

Cuidado para limpar as tabelas antes de colar as informações

\*\*\*Dica: 1) Grave uma macro para descobrir como se filtra a concessionária desejada e o tipo do carro (novo ou usado)

2) Você vai precisar de uma estrutura de repetição para varrer as concessionárias existentes



## SOLUÇÃO

**Parte 1 - Construção da MsgBox + InputBox ([página 251](#))**

**Parte 2 - Como percorrer cada aba da planilha ([página 252](#))**

**Parte 3 - Como obter o código de filtrar a tabela ([página 253](#))**

**Parte 4 - Como copiar os dados da tabela e selecionar a aba certa ([página 255](#))**

**Parte 5 - Código Final ([página 256](#))**

```
Sub compila()

    resposta = MsgBox("Deseja realmente executar a macro?", vbYesNo)

    If resposta = vbYes Then

        tipo = InputBox("Você deseja compilar os carros novos ou usados?", "Tipo dos carros", "Novo/Usado")
        |
    End If

End Sub
```

A **PARTE 1** do código basicamente é começar perguntando ao usuário se ele deseja executar a macro e caso a resposta seja Sim, perguntar a ele se ele deseja compilar para os carros novos ou usados.

Aqui é importante dar a ele um exemplo de como ele deve escrever a resposta “Novo/Usado”, pois esse padrão será importante na hora que a gente for filtrar a nossa resposta de acordo com o tipo que ele preencher.

```
Sub compila()

    resposta = MsgBox("Deseja realmente executar a macro?", vbYesNo)

    If resposta = vbYes Then

        tipo = InputBox("Você deseja compilar os carros novos ou usados?", "Tipo dos carros", "Novo/Usado")

        Sheets("Concessionárias").Activate

        For linha = 2 To 9

            concessionaria = Cells(linha, 1).Value

        Next

    End If

End Sub
```

A
1 Unidade
2 AD1 - Barra
3 AD2 - Botafogo
4 AD3 - Jardim Botânico
5 AD4 - Tijuca
6 AD5 - Recreio
7 AD6 - Méier
8 AD7 - Copacabana
9 AD8 - Flamengo
10
11
12
13
14
15
...

Na **PARTE 2**, precisamos pensar que devemos percorrer cada uma das abas do nosso arquivo. Os nomes das abas basicamente tem em sua composição a unidade. Para nos auxiliar, vamos utilizar cada uma das unidades da aba “Concessionárias”, da linha 2 até a linha 9, para armazenar o nome da unidade para em seguida utilizar no filtro da tabela na aba “Resumo”.

## Módulo 7 – Exercícios – Desafio Compilação Concessionárias (Parte 3)

253

The first screenshot shows the 'Unidade' column being filtered by 'AD1 - Barra'. The second screenshot shows the 'Novo/Usado' column being filtered by 'Novo'.

Na **PARTES 3**, precisamos do código que filtra a nossa tabela da aba “Resumo”, pois o objetivo é filtrá-la, para cada Unidade e Tipo do carro, para poder copiar a tabela e colar na aba correspondente.

Para descobrir este código, gravamos uma macro simples filtrando as colunas A e F. Podemos chamar essa macro de **filtrar\_tabela**.

The screenshot shows the 'Dados' tab selected in the ribbon. A red arrow points to the 'Filtro' button in the 'Classificar e Filtrar' group.

Em seguida, não esqueça de Parar a gravação e Limpar os filtros da tabela para voltar à situação inicial sem os filtros.

```

Sub compila()
    resposta = MsgBox("Deseja realmente executar a macro?", vbYesNo)
    If resposta = vbYes Then
        tipo = InputBox("Você deseja compilar os carros novos ou usados?", "Tipo dos carros", "Novo/Usado")
        Sheets("Concessionárias").Activate
        For linha = 2 To 9
            concessionaria = Cells(linha, 1).Value
            Sheets("Resumo").Activate
            ActiveSheet.Range("$A$1:$F$1600").AutoFilter Field:=1, Criteria1:= _
                "AD1 - Barra"
            ActiveSheet.Range("$A$1:$F$1600").AutoFilter Field:=6, Criteria1:="Novo"
        Next
    End If
End Sub

```

```

Sub compila()
    resposta = MsgBox("Deseja realmente executar a macro?", vbYesNo)
    If resposta = vbYes Then
        tipo = InputBox("Você deseja compilar os carros novos ou usados?", "Tipo dos carros", "Novo/Usado")
        Sheets("Concessionárias").Activate
        For linha = 2 To 9
            concessionaria = Cells(linha, 1).Value
            Sheets("Resumo").Activate
            ActiveSheet.Range("$A$1:$F$1600").AutoFilter Field:=1, Criteria1:="AD1 - Barra"
            ActiveSheet.Range("$A$1:$F$1600").AutoFilter Field:=6, Criteria1:="Novo"
        Next
    End If
End Sub

```

Ainda na **PARTE 3**, voltamos para o ambiente VBA, copiamos o código da macro gravada e colamos no nosso código principal. Logo após armazenar o nome da concessionária na variável **concessionaria** precisamos selecionar a aba “Resumo” pois é nela que vamos aplicar os filtros.

Repare que o código do filtro trouxe um underline. Isso é apenas um recurso para pular de linha e continuar escrevendo o código na linha de baixo. Porém, podemos manter tudo em uma linha mesmo, não faz diferença.

```
Sub compila()
    resposta = MsgBox("Deseja realmente executar a macro?", vbYesNo)
    If resposta = vbYes Then
        tipo = InputBox("Você deseja compilar os carros novos ou usados?", "Tipo dos carros", "Novo/Usado")
        Sheets("Concessionárias").Activate
        For linha = 2 To 9
            concessionaria = Cells(linha, 1).Value
            Sheets("Resumo").Activate
            ActiveSheet.Range("$A$1:$F$1600").AutoFilter Field:=1, Criteria1:=concessionaria
            ActiveSheet.Range("$A$1:$F$1600").AutoFilter Field:=6, Criteria1:=tipo
            ult_linha = Range("A1").End(xlDown).Row
            Range("A1:F" & ult_linha).Copy
            nome_aba = Mid(concessionaria, 7) & " - " & tipo & "s"
            Next
        End If
    End Sub
```

O nome das abas tem a estrutura abaixo. Como ela não possui o código no começo, precisamos usar uma fórmula de texto Mid para extrair apenas a partir do caractere 7. Se não colocarmos nada no 3º argumento do Mid é como se estivéssemos dizendo que queremos tudo após o caractere 7 (mesma ideia do 100 que utilizamos nos exercícios de fórmulas de texto. Além disso, o tipo está no plural, então concatenamos um "s" para formar o nome correto da aba.



Barra - Novos | Barra - Usados | Botafogo - Novos | Botafogo - Usados

AD1 - Barra

Ainda na **PARTE 4**, primeiro devemos pensar que os critérios do filtro precisam variar de acordo com a concessionária da variável **concessionaria** e do tipo (Usado/Novo) da variável **tipo**. Por isso, já mudamos os critérios para considerar estas duas variáveis.

Além disso, após filtrar a tabela na aba “Resumo”, precisamos copiar os dados desde a primeira linha até a última linha preenchida. Para isso, criamos uma variável **ult\_linha** para armazenar a última linha da tabela filtrada e depois a usamos no Range de cópia.

Por fim, como queremos colar estes dados na tabela correspondente, precisamos selecioná-la antes de usar o comando **.PasteSpecial** de colagem. Para isso, criamos uma variável chamada **nome\_aba** que irá receber o nome da aba que deverá ser selecionada para que possamos colar os dados corretamente.

```
Sub compila()
    resposta = MsgBox("Deseja realmente executar a macro?", vbYesNo)
    If resposta = vbYes Then
        tipo = InputBox("Você deseja compilar os carros novos ou usados?", "Tipo dos carros", "Novo/Usado")
        Sheets("Concessionárias").Activate
        For linha = 2 To 9
            concessionaria = Cells(linha, 1).Value
            Sheets("Resumo").Activate
            ActiveSheet.Range("$A$1:$F$1600").AutoFilter Field:=1, Criterial:=concessionaria
            ActiveSheet.Range("$A$1:$F$1600").AutoFilter Field:=6, Criterial:=tipo
            ult_linha = Range("A1").End(xlDown).Row
            Range("A1:F" & ult_linha).Copy
            nome_aba = Mid(concessionaria, 7) & " - " & tipo & "s"
            Sheets(nome_aba).Activate
            Range("A1").PasteSpecial
            Sheets("Concessionárias").Activate
        Next
    End If
End Sub
```

Ainda na **PARTE 5**, após armazenar o nome da aba e selecionar a aba correta para colagem dos valores, usamos o comando PasteSpecial.

Por fim, é importante finalizar com o comando:

Sheets("Concessionárias").Activate

Pois o código precisa voltar na aba concessionárias para armazenar um novo valor de Unidade para iniciar um novo loop para a próxima unidade da empresa.

## Módulo 7 – Exercícios – Desafio Compilação Concessionárias (Parte 5)

257

The screenshot shows a Microsoft Excel spreadsheet titled "Concessionárias.xlsxm - Excel". The "Dados" (Data) tab is selected. A yellow button labeled "Rodar Compilação" is positioned in the center of the data area. The data table contains information about car sales, including the unit (Unidade), date (Data), quantity (Quantidade), car model (Carro), value (Valor), and status (Novo/Usado). The table is filtered, showing only entries for AD8 - Flamengo. The "Resumo" tab is currently selected at the bottom of the sheet.

Ao executarmos a macro, veremos que ela funcionará perfeitamente, colando os dados filtrados em cada uma das abas do arquivo.

Porém, ainda temos algumas coisas para fazer antes de deixar esse código 100% otimizado.

Um exemplo é o problema ao lado: na aba “Resumo”, as células continuam copiadas, e não é legal deixar essas células dessa forma. Além disso, a tabela também permaneceu filtrada. Seria interessante se pudéssemos também retirar os filtros aplicados antes de finalizar a execução do código.

```
Sub compila()

resposta = MsgBox("Deseja realmente executar a macro?", vbYesNo)

If resposta = vbYes Then

    For Each aba In ThisWorkbook.Sheets

        If aba.Name <> "Enunciado" And aba.Name <> "Concessionárias" And aba.Name <> "Resumo" Then

            aba.Activate

            Range("A2:F100000").ClearContents

        End If

    Next

    tipo = InputBox("Você deseja compilar os carros novos ou usados?", "Tipo dos carros", "Novo/Usado")

    Sheets("Concessionárias").Activate

    For linha = 2 To 9

        concessionaria = Cells(linha, 1).Value

        Sheets("Resumo").Activate

        ActiveSheet.Range("$A$1:$F$1600").AutoFilter Field:=1, Criteria1:=concessionaria
        ActiveSheet.Range("$A$1:$F$1600").AutoFilter Field:=6, Criteria1:=tipo

        ult_linha = Range("A1").End(xlDown).Row

        Range("A1:F" & ult_linha).Copy

        nome_aba = Mid(concessionaria, 7) & " - " & tipo & "s"
```

O primeiro ajuste que devemos fazer é deletar os dados de todas as abas sempre antes de rodar a macro. Porém, não podemos deletar de todas as abas, apenas aquelas que são diferentes de “Enunciado”, “Concessionárias” e da própria “Resumo”.

Para isso, usamos uma estrutura For para percorrer cada aba e uma estrutura If para testar o nome de cada aba. E se o nome da aba for diferente de “Enunciado” **E** “Concessionárias” **E** “Resumo”, então os valores do intervalo da coluna A até F podem ser excluídos.

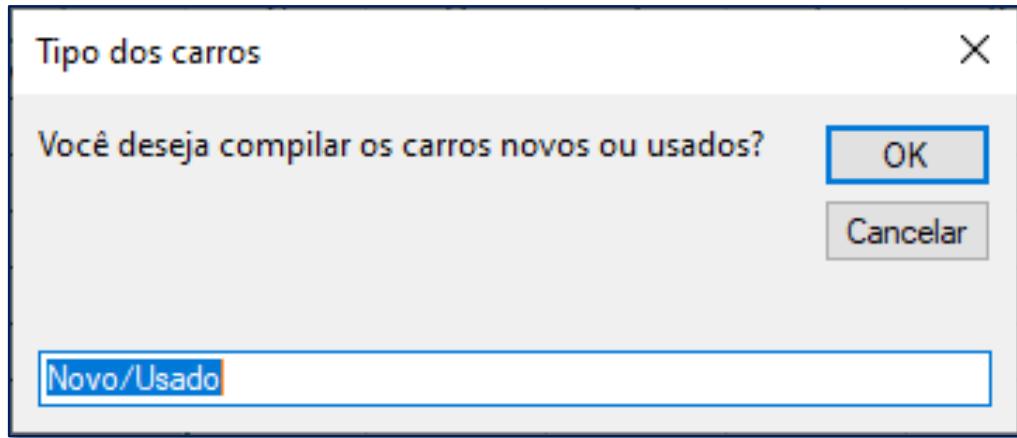
```
Next  
  
    tipo = InputBox("Você deseja compilar os carros novos ou usados?", "Tipo dos carros", "Novo/Usado")  
  
    Sheets("Concessionárias").Activate  
  
    For linha = 2 To 9  
  
        concessionaria = Cells(linha, 1).Value  
  
        Sheets("Resumo").Activate  
  
        ActiveSheet.Range("$A$1:$F$1600").AutoFilter Field:=1, Criteria1:=concessionaria  
        ActiveSheet.Range("$A$1:$F$1600").AutoFilter Field:=6, Criteria1:=tipo  
  
        ult_linha = Range("A1").End(xlDown).Row  
  
        Range("A1:F" & ult_linha).Copy  
  
        nome_aba = Mid(concessionaria, 7) & " - " & tipo & "s"  
  
        Sheets(nome_aba).Activate  
  
        Range("A1").PasteSpecial  
  
        Sheets("Concessionárias").Activate  
  
    Next  
  
    Sheets("Resumo").Activate  
  
    ActiveSheet.ShowAllData  
  
End If  
  
End Sub
```

Outro ajuste seria retirar os filtros da tabela da aba “Resumo”. Para saber o código por trás de retirar os filtros da tabela, lembre-se que você pode gravar uma macro fazendo isso, como já praticamos diversas vezes.

O código então está mostrado ao lado: devemos selecionar a aba “Resumo” e depois usar a estrutura:

## ActiveSheet.ShowAllData

para retirar todos os filtros da tabela original na aba “Resumo”.



Para fechar esse código, um último ajuste:

A nossa InputBox pede uma resposta que deve ser ou 'Novo' ou 'Usado'. Apesar de termos sugerido a maneira como a pessoa deve escrever aquela resposta, ela ainda assim pode ter dúvidas e acabar escrevendo uma resposta que não é a correta. Isso irá ocasionar problemas porque a macro depende de uma resposta certa para filtrar corretamente a coluna F da tabela da aba "Resumo".

	A	B	C	D	E	F
1	Unidade	Data	Quantidade	Carro	Valor	Novo/Usado
2	AD7 - Copacabana	01/01/2016		1 SpaceFox	R\$ 39.714,00	Usado
3	AD8 - Flamengo	01/01/2016		1 Voyage	R\$ 26.340,00	Usado
4	AD6 - Méier	03/01/2016		1 Jetta	R\$ 59.994,00	Usado

```
Sub compila()

resposta = MsgBox("Deseja realmente executar a macro?", vbYesNo)

If resposta = vbYes Then

    For Each aba In ThisWorkbook.Sheets

        If aba.Name <> "Enunciado" And aba.Name <> "Concessionárias" And aba.Name <> "Resumo" Then

            aba.Activate

            Range("A2:F100000").ClearContents

        End If

    Next

Alon:

    tipo = InputBox("Você deseja compilar os carros novos ou usados?", "Tipo dos carros", "Novo/Usado")

    If tipo <> "Novo" And tipo <> "Usado" Then

        MsgBox ("Favor, inserir somente 'Novo' ou 'Usado'.") 

        GoTo Alon

    End If

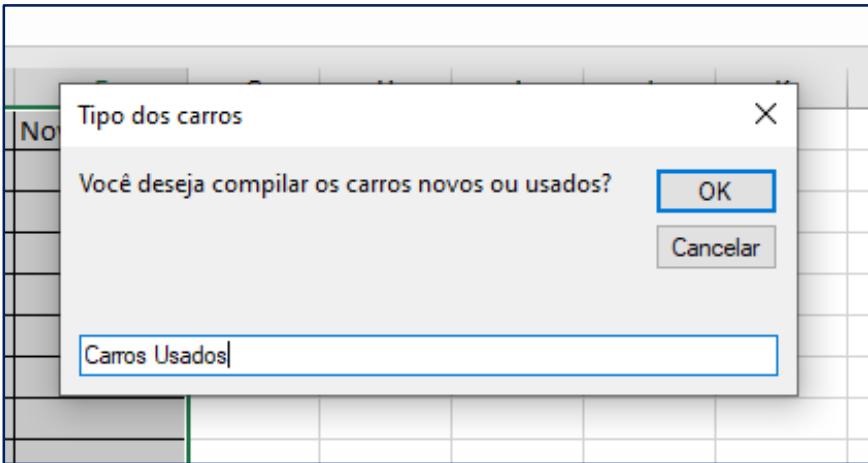
Sheets("Concessionárias").Activate

For linha = 2 To 9
```

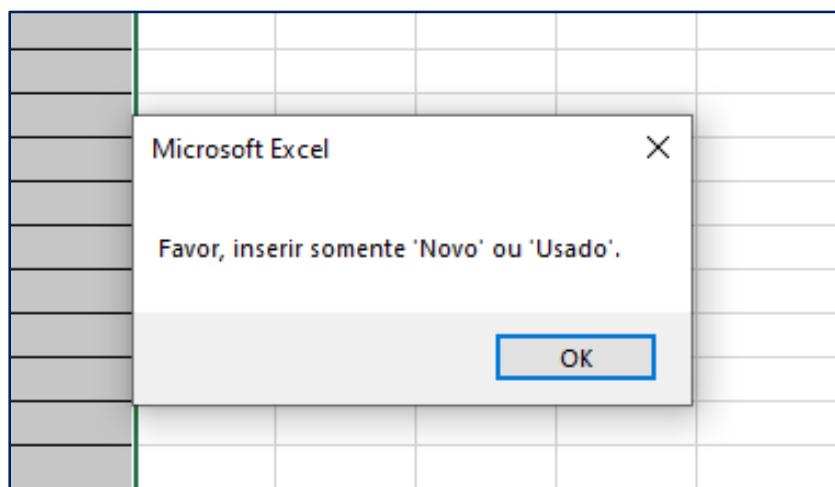
O que podemos fazer então é criar uma estrutura If, logo após o resultado do InputBox, para testar se a resposta está correta (ou a pessoa escreveu 'Novo' ou escreveu 'Usado').

E caso a pessoa tenha escrito errado, fazemos com que o código volte para a linha do InputBox. Esse movimento de voltar para um ponto específico do código é feito através da estrutura GoTo (Ir para). Criamos uma palavra chave para usar como uma espécie de marca no nosso código e permitimos assim que ele vá para este ponto toda vez que usamos a estrutura GoTo.

No próximo módulo de tratamento de erros veremos com mais detalhes esta estrutura, então pode ficar tranquilo.



Agora, toda vez que o usuário não escrever a resposta correta, uma mensagem mais informativa será exibida e ele saberá a maneira correta de preencher o campo.



No próximo exercício, temos uma tabela com informações de diferentes plataformas, para todos os meses do ano de 2015. O objetivo é registrar o volume extraído de cada célula da coluna D na respectiva aba (mês) e na respectiva coluna de plataforma, de acordo com a plataforma na coluna C da aba “Base”. Isso deverá ser feito para cada linha da tabela da aba “Base”. Ou seja, de acordo com o Mês e com a plataforma, registrar o volume na célula correta. A imagem ao lado exemplifica para um valor do mês de janeiro, plataforma ODP2.

A ideia basicamente será:

1. Criar uma estrutura de repetição para percorrer cada linha da tabela na aba “Base”.
2. Selecionar a aba do mês correto e encontrar a última linha e a coluna certa a ser preenchida com cada registro.
3. Lembrar de limpar todos os dados de todas as abas dos meses antes de executar a macro.

	A	B	C	D	E	F	G	H	I
1	Mês	Data	Plataforma	Volume Extraído					
2	Janeiro	01/01/2015	ODP2	406.846,00					
3	Janeiro	01/01/2015	MDP1	55.791,00					
4	Janeiro	01/01/2015	MDP2	270.615,00					
5	Janeiro	01/01/2015	ODP1	425.057,00					
6	Janeiro	01/01/2015	ODP4	319.305,00					
7	Janeiro	01/01/2015	ODP1	109.599,00					
8	Janeiro	01/01/2015	MDP1	27.776,00					
9	Janeiro	01/01/2015	ODP3	260.381,00					
10	Janeiro	01/01/2015	MDP3	413.193,00					
11	Janeiro	01/01/2015	ODP2	404.792,00					
12	Janeiro	02/01/2015	MDP1	404.662,00					
13	Janeiro	02/01/2015	MDP3	214.702,00					
14	Janeiro	02/01/2015	MDP3	477.416,00					
15	Janeiro	02/01/2015	ODP2	31.477,00					
16	Janeiro	02/01/2015	ODP1	206.158,00					
17	Janeiro	03/01/2015	ODP1	466.814,00					

	A	B	C	D	E	F	G	H
1	Extração	MDP1	MDP2	MDP3	ODP1	ODP2	ODP3	ODP4
2		1						
3		2						
4		3						
5		4						
6		5						
7		6						

Cada uma das partes da solução é mostrada no resumo abaixo. Tente exercitar e verificar se você consegue resolver o exercício sem conferir a solução. Caso você tenha dúvidas, pule apenas para a parte onde você tiver travado e continue a partir dali.

Tanto neste exercício quanto nas otimizações que faremos mais a frente, usaremos como estrutura de repetição de repetição o Do Until. Porém, na página 268, também será mostrada a solução final alternativa usando como estrutura de repetição o For.

### SOLUÇÃO

**Parte 1** - Construção do Loop de repetição na aba “Base” ([página 265](#))

**Parte 2** - Descobrindo a célula correta (linha e coluna) para registro do valor ([página 266](#))

**Parte 3** - Finalizando o Código ([página 267](#))

**Parte 4** - Código Final: Do Until x For ([página 268](#))

```
Sub compila()
    Sheets("Base").Activate
    linha = 2
    Do Until Cells(linha, 1).Value = ""
        mes = Cells(linha, 1).Value
        plataforma = Cells(linha, 3).Value
        volume = Cells(linha, 4).Value
        Sheets(mes).Activate
        |
        Loop
End Sub
```

Na **PARTE 1** construímos o nosso código em cima da estrutura de repetição que vai percorrer cada uma das linhas da aba “Base”. Dessa vez, em vez do For, usamos a estrutura Do Until.

Com Do Until, precisamos executar as repetições até que uma célula da coluna A, por exemplo, da aba “Base” seja igual a vazio.

Feito isso, armazenamos em 3 variáveis o mês daquela linha, a plataforma e o volume. Para o primeiro Loop, por exemplo, os valores armazenados estão marcados nas células da imagem abaixo.

	A	B	C	D
1	Mês	Data	Plataforma	Volume Extraído
2	Janeiro	01/01/2015	ODP2	406.846,00
3	Janeiro	01/01/2015	MDP1	55.791,00
4	Janeiro	01/01/2015	MDP2	270.615,00

Por fim, não podemos esquecer de selecionar a aba do mês utilizando a variável **mes** para poder fazer o registro corretamente.

```

Sub compila()
    Sheets("Base").Activate
    linha = 2
    Do Until Cells(linha, 1).Value = ""
        mes = Cells(linha, 1).Value
        plataforma = Cells(linha, 3).Value
        volume = Cells(linha, 4).Value
        Sheets(mes).Activate
        coluna = Cells.Find(plataforma).Column
        linha_registro = Cells(1000000, coluna).End(xlUp).Row + 1
        Cells(linha_registro, coluna).Value = volume
        Sheets("Base").Activate
        linha = linha + 1
    Loop
End Sub

```



**Não esqueça de atualizar a variável linha com o comando `linha = linha + 1`. Explicamos este detalhe repetidas vezes nas aulas de Do Until e Do While. Se você não fizer isso, a linha sempre será igual a 2, a célula analisada no teste do Until sempre será `Cells(2, 1).Value` e essa célula nunca será igual a vazio. Sem essa `linha = linha + 1` seu código entrará em Loop infinito!!!**

A seguir, precisamos encontrar a coluna correta de acordo com a plataforma. Para isso, usamos um novo conceito: o **Cells.Find**. Este comando faz exatamente a mesma coisa que a ferramenta Localizar (atälho CTRL + L) do Excel: ele procura por um texto em todas as células da planilha e seleciona a célula onde aquele texto está escrito. Como não queremos a célula e sim a coluna da célula contendo a informação, usamos a propriedade `.Column` em conjunto. No exemplo abaixo, o `Cells.Find` vai encontrar a plataforma ODP2 no primeiro Loop.

The screenshot shows a Microsoft Excel spreadsheet with data in columns A through I. Row 1 is labeled 'Extração' and contains values MDP1, MDP2, MDP3, ODP1, ODP2, ODP3, ODP4. Row 2 contains values 1, 2, 3, 4, 5, 6, 7, 8, 9. Row 3 contains values 2, 3, 4, 5, 6, 7, 8, 9, 10. A 'Localizar e substituir' (Find & Replace) dialog box is open over the spreadsheet. The 'Localizar' (Find) input field has 'ODP2' typed into it. A red arrow points from this text in the dialog to the cell F2 in the spreadsheet, which is highlighted with a green border. The dialog also includes buttons for 'Substituir' (Replace), 'Opções' (Options), 'Localizar tudo' (Find All), 'Localizar próxima' (Find Next), and 'Fechar' (Close).

A coluna será armazenada na variável **coluna**. Usamos também o comando `End(xlUp)` para encontrar a última linha preenchida daquela coluna, assim como já fizemos em outros exercícios anteriores.

```
Sub compila()

    For Each aba In ThisWorkbook.Sheets

        If aba.Name <> "Base" Then

            aba.Activate

            Range("B2:H100000").ClearContents

        End If

    Next

    Sheets("Base").Activate

    linha = 2

    Do Until Cells(linha, 1).Value = ""
```

Até a página anterior, o nosso código já estava finalizado. O único detalhe que falta é, no início do código, limpar todas as células das tabelas das abas dos meses. Assim, será possível sempre fazer uma nova compilação cada vez que o código for executado.

Esse acréscimo é mostrado ao lado. De resto, o código é exatamente igual ao da página anterior.

## Solução DO UNTIL

```
Sub compila()
    For Each aba In ThisWorkbook.Sheets
        If aba.Name <> "Base" Then
            aba.Activate
            Range("B2:H100000").ClearContents
        End If
    Next
    Sheets("Base").Activate
    linha = 2
    Do Until Cells(linha, 1).Value = ""
        mes = Cells(linha, 1).Value
        plataforma = Cells(linha, 3).Value
        volume = Cells(linha, 4).Value
        Sheets(mes).Activate
        coluna = Cells.Find(plataforma).Column
        linha_registro = Cells(1000000, coluna).End(xlUp).Row + 1
        Cells(linha_registro, coluna).Value = volume
        Sheets("Base").Activate
        linha = linha + 1
    Loop
End Sub
```

## Solução FOR

```
Sub compila2()
    For Each aba In ThisWorkbook.Sheets
        If aba.Name <> "Base" Then
            aba.Activate
            Range("B2:H100000").ClearContents
        End If
    Next
    Sheets("Base").Activate
    ult_linha = Range("A1").End(xlDown).Row
    For linha = 2 To ult_linha
        mes = Cells(linha, 1).Value
        plataforma = Cells(linha, 3).Value
        volume = Cells(linha, 4).Value
        Sheets(mes).Activate
        coluna = Cells.Find(plataforma).Column
        linha_registro = Cells(1000000, coluna).End(xlUp).Row + 1
        Cells(linha_registro, coluna).Value = volume
        Sheets("Base").Activate
    Next
End Sub
```

1 Mês	B	C	D	F	G	H	I	J	K	L	M
2 Janeiro	207960	33538	251761	374599	109425	120461	335989				
	259871	311186	497851	157947	182586	78651	51805	5846	260381	319305	
3 Janeiro	32746	302793	51840	310134	297984	364237	405843	4792	381245	227180	
4	376167	13351	142712	301856	32798	139042	283026				
5	456919	70014	7931	89083	325578	97153	18524				
6	153448	420627	446023	404146	372289	217854	192567				
7	123133	337950	245197	35048	299072	272673	31875				
8	396038	273007	138497	424597	242348	273568	311128				
9	98992	33101	166436	115689	53459	88261	153761				
10	234257	484072	484710	417750	320099	109385	461333				
11	450246	148754	171709	90895	387829	350208	187213				
12	311538	252601	77491	238265	374814		433156				
13	173511	355249	94683	491786		348183					
14	397957	392804	24947		116911		161389				
15	18223	181468	237025	441631		129817					
16	432906	300325	260332			470571					
17	311827		443039			95509					
18			57915			309774					
19						441491					
20	272149	440940	137040	304774	109448	311944	209700				
21	196225	52018	463226	440042	130999	72051	102320				
22	186629	278409	389407	464276	109068	494772	149158				
23	359393	471717	281252	229174	349665	168348	252740				
24	177144	411316	398668	12012	162982	204941	209639				

Ao executar qualquer uma das duas macros, você verá a sua tela piscando e os valores sendo registrados de maneira sequencial. O seu código irá demorar, possivelmente, de 1 a 2 minutos até finalizar.

Por mais que este tempo ainda seja muito inferior do que levaríamos se fossemos fazer essa tarefa de forma manual, poderíamos torná-la ainda mais rápida fazendo algumas otimizações.

Para a primeira dica de otimização do código, precisamos entender um detalhe importante. Em geral, diversas operações do VBA exigem bastante processamento do seu computador, principalmente códigos complexos ou com estruturas de repetição que irão executar uma série de comandos centenas de vezes.

Pensando nisso, existem técnicas que podem ser aplicadas no código a fim de obter um melhor desempenho do mesmo. Nesta e na próxima página exemplificamos duas:

### 1. Desativando a atualização de tela

Ao executar um código VBA, o padrão do Excel é atualizar a tela com as alterações do seu código, como por exemplo, alteração ou seleção de células e abas. Isso faz com que o código fique muito lento, tornando a execução pouco otimizada.

Para desativar esta atualização de tela, basta incluir no início do seu código a propriedade:

#### **Application.ScreenUpdating**

Definida como falsa, e incluindo no final do código a sua ativação novamente, definindo-a como True, para voltar ao padrão após a execução da macro.

```
Sub Minha_Macro()
```

```
    Application.ScreenUpdating = False
```

```
    Seu código...
```

```
    Application.ScreenUpdating = True
```

```
End Sub
```

Para a primeira dica de otimização do código, precisamos entender um detalhe importante. Em geral, diversas operações do VBA exigem bastante processamento do seu computador, principalmente códigos complexos ou com estruturas de repetição que irão executar uma série de comandos centenas de vezes.

Pensando nisso, existem técnicas que podem ser aplicadas no código a fim de obter um melhor desempenho do mesmo. Nesta e na próxima página exemplificamos duas:

### 2. Desativando o cálculo automático

Em situações onde a planilha possui diversas fórmulas que exigem uma capacidade maior de processamento, normalmente as macros ficam bem mais lentas devido ao padrão do Excel para o cálculo de fórmulas, que é automático. Podemos desativar esta propriedade durante o andamento do código e depois retornar ao estado padrão. Para desabilitar o cálculo automático, basta inserir no início do seu código a propriedade:

**Application.Calculation = xlCalculationManual**

Incluindo o **xlCalculationAutomatic** ao final do código para retornar ao cálculo automático padrão.

*Sub Minha\_Macro()*

*Application.Calculation = xlCalculationManual*

*Seu código...*

*Application.Calculation = xlCalculationAutomatic*

*End Sub*

```
Sub compila()
    Application.ScreenUpdating = False
    Application.Calculation = xlCalculationManual

    For Each aba In ThisWorkbook.Sheets
        If aba.Name <> "Base" Then
            aba.Activate
            Range("B2:H100000").ClearContents
        End If
    Next

    Sheets("Base").Activate
    linha = 2

    Do Until Cells(linha, 1).Value = ""
        mes = Cells(linha, 1).Value
        plataforma = Cells(linha, 3).Value
        volume = Cells(linha, 4).Value

        Sheets(mes).Activate
        coluna = Cells.Find(plataforma).Column
        linha_registro = Cells(1000000, coluna).End(xlUp).Row + 1
        Cells(linha_registro, coluna).Value = volume

        Sheets("Base").Activate
        linha = linha + 1
    Loop

    Application.ScreenUpdating = False
    Application.Calculation = xlCalculationAutomatic
End Sub
```

Incluindo as otimizações explicadas anteriormente, o nosso código final está mostrado ao lado.

```
Sub compila()

Application.ScreenUpdating = False
Application.Calculation = xlCalculationManual

For Each aba In ThisWorkbook.Sheets

    If aba.Name <> "Base" Then
        aba.Range("B2:H100000").ClearContents
    End If
Next

linha = 2

Do Until Sheets("Base").Cells(linha, 1).Value = ""

    mes = Sheets("Base").Cells(linha, 1).Value
    plataforma = Sheets("Base").Cells(linha, 3).Value
    volume = Sheets("Base").Cells(linha, 4).Value

    coluna = Sheets(mes).Cells.Find(plataforma).Column

    linha_registro = Sheets(mes).Cells(1000000, coluna).End(xlUp).Row + 1

    Sheets(mes).Cells(linha_registro, coluna).Value = volume

    linha = linha + 1

Loop

Application.ScreenUpdating = False
Application.Calculation = xlCalculationAutomatic

End Sub
```

Outra modificação que podemos fazer é referente à seleção de abas no código. Até agora, quando precisávamos executar um comando em alguma aba, primeiro ativávamos a aba para somente depois executar o comando.

```
Sheets(mes).Activate

coluna = Cells.Find(plataforma).Column

linha_registro = Cells(1000000, coluna).End(xlUp).Row + 1

Cells(linha_registro, coluna).Value = volume
```

Porém, isto não é necessário. Podemos utilizar a estrutura abaixo para executar os comandos diretamente na aba desejada, sem a necessidade de ativá-la.

```
coluna = Sheets(mes).Cells.Find(plataforma).Column

linha_registro = Sheets(mes).Cells(1000000, coluna).End(xlUp).Row + 1

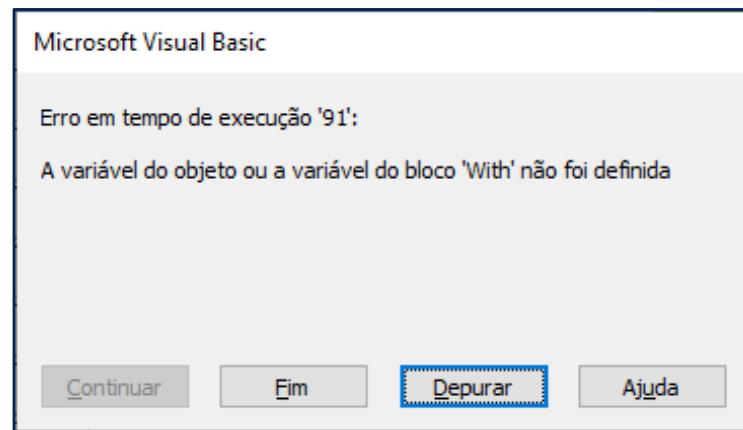
Sheets(mes).Cells(linha_registro, coluna).Value = volume
```

O resultado final com todas as modificações é mostrado ao lado.

Módulo 8

# Tratamento de Erros

Imagine que você terminou de desenvolver aquele projeto que era o projeto e você está ansioso para mostrar para o seu chefe a planilha que você desenvolveu. Você testou o seu código diversas vezes, ele aparentemente está funcionando perfeitamente, porém, quando você manda o arquivo para o seu chefe e ele aperta no botão para executar a macro, 1 minuto depois ele te envia uma mensagem com o print abaixo:



Naturalmente, você vai entrar em desespero e começar a imaginar como deve ser a vida de desempregado.

Erros no VBA acontecem a todo o momento, então não pense que esta será a sua carta de demissão. Ninguém irá te demitir por isso, a menos que nenhuma das suas macros funcione. Neste ponto, temos que entender que, tão importante quanto criar macros muito rápidas e otimizadas é se antecipar aos possíveis erros que podem ocorrer durante a sua execução e realizar os devidos tratamentos no seu código.

Basicamente existem duas maneiras de tratar erros no VBA, por meio das duas estruturas abaixo:

(1) **On Error Resume Next**

(2) **On Error GoTo Label**

O On Error Resume Next, ao encontrar um erro, simplesmente o ignora e passa para a próxima linha de código.

Já o On Error GoTo Label, ao encontrar um erro, executa algum comando pré-especificado no ponto do código onde encontrar a palavra-chave **Label**.

Para começar, vamos entender como funciona a estrutura **On Error Resume Next**.

A	B	C	D	
1	Funcionário	Quantidade Serviços	Serviços Finalizados	% Finalizado
2	Ailton Castro Morett Ceppas	7	4	
3	Ana Marinho	3	0	
4	Bianca Ferezin	0	0	
5	Bruna Chein	7	3	
6	Eduardo Ferreira Xavier	0	0	
7	Gabriel Simões Ribeiro	5	1	
8	Júlio Almeida Reis	7	6	
9	Luana Santana	0	0	
10	Maike da Silva Pereira	8	6	
11	Myllena Negrelli	8	0	
12	Pedro Delgado	7	7	
13	Ulisses Quinto	9	7	
14	Vitor Almeida Campos	3	2	
15				
16				
17				

Neste primeiro exemplo vamos calcular o percentual de serviços finalizados, que nada mais é do que a quantidade de serviços finalizados dividido pela quantidade de serviços total.

Para isso, teremos que utilizar uma estrutura de repetição.

```
Sub serviços()
    Dim linha As Integer
    linha = 2
    Do Until Cells(linha, 1).Value = ""
        Cells(linha, 4).Value = Cells(linha, 3).Value / Cells(linha, 2).Value
        linha = linha + 1
    Loop
End Sub
```

A solução final está mostrada ao lado.

```

Sub serviços()
    Dim linha As Integer
    linha = 2
    Do Until Cells(linha, 1).Value = ""
        Cells(linha, 4).Value = Cells(linha, 3).Value / Cells(linha, 2).Value
        linha = linha + 1
    Loop
End Sub

```

```

Sub serviços()
    Dim linha As Integer
    linha = 2
    Do Until Cells(linha, 1).Value = ""
        Cells(linha, 4).Value = Cells(linha, 3).Value / Cells(linha, 2).Value
        linha = linha + 1
    Loop
End Sub

```

Porém, quando executamos o código, ele exibe uma mensagem de erro. Ao clicar em “Depurar” ele vai marcar a linha onde fazemos a divisão dos valores.

Se passarmos o mouse em cima da variável linha, repare que o erro ocorreu no momento em que linha = 4. Ao investigar essa linha na tabela, vemos que, como a quantidade de serviços é igual a zero, então ao fazer 0 / 0 o código retorna um erro. Repare que, apesar da linha 3 ter funcionado corretamente, o problema é quando a quantidade de serviços é igual a zero, pois temos uma divisão por zero.

	A	B	C	D
1	Funcionário	Quantidade Serviços	Serviços Finalizados	% Finalizado
2	Ailton Castro Morett Ceppas	7	4	57%
3	Ana Marinho	3	0	0%
4	Bianca Ferezin	0	0	
5	Bruna Chein	7	3	
6	Eduardo Ferreira Xavier	0	0	

```
Sub serviços()  
    On Error Resume Next  
    Dim linha As Integer  
    linha = 2  
    Do Until Cells(linha, 1).Value = ""  
        Cells(linha, 4).Value = Cells(linha, 3).Value / Cells(linha, 2).Value  
        linha = linha + 1  
    Loop  
End Sub
```

Para contornar esse problema, utilizamos a estrutura On Error Resume Next logo no início do nosso código. Isso fará com que qualquer erro encontrado seja ignorado e a próxima linha seja executada.

O resultado final é mostrado abaixo:

A	B		C	
1	Funcionário	Quantidade Serviços	Serviços Finalizados	% Finalizado
2	Ailton Castro Morett Ceppas	7	4	57%
3	Ana Marinho	3	0	0%
4	Bianca Ferezin	0	0	
5	Bruna Chein	7	3	43%
6	Eduardo Ferreira Xavier	0	0	
7	Gabriel Simões Ribeiro	5	1	20%
8	Júlio Almeida Reis	7	6	86%
9	Luana Santana	0	0	
10	Maike da Silva Pereira	8	6	75%
11	Myllena Negrelli	8	0	0%
12	Pedro Delgado	7	7	100%
13	Ulisses Quinto	9	7	78%
14	Vitor Almeida Campos	3	2	67%
15				

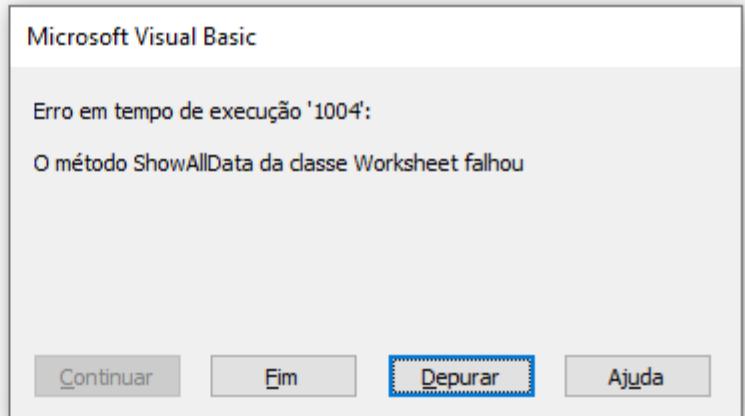
A	B	C	D	E
1 Descrição	Unidades	Litros	Perda	Origem
2 SKU 1	785	981	157	Nacional
4 SKU 3	851	1064	170	Nacional
5 SKU 4	587	734	117	Nacional
7 SKU 6	113	141	23	Nacional
8 SKU 7	364	455	73	Nacional
9 SKU 8	1000	1250	200	Nacional
10 SKU 9	607	759	121	Nacional
12 SKU 11	463	579	93	Nacional
13 SKU 12	853	1066	171	Nacional
17 SKU 16	153	191	31	Nacional
18 SKU 17	593	741	119	Nacional
19 SKU 18	564	705	113	Nacional
20 SKU 19	207	259	41	Nacional
22 SKU 21	316	395	63	Nacional
23 SKU 22	882	1103	176	Nacional
24 SKU 23	237	296	47	Nacional
26 SKU 25	818	1023	164	Nacional
27 SKU 26	428	535	86	Nacional
28 SKU 27	107	134	21	Nacional
29 SKU 28	645	806	129	Nacional
30 SKU 29	648	810	130	Nacional
31 SKU 30	565	706	113	Nacional
32 SKU 31	810	1013	162	Nacional
33 SKU 32	722	903	144	Nacional

Limpar Filtros

No próximo exemplo vamos gravar uma macro para percorrer todas as abas e retirar todos os filtros das tabelas. O código para retirar os filtros está mostrado abaixo.

```
Sub limpa_abas ()  
    For Each aba In ThisWorkbook.Sheets  
        aba.Activate  
        ActiveSheet.ShowAllData  
    Next  
End Sub
```

```
Sub limpa_abas()
    For Each aba In ThisWorkbook.Sheets
        aba.Activate
        ActiveSheet.ShowAllData
    Next
End Sub
```



```
Sub limpa_abas()
    For Each aba In ThisWorkbook.Sheets
        aba.Activate
        ActiveSheet.ShowAllData
    Next
End Sub
```

A	B	C	D	E
1 Funcionário	Data de Contratação	Idade	Cargo	Salário
2 Adriana de Azevedo	14/nov/15	53	Gerente 2	12320
3 Adriana Mosqueira Rodrigues Lopes	14/nov/15	36	Estagiário 1	1600
4 Adriana Passos	14/nov/15	40	Gerente 2	12320
5 Adriana Torres Carneiro	14/nov/15	42	Estagiário 2	2000
6 Adriane Chagas	14/nov/15	48	Estagiário 2	2000
7 Adriane de Carvalho Gomes	14/nov/15	58	Analista Senior	4650
8 Adriane Pinheiro	14/nov/15	61	Coordenador 2	8100
9 Adriano De Biase	14/nov/15	35	Analista Júnior	2850
10 Adriano Novaes Silva	14/nov/15	50	Coordenador 2	8100
11 Adrielle Figueiredo Gab	...	...	...	...
12 Adrielle Penna Forte	...	...	...	...
13 Adrielle Vieira	...	...	...	...

Porém, ao tentar executar, a macro exibe um erro. Isso porque nem todas as tabelas das abas estão filtradas. Assim, quando ele chega em uma tabela sem filtros (por exemplo, na aba “Base Funcionários”, a macro retorna um erro.

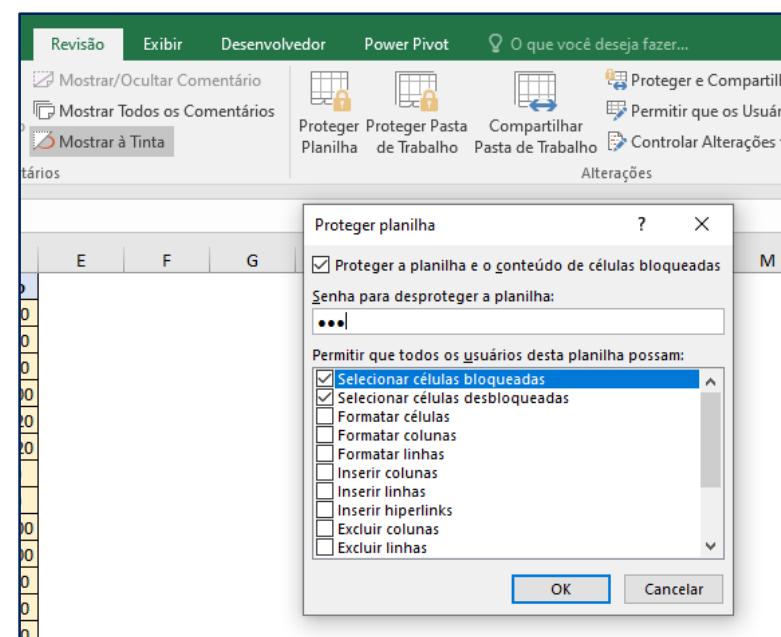
```
Sub limpa_abas()
    On Error Resume Next
    For Each aba In ThisWorkbook.Sheets
        aba.Activate
        ActiveSheet.ShowAllData
    Next
End Sub
```

Sabemos que, se a tabela não possuir filtros, então não precisamos nos preocupar em retirar os filtros. Assim, podemos ignorar este erro utilizando mais uma vez a estrutura On Error Resume Next no início do nosso código.

A	B	C	D	
1	Funcionário	Idade	Cargo	Salário
2	Allan Paes de Sousa Guedes	61	Coordenador 2	R\$8.100
3	Ananda Coimbra de Gouvêa	62	Gerente 1	R\$9.450
4	Anízio Vieira	31	Analista Júnior	R\$2.850
5	Anna del Prado Kling	33	Gerente 3	R\$16.300
6	Bernardo Ribeiro	61	Gerente 2	R\$12.320
7	Caio Teixeira Caldas	23	Gerente 2	R\$12.320
8	Camila da Silva	55	Jovem Aprendiz	R\$800
9	Diego de Barros	35	Jovem Aprendiz	R\$800
10	Emilaine Mota	62	Gerente 3	R\$16.300
11	Gabriel Rebouças	30	Gerente 3	R\$16.300
12	Lucas Rodrigues Villanova	31	Estagiário 2	R\$2.000
13	Luiz Vieira Branco de Matos	27	Analista Júnior	R\$2.850
14	Matheus da Silva Miranda	40	Coordenador 2	R\$8.100
15	Matheus Testahy Barros Afonso	61	Estagiário 2	R\$2.000
16	Rachel Restum	50	Estagiário 1	R\$1.600
17	Rubens Valente	28	Gerente 1	R\$9.450
18	Thaís Tardem Pereira	32	Analista Pleno	R\$3.700
19	Thiago Costa	32	Analista Júnior	R\$2.850
20				
21				
22				
23				
24				
25				

Funcionarios

Em alguns casos, ignorar os erros pode não ser a melhor opção. Vamos entender porquê. Imagina que a gente tenha uma planilha com senha e queremos registrar uma nova informação. Para simular esse problema, você pode aplicar uma senha à planilha na guia Revisão, em Proteger Planilha. Pode colocar qualquer senha, como por exemplo, 123.



```
Sub cadastro()

funcionario = InputBox("Digite o nome do novo funcionário")
idade = InputBox("Digite a idade do novo funcionário")
cargo = InputBox("Digite o cargo do novo funcionário")
salario = InputBox("Digite o salario do novo funcionário")

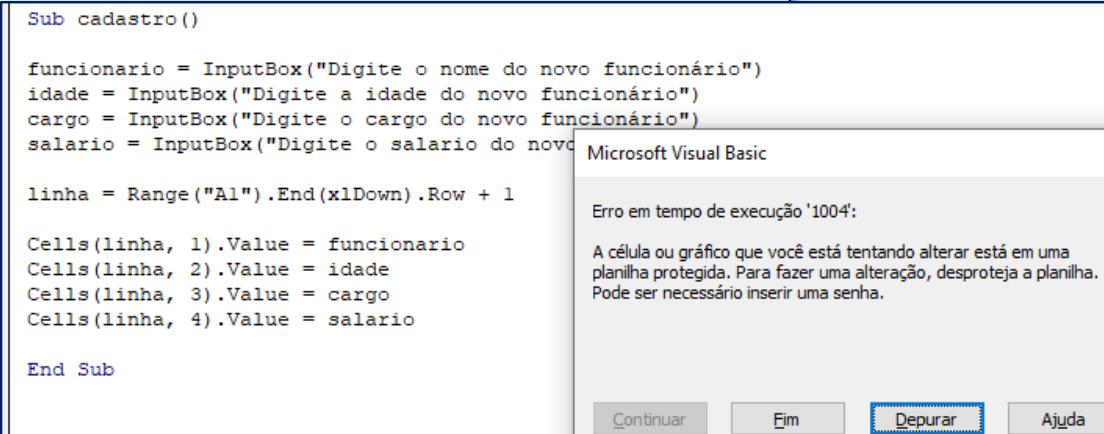
linha = Range("A1").End(xlDown).Row + 1

Cells(linha, 1).Value = funcionario
Cells(linha, 2).Value = idade
Cells(linha, 3).Value = cargo
Cells(linha, 4).Value = salario

End Sub
```

O código que cadastra um novo funcionário e o seu respectivo nome, idade, cargo e salário é mostrada ao lado.

Porém, ao tentar executá-la, o código retorna um erro. Isso porque a planilha está protegida e assim não é possível adicionar novas informações nela.



```
Sub cadastro()

On Error GoTo tratar

funcionario = InputBox("Digite o nome do novo funcionário")
idade = InputBox("Digite a idade do novo funcionário")
cargo = InputBox("Digite o cargo do novo funcionário")
salario = InputBox("Digite o salario do novo funcionário")

linha = Range("A1").End(xlDown).Row + 1

codigo:

Cells(linha, 1).Value = funcionario
Cells(linha, 2).Value = idade
Cells(linha, 3).Value = cargo
Cells(linha, 4).Value = salario

ActiveSheet.Protect Password:=123
|
Exit Sub

tratar:

ActiveSheet.Unprotect

GoTo codigo

End Sub
```

Aqui não podemos usar o On Error Resume Next pois este erro não podemos ignorar, já que, querendo ou não, precisaremos informar a senha antes de cadastrar um novo funcionário.

Para resolver este problema, usamos a estrutura **GoTo**.

A estrutura GoTo permite que a gente vá até um determinado ponto do código (exemplo, ir para o ponto **tratar**) e neste ponto escrevemos os códigos que queremos executar em caso de erro, neste caso, solicitar a senha para o desbloqueio da planilha.

Em seguida, para voltar para o ponto onde preenchemos os valores nas células, usamos um outro GoTo (**GoTo codigo**). Em seguida, protegemos novamente a planilha com uma senha.

Por fim, para o código não executar novamente o trecho **tratar**, simplesmente finalizamos a macro com o código **Exit Sub**.

# Módulo 9

# Function

O VBA possui dois tipos de estruturas para a construção de códigos: **Sub** e **Function**.

Uma **Sub** executa uma série de comandos.

Uma **Function** executa uma série de comandos e retorna um valor como resposta.

Até o momento criamos apenas Subs. Apesar de parecer algo absolutamente novo, uma Function é muito parecida com uma fórmula qualquer no Excel. Ela vai ter um nome, vai pedir alguns argumentos e vai retornar um único valor como resposta. A grande vantagem é que, através de Functions, seremos capazes de criar fórmulas que não existem prontas no Excel, realizando cálculos que não conseguiríamos fazer por não existir uma fórmula com essa finalidade.

- **Function no VBA**

**Function nome\_da\_funcao (arg1 As Tipo, arg2 As Tipo, arg3 As Tipo, ...) As Tipo**

**'comandos**

**nome\_da\_funcao = valor que vai ser retornado como resposta**

**End Function**



**Não só os argumentos devem ter um tipo definido mas a Function também, conforme o valor de retorno dela.**

Como exemplo bem simples, a **Function ao lado calcula a área de um retângulo/quadrado**. Ela pede dois argumentos: x e y, e esses argumentos devem ser do tipo Double. Isso significa que se a pessoa escrever um texto como argumento a Function não vai funcionar. Além disso, a função em si é do tipo Double, porque ao final ela retornará um resultado do tipo Double.

Em seguida, temos o valor de retorno:  $\text{Area} = x * y$

A Function não é executada assim como uma Sub. Para utilizar a function, você deve ir até uma célula do Excel e escrever como se fosse uma fórmula.



Functions também são criadas em Módulos exatamente igual às Subs!

- **Função que calcula a área de um quadrilátero**

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

$$\text{Area} = x * y$$

End Function

	A	
1	=area	
2	<i>f<sub>x</sub></i> Area	

	A	
1	=Area(2;3)	
2		

	A	
1		6

Um outro exemplo seria criar uma function para **converter uma temperatura de graus Celsius para graus Fahrenheit**. Aposto que nunca na sua vida você conseguiu decorar essa fórmula no ensino médio. É mentira? Então, imagina se na época você tivesse aprendido sobre as functions do VBA com a Hashtag? Aposto que não teria sofrido tanto em química...

Resumindo, Functions permitem que a gente crie fórmulas personalizadas para realizar cálculos que não seriam possíveis com uma fórmula existente no VBA.

- **Função que converte °C em °F**

Function **CparaF (tempC As Double) As Double**

Dim tempF As Double

tempF = 9 \* tempC / 5 + 32

**CparaF = tempF**

End Function

A	
1	=cpa
2	<small>(fx)</small> CparaF

A	
1	=CparaF(20)
2	

A	
1	68
2	

A	B	C	D	E
2	Crie uma Function VBA que some os dois valores da tabela			
4	Valor 1	Valor 2	Soma	
5				
6				
7				

```
Function minhasoma(num1 As Double, num2 As Double) As Double  
  
minhasoma = num1 + num2  
  
End Function
```

A	B	C	D	E
2	Crie uma Function VBA que some os dois valores da tabela			
4	Valor 1	Valor 2	Soma	
5	5		=minhasoma(B5;C5)	
6				

Vamos começar com um exemplo bem simples. Queremos criar uma Function que irá somar dois valores.

O código é bem simples e também está mostrado ao lado.

Criada a Function, é só você ir até uma célula do Excel e passar para a fórmula os argumentos necessários.

A	B	C	D	E
2	Crie uma Function VBA que some os dois valores da tabela			
4	Valor 1	Valor 2	Soma	
5	5	7	12	
6				

A	B	C	D	E
1				
2				
3				
4	Valor 1	Valor 2	Soma	
5	5		=minhasoma(	
6				
7				

Você deve ter reparado que, diferente das fórmulas que existem no Excel, ao iniciar uma Function não aparecem os nomes dos argumentos da fórmula.

A	B	C	D	E
1				
2				
3				
4	Valor 1	Valor 2	Soma	
5	5		=minhasoma(num1;num2)	
6				

Para mostrar quais são os argumentos da Function, após inicia-la, basta usar o atalho:

**CTRL + SHIFT + A**

E os argumentos irão aparecer na fórmula.

	A	B	C	D	E	F
1	Crie uma Function VBA que concatene os valores					
3	Primeiro Nome	Segundo Nome	Sobrenome	CPF	Informações Completas	
4	Diego	Amorim	dos Santos	140.472.294-62		
5						

```
Function meuconcatenar(intervalo As Range) As String
    Dim texto As String
    texto = ""
    For Each celula In intervalo
        texto = texto & celula.Value
    Next
    meuconcatenar = texto
End Function
```

No próximo exemplo, queremos criar uma função que receba um único argumento: um intervalo de células. E para cada célula desse intervalo, queremos concatenar para criar uma única célula com todas as informações daquele intervalo.

A solução final é mostrada ao lado. Porém, se você reparar, os textos das células foram concatenados, mas todos colados. A ideia é que a gente acrescente um espaço entre as palavras para melhorar a visualização na célula.

	A	B	C	D	E	F
1	Crie uma Function VBA que concatene os valores					
3	Primeiro Nome	Segundo Nome	Sobrenome	CPF	Informações Completas	
4	Diego	Amorim	dos Santos	140.472.294-62	Amorimdos Santos140.472.294-62	
5						

```
Function meuconcatenar(intervalo As Range) As String  
  
Dim texto As String  
  
texto = ""  
  
For Each celula In intervalo  
  
    texto = texto & " " & celula.Value  
  
Next  
  
meuconcatenar = texto  
  
End Function
```

Podemos então concatenar um espaço entre a variável texto e a célula celula.Value do intervalo. Essa variável texto é apenas uma variável para armazenar os valores concatenados ao longo do For Each para que depois seja possível passa-lo como resposta da função meuconcatenar.

Ao refazer a função no Excel, aparentemente está certo. Porém, se verificarmos a fundo (copiar a célula F4 e colar como valor) vemos que existe um espaço sobrando no início da palavra.

Isso porque, no primeiro Loop do For Each, a variável texto estava vazia, mas ao concatenar com o " " e celula.Value, foi inserido um " " logo no começo.

	A	B	C	D	E	F
1	Crie uma Function VBA que concatene os valores					
3	Primeiro Nome	Segundo Nome	Sobrenome	CPF	Informações Completas	
4	Diego	Amorim	dos Santos	140.472.294-62	Diego Amorim dos Santos 140.472.294-62	
5					Diego Amorim dos Santos 140.472.294-62	
6						



```
Function meuconcatenar(intervalo As Range) As String  
  
Dim texto As String  
  
texto = ""  
  
For Each celula In intervalo  
  
    If texto = "" Then  
  
        texto = texto & celula.Value  
  
    Else  
  
        texto = texto & " " & celula.Value  
  
    End If  
  
Next  
  
meuconcatenar = texto  
  
End Function
```

O problema desses espaços indesejados dentro dos textos é que, em situações onde você precisar encontrar este nome na planilha, esse espaço a mais pode dificultar esta busca na hora de comparar com informações de outras tabelas.

Para corrigir isso, precisamos modificar o nosso código para que apenas quando a variável **texto** possuir algum valor diferente de vazio deverá ser adicionado este espaço entre o texto e celula.Value.

O resultado final é mostrado ao lado.

O enunciado do próximo exercício é mostrado na imagem abaixo. Antes de ir para a solução final na próxima página, tente praticar seus conhecimentos para criar a Function necessária.

	A	B	C	D	E	F	G	H	I	J
1										
2	Crie uma Function VBA que calcule o salário com imposto dos funcionários sob a ótica do empregador. O cálculo do salário com imposto é feito somando o imposto ao valor do salário. Assim, se o salário sem imposto do funcionário é R\$15.000, seu salário com imposto será: R\$ 15.000 + 10% * R\$15.000 (isso porque estamos analisando a ótica do empresário). Depois, crie uma Sub que vai calcular o valor do salário para todos os funcionários da empresa (de todas as abas).									
3										
4										
5										
6	Nome	Hora Normal	Hora Extra	Salário Com Imposto						
7	Fabio	500	100							
8	Sofia	1000	100							
9	Álvaro	900	50							
10	Sofia	1200	500							
11	Camila	800	100							
12										
13										
14	<b>Calcular Salário Com Imposto</b>									
15										
16										
17										
18										

R\$/hora normal	R\$ 15,00
R\$/hora extra	R\$ 25,00
<b>Renda</b>	<b>Imposto (% Renda)</b>
Até R\$ 12.000	-
De R\$12.000 até R\$ 18.000	10%
Acima de R\$18.000	12,5%

O exercício está dividido em duas partes: criar uma function para calcular o salário com impostos na primeira aba e depois uma Sub para utilizar essa function em cada uma das outras abas e calcular os respectivos salários com impostos.

A solução para a primeira parte, de criar a Function, é mostrada no gabarito abaixo. Como precisamos de informações para que a function funcione, ela irá receber um total de 4 argumentos, pois ela precisará saber: quantidade de horas trabalhadas, quantidade de horas extras, preço das horas trabalhadas e preço das horas extras.

```
Function SalarioComImposto(qtd_normal As Double, qtd_extra As Double, preco_normal As Double, preco_extra As Double) As Double  
  
    salario_total = qtd_normal * preco_normal + qtd_extra * preco_extra  
  
    If salario_total <= 12000 Then  
  
        SalarioComImposto = salario_total  
  
    ElseIf salario_total| <= 18000 Then  
  
        SalarioComImposto = salario_total * 1.1  
  
    Else  
  
        SalarioComImposto = salario_total * 1.125  
  
    End If  
  
End Function
```

No Excel, a fórmula ficará assim. Lembrando que, antes de arrastar a fórmula para baixo, precisamos travar as células H6 e H7 com um cifrão para que elas não se movimentem junto com a fórmula.

Crie uma Function VBA que calcule o salário com imposto dos funcionários sob a ótica do empregador. O cálculo do salário com imposto é feito valor do salário. Assim, se o salário sem imposto do funcionário é R\$15.000, seu salário com imposto será: R\$ 15.000 + 10% \* R\$15.000 (isso p analisando a ótica do empresário). Depois, crie uma Sub que vai calcular o valor do salário para todos os funcionários da empresa (de todas as :

Nome	Hora Normal	Hora Extra	Salário Com Imposto
Fabio	500	=SalarioComImposto(C7;D7;\$H\$6;\$H\$7)	
Sofia	1000	100	
Álvaro	900	50	
Sofia	1200	500	
Camila	800	100	

Calcular Salário Com

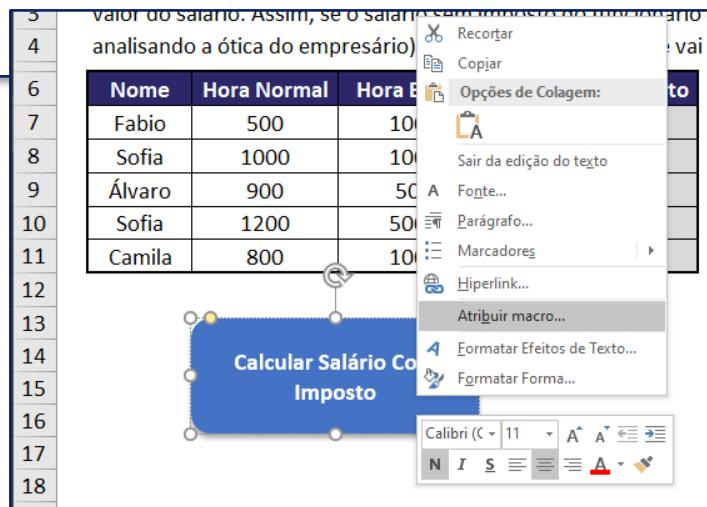
R\$/hora normal	R\$ 15,00
R\$/hora extra	R\$ 25,00

Renda	Imposto (% Renda)
Até R\$ 12.000	-

Nome	Hora Normal	Hora Extra	Salário Com Imposto
Fabio	500	100	10.000,00
Sofia	1000	100	19.250,00
Álvaro	900	50	16.225,00
Sofia	1200	500	34.312,50
Camila	800	100	15.950,00

Melhor do que ter que digitar a fórmula no Excel é criar uma macro que irá fazer isso automaticamente: preencher com a Function os valores das células sem que a gente precise entrar no Excel, digitar a fórmula, trancar as células e copiar as fórmulas para as demais células. Você também pode atribuir a macro ao botão da planilha para tornar a execução ainda mais automática.

```
Sub calcula_salario()
    For linha = 7 To 11
        Cells(linha, 5).Value = SalarioComImposto(Cells(linha, 3).Value, Cells(linha, 4).Value, Cells(6, 8).Value, Cells(7, 8).Value)
    Next
End Sub
```

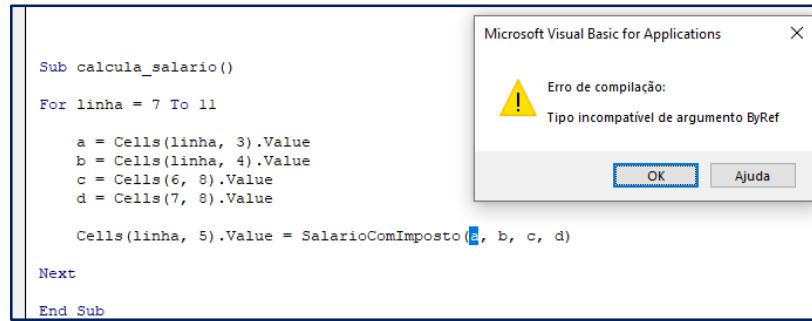


Nome	Hora Normal	Hora Extra	Salário Com Imposto
Fabio	500	100	10.000,00
Sofia	1000	100	19.250,00
Álvaro	900	50	16.225,00
Sofia	1200	500	34.312,50
Camila	800	100	15.950,00

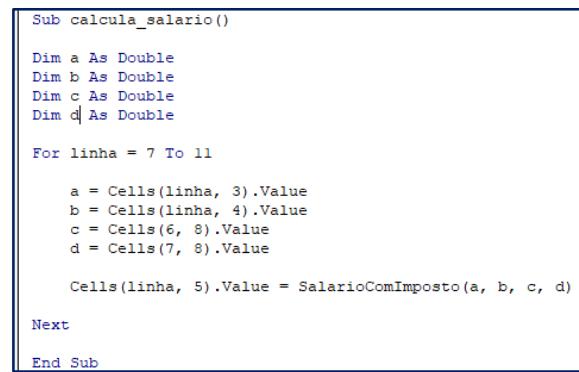


### ATENÇÃO !

Caso você tenha decidido por armazenar em variáveis os 4 argumentos utilizados na Function, você provavelmente se deparou com este erro:



Isso acontece porque, ao passar os argumentos como variáveis diretamente na Function, eles precisam ter um tipo bem definido (no caso, a nossa Function precisa de argumentos do tipo Double). Porém, em nenhum momento do código declaramos o tipo destas variáveis. Assim, ao trabalhar com variáveis e Functions, precisamos obrigatoriamente declarar estas variáveis no início do código:



	A	B	C	D	E
1	Nome	Hora Normal	Hora Extra	Salário com imposto	
2	Gabriel Mesquita	800	100		
3	João Haddad	1200	170		
4	Amanda Marques Ribeiro	1400	280		
5	Guilherme Nunez	800	90		
6	Adelino Gomes	700	70		
7	Audir de Avila Goulart	800	110		
8	Zilma Guimarães	500	60		
9	Gil Bonder	1100	160		
10	Gustavo de Melo Teixeira	1500	190		
11	Andre Campos	600	90		
12	André Melo Soledade	700	140		
13	Natalia Morgan Loureiro	1100	160		
14	Luca Costa	1300	160		
15	Jessica Oliveira Lima	1000	100		
16	Caio Scalabrin	700	90		
17	Vinicius Freitas	1000	200		
18	Pedro Carrera	1300	220		
19	Isabella Bernardo	1500	250		
20	Felipe Sousa Melo	1200	130		
21	Pedro Fontes	600	60		
22	Maria Lobo	900	100		
23	Bruno Vargas	700	100		
24	Bárbara Spenchutt Vieira	1300	220		
25	Andre Ramos	1100	180		

Ainda neste arquivo temos diversas abas de áreas diferentes. Queremos utilizar a Function que acabamos de criar para registrar o salário com impostos de todos os funcionários, de todas as abas, de uma maneira automática.

Se você quiser praticar, você pode criar uma nova Sub e fazer o seu código antes de passar para a próxima página com a solução final. Você já consegue fazer, basta confiar.

```
Sub compila_funcionarios()

Dim qtd_normal As Double
Dim qtd_extra As Double
Dim preco_normal As Double
Dim preco_extra As Double

Sheets("Exemplo Funcionários").Activate

qtd_normal = Range("H6").Value
qtd_extra = Range("H7").Value

For Each aba In ThisWorkbook.Sheets

    If aba.Name <> "Exemplo Funcionários" Then

        aba.Activate

        linha = 2

        Do Until Cells(linha, 1).Value = ""

            preco_normal = Cells(linha, 2).Value
            preco_extra = Cells(linha, 3).Value

            Cells(linha, 4).Value = SalarioComImposto(qtd_normal, qtd_extra, preco_normal, preco_extra)

            linha = linha + 1

        Loop

    End If

Next| 

End Sub
```

A solução final é mostrada ao lado.

Lembrando que, como utilizamos na solução variáveis para os 4 argumentos da função **SalarioComImposto**, precisamos declarar estas variáveis no começo do nosso código.

Todas as estruturas utilizadas já foram exercitadas ao longo do curso. Portanto, se tiver alguma dúvida, é só voltar nas páginas respectivas às estruturas utilizadas.

Podemos dizer que a principal vantagem de uma Function é que ela pode ser criada apenas uma vez e sempre que uma macro precisar executar aquela mesma rotina, é possível chamá-la no nosso código, deixando-o muito mais legível e otimizado.

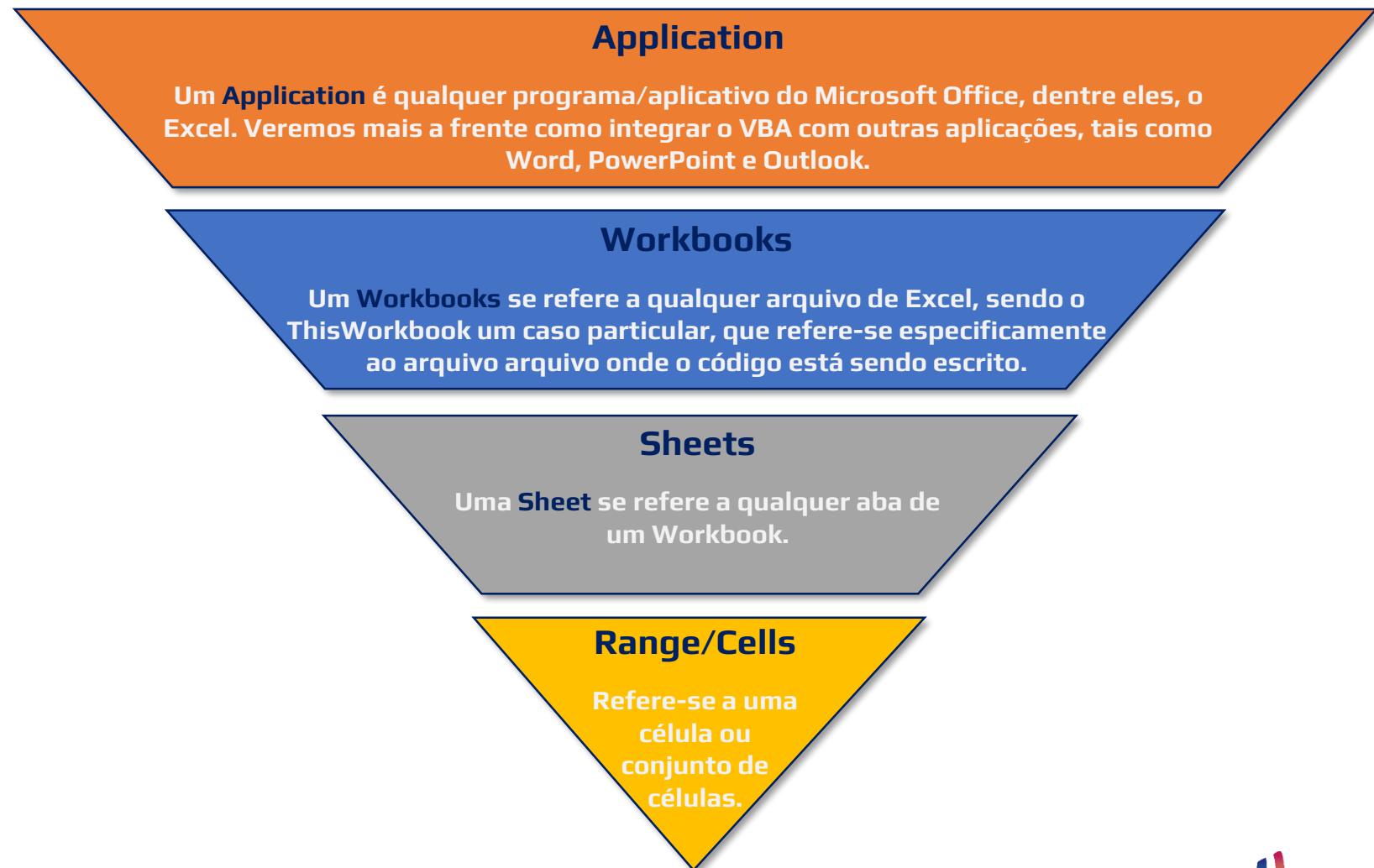
**Módulo 10**

**Trabalhando com  
vários arquivos**

- **Hierarquia do VBA**

Neste módulo veremos como podemos trabalhar com diferentes arquivos do Excel no nosso computador. Até agora vimos como executar diversas ações em células e abas de um arquivo. Porém, em boa parte das nossas aplicações, precisamos manipular diferentes arquivos Excel de uma vez, como por exemplo, fechar vários arquivos de uma vez, copiar dados de vários arquivos e compilar em um único, criar novos arquivos, e por ai vai.

O que precisamos começar a entender para dar prosseguimento é que **existe uma hierarquia no Excel, composta por 4 grandes objetos** e tudo o que é feito em um objeto afetará igualmente todos os que estiverem abaixo na hierarquia.



- **Application**

Cada um desses objetos possuem suas propriedades específicas. Exemplos para o Application são mostrados abaixo (alguns até já conhecidos):

- **Application.Quit**

Fecha todos os arquivos abertos do Microsoft Excel.

- **Application.DisplayFullScreen = True**

Aplica o modo Tela Inteira ao arquivo Excel.

- **Application.ScreenUpdating = False**

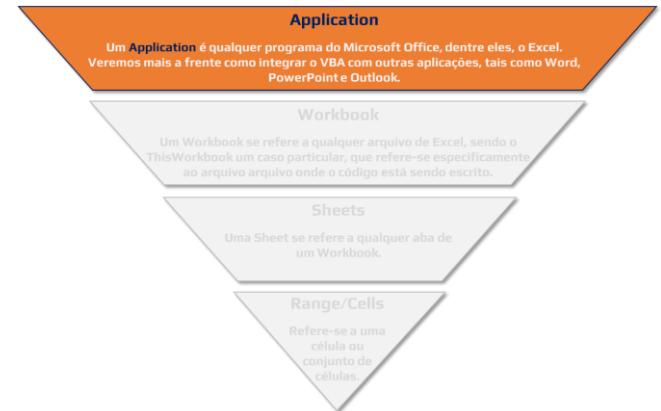
Desativa a atualização de tela do arquivo Excel.

- **Application.DisplayAlerts = False**

Exibe ou oculta as caixas de mensagem que o arquivo emite durante a execução da macro.

- **Application.Calculation = xlCalculationManual**

Modifica o cálculo automático do Excel para manual.



- **Workbooks**

Para o objeto Workbooks temos as seguintes possibilidades:

- **Workbooks.Open**

Abre um arquivo Excel.

- **Workbook.Save**

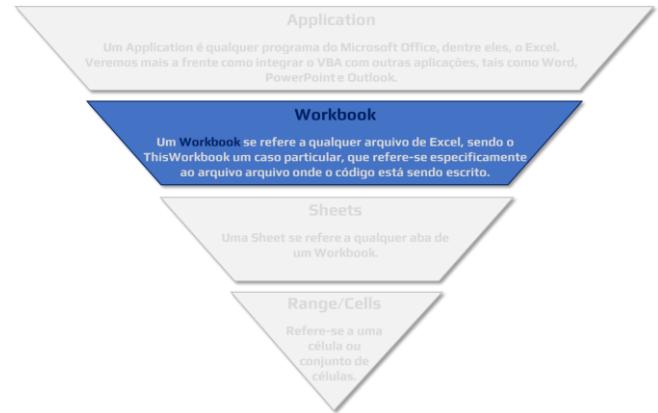
Salva um arquivo Excel.

- **Workbook.Add**

Cria um novo arquivo Excel.

- **Workbooks.Close**

Fecha um arquivo Excel.



- **Sheets**

Para o objeto Sheets temos as seguintes possibilidades:

- **Sheets.Add**

Cria uma nova aba no arquivo Excel.

- **Sheets.Delete**

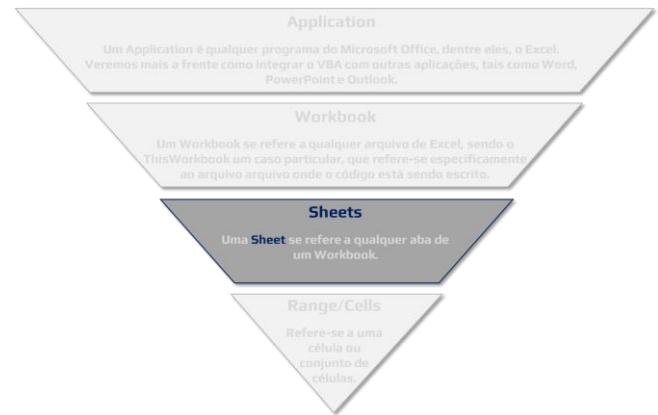
Exclui uma aba.

- **Sheets.Visible = False**

Oculta uma aba do Excel.

- **Sheets.Select**

Seleciona uma aba.



- **Range/Cells**

Para o objeto Range/Cells temos as seguintes possibilidades:

- **Range("A1:C3").Copy**

Copia o intervalo de células.

- **Range("A:A").ClearContents**

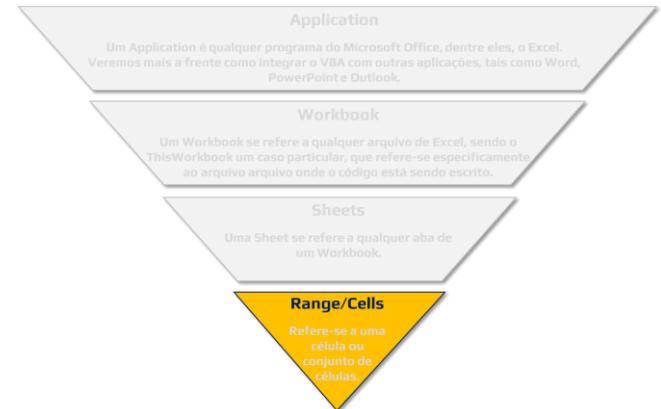
Limpa o conteúdo da célula sem excluir a célula.

- **Range("A1").AddComment**

Adiciona um comentário na célula.

- **Cells(1, 1).Value = "VBA"**

Escreve em uma célula.



# Módulo 10 – Trabalhando com vários arquivos – Exercício Arquivos

306

The image shows two Microsoft Excel windows side-by-side. The left window is titled "Resumo Geral.xlsxm - Excel" and displays a table of employees with columns "Nome", "Área", "Salário", and "Idade". The right window is titled "Areas.xlsxm - Excel" and also displays a similar table of employees with the same columns. A black rectangular button with the text "Registrar Novo Funcionário" is overlaid on the left window, positioned over the employee data table.

	A	B	C	D	E	F	G
1	Nome	Área	Salário	Idade			
2	Gabriel Mesquita	Industrial	4658	36			
3	Hanna Vaz	Industrial	3986	48			
4	Mathheus Alvarez	Industrial	3997	39			
5	Felipe Rovetta	Administrativo	1600	33			
6	Adriano Carneiro Golcaives	Administrativo	2530	33			
7	Yuske Ferreira	Administrativo	3686	31			
8	William Cunha Felipe	Comercial	4871	49			
9	Thalles Jesus Freitas Silva	Comercial	4686	45			
10	Victor Freire de Souza	Comercial	2007	45			
11	Olívia e Carvalho	Comercial	4102	37			
12	Júlia Santoro Silva	Logística	4849	39			
13	Norman Andrade Souza	Logística	4972	32			
14	Vicente Dias	Logística	1236	48			
15	Breno Moreira	Logística	1332	39			
16	Paulo Rodrigues Pereira	Industrial	1311	39			
17	Pedro Alcoforado	Industrial	1721	36			
18	Lucas Alvarenga dos Santos	Industrial	3881	35			
19	Bernardo Costa Silva	Administrativo	4737	35			
20	Arthur de Oliveira	Comercial	4707	40			
21	Bruno Vargas	Industrial	1937	38			
22	Jéssica Neves Heimlich	Industrial	3030	43			
23	Patrick da Silva Farias	Industrial	1065	31			
24	Roberto Leite	Industrial	3055	34			
25	Yasmini de Almeida Richa	Industrial	4909	42			
26	Ramon Garcia Bittencourt	Industrial	4367	45			
27	Julia Bach	Industrial	3872	45			
28	Miguel Reis	Administrativo	3672	49			
29	Vitor Neves Bastos	Administrativo	1813	50			
30	Leila Freire	Logística	2582	48			

Resumo Funcionários

	A	B	C	D	E	F	G
1	Nome	Área	Salário	Idade			
2	Gabriel Mesquita	Industrial	4658	36			
3	João Haddad	Industrial	1202	46			
4	Amanda Marques Ribeiro	Industrial	2033	50			
5	Guilherme Nunez	Industrial	1600	49			
6	Adelino Gomes	Industrial	4226	49			
7	Audir de Avila Goulart	Industrial	2928	36			
8	Zilma Guimarães	Industrial	3610	36			
9	Gil Bonder	Industrial	2224	30			
10	Gustavo de Melo Teixeira	Industrial	2429	30			
11	Andre Campos	Industrial	2270	32			
12	André Melo Soledade	Industrial	4457	42			
13	Natalia Morgan Loureiro	Industrial	1141	42			
14	Luca Costa	Industrial	3710	40			
15	Jessica Oliveira Lima	Industrial	4239	38			
16	Caio Scalabrin	Industrial	4912	48			
17	Vinicius Freitas	Industrial	3301	37			
18	Pedro Carrera	Industrial	1359	41			
19	Isabella Bernardo	Industrial	4751	44			
20	Felipe Sousa Melo	Industrial	1951	48			
21	Pedro Fontes	Industrial	1806	38			
22	Maria Lobo	Industrial	3111	33			
23	Bruno Vargas	Industrial	1937	38			
24	Bárbara Spennuth Vieira	Industrial	2047	43			
25	André Ramos	Industrial	3850	37			
26	Livia Melo	Industrial	2317	43			
27	Jéssica Neves Heimlich	Industrial	3030	43			
28	Carolina Monteiro	Industrial	4287	49			
29	Luis Werneck	Industrial	2570	41			
30	Cícero Barcelos Taubari	Industrial	2623	37			

Em nosso primeiro exercício temos dois arquivos: o da esquerda, chamado “Resumo Geral”, possui uma tabela com todos os funcionários de uma determinada empresa.

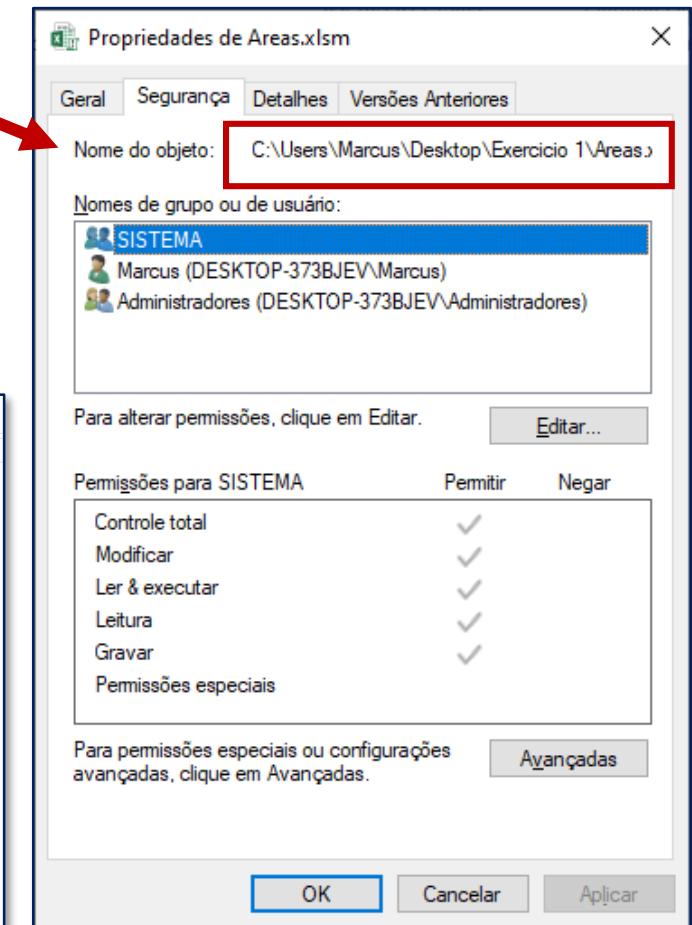
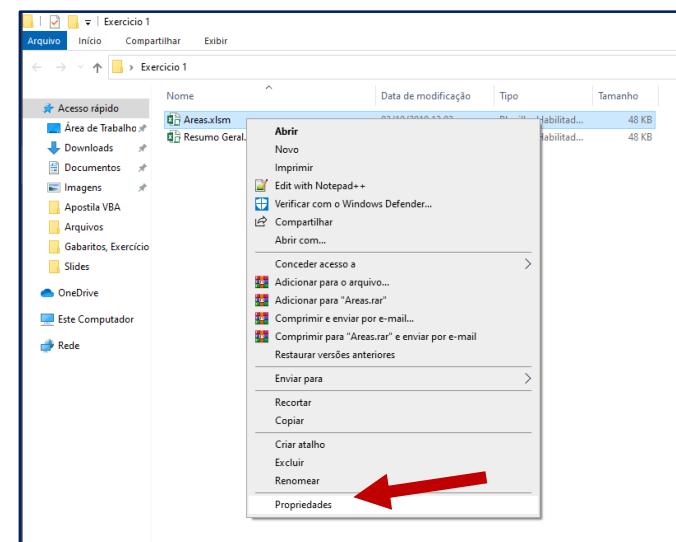
Devemos, em primeiro lugar, construir uma macro que seja capaz de adicionar um novo funcionário nesta tabela, mas que também seja capaz de abrir o arquivo da direito (“Areas”) e registrar o funcionário na aba correta, de acordo com a sua área.

```
Sub cadastra_funcionario()

Workbooks.Open (
    Open(Filename As String, [UpdateLi
    [Delimiter], [Editable], [Notify], [Conve
End Sub
```

Vamos começar o nosso código vendo como abrir um arquivo Excel. Para isso, usamos o comando Workbooks.Open. Será necessário informar o caminho onde o arquivo que queremos abrir está localizado.

Para saber este caminho, tem uma forma bem simples: basta você abrir uma janela do seu Windows Explorer, localizar o arquivo que deseja abrir (Areas.xlsx), clicar nele com o botão direito e ir na opção Propriedades. Em seguida, na aba Segurança, copie o caminho que aparece em Nome do objeto.



```
Sub cadastra_funcionario()
    Workbooks.Open ("C:\Users\Marcus\Desktop\Exercicio 1\Areas.xlsx")
End Sub
```

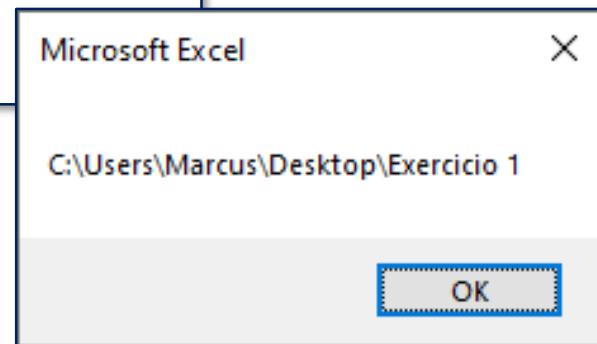
Chamamos a atenção para dois detalhes:

1. O nome do caminho deve estar entre aspas.
2. Ao final do nome do arquivo também é necessário escrever a extensão deste arquivo, no caso, .xlsx.

O primeiro código ao lado já é capaz de abrir o arquivo do seu computador. Porém, ele não está 100% automatizado, pois se enviássemos este arquivo para uma outra pessoa, ela também teria que copiar o caminho e colar no código, o que seria zero prático.

```
Sub cadastr_funcionario()
    'caminho arquivo Áreas
    Workbooks.Open ("C:\Users\Marcus\Desktop\Exercicio 1\Areas.xlsx")
    'caminho arquivo Resumo Geral
    Workbooks.Open ("C:\Users\Marcus\Desktop\Exercicio 1\Resumo Geral.xlsx")
End Sub
```

```
Sub cadastr_funcionario()
    caminho = ThisWorkbook.Path
    'caminho arquivo Áreas
    Workbooks.Open (caminho & "\Areas.xlsx")
    MsgBox (caminho.Path)
End Sub
```



Para melhorar esse código, o primeiro passo é que os dois arquivos (“Áreas” e “Resumo Geral”) devem estar na mesma pasta.

Assim, uma vez que os dois arquivos estão na mesma pasta, podemos reparar que o caminho dos dois é exatamente igual, mudando apenas o nome do arquivo, obviamente.

Uma maneira de conseguir descobrir facilmente o caminho do arquivo onde estamos criando o código é por meio do comando `ThisWorkbook.Path`.

Podemos ver, pela `MsgBox`, que este comando retorna exatamente o caminho da pasta do arquivo, que podemos utilizar armazenar em uma variável e concatenar com o nome do arquivo.

```
Sub cadastra_funcionario()

ult_linha = Range("A1").End(xlDown).Row + 1

Cells(ult_linha, 1).Value = InputBox("Digite o nome do novo funcionário")
Cells(ult_linha, 2).Value = InputBox("Digite a área do novo funcionário")
Cells(ult_linha, 3).Value = InputBox("Digite o salário do novo funcionário")
Cells(ult_linha, 4).Value = InputBox("Digite a idade do novo funcionário")

nome = Cells(ult_linha, 1).Value
area = Cells(ult_linha, 2).Value
salario = Cells(ult_linha, 3).Value
idade = Cells(ult_linha, 4).Value

caminho = ThisWorkbook.Path

Workbooks.Open (caminho & "\Areas.xlsm")

End Sub
```

Continuando a nossa macro, devemos pensar o que deve ser feito antes de abrir o arquivo “Áreas”.

Como queremos começar registrando o novo funcionário no arquivo “Resumo Geral” onde estamos digitando o nosso código, devemos:

1. Descobrir a última linha da tabela para preencher um novo funcionário logo abaixo.
2. Criar 4 InputBox para preencher as células da tabela onde queremos armazenar as informações.
3. Para podermos escrever no arquivo “Áreas”, armazenamos as informações em 4 variáveis.

```
Sub cadastr_funcionario()

ult_linha = Range("A1").End(xlDown).Row + 1

Cells(ult_linha, 1).Value = InputBox("Digite o nome do novo funcionário")
Cells(ult_linha, 2).Value = InputBox("Digite a área do novo funcionário")
Cells(ult_linha, 3).Value = InputBox("Digite o salário do novo funcionário")
Cells(ult_linha, 4).Value = InputBox("Digite a idade do novo funcionário")

nome = Cells(ult_linha, 1).Value
area = Cells(ult_linha, 2).Value
salario = Cells(ult_linha, 3).Value
idade = Cells(ult_linha, 4).Value

caminho = ThisWorkbook.Path

Workbooks.Open (caminho & "\Areas.xlsm")

Sheets(area).Activate

linha_registro = Range("A1").End(xlDown).Row + 1

Cells(linha_registro, 1).Value = nome
Cells(linha_registro, 2).Value = area
Cells(linha_registro, 3).Value = salario
Cells(linha_registro, 4).Value = idade

End Sub
```

Após abrir o arquivo, devemos ativar a aba certa para registrar o novo funcionário. Esta aba será de acordo com a área informada na InputBox anterior.

Em seguida, seguimos o mesmo procedimento de encontrar a linha de registro nessa aba e depois escrevemos em cada uma das respectivas células os valores que armazenamos nas variáveis.

```
Sub cadastrando_funcionario()

ult_linha = Range("A1").End(xlDown).Row + 1

Cells(ult_linha, 1).Value = InputBox("Digite o nome do novo funcionário")
Cells(ult_linha, 2).Value = InputBox("Digite a área do novo funcionário")
Cells(ult_linha, 3).Value = InputBox("Digite o salário do novo funcionário")
Cells(ult_linha, 4).Value = InputBox("Digite a idade do novo funcionário")

nome = Cells(ult_linha, 1).Value
area = Cells(ult_linha, 2).Value
salario = Cells(ult_linha, 3).Value
idade = Cells(ult_linha, 4).Value

caminho = ThisWorkbook.Path

Workbooks.Open (caminho & "\Areas.xlsx")
Sheets(area).Activate

linha_registro = Range("A1").End(xlDown).Row + 1

Cells(linha_registro, 1).Value = nome
Cells(linha_registro, 2).Value = area
Cells(linha_registro, 3).Value = salario
Cells(linha_registro, 4).Value = idade

ActiveWorkbook.Save
ActiveWorkbook.Close

MsgBox ("Macro executada com sucesso!")

End Sub
```

Para finalizar a macro, devemos salvar e fechar o arquivo Áreas.

Um ponto importante aqui é que, como abrimos o arquivo Áreas através do comando .Open, ele passou a ser o nosso arquivo ativo.

Por isso, para conseguir salvar e fechar o arquivo “Áreas” em vez do arquivo “Resumo Geral”, utilizamos o comando ActiveWorkbook, que executará os comandos de salvar e fechar no arquivo ativo, que no caso, é o “Areas.xlsx”.

Não poderíamos utilizar o comando ThisWorkbook porque ele se trata na verdade do arquivo onde estamos escrevendo o nosso código, que no caso é o “Resumo Geral.xlsx”.

Por fim, podemos exibir para o usuário uma mensagem de finalização da macro.

```
Sub cadastra_funcionario()

ult_linha = Range("A1").End(xlDown).Row + 1

Cells(ult_linha, 1).Value = InputBox("Digite o nome do novo funcionário")
Cells(ult_linha, 2).Value = InputBox("Digite a área do novo funcionário")
Cells(ult_linha, 3).Value = InputBox("Digite o salário do novo funcionário")
Cells(ult_linha, 4).Value = InputBox("Digite a idade do novo funcionário")

nome = Cells(ult_linha, 1).Value
area = Cells(ult_linha, 2).Value
salario = Cells(ult_linha, 3).Value
idade = Cells(ult_linha, 4).Value

caminho = ThisWorkbook.Path

Set arq_areas = Workbooks.Open(caminho & "\Areas.xlsx")

arq_areasActivate
Sheets(area).Activate

linha_registro = Range("A1").End(xlDown).Row + 1

Cells(linha_registro, 1).Value = nome
Cells(linha_registro, 2).Value = area
Cells(linha_registro, 3).Value = salario
Cells(linha_registro, 4).Value = idade

arq_areasSave
arq_areasClose

MsgBox ("Macro executada com sucesso!")

End Sub
```

O problema de trabalhar com vários arquivos ao mesmo tempo é que podemos ter dificuldades de ativar o arquivo certo usando a lógica do código anterior.

Algo mais otimizado seria declarar uma variável que recebe o Workbook. Para isso, utilizariámos o comando Set.

Já falamos sobre este comando na página 176, onde mostramos como declarar uma célula como variável. Inclusive, falamos no segundo parágrafo desta página que usamos o Set para declarar qualquer objeto no Excel: células, abas, gráficos ou arquivos (Workbooks).

Isso nos permitirá o controle de selecionar o arquivo desejado, ou salvar um arquivo sem necessariamente que ele esteja ativado.

## Módulo 10 – Trabalhando com vários arquivos – Exercício Estoque

The image shows three Excel windows side-by-side:

- Base Vendas.xlsxm - Excel**: Shows a table of sales data with columns: Nº da Venda, Data, Marca, Valor, Quantidade em Estoque, and Disponibilidade no Estoque. A macro is running, indicated by a yellow tooltip "Registrar Venda".
- Estoque.xlsxm - Excel**: Shows a table of stock quantities for different motorcycle brands: Honda (1), Yamaha (7), Kawasaki (2), BMW (0), Suzuki (3), Dafra (0), Flash (5), Sundown (1), and Vespa (0).
- Base Vendas.xlsxm - Excel** (bottom): Shows a smaller table of motorcycle prices: Moto (Honda \$5.570, Yamaha \$6.480, Kawasaki \$20.890, BMW \$29.800, Suzuki \$6.490, Dafra \$7.500, Flash \$10.000, Sundown \$13.500, Vespa \$27.900).

A red arrow points from the bottom-left table to the "Vendas Diárias" tab in the main window, indicating the source of data for the sales table.

Neste próximo exercício queremos poder registrar uma venda nova venda na aba **Vendas Diárias** do arquivo **Base Vendas**.

Além disso, a macro deve procurar corretamente o preço da moto vendida na aba **Dados** do arquivo **Base Vendas**.

Por fim, a cada venda, o arquivo de estoque deve ser consultado para sabermos se ainda há estoque daquela moto, se tiver, registra a venda com o status disponível, caso contrário, registra como indisponível no arquivo **Base Vendas**.

```
Sub registra_venda()

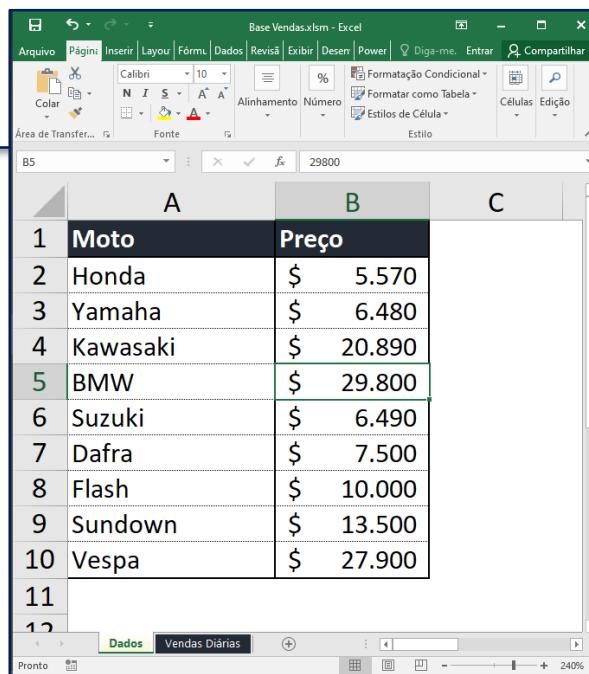
    marca = InputBox("Digite o nome da marca")

    Sheets("Dados") .Activate

    linha_marca = Cells.Find(marca) .Row
    preco_marca = Cells(linha_marca, 2) .Value

    |

End Sub
```



A screenshot of Microsoft Excel showing a table titled "Base Vendas.xlsx". The table has two columns: "Moto" and "Preço". The data is as follows:

	A	B
1	Moto	Preço
2	Honda	\$ 5.570
3	Yamaha	\$ 6.480
4	Kawasaki	\$ 20.890
5	BMW	\$ 29.800
6	Suzuki	\$ 6.490
7	Dafra	\$ 7.500
8	Flash	\$ 10.000
9	Sundown	\$ 13.500
10	Vespa	\$ 27.900
11		
12		

Começamos preparando a base para o nosso código. Primeiro criamos uma variável marca para receber a marca da moto que será vendida.

Em seguida, selecionamos a aba "Dados" para descobrir o preço da moto. Usamos a estrutura Find + Row para encontrar a linha daquela moto e, como sabemos que o preço está na segunda coluna, basta usar o Cells.

Se você ficou com alguma dúvida sobre a estrutura Find + Row, basta voltar na página 266 que é o momento onde explicamos esta combinação.

```
Sub registra_venda()  
  
    marca = InputBox("Digite o nome da marca")  
  
    Sheets("Dados").Activate  
  
    linha_marca = Cells.Find(marca).Row  
    preco_marca = Cells(linha_marca, 2).Value  
  
    Sheets("Vendas Diárias").Activate  
  
    Workbooks.Open (ThisWorkbook.Path & "\Estoque.xlsx")  
|  
End Sub
```

Voltamos para a aba “Vendas Diárias”, pois será a aba onde faremos o registro da venda.

Em seguida, abrimos o arquivo “Estoque.xlsx” utilizando a mesma lógica com o ThisWorkbook.Path.

Mais uma vez, aqui é importante que tanto o arquivo “Base Vendas” quanto o arquivo “Estoque” estejam na mesma pasta!

```
Sub registra_venda()

    marca = InputBox("Digite o nome da marca")

    Sheets("Dados").Activate

    linha_marca = Cells.Find(marca).Row
    preco_marca = Cells(linha_marca, 2).Value

    Sheets("Vendas Diárias").Activate

    Workbooks.Open(ThisWorkbook.Path & "\Estoque.xlsxm")

    linha_estoque = Cells.Find(marca).Row
    qtd_estoque = Cells(linha_estoque, 2).Value

    If qtd_estoque > 0 Then

        Cells(linha_estoque, 2).Value = Cells(linha_estoque, 2).Value - 1

    End If

    ActiveWorkbook.Save
    ActiveWorkbook.Close

End Sub
```

Após abrir o arquivo de estoque, precisamos:

1. Descobrir a linha da marca.
2. Descobrir a quantidade em estoque.
3. Testar se o estoque é maior que zero. Se for, diminuímos 1 devido a venda feita, caso contrário, não fazemos nada pois não há estoque.
4. Por fim, devemos salvar o arquivo e fechá-lo.

	Marca	Quantidade em Estoque
1	Honda	1
2	Yamaha	7
3	Kawasaki	2
4	BMW	0
5	Suzuki	3
6	Dafra	0
7	Flash	5
8	Sundown	1
9	Vespa	0
10		
11		
12		
13		
14		
15		

```

Sub registra_venda()

marca = InputBox("Digite o nome da marca")

Sheets("Dados").Activate

linha_marca = Cells.Find(marca).Row
preco_marca = Cells(linha_marca, 2).Value

Sheets("Vendas Diárias").Activate

Workbooks.Open (ThisWorkbook.Path & "\Estoque.xlsx")

linha_estoque = Cells.Find(marca).Row
qtd_estoque = Cells(linha_estoque, 2).Value

If qtd_estoque > 0 Then

    Cells(linha_estoque, 2).Value = Cells(linha_estoque, 2).Value - 1

End If

ActiveWorkbook.Save
ActiveWorkbook.Close

linha_registro = Range("A1").End(xlDown).Row + 1

Cells(linha_registro, 1).Value = linha_registro - 1
Cells(linha_registro, 2).Value = Date
Cells(linha_registro, 3).Value = marca
Cells(linha_registro, 4).Value = preco_marca
Cells(linha_registro, 5).Value = qtd_estoque

If qtd_estoque > 0 Then
    Cells(linha_registro, 6).Value = "Disponível"
Else
    Cells(linha_registro, 6).Value = "Indisponível"
End If

End Sub

```

Para finalizar o código, após salvar e fechar o arquivo de Estoque, é necessário:

1. Descobrir a última linha do registro na aba “Vendas Diárias”.
2. Cadastrar todas as informações em cada coluna.
3. Para a coluna de disponibilidade, testar se a quantidade em estoque antes da venda é maior que zero. Se for, o status é “Disponível”. Se não, o status é “Indisponível”.

O código finalmente está finalizado e você já pode atribuí-lo ao botão e testar o seu funcionamento.

	A	B	C	D	E	F
1	Nº da Venda	Data	Marca	Valor	Quantidade em Estoque	Disponibilidade no Estoque
2	1	03/07/2019	Yamaha	R\$ 6.480,00	9	Disponível
3	2	04/07/2019	Vespa	R\$ 27.900,00	3	Disponível
4	3	06/07/2019	Sundown	R\$ 13.500,00	2	Disponível
5	4	08/07/2019	Vespa	R\$ 27.900,00	2	Disponível
6	5	08/07/2019	Dafra	R\$ 7.500,00	1	Disponível
7	6	09/07/2019	Yamaha	R\$ 6.480,00	8	Disponível
8	7	11/07/2019	Vespa	R\$ 27.900,00	1	Disponível
9	8	12/07/2019	Sundown	R\$ 13.500,00	0	Indisponível

# Módulo 10 – Trabalhando com vários arquivos – Percorrendo arquivos da pasta

319

The screenshot shows two Excel windows side-by-side. The left window is titled '0 - Compilado.xlsxm - Excel' and its 'Compilado' sheet contains a table with columns: SKU, Tamanho Pedido, Loja, and Data da Venda. A button labeled 'Compilar Vendas' is located in the center of this sheet. The right window is titled '1 - Janeiro.xlsx - Excel' and its 'Janeiro' sheet shows the same table structure with data for January. Below the windows, a file explorer window is open, displaying a list of files in a folder, including various Excel files like '0 - Compilado.xlsxm', '1 - Janeiro.xlsx', '2 - Fevereiro.xlsx', etc., and a text file 'Anotações Gerais.txt'.

No próximo exercício, queremos fazer algo totalmente novo: percorrer todos os arquivos de vendas em uma pasta do computador e compilar os dados no nosso arquivo “Compilado.xlsxm”.

Cada arquivo representa as vendas de um mês específico e todos eles possuem exatamente a mesma estrutura, mudando apenas os valores. Além disso, ainda há outros arquivos que não nos interessam e que vamos precisar ignorar.

Os comandos que vamos aprender são realmente muito legais e que com certeza vão otimizar ainda mais as suas macros.

Em primeiro lugar, como vamos trabalhar com uma pasta do nosso computador, precisamos declarar esta pasta no nosso código. Esta pasta, assim como células, abas e arquivos é um objeto especial, então precisamos utilizar o comando Set para inicializa-lo. Vamos chamar a nossa variável de **pasta**.

O código necessário para criar um objeto do tipo “pasta de um computador” é o **CreateObject(“Scripting.FileSystemObject”)**. Este comando é padrão sempre que trabalhamos neste tipo de situação. Além disso, para especificar a pasta do computador que ele deve armazenar, usamos o comando GetFolder e passamos o caminho atual do arquivo onde estamos escrevendo o código porque nele temos todos os arquivos que serão utilizados.

```
Sub compilacao()
    Set pasta = CreateObject("Scripting.FileSystemObject").GetFolder(ThisWorkbook.Path)
    For Each arquivo In pasta.Files
        |
        Next
End Sub
```

Além disso, precisamos também criar a nossa estrutura de repetição para percorrer cada um dos arquivos na nossa pasta. A estrutura de repetição será o For Each e está mostrada no código acima.

```
Sub compilacao()
    Set pasta = CreateObject("Scripting.FileSystemObject").GetFolder(ThisWorkbook.Path)
    For Each arquivo In pasta.Files
        If InStr(arquivo.Name, ".xlsx") > 0 Then
            |
        End If
    Next
End Sub
```

Nome	Data de modificação	Tipo	Tamanho
0 - Compilado.xlsxm	09/10/2019 13:13	Planilha Habilidade...	46 KB
1 - Janeiro.xlsx	09/10/2019 13:10	Planilha do Micro...	15 KB
2 - Fevereiro.xlsx	09/10/2019 13:10	Planilha do Micro...	13 KB
3 - Março.xlsx	09/10/2019 13:10	Planilha do Micro...	13 KB
4 - Abril.xlsx	09/10/2019 13:10	Planilha do Micro...	14 KB
5 - Maio.xlsx	09/10/2019 13:10	Planilha do Micro...	13 KB
6 - Junho.xlsx	09/10/2019 13:10	Planilha do Micro...	13 KB
7 - Julho.xlsx	09/10/2019 13:10	Planilha do Micro...	15 KB
8 - Agosto.xlsx	09/10/2019 13:10	Planilha do Micro...	14 KB
9 - Setembro.xlsx	09/10/2019 13:10	Planilha do Micro...	14 KB
10 - Outubro.xlsx	09/10/2019 13:10	Planilha do Micro...	13 KB
11 - Novembro.xlsx	09/10/2019 13:10	Planilha do Micro...	15 KB
12 - Dezembro.xlsx	09/10/2019 13:10	Planilha do Micro...	15 KB
Anotações Gerais.txt	09/10/2019 13:10	Documento de Te...	0 KB
Arquivo de Texto.docx	09/10/2019 13:10	Documento do Mi...	0 KB

Após iniciar a nossa estrutura de repetição, precisamos checar se o arquivo que está sendo analisado durante o loop deve ser de fato aberto. Devemos fazer isso porque temos 4 tipos de arquivo na nossa pasta, como podemos ver na imagem à esquerda.

1. **.xlsm** → Arquivo com macros.
2. **.xlsx** → Arquivo Excel sem macros.
3. **.txt** → Arquivo de texto.
4. **.docx** → Arquivo Word.

Porém, os nossos arquivos de vendas são do tipo Excel, então só vamos querer abrir aqueles com extensão **.xlsx**.

Para isso, utilizamos a função **InStr** que pesquisa por um caractere dentro de um texto e retorna a posição dele. No caso, se o **InStr** retornar um valor maior que zero, significa que ele encontrou uma posição para o texto “**.xlsx**”. (Obs: explicamos sobre a função **InStr** na página 218)

```
Sub compilacao()

Set pasta = CreateObject("Scripting.FileSystemObject").GetFolder(ThisWorkbook.Path)

For Each arquivo In pasta.Files

    If InStr(arquivo.Name, "xlsx") > 0 Then

        linha_compilacao = Range("A100000").End(xlUp).Row + 1

        Workbooks.Open (ThisWorkbook.Path & "\" & arquivo.Name)

    End If

Next

End Sub
```

The screenshot shows a Microsoft Excel window titled "0 - Compilado.xlsxm - Excel". The ribbon is visible at the top with tabs like Arquivo, Página, Inserir, Layout, Fórmula, Dados, Revisão, Exibir, Desenhar, Power, Diga-me, Entrar, Compartilhar, and others. The main area displays a table with four columns: SKU, Tamanho Pedido, Loja, and Data da Venda. The first row has these column headers. Rows 2 through 7 are empty. A black button labeled "Compilar Vendas" is positioned on the right side of the table. The status bar at the bottom shows "Pronto" and "85%". The tab bar at the bottom indicates the active sheet is "Compilado".

Testado o nome do arquivo, se este possui o texto “xlsx” então queremos abri-lo.

Logo após abrir, precisamos descobrir a última linha a ser colado os valores de vendas no arquivo “Compilado”.

Como você pode ver na imagem ao lado, inicialmente a tabela não possui dados, então para descobrir a última linha a ser preenchida, usamos a estrutura End(xlUp) que já vimos anteriormente.

Feito isso, abrimos o arquivo com a instrução Workbooks.Open.

```
Sub compilacao()

Set pasta = CreateObject("Scripting.FileSystemObject").GetFolder(ThisWorkbook.Path)

For Each arquivo In pasta.Files

    If InStr(arquivo.Name, "xlsx") > 0 Then

        linha_compilacao = Range("A100000").End(xlUp).Row + 1

        Workbooks.Open (ThisWorkbook.Path & "\& arquivo.Name)

        linha_copia = Range("A1").End(xlDown).Row

        Range("A2:D" & linha_copia).Copy

        ThisWorkbook.Activate

        Cells(linha_compilacao, 1).PasteSpecial

    End If

Next

End Sub
```

The screenshot shows a Microsoft Excel window titled "1 - Janeiro.xlsx - Excel". The table has four columns: SKU, Tamanho Pedido, Loja, and Data da Venda. The data is as follows:

	A	B	C	D
1	SKU	Tamanho Pedido	Loja	Data da Venda
2	HL1001	2	Fortaleza	01/01/2018
3	HL1021	1	Recife	01/01/2018
4	HL1008	5	Salvador	01/01/2018
5	HL1020	4	Recife	01/01/2018
6	HL1007	4	Campinas	01/01/2018

Após abrir o arquivo, precisamos encontrar a última linha preenchida, para saber até onde devemos copiar as informações. Como esta planilha possui dados, então podemos usar o End(xlDown). Abaixo é mostrado um exemplo de um arquivo de vendas para o mês de janeiro.

Vamos então copiar o intervalo de células e depois ativar o ThisWorkbook (que é o arquivo onde estamos criando o código e onde queremos colar as informações). Por fim, colamos a partir da linha **linha\_compilacao**, que descobrimos anteriormente ser a última linha preenchida na tabela do arquivo de "Compilado".

```
Sub compilacao()

Set pasta = CreateObject("Scripting.FileSystemObject").GetFolder(ThisWorkbook.Path)

For Each arquivo In pasta.Files

    If InStr(arquivo.Name, "xlsx") > 0 Then

        linha_compilacao = Range("A100000").End(xlUp).Row + 1

        Workbooks.Open (ThisWorkbook.Path & "\" & arquivo.Name)

        linha_copia = Range("A1").End(xlDown).Row

        Range("A2:D" & linha_copia).Copy

        ThisWorkbook.Activate

        Cells(linha_compilacao, 1).PasteSpecial

        Application.CutCopyMode = False

        Workbooks(arquivo.Name).Close

    End If

Next

End Sub
```

The screenshot shows a Microsoft Excel spreadsheet titled '0 - Compilado.xlsx'. The data is organized into columns A, B, and C. Column A contains numerical values ranging from 1 to 5. Column B contains city names such as 'Guarulhos', 'São Paulo', 'Rio Horizonte', 'Porto Alegre', 'Salvador', 'João Pessoa', 'Recife', 'Campinas', 'Fortaleza', and 'Porto Velho'. Column C contains dates in the format 'dd/MM/yy' such as '30/08/2018' and '01/09/2018'. The rows are numbered from 2196 to 2254. The status bar at the bottom indicates 'Média: 21681,1359 Contagem: 780 Min: 1 Máx: 43373 Soma: 0455643'.

A	B	C
2196	HL1023	5 Guarulhos 30/08/2018
2197	HL1003	2 São Paulo 30/08/2018
2198	HL1009	4 Rio Horizonte 30/08/2018
2199	HL1021	5 Porto Alegre 30/08/2018
2200	HL1021	5 Rio de Janeiro 30/08/2018
2201	HL1003	5 Salvador 30/08/2018
2202	HL1003	1 João Pessoa 30/08/2018
2203	HL1026	3 Rio de Janeiro 30/08/2018
2204	HL1003	5 Belo Horizonte 30/08/2018
2205	HL1015	5 Recife 31/08/2018
2206	HL1009	2 Campinas 31/08/2018
2207	HL1018	4 Porto Alegre 31/08/2018
2208	HL1021	5 Rio de Janeiro 31/08/2018
2209	HL1003	4 Fortaleza 31/08/2018
2210	HL1003	3 Fortaleza 01/09/2018
2211	HL1007	3 Rio de Janeiro 01/09/2018
2212	HL1003	4 Rio de Janeiro 01/09/2018
2213	HL1021	2 Rio Horizonte 01/09/2018
2214	HL1021	5 Rio de Janeiro 01/09/2018
2215	HL1023	5 Rio Horizonte 01/09/2018
2216	HL1009	1 Salvador 01/09/2018
2217	HL1015	5 João Pessoa 01/09/2018
2218	HL1021	5 Belo Horizonte 01/09/2018
2219	HL1020	4 Recife 01/09/2018
2220	HL1023	4 Fortaleza 01/09/2018
2221	HL1018	4 Rio de Janeiro 01/09/2018
2222	HL1003	3 Belo Horizonte 01/09/2018
2223	HL1003	2 Fortaleza 01/09/2018
2224	HL1003	2 Guarulhos 01/09/2018
2225	HL1018	4 São Paulo 04/09/2018

Após copiar os dados do arquivo aberto, é importante desativar o modo de recorte, que nada mais é do que o tracejado que fica em volta de uma célula quando copiamos ela. Quando formos fechar este arquivo, isso pode dar problemas na nossa macro, pois com o modo de recorte ativo o Excel acha que ainda não terminamos de mexer naquele arquivo, então usamos o comando:

## Application.CutCopyMode = False

Por fim, para não ficarmos com todos os arquivos de vendas abertos, fechamos o arquivo **arquivo.Name**.

Finalmente, você pode executar a macro, ela já vai estar funcionando. Porém, ainda precisaremos fazer alguns ajustes para deixá-la 100%.

# Módulo 10 – Trabalhando com vários arquivos – Percorrendo arquivos da pasta (Parte 6)

325

The screenshot shows an Excel spreadsheet titled "0 - Compilado.xlsxm - Excel". The ribbon is visible with the "Dados" tab selected. A "Classificar" (Sort) dialog box is open, showing the settings: "Classificar por" is set to "Data da Venda" and "Ordem" is set to "Do Mais Antigo para o Mais Novo". The main table in the spreadsheet has columns labeled A, B, C, and D, with data entries such as HL1001 through HL1021 across rows 2 to 30. The "Compilado" sheet is active.

O primeiro ajuste que devemos fazer é garantir que a coluna D de Data da Venda estará ordenada corretamente após a execução da macro.

Para isso, podemos incluir ao final do código os comandos que vão ordenar a tabela por ordem cronológica. Como não conhecemos este código, vamos gravar uma macro para descobri-lo.

Já fizemos anteriormente este exercício de ordenar uma coluna (página 24), mas a ideia basicamente é gravar uma macro em que:

1. Selecione todas as colunas de A até D.
2. Vamos na guia **Dados** → **Classificar** → Classificar por “Data da Venda”.
3. Após clicar em Ok, lembrar de parar a gravação.

```
Sub compilacao()
    Range("A2:D100000").ClearContents 
    Set pasta = CreateObject("Scripting.FileSystemObject").GetFolder(ThisWorkbook.Path)
    For Each arquivo In pasta.Files
        If InStr(arquivo.Name, "xlsx") > 0 Then
            linha_compilacao = Range("A100000").End(xlUp).Row + 1
            Workbooks.Open (ThisWorkbook.Path & "\" & arquivo.Name)
            linha_copia = Range("A1").End(xlDown).Row
            Range("A2:D" & linha_copia).Copy
            ThisWorkbook.Activate
            Cells(linha_compilacao, 1).PasteSpecial
            Application.CutCopyMode = False
            Workbooks(arquivo.Name).Close
        End If
    Next
    Columns("A:D").Select
    ActiveWorkbook.Worksheets("Compilado").Sort.SortFields.Clear
    ActiveWorkbook.Worksheets("Compilado").Sort.SortFields.Add Key:=Range("D2:D100000"), SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
    With ActiveWorkbook.Worksheets("Compilado").Sort
        .SetRange Range("A1:D100000")
        .Header = xlYes
        .MatchCase = False
        .Orientation = xlTopToBottom
        .SortMethod = xlPinYin
        .Apply
    End With
End Sub
```

A macro que gravamos para classificar a coluna de data deve ser colado ao final do código, pois a classificação deve ser feita logo após todos os comandos serem executados.

É importante fazer uma pequena modificação no código para fazer com que ele classifique a coluna de data independente da quantidade de linhas (pois a tabela poderá aumentar). Assim, fazemos com que a classificação seja feita considerando desde a linha 1 até uma linha bem lá embaixo, podendo ser a linha 100000.

Além disso, na primeira linha do código, usamos o comando `ClearContents` para apagar todos os valores da tabela sempre antes de executar a macro, para garantir que sempre serão registrados novos valores, sem sobrepor o que já estava registrado na planilha. Aqui usamos a mesma ideia de apagar desde a linha 2 (pois não queremos apagar a linha 1 de título) até uma linha bem lá embaixo, por exemplo, a 100000.

```
Sub compilacao()
    Application.ScreenUpdating = False
    Application.Calculation = xlCalculationManual

    Range("A2:D100000").ClearContents

    Set pasta = CreateObject("Scripting.FileSystemObject").GetFolder(ThisWorkbook.Path)

    For Each arquivo In pasta.Files
        If InStr(arquivo.Name, "xlsx") > 0 Then
            linha_compilacao = Range("A100000").End(xlUp).Row + 1
            Workbooks.Open (ThisWorkbook.Path & "\" & arquivo.Name)
        End If
    Next
End Sub
```

```
    ...
    End With

    Application.ScreenUpdating = True
    Application.Calculation = xlCalculationAutomatic

    MsgBox("Macro executada com sucesso!")

End Sub
```

O código da página anterior já executa exatamente o que queremos. Porém, ainda podemos otimizar um pouco mais, fazendo ele rodar ainda mais rápido.

Para isso, usamos as nossas conhecidas estruturas otimizadoras: ScreenUpdating e Calculation.

Lembrando que sempre que as desativarmos no início do código, precisamos também ativá-las ao final do código!

Módulo 11

# Eventos no VBA

Uma vez que você tenha criado uma macro, você precisa executá-la de alguma forma, seja através de um botão, seja através do atalho F5, como já fizemos diversas vezes anteriormente. Caso contrário, todos os códigos que criamos teriam sido em vão, pois aparentemente eles não são capazes de serem executados sozinhos. Será mesmo que não existe uma maneira de fazer isso?

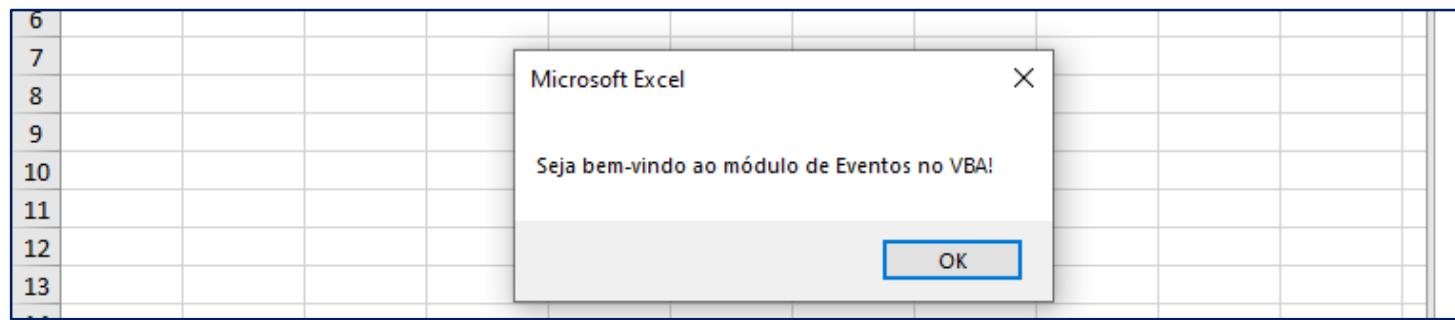
Na verdade, existe sim. Podemos executar automaticamente uma macro no momento em que um determinado evento ocorre, sem a necessidade de apertar um botão ou usar o atalho F5 no ambiente VBA. Isso ocorreria de forma automática. Mas como assim uma “macro que é executada quando um **evento** ocorre”?

De maneira objetiva, um **evento** nada mais é do que alguma coisa que acontece enquanto estamos trabalhando no Excel. Estes eventos ocorrem o tempo todo, sem você nem perceber ou ao menos se tocar de que aquilo é um evento.

Pra ficar um pouco mais claro, abaixo temos alguns exemplos de eventos no Excel:

- Abrir um arquivo.
- Fechar um arquivo.
- Salvar um arquivo.
- Ativar uma aba.
- Selecionar uma célula.
- Alterar o conteúdo de uma célula.
- E assim vai...

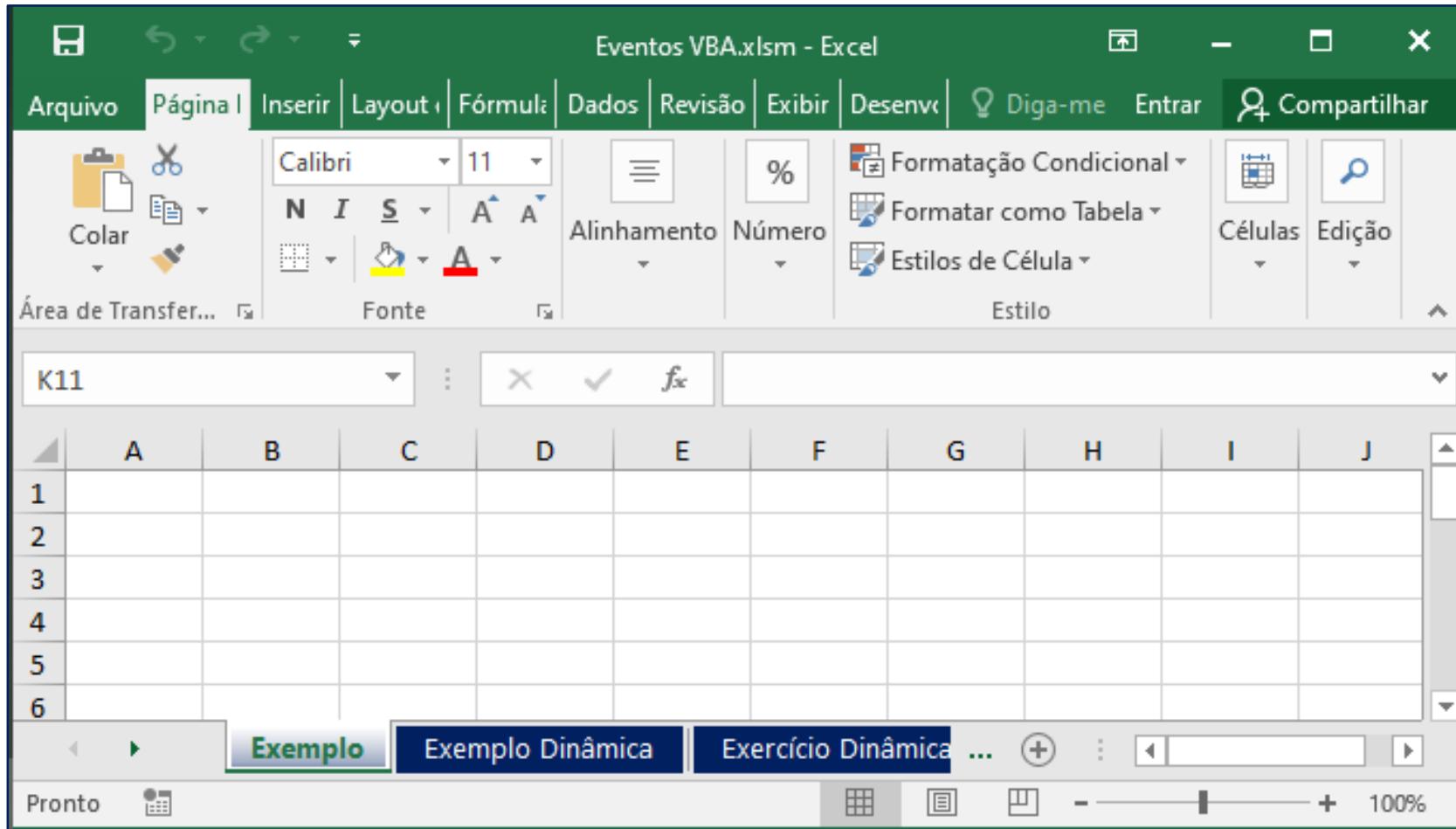
Isso significa que podemos criar macros que são executadas automaticamente quando o usuário modifica o valor de uma célula, ou então toda vez que ele muda de aba, ou então toda vez que ele abre um arquivo, como por exemplo, uma MsgBox com a seguinte mensagem:



### Por que criar eventos no VBA?

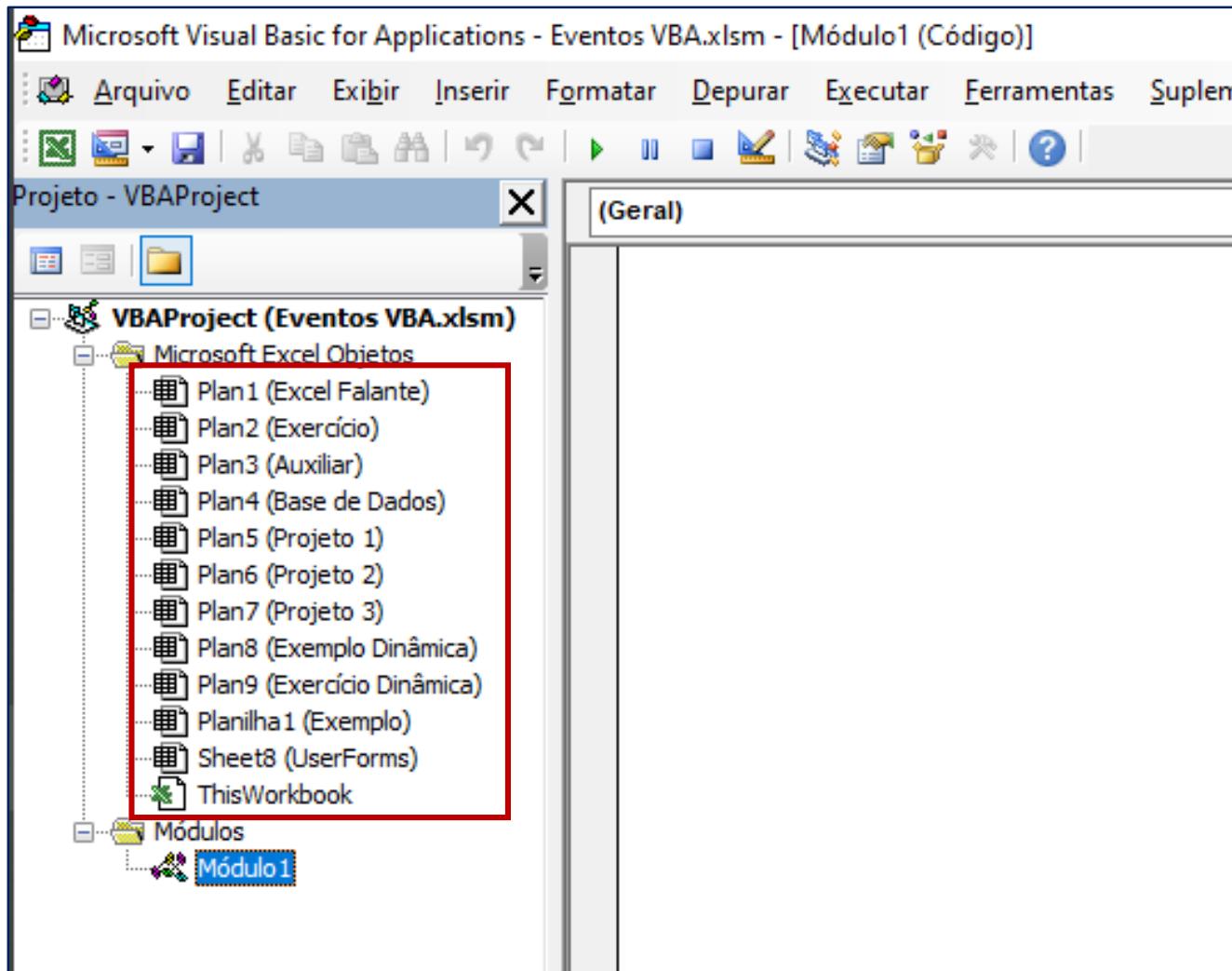
O maior motivo para criar eventos é porque você pode criar macros que são executadas automaticamente, como já citamos acima. Isso permite adicionar maior interatividade dos arquivos com o usuário, melhorar a experiência de quem estiver utilizando a sua planilha e, o mais importante, executar atividades que seriam impossíveis sem os eventos em VBA. (como você acha que criaria uma macro que executa automaticamente toda vez que um arquivo é aberto, sem a necessidade de clicar em um botão?)

A partir da próxima página, começaremos a praticar e entender melhor o funcionamento destes códigos.



Vamos começar a trabalhar com um arquivo novo. Neste, vamos criar alguns eventos para entender quais as possibilidades temos a partir de agora.

Assim, para iniciar, vamos até o ambiente VBA para começar a criar os nossos códigos.



Até agora, o que fizemos basicamente para criar os nossos códigos foi ir até a guia **Inserir** do ambiente VBA e selecionar a opção **Módulo**.

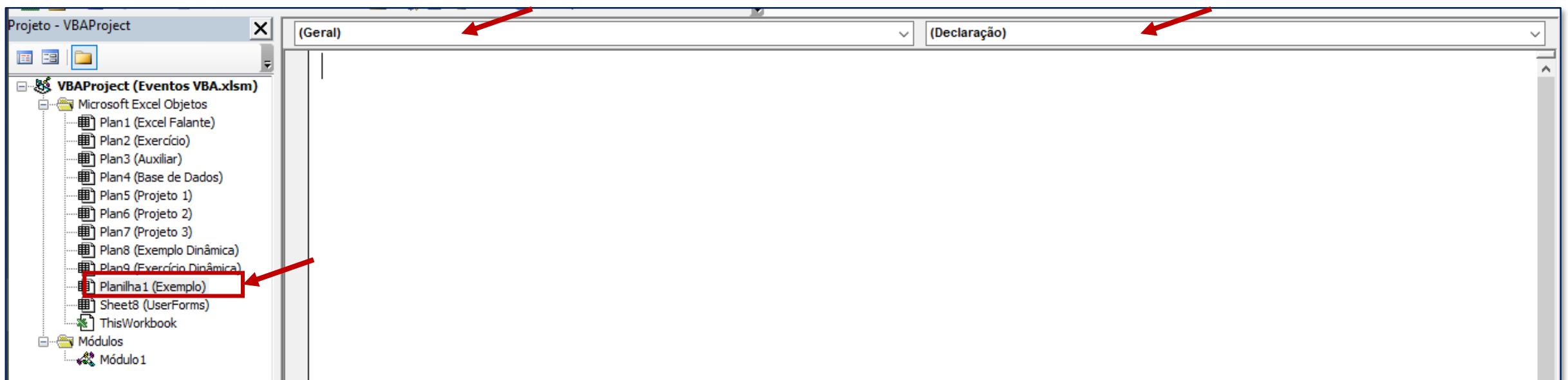
Fizemos assim até agora por 2 motivos:

1. As macros criadas em módulos funcionam para todas as abas do arquivo.
2. Uma macro escrita em um módulo pode ser atribuída a um botão.

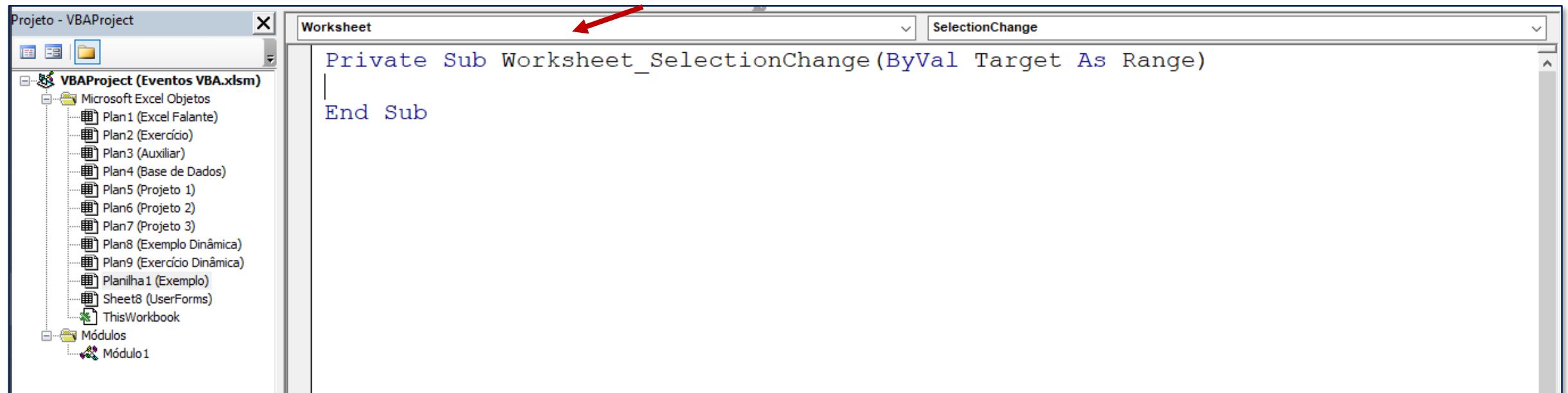
Porém, podemos escrever nossas macros diretamente nas abas. Se você der um duplo clique em uma das abas na lista da esquerda, irá abrir uma nova página em branco para que a gente possa escrever o nosso código. A diferença é que, um código criado diretamente em uma aba, afetará apenas aquela aba.

Para começar com um exemplo, vamos criar uma macro que é executada toda vez que selecionamos a aba “Exemplo”. Para isso, precisamos dar um duplo clique na aba “Exemplo” para criar o nosso código ali.

Em seguida, vamos ter que começar a trabalhar com as duas caixas de opções que ignoramos até agora, bem onde está escrito: (Geral) e (Declaração).



Começamos mudando a caixinha da esquerda de (Geral) para Worksheet. É dessa forma que o VBA entende que queremos trabalhar com eventos. Automaticamente, será criada uma Private Sub, e a caixinha da direita será modificada de (Declaração) para SelectionChange. O Private Sub significa que esta é uma Sub que é particular (privada) à aba “Exemplo”, como já comentamos na página anterior. Já o SelectionChange é um dos eventos que existem, sendo este o que é criado por padrão. Mais a frente vamos falar deste em particular.

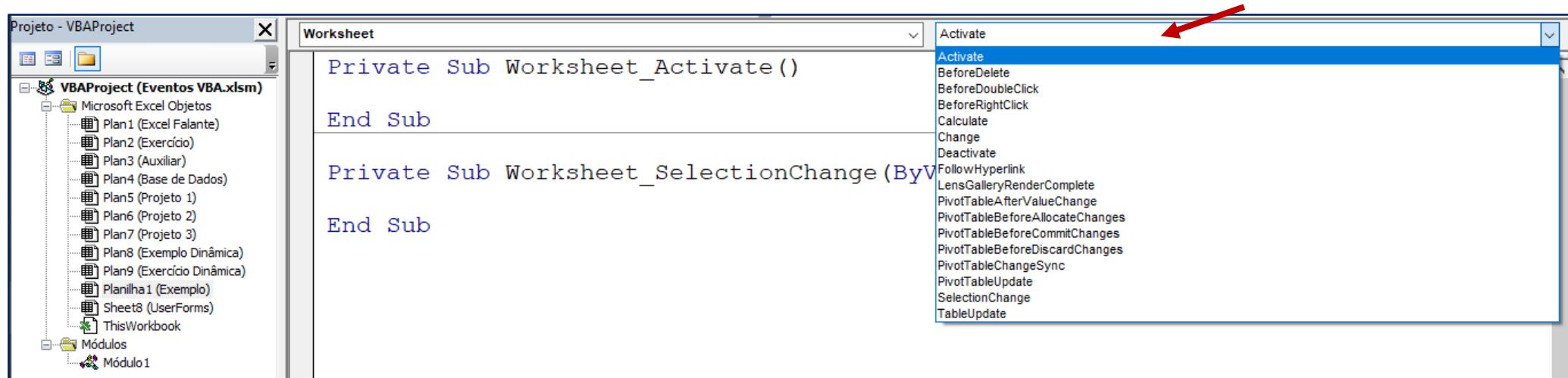


The screenshot shows the Microsoft Visual Basic Editor (VBE) interface. On the left is the Project Explorer window titled "Projeto - VBAPercent23ct". It lists the "VBAPercent23ct (Eventos VBA.xlsm)" project, which contains several worksheets like Plan1 through Plan9, a "ThisWorkbook" object, and a "Módulos" folder with "Módulo1". The main editor window has a title bar "Worksheet" and a dropdown menu "SelectionChange". The code pane displays the following VBA code:

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

A red arrow points to the word "Worksheet" in the title bar of the editor window.

Para começar por um exemplo bem simples, vamos clicar na caixinha da direita. Abrirá uma lista de opções, que nada mais são do que eventos que existem no VBA, que vão acontecer especificamente para aquela aba (worksheet). Vamos começar com o Activate. Este evento irá nos possibilitar executar uma macro toda vez que a aba “Exemplo” for ativada/selecionada. Se quiser, você pode apagar o evento SelectionChange criado automaticamente. Não o usaremos para nada.



O que você precisa entender da estrutura deste código que é criado automaticamente é:

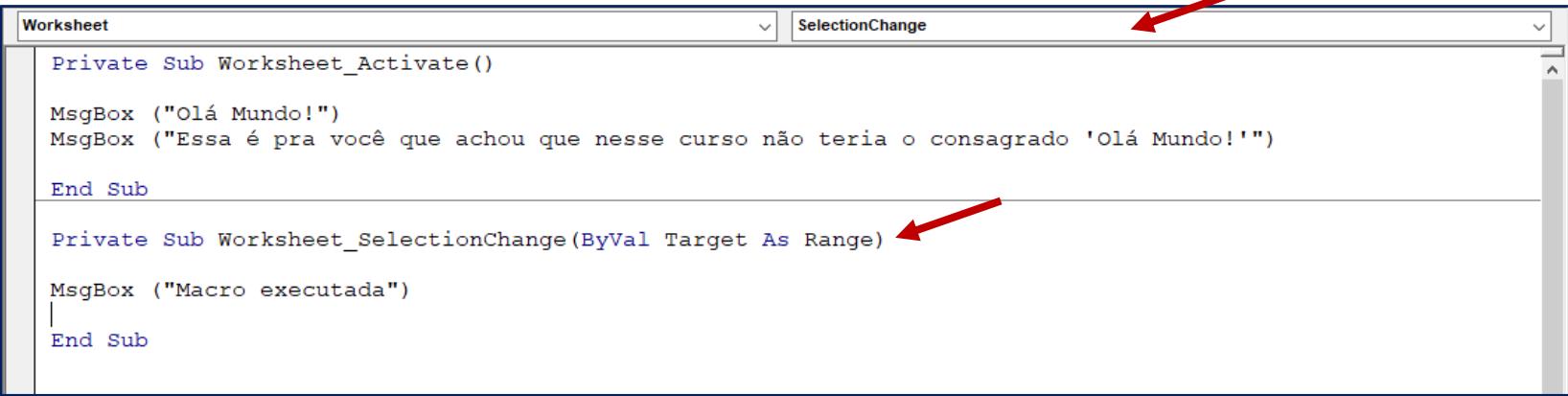
Private Sub Worksheet\_Activate()

Palavra-chave para  
chamar a Sub

Nome do evento que  
ativará a macro

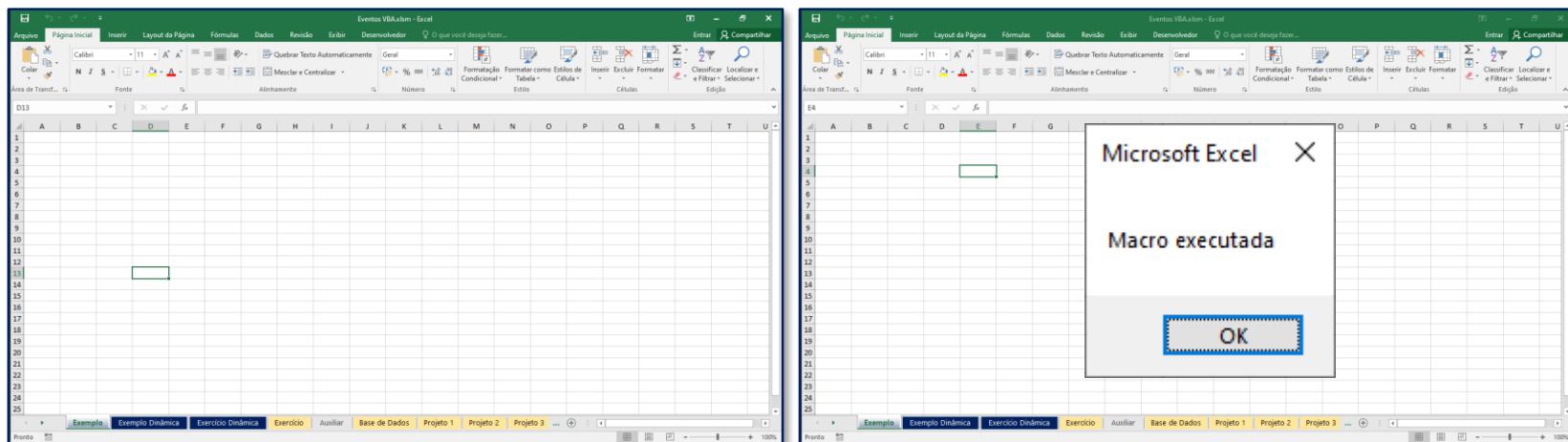
# Módulo 11 – Eventos no VBA – Como criar eventos no VBA (Parte 2)

335



```
Worksheet
Private Sub Worksheet_Activate()
    MsgBox ("Olá Mundo!")
    MsgBox ("Essa é pra você que achou que nesse curso não teria o consagrado 'Olá Mundo!'")
End Sub

Private Sub Worksheet_SelectionChange(ByVal Target As Range)
    MsgBox ("Macro executada")
End Sub
```



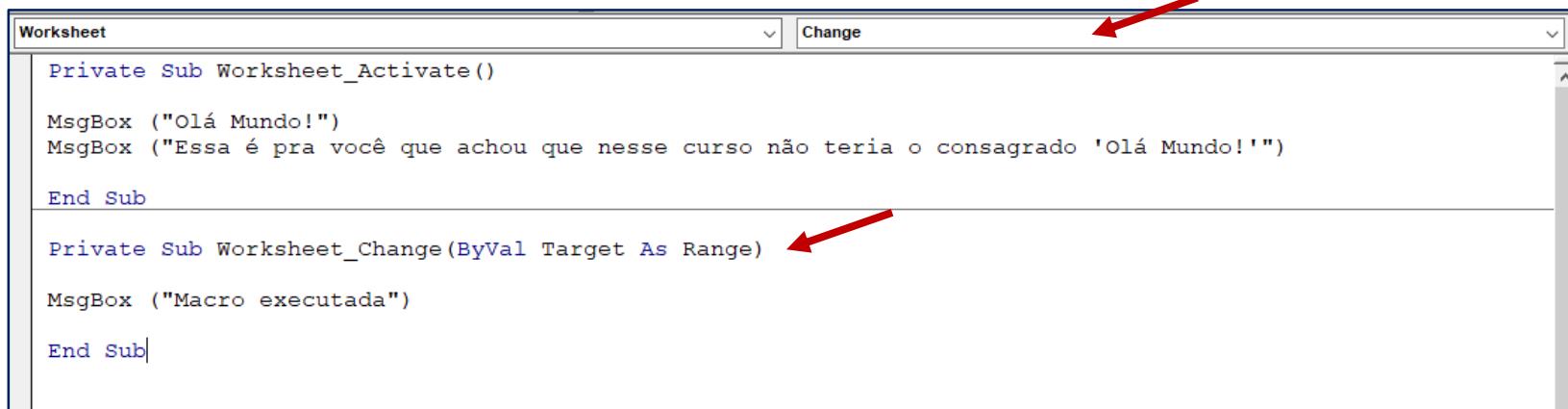
Agora vamos entender o SelectionChange. Essa macro basicamente será executada toda vez que a gente mudar uma seleção de uma célula. Traduzindo: sempre que a gente selecionar uma nova célula na planilha.

Apenas para teste, vamos associar a esse evento a MsgBox ao lado.

Para executar o código, basta mudar a célula selecionada na aba “Exemplo”.

## Módulo 11 – Eventos no VBA – Como criar eventos no VBA (Parte 2)

336

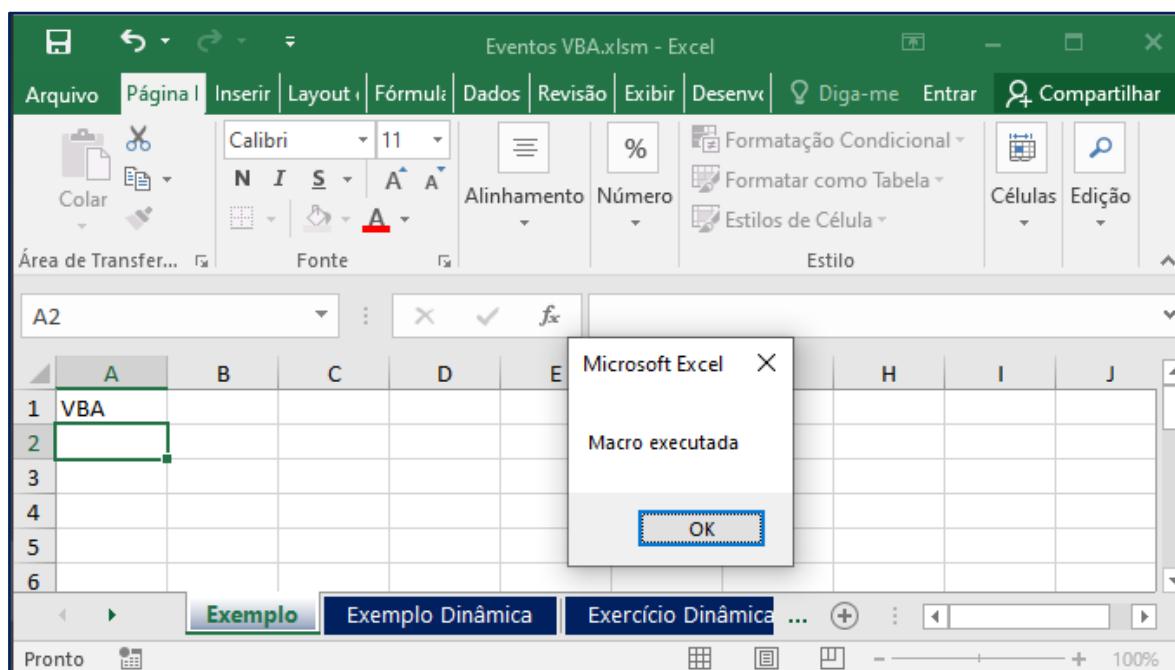


The screenshot shows the VBA editor with the code for the Worksheet\_Change event. The code is as follows:

```
Worksheet
Private Sub Worksheet_Activate()
    MsgBox ("Olá Mundo!")
    MsgBox ("Essa é pra você que achou que nesse curso não teria o consagrado 'Olá Mundo!'")
End Sub

Private Sub Worksheet_Change(ByVal Target As Range)
    MsgBox ("Macro executada")
End Sub
```

Two red arrows point to the event names: "Worksheet\_Change" and "ByVal Target As Range".



Outro evento muito utilizado é o Change. Este evento é executado toda vez que modificamos o valor de uma célula.

Repare na diferença para o anterior (SelectionChange): enquanto o Change só é executado quando o valor de uma célula é modificado, o SelectionChange será executado toda vez que uma célula simplesmente for selecionada.

The image shows two side-by-side Excel windows. The left window, titled 'Eventos VBA.xlsxm - Excel', displays a dynamic table named 'Exemplo' with columns: Descrição, Marca, Faturado, Perda, Fábrica, and Origem. The right window, titled 'Eventos VBA.xlsxm - Excel', shows a summary report for the 'Exemplo' table. The report includes labels for 'Soma de Faturado', 'Rótulos de Coluna', 'Rótulos de Linha', and categories 'Fábrica RJ' and 'Fábrica SP'. It lists totals for each brand under each factory, ending with a 'Total Geral' row.

	A	B	C	D	E	F
1	Descrição	Marca	Faturado	Perda	Fábrica	Origem
2	Produto 1	Marca A	R\$ 20.822,00	R\$ 1.500,00	Fábrica SP	Importado
3	Produto 2	Marca B	R\$ 75.789,00	R\$ 16.848,00	Fábrica RJ	Nacional
4	Produto 3	Marca A	R\$ 20.173,00	R\$ 3.557,00	Fábrica SP	Importado
5	Produto 4	Marca C	R\$ 29.707,00	R\$ 3.300,00	Fábrica RJ	Nacional
6	Produto 5	Marca C	R\$ 32.568,00	R\$ 17.271,00	Fábrica RJ	Nacional
7	Produto 6	Marca A	R\$ 32.488,00	R\$ 17.691,00	Fábrica SP	Nacional
8	Produto 7	Marca A	R\$ 22.528,00	R\$ 181,00	Fábrica SP	Importado
9	Produto 8	Marca C	R\$ 71.753,00	R\$ 18.067,00	Fábrica RJ	Nacional
10	Produto 9	Marca C	R\$ 54.213,00	R\$ 45.986,00	Fábrica SP	Nacional
11	Produto 10	Marca A	R\$ 55.171,00	R\$ 46.029,00	Fábrica RJ	Nacional
12	Produto 11	Marca C	R\$ 54.213,00	R\$ 47.060,00	Fábrica SP	Nacional
13	Produto 12	Marca B	R\$ 66.952,00	R\$ 31.412,00	Fábrica SP	Nacional
14	Produto 13	Marca A	R\$ 65.536,00	R\$ 25.875,00	Fábrica SP	Nacional
15	Produto 14	Marca C	R\$ 55.473,00	R\$ 32.267,00	Fábrica SP	Nacional
16	Produto 15	Marca C	R\$ 53.158,00	R\$ 43.025,00	Fábrica SP	Importado
17	Produto 16	Marca A	R\$ 73.278,00	R\$ 53.866,00	Fábrica RJ	Nacional
18	Produto 17	Marca C	R\$ 20.412,00	R\$ 9.120,00	Fábrica RJ	Nacional
19	Produto 18	Marca C	R\$ 88.358,00	R\$ 166,00	Fábrica SP	Nacional
20	Produto 19	Marca A	R\$ 70.381,00	R\$ 40.897,00	Fábrica RJ	Importado
21	Produto 20	Marca B	R\$ 62.129,00	R\$ 46.566,00	Fábrica RJ	Importado
22	Produto 21	Marca B	R\$ 76.050,00	R\$ 57.852,00	Fábrica RJ	Nacional
23	Produto 22	Marca A	R\$ 26.407,00	R\$ 8.586,00	Fábrica SP	Importado
24	Produto 23	Marca A	R\$ 51.520,00	R\$ 35.287,00	Fábrica RJ	Nacional
25	Produto 24	Marca A	R\$ 71.738,00	R\$ 3.140,00	Fábrica RJ	Nacional
26	Produto 25	Marca A	R\$ 37.012,00	R\$ 16.579,00	Fábrica RJ	Importado
27	Produto 26	Marca B	R\$ 46.892,00	R\$ 20.622,00	Fábrica RJ	Nacional
28	Produto 27	Marca B	R\$ 25.370,00	R\$ 1.436,00	Fábrica SP	Importado
29	Produto 28	Marca B	R\$ 74.877,00	R\$ 32.861,00	Fábrica RJ	Nacional

O próximo exemplo é muito útil e trata-se de como otimizar o uso da tabela dinâmica.

Se você conhece um pouco sobre Tabela Dinâmica, deve saber que ela não se atualiza automaticamente quando você muda um valor na tabela (tente alterar um valor da coluna de Faturado, a tabela da aba de exercício não vai alterar).

The screenshot shows a Microsoft Excel spreadsheet with a dynamic table named 'Fábrica RJ'. The table has columns A through E. Row 2 contains the label 'Soma de Faturado' and 'Rótulos de Coluna'. Row 3 contains 'Rótulos de Linha' and 'Fábrica RJ'. Rows 4, 5, and 6 show data for 'Marca A', 'Marca B', and 'Marca C' respectively, with values R\$ 855.555,00, R\$ 1.000,00, and R\$ 3.400,00. Row 7 is the 'Total Geral' row, which contains 'R\$ 5.400,00' and '11824341'. A context menu is open over the value '11824341' in cell E7. The menu options include 'Copiar', 'Formatar células...', 'Formato de Número...', 'Atualizar' (which is highlighted in grey), 'Classificar', 'Remover "Soma de Faturado"', 'Resumir Valores por', 'Mostrar Valores como', 'Mostrar Detalhes', 'Configurações do Campo de Valor...', 'Opções da Tabela Dinâmica...', and 'Mostrar Lista de Campos'. The ribbon at the bottom shows tabs for 'Exemplo Dinâmica', 'Exercício Dinâmica' (which is selected), 'Exercício', 'Auxiliar', 'Base de Dados', and 'Projeto 1'. The status bar at the bottom left says 'Pronto'.

A	B	C	D	E
2	Soma de Faturado	Rótulos de Coluna		
3	Rótulos de Linha	Fábrica RJ		Total Geral
4	Marca A	R\$ 855.555,00	1680867	
5	Marca B	R\$ 1.000,00	2017270	
6	Marca C	R\$ 3.400,00	8126204	
7	<b>Total Geral</b>	<b>R\$ 5.400,00</b>	<b>11824341</b>	
8				
9				
10				
11				
12				
13				
14				
15				

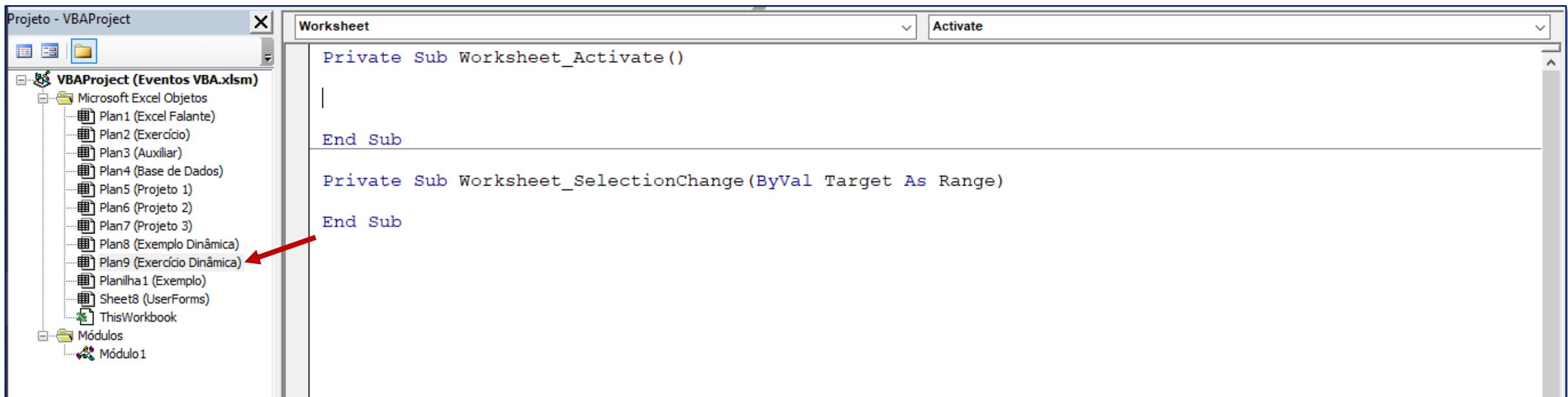
A solução para isso é até relativamente simples. Você clica com o botão direito na tabela dinâmica e seleciona a opção **Atualizar**. Todos os valores da tabela dinâmica se atualizarão de acordo com as mudanças feitas na tabela original.

Porém está totalmente sujeita a falhas. Alguém pode esquecer de atualizar ou até mesmo a pessoa que for mexer nesse arquivo pode não saber sobre tabelas dinâmicas e que existe esse problema com elas.

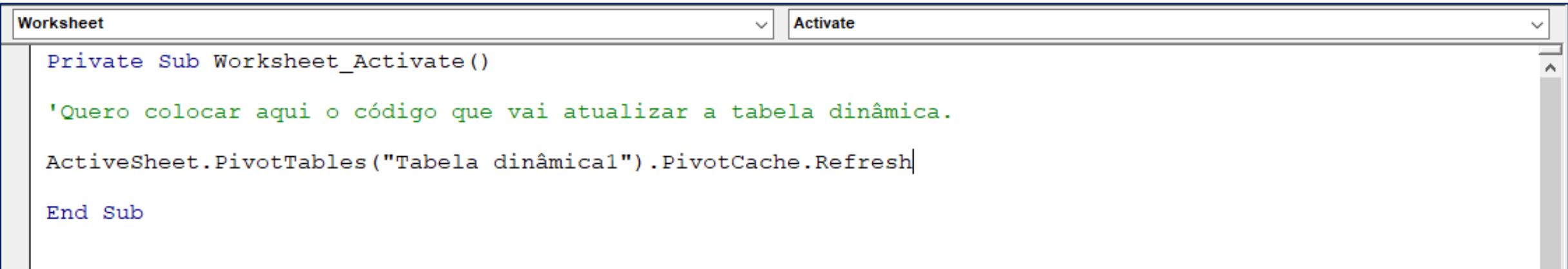
O ideal seria a gente criar uma macro que atualizasse a tabela dinâmica de forma automática toda vez que a gente selecionasse a aba onde a tabela dinâmica está.

Começamos abrindo o ambiente VBA e dando dois cliques na aba referente ao “Exercício Dinâmica”, pois queremos que a macro seja executada nesta aba. Em seguida, escolhemos o evento Activate, pois a macro deve rodar sempre que a gente selecionar esta aba. Lembrando que por padrão ele cria uma Sub para o evento SelectionChange. Você pode apagar esta Sub pois não a usaremos.

Dentro da Sub referente ao Worksheet\_Activate, queremos colocar o código para atualizar uma tabela dinâmica. Como não sabemos de cabeça, podemos gravar uma macro executando a ação de atualizar mostrada na página anterior, copiar o código e colar no nosso evento.



Agora, com esse evento, sempre que você mudar qualquer valor na sua tabela original, ao voltar para a aba da tabela dinâmica, os valores serão atualizados automaticamente. Faça o teste e verá a mágica acontecendo!



The screenshot shows the Microsoft Excel VBA editor with the 'Worksheet' tab selected in the ribbon. The code window contains the following VBA code:

```
Private Sub Worksheet_Activate()
    'Quero colocar aqui o código que vai atualizar a tabela dinâmica.
    ActiveSheet.PivotTables("Tabela dinâmica1").PivotCache.Refresh
End Sub
```

No próximo exercício queremos criar uma folha de ponto. A ideia basicamente é que a pessoa registre a entrada na coluna de entrada (com um 'x', por exemplo), e automaticamente a hora deverá ser preenchida na coluna de hora de entrada. A mesma coisa para a marcação da saída. Como queremos que uma macro seja executada automaticamente sempre que uma célula da coluna de entrada seja preenchida, queremos que a macro seja executada sempre que o valor de uma célula seja alterado. O evento que vai permitir a gente fazer isso é o **Change**.

A tabela abaixo é uma folha de pontos dos funcionários da Loja Hash&Botafogo de um dia. Como deve funcionar essa folha de pontos?

O funcionário, ao chegar na empresa, marca um x na coluna de entrada (pode marcar qualquer coisa na verdade). Assim que sua marcação estiver completa, a planilha deve automaticamente marcar o horário de sua entrada na coluna D, sempre na mesma linha da marcação do X.

O mesmo deve acontecer na sua saída. Quando houver uma marcação na coluna E, a coluna F deve ser preenchida com o horário dessa marcação automaticamente.

Além disso, a macro deve registrar o horário da última compilação na aba "Auxiliar"

Funcionário	Entrada	Hora de Entrada	Saída	Hora de Saída
João				
Lira				
Alon				
Barreto				
Fred				
Raíssa				
Marcus				
Fábio				
Bruna				

	A	B	C	D	E	F	G
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							

**Funcionário** **Entrada** **Hora de Entrada** **Saída** **Hora de Saída**

João				
Lira				
Alon				
Barreto				
Fred				
Raíssa				
Marcus				
Fábio				
Bruna				

Primeiro devemos pensar o seguinte: queremos que a macro seja executada sempre que alguém preencher uma célula da planilha, porém, apenas se esta célula estiver dentro dos intervalos de entrada (C9:C17) e de saída (E9:E17).

Não queremos que NADA aconteça se alguém preencher com um 'x' a célula A1, por exemplo, concorda?

Então vamos precisar, de alguma forma, limitar a macro para que ela só execute se e somente se alguém alterar o valor de uma célula que esteja dentro dos dois intervalos mencionados acima.

Assim, de alguma maneira, teremos que ser capazes de identificar, através do código, qual foi a célula modificada, para ai sim verificar se esta célula está dentro do intervalo desejado.

```
Private Sub Worksheet_Change(ByVal Target As Range)  
    MsgBox (Target.Address)|  
End Sub
```

A screenshot of Microsoft Excel showing a table of employee check-in and check-out times. The table has columns for Funcionário, Entrada, Hora de Entrada, Saída, and Hora de Saída. In row 10, under the 'Entrada' column, there is an 'x' in cell C10. A message box titled 'Microsoft Excel' is displayed, showing the address '\$C\$9'.

Funcionário	Entrada	Hora de Entrada	Saída	Hora de Saída
João	x			
Lira				
Alon				
Barreto				
Fred				
Raíssa				
Marcus				
Fábio				
Bruna				

A maneira como identificamos a célula selecionada é através do objeto **Target**. Você deve ter se perguntado o que seria esse Target As Range.

Bom, do que sabemos, qualquer coisa que venha antes de um As Range possivelmente é uma variável declarada como Range. E é isso mesmo. Na página 176, falamos que era possível declarar células como variáveis por meio da instrução Set. O Target é exatamente como se fosse uma célula, porém ele varia de acordo com a seleção que a gente fizer.

Um Target possui as mesmas propriedades de uma célula, tais como: Activate, Value, ClearContents, Row, Column, etc. Uma delas, a Address, retorna o endereço da célula, como é mostrado ao lado. Ao escrever um 'x' na célula C9 a MsgBox acima retornou o endereço da célula modificada, que nada mais é que o nosso Target.

```
Private Sub Worksheet_Change(ByVal Target As Range)  
  
If Target.Row >= 9 And Target.Row <= 17 Then  
  
    If Target.Column = 3 Or Target.Column = 5 Then  
  
        MsgBox ("teste")  
  
    End If  
  
End If  
  
End Sub
```

Para fazer que o evento ocorra apenas quando mudarmos o valor de uma célula que esteja em um dos dois intervalos especificados anteriormente, precisamos testar tanto a linha (Row) do Target quanto a coluna (Column). E ai, se a linha estiver entre as linhas 9 e 17 e também a coluna for igual a 3 (coluna C) ou igual a 5 (Coluna E), então criamos a lógica ao lado.

Você já pode testar este evento e verificar que ele só ocorre quando estes critérios de linha e coluna forem atendidos. No caso, o valor de resposta é uma simples MsgBox("teste").

```
Private Sub Worksheet_Change(ByVal Target As Range)  
  
If Target.Row >= 9 And Target.Row <= 17 Then  
  
    If Target.Column = 3 Or Target.Column = 5 Then  
  
        Cells(Target.Row, Target.Column + 1).Value = Now  
  
    End If  
  
End If  
  
End Sub
```

O que queremos no final das contas é escrever a data e horário na coluna D ou F. Para isso, usamos a estrutura Cells.

Nós já sabemos em qual linha devemos escrever data/hora, que no caso é a mesma linha do Target. Também sabemos em qual coluna devemos escrever, que no caso é a coluna do Target, só que + 1, pois queremos escrever na coluna exatamente ao lado.

O resultado final é mostrado na imagem ao lado.

	A	B	C	D	E	F	G
7							
8							
9	Funcionário	Entrada	Hora de Entrada	Saída	Hora de Saída		
10	João	x	10/03/2020 17:01				
11	Lira						
12	Alon						
13	Barreto						
14	Fred						
15	Raíssa						
16	Marcus						
17	Fábio						
18	Bruna						
19							

## Módulo 11 – Eventos no VBA – Evento Change: Folha de Ponto (Resolução Parte 3)

346

The screenshot shows two Excel windows side-by-side. The left window, titled 'Eventos VBA.xlsm - Excel', contains a table with columns 'Funcionário', 'Entrada', 'Hora de Entrada', 'Saída', and 'Hora de Saída'. The 'Hora de Entrada' column for João contains the value '10/03/2020 17:01'. The right window, also titled 'Eventos VBA.xlsm - Excel', has its tabs 'Exercício' and 'Auxiliar' selected. The 'Auxiliar' tab is active, showing cell A1 with the text 'Data e Hora da última compilação:'.

O próximo passo é escrever na célula A2 da aba “Auxiliar” a hora da última atualização, de acordo com a planilha de ponto da aba “Exercício”.

```
Private Sub Worksheet_Change(ByVal Target As Range)
If Target.Row >= 9 And Target.Row <= 17 Then
    If Target.Column = 3 Or Target.Column = 5 Then
        Cells(Target.Row, Target.Column + 1).Value = Now
        Sheets("Auxiliar").Activate
        Range("A2").Value = Now
    End If
End If
End Sub
```



```
Private Sub Worksheet_Change(ByVal Target As Range)
If Target.Row >= 9 And Target.Row <= 17 Then
    If Target.Column = 3 Or Target.Column = 5 Then
        Cells(Target.Row, Target.Column + 1).Value = Now
        Sheets("Auxiliar").Range("A2").Value = Now
    End If
End If
End Sub
```



Para escrever em outra aba, poderíamos tentar o primeiro código da esquerda, onde primeiro ativamos a aba “Auxiliar” para só depois escrever a hora na célula A2.

O problema é que, como agora estamos criando um evento, cujo código é escrito em uma aba específica, não importa a aba que esteja ativa, o código sempre irá escrever na aba onde o código está sendo escrito, no caso, a aba “Exercício”.

Por isso, agora teremos que fazer diferente. Em vez de primeiro ativar a aba, escrevemos o nome da aba antes do comando Range, assim, ele entenderá que deve escrever em outra aba, no caso, a aba “Auxiliar”.

```
Private Sub Worksheet_Change(ByVal Target As Range)
    If Target.Row >= 9 And Target.Row <= 17 Then
        If Target.Column = 3 Or Target.Column = 5 Then
            If Target.Value <> "" Then
                Cells(Target.Row, Target.Column + 1).Value = Now
                Sheets("Auxiliar").Range("A2").Value = Now
            End If
        End If
    End If
End Sub
```

Por fim, para evitar que o evento seja executado também quando alguém **deleta um valor** nestes intervalos (não é o comportamento que queremos), então incluímos um teste para verificar, antes de mais nada, se o valor do Target é diferente de vazio. Se for, executa o evento, caso contrário, não faz nada.

O código final do exercício é mostrado ao lado.

## Módulo 11 – Eventos no VBA – Evento Change: Desafio Projetos (Explicação)

349

The image shows two Excel windows side-by-side. The top window, titled 'Eventos VBA.xlsm - Excel', displays a table with columns A through F. Row 100 is currently selected, showing values: 99, 13/02/2018, Comercial, R\$ 3.600,00, BH, Projeto 2. The bottom window, also titled 'Eventos VBA.xlsm - Excel', shows a similar table with data from rows 31 to 42. Row 38 is selected, showing values: 100, 10/03/2020, Comercial, R\$ 5.000,00, SP, Projeto 1. A red arrow points from the bottom window towards the top window, indicating a relationship or flow between the two datasets.

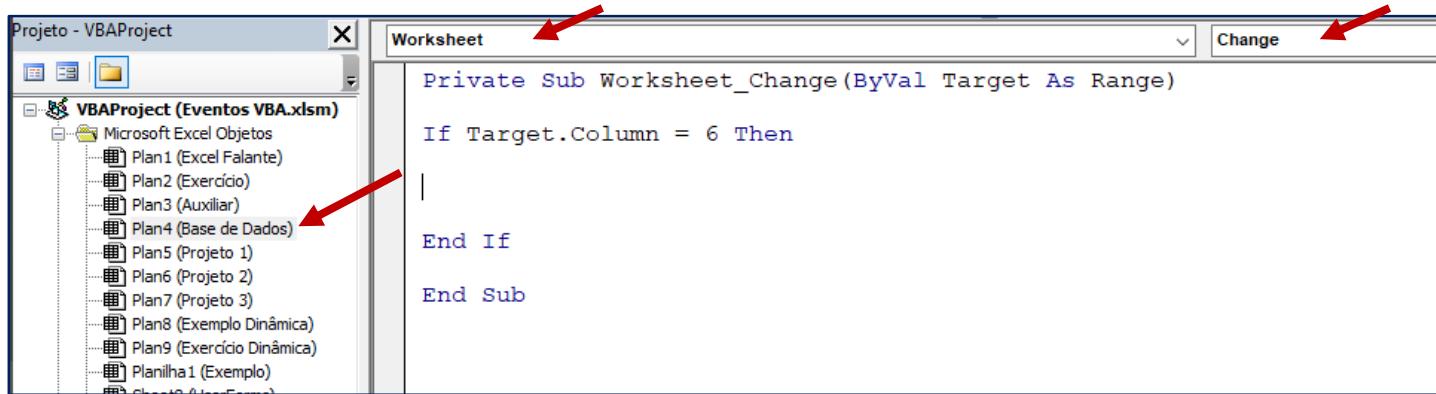
	A	B	C	D	E	F
91	90	03/01/2018	Financeiro	R\$ 4.400,00	RJ	Projeto 2
92	91	04/01/2018	Financeiro	R\$ 4.600,00	BH	Projeto 2
93	92	08/01/2018	Vendas	R\$ 6.400,00	RJ	Projeto 1
94	93	11/01/2018	Logística	R\$ 3.600,00	SP	Projeto 1
95	94	20/01/2018	RH	R\$ 4.200,00	MG	Projeto 3
96	95	22/01/2018	Comercial	R\$ 2.800,00	BH	Projeto 3
97	96	28/01/2018	Marketing	R\$ 800,00	SP	Projeto 2
98	97	07/02/2018	RH	R\$ 2.600,00	MG	Projeto 2
99	98	12/02/2018	Customer Care	R\$ 6.200,00	BH	Projeto 2
100	99	13/02/2018	Comercial	R\$ 3.600,00	BH	Projeto 2
101	100	10/03/2020	Comercial	R\$ 5.000,00	SP	Projeto 1

	A	B	C	D	E	F
31	80	18/11/2017	Financeiro	R\$ 2.900,00	RJ	Projeto 1
32	85	10/12/2017	Logística	R\$ 4.300,00	MG	Projeto 1
33	86	23/12/2017	Comercial	R\$ 2.100,00	BH	Projeto 1
34	88	29/12/2017	Vendas	R\$ 3.300,00	RJ	Projeto 1
35	89	01/01/2018	Marketing	R\$ 4.100,00	SP	Projeto 1
36	92	08/01/2018	Vendas	R\$ 6.400,00	RJ	Projeto 1
37	93	11/01/2018	Logística	R\$ 3.600,00	SP	Projeto 1
38	100	10/03/2020	Comercial	R\$ 5.000,00	SP	Projeto 1

Neste próximo exercício queremos criar uma código que cadastre automaticamente os novos projetos conforme adicionamos uma nova linha na tabela da aba “Base de Dados”. Ou seja, ao registrar um novo projeto nesta base, a macro deverá copiar a linha adicionada e colar na aba correta, de acordo com o projeto preenchido na coluna F.

Como isso deverá ser feito de forma automática, então precisamos criar um evento para que, sempre que o último valor da linha for preenchido (no caso, o projeto na coluna F), a macro deve ser executada automaticamente.



Começando a nossa solução, devemos em primeiro lugar dar um duplo clique na aba “Base de Dados” para iniciar o nosso evento que deverá acontecer sempre que uma determinada célula desta mesma aba seja modificada, através do evento **Change**.

Agora, pensando na lógica do código, devemos levar em consideração exatamente o fato de que, logo após preencher a última coluna (a coluna F, que é a coluna 6 do Excel) devemos executar a macro. Por isso o evento **Change**.

Como temos esse limitante, o primeiro passo é testar se a coluna do nosso Target (da célula modificada) é igual a 6. Se for, então podemos executar toda a nossa macro, caso contrário, ainda não temos informações suficientes para copiar a linha e portanto nada deverá acontecer.

```
Private Sub Worksheet_Change(ByVal Target As Range)
If Target.Column = 6 Then
    linha_copia = Target.Row
    Range("A" & linha_copia & ":F" & linha_copia).Copy
End If
End Sub
```

Após o nosso teste inicial, devemos armazenar a linha do Target para saber qual linha deverá ser copiada.

Para copiar todas as células das colunas A até F, que estão na linha do Target, usamos o comando Range ao lado, concatenando o valor da linha para que a macro seja automática, independente da linha onde o projeto for registrado.

```
Private Sub Worksheet_Change(ByVal Target As Range)
If Target.Column = 6 Then
    linha_copia = Target.Row
    Range("A" & linha_copia & ":F" & linha_copia).Copy
    linha_registro = Sheets(Target.Value).Range("A1").End(xlDown).Row + 1
End If
End Sub
```

A	B	C	D	E	F
90	03/01/2018	Financeiro	R\$ 4.400,00	RJ	Projeto 2
91	04/01/2018	Financeiro	R\$ 4.600,00	BH	Projeto 2
92	08/01/2018	Vendas	R\$ 6.400,00	RJ	Projeto 1
93	11/01/2018	Logística	R\$ 3.600,00	SP	Projeto 1
94	20/01/2018	RH	R\$ 4.200,00	MG	Projeto 3
95	22/01/2018	Comercial	R\$ 2.800,00	BH	Projeto 3
96	28/01/2018	Marketing	R\$ 800,00	SP	Projeto 2
97	07/02/2018	RH	R\$ 2.600,00	MG	Projeto 2
98	12/02/2018	Customer Care	R\$ 6.200,00	BH	Projeto 2
99	13/02/2018	Comercial	R\$ 3.600,00	BH	Projeto 2
100	10/03/2020	Comercial	R\$ 5.000,00	SP	Projeto 1
101					

O próximo passo é descobrir qual é a última linha preenchida da aba onde vamos colar a informação.

A aba onde vamos colar vai depender do Projeto que a pessoa digitar na coluna F. Se a pessoa digitar “Projeto 1”, então precisaremos descobrir a última linha preenchida na tabela da aba “Projeto 1”, e só depois colar as informações.

A maneira que temos para saber a aba que deverá ser colada é exatamente o valor da célula do nosso Target. Assim, podemos utilizá-lo dentro de um comando Sheets para saber a aba onde deveremos encontrar a última linha preenchida.

Para isso, criamos a variável **linha\_registro** e executamos o procedimento de sempre para encontrar a última linha, com a diferença que agora devemos especificar o nome da aba antes de mais nada. O +1 é por conta de querermos adicionar na linha seguinte a última da tabela.

```
Private Sub Worksheet_Change(ByVal Target As Range)
If Target.Column = 6 Then
    linha_copia = Target.Row
    Range("A" & linha_copia & ":F" & linha_copia).Copy
    linha_registro = Sheets(Target.Value).Range("A1").End(xlDown).Row + 1
    Sheets(Target.Value).Cells(linha_registro, 1).PasteSpecial
End If
End Sub
```

Feito isso, só falta colar a linha usando o comando PasteSpecial.

O **Sheets(Target.Value)** representa a aba do projeto e o **Cells(linha\_registro, 1).Value** representa a célula a partir de onde será colada a informação, sempre na próxima linha disponível para um novo cadastro, de acordo com a variável **linha\_registro**.

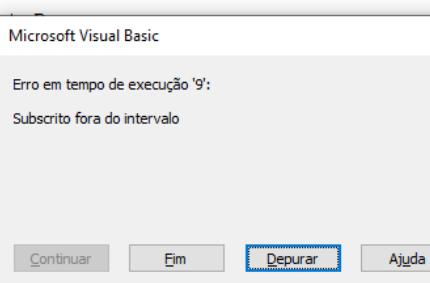
Nossa macro já estará funcionando perfeitamente.

	A	B	C	D	E	F	G
91	90	03/01/2018	Financeiro	R\$ 4.400,00	RJ	Projeto 2	
92	91	04/01/2018	Financeiro	R\$ 4.600,00	BH	Projeto 2	
93	92	08/01/2018	Vendas	R\$ 6.400,00	RJ	Projeto 1	
94	93	11/01/2018	Logística	R\$ 3.600,00	SP	Projeto 1	
95	94	20/01/2018	RH	R\$ 4.200,00	MG	Projeto 3	
96	95	22/01/2018	Comercial	R\$ 2.800,00	BH	Projeto 3	
97	96	28/01/2018	Marketing	R\$ 800,00	SP	Projeto 2	
98	97	07/02/2018	RH	R\$ 2.600,00	MG	Projeto 2	
99	98	12/02/2018	Customer Care	R\$ 6.200,00	BH	Projeto 2	
100	99	13/02/2018	Comercial	R\$ 3.600,00	BH	Projeto 2	
101	100	10/03/2020	Comercial	R\$ 5.000,00	SP	Projeto 1	
102							

	A	B	C	D	E	F	G
31	80	18/11/2017	Financeiro	R\$ 2.900,00	RJ	Projeto 1	
32	85	10/12/2017	Logística	R\$ 4.300,00	MG	Projeto 1	
33	86	23/12/2017	Comercial	R\$ 2.100,00	BH	Projeto 1	
34	88	29/12/2017	Vendas	R\$ 3.300,00	RJ	Projeto 1	
35	89	01/01/2018	Marketing	R\$ 4.100,00	SP	Projeto 1	
36	92	08/01/2018	Vendas	R\$ 6.400,00	RJ	Projeto 1	
37	93	11/01/2018	Logística	R\$ 3.600,00	SP	Projeto 1	
38	100	10/03/2020	Comercial	R\$ 5.000,00	SP	Projeto 1	
39							
40							
41							
42							

	A	B	C	D	E	F	G
91	90	03/01/2018	Financeiro	R\$ 4.400,00	RJ	Projeto 2	
92	91	04/01/2018	Financeiro	R\$ 4.600,00	BH	Projeto 2	
93	92	08/01/2018	Vendas	R\$ 6.400,00	RJ	Projeto 1	
94	93	11/01/2018	Logística	R\$ 3.600,00	SP	Projeto 1	
95	94	20/01/2018	RH	R\$ 4.200,00	MG	Projeto 3	
96	95	22/01/2018	Comercial	R\$ 2.800,00	BH	Projeto 3	
97	96	28/01/2018	Marketing	R\$ 800,00	SP	Projeto 2	
98	97	07/02/2018	RH	R\$ 2.600,00	MG	Projeto 2	
99	98	12/02/2018	Customer Care	R\$ 6.200,00	BH	Projeto 2	
100	99	13/02/2018	Comercial	R\$ 3.600,00	BH	Projeto 2	
101	100	10/03/2020	Comercial	R\$ 5.000,00	SP	Projeto 2	
102							

```
Private Sub Worksheet_Change(ByVal Target As Range)
If Target.Column = 6 Then
    linha_copia = Target.Row
    Range("A" & linha_copia & ":F" & linha_copia).Copy
    linha_registro = Sheets(Target.Value).Range("A1").End(xlDown).Row + 1
    Sheets(Target.Value).Cells(linha_registro, 1).PasteSpecial
End If
End Sub
```



```
Private Sub Worksheet_Change(ByVal Target As Range)
If Target.Column = 6 Then
    linha_copia = Target.Row
    Range("A" & linha_copia & ":F" & linha_copia).Copy
    | linha_registro = Sheets(Target.Value).Range("A1").End(xlDown).Row + 1
    Sheets(Target.Value).Cells(linha_registro, 1).PasteSpecial
End If
End Sub
```

Ashtaq

```
Private Sub Worksheet_Change(ByVal Target As Range)
    If Target.Column = 6 And Target.Value <> "" Then
        linha_copia = Target.Row
        Range("A" & linha_copia & ":F" & linha_copia).Copy
        linha_registro = Sheets(Target.Value).Range("A1").End(xlDown).Row + 1
        Sheets(Target.Value).Cells(linha_registro, 1).PasteSpecial
    End If
End Sub
```

Para evitar este problema, podemos usar uma lógica parecida com o exercício anterior de folha de ponto, que basicamente é testar se, além da coluna do Target ser igual a 6, o valor também deverá ser diferente de vazio para que o evento seja executado.

Caso contrário não faz sentido executar a macro pois sem o nome do projeto não é possível saber a aba onde a linha deve ser colada.

A	B	C	D	E	F	G
94	93	11/01/2018	Logística	R\$ 3.600,00	SP	Projeto 1
95	94	20/01/2018	RH	R\$ 4.200,00	MG	Projeto 3
96	95	22/01/2018	Comercial	R\$ 2.800,00	BH	Projeto 3
97	96	28/01/2018	Marketing	R\$ 800,00	SP	Projeto 2
98	97	07/02/2018	RH	R\$ 2.600,00	MG	Projeto 2
99	98	12/02/2018	Customer Care	R\$ 6.200,00	BH	Projeto 2
100	99	13/02/2018	Comercial	R\$ 3.600,00	BH	Projeto 2
101	100	10/03/2020	Comercial	R\$ 5.000,00	SP	Projeto 3
102						
103						
104						
105						

Outro detalhe que devemos acertar também é se, caso a pessoa decida preencher os valores a partir da coluna F, então a linha inteira será copiada e colada na aba do projeto, sem que nenhuma informação tenha sido preenchida nas outras colunas.

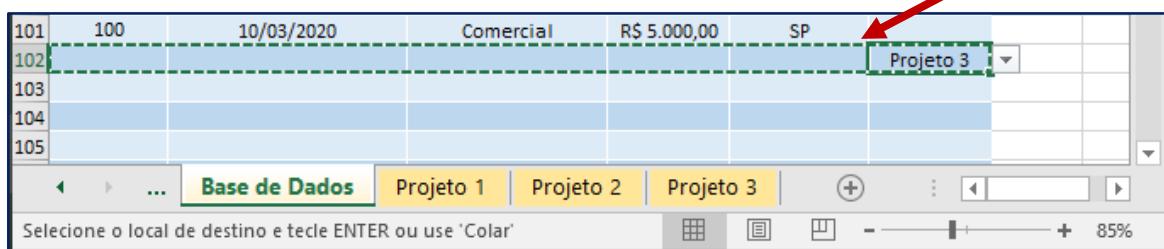
Para evitar este problema, podemos fazer o seguinte: verificar o valor de cada uma das células, da coluna A até E (coluna 1 até 5) e se algum valor estiver vazio, simplesmente não executa a macro.

Precisaremos então adicionar uma estrutura de repetição associada a um teste lógico para conseguir fazer isso.

```
Private Sub Worksheet_Change(ByVal Target As Range)
If Target.Column = 6 And Target.Value <> "" Then
    For coluna = 1 To 5
        If Cells(Target.Row, coluna).Value = "" Then
            MsgBox ("Favor preencher todas as informações do projeto")
            Exit Sub
        End If
    Next
    linha_copia = Target.Row
    Range("A" & linha_copia & ":F" & linha_copia).Copy
    linha_registro = Sheets(Target.Value).Range("A1").End(xlDown).Row + 1
    Sheets(Target.Value).Cells(linha_registro, 1).PasteSpecial
End If
End Sub
```

Adicionamos então ao código a estrutura For para percorrer cada uma das células da linha do Target, desde a coluna 1 até a coluna 5. E ai, se alguma célula de alguma coluna possuir um valor vazio, a macro simplesmente finaliza (Exit Sub).

Para tornar o código ainda mais otimizado, podemos exibir uma mensagem de texto para a pessoa saber que ainda faltam informações a serem preenchidas naquela linha.



101	100	10/03/2020	Comercial	R\$ 5.000,00	SP	Projeto 3
102						
103						
104						
105						

Base de Dados    Projeto 1    Projeto 2    Projeto 3    +    85%

Selezione o local de destino e tecle ENTER ou use 'Colar'

```

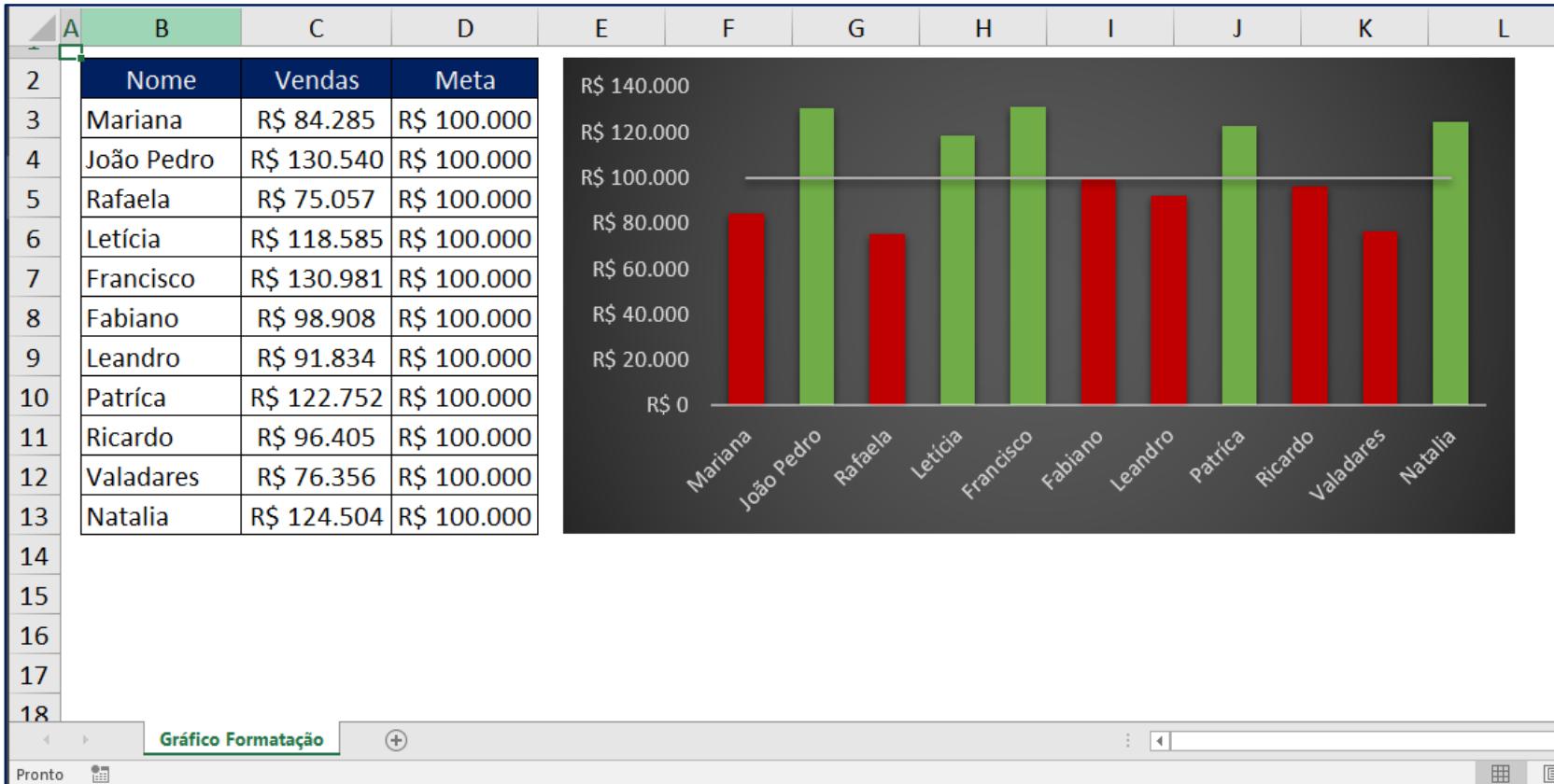
Private Sub Worksheet_Change(ByVal Target As Range)
    If Target.Column = 6 And Target.Value <> "" Then
        For coluna = 1 To 5
            If Cells(Target.Row, coluna).Value = "" Then
                MsgBox ("Favor preencher todas as informações do projeto")
                Exit Sub
            End If
        Next
        linha_copia = Target.Row
        Range("A" & linha_copia & ":F" & linha_copia).Copy
        linha_registro = Sheets(Target.Value).Range("A1").End(xlDown).Row + 1
        Sheets(Target.Value).Cells(linha_registro, 1).PasteSpecial
        Application.CutCopyMode = False
        MsgBox ("Custo registrado com sucesso!")
    End If
End Sub

```

Por fim, apenas por questões visuais, podemos também retirar esse tracejado que aparece nas células após elas serem copiadas.

Para desativá-lo, utilizamos a estrutura `CutCopyMode = False`, como já vimos anteriormente.

Para fechar, podemos ainda exibir uma mensagem final para avisar ao usuário que a macro foi executada com sucesso e os dados foram colados na respectiva aba.



No próximo exercício vamos criar uma formatação condicional para o gráfico.

A ideia é que as colunas de vendas dos vendedores no gráfico ao lado mudem de cor de acordo com a meta estabelecida de R\$ 100.000.

A princípio, as cores foram modificadas manualmente, mas a ideia é que esta formatação seja automática e aconteça sempre após modificarmos um valor de vendas de algum vendedor.

```
Private Sub Worksheet_Change(ByVal Target As Range)
If Target.Column = 3 Then
    |
End If
End Sub
```

Seguindo a mesma lógica dos exercícios anteriores, vamos utilizar o evento **Change**, pois queremos que a macro seja executada automaticamente sempre que alguém alterar o valor de vendas. Como se trata de um evento, devemos criar o código diretamente dentro da aba “Gráficos”. Dê então um duplo clique nesta aba, no canto direito do ambiente VBA e em seguida inicie o código ao lado.

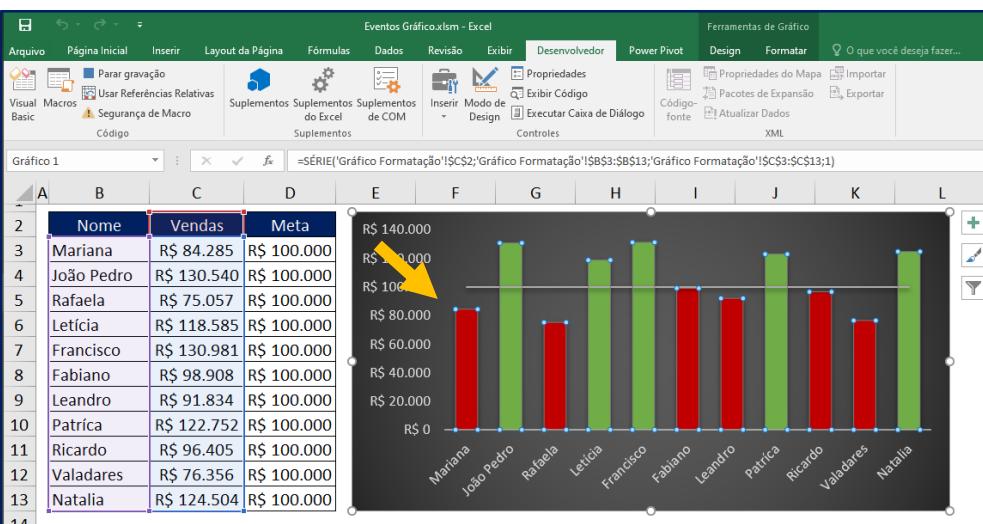
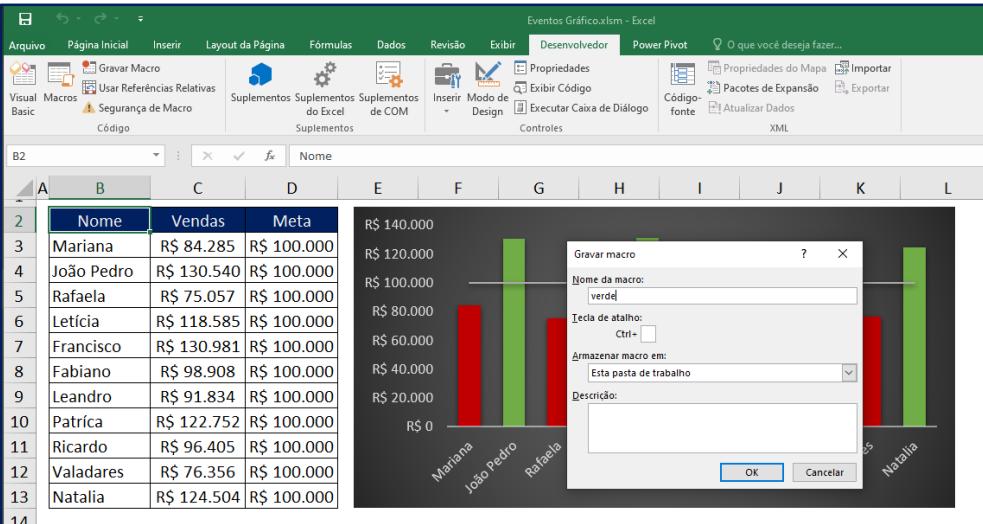
Devemos pensar que esta macro só deve rodar caso a célula alterada esteja na coluna C (coluna de número 3), pois é nessa coluna onde estão as vendas dos vendedores.

Assim, devemos iniciar testando se a coluna do nosso Target é igual a 3, e se e somente se isso for verdade, a macro deverá ser executada.

Agora, teremos que descobrir como fazer para mudar a cor de uma coluna do gráfico usando o VBA.

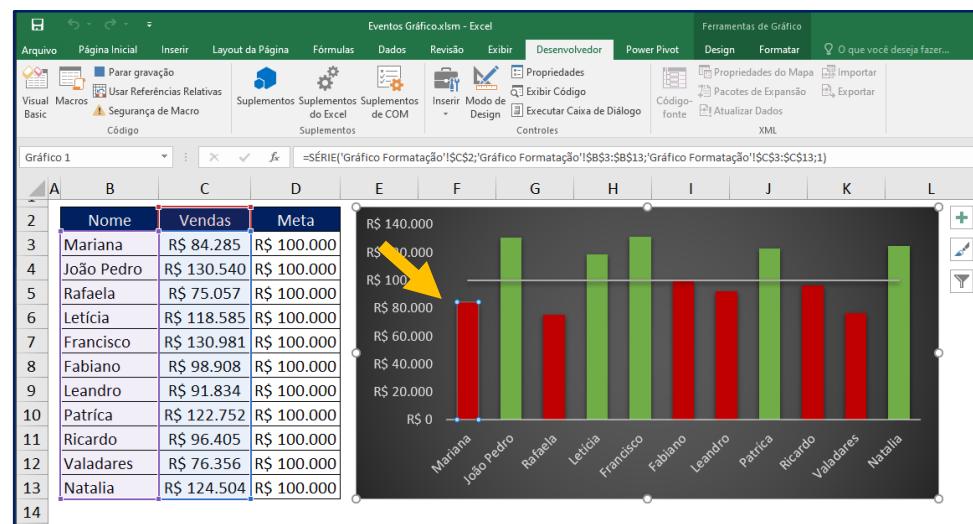
# Módulo 11 – Eventos no VBA – Evento Change: Formatação de Gráfico (Resolução Parte 1)

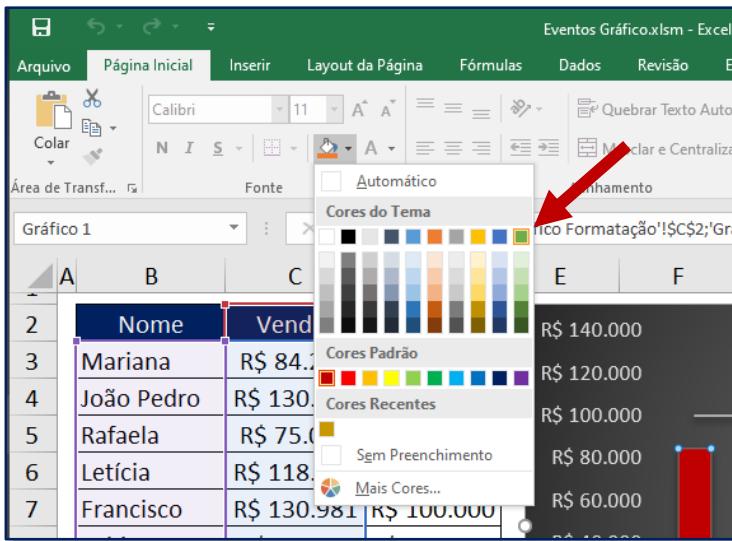
361



Caso você tenha pensado em gravar uma macro, tá voando! É exatamente o que iremos fazer. Grave então uma macro, chamada de verde, por exemplo, e siga o passo a passo:

1. Clique em uma coluna do gráfico. Isso fará com que todas as colunas sejam selecionadas.
2. Clique em uma coluna específica para selecionar apenas ela. Por exemplo, a coluna com as vendas da Mariana.

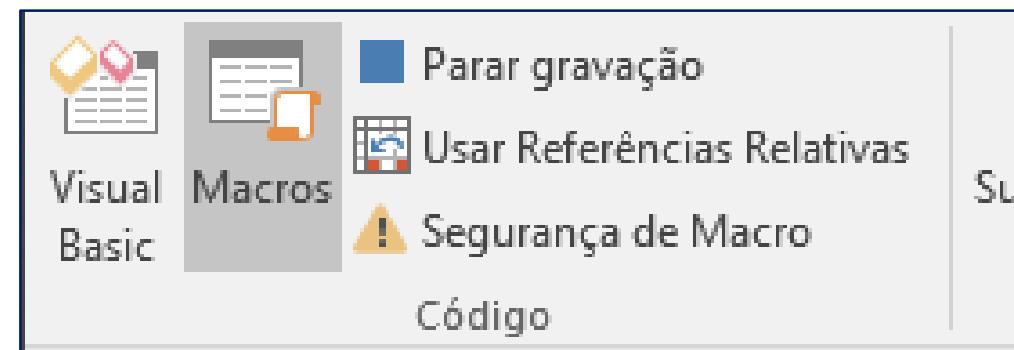
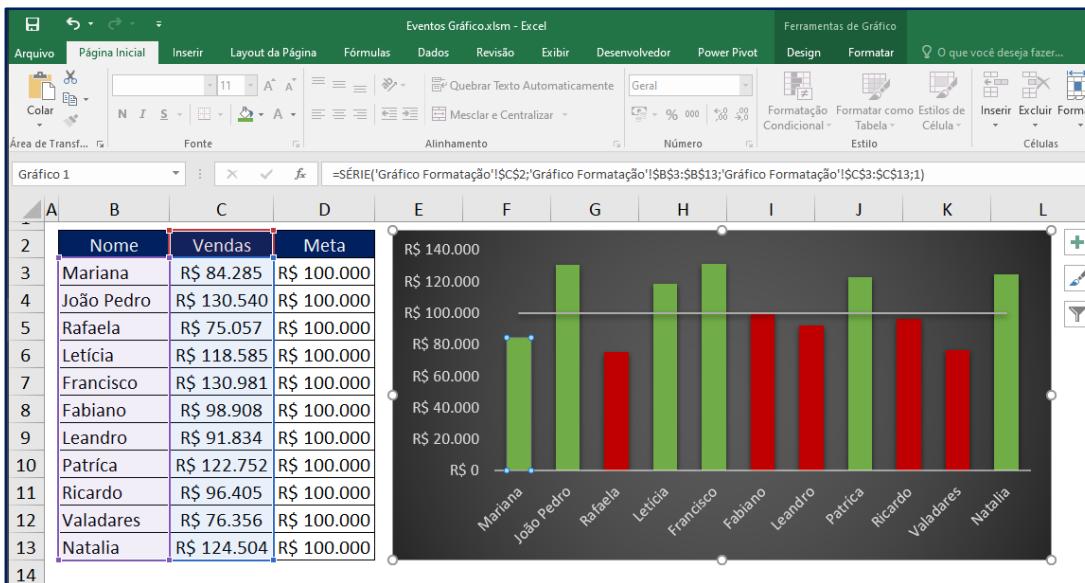




Caso você tenha pensado em gravar uma macro, tá voando! É exatamente o que iremos fazer. Grave então uma macro, chamada de verde, por exemplo, e siga o passo a passo:

3. Vá até a Página Inicial e escolha a cor Verde.
4. Por fim, clique em Parar Gravação.

Faça o mesmo passo a passo para descobrir o código da cor vermelha.



The screenshot shows the Microsoft Excel VBA Editor. On the left, the Project Explorer window displays a project named 'VBAProject (Eventos Gráfico.xlsxm)' containing a 'Planilha1 (Gráfico Formatação)' sheet and a 'Módulos' folder with a single module named 'Módulo1'. The main code editor window shows the following VBA code:

```
Sub verde()
    ' verde Macro
    ' 
    ' 
    ActiveSheet.ChartObjects("Gráfico 1").Activate
    ActiveChart.FullSeriesCollection(1).Select
    ActiveChart.FullSeriesCollection(1).Points(1).Select
    With Selection.Format.Fill
        .Visible = msoTrue
        .ForeColor.ObjectThemeColor = msoThemeColorAccent6
        .ForeColor.TintAndShade = 0
        .ForeColor.Brightness = 0
        .Transparency = 0
        .Solid
    End With
End Sub

Sub vermelho()
    ' vermelho Macro
    ' 
    ' 
    ActiveSheet.ChartObjects("Gráfico 1").Activate
    ActiveChart.FullSeriesCollection(1).Select
    ActiveChart.FullSeriesCollection(1).Points(1).Select
    With Selection.Format.Fill
        .Visible = msoTrue
        .ForeColor.RGB = RGB(192, 0, 0)
        .Transparency = 0
        .Solid
    End With
End Sub
```

The 'Propriedades' (Properties) window on the left is visible, showing 'Módulo1 Módulo' selected under 'Alfabético' (Alphabetical).

Os códigos serão salvos em módulos, como já sabemos. Basta clicar em **Módulo1** para visualizar.

Estes dois códigos vamos copiar e colar no nosso código do evento.

```
Private Sub Worksheet_Change(ByVal Target As Range)
    If Target.Column = 3 Then
        If Target.Value >= 100000 Then
            ActiveSheet.ChartObjects("Gráfico 1").Activate
            ActiveChart.FullSeriesCollection(1).Select
            ActiveChart.FullSeriesCollection(1).Points(1).Select
            With Selection.Format.Fill
                .Visible = msoTrue
                .ForeColor.ObjectThemeColor = msoThemeColorAccent6
                .ForeColor.TintAndShade = 0
                .ForeColor.Brightness = 0
                .Transparency = 0
                .Solid
            End With
        Else
            ActiveSheet.ChartObjects("Gráfico 1").Activate
            ActiveChart.FullSeriesCollection(1).Select
            ActiveChart.FullSeriesCollection(1).Points(1).Select
            With Selection.Format.Fill
                .Visible = msoTrue
                .ForeColor.RGB = RGB(192, 0, 0)
                .Transparency = 0
                .Solid
            End With
        End If
    End If
End Sub
```

O código completo está mostrado ao lado. Logo após o nosso If inicial, devemos criar um novo If para testar se o valor do Target (a célula que foi atualizada com as vendas, na coluna C) é maior ou igual a 100000. Se for, queremos utilizar o código de pintar a coluna de verde, caso contrário (Else) queremos pintar a coluna de vermelho.

Vale a pena entender minimamente o código gravado pela macro. No trecho:

```
ActiveChart.FullSeriesCollection(1).Points(1).Select
```

O comando Points(1) significa que a coluna de número 1 (referente à mariana) foi pintada de verde. Porém, queremos que este código seja automático para qualquer vendedor.

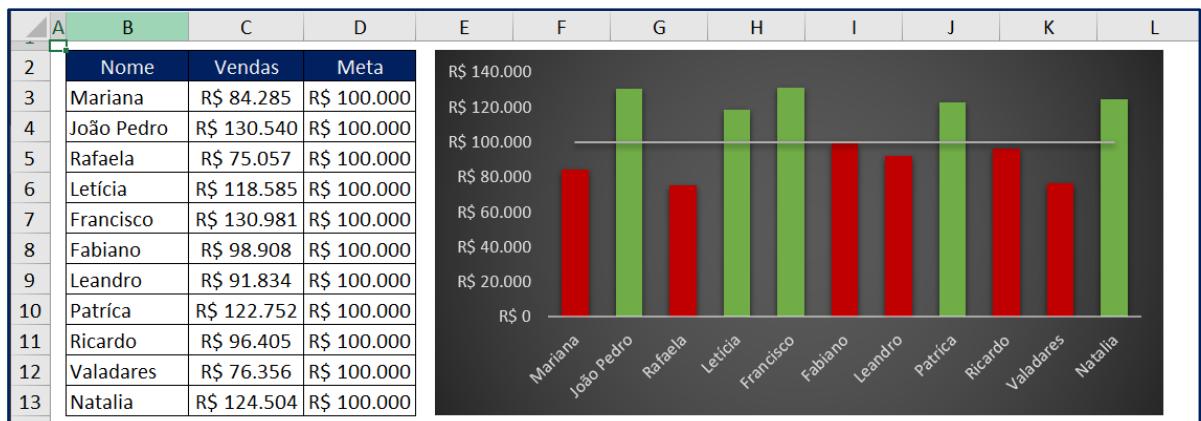
```

Private Sub Worksheet_Change(ByVal Target As Range)
    If Target.Column = 3 Then
        If Target.Value >= 100000 Then
            ActiveSheet.ChartObjects("Gráfico 1").Activate
            ActiveChart.FullSeriesCollection(1).Select
            ActiveChart.FullSeriesCollection(1).Points(Target.Row - 2).Select
            With Selection.Format.Fill
                .Visible = msoTrue
                .ForeColor.ObjectThemeColor = msoThemeColorAccent6
                .ForeColor.TintAndShade = 0
                .ForeColor.Brightness = 0
                .Transparency = 0
                .Solid
            End With
        Else
            ActiveSheet.ChartObjects("Gráfico 1").Activate
            ActiveChart.FullSeriesCollection(1).Select
            ActiveChart.FullSeriesCollection(1).Points(Target.Row - 2).Select
            With Selection.Format.Fill
                .Visible = msoTrue
                .ForeColor.RGB = RGB(192, 0, 0)
                .Transparency = 0
                .Solid
            End With
        End If
    End If
End Sub

```

Como a coluna da Mariana é a primeira, então ela recebe a posição 1 dentro do Points. Já o João Pedro recebe a posição 2, e por ai vai. Para tornar essa macro automática para a posição de qualquer um dos vendedores, no lugar do 1, trocamos para Target.Row - 2. Por quê?

No gráfico, a coluna da Mariana é a número 1, e a sua linha na tabela é a número 3. Se fizermos Target.Row - 2 para ela teremos  $= 3 - 2 = 1$ , que é a posição da sua coluna dentro do gráfico. Essa lógica vale para a coluna de qualquer vendedor.



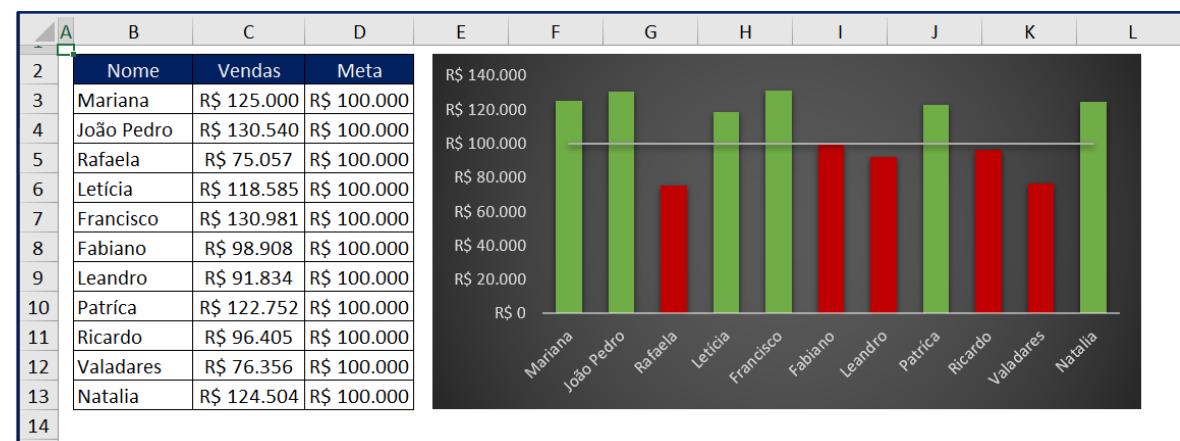
```

Private Sub Worksheet_Change(ByVal Target As Range)
    If Target.Column = 3 Then
        If Target.Value >= 100000 Then
            ActiveSheet.ChartObjects("Gráfico 1").Activate
            ActiveChart.FullSeriesCollection(1).Select
            ActiveChart.FullSeriesCollection(1).Points(Target.Row - 2).Select
            With Selection.Format.Fill
                .Visible = msoTrue
                .ForeColor.ObjectThemeColor = msoThemeColorAccent6
                .ForeColor.TintAndShade = 0
                .ForeColor.Brightness = 0
                .Transparency = 0
                .Solid
            End With
        Else
            ActiveSheet.ChartObjects("Gráfico 1").Activate
            ActiveChart.FullSeriesCollection(1).Select
            ActiveChart.FullSeriesCollection(1).Points(Target.Row - 2).Select
            With Selection.Format.Fill
                .Visible = msoTrue
                .ForeColor.RGB = RGB(192, 0, 0)
                .Transparency = 0
                .Solid
            End With
        End If
        Range("A1").Select
    End If
End Sub

```

Por fim, para a macro encerrar sem que o gráfico fique selecionado, finalizamos o código com a instrução Range("A1").Select.

A macro já está pronta para ser usada. Você pode mudar qualquer valor de vendas e a formatação da coluna será atualizada automaticamente.





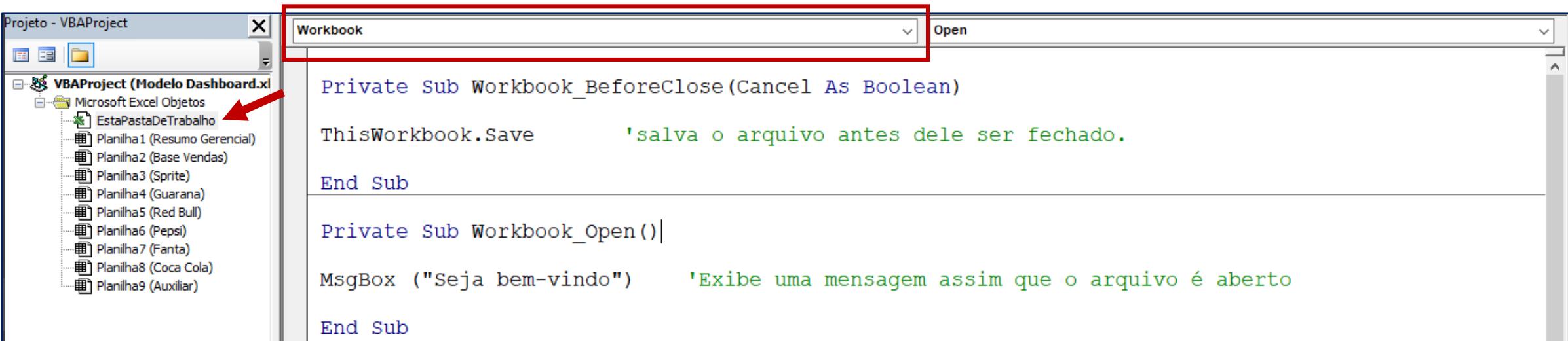
Até agora vimos como criar eventos que eram particulares de determinadas abas. Além destes, existem também outros tipos de eventos, que são os que ocorrem com o arquivo como um todo.

Exemplos disso são, basicamente: abrir um arquivo, fechar um arquivo, salvar um arquivo, e por ai vai.

No exemplo ao lado, toda vez que abrimos o arquivo, uma MsgBox é exibida automaticamente com uma mensagem de "Seja bem-vindo". Veremos como criar este tipo de evento, que se parece em muito com o que já vimos até agora.

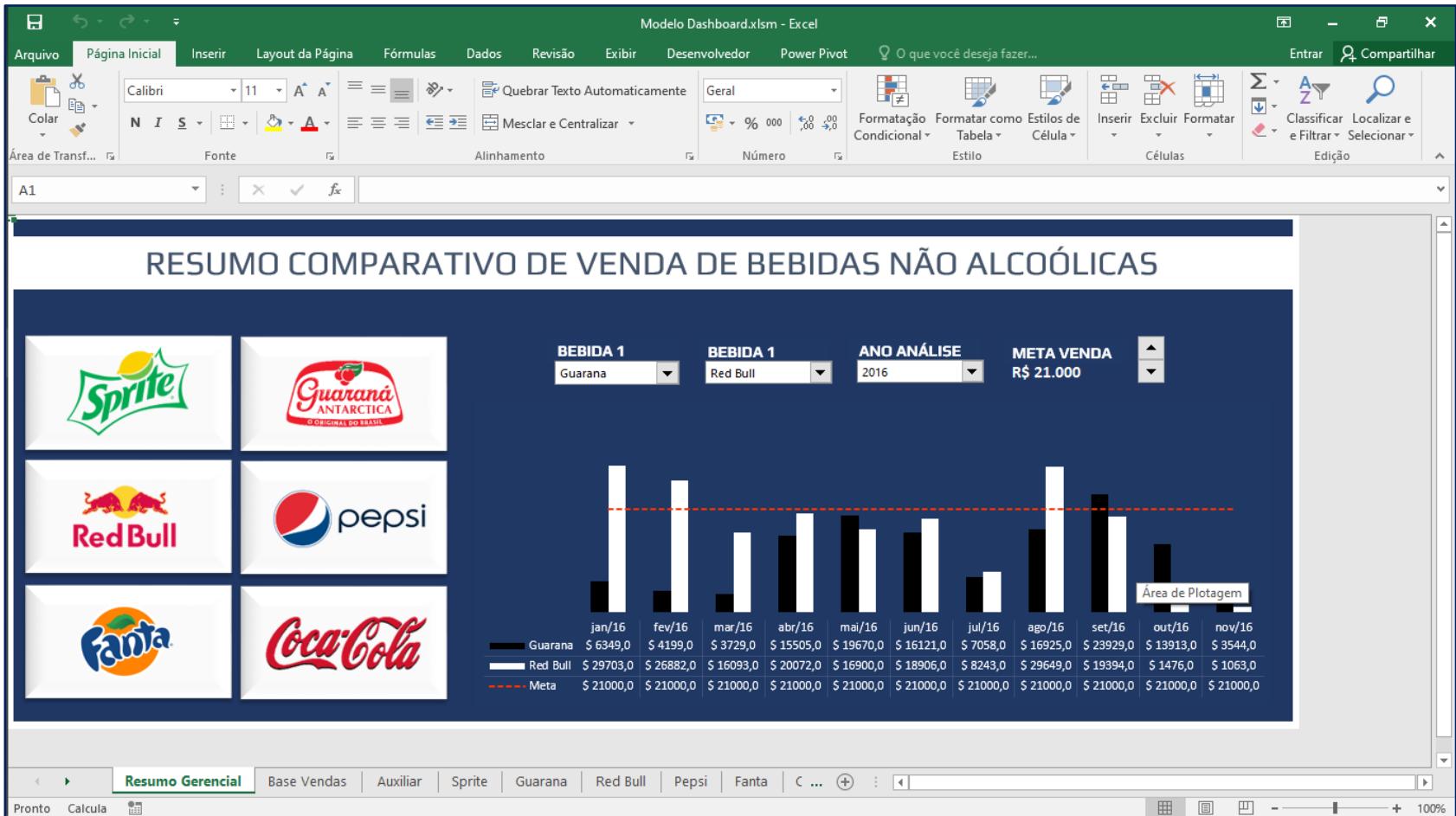
Os eventos de arquivos são até mais simples que os eventos que vimos até agora. Em primeiro lugar, para criar um evento de arquivo, devemos dar um duplo clique na opção de **EstaPastaDeTrabalho**, que não tínhamos usado até agora. Feito isso, escolhemos também a opção Workbook na lista de opções na parte superior, assim como fizemos com o Worksheet. Automaticamente, será criado um evento chamado **Workbook\_Open**, onde podemos criar uma macro para ser executada sempre que o arquivo for aberto. Além deste evento, podemos escolher diversos outros, como o **Before\_Close**, que será executado sempre antes de um arquivo ser fechado.

Os dois códigos abaixo são exemplos de macros ativadas por meio dos dois eventos descritos acima e que você pode testar em qualquer arquivo.



The screenshot shows the Microsoft Excel VBA Editor interface. On the left, the Project Explorer window displays a project named "VBAProject (Modelo Dashboard.xls)" containing a folder "Microsoft Excel Objetos" which includes "EstaPastaDeTrabalho". A red arrow points to this folder. The main code editor window has a title bar "Workbook" with a dropdown menu and a button "Open". The code itself is as follows:

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)  
    ThisWorkbook.Save      'salva o arquivo antes dele ser fechado.  
End Sub  
  
Private Sub Workbook_Open()  
    MsgBox ("Seja bem-vindo")      'Exibe uma mensagem assim que o arquivo é aberto  
End Sub
```



Voltando ao nosso arquivo inicial, queremos deixá-lo um pouco mais apresentável e com menos cara de Excel. A ideia deste arquivo, que chamamos de Dashboard, é ser bem visual e com o máximo de foco no que realmente importa para quem estiver analisando esta planilha, que no caso são os gráficos, tabelas e botões.

O objetivo então é criar um evento que será executado automaticamente sempre que a gente abrir o arquivo e que seja capaz de colocar o arquivo em tela cheia, tirar as guias e menus que aparecem no topo, os nomes das abas que aparecem na parte inferior e a barra de fórmulas.

```
Workbook BeforeClose
Private Sub Workbook_BeforeClose(Cancel As Boolean)
    Application.DisplayFormulaBar = True
    Application.DisplayFullScreen = False
    Application.ActiveWindow.DisplayWorkbookTabs = True
    ThisWorkbook.Save
End Sub

Private Sub Workbook_Open()
    Sheets("Resumo Gerencial").Activate
    Application.DisplayFormulaBar = False
    Application.DisplayFullScreen = True
    Application.ActiveWindow.DisplayWorkbookTabs = False
End Sub
```

Para fechar, tão importante quanto fazer as configurações no momento em que o arquivo abre é desfazê-las antes do arquivo ser fechado, para evitar que as configurações que fizemos para este arquivo em particular afetem os demais arquivos que abrirmos. Para isso, usamos o evento BeforeClose para executar esta macro de reconfiguração antes do arquivo ser fechado.

Ao final deste evento podemos também solicitar que o arquivo seja salvo automaticamente, caso o usuário esqueça de fechar.



O resultado final está mostrado ao lado. Ao abrir o arquivo, ele será configurado para ter uma visualização bem mais limpa e agradável ao usuário.

E lembrando que, ao fechar o arquivo, todas as configurações originais serão refeitas graças ao nosso evento BeforeClose e futuros arquivos que sejam abertos não serão afetados.

Módulo 12

# UserForms

Agora vamos entrar em um tema totalmente novo e muito importante no VBA, que são os **UserForms**. Userforms são formulários que permitem uma interação mais prática e visual entre a planilha e o usuário.

Ao lado temos um exemplo de cadastro de um novo funcionário da empresa ou exclusão de um novo funcionário. Cada botão abre uma caixinha na tela que pode ser preenchida facilmente por meio de botões e caixas de texto.

A ideia do Userform é tornar a interação muito mais intuitiva ao usuário, sem que ele precise necessariamente entender alguma coisa sobre Excel para utilizar a sua planilha. Esta mesma lógica pode ser utilizada em diferentes situações, como uma planilha de cadastro de vendas, ou de cadastro de clientes e serviços prestados, e por ai vai.

	A	B	C	D	E	F	G	H
1	Nome	Sexo	Área	CPF	Salário			
2	Adriane de Carvalho Gomes	Feminino	RH	082.442.274-27	R\$ 4.650,00			
3	Elvis de Freitas Derossi	Masculino	RH	104.030.731-00	R\$ 12.320,00			
4	Gabrielle Andrade da Silva	Feminino	Administrativo	081.419.765-67	R\$ 12.320,00			
5	Gustavo de Vasconcelos	Masculino	Administrativo	137.532.186-15	R\$ 16.300,00			
6	Isabella Ronfini	Feminino	RH	133.655.149-67	R\$ 12.320,00			
7	Izabelle Anderson	Feminino	Marketing	154.896.314-20	R\$ 15.000,00			
8	João Lira	Masculino	RH	152.896.317-24	R\$ 5.000,00			
9	João Martins	Masculino	Administrativo	187.542.124-52	R\$ 12.000,00			
10	Juan Fernandes	Masculino	RH	107.000.473-49	R\$ 16.300,00			
11	Júlio Fraga	Masculino	Compras	133.351.567-51	R\$ 7.200,00			
12	Leticia Mesquita	Feminino	Administrativo	136.725.629-63	R\$ 9.450,00			

Cadastro de Funcionário

Nome:

Sexo:  Feminino  Masculino

Área:

CPF:

Salário:



OK X

Excluir Funcionário

Nome:

OU

CPF:

**EXCLUIR** CANCELAR

## Módulo 12 – UserForms – Criando um Userform no VBA

373

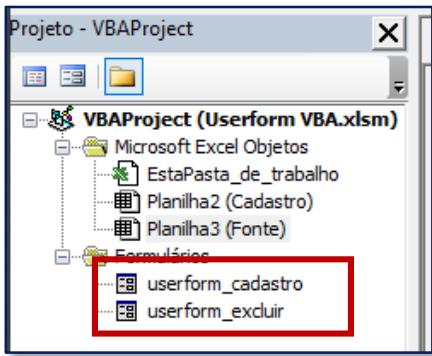
A	B	C	D	E	F	G	H	I
1	Nome	Sexo	Área	CPF	Salário			
2	Adriane de Carvalho Gomes	Feminino	RH	082.442.274-27	R\$ 4.650,00			
3	Andressa Rotsztejn	Feminino	Marketing	066.779.932-62	R\$ 8.100,00			
4	Beatriz Leite Júnior	Feminino	Administrativo	134.819.231-78	R\$ 4.650,00			
5	Bruna dos Santos Gomes	Feminino	Administrativo	073.033.309-74	R\$ 4.650,00			
6	Carolina Codeceira	Feminino	Marketing	085.347.936-87	R\$ 8.100,00			
7	Daniela Muller Braga	Feminino	Logística	127.900.156-18	R\$ 2.000,00			
8	Elvis de Freitas Derossi	Masculino	RH	104.030.731-00	R\$ 12.320,00			
9	Gabrielle Andrade da Silva	Feminino	Administrativo	081.419.765-67	R\$ 12.320,00			
10	Gustavo de Vasconcelos	Masculino	Administrativo	137.532.186-15	R\$ 16.300,00			
11	Isabella Ronfini	Feminino	RH	133.655.149-67	R\$ 12.320,00			
12	Juan Fernandes	Masculino	RH	107.000.473-49	R\$ 16.300,00			
13	Júlio Fraga	Masculino	Compras	133.351.567-51	R\$ 7.200,00			
14	Leticia Mesquita	Feminino	Administrativo	136.725.629-63	R\$ 9.450,00			
15	Marcela de Freitas	Feminino	Financeiro	140.745.317-54	R\$ 4.650,00			
16	Marianna Pestana	Feminino	Marketing	110.857.529-80	R\$ 3.700,00			
17	Matheus de Souza	Masculino	Marketing	136.163.401-29	R\$ 7.200,00			
18	Pedro Costa	Masculino	Logística	127.511.539-73	R\$ 16.300,00			
19	Rafael Machado Cardoso	Masculino	Logística	151.782.237-21	R\$ 16.300,00			

Novo  
Funcionário

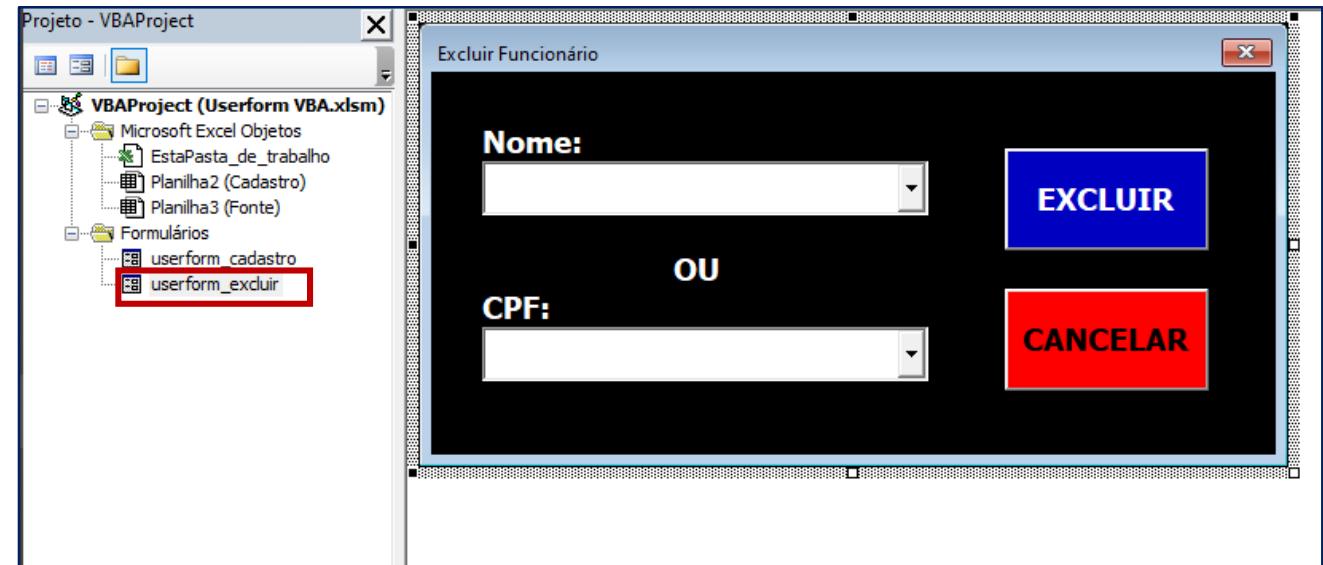
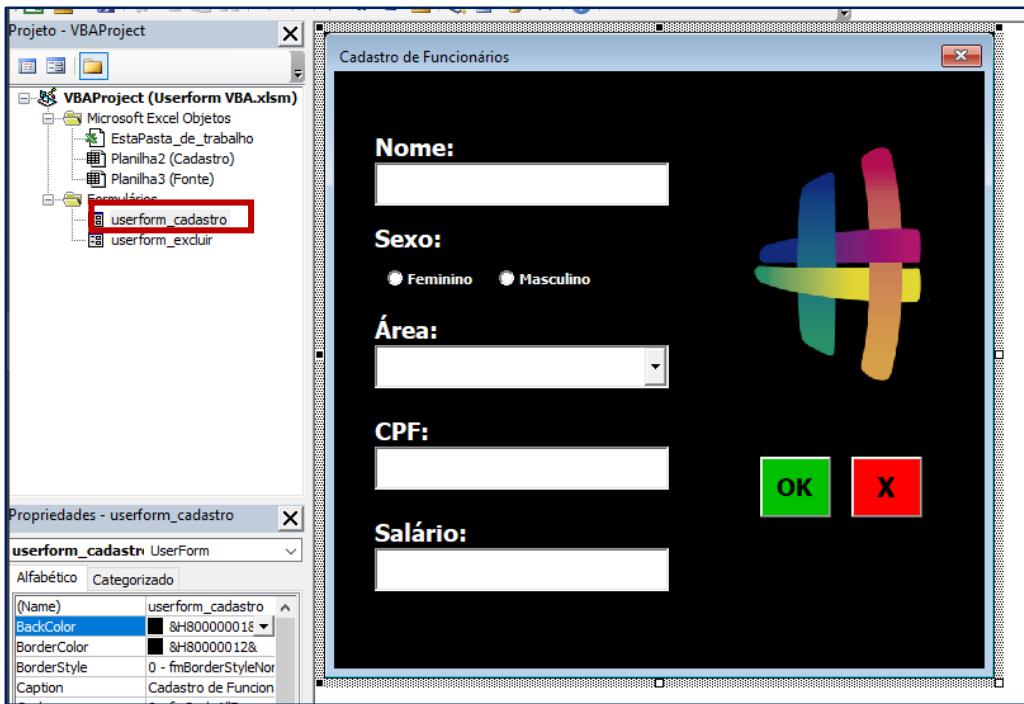
Excluir  
Funcionário

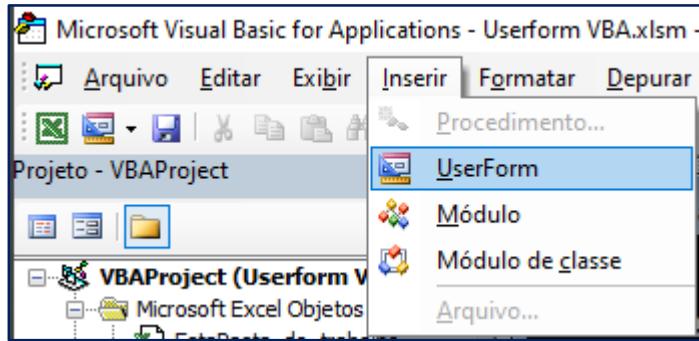
Vamos começar montando o formulário do exemplo mostrado na página anterior.

Primeiro, vamos até o ambiente do VBA para dar início.



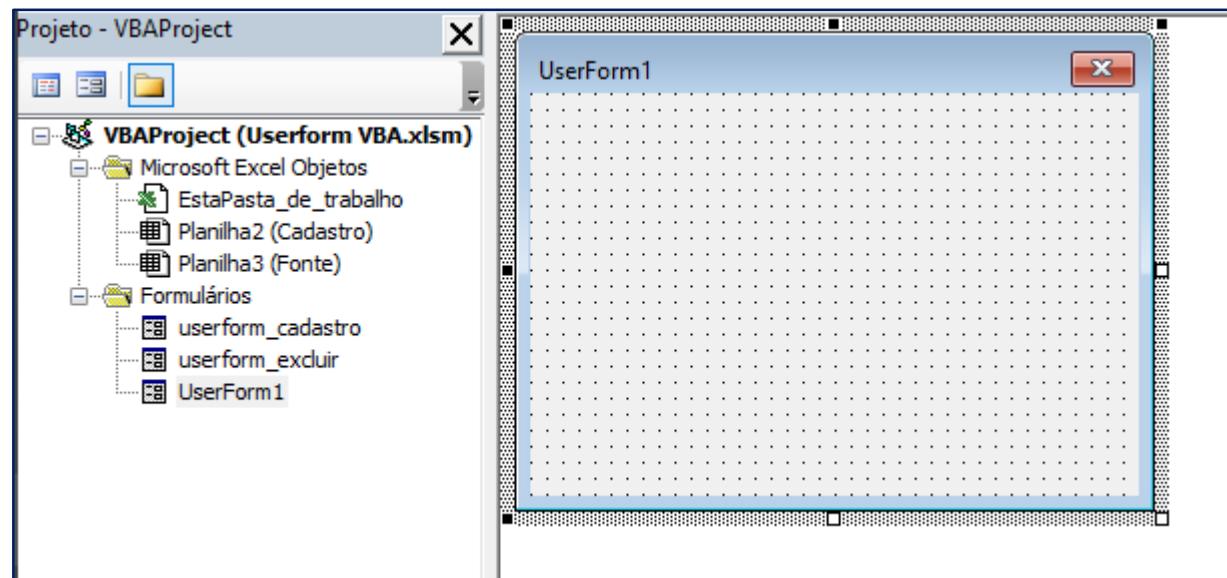
No ambiente do VBA já terão dois exemplos de userforms que você pode dar um duplo clique para visualizar. Os dois userforms são essencialmente os mesmos mostrados no exemplo do início do módulo.

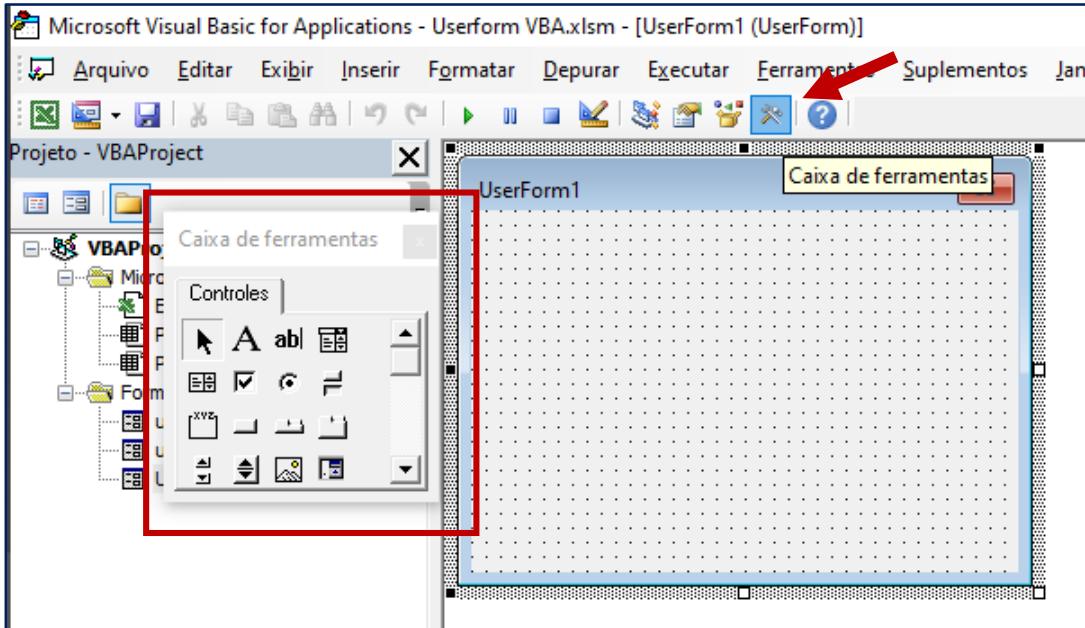




Porém, o nosso objetivo é partir do zero. Então, para criar um novo Userform, você também seguirá o procedimento parecido com a opção de inserir um Módulo: clique na guia **Inserir** e em seguida, clique em **UserForm**.

O userform totalmente em branco aparecerá na tela. Este userform é como se fosse uma imagem. Você pode aumentar ou diminuir o seu tamanho simplesmente arrastando suas extremidades com o mouse.

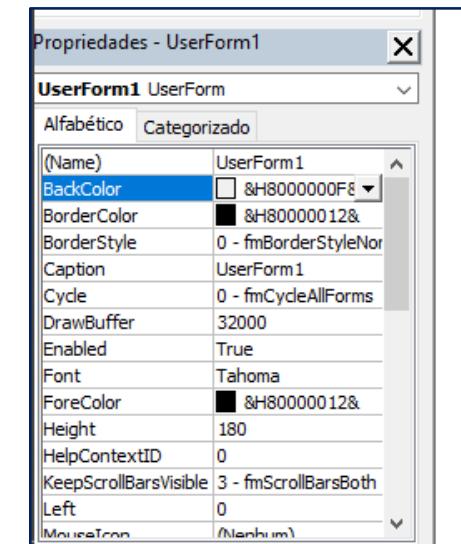




A janela que aparece é literalmente um espaço onde vamos desenhar cada um dos nossos botões. Para começar a desenhá-los e dimensioná-los dentro dessa janela, primeiramente ativamos a **Caixa de Ferramentas**, para que abram todas as opções de controle possíveis de serem adicionadas no UserForm.

Ao criar um UserForm também aparece, no canto inferior esquerdo, uma janela de Propriedades do UserForm, onde poderemos alterar as diferentes características deste userform, tais como: nome, altura, largura, cor, etc.

**Se esta caixinha de propriedades não estiver aparecendo para você, é só ir até a guia Exibir e procurar pela opção: Janela ‘Propriedades’ (ou usar o atalho F4).**



The screenshot shows an Excel spreadsheet with a list of employees in columns A through E. Column A contains names, column B contains sex, column C contains area, column D contains CPF, and column E contains salary. A user form titled "UserForm1" is overlaid on the spreadsheet. The user form has two buttons: "Novo Funcionário" (New Employee) and "Excluir Funcionário" (Delete Employee). The user form also has a close button ("X"). At the bottom of the user form, there are tabs for "Cadastro" (Cadastro), "Fonte" (Fonte), and a plus sign icon.

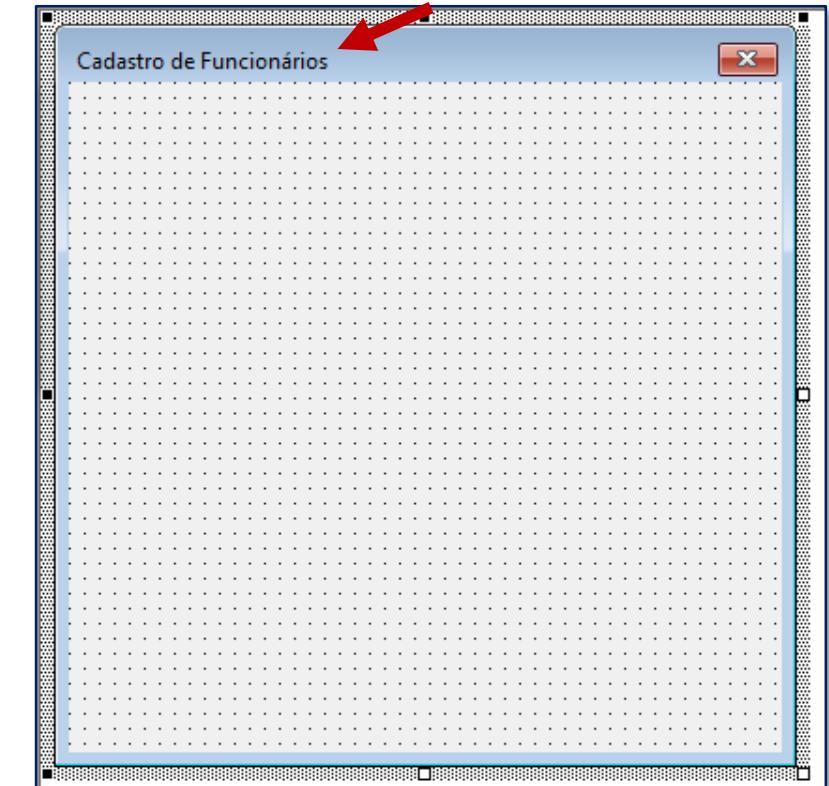
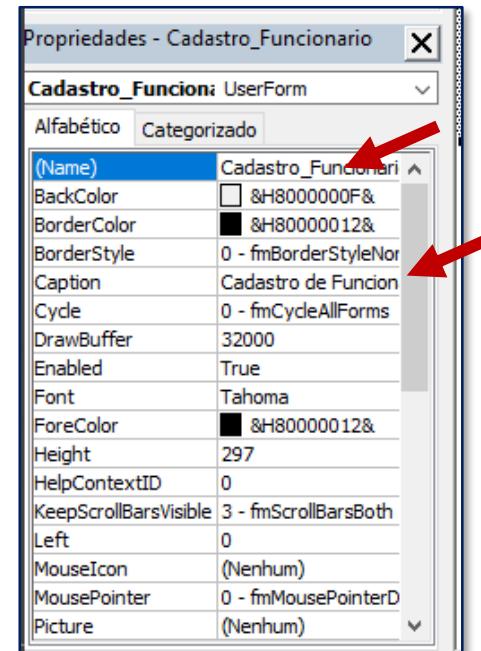
	A	B	C	D	E	F	G	H
1	Nome	Sexo	Área	CPF	Salário			
2	Adriane de Carvalho Gomes	Feminino	RH	082.442.274-27	R\$ 4.650,00			
3	Andressa Rotsztejn	Feminino	Marketing	066				
4	Beatriz Leite Júnior	Feminino	Administrativo	134				
5	Bruna dos Santos Gomes	Feminino	Administrativo	073				
6	Carolina Codeceira	Feminino	Marketing	085				
7	Daniela Muller Braga	Feminino	Logística	127				
8	Elvis de Freitas Derossi	Masculino	RH	104				
9	Gabrielle Andrade da Silva	Feminino	Administrativo	081				
10	Gustavo de Vasconcelos	Masculino	Administrativo	137				
11	Isabella Ronfini	Feminino	RH	133.055.145-07	R\$ 12.520,00			
12	Juan Fernandes	Masculino	RH	107.000.473-49	R\$ 16.300,00			
13	Júlio Fraga	Masculino	Compras	133.351.567-51	R\$ 7.200,00			
14	Leticia Mesquita	Feminino	Administrativo	136.725.629-63	R\$ 9.450,00			
15	Marcela de Freitas	Feminino	Financeiro	140.745.317-54	R\$ 4.650,00			
16	Marianna Pestana	Feminino	Marketing	110.857.529-80	R\$ 3.700,00			
17	Matheus de Souza	Masculino	Marketing	136.163.401-29	R\$ 7.200,00			
18	Pedro Costa	Masculino	Logística	127.511.539-73	R\$ 16.300,00			
19	Rafael Machado Cardoso	Masculino	Logística	151.782.237-21	R\$ 16.300,00			

A princípio, se executarmos (F5) o formulário do jeito que está, ele já irá exibir alguma coisa.

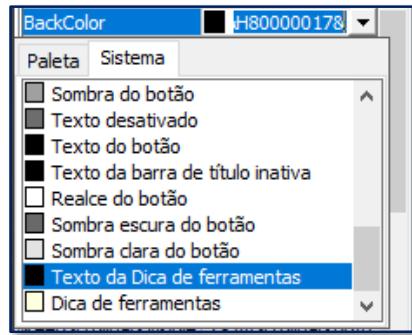
Obviamente ainda temos muito o que fazer, como adicionar os botões ao formulário, trocar o nome que aparece no cabeçalho do Userform, mudar a cor, tamanho, e por ai vai.

Primeiro, vamos começar mudando o nome que aparece no cabeçalho do UserForm. Para isso, precisamos mudar as propriedades do UserForm. Mas antes, você precisa entender a diferença entre (Name) e Caption. Pode ser intuitivo pensar que devemos trocar o (Name) do UserForm para mudar o cabeçalho. Porém, o (Name) é similar ao nome de uma variável. Isso porque, dentro do código, veremos que a maneira como referenciamos o UserForm é através do seu (Name).

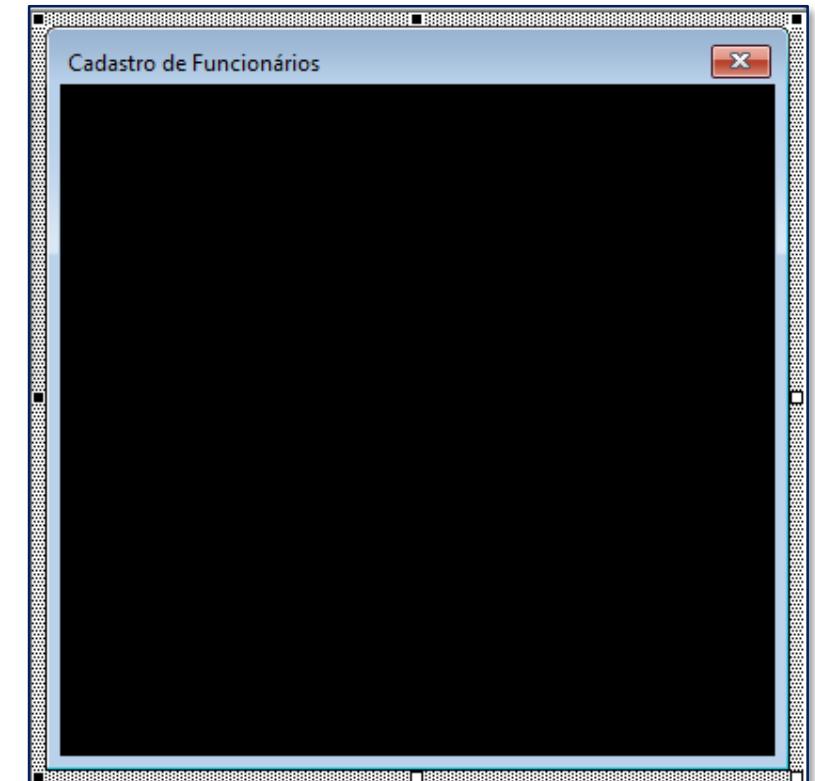
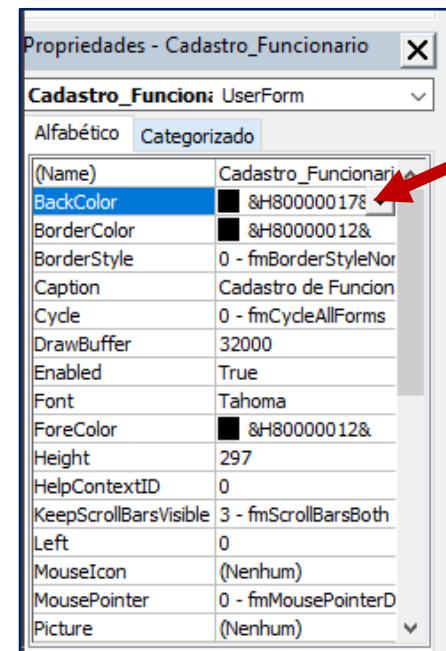
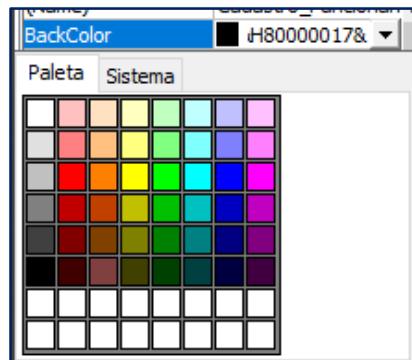
Para mudar o texto que aparece no cabeçalho do UserForm devemos na verdade mudar a propriedade **Caption**. Você pode colocar o texto: **Cadastro Funcionários**. Também é uma boa prática dar um **(Name)** mais intuitivo para o userform, para ficar mais fácil de criar os códigos mais para frente. Nesta propriedade, você pode alterar UserForm1 para **Cadastro\_Funcionario**.



Podemos também mudar a cor de fundo do UserForm. Para isso, você pode mudar a cor na propriedade **BackColor**.



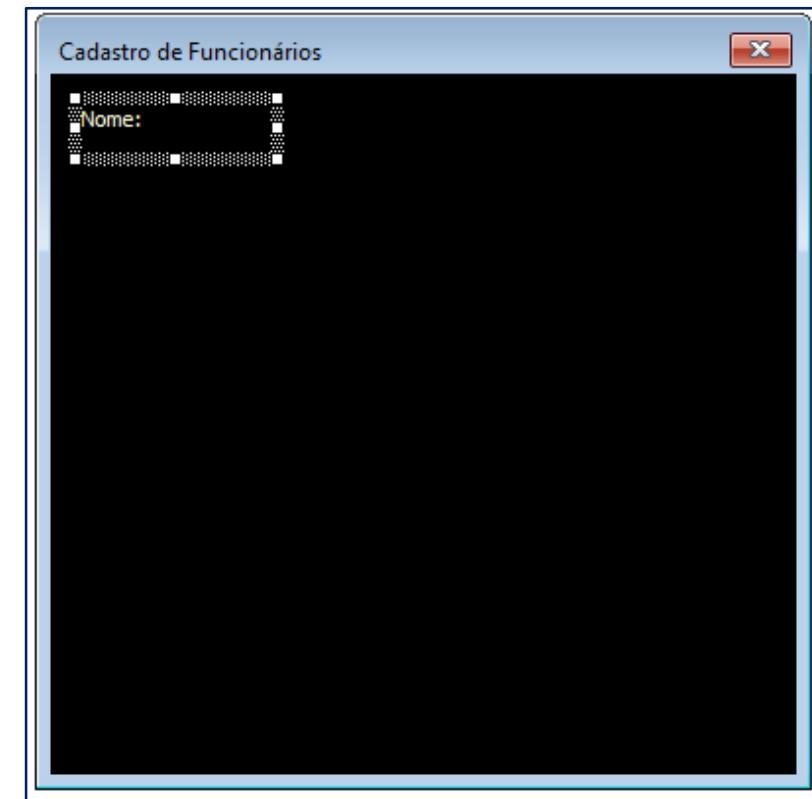
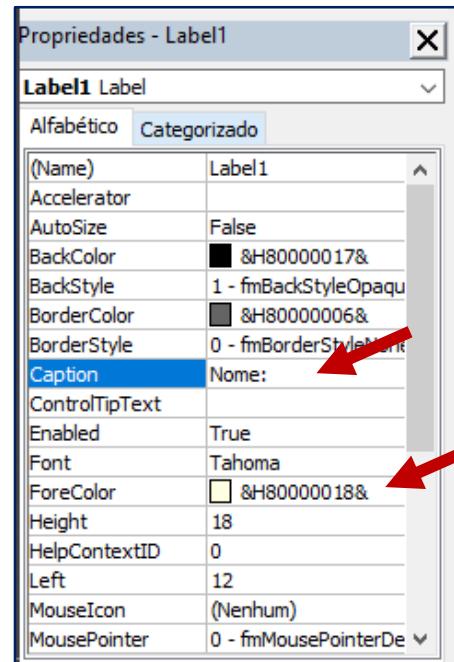
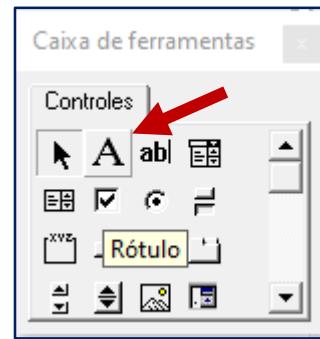
Na opção **Sistema**, você não tem muitas opções de cores, mas se quiser algo mais personalizado pode ir na opção **Paleta**.



O próximo passo é inserir o nosso primeiro objeto, que no caso será o **Rótulo**, que é um texto que colocamos no UserForm para deixar as opções mais intuitivas. Para isso, você terá que usar a caixa de ferramentas, e clicar na letra A indicada na imagem ao lado. Lembre-se: se esta caixinha não estiver aparecendo para você, clique no símbolo de ferramentas indicado na imagem abaixo.

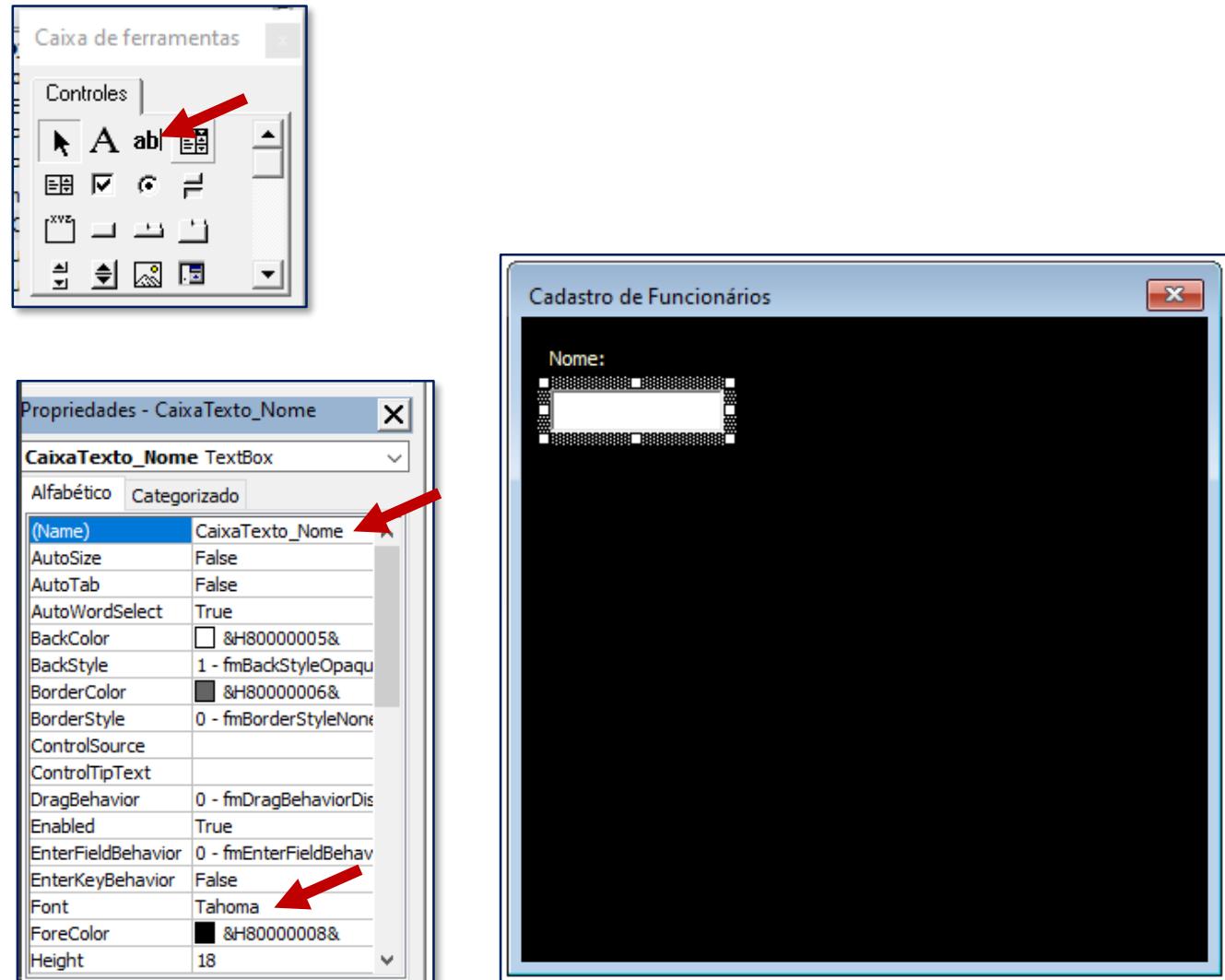


Feito isso, você irá desenhar o Rótulo no seu UserForm. A princípio, o texto aparecerá com a fonte preta. Para conseguir enxerga-lo, você deverá mudar a sua propriedade **ForeColor**. Além disso, mude também o Caption do Rótulo para **Nome:**.

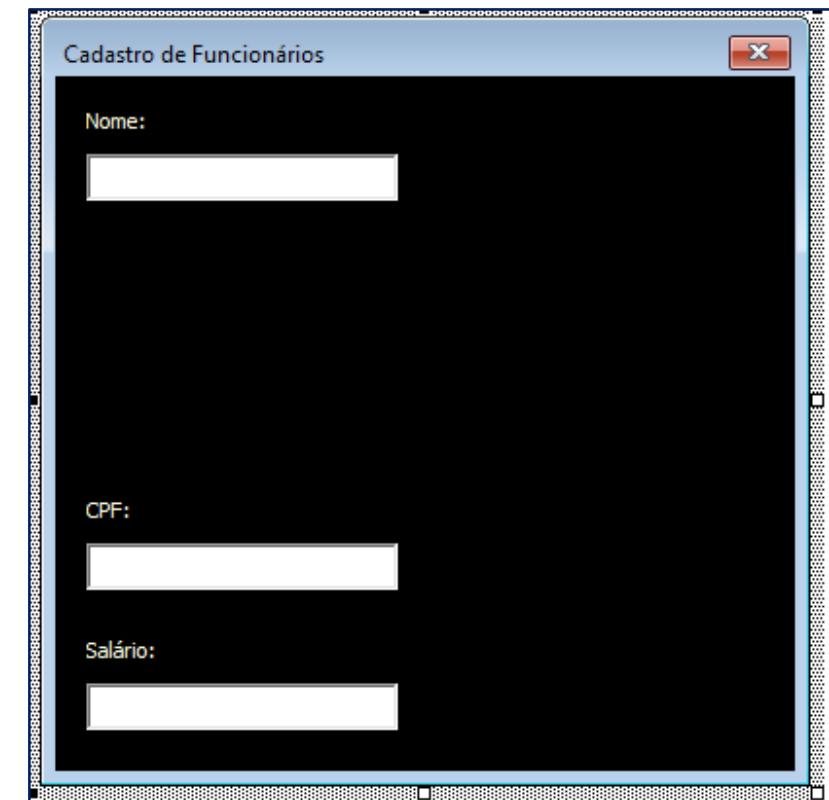
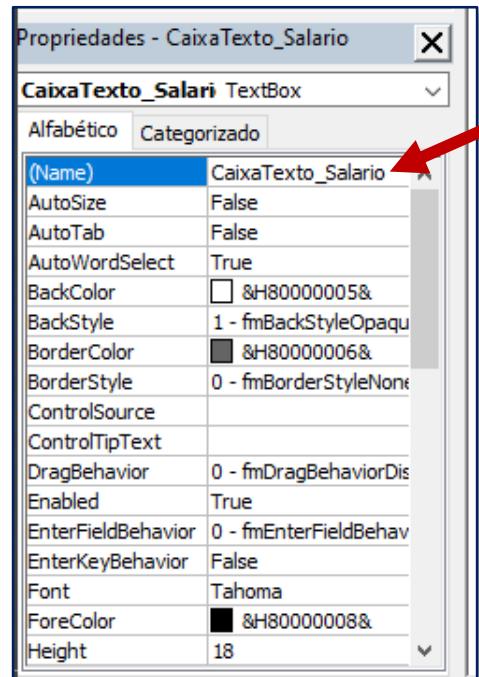
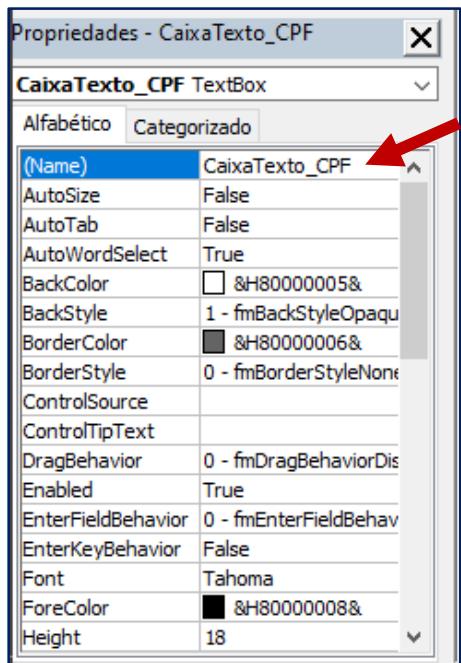


Outro objeto que vamos inserir é a **Caixa de Texto**, para permitir ao usuário inserir algum texto, no caso, o seu nome. A maneira de criar este e qualquer objeto é sempre clicar na opção dentro da Caixa de Ferramentas e desenhar no UserForm.

Com relação às suas propriedades, você pode mudar a Font (por exemplo, tamanho do texto) e também mudar o (Name) para CaixaTexto\_Nome. Estes nomes que estamos dando aos objetos na propriedade (Name) serão muito importantes na hora que começarmos a criar códigos por trás desses objetos.



Como exercício, tente agora inserir mais dois Rótulos e duas Caixas de Texto: um para preencher o CPF e a outra para preencher o salário. Lembre-se também de mudar a propriedade (Name) dos objetos Caixa de Texto.



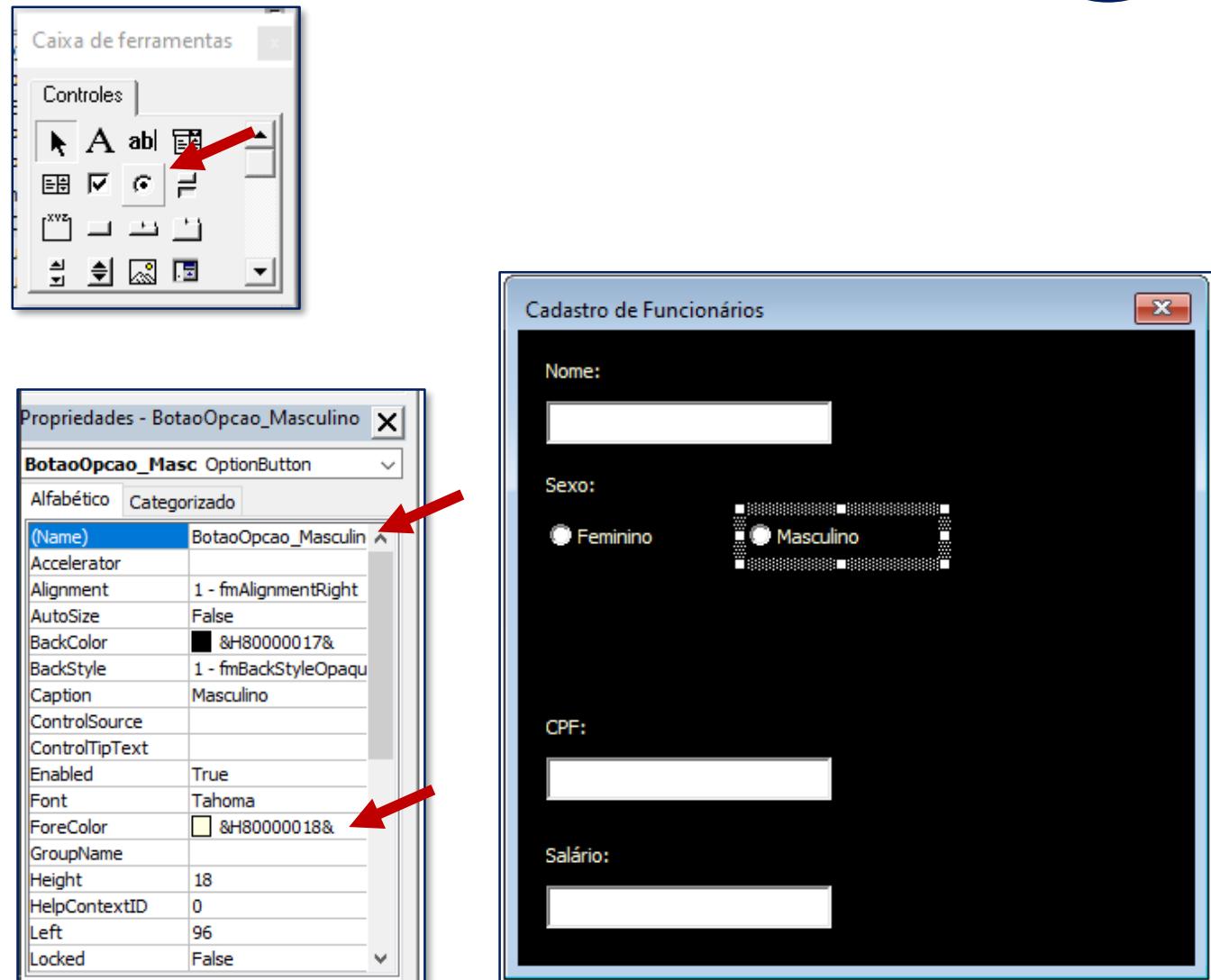
Agora vamos adicionar o **Botão de Opção**, para que o usuário possa marcar entre as opções: Feminino e Masculino. Vamos fazer as configurações necessárias:

- Alterar o tamanho
- Mudar a cor da letra em ForeColor para branco

Além disso, vamos também mudar o (Name) dos botões para:

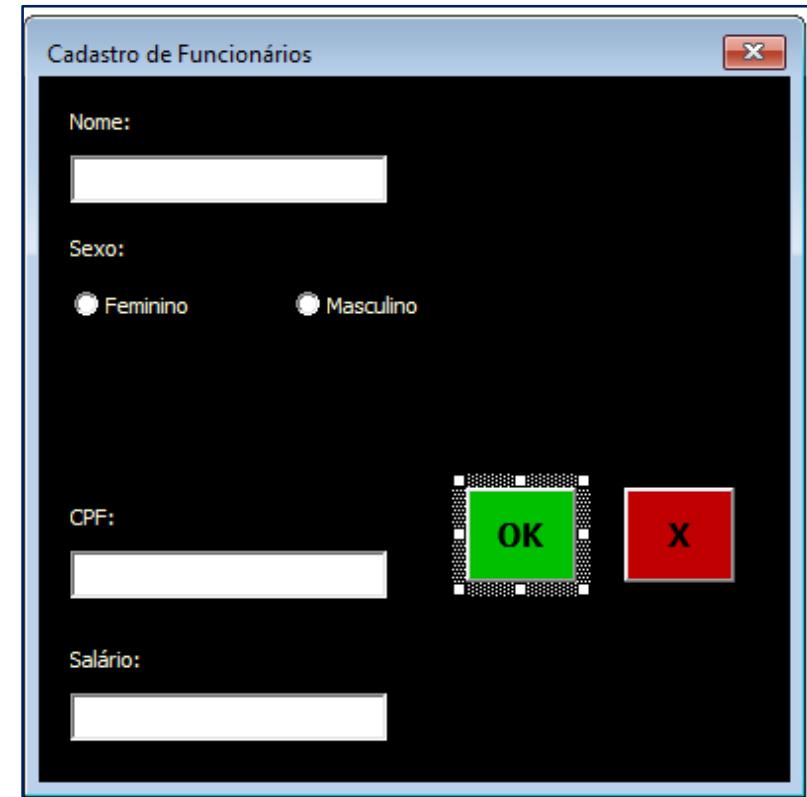
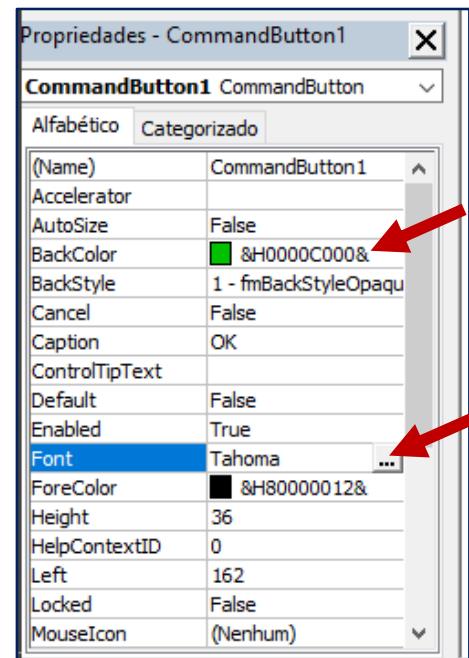
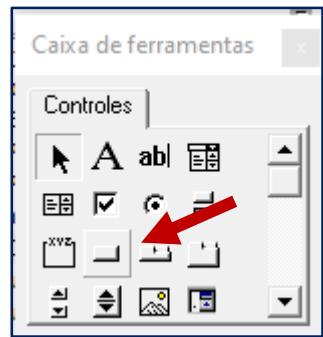
- BotaoOpcão\_Feminino
- BotaoOpcão\_Masculino

Coloque também um cabeçalho para essas opções. Pode chamá-lo de 'Sexo':



Os próximos dois botões que vamos inserir no UserForm são o **Botão de Comando**. A opção está mostrada na imagem ao lado. Para mudar a cor de fundo e editar a fonte da letra, é só você mudar as propriedades **BackColor** e **Font**, respectivamente.

Lembre-se também de mudar o **Caption** de cada botão e o **(Name)** de cada um para **BotaoComando\_OK** e **BotaoComando\_Cancelar**.

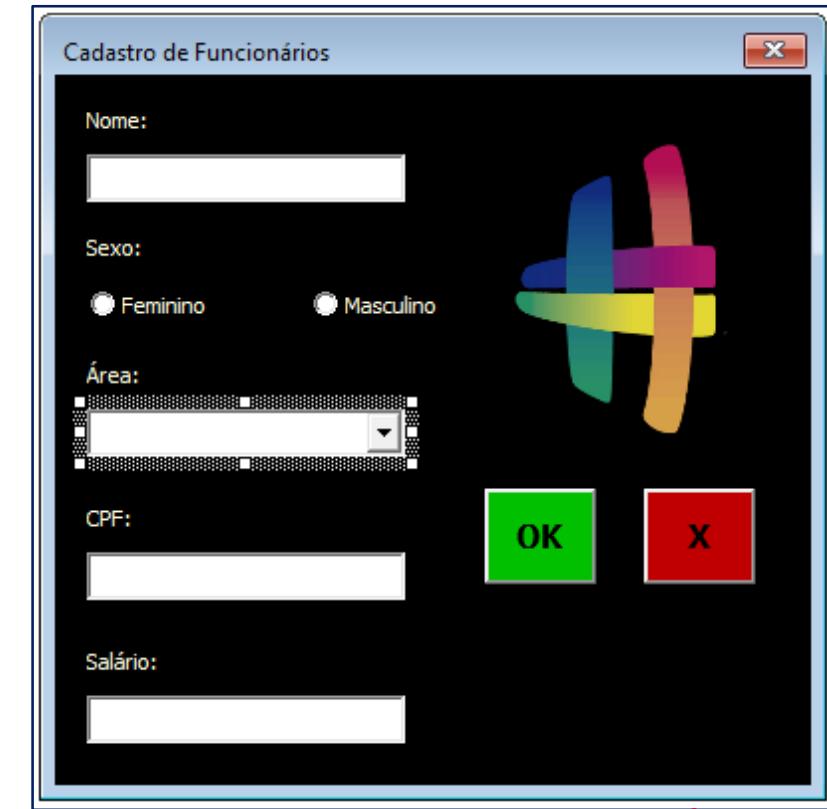
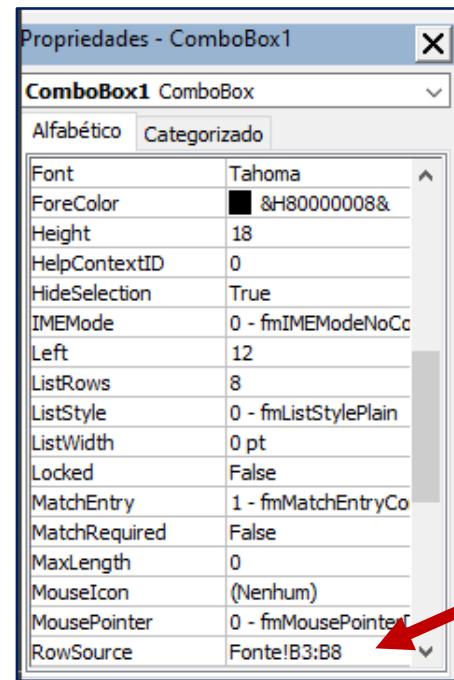
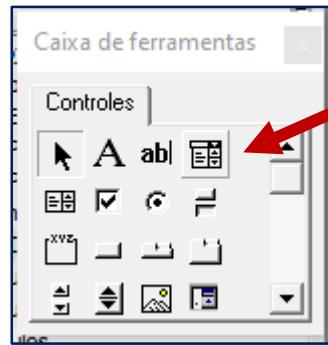


Outro botão muito utilizado é a **Caixa de Combinação**. Ao clicar neste botão, é exibida uma lista de opções onde o usuário pode escolher uma delas. Para passar a lista de opções deste botão, usamos a propriedade RowSource.

Nela, identificamos na nossa planilha o intervalo onde esta lista se encontra. No caso, já deixamos uma lista com as áreas no intervalo B3:B8 da aba Fonte. Para que o botão identifique a aba do arquivo onde estão os dados, precisamos escrever exatamente desta forma: **Fonte!B3:B8**.

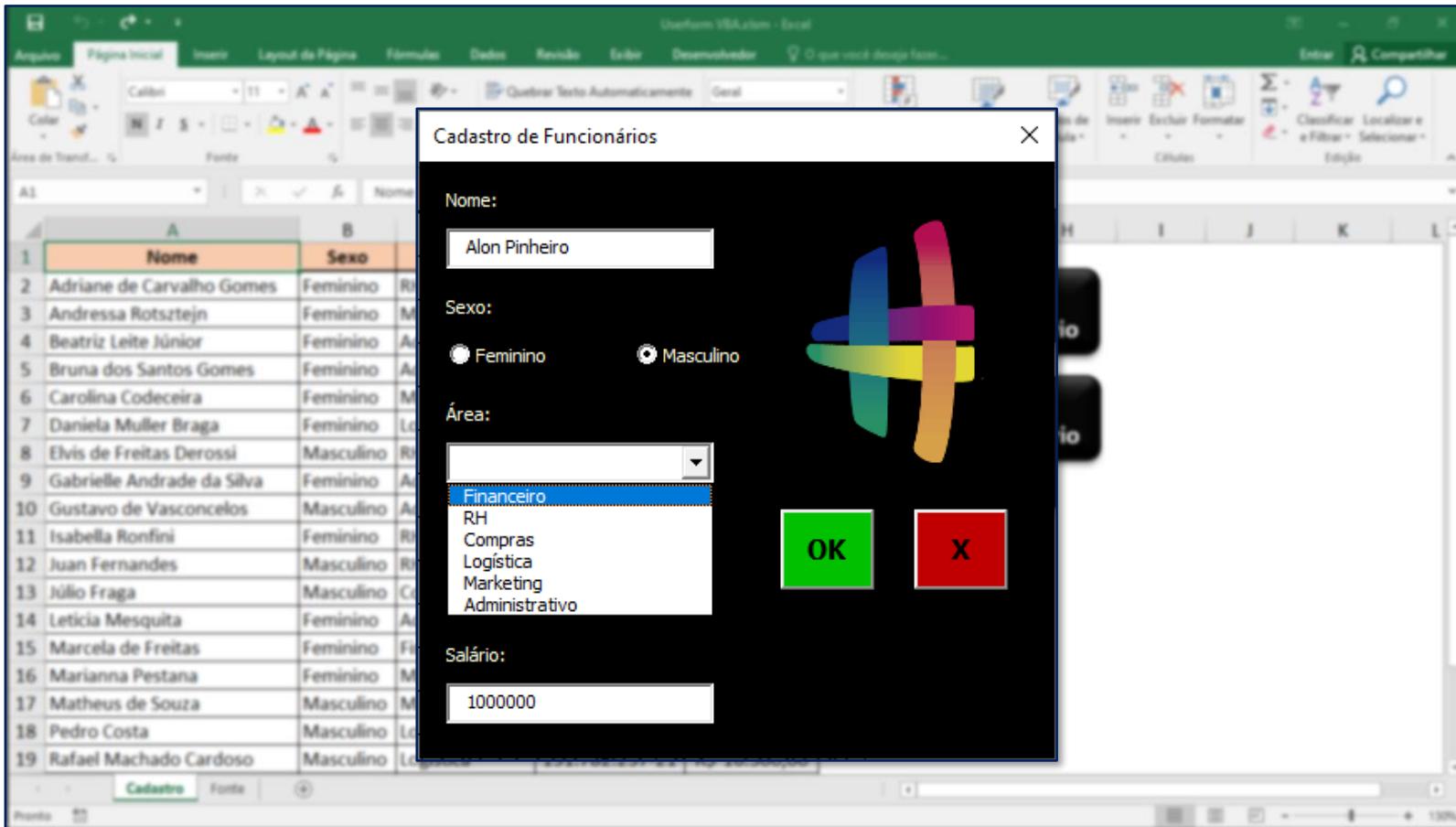
	B	C	D	E
2	<b>Área</b>			
3	Financeiro			
4	RH			
5	Compras			
6	Logística			
7	Marketing			
8	Administrativo			
9				
10				

Cadastro      **Fonte**



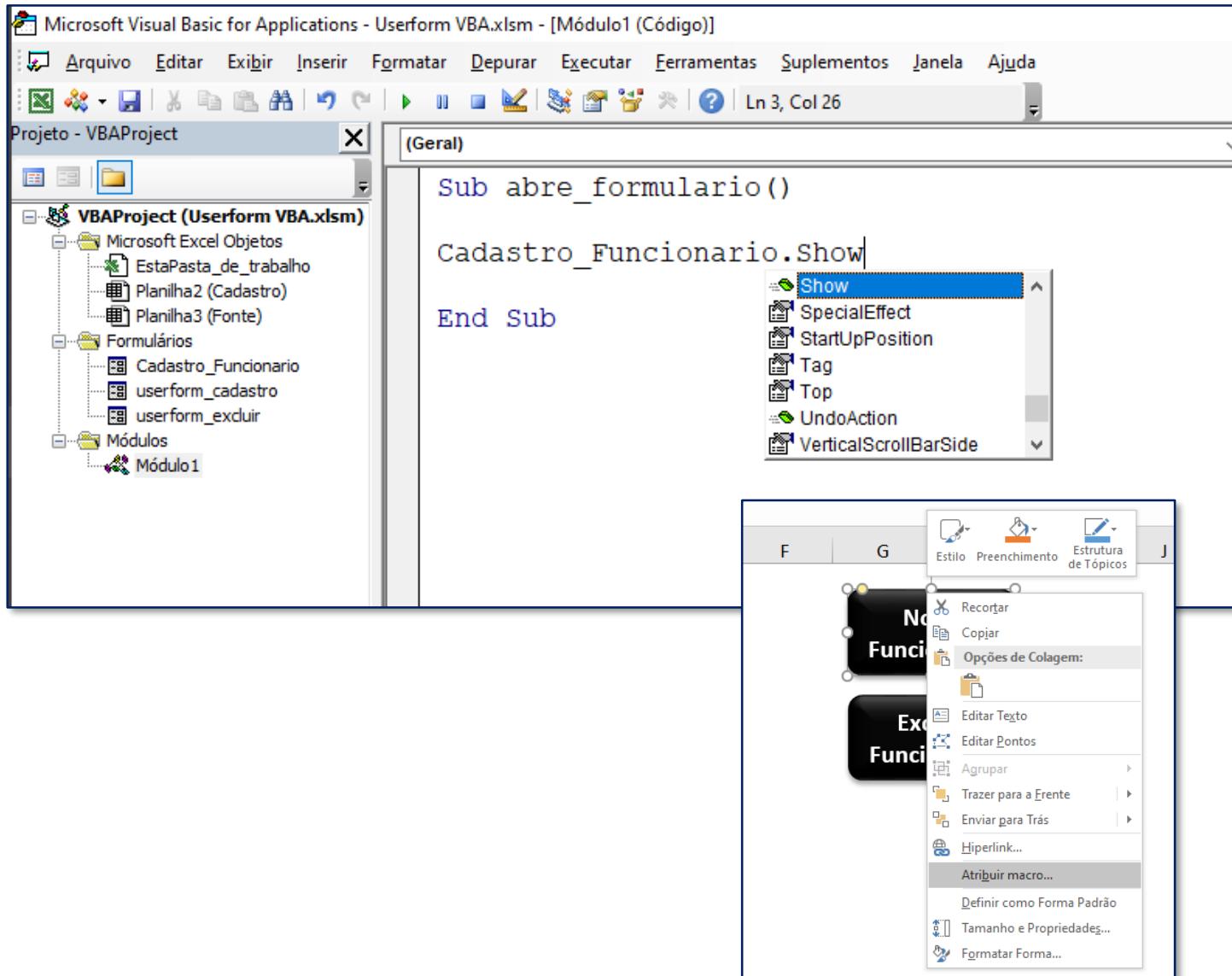
## Módulo 12 – UserForms – Caixa de Combinação

386



O layout do nosso userform já está pronto. Se você executá-lo (atälho F5) ele será exibido e você já pode interagir com ele.

Porém, este formulário não faz nada. Ainda precisamos criar toda a lógica por trás que nos permitirá armazenar em um código todos estes valores preenchidos e registrar na nossa planilha.



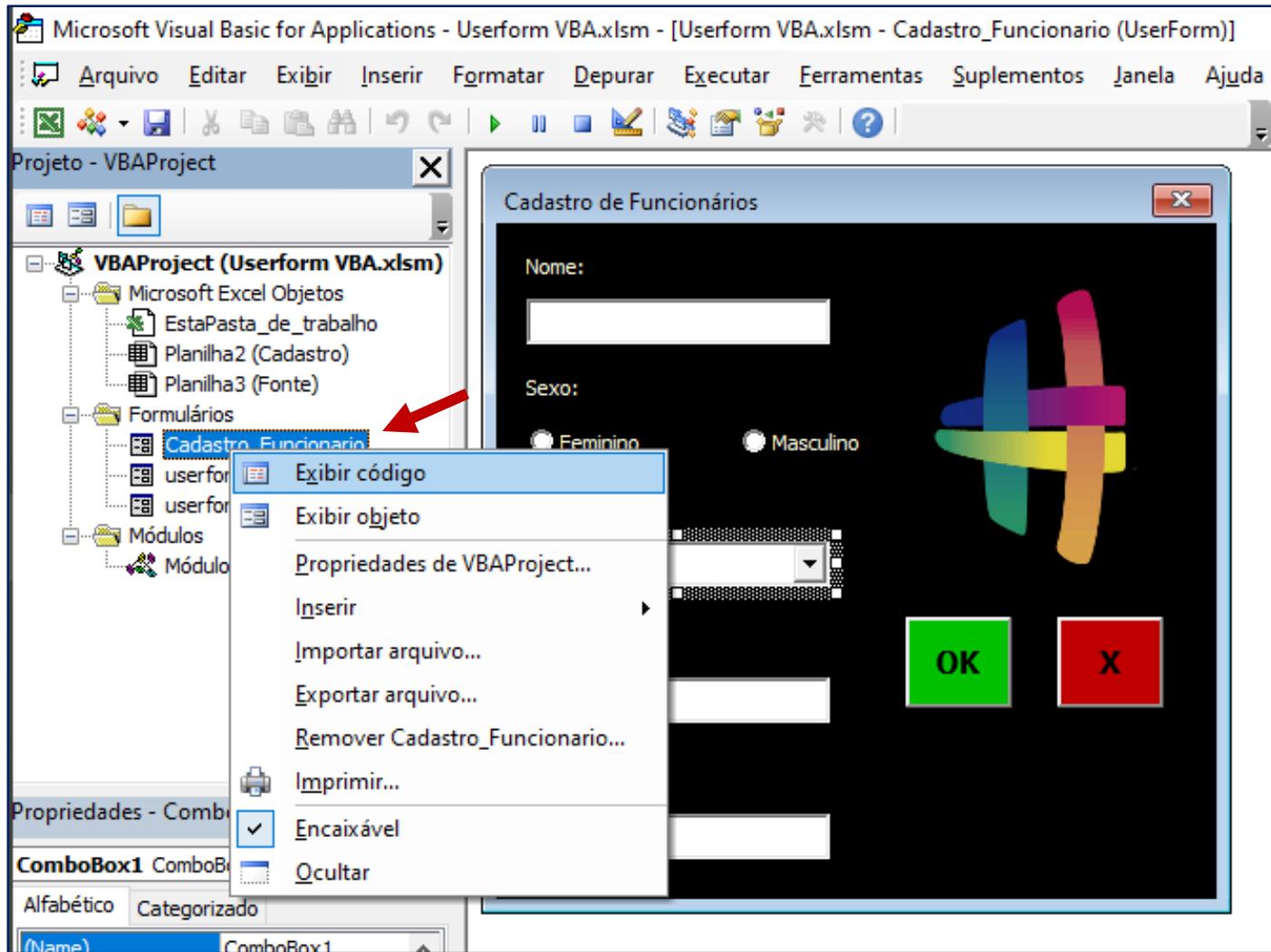
Obviamente na prática não vamos ficar clicando no botão de executar no ambiente VBA ou usando o atalho F5. O mais prático seria clicar em um botão que automaticamente exibe esse formulário para a gente.

Para isso, precisamos criar um novo módulo e nele uma nova Sub. Para exibir o formulário, utilizamos o comando ao lado.

Se você não tivesse renomeado o userform como fizemos anteriormente, você teria que fazer algo do tipo:

## UserForm1.Show

Agora, esta Sub você poderá associar ao botão da nossa planilha.



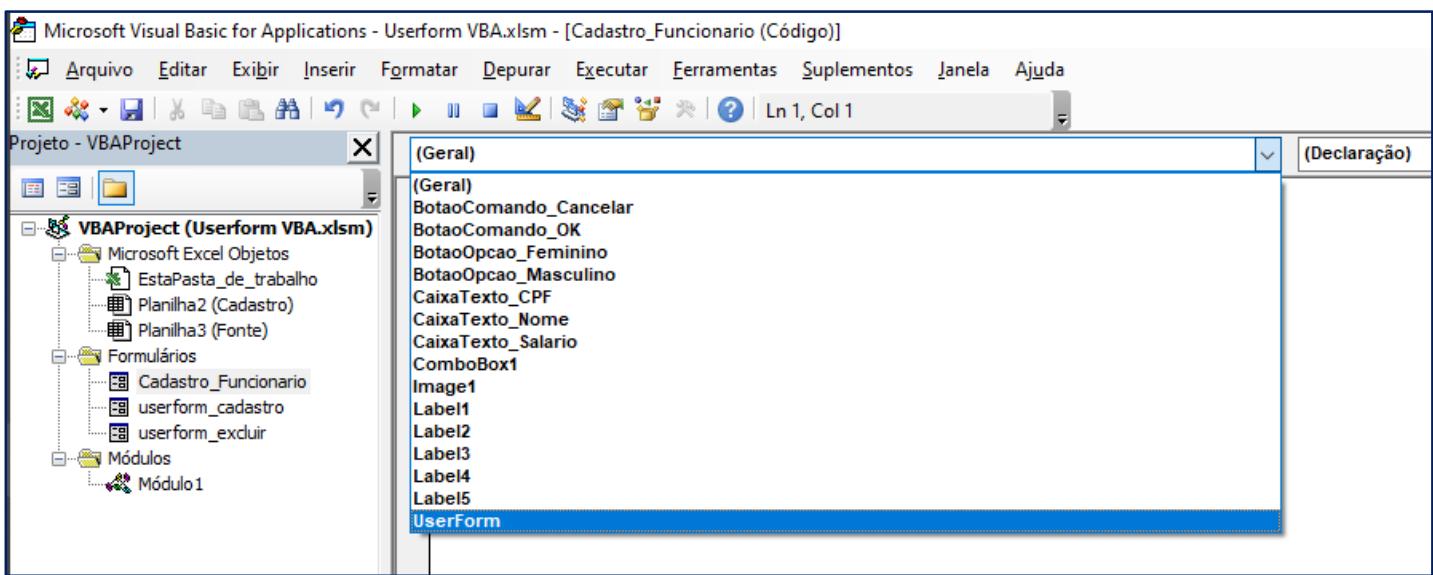
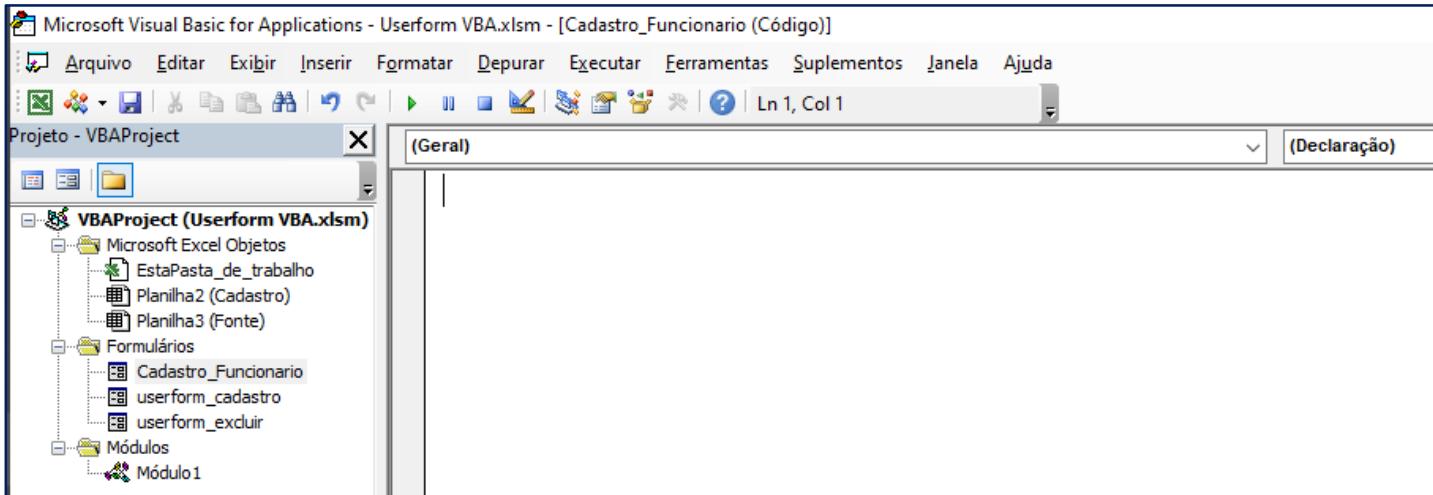
Com relação ao preenchimento das opções da Caixa de Combinação, a maneira como fizemos anteriormente, com o RowSource, não é automática. Isso significa que, se adicionarmos uma nova linha com uma nova área na aba Fonte, o botão não se atualizará automaticamente.

Uma maneira melhor que temos para preencher este botão é através de um código. E a ideia é que, sempre que abrirmos o UserForm, esta lista de opções deve se atualizar automaticamente. O que queremos fazer aqui então é semelhante ao que já vimos em Eventos: executar um código para atualizar a caixa de combinação sempre que abrirmos o formulário.

Para criar este código, clicamos no UserForm com o botão direito e em seguida, na opção Exibir Código.

## Módulo 12 – UserForms – Preenchendo a Caixa de Combinação (Parte 1)

389



Será aberta uma janela em branco onde podemos começar a criar o nosso código. Como queremos na verdade trabalhar com um Evento de UserForm (não se assuste, a lógica é exatamente a mesma da que já vimos no módulo anterior), temos que pensar qual será o objeto que poderá sofrer uma ação. No módulo anterior, trabalhamos basicamente com Workbook e Worksheet. Aqui, vamos trabalhar com o objeto UserForm.

A lista ao lado mostra todos os objetos que criamos, tais como o botão de opção, a caixa de texto, o botão de comando, e por ai vai. Vamos escolher o objeto UserForm.

```
UserForm
Private Sub UserForm_Click()
End Sub
```

```
UserForm
Private Sub UserForm_Initialize()
    'CaixaCombinacao_Area.RowSource = "Fonte!B3:B8"
    linha_area = Sheets("Fonte").Range("B2").End(xlDown).Row
    CaixaCombinacao_Area.RowSource = "Fonte!B3:B" & linha_area
End Sub
```

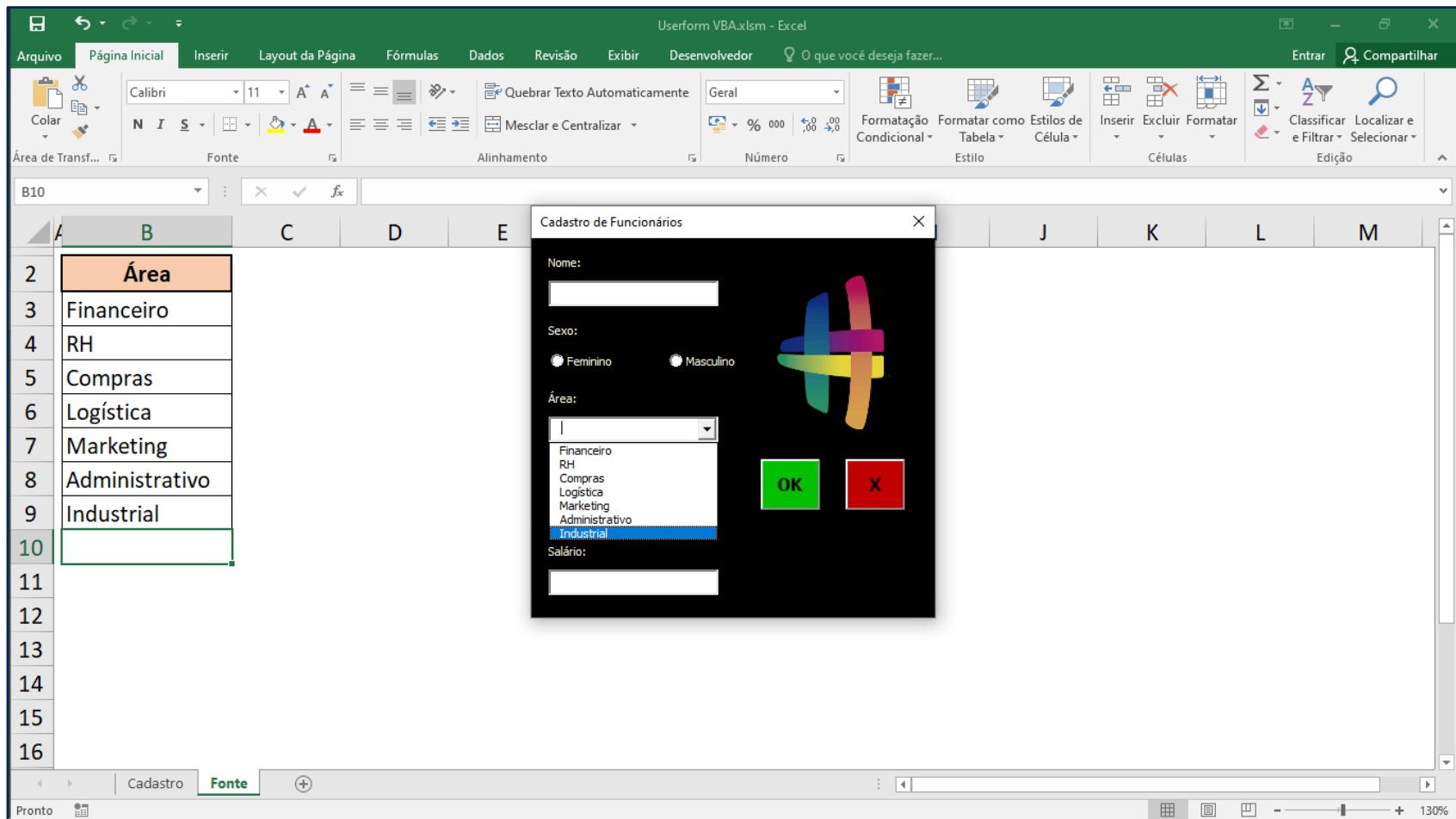
O evento que aparece automaticamente é o de clicar no UserForm. Porém, como queremos que a nossa caixa de combinação mude sempre que abrirmos o UserForm, vamos escolher a opção **Initialize**.

Se fossemos configurar o RowSource como fizemos anteriormente através de um código, usariamos a estrutura comentada em verde. CaixaCombinacao\_Area é o nome que demos à caixa de combinação. Se você não tiver feito isso, volte no formulário, selecione a caixa de combinação e mude a propriedade (Name).

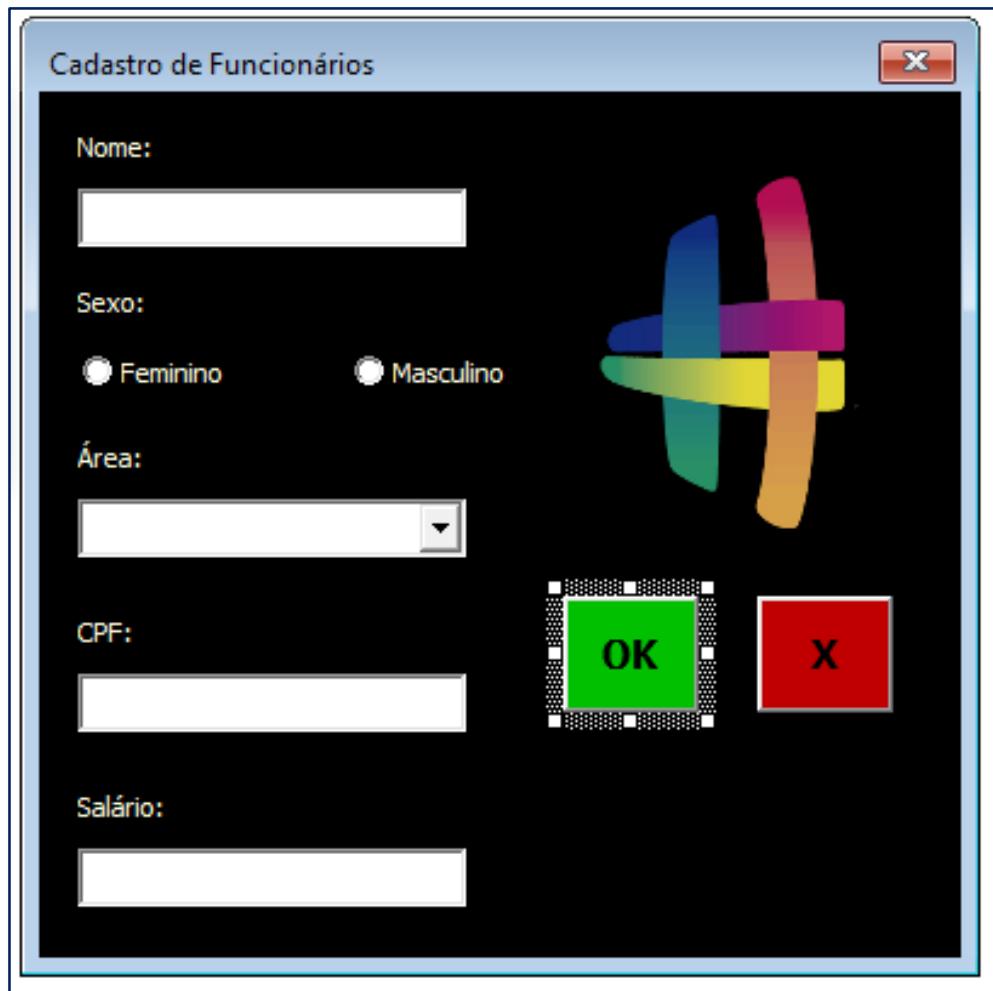
Como queremos que esta lista de áreas seja automática e considere sempre a última linha, criamos o código ao lado.

## Módulo 12 – UserForms – Preenchendo a Caixa de Combinação (Parte 2)

391



Pronto. Agora a caixa de combinação está automática para qualquer quantidade de linhas na tabela de áreas.



Agora que o UserForm está 100% funcional, precisamos fazer com que, ao clicar no botão de Ok, todas as informações preenchidas no formulário sejam registradas na nossa tabela.

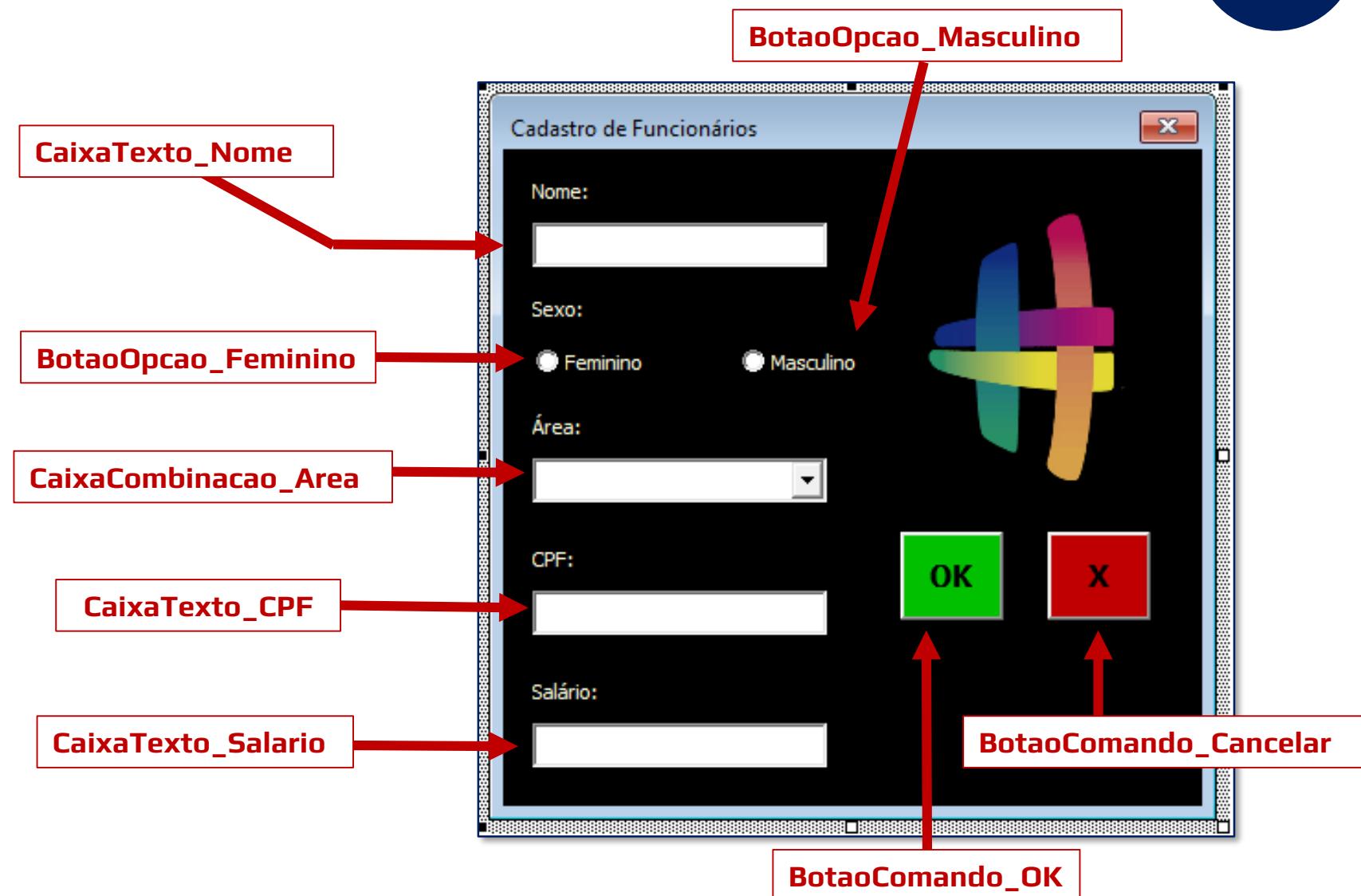
Mais uma vez, queremos que uma macro seja executada assim que uma determinada ação for executada, no caso, clicar no botão de Ok.

Você pode então dar um duplo clique neste botão que você será levado diretamente para a janela de código e o evento BotaoComando\_OK\_Click será criado.

```
BotaoComando_OK
Private Sub BotaoComando_OK_Click()
|
End Sub
```

Antes de prosseguir, é importante agora relembrar os nomes de cada um dos objetos do nosso userform.

Precisaremos deles para conseguir armazenar, dentro do código, os valores registrados pelo usuário e em seguida registrar nas células da tabela da aba Cadastro.



```

Botaocomando_OK
Click

Private Sub Botaocomando_OK_Click()
Sheets("Cadastro").Activate
ult_linha = Range("A1").End(xlDown).Row + 1
Cells(ult_linha, 1).Value = CaixaTexto_Nome.Value
If Botaooacao_Feminino.Value = True Then
    Cells(ult_linha, 2).Value = "Feminino"
ElseIf Botaooacao_Masculino.Value = True Then
    Cells(ult_linha, 2).Value = "Masculino"
End If
Cells(ult_linha, 3).Value = CaixaCombinacao_Area.Value
Cells(ult_linha, 4).Value = CaixaTexto_CPF.Value
Cells(ult_linha, 5).Value = CaixaTexto_Salario.Value
End Sub

```

O código final associado ao botão de comando OK está mostrado ao lado. Basicamente, precisamos descobrir a última linha onde será preenchido o novo funcionário. Em seguida, em cada uma das células da nova linha, preenchemos as informações registradas nos botões pelo usuário.

O código está praticamente pronto. Se você voltar na planilha e registrar um novo funcionário, ele será registrado automaticamente na tabela.

	A	B	C	D	E
20	Renan Freire	Masculino	Marketing	101.981.446-34	R\$ 7.200,00
21	Thaís Ribeiro	Feminino	Financeiro	121.464.801-28	R\$ 16.300,00
22	Victor Ruzza de Carvalho	Masculino	Logística	097.867.609-77	R\$ 1.600,00
23	Yuri Araujo Senna	Masculino	RH	144.402.781-94	R\$ 12.320,00
24	Alon	Masculino	Financeiro	12345678910	100000
25					

← → Cadastro Fonte + : ↻ ↻

```
BotaoComando_OK
    Click

Private Sub BotaoComando_OK_Click()
    Sheets("Cadastro").Activate
    ult_linha = Range("A1").End(xlDown).Row + 1
    Cells(ult_linha, 1).Value = CaixaTexto_Nome.Value
    If BotaoOpcao_Feminino.Value = True Then
        Cells(ult_linha, 2).Value = "Feminino"
    ElseIf BotaoOpcao_Masculino.Value = True Then
        Cells(ult_linha, 2).Value = "Masculino"
    End If
    Cells(ult_linha, 3).Value = CaixaCombinacao_Area.Value
    Cells(ult_linha, 4).Value = CaixaTexto_CPF.Value
    Cells(ult_linha, 5).Value = CaixaTexto_Salario.Value
    Range("A" & ult_linha - 1 & ":E" & ult_linha - 1).Copy
    Range("A" & ult_linha & ":E" & ult_linha).PasteSpecial xlPasteFormats
    Application.CutCopyMode = False
End Sub
```

Uma otimização que poderíamos fazer antes de executar a macro anterior é formatar, pelo código, da mesma forma que a tabela já estava: bordas, formatos de moeda, e por ai vai.

As linhas de comando que farão isso estão marcadas ao lado. Sempre que adicionarmos uma nova linha, queremos formatá-la da mesma forma que a linha anterior. Quem nos informa a linha onde preenchemos um novo cadastro é a variável **ult\_linha**. É nela que queremos colar o formato (xlPasteFormats).

Porém, antes precisamos copiar essa formatação de algum lugar: no caso, da linha imediatamente anterior à **ult\_linha** (por isso o **ult\_linha - 1** dentro do Range de cópia).

Feito isso, desativamos o modo de recorte com o comando **CutCopyMode = False**, como já fizemos em exemplos passados.

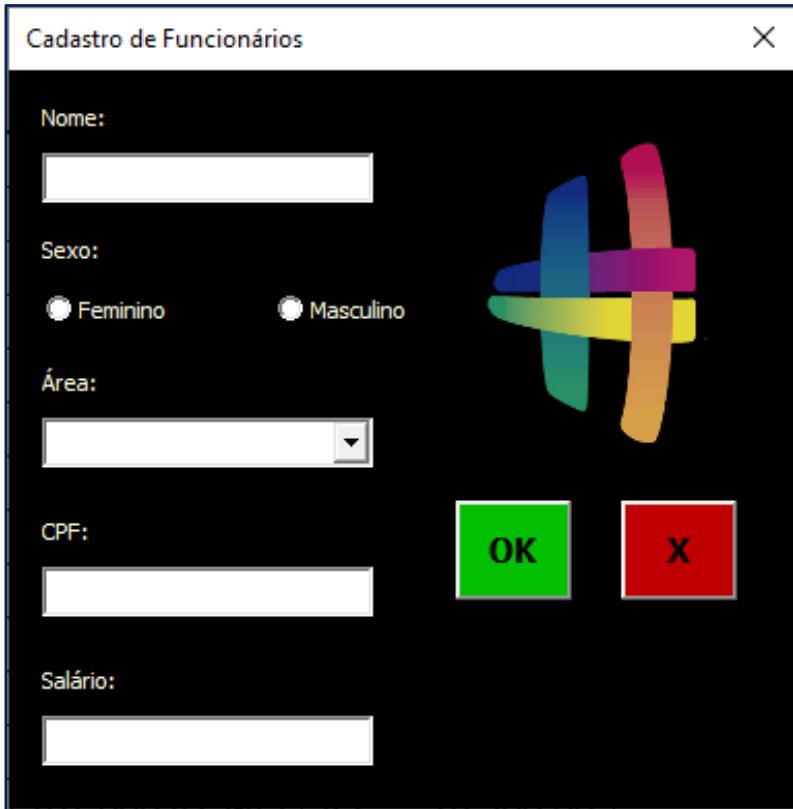
	A	B	C	D	E	
15	Marcela de Freitas	Feminino	Financeiro	140.745.317-54	R\$ 4.650,00	
16	Marianna Pestana	Feminino	Marketing	110.857.529-80	R\$ 3.700,00	
17	Matheus de Souza	Masculino	Marketing	136.163.401-29	R\$ 7.200,00	
18	Pedro Costa	Masculino	Logística	127.511.539-73	R\$ 16.300,00	
19	Rafael Machado Cardoso	Masculino	Logística	151.782.237-21	R\$ 16.300,00	
20	Renan Freire	Masculino	Marketing	101.981.446-34	R\$ 7.200,00	
21	Thaís Ribeiro	Feminino	Financeiro	121.464.801-28	R\$ 16.300,00	
22	Victor Ruzza de Carvalho	Masculino	Logística	097.867.609-77	R\$ 1.600,00	
23	Yuri Araujo Senna	Masculino	RH	144.402.781-94	R\$ 12.320,00	
24	Alon	Masculino	Financeiro	123.456.789-10	R\$ 1.000.000,00	
25						
26						
27						
28						
29						
30						
31						
32						
33						

Pronto! Agora toda nova linha da tabela terá o mesmo formato da tabela.

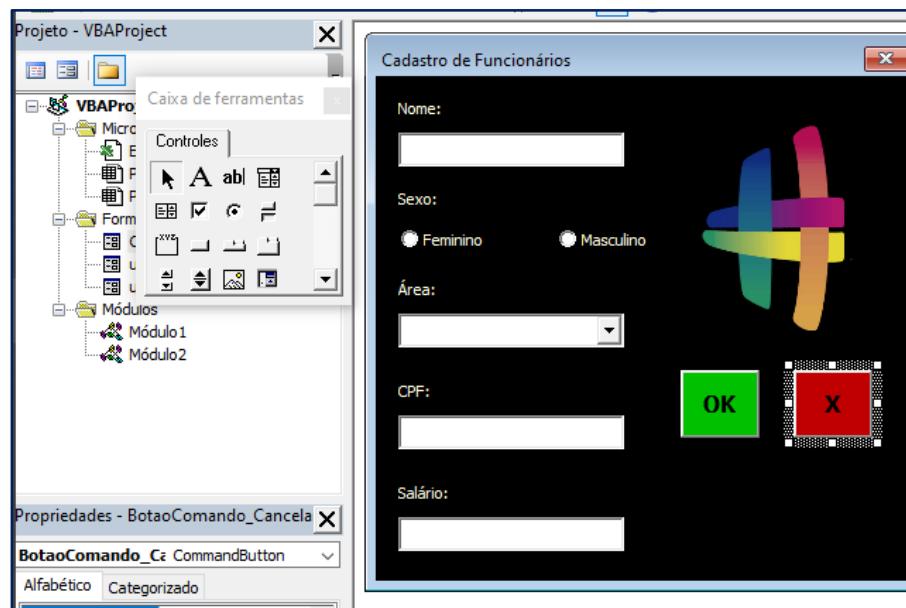
```
Private Sub BotaoComando_OK_Click()
    Sheets("Cadastro").Activate
    ult_linha = Range("A1").End(xlDown).Row + 1
    Cells(ult_linha, 1).Value = CaixaTexto_Nome.Value
    If BotaoOpcao_Feminino.Value = True Then
        Cells(ult_linha, 2).Value = "Feminino"
    ElseIf BotaoOpcao_Masculino.Value = True Then
        Cells(ult_linha, 2).Value = "Masculino"
    End If
    Cells(ult_linha, 3).Value = CaixaCombinacao_Area.Value
    Cells(ult_linha, 4).Value = CaixaTexto_CPF.Value
    Cells(ult_linha, 5).Value = CaixaTexto_Salario.Value
    Range("A" & ult_linha - 1 & ":E" & ult_linha - 1).Copy
    Range("A" & ult_linha & ":E" & ult_linha).PasteSpecial xlPasteFormats
    Application.CutCopyMode = False
    Columns("A:E").Select
    ActiveWorkbook.Worksheets("Cadastro").Sort.SortFields.Clear
    ActiveWorkbook.Worksheets("Cadastro").Sort.SortFields.Add Key:=Range("A:A")
    , SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
    With ActiveWorkbook.Worksheets("Cadastro").Sort
        .SetRange Range("A:E")
        .Header = xlYes
        .MatchCase = False
        .Orientation = xlTopToBottom
        .SortMethod = xlPinYin
        .Apply
    End With
    Range("A1").Select
End Sub
```

Outra otimização que podemos fazer é colocar a nossa tabela em ordem alfabética ao finalizar a execução do código.

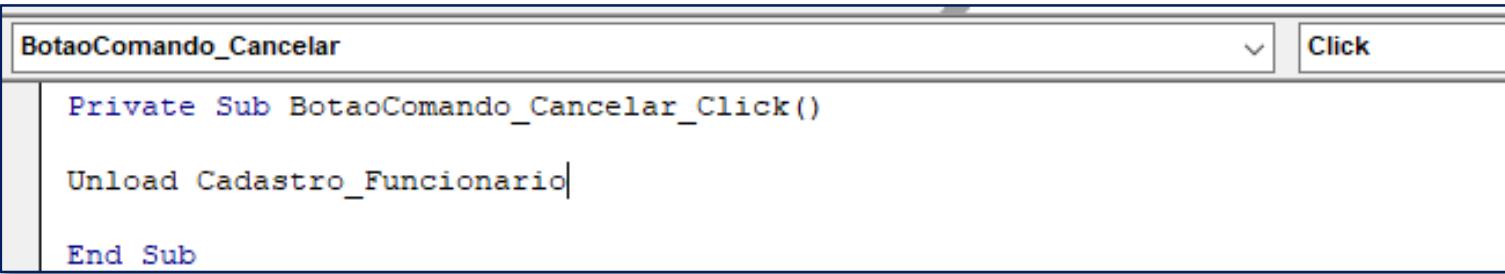
Para ter acesso aos comandos que vão classificar a nossa tabela, podemos gravar uma macro usando a opção **Classificar** da guia **Dados**. Já fizemos este exercício várias vezes, então caso você tenha esquecido do passo a passo para fazer isso, você pode voltar nas páginas 24, 246 e 247 para relembrar o passo a passo.



O último botão que falta a gente configurar é o de Cancelar. A ideia é que, se a pessoa abrir o formulário e desistir de fazer um novo cadastro, ela deve conseguir clicar no botão vermelho de Cancelar e fechar o UserForm.



Para fazer isso, voltamos no ambiente VBA e damos um duplo clique no botão vermelho. Isso abrirá o código onde começaremos a escrever os comandos para fazer com que o UserForm se encerre assim que a gente clicar nesse botão.



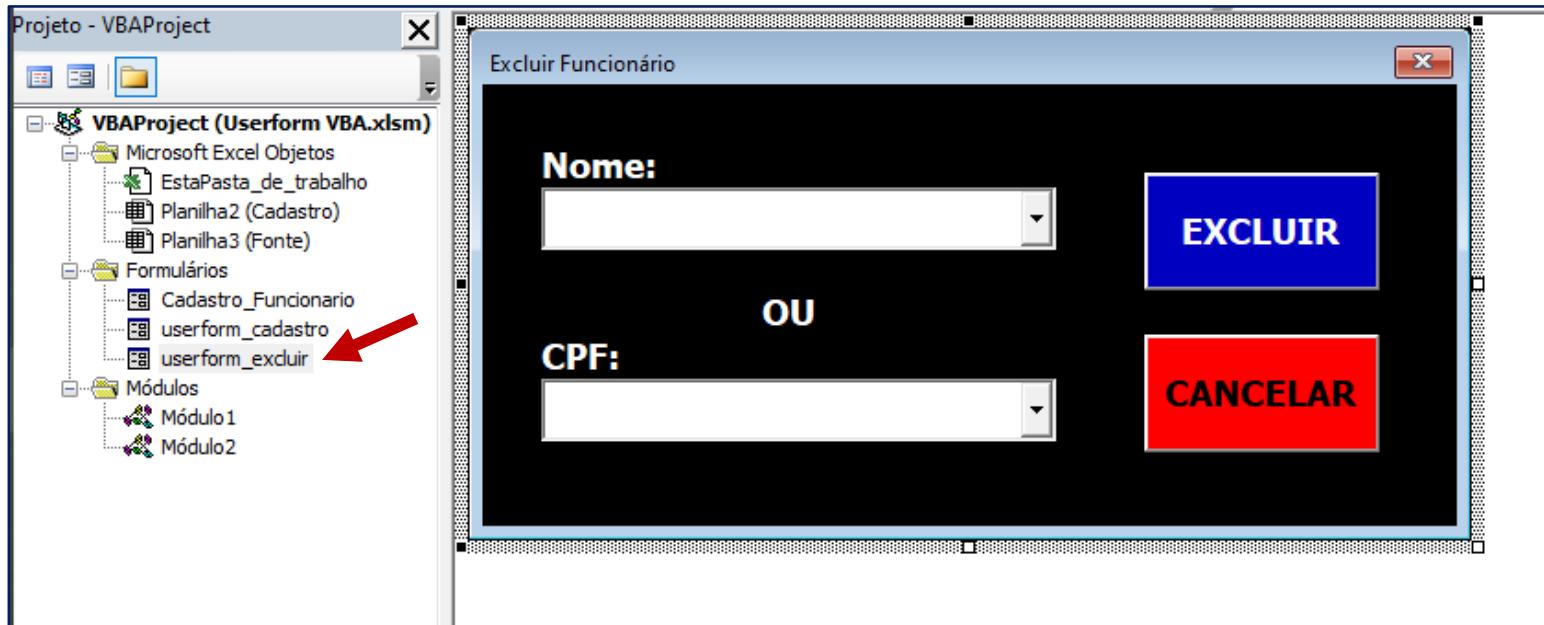
The screenshot shows the Microsoft Visual Basic Editor with a dark blue header bar. The title bar reads "BotaoComando\_Cancelar" and the tab bar has "Click" selected. The code window contains the following VBA code:

```
Private Sub BotaoComando_Cancelar_Click()
    Unload Cadastro_Funcionario
End Sub
```

A macro é simples e já a conhecemos de certa maneira. O único comando que queremos executar é o de fechar o formulário por meio do código `Unload`.

## Módulo 12 – UserForms – Excluir Funcionário

400

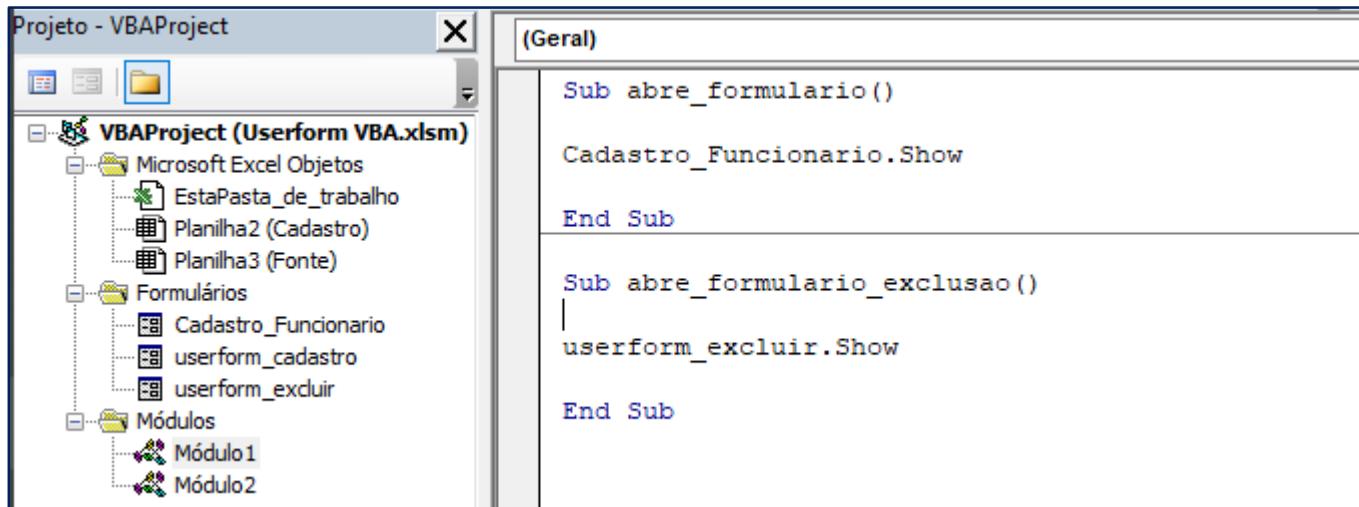


No próximo exercício vamos partir do UserForm **userfor\_excluir**. Se você quiser praticar e criar o formulário do zero, você pode seguir o mesmo passo a passo ensinado para o UserForm anterior.

A ideia deste UserForm é que, em cada uma das caixas de combinação, apareça uma lista com todos os nomes (ou CPFs) dos funcionários e que, uma vez selecionado um funcionário específico, queremos busca-lo na tabela de Cadastro e excluir a linha deste funcionário.

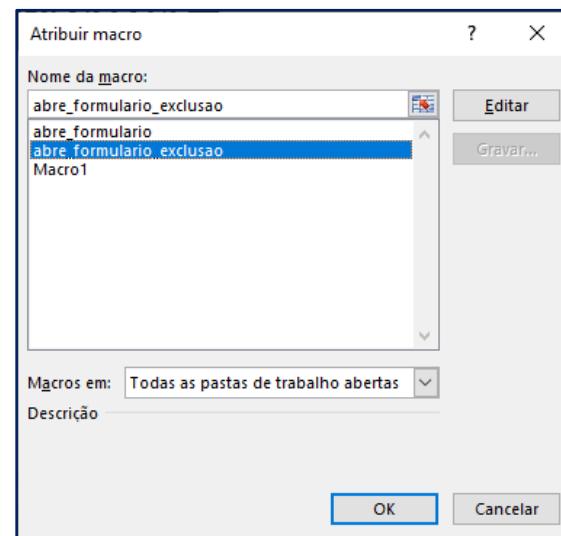
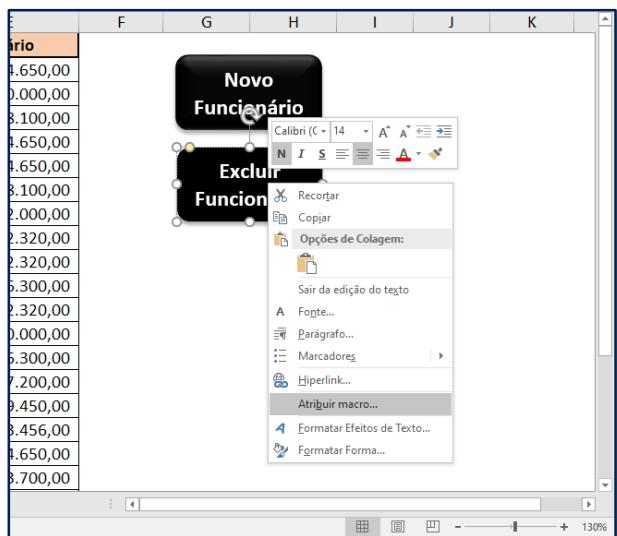
# Módulo 12 – UserForms – Excluir Funcionário

401



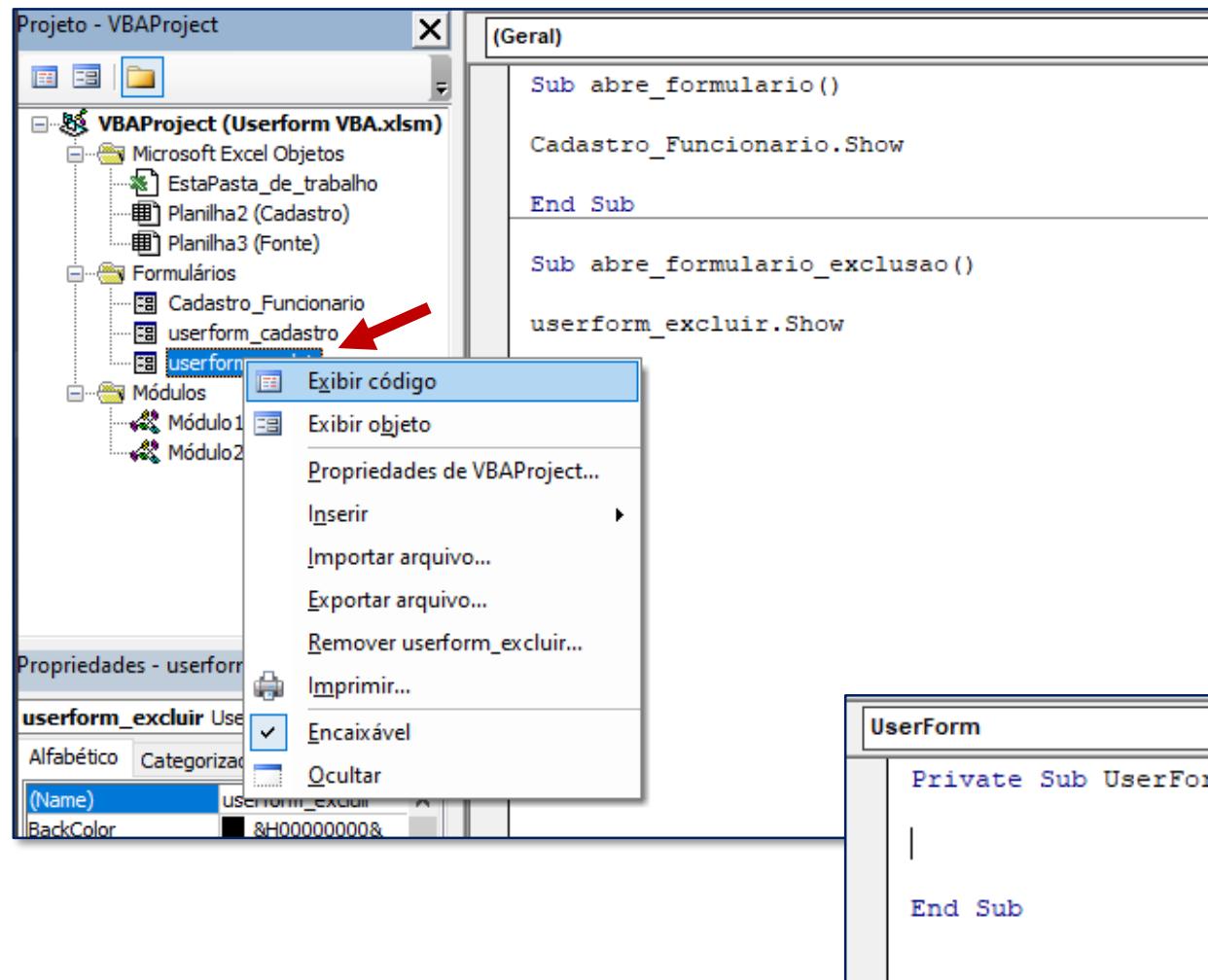
```
Sub abre_formulario()
    Cadastro_Funcionario.Show
End Sub

Sub abre_formulario_exclusao()
    userform_excluir.Show
End Sub
```



Em primeiro lugar devemos criar uma nova Sub para abrir o nosso formulário de exclusão. Fazemos isso no nosso Módulo1, onde já criamos a nossa primeira macro.

Em seguida, é só atribuir esta Sub ao botão de exclusão na nossa planilha.



Agora vamos seguir para a página de códigos do nosso **userform\_excluir**. Basta clicar com o botão direito e ir em **Exibir Código**.

O primeiro passo é criar um código que irá preencher a lista de opções das caixas de combinação, começando com a de nome. A lógica é a mesma do exercício anterior. Vamos associar este código ao evento **Initialize** do UserForm.

```
UserForm

Private Sub UserForm_Initialize()

    ult_linha = Range("A1").End(xlDown).Row

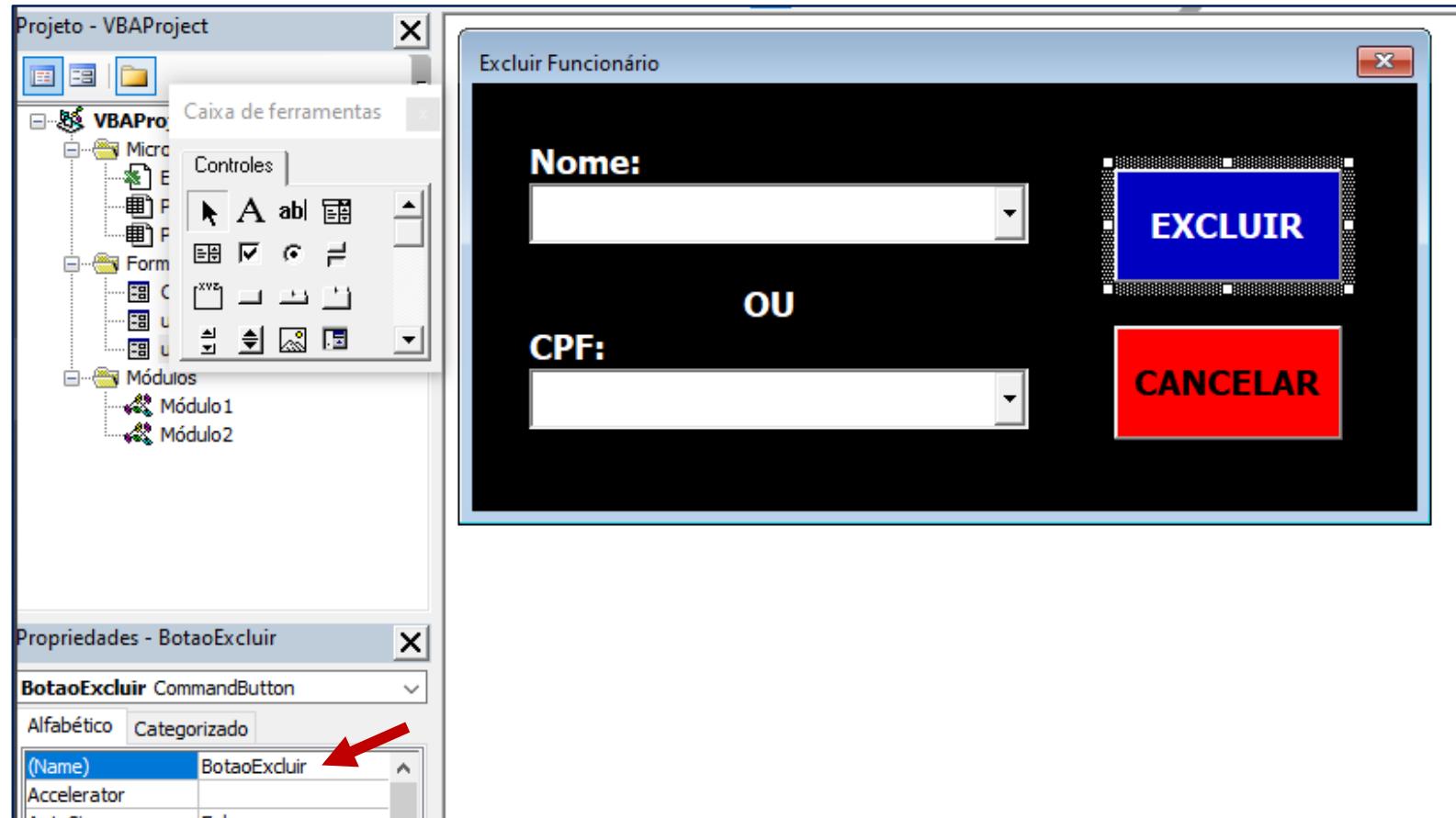
    caixacomb_nome.RowSource = "Cadastro!A2:A" & ult_linha
    caixacomb_cpf.RowSource = "Cadastro!D2:D" & ult_linha

End Sub
```



Usando a mesma lógica para o caso da caixa de combinação das áreas, descobrimos a última linha preenchida na tabela de cadastro e alteramos a propriedade RowSource das caixas de combinação **caixacomb\_nome** e **caixacomb\_cpf**.

Esse código já está funcionando e se executarmos na planilha veremos que as listas de opção já estão funcionando perfeitamente.



O próximo passo é criar uma macro que será executada ao clicarmos no botão EXCLUIR.

Lembre-se de mudar o (Name) para um nome mais intuitivo, no caso, **BotaoExcluir**, por exemplo.

Em seguida, para começar o código, clique duas vezes no botão EXCLUIR.

```
BotaoExcluir  
Private Sub BotaoExcluir_Click()  
  
If caixacomb_nome.Value <> "" Then  
  
    linha_funcionario = Cells.Find(caixacomb_nome.Value).Row  
  
    Rows(linha_funcionario).Delete  
  
ElseIf caixacomb_cpf.Value <> "" Then  
  
    linha_funcionario = Cells.Find(caixacomb_cpf.Value).Row  
  
    Rows(linha_funcionario).Delete  
  
End If  
  
Unload userform_excluir  
|  
End Sub
```

O código utilizado para o botão de Excluir está mostrado ao lado. Como temos duas caixas de combinação, temos que testar cada uma para ver qual delas possui um valor preenchido. Para isso, usamos o teste lógico If.

Feito isso, usamos o comando Cells.Find para localizar ou nome do funcionário (ou o CPF) e na verdade armazenamos o valor da linha da célula onde o comando Find encontrar o nome/CPF. Já vimos em aulas passadas que o Find é equivalente ao Localizar (CTRL + L) do Excel.

A seguir, usamos o comando Rows para excluir (Delete) a linha deste funcionário.

Por fim, fechamos o formulário com o comando Unload.

```
Private Sub BotaoExcluir_Click()
    If caixacomb_nome.Value <> "" Then
        linha_funcionario = Cells.Find(caixacomb_nome.Value).Row
        nome = caixacomb_nome.Value
        Rows(linha_funcionario).Delete
        MsgBox ("O(a) funcionário(a) " & nome & " foi excluído com sucesso!")
    ElseIf caixacomb_cpf.Value <> "" Then
        linha_funcionario = Cells.Find(caixacomb_cpf.Value).Row
        nome = Cells(linha_funcionario, 1).Value
        Rows(linha_funcionario).Delete
        MsgBox ("O(a) funcionário(a) " & nome & " foi excluído com sucesso!")
    End If
    Unload userform_excluir
End Sub
```

Podemos finalizar o código do botão de Excluir exibindo uma MsgBox informando um texto dizendo que a macro foi finalizada com sucesso e também o nome do funcionário excluído.

Para o caso em que o **nome** tenha sido preenchido na caixa de combinação, basta concatenar a variável nome com o texto dentro da MsgBox.

Porém, no caso do **cpf** ter sido preenchido, não teremos tão fácil o nome do funcionário. Mas sabemos que o nome do funcionário estará na mesma linha do CPF encontrado, só que na coluna 1 (coluna A). Então, usamos a estrutura Cells para descobrir o nome do Funcionário a partir da linha encontrada ao procurar pelo CPF.

## Módulo 12 – UserForms – Excluir Funcionário (Parte 3)

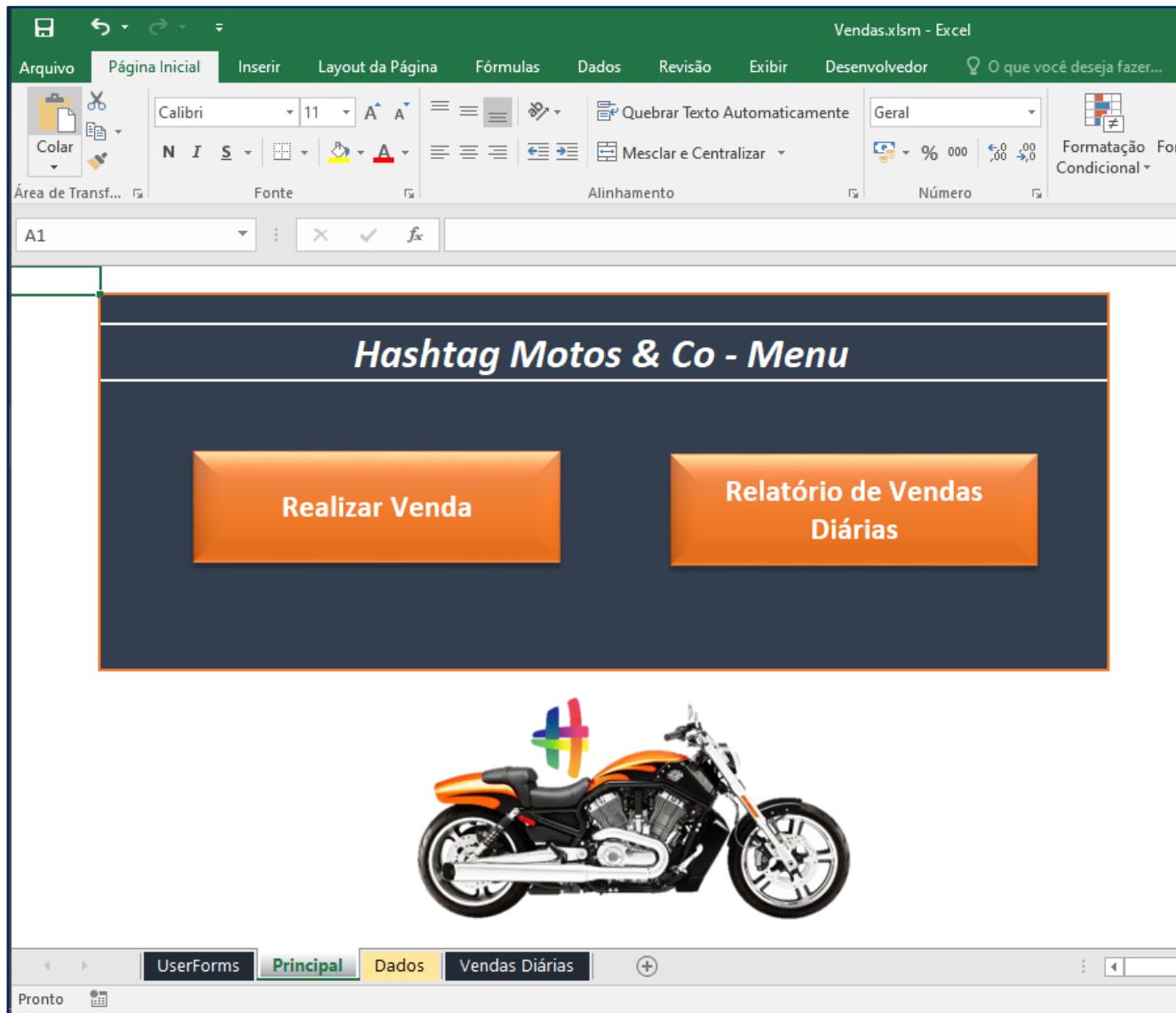
407

The screenshot shows a Microsoft Excel window with a UserForm titled "Excluir Funcionário" overlaid on a spreadsheet. The UserForm has fields for "Nome", "CPF", and "Área". It also features three buttons: "Novo Funcionário" (black), "Excluir Funcionário" (blue), and "Cancelar" (red). A message box from Microsoft Excel is visible, stating "O(a) funcionário(a) Adriane de Carvalho Gomes foi excluído com sucesso!" (The employee Adriane de Carvalho Gomes was successfully deleted). The Excel ribbon at the top includes tabs like Arquivo, Página Inicial, Inserir, Layout da Página, Fórmulas, Dados, Revisão, Exibir, Desenvolvedor, and XML.

	A	B	C	D	E	F	G	H
1	Nome	Sexo	Área	CPF	Salário			
2	Alon	Masculino	Financeiro					
3	Andressa Rotsztejn	Feminino	Marketing					
4	Beatriz Leite Júnior	Feminino	Administrador					
5	Bruna dos Santos Gomes	Feminino	Administrador					
6	Carolina Codeceira	Feminino	Marketing					
7	Daniela Muller Braga	Feminino	Logística					
8	Elvis de Freitas Derossi	Masculino	RH					
9	Gabrielle Andrade da Silva	Feminino	Administrador					
10	Gustavo de Vasconcelos	Masculino	Administrador					
11	Isabella Ronfini	Feminino	RH					
12	João	Masculino	RH	109.876.543-21	R\$ 200.000,00			
13	Juan Fernandes	Masculino	RH	107.000.473-49	R\$ 16.300,00			
14	Iúlio Fraga	Masculino	Compras	133.351.567-51	R\$ 7.200,00			

Finalmente, o nosso botão de Excluir funcionários já está funcionando perfeitamente.

Como exercício, tente criar o código necessário para o botão CANCELAR. A ideia basicamente é que, ao clicar nele, o UserForm simplesmente feche. O procedimento é exatamente o mesmo utilizado para o botão X do UserForm de cadastro de um novo funcionário (página 399).



Neste próximo exercício sobre UserForms vamos criar um sistema completo de vendas de motos.

Primeiro, vamos apresentar os arquivos que temos. O primeiro arquivo é o Vendas.xlsx. Nele, temos 4 abas. A primeira, UserForms, trata-se apenas de uma aba de explicação. Na segunda aba temos o nosso Menu, com dois botões:

- **Realizar Venda**: que será o botão que ativará o nosso UserForm.
- **Relatório de Vendas**: um botão que nos leva até a aba de Vendas Diárias.

# Módulo 12 – UserForms – Registro de Vendas (Explicação)

409

The screenshot shows the 'Dados' sheet in Excel. The table has columns labeled Moto, Preço, Modelo, Preço Modelo, Itens, and Preço Itens. The data includes:

	A	B	C	D	E	F
1	Moto	Preço	Modelo	Preço Modelo	Itens	Preço Itens
2	Honda	\$ 5.570	Básico	\$ -	Manual	\$ -
3	Yamaha	\$ 6.480	Completo	\$ 1.000	Automático	\$ 2.500
4	Kawasaki	\$ 20.890	Especial	\$ 2.000		
5	BMW	\$ 29.800	Luxo	\$ 3.000		
6	Harley-Davidson	\$ 40.500	Exclusive	\$ 5.000		
7	Suzuki	\$ 6.490				
8	Vespa	\$ 27.900				

The screenshot shows the 'Vendas Diárias' sheet in Excel. The table has columns labeled Nº da Venda, Data, Marca, Categoria, Câmbio, Desconto, Valor, Disponibilidade no Estoque, and Opcionais. The data includes:

	A	B	C	D	E	F	G	H	I	J
1	Nº da Venda	Data	Marca	Categoria	Câmbio	Desconto	Valor	Disponibilidade no Estoque	Opcionais	
2	1	13/09/2017	BMW	Completo	Automático	6%	\$31.302,00	Disponível	Banco de Couro, Capacete	
3	2	27/11/2017	Harley-Davidson	Completo	Automático	2%	\$43.120,00	Disponível	Jaqueta Preta, Capacete, Adesivos	
4										
5										
6										
7										
8										

Na terceira aba, temos a base de Dados com as informações de Preços, Modelo e Itens de cada moto. Estas informações serão utilizadas para a construção do nosso UserForm final. Por fim, na quarta aba, temos uma base com o registro de todas as vendas realizadas. Ou seja, sempre que cadastrarmos uma nova venda, nesta aba deverá ser adicionada uma nova linha, com as informações desta venda.

## Módulo 12 – UserForms – Registro de Vendas (Explicação)

410

	A	B	C	D	E	F
1	Marca	Categoria	Quantidade em Estoque			
2	Honda	Básico	1			
3	Honda	Completo	1			
4	Honda	Especial	0			
5	Honda	Luxo	2			
6	Honda	Exclusive	2			
7	Yamaha	Básico	1			
8	Yamaha	Completo	0			
9	Yamaha	Especial	5			
10	Yamaha	Luxo	1			
11	Yamaha	Exclusive	0			

Além do arquivo anterior, também temos outro arquivo para controle de Estoque.

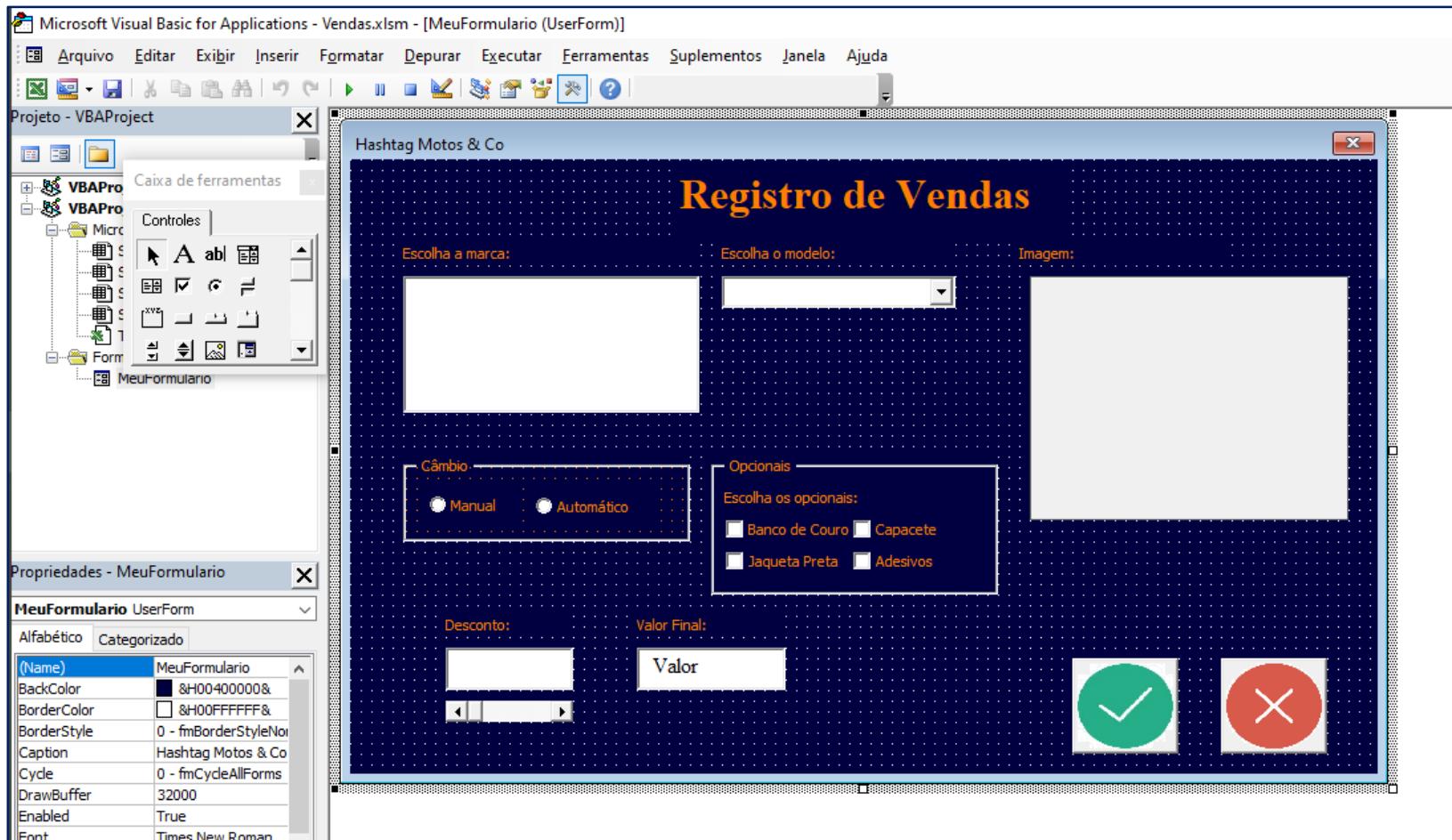
A ideia é que este arquivo seja atualizado toda vez que uma nova venda for realizada no arquivo de Vendas mostrado anteriormente.



O UserForm final que queremos construir é mostrado na imagem ao lado. Após preencher todos os campos queremos cadastrar automaticamente estas informações da venda na nossa aba “Dados”.

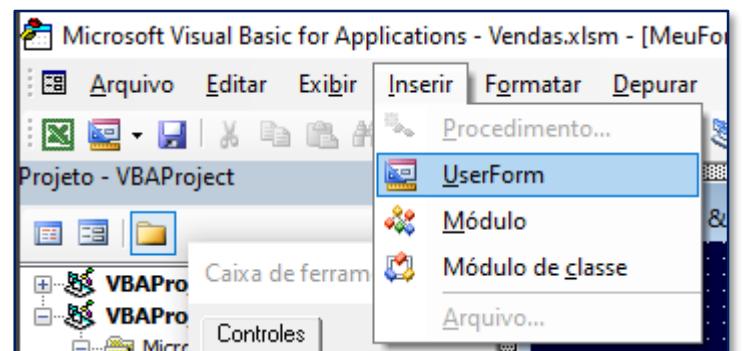
# Módulo 12 – UserForms – Construindo o Layout do Formulário

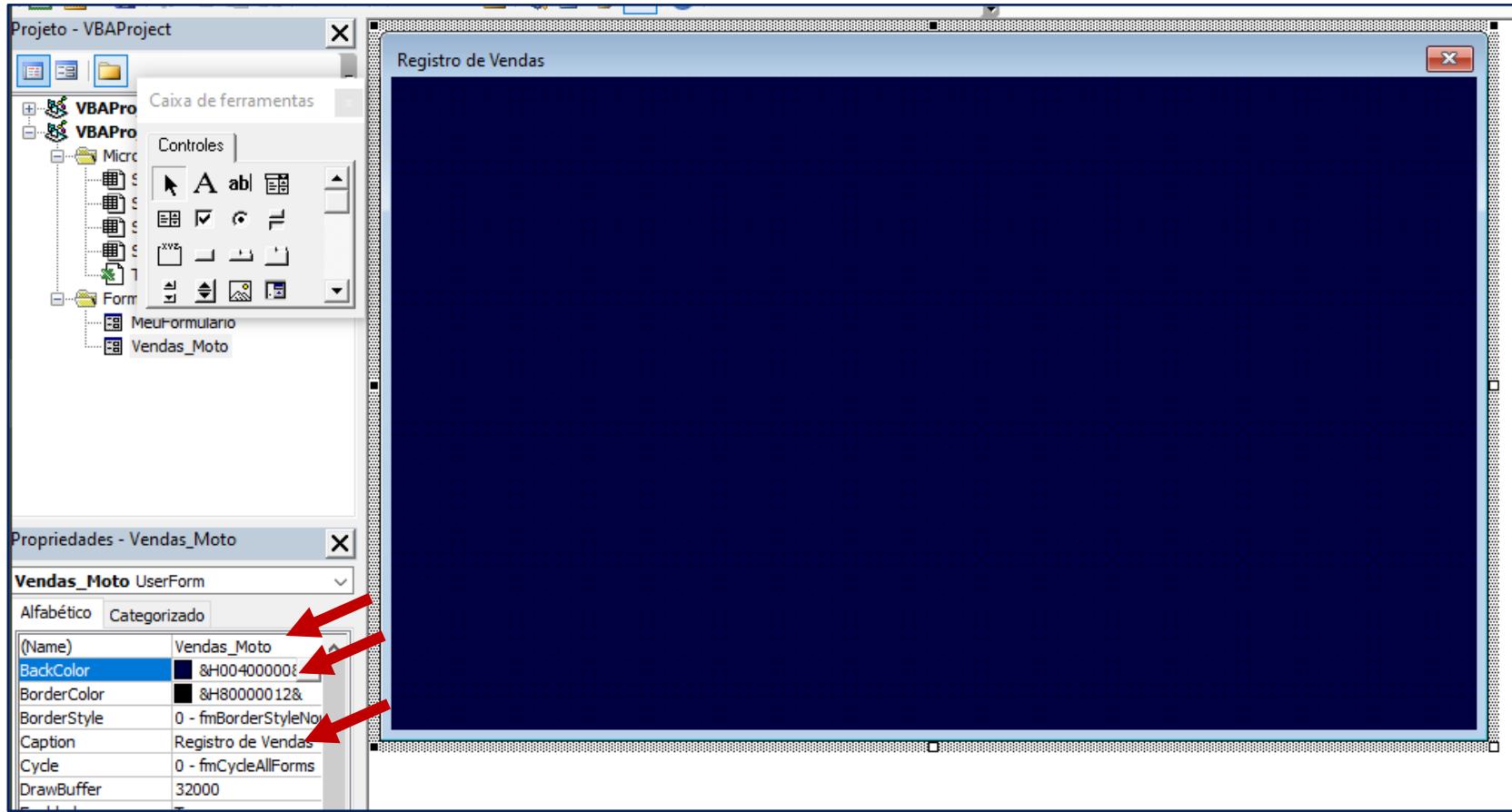
412



Para dar inicio, abra o ambiente VBA. Se você reparar, já temos um modelo de UserForm pronto no arquivo disponibilizado, e a ideia é que a gente crie exatamente este formulário do zero. Boa parte dos botões já são conhecidos, então conseguiremos construí-lo sem muita dificuldade.

Vamos começar inserindo um novo UserForm.





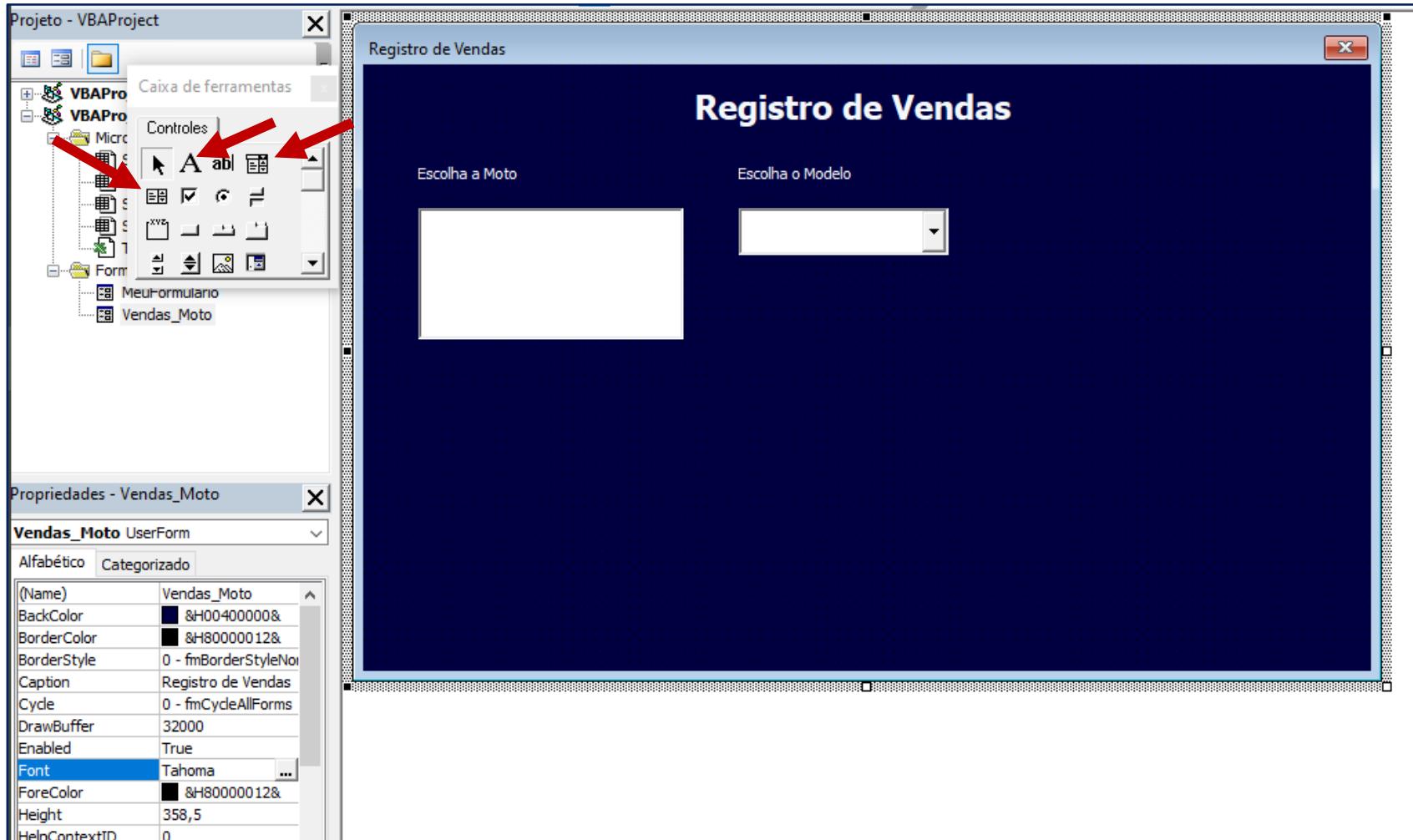
## ATENÇÃO !

Se a Caixa de Ferramentas acima não estiver aparecendo para você, clique em no ícone com um martelo:



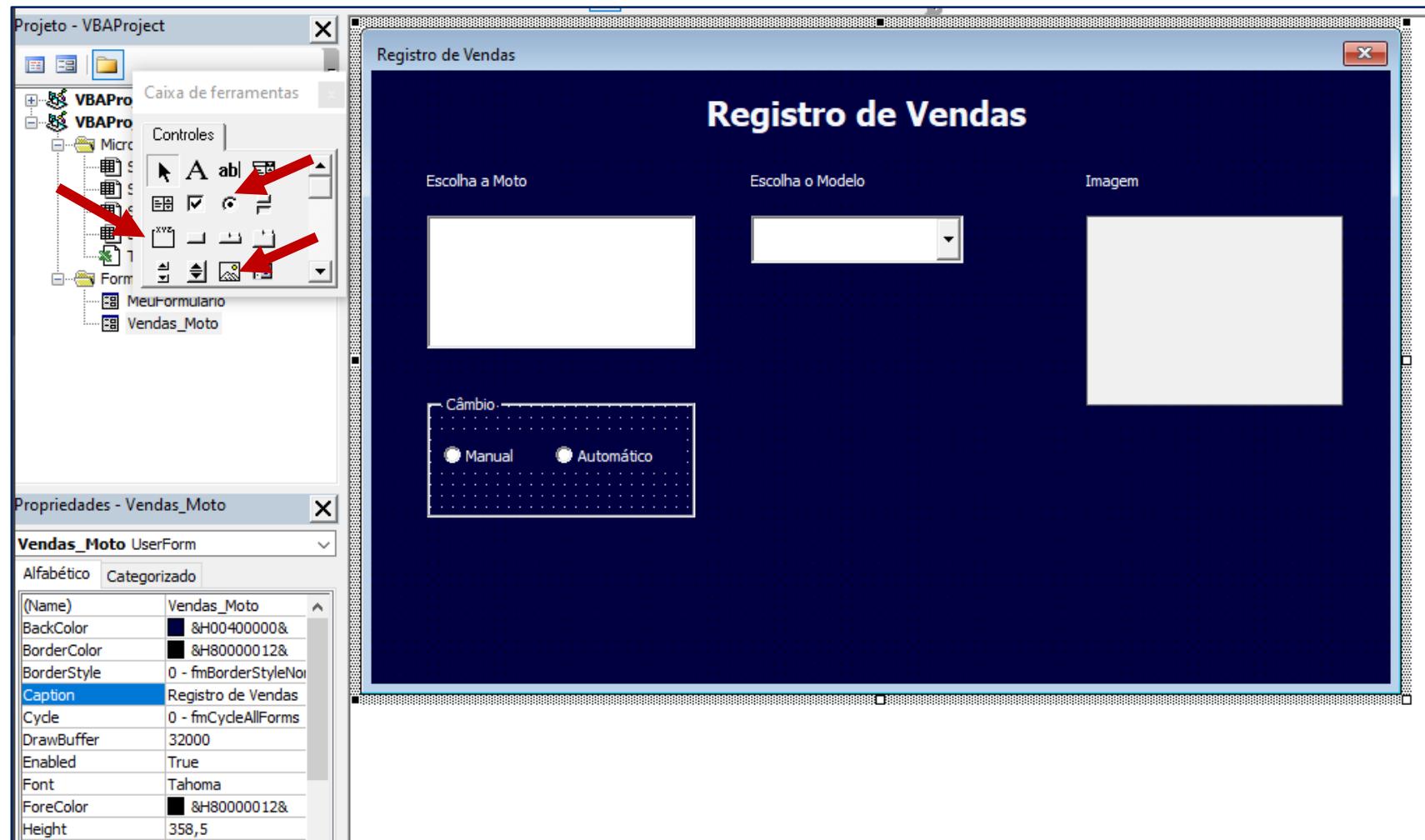
Após inserir o novo formulário, siga os passos descritos a partir de agora:

1. Aumentar o tamanho do UserForm.
2. Mudar a propriedade **(Name)** para: **Vendas\_Moto**.
3. Mudar a propriedade **BackColor** para a cor que quiser.
4. Mudar o **Caption** para: **Registro de Vendas**.



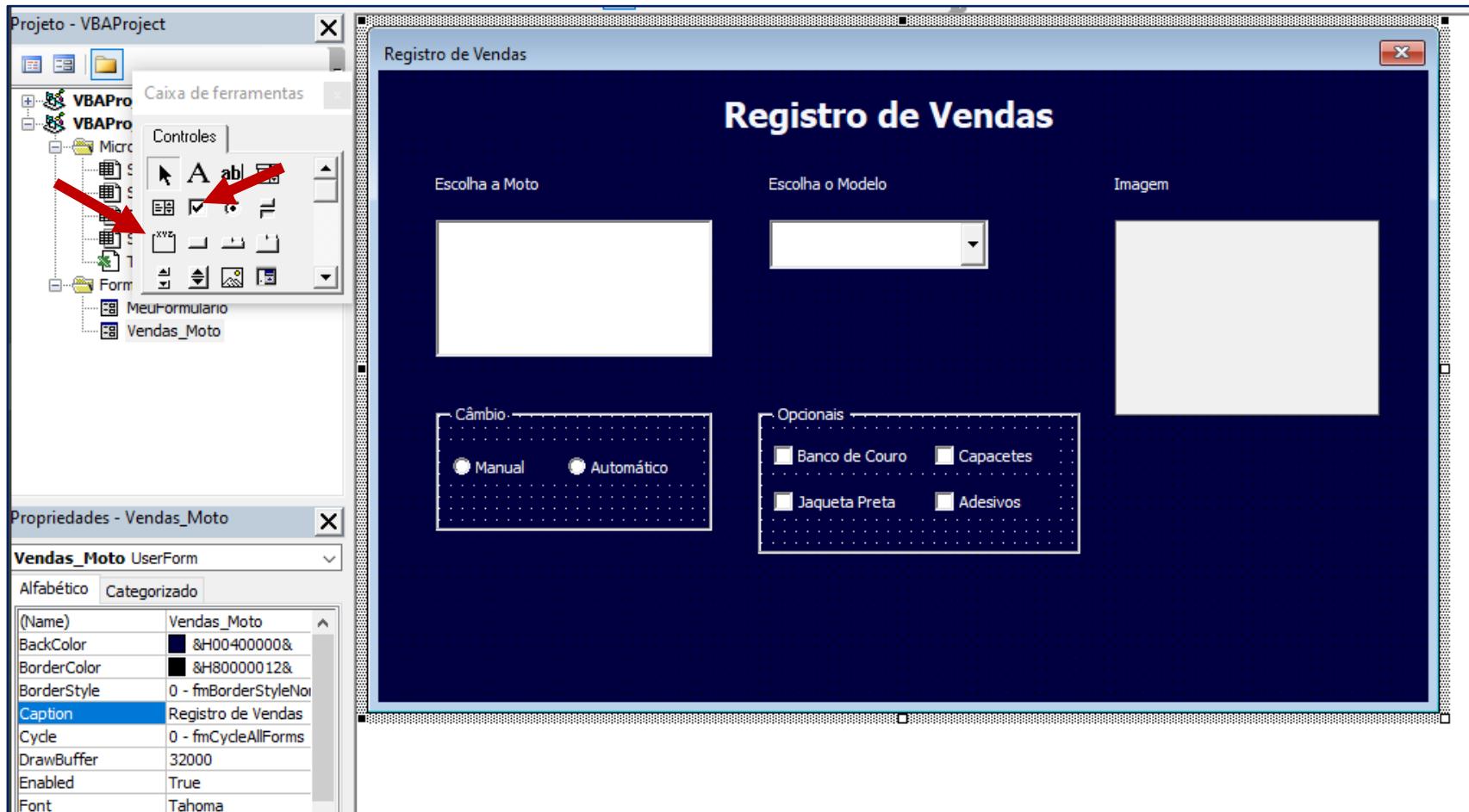
Em seguida:

5. Adicione os objetos Rótulo, Caixa de Combinação e Caixa de Listagem.
6. Posicione-os adequadamente no UserForm.
7. Para mudar a cor da fonte do Rótulo, mude a propriedade **ForeColor** para uma cor mais adequada.
8. Para aumentar o tamanho do texto do Rótulo do título, mude a propriedade **Font**. Lá terão outras opções de formatação de texto.

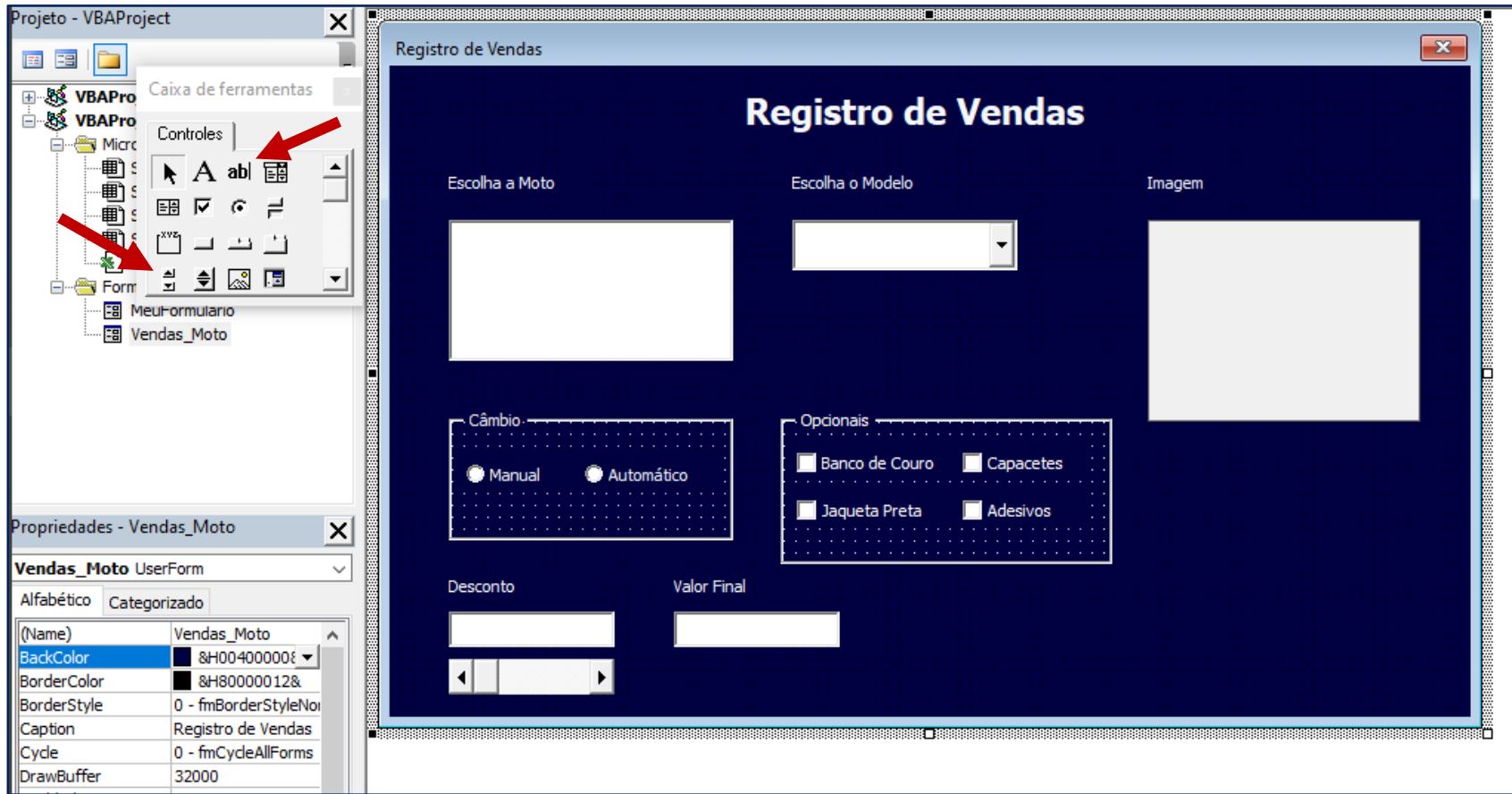


Em seguida:

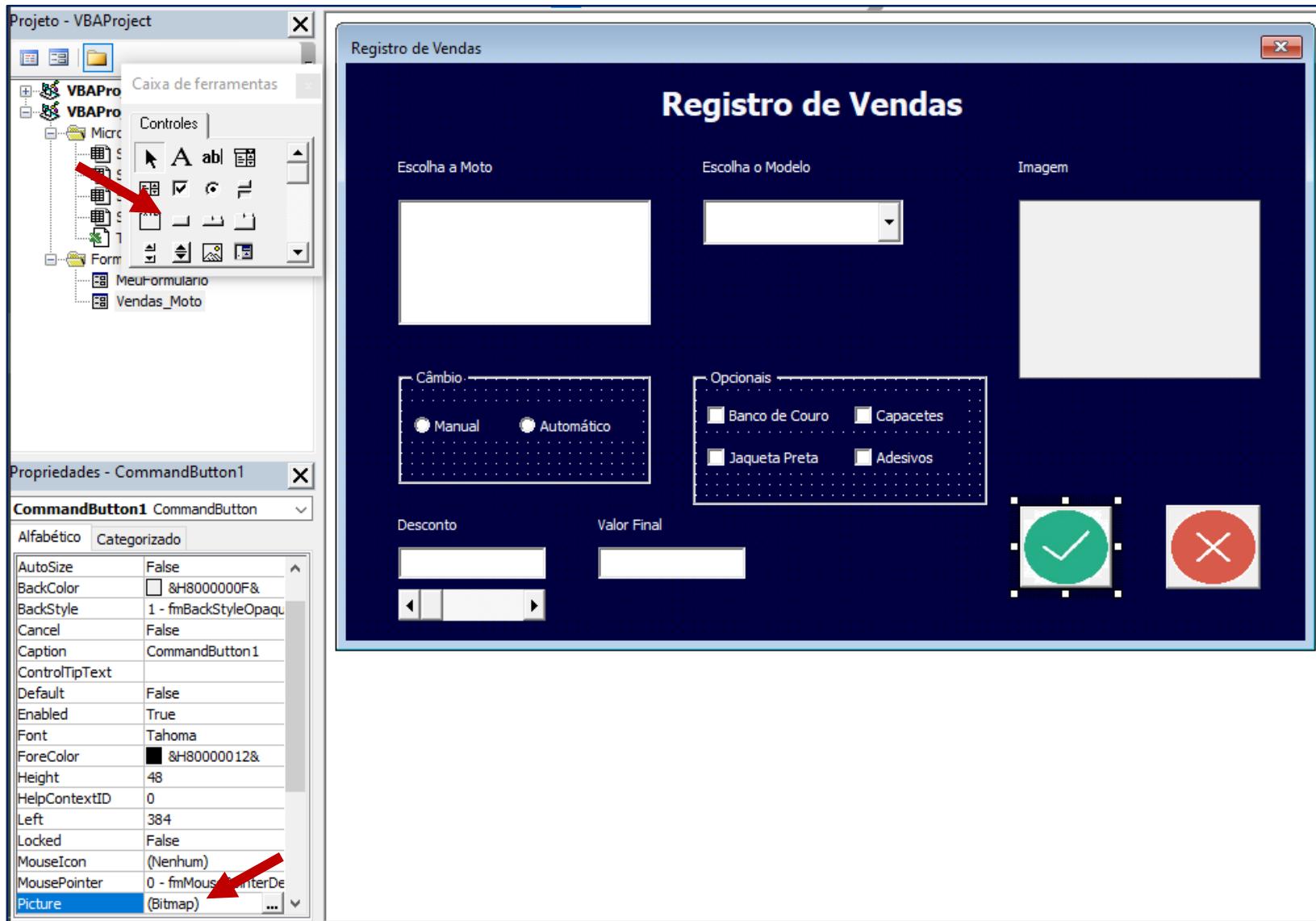
9. Adicione os objetos Imagem, Quadro e por cima deste, dois botões de opção.
10. Posicione-os adequadamente no UserForm.
11. Para mudar a cor da fonte do Quadro e dos botões de opção, mude a propriedade **ForeColor** para uma cor mais adequada.
12. Mude a propriedade **Caption** do Quadro para **Câmbio**.



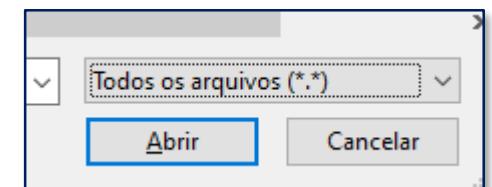
13. Adicione os objetos Quadro e Caixa de Seleção.
14. Posicione-os adequadamente no UserForm.
15. Para mudar a cor da fonte do Quadro e das caixas de seleção, mude a propriedade **ForeColor** para uma cor mais adequada.
16. Mude a propriedade **Caption** do Quadro para **Opcionais**.

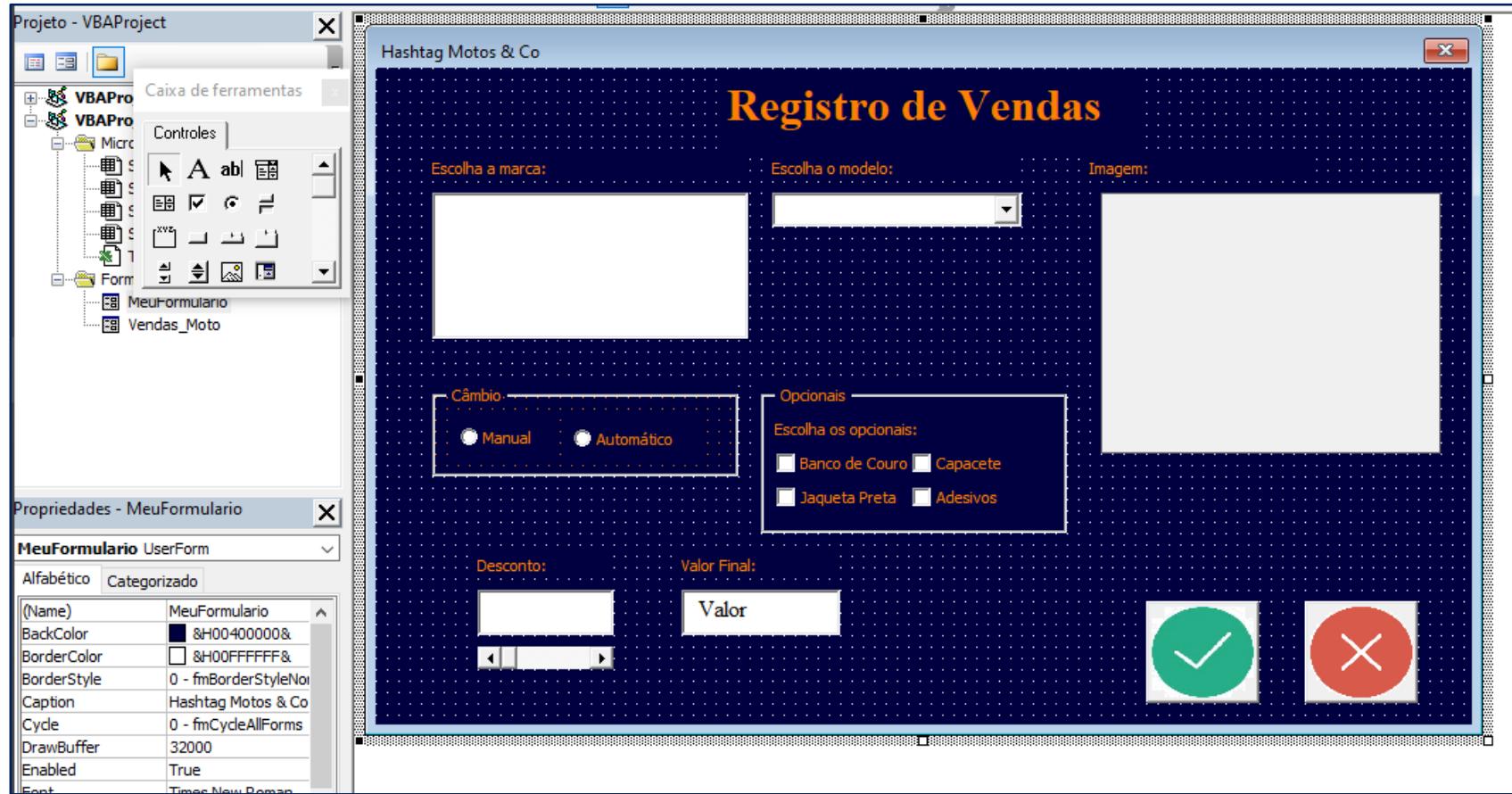


17. Adicione os objetos Barra de Rolagem e duas caixas de texto.
18. Posicione-os adequadamente no UserForm.
19. Para mudar a cor da fonte das caixas de texto, mude a propriedade **ForeColor** para uma cor mais adequada.
20. Clique na Barra de Rolagem e troque o valor do Max para 30, pois a ideia é que esta barra de rolagem varie de um mínimo (0) até um máximo (30) pois é por meio dessa barra de rolagem que vamos alterar o valor do desconto na venda.



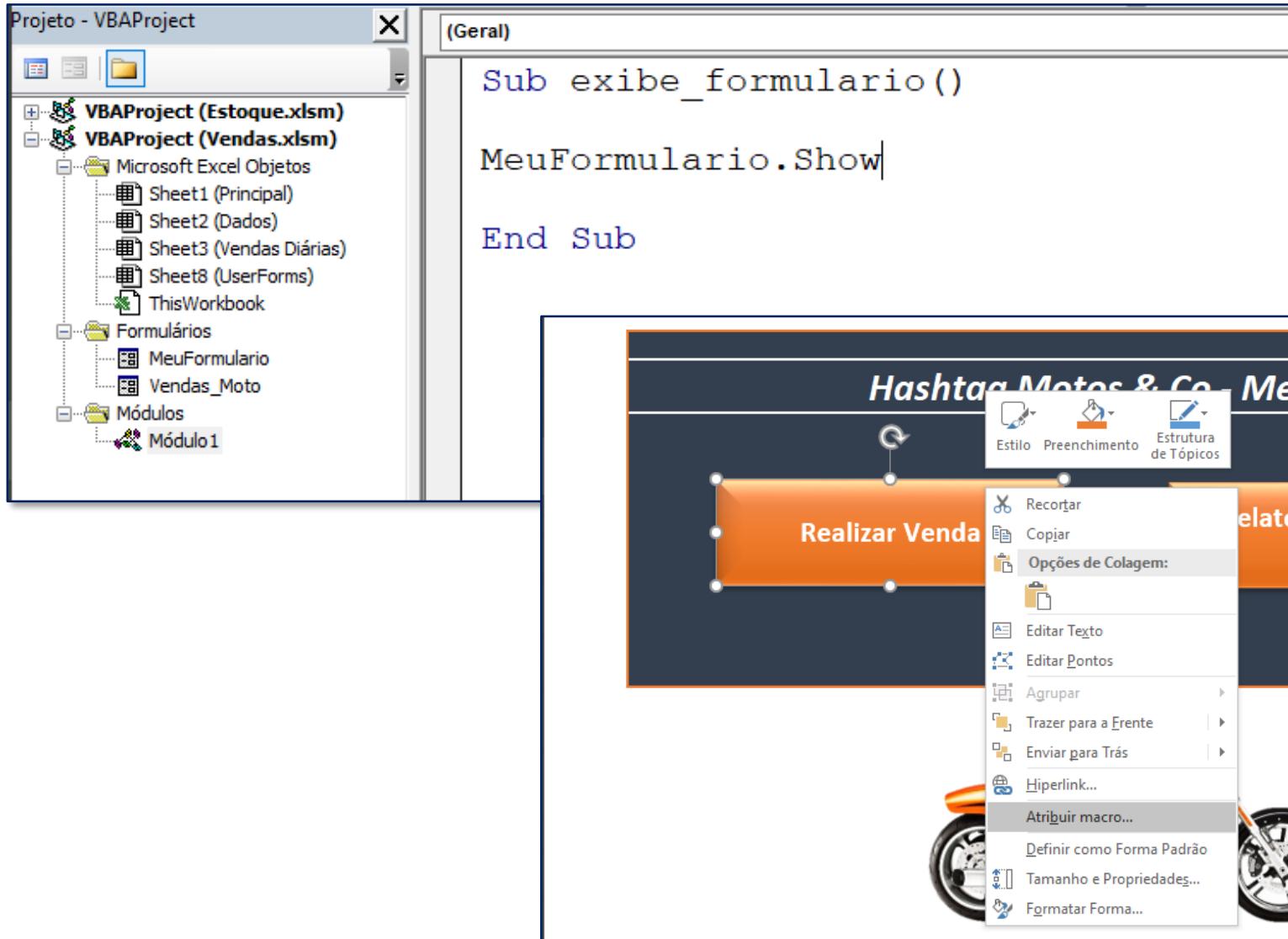
21. Adicione dois objetos de botão de comando, uma para o nosso OK e outro para a opção de Cancelar.
22. Para inserir uma imagem nestes botões, altere a propriedade **Picture** de cada um deles para selecionar uma imagem da pasta.
23. Lembrando que você deve marcar a opção **Todos os Arquivos** para que as imagens apareçam na pasta.





Com as configurações anteriores nós conseguimos finalizar o nosso UserForm.

Apenas para facilitar, vamos trabalhar em cima do UserForm já pronto do arquivo, para garantir que você e eu vamos iniciar os códigos exatamente do mesmo ponto. Mas se você quiser seguir com o seu, sem problemas. Só lembre de alterar a propriedade (Name) de cada um dos objetos para dar nomes mais intuitivos a eles.



Lembre-se que sempre queremos abrir o nosso formulário clicando em um botão da planilha. Para fazermos isso, criamos um novo Módulo, na guia Inserir, e nele escrevemos uma macro para mostrar o nosso UserForm, como já fizemos anteriormente.

O código é mostrado ao lado.

Em seguida, você pode voltar na planilha, clicar no botão de Vendas com o botão direito e atribuir esta macro a ele.

## Módulo 12 – UserForms – Inicializando o Formulário

421

The screenshot shows a Microsoft Excel window with the title bar 'Vendas.xlsm - Excel'. The ribbon tabs include Arquivo, Página Inicial, Inserir, Layout da Página, Fórmulas, Dados, Revisão, Exibir, Desenvolvedor, and Entrar. The 'Página Inicial' tab is selected. A UserForm titled 'Hashtag Motos & Co' is displayed in the foreground. The UserForm has three input fields: 'Escolha a marca:' (Choose brand:), 'Escolha o modelo:' (Choose model:), and 'Imagem:' (Image:). Below these fields is a table with columns labeled A through F. Column A is 'Moto', column B is 'Preço', column C is 'Modelo', column D is 'Preço Modelo', column E is 'Itens', and column F is 'Preço Itens'. The table contains the following data:

	A	B	C	D	E	F
1	Moto	Preço	Modelo	Preço Modelo	Itens	Preço Itens
2	Honda	\$ 5.570	Básico	\$ -	Manual	\$ -
3	Yamaha	\$ 6.480	Completo	\$ 1.000	Automático	\$ 2.500
4	Kawasaki	\$ 20.890	Especial	\$ 2.000		
5	BMW	\$ 29.800	Luxo	\$ 3.000		
6	Harley-Davidson	\$ 40.500	Exclusive	\$ 5.000		
7	Suzuki	\$ 6.490				
8	Vespa	\$ 27.900				
9						

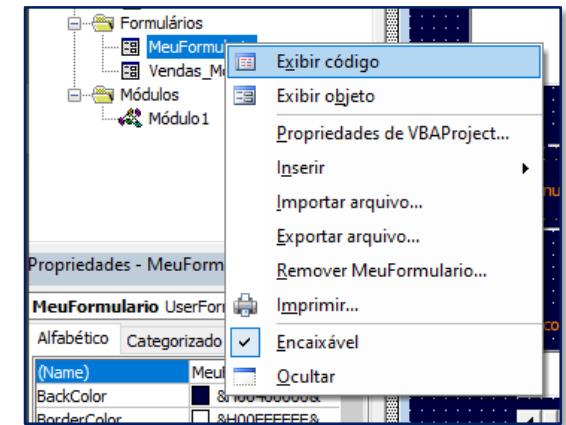
Ao clicar no botão, o formulário abrirá automaticamente.

A partir daqui, precisamos começar a preencher os nossos objetos. Para começar, queremos preencher com as opções de Marca e Modelo os objetos Caixa de Listagem e Caixa de Combinação, respectivamente, de acordo com as informações na aba "Dados".

Para preencher as opções nestes dois botões, vamos seguir a mesma lógica dos exercícios anteriores. Queremos que estes dois botões se atualizem com as opções automaticamente sempre que a gente abrir o formulário.

Para começar criando o código, primeiro vamos clicar com o botão direito em nosso formulário (como mostrado ao lado) e depois clicar em Exibir Código.

Na nossa janela de programação, vamos escolher o evento `UserForm_Initialize`, e qualquer código dentro desta Sub será executado assim que alguém inicializar (abrir) o UserForm.

A screenshot of the Microsoft Excel VBA Editor. The left pane shows the 'Projeto - VBAProject' browser with several workbooks and their contents listed. The right pane shows the code for the 'UserForm' object. The code consists of a single sub-procedure:

```
Private Sub UserForm_Initialize()  
    End Sub
```

```
UserForm
Initialize

Private Sub UserForm_Initialize()

    'descobre a última linha preenchida na coluna de Moto da aba "Dados"
    ult_linha_marca = Sheets("Dados").Range("A1").End(xlDown).Row

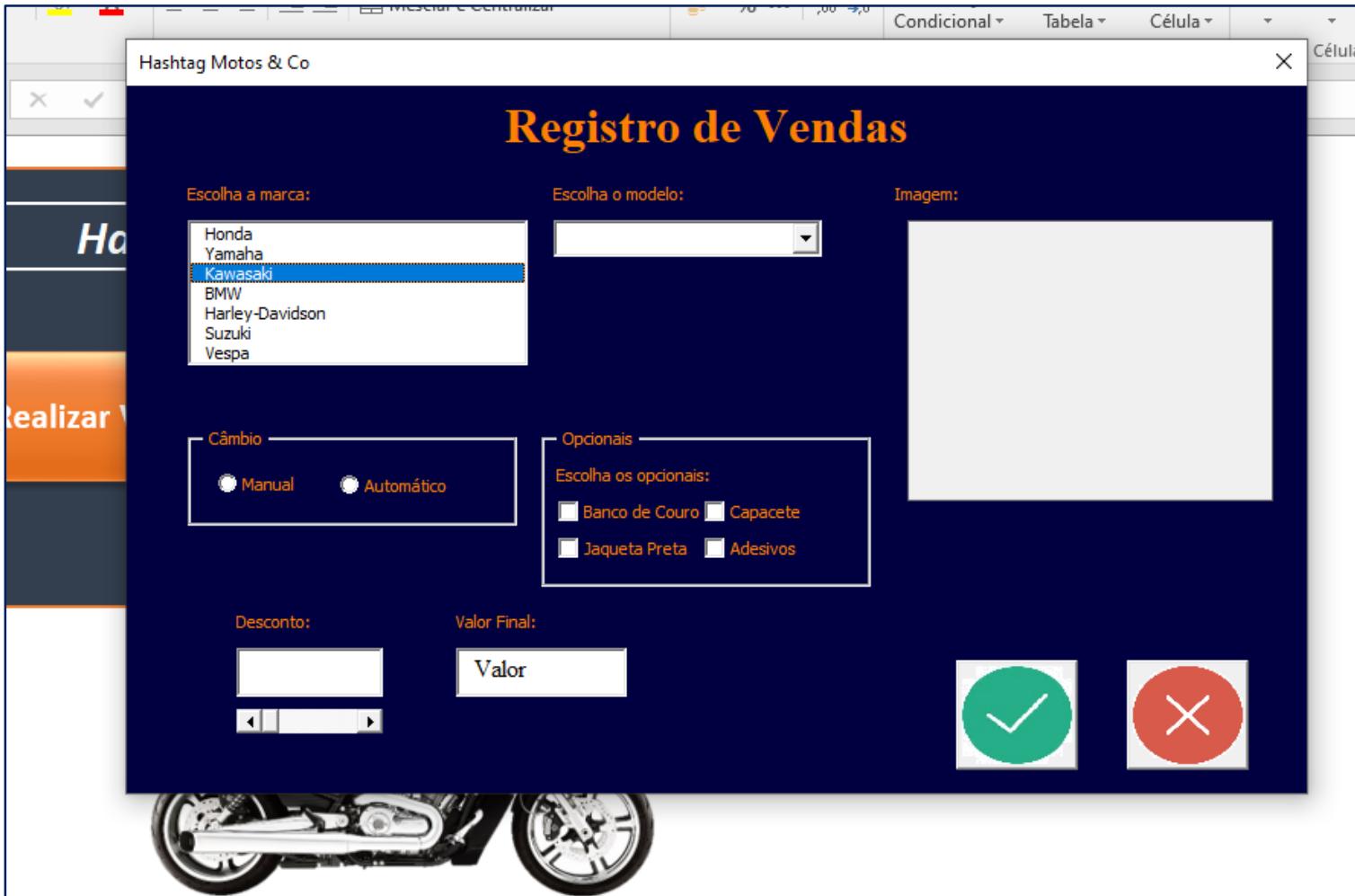
    'utiliza a propriedade RowSource para passar a lista de Motos para a Caixa de Listagem
    ListaMarca.RowSource = "Dados!A2:A" & ult_linha_marca

    'descobre a última linha preenchida na coluna de Modelo da aba "Dados"
    ult_linha_modelo = Sheets("Dados").Range("C1").End(xlDown).Row

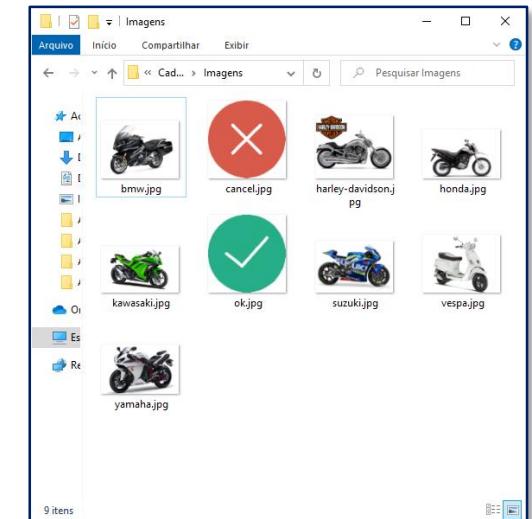
    'utiliza a propriedade RowSource para passar a lista de Modelos para a Caixa de Combinação
    CaixaModelo.RowSource = "Dados!C2:C" & ult_linha_modelo

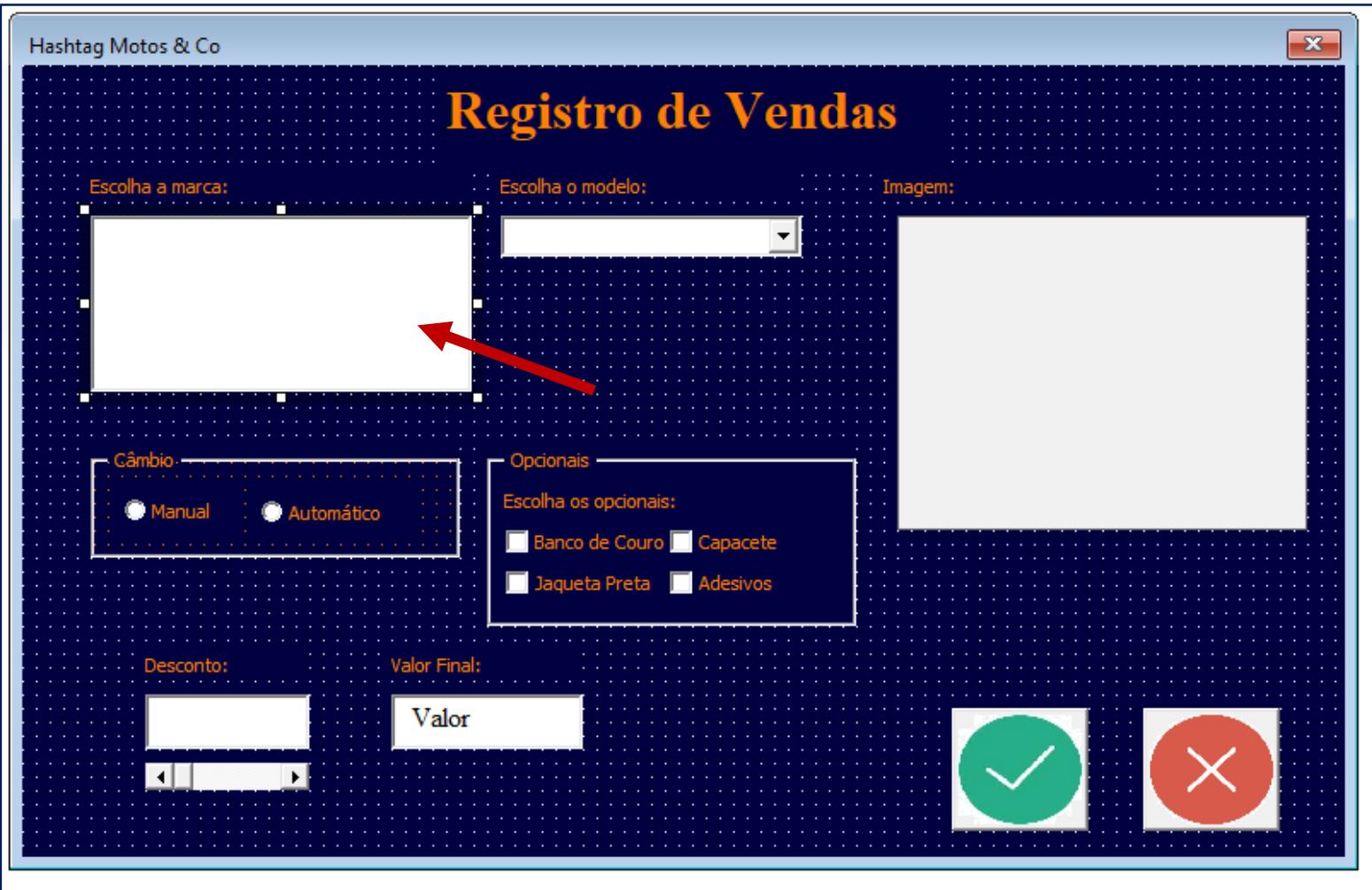
End Sub
```

O código ao lado será utilizado para preencher as opções de ambos os botões, alterando basicamente a propriedade **RowSource** destes botões, como já vimos anteriormente.



O próximo passo no exercício é modificar a Imagem da foto de acordo com a marca selecionada na Caixa de Listagem. Na pasta do exercício temos uma série de imagens que queremos adicionar ao UserForm de forma automática, conforme o usuário muda a marca selecionada.





O código que queremos criar (alterar a imagem automaticamente) deve estar associado ao evento de clicar na caixa de listagem, portanto, o nosso código deverá ser criado na Caixa de Listagem.

Para isso, volte ao UserForm e dê um duplo clique na Caixa de Listagem, indicada ao lado.

Para carregar a imagem de cada moto, devemos passar o caminho completo do nosso computador onde as imagens estão. Como isso deve ser feito de forma dinâmica (a cada moto selecionada a imagem deve se atualizar) então primeiro criamos uma variável para armazenar o nome da moto, de acordo com o valor selecionado na caixa de listagem. Isto será útil porque os nomes dos arquivos também são os nomes das opções na caixa de listagem, então podemos usá-lo ao nosso favor para tornar dinâmico o caminho inteiro da imagem.

O caminho completo da pasta podemos descobrir usando o comando ThisWorkbook.Path, que já vimos anteriormente no curso. Como as imagens estão em uma pasta dentro desta pasta, adicionamos também o nome desta pasta. Por fim, concatenamos o texto com o nome do arquivo e a sua extensão (.jpg). Todo o caminho deve ser o argumento da função LoadPicture para que o código funcione.

```
ListaMarca
```

```
Private Sub ListaMarca_Click()
    nome_moto = ListaMarca.Value
    ImagemMoto.Picture = LoadPicture(ThisWorkbook.Path & "\Imagens\" & nome_moto & ".jpg")
End Sub
```



O próximo passo é fazer com que o valor da caixa de texto abaixo altere de acordo com a barra de rolagem.

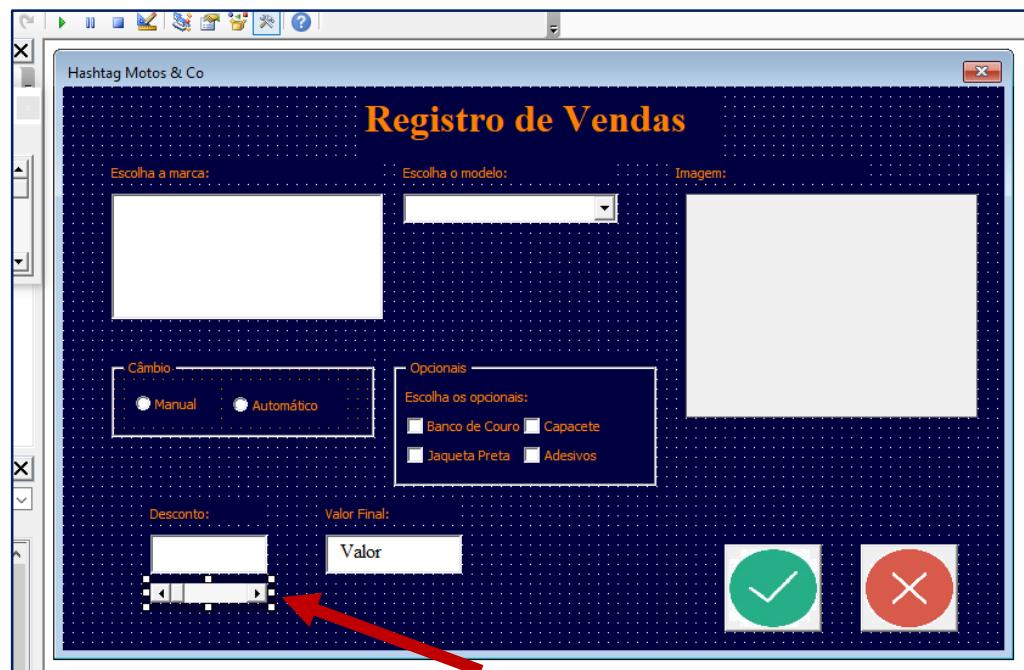
Isso quer dizer que o valor da caixa de texto do desconto deve variar sempre que a gente mudar o valor na barra de rolagem.

Como este evento está associado ao clique na barra de rolagem, podemos voltar para o ambiente VBA e dessa vez dar um duplo clique na barra de rolagem.

O evento que iremos mudar pode ser o Click ou Change, tanto faz.

E o código final é mostrado abaixo.

Concatenamos o sinal de porcentagem ao final para que o valor seja mostrado em uma formatação de porcentagem.



```
BarraDesconto
Private Sub BarraDesconto_Change()
    TextoDesconto.Value = BarraDesconto.Value & "%"
End Sub
```

Hashtag Motos & Co

## Registro de Vendas

Escolha a marca:

- Honda
- Yamaha**
- Kawasaki
- BMW
- Harley-Davidson
- Suzuki
- Vespa

Escolha o modelo:

Básico

Imagen:

Câmbio

Manual     Automático

Opcionais

Escolha os opcionais:

Banco de Couro     Capacete  
 Jaqueta Preta     Adesivos

Desconto:

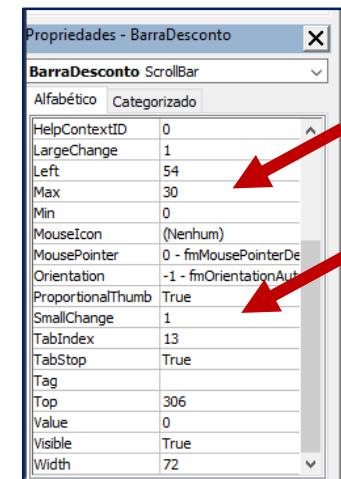
10%

Valor Final:

Valor

O desconto deve variar de 0 a 30. Caso você não tenha feito essa configuração, volte até o passo 20 da página 417, onde configuramos a propriedade Max do botão. Ou então você pode se guiar pela imagem abaixo.

Lá você também pode mudar a propriedade SmallChange para configurar de quanto em quanto o valor deve mudar. O padrão é de 1 em 1.



Hashtag Motos & Co

## Registro de Vendas

Escolha a marca:

Honda  
 Yamaha  
 Kawasaki  
 BMW  
**Harley-Davidson**  
 Suzuki  
 Vespa

Escolha o modelo:

Completo

Imagen:



Câmbio:

Manual     Automático

Opcionais:

Escolha os opcionais:

Banco de Couro     Capacete  
 Jaqueta Preta     Adesivos

Desconto:

20%

Valor Final:

Valor



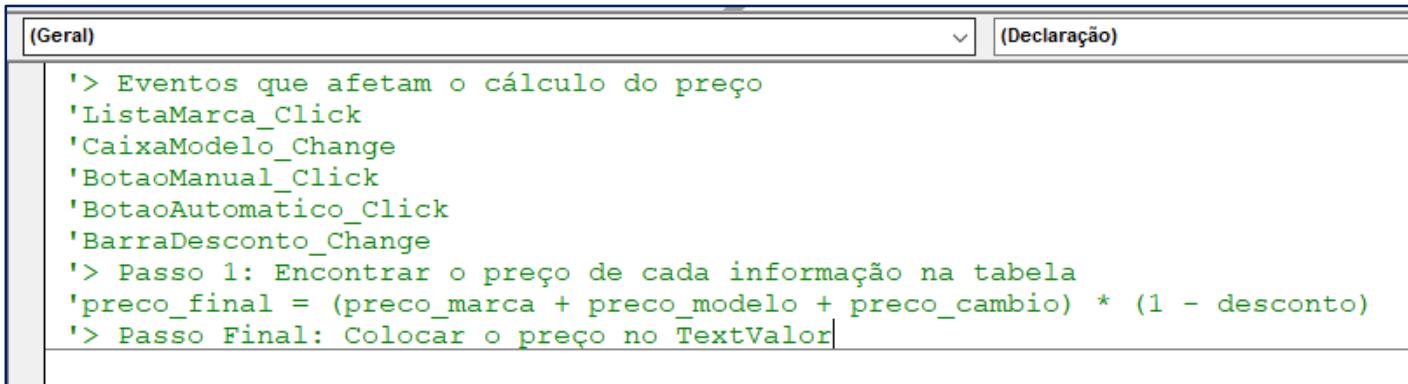

O que queremos fazer agora é calcular o preço final da moto conforme selecionamos:

- Marca
- Modelo
- Câmbio
- Desconto

Os opcionais são bônus e não afetarão o valor final. Os preços de todos os demais serão buscados na aba Dados e devem ser somados na caixa de texto correspondente ao Valor Final.

	A	B	C	D	E	F
1	Moto	Preço	Modelo	Preço Modelo	Itens	Preço Itens
2	Honda	\$ 5.570	Básico	\$ -	Manual	\$ -
3	Yamaha	\$ 6.480	Completo	\$ 1.000	Automático	\$ 2.500
4	Kawasaki	\$ 20.890	Especial	\$ 2.000		
5	BMW	\$ 29.800	Luxo	\$ 3.000		
6	Harley-Davidson	\$ 40.500	Exclusive	\$ 5.000		
7	Suzuki	\$ 6.490				
8	Vespa	\$ 27.900				
9						

« »    
 UserForms Principal **Dados** Vendas Diárias +



```
(Geral) (Declaração)
'> Eventos que afetam o cálculo do preço
'ListaMarca_Click
'CaixaModelo_Change
'BotaoManual_Click
'BotaoAutomatico_Click
'BarraDesconto_Change
'> Passo 1: Encontrar o preço de cada informação na tabela
'preco_final = (preco_marca + preco_modelo + preco_cambio) * (1 - desconto)
'> Passo Final: Colocar o preço no TextValor|
```

Apenas por questão de organização, vamos listar os pontos importantes do que precisamos fazer para encontrar o preço final.

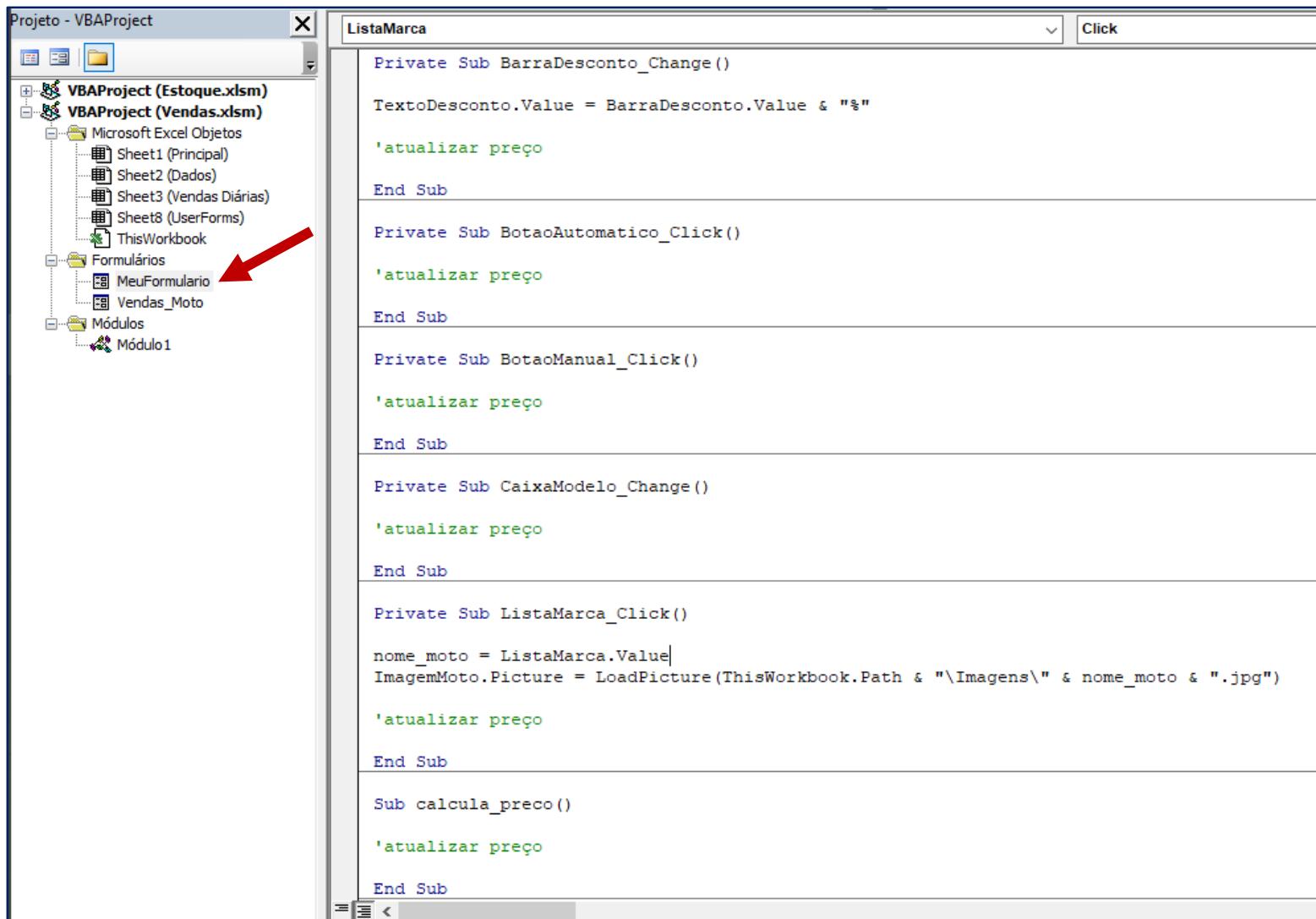
Exiba o código do formulário (caso você não esteja na janela de programação do UserForm) e escreva os comentários ao lado.

Devemos entender que existem 5 eventos diferentes que irão afetar o valor final, e sempre que qualquer um deles for executado, o valor final deverá se atualizar.

Portanto, em cada um desses eventos, teremos que criar uma macro que calcule automaticamente o preço final. Isso significa que precisaremos de pelo menos 5 eventos, conforme especificado nos comentários ao lado.

## Módulo 12 – UserForms – Calculando o Preço Final da Moto (Parte 2)

432



The screenshot shows the VBA Project Explorer on the left and the code editor on the right. The project contains two main files: 'VBAProject (Estoque.xlsm)' and 'VBAProject (Vendas.xlsm)'. Under 'Vendas.xlsm', there are several objects: Microsoft Excel Objetos (Sheet1, Sheet2, Sheet3, Sheet8), ThisWorkbook, Formulários (MeuFormulario, Vendas\_Moto), and Módulos (Módulo1). A red arrow points to the 'MeuFormulario' object in the 'Formulários' folder. The code editor displays five macros:

```
Private Sub BarraDesconto_Change()
    TextoDesconto.Value = BarraDesconto.Value & "%"
    'atualizar preço
End Sub

Private Sub BotaoAutomatico_Click()
    'atualizar preço
End Sub

Private Sub BotaoManual_Click()
    'atualizar preço
End Sub

Private Sub CaixaModelo_Change()
    'atualizar preço
End Sub

Private Sub ListaMarca_Click()
    nome_moto = ListaMarca.Value
    ImagemMoto.Picture = LoadPicture(ThisWorkbook.Path & "\Imagens\" & nome_moto & ".jpg")
    'atualizar preço
End Sub

Sub calcula_preco()
    'atualizar preço
End Sub
```

Assim, teremos as 5 macros ao lado, uma para cada evento. Repare que em cada uma dessas macros existe uma linha de comentário que diz que ali deve entrar um código para atualizar o preço final.

Uma sexta macro, ao final, chamada **calcula\_preco**, será responsável por fazer esses cálculos e simplesmente o que faremos é, em cada uma das outras 5 macros, vamos chamar a macro **calcula\_preco**. Isso será muito melhor do que repetir o código 5 vezes, você vai ver.

Repare que todos os códigos estão sendo feitos dentro do **MeuFormulario**.

```
Sub calcula_preco()

'atualizar preço

If ListaMarca.Value <> "" Then

    linha_marca = Sheets("Dados").Cells.Find(ListaMarca.Value).Row
    valor_marca = Sheets("Dados").Cells(linha_marca, 2).Value

End If

If CaixaModelo.Value <> "" Then

    linha_modelo = Sheets("Dados").Cells.Find(CaixaModelo.Value).Row
    valor_modelo = Sheets("Dados").Cells(linha_modelo, 4).Value

End If

If BotaoManual.Value = True Then

    linha_cambio = Sheets("Dados").Cells.Find("Manual").Row
    valor_cambio = Sheets("Dados").Cells(linha_cambio, 6).Value

ElseIf BotaoAutomatico.Value = True Then

    linha_cambio = Sheets("Dados").Cells.Find("Automático").Row
    valor_cambio = Sheets("Dados").Cells(linha_cambio, 6).Value

End If

desconto = BarraDesconto.Value / 100

valor_final = (valor_marca + valor_modelo + valor_cambio) * (1 - desconto)

TextoValor.Value = Format(valor_final, "Currency")

End Sub
```

O código completo da Sub calcula\_preco está mostrada na imagem ao lado. O resultado final que queremos é calcular o valor final da venda de acordo com cada um dos objetos do UserForm que podem afetar este valor final, que no caso são:

- ListaMarca,
- CaixaModelo
- BotaoManual
- BotaoAutomatico

Como estes botões estão inicialmente desmarcados, antes de mais nada devemos usar uma estrutura If para testar se o botão possui ou não algum valor, e somente executar o comando caso algum valor tenha sido preenchido.

```

Sub calcula_preco()
    'atualizar preço
    If ListaMarca.Value <> "" Then
        linha_marca = Sheets("Dados").Cells.Find(ListaMarca.Value).Row
        valor_marca = Sheets("Dados").Cells(linha_marca, 2).Value
    End If

    If CaixaModelo.Value <> "" Then
        linha_modelo = Sheets("Dados").Cells.Find(CaixaModelo.Value).Row
        valor_modelo = Sheets("Dados").Cells(linha_modelo, 4).Value
    End If

    If BotaoManual.Value = True Then
        linha_cambio = Sheets("Dados").Cells.Find("Manual").Row
        valor_cambio = Sheets("Dados").Cells(linha_cambio, 6).Value
    ElseIf BotaoAutomatico.Value = True Then
        linha_cambio = Sheets("Dados").Cells.Find("Automático").Row
        valor_cambio = Sheets("Dados").Cells(linha_cambio, 6).Value
    End If

    desconto = BarraDesconto.Value / 100
    valor_final = (valor_marca + valor_modelo + valor_cambio) * (1 - desconto)
    TextoValor.Value = Format(valor_final, "Currency")
End Sub

```

Dentro de cada If, devemos basicamente fazer duas coisas:

1. Descobrir qual é a linha onde a opção marcada no botão encontra-se na aba “Dados”, usando o comando .Find + .Row.
2. Em seguida, sabendo a linha da opção marcada, descobrimos o preço.

Apenas um exemplo de como estes dois passos funcionam, caso a opção de Moto marcada seja “Kawasaki”, o comando Find vai encontrar essa opção na célula A4, e o comando Row vai retornar a linha 4. O preço da “Kawasaki” encontra-se exatamente na mesma linha, da coluna B (coluna 2).

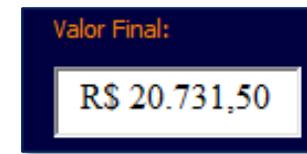
	A	B	C	D	E	F
1	Moto	Preço	Modelo	Preço Modelo	Itens	Preço Itens
2	Honda	\$ 5.570	Básico	\$ -	Manual	\$ -
3	Yamaha	\$ 6.480	Completo	\$ 1.000	Automático	\$ 2.500
4	Kawasaki	\$ 20.890	Especial	\$ 2.000		
5	BMW	\$ 29.800	Luxo	\$ 3.000		
6	Harley-Davidson	\$ 40.500	Exclusive	\$ 5.000		
7	Suzuki	\$ 6.490				
8	Vespa	\$ 27.900				
9						

```
Sub calcula_preco()
    'atualizar preço
    If ListaMarca.Value <> "" Then
        linha_marca = Sheets("Dados").Cells.Find(ListaMarca.Value).Row
        valor_marca = Sheets("Dados").Cells(linha_marca, 2).Value
    End If
    If CaixaModelo.Value <> "" Then
        linha_modelo = Sheets("Dados").Cells.Find(CaixaModelo.Value).Row
        valor_modelo = Sheets("Dados").Cells(linha_modelo, 4).Value
    End If
    If BotaoManual.Value = True Then
        linha_cambio = Sheets("Dados").Cells.Find("Manual").Row
        valor_cambio = Sheets("Dados").Cells(linha_cambio, 6).Value
    ElseIf BotaoAutomatico.Value = True Then
        linha_cambio = Sheets("Dados").Cells.Find("Automático").Row
        valor_cambio = Sheets("Dados").Cells(linha_cambio, 6).Value
    End If
    desconto = BarraDesconto.Value / 100
    valor_final = (valor_marca + valor_modelo + valor_cambio) * (1 - desconto)
    TextoValor.Value = Format(valor_final, "Currency")
End Sub
```

Além disso, devemos armazenar o valor do desconto aplicado no botão BarraDesconto.

Em seguida, usamos todas as variáveis a seguir: **valor\_marca**, **valor\_modelo**, **valor\_cambio** e **desconto**, para calcular o valor final da venda, armazenado na variável **valor\_final**.

Por fim, devemos atribuir este **valor\_final** à caixa de texto Valor Final:



Perceba também que, para aplicar a formatação de moeda, utilizamos o comando **Format**.

```
Private Sub BarraDesconto_Change()  
  
TextoDesconto.Value = BarraDesconto.Value & "%"  
  
'atualizar preço  
Call calcula_preco  
  
End Sub  
  
Private Sub BotaoAutomatico_Click()  
  
'atualizar preço  
Call calcula_preco  
  
End Sub  
  
Private Sub BotaoManual_Click()  
  
'atualizar preço  
Call calcula_preco  
  
End Sub  
  
Private Sub CaixaModelo_Change()  
  
'atualizar preço  
Call calcula_preco  
  
End Sub  
  
Private Sub ListaMarca_Click()  
  
nome_moto = ListaMarca.Value  
ImagenMoto.Picture = LoadPicture(ThisWorkbook.Path & "\Imagens\" & nome_moto & ".jpg")  
  
'atualizar preço  
Call calcula_preco  
  
End Sub
```

Antes de executar o nosso UserForm, ainda falta fazer com que as 5 macros mencionadas anteriormente executem a Sub `calcula_preco`. Para fazer com que várias macros executem um código que está em outra Sub, devemos utilizar a função **Call**.

Lembrando que devemos executar essa Sub sempre que algum botão for modificado, pois queremos que o valor final da venda se atualize enquanto a gente estiver com o UserForm aberto.

Agora sim os nossos códigos para cadastro dos valores nos botões estão prontos, e o resultado final é mostrado na página seguinte. Claro que ainda precisamos registrar esta venda na nossa planilha, pois por enquanto nada acontecerá ao tentarmos clicar em OK pois este código para registro na planilha ainda não foi construído.

Agora vamos finalizar este código para conseguir fazer o que falta: registrar as informações de venda preenchidas no UserForm. A macro de registrar os valores na planilha deve ser executada sempre ao clicar no botão de OK. Portanto, a macro deverá estar associada ao evento Click deste botão.

Para começar o código, voltamos no ambiente do VBA, selecionamos o UserForm de vendas e damos um duplo clique no botão de OK.



BotaoConfirmar

Click

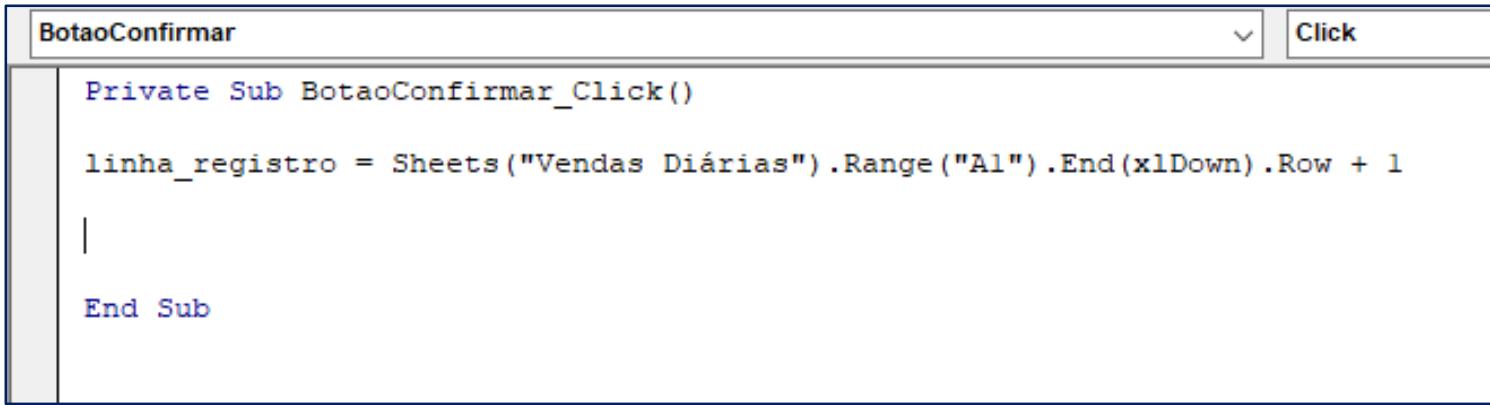
```
Private Sub BotaoConfirmar_Click()
|
End Sub
```

Antes de começar o código, devemos entender o nosso objetivo. Queremos fazer o seguinte:

Ao clicar no botão de OK, todas as informações de vendas devem ser registradas na tabela da aba **Vendas Diárias**, mostrada na imagem abaixo.

Portanto, antes de mais nada sabemos que vamos precisar descobrir a última linha preenchida desta tabela, para registrar a nossa nova venda.

	A	B	C	D	E	F	G	H	I	
1	Nº da Venda	Data	Marca	Categoria	Câmbio	Desconto	Valor	Disponibilidade no Estoque	Opcionais	
2	1	13/09/2017	BMW	Completo	Automático	6%	\$31.302,00	Disponível	Banco de Couro, Capacete	
3	2	27/11/2017	Harley-Davidson	Completo	Automático	2%	\$43.120,00	Disponível	Jaqueta Preta, Capacete, Adesivos	
4										
5										
6										



The screenshot shows the Microsoft Visual Basic Editor with a single module named "BotaoConfirmar". The "Click" event is selected, displaying the following VBA code:

```
Private Sub BotaoConfirmar_Click()
    linha_registro = Sheets("Vendas Diárias").Range("A1").End(xlDown).Row + 1
    |
End Sub
```

Primeiro começamos encontrando a última linha preenchida na tabela da aba Vendas Diárias.

Vamos armazenar a linha de registro em uma variável chamada **linha\_registro**.

```
BotaoConfirmar
Private Sub BotaoConfirmar_Click()
    Sheets("Vendas Diárias").Activate
    linha_registro = Range("A1").End(xlDown).Row + 1
    'O número da venda é sempre o número da linha de registro menos um.
    Cells(linha_registro, 1).Value = linha_registro - 1
    Cells(linha_registro, 2).Value = Date
    Cells(linha_registro, 3).Value = ListaMarca.Value
    Cells(linha_registro, 4).Value = CaixaModelo.Value
    If BotaoManual.Value = True Then
        Cells(linha_registro, 5).Value = "Manual"
    ElseIf BotaoAutomatico.Value = True Then
        Cells(linha_registro, 5).Value = "Automático"
    End If
    Cells(linha_registro, 6).Value = TextoDesconto.Value
    Cells(linha_registro, 7).Value = TextoValor.Value
End Sub
```

Em seguida, vamos registrar em cada coluna desta linha de registro os valores preenchidos nos botões do formulário.

```
Cells(linha_registro, 5).Value = "Automático"  
End If  
  
Cells(linha_registro, 6).Value = TextoDesconto.Value  
Cells(linha_registro, 7).Value = TextoValor.Value  
  
If Opcional1.Value = True Then  
    Cells(linha_registro, 9).Value = "Banco de Couro"  
End If  
  
If Opcional2.Value = True Then  
    If Cells(linha_registro, 9).Value = "" Then  
        Cells(linha_registro, 9).Value = "Jaqueta Preta"  
    Else  
        Cells(linha_registro, 9).Value = Cells(linha_registro, 9).Value & ", Jaqueta Preta"  
    End If  
End If  
  
End Sub
```

Para preencher os opcionais selecionadas nas 4 caixas de seleção, primeiro precisamos testar se a caixa de seleção foi marcada e, se for, registrar o opcional na coluna de opcionais.

Porém, como podemos marcar mais de 1 opcional, quando formos escrever os opcionais marcados dentro da célula, a partir do segundo, precisamos testar se já tem algum valor escrito na célula. Se não tiver, simplesmente registramos com o nome do opcional. Caso contrário, precisamos acrescentar ao valor que já está na célula o texto do opcional, acrescentando uma vírgula para separar os textos dentro da célula.

```
Cells(linha_registro, 5).Value = "Automático"

End If

Cells(linha_registro, 6).Value = TextoDesconto.Value
Cells(linha_registro, 7).Value = TextoValor.Value

If Opcional1.Value = True Then
    Cells(linha_registro, 9).Value = "Banco de Couro"
End If

If Opcional2.Value = True Then
    If Cells(linha_registro, 9).Value = "" Then
        Cells(linha_registro, 9).Value = "Jaqueta Preta"
    Else
        Cells(linha_registro, 9).Value = Cells(linha_registro, 9).Value & ", Jaqueta Preta"
    End If
End If

If Opcional3.Value = True Then
    If Cells(linha_registro, 9).Value = "" Then
        Cells(linha_registro, 9).Value = "Capacete"
    Else
        Cells(linha_registro, 9).Value = Cells(linha_registro, 9).Value & ", Capacete"
    End If
End If

If Opcional4.Value = True Then
    If Cells(linha_registro, 9).Value = "" Then
        Cells(linha_registro, 9).Value = "Adesivos"
    Else
        Cells(linha_registro, 9).Value = Cells(linha_registro, 9).Value & ", Adesivos"
    End If
End If

End Sub
```

Fazemos esse processo para cada um dos 4 opcionais.

## Módulo 12 – UserForms – Registrando a Venda da Moto (Parte 5)

443

The screenshot displays two Microsoft Excel windows side-by-side.

**Vendas.xlsxm - Excel** (Top Window):

Nº da Venda	Data	Marca	Categoria	Câmbio	Desconto	Valor	Disponibilidade no Estoque	Opcionais
1	13/09/2017	BMW	Completo	Automático	6%	\$31.302,00	Disponível	Banco de Couro, Capacete
2	27/11/2017	Harley-Davidson	Completo	Automático	2%	\$43.120,00	Disponível	Jaqueta Preta, Capacete, Adesivos

**Estoque.xlsxm - Excel** (Bottom Window):

	Marca	Categoria	Quantidade em Estoque
1	Honda	Básico	1
2	Honda	Completo	1
3	Honda	Especial	0
4	Honda	Luxo	2
5	Honda	Exclusive	2
6	Yamaha	Básico	1

O próximo passo é verificar a disponibilidade das motos no arquivo **Estoque.xlsxm**.

Portanto, durante a execução do UserForm, vamos precisar abrir o arquivo de estoque e fazer a verificação da disponibilidade.

Além disso, caso haja disponibilidade, a venda será realizada e no arquivo de estoque a Quantidade em Estoque deverá se atualizar automaticamente.



Volte para o UserForm e dê um duplo clique no botão de **OK** para voltar para o código que estávamos escrevendo no botão BotaoConfirmar.

```
If Opcional4.Value = True Then
    If Cells(linha_registro, 9).Value = "" Then
        Cells(linha_registro, 9).Value = "Adesivos"
    Else
        Cells(linha_registro, 9).Value = Cells(linha_registro, 9).Value & ", Adesivos"
    End If
End If
```

```
If Opcional4.Value = True Then  
    If Cells(linha_registro, 9).Value = "" Then  
        Cells(linha_registro, 9).Value = "Adesivos"  
    Else  
        Cells(linha_registro, 9).Value = Cells(linha_registro, 9).Value & ", Adesivos"  
    End If  
End If  
  
Workbooks.Open (ThisWorkbook.Path & "\Estoque.xlsm")  
  
|  
  
End Sub
```

Damos sequência ao nosso código abrindo o arquivo Estoque.xlsm.

```
End If  
End If  
  
Workbooks.Open (ThisWorkbook.Path & "\Estoque.xlsx")  
  
linha = 2  
  
Do Until Cells(linha, 1).Value = ListaMarca.Value And Cells(linha, 2).Value = CaixaModelo.Value  
  
    linha = linha + 1  
  
Loop  
  
|  
  
End Sub
```

	A	B	C	
1	Marca	Categoria	Quantidade em Estoque	
2	Honda	Básico	1	
3	Honda	Completo	1	
4	Honda	Especial	0	
5	Honda	Luxo	2	
6	Honda	Exclusive	2	
7	Yamaha	Básico	1	
8	Yamaha	Completo	0	
9	Yamaha	Especial	5	
10	Yamaha	Luxo	1	
11	Yamaha	Exclusive	0	
12	Kawasaki	Básico	2	
13	Kawasaki	Completo	1	
14	Kawasaki	Especial	5	

Em seguida, devemos encontrar a linha onde está a linha da moto a ser vendida. Porém, não basta aqui usar o comando Find porque a linha depende não só da Marca mas da Categoria também, então devemos fazer uma procura de acordo com duas condições, então o Findo não servirá.

Neste caso, poderemos substituir pela estrutura Do Until com duas condições, uma avaliando a coluna A e depois a coluna B da tabela. A cada Loop a variável linha vai atualizando e quando Marca e Categoria forem encontrados, o último valor da linha vai ser exatamente o número da linha onde temos a Marca e Categoria.

```
Workbooks.Open (ThisWorkbook.Path & "\Estoque.xlsx")
linha = 2
Do Until Cells(linha, 1).Value = ListaMarca.Value And Cells(linha, 2).Value = CaixaModelo.Value
    linha = linha + 1
Loop
'aqui a variável "linha" vai ter o valor da linha onde estão a marca e o modelo selecionado
qtd_estoque = Cells(linha, 3).Value
If qtd_estoque > 0 Then
    Cells(linha, 3).Value = Cells(linha, 3).Value - 1
End If
ActiveWorkbook.Save
ActiveWorkbook.Close
ThisWorkbook.Activate
|
End Sub
```

Após encontrada a linha, armazenamos a quantidade em estoque, que encontra-se na coluna C, na variável **qtd\_estoque**.

Em seguida, verificamos se o estoque é maior do que zero. Se for, diminuímos uma venda, se não, não faz nada.

Para fechar esta parte, salvamos o ActiveWorkbook, que é o Estoque.xlsx (ele é o arquivo ativo porque abrimos ele) e depois fechamos.

Para garantir que voltamos para o arquivo de Vendas, usamos o comando ThisWorkbook.Activate.

```
linha = linha + 1
Loop
'aqui a variável "linha" vai ter o valor da linha onde estão a marca e o modelo selecionado
qtd_estoque = Cells(linha, 3).Value
If qtd_estoque > 0 Then
    Cells(linha, 3).Value = Cells(linha, 3).Value - 1
End If
ActiveWorkbook.Save
ActiveWorkbook.Close
ThisWorkbook.Activate
If qtd_estoque > 0 Then
    Cells(linha_registro, 8).Value = "Disponivel"
    resp = MsgBox("Venda realizada com sucesso", vbExclamation, "Confirmação")
Else
    Cells(linha_registro, 8).Value = "Indisponível"
    resp = MsgBox(ListaMarca.Value & " " & CaixaModelo.Value & " indisponivel. Favor solicitar à montadora", vbExclamation, "Aviso")
End If
Sheets("Principal").Select
|
Unload MeuFormulario
End Sub
```

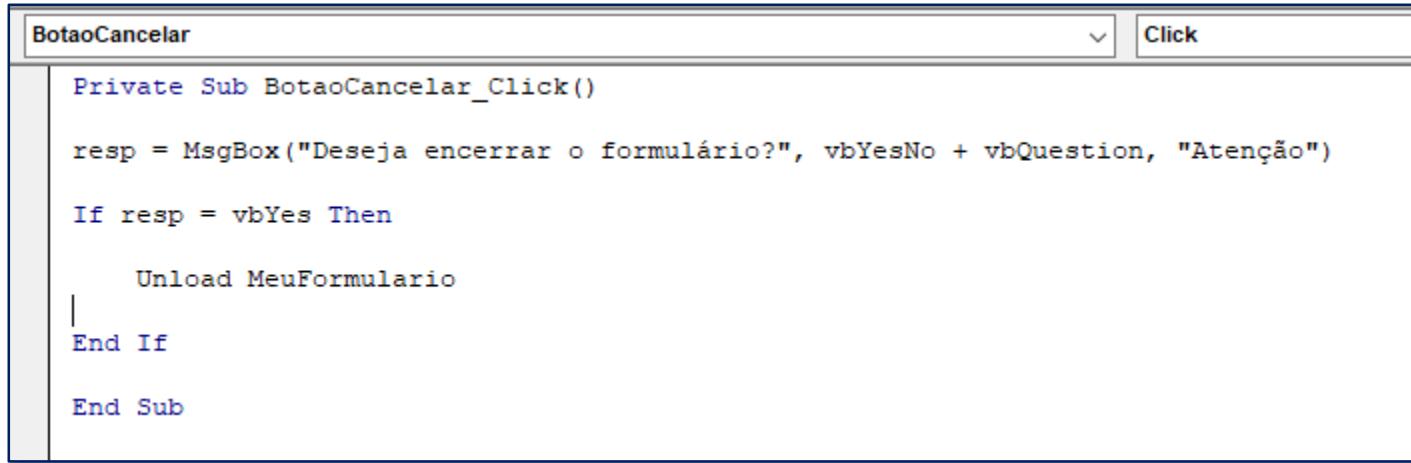
O próximo passo agora é registrar o status da moto, que poderá ser “Disponível” ou “Indisponível” de acordo com a variável **qtd\_estoque**. Para melhorar a experiência do usuário, podemos exibir mensagens de aviso para dizer se a venda foi realizada ou se o produto estava indisponível.

Por fim, selecionamos a aba Principal com o comando Select e fechamos o formulário.



Finalmente, vamos configurar o botão de Cancelar. Ele será bem simples. Basicamente ele vai conter o comando para fechar (Unload) o formulário.

Clique duas vezes no botão para começar o código.



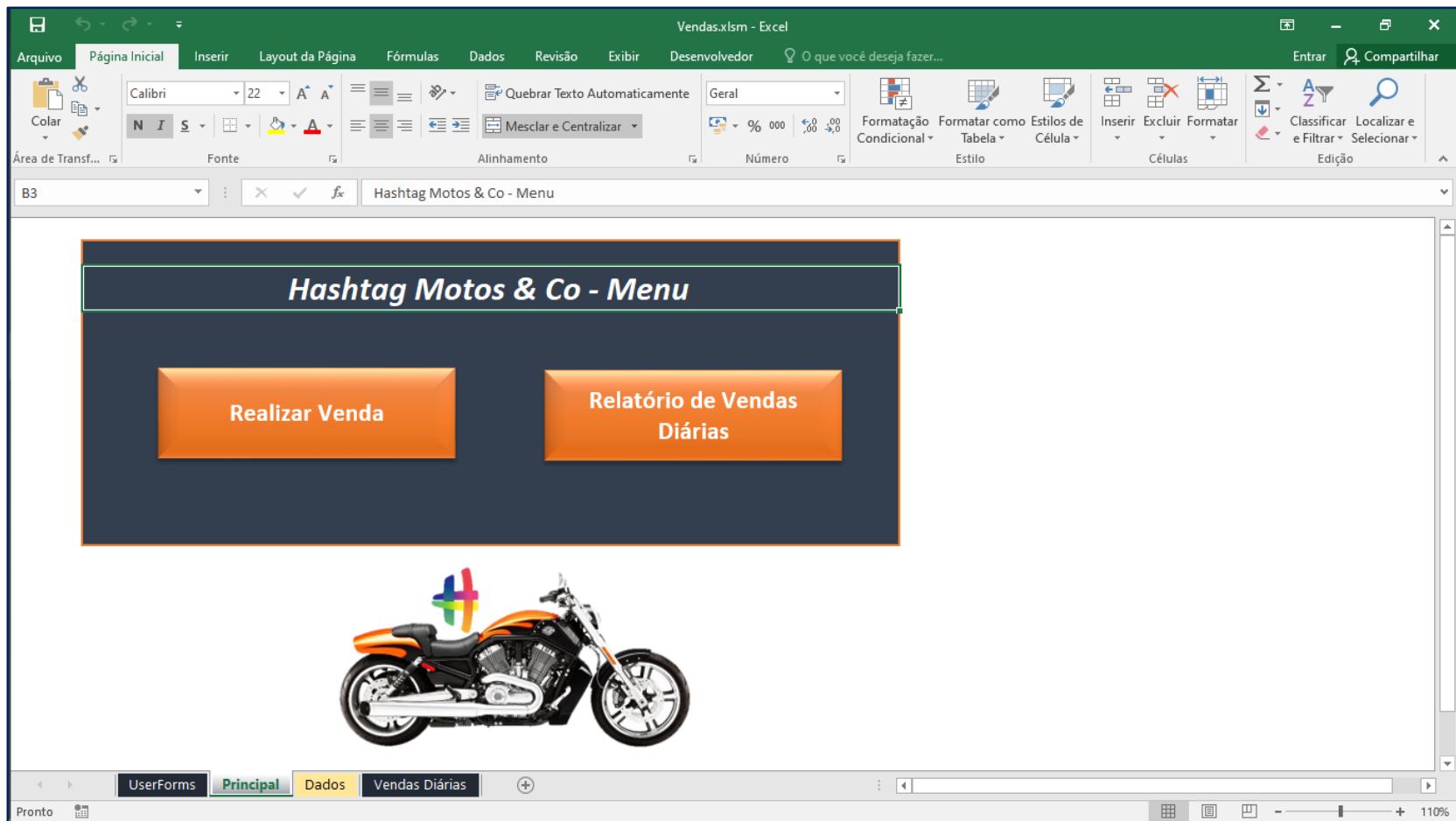
The screenshot shows a Microsoft Visual Studio code editor window titled "BotaoCancelar". The tab bar indicates the code is for the "Click" event. The code itself is as follows:

```
Private Sub BotaoCancelar_Click()
    resp = MsgBox("Deseja encerrar o formulário?", vbYesNo + vbQuestion, "Atenção")
    If resp = vbYes Then
        Unload MeuFormulario
    End If
End Sub
```

Como pode acontecer de o usuário clicar sem querer neste botão, podemos exibir uma mensagem perguntando se de fato ele deseja fechar o formulário.

## Módulo 12 – UserForms – Botão Cancelar

451



Finalmente concluímos o nosso formulário e o módulo de UserForms.

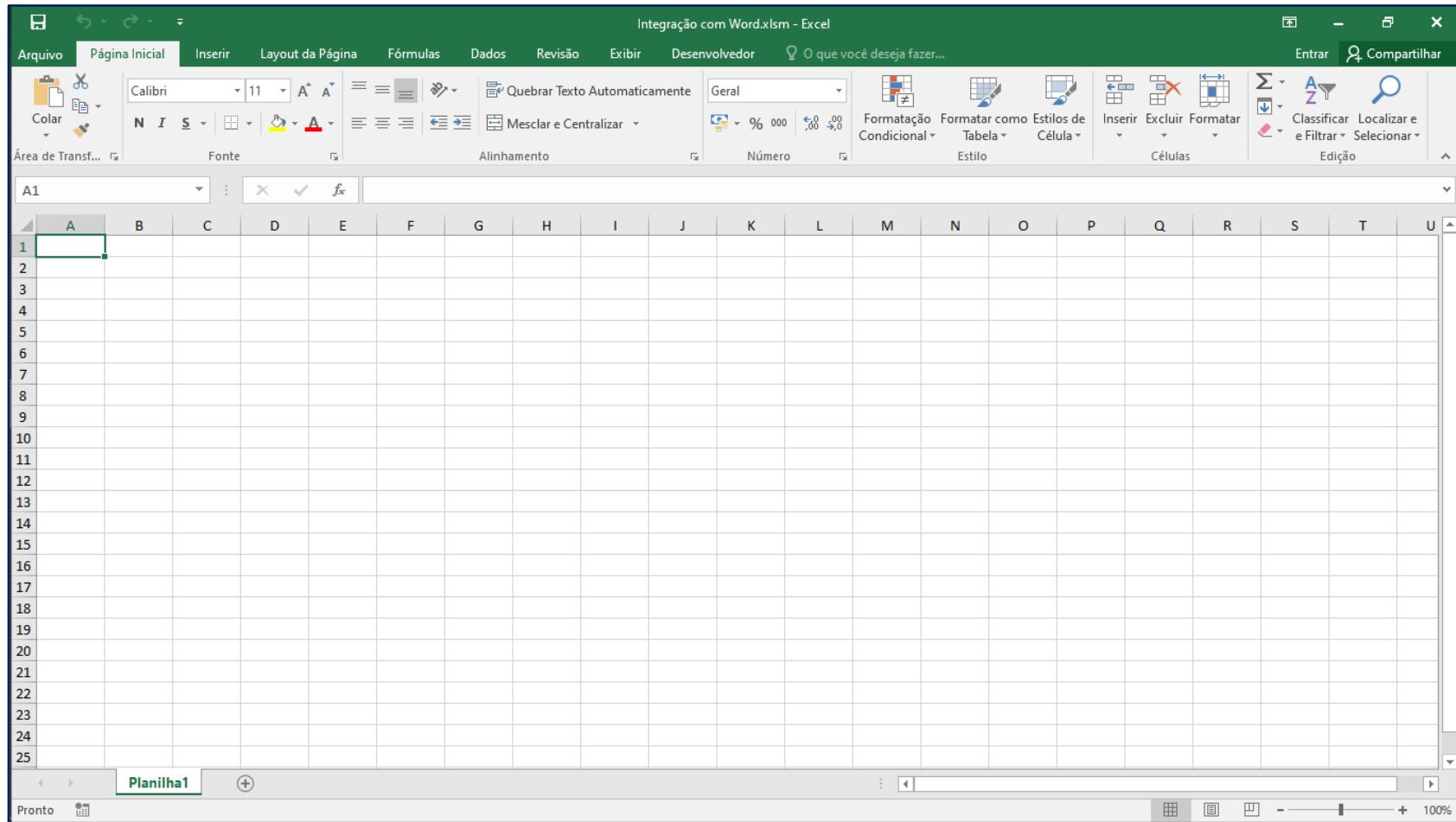
Com certeza a partir de agora você já será capaz de criar ferramentas muito avançadas, proporcionando a melhor experiência possível do usuário durante a utilização das suas planilhas.

Módulo 13

# Integração VBA com Word

# Módulo 13 – Integração com Word – Abrindo um Documento em Word do Zero

452



A partir deste módulo, daremos um grande passo em direção a novas possibilidades além do Excel. Vamos começar a realizar integrações do VBA com o Excel, e nos próximos módulos, com Outlook e PowerPoint.

Abrimos um arquivo em branco e vamos começar a criar os códigos normalmente no ambiente VBA.

```
Sub word()  
  
Set objeto_word = CreateObject("Word.Application")  
  
objeto_word.Visible = True  
  
Set objeto_documento = objeto_word.documents.Add  
  
|  
  
End Sub
```

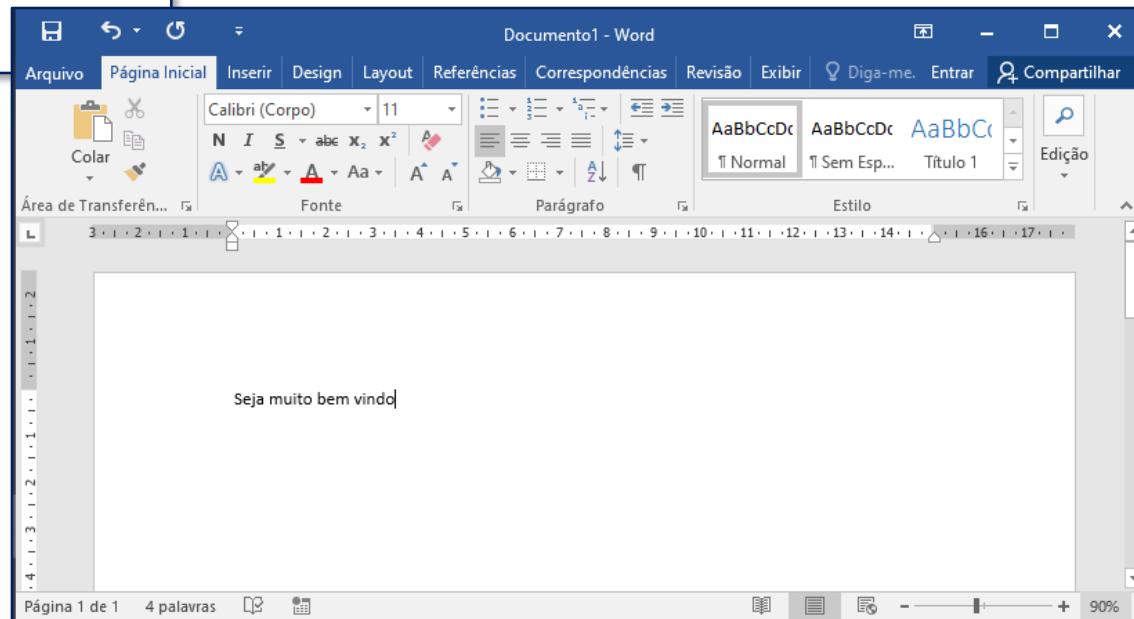
Vamos criar uma macro chamada **Word**.

Uma vez que agora vamos trabalhar com um novo objeto no VBA (um aplicativo do Word), precisamos criar este objeto através da instrução `CreateObject`. Em seguida, fazemos esta aplicação se tornar visível.

O arquivo Word será criado através da instrução `.Add`, e vamos atribuir este documento a uma variável chamada `objeto_documento`. Lembrando que o `Set` será utilizado para variáveis que armazenam objetos, tais como células, gráficos, planilhas, arquivos Excel, Word, etc.

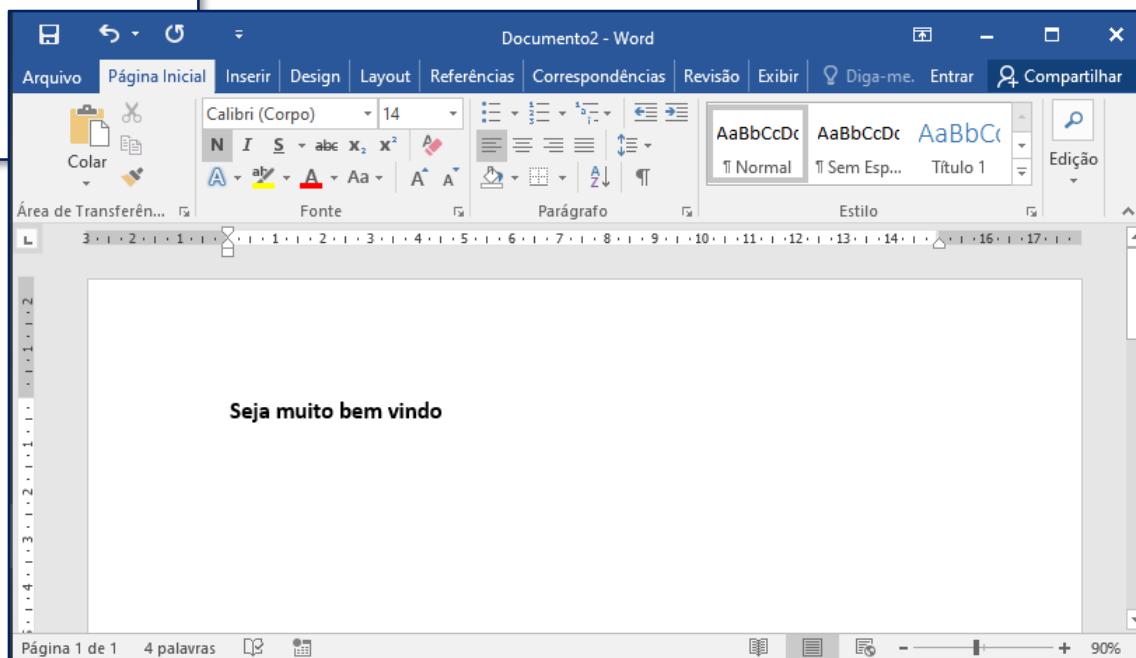
```
Sub word()
    Set objeto_word = CreateObject("Word.Application")
    objeto_word.Visible = True
    Set objeto_documento = objeto_word.documents.Add
    Set selecao = objeto_word.Selection
    selecao.typetext ("Seja muito bem vindo")
    |
End Sub
```

Declaramos então outra variável chamada seleção, que nada mais é do que o local onde o cursor do word estará posicionado. Para escrever um texto no arquivo utilizamos a função **typetext**.



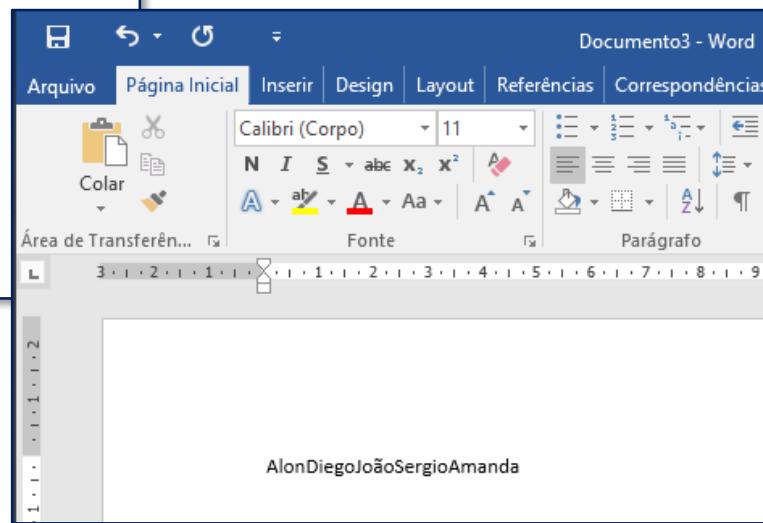
```
Sub word()
    Set objeto_word = CreateObject("Word.Application")
    objeto_word.Visible = True
    Set objeto_documento = objeto_word.documents.Add
    Set selecao = objeto_word.Selection
    selecao.Font.Bold = True
    selecao.Font.Size = 14
    selecao.typetext ("Seja muito bem vindo")
End Sub
```

Se quisermos alterar a formatação do texto, tais como Negrito e Tamanho do Texto, utilizamos as duas linhas de código ao lado logo antes de escrever o texto com a instrução typetext.



	A	B
1	Funcionários	
2	Alon	
3	Diego	
4	João	
5	Sergio	
6	Amanda	
7		
8		

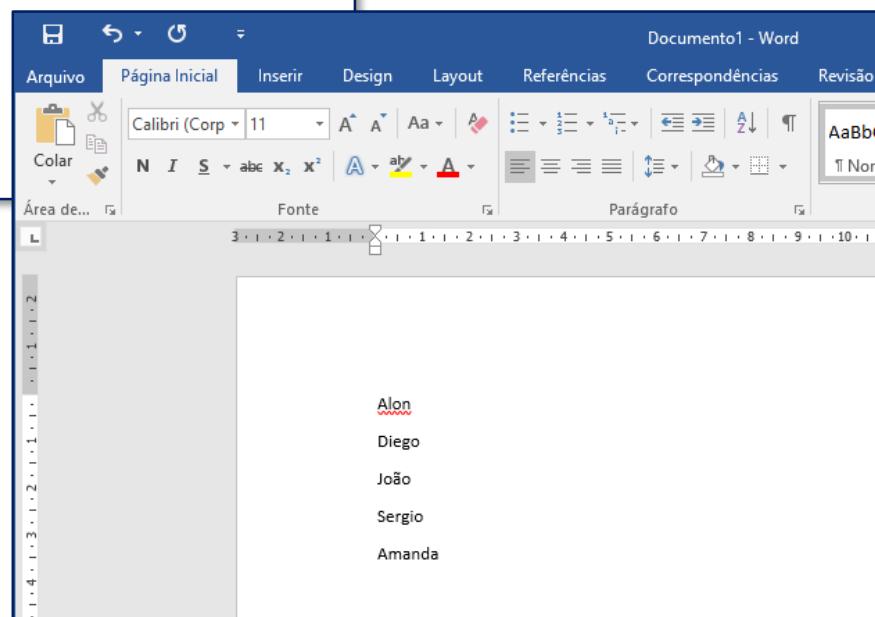
```
Sub word()
    Set objeto_word = CreateObject("Word.Application")
    objeto_word.Visible = True
    Set objeto_documento = objeto_word.documents.Add
    Set selecao = objeto_word.Selection
    linha = 2
    Do Until Cells(linha, 1).Value = ""
        selecao.typetext (Cells(linha, 1).Value)
        linha = linha + 1
    Loop
End Sub
```



Imagine agora que temos uma lista de nomes de funcionários na nossa planilha, da linha 2 até a linha 6, e queremos escrever estes nomes, um embaixo do outro, no nosso arquivo Word. Podemos utilizar a estrutura de repetição Do Until para percorrer cada linha desta lista de nomes no Excel e com a função typetext escrever estes nomes no arquivo Word.

Porém, o código simplesmente escreve todos os nomes na mesma linha.

```
Sub word()
    Set objeto_word = CreateObject("Word.Application")
    objeto_word.Visible = True
    Set objeto_documento = objeto_word.documents.Add
    Set selecao = objeto_word.Selection
    linha = 2
    Do Until Cells(linha, 1).Value = ""
        selecao.typetext (Cells(linha, 1).Value & Chr(13))
        linha = linha + 1
    Loop
End Sub
```



Para pular uma linha logo após preencher cada nome, você pode concatenar o valor da célula com o código **Chr(13)**. A função Chr retorna diferentes caracteres de acordo com o número passado como argumento. O número 13 em particular irá inserir uma quebra de linha, como mostrado na imagem abaixo.

# Módulo 13 – Integração com Word – Registrando Atividades dos Funcionários (Parte 1)

458

The screenshot shows a Microsoft Excel window titled "Integração com Word - Funcionários.xlsxm - Excel". The ribbon menu includes Arquivo, Página Inicial, Inserir, Layout da Página, Fórmulas, Dados, Revisão, Exibir, Desenvolvedor, and Ajuda. The "Página Inicial" tab is selected. The table in the spreadsheet has columns A and B. Column A contains names from 1 to 7, and column B contains their respective activity descriptions. A black button labeled "Compilar Atividades" is overlaid on the right side of the table.

	Nome	Descrição das atividades
1	Alon	Gravar vídeos youtube, Dar aula presencial
2	João	Gravar anúncios, Gravar vídeos youtube, Scriptar vídeos
3	Diego	Fazer controle financeiro, Editar vídeos, Responder e-mail, Criar planilhas
4	Sergio	Falar com alunos, Subir anúncios, Responder Inbox WhatsApp
5	Amanda	Responder E-mail, Responder Inbox Instagram, Criar artes
6	Hecio	Edita vídeos, Criar Roteiros, Atendimento ao Cliente
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		

No próximo exercício vamos criar um código para preencher cada um dos funcionários ao lado e a sua lista de atividades, como mostrado na imagem abaixo:

<b>Alon</b>
- Gravar vídeos youtube
- Dar aula presencial
<b>João</b>
- Gravar anúncios
- Gravar vídeos youtube
- Scriptar vídeos
<b>Diego</b>
- Fazer controle financeiro
- Editar vídeos
- Responder e-mail
- Criar planilhas
<b>Sergio</b>
- Falar com alunos
- Subir anúncios
- Responder Inbox WhatsApp

```
Sub word()
    Set objeto_word = CreateObject("Word.Application")
    Set objeto_documento = objeto_word.documents.Add
    objeto_word.Visible = True
    Set selecao = objeto_word.Selection
    linha = 2

    Do Until Cells(linha, 1).Value = ""
        selecao.typetext (Cells(linha, 1).Value & Chr(13))
        linha = linha + 1
    Loop
    |
End Sub
```

A lógica por trás do código que utilizaremos já havia sido construída, e agora vamos utilizá-lo neste novo arquivo e fazer as adaptações necessárias.

O código é exatamente o mesmo, apenas trocando a posição de algumas linhas por questão de organização.

```
Sub word()
    Set objeto_word = CreateObject("Word.Application")
    Set objeto_documento = objeto_word.documents.Add
    objeto_word.Visible = True
    Set selecao = objeto_word.Selection
    linha = 2

    Do Until Cells(linha, 1).Value = ""
        selecao.Font.Bold = True
        selecao.Font.Size = 14
        selecao.typetext (Cells(linha, 1).Value & Chr(13))

        selecao.Font.Bold = False
        selecao.Font.Size = 11

        atividades = WorksheetFunction.Substitute(Cells(linha, 2).Value, ",",
        Chr(13) & "-")
        | selecao.typetext ("-" & atividades & Chr(13) & Chr(13))

        linha = linha + 1
    Loop
End Sub
```

Para organizar a estrutura do texto da forma mostrada na página 458, basicamente vamos incluir as linhas de código ao lado.

Queremos colocar cada nome em maiúscula, então antes de mais nada, ativamos as configurações de negrito e tamanho do texto. Após escrever o nome que está na coluna A, desativamos o negrito e voltamos o tamanho do texto para 11 e ai sim escrevemos a lista de atividades.

Como esta lista de atividades está separada por vírgulas, seria mais interessante para nós substituir a vírgula por quebras de linha, através do **Chr(13)**.

Já para colocar o hífen no início de cada atividade, concatenamos o texto “-” logo após cada Chr(13), que são as nossas quebras de linha.

The screenshot shows a Microsoft Word document window titled "Documento4 - Word". The ribbon menu is visible at the top, showing tabs like Arquivo, Página Inicial, Inserir, Design, Layout, Referências, Correspondências, Revisão, Exibir, Diga-me, Entrar, and Compartilhar. The "Página Inicial" tab is selected. The main content area contains the following text:

**Alan**

- Gravar vídeos youtube
- Dar aula presencial

**João**

- Gravar anúncios
- Gravar vídeos youtube
- Scriptar vídeos

**Diego**

- Fazer controle financeiro
- Editar vídeos
- Responder e-mail
- Criar planilhas

**Sergio**

- Falar com alunos
- Subir anúncios
- Responder inbox WhatsApp

At the bottom left, it says "Página 1 de 2 68 palavras". The status bar at the bottom right shows "70%".

O resultado final é mostrado ao lado.

```
Sub word()

Set objeto_word = CreateObject("Word.Application")
Set objeto_documento = objeto_word.documents.Add
objeto_word.Visible = True
Set selecao = objeto_word.Selection

linha = 2

Do Until Cells(linha, 1).Value = ""

selecao.Font.Bold = True
selecao.Font.Size = 14

selecao.typetext (Cells(linha, 1).Value & Chr(13))

selecao.Font.Bold = False
selecao.Font.Size = 11

atividades = WorksheetFunction.Substitute(Cells(linha, 2).Value, ",", Chr(13) & "-")

selecao.typetext ("-" & atividades & Chr(13) & Chr(13))

linha = linha + 1

Loop

objeto_documento.Saveas2 (ThisWorkbook.Path & "\Registro de Funcionários.docx")

End Sub
```

Após criar o arquivo, é importante salvá-lo em alguma pasta do nosso computador. Fazemos isso com o comando SaveAs2, e passamos o caminho da pasta como argumento.

No exemplo, salvamos na mesma pasta do arquivo Excel onde estamos criando o código VBA.

Nome	Data de modificaç...	Tipo	Tamanho
Integração com Word - Funcionários.xlsxm	13/03/2020 16:06	Planilha Habilitad...	16 KB
Registro de Funcionários.docx	13/03/2020 16:14	Documento do Mi...	12 KB

```
Sub word()
    On Error Resume Next 2
    Set objeto_word = CreateObject("Word.Application")
    Set objeto_documento = objeto_word.documents.Add
    objeto_word.Visible = True
    Set selecao = objeto_word.Selection
    linha = 2

    Do Until Cells(linha, 1).Value = ""
        selecao.Font.Bold = True
        selecao.Font.Size = 14
        selecao.typetext (Cells(linha, 1).Value & Chr(13))
        selecao.Font.Bold = False
        selecao.Font.Size = 11
        atividades = WorksheetFunction.Substitute(Cells(linha, 2).Value, "", Chr(13) & "-")
        selecao.typetext ("-" & atividades & Chr(13) & Chr(13))
        linha = linha + 1
    Loop 1
    Kill ThisWorkbook.Path & "\Registro de Funcionários.docx" 1
    objeto_documento.Saveas2 (ThisWorkbook.Path & "\Registro de Funcionários.docx")
    objeto_documento.Close 3
End Sub
```

Para finalizar este código, devemos pensar em 3 coisas:

1. Antes de salvar o nosso arquivo atualizado, devemos excluir o arquivo anterior, pois queremos sempre manter salvo um arquivo com as informações mais atualizadas dos funcionários. Para excluir um arquivo, usamos o comando **Kill**, e indicamos o caminho do arquivo que queremos excluir.
2. Porém, se o arquivo não existir, ocorrerá um erro, pois o código tentará apagar um arquivo que não existe. Para corrigir isso, usamos a instrução **On Error Resume Next**, que já vimos no módulo de tratamento de erros. Com este tratamento, ele simplesmente irá ignorar o erro caso não encontre o arquivo na pasta. Isso é exatamente o que queremos, pois se o arquivo não existir, simplesmente não faremos nada.
3. Por fim, fechamos o arquivo Word com o comando **.Close**.

# Módulo 13 – Integração com Word – Geração de Contratos (Explicação)

464

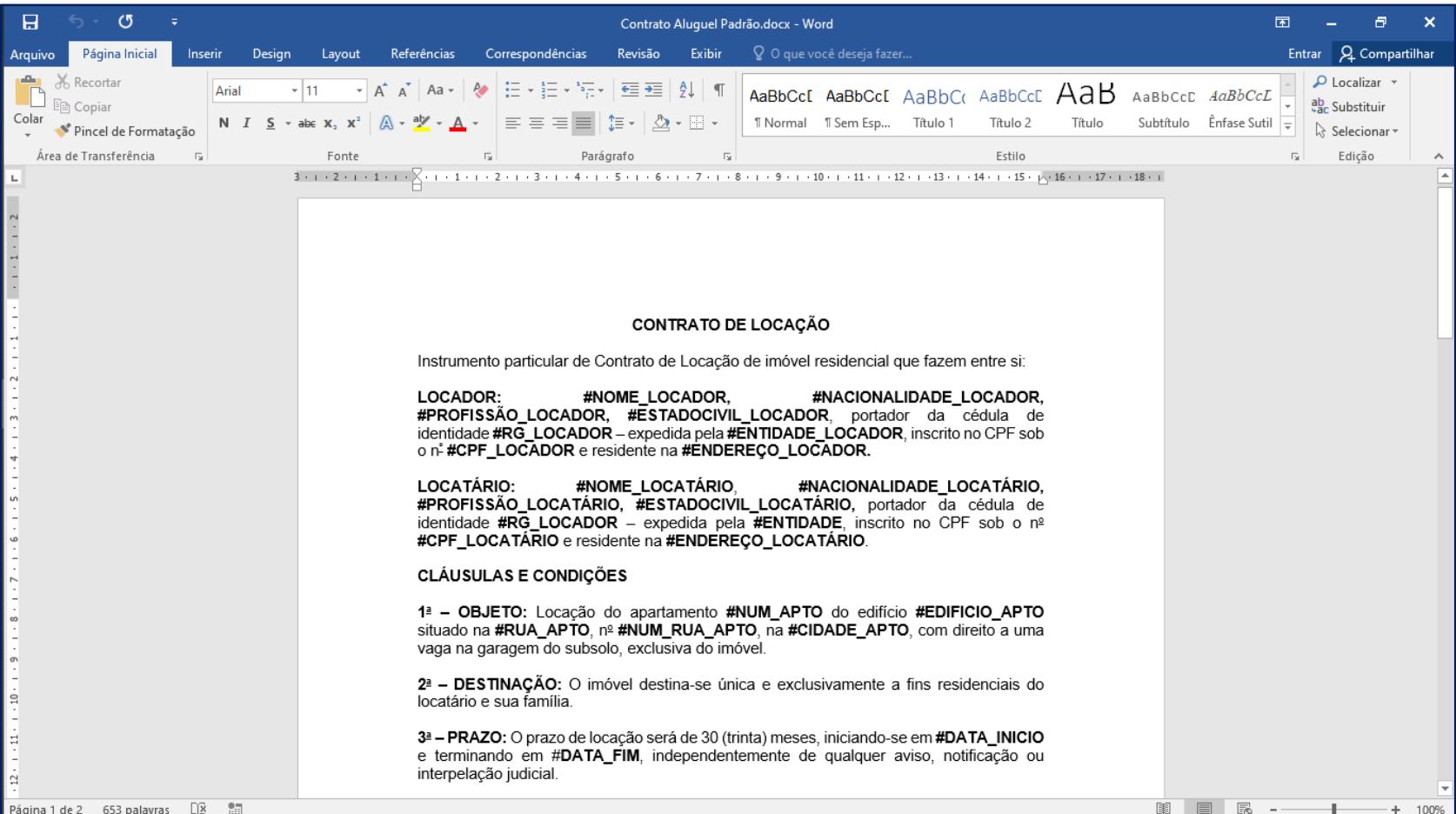
The screenshot shows a Microsoft Excel spreadsheet titled "Contratos.xlsxm - Excel". The table contains the following data:

	A	B	C	D	E	F	G	H	I
1	#NOME_LOCADOR	#NACIONALIDADE_LOCADOR	#PROFISSÃO_LOCADOR	#ESTADOCIVIL_LOCADOR	#RG_LOCADOR	#ENTIDADE_LOCADOR	#CPF_LOCADOR	#ENDERECO_LOCADOR	#NOM
2	João Paulo Rodrigues	Brasileiro	Engenheiro	Solteiro	28.431.875-4	DETTRAN-RJ	154.521.548-85	Rua Decio Vilares, 20, apto 103, Copacabana	Marin
3	Sergio Tranjan	Brasileiro	Publicitário	Casado	49.772.293-4	DETTRAN-RJ	173.273.492-92	Rua Barão da Torre, 45, apto 1007, Ipanema	Cristia
4	Carolina Souza	Brasileira	Estudante	Noiva	38.283.283-7	DETTRAN-SP	152.896.857-56	Rua Helena, 105, apto 53, Vila Olímpia	Rafael
5	Gabriel Pereira	Brasileiro	Estudante	Solteiro	12.424.274-42	DETTRAN-RS	449.212.283-43	Rua Victor Silva, 52, apto 107, Vila Conceição	Maria
6	Alexandre Donati	Brasileiro	Jornalista	Casado	18.274.382-91	DETTRAN-SP	183.183.481-52	Rua Caubi, 17, apto 907, Perdizes	Bernard

Below the table is an orange button labeled "Gerar Contratos".

No próximo exercício queremos criar uma macro capaz de gerar automaticamente uma série de contratos de locação, de acordo com a lista de locadores, no nosso arquivo Excel de Contratos.

Neste arquivo temos uma tabela com todas as informação de cada um dos locadores. Estas informações devem ser preenchidas nos seus devidos espaços no arquivo Word de contratos.

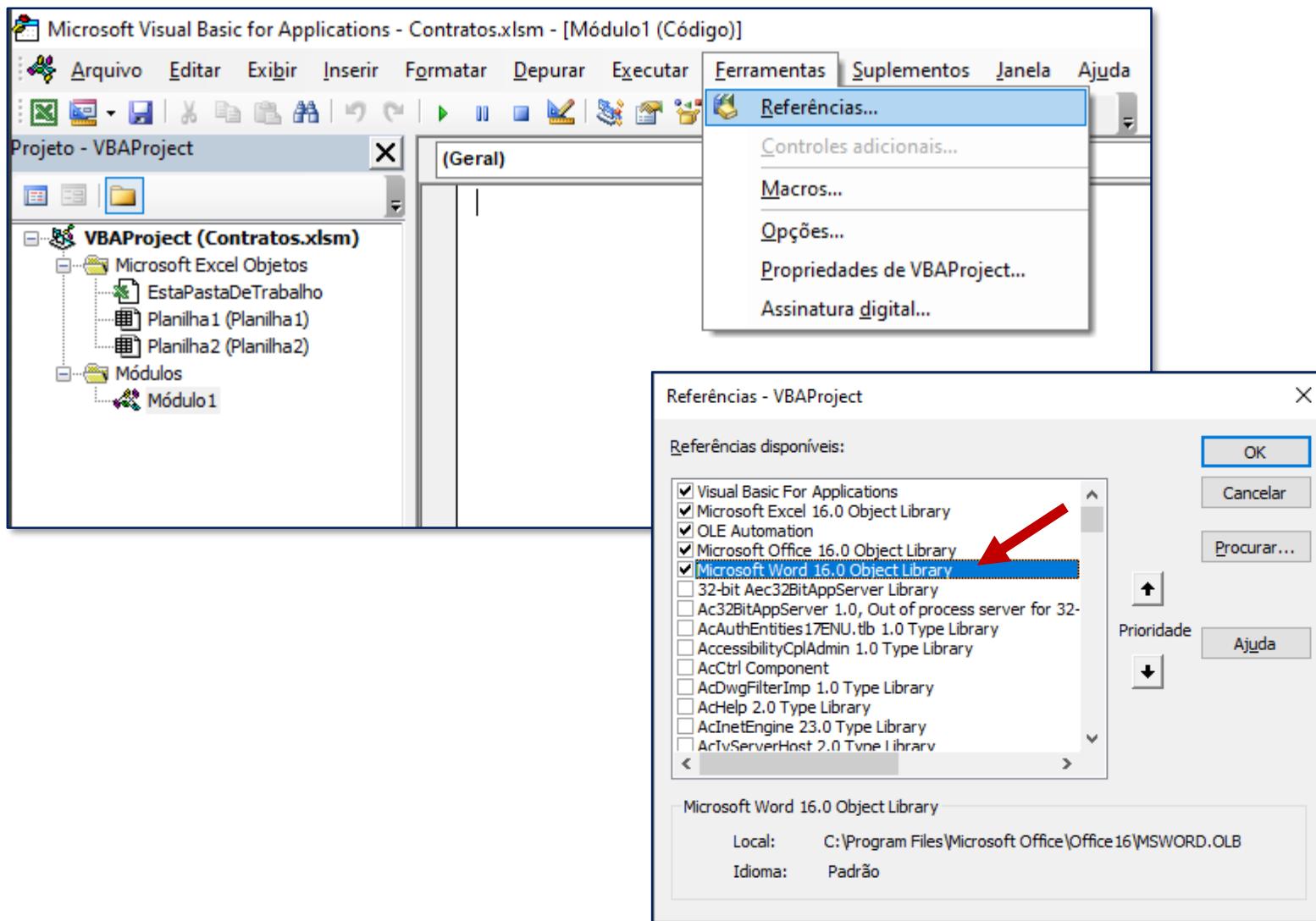


O modelo de arquivo em Word está mostrado na imagem ao lado. Repare que em cada espaço onde devem entrar as informações dos locadores temos um nome precedido de uma hashtag. Este é um padrão que devemos seguir para conseguir encontrar, via código, cada uma das informações e preencher no seu devido local.

O objetivo é que a macro abra este modelo de contrato e crie um documento para cada locador da tabela de contratos no arquivo Excel.

Nome	Data de modificação	Tipo	Tamanho
Contratos	08/01/2020 13:29	Pasta de arquivos	
Contrato Aluguel Padrão.docx	13/12/2019 10:56	Documento do Mi...	18 KB
Contratos.xlsx	13/12/2019 10:56	Planilha Habilitad...	19 KB

Todos os arquivos Word gerados deverão ser salvos na pasta chamada Contratos, que está junta dos arquivos do modelo de contrato e da planilha Excel com as informações dos locadores.



Antes de começar a criar o código, é necessário habilitar uma opção para possibilitar comandos de integração com o Word mais completos.

Para isso, no ambiente VBA, vá na guia **Ferramentas** → **Referências**.

Em seguida, procure pela opção **Microsoft Word 16.0 Object Library** e marque-a.

**CONTRATO DE LOCAÇÃO**

Instrumento particular de Contrato de Locação de imóvel residencial que fazem entre si:

**LOCADOR:** #NOME\_LOCADOR, #NACIONALIDADE\_LOCADOR, #PROFISSÃO\_LOCADOR, #ESTADOCIVIL\_LOCADOR, portador da cédula de identidade #RG\_LOCADOR – expedida pela #ENTIDADE\_LOCADOR, inscrito no CPF sob o nº #CPF\_LOCADOR e residente na #ENDERECO\_LOCADOR.

**LOCATÁRIO:** #NOME\_LOCATÁRIO, #NACIONALIDADE\_LOCATÁRIO, #PROFISSÃO\_LOCATÁRIO, #ESTADOCIVIL\_LOCATÁRIO, portador da cédula de identidade #RG\_LOCADOR – expedida pela #ENTIDADE, inscrito no CPF sob o nº #CPF\_LOCATÁRIO e residente na #ENDERECO\_LOCATÁRIO.

**CLÁUSULAS E CONDIÇÕES**

**1º – OBJETO:** Locação do apartamento #NUM\_APTO do edifício #EDIFICIO\_APTO situado na #RUA\_APTO, nº #NUM\_RUA\_APTO, na #CIDADE\_APTO, com direito a uma vaga na garagem do subsolo, exclusiva do imóvel.

**2º – DESTINAÇÃO:** O imóvel destina-se única e exclusivamente a fins residenciais do locatário e sua família.

**3º – PRAZO:** O prazo de locação será de 30 (trinta) meses, iniciando-se em #DATA\_INICIO e terminando em #DATA\_FIM, independentemente de qualquer aviso, notificação ou interrupção judicial.

Antes de começar a programar, vamos dar mais uma olhada no modelo de contrato no Word.

Como dito, cada local onde será inserida uma nova informação contém uma palavra-chave, iniciada por uma hashtag e por um nome, como se fosse o de uma variável.

É importante que você consiga identificar no seu texto padrão quais palavras deverão mudar de acordo com o novo locador e o que não irá mudar por ser um texto padrão.

Para o que for mudar, basicamente as informações de cada locador, criamos estas palavras-chave para serem substituídas após a execução do código.

# Módulo 13 – Integração com Word – Geração Automática de Contratos (Parte 1)

469

The screenshot shows a Microsoft Excel spreadsheet titled "Contratos.xlsxm - Excel". The table contains the following data:

	#NOME_LOCADOR	#NACIONALIDADE_LOCADOR	#PROFISSÃO_LOCADOR	#ESTADOCIVIL_LOCADOR	#RG_LOCADOR	#ENTIDADE_LOCADOR	#CPF_LOCADOR	#ENDERECO_LOCADOR	#NOM
1	Juão Paulo Rodrigues	Brasileiro	Engenheiro	Solteiro	28.451.873-4	DETRAN-RJ	154.021.548-80	Rua Decio Vilaes, 20, apto 103, Copacabana	Marília
2	Sergio Tranjan	Brasileiro	Publicitário	Casado	49.772.293-4	DETRAN-RJ	173.273.492-92	Rua Barão da Torre, 45, apto 1007, Ipanema	Cristina
3	Carolina Souza	Brasileira	Estudante	Noiva	38.283.283-7	DETRAN-SP	152.896.857-56	Rua Helena, 105, apto 53, Vila Olímpia	Rafael
4	Gabriel Pereira	Brasileiro	Estudante	Solteiro	12.424.274-42	DETRAN-RS	449.212.283-43	Rua Victor Silva, 52, apto 107, Vila Conceição	Maria
5	Alexandre Donati	Brasileiro	Jornalista	Casado	18.274.382-91	DETRAN-SP	183.183.481-52	Rua Caubi, 17, apto 907, Perdizes	Bernardo

Below the table is an orange button labeled "Gerar Contratos".

Estas palavras-chave também estão presentes no cabeçalho da tabela do Excel. Este padrão será muito importante para que seja possível que o código consiga preencher cada informação na sua devida posição no contrato em Word.

```
Sub gera_contratos()
    Set objeto_word = CreateObject("Word.Application")
    objeto_word.Visible = True
    Set arquivo_contrato = objeto_word.Documents.Open(ThisWorkbook.Path & "\Contrato Aluguel Padrão.docx")
|
End Sub
```

Começamos criando o nosso objeto que irá armazenar o aplicativo do Word.

Em seguida, o objeto que de fato receberá o arquivo modelo do Word, chamado **arquivo\_contrato**.

```
Sub gera_contratos()
    Set objeto_word = CreateObject("Word.Application")
    objeto_word.Visible = True
    Set arquivo_contrato = objeto_word.Documents.Open(ThisWorkbook.Path & "\Contrato Aluguel Padrão.docx")
    Set selecao = arquivo_contrato.Application.Selection
    selecao.Find.Text = "#NOME_LOCADOR"
    selecao.Find.Replacement.Text = "João Paulo"
    selecao.Find.Execute Replace:=wdReplaceAll
End Sub
```



**Por enquanto, você deverá fechar o arquivo com o modelo de contrato do Word antes de executar o código acima, pois caso o contrato esteja aberto, a macro entrará em conflito, pois ela tentará abrir um arquivo que já está aberto. Mais para frente vamos corrigir essa questão.**

Podemos começar fazendo um teste simples para ver o funcionamento correto.

Primeiro definimos uma variável chamada seleção que basicamente irá armazenar a posição do cursor do mouse no arquivo Word.

Em seguida, as 3 linhas de código ao lado serão responsáveis por:

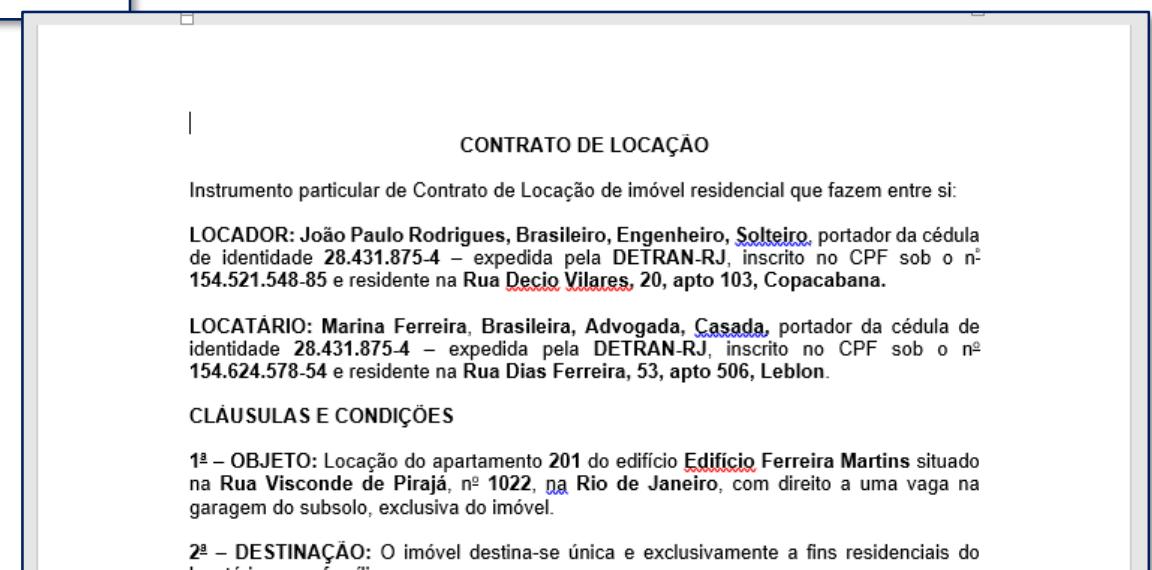
1. Encontrar o texto correspondente à palavra-chave.
2. Informar o texto que irá substituir aquele encontrado.
3. Informar que todas as vezes que o texto for encontrado, ele deverá ser substituído.

```
Sub gera_contratos()
    Set objeto_word = CreateObject("Word.Application")
    objeto_word.Visible = True
    Set arquivo_contrato = objeto_word.Documents.Open(ThisWorkbook.Path & "\Contrato Aluguel Padrão.docx")
    Set selecao = arquivo_contrato.Application.Selection
    ult_coluna = Range("A1").End(xlToRight).Column
    For coluna = 1 To ult_coluna
        selecao.Find.Text = Cells(1, coluna).Value
        selecao.Find.Replacement.Text = Cells(2, coluna).Value
        selecao.Find.Execute Replace:=wdReplaceAll
    Next
End Sub
```

A variável **ult\_coluna** será responsável por encontrar a última coluna que possui informações a serem adicionadas. Será necessário realizar um Loop de repetição para cada informação ao longo das colunas. O código acima substitui as informações apenas para o João Paulo Rodrigues, que está na **linha 2** da tabela do Excel. Ainda precisamos generalizar para todos os nomes da tabela.

O resultado final para o João é mostrado na imagem ao lado e você pode ver que todas as informações são substituídas corretamente.

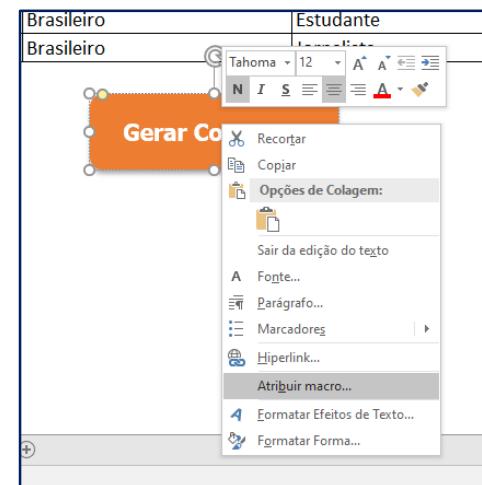
Entendida a lógica do que queremos fazer, vamos começar generalizando para o primeiro locador (João) o texto procurado e o texto de substituição para considerar todos os nomes da tabela do Excel. Cada texto com as informações do João usado para substituir está distribuído ao longo da linha 1 da tabela.



```
Sub gera_contratos()
    Set objeto_word = CreateObject("Word.Application")
    objeto_word.Visible = True
    ult_linha = Range("A1").End(xlDown).Row
    For linha = 2 To ult_linha
        Set arquivo_contrato = objeto_word.Documents.Open(ThisWorkbook.Path & "\Contrato Aluguel Padrão.docx")
        Set selecao = arquivo_contrato.Application.Selection
        ult_coluna = Range("A1").End(xlToLeft).Column
        For coluna = 1 To ult_coluna
            selecao.Find.Text = Cells(1, coluna).Value
            selecao.Find.Replacement.Text = Cells(linha, coluna).Value
            selecao.Find.Execute Replace:=wdReplaceAll
        Next
        arquivo_contrato.SaveAs2 (ThisWorkbook.Path & "\Contratos\Contrato de Locação " & Cells(linha, 1).Value & ".docx")
    Next
End Sub
```

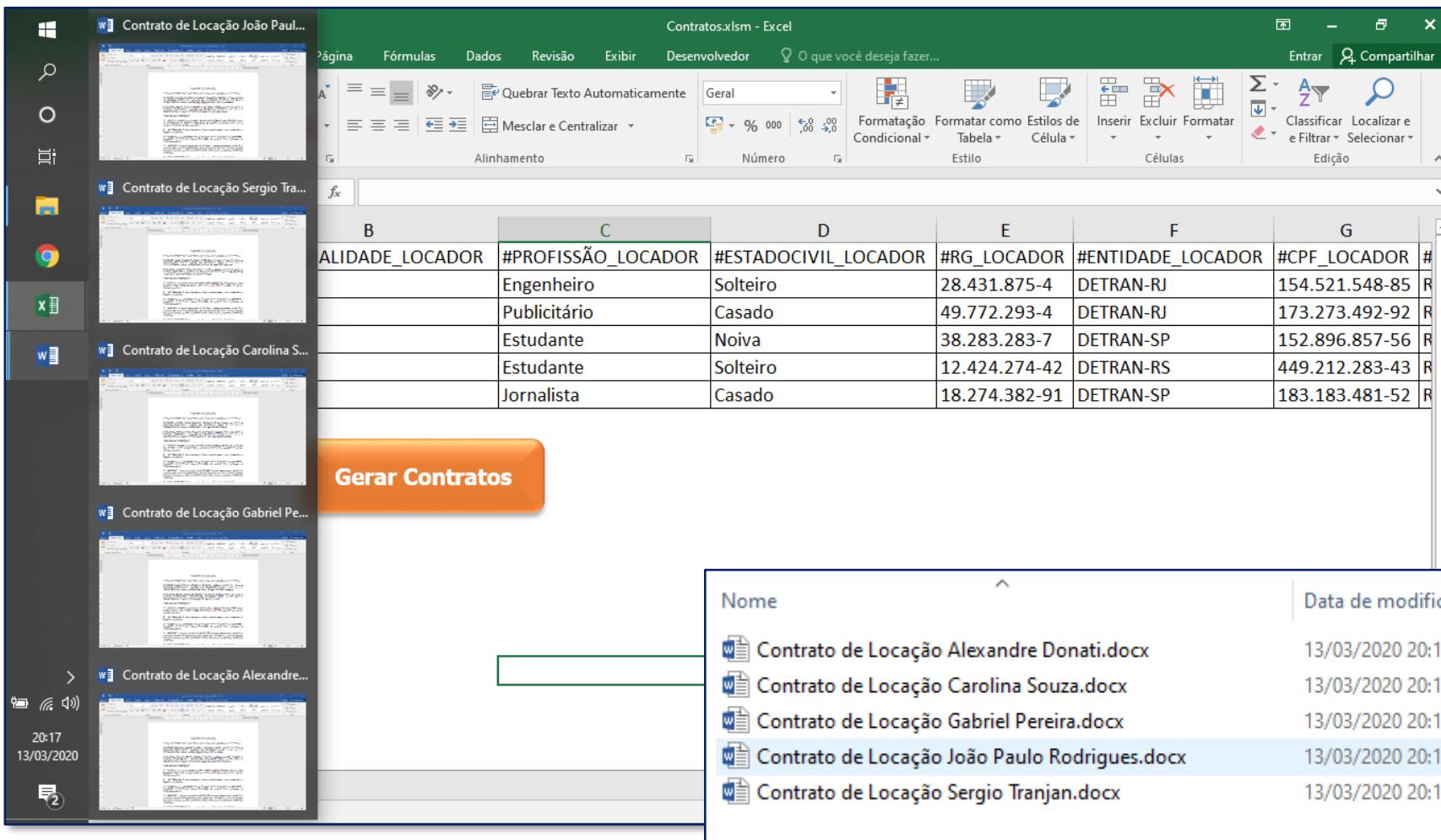
Lembre-se de que os arquivos de contrato devem ser salvos na pasta **Contratos**. Portanto, é necessário incluir o nome da pasta onde o arquivo será salvo.

O código modificado para funcionar para todos os nomes da tabela é mostrado ao lado. Foi criado um loop a mais e uma variável linha para descobrir a última linha preenchida na tabela, para que seja possível variar a linha para cada nome e criar um contrato diferente a cada loop. Ao fim também é utilizado o comando SaveAs2 para salvar cada arquivo de contrato criado. Lembre-se de atribuir a macro a um botão para executá-la facilmente.



# Módulo 13 – Integração com Word – Geração Automática de Contratos (Parte 5)

475



The screenshot shows a Microsoft Excel spreadsheet titled "Contratos.xlsxm - Excel". The spreadsheet contains a table with columns labeled B through G. Column B is "ALIDADE\_LOCADOR", column C is "#PROFISSÃO\_LOCADOR", column D is "#ESTADOCIVIL\_LOCADOR", column E is "#RG\_LOCADOR", column F is "#ENTIDADE\_LOCADOR", and column G is "#CPF\_LOCADOR". The data includes rows for Engenheiro, Solteiro, 28.431.875-4, DETRAN-RJ, 154.521.548-85, R; Publicitário, Casado, 49.772.293-4, DETRAN-RJ, 173.273.492-92, R; Estudante, Noiva, 38.283.283-7, DETRAN-SP, 152.896.857-56, R; Estudante, Solteiro, 12.424.274-42, DETRAN-RS, 449.212.283-43, R; and Jornalista, Casado, 18.274.382-91, DETRAN-SP, 183.183.481-52, R.

A large orange button labeled "Gerar Contratos" is positioned below the table. In the bottom right corner of the Excel window, there is a callout bubble containing a table of five generated Word documents:

Nome	Data de modificação	Tipo	Tamanho
Contrato de Locação Alexandre Donati.docx	13/03/2020 20:16	Documento do Mi...	18 KB
Contrato de Locação Carolina Souza.docx	13/03/2020 20:16	Documento do Mi...	18 KB
Contrato de Locação Gabriel Pereira.docx	13/03/2020 20:16	Documento do Mi...	18 KB
Contrato de Locação João Paulo Rodrigues.docx	13/03/2020 20:16	Documento do Mi...	18 KB
Contrato de Locação Sergio Tranjan.docx	13/03/2020 20:16	Documento do Mi...	18 KB

Ao executar o código, vemos que todos os arquivos de contrato são criados. Porém, todos eles ficam abertos. A ideia é fechá-los logo após a execução.

```
Sub gera_contratos()

Set objeto_word = CreateObject("Word.Application")
objeto_word.Visible = True
ult_linha = Range("A1").End(xlDown).Row
For linha = 2 To ult_linha
    Set arquivo_contrato = objeto_word.Documents.Open(ThisWorkbook.Path & "\Contrato Aluguel Padrão.docx")
    Set selecao = arquivo_contrato.Application.Selection
    ult_coluna = Range("A1").End(xlToRight).Column
    For coluna = 1 To ult_coluna
        selecao.Find.Text = Cells(1, coluna).Value
        selecao.Find.Replacement.Text = Cells(linha, coluna).Value
        selecao.Find.Execute Replace:=wdReplaceAll
    Next
    arquivo_contrato.SaveAs2 (ThisWorkbook.Path & "\Contratos\Contrato de Locação " & Cells(linha, 1).Value & ".docx")
    Cells(ult_linha + 2, 1).Value = "Progresso: " & (linha - 1) & "/" & (ult_linha - 1)
Next
objeto_word.Quit
MsgBox ("Contratos gerados com sucesso")
End Sub
```

Para fechar todos os arquivos abertos, podemos simplesmente usar o comando `Quit` para fechar a aplicação, que é representada no nosso código pela variável `objeto_word`.

Duas otimizações que podemos fazer são as seguintes:

1. Escrever na planilha um texto informando o progresso de criação dos contratos, uma espécie de Progresso: 3/5 para que o usuário possa saber que o código está sendo executado.
2. Exibir uma mensagem para informar que a macro foi finalizada com sucesso.

# Módulo 13 – Integração com Word – Geração Automática de Contratos (Parte 6)

477

Screenshot of Microsoft Excel showing the generation of automatic contracts. The spreadsheet contains data for locators and their details. An orange button labeled "Gerar Contratos" is visible, and a confirmation dialog box says "Contratos gerados com sucesso".

	A	B	C	D	E	F	G
1	#NOME_LOCADOR	#NACIONALIDADE_LOCADOR	#PROFISSÃO_LOCADOR	#ESTADOCIVIL_LOCADOR	#RG_LOCADOR	#ENTIDADE_LOCADOR	Progresso: 5/5
2	João Paulo Rodrigues	Brasileiro	Engenheiro	Solteiro	28.431.875-4	DETAN-RJ	154.521.548-85
3	Sergio Tranjan	Brasileiro	Publicitário	Casado	49.772.293-4	DETAN-RJ	173.273.492-92
4	Carolina Souza	Brasileira	Estudante	Noiva	38.283.283-7	DETAN-SP	152.896.857-56
5	Gabriel Pereira	Brasileiro	Estudante	Solteiro	12.424.274-42	DETAN-RS	449.212.283-43
6	Alexandre Donati	Brasileiro	Jornalista		18.274.382-91	DETAN-SP	183.183.481-52
7							
8	Progresso: 5/5						
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							

Planilha1

The screenshot shows a Microsoft Excel window titled "Vendas.xlsxm - Excel". The ribbon menu is visible at the top. In the center, there is a table with data. A black button labeled "Gerar Relatório" is overlaid on the first row of the table. The table has columns for months (Setembro, 2019, Mês, 2017, 2018, 2019) and rows for product categories (Celular, Tablet, Notebook, Relogio, Televisão) and a total row. Below the table, there are four empty rows labeled 16 through 19, each containing a VBA code snippet:

```
#NOME_MES  
#ANO  
#FATURAMENTO  
#SITUACAO_MES
```

The status bar at the bottom shows "Base Vendas" and "130%".

Vamos agora concluir com o último exercício de integração com o Word, e também um dos mais completos que já fizemos agora.

**Ao final deste exercício você será capaz de construir um código de geração de relatórios em Word de uma maneira 100% automática e integrada com o Excel.**

A criação de relatórios é uma das tarefas mais comuns no dia a dia de uma empresa. A maioria das pessoas, no entanto, não tem conhecimento de VBA e demoram horas e horas para executar esta tarefa que faremos em segundos.

## Módulo 13 – Integração com Word – Criação de Relatório Automático (Explicação)

479

The screenshot shows a Microsoft Excel interface with the following details:

- Worksheet:** Vendas.xlsxm - Excel
- Toolbar:** Arquivo, Página Inicial, Inserir, Layout da Página, Fórmulas, Dados, Revisão, Exibir, Desenvolvedor, Entrar, Compartilhar.
- Fonte:** Calibri, Size 11, Bold, Italic, Underline, Font Color.
- Alinhamento:** Quebrar Texto Automaticamente, Mesclar e Centralizar.
- Número:** Geral, % 000, €, £, \$.
- Estilo:** Formatação Condicional, Formatar como Tabela, Estilos de Célula.
- Células:** Inserir, Excluir, Formatar.
- Edição:** Classificar e Filtrar, Localizar e Selecionar.

**Data Tables:**

	B	C	D	E	F	G	H	I	J	K	L	M
1	Setembro	2019	Mês	2017	2018	2019						
2	Celular	R\$ 30.361	Janeiro	R\$ 24.400	R\$ 21.472	R\$ 23.190						
3	Tablet	R\$ 6.665	Fevereiro	R\$ 21.000	R\$ 23.310	R\$ 32.634						
4	Notebook	R\$ 17.032	Março	R\$ 25.300	R\$ 24.288	R\$ 27.203						
5	Relogio	R\$ 5.184	Abril	R\$ 30.700	R\$ 38.068	R\$ 45.682						
6	Televisão	R\$ 14.810	Maio	R\$ 17.000	R\$ 15.810	R\$ 20.079						
7	Total:	R\$ 74.050	Junho	R\$ 38.300	R\$ 39.832	R\$ 50.985						
8			Julho	R\$ 52.800	R\$ 53.328	R\$ 46.395						
9			Agosto	R\$ 37.400	R\$ 46.750	R\$ 68.255						
10			Setembro	R\$ 47.600	R\$ 50.932	R\$ 74.050						
11			Outubro	R\$ 55.900	R\$ 76.583							
12			Novembro	R\$ 60.404	R\$ 72.485							
13			Dezembro	R\$ 82.800	R\$ 103.500							
14												
15												
16	#NOME_MES											
17	#ANO											
18	#FATURAMENTO											
19	#SITUACAO_MES											

**Buttons:**

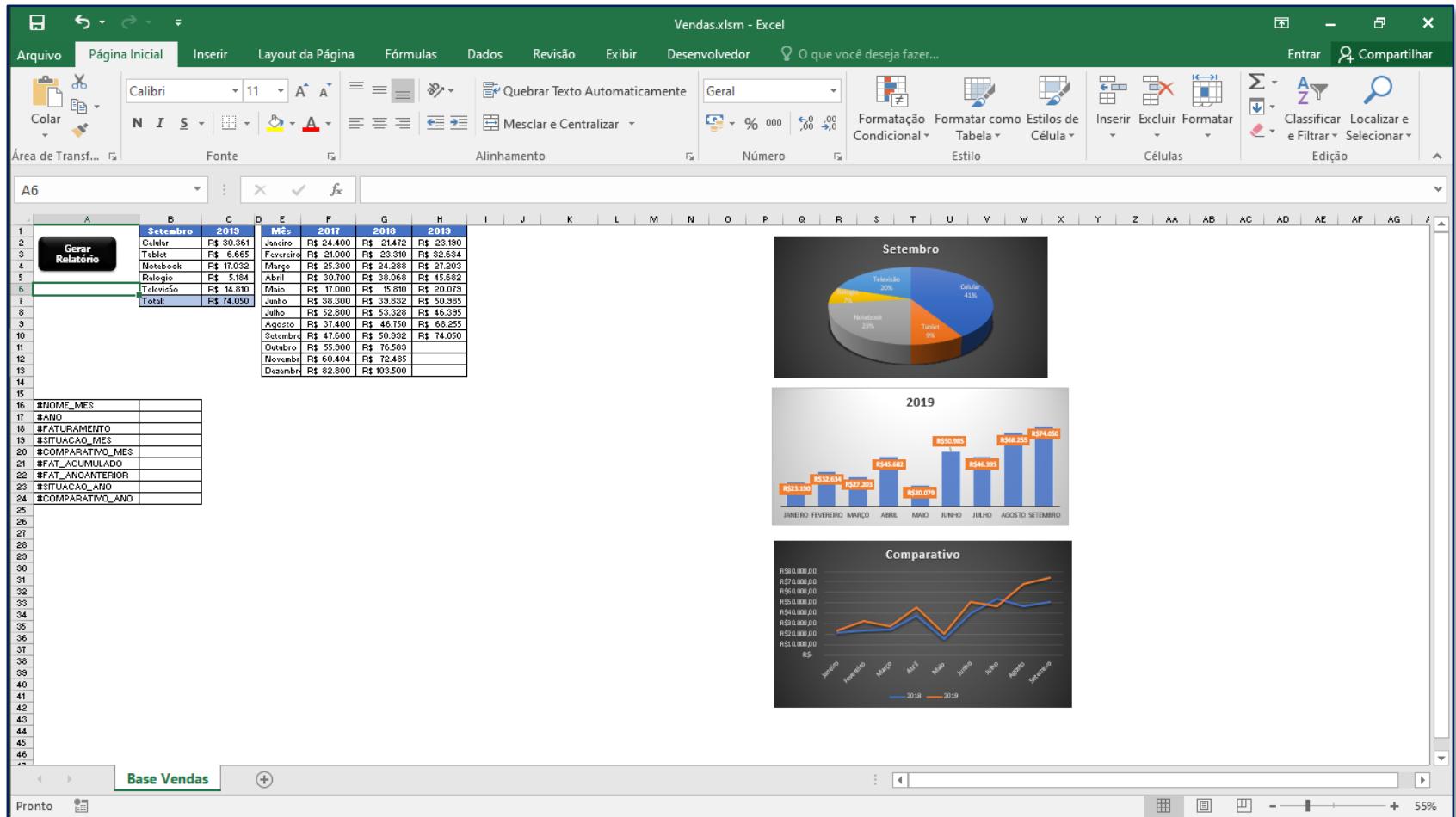
- Gerar Relatório (in a black button)

**Formulas:**

- Base Vendas

No arquivo ao lado temos uma tabela com os totais de faturamento para cada mês, em 3 anos distintos. Além disso, temos também outra tabela, nas colunas B e C, que mostra a composição do faturamento total de acordo com as vendas de cada um dos 5 produtos.

O objetivo é que a gente escolha um mês e um ano de análise, e gere um relatório com diversos indicadores que veremos mais a frente.



Mais a esquerda, no mesmo arquivo, temos também 3 gráficos, que mostram as seguintes informações:

- Gráfico de pizza:** composição do faturamento total para o mês de análise.
- Gráfico de colunas:** faturamento mensal para o ano de análise selecionado, de janeiro até o mês também selecionado na célula B1.
- Gráfico de linhas:** comparativo entre o faturamento dos meses do ano de análise e o faturamento dos mesmos meses do ano anterior.

The screenshot shows a Microsoft Word document titled "Relatório Mensal.docx". The ribbon menu is visible at the top, showing tabs like Arquivo, Página Inicial, Inserir, Design, Layout, Referências, Correspondências, Revisão, Exibir, and uma barra de busca. The "Página Inicial" tab is selected. The main content area contains the following text:

**Relatório de Vendas #NOME\_MES/#ANO**

O mês de #NOME\_MES fechou com #FATURAMENTO de faturamento, o que representa #SITUACAO\_MES com relação ao mês anterior no valor de #COMPARATIVO\_MES.

A proporção de faturamento entre os produtos da empresa foi a seguinte:  
#GRAFICO1

Até o mês de #NOME\_MES a empresa acumulou um faturamento total de #FAT\_ACUMULADO no ano, distribuído da seguinte forma:  
#GRAFICO2

Com relação ao ano anterior, no qual a empresa obteve um faturamento acumulado até o mês de #NOME\_MES totalizando #FAT\_ANOANTERIOR, houve #SITUACAO\_ANO de #COMPARATIVO\_ANO.

Página 1 de 1 95 palavras

O modelo do relatório em Word que será criado está mostrado ao lado. Mais uma vez, temos diversas palavras-chave, onde basicamente faremos as substituições das informações variáveis daquele mês.

Além disso, temos também as palavras-chaves identificando a posição no texto onde serão colados os gráficos, cuja estrutura é: **#GRAFICO**.

**Relatório de Vendas Setembro/2019**

O mês de **Setembro** fechou com R\$ 74.050,00 de faturamento, o que representa um aumento com relação ao mês anterior no valor de R\$ 5.795,00.

A proporção de faturamento entre os produtos da empresa foi a seguinte:

The pie chart illustrates the sales distribution across four categories: Celular (41%), Notebook (28%), Telefone (20%), and Outros (11%).

Categoria	Porcentagem
Celular	41%
Notebook	28%
Telefone	20%
Outros	11%

Até o mês de **Setembro**, a empresa acumulou um faturamento total de R\$ 388.471,94 no ano, distribuído da seguinte forma:

The bar chart displays monthly sales volumes for 2019, with specific values labeled above each bar.

Mês	Faturamento (R\$)
JANEIRO	313.190
FEVEREIRO	332.834
MARÇO	352.703
ABRIL	365.682
MAIO	320.679
JUNHO	370.181
JULHO	340.195
AGOSTO	368.225
SETEMBRO	374.050

Com relação ao ano anterior, no qual a empresa obteve um faturamento acumulado até o mês de **Setembro** totalizando R\$ 313.790,00, houve um crescimento de R\$ 74.681,94.

O comparativo com relação ao ano anterior está indicado no gráfico

The line graph compares monthly sales volumes between 2018 (blue line) and 2019 (orange line) from January to September. The Y-axis represents sales in R\$.

Mês	2018 (R\$)	2019 (R\$)
JANEIRO	313.190	332.834
FEVEREIRO	332.834	352.703
MARÇO	352.703	365.682
ABRIL	365.682	320.679
MAIO	320.679	370.181
JUNHO	370.181	340.195
JULHO	340.195	368.225
AGOSTO	368.225	374.050
SETEMBRO	374.050	-

Neste relatório queremos basicamente fazer uma análise comparativa do faturamento, para que seja possível avaliar se a empresa está tendo um crescimento ou decrescimento.

Ao lado temos um exemplo já pronto de como queremos preencher o modelo do relatório, para o mês de **setembro de 2019**.

Nome	Data de modificação	Tipo	Tamanho
 Relatórios	08/01/2020 13:41	Pasta de arquivos	
 Relatório Mensal.docx	18/12/2019 09:02	Documento do Mi...	14 KB
 Vendas.xlsxm	18/12/2019 09:21	Planilha Habilitad...	28 KB

Além de criar estes relatórios, queremos também salvá-los como PDF na pasta **Relatórios**.

Relatório de Vendas Setembro/2019

O mês de Setembro fechou com R\$ 74.050,00 de faturamento, o que representa um aumento com relação ao mês anterior no valor de R\$ 5.795,00.

A proporção de faturamento entre os produtos da empresa foi a seguinte:

**Setembro**

Categoria	Porcentagem
Televisão	20%
Computador	30%
Tablet	30%
Celular	20%

Até o mês de Setembro a empresa acumulou um faturamento total de R\$ 388.471,94 no ano, distribuído da seguinte forma:

**2019**

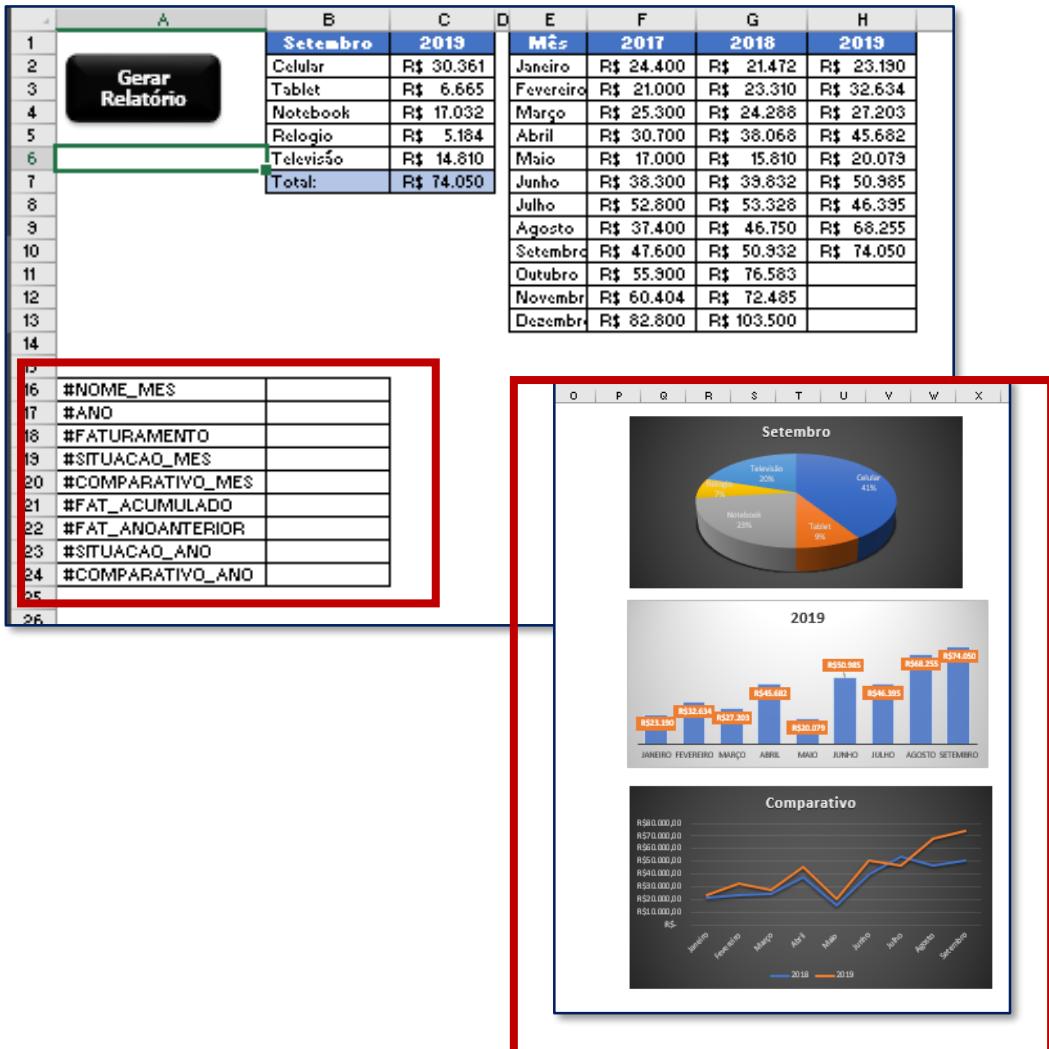
Mês	Faturamento (R\$)
JANEIRO	R\$ 61.850
FEVEREIRO	R\$ 62.350
MARÇO	R\$ 67.050
ABRIL	R\$ 65.642
MAYO	R\$ 60.079
JUNHO	R\$ 60.095
JULHO	R\$ 60.195
AGOSTO	R\$ 61.253
SETEMBRO	R\$ 74.050

Com relação ao ano anterior, no qual a empresa obteve um faturamento acumulado até o mês de Setembro totalizando R\$ 313.790,00, houve um crescimento de R\$ 74.681,94.

Sendo este o resultado final esperado.

Com este conhecimento, sem dúvidas o seu nível de VBA será elevado a outro patamar, e o seu trabalho será muito mais otimizado e eficiente, chamando muito a atenção de qualquer um que trabalhe com você, especialmente o seu chefe!

A partir da próxima página vamos começar os trabalhos.



Basicamente, as informações que devemos copiar da nossa planilha Excel e coladas no arquivo Word estão destacadas ao lado. Abaixo, descrevemos o que significa cada informação:

1. **#NOME\_MES**: mês de análise (célula B1).
2. **#ANO**: ano de análise (célula C1).
3. **#FATURAMENTO**: faturamento do mês/ano.
4. **#SITUACAO\_MES**: indica se teve “um aumento” ou “uma queda” do faturamento em relação ao mês anterior.
5. **#COMPARATIVO\_MES**: diferença, em R\$, entre os faturamentos do mês atual e do mês anterior.
6. **#FAT\_ACUMULADO**: soma do faturamento acumulado no ano de análise até o mês de análise.
7. **#FAT\_ANOANTERIOR**: soma do faturamento acumulado de janeiro até o mês de análise, só que do ano anterior ao de análise.
8. **#SITUACAO\_ANO**: indica se teve “um aumento” ou “uma queda” do faturamento em relação ao ano anterior.
9. **#COMPARATIVO\_ANO**: diferença, em R\$, do faturamento acumulado atual e o faturamento do ano anterior

The screenshot shows a Microsoft Word document titled "Relatório Mensal.docx". The ribbon menu is visible at the top, showing tabs like Arquivo, Página Inicial, Design, Layout, etc. The main content area contains the following text:

**Relatório de Vendas #NOME\_MES/#ANO**

O mês de #NOME\_MES fechou com #FATURAMENTO de faturamento, o que representa #SITUACAO\_MES com relação ao mês anterior no valor de #COMPARATIVO\_MES.

A proporção de faturamento entre os produtos da empresa foi a seguinte:

#GRAFICO1

Até o mês de #NOME\_MES a empresa acumulou um faturamento total de #FAT\_ACUMULADO no ano, distribuído da seguinte forma:

#GRAFICO2

Com relação ao ano anterior, no qual a empresa obteve um faturamento acumulado até o mês de #NOME\_MES totalizando #FAT\_ANOANTERIOR, houve #SITUACAO\_ANO de #COMPARATIVO\_ANO.

Página 1 de 1 95 palavras

Cada um dos indicadores da página anterior serão devidamente preenchidos nos respectivos lugares do arquivo Word.

```
Sub gera_relatorio()
    Set objeto_word = CreateObject("Word.Application")
    objeto_word.Visible = True
    Set arquivo_relatorio = objeto_word.documents.Open(ThisWorkbook.Path & "\Relatório Mensal.docx")
    Set selecao = arquivo_relatorio.Application.Selection
    |
End Sub
```

Primeiro começamos declarando o nosso objeto Word, o arquivo que iremos manipular e também a variável **selecao** para que seja possível posicionar o cursor nos locais adequados no arquivo Word.

The screenshot shows a Microsoft Excel window titled "Vendas.xlsxm - Excel". The ribbon menu includes Arquivo, Página Inicial, Inserir, Layout da Página, Fórmulas, Dados, Revisão, Exibir, Desenvolvedor, and Ajuda. The "Página Inicial" tab is selected. The "Fonte" group contains buttons for Calibri, 11pt, bold, italic, underline, and font color. The "Alinhamento" group contains buttons for center, right, and left alignment, and for horizontal and vertical text orientation. The "Número" group contains buttons for decimal separator, thousands separator, and currency symbols. The "Área de Transferência" button is visible. The main content area displays a table with data for September 2019 and a larger table for monthly sales from January to December 2017-2019. A black button labeled "Gerar Relatório" is overlaid on the table. Below the tables, rows 16 to 25 show code snippets: #NOME\_MES, #ANO, #FATURAMENTO, #SITUACAO\_MES, #COMPARATIVO\_MES, #FAT\_ACUMULADO, #FAT\_ANOANTERIOR, #SITUACAO\_ANO, and #COMPARATIVO\_ANO. The bottom tabs include "Base Vendas" and a plus sign tab.

Em seguida, vamos salvar em duas variáveis as informações de mês e ano, que estão nas células B1 e C1, respectivamente.

Além disso, devemos armazenar o valor do faturamento para aquele ano, que no caso vai estar na tabela de meses/anos.

A lógica de preenchimento da planilha é a seguinte:

O usuário irá preencher o valor do faturamento na tabela da direita e também na célula C7, de total. Os faturamentos de cada produto são preenchidos automaticamente de acordo com uma fórmula que calcula a proporção de cada um em relação ao total.

Portanto, teremos que conseguir encontrar, na tabela da direita, o faturamento correto, de acordo com o mês e ano selecionado.

```
Sub gera_relatorio()
    Set objeto_word = CreateObject("Word.Application")
    objeto_word.Visible = True
    Set arquivo_relatorio = objeto_word.documents.Open(ThisWorkbook.Path & "\Relatório Mensal.docx")
    Set selecao = arquivo_relatorio.Application.Selection
    mes = Cells(1, 2).Value
    ano = Cells(1, 3).Value
    Range("B16").Value = mes
    Range("B17").Value = ano
    linha_mes = Range("E:E").Find(mes).Row
    coluna_ano = Range("F1:XFD1").Find(ano).Column
    faturamento = Cells(linha_mes, coluna_ano).Value
    Range("B18").Value = faturamento
End Sub
```

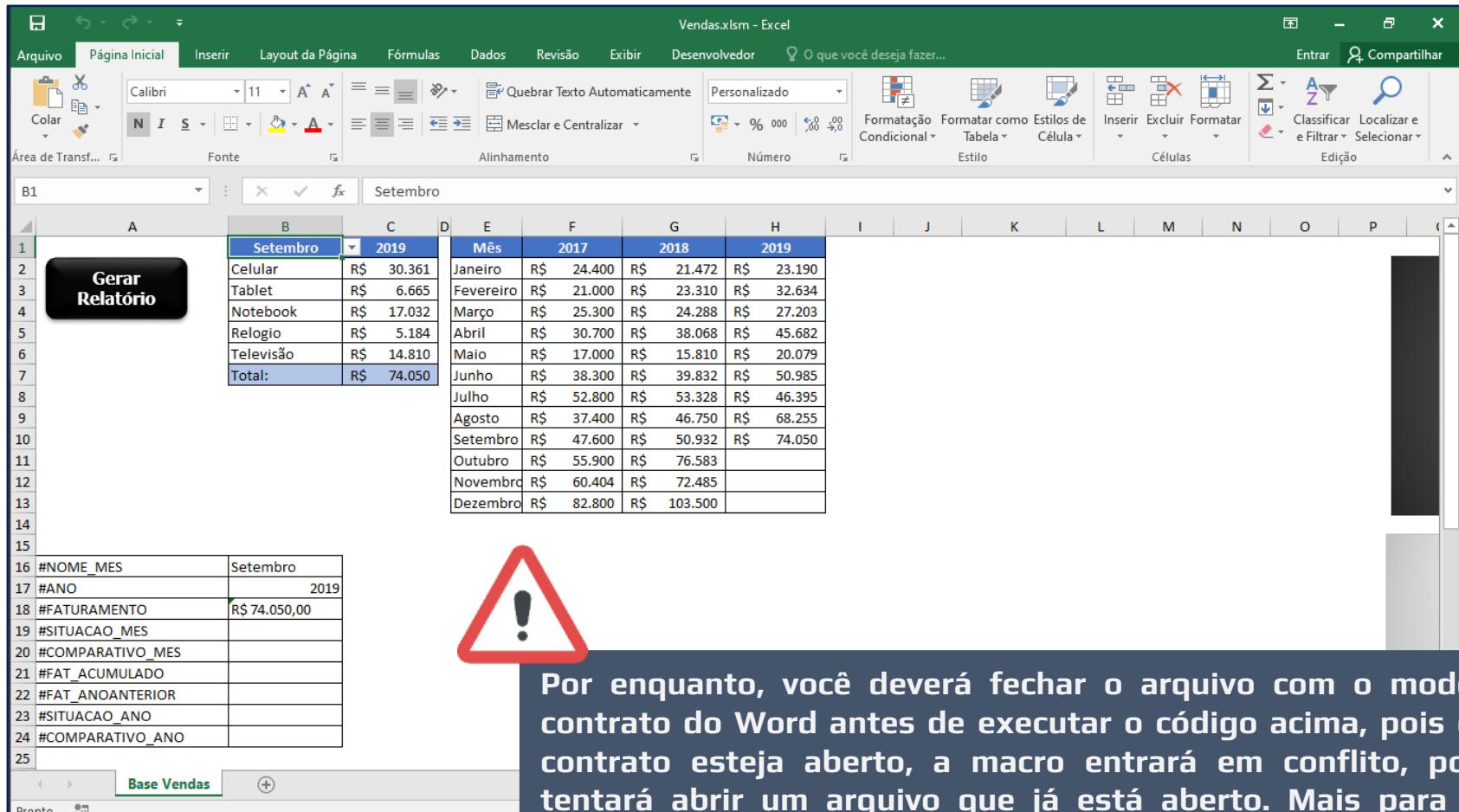
Para encontrar a posição do mês na tabela da direita, não podemos usar simplesmente o comando **Cells.Find**, pois temos o mês aparecendo mais de uma vez. Para encontrar o mês correto, procuramos especificamente na coluna E e armazenamos em uma variável chamada **linha\_mes**.

Já para o ano, a lógica é a mesma. Como o ano se repete, devemos na verdade procurar, na linha 1, apenas a partir da coluna F. Para não precisarmos nos preocupar com a quantidade de anos, nosso limite será a coluna XFD, que é a última coluna do Excel. O número da coluna é armazenado na variável **coluna\_ano**.

Por fim, escrevemos nas células B16, B17 e B18 os valores de **mes**, **ano** e **faturamento**, respectivamente.

```
Sub gera_relatorio()
    Set objeto_word = CreateObject("Word.Application")
    objeto_word.Visible = True
    Set arquivo_relatorio = objeto_word.documents.Open(ThisWorkbook.Path & "\Relatório Mensal.docx")
    Set selecao = arquivo_relatorio.Application.Selection
    mes = Cells(1, 2).Value
    ano = Cells(1, 3).Value
    Range("B16").Value = mes
    Range("B17").Value = ano
    linha_mes = Range("E:E").Find(mes).Row
    coluna_ano = Range("F1:XFD1").Find(ano).Column
    faturamento = Cells(linha_mes, coluna_ano).Value
    Range("B18").Value = Format(faturamento, "Currency")
    |
End Sub
```

Apenas um detalhe: o valor de faturamento precisa ter uma formatação de moeda para que, ao colarmos este valor no modelo de relatório em Word, esta formatação também seja levada. Para isso, podemos utilizar a função Format que já conhecemos, e formatar como “Currency”.



The screenshot shows an Excel spreadsheet titled "Vendas.xlsxm - Excel". The main table (rows 1-15) contains sales data for various products across three years (2017, 2018, 2019). A summary row (row 16) uses formulas to calculate the total for each column. A button labeled "Gerar Relatório" is visible on the left. A red warning triangle icon with an exclamation mark is overlaid on the screen, pointing towards the bottom right.

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1		Setembro	2019		Mês	2017	2018	2019							
2		Celular	R\$ 30.361	Janeiro	R\$ 24.400	R\$ 21.472	R\$ 23.190								
3		Tablet	R\$ 6.665	Fevereiro	R\$ 21.000	R\$ 23.310	R\$ 32.634								
4		Notebook	R\$ 17.032	Março	R\$ 25.300	R\$ 24.288	R\$ 27.203								
5		Relogio	R\$ 5.184	Abril	R\$ 30.700	R\$ 38.068	R\$ 45.682								
6		Televisão	R\$ 14.810	Maio	R\$ 17.000	R\$ 15.810	R\$ 20.079								
7		Total:	R\$ 74.050	Junho	R\$ 38.300	R\$ 39.832	R\$ 50.985								
8				Julho	R\$ 52.800	R\$ 53.328	R\$ 46.395								
9				Agosto	R\$ 37.400	R\$ 46.750	R\$ 68.255								
10				Setembro	R\$ 47.600	R\$ 50.932	R\$ 74.050								
11				Outubro	R\$ 55.900	R\$ 76.583									
12				Novembrc	R\$ 60.404	R\$ 72.485									
13				Dezembro	R\$ 82.800	R\$ 103.500									
14															
15															
16	#NOME_MES	Setembro													
17	#ANO		2019												
18	#FATURAMENTO		R\$ 74.050,00												
19	#SITUACAO_MES														
20	#COMPARATIVO_MES														
21	#FAT_ACUMULADO														
22	#FAT_ANOANTERIOR														
23	#SITUACAO_ANO														
24	#COMPARATIVO_ANO														
25															

Por enquanto, você deverá fechar o arquivo com o modelo de contrato do Word antes de executar o código acima, pois caso o contrato esteja aberto, a macro entrará em conflito, pois ela tentará abrir um arquivo que já está aberto. Mais para frente vamos corrigir essa questão.

Você já pode executar o código anterior para ver o funcionamento. Por enquanto, ele apenas abrirá o arquivo Word e escreverá o valor das 3 variáveis que criamos em cada uma das células da tabela auxiliar a partir da linha 16.

```
Sub gera_relatorio()

Set objeto_word = CreateObject("Word.Application")
objeto_word.Visible = True

Set arquivo_relatorio = objeto_word.documents.Open(ThisWorkbook.Path & "\Relatório Mensal.docx")

Set selecao = arquivo_relatorio.Application.Selection

mes = Cells(1, 2).Value
ano = Cells(1, 3).Value

Range("B16").Value = mes
Range("B17").Value = ano

linha_mes = Range("E:E").Find(mes).Row
coluna_ano = Range("F1:XF1").Find(ano).Column

faturamento = Cells(linha_mes, coluna_ano).Value

Range("B18").Value = Format(faturamento, "Currency")

If mes <> "Janeiro" Then
    faturamento_mesanterior = Cells(linha_mes - 1, coluna_ano).Value
Else
    faturamento_mesanterior = Cells(13, coluna_ano - 1).Value
End If

If faturamento >= faturamento_mesanterior Then
    Range("B19").Value = "um aumento"
    Range("B20").Value = Format(faturamento - faturamento_mesanterior, "Currency")
Else
    Range("B19").Value = "uma diminuição"
    Range("B20").Value = Format(faturamento_mesanterior - faturamento, "Currency")
End If

End Sub
```

O próximo passo é criar a lógica para preencher os campos de #SITUACAO\_MES e #COMPARATIVO\_MES. Um único detalhe que temos que nos atentar aqui é que o comparativo do mês anterior depende da informação de faturamento do mês anterior. Porém, temos duas situações para conseguir encontrar o faturamento do mês anterior:

1. Se o mês for de fevereiro até dezembro, o faturamento do mês anterior é simplesmente o valor que está na linha imediatamente acima da **linha\_mes**.
2. Porém, caso o mês de análise seja janeiro, então o valor do faturamento do mês anterior não estará na linha acima, e sim na coluna da esquerda (coluna\_ano - 1), na linha 13 (pois dezembro sempre está na linha 13).

Por isso, criamos uma condição If para tratar estes dois casos.

```
Sub gera_relatorio()

Set objeto_word = CreateObject("Word.Application")
objeto_word.Visible = True

Set arquivo_relatorio = objeto_word.documents.Open(ThisWorkbook.Path & "\Relatório Mensal.docx")

Set selecao = arquivo_relatorio.Application.Selection

mes = Cells(1, 2).Value
ano = Cells(1, 3).Value

Range("B16").Value = mes
Range("B17").Value = ano

linha_mes = Range("E:E").Find(mes).Row
coluna_ano = Range("F1:XF1").Find(ano).Column

faturamento = Cells(linha_mes, coluna_ano).Value

Range("B18").Value = Format(faturamento, "Currency")

If mes <> "Janeiro" Then

    faturamento_mesanterior = Cells(linha_mes - 1, coluna_ano).Value

Else

    faturamento_mesanterior = Cells(13, coluna_ano - 1).Value

End If

If faturamento >= faturamento_mesanterior Then
    Range("B19").Value = "um aumento"
    Range("B20").Value = Format(faturamento - faturamento_mesanterior, "Currency")
Else
    Range("B19").Value = "uma diminuição"
    Range("B20").Value = Format(faturamento_mesanterior - faturamento, "Currency")
End If

End Sub
```

Também precisaremos fazer um *check* para verificar se o faturamento é maior ou menor que o do mês anterior. Dependendo deste teste, escreveremos nas células B19 e B20 “um aumento”/diferença entre os faturamentos, ou “uma queda”/diferença entre os faturamentos.

The screenshot shows a Microsoft Excel spreadsheet titled "Vendas.xlsxm - Excel". The ribbon menu is visible at the top, showing tabs like Arquivo, Página Inicial, Inserir, Layout da Página, Fórmulas, Dados, Revisão, Exibir, Desenvolvedor, and Entrar.

The main content consists of two parts:

- Summary Table:** A table in rows 1 to 7 showing sales by device type (Celular, Tablet, Notebook, Relogio, Televisao) and their total for September 2019. The total for September is R\$ 74.050.
- VBA Code:** A series of formulas starting from row 16, which generate the summary values. The formulas include:
  - #NOME\_MES: Setembro
  - #ANO: 2019
  - #FATURAMENTO: R\$ 74.050,00
  - #SITUACAO\_MES: um aumento
  - #COMPARATIVO\_MES: R\$ 5.795,00
  - #FAT\_ACUMULADO: (empty)
  - #FAT\_ANOANTERIOR: (empty)
  - #SITUACAO\_ANO: (empty)
  - #COMPARATIVO\_ANO: (empty)

The bottom of the screen shows the "Base Vendas" tab selected and the status bar indicating "Pronto" (Ready).

Você também já pode testar o código anterior.

**Antes de rodar o código, lembre que você deve fechar o arquivo Word aberto pois é o próprio código VBA que abrirá este arquivo para a gente.**

```
Range("B18").Value = Format(faturamento, "Currency")
If mes <> "Janeiro" Then
    faturamento_mesanterior = Cells(linha_mes - 1, coluna_ano).Value
Else
    faturamento_mesanterior = Cells(13, coluna_ano - 1).Value
End If
If faturamento >= faturamento_mesanterior Then
    Range("B19").Value = "um aumento"
    Range("B20").Value = Format(faturamento - faturamento_mesanterior, "Currency")
Else
    Range("B19").Value = "uma diminuição"
    Range("B20").Value = Format(faturamento_mesanterior - faturamento, "Currency")
End If

faturamento_acumulado = WorksheetFunction.Sum(Range(Cells(2, coluna_ano), Cells(linha_mes, coluna_ano)))
faturamento_acumulado_anoanterior = WorksheetFunction.Sum(Range(Cells(2, coluna_ano - 1), Cells(linha_mes, coluna_ano - 1)))

Range("B21").Value = Format(faturamento_acumulado, "Currency")
Range("B22").Value = Format(faturamento_acumulado_anoanterior, "Currency")
```

Dando continuidade, precisamos fazer a soma do acumulado do faturamento no mês de análise e do acumulado do faturamento do mês anterior.

Para isso, usaremos a fórmula SUM do Excel. Repare ao lado como ficou a estrutura.

Meio estranha, não?

Escrevemos o Range de uma maneira diferente do que vimos até agora. Não é nada demais, mas o que você precisa entender é que as duas formas abaixo são equivalentes.

```
Sub ExemploRange()
    Range ("A1:B10")
    Range(Cells(1,1), Cells(10, 2))
End Sub
```

Ou seja, podemos substituir o texto "A1:B10" dentro do Range pela sua representação na forma Cells. A célula A1 é representada por Cells(1, 1) e a célula B10 é representada por Cells(10, 2). **Repare que aqui não colocamos o Value porque não queremos o valor dentro destas células, e sim a célula propriamente dita (lembre-se do conceito do objeto célula).**

Assim, por meio de dois comandos Cells, especificamos dentro do Range a célula de início e a célula de fim. A vantagem de fazer dessa maneira é que temos a facilidade de manipular os números das linhas e colunas deste Range. Na nossa situação, tanto a linha é variável (de acordo com o mês de análise) quanto o ano é variável (de acordo com o ano de análise).

Assim, para conseguirmos fazer a soma do acumulado do mês, devemos ter um Range começando sempre da linha 2 (referente ao mês de janeiro) da coluna coluna\_ano (depende do ano de análise selecionado) até a célula do mês de análise, que está na linha **linha\_mes**, na coluna **coluna\_ano**. A mesma lógica vale para o faturamento acumulado do ano anterior, com a diferença de que a coluna do ano anterior ao ano de análise é a coluna número (coluna\_ano - 1), pois é a coluna à esquerda daquela do ano de análise.

```
faturamento_acumulado = WorksheetFunction.Sum(Range(Cells(2, coluna_ano), Cells(linha_mes, coluna_ano)))
faturamento_acumulado_anoanterior = WorksheetFunction.Sum(Range(Cells(2, coluna_ano - 1), Cells(linha_mes, coluna_ano - 1)))
```

Módulo 13 – Integração com Word – Criação de Relatório Automático (Resolução Parte 5)

498

Vendas.xlsx - Excel

Arquivo Página Inicial Inserir Layout da Página Fórmulas Dados Revisão Exibir Desenvolvedor O que você deseja fazer... Entrar Compartilhar

Calibri 11 A A Quebrar Texto Automaticamente Personalizado Formatação Condicional Mesclar e Centralizar Número Formatar como Tabela Estilos de Célula Inserir Excluir Formatar Células Edição

B1 : Setembro

**Gerar Relatório**

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Setembro	2019		Mês	2017	2018	2019								
2	Celular	R\$ 30.361		Janeiro	R\$ 24.400	R\$ 21.472	R\$ 23.190								
3	Tablet	R\$ 6.665		Fevereiro	R\$ 21.000	R\$ 23.310	R\$ 32.634								
4	Notebook	R\$ 17.032		Março	R\$ 25.300	R\$ 24.288	R\$ 27.203								
5	Relogio	R\$ 5.184		Abri	R\$ 30.700	R\$ 38.068	R\$ 45.682								
6	Televisão	R\$ 14.810		Maio	R\$ 17.000	R\$ 15.810	R\$ 20.079								
7	Total:	R\$ 74.050		Junho	R\$ 38.300	R\$ 39.832	R\$ 50.985								
8				Julho	R\$ 52.800	R\$ 53.328	R\$ 46.395								
9				Agosto	R\$ 37.400	R\$ 46.750	R\$ 68.255								
10				Setembro	R\$ 47.600	R\$ 50.932	R\$ 74.050								
11				Outubro	R\$ 55.900	R\$ 76.583									
12				Novembrc	R\$ 60.404	R\$ 72.485									
13				Dezembro	R\$ 82.800	R\$ 103.500									
14															
15															
16	#NOME_MES	Setembro													
17	#ANO	2019													
18	#FATURAMENTO	R\$ 74.050,00													
19	#SITUACAO_MES	um aumento													
20	#COMPARATIVO_MES	R\$ 5.795,00													
21	#FAT_ACUMULADO	R\$ 388.471,94													
22	#FAT_ANOANTERIOR	R\$ 313.790,00													
23	#SITUACAO_ANO														
24	#COMPARATIVO_ANO														
25															

Base Vendas

Pronto

Mais uma vez já podemos executar a macro e visualizar o resultado parcial.

```
If faturamento >= faturamento_mesanterior Then
    Range("B19").Value = "um aumento"
    Range("B20").Value = Format(faturamento - faturamento_mesanterior, "Currency")
Else
    Range("B19").Value = "uma diminuição"
    Range("B20").Value = Format(faturamento_mesanterior - faturamento, "Currency")
End If

faturamento_acumulado = WorksheetFunction.Sum(Range(Cells(2, coluna_ano), Cells(linha_mes, coluna_ano)))

If ano > 2017 Then
    faturamento_acumulado_anoanterior = WorksheetFunction.Sum(Range(Cells(2, coluna_ano - 1), Cells(linha_mes, coluna_ano - 1)))
Else
    faturamento_acumulado_anoanterior = 0
End If

Range("B21").Value = Format(faturamento_acumulado, "Currency")
Range("B22").Value = Format(faturamento_acumulado_anoanterior, "Currency")

If faturamento_acumulado >= faturamento_acumulado_anoanterior Then
    Range("B23").Value = "um crescimento"
    Range("B24").Value = Format(faturamento_acumulado - faturamento_acumulado_anoanterior, "Currency")
Else
    Range("B23").Value = "uma queda"
    Range("B24").Value = Format(faturamento_acumulado_anoanterior - faturamento_acumulado, "Currency")
End If
|
```

Para o cálculo do faturamento acumulado do ano anterior só temos que tomar cuidado com um detalhe: só existirá faturamento acumulado do ano anterior se o ano de análise for maior que 2017. Se for igual, o faturamento do ano anterior é igual a zero, pois não temos dados para 2016 ou antes. Por isso, precisamos fazer o *check* indicado ao lado antes de calcular o faturamento do ano anterior.

Para informar o status e também a diferença de faturamento nas células B23 e B24 usamos a mesma lógica da página 493.

# Módulo 13 – Integração com Word – Criação de Relatório Automático (Resolução Parte 6)

500

The screenshot shows a Microsoft Excel interface with the following details:

- Worksheet:** Vendas.xlsxm - Excel
- Toolbar:** Arquivo, Página Inicial, Inserir, Layout da Página, Fórmulas, Dados, Revisão, Exibir, Desenvolvedor, Entrar, Compartilhar.
- Fonte:** Calibri, Size 11, Bold, Italic.
- Alinhamento:** Quebrar Texto Automaticamente, Mesclar e Centralizar.
- Número:** Personalizado, % 000, 0,00.
- Estilo:** Formatação Condicional, Formatar como Tabela, Estilos de Célula.
- Células:** Inserir, Excluir, Formatar.
- Edição:** Classificar, Localizar e e Filtrar, Selecionar.

**Data Tables:**

- Table 1 (B1):** A summary table with columns for Product (A), Month (B), and Year (C). The table includes rows for Celular, Tablet, Notebook, Relogio, and Televisao, along with a Total row (R\$ 74.050).
- Table 2 (D1):** A monthly sales comparison table from January to December, comparing 2017, 2018, and 2019.
- Table 3 (B16):** A detailed breakdown of sales by month and year, including:
  - #NOME\_MES: Setembro
  - #ANO: 2019
  - #FATURAMENTO: R\$ 74.050,00
  - #SITUACAO\_MES: um aumento
  - #COMPARATIVO\_MES: R\$ 5.795,00
  - #FAT\_ACUMULADO: R\$ 388.471,94
  - #FAT\_ANOANTERIOR: R\$ 313.790,00
  - #SITUACAO\_ANO: um crescimento
  - #COMPARATIVO\_ANO: R\$ 74.681,94

**Bottom Navigation:** Pronto, Base Vendas, +.

Ao executar o código, o resultado final parcial já pode ser verificado ao lado.

```
If faturamento >= faturamento_mesanterior Then
    Range("B19").Value = "um aumento"
    Range("B20").Value = Format(faturamento - faturamento_mesanterior, "Currency")
Else
    Range("B19").Value = "uma diminuição"
    Range("B20").Value = Format(faturamento_mesanterior - faturamento, "Currency")
End If

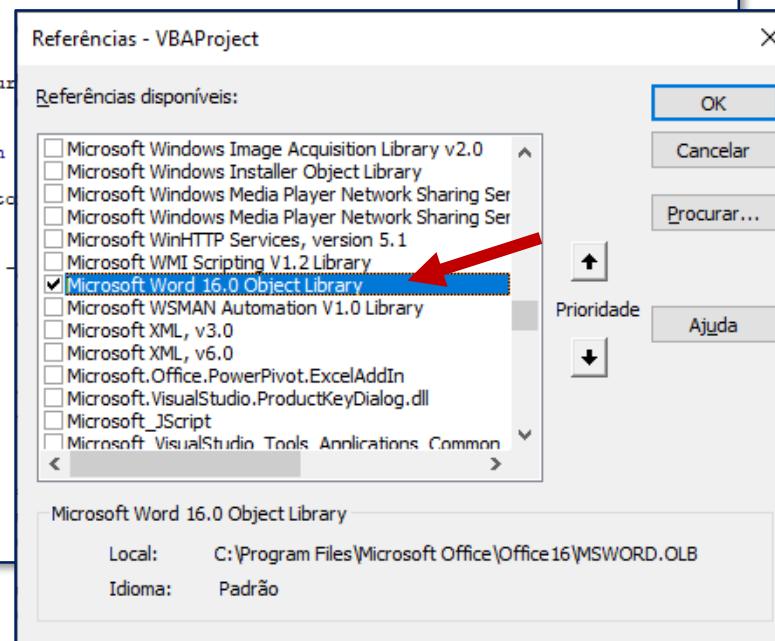
faturamento_acumulado = WorksheetFunction.Sum(Range(Cells(2, coluna_ano), Cells(linha_mes, coluna_ano)))

If ano > 2017 Then
    faturamento_acumulado_anoanterior = WorksheetFunction.Sum(Range(Cells(2, coluna_ano - 1), Cells(linha_mes, coluna_ano - 1)))
Else
    faturamento_acumulado_anoanterior = 0
End If

Range("B21").Value = Format(faturamento_acumulado, "Currency")
Range("B22").Value = Format(faturamento_acumulado_anoanterior, "Cur

If faturamento_acumulado >= faturamento_acumulado_anoanterior Then
    Range("B23").Value = "um crescimento"
    Range("B24").Value = Format(faturamento_acumulado - faturamento_acumulado_anoanterior)
Else
    Range("B23").Value = "uma queda"
    Range("B24").Value = Format(faturamento_acumulado_anoanterior - faturamento_acumulado)
End If

For linha = 16 To 24
    selecao.Find.Text = Cells(linha, 1).Value
    selecao.Find.Replacement.Text = Cells(linha, 2).Value
    selecao.Find.Execute Replace:=wdReplaceAll
Next
```



Para começar a preencher os valores da tabela auxiliar abaixo no arquivo Word, precisamos primeiramente garantir que a opção de integração avançada com o Word está habilitada na guia **Ferramentas → Referências**, isso no ambiente VBA.

15		
16	#NOME_MES	Setembro
17	#ANO	2019
18	#FATURAMENTO	R\$ 74.050,00
19	#SITUACAO_MES	um aumento
20	#COMPARATIVO_MES	R\$ 5.795,00
21	#FAT_ACUMULADO	R\$ 388.471,94
22	#FAT_ANOANTERIOR	R\$ 313.790,00
23	#SITUACAO_ANO	um crescimento
24	#COMPARATIVO_ANO	R\$ 74.681,94
25		

Base Vendas

Em seguida, criamos o Loop For para: localizar e substituir cada um dos textos da nossa tabela auxiliar acima no arquivo Word.

Relatório Mensal.docx - Word

Arquivo Página Inicial Inserir Design Layout Referências Correspondências Revisão Exibir O que você deseja fazer... Entrar Compartilhar

Recortar Copiar Colar Pincel de Formatação Área de Transferência

Fonte Parágrafo Estilo

Normal Sem Espaçamento Título 1 Título 2 Título Subtítulo Ênfase Sutil

Relatório de Vendas Setembro/2019

O mês de **Setembro** fechou com R\$ 74.050,00 de faturamento, o que representa **um aumento** com relação ao mês anterior no valor de R\$ 5.795,00.

A proporção de faturamento entre os produtos da empresa foi a seguinte:

#GRAFICO1

Até o mês de **Setembro** a empresa acumulou um faturamento total de R\$ 388.471,94 no ano, distribuído da seguinte forma:

#GRAFICO2

Com relação ao ano anterior, no qual a empresa obteve um faturamento acumulado até o mês de **Setembro** totalizando R\$ 313.790,00, houve **um crescimento** de R\$ 74.681,94.

Página 1 de 1 102 palavras 100%

E o resultado final parcial é mostrado ao lado.

Vendas.xlsx - Excel

Arquivo Página Inicial Inserir Layout da Página Fórmulas Dados Revisão Exibir Desenvolvedor O que você deseja fazer... Entrar Compartilhar

Visual Basic Gravar Macro Usar Referências Relativas Suplementos do Excel Suplementos do COM Inserir Modo de Design Propriedades Exibir Código Executar Caixa de Diálogo Controles Código-fonte Propriedades do Mapa Importar Pacotes de Expansão Exportar Atualizar Dados XML

B1 A B C E F G H I J K L M N O P Q R S T U V W X Y Z AA AB AC AD

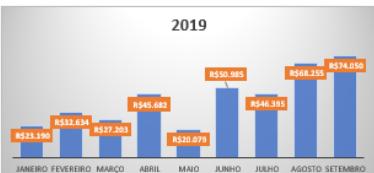
**Outubro**

	B	C	E	F	G	H
	Outubro	2019	Mês	2017	2018	2019
1 Celular	R\$ 30.361	Janeiro	R\$ 24.400	R\$ 21.472	R\$ 23.190	
2 Tablet	R\$ 6.685	Fevereiro	R\$ 21.000	R\$ 23.310	R\$ 32.634	
3 Notebook	R\$ 17.032	Março	R\$ 25.300	R\$ 24.288	R\$ 27.203	
4 Relógio	R\$ 5.184	Abri	R\$ 30.700	R\$ 38.068	R\$ 45.682	
5 Televisão	R\$ 14.810	Mai	R\$ 17.000	R\$ 15.510	R\$ 20.079	
6 Total:	R\$ 74.050	Junho	R\$ 38.300	R\$ 39.832	R\$ 50.395	
7		Julho	R\$ 52.800	R\$ 53.328	R\$ 46.395	
8		Agosto	R\$ 27.400	R\$ 46.750	R\$ 68.255	
9		Setembro	R\$ 27.600	R\$ 50.932	R\$ 74.050	
10		Outubro	R\$ 55.900	R\$ 76.583	R\$ 74.050	
11		Novembro	R\$ 60.404	R\$ 72.485		
12		Dezembro	R\$ 82.800	R\$ 103.500		
13						
14						
15						
16 #NOME_MES	Setembro					
17 #ANO	2019					
18 #FATURAMENTO	R\$ 74.050,00					
19 #SITUAÇÃO_MES	um aumento					
20 #COMPARATIVO_MES	R\$ 5.795,00					
21 #FAT_ACUMULADO	R\$ 388.471,94					
22 #FAT_ANOANTERIOR	R\$ 313.790,00					
23 #SITUAÇÃO_ANO	um crescimento					
24 #COMPARATIVO_ANO	R\$ 74.681,94					
25						
26						
27						
28						
29						
30						
31						
32						
33						
34						
35						
36						
37						
38						
39						
40						
41						
42						

Base Vendas

Pronto

Estamos bem encaminhados e agora só faltam os gráficos.



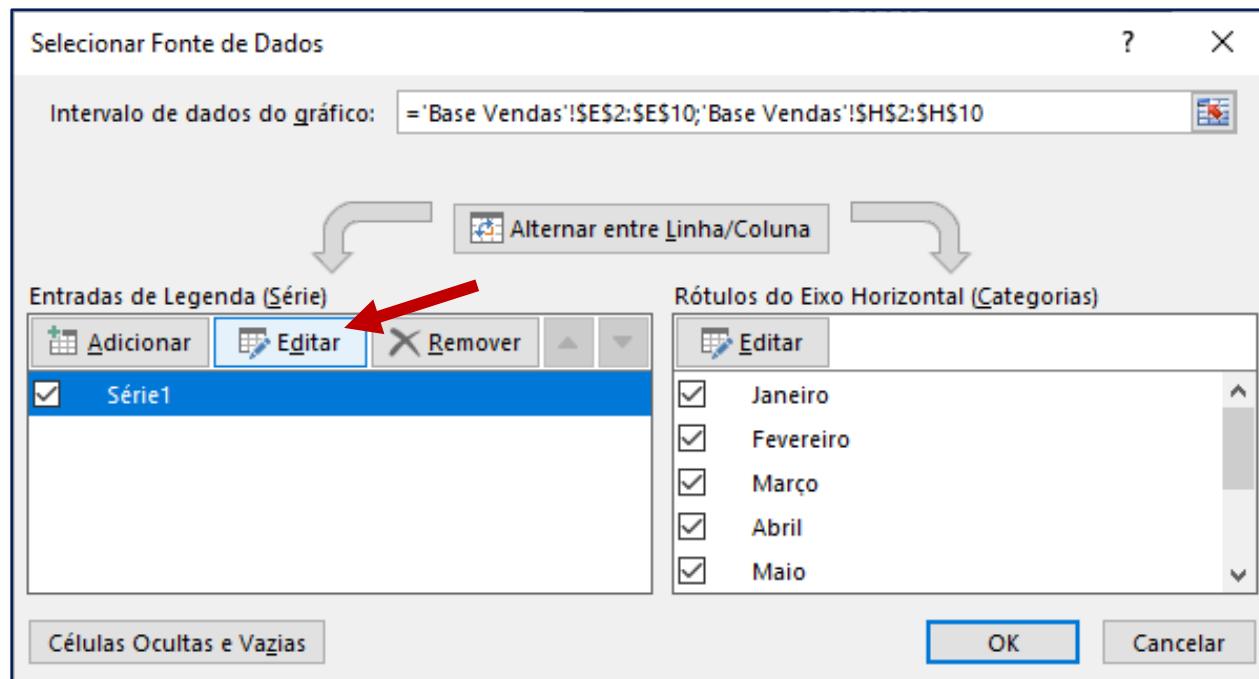
The screenshot shows a Microsoft Excel interface with a green ribbon bar. The 'Desenvolvedor' tab is selected. On the left, there's a table titled 'Base Vendas' with columns for Product (A), Month (B), Year (C), and Sales (D-E). A red arrow points from the table to the context menu of a column chart. The context menu is open, showing options like 'Preenchimento', 'Estrutura de Tópicos', 'Recortar', 'Copiar', and 'Opções de Colagem:'. A second red arrow points specifically to the 'Selecionar Dados...' option in the menu.

O ponto importante agora, antes de copiar e colar os gráficos, é fazer com que eles fiquem automáticos de acordo com a data de análise. Como assim?

De acordo com o mês e ano de análise, os dados mostrados no gráfico devem ser atualizados automaticamente antes de copiá-los para o Word.

Para fazer isso, vamos gravar uma macro para obter o código que altera os valores selecionados. Para fazer essa atualização, antes de gravar a macro você pode adicionar um valor aleatório para o mês de outubro, depois:

1. Gravar macro
2. Clicar com o botão direito no gráfico de colunas.
3. Clicar na opção de Selecionar Dados.



Aqui teremos que editar tanto os valores do lado esquerdo, quando o eixo X do lado direito.

Começamos selecionando a Série1 e clicando no botão de Editar.

The screenshot shows a Microsoft Excel spreadsheet with a pivot table and a 'Editar Série' (Edit Series) dialog box.

**Pivot Table Data:**

	B	C	D	E	F	G	H	I	J	K
1	A	Outubro	= 2019	Mês	2017	2018	2019			
2	Celular	R\$ 30.361	Janeiro	R\$ 24.400	R\$ 21.472	R\$ 23.190				
3	Tablet	R\$ 6.665	Fevereiro	R\$ 21.000	R\$ 23.310	R\$ 32.634				
4	Notebook	R\$ 17.032	Março	R\$ 25.300	R\$ 24.288	R\$ 27.203				
5	Relogio	R\$ 5.184	Abri	R\$ 30.700	R\$ 38.068	R\$ 45.682				
6	Televisão	R\$ 14.810	Maio	R\$ 17.000	R\$ 15.810	R\$ 20.0	Área de Plotagem			
7	Total:	R\$ 74.050	Junho	R\$ 38.300	R\$ 39.832	R\$ 50.969				
8			Julho	R\$ 52.800	R\$ 53.328	R\$ 46.395				
9			Agosto	R\$ 37.400	R\$ 46.750	R\$ 68.255				
10			Setembro	R\$ 47.600	R\$ 50.932	R\$ 74.050				
11			Outubro	R\$ 55.900	R\$ 76.583	R\$ 74.050				
12			Novembro	R\$ 60.404	R\$ 72.485					
13			Dezembro	R\$ 82.800	R\$ 103.500					

**Edit Series Dialog Box:**

**Nome da série:** Selecionar um intervalo

**Valores da série:** ='Base Vendas'!SH\$2:\$H\$10 = R\$ 23.190; ...

Buttons: OK, Cancelar

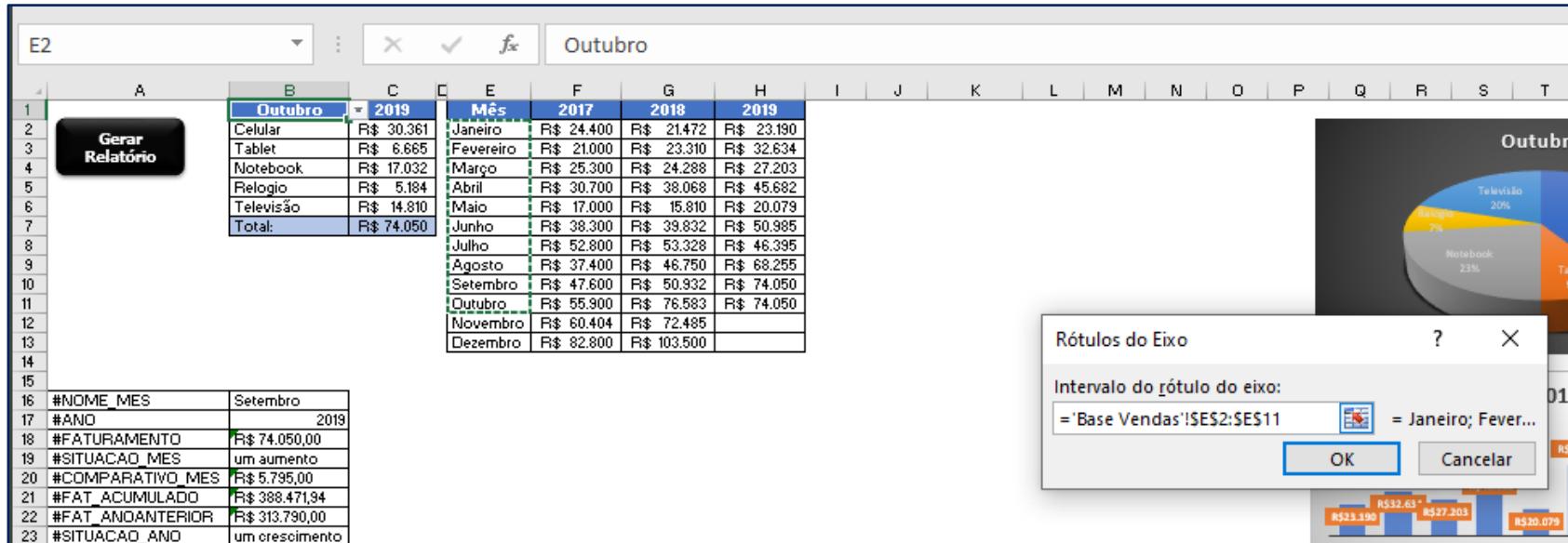
Na opção de **Valores da série**: aumente o tamanho do intervalo para abranger também o mês de outubro.

Em seguida clique em Ok.

The screenshot shows a Microsoft Excel spreadsheet with two main sections:

- Pivot Table Area:** Contains a table titled "Outubro" with columns for "Mes" (Month) and years "2017", "2018", and "2019". The table includes rows for various products like Celular, Tablet, Notebook, Relógio, Televisão, and a total row.
- Dialog Box:** A "Selecionar Fonte de Dados" (Select Data Source) dialog box is open. It has tabs for "Entradas de Legenda (Série)" (Legend Entries (Series)) and "Rótulos do Eixo Horizontal (Categorias)" (Horizontal Axis Labels (Categories)). The "Rótulos do Eixo Horizontal" tab is selected, showing a list of months from Janeiro to Maio with checkboxes next to them. An arrow points to the "Editar" (Edit) button in this tab.

Para atualizar também o eixo X do gráfico, clique na opção Editar da direita.



Aumente o intervalo de meses da coluna E para abranger também o mês de outubro.

Clique em Ok.

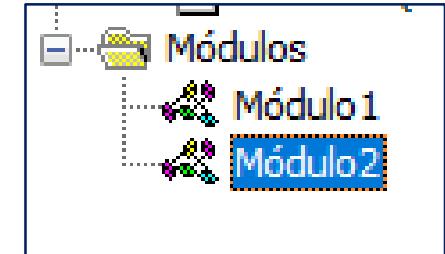
The screenshot shows a Microsoft Excel spreadsheet titled "Base Vendas". A pivot table is displayed in the top-left corner, with the title "Outubro" selected. The pivot table includes columns for Product (Celular, Tablet, Notebook, Relógio, Televisão) and Month (2019, 2017, 2018, 2019). Below the pivot table, there is a large block of formulas starting with "#NOME\_MES" and ending with "#COMPARATIVO\_ANO". In the center, a "Selecionar Fonte de Dados" (Select Data Source) dialog box is open, showing the range = 'Base Vendas'!\$E\$2:\$E\$11; 'Base Vendas'!\$H\$2:\$H\$11. To the right of the dialog, a chart is being edited, showing two series: 2018 (blue line) and 2019 (orange line) over time from Janeiro to Setembro. The chart has a dark background and a legend at the bottom. On the far right, a ribbon menu is visible with tabs like Arquivo, Página Inicial, Inserir, and Layout da Pág. A dropdown menu is open under the "Inserir" tab, specifically the "Gráfico" section, with "Gráfico 4" selected. A tooltip for "Gerar Relatório" is shown in the bottom right corner of the screen.

Finalmente, clique em Ok de novo.

Após isso, **não esqueça de clicar no botão de Parar Gravação!**

```
Sub Macrol()
    '
    ' Macrol Macro
    '
    '
    ActiveSheet.ChartObjects("Gráfico 4").Activate
    ActiveChart.FullSeriesCollection(1).Values = "'Base Vendas' !$H$2:$H$11"
    ActiveChart.FullSeriesCollection(1).XValues = "'Base Vendas' !$E$2:$E$11"
End Sub
```

O código gravado é mostrado na imagem ao lado. Lembre-se que ele estará em um segundo módulo.



Copie este código ...

```
faturamento_acumulado = WorksheetFunction.Sum(Range(Cells(2, coluna_ano), Cells(linha_mes, coluna_ano)))  
  
If ano > 2017 Then  
    faturamento_acumulado_anoanterior = WorksheetFunction.Sum(Range(Cells(2, coluna_ano - 1), Cells(linha_mes, coluna_ano - 1)))  
Else  
    faturamento_acumulado_anoanterior = 0  
End If  
  
Range("B21").Value = Format(faturamento_acumulado, "Currency")  
Range("B22").Value = Format(faturamento_acumulado_anoanterior, "Currency")  
  
If faturamento_acumulado >= faturamento_acumulado_anoanterior Then  
    Range("B23").Value = "um crescimento"  
    Range("B24").Value = Format(faturamento_acumulado - faturamento_acumulado_anoanterior, "Currency")  
Else  
    Range("B23").Value = "uma queda"  
    Range("B24").Value = Format(faturamento_acumulado_anoanterior - faturamento_acumulado, "Currency")  
End If  
  
For linha = 16 To 24  
  
    selecao.Find.Text = Cells(linha, 1).Value  
    selecao.Find.Replacement.Text = Cells(linha, 2).Value  
    selecao.Find.Execute Replace:=wdReplaceAll  
  
Next  
  
ActiveSheet.ChartObjects("Gráfico 4").Activate  
ActiveChart.FullSeriesCollection(1).Values = "'Base Vendas'!$H$2:$H$11"  
ActiveChart.FullSeriesCollection(1).XValues = "'Base Vendas'!$E$2:$E$11"  
  
End Sub
```

... e cole ao final do nosso código em construção.

Teremos apenas que fazer algumas adaptações pois o mesmo não está automático de acordo com as opções de mês e ano selecionado.

```
For linha = 16 To 24  
  
    selecao.Find.Text = Cells(linha, 1).Value  
    selecao.Find.Replacement.Text = Cells(linha, 2).Value  
    selecao.Find.Execute Replace:=wdReplaceAll  
  
    Next  
  
    endereco_iniciovalores = Cells(2, coluna_ano).Address  
    endereco_fimvalores = Cells(linha_mes, coluna_ano).Address  
    endereco_completovalores = endereco_iniciovalores & ":" & endereco_fimvalores  
  
    endereco_iniciomes = Cells(2, 5).Address  
    endereco_fimmes = Cells(linha_mes, 5).Address  
    endereco_completomes = endereco_iniciomes & ":" & endereco_fimmes  
  
    ActiveSheet.ChartObjects("Gráfico 4").Activate  
        ActiveChart.FullSeriesCollection(1).Values = "'Base Vendas'" & endereco_completovalores  
        ActiveChart.FullSeriesCollection(1).XValues = "'Base Vendas'" & endereco_completomes  
  
End Sub
```

A comando **Address** é apenas para retornar a informação de endereço da célula, em um formato de texto, que é exatamente o que os valores do gráfico precisam, um intervalo de células no formato de texto.

Para automatizar os intervalos de seleção, tanto dos valores quanto dos meses, criamos 3 variáveis:

### 1. **endereco\_iniciovalores:**

De acordo com o número da coluna selecionada, sabemos que o gráfico deve começar na linha do mês de janeiro (linha 2).

### 2. **endereco\_fimvalores:**

É a última célula do intervalo, que termina na linha **linha\_mes**, e está na coluna **coluna\_ano**.

### 3. **endereco\_completovalores:**

Apenas a concatenação das duas variáveis acima para compor o intervalo final de valores.

```
For linha = 16 To 24  
  
    selecao.Find.Text = Cells(linha, 1).Value  
    selecao.Find.Replacement.Text = Cells(linha, 2).Value  
    selecao.Find.Execute Replace:=wdReplaceAll  
  
Next  
  
endereco_iniciovalores = Cells(2, coluna_ano).Address  
endereco_fimvalores = Cells(linha_mes, coluna_ano).Address  
endereco_completovalores = endereco_iniciovalores & ":" & endereco_fimvalores  
  
endereco_iniciomes = Cells(2, 5).Address  
endereco_fimmes = Cells(linha_mes, 5).Address  
endereco_completomes = endereco_iniciomes & ":" & endereco_fimmes  
  
ActiveSheet.ChartObjects("Gráfico 4").Activate  
    ActiveChart.FullSeriesCollection(1).Values = "'Base Vendas'!" & endereco_completovalores  
    ActiveChart.FullSeriesCollection(1).XValues = "'Base Vendas'!" & endereco_completomes  
  
End Sub
```

A mesma lógica anterior é aplicada para o intervalo de meses do eixo X, com a diferença de que os meses sempre vão estar na coluna E (coluna 5).

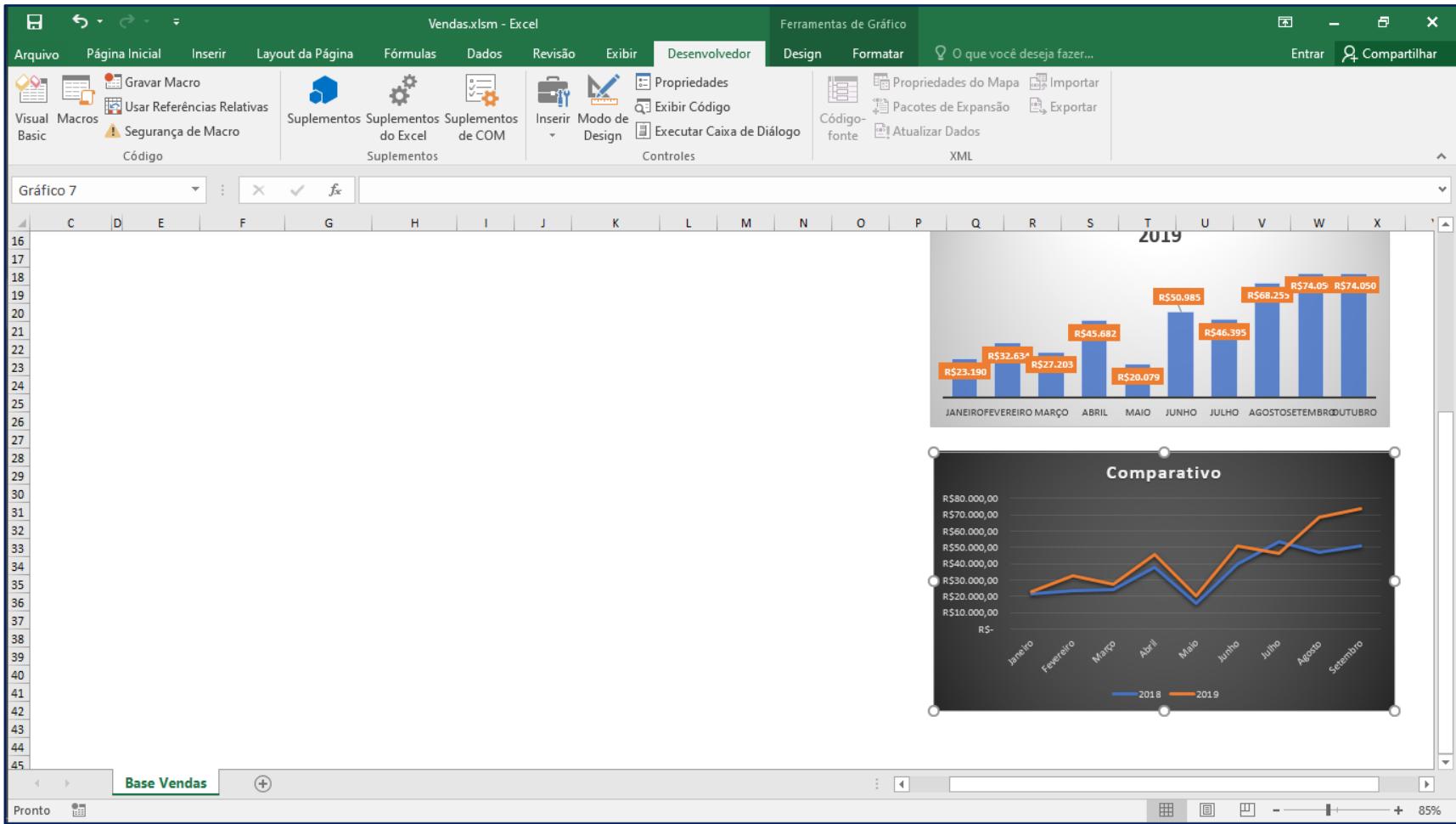
Por fim, usamos as duas variáveis:

**endereco\_completovalores**  
**endereco\_completomes**

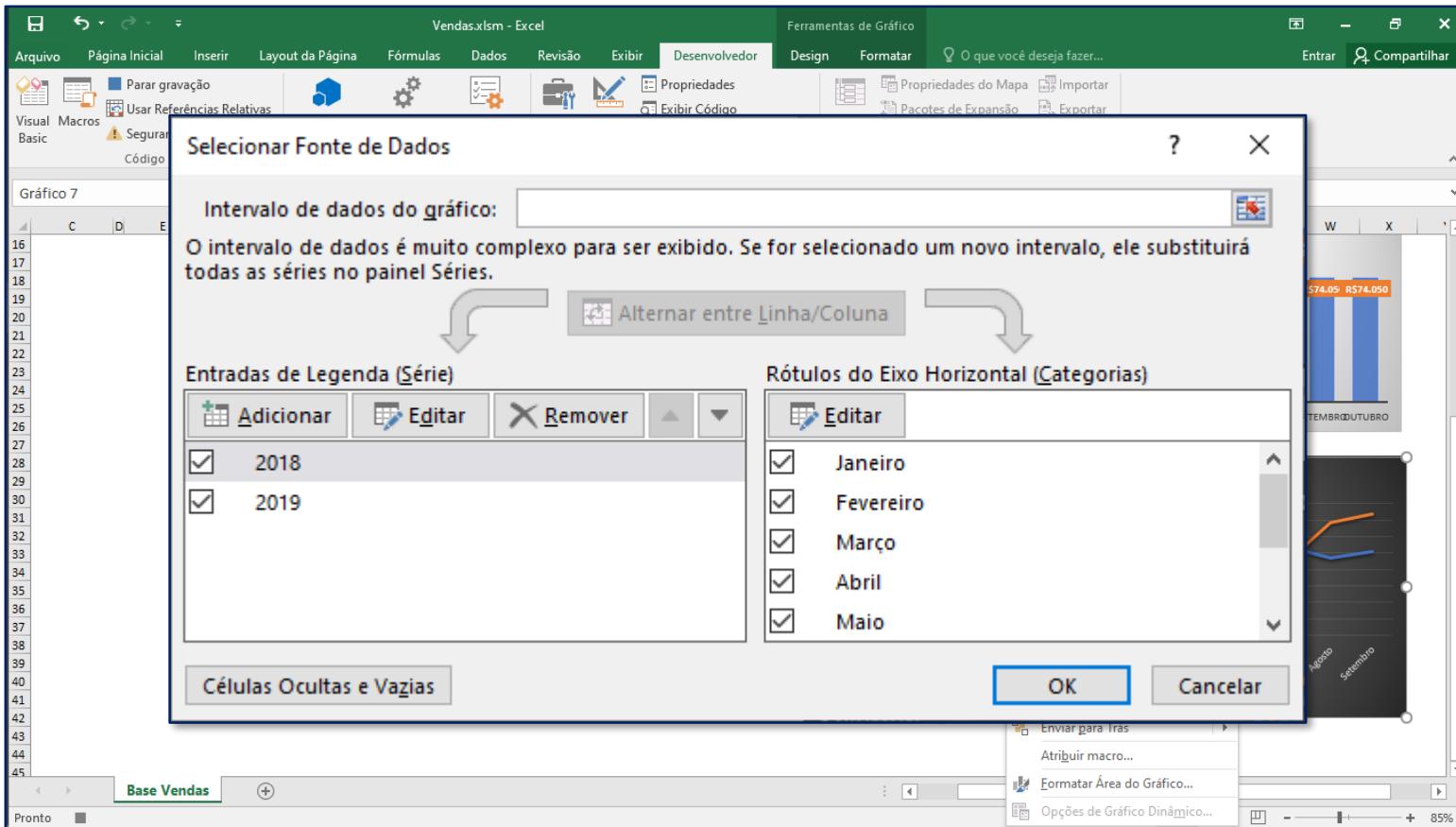
Como os intervalos para criação do nosso gráfico.

# Módulo 13 – Integração com Word – Criação de Relatório Automático (Resolução Parte 10)

514



Para descobrir o código que atualiza os valores do gráfico de linhas, fazemos o mesmo processo anterior de gravação de macro.



Dessa vez, vamos atualizar tanto os valores para 2019, quanto para 2018.

Vendas.xlsxm - Excel

Arquivo Página Inicial Inserir Layout da Página Fórmulas Dados Revisão Exibir Desenvolvedor Design Formatar O que você deseja fazer.

Parar gravação Usar Referências Relativas Suplementos do Excel Suplementos de COM Inserir Modo de Design Propriedades Executar Caixa de Diálogo Código-fonte Propriedades do Mapa Importar Pacotes de Expansão Exportar Atualizar Dados XML

Visual Basic Segurança de Macro Código

H2

	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	2019		Mês	2017	2018	2019										
2	R\$ 30.361		Janeiro	R\$ 24.400	R\$ 21.472	R\$ 23.190										
3	R\$ 6.665		Fevereiro	R\$ 21.000	R\$ 23.310	R\$ 32.634										
4	R\$ 17.032		Março	R\$ 25.300	R\$ 24.288	R\$ 27.203										
5	R\$ 5.184		Abril	R\$ 30.700	R\$ 38.068	R\$ 45.682										
6	R\$ 14.810		Maio	R\$ 17.000	R\$ 15.810	R\$ 20.079										
7	R\$ 74.050		Junho	R\$ 38.300	R\$ 39.832	R\$ 50.985										
8			Julho	R\$ 52.800	R\$ 53.328	R\$ 46.395										
9			Agosto	R\$ 37.400	R\$ 46.750	R\$ 68.255										
10			Setembro	R\$ 47.600	R\$ 50.932	R\$ 74.050										
11			Outubro	R\$ 55.900	R\$ 76.583	R\$ 74.050										
12			Novembro	R\$ 60.404	R\$ 72.485											
13			Dezembro	R\$ 82.800	R\$ 103.500											
14																
15																
16																
17																
18																
19																

Editar Série

Nome da série:  
= 'Base Vendas'!SH\$1 = 2019

Valores da série:  
= 'Base Vendas'!SH\$2:\$H\$11 = \$ 23189,760 ;...

OK Cancelar

Começamos com o de 2019.

# Módulo 13 – Integração com Word – Criação de Relatório Automático (Resolução Parte 10)

517

Vendas.xlsxm - Excel

Arquivo Página Inicial Inserir Layout da Página Fórmulas Dados Revisão Exibir Desenvolvedor

Visual Basic Macros Usar Referências Relativas Segurança de Macro Suplementos do Excel Suplementos de COM Inserir Modo de Design Executar Caixa de Diálogo Propriedades Exibir Código Executar Caixa de Diálogo Propriedades do Mapa Importar Pacotes de Expansão Exportar Código-fonte Atualizar Dados XML

Código Suplementos

G2

	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	2019		Mês	2017		2018		2019									
2	R\$ 30.361		Janeiro	R\$ 24.400	R\$ 21.472	R\$ 23.190											
3	R\$ 6.665		Fevereiro	R\$ 21.000	R\$ 23.310	R\$ 32.634											
4	R\$ 17.032		Março	R\$ 25.300	R\$ 24.288	R\$ 27.203											
5	R\$ 5.184		AbriL	R\$ 30.700	R\$ 38.068	R\$ 45.682											
6	R\$ 14.810		Maio	R\$ 17.000	R\$ 15.810	R\$ 20.079											
7	R\$ 74.050		Junho	R\$ 38.300	R\$ 39.832	R\$ 50.985											
8			Julho	R\$ 52.800	R\$ 53.328	R\$ 46.395											
9			Agosto	R\$ 37.400	R\$ 46.750	R\$ 68.255											
10			Setembro	R\$ 47.600	R\$ 50.932	R\$ 74.050											
11			Outubro	R\$ 55.900	R\$ 76.583	R\$ 74.050											
12			Novembro	R\$ 60.404	R\$ 72.485												
13			Dezembro	R\$ 82.800	R\$ 103.500												
14																	
15																	
16																	
17																	

Editar Série

Nome da série:  
= 'Base Vendas'!\$G\$1 = 2018

Valores da série:  
= 'Base Vendas'!\$G\$2:\$G\$11 = \$ 21472,0; ...

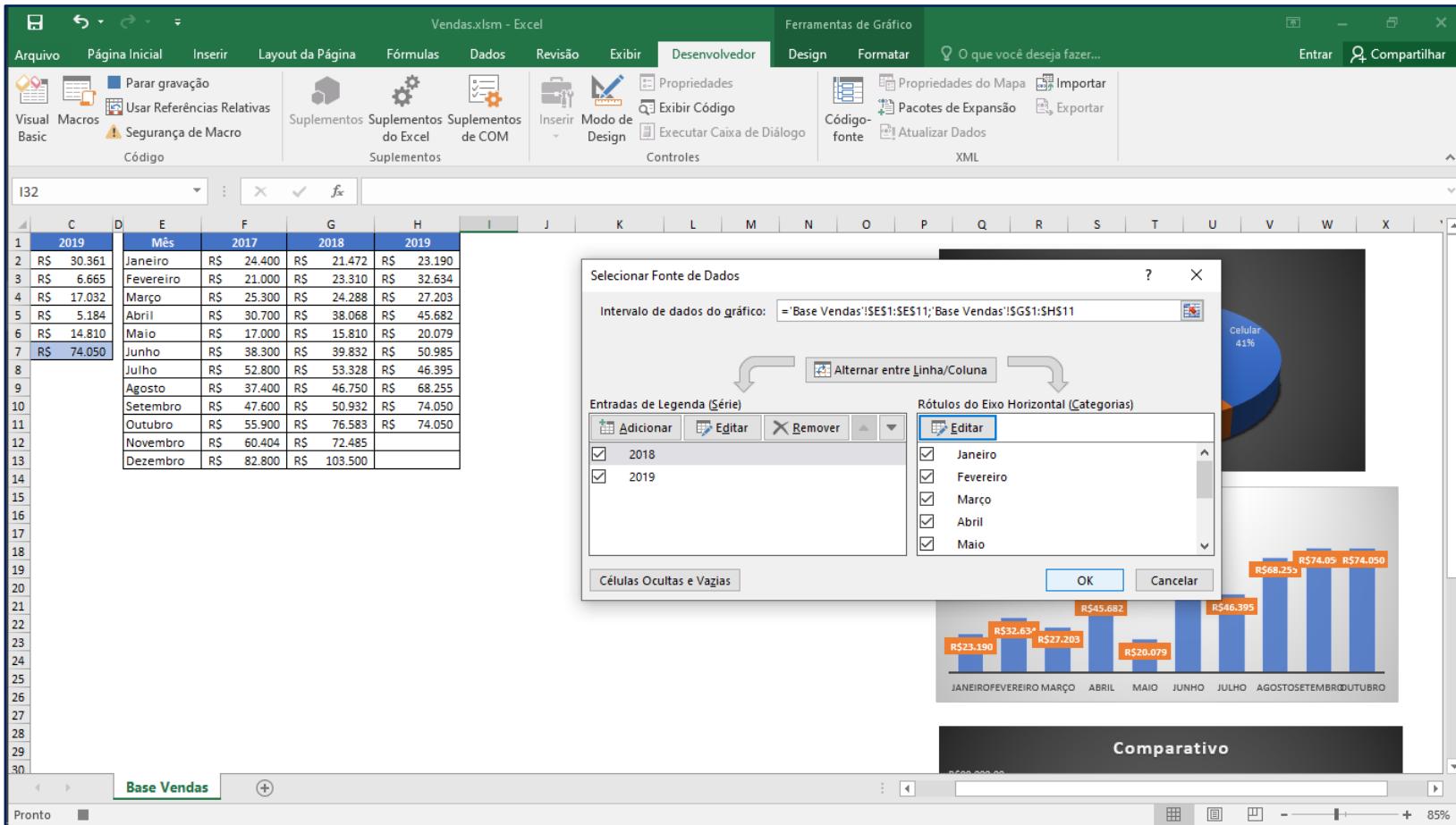
OK Cancelar

Em seguida para 2018.

E por fim, para os meses.

# Módulo 13 – Integração com Word – Criação de Relatório Automático (Resolução Parte 10)

519



Lembre-se de clicar na opção de Parar Gravação ao final das configurações do gráfico.

```
Next  
  
endereco_iniciovalores = Cells(2, coluna_ano).Address  
endereco_fimvalores = Cells(linha_mes, coluna_ano).Address  
endereco_completovalores = endereco_iniciovalores & ":" & endereco_fimvalores  
  
endereco_iniciomes = Cells(2, 5).Address  
endereco_fimmes = Cells(linha_mes, 5).Address  
endereco_completomes = endereco_iniciomes & ":" & endereco_fimmes  
  
ActiveSheet.ChartObjects("Gráfico 4").Activate  
ActiveChart.FullSeriesCollection(1).Values = "'='Base Vendas'!" & endereco_completovalores  
ActiveChart.FullSeriesCollection(1).XValues = "'='Base Vendas'!" & endereco_completomes  
  
  
ActiveSheet.ChartObjects("Gráfico 7").Activate  
  
ActiveChart.FullSeriesCollection(2).Values = "'='Base Vendas'!$H$2:$H$11"  
ActiveChart.FullSeriesCollection(1).Values = "'='Base Vendas'!$G$2:$G$11"  
ActiveChart.FullSeriesCollection(1).XValues = "'='Base Vendas'!$E$2:$E$11"  
  
End Sub
```

Copie o código gravado e cole ao final do nosso código principal.

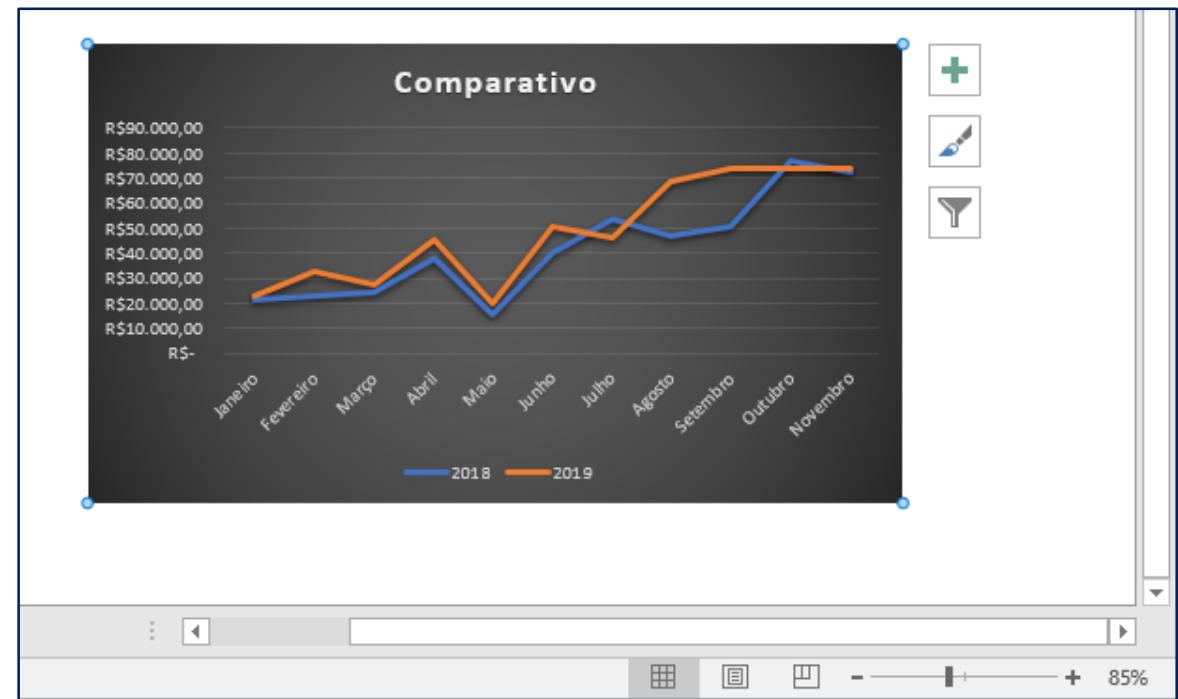
```
ActiveSheet.ChartObjects("Gráfico 7").Activate  
  
ActiveChart.FullSeriesCollection(2).Values = "'Base Vendas'" & endereco_completovalores  
  
endereco_iniciovaloresanoanterior = Cells(2, coluna_ano - 1).Address  
endereco_fimvaloresanoanterior = Cells(linha_mes, coluna_ano - 1).Address  
endereco_completovaloresanoanterior = endereco_iniciovaloresanoanterior & ":" & endereco_fimvaloresanoanterior  
  
ActiveChart.FullSeriesCollection(1).Values = "'Base Vendas'" & endereco_completovaloresanoanterior  
ActiveChart.FullSeriesCollection(1).XValues = "'Base Vendas'" & endereco_completomes  
  
End Sub
```

Como temos dois gráficos de linha, temos a série 1 (referente ao ano anterior) e a série 2 (referente ao ano atual).

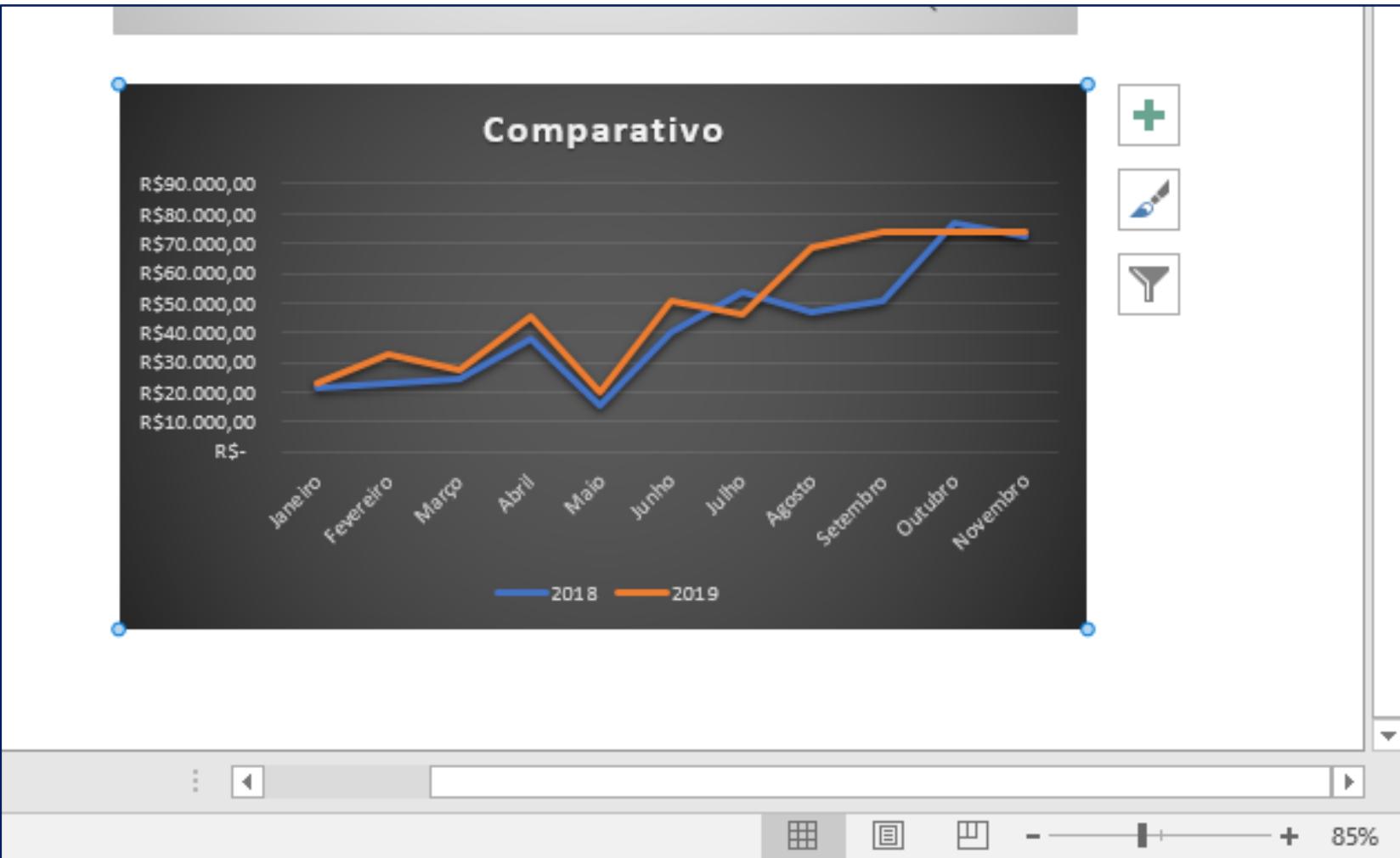
Para preencher essas duas séries de valores, seguiremos a mesma lógica de criar um intervalo para os valores do ano anterior.

A screenshot of Microsoft Excel showing a table of sales data for November 2019. The table has columns for Product (A), Month (B), and Value (C). The value for November 2019 is R\$ 74.050. A black button labeled "Gerar Relatório" is visible on the left.

A	B	C	D
1	Novembro	2019	
2	Celular	R\$ 30.361	Jane
3	Tablet	R\$ 6.665	Feve
4	Notebook	R\$ 17.032	Marc
5	Relogio	R\$ 5.184	Abril
6	Televisão	R\$ 14.810	Mai
7	Total:	R\$ 74.050	Junh
8			Julho



Podemos ver agora que, após preencher mais uma valor para um novo mês (por exemplo, Novembro de 2019) e executar a macro, esta nova informação é inserida no gráfico automaticamente.



Antes de continuar, precisamos nos atentar a mais um detalhe: a legenda do gráfico não está mudando automaticamente. Devemos incluir esta atualização no nosso código.

```
selecao.Find.Text = Cells(linha, 1).Value
selecao.Find.Replacement.Text = Cells(linha, 2).Value
selecao.Find.Execute Replace:=wdReplaceAll

Next

endereco_iniciovalores = Cells(2, coluna_ano).Address
endereco_fimvalores = Cells(linha_mes, coluna_ano).Address
endereco_completovalores = endereco_iniciovalores & ":" & endereco_fimvalores

endereco_iniciomes = Cells(2, 5).Address
endereco_fimmes = Cells(linha_mes, 5).Address
endereco_completomes = endereco_iniciomes & ":" & endereco_fimmes

ActiveSheet.ChartObjects("Gráfico 4").Activate
ActiveChart.FullSeriesCollection(1).Values = "'=Base Vendas'" & endereco_completovalores
ActiveChart.FullSeriesCollection(1).XValues = "'=Base Vendas'" & endereco_completomes

ActiveSheet.ChartObjects("Gráfico 7").Activate

ActiveChart.FullSeriesCollection(2).Values = "'=Base Vendas'" & endereco_completovalores

endereco_iniciovaloresanoanterior = Cells(2, coluna_ano - 1).Address
endereco_fimvaloresanoanterior = Cells(linha_mes, coluna_ano - 1).Address
endereco_completovaloresanoanterior = endereco_iniciovaloresanoanterior & ":" & endereco_fimvaloresanoanterior

ActiveChart.FullSeriesCollection(1).Values = "'=Base Vendas'" & endereco_completovaloresanoanterior
ActiveChart.FullSeriesCollection(1).XValues = "'=Base Vendas'" & endereco_completomes

ActiveChart.FullSeriesCollection(1).Name = ano - 1
ActiveChart.FullSeriesCollection(2).Name = ano

End Sub
```

A adaptação é bem simples e está mostrada ao lado.

```
ActiveSheet.ChartObjects("Gráfico 4").Activate
ActiveChart.FullSeriesCollection(1).Values = "'Base Vendas'!" & endereco_completovalores
ActiveChart.FullSeriesCollection(1).XValues = "'Base Vendas'!" & endereco_completomes

ActiveSheet.ChartObjects("Gráfico 7").Activate

ActiveChart.FullSeriesCollection(2).Values = "'Base Vendas'!" & endereco_completovalores

If ano > 2017 Then

    endereco_iniciovaloresanoanterior = Cells(2, coluna_ano - 1).Address
    endereco_fimvaloresanoanterior = Cells(linha_mes, coluna_ano - 1).Address
    endereco_completovaloresanoanterior = endereco_iniciovaloresanoanterior & ":" & endereco_fimvaloresanoanterior

    ActiveChart.FullSeriesCollection(1).Values = "'Base Vendas'!" & endereco_completovaloresanoanterior

Else

    ActiveChart.FullSeriesCollection(1).Values = 0

End If

    ActiveChart.FullSeriesCollection(1).XValues = "'Base Vendas'!" & endereco_completomes

If ano > 2017 Then

    ActiveChart.FullSeriesCollection(1).Name = ano - 1

Else

    ActiveChart.FullSeriesCollection(1).Name = "-"

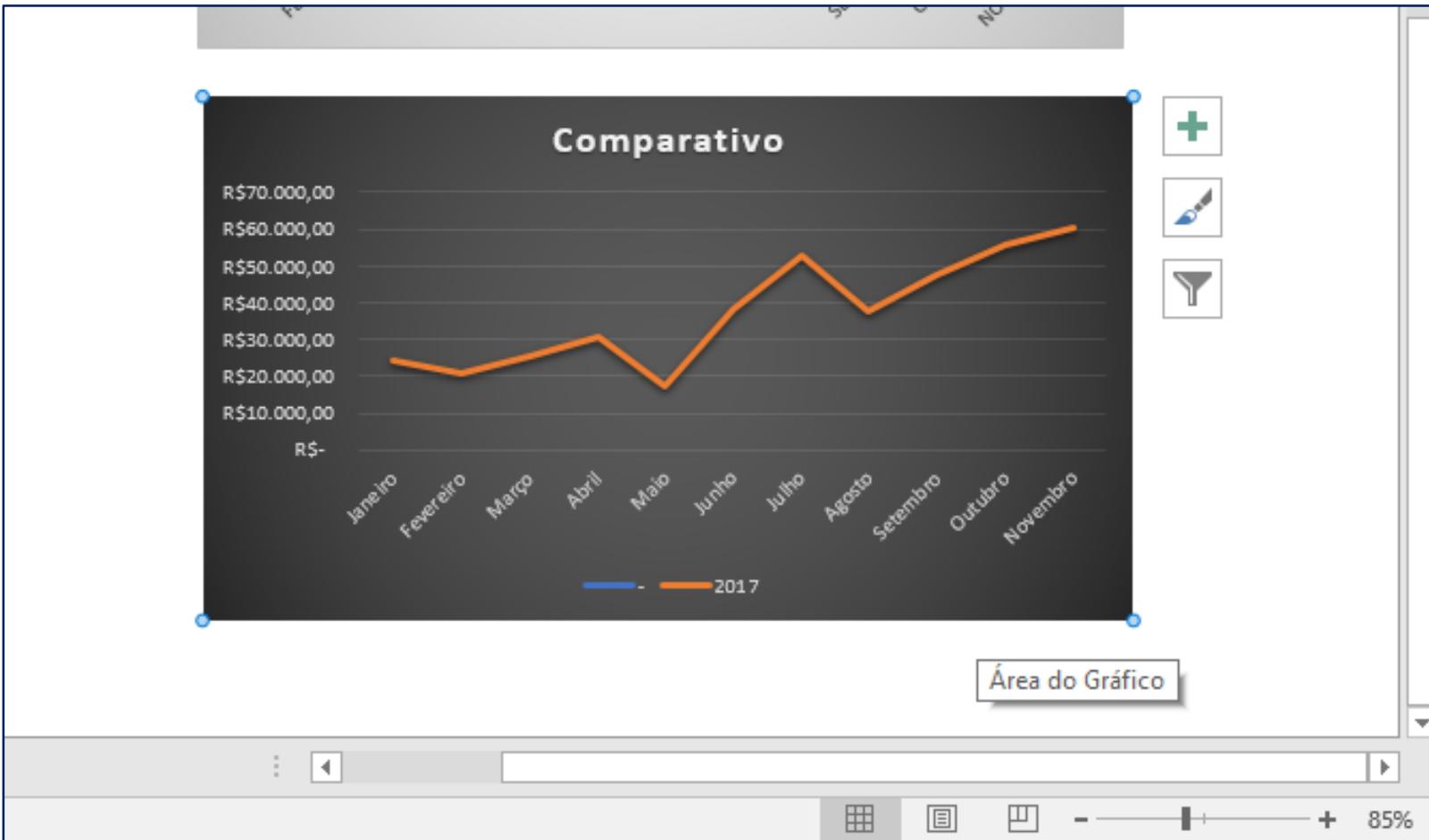
End If

    ActiveChart.FullSeriesCollection(2).Name = ano

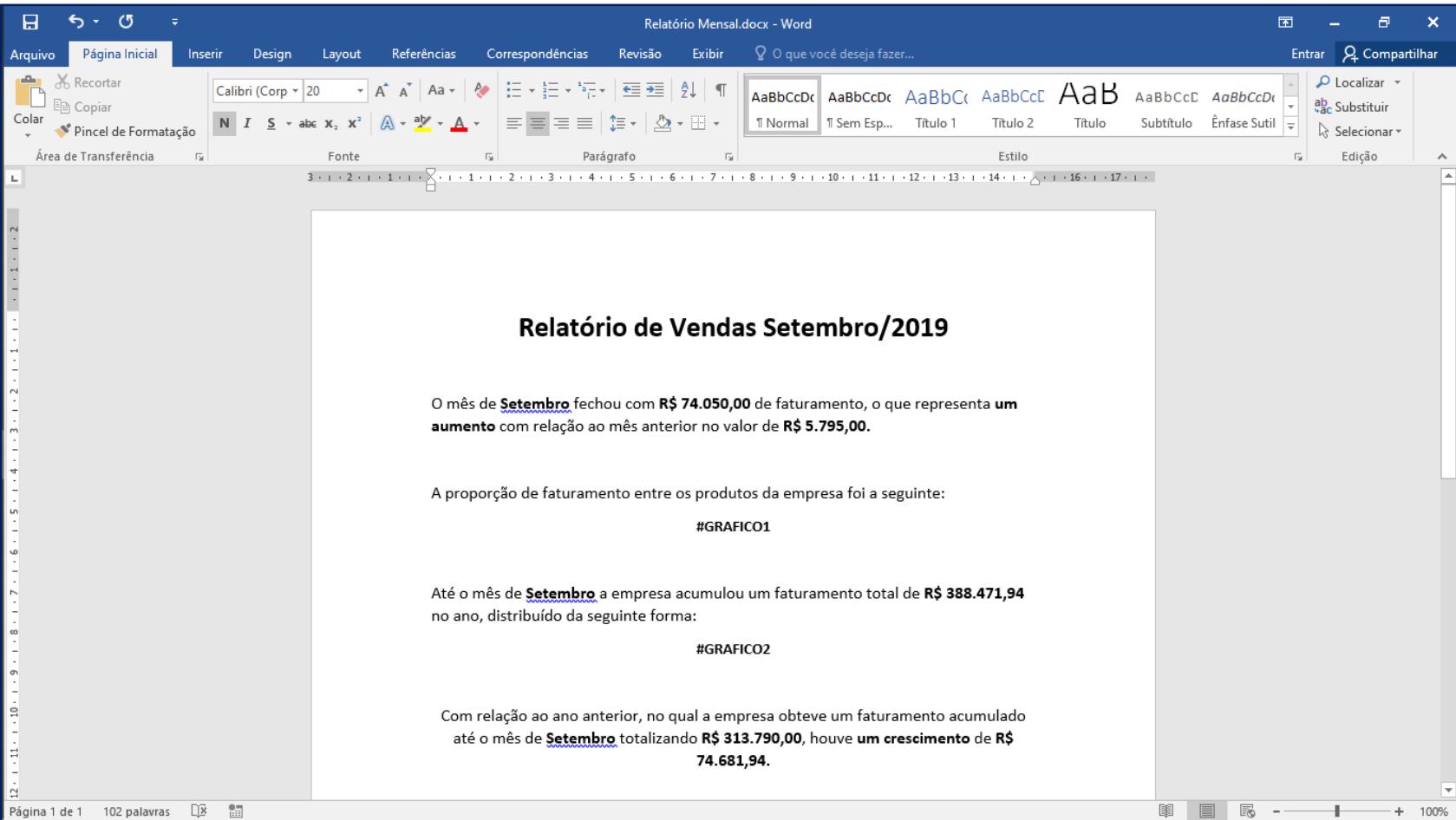
End Sub
```

Porém, temos um outro problema com a legenda e com os dados do faturamento do ano anterior: aquele mesmo relacionado ao fato de que, antes de 2017, não temos dados.

Portanto, devemos primeiramente fazer um *check* para testar se o ano de análise é 2017. Se não for, pode alterar os valores e a legenda, se não, os valores para o ano anterior são iguais a zero e a legenda deve ser igual a “-”.



Agora sim teremos o nosso gráfico a prova de problemas.



The screenshot shows a Microsoft Word document titled "Relatório Mensal.docx - Word". The ribbon menu is visible at the top, showing tabs like Arquivo, Página Inicial, Inserir, Design, Layout, Referências, Correspondências, Revisão, Exibir, and O que você deseja fazer... The "Página Inicial" tab is selected. The main content area contains the following text:

**Relatório de Vendas Setembro/2019**

O mês de **Setembro** fechou com **R\$ 74.050,00** de faturamento, o que representa **um aumento** com relação ao mês anterior no valor de **R\$ 5.795,00**.

A proporção de faturamento entre os produtos da empresa foi a seguinte:

#GRAFICO1

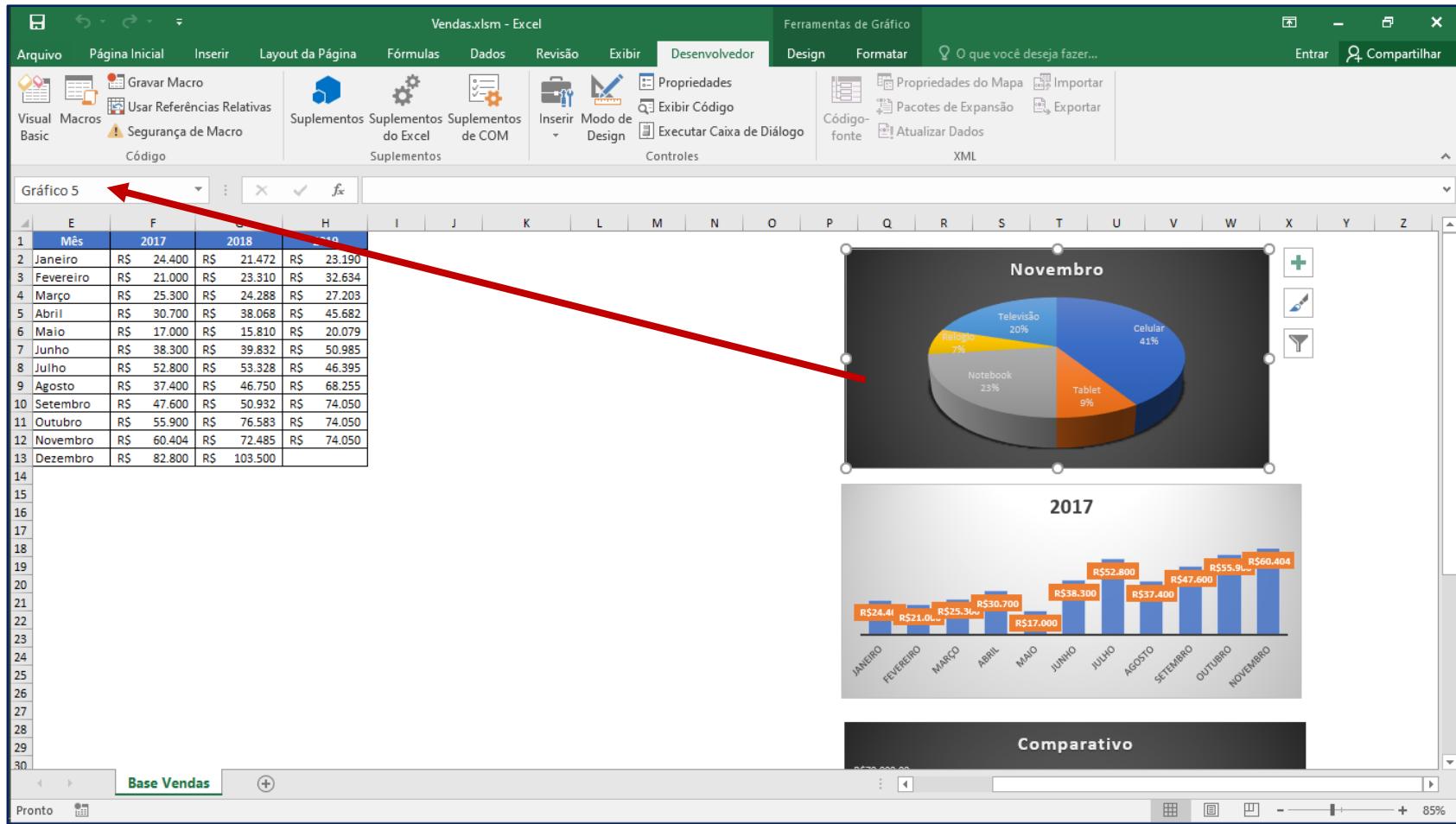
Até o mês de **Setembro** a empresa acumulou um faturamento total de **R\$ 388.471,94** no ano, distribuído da seguinte forma:

#GRAFICO2

Com relação ao ano anterior, no qual a empresa obteve um faturamento acumulado até o mês de **Setembro** totalizando **R\$ 313.790,00**, houve **um crescimento** de **R\$ 74.681,94**.

Página 1 de 1 102 palavras 100%

Agora que já acertamos tudo o que era necessário para automatizar os nossos gráficos, vamos colá-los nas posições certas no arquivo Word.



Primeiro precisamos saber qual é o nome do gráfico em questão. Para isso, basta selecionar o gráfico e olhar o nome que aparece na Caixa de Nome.

Caso essa caixa não esteja aparecendo para você, vá na guia Exibir e marque a opção Barra de Fórmulas.

```
Else  
    ActiveChart.FullSeriesCollection(1).Name = "-"  
End If  
  
ActiveChart.FullSeriesCollection(2).Name = ano  
  
selecao.Find.Text = "#GRAFICO1"  
selecao.Find.Replacement.Text = ""  
selecao.Find.Execute  
  
ActiveSheet.ChartObjects("Gráfico 5").Copy  
  
selecao.PasteSpecial DataType:=wdPasteBitmap  
  
|  
  
End Sub
```

Primeiro precisamos encontrar o texto "#GRAFICO1" para então substituí-lo por nada (pois queremos na verdade colar o gráfico por cima).

Isso fará com que o cursor do Word se posicione exatamente onde queremos colar o gráfico. Sabemos a posição do cursor graças à variável **selecao** que já definimos anteriormente.

Em seguida, copiamos o gráfico e colamos sobre a variável selecao. Repare no tipo que usamos para colar: wdPasteBitmap. Este comando é importante para o caso de você querer trabalhar apenas com o arquivo em Word, pois apenas o PasteSpecial irá colar o gráfico vinculado à planilha, o que pode ser ruim. Porém, como no nosso caso iremos salvar o arquivo como PDF, isso não fará diferença.

Relatório Mensal.docx - Word

Arquivo Página Inicial Inserir Design Layout Referências Correspondências Revisão Exibir O que você deseja fazer... Entrar Compartilhar

Recortar Copiar Colar Pincel de Formatação Área de Transferência

Fonte Parágrafo Estilo

Normal Sem Espaçamento Título 1 Título 2 Subtítulo Ênfase Sutil

Localizar Substituir Selecionar

Relatório de Vendas Novembro/2017

O mês de **Novembro** fechou com R\$ 60.404,00 de faturamento, o que representa **um aumento** com relação ao mês anterior no valor de **R\$ 4.504,00**.

A proporção de faturamento entre os produtos da empresa foi a seguinte:

The pie chart displays the following data:

Produto	Porcentagem
Celular	41%
Notebook	23%
Tablet	9%
Televisão	20%
Relógio	7%

Até o mês de **Novembro** a empresa acumulou um faturamento total de **R\$ 410.804,00** no ano, distribuído da seguinte forma:

#GRAFICO2

Página 1 de 1 101 palavras

Ao executar o código já temos o resultado em Word, mostrado ao lado.

```
Else  
    ActiveChart.FullSeriesCollection(1).Name = "-"  
End If  
  
ActiveChart.FullSeriesCollection(2).Name = ano  
  
'--- Copia e cola gráfico 1 ---  
selecao.Find.Text = "#GRAFICO1"  
selecao.Find.Replacement.Text = ""  
selecao.Find.Execute  
  
ActiveSheet.ChartObjects("Gráfico 5").Copy  
selecao.PasteSpecial DataType:=wdPasteBitmap  
  
'--- Copia e cola gráfico 2 ---  
selecao.Find.Text = "#GRAFICO2"  
selecao.Find.Replacement.Text = ""  
selecao.Find.Execute  
  
ActiveSheet.ChartObjects("Gráfico 4").Copy  
selecao.PasteSpecial DataType:=wdPasteBitmap  
  
'--- Copia e cola gráfico 3 ---  
selecao.Find.Text = "#GRAFICO3"  
selecao.Find.Replacement.Text = ""  
selecao.Find.Execute  
  
ActiveSheet.ChartObjects("Gráfico 7").Copy  
selecao.PasteSpecial DataType:=wdPasteBitmap  
  
|  
  
End Sub
```

Este processo de copiar e colar também vale para os outros dois gráficos, e o código final está mostrado ao lado. A estrutura será exatamente igual para os 3, mudando apenas a palavra-chave a ser buscada no Word, bem como o nome do gráfico no arquivo Excel.

Módulo 13 – Integração com Word – Criação de Relatório Automático (Resolução Parte 13)

532

**Relatório Mensal.docx - Word**

O mês de **Novembro** fechou com **R\$ 60.404,00** de faturamento, o que representa um aumento com relação ao mês anterior no valor de **R\$ 4.504,00**.

A proporção de faturamento entre os produtos da empresa foi a seguinte:

**Novembro**

Categoria	Porcentagem
Computador	41%
Monitor	20%
Teclado	20%
Mouse	19%

Até o mês de **Novembro**, a empresa acumulou um faturamento total de **R\$ 410.804,00** no ano, distribuído da seguinte forma:

**2017**

Mês	Faturamento (R\$)
JANEIRO	10.000,00
FEVEREIRO	10.000,00
MARÇO	10.000,00
ABRIL	10.000,00
MAYO	10.000,00
JUNHO	10.000,00
JULHO	10.000,00
AGOSTO	10.000,00
SETEMBRO	10.000,00
OUTUBRO	10.000,00
NOVEMBRO	10.000,00

Com relação ao ano anterior, no qual a empresa obteve um faturamento acumulado até o mês de **Novembro**, totalizando **R\$ 0,00**, houve um crescimento de **R\$ 410.804,00**.

O comparativo com relação ao ano anterior está indicado no gráfico

**Comparativo**

Mês	2016 (R\$)	2017 (R\$)
JANEIRO	10.000,00	10.000,00
FEVEREIRO	10.000,00	10.000,00
MARÇO	10.000,00	10.000,00
ABRIL	10.000,00	10.000,00
MAYO	10.000,00	10.000,00
JUNHO	10.000,00	10.000,00
JULHO	10.000,00	10.000,00
AGOSTO	10.000,00	10.000,00
SETEMBRO	10.000,00	10.000,00
OUTUBRO	10.000,00	10.000,00
NOVEMBRO	10.000,00	10.000,00

O resultado final é mostrado ao lado.

```
ActiveSheet.ChartObjects("Gráfico 5").Copy
selecao.PasteSpecial DataType:=wdPasteBitmap

'--- Copia e cola gráfico 2 ---
selecao.Find.Text = "#GRAFICO2"
selecao.Find.Replacement.Text = ""
selecao.Find.Execute

ActiveSheet.ChartObjects("Gráfico 4").Copy
selecao.PasteSpecial DataType:=wdPasteBitmap

'--- Copia e cola gráfico 3 ---
selecao.Find.Text = "#GRAFICO3"
selecao.Find.Replacement.Text = ""
selecao.Find.Execute

ActiveSheet.ChartObjects("Gráfico 7").Copy
selecao.PasteSpecial DataType:=wdPasteBitmap

'--- Salvar arquivo como PDF na pasta de relatórios
nome_arquivo = ThisWorkbook.Path & "\Relatórios\Relatório " & mes & "-" & ano & ".pdf"
arquivo_relatorio.ExportAsFixedFormat outputfilename:=nome_arquivo, ExportFormat:=wdExportFormatPDF

|
|  
End Sub
```

Por fim, falta agora salvar o arquivo como PDF.

Queremos salvar na pasta Relatórios, no mesmo caminho do arquivo Excel onde estamos criando o nosso código.

Criamos portanto uma variável nome\_arquivo para armazenar o caminho completo do arquivo a ser salvo, incluindo a extensão “.pdf” ao final do nome do arquivo.

Para salvar o arquivo como PDF, utilizamos a última linha de código mostrada ao lado.

```
selecao.Find.Execute

ActiveSheet.ChartObjects("Gráfico 5").Copy
selecao.PasteSpecial DataType:=wdPasteBitmap

'--- Copia e cola gráfico 2 ---
selecao.Find.Text = "#GRAFICO2"
selecao.Find.Replacement.Text = ""
selecao.Find.Execute

ActiveSheet.ChartObjects("Gráfico 4").Copy
selecao.PasteSpecial DataType:=wdPasteBitmap

'--- Copia e cola gráfico 3 ---
selecao.Find.Text = "#GRAFICO3"
selecao.Find.Replacement.Text = ""
selecao.Find.Execute

ActiveSheet.ChartObjects("Gráfico 7").Copy
selecao.PasteSpecial DataType:=wdPasteBitmap

'--- Salvar arquivo como PDF na pasta de relatórios
nome_arquivo = ThisWorkbook.Path & "\Relatórios\Relatório " & mes & "-" & ano & ".pdf"
arquivo_relatorio.ExportAsFixedFormat outputfilename:=nome_arquivo, ExportFormat:=wdExportFormatPDF

arquivo_relatorio.Close savechanges:=wdDoNotSaveChanges

MsgBox ("Relatório criado com sucesso!")

objeto_word.Visible = False

End Sub
```

Por fim, o arquivo Word deverá ser fechado sem alterações, pois ele é o nosso modelo a ser utilizado para a criação de outros relatórios.

Podemos também exibir uma mensagem indicando que a macro foi executada e o relatório gerado com sucesso,

Por fim, tornamos a nossa aplicação Word não mais visível, pois já terminamos de criar e salvar o relatório.

The screenshot shows a Microsoft Excel spreadsheet titled "Vendas.xlsx - Excel". The main table displays sales data for various products across four years (2017, 2018, 2019, 2020). A callout box highlights the "Gerar Relatório" button. A message box in the center says "Relatório criado com sucesso!" (Report created successfully!). The ribbon at the top has the "Ferramentas de Gráfico" tab selected. In the bottom left, there's a code editor window showing VBA code related to the report generation.

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1		Julho	2019		Mês	2017	2018	2019											
2		Celular	R\$ 30.361		Janeiro	R\$ 24.400	R\$ 21.472	R\$ 23.190											
3		Tablet	R\$ 6.665		Fevereiro	R\$ 21.000	R\$ 23.310	R\$ 32.634											
4		Notebook	R\$ 17.032		Marco	R\$ 25.300	R\$ 24.288	R\$ 27.203											
5		Relogio	R\$ 5.184		Abrii	R\$ 30.700	R\$ 38.068	R\$ 45.682											
6		Televisão	R\$ 14.810		Maio	R\$ 17.000	R\$ 15.810	R\$ 20.079											
7		Total:	R\$ 74.050		Junho	R\$ 38.300	R\$ 39.832	R\$ 50.985											
8					Julho	R\$ 52.800	R\$ 53.328	R\$ 46.395											
9					Agosto	R\$ 37.400	R\$ 46.75												
10					Setembro	R\$ 47.600	R\$ 50.9												
11					Outubro	R\$ 55.900	R\$ 76.5												
12					Novembro	R\$ 60.404	R\$ 72.4												
13					Dezembro	R\$ 82.800	R\$ 103.5												
14																			
15	#NOME_MES	Julho																	
16	#ANO	2019																	
17	#FATURAMENTO	R\$ 46.395,36																	
18	#SITUACAO_MES	uma diminuição																	
19	#COMPARATIVO_MES	R\$ 4.589,60																	
20	#FAT_ACUMULADO	R\$ 246.166,94																	
21	#FAT_ANOANTERIOR	R\$ 216.108,00																	
22	#SITUACAO_ANO	um crescimento																	
23	#COMPARATIVO_ANO	R\$ 30.058,94																	
24																			
25																			
26																			
27																			
28																			
29																			
30																			

Finalmente, a nossa macro está completa.

Com isso, fechamos o módulo com uma ferramenta totalmente avançada e extremamente útil para o dia a dia de uma empresa.

Como sugestão, tente refazer todo este código sozinho para treinar os seus conhecimentos. É muito importante que você pratique diversas vezes para garantir que todos os conceitos estão claros. Lembrando que você pode consultar todo o passo a passo sempre que precisar!

Módulo 14

# Integração VBA com Outlook

The screenshot shows a Microsoft Excel spreadsheet titled "Enviar E-mail.xlsxm - Excel". The table has columns A, B, and C. Column A contains email addresses, column B contains vendor names, and column C contains the body of the email. An orange button labeled "Enviar Relatórios" is overlaid on the right side of the table.

	A	B	C	D	E	F
1	E-mail	Vendedor	Corpo			
2	<a href="mailto:joaoprli@poli.ufrj.br">joaoprli@poli.ufrj.br</a>	João Paulo	segue em anexo o relatório de vendas com o seu desempenho neste ano.			
3	<a href="mailto:sergio@hashtagtreinamentos.com">sergio@hashtagtreinamentos.com</a>	Sergio Tranjan	segue em anexo o relatório de vendas com o seu desempenho neste ano.			
4	<a href="mailto:jessica.hollander@uol.com.br">jessica.hollander@uol.com.br</a>	Jessica Hollander	segue em anexo o relatório de vendas com o seu desempenho neste ano.			
5	<a href="mailto:d.amorim.santos96@gmail.com">d.amorim.santos96@gmail.com</a>	Diego Amorim	segue em anexo o relatório de vendas com o seu desempenho neste ano.			
6	<a href="mailto:luiza.franca@gmail.com">luiza.franca@gmail.com</a>	Luiza França	segue em anexo o relatório de vendas com o seu desempenho neste ano.			
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						

Neste módulo veremos como integrar o Outlook com o VBA. Mas o que é o Outlook?

O Microsoft Outlook é um programa da Microsoft, que está integrado no pacote Office. Ele fornece serviço gratuito de e-mails da Microsoft, assim como o Gmail é o serviço de e-mails do Google.

Assim, basicamente a ideia agora é integrar o VBA ao Outlook para que seja possível automatizar e envio de e-mails para diferentes destinatários, assim como enviar anexos, enviar em cópia, cópia oculta, e por ai vai @

The screenshot shows a Microsoft Excel spreadsheet titled "Enviar E-mail.xlsxm - Excel". The spreadsheet contains two main sections:

- Top Section (Table A1):** A table with columns A, B, and C. Column A lists emails, column B lists vendors, and column C lists the body of the email. An orange button labeled "Enviar Relatórios" is positioned to the right of the table.
- Bottom Section (Table 11):** A table showing file attachments. It includes columns for Nome (Name), Data de modificação (Last modified date), Tipo (Type), and Tamanho (Size). One attachment, "Relatórios Vendas", is highlighted.
- Right Panel:** A preview pane showing five Excel files: "Vendas - Diego Amorim.xlsx", "Vendas - Jessica Hollander.xlsx", "Vendas - João Paulo.xlsx", "Vendas - Luiza França.xlsx", and "Vendas - Sergio Tranjan.xlsx".

A ideia do nosso primeiro exercício é criar um código capaz de enviar e-mails automáticos para todos aqueles que estão na nossa lista de e-mail ao lado.

Além disso, também queremos enviar anexos a esses destinatários, que estão disponíveis na própria pasta do exercício.

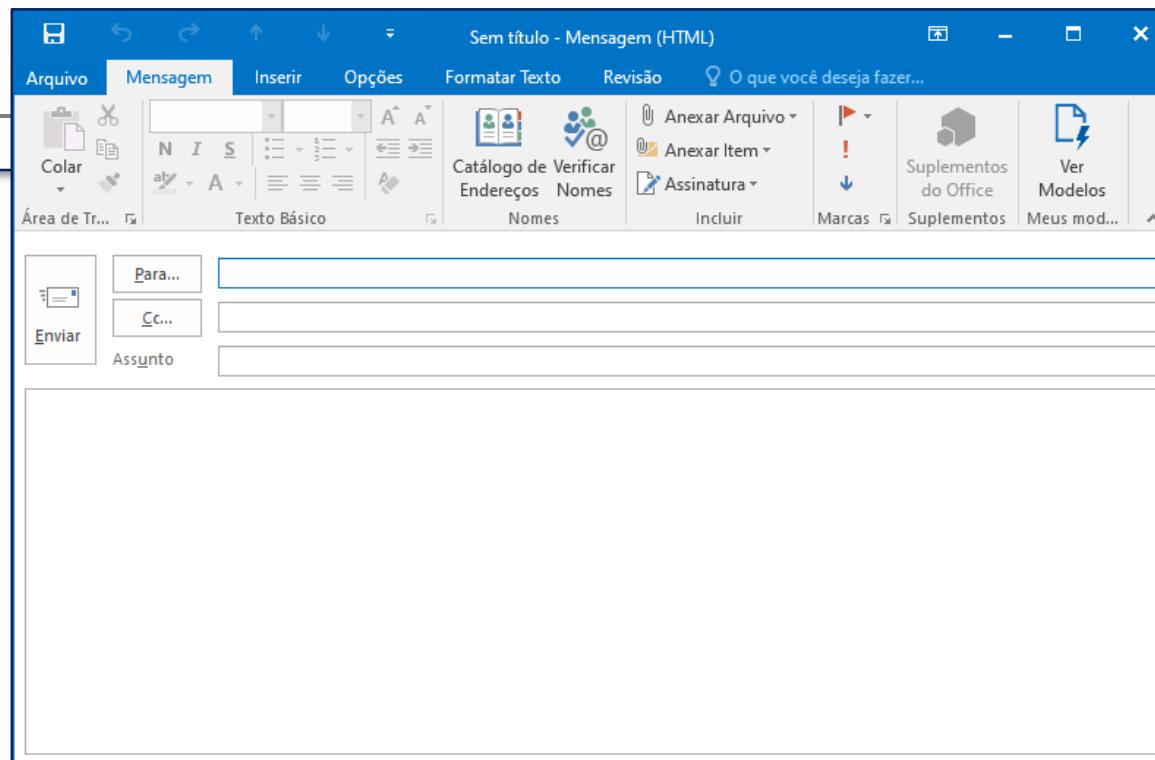
```
Sub envia_email()

Set objeto_outlook = CreateObject("Outlook.Application")

Set Email = objeto_outlook.CreateItem(0)

Email.Display

End Sub
```



Para começar o nosso código, temos que começar declarando o nosso objeto para armazenar o programa do Outlook.

Para criar a caixa de e-mail mostrada abaixo, onde vamos preencher todas as informações necessárias para enviar os e-mails, é necessário criar o objeto **Email**, por meio da propriedade **CreateItem**.

Além disso, também será necessário o comando **.Display** para que seja possível visualizar a caixa de e-mail.

Você pode criar um e-mail de qualquer servidor (incluindo o Gmail) para utilizar no Outlook.

```
' .Display  
' .To           -> Para quem vai mandar  
' .CC            -> Cópia  
' .BCC           -> Cópia Oculta  
' .Subject        -> Assunto  
' .HtmlBody       -> Corpo do e-mail  
' .Attachments.Add -> Anexos
```

Ao lado, temos uma breve colá de algumas propriedades importantes que precisaremos usar para conseguir informar para quem vamos enviar o e-mail, se o mesmo terá cópia, e assim vai.

```
Sub envia_email()

Set objeto_outlook = CreateObject("Outlook.Application")
Set Email = objeto_outlook.CreateItem(0)

Email.Display

Email.to = "mvcavalcanti@poli.ufrj.br"
Email.cc = "mvcavalcanti@ppe.ufrj.br"

Email.Subject = "Fala! E-mail de teste aqui"

Email.htmlbody = "Olá! Estou enviando este e-mail pelo excel para testar"

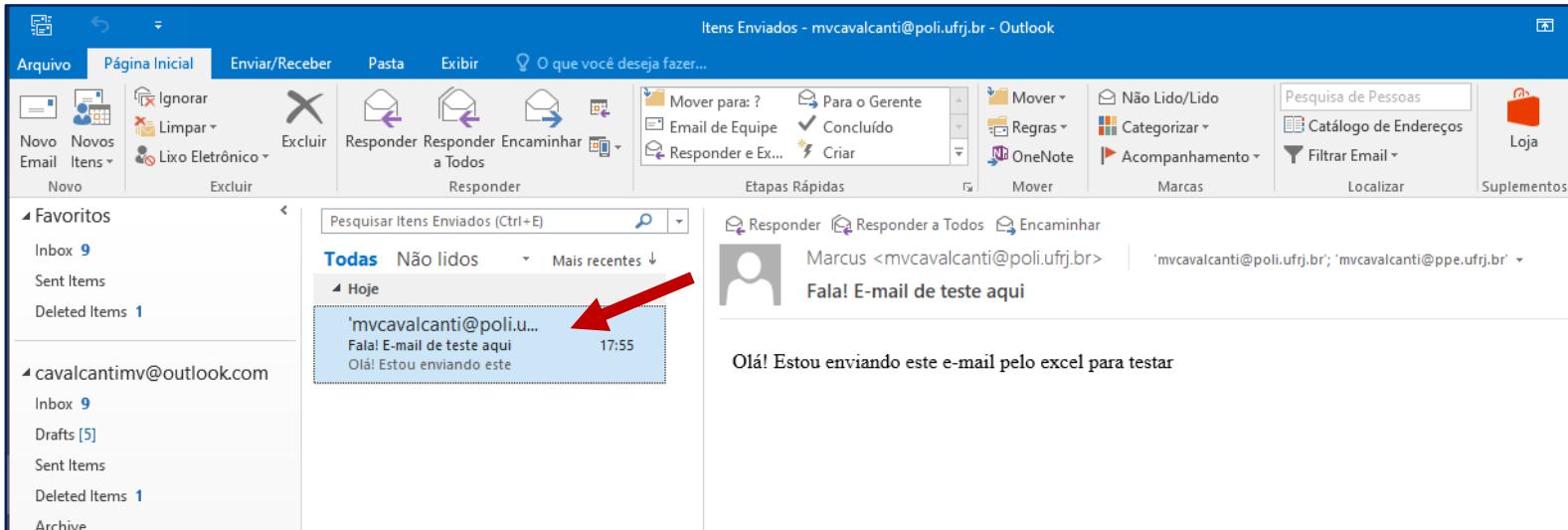
Email.send

|
|  
End Sub
```

Os comandos descritos anteriormente são usados no nosso código, como mostrado ao lado.

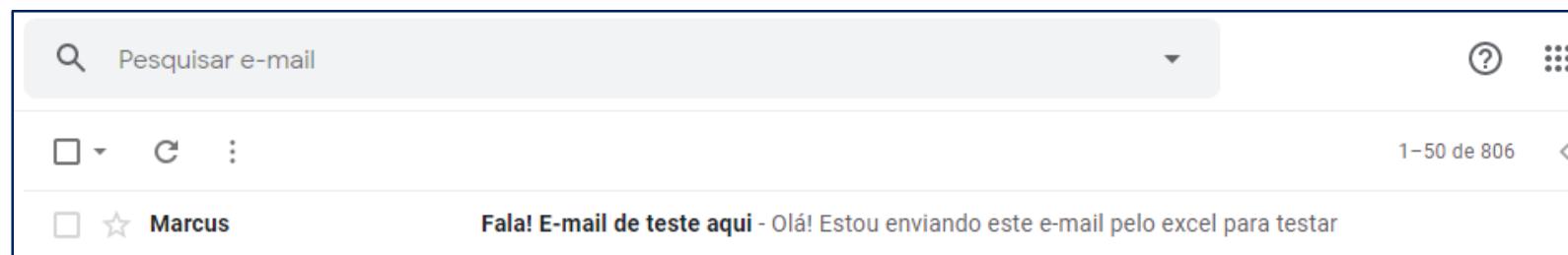
# Módulo 14 – Integração com Outlook – Preenchendo e enviando o primeiro e-mail

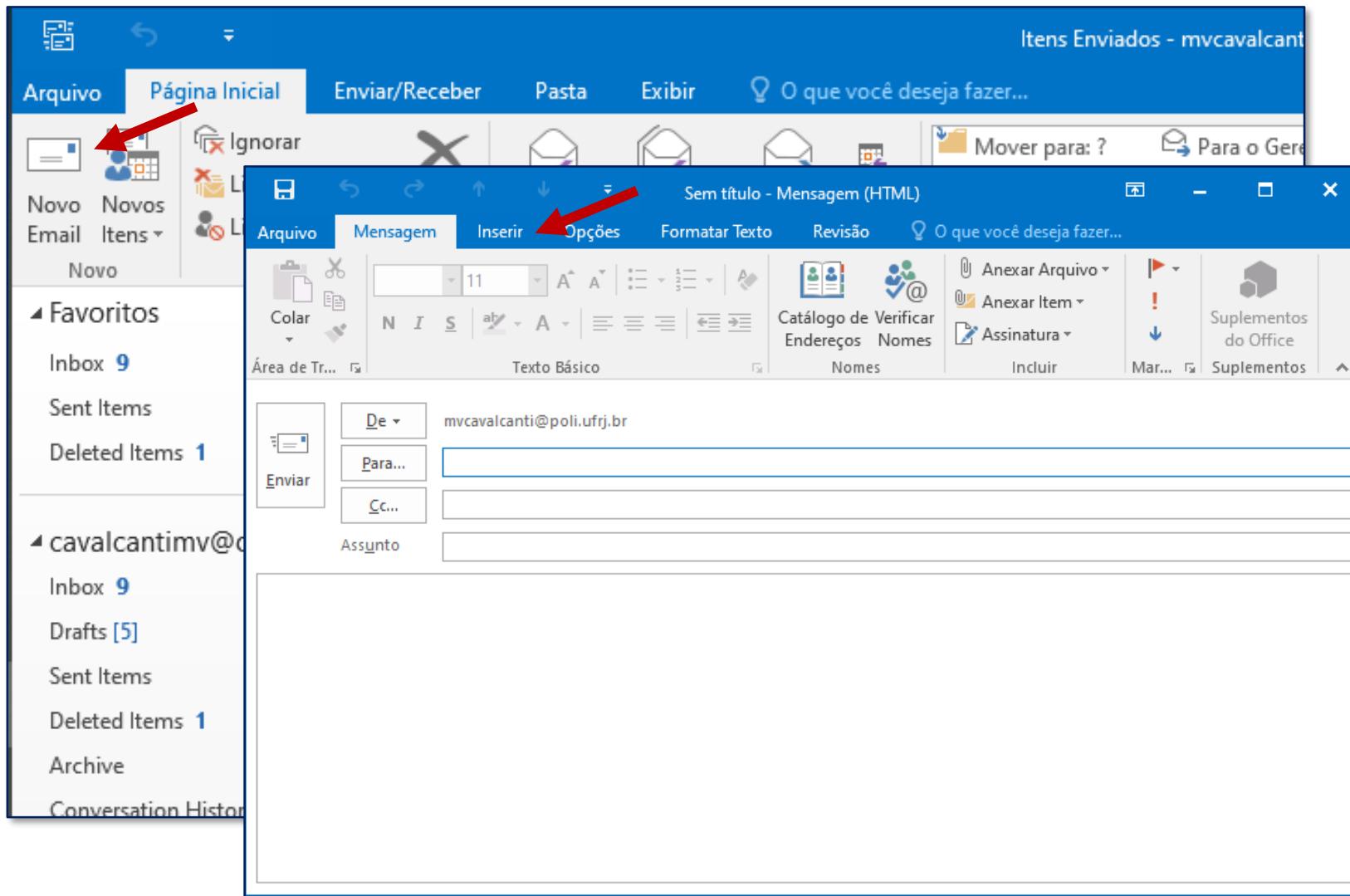
541



Após executar o código simples mostrado na página anterior, já seremos capazes de enviar um e-mail.

Podemos vê-lo na caixa de enviados do Outlook e também na Caixa de Entrada do e-mail para o qual enviamos o e-mail.

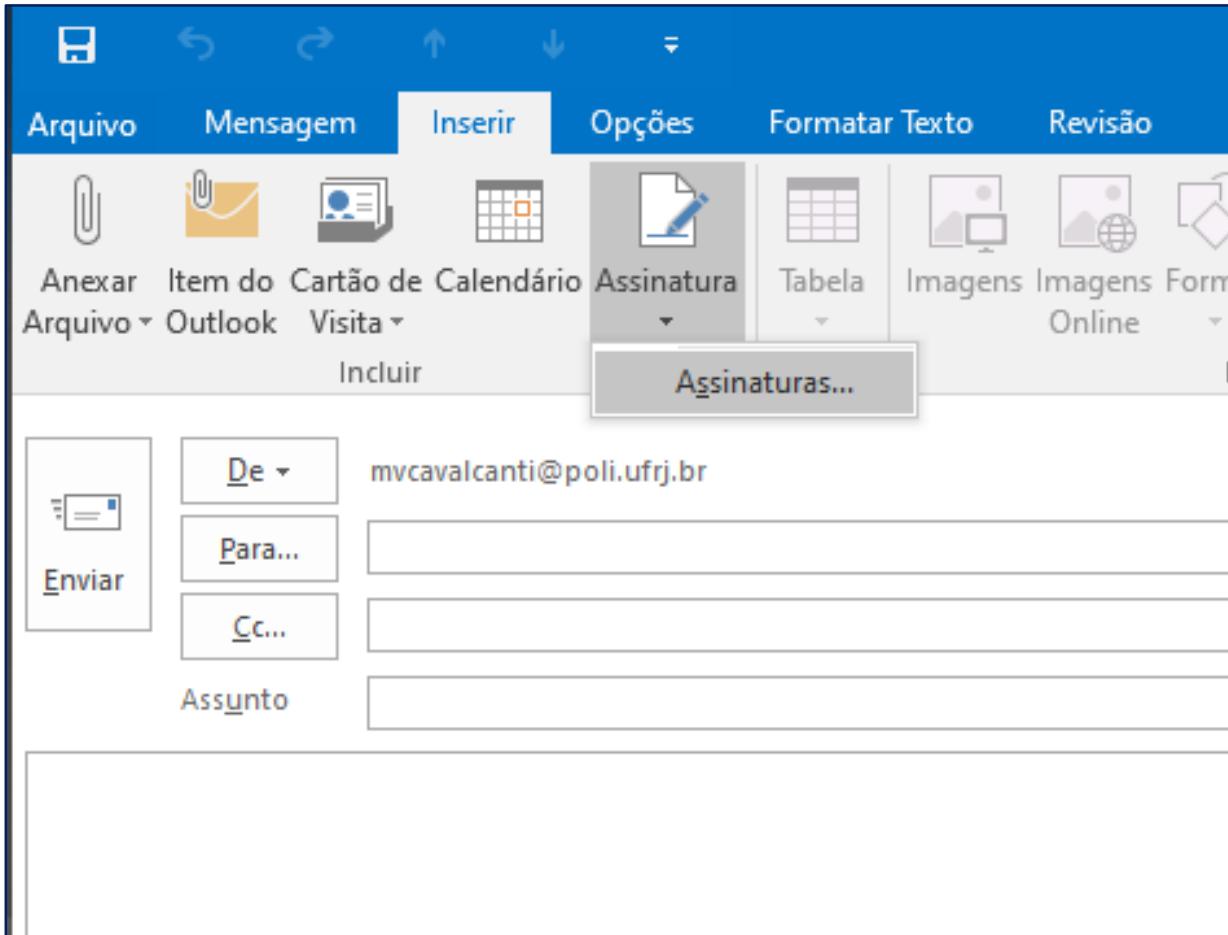




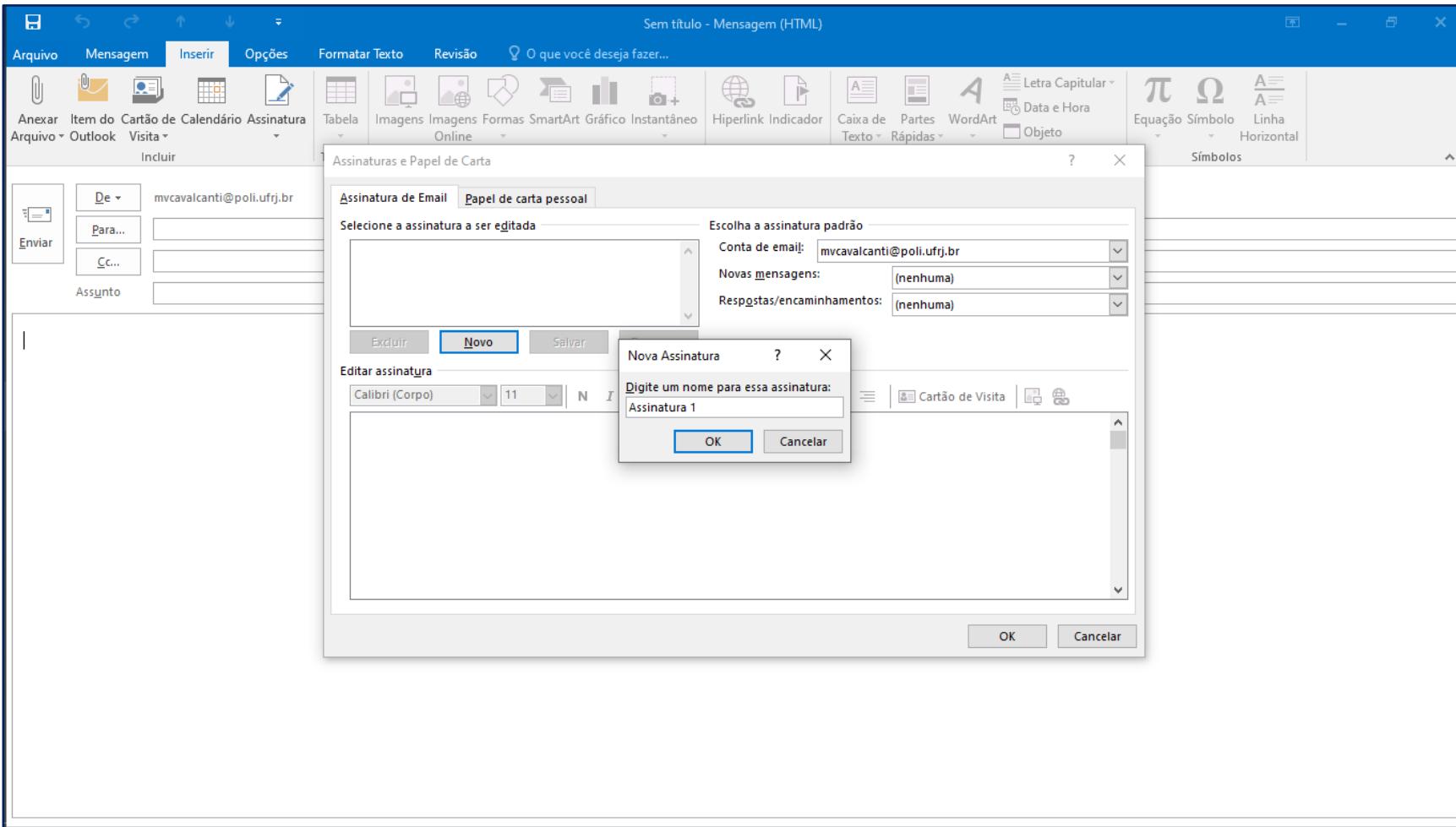
Vamos agora criar uma assinatura padrão para os nossos e-mails.

Uma assinatura basicamente é uma identificação personalizada criada pelo remetente do e-mail, geralmente contendo informações da empresa, telefone e e-mail para contato e logo da empresa.

Para criar uma nova assinatura, clique na opção de **Novo Email**. Em seguida, na opção que abrir, vá na guia **Inserir**.

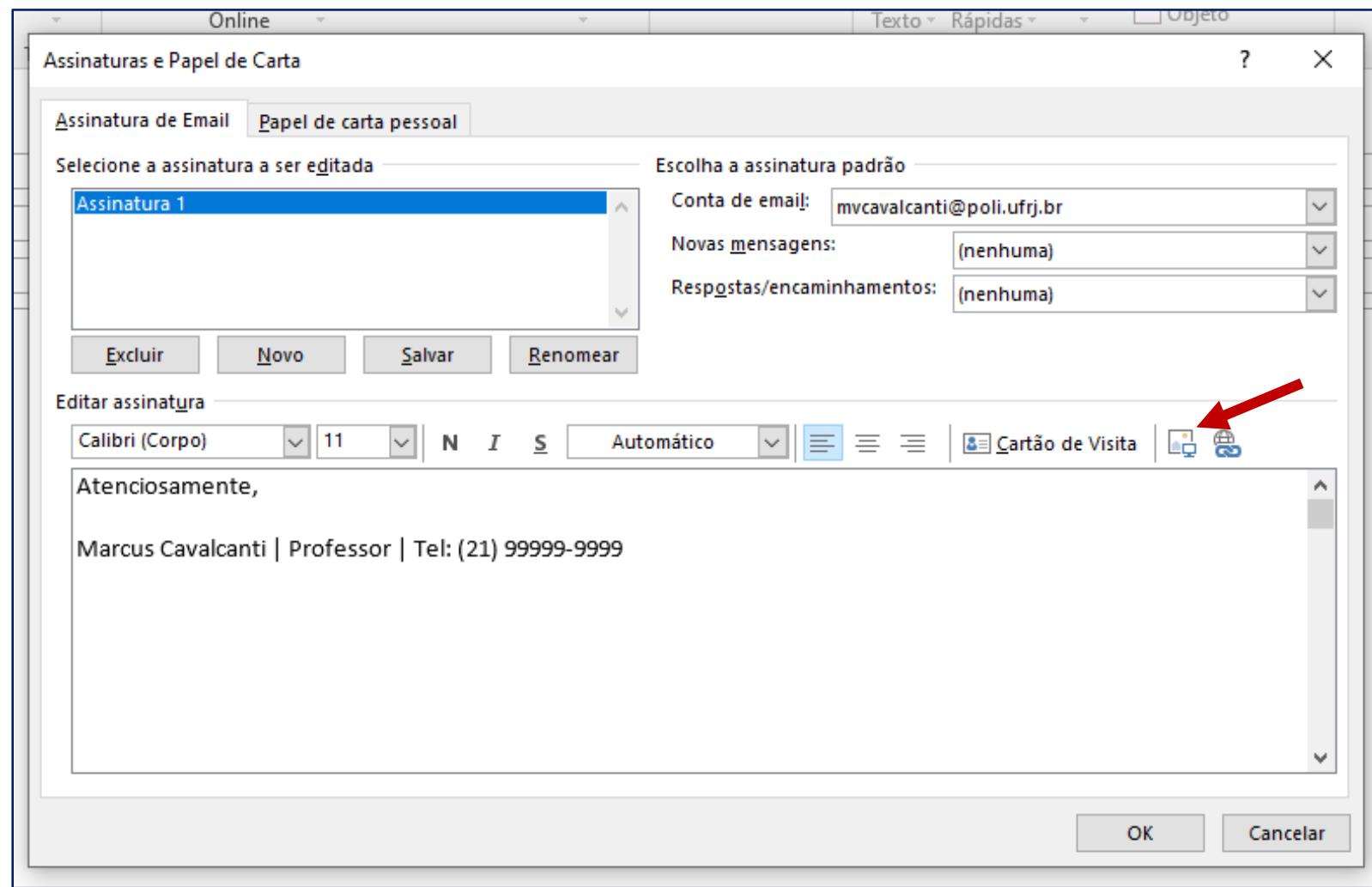


Clique na opção de Assinaturas para criar uma nova assinatura.



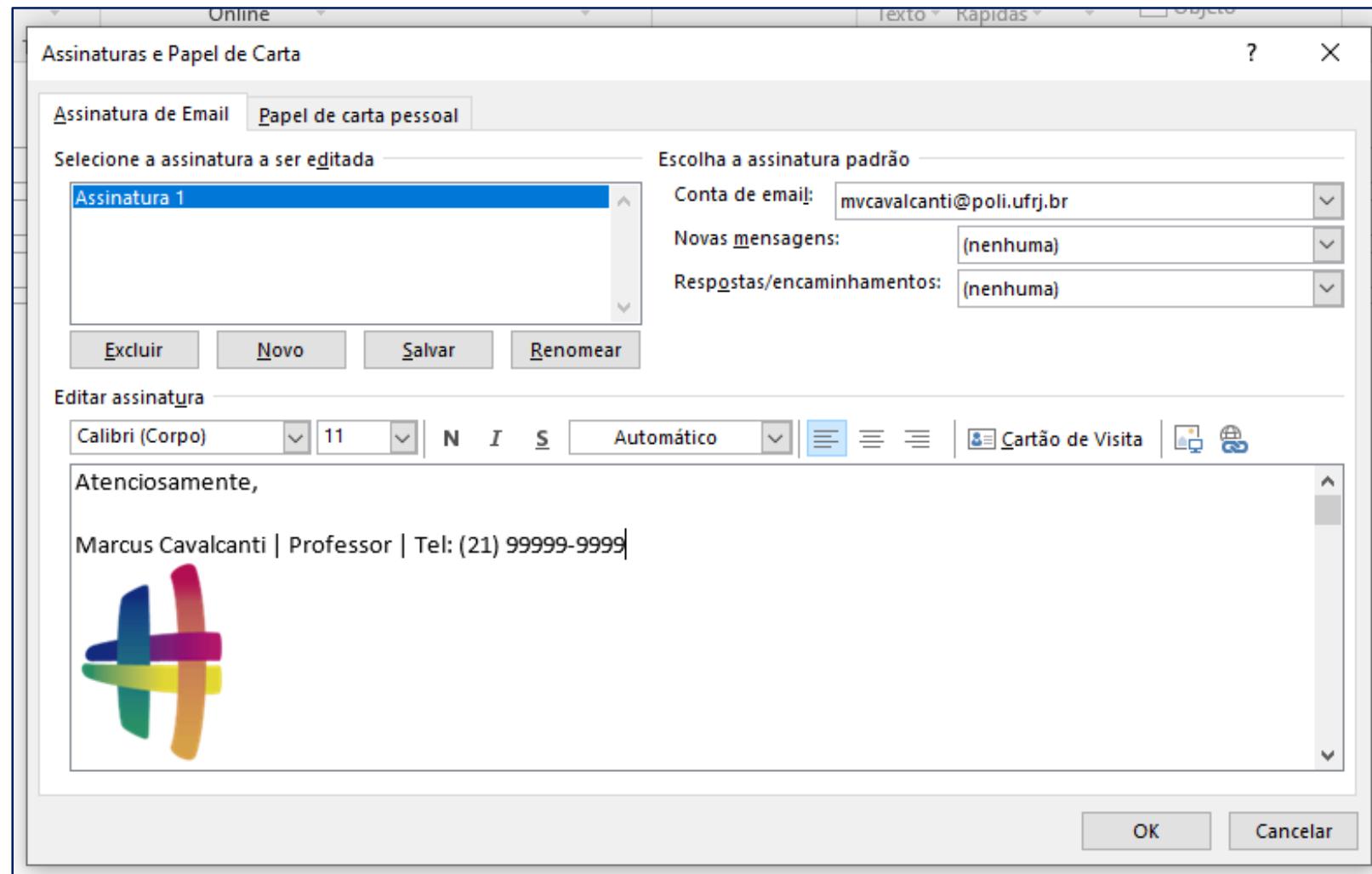
Clique na opção Novo.

Você pode chamar a sua assinatura de **Assinatura 1**. Em seguida, clique em Ok.

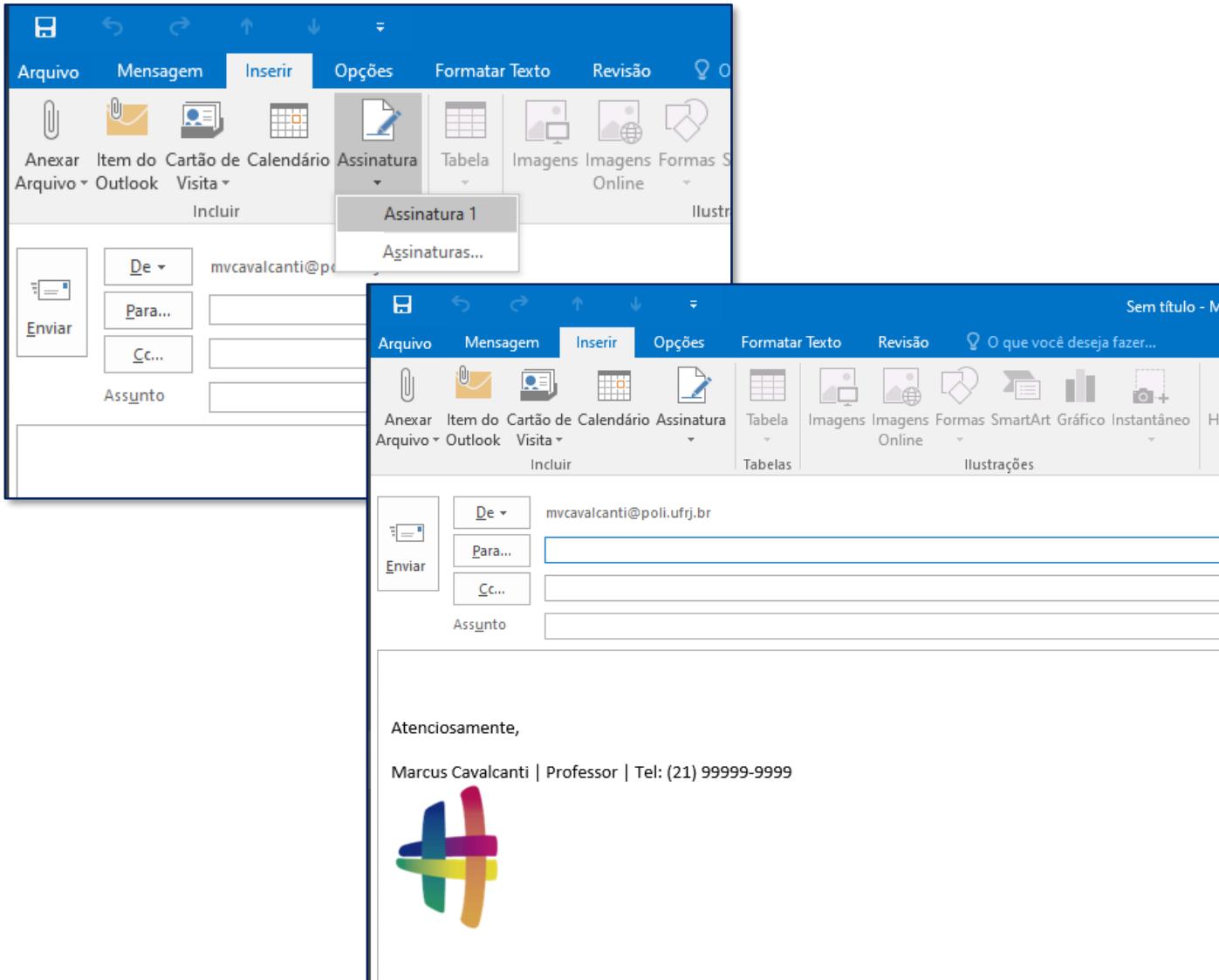


Escreva na caixa de texto abaixo o texto que você quer que apareça na sua assinatura.

Em seguida, clique na opção de Imagem indicada ao lado para inserir uma imagem personalizada. Deixamos uma imagem na pasta do exercício com a logo da hashtag que você pode usar como exemplo.



Em seguida, basta clicar em Ok.



Após criada, a assinatura já pode ser usada.

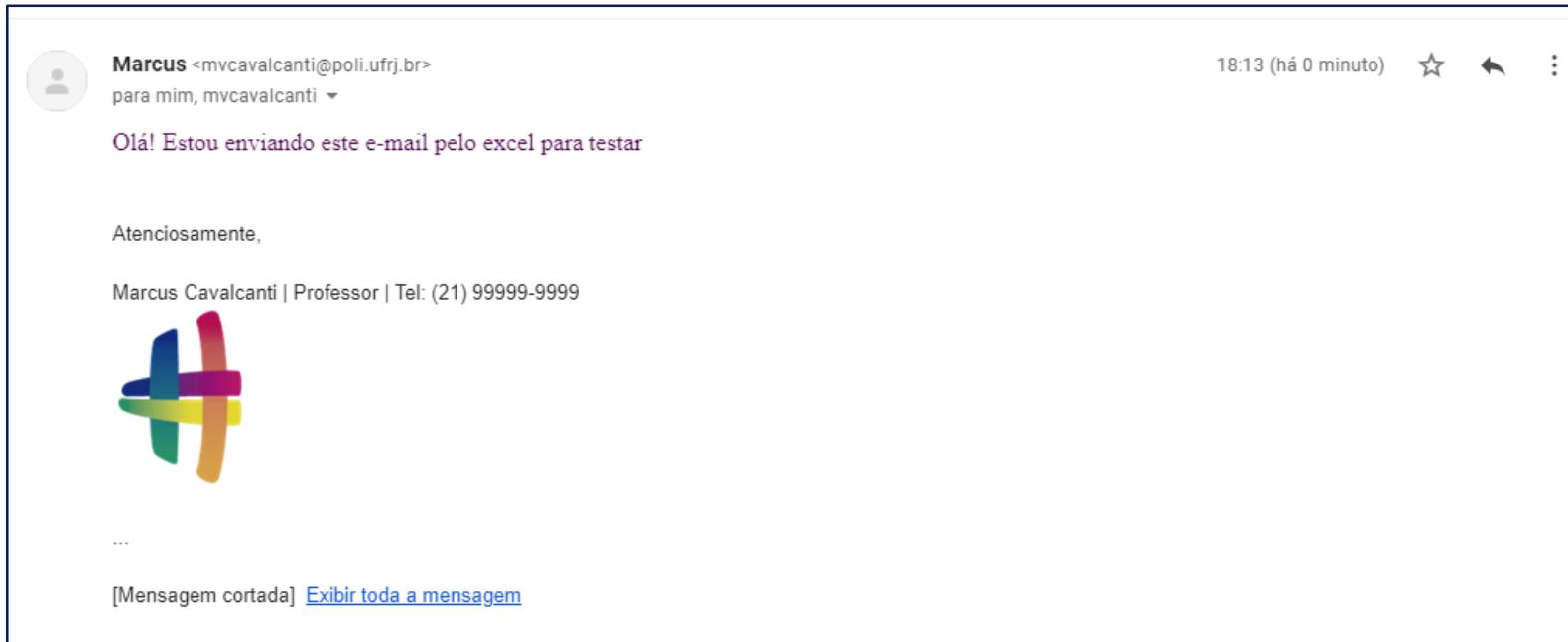
```
Sub envia_email()

Set objeto_outlook = CreateObject("Outlook.Application")
Set Email = objeto_outlook.CreateItem(0)
Email.Display
assinatura = Email.htmlbody
Email.to = "mvcavalcanti@poli.ufrj.br"
Email.cc = "mvcavalcanti@ppe.ufrj.br"
Email.Subject = "Fala! E-mail de teste aqui"
Email.htmlbody = "Olá! Estou enviando este e-mail pelo excel para testar" & assinatura
Email.send

End Sub
```

A partir de agora, sempre que a gente quiser utilizar a assinatura, basta abrir a caixa de e-mail e salvar o conteúdo do e-mail em uma variável, que podemos chamar de **assinatura**. Caso contrário, a assinatura que já aparece padrão será sobreescrita pela mensagem do corpo de e-mail (htmlbody).

Em seguida, podemos concatenar esta assinatura ao texto do corpo de e-mail.



The screenshot shows an Outlook email window. The recipient is Marcus <mvcavalcanti@poli.ufrj.br>, with a note to self "para mim, mvcavalcanti". The timestamp is 18:13 (há 0 minuto). The message body contains the text: "Olá! Estou enviando este e-mail pelo excel para testar", "Atenciosamente," and "Marcus Cavalcanti | Professor | Tel: (21) 99999-9999". Below the text is a colorful logo consisting of overlapping blue, green, yellow, and red shapes. At the bottom left, there is a link "[Mensagem cortada] Exibir toda a mensagem".

Após executar a macro, o e-mail será enviado, incluindo a assinatura.

## Módulo 14 – Integração com Outlook – Enviando vários E-mails

550

The screenshot shows a Microsoft Excel spreadsheet titled "Enviar E-mail.xlsxm - Excel". The table has columns A, B, and C. Column A contains email addresses, column B contains names, and column C contains a descriptive message. An orange button labeled "Enviar Relatórios" is overlaid on the right side of the table.

	A	B	C	D	E	F
1	E-mail	Vendedor	Corpo			
2	<a href="mailto:joaoprlira@poli.ufrj.br">joaoprlira@poli.ufrj.br</a>	João Paulo	segue em anexo o relatório de vendas com o seu desempenho neste ano.			
3	<a href="mailto:sergio@hashtagtreinamentos.com">sergio@hashtagtreinamentos.com</a>	Sergio Tranjan	segue em anexo o relatório de vendas com o seu desempenho neste ano.			
4	<a href="mailto:jessica.hollander@uol.com.br">jessica.hollander@uol.com.br</a>	Jessica Hollander	segue em anexo o relatório de vendas com o seu desempenho neste ano.			
5	<a href="mailto:d.amorim.santos96@gmail.com">d.amorim.santos96@gmail.com</a>	Diego Amorim	segue em anexo o relatório de vendas com o seu desempenho neste ano.			
6	<a href="mailto:luiza.franca@gmail.com">luiza.franca@gmail.com</a>	Luiza França	segue em anexo o relatório de vendas com o seu desempenho neste ano.			
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						

Agora a ideia é de fato considerar a lista de e-mails do nosso arquivo para que possamos enviar para todos os destinatários de uma vez.

```
Sub envia_email()

Set objeto_outlook = CreateObject("Outlook.Application")
ult_linha = Range("A1").End(xlDown).Row
For linha = 2 To ult_linha
    Set Email = objeto_outlook.CreateItem(0)
    Email.Display
    assinatura = Email.htmlbody
    Email.To = Cells(linha, 1).Value
    Email.Subject = "Relatório de Vendas"
    Email.htmlbody = "Olá " & Cells(linha, 2).Value & ", " & Cells(linha, 3).Value & "Abraço!" & assinatura
    Email.send
Next
End Sub
```

Como queremos enviar o e-mail repetidas vezes, então precisamos criar uma estrutura de repetição, percorrendo a linha 2 até a última linha preenchida da tabela.

Basicamente o que teremos que mudar é o destinatário (Email.to) e o corpo do e-mail (htmlbody) de acordo com cada destinatário.

## Módulo 14 – Integração com Outlook – Enviando vários E-mails

552

The screenshot shows the Microsoft Outlook interface with the ribbon menu at the top. The main window displays a list of sent emails under the 'Itens Enviados' tab. A red box highlights the list of recipients. The first recipient is Marcus <mvcavalcanti@poli.ufrj.br> with the message: 'Olá Sergio Tranjan, segue em anexo o relatório de vendas com o seu desempenho neste ano. Abraço!' The second recipient is 'sergio@hashtagtreinamentos.com'. Below these, several other recipients are listed with their messages. The right pane shows the details of the selected email to Sergio.

Arquivo Página Inicial Enviar/Receber Pasta Exibir O que você deseja fazer...

Novo Novos Email Itens Novo Novos Email Itens Ignorar Limpar Lixo Eletrônico Excluir Excluir Responder Responder a Todos Encaminhar a Todos Responder e Ex... Mover para: ? Para o Gerente Email de Equipe Concluído Mover Mover para: ? Não Lido/Lido Regras Categorizar OneNote Criar Pesquisa de Pessoas Catálogo de Endereços Loja Localizar Suplementos Enviar/Receber Todas as Pastas Enviar/Receber

Apartado Favoritos:

- Inbox 9
- Sent Items
- Deleted Items 1

Apartado cavalcantimv@outlook.com:

- Inbox 9
- Drafts [5]
- Sent Items
- Deleted Items 1
- Archive
- Conversation History
- Junk Email
- Outbox [2]
- RSS Feeds
- Pastas de Pesquisa

Apartado mvcavalcanti@poli.ufrj.br:

- Caixa de Entrada 805
- Itens Enviados
- Itens Excluídos
- Caixa de Saída
- Calendário

Itens: 7

Após executar a macro, todos os e-mails serão enviados de forma automática para cada destinatário da lista de e-mails.

Nome	Data de modificação	Tipo	Tamanho
Vendas - Diego Amorim.xlsx	03/01/2020 08:47	Planilha do Micro...	16 KB
Vendas - Jessica Hollander.xlsx	03/01/2020 08:47	Planilha do Micro...	16 KB
Vendas - João Paulo.xlsx	03/01/2020 08:47	Planilha do Micro...	17 KB
Vendas - Luiza França.xlsx	03/01/2020 08:47	Planilha do Micro...	16 KB
Vendas - Sergio Tranjan.xlsx	03/01/2020 08:47	Planilha do Micro...	16 KB

Agora queremos incluir os anexos da pasta de Relatórios. A ideia é que cada planilha seja enviada corretamente para cada destinatário.

```
Sub envia_email()

Set objeto_outlook = CreateObject("Outlook.Application")
ult_linha = Range("A1").End(xlDown).Row
For linha = 2 To ult_linha
    Set Email = objeto_outlook.CreateItem(0)
    Email.Display
    assinatura = Email.htmlbody
    Email.To = Cells(linha, 1).Value
    Email.Subject = "Relatório de Vendas"
    Email.htmlbody = "Olá " & Cells(linha, 2).Value & ", " & Cells(linha, 3).Value & "Abraço!" & assinatura
    Email.Attachments.Add (ThisWorkbook.Path & "\Relatórios Vendas\Vendas - " & Cells(linha, 2).Value & ".xlsx")
    Email.send
Next
End Sub
```

O código utilizado para enviar um anexo está indicado na imagem ao lado.

# Módulo 14 – Integração com Outlook – Colocando Anexo no E-mail

555

The screenshot shows the Microsoft Outlook interface with the ribbon menu at the top. The main window displays an email message sent from Marcus Cavalcanti to Sergio Tranjan. The message subject is 'Relatório de Vendas' and the body contains a greeting and the attached file 'Vendas - Sergio Tranjan...'. The file is described as '19 KB'. The Outlook ribbon tabs include Arquivo, Página Inicial, Enviar/Receber, Pasta, Exibir, and O que você deseja fazer... (Search). The left sidebar shows the navigation pane with sections like Favorites, Inbox (9), Sent Items, Deleted Items (1), and several other accounts listed. The bottom status bar shows 'Itens: 12'.

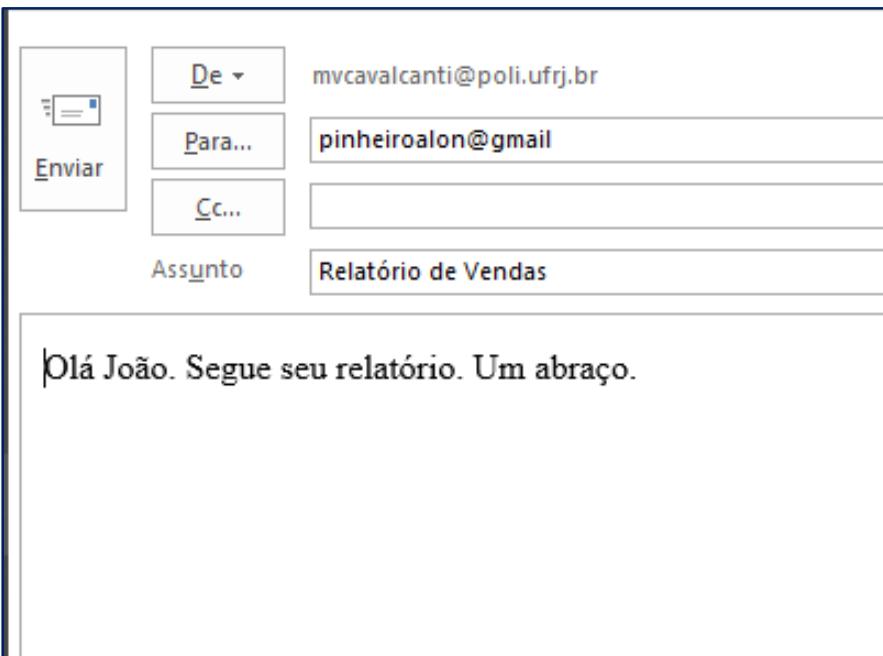
Já podemos executar o código anterior e os e-mails serão enviados com os seus devidos anexos.

```
Sub enviar_email()

Set objeto_outlook = CreateObject("Outlook.Application")
Set email = objeto_outlook.CreateItem(0)
email.Display
email.To = "pinheiroalon@gmail"
email.Subject = "Relatório de Vendas"
email.htmlbody = "Olá João. Segue seu relatório. Um abraço."
End Sub
```

Vamos agora fazer um parênteses com relação à formatação dos nossos textos no corpo de e-mail, dessa vez com o exemplo bem simples, mostrado ao lado.

Você deve ter reparado que, até agora, todos os nossos textos têm sido enviados sempre na mesma linha, como mostrado no corpo de e-mail abaixo.



```
Sub enviar_email()

Set objeto_outlook = CreateObject("Outlook.Application")

Set email = objeto_outlook.CreateItem(0)

email.Display

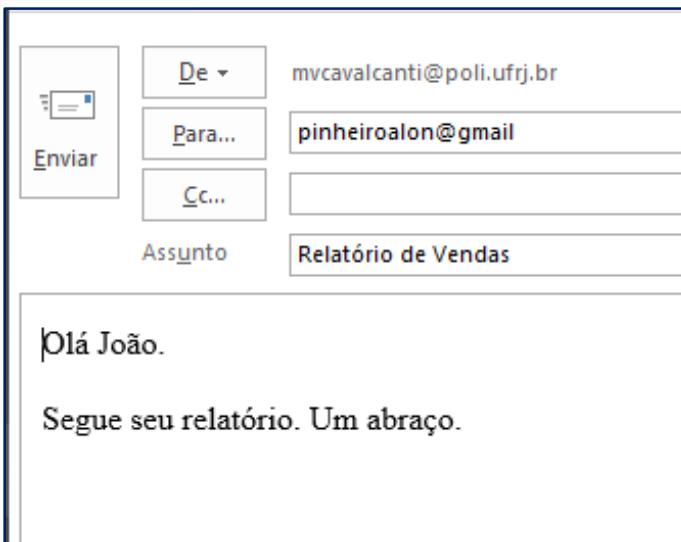
email.To = "pinheiroalon@gmail"

email.Subject = "Relatório de Vendas"

email.htmlbody = "<p>Olá João. </p>Segue seu relatório. Um abraço."

End Sub
```

Para poder fazer essa quebra de linha (ou criação de um novo parágrafo) precisamos utilizar um comando básico de HTML, que é a linguagem usada por trás do Outlook. Basicamente, o código para fazer com que determinado texto fique em um parágrafo a parte é a presença de um `<p>` e um `</p>` envolvendo o texto. Ao lado, temos um exemplo. O texto “Olá João” será mostrado em um parágrafo próprio graças aos comandos `<p>` e `</p>`.



```
Sub enviar_email()

Set objeto_outlook = CreateObject("Outlook.Application")

Set email = objeto_outlook.CreateItem(0)

email.Display

email.To = "pinheiroalon@gmail.com"

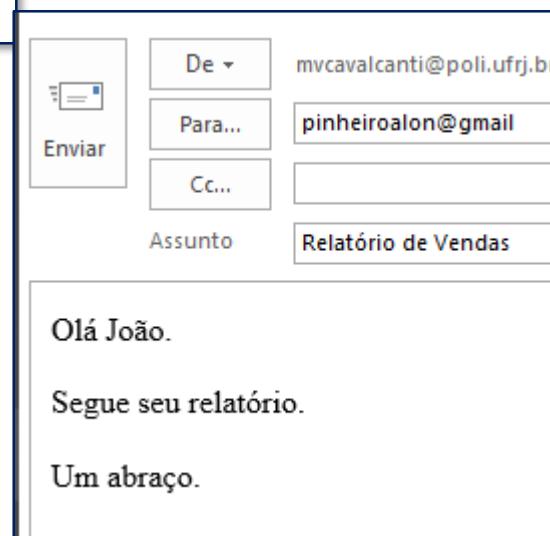
email.Subject = "Relatório de Vendas"

email.htmlbody = "<p>Olá João. </p>" & _
"<p>Segue seu relatório.</p>" & _
"<p>Um abraço.</p>"

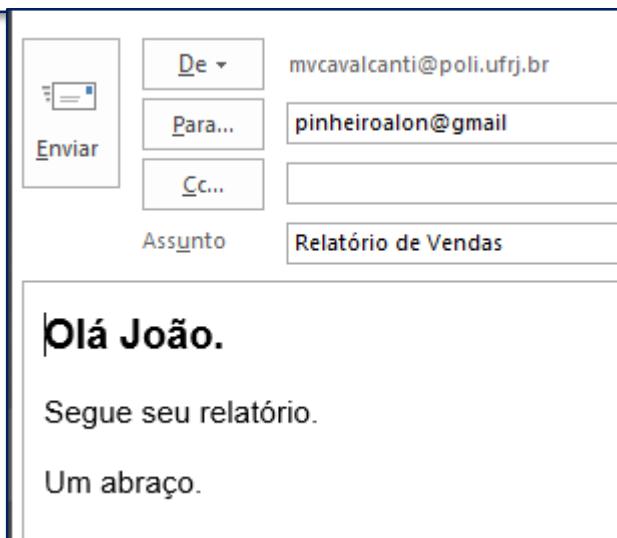
End Sub
```

Para separar cada linha em um novo parágrafo, basta seguir a mesma lógica.

Repare que concatenamos um underline ao final de cada frase. Isso é apenas um recurso do VBA que nos permite quebrar uma linha no código e melhorar a visualização dentro do ambiente VBA. **Entenda: não é quebrar uma linha do texto entre aspas, e sim quebrar uma linha no código criado no ambiente VBA.**



```
Sub enviar_email()
    Set objeto_outlook = CreateObject("Outlook.Application")
    Set email = objeto_outlook.CreateItem(0)
    email.Display
    email.To = "pinheiroalon@gmail.com"
    email.Subject = "Relatório de Vendas"
    email.htmlbody = "<p style=""font-size:20px;font-family:arial;font-weight:bold"">Olá João. </p>" &
                    "<p style=""font-size:15px;font-family:arial;font-weight:normal"">Segue seu relatório.</p>" &
                    "<p style=""font-size:15px;font-family:arial;font-weight:normal"">Um abraço.</p>"
End Sub
```



Se quisermos adicionar mais formatações a estes textos, podemos utilizar outros códigos, tais como mostrados na imagem ao lado. Você pode encontrar esses e outros códigos buscando por códigos de formatação em HTML para Outlook.

De qualquer forma, os mais utilizados são mostrados no exemplo:

1. Tamanho
2. Fonte
3. Negrito ou normal

```
Sub envia_email()

Set objeto_outlook = CreateObject("Outlook.Application")
ult_linha = Range("A1").End(xlDown).Row

For linha = 2 To ult_linha

    Set Email = objeto_outlook.CreateItem(0)
    Email.Display

    assinatura = Email.htmlbody

    Email.To = Cells(linha, 1).Value
    Email.Subject = "Relatório de Vendas"

    Email.htmlbody = "<p>Olá, " & Cells(linha, 2).Value & ".</p>" & _
                    "<p>" & Cells(linha, 3).Value & "</p>" & _
                    "<p>Um abraço, Marcus</p>" & _
                    assinatura

    Email.Attachments.Add (ThisWorkbook.Path & "\Relatórios Vendas\Vendas - " & Cells(linha, 2).Value & ".xlsx")
    Email.send

Next

End Sub
```

Trazendo para o nosso código principal as adaptações de criação de parágrafos em HTML mostradas nas páginas anteriores, o código é o mostrado ao lado.

# Módulo 14 – Integração com Outlook – Organizando o E-mail em Parágrafos

561

Relatório de Vendas - Mensagem (HTML)

Arquivo Mensagem Inserir Opções Formatar Texto Revisão O que você deseja fazer...

Recortar Copiar Colar Pincel de Formulação Área de Transferência

10 A A N I S ab A Texto Básico

Catálogo de Verificar Endereços Nomes Nomes Anexar Anexar Assinatura Arquivo Item Incluir Marcas Suplementos do Office Suplementos

De: mvcavalcanti@poli.ufrj.br  
Para: joaopririra@poli.ufrj.br  
Assunto: Relatório de Vendas

Olá, João Paulo.  
segue em anexo o relatório de vendas com o seu desempenho neste ano.  
Um abraço, Marcus

Atenciosamente,

Marcus Cavalcanti | Professor | Tel: (21) 99999-9999



E um novo resultado é mostrado ao lado.

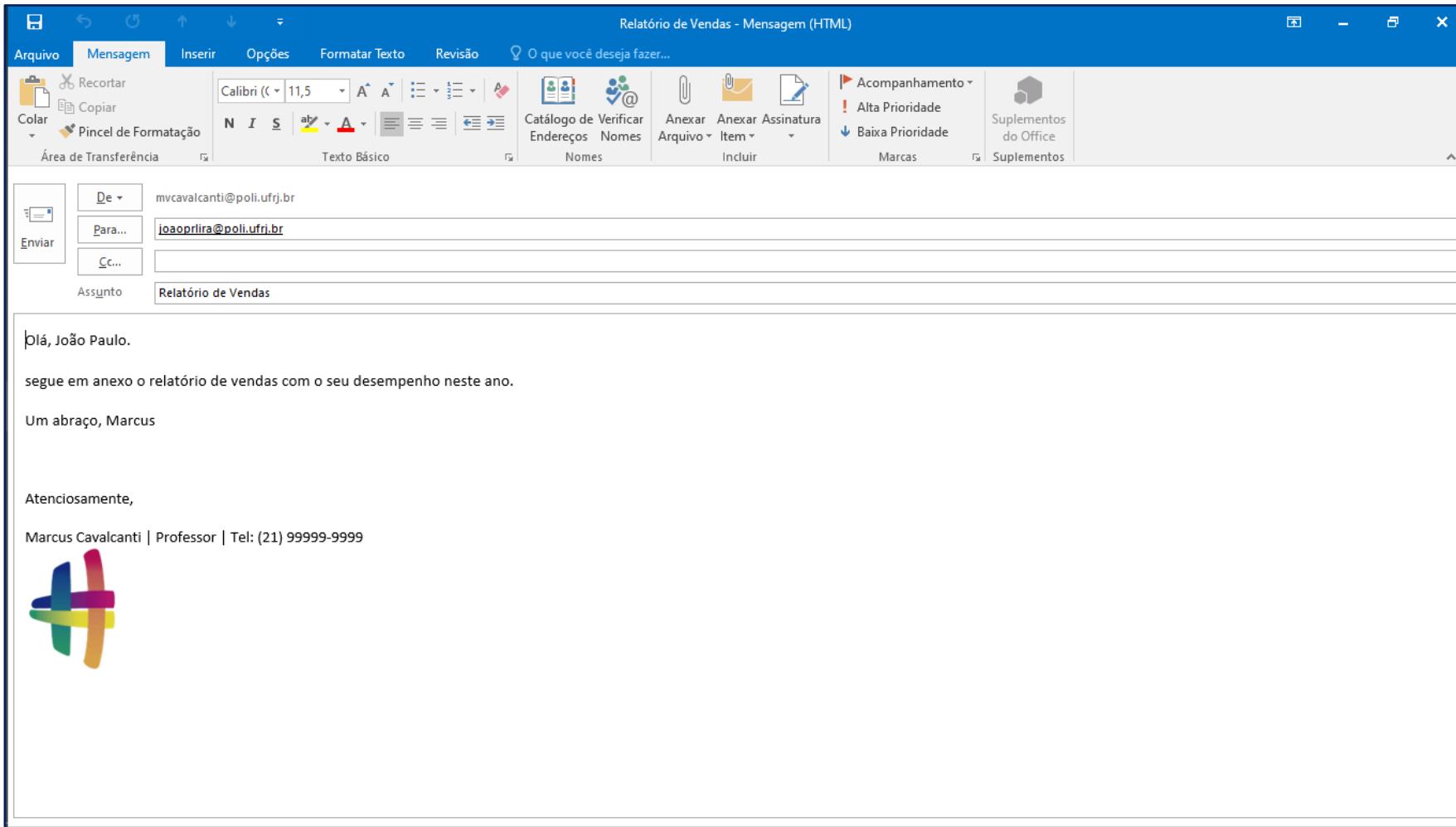
```
Sub envia_email()

Set objeto_outlook = CreateObject("Outlook.Application")
ult_linha = Range("A1").End(xlDown).Row
For linha = 2 To ult_linha
    Set Email = objeto_outlook.CreateItem(0)
    Email.Display
    assinatura = Email.htmlbody
    Email.To = Cells(linha, 1).Value
    Email.Subject = "Relatório de Vendas"
    Email.htmlbody = "<p style=""font-size:15px"">Olá, " & Cells(linha, 2).Value & ".</p>" & _
                    "<p style=""font-size:15px"">" & Cells(linha, 3).Value & "</p>" & _
                    "<p style=""font-size:15px"">Um abraço, Marcus</p>" & _
                    assinatura
    Email.Attachments.Add (ThisWorkbook.Path & "\Relatórios Vendas\Vendas - " & Cells(linha, 2).Value & ".xlsx")
    Email.send
Next
End Sub
```

Podemos também adicionar o código de tamanho da fonte para aumentar um pouco mais o tamanho das letras no corpo de e-mail.

Módulo 14 – Integração com Outlook – Formatando o Corpo do E-mail

563

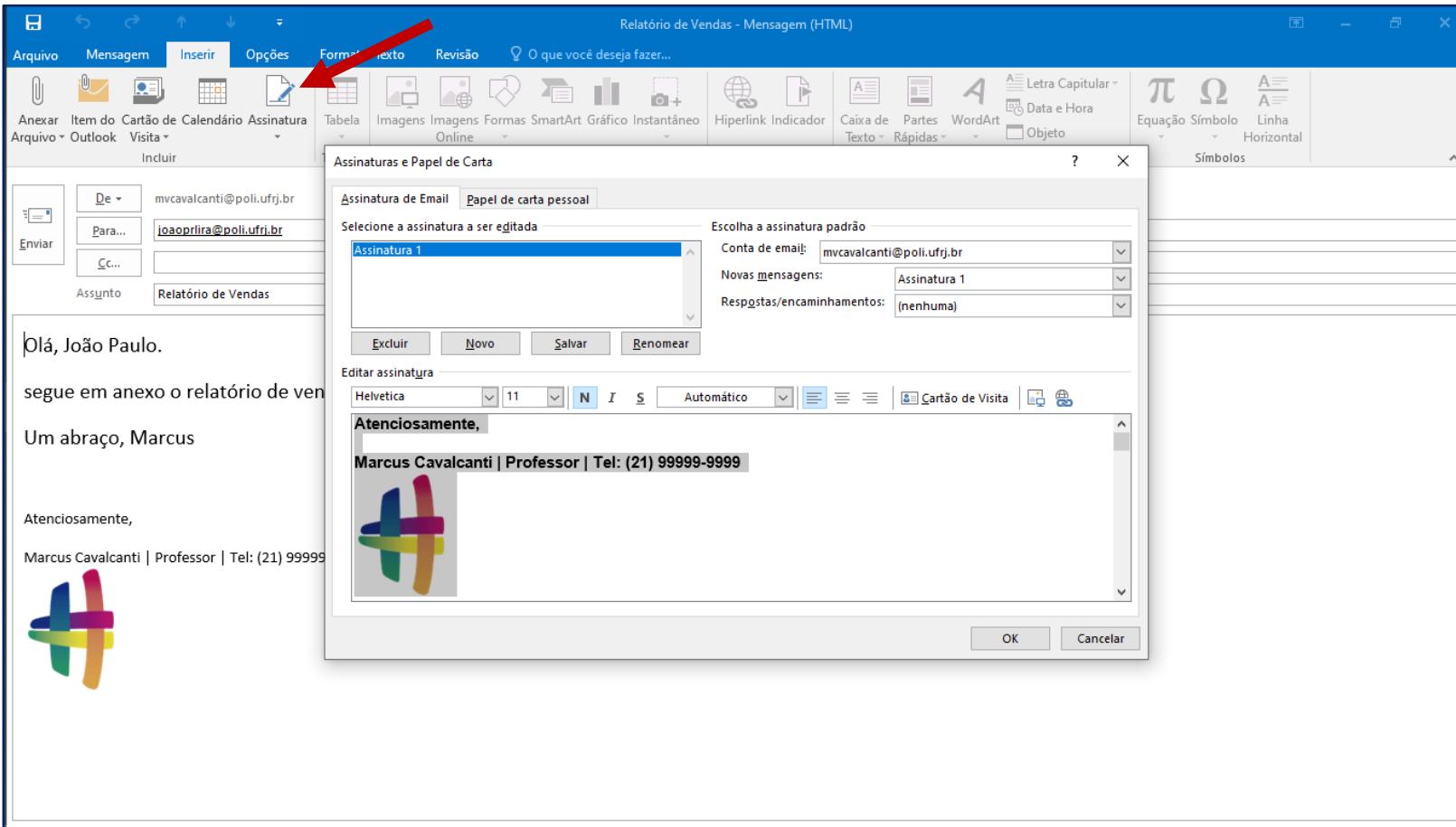


O resultado é mostrado ao lado.

Você pode também testar outros tamanhos. O 15px seria mais ou menos o equivalente ao tamanho 11,5 que estamos mais acostumados.

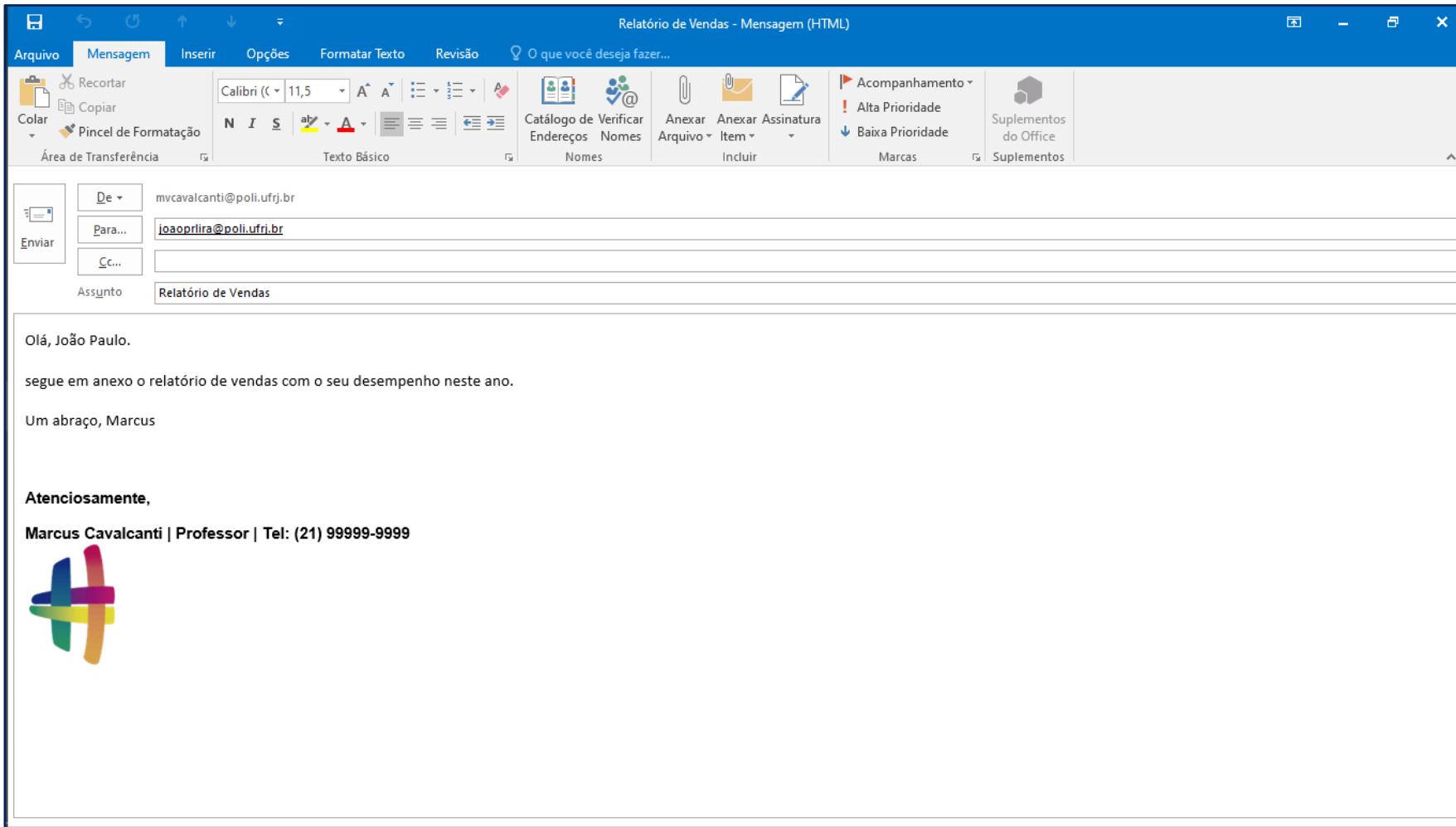
## Módulo 14 – Integração com Outlook – Outros tipos de Formatação de Fonte

564



Para ver mais uma possibilidade, vamos editar a nossa assinatura. Vamos colocá-la em negrito e na fonte Helvetica.

Por fim, clique em Ok.



Ao executar novamente o nosso código, já podemos ver a diferença da nossa nova assinatura.

Seria interessante também deixar o texto do corpo de e-mail com uma formatação similar, para manter um padrão.

```
Sub envia_email()

    Set objeto_outlook = CreateObject("Outlook.Application")
    ult_linha = Range("A1").End(xlDown).Row
    For linha = 2 To ult_linha

        Set Email = objeto_outlook.CreateItem(0)
        Email.Display
        assinatura = Email.htmlbody
        Email.To = Cells(linha, 1).Value
        Email.Subject = "Relatório de Vendas"
        Email.htmlbody = "<p style=""font-size:15px;font-family:helvetica;font-weight:bold"">Olá, " & Cells(linha, 2).Value & ".</p>" &_
                        "<p style=""font-size:15px;font-family:helvetica;font-weight:bold"">" & Cells(linha, 3).Value & "</p>" & _
                        "<p style=""font-size:15px;font-family:helvetica;font-weight:bold"">Um abraço, Marcus</p>" & _
                        assinatura
        Email.Attachments.Add (ThisWorkbook.Path & "\Relatórios Vendas\Vendas - " & Cells(linha, 2).Value & ".xlsx")
        Email.send
    Next
End Sub
```

## Módulo 14 – Integração com Outlook – Outros tipos de Formatação de Fonte

567

Relatório de Vendas - Mensagem (HTML)

Arquivo Mensagem Inserir Opções Formatar Texto Revisão O que você deseja fazer...

Recortar Copiar Colar Pincel de Formulação Área de Transferência

11,5 A A N I S Texto Básico

Catálogo de Verificar Endereços Nomes Nomes Anexar Anexar Assinatura Arquivo Item Incluir Suplementos do Office Marcas Suplementos

De: mvcavalcanti@poli.ufrj.br  
Para: joaopririra@poli.ufrj.br  
Assunto: Relatório de Vendas

Olá, João Paulo.  
segue em anexo o relatório de vendas com o seu desempenho neste ano.  
Um abraço, Marcus

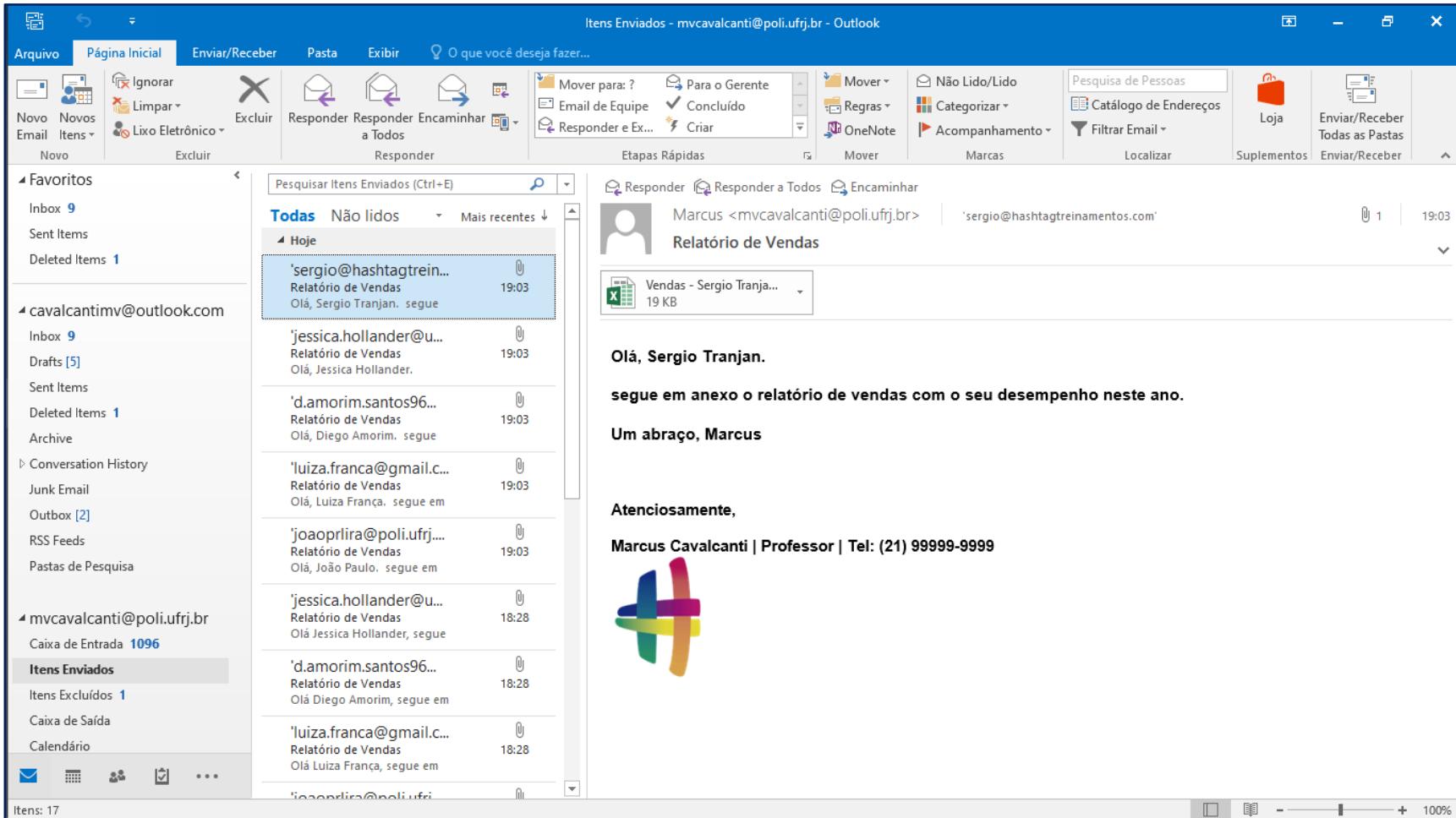
Atenciosamente,  
Marcus Cavalcanti | Professor | Tel: (21) 99999-9999



E o resultado final é mostrado ao lado.

# Módulo 14 – Integração com Outlook – Outros tipos de Formatação de Fonte

568



Finalmente, você pode executar a macro final com todas as alterações.

Fique tranquilo quanto aos códigos em HTML. Não há necessidade de outros códigos além destes, pois se tratam dos principais, uma vez que o mais importante em um e-mail é o seu conteúdo e os anexos.

Com isso, fechamos mais um módulo de integração do VBA. Com estes códigos, você será capaz de ir ainda mais além na automatização de envio de e-mails, que também fazem parte do dia a dia de qualquer um em uma empresa.

Módulo 15

# Integração VBA com PowerPoint

# Módulo 15 – Integração com PowerPoint – Ferramenta 1: Dashboard no PPT

569

Dashboard.xlsxm - Excel

Arquivo Página Inicial Inserir Layout da Página Fórmulas Dados Revisão Exibir Desenvolvedor O que você deseja fazer... Entrar Compartilhar

Fonte Alinhamento Número Estilo Células Edição

B7

2 Desenvolva um botão de seleção que faça uma leitura da base de vendas por ano e retorne aos número de vendas do ano.

	jan	fev	mar	abr	mai	jun	jul	ago	set	out	nov	dez
2014 Vendas	30	23	21	36	34	39	24	30	25	32	29	22
2012 Vendas	33	36	31	30	38	26	27	26	23	36	38	30
2013 Vendas	26	27	21	23	40	21	38	38	22	39	22	31
2014 Vendas	30	23	21	36	34	39	24	30	25	32	29	22
2015 Vendas	23	36	37	38	27	36	35	36	32	28	32	32

jan fev mar abr mai jun jul ago set out nov dez

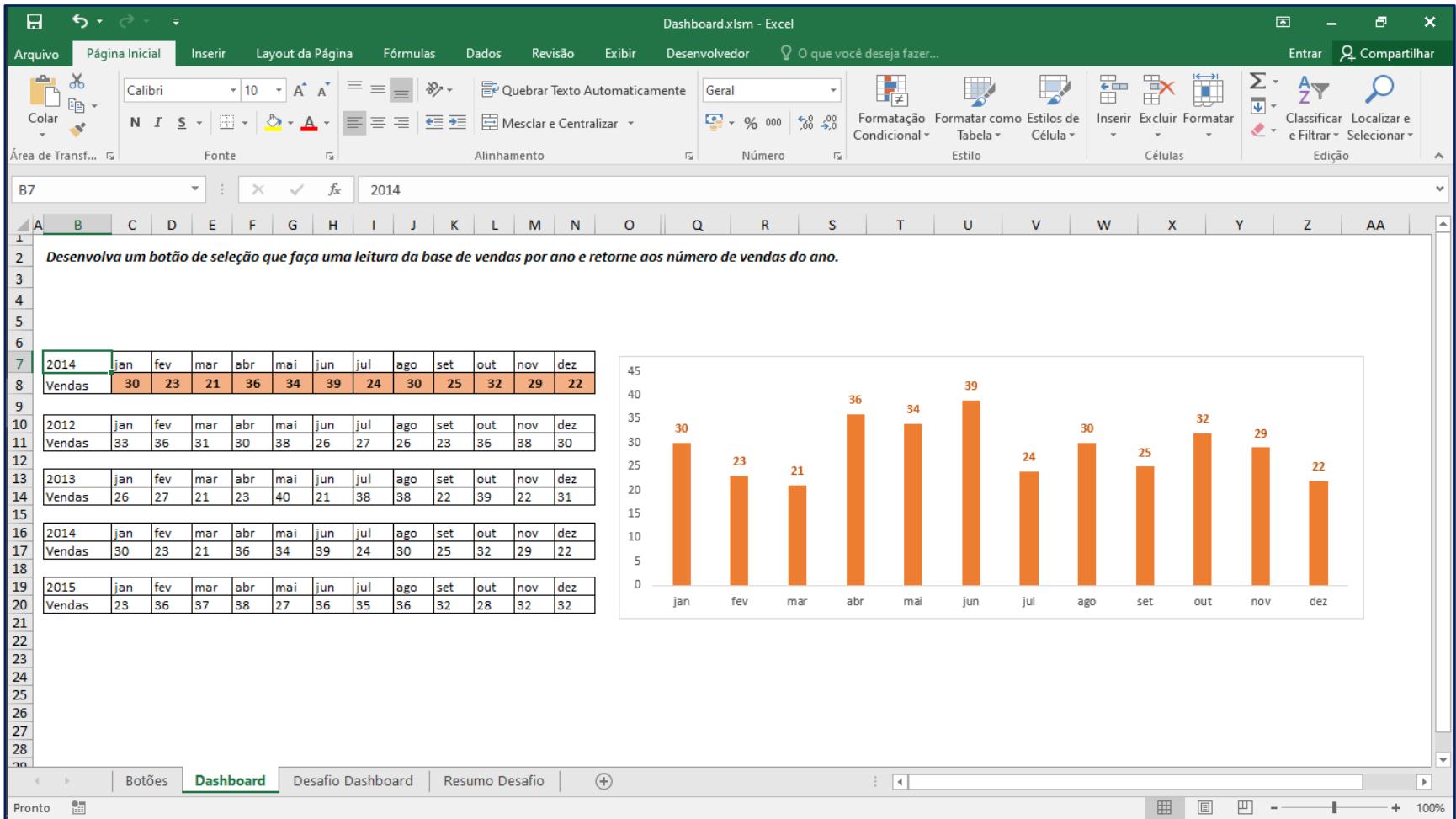
Mês	Vendas
jan	30
fev	23
mar	21
abr	36
mai	34
jun	39
jul	24
ago	30
set	25
out	32
nov	29
dez	22

Chegamos ao nosso último módulo de integração. Conseguimos visualizar o poder do VBA para criação de ferramentas bem avançadas de integração com Word e Outlook. Agora vamos fechar com o PowerPoint.

Ao longo deste módulo criaremos 3 ferramentas de integração, começando do básico, até chegar em uma ferramenta mais completa e avançada.

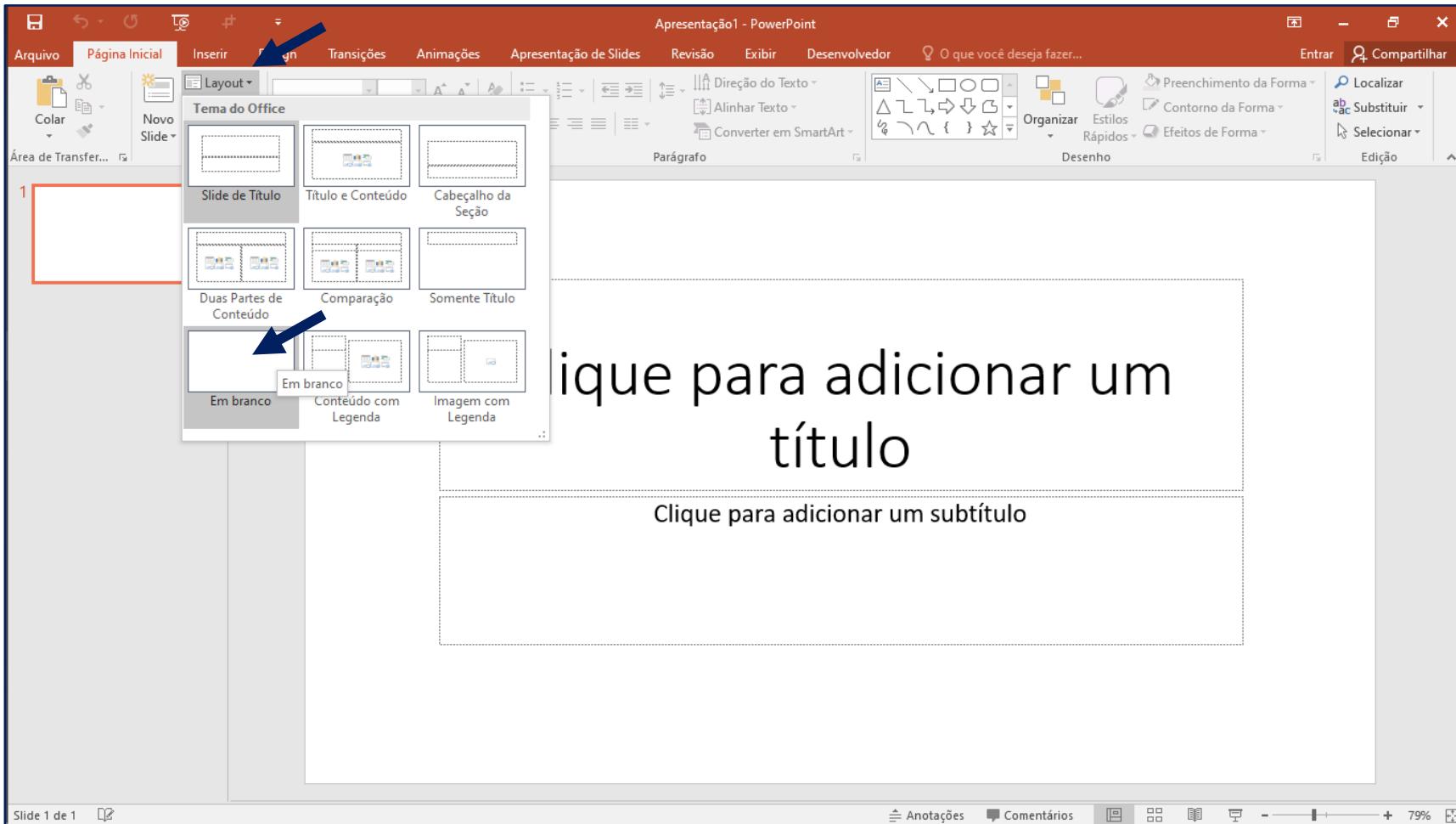
# Módulo 15 – Integração com PowerPoint – Ferramenta 1: Dashboard no PPT

570



No exemplo ao lado temos a nossa primeira ferramenta. Queremos criar um dashboard em uma apresentação de PowerPoint utilizando o gráfico da planilha Excel ao lado.

Você deve reparar que a lógica dentro do Excel já foi criada, então nas células em laranja já temos uma fórmula que faz o valor de vendas mudar de acordo com o ano da célula B7. Portanto, basicamente o que precisamos fazer é possibilitar o código de mudar essa célula B7, que automaticamente mudará o gráfico, fazendo com que a apresentação fique dinâmica.

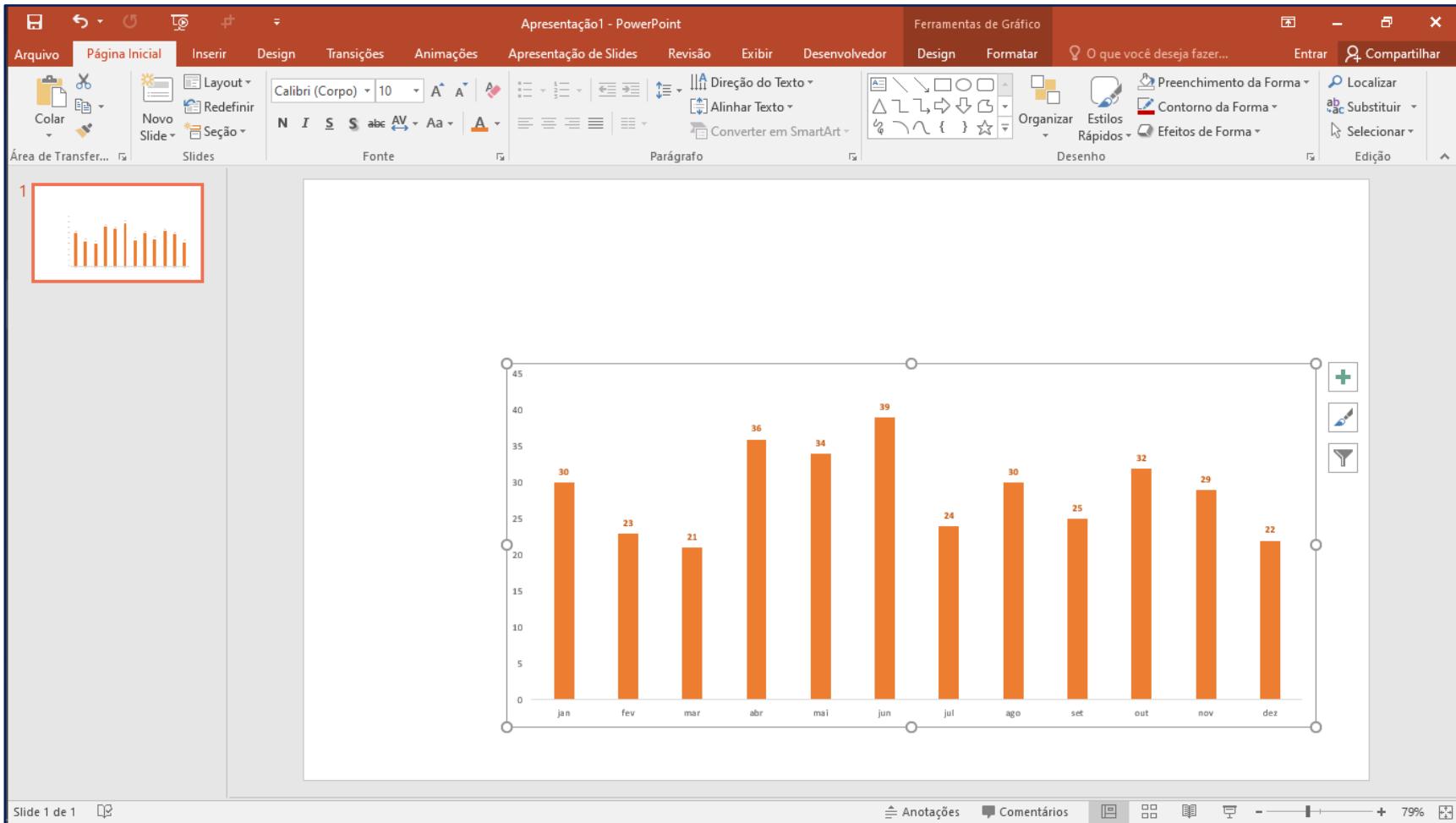


Como ainda não temos uma apresentação, podemos simplesmente abrir uma em branco.

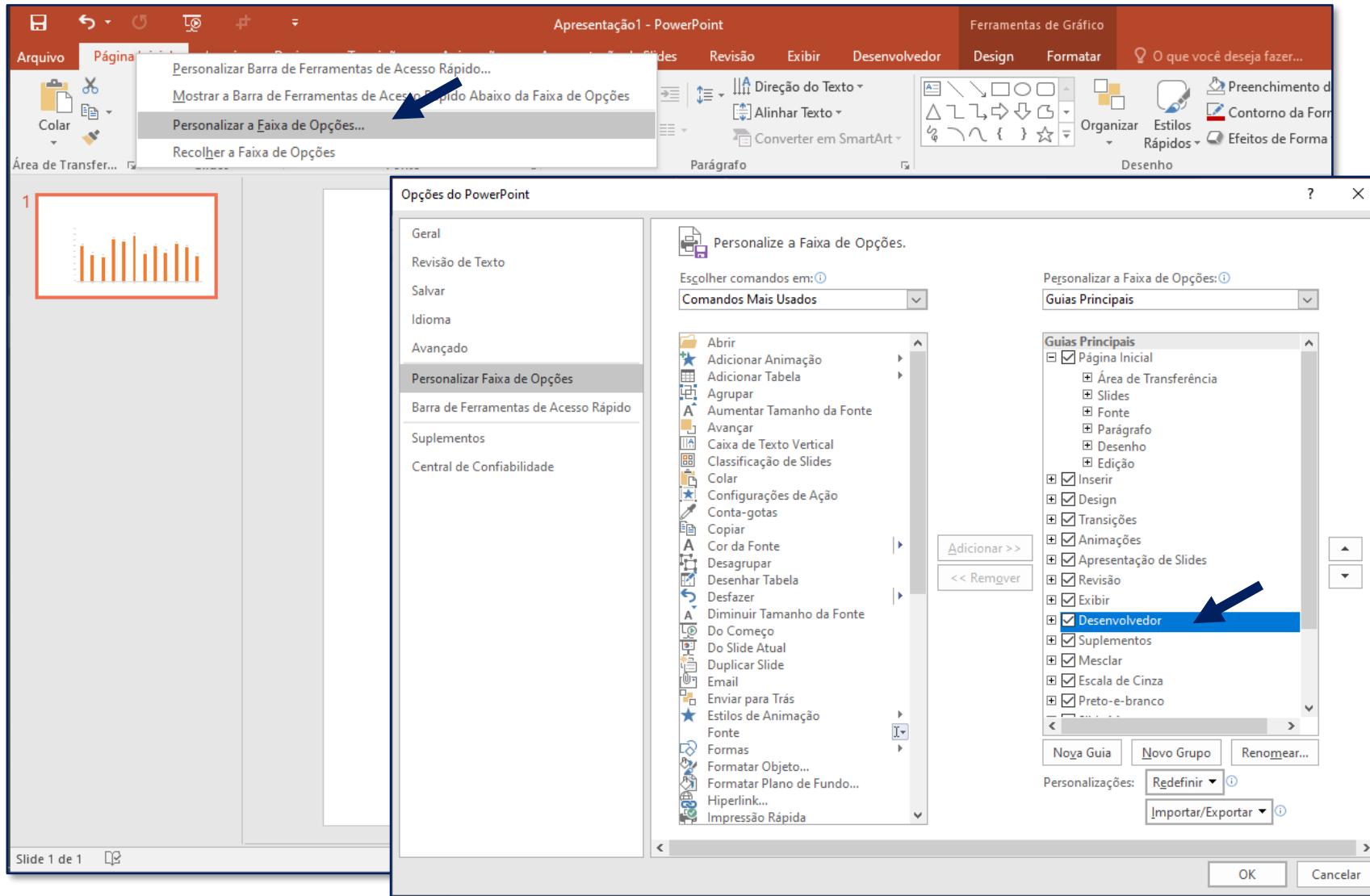
Para trocar o slide para uma folha sem nenhuma caixa de texto, você pode ou deletá-los ou na guia Página Inicial escolher a opção de Layout Em Branco.

# Módulo 15 – Integração com PowerPoint – Criando a apresentação e o botão de controle

572



Em seguida, simplesmente copie o gráfico da planilha Excel e cole neste slide em branco.



Em seguida, você terá que habilitar uma nova guia no seu PowerPoint: a guia **Desenvolvedor**.

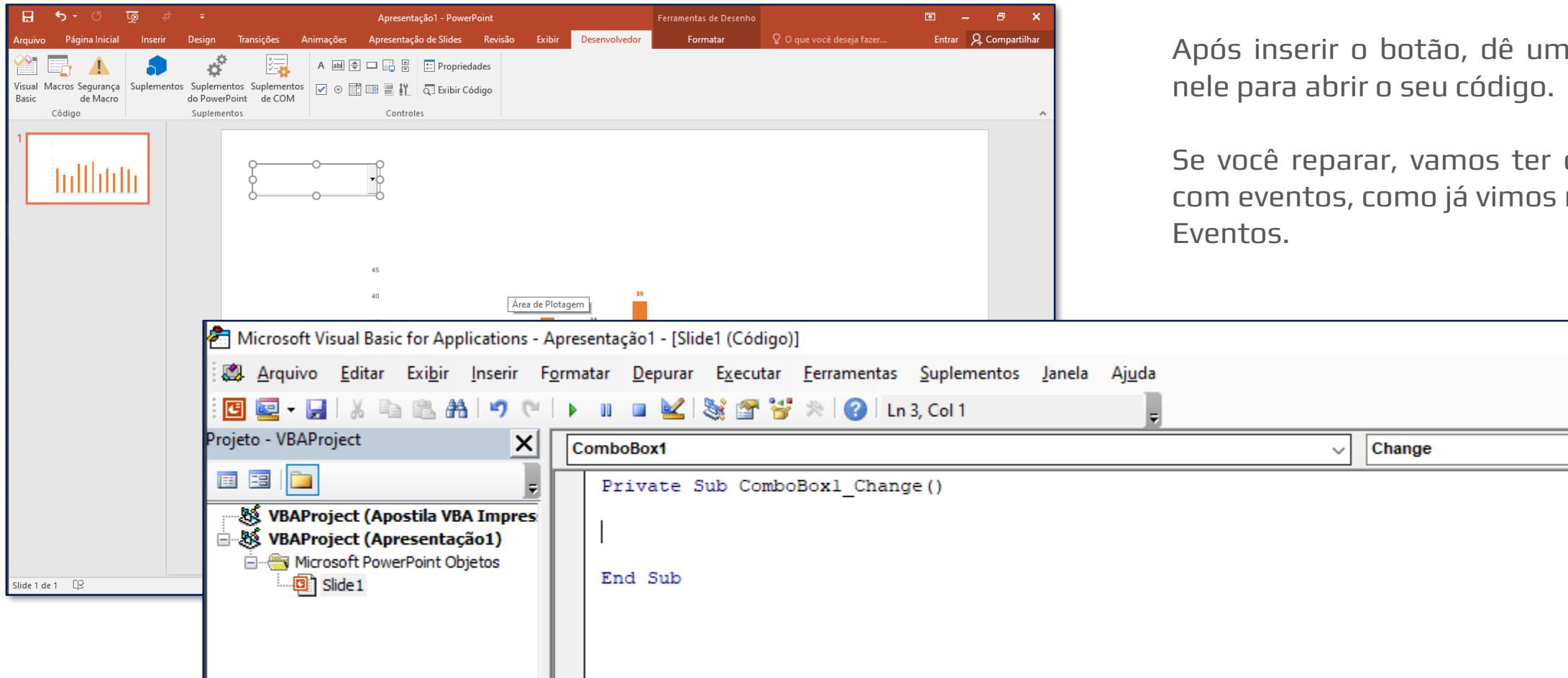
A forma mais fácil de fazer isso é clicar com o botão direito em uma das guias (na Página Inicial, por exemplo) e no menu que abrir escolher a opção de **Personalizar a Faixa de Opções**.

Na janela que abrir, você irá marcar a opção **Desenvolvedor** do lado direito.

The screenshot shows a Microsoft PowerPoint slide titled "Apresentação1 - PowerPoint". The ribbon is visible at the top, with the "Desenvolvedor" (Developer) tab selected. A callout box points to the "Caixa de Combinação (Controle ActiveX)" (Combination Box Control) icon in the "Controles" (Controls) group. On the slide, there is a bar chart with the following data:

Categoria	Venda (Valor)
1	30
2	23
3	31
4	36
5	34
6	39

O próximo passo é inserir a Caixa de Combinação, que é o botão que usaremos para preencher as opções de ano e, de acordo com a opção escolhida, mudar o gráfico com as vendas.



Após inserir o botão, dê um duplo clique nele para abrir o seu código.

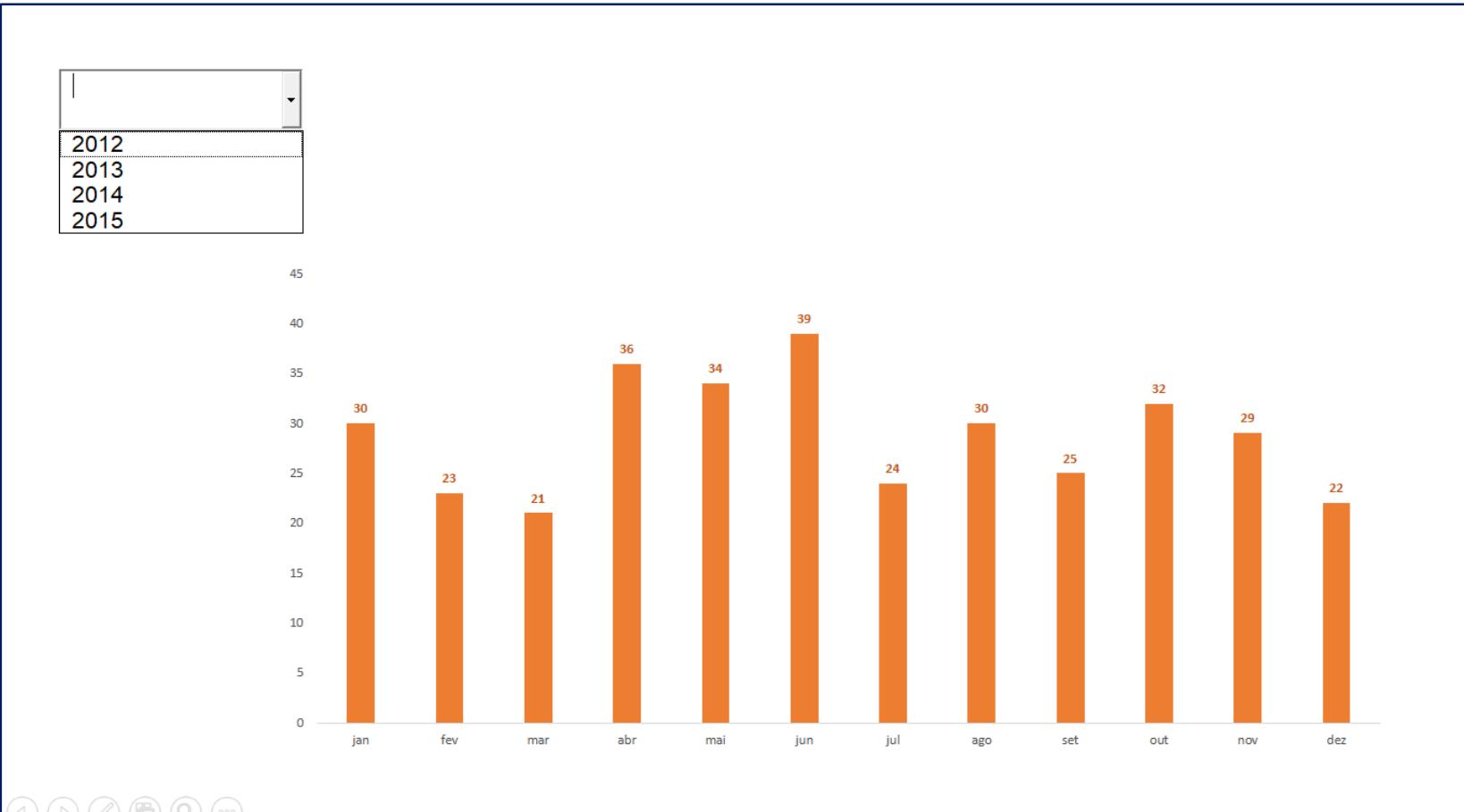
Se você reparar, vamos ter que trabalhar com eventos, como já vimos no módulo de Eventos.

```
Private Sub ComboBox1_DropButtonClick()  
  
If ComboBox1.ListCount = 0 Then  
  
    For ano = 2012 To 2015  
  
        ComboBox1.AddItem (ano)  
  
    Next  
  
End If  
  
End Sub
```

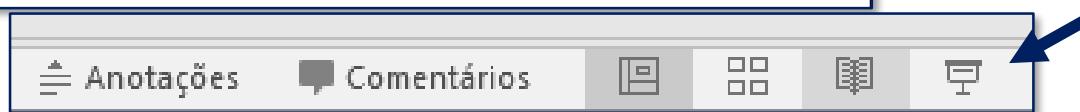
Só temos que entender o evento certo a ser criado. Queremos que a lista de anos apareça sempre que a gente clicar na setinha do botão que abre as opções. Portanto, o evento que teremos que usar é o DropButtonClick.

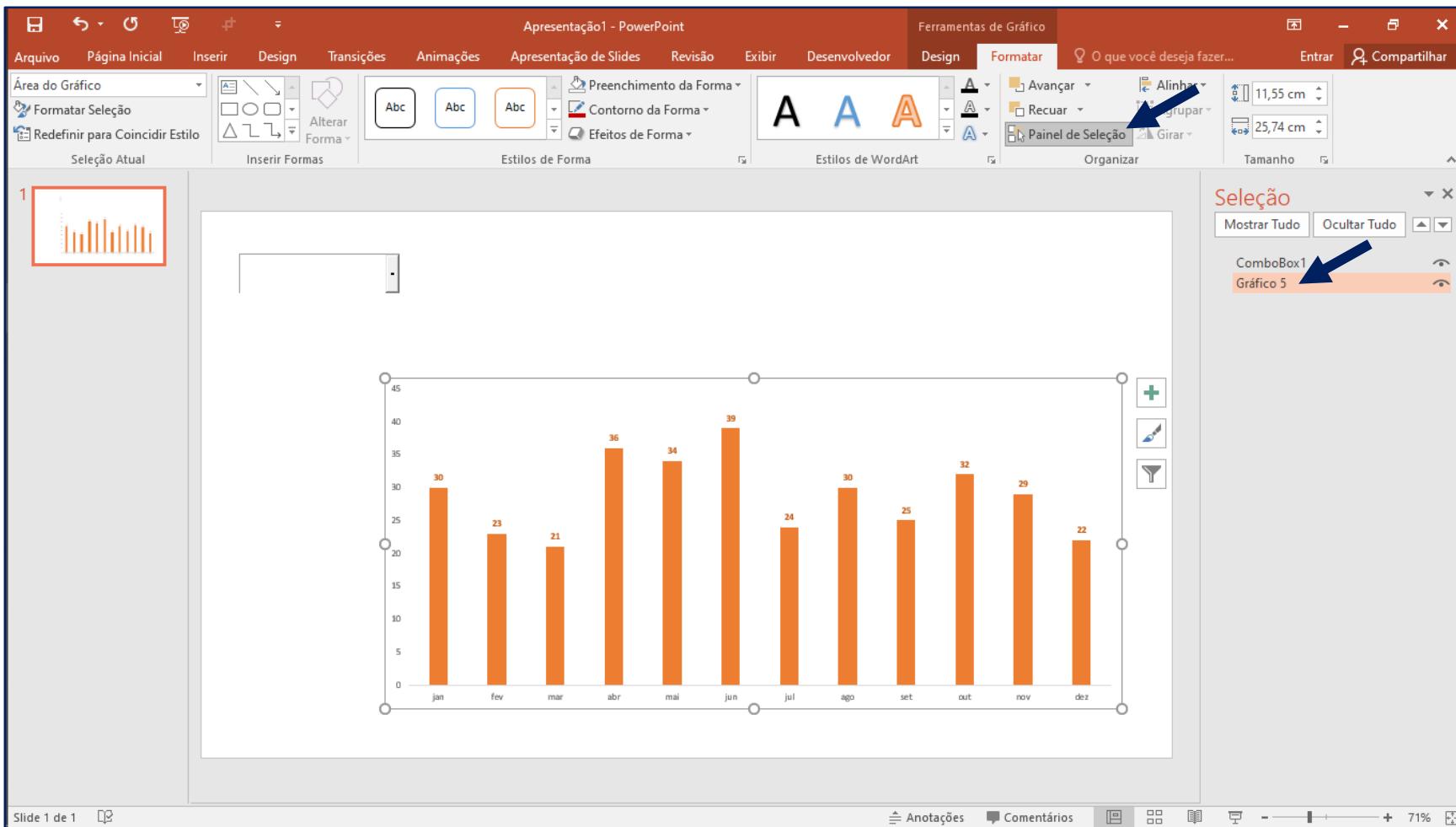
Escolhido o evento, usaremos o código ao lado para preencher as opções do botão. Basicamente temos que testar se existe uma lista preenchida de opções, uma vez que sempre que clicarmos nesse botão ele ficará adicionando listas de forma infinita. A maneira de checar isso é através do If + ListCount. Ou seja, se a contagem de valores na lista for igual a zero, ai sim os anos deverão ser preenchidos através do For. Caso contrário, ele não deverá fazer nada.

Se você lembrar, usávamos a propriedade RowSource para preencher os valores dessas opções. Porém, esta opção não existe no VBA do PowerPoint. **Lembre-se que este código não está sendo criado no VBA do Excel, e sim no VBA do PowerPoint.**



O botão só irá funcionar quando você colocar a apresentação em tela cheia. Para fazer isso, você pode ou clicar no botão indicado abaixo ou usar o atalho F5.





O próximo passo agora é fazer com que o gráfico mude de acordo com o botão.

Para isso, precisamos descobrir qual é o nome que o PowerPoint dá para o gráfico, pois precisaremos desse nome dentro do nosso código.

Você consegue descobrir o nome do gráfico selecionando-o, e na guia Formatar clique na opção de Painel de Seleção.

Repare que na imagem o nome do gráfico é **Gráfico 5**.

ComboBox1

```

Private Sub ComboBox1_Change()
    Me.Shapes("Chart 5").Chart.ChartData.Workbook.Worksheets(1).Range("B7").Value = ComboBox1.Value
End Sub

Private Sub ComboBox1_DropButtonClick()
If ComboBox1.ListCount = 0 Then
    For ano = 2012 To 2014
        ComboBox1.AddItem ano
    Next
End If
End Sub

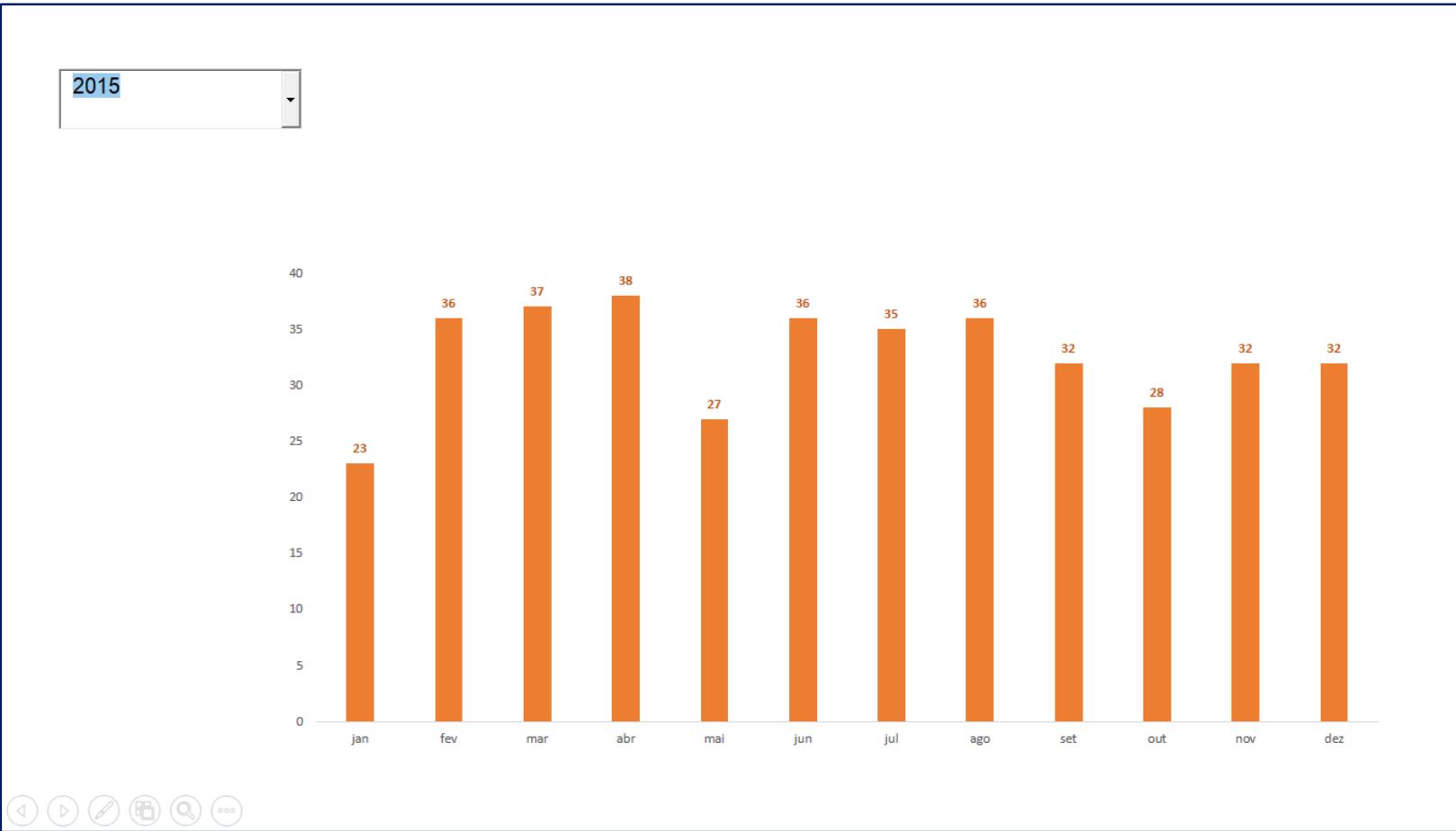
```

*Desenvolva um botão de seleção que faça uma leitura da base de vendas por ano e retorne aos número de vendas.*

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	Q
1																
2																
3																
4																
5																
6																
7	2014	jan	fev	mar	abr	mai	jun	jul	ago	set	out	nov	dez			
8	Vendas	30	23	21	36	34	39	24	30	25	32	29	22			
9																
10	2012	jan	fev	mar	abr	mai	jun	jul	ago	set	out	nov	dez			
11	Vendas	33	36	31	30	38	26	27	26	23	36	38	30			
12																
13	2013	jan	fev	mar	abr	mai	jun	jul	ago	set	out	nov	dez			
14	Vendas	26	27	21	23	40	21	38	38	22	39	22	31			
15																
16	2014	jan	fev	mar	abr	mai	jun	jul	ago	set	out	nov	dez			
17	Vendas	30	23	21	36	34	39	24	30	25	32	29	22			
18																
19	2015	jan	fev	mar	abr	mai	jun	jul	ago	set	out	nov	dez			
20	Vendas	23	36	37	38	27	36	35	36	32	28	32	32			
21																
22																

Para fazer o gráfico mudar de acordo com o botão, teremos que criar um outro evento, que é o **ComboBox1\_Change**, pois queremos que o gráfico se atualize sempre que a gente mudar alguma opção no botão.

O código para fazer isso é mostrado na imagem ao lado. Fique tranquilo que ele é padrão. Basicamente a ideia é que a gente deverá preencher a célula B7 da planilha para que o gráfico mude. E ai se o gráfico muda no Excel, como copiamos ele para dentro do PowerPoint, ele também será atualizado na apresentação.



Agora, para testar a apresentação basta coloca-la em tela cheia e testar o botão. Você verá que o gráfico se atualizar automaticamente.

Isso faz com que a apresentação fique muito mais dinâmica. Com relação à formatação, você pode já criar toda a formatação no gráfico que você desejar diretamente no Excel.

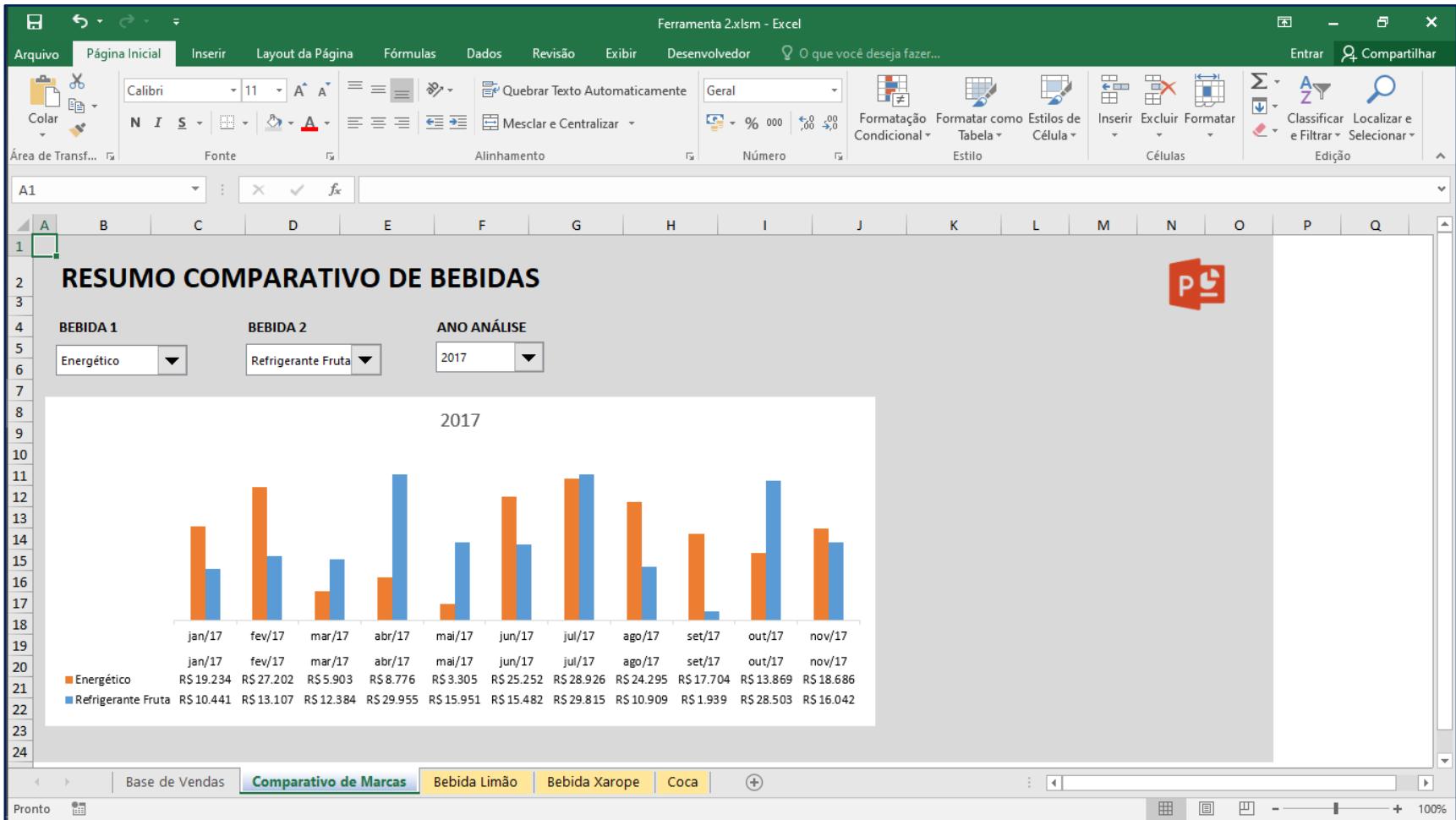
# Módulo 15 – Integração com PowerPoint – Erros Comuns e Atualizando

581

The screenshot shows a Microsoft PowerPoint slide titled "Apresentação1.pptm - PowerPoint". The slide contains a bar chart with data for each month from January to December, showing values ranging from 23 to 38. Above the chart is a small network diagram with the year "2015" at the top. A callout bubble points to the chart area. On the right side of the slide, there is a "Seleção" (Selection) pane showing "ComboBox1" and "Gráfico 5" selected. Below the slide, the VBA editor is open with the following code:

```
ComboBox1_Change()
Me.Shapes("Chart 5").ChartData.Workbook.Worksheets(1).Range("B7").Value = ComboBox1.Value
End Sub
```

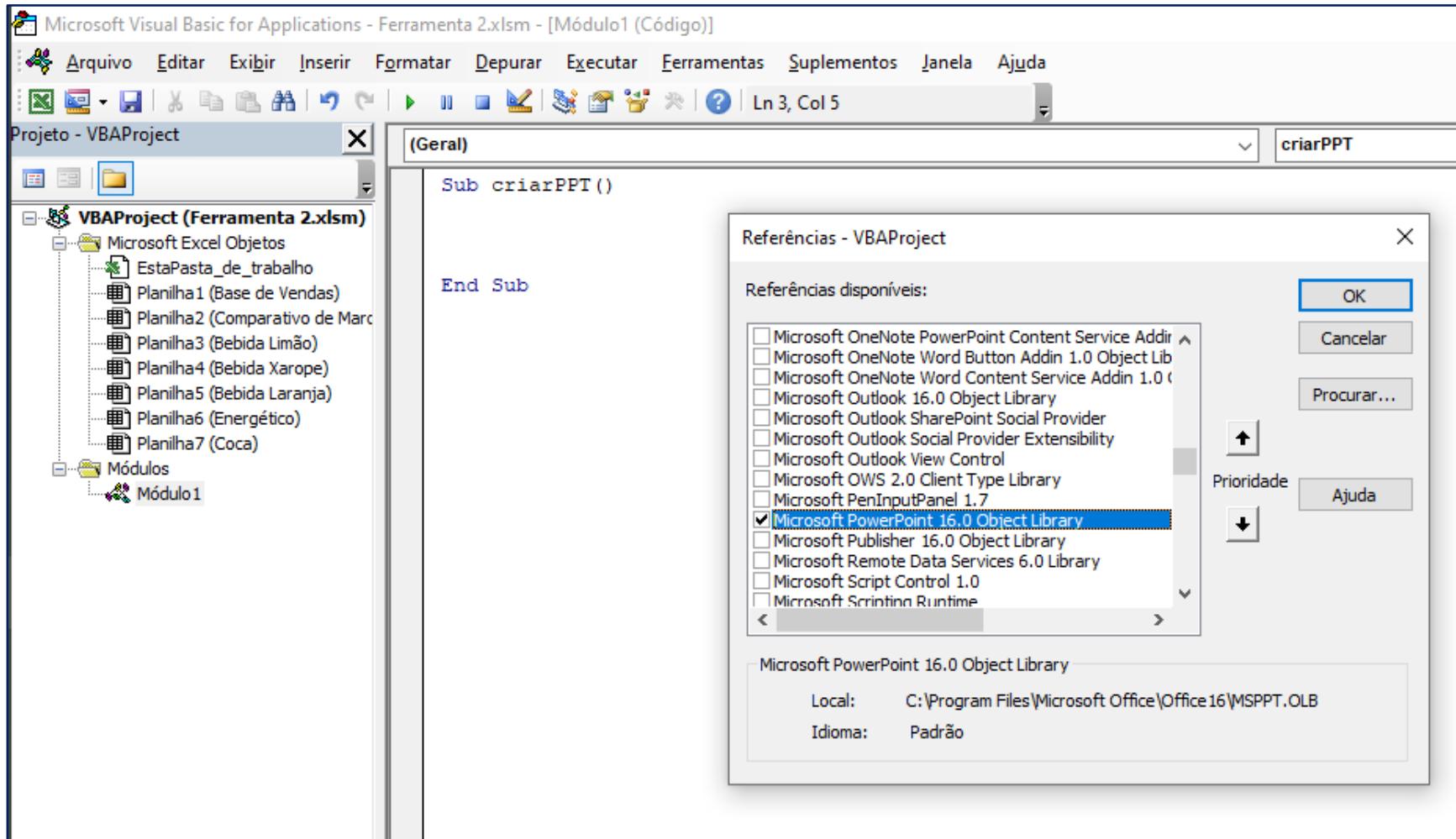
Caso você tenha algum problema futuramente com este código, possivelmente será por conta de mudança de pasta dos arquivos. Então para corrigir o código, você poderá simplesmente copiar o gráfico mais uma vez para o PowerPoint, descobrir o seu novo nome e atualizar no código da macro. Assim você evitará qualquer problema com a sua macro.



Em nossa segunda ferramenta, vamos construir um código que copie cada um dos gráficos das abas do nosso arquivo Excel e cole em uma apresentação de PowerPoint.

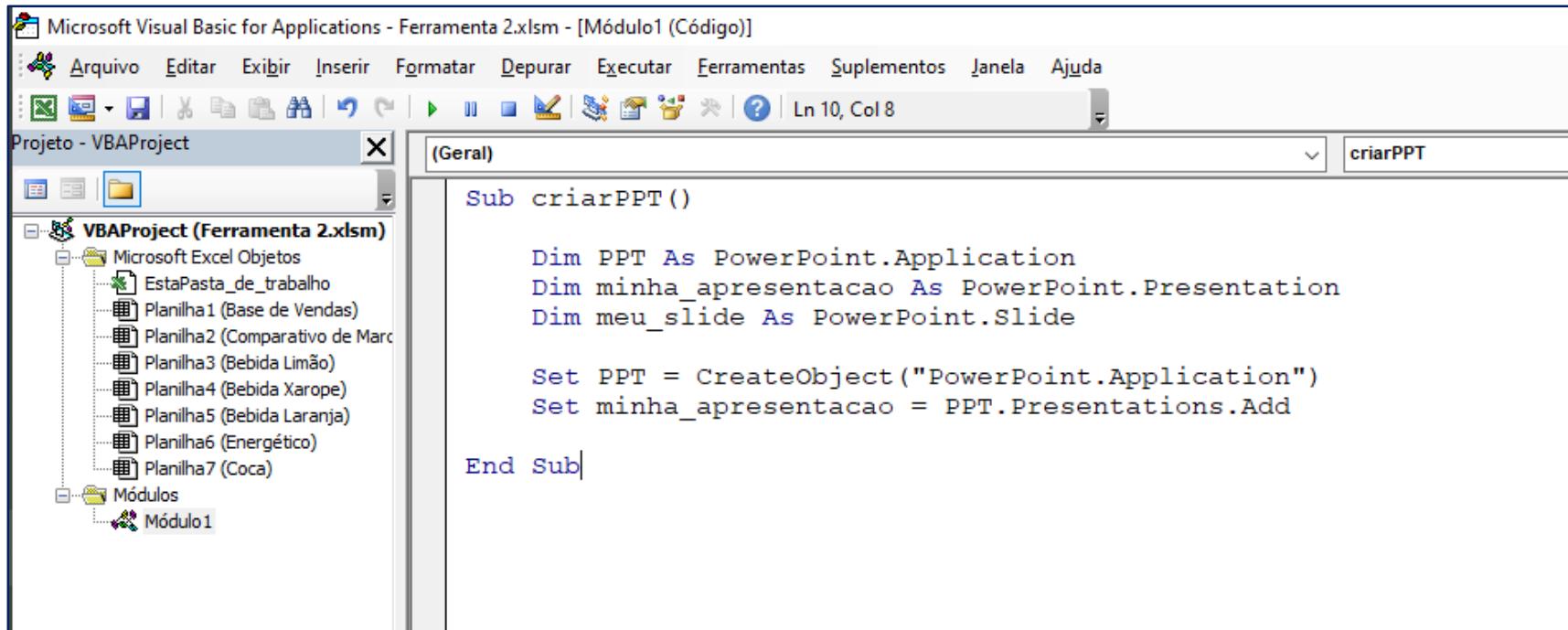
Este é um trabalho comum de se fazer após criar os gráficos no Excel: copiar cada um deles e colar em uma apresentação para o seu cliente ou para o seu gestor.

A ideia é executar este código sempre que clicarmos no botão do PowerPoint no canto superior direito, que simplesmente baixamos do Google e inserimos como Imagem pela guia Inserir do Excel.



Vamos começar criando a nossa Sub **criarPPT**.

Porém, antes de continuar, lembre-se que você deverá habilitar a biblioteca do Microsoft PowerPoint no seu VBA do Excel. Para fazer isso, você vai na guia **Ferramentas** e depois na opção **Referências**.



The screenshot shows the Microsoft Visual Basic for Applications (VBA) editor interface. The title bar reads "Microsoft Visual Basic for Applications - Ferramenta 2.xlsxm - [Módulo1 (Código)]". The menu bar includes Arquivo, Editar, Exibir, Inserir, Formatar, Depurar, Executar, Ferramentas, Suplementos, Janela, and Ajuda. The toolbar has various icons for file operations. The Project Explorer on the left shows "VBAProject (Ferramenta 2.xlsxm)" with a folder "Microsoft Excel Objetos" containing "EstaPasta\_de\_trabalho" and several sheets like "Planilha1" through "Planilha7". A module named "Módulo1" is selected. The code window displays the following VBA code:

```
Sub criarPPT()

    Dim PPT As PowerPoint.Application
    Dim minha_apresentacao As PowerPoint.Presentation
    Dim meu_slide As PowerPoint.Slide

    Set PPT = CreateObject("PowerPoint.Application")
    Set minha_apresentacao = PPT.Presentations.Add

End Sub
```

Começamos seguindo a mesma lógica de sempre: declarar os objetos que precisaremos para o nosso código.

Primeiro declaramos a variável PPT que será o nosso programa do PowerPoint dentro do VBA do Excel e em seguida declaramos um arquivo PowerPoint através da variável **minha\_apresentacao**.

```
Sub criarPPT()

    Dim PPT As PowerPoint.Application
    Dim minha_apresentacao As PowerPoint.Presentation
    Dim meu_slide As PowerPoint.Slide

    On Error Resume Next
    Set PPT = GetObject(, "PowerPoint.Application")

    If PPT Is Nothing Then
        'não tenho PPT criado
        Set PPT = CreateObject("PowerPoint.Application")
        Set minha_apresentacao = PPT.Presentations.Add
    Else
        'já tenho uma apresentação criada
        Set minha_apresentacao = PPT.ActivePresentation
    End If

    'adicionar slide com o gráfico
    'ajustar o gráfico para o tamanho do slide

End Sub
```

A ideia é criar um arquivo PowerPoint do zero para adicionar os nossos gráficos. O detalhe é que todos os gráficos deverão ficar na mesma apresentação. Então, o comando .Add de criar apresentação só deve ser utilizado caso não haja nenhum arquivo PowerPoint aberto, se não ele vai criar vários e vários arquivos.

Para verificar se existe um objeto PowerPoint aberto, usamos o comando GetObject, ou seja, vamos começar tentando “pegar” algum objeto PowerPoint que esteja aberto. Se não existir nenhum, a maneira como saberemos isso é através do teste PPT is Nothing, ou seja, Se PPT “é nada” então pode criar um novo arquivo PowerPoint, se não vamos usar como minha\_apresentacao a apresentação que estiver ativa.

O tratamento de erro On Error é importante para evitar qualquer problema com o comando GetObject caso ele não encontre nenhuma apresentação aberta.

```
Sub criarPPT()

    Dim PPT As PowerPoint.Application
    Dim minha_apresentacao As PowerPoint.Presentation
    Dim meu_slide As PowerPoint.Slide

    On Error Resume Next
    Set PPT = GetObject(, "PowerPoint.Application")

    If PPT Is Nothing Then
        'não tenho PPT criado
        Set PPT = CreateObject("PowerPoint.Application")
        Set minha_apresentacao = PPT.Presentations.Add
    Else
        'já tenho uma apresentação criada
        Set minha_apresentacao = PPT.ActivePresentation
    End If

    'adicionar slide com o gráfico
    'percorrer todos os arquivos da aba
    For Each grafico In ThisWorkbook.ActiveSheet.ChartObjects

        Set meu_slide = minha_apresentacao.Slides.Add(1, ppLayoutBlank)
        grafico.Chart.ChartArea.Copy
        meu_slide.Shapes.Paste
    Next

    'ajustar o gráfico para o tamanho do slide
    |
```

End Sub

Em seguida, vamos percorrer cada um dos gráficos que tivermos na nossa lista de gráficos (ChartObjects) dentro de uma aba.

Para cada gráfico, vamos:

1. Adicionar um novo slide à nossa apresentação e atribuí-lo à variável **meu\_slide**.
2. Copiar o gráfico que estiver selecionado naquele momento do loop For Each.
3. Colar o gráfico em **meu\_slide**.

```
Sub criarPPT()

    Dim PPT As PowerPoint.Application
    Dim minha_apresentacao As PowerPoint.Presentation
    Dim meu_slide As PowerPoint.Slide

    On Error Resume Next
    Set PPT = GetObject(, "PowerPoint.Application")

    If PPT Is Nothing Then
        'não tenho PPT criado
        Set PPT = CreateObject("PowerPoint.Application")
        Set minha_apresentacao = PPT.Presentations.Add
    Else
        'já tenho uma apresentação criada
        Set minha_apresentacao = PPT.ActivePresentation
    End If

    'adicionar slide com o gráfico
    'percorrer todos os arquivos da aba
    For Each grafico In ThisWorkbook.ActiveSheet.ChartObjects

        Set meu_slide = minha_apresentacao.Slides.Add(1, ppLayoutBlank)
        grafico.Chart.ChartArea.Copy
        meu_slide.Shapes.Paste
    Next

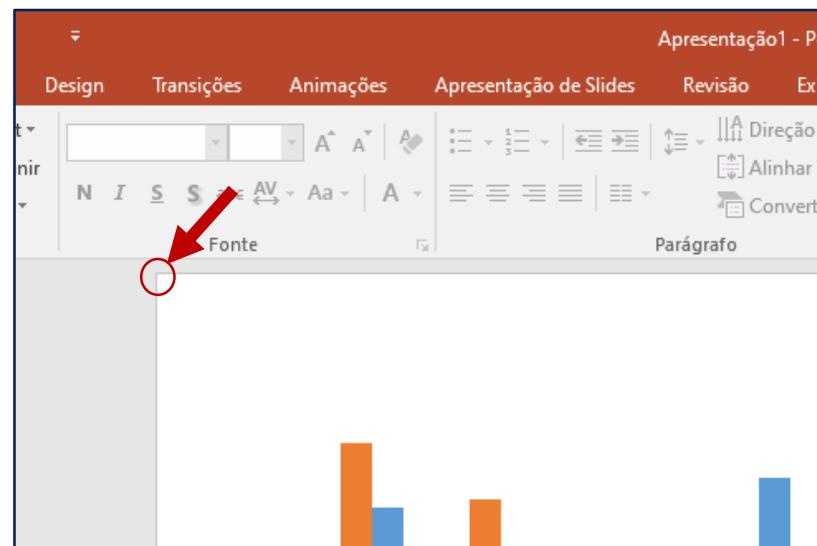
    'ajeitar o gráfico para o tamanho do slide
    altura = minha_apresentacao.PageSetup.SlideHeight
    largura = minha_apresentacao.PageSetup.SlideWidth

    meu_slide.Shapes(1).Height = altura
    meu_slide.Shapes(1).Width = largura

    meu_slide.Shapes(1).Left = 0
    meu_slide.Shapes(1).Top = 0

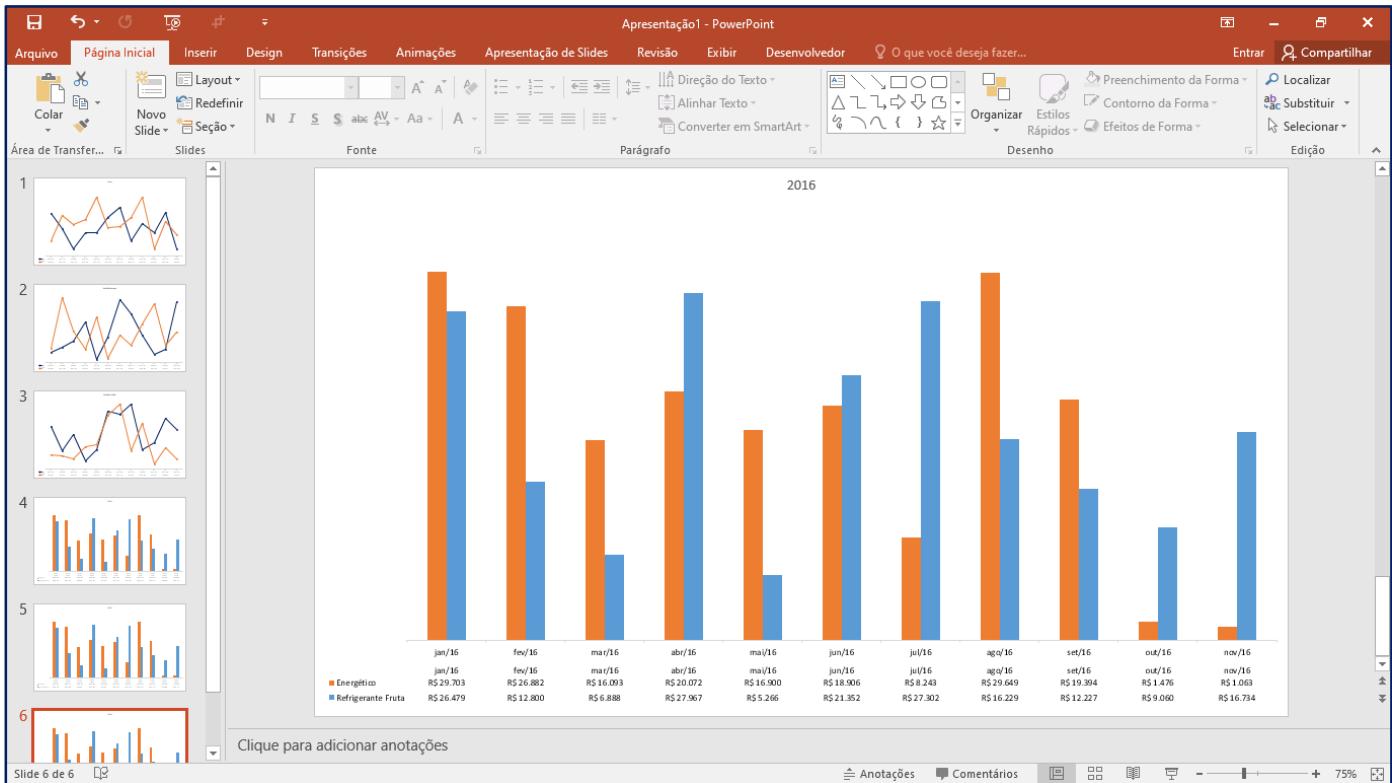
End Sub
```

Para fazer com que o gráfico tenha exatamente o mesmo tamanho do slide, configuramos as suas propriedades, como mostrado no código ao lado. Para garantir que ele sempre começará a ser posicionado bem no início do slide, é como se tivéssemos que configurar o seu início para o ponto (0, 0) do gráfico, ou seja, largura = 0 e altura = 0.



# Módulo 15 – Integração com PowerPoint – Redimensionar e ajeitar criação de gráfico

588



Cada aba do nosso arquivo Excel possui um gráfico diferente. Após executar a macro separadamente em cada aba, o gráfico daquela aba será colado na apresentação do PowerPoint.

The screenshot shows a Microsoft Excel spreadsheet titled "Ferramenta 3 - Integração VBA com PowerPoint.xls - Excel". The data is organized into columns: Ano, Vendas Totais, Valor Vendido, Lucro Total, Margem, and Status. A blue button labeled "Gerar Relatórios" is positioned over the data area. The status column contains values like "30% Não OK", "28% Não OK", etc.

	A	B	C	D	E	F	G	H	I	J	K
1	Ano	Vendas Totais	Valor Vendido	Lucro Total	Margem	Status					
2	2020	2134	R\$ 3.200.000,00	R\$ 960.000,00	30%	Não OK					
3	2019	2227	R\$ 3.340.028,92	R\$ 935.208,10	28%	Não OK					
4	2018	2276	R\$ 3.412.741,24	R\$ 682.548,25	20%	Não OK					
5	2017	2296	R\$ 3.443.626,33	R\$ 1.205.269,22	35%	Não OK					
6	2016	2146	R\$ 3.217.875,46	R\$ 804.468,86	25%	Não OK					
7	2015	2198	R\$ 3.295.621,72	R\$ 988.686,52	30%	Não OK					
8	2014	2049	R\$ 3.072.512,73	R\$ 860.303,56	28%	Não OK					
9	2013	2015	R\$ 3.022.322,63	R\$ 604.464,53	20%	Não OK					
10	2012	1939	R\$ 2.907.079,81	R\$ 1.017.477,93	35%	Não OK					
11	2011	1834	R\$ 2.750.316,97	R\$ 687.579,24	25%	Não OK					
12											
13											
14											
15											
16											
	Alunos										

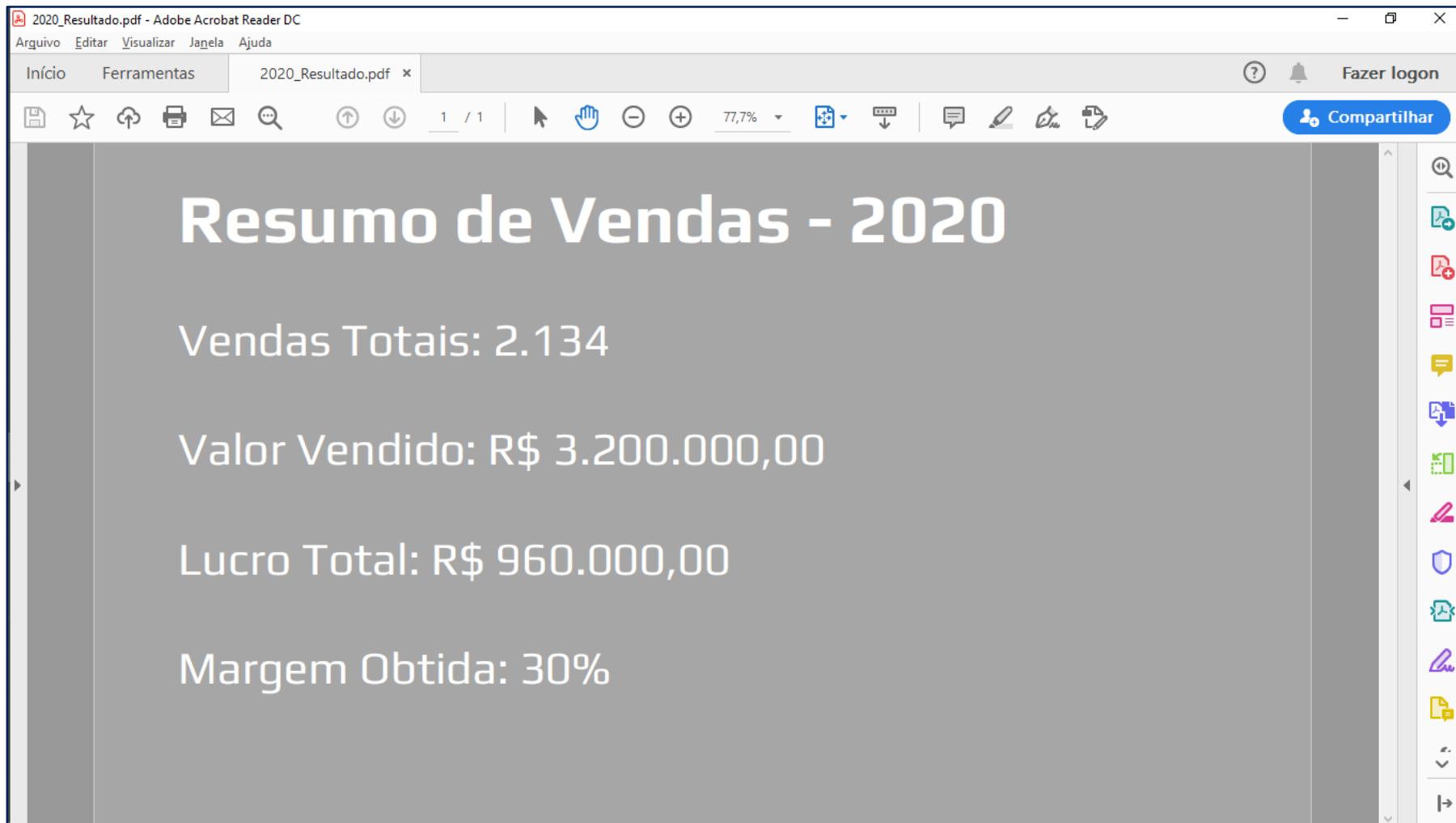
Nosso terceiro e último exercício será uma macro que irá gerar um relatório para cada um dos anos de análise.

Cada ano possui uma série de indicadores que deverão ser copiados, colados em uma apresentação em PowerPoint e salvos como PDF.

The screenshot shows a Microsoft PowerPoint slide titled "Resumo de Vendas - AAAA". The slide contains four text elements: "Vendas Totais: XXXX", "Valor Vendido: YYYY", "Lucro Total: LLLL", and "Margem Obtida: MMMM". The "Fonte" tab is selected in the ribbon. The slide is part of a presentation named "Modelo.pptx".

Na pasta do exercício já temos um modelo de apresentação.

A ideia é que, para cada indicador, temos um campo onde inseri-lo dentro do gráfico. Aqui vamos usar a mesma lógica dos exercícios de integração com o Word, que foi de criar palavras-chave para conseguirmos facilmente substitui-las pelos valores da nossa planilha.



A imagem ao lado mostra um exemplo de como deverá ficar o nosso relatório final, já salvo como PowerPoint.

```
Sub CriarRelatorio()

    'percorrer todas as linhas
    ult_linha = Range("A1").End(xlDown).Row
    For linha = 2 To ult_linha
        ano = Cells(linha, 1).Value
        vendas = Cells(linha, 2).Value
        valor = Cells(linha, 3).Value
        lucro = Cells(linha, 4).Value
        margem = Cells(linha, 5).Value

        situacao = GerarRelatorio(ano, vendas, valor, lucro, margem)
        Cells(linha, 6).Value = situacao

    Next

    'para cada linha registrar relatório
    'registrar status dos relatórios

End Sub

Function GerarRelatorio(ano As Double, vendas As Double, valor As Double, lucro As Double, margem As Double) As String

    GerarRelatorio = "Não OK"
    GerarRelatorio = "OK"

End Function
```

O esqueleto do nosso código é mostrado ao lado. Primeiro criamos uma Sub para Criar o Relatório. Nesta Sub, devemos percorrer um Loop desde a linha 2 até a última linha da tabela, pois queremos executar o código de criação das apresentações repetidas vezes, uma para cada ano.

Para fins de organização, criamos uma Function chamada GerarRelatorio para executar toda a lógica de abrir o PPT, substituir os valores e salvar como PDF. Esta Function será chamada na própria Sub e foi feita apenas para fins de organização.

```
Sub CriarRelatorio()

    Dim ano As Double, vendas As Double, valor As Double, lucro As Double, margem As Double

    'percorrer todas as linhas
    ult_linha = Range("A1").End(xlDown).Row
    For linha = 2 To ult_linha
        ano = Cells(linha, 1).Value
        vendas = Cells(linha, 2).Value
        valor = Cells(linha, 3).Value
        lucro = Cells(linha, 4).Value
        margem = Cells(linha, 5).Value

        situacao = GerarRelatorio(ano, vendas, valor, lucro, margem)
        Cells(linha, 6).Value = situacao

    Next

    'para cada linha registrar relatório
    'registrar status dos relatórios

End Sub

Function GerarRelatorio(ano As Double, vendas As Double, valor As Double, lucro As Double, margem As Double) As String

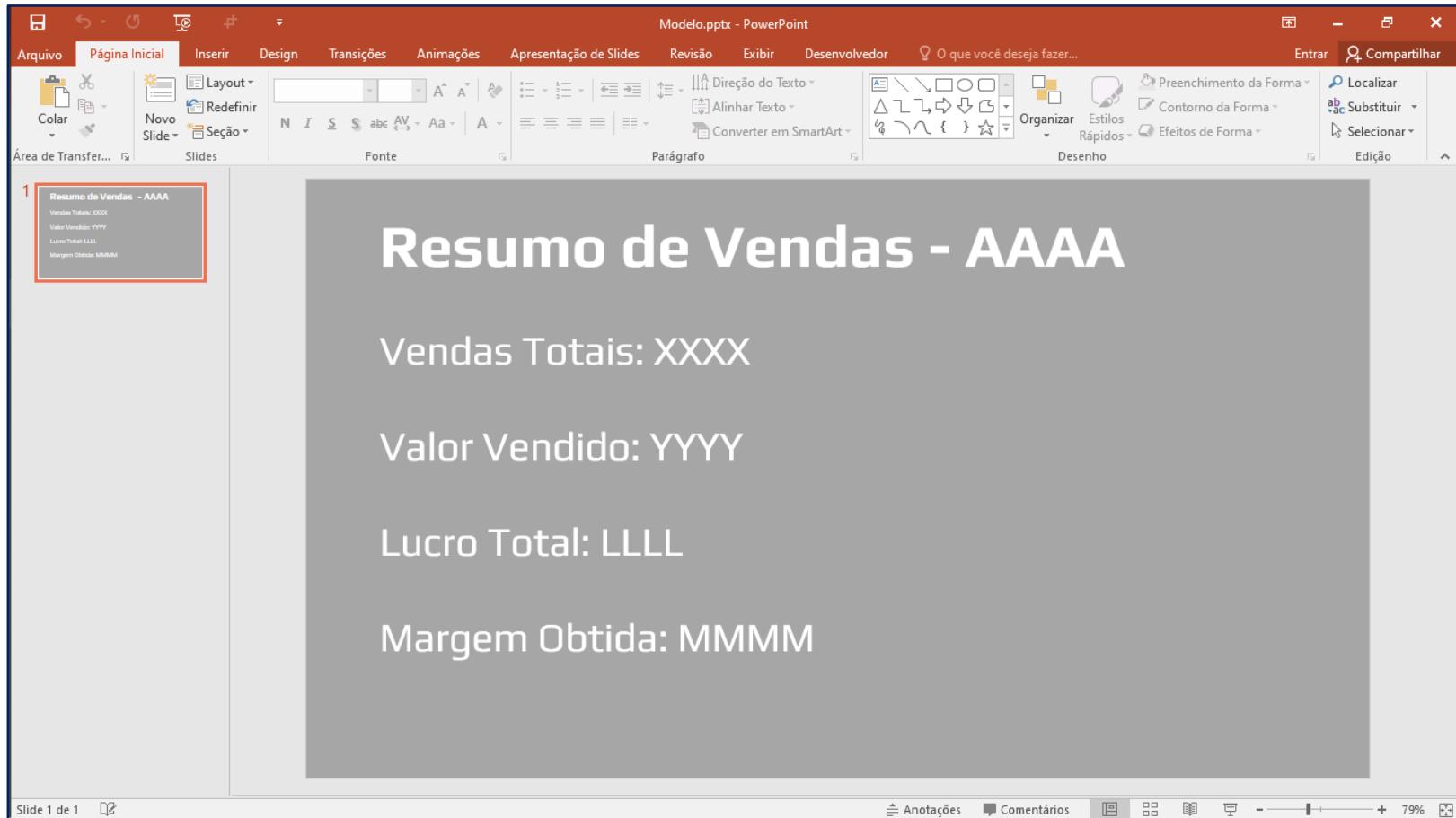
    GerarRelatorio = "Não OK"
    GerarRelatorio = "OK"

    Dim PPT As PowerPoint.Application
    Dim apresentacao As PowerPoint.Presentation

    Set PPT = New PowerPoint.Application
    Set apresentacao = PPT.Presentations.Open(ThisWorkbook.Path & "\Modelo.pptx")

End Function
```

O início do nosso código na Function segue a mesma estrutura de sempre: declarar o programa PowerPoint e também declarar o arquivo em PowerPoint que iremos usar. Como já temos um modelo padrão pronto, apenas vamos abrí-lo através do comando .Open.



Modelo.pptx - PowerPoint

Arquivo Página Inicial Inserir Design Transições Animações Apresentação de Slides Revisão Exibir Desenvolvedor O que você deseja fazer... Entrar Compartilhar

Colar Novo Slide Seção

Área de Transfer... Slides

Fonte Parágrafo Desenho Edição

1 Resumo de Vendas - AAAA

Vendas Totais: XXXX

Valor Vendido: YYYY

Lucro Total: LLLL

Margem Obtida: MMMM

Slide 1 de 1 Anotações Comentários 79%

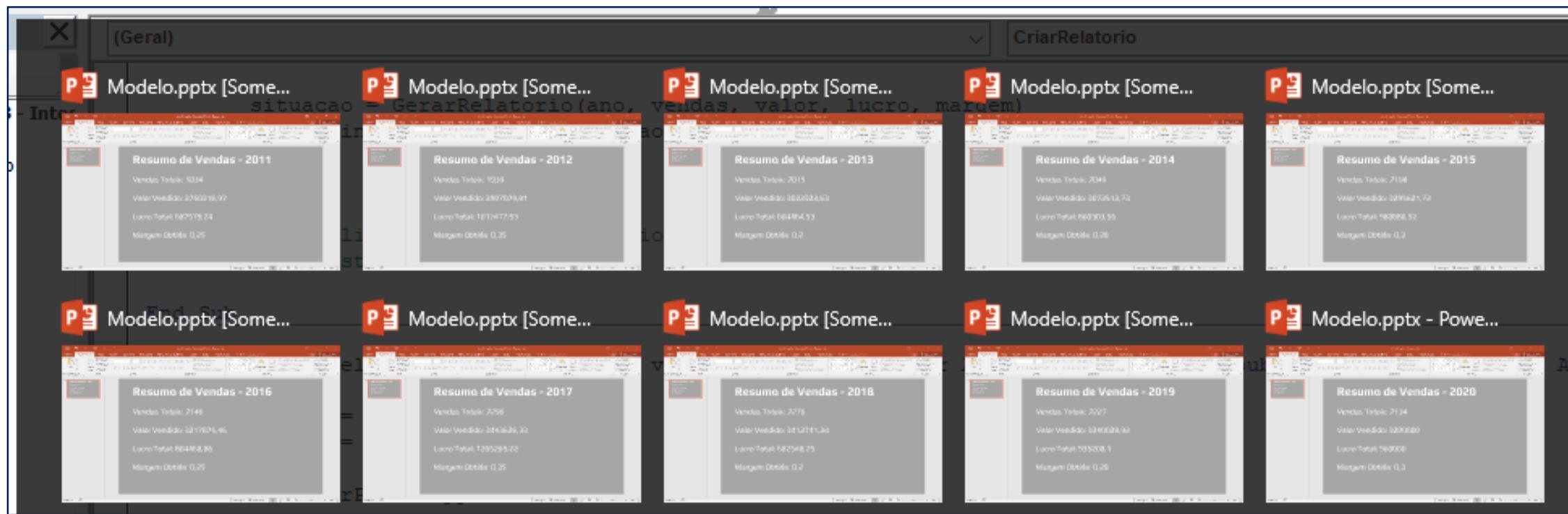
Lembrando que cada valor possui a sua palavra-chave correspondente que deverá ser localizada e substituída.

```
Function GerarRelatorio(ano As Double, vendas As Double, valor As Double, lucro As Double, margem As Double) As String  
  
GerarRelatorio = "Não OK"  
GerarRelatorio = "OK"  
  
Dim PPT As PowerPoint.Application  
Dim apresentacao As PowerPoint.Presentation  
  
Set PPT = New PowerPoint.Application  
Set apresentacao = PPT.Presentations.Open(ThisWorkbook.Path & "\Modelo.pptx")  
  
'trocar os códigos pelas variáveis  
For Each objeto In apresentacao.Slides(1).Shapes  
  
    If objeto.HasTextFrame Then  
        |  
        Set teste = objeto.TextFrame.TextRange.Replace("AAAA", ano)  
        Set teste = objeto.TextFrame.TextRange.Replace("XXXX", vendas)  
        Set teste = objeto.TextFrame.TextRange.Replace("YYYY", valor)  
        Set teste = objeto.TextFrame.TextRange.Replace("LLLL", lucro)  
        Set teste = objeto.TextFrame.TextRange.Replace("MMMM", margem)  
    End If  
  
Next  
  
End Function
```

Para isso, usamos o comando de substituir mostrado abaixo (Replace).

O loop For Each se dá por conta da necessidade de percorrer cada uma das caixas de texto do nosso PowerPoint, pois temos mais de uma. Além disso, para não correr o risco de ainda assim selecionar um objeto que não tenha nenhum texto, fazemos um teste if para verificar se o objeto em questão possui algum texto (HasTextFrame) e, caso tenha, aí sim executa os códigos de substituição.

Se você executar este código, já verá um resultado. Todos os slides são gerados, separadamente para cada ano. O código já está praticamente pronto, o que falta agora é salvar estas apresentações como PDF na pasta de Relatórios e fechar os arquivos.



```
Function GerarRelatorio(ano As Double, vendas As Double, valor As Double, lucro As Double, margem As Double) As String
GerarRelatorio = "Não OK"

Dim PPT As PowerPoint.Application
Dim apresentacao As PowerPoint.Presentation

Set PPT = New PowerPoint.Application
Set apresentacao = PPT.Presentations.Open(ThisWorkbook.Path & "\Modelo.pptx")

'trocar os códigos pelas variáveis
For Each objeto In apresentacao.Slides(1).Shapes

    If objeto.HasTextFrame Then

        Set teste = objeto.TextFrame.TextRange.Replace("AAAA", ano)
        Set teste = objeto.TextFrame.TextRange.Replace("XXXX", Format(vendas, "#,##0"))
        Set teste = objeto.TextFrame.TextRange.Replace("YYYY", Format(valor, "Currency"))
        Set teste = objeto.TextFrame.TextRange.Replace("LLLL", Format(lucro, "Currency"))
        Set teste = objeto.TextFrame.TextRange.Replace("MMMM", Format(margem, "0%"))

    End If

Next

apresentacao.SaveAs ThisWorkbook.Path & "\Relatórios\" & ano & "_Resultado", ppSaveAsPDF

GerarRelatorio = "OK"

End Function
```

Para salvar como PDF, usamos o código marcado ao lado. Este também é outro código padrão, então uma boa memória será necessária aqui para códigos futuros. Ou então você pode simplesmente consultar esta solução sempre que precisar.

Nome	Data de modificação	Tipo	Tamanho
2011_Resultado.pdf	17/03/2020 08:51	Adobe Acrobat D...	46 KB
2012_Resultado.pdf	17/03/2020 08:51	Adobe Acrobat D...	46 KB
2013_Resultado.pdf	17/03/2020 08:51	Adobe Acrobat D...	46 KB
2014_Resultado.pdf	17/03/2020 08:51	Adobe Acrobat D...	46 KB
2015_Resultado.pdf	17/03/2020 08:51	Adobe Acrobat D...	46 KB
2016_Resultado.pdf	17/03/2020 08:51	Adobe Acrobat D...	46 KB
2017_Resultado.pdf	17/03/2020 08:51	Adobe Acrobat D...	46 KB
2018_Resultado.pdf	17/03/2020 08:51	Adobe Acrobat D...	46 KB
2019_Resultado.pdf	17/03/2020 08:51	Adobe Acrobat D...	46 KB
2020_Resultado.pdf	17/03/2020 08:51	Adobe Acrobat D...	46 KB

Agora, após executar o código, teremos todos os arquivos salvos em PDF na nossa pasta de Relatórios.

```
Sub CriarRelatorio()

Dim ano As Double, vendas As Double, valor As Double, lucro As Double, margem As Double

'percorrer todas as linhas
ult_linha = Range("A1").End(xlDown).Row
For linha = 2 To ult_linha
    ano = Cells(linha, 1).Value
    vendas = Cells(linha, 2).Value
    valor = Cells(linha, 3).Value
    lucro = Cells(linha, 4).Value
    margem = Cells(linha, 5).Value

    situacao = GerarRelatorio(ano, vendas, valor, lucro, margem)
    Cells(linha, 6).Value = situacao

Next

Shell "TASKKILL /F /IM powerpnt.exe", vbHide

End Sub
```

Para encerrar, falta agora um código para encerrar todos os arquivos PowerPoint de uma vez, forçadamente e sem salvar, pois não queremos modificar o nosso modelo para não comprometer relatórios futuros.

O comando usado para isso é mostrado ao lado.

# Módulo 15 – Integração com PowerPoint – Fechar PPT forçadamente e Finalizando

600

The screenshot shows a Microsoft Excel spreadsheet titled "Ferramenta 3 - Integração VBA com PowerPoint.xlsxm - Excel". The spreadsheet contains a table with columns: Ano, Vendas Totais, Valor Vendido, Lucro Total, Margem, and Status. The data for 2020 is: Ano 2020, Vendas Totais 2134, Valor Vendido R\$ 3.200.000,00, Lucro Total R\$ 960.000,00, Margem 30%, Status OK. Below the table, there is a chart showing a blue bar from 0 to 1000, with a value of 2134 marked. In the background, an Adobe Acrobat Reader DC window is open, showing a PDF document titled "Resumo de Vendas - 2020". The PDF content matches the Excel data, displaying: Resumo de Vendas - 2020, Vendas Totais: 2.134, Valor Vendido: R\$ 3.200.000,00, Lucro Total: R\$ 960.000,00, and Margem Obtida: 30%.

Ano	Vendas Totais	Valor Vendido	Lucro Total	Margem	Status
2020	2134	R\$ 3.200.000,00	R\$ 960.000,00	30%	OK

E finalmente encerramos o nosso código. Você já pode executá-lo e ele funcionará perfeitamente.

E com isto também se encerra o nosso curso de VBA. Percorremos um longo caminho, construindo diferentes ferramentas, desde o básico até o mais avançado, por meio de integração entre os diferentes programas da Microsoft. O que imagino é que você tenha ficado com um pouco de receio com relação a alguns códigos usados, especialmente nestes 3 últimos módulos: “é muita coisa, vou ter que decorar tudo isso?”

De fato são comandos novos e que não conseguimos obtê-los de forma fácil, por meio de gravação de macro, por exemplo. Em determinado ponto isso de fato iria acontecer, pois o VBA é uma linguagem muito rica e com infinitas possibilidades. A melhor sugestão que tenho a dar para você é que também utilize o Google como fonte de ajuda. Isto porque o VBA (e a maioria das outras linguagens de programação) já chegaram em um nível de utilização que praticamente 99% dos problemas já foram enfrentados por outras pessoas, mas soluções também foram propostas por outras pessoas. Então praticamente qualquer dúvida que você tenha é muito provável que exista uma exatamente igual na internet, porém com uma solução.

# Considerações Finais

602

A screenshot of a web browser displaying the Stack Overflow homepage. The search bar at the top contains the text '[vba]'. The main content area shows a list of questions tagged 'vba'. The first question is titled 'Como fazer a execução esperar um comando de tecla? [fechada]' and has -2 votes. The second question is titled 'Como eu escrevo essa fórmula em .formula?' and has -1 vote. A sidebar on the right is titled 'Em destaque no Meta' and lists several meta posts. Below that is a section titled 'Tags relacionadas' with links to 'excel' and 'excel-vba'.

Uma ótima sugestão para buscar uma solução para o seu problema é o site **StackOverFlow**. Este é um site de perguntas muito ativo e com milhões de publicações.

Lá você encontra uma infinidade de perguntas e respostas, de pessoas que já passaram pelo mesmo problema que você.

The screenshot shows a Google search results page with the following details:

- Search Query:** how to loop through checkboxes in vba stackoverflow
- Results Count:** Aproximadamente 60.300 resultados (0,51 segundos)
- Top Result:** [How to loop through CheckBoxes on UserForm? - Stack Overflow](#)
  - 1 resposta
  - 15 de mai. de 2017 - By selecting an OptionButton I want all **Checkboxes** on the active form to deselect AND UNCHECK. How to fix this? Alternatively: Is it possible to have a change event on the UserForm that states that if a **CheckBox** is Enabled= False Then Value=False.
- Second Result:** [Excel VBA: How to loop through checkboxes based on name ...](#)
  - 1 resposta
  - 17 de ago. de 2016
  - Loop Through CheckBox Controls in VBA UserForm      1 de ago. de 2013
  - Loop through checkboxes in VBA      23 de jan. de 2019
  - Loop through checkboxes on a sheet      27 de jul. de 2016
- Bottom Result:** [Excel VBA: How to loop through checkboxes based on name ...](#)
  - 1 resposta
  - 17 de ago. de 2016 - As a follow up answer to my comment, here is what I think you are looking for: arow = lastAns - qRow Dim i As Long, ctl As Control For i = 1 To 4 ...

Ao lado, um exemplo de como você pode pesquisar a sua dúvida no Google. Sim, você não está vendo errado, a busca foi em inglês mesmo.

Isso porque buscar pelas dúvidas em inglês garante que você tenha um leque muito maior de respostas, pois existem muito mais fóruns em inglês do que em português, isso é fato. É claro que isso não quer dizer que você não poderá pesquisar suas dúvidas em português, mas pesquisando em inglês você terá mais opções e, em geral, respostas bem mais completas. Então também é importante que você se preocupe um pouco com o inglês também, nem que seja utilizando o Translate do Google para traduzir os textos. Mas fique tranquilo pois esta sugestão é para ser utilizada em situações em que o seu nível de VBA já é bem elevado, então com bastante prática uma hora isso será bem fluido e natural.

De qualquer forma, com bastante prática tenho certeza que todos os conceitos ficarão cada vez mais claros para você. Então não deixe de praticar, não só estes últimos 3 últimos módulos, bem como todos os anteriores. Quanto mais contato você tiver com os códigos, mais facilidade você terá. Lembrando que qualquer dúvida você tem a ajuda do nosso suporte com qualquer questão.

Por fim, desejamos que você tenha muito sucesso nesta sua caminhada, e que o curso te ajude a se tornar um profissional ainda melhor do que o que você já é!

Até mais!

Equipe Hashtag.

