**Centro paula Souza**

**Faculdade de tecnologia de mogi-mirim**

**Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas**

Danilo Cesar Tobinaga

Geremias Sodré Novais

Matheus

Projeto de aplicativo em Excel para controle de movimentação de produtos

**Mogi-Mirim – SP**

**2017**

**Danilo Cesar Tobinaga**

**Geremias Sodré Novais**

**Matheus**

**Projeto de aplicativo em Excel para controle de movimentação de produtos**

**Trabalho computacional apresentado à disciplina Programação em Microinformática – componente da matriz curricular do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Faculdade de tecnologia de Mogi-Mirim – como instrumento de avaliação para a obtenção de aprovação na Disciplina de Programação em Microinformática.**

Orientador: Prof. Me. Marcos Roberto de Moraes

**MOGI-MIRIM**

**2017**

Danilo Cesar Tobinaga

Geremias Sodré Novais

Matheus

**Projeto de aplicativo em Excel para controle de movimentação de produtos**

Este trabalho computacional, submetido como instrumento de avaliação à disciplina Programação em Microinformática – componente curricular do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Faculdade de tecnologia de Mogi-Mirim – foi avaliado e aprovado para a obtenção de aprovação na disciplina Programação em Microinformática.

Mogi-Mirim, de de 2017.

**Prof. Luíz Henrique Biazotto**

**Coordenador do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas**

**BANCA EXAMINADORA**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Me. Marcos Roberto de Moraes

Resumo

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Palavras-chave: 5 – separadas por ;

Abstract

Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Keywords: 5 – separadas por ;

**Sumário**

[1 introdução 6](#_Toc485603517)

[2 Desenvolvimento 7](#_Toc485603518)

[2.1 referencial teórico 7](#_Toc485603519)

[2.1.1 Excel 7](#_Toc485603520)

[2.1.2 vba 7](#_Toc485603521)

[2.1.3 objetos específicos do excel 7](#_Toc485603522)

[2.1.3.1 objeto application 7](#_Toc485603523)

[2.1.3.2 objeto workbook 8](#_Toc485603524)

[2.1.3.3 objeto worksheet 10](#_Toc485603525)

[2.1.3.4 objeto range 10](#_Toc485603526)

[2.1.3.4.1 representação de um objeto range 11](#_Toc485603527)

[2.1.4 UserForms 12](#_Toc485603528)

[2.1.4.1 Controles de seleção 13](#_Toc485603529)

[2.1.4.2 controles de interfaceamento 14](#_Toc485603530)

[2.1.4.3 controles de opção 14](#_Toc485603531)

[2.2 Metodologia 16](#_Toc485603532)

[2.2.1 A aplicação 16](#_Toc485603533)

[2.2.2 A construção da aplicação 19](#_Toc485603534)

[2.2.3 a programação 21](#_Toc485603535)

[2.3 resultados 22](#_Toc485603536)

# introdução

A ideia deste trabalho é desenvolver um aplicativo baseado nas ferramentas do Software Micrososft Office Excel para automatizar um processo de controle de entrada e saída de produtos, de modo a tornar este processo mais rápido e simples de realizar.

Esta aplicação será construída através dos recursos avançados do Excel e as ferramentas de programação do Visual Basic.

A aplicação consiste em três planilhas e um formulário, sendo uma planilha para registrar os movimentos de entrada e saída no estoque de produtos, duas que formarão uma base de dados para cadastro de produtos e fornecedores, e contabilização de itens do estoque de produtos, respectivamente, e um formulário, que será a interface customizada contendo todo o controle do processo e que será responsável por interagir com o usuário para que o processo seja realizado.

# Desenvolvimento

## referencial teórico

### Excel

O Excel é uma ferramenta incrivelmente poderosa para tornar significativa uma vasta quantidade de dados. Mas ele também funciona muito bem para cálculos simples e para rastrear de quase todos os tipos de informações. A chave para desbloquear todo esse potencial é a grade de células. As células podem conter números, texto ou fórmulas e podem ser agrupadas em linhas e colunas. Isso permite adicionar dados, classificá-los e filtrá-los, inseri-los em tabelas e criar gráficos incríveis (Tarefas básicas no Excel, 2017).

### vba

VBA é uma plataforma de desenvolvimento para aplicativos Microsoft Office. Essa plataforma usa uma linguagem de programação muito parecida com Visual Basic, porem com algumas particularidades herdadas da própria plataforma. Com VBA é possível fazer quase tudo dentro dos aplicativos do Office, desde criar contratos automaticamente no Word, criar apresentações no Power Point em questões de segundos, fazer eficazes sistemas de cadastro no Excel entre outras coisas (Estudo sobre VBA: Introdução, 2017).

### objetos específicos do excel

São eles: **Application**, **WorkBooks**, **WorkSheets** e **Range** (FERNANDES, 2005).

#### objeto application

O objeto **Application** representa o próprio Excel e seus objetos derivados. Através dele, podemos configurar o Excel em termos de visualização, execuções e outras funcionalidades.

O objeto **Application** possui várias propriedades. A seguir alguns exemplos (FERNANDES, 2005):

**Propriedade DisplayFormulaBar**

Exibe ou oculta a barra de fórmulas do Excel.

Public Sub BarraFormula()

Application.DisplayFormulaBar = False

MsgBox “A barra de fórmulas foi desabilitada”

Application.DisplayFormulaBar = True

MsgBox “A barra de fórmulas foi habilitada”

End Sub

**Propriedade DisplayFullScreen**

Exibe o Excel em tela cheia, diferente de maximizar a aplicação.

Public Sub TelaCheia()

Application.DisplayFullScreen = True

MsgBox “A tela está no modo FullScreen”

Application.DisplayFullScreen = False

MsgBox “A tela voltou ao modo normal”

End Sub

#### objeto workbook

Um objeto **Workbook** representa uma pasta de trabalho aberta. Cada objeto **Workbook** é armazenado em uma coleção **Workbooks**, que faz parte das coleções do objeto **Application**, visto anteriormente (FERNANDES, 2005).

Como um objeto **Workbook** representa uma pasta de trabalho, podemos acessá-lo como um índice da coleção **Workbooks** da seguinte maneira: **Workbooks(1)** ou pelo nome do arquivo **Workbooks(“Nome do arquivo”)**.

O código à seguir ajuda a perceber a diferença entre o objeto **Application** e o objeto **Workbook**:

Public Sub Dif()

MsgBox Application.Name

MsgBox Workbooks(1).Name

End Sub

O nome retornado pelo objeto **Application** é do Microsoft Excel, que é a aplicação que está rodando, enquanto o nome retornado pelo objeto **Workbook** é, caso ainda não tenha sido salva a aplicação, **Pasta1**.

A seguir, alguns exemplos do objeto **Workbook**:

**Propriedade FullName**

Retorna o **Path** e o **Name** do objeto. Se uma pasta chamada **Teste.xls** fosse salva em **C:\Meus Documentos**,

Workbooks(1).FullName

Retornaria **“C\Meus Documentos\Teste.xls”**

**Método Activate**

Este método torna uma pasta de trabalho ativa. Exemplo:

Workbooks(1).Activate

#### objeto worksheet

O objeto **Worksheet** representa uma planilha especificada na coleção **Worksheets**.

Existem várias propriedades e métodos que são aplicados aos objetos **Worksheets**.

Alguns exemplos (FERNANDES, 2005):

**Propriedade Visible**

Controla a visibilidade de uma planilha. É possível utilizar três valores:

* **xlSheetHidden** – Torna a planilha oculta.
* **xlSheetVisible** – Torna a planilha visível.
* **xlSheetVeryHidden** – Só permite tornar uma planilha visível por código, ou seja, o usuário não poderá solicitar a exibição da planilha específica.

**Método Activate**

O método **Activate** torna uma planilha da coleção **Worksheets** ativa. Sintaxe:

Worksheets(1).Activate

No exemplo acima, a primeira **Worksheet** da lista de planilhas será ativada.

#### objeto range

O objeto **Range** é quem faz o trabalho pesado no Excel. Esse objeto representa uma célula, uma coluna, uma linha, um conjunto de células, até todas as células de uma planilha; por isso, é tão genérico e funcional (FERNANDES, 2005).

##### representação de um objeto range

Um objeto **Range** pode ser representado de diversas formas. Pode ser (FERNANDES, 2005):

* Uma célula:

Range (“A1”)

* Um intervalo de células:

Range (“B1:D4”)

* Uma intersecção de intervalos:

Range (“B1:D4 C2:E6”)

* Uma união de intervalos:

Range (“B1:D4”, “C2:E6”)

* Um objeto Range também pode ser representado pelas células limites que o compõem, na seguinte sintaxe:

Range (cell1, cell2)

Por exemplo, **Range (“A1:B2”)** é equivalente a **Range (Cells(1,1), Cells(1,2))**.

* Finalmente, também pode ser representado por um conjunto de linhas ou colunas:

Range (Columns(1), Columns(3))

A seguir, algumas propriedades (FERNANDES, 2005):

**Propriedade Cells**

A propriedade **Cells** é uma representante de todas as células de dentro do Range. Através dela, pode-se obter informações específicas a respeito de cada célula ou do conjunto como um todo.

**Propriedade Value**

Retorna ou define um determinado valor de uma célula do **Range**.

Exemplo:

Private Sub Worksheet\_SelectionChange(ByVal Target As Range)

Dim Celula As Range, i As Integer

For Each Celula In Target

i = i + 1

Celula.Value = i

Next

End Sub

Aqui, cada célula selecionada será preenchida com o seu index (posição) no **Range**.

### UserForms

UserForms são formulários personalizados pelo programador para fazer um interfaceamento entre a pasta de trabalho Excel e o usuário da aplicação, visando facilidade e controle.

Através de um UserForm, aliado aos objetos e recursos de programação do ambiente VBA, é possível fazer com que o usuário da aplicação trabalhe apenas em um ambiente personalizado, sem precisar digitar nas planilhas (FERNANDES, 2005).

No ambiente de programação VBA, é possível inserir um UserForm no menu **Inserir/UserForm**.

A seguir, um UserForm vazio (Figura 1):

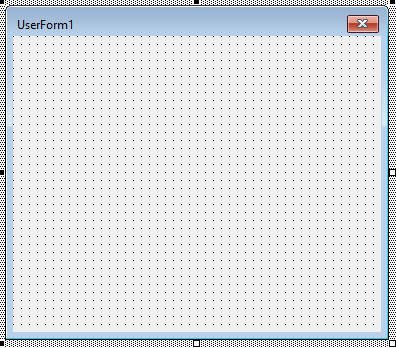


Figura - UserForm padrão (Próprio Autor)

Junto ao formulário, tem a caixa de ferramentas. São os recursos gráficos do ambiente VBA para formulários, dispostos segundo a Figura 2 (Próprio Autor):

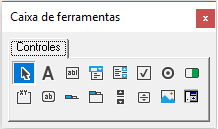


Figura - Caixa de Ferramentas (Proprio Autor)

Abaixo, alguns controles serão explicados.

#### Controles de seleção

São as ferramentas contidas na caixa de ferramentas, usadas para implementar controles no formulário. A seguir uma breve descrição de cada uma delas (FERNANDES, 2005):

**Selecionar objetos**

Essa ferramenta serve para dimensionar e selecionar controles.

#### controles de interfaceamento

**Rótulo**

O rótulo serve para colocarmos textos dentro de um formulário (Figura 3).



Figura - Rótulo (Próprio Autor)

**Caixas de Texto (TextBox)**

A caixa de texto permite a digitação de qualquer caractere alfanumérico (Figura 4).



Figura -Caixa de texto (Próprio Autor)

**Botão de comando (CommandButton)**

Os botões de comando são os principais responsáveis por executar as funções finais de uma aplicação, ou seja, àquelas que determinam alguma consequência para a aplicação (Figura 5).



Figura - Botão de comando (Próprio Autor)

#### controles de opção

**Caixa de seleção (Checkbox)**

A caixa de seleção é um controle booleano, ou seja, permite escolher entre **Sim** e **Não** para um determinado item (Figura 6).



Figura - Caixa de seleção (Próprio Autor)

**Botão de opção (OptionButton)**

Semelhantes às caixas de seleção, porém não trabalham sozinhos; são necessários dois botões ou mais para que possa haver uma alternância entre os botões (Figura 7).



Figura - Botão de opção (Próprio Autor)

**Caixa de combinação (Combobox)**

As caixas de combinação servem para mostrar ao usuário múltiplas opções (Figura 8).



Figura - Caixa de combinação (Próprio Autor)

**Caixa de Listagem (ListBox)**

Basicamente, possuem as mesmas funções de uma caixa de combinação, porém exibem no formato de lista aberta (Figura 9).

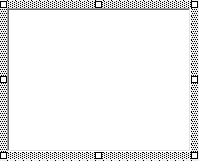


Figura - Caixa de Listagem (Próprio Autor)

## Metodologia

### A aplicação

A aplicação consiste numa customização através de um formulário, feito em VBA, para controlar um conjunto de 3 planilhas, que juntas realizam o controle de entrada e saída de produtos.

Uma planilha, intitulada “Movimentacao de produtos”, é responsável por armazenar os registros de movimentações de entrada e saída de produtos, recebendo as informações geradas pelo formulário quando o usuário aperta, no formulário o botão “Registrar”.

A planilha Movimentacao de Produtos é ilustrada na figura 10:



Figura - Planilha Movimentacao de Produtos (Próprio Autor)

O formulário é o agente que faz a interação com o usuário. Através dele é que o usuário fará cada registro de entrada ou de saída de produtos. O formulário é ilustrado na figura 11:

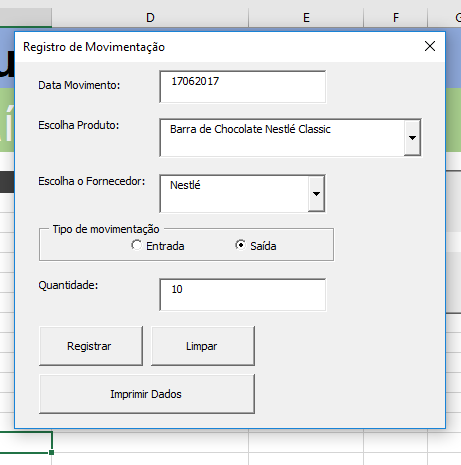


Figura - Formulário Registro de Movimentação (Próprio Autor)

A planilha “Fornecedores” contém a relação de fornecedores cadastrados, segundo a figura 12:



Figura - Planilha Fornecedores (Próprio Autor)

Por último, a planilha “Estoque Produtos” mostra a relação de produtos existentes e suas quantidades no estoque, segundo a figura 13:

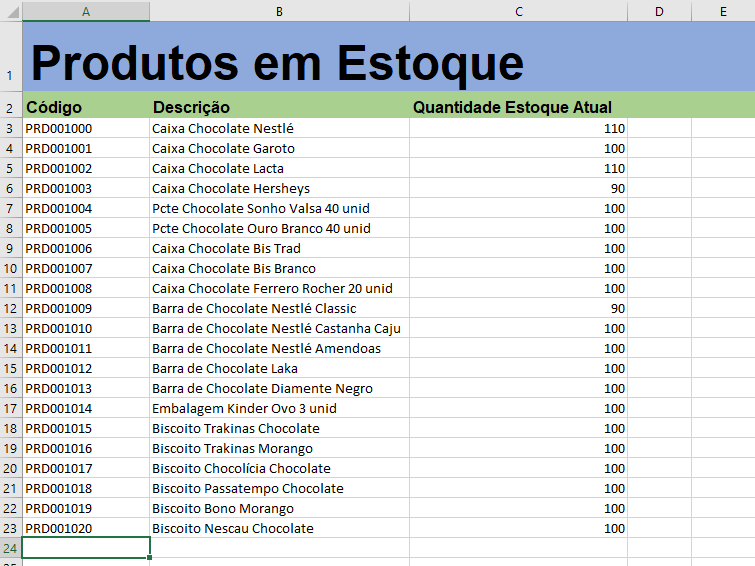


Figura - Planilha Estoque Produtos (Próprio Autor)

### A construção da aplicação

O início do trabalho de construção deste programa se deu com a elaboração de cada uma das três planilhas constantes no projeto e ilustradas nas fotos anteriores.

A próxima etapa foi a construção do formulário no ambiente VBA, bem como a designação dos botões de comando, rótulos, caixas de opção e caixas de combinação em seus devidos lugares; e suas nomeações como objetos. O formulário, no ambiente VBA, é ilustrado a seguir, na figura 14:

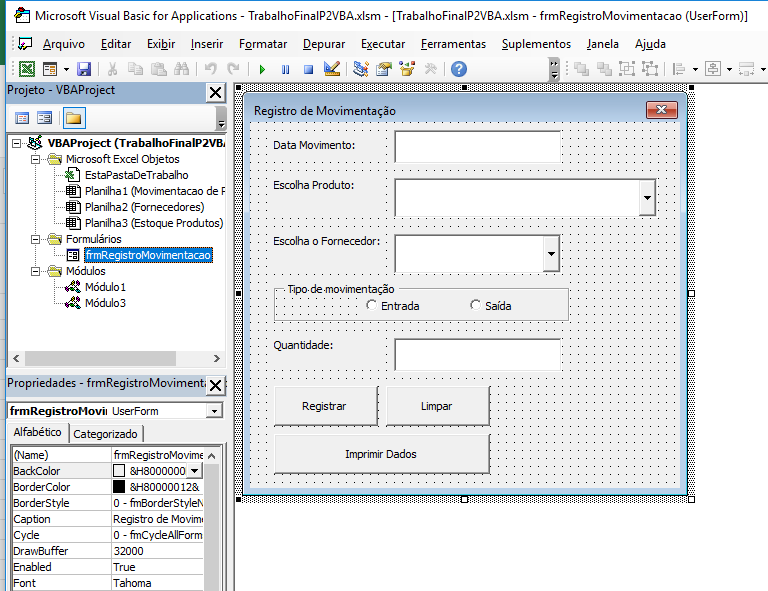


Figura - Visão geral do design do Formulário (Próprio Autor)

Para este formulário os recursos utilizados foram:

|  |  |
| --- | --- |
| **Objetos** | **Quantidade** |
| Formulário | 1 |
| Caixa de texto | 2 |
| Botão de comando | 3 |
| Rótulos | 4 |
| Botões de opção | 2 |
| Frame | 1 |

Figura - Tabela de objetos utilizados (Próprio Autor)

Os objetos que foram nomeados tiveram as seguintes identificações:

* Formulário – frmRegistroMovimentacao (name);
* Caixa de texto para “Data Movimento” – txtDataMovimento (name);
* Caixa de texto para “Quantidade” – txtQutde (name);
* Botão de comando para “Registrar” – btnRegistrar (name);
* Botão de comando para “Limpar” – btnLimpar (name);
* Botão de comando para “Imprimir Dados” – btnImprimirDados (name);
* Os 4 rótulos, como não sofreriam qualquer tratamento por parte do programa, não foram renomeados;
* Botão de opção para entrada – optEntProd
* Botão de opção para saída – optSaiProd
* O Frame também não sofreu tratamento lógico, então não foi renomeado.

Os nomes que aparecem nos corpos dos objetos são apenas nomes aparentes, da propriedade **Caption**, servem apenas para identificar os objetos graficamente para o usuário. Os nomes de objetos, que são os utilizados pelo programador, são os designados na propriedade **Name.**

### a programação

A maior parte do trabalho de programação se concentrou no botão de comando “Registrar”, que é o principal comando da aplicação. É nele que a lógica central de controle é executada.

Quando esse botão é pressionado, o programa verifica e garante todas as condições de preenchimento previstas, atualiza o conteúdo da planilha “Estoque Produtos”, registra a movimentação do produto na planilha “Movimentação de produtos” e apaga os campos para um novo ciclo de registro de produtos.

Os outros dois botões executam funções específicas no processo. O botão imprimir realiza a função de impressão dos dados da planilha “Movimentação de Produtos”.

Já o botão Limpar apaga os campos.

## resultados

Um teste será realizado para demonstrar o funcionamento do programa. A situação inicial é mostrada na figura 16:



Figura - Início do teste de funcionamento (Próprio Autor)

É notável que a última movimentação antes do teste consta na linha 10.

Ao pressionar o botão “Registrar Movimentação”, o formulário aparece conforme a figura 17:

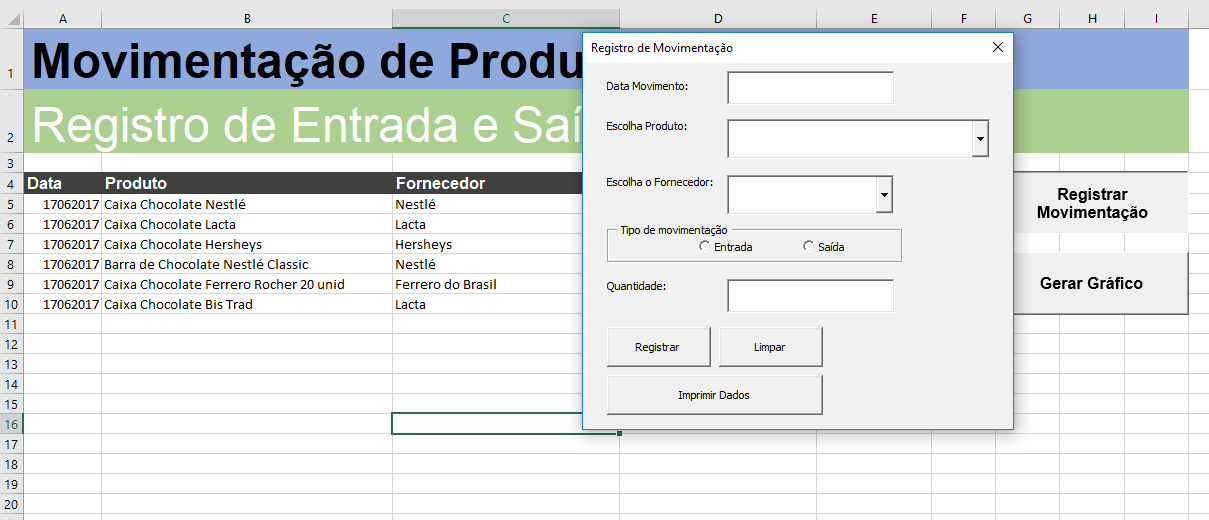


Figura - Condição inicial do formulário ao pressionar o botão "Registrar Movimentação" (Próprio Autor)

Os campos são monitorados, se o usuário deixar de preencher algum campo ou inserir uma informação diferente da qual é esperada pelo programa, uma mensagem aparece na tela notificando o usuário, como na figura 18:

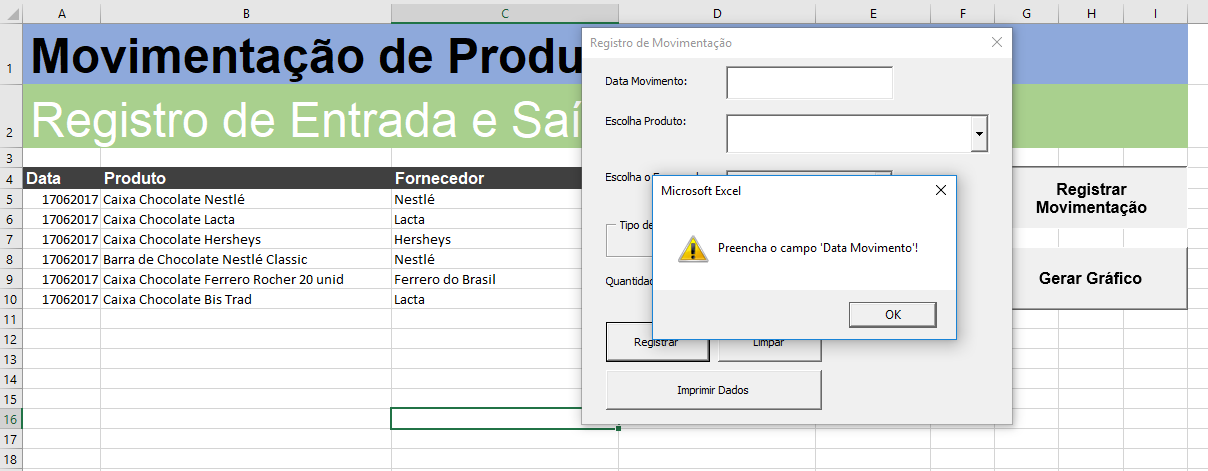


Figura - Mensagem de notificação ao usuário (Próprio Autor)

Ao preencher corretamente, como na figura 19:

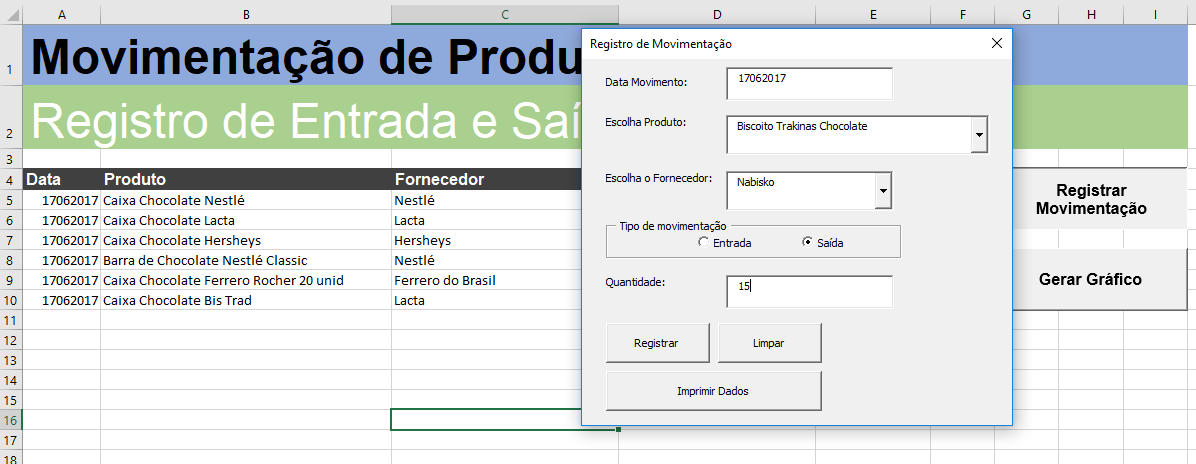


Figura - Preenchimento esperado pelo programa (Próprio Autor)

Uma mensagem aparece na tela avisando que o registro foi realizado, como na figura 20:

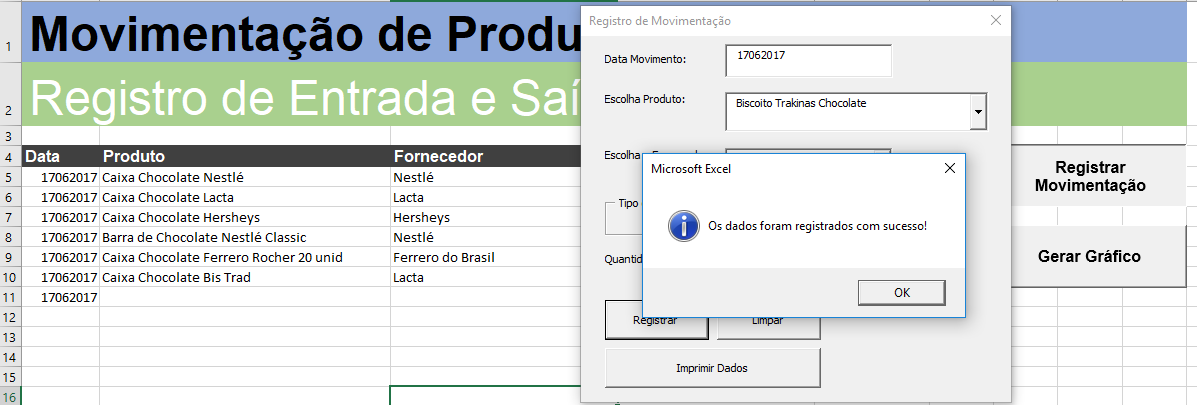


Figura - Confirmação de Registro (Próprio Autor)

O programa realiza o registro na planilha “Movimentação de Produtos”, como na figura 21:



Figura - Atualização do Registro (Próprio Autor)

É notável que o este último registro consta na linha seguinte, a linha 11.

E atualiza o estoque na planilha “Estoque Produtos”, conforme a figura 22:

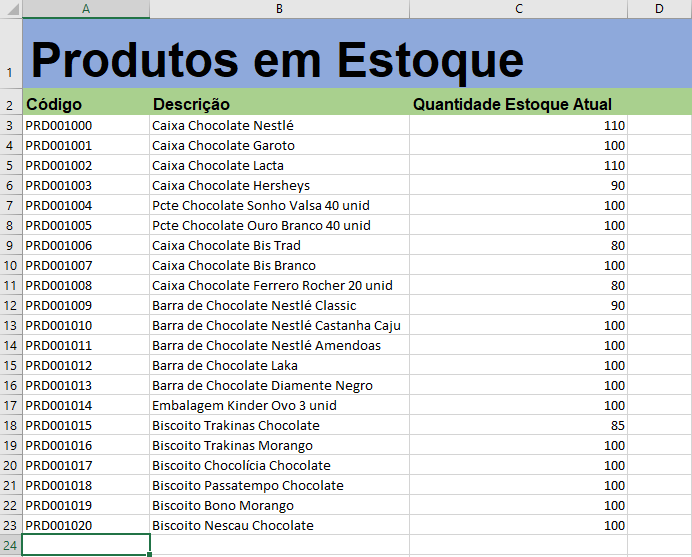


Figura - Estoque atualizado (Próprio Autor)

Após a atualização do registro, o produto escolhido – que foi o da linha 18 da planilha “Estoque Produtos” – sofreu uma diminuição em sua quantidade no estoque, uma vez que o movimento foi de saída dos estoques do produto.

**CONCLUSÃO:**

O funcionamento do programa desenvolvido reforça a ideia de que um software customizado pode auxiliar significativamente na melhoria dos processos de trabalho e controle nos estabelecimentos comerciais e empresariais em geral.

Um programa feito sob medida tem a capacidade de facilitar o cotidiano das pessoas, trazendo consigo muita facilidade, eficiência e agilidade no momento de lidar com os processos de trabalho, uma vez que dispensa dos usuários um conhecimento avançado em operação de programas dispendiosos e padroniza as operações a fim de obter ganhos de performance.

**REFERÊNCIAS BIBLIOGRÁFICAS**

BRASIL, M. (18 de Junho de 2017). *Tarefas Básicas no Excel*. Fonte: support.office.com: https://support.office.com/pt-br/article/Tarefas-b%c3%a1sicas-no-Excel-dc775dd1-fa52-430f-9c3c-d998d1735fca?ui=pt-BR&rs=pt-BR&ad=BR

FERNANDES, M. (2005). *Desenvolvendo Aplicações Poderosas com Excel e VBA.* Florianópolis: Visual Books Editora.

ROMAGNOLO, C. A. (18 de Junho de 2017). *Estudos sobre VBA: Introdução*. Fonte: Oficina da Net: https://www.oficinadanet.com.br/artigo/excel/estudos-sobre-vba-introducao

**APÊNDICE:**

Código-fonte do programa de computador:

Private Sub btnRegistrar\_Click()

'Declaração de variáveis e outros identificadores

Dim DataMovimento As String 'Variável de cópia do conteúdo do objeto txtDataMovimento

Dim Quantidade As Integer 'Variável de cópia do conteúdo do objeto txtQuantidade

Dim Produto As Variant 'Variável de cópia do conteúdo do objeto comboProduto

Dim Fornecedores As Variant 'Variável de cópia do conteúdo do objeto comboFornecedores

Dim InOut As String 'Variável de cópia do conteúdo dos objetos optbtnEntrada e optbtnSaida

Dim PlanMovimentacao As Worksheet 'Declaração das planilhas como objetos Worksheet

Dim PlanFornecedores As Worksheet

Dim PlanEstoques As Worksheet

Dim linha As Integer

Dim linha3 As Integer

'Setup das planilhas - Quem será quem?

'Setando a planilha 1 - Movimentação de Produtos

Set PlanMovimentacao = Worksheets(1)

'Encontra a primeira linha vazia da planilha

linha = PlanMovimentacao.Cells(Rows.Count, 1).End(xlUp).Offset(1, 0).Row

'Setando a planilha 2 - Fornecedores

Set PlanFornecedores = Worksheets(2)

'Setando a planilha 3 - Estoque de Produtos

Set PlanEstoques = Worksheets(3)

'IMPORTANTE !!! !!! !!! !!!

'OBS: O método "linha = PlanMovimentacao.Cells(Rows.Count, 1).End(xlUp).Offset(1, 0).Row"

'só deve ser empragado na planilha a ser preenchida pelo procedimento no momento da execução.

'Setar nas outras duas causa um erro de cálculo no método, de modo que a primeira linha se

'torna a linha 24 da planilha a ser atualizada pelo programa, durante sua execução!!!

'INTERPRETAÇÃO

'Colocando o método "linha = PlanMovimentacao.Cells(Rows.Count, 1).End(xlUp).Offset(1, 0).Row"

'nas três planilhas, o programa examinará as três planilhas sucessivamente em busca da primeira

'linha vazia e usará a linha vazia da última planilha examinada, que é a planilha

'"Estoques de Produtos" com a linha 24 como primeira linha vazia, como base para o progresso

'da execução do procedimento de registro!!!!!!

'Estrutura Condicional:

'Verificar a conformidade do preenchimento do usuário

'Preenchimento de acordo ou não?

'Preenchimento do Objeto TextBox txtDataMovimento.Text

'Estar atento para a ordem dos ifs para que se faça o

'condicionamento da forma correta!

'Condição 1 - Estar Preenchido

If Trim(txtDataMovimento.Text) = "" Then

MsgBox "Preencha o campo 'Data Movimento'!", vbExclamation

Exit Sub

End If

'Condição 2 - Campo ser numérico

If Not IsNumeric(txtDataMovimento.Text) Then

MsgBox "O campo Data deve ser numérico!", vbOKOnly

txtDataMovimento.Text = ""

txtDataMovimento.SetFocus

Exit Sub

End If

'Preenchimento do Objeto TextBox txtQutde.Text

'Estar atento para a ordem dos ifs para que se faça o

'condicionamento da forma correta!

'Condição 1 - Estar Preenchido

If Trim(txtQutde.Text) = "" Then

MsgBox "Preencha o campo 'Quantidade'!", vbExclamation

Exit Sub

End If

'Condição 2 - Campo ser numérico

If Not IsNumeric(txtQutde.Text) Then

MsgBox "O campo Quantidade deve ser numérico!", vbOKOnly

txtQuantidade.Text = ""

txtQuantidade.SetFocus

Exit Sub

End If

'Preenchimento dos objetos combobox comboProduto e comboFornecedor

If frmRegistroMovimentacao.comboProduto.Value = "" Or \_

frmRegistroMovimentacao.comboFornecedor.Value = "" Then

MsgBox "Preenchimento obrigatório!", vbCritical, "ATENÇÃO!!!!!"

Exit Sub

End If

'Estrutura de controle dos objetos optionButtons optEntProd e optSaiProd

'Aproveitado a PlanMovimentacao como Worksheet

PlanMovimentacao.Select

'Testar se o controle de preenchimento foi clicado

If frmRegistroMovimentacao.optEntProd.Value = False And \_

frmRegistroMovimentacao.optSaiProd.Value = False Then

MsgBox "Selecione o tipo de movimentação"

Exit Sub

End If

'condição para registrar o tipo de movimento e atualizar o conteúdo da planilha Estoque produtos

If frmRegistroMovimentacao.optEntProd.Value = True And frmRegistroMovimentacao.optSaiProd.Value = False Then

For linha3 = 1 To Planilha3.Cells(Rows.Count, 1).End(xlUp).Offset(1, 0).Row

If Planilha3.Cells(linha3, 2).Value = comboProduto.Value Then

Planilha3.Cells(linha3, 3).Value = Planilha3.Cells(linha3, 3).Value + txtQutde.Text

End If

Next

InOut = "Entrada"

Else

For linha3 = 1 To Planilha3.Cells(Rows.Count, 1).End(xlUp).Offset(1, 0).Row

If Planilha3.Cells(linha3, 2).Value = comboProduto.Value Then

Planilha3.Cells(linha3, 3).Value = Planilha3.Cells(linha3, 3).Value - txtQutde.Text

End If

Next

InOut = "Saída"

End If

'Procedimento de integração entre o formulário e as planilhas - Transferência da informação:

'Etapa 1:

'Transferência dos dados do formulário para as suas

'Variáveis Receptoras específicas

DataMovimento = txtDataMovimento.Text

Quantidade = txtQutde.Text

Produto = comboProduto.Value

Fornecedor = comboFornecedor.Value

'Etapa 2:

'Transferência dos dados contidos em cada variável para as

'coordenadas designadas nas planilhas

'Ex: ws.Cells(linha, 1).Value = codigoProduto

PlanMovimentacao.Cells(linha, 1).Value = DataMovimento

PlanMovimentacao.Cells(linha, 5).Value = Quantidade

PlanMovimentacao.Cells(linha, 2).Value = Produto

PlanMovimentacao.Cells(linha, 3).Value = Fornecedor

PlanMovimentacao.Cells(linha, 4).Value = InOut

'Mensagem do resultado do procedimento:

MsgBox "Os dados foram registrados com sucesso!", vbInformation

'Reset dos campos preenchidos após o registro

LimparPreenchimento

End Sub

Sub LimparPreenchimento()

'Reset dos campos preenchidos após o registro

'Ex: txtCodigoProduto.Text = ""

txtDataMovimento.Text = ""

txtQutde.Text = ""

comboProduto = Clear

comboFornecedor = Clear

'Focar o cursor em txtDataMovimento

txtDataMovimento.SetFocus

End Sub

Private Sub btnLimpar\_Click()

LimparPreenchimento

End Sub

Private Sub UserForm\_Initialize()

'Preenchimento do objeto ComboBox comboProduto

'Setando a primeira linha da varredura

linha1 = 3

'Laço de repetição para localizar e exibir todos os ítens da coluna 2 da planilha 3

'no objeto ComboBox comboProduto

'Esse é o procedimento mais simples, porém funciona!!!!!!

Do Until Planilha3.Cells(linha1, 2) = ""

comboProduto.AddItem Planilha3.Cells(linha1, 2)

linha1 = linha1 + 1

Loop

'Preenchimento do objeto ComboBox comboFornecedor

'Setando a primeira linha da varredura

linha2 = 3

'Laço de repetição para localizar e exibir todos os ítens da coluna 2 da planilha 2

'no objeto ComboBox comboFornecedor

'Esse é o procedimento mais simples, porém funciona!!!!!!

Do Until Planilha2.Cells(linha2, 2) = ""

comboFornecedor.AddItem Planilha2.Cells(linha2, 2)

linha2 = linha2 + 1

Loop

End Sub

Private Sub btnImprimirDados\_Click()

Imprimir

End Sub