



UNIVERSIDADE FEDERAL DO PARANÁ

CURSO DE AGRONOMIA

Victor Hugo Basegio

Rogério Antonio Pinto da Silva Filho

Yan Matheus Iliano

Sistema de Controle de Estocagem e Armazenamento

Curitiba

2019

Victor Hugo Basegio GRR20194251

Rogério Antonio Pinto da Silva Filho GRR20194178

Yan Matheus Iliano GRR20194168

Sistema de Controle de Estocagem e Armazenamento

Relatório apresentado à disciplina Fundamentos da
Programação de Computadores do Curso de Agronomia da
Universidade Federal Paraná.

Orientador: Prof. Jackson Antônio do Prado Lima

Curitiba, junho de 2019

SUMÁRIO

1 INTRODUÇÃO	4
2 OBJETIVOS.....	5
3 DESENVOLVIMENTO	6
3.1 JANELA.....	8
3.2 COMANDOS.....	10
3.3 APLICATIVO.....	11
4 CONCLUSÃO	12
REFERÊNCIAS.....	13

1 Introdução

Neste relatório, será apresentado informações relativas ao trabalho realizado em phyton sobre Estocagem e Armazenamento.

Python é uma linguagem de programação criada por Guido van Rossum em 1991, sendo uma linguagem que foi criada para produzir código bom e fácil de manter de maneira rápida.

2. OBJETIVOS

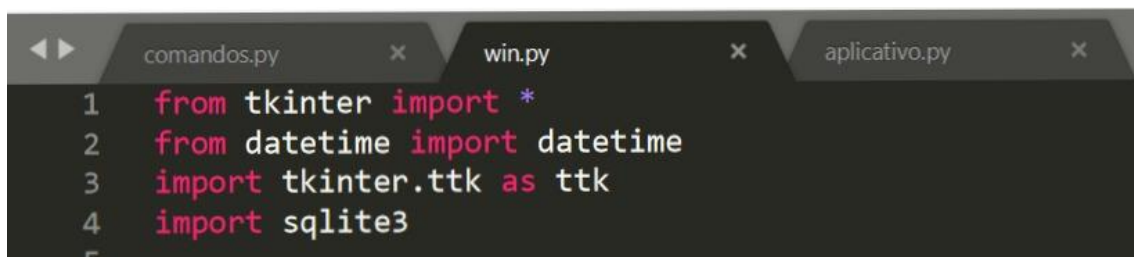
O objetivo foi criar um programa que armazenasse e estocasse qualquer coisa, com o intuito de fazer o usuário ter controle sobre seu estoque. O programa pede para o usuário informar além do nome, os preço e quantidades iniciais do produto, havendo também a possibilidade de excluir ou alterar esses valores após o salvamento no banco de dados.

3 DESENVOLVIMENTO

O trabalho é dividido em três pastas principais: Janela, aplicativo(executando o programa) e comandos(conectando a janela com o aplicativo).

3.1 JANELA

Bibliotecas:

A screenshot of a code editor with three tabs: 'comandos.py', 'win.py', and 'aplicativo.py'. The 'comandos.py' tab is active, showing the following Python code:

```
1 from tkinter import *
2 from datetime import datetime
3 import tkinter.ttk as ttk
4 import sqlite3
5
```

Janela: Criação da classe, podendo utilizar tudo q há nela em outro arquivo.

A screenshot of a code editor showing the definition of the 'Janela' class in a file. The code is as follows:

```
5
6 class janela():
7     #Janela
8     win = Tk()
9     win.title("AgropopsOS 2.0.0")
10    win.configure(background = "khaki1")
11
```

Três principais variáveis: nome, quantidade e preço do produto, definidas pelo usuário.

```

12
13     #Aqui ficam as variaveis, por isso o "Var"
14     nome = StringVar()
15     quant = StringVar()
16     preco = StringVar()
17
18

```

Parte estética da nossa janela: Apenas textos, sem impotência na execução do aplicativo.

```

16     preco = StringVar()
17
18
19     #Texto
20     label_nome = Label(win , text = "Nome:" , bg = "khaki1" , fg = "black" , font = "Arial 12")
21     label_quantidade = Label(win , text = "Quantidade:" , bg = "khaki1" , fg = "black" , font = "Arial 12")
22     label_preco = Label(win , text = "Preço de compra:" , bg = "khaki1" , fg = "black" , font = "Arial 12")
23     label_hora = Label(win , text = datetime.now() , bg = "khaki1" , fg = "black" , font = "Arial 8")
24     label_marca = Label(win , text = "AgropopsOS®" , bg = "khaki1" , fg = "black" , font = "Arial 7")
25
26
27     #Caixa de entrada
28     entnome = Entry(win , width = 20 , bg = "white" , textvariable = nome)

```

Caixa de entrada: Também definida pela biblioteca tkinter(função Entry), havendo três caixas de entrada(nome, quantidade e preço do produto). Há também o relacionamento da variável de texto com as três principais variáveis.

```

22     label_preco = Label(win , text = "Preço de compra:" , bg = "khaki1" , fg = "black" , font = "Arial 12")
23     label_hora = Label(win , text = datetime.now() , bg = "khaki1" , fg = "black" , font = "Arial 8")
24     label_marca = Label(win , text = "AgropopsOS®" , bg = "khaki1" , fg = "black" , font = "Arial 7")
25
26
27     #Caixa de entrada
28     entnome = Entry(win , width = 20 , bg = "white" , textvariable = nome)
29     entquant = Entry(win , width = 20 , bg = "white" , textvariable = quant)
30     entpreco = Entry(win , width = 20 , bg = "white" , textvariable = preco)
31

```

Criação dos botões: Que irão se conectar com as funções da outra pasta

```

33     #botao
34     botao_adicionar = Button(win , width = 20 , bg = "gray" , fg = "white" , text = "Adicionar")
35     botao_atualizar = Button(win , width = 20 , bg = "gray" , fg = "white" , text = "Atualizar selecionados")
36     botao_fechar = Button(win , width = 20 , bg = "red" , fg = "white" , text = "Fechar")
37     botao_deletar = Button(win , width = 20 , bg = "gray" , fg = "white" , text = "Deletar")
38     botao_visualizar = Button(win , width = 20 , bg = "gray" , fg = "white" , text = "Ver lista")
39     botao_pesquisar = Button(win , width = 20 , bg = "gray" , fg = "white" , text = "Pesquisar")
40
41     #Lista
42     lista = Listbox(win , width = 100)
43     scroll_lista = Scrollbar(win)
44     scroll_listax = Scrollbar(win , width = 17)

```

Lista: Onde aparecerá o nome e preço dos produtos, a função Scrollbar realiza o rolamento da tela, sendo criadas duas(uma no eixo X e outra no eixo Y), sendo uma função do tkinter.

```

40
41     #Lista
42     lista = Listbox(win , width = 100)
43     scroll_lista = Scrollbar(win)
44     scroll_listax = Scrollbar(win , width = 17)
45
46     #Decisões dos elementos na grade

```


3.2 COMANDOS

Iniciar: Criando uma tabela (se ela ainda não existir ainda), chamada de estoque. Havendo um código único para cada produto.

```

3  db = "banco.db"
4  #Conexão principal e criação da tabela
5
6  def iniciar():
7      conn = sqlite3.connect(db)
8      cur = conn.cursor()
9      cur.execute("CREATE TABLE IF NOT EXISTS estoque(codigo INTEGER PRIMARY KEY, txtproduto TEXT, p
10
11      conn.commit()
12      conn.close()
13

```

Adicionar: Inserindo os valores informados pelo usuário nas variáveis.

```

12      conn.close()
13
14  def adicionar(txtproduto, produto, txtquant, quant, txtpreco, preco, txthora, hora):
15
16      conn = sqlite3.connect(db)
17      cur = conn.cursor()
18      cur.execute("INSERT INTO estoque VALUES(NULL, ?, ?, ?, ?, ?, ?, ?, ?)" , (txtproduto, produto, txtqu
19
20      conn.commit()
21      conn.close()
22
23

```

Deletar: Função do sqlite3, que procura o código do produto selecionado e então ocorre o deleta do banco de dados.

```

25  def deletar(codigo):
26      conn = sqlite3.connect(db)
27      cur = conn.cursor()
28      cur.execute("DELETE FROM estoque WHERE codigo = ?", (codigo,))
29
30      conn.commit()
31      conn.close()
32

```

Pesquisa: Seleciona o produto presente no estoque igual ao produto pesquisado.

```

33 def pesquisar(produto = "", quant = "", preco = ""):
34     conn = sqlite3.connect(db)
35     cur = conn.cursor()
36     cur.execute("SELECT * FROM estoque WHERE produto = ? or quant = ? or preco = ?", (produto, quant, preco))
37     linhas = cur.fetchall()
38
39     conn.commit()
40     conn.close()
41
42     return linhas
43
44 def atualizar(codigo, produto, quant, preco, hora):

```

Atualização: Também uma função do sqlite3, que realiza o update dos novos valores selecionados.

```

39     conn.commit()
40     conn.close()
41
42     return linhas
43
44 def atualizar(codigo, produto, quant, preco, hora):
45     conn = sqlite3.connect(db)
46     cur = conn.cursor()
47     cur.execute("UPDATE estoque SET produto = ?, quant = ?, preco = ?, hora = ? WHERE codig
48

```

Vizualização: Seleciona todos os produtos presentes no banco de dados e os apresenta para o usuário.

```

52 def visualizar():
53     conn = sqlite3.connect(db)
54     cur = conn.cursor()
55     cur.execute("SELECT * FROM estoque")
56     linhas = cur.fetchall()
57
58
59     conn.commit()
60     conn.close()
61
62     return linhas
63

```

3.3 APLICATIVO

Inicialmente foi feita a importação da janela, dos comandos, e da biblioteca datetime.

```
1 from win import *
2 import comandos as com
3 from datetime import date
4
5
6 geral = janela()
7
8
```

Botões: Conecta todas as funções com os botões do programa.

```
65 geral.botao_visualizar.configure(command = comando_visualizar)
66 geral.botao_pesquisar.configure(command = comando_pesquisar)
67 geral.botao_adicionar.configure(command = comando_adicionar)
68 geral.botao_atualizar.configure(command = comando_atualizar)
69 geral.botao_deletar.configure(command = comando_deletar)
70 geral.botao_fechar.configure(command = geral.win.destroy)
71
72 com.iniciar()
73 geral.run()
74
```

4 Conclusão

Apesar da dificuldade e sendo finalizado na semana de provas, conseguimos realizar um bom trabalho, tivemos inúmeras dificuldades como na parte de atualização dos valores selecionados. O conteúdo de vetores trabalhados dentro da sala de aula foi bem importante para a realização do nosso aplicativo.

