Relatório Técnico

(SA_Somativa2)

Aula: T.I. – Python
Professor(a): Eduardo Francisco Maiese Furlanetti
Alunos: Henrique M. - João - Joel - Kayque - Victor
Curso: 1ºDEVT - SENAI
Data: 02/06/2025

Sumário

1. Problemática

Área: *Logística*

Gerador de Etiquetas

No setor de logística, a organização e a identificação correta de produtos são fundamentais para garantir um controle eficiente do estoque e da distribuição. A falta de etiquetas padronizadas, contendo informações como **nome, código ou QR Code,** pode gerar erros, extravios e atrasos. Para resolver esse problema, desenvolvemos um programa que gera automaticamente etiquetas personalizadas, facilitando a identificação e a rastreabilidade dos itens.

---

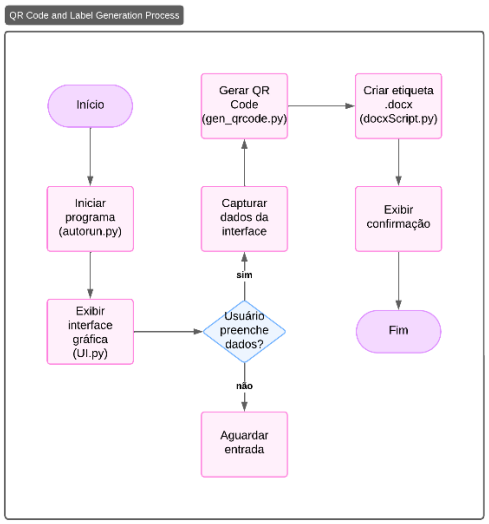
2. Divisão de Tarefas

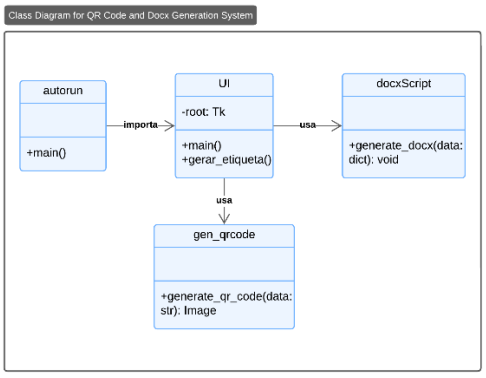
INTEGRANTES	TAREFAS
Henrique M.	Código/UI
João	Código/docx
Joel	Relatório e Apresentação
Kayque	Código/qrCode
Victor	Código/UI

*Todos os integrantes contribuíram com ideias e soluções durante a criação do código final e suas vertentes, colaborando ativamente na definição das funcionalidades, na divisão das tarefas e na resolução dos desafios encontrados ao longo do desenvolvimento.

---

3. Fluxogramas





4. Programação

autorun.py

```
# Bibliotecas
import time

# Importa e inicia a interface gráfica
import UI
UI
print("UI concluído")

# Importa e gera o QR Code
import gen_qrcode
gen_qrcode
print("QR CODE gerado")
time.sleep(1)  # Espera 1 segundo

# Importa o código do produto da UI
from UI import prod_codigo

# Importa e gera o documento Word
import docxScript
docxScript

print("Documento gerado!")
```

Este arquivo **orquestra a execução do programa**. Ele importa os módulos responsáveis por criar a interface gráfica, gerar o QR Code e gerar o documento. A execução de cada módulo é confirmada com uma mensagem no console, e há um pequeno atraso com sleep(1) para melhorar a fluidez da execução.

docxScript.py

```python
# Bibliotecas
from docx import Document
from docx.shared import Inches
from docx.enum.text import WD_ALIGN_PARAGRAPH
import os

# Importa variáveis da interface
from UI import prod_codigo
from UI import nome_produto

# Cria documento Word
doc = Document()

# Define as configurações da página: 2x3 polegadas e sem margens
section = doc.sections[0]
section.page_width = Inches(2)
section.page_height = Inches(3)
section.top_margin = section.right_margin = section.left_margin =
section.bottom_margin = 0

# Adiciona título e código do produto
title = doc.add_heading(f"Etiqueta: {nome_produto}", level=1)
text = doc.add_paragraph(str(prod_codigo))

# Insere QR Code da imagem gerada
qrcode_p = doc.add_paragraph("")
run = qrcode_p.add_run()
run.add_picture(f"qrcode_{prod_codigo}.png", width=Inches(1.0))

# Centraliza todos os elementos
title.alignment = WD_ALIGN_PARAGRAPH.CENTER
text.alignment = WD_ALIGN_PARAGRAPH.CENTER
qrcode_p.alignment = WD_ALIGN_PARAGRAPH.CENTER

# Salva o documento na pasta Docs/
filename = f"Docs/tag_{prod_codigo}.docx"
os.makedirs(os.path.dirname(filename), exist_ok=True)
doc.save(filename)
print(f"Documento salvo como: {os.path.abspath(filename)}")
```

Este módulo **gera o arquivo .docx da etiqueta**. Ele:

- Cria um documento do Word com tamanho pequeno (2x3 polegadas);
- Adiciona o nome do produto como título e o código gerado como parágrafo;
- Insere o QR Code correspondente ao código do produto;
- Centraliza todos os elementos e salva o arquivo em uma pasta chamada Docs.

gen_qrcode.py

```
# Bibliotecas
import qrcode
import qrcode.constants

# Importa variável da interface
from UI import prod_codigo

# Configura o QR Code
qr = qrcode.QRCode(
    version=1,  # Versão do QR Code (tamanho fixo)
    box_size=5,  # Tamanho dos blocos
    border=1,  # Borda ao redor
    error_correction=qrcode.constants.ERROR_CORRECT_Q  # Correção de erro (Q
(Quartile) = 25% dos dados)
)

# Adiciona dados e gera o QR Code
qr.add_data(prod_codigo)
qr.make(fit=True)

# Cria e salva a imagem do QR Code
img = qr.make_image(fill_color="black", back_color="white")
img.save(f"qrcode_{prod_codigo}.png")
```

Este módulo **gera a imagem do QR Code**. Ele:

- Usa a biblioteca qrcode com configurações específicas (tamanho, borda, correção de erro);
- Adiciona o código do produto como dado principal do QR Code;
- Gera a imagem em preto e branco e a salva com nome correspondente ao código (qrcode_<código>.png).

UI.py

```
# Bibliotecas
import tkinter as tk
import random
import string

# Função para gerar o código da etiqueta
def gerar_codigo():
    global prod_codigo, status, r, nome_produto
    nome_produto = entrada_nome.get()  # Captura o nome digitado
    if not nome_produto.strip():  # Verifica se está vazio
        status.config(text="Por favor, insira o nome do produto.", fg="red")  #
Mostra erro
        return
    # Gera um código de 9 caracteres (letras e números)
    codigo = ''.join(random.choice(string.ascii_uppercase + string.digits) for _
in range(9))
    prod_codigo = codigo  # Salva o código
    status.config(text=f"Etiqueta gerada para '{nome_produto}' com sucesso!",
fg="green")  # Mensagem de sucesso
    r.destroy()  # Fecha a janela

# Cria a janela principal
r = tk.Tk()
r.title("Gerador de Etiquetas")
r.geometry("400x250")

# Configura o layout para centralizar os elementos
r.grid_rowconfigure(0, weight=1)
r.grid_rowconfigure(7, weight=1)
r.grid_columnconfigure(0, weight=1)
r.grid_columnconfigure(1, weight=1)

# Título da aplicação
tk.Label(r, text="Gerador de Etiquetas", font=("Helvetica", 16,
"bold")).grid(row=1, column=0, columnspan=2, pady=10)

# Campo para digitar o nome do produto
tk.Label(r, text="Nome do produto:").grid(row=2, column=0, sticky="e", padx=5)
entrada_nome = tk.Entry(r, width=30)
entrada_nome.grid(row=2, column=1, padx=5)

# Botão para gerar o código
tk.Button(r, text="Gerar/Usar Código", command=gerar_codigo, bg="green",
fg="white").grid(row=3, column=0, columnspan=2, pady=10)

# Label para mostrar o código (não está sendo usado)
codigo_label = tk.Label(r, text="", font=("Helvetica", 12))
codigo_label.grid(row=4, column=0, columnspan=2)
```

```python
# Label para mensagens de status
status = tk.Label(r, text="", font=("Helvetica", 10))
status.grid(row=5, column=0, columnspan=2)

# Inicia a interface gráfica
r.mainloop()
```

Este módulo contém a **interface gráfica para entrada de dados**:

- Exibe uma janela com um campo para digitar o nome do produto;
- Ao clicar no botão, o programa valida o nome e gera um código aleatório de 9 caracteres;
- Mostra mensagens de erro ou sucesso;
- Fecha a janela automaticamente após a geração do código.

---

5. Descritivo Geral (breve)

O programa inicia com uma interface gráfica onde o usuário pode preencher campos como nome, código e outras informações de um produto. Ao clicar em "Gerar Etiqueta", o programa:

1. Gera um **QR Code** com base nos dados inseridos;
2. Cria um documento Word (`.docx`) com esses dados organizados em formato de etiqueta;
3. Exibe uma mensagem de sucesso e permite salvar o arquivo.

Esse processo padroniza e automatiza a geração de etiquetas para controle logístico.