

Ficha de Avaliação Final	
<b>Curso:</b>	UFCD 10793
<b>UFCD/Módulo/Temática:</b>	UFCD 10793 - Fundamentos de Python
<b>Ação:</b>	10793_2/AT & 10793_5/N
<b>Formador/a:</b>	Sandra Liliana Meira de Oliveira
<b>Data:</b>	março de 2025
<b>Nome do Formando/a:</b>	

O presente documento é composto por uma parte teórica e uma parte prática. A resolução da parte prática corresponde à realização da Ficha de Avaliação Final.

Impressoras Zebra & Liguagem ZPL .....	2
Principais características do ZPL: .....	2
Comandos comuns: .....	2
Documentação e recursos .....	3
Simuladores e editores online para ZPL .....	3
Exemplos .....	4

# Teoria

## Impressoras Zebra & Liguagem ZPL

A linguagem **ZPL (Zebra Programming Language)** é uma linguagem de descrição de etiquetas desenvolvida pela empresa **Zebra Technologies**. É usada principalmente para configurar e imprimir etiquetas em impressoras térmicas Zebra, suportando elementos como texto, códigos de barras, QR Codes, gráficos e formatações específicas para etiquetas.

### Principais características do ZPL:

- **Baseada em comandos ASCII** que começam com ^ (caret) ou ~ (til).
- Muito leve e rápida, ideal para dispositivos com poucos recursos.
- Pode ser gerada manualmente ou programaticamente (ex: por scripts Python, C#, etc).
- Permite o envio direto para a impressora através de ficheiros .zpl, porta COM, USB ou rede (socket TCP/IP).
- Suporta diferentes tipos de códigos de barras, QR Codes, fontes escaláveis e bitmaps.

### Comandos comuns:

- ^XA — Início de etiqueta
- ^FOx,y — Define posição (Field Origin)
- ^A — Define a fonte
- ^FD — Field Data (texto a imprimir)
- ^BC — Código de barras Code128
- ^BQN — QR Code
- ^XZ — Fim da etiqueta

### Exemplo básico:

```
zpl
CopyEdit
^XA
^FO50,50^A0N,50,50^FDolá Mundo!^FS
^FO50,150^BCN,100,Y,N,N
^FD1234567890^FS
^XZ
```

## Documentação e recursos

### **Zebra ZPL II Programming Guide (PDF oficial)**

<https://www.zebra.com/content/dam/zebra/manuals/en-us/printers/zpl-zbi2-pm-en.pdf>

### **Zebra Developer Portal - Página dedicada ao ZPL**

<https://developer.zebra.com/community/technologies/barcode-printers/zpl>

### **Zebra Support and Downloads**

<https://www.zebra.com/us/en/support-downloads.html>

### **Zebra Brasil - Suporte técnico em português**

<https://www.zebra.com/br/pt/support-downloads.html>

### **DevMedia – Impressão de etiquetas com ZPL (explicação e exemplos)**

<https://www.devmedia.com.br/impressao-de-etiquetas-com-zpl/41902>

### **Fórum Clube Delphi – Pesquisar por “ZPL” para exemplos e dúvidas**

<https://www.clubedelphi.com.br/forum>

### **Como imprimir etiquetas em impressora Zebra usando ZPL - Smart Solutions**

[https://www.youtube.com/watch?v=IRYZ8xqUI\\_k](https://www.youtube.com/watch?v=IRYZ8xqUI_k)

### **Zebra ZPL - Como imprimir etiqueta simples - Hacatec Brasil**

<https://www.youtube.com/watch?v=bXM3Nq4Malo>

### **Imprimindo QR Code em Impressora Zebra com ZPL - Imporium**

<https://www.youtube.com/watch?v=LnQkxUINmfA>

### **Como montar etiquetas com ZPL e visualizar no browser (Visual ZPL) - Impressoras Zebra**

<https://www.youtube.com/watch?v=VITdRB6q7gM>

## Simuladores e editores online para ZPL

### **Labelary Online ZPL Viewer and Converter**

Editor e renderizador online de etiquetas ZPL. Permite ver a etiqueta gerada diretamente no navegador.

<https://labelary.com/viewer.html>

### Labelary REST API Documentation

Caso queiras integrar a renderização de etiquetas nos teus próprios scripts ou aplicações (ex: em Python).

<https://labelary.com/service.html>

### ZPL Designer (projeto open-source para edição gráfica)

Ferramenta open-source para Windows que permite desenhar etiquetas e gerar o código ZPL correspondente.

<https://github.com/bozhu/ZPLDesigner>

(Download: <https://github.com/bozhu/ZPLDesigner/releases>)

### ZebraDesigner Essentials (software oficial da Zebra para Windows)

Ferramenta gráfica gratuita da Zebra para criação de etiquetas, com exportação para ZPL.

<https://www.zebra.com/us/en/products/software/barcode-printers/zebradesigner.html>

## Exemplos

### Usar curl com o Labelary (linha de comandos)

```
curl -o etiqueta.png -H "Content-Type: application/x-www-form-urlencoded" \  
--data '^XA^F050,50^A0N,50,50^FD01á Mundo!^FS^XZ' \  
https://api.labelary.com/v1/printers/8dpmm/labels/4x6/0/
```

Explicação dos parâmetros do URL:

- 8dpmm: resolução (8 dots per mm = 203 dpi)
- 4x6: tamanho da etiqueta (largura x altura, em polegadas)
- 0: rotação da etiqueta

O resultado será guardado num ficheiro chamado etiqueta.png.

## Usar Labelary com Python

Instalar primeiro a biblioteca requests: `pip install requests`

```
import requests

zpl = "^XA^F050,50^A0N,50,50^FD01á Mundo!^FS^XZ"
url = "https://api.labelary.com/v1/printers/8dpmm/labels/4x6/0/"

headers = {'Content-Type': 'application/x-www-form-urlencoded'}
response = requests.post(url, data=zpl, headers=headers)

if response.status_code == 200:
    with open("etiqueta.png", "wb") as f:
        f.write(response.content)
    print("Etiqueta gerada com sucesso.")
else:
    print(f"Erro: {response.status_code}")
    print(response.text)
```

## Exemplo avançado

Considera o ficheiro produtos.csv onde se encontram os seguintes dados:

```
nome,codigo
Produto A,123456789012
Produto B,987654321098
Produto C,192837465564
```

Vamos ver como é que a partir desse ficheiro, utilizando um exemplo completo em python:

- Lê os dados de um CSV.
- Gera uma etiqueta ZPL para cada linha.
- Envia cada etiqueta ao **Labelary API**.
- Guarda os ficheiros como PNG.

```
import csv
import requests
import os

# Pasta de saída
os.makedirs("etiquetas", exist_ok=True)

# Configuração da impressora e etiqueta
dpi = "8dpm"
size = "4x6"
rotation = "0"
labelary_url = f"https://api.labelary.com/v1/printers/{dpi}/labels/{size}/{rotation}/"
headers = {'Content-Type': 'application/x-www-form-urlencoded'}

# Lê os dados do CSV e gera etiquetas
with open("produtos.csv", newline='', encoding="utf-8") as csvfile:
    reader = csv.DictReader(csvfile)
    for i, row in enumerate(reader):
        nome = row["nome"]
        codigo = row["codigo"]

        # Geração do ZPL para cada produto
        zpl = f"""
^XA
^F050,50^A0N,40,40^FD{nome}^FS
^F050,120^BCN,100,Y,N,N
^FD{codigo}^FS
^XZ
"""

        response = requests.post(labelary_url, data=zpl.strip(), headers=headers)

        if response.status_code == 200:
            with open(f"etiquetas/etiqueta_{i+1}.png", "wb") as f:
                f.write(response.content)
            print(f"Etiqueta {i+1} gerada para {nome}")
        else:
            print(f"Erro ao gerar etiqueta {i+1} - {nome}: {response.status_code}")
```

## Resultado:

- São geradas imagens .png de etiquetas na pasta etiquetas/.
- Cada imagem terá o nome do produto e um código de barras correspondente.

Vamos ver agora como podemos:

- Gerar **PDFs** em vez de PNGs
- Criar **várias etiquetas** numa só página

- Usar **QR Codes** em vez de Code128
- Obter um **template parametrizável**

#### O exemplo seguinte:

- Lê dados de um CSV (nome e código)
- Gera etiquetas ZPL com:
  - a. Nome do produto
  - b. Código de barras (Code128)
  - c. QR Code (opcional)
- Converte para PNG e também para PDF
- Junta várias etiquetas num único PDF final

Este script usa as bibliotecas requests, reportlab, e Pillow. Certifica-te de as instalar antes de correr o código:

```
pip install requests pillow reportlab
```

```
import csv
import requests
import os
from PIL import Image
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import A4

# === CONFIGURAÇÕES ===

# Tamanho da etiqueta (em polegadas) para o Labelary
dpi = "8dpmm" # 8dpmm = 203 dpi
size = "4x6" # Largura x altura em polegadas
rotation = "0"
labelary_url = f"https://api.labelary.com/v1/printers/{dpi}/labels/{size}/{rotation}/"
headers = {'Content-Type': 'application/x-www-form-urlencoded'}

# Ficheiros
csv_input = "produtos.csv"
output_folder = "etiquetas"
pdf_final = "etiquetas_todas.pdf"

# === CRIA PASTA DE SAÍDA ===
os.makedirs(output_folder, exist_ok=True)
```

```
# Ficheiros
csv_input = "produtos.csv"
output_folder = "etiquetas"
pdf_final = "etiquetas_todas.pdf"

# === CRIA PASTA DE SAÍDA ===
os.makedirs(output_folder, exist_ok=True)

# === LÊ CSV E GERA ETIQUETAS ===
etiqueta_paths = []

with open(csv_input, newline='', encoding="utf-8") as csvfile:
    reader = csv.DictReader(csvfile)
    for i, row in enumerate(reader):
        nome = row["nome"]
        codigo = row["codigo"]

        # === GERA CÓDIGO ZPL PARA CADA PRODUTO ===
        zpl = f"""
^XA
^F050,50^A0N,40,40^FD{nome}^FS

^F050,120^BCN,100,Y,N,N
^FD{codigo}^FS

^F050,250^BQN,2,5
^FDLA,{codigo}^FS

^XZ
"""

        # === ENVIA PARA O LABELARY E GUARDA COMO PNG ===
        response = requests.post(labelary_url, data=zpl.strip(), headers=headers)

        if response.status_code == 200:
            img_path = os.path.join(output_folder, f"etiqueta_{i+1}.png")
            with open(img_path, "wb") as f:
                f.write(response.content)
            etiqueta_paths.append(img_path)
            print(f"[✓] Etiqueta {i+1} gerada para '{nome}'")
        else:
            print(f"[X] Erro ao gerar etiqueta {i+1}: {response.status_code} - {response.text}")

# === CRIA PDF ÚNICO COM TODAS AS ETIQUETAS ===

c = canvas.Canvas(pdf_final, pagesize=A4)
page_width, page_height = A4
x_margin = 30
y_margin = 30
x_spacing = 20
y_spacing = 20
```



```
# Tamanho padrão das etiquetas (em pixels)
label_width, label_height = 800, 480 # aproximado (4x6" @ 203dpi)

x = x_margin
y = page_height - y_margin - label_height * 0.24 # Escala para caber na página

labels_per_page = 0
for i, path in enumerate(etiqueta_paths):
    img = Image.open(path)
    img.thumbnail((label_width * 0.24, label_height * 0.24)) # Escalar para caber no A4
    img_path_temp = os.path.join(output_folder, f"_tmp_{i}.jpg")
    img.save(img_path_temp, "JPEG")

    c.drawImage(img_path_temp, x, y)

    x += img.width + x_spacing
    if x + img.width > page_width - x_margin:
        x = x_margin
        y -= img.height + y_spacing
        if y < y_margin:
            c.showPage()
            x = x_margin
            y = page_height - y_margin - img.height

    labels_per_page += 1

c.save()
print(f"[✓] PDF final criado com {labels_per_page} etiquetas: {pdf_final}")
```

**Como criar um GUI simples usando TKinter (já vem incluído com o Python na maioria das distribuições (Windows/Linux/macOS)) para:**

- Selecionar o ficheiro CSV
- Gerar as etiquetas (PNG + PDF)
- Exibir mensagens de progresso

O script continua a usar o **Labelary API** para renderizar as etiquetas e reportlab para o PDF.

```
import tkinter as tk
from tkinter import filedialog, messagebox
import csv
import requests
import os
from PIL import Image
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import A4

# === Função principal que gera as etiquetas ===
def gerar_etiquetas(csv_path):
    dpi = "8dpm"
    size = "4x6"
    rotation = "0"
    labelary_url = f"https://api.labelary.com/v1/printers/{dpi}/labels/{size}/{rotation}/"
    headers = {'Content-Type': 'application/x-www-form-urlencoded'}

    output_folder = "etiquetas"
    pdf_final = "etiquetas_todas.pdf"
    os.makedirs(output_folder, exist_ok=True)
    etiqueta_paths = []

    try:
        with open(csv_path, newline='', encoding="utf-8") as csvfile:
            reader = csv.DictReader(csvfile)
            for i, row in enumerate(reader):
                nome = row["nome"]
                codigo = row["codigo"]

                # === Gera ZPL com texto, código de barras e QR Code ===
                zpl = f"""
                ^XA
                ^F050,50^A0N,40,40^FD{nome}^FS
                ^F050,120^BCN,100,Y,N,N
                ^FD{codigo}^FS
                ^F050,250^BQN,2,5
                ^FDLA,{codigo}^FS
                ^XZ
                """

                response = requests.post(labelary_url, data=zpl.strip(), headers=headers)

                if response.status_code == 200:
                    img_path = os.path.join(output_folder, f"etiqueta_{i+1}.png")
                    with open(img_path, "wb") as f:
                        f.write(response.content)
                    etiqueta_paths.append(img_path)
                    print(f"Etiqueta {i+1} gerada")
                else:
                    print(f"Erro {response.status_code}: {response.text}")
                    messagebox.showerror("Erro", f"Falha ao gerar etiqueta {i+1} ({nome})")

    except Exception as e:
        messagebox.showerror("Erro", str(e))
    return
```

```
# === Geração do PDF ===
c = canvas.Canvas(pdf_final, pagesize=A4)
page_width, page_height = A4
x_margin = 30
y_margin = 30
x_spacing = 20
y_spacing = 20
label_width, label_height = 800, 480 # px

x = x_margin
y = page_height - y_margin - label_height * 0.24

for i, path in enumerate(etiqueta_paths):
    img = Image.open(path)
    img.thumbnail((label_width * 0.24, label_height * 0.24))
    temp_jpg = os.path.join(output_folder, f"_tmp_{i}.jpg")
    img.save(temp_jpg, "JPEG")

    c.drawImage(temp_jpg, x, y)

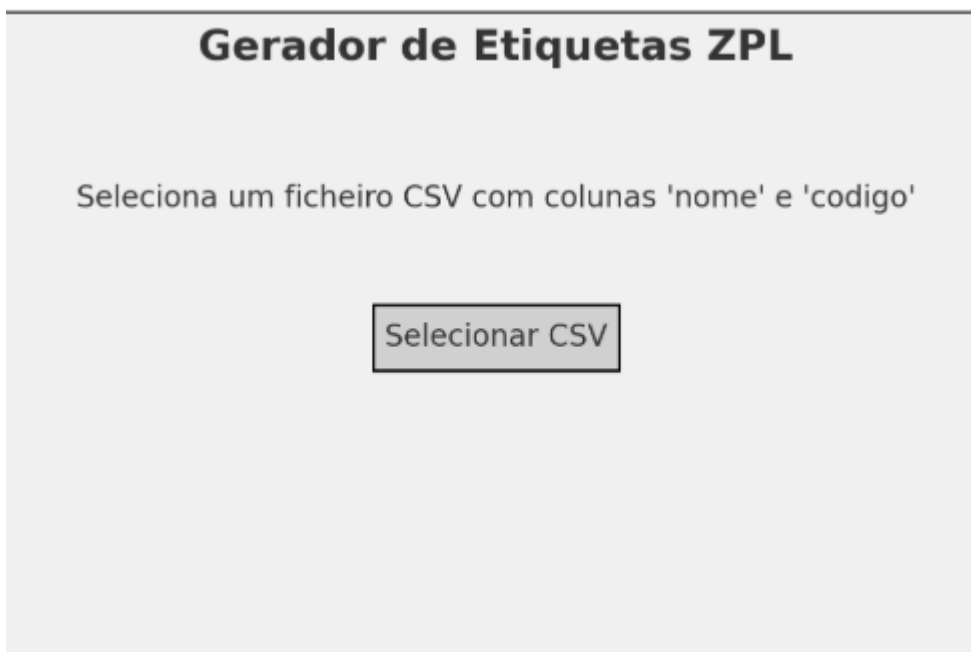
    x += img.width + x_spacing
    if x + img.width > page_width - x_margin:
        x = x_margin
        y -= img.height + y_spacing
        if y < y_margin:
            c.showPage()
            x = x_margin
            y = page_height - y_margin - img.height

c.save()
messagebox.showinfo("Concluído", f"PDF gerado com sucesso: {pdf_final}")

# === Função chamada ao clicar no botão "Selecionar CSV" ===
def escolher_csv():
    file_path = filedialog.askopenfilename(
        title="Selecione o ficheiro CSV",
        filetypes=[("Ficheiros CSV", "*.csv")]
    )
    if file_path:
        gerar_etiquetas(file_path)
```

```
# === Interface gráfica (Tkinter) ===  
janela = tk.Tk()  
janela.title("Gerador de Etiquetas ZPL")  
janela.geometry("400x150")  
  
label = tk.Label(janela, text="Selecione um ficheiro CSV com colunas 'nome' e 'codigo'")  
label.pack(pady=20)  
  
botao = tk.Button(janela, text="Selecionar CSV", command=escolher_csv)  
botao.pack()  
  
janela.mainloop()
```

O interface ficará similar a:



A janela inclui:

- Um **título** no topo: "Gerador de Etiquetas ZPL"
- Uma **instrução central** que orienta o utilizador a selecionar um ficheiro CSV com colunas nome e código
- Um **botão "Selecionar CSV"** que abre o seletor de ficheiros

Vamos agora melhorar o layout com os seguintes elementos adicionais:

- **Escolha da pasta de destino para salvar os ficheiros**
- **Seleção do tamanho da etiqueta** (ex: 4x6, 2x1, etc.)
- Layout mais organizado com espaçamentos, cores suaves e ícones opcionais

```
import tkinter as tk
from tkinter import filedialog, messagebox, ttk
import csv
import requests
import os
from PIL import Image
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import A4

# === Função para gerar etiquetas ===
def gerar_etiquetas(csv_path, output_folder, tamanho_etiqueta):
    dpi = "8dppm"
    rotation = "0"
    labelary_url = f"https://api.labelary.com/v1/printers/{dpi}/labels/{tamanho_etiqueta}/{rotation}/"
    headers = {'Content-Type': 'application/x-www-form-urlencoded'}

    pdf_final = os.path.join(output_folder, "etiquetas_todas.pdf")
    os.makedirs(output_folder, exist_ok=True)
    etiqueta_paths = []

    try:
        with open(csv_path, newline='', encoding="utf-8") as csvfile:
            reader = csv.DictReader(csvfile)
            for i, row in enumerate(reader):
                nome = row["nome"]
                codigo = row["codigo"]

                # Comando ZPL com nome, código de barras e QR Code
                zpl = f"""
^XA
^F050,50^A0N,40,40^FD{nome}^FS
^F050,120^BCN,100,Y,N,N
^FD{codigo}^FS
^F050,250^BQN,2,5
^FDLA,{codigo}^FS
^XZ
"""

                response = requests.post(labelary_url, data=zpl.strip(), headers=headers)

                if response.status_code == 200:
                    img_path = os.path.join(output_folder, f"etiqueta_{i+1}.png")
                    with open(img_path, "wb") as f:
                        f.write(response.content)
                    etiqueta_paths.append(img_path)
                else:
                    messagebox.showerror("Erro", f"Erro ao gerar etiqueta {i+1}: {response.status_code}")
                    return
    except Exception as e:
        messagebox.showerror("Erro", str(e))
        return
```

```
# Criação do PDF com todas as etiquetas
c = canvas.Canvas(pdf_final, pagesize=A4)
page_width, page_height = A4
x_margin, y_margin = 30, 30
x_spacing, y_spacing = 20, 20
label_width, label_height = 800, 480 # pixels simulados

x = x_margin
y = page_height - y_margin - label_height * 0.24

for i, path in enumerate(etiqueta_paths):
    img = Image.open(path)
    img.thumbnail((label_width * 0.24, label_height * 0.24))
    temp_jpg = os.path.join(output_folder, f"_tmp_{i}.jpg")
    img.save(temp_jpg, "JPEG")

    c.drawImage(temp_jpg, x, y)

    x += img.width + x_spacing
    if x + img.width > page_width - x_margin:
        x = x_margin
        y -= img.height + y_spacing
        if y < y_margin:
            c.showPage()
            x = x_margin
            y = page_height - y_margin - img.height

c.save()
messagebox.showinfo("Concluído", f"PDF criado: {pdf_final}")
```

# === Funções de interface ===

```
def escolher_csv():
    caminho = filedialog.askopenfilename(
        title="Selecionar ficheiro CSV",
        filetypes=[("Ficheiros CSV", "*.csv")]
    )
    if caminho:
        csv_entry.delete(0, tk.END)
        csv_entry.insert(0, caminho)

def escolher_pasta():
    caminho = filedialog.askdirectory(title="Selecionar pasta de destino")
    if caminho:
        pasta_entry.delete(0, tk.END)
        pasta_entry.insert(0, caminho)
```

```
def iniciar_geracao():
    csv_path = csv_entry.get()
    output_folder = pasta_entry.get()
    tamanho = tamanho_combo.get()

    if not csv_path or not output_folder:
        messagebox.showwarning("Atenção", "Por favor, seleciona o ficheiro CSV e a pasta de destino.")
        return

    gerar_etiquetas(csv_path, output_folder, tamanho)

# === GUI ===

janela = tk.Tk()
janela.title("Gerador de Etiquetas ZPL")
janela.geometry("500x300")
janela.resizable(False, False)

# === Layout ===

frame = tk.Frame(janela, padx=20, pady=20)
frame.pack(fill="both", expand=True)

tk.Label(frame, text="Ficheiro CSV:").grid(row=0, column=0, sticky="w")
csv_entry = tk.Entry(frame, width=40)
csv_entry.grid(row=0, column=1, padx=5)
tk.Button(frame, text="Selecionar", command=escolher_csv).grid(row=0, column=2)

tk.Label(frame, text="Pasta de destino:").grid(row=1, column=0, sticky="w")
pasta_entry = tk.Entry(frame, width=40)
pasta_entry.grid(row=1, column=1, padx=5)
tk.Button(frame, text="Selecionar", command=escolher_pasta).grid(row=1, column=2)

tk.Label(frame, text="Tamanho da etiqueta:").grid(row=2, column=0, sticky="w")
tamanho_combo = ttk.Combobox(frame, values=["4x6", "2x1", "3x2", "4x2"], width=10)
tamanho_combo.grid(row=2, column=1, sticky="w")
tamanho_combo.set("4x6")

tk.Button(janela, text="Gerar Etiquetas", bg="#4CAF50", fg="white", height=2, command=iniciar_geracao).pack(pady=15)

janela.mainloop()
```

### Vamos agora acrescentar:

- Pré-visualização da etiqueta gerada
- Mensagens de progresso em tempo real
- Tema escuro ou personalização de cores

Vamos implementar o **Passo 1: Pré-visualização da última etiqueta gerada.**

### O que será feito:

- Integraremos um **widget Label com PhotoImage** (usando Pillow) para mostrar a última etiqueta gerada.
- A imagem será atualizada dinamicamente sempre que uma nova etiqueta for processada.



Certifica-te de ter a biblioteca **Pillow** instalada (**pip install pillow**)

```
import tkinter as tk
from tkinter import filedialog, messagebox, ttk
from PIL import Image, ImageTk
import csv
import requests
import os
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import A4

# === Função para gerar etiquetas com pré-visualização ===
def gerar_etiquetas(csv_path, output_folder, tamanho_etiqueta):
    dpi = "8dppm"
    rotation = "0"
    labelary_url = f"https://api.labelary.com/v1/printers/{dpi}/labels/{tamanho_etiqueta}/{rotation}/"
    headers = {'Content-Type': 'application/x-www-form-urlencoded'}

    pdf_final = os.path.join(output_folder, "etiquetas_todas.pdf")
    os.makedirs(output_folder, exist_ok=True)
    etiqueta_paths = []

    try:
        with open(csv_path, newline='', encoding="utf-8") as csvfile:
            reader = csv.DictReader(csvfile)
            total = sum(1 for _ in reader)
            csvfile.seek(0)
            next(reader) # skip header again

            for i, row in enumerate(reader):
                nome = row["nome"]
                codigo = row["codigo"]

                status_label.config(text=f"A gerar etiqueta {i+1} de {total}...")
                janela.update_idletasks()

                zpl = f"""
^XA
^F050,50^A0N,40,40^FD{nome}^FS
^F050,120^BCN,100,Y,N,N
^FD{codigo}^FS
^F050,250^BQN,2,5
^FDLA,{codigo}^FS
^XZ
"""

                response = requests.post(labelary_url, data=zpl.strip(), headers=headers)

                if response.status_code == 200:
                    img_path = os.path.join(output_folder, f"etiqueta_{i+1}.png")
                    with open(img_path, "wb") as f:
                        f.write(response.content)
                    etiqueta_paths.append(img_path)
                    atualizar_preview(img_path)
                else:
                    messagebox.showerror("Erro", f"Erro ao gerar etiqueta {i+1}: {response.status_code}")
                    return
    except Exception as e:
        messagebox.showerror("Erro", str(e))
        return
```



```

criar_pdf(etiqueta_paths, output_folder, pdf_final)
status_label.config(text="Concluído!")
messagebox.showinfo("Concluído", f"PDF criado: {pdf_final}")

# === Criar PDF com etiquetas ===
✓ def criar_pdf(etiqueta_paths, output_folder, pdf_final):
    c = canvas.Canvas(pdf_final, pagesize=A4)
    page_width, page_height = A4
    x_margin, y_margin = 30, 30
    x_spacing, y_spacing = 20, 20
    label_width, label_height = 800, 480

    x = x_margin
    y = page_height - y_margin - label_height * 0.24

    for i, path in enumerate(etiqueta_paths):
        img = Image.open(path)
        img.thumbnail((label_width * 0.24, label_height * 0.24))
        temp_jpg = os.path.join(output_folder, f"_tmp_{i}.jpg")
        img.save(temp_jpg, "JPEG")
        c.drawImage(temp_jpg, x, y)

        x += img.width + x_spacing
        if x + img.width > page_width - x_margin:
            x = x_margin
            y -= img.height + y_spacing
        if y < y_margin:
            c.showPage()
            x = x_margin
            y = page_height - y_margin - img.height

    c.save()

# === Atualiza o painel de imagem ===
✓ def atualizar_preview(caminho_imagem):
    try:
        img = Image.open(caminho_imagem)
        img.thumbnail((250, 150))
        img_tk = ImageTk.PhotoImage(img)
        preview_label.config(image=img_tk)
        preview_label.image = img_tk
    except:
        pass

```

```
# === Interface ===

def escolher_csv():
    caminho = filedialog.askopenfilename(title="Selecionar CSV", filetypes=[("CSV", "*.csv")])
    if caminho:
        csv_entry.delete(0, tk.END)
        csv_entry.insert(0, caminho)

def escolher_pasta():
    caminho = filedialog.askdirectory(title="Selecionar pasta")
    if caminho:
        pasta_entry.delete(0, tk.END)
        pasta_entry.insert(0, caminho)

def iniciar_geracao():
    csv_path = csv_entry.get()
    output_folder = pasta_entry.get()
    tamanho = tamanho_combo.get()

    if not csv_path or not output_folder:
        messagebox.showwarning("Atenção", "Seleciona o CSV e a pasta de destino.")
        return

    gerar_etiquetas(csv_path, output_folder, tamanho)

# === GUI ===

janela = tk.Tk()
janela.title("Gerador de Etiquetas ZPL")
janela.geometry("700x400")
janela.resizable(False, False)

frame = tk.Frame(janela, padx=20, pady=10)
frame.pack(fill="x")

tk.Label(frame, text="Ficheiro CSV:").grid(row=0, column=0, sticky="w")
csv_entry = tk.Entry(frame, width=40)
csv_entry.grid(row=0, column=1, padx=5)
tk.Button(frame, text="Selecionar", command=escolher_csv).grid(row=0, column=2)

tk.Label(frame, text="Pasta de destino:").grid(row=1, column=0, sticky="w")
pasta_entry = tk.Entry(frame, width=40)
pasta_entry.grid(row=1, column=1, padx=5)
tk.Button(frame, text="Selecionar", command=escolher_pasta).grid(row=1, column=2)

tk.Label(frame, text="Tamanho da etiqueta:").grid(row=2, column=0, sticky="w")
tamanho_combo = ttk.Combobox(frame, values=["4x6", "2x1", "3x2", "4x2"], width=10)
tamanho_combo.grid(row=2, column=1, sticky="w")
tamanho_combo.set("4x6")
```

```
# Status
status_label = tk.Label(janela, text="Pronto", anchor="w")
status_label.pack(fill="x", padx=20, pady=5)

# Pré-visualização
preview_frame = tk.LabelFrame(janela, text="Pré-visualização da última etiqueta")
preview_frame.pack(padx=20, pady=10, fill="both", expand=True)
preview_label = tk.Label(preview_frame)
preview_label.pack(padx=10, pady=10)

# Botão final
tk.Button(janela, text="Gerar Etiquetas", bg="#4CAF50", fg="white", height=2, command=iniciar_geracao).pack(pady=10)

janela.mainloop()
```

vamos agora implementar o **Passo 2: Mensagens de progresso em tempo real**.

Já começámos isso no código anterior com esta linha:

```
status_label.config(text=f"A gerar etiqueta {i+1} de {total}...")
```

Vamos agora **refinar e organizar melhor o feedback visual**, incluindo:

- Mensagens mais detalhadas no status\_label
- Atualizações durante:
  - Leitura do CSV
  - Geração de cada etiqueta
  - Criação do PDF
  - Conclusão

### O que será adicionado:

1. Mensagem: A preparar leitura do CSV...
2. Mensagem: Etiqueta X de Y gerada
3. Mensagem: A criar PDF final...
4. Mensagem: PDF concluído com sucesso!

Estas mensagens manterão o utilizador informado ao longo de todo o processo.

Para tal basta substituir o conteúdos da função gerar\_etiquetas por:

```
def gerar_etiquetas(csv_path, output_folder, tamanho_etiqueta):
    dpi = "8dpm"
    rotation = "0"
    labelary_url = f"https://api.labelary.com/v1/printers/{dpi}/labels/{tamanho_etiqueta}/{rotation}/"
    headers = {'Content-Type': 'application/x-www-form-urlencoded'}

    pdf_final = os.path.join(output_folder, "etiquetas_todas.pdf")
    os.makedirs(output_folder, exist_ok=True)
    etiqueta_paths = []

    try:
        status_label.config(text="A preparar leitura do CSV...")
        janela.update_idletasks()

        with open(csv_path, newline='', encoding="utf-8") as csvfile:
            reader = csv.DictReader(csvfile)
            total = sum(1 for _ in reader)
            csvfile.seek(0)
            next(reader) # Skip header

            for i, row in enumerate(reader):
                nome = row["nome"]
                codigo = row["codigo"]

                status_label.config(text=f"A gerar etiqueta {i+1} de {total}...")
                janela.update_idletasks()

                zpl = f"""
^XA
^F050,50^A0N,40,40^FD{nome}^FS
^F050,120^BCN,100,Y,N,N
^FD{codigo}^FS
^F050,250^BQN,2,5
^FDLA,{codigo}^FS
^XZ
"""

                response = requests.post(labelary_url, data=zpl.strip(), headers=headers)

                if response.status_code == 200:
                    img_path = os.path.join(output_folder, f"etiqueta_{i+1}.png")
                    with open(img_path, "wb") as f:
                        f.write(response.content)
                    etiqueta_paths.append(img_path)
                    atualizar_preview(img_path)

                    status_label.config(text=f"Etiqueta {i+1} de {total} gerada com sucesso.")
                    janela.update_idletasks()
                else:
                    messagebox.showerror("Erro", f"Erro ao gerar etiqueta {i+1}: {response.status_code}")
                    return

            # Criação do PDF
            status_label.config(text="A criar PDF final...")
            janela.update_idletasks()
            criar_pdf(etiqueta_paths, output_folder, pdf_final)

            status_label.config(text="PDF concluído com sucesso!")
            janela.update_idletasks()
            messagebox.showinfo("Concluído", f"PDF criado: {pdf_final}")

    except Exception as e:
        status_label.config(text="Erro durante o processo.")
        messagebox.showerror("Erro", str(e))
```

Vamos agora aplicar o **Passo 3: Tema escuro (Dark Mode)** ao interface Tkinter.

### Objetivo:

Dar um aspeto mais moderno e confortável para os olhos, especialmente em ambientes com pouca luz. 😊

### Estratégia:

1. Definir **cores escuras** para o fundo, botões, caixas de texto e textos.
2. Usar uma **paleta cinza/azul escura** com bom contraste.
3. Garantir que widgets como Combobox, Entry, Label e Button seguem o mesmo tema.

⚠ Algumas limitações gráficas do ttk em Tkinter não permitem mudar todas as cores por defeito. Vamos contornar isso quando necessário.

Paleta sugerida:

Fundo janela	#2E2E2E
Texto normal	#FFFFFF
Entradas/caixas	#3C3C3C
Botões	#4CAF50
Fundo Combobox	#3C3C3C
Texto Combobox	#FFFFFF

### Como podemos aplicar?

Logo depois de criares a janela (janela = tk.Tk()), insere este bloco para definir o tema escuro:

```
# === Tema escuro ===
dark_bg = "#2E2E2E"
dark_entry = "#3C3C3C"
text_color = "#FFFFFF"
accent_color = "#4CAF50"

janela.configure(bg=dark_bg)
frame.configure(bg=dark_bg)

for widget in frame.winfo_children():
    if isinstance(widget, tk.Label):
        widget.configure(bg=dark_bg, fg=text_color)
    elif isinstance(widget, tk.Entry):
        widget.configure(bg=dark_entry, fg=text_color, insertbackground=text_color)
    elif isinstance(widget, tk.Button):
        widget.configure(bg=accent_color, fg="white", activebackground="#45A049")
    elif isinstance(widget, ttk.Combobox):
        style = ttk.Style()
        style.theme_use('clam')
        style.configure("TCombobox", fieldbackground=dark_entry, background=dark_entry, foreground=text_color)

status_label.configure(bg=dark_bg, fg=text_color)
preview_frame.configure(bg=dark_bg, fg=text_color)
preview_label.configure(bg=dark_bg)
```

Vamos agora incluir as seguintes funcionalidades:

#### Funcionalidades a incluir:

1. **Escolha detalhada do tamanho da etiqueta** (ex: largura e altura em polegadas, não apenas predefinidos como "4x6")
2. **Personalização de fontes** (tamanho e tipo ZPL)
3. **Opção de envio direto para impressora Zebra em rede (IP + porta)**

#### Plano de implementação:

##### 1. Tamanho da etiqueta personalizável

Substituir o Combobox com tamanhos fixos por dois campos:

- Largura (pol)
- Altura (pol)

##### 2. Fontes personalizáveis

Adicionar:

- Campo para **tamanho da fonte** (ex: 30, 40, 50)
- Campo para **tipo de fonte ZPL** (^A0, ^A1, etc.)

##### 3. Envio direto para impressora em rede

Adicionar:

- Campo para **IP da impressora Zebra**
- Campo para **porta** (normalmente 9100)
- Caixa de verificação: "Enviar diretamente para a impressora?"

#### Passo 1: Tamanho da etiqueta personalizável

##### O que vamos alterar:

- Remover o Combobox com opções fixas como "4x6"

- Adicionar **dois campos de entrada (Entry)** para o utilizador indicar:
  - **Largura (em polegadas)**
  - **Altura (em polegadas)**

#### Alterações no layout:

##### Substitui esta secção do código:

```
tk.Label(frame, text="Tamanho da etiqueta:").grid(row=2, column=0, sticky="w")
tamanho_combo = ttk.Combobox(frame, values=["4x6", "2x1", "3x2", "4x2"], width=10)
tamanho_combo.grid(row=2, column=1, sticky="w")
tamanho_combo.set("4x6")
```

##### Por esta:

```
tk.Label(frame, text="Largura da etiqueta (pol):").grid(row=2, column=0, sticky="w")
largura_entry = tk.Entry(frame, width=10)
largura_entry.grid(row=2, column=1, sticky="w", padx=5)
largura_entry.insert(0, "4")

tk.Label(frame, text="Altura da etiqueta (pol):").grid(row=3, column=0, sticky="w")
altura_entry = tk.Entry(frame, width=10)
altura_entry.grid(row=3, column=1, sticky="w", padx=5)
altura_entry.insert(0, "6")
```

#### Alteração na função *iniciar\_geracao*

##### Substitui:

```
tamanho = tamanho_combo.get()
```

##### Por:

```
largura = largura_entry.get()
altura = altura_entry.get()
tamanho = f"{largura}x{altura}"
```

## Passo 2: Personalização de fontes ZPL

#### O que vamos adicionar:

- Campo para o utilizador escolher o **tipo de fonte ZPL** (^A0, ^A1, etc.)
- Campo para definir o **tamanho da fonte** (largura e altura dos caracteres)



## Atualizações no layout

Adiciona esta secção ao layout logo abaixo da altura da etiqueta:

```
tk.Label(frame, text="Fonte ZPL (ex: A0, A1):").grid(row=4, column=0, sticky="w")
fonte_entry = tk.Entry(frame, width=10)
fonte_entry.grid(row=4, column=1, sticky="w", padx=5)
fonte_entry.insert(0, "A0")

tk.Label(frame, text="Tamanho da fonte (larg x alt):").grid(row=5, column=0, sticky="w")
fonte_l_entry = tk.Entry(frame, width=5)
fonte_l_entry.grid(row=5, column=1, sticky="w", padx=(0, 2))
fonte_l_entry.insert(0, "40")

fonte_h_entry = tk.Entry(frame, width=5)
fonte_h_entry.grid(row=5, column=1, sticky="e", padx=(2, 0))
fonte_h_entry.insert(0, "40")
```

## Alterações no código ZPL dentro de gerar\_etiquetas

Substitui esta linha do ZPL:

```
^F050,50^A0N,40,40^FD{nome}^FS
```

Por este novo bloco:

```
^F050,50^{fonte}N,{fonte_h},{fonte_l}^FD{nome}^FS
```

E no início da função gerar\_etiquetas, recolhe estes valores dos Entry:

```
fonte = fonte_entry.get()
fonte_l = fonte_l_entry.get()
fonte_h = fonte_h_entry.get()
```

(Nota: certifica-te de que estão acessíveis como variáveis globais ou passam como argumento, se estiveres a modularizar.)



## Resultado:

O utilizador pode agora escolher:

- Fonte ZPL (A0, A1, etc.)
- Largura e altura da fonte (ex: 40 x 40)

## Passo 3: Envio direto para impressora Zebra em rede?

### O que vamos adicionar:

1. Campo para inserir o **endereço IP da impressora Zebra**
2. Campo para definir a **porta** (por padrão, é 9100)
3. Caixa de seleção (Checkbox) para ativar/desativar o envio direto à impressora

## Atualizações no layout

Adiciona o seguinte bloco abaixo dos campos de fonte:

```
tk.Label(frame, text="IP da impressora Zebra:").grid(row=6, column=0, sticky="w")
ip_entry = tk.Entry(frame, width=15)
ip_entry.grid(row=6, column=1, sticky="w", padx=5)
ip_entry.insert(0, "192.168.1.100")

tk.Label(frame, text="Porta:").grid(row=7, column=0, sticky="w")
porta_entry = tk.Entry(frame, width=6)
porta_entry.grid(row=7, column=1, sticky="w", padx=5)
porta_entry.insert(0, "9100")

enviar_check = tk.IntVar()
tk.Checkbutton(frame, text="Enviar diretamente para a impressora", variable=enviar_check, bg=frame["bg"], fg="white",
               selectcolor=frame["bg"]).grid(row=8, column=0, columnspan=2, sticky="w", pady=(5, 10))
```

## Lógica de envio direto para a impressora Zebra

Na função gerar\_etiquetas, depois de gerar o zpl, insere este bloco **em substituição ou paralelo ao envio para Labelary**:

```
# Se envio direto estiver ativado
if enviar_check.get() == 1:
    try:
        import socket
        ip = ip_entry.get()
        porta = int(porta_entry.get())

        status_label.config(text=f"A enviar etiqueta {i+1} para {ip}:{porta}...")
        janela.update_idletasks()

        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as zebra_socket:
            zebra_socket.connect((ip, porta))
            zebra_socket.sendall(zpl.encode('utf-8'))

        status_label.config(text=f"Etiqueta {i+1} enviada com sucesso.")
    except Exception as e:
        messagebox.showerror("Erro de envio", f"Falha ao enviar etiqueta para a impressora: {e}")
        return
else:
    # Enviar para Labelary (continua como antes)
    response = requests.post(labelary_url, data=zpl.strip(), headers=headers)

    if response.status_code == 200:
        img_path = os.path.join(output_folder, f"etiqueta_{i+1}.png")
        with open(img_path, "wb") as f:
            f.write(response.content)
        etiqueta_paths.append(img_path)
        atualizar_preview(img_path)
    else:
        messagebox.showerror("Erro", f"Erro ao gerar etiqueta {i+1}: {response.status_code}")
        return
```

## Resultado:

O utilizador agora pode:

- Introduzir o IP e porta da impressora Zebra
- Marcar a caixa “Enviar diretamente”
- O script envia a etiqueta por **socket TCP/IP direto**, sem Labelary

Apresenta-se de seguida o código total:

```
import tkinter as tk
from tkinter import filedialog, messagebox, ttk
from PIL import Image, ImageTk
import csv
import requests
import socket
import os
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import A4

# === Função para atualizar imagem de pré-visualização ===
def atualizar_preview(caminho_imagem):
    try:
        img = Image.open(caminho_imagem)
        img.thumbnail((250, 150))
        img_tk = ImageTk.PhotoImage(img)
        preview_label.config(image=img_tk)
        preview_label.image = img_tk
    except:
        pass

# === Função principal ===
def gerar_etiquetas(csv_path, output_folder, largura, altura, fonte, fonte_l, fonte_h):
    dpi = "8dpmm"
    rotation = "0"
    tamanho_etiqueta = f"{largura}x{altura}"
    labelary_url = f"https://api.labelary.com/v1/printers/{dpi}/labels/{tamanho_etiqueta}/{rotation}/"
    headers = {'Content-Type': 'application/x-www-form-urlencoded'}

    pdf_final = os.path.join(output_folder, "etiquetas_todas.pdf")
    os.makedirs(output_folder, exist_ok=True)
    etiqueta_paths = []

    try:
        status_label.config(text="A preparar leitura do CSV...")
        janela.update_idletasks()

        with open(csv_path, newline='', encoding="utf-8") as csvfile:
            reader = csv.DictReader(csvfile)
            total = sum(1 for _ in reader)
            csvfile.seek(0)
            next(reader)
```

```

for i, row in enumerate(reader):
    nome = row["nome"]
    codigo = row["codigo"]

    status_label.config(text=f"A gerar etiqueta {i+1} de {total}...")
    janela.update_idletasks()

    zpl = f"""
^XA
^F050,50^{fonte}N,{fonte_h},{fonte_l}^FD{nome}^FS
^F050,120^BCN,100,Y,N,N
^FD{codigo}^FS
^F050,250^BQN,2,5
^FDLA,{codigo}^FS
^XZ
"""

    if enviar_check.get() == 1:
        try:
            ip = ip_entry.get()
            porta = int(porta_entry.get())

            status_label.config(text=f"A enviar etiqueta {i+1} para {ip}:{porta}...")
            janela.update_idletasks()

            with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as zebra_socket:
                zebra_socket.connect((ip, porta))
                zebra_socket.sendall(zpl.encode('utf-8'))

            status_label.config(text=f"Etiqueta {i+1} enviada com sucesso.")
        except Exception as e:
            messagebox.showerror("Erro de envio", f"Falha ao enviar etiqueta: {e}")
            return

    else:
        response = requests.post(labelary_url, data=zpl.strip(), headers=headers)

        if response.status_code == 200:
            img_path = os.path.join(output_folder, f"etiqueta_{i+1}.png")
            with open(img_path, "wb") as f:
                f.write(response.content)
            etiqueta_paths.append(img_path)
            atualizar_preview(img_path)
        else:
            messagebox.showerror("Erro", f"Erro ao gerar etiqueta {i+1}: {response.status_code}")
            return

    status_label.config(text="A criar PDF final...")
    janela.update_idletasks()
    criar_pdf(etiqueta_paths, output_folder, pdf_final)

```

```
status_label.config(text="A criar PDF final...")
janela.update_idletasks()
criar_pdf(etiqueta_paths, output_folder, pdf_final)

status_label.config(text="PDF concluído com sucesso!")
messagebox.showinfo("Concluído", f"PDF criado: {pdf_final}")
```

```
✓ except Exception as e:
    status_label.config(text="Erro durante o processo.")
    messagebox.showerror("Erro", str(e))

# === Cria o PDF ===
✓ def criar_pdf(etiqueta_paths, output_folder, pdf_final):
    c = canvas.Canvas(pdf_final, pagesize=A4)
    page_width, page_height = A4
    x_margin, y_margin = 30, 30
    x_spacing, y_spacing = 20, 20
    label_width, label_height = 800, 480

    x = x_margin
    y = page_height - y_margin - label_height * 0.24

    for i, path in enumerate(etiqueta_paths):
        img = Image.open(path)
        img.thumbnail((label_width * 0.24, label_height * 0.24))
        temp_jpg = os.path.join(output_folder, f"_tmp_{i}.jpg")
        img.save(temp_jpg, "JPEG")
        c.drawImage(temp_jpg, x, y)

        x += img.width + x_spacing
        ✓ if x + img.width > page_width - x_margin:
            x = x_margin
            y -= img.height + y_spacing
        ✓ if y < y_margin:
            c.showPage()
            x = x_margin
            y = page_height - y_margin - img.height

    c.save()
```

```
# === Interface ===
def escolher_csv():
    caminho = filedialog.askopenfilename(title="Selecionar CSV", filetypes=[("CSV", "*.csv")])
    if caminho:
        csv_entry.delete(0, tk.END)
        csv_entry.insert(0, caminho)

def escolher_pasta():
    caminho = filedialog.askdirectory(title="Selecionar pasta")
    if caminho:
        pasta_entry.delete(0, tk.END)
        pasta_entry.insert(0, caminho)

def iniciar_geracao():
    csv_path = csv_entry.get()
    output_folder = pasta_entry.get()
    largura = largura_entry.get()
    altura = altura_entry.get()
    fonte = fonte_entry.get()
    fonte_l = fonte_l_entry.get()
    fonte_h = fonte_h_entry.get()

    if not csv_path or not output_folder:
        messagebox.showwarning("Atenção", "Seleciona o CSV e a pasta de destino.")
        return

    gerar_etiquetas(csv_path, output_folder, largura, altura, fonte, fonte_l, fonte_h)

# === GUI ===
janela = tk.Tk()
janela.title("Gerador de Etiquetas ZPL")
janela.geometry("750x500")
janela.configure(bg="#2E2E2E")

frame = tk.Frame(janela, padx=20, pady=10, bg="#2E2E2E")
frame.pack(fill="x")

# Campos CSV e pasta
tk.Label(frame, text="Ficheiro CSV:", bg="#2E2E2E", fg="white").grid(row=0, column=0, sticky="w")
csv_entry = tk.Entry(frame, width=40, bg="#3C3C3C", fg="white", insertbackground="white")
csv_entry.grid(row=0, column=1, padx=5)
tk.Button(frame, text="Selecionar", command=escolher_csv, bg="#4CAF50", fg="white").grid(row=0, column=2)

tk.Label(frame, text="Pasta de destino:", bg="#2E2E2E", fg="white").grid(row=1, column=0, sticky="w")
pasta_entry = tk.Entry(frame, width=40, bg="#3C3C3C", fg="white", insertbackground="white")
pasta_entry.grid(row=1, column=1, padx=5)
tk.Button(frame, text="Selecionar", command=escolher_pasta, bg="#4CAF50", fg="white").grid(row=1, column=2)

# Tamanho
tk.Label(frame, text="Largura (pol):", bg="#2E2E2E", fg="white").grid(row=2, column=0, sticky="w")
largura_entry = tk.Entry(frame, width=10, bg="#3C3C3C", fg="white")
largura_entry.grid(row=2, column=1, sticky="w", padx=5)
largura_entry.insert(0, "4")
```

```
tk.Label(frame, text="Altura (pol):", bg="#2E2E2E", fg="white").grid(row=3, column=0, sticky="w")
altura_entry = tk.Entry(frame, width=10, bg="#3C3C3C", fg="white")
altura_entry.grid(row=3, column=1, sticky="w", padx=5)
altura_entry.insert(0, "6")

# Fonte ZPL
tk.Label(frame, text="Fonte ZPL (ex: A0):", bg="#2E2E2E", fg="white").grid(row=4, column=0, sticky="w")
fonte_entry = tk.Entry(frame, width=10, bg="#3C3C3C", fg="white")
fonte_entry.grid(row=4, column=1, sticky="w", padx=5)
fonte_entry.insert(0, "A0")

tk.Label(frame, text="Tamanho da fonte (LxA):", bg="#2E2E2E", fg="white").grid(row=5, column=0, sticky="w")
fonte_l_entry = tk.Entry(frame, width=5, bg="#3C3C3C", fg="white")
fonte_l_entry.grid(row=5, column=1, sticky="w", padx=(0, 2))
fonte_l_entry.insert(0, "40")
fonte_h_entry = tk.Entry(frame, width=5, bg="#3C3C3C", fg="white")
fonte_h_entry.grid(row=5, column=1, sticky="e", padx=(2, 0))
fonte_h_entry.insert(0, "40")

# Impressora Zebra

tk.Label(frame, text="IP da impressora:", bg="#2E2E2E", fg="white").grid(row=6, column=0, sticky="w")
ip_entry = tk.Entry(frame, width=15, bg="#3C3C3C", fg="white")
ip_entry.grid(row=6, column=1, sticky="w", padx=5)
ip_entry.insert(0, "192.168.1.100")

tk.Label(frame, text="Porta:", bg="#2E2E2E", fg="white").grid(row=7, column=0, sticky="w")
porta_entry = tk.Entry(frame, width=6, bg="#3C3C3C", fg="white")
porta_entry.grid(row=7, column=1, sticky="w", padx=5)
porta_entry.insert(0, "9100")

enviar_check = tk.IntVar()
tk.Checkbutton(frame, text="Enviar diretamente para a impressora", variable=enviar_check,
               bg="#2E2E2E", fg="white", selectcolor="#2E2E2E").grid(row=8, column=0, columnspan=2, sticky="w")

# Status e preview
status_label = tk.Label(janela, text="Pronto", anchor="w", bg="#2E2E2E", fg="white")
status_label.pack(fill="x", padx=20, pady=5)

preview_frame = tk.LabelFrame(janela, text="Pré-visualização da última etiqueta", bg="#2E2E2E", fg="white")
preview_frame.pack(padx=20, pady=10, fill="both", expand=True)
preview_label = tk.Label(preview_frame, bg="#2E2E2E")
preview_label.pack(padx=10, pady=10)

# Botão principal
tk.Button(janela, text="Gerar Etiquetas", bg="#4CAF50", fg="white", height=2, command=iniciar_geracao).pack(pady=10)

janela.mainloop()
```

Na pasta correspondente a este enunciado encontra-se a script

gerador\_etiquetas.py com este código. A aplicação ficou com o seguinte aspeto:



Abaixo descreve-se como podes gerar o executável.

Vamos usar o **PyInstaller**, que converte scripts Python em executáveis standalone.

Para tal instala a seguinte biblioteca

```
pip install pyinstaller
```

Abre o terminal (CMD ou PowerShell). Vai à pasta onde tens o `gerador_etiquetas.py`.

Exemplo:

```
cd C:\Users\Joao\Desktop
```



Cria o ficheiro `.exe` utilizando

```
pyinstaller --onefile --windowed gerador_etiquetas.py
```

- `--onefile` : cria um único `.exe`
- `--windowed` : evita que o terminal apareça ao abrir a aplicação (útil para apps gráficas)

Ir à pasta `dist`

Após a execução, o executável estará em:

```
dist/gerador_etiquetas.exe
```

Para testes:

Corre o `.exe` em qualquer computador com Windows – **não é necessário instalar Python.**

Cria agora um ficheiro `.ico`

Tendo o ícone gera o `.exe` com o mesmo usando o seguinte comando

```
pyinstaller --onefile --windowed --icon=icone.ico gerador_etiquetas.py
```

Ou seja,

### 1. Coloca os ficheiros na mesma pasta:

- `gerador_etiquetas.py`
- `icone.ico`

### 2. Abre o terminal (CMD ou PowerShell) nessa pasta

### 3. Executa este comando:

```
bash
```

[Copy](#)[Edit](#)

```
pyinstaller --onefile --windowed --icon=icone.ico gerador_etiquetas.py
```

### 4. Resultado:

O executável `.exe` estará dentro da pasta `dist`, com o ícone personalizado aplicado.

A título opcional podemos criar um instalador profissional 😊

## ✓ Pré-requisitos

#### 1. Instalar o Inno Setup Compiler

- Site oficial: <https://jrsoftware.org/isdl.php>
- Descarrega e instala a versão estável

## 📁 Estrutura da pasta (antes de compilar)

Cria uma pasta chamada `GeradorEtiquetasApp` com esta estrutura:

```
GeradorEtiquetasApp/
```

```
├─ gerador_etiquetas.exe ← gerado com PyInstaller
```

```
├─ icone.ico ← ícone da aplicação
```

## Criar o script .iss (Inno Setup Script)

Guarda o seguinte conteúdo como `gerador_etiquetas_instalador.iss`:

```
ini
[Setup]
AppName=Gerador de Etiquetas ZPL
AppVersion=1.0
DefaultDirName={pf}\\GeradorEtiquetasZPL
DefaultGroupName=Gerador de Etiquetas ZPL
AllowNoIcons=yes
OutputBaseFilename=Instalador_Gerador_Etiquetas
SetupIconFile=icone.ico
Compression=lzma
SolidCompression=yes

[Languages]
Name: "portuguese"; MessagesFile: "compiler:Languages\\Portuguese.isl"

[Files]
Source: "gerador_etiquetas.exe"; DestDir: "{app}"; Flags: ignoreversion
Source: "icone.ico"; DestDir: "{app}"; Flags: ignoreversion


[Icons]
Name: "{group}\\Gerador de Etiquetas ZPL"; Filename: "{app}\\gerador_etiquetas.exe"
Name: "{userdesktop}\\Gerador de Etiquetas ZPL"; Filename: "{app}\\gerador_etiquetas.exe"; Ta

[Tasks]
Name: "desktopicon"; Description: "Criar atalho no ambiente de trabalho"; GroupDescription: "C
```

## Compilar o instalador

1. Abre o Inno Setup Compiler
2. Vai a **File > Open**, e abre o `gerador_etiquetas_instalador.iss`
3. Clica em **Compile (F9)**
4. O ficheiro final `.exe` estará na pasta do script, com o nome:

Instalador\_Gerador\_Etiquetas.exe

 Copy  Edit

## Resultado

- Instalação em `C:\Program Files\GeradorEtiquetasZPL`
- Atalho no menu iniciar e no ambiente de trabalho (opcional)
- Ícone personalizado incluído

Na pasta associada a este enunciado encontra-se o ficheiro `.iss`. Como podes utilizar?

1. Gera o `gerador_etiquetas.exe` com PyInstaller

```
bash Copy Edit  
  
pyinstaller --onefile --windowed --icon=icone.ico gerador_etiquetas.py
```

2. Copia para uma pasta com o nome:

```
Copy Edit  
  
GeradorEtiquetasApp/  
├─ gerador_etiquetas.exe  
└─ icone.ico
```

3. Abre o `.iss` no Inno Setup

- Vai a File > Open
- Abre o ficheiro `gerador_etiquetas_instalador.iss`

4. Clica em Compile (ou F9)

O Inno Setup vai gerar um instalador `.exe` com:

- Nome: `Instalador_Gerador_Etiquetas.exe`
- Assistente de instalação em português
- Atalho no ambiente de trabalho (opcional)
- Ícone da aplicação incluído

**Podes também gerar um instalador empresarial:**

## Opção 1: Pacote .msi (instalador empresarial)

Um .msi é útil para:

- Distribuição em ambientes corporativos
- Instalação silenciosa por administradores
- Políticas de rede e domínio (GPO)

### Ferramenta recomendada: WiX Toolset

Passos:

#### 1. Instalar o WiX Toolset

- Site: <https://wixtoolset.org/releases/>
- Instala também o WiX Toolset Visual Studio Extension (se usas VS)

#### 2. Criar ficheiro WiX .WXS com este conteúdo:

```

xml
Copy Edit

<?xml version="1.0" encoding="UTF-8"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
  <Product Id="*" Name="Gerador de Etiquetas ZPL" Language="1046" Version="1.0.0.0" Manufact
    <Package InstallerVersion="500" Compressed="yes" InstallScope="perMachine" />

    <MajorUpgrade DowngradeErrorMessage="Já está instalada uma versão mais recente." />
    <MediaTemplate />

    <Feature Id="ProductFeature" Title="Gerador de Etiquetas ZPL" Level="1">
      <ComponentGroupRef Id="AppComponents" />
    </Feature>
  </Product>

  <Fragment>
    <Directory Id="TARGETDIR" Name="SourceDir">
      <Directory Id="ProgramFilesFolder">
        <Directory Id="INSTALLFOLDER" Name="GeradorEtiquetasZPL" />
      </Directory>
    </Directory>
  </Fragment>

  <Fragment>
    <ComponentGroup Id="AppComponents" Directory="INSTALLFOLDER">
      <Component Id="MainExecutable" Guid="*">
        <File Source="gerador_etiquetas.exe" KeyPath="yes" />
      </Component>
      <Component Id="Icon" Guid="*">
        <File Source="icone.ico" />
      </Component>
    </ComponentGroup>
  </Fragment>
</Wix>

```

### 3. Compilar com o WiX (via terminal ou Visual Studio)

```
bash

candle setup.wxs
light setup.wixobj -o GeradorEtiquetas.msi
```

[Copy](#)[Edit](#)

### ✓ Opção 2: Pacote portátil .zip

Ideal para:

- Executar sem instalar
- Distribuir via email ou pen drive

### Como criar:

1. Cria uma pasta chamada GeradorEtiquetas\_Portatil
2. Coloca dentro:
  - gerador\_etiquetas.exe
  - icone.ico
3. Comprime a pasta como .zip

## Prática

1. Instala a biblioteca pywin32 com o comando `pip install pywin32`
2. Reproduz o seguinte exercício. O mesmo usado permite imprimir etiquetas em impressoras Zebra.

```
import win32print
import win32api

printer_name = "Zebra" # Substituir pelo nome correto da impressora
zpl_command = "^XA^F050,50^ADN,36,20^FDZPL Test^FS^XZ"

# Abre a impressora
printer_handle = win32print.OpenPrinter(printer_name)
try:
    job = win32print.StartDocPrinter(printer_handle, 1, ("ZPL Print Job", None, "RAW"))
    win32print.StartPagePrinter(printer_handle)
    win32print.WritePrinter(printer_handle, zpl_command.encode('utf-8'))
    win32print.EndPagePrinter(printer_handle)
    win32print.EndDocPrinter(printer_handle)
finally:
    win32print.ClosePrinter(printer_handle)
```

3. Em vez de indicares diretamente o código altera o mesmo para listar as impressoras existentes na rede destacando a predefinida.

```
# Importa bibliotecas do Windows necessárias para aceder às impressoras
import win32print
import win32api

# Função para listar todas as impressoras instaladas no sistema
def listar_impressoras():
    printers = win32print.EnumPrinters(win32print.PRINTER_ENUM_LOCAL | win32print.PRINTER_ENUM_CONNECTIONS)

    print("🖨 Impressoras disponíveis no sistema:\n")
    for flags, description, name, comment in printers:
        print(f"- {name}")

    default = win32print.GetDefaultPrinter()
    print(f"\n✅ Impressora predefinida: {default}")
    return [name for flags, description, name, comment in printers]
```



```
# Função para enviar um comando ZPL para uma impressora específica
def enviar_zpl(printer_name, zpl_command):
    try:
        # Abre a impressora
        printer_handle = win32print.OpenPrinter(printer_name)

        # Inicia o "documento" de impressão (raw, sem drivers gráficos)
        job = win32print.StartDocPrinter(printer_handle, 1, ("ZPL Print Job", None, "RAW"))
        win32print.StartPagePrinter(printer_handle)

        # Envia os dados (em bytes) para a impressora
        win32print.WritePrinter(printer_handle, zpl_command.encode('utf-8'))

        # Finaliza a página e o trabalho
        win32print.EndPagePrinter(printer_handle)
        win32print.EndDocPrinter(printer_handle)
        print("Comando ZPL enviado com sucesso!")

    except Exception as e:
        print(f"Erro ao imprimir: {e}")

    finally:
        # Fecha a ligação à impressora
        win32print.ClosePrinter(printer_handle)

# Função principal
def main():
    impressoras = listar_impressoras()

    # Pede ao utilizador para escrever o nome da impressora desejada
    printer_name = input("\nIntroduz o nome exato da impressora a usar: ").strip()

    if printer_name not in impressoras:
        print("Impressora não encontrada na lista. Verifica o nome e tenta novamente.")
        return

    # Comando ZPL de teste (podes personalizar)
    zpl = "^XA^F050,50^ADN,36,20^FDZPL Test^FS^XZ"

    # Envia o comando para a impressora
    enviar_zpl(printer_name, zpl)

# Início do script
if __name__ == "__main__":
    main()
```

#### 4. Vamos agora alterar a script para:

- i. Gerar etiquetas com base em dados externos (como CSV, base de dados).
- ii. Criar QR Codes ou códigos de barras em ZPL.

iii. Gerar múltiplas etiquetas numa só impressão.

Cria um ficheiro **dados.csv** com a seguinte informação:

```
Nome,Codigo
João,12345
Maria,67890
```

O código em python para resolver os pontos anteriores são (reproduz cada uma delas):

### Script 1: Geração de etiquetas a partir de CSV e envio para impressora

```
import csv
import win32print

# Função para enviar ZPL para a impressora
def enviar_zpl(printer_name, zpl_command):
    try:
        printer_handle = win32print.OpenPrinter(printer_name)
        job = win32print.StartDocPrinter(printer_handle, 1, ("ZPL CSV Job", None, "RAW"))
        win32print.StartPagePrinter(printer_handle)
        win32print.WritePrinter(printer_handle, zpl_command.encode('utf-8'))
        win32print.EndPagePrinter(printer_handle)
        win32print.EndDocPrinter(printer_handle)
        print("Etiquetas enviadas com sucesso.")
    except Exception as e:
        print(f"Erro: {e}")
    finally:
        win32print.ClosePrinter(printer_handle)

# Gera ZPL com base num ficheiro CSV
def gerar_zpl_de_csv(caminho_csv):
    with open(caminho_csv, newline='', encoding='utf-8') as ficheiro:
        leitor = csv.DictReader(ficheiro)
        etiquetas = []

        for linha in leitor:
            nome = linha['Nome']
            codigo = linha['Codigo']
            zpl = f"XA^F050,50^ADN,36,20^FDNome: {nome}^FS^F050,100^FDCódigo: {codigo}^FS^XZ"
            etiquetas.append(zpl)

        return ''.join(etiquetas)

# Script principal
def main():
    caminho_csv = "dados.csv"
    printer_name = input("Nome da impressora: ").strip()
    zpl = gerar_zpl_de_csv(caminho_csv)
    enviar_zpl(printer_name, zpl)

if __name__ == "__main__":
    main()
```

## Script 2: Geração de etiqueta com QR Code e Código de Barras

```
import win32print

def enviar_zpl(printer_name, zpl_command):
    try:
        printer_handle = win32print.OpenPrinter(printer_name)
        job = win32print.StartDocPrinter(printer_handle, 1, ("ZPL Códigos", None, "RAW"))
        win32print.StartPagePrinter(printer_handle)
        win32print.WritePrinter(printer_handle, zpl_command.encode('utf-8'))
        win32print.EndPagePrinter(printer_handle)
        win32print.EndDocPrinter(printer_handle)
        print("Etiqueta com códigos enviada com sucesso.")
    except Exception as e:
        print(f"Erro: {e}")
    finally:
        win32print.ClosePrinter(printer_handle)

def gerar_zpl_com_codigos(dado):
    zpl = f"""
^XA
^F050,50^BQN,2,6^FDLA,{dado}^FS
^F050,200^BCN,100,Y,N,N^FD{dado}^FS
^XZ
"""
    return zpl

def main():
    dado = input("Conteúdo para QR/Code128: ").strip()
    printer_name = input("Nome da impressora: ").strip()
    zpl = gerar_zpl_com_codigos(dado)
    enviar_zpl(printer_name, zpl)

if __name__ == "__main__":
    main()
```

## Script 3: Impressão de múltiplas etiquetas numa só impressão

```
import win32print

def enviar_zpl(printer_name, zpl_command):
    try:
        printer_handle = win32print.OpenPrinter(printer_name)
        job = win32print.StartDocPrinter(printer_handle, 1, ("ZPL Várias", None, "RAW"))
        win32print.StartPagePrinter(printer_handle)
        win32print.WritePrinter(printer_handle, zpl_command.encode('utf-8'))
        win32print.EndPagePrinter(printer_handle)
        win32print.EndDocPrinter(printer_handle)
        print("Etiquetas enviadas com sucesso.")
    except Exception as e:
        print(f"Erro: {e}")
    finally:
        win32print.ClosePrinter(printer_handle)

def juntar_etiquetas_zpl(lista_zpl):
    return ''.join(lista_zpl)

def main():
    printer_name = input("Nome da impressora: ").strip()

    etiquetas = [
        "^XA^F050,50^FDProduto A^FS^XZ",
        "^XA^F050,50^FDProduto B^FS^XZ",
        "^XA^F050,50^FDProduto C^FS^XZ"
    ]

    zpl_unificado = juntar_etiquetas_zpl(etiquetas)
    enviar_zpl(printer_name, zpl_unificado)

if __name__ == "__main__":
    main()
```

## Script Global: etiquetas\_zebra.py com menu interativo no terminal

```
import csv
import win32print

# Envia comando ZPL para a impressora
def enviar_zpl(printer_name, zpl_command):
    try:
        printer_handle = win32print.OpenPrinter(printer_name)
        job = win32print.StartDocPrinter(printer_handle, 1, ("ZPL Job", None, "RAW"))
        win32print.StartPagePrinter(printer_handle)
        win32print.WritePrinter(printer_handle, zpl_command.encode('utf-8'))
        win32print.EndPagePrinter(printer_handle)
        win32print.EndDocPrinter(printer_handle)
        print("\nEtiqueta(s) enviada(s) com sucesso.")
    except Exception as e:
        print(f"\nErro ao imprimir: {e}")
    finally:
        win32print.ClosePrinter(printer_handle)
```

```
# Opção 1: Etiquetas a partir de CSV
def gerar_zpl_de_csv(caminho_csv):
    try:
        with open(caminho_csv, newline='', encoding='utf-8') as ficheiro:
            leitor = csv.DictReader(ficheiro)
            etiquetas = []

            for linha in leitor:
                nome = linha['Nome']
                codigo = linha['Codigo']
                zpl = f"^XA^F050,50^ADN,36,20^FDNome: {nome}^FS^F050,100^FDCódigo: {codigo}^FS^XZ"
                etiquetas.append(zpl)

            return ''.join(etiquetas)
    except FileNotFoundError:
        print(f"\nFicheiro '{caminho_csv}' não encontrado.")
        return ""

# Opção 2: Etiqueta com QR Code + Código de Barras
def gerar_zpl_com_codigos(dado):
    zpl = f"""
^XA
^F050,50^BQN,2,6^FDLA,{dado}^FS
^F050,200^BCN,100,Y,N,N^FD{dado}^FS
^XZ
"""
    return zpl

# Opção 3: Várias etiquetas simples de uma vez
def juntar_etiquetas_zpl(lista_zpl):
    return ''.join(lista_zpl)

# Menu principal
def main():
    print("=== Impressão de Etiquetas Zebra ===")
    print("1. Imprimir etiquetas a partir de ficheiro CSV")
    print("2. Imprimir etiqueta com QR Code e Código de Barras")
    print("3. Imprimir várias etiquetas simples")
    escolha = input("Escolhe uma opção (1, 2 ou 3): ").strip()

    printer_name = input("Nome da impressora: ").strip()

    if escolha == '1':
        caminho_csv = input("Caminho para o ficheiro CSV (ex: dados.csv): ").strip()
        zpl = gerar_zpl_de_csv(caminho_csv)
        if zpl:
            enviar_zpl(printer_name, zpl)

    elif escolha == '2':
        dado = input("Conteúdo para o QR Code e código de barras: ").strip()
        zpl = gerar_zpl_com_codigos(dado)
        enviar_zpl(printer_name, zpl)
```

```
elif escolha == '3':
    etiquetas = [
        "^XA^F050,50^FDProduto A^FS^XZ",
        "^XA^F050,50^FDProduto B^FS^XZ",
        "^XA^F050,50^FDProduto C^FS^XZ"
    ]
    zpl = juntar_etiquetas_zpl(etiquetas)
    enviar_zpl(printer_name, zpl)

else:
    print("Opção inválida.")

if __name__ == "__main__":
    main()
```

5. Vamos transformar o script com menu num **pequeno programa com interface gráfica em**

**Tkinter**, mantendo as três funcionalidades:

- i. Geração de etiquetas a partir de CSV
- ii. Criação de etiqueta com QR Code e Código de Barras
- iii. Impressão de múltiplas etiquetas

```
import tkinter as tk
from tkinter import filedialog, messagebox
import csv
import win32print

# Função para enviar ZPL para a impressora
def enviar_zpl(printer_name, zpl_command):
    try:
        printer_handle = win32print.OpenPrinter(printer_name)
        job = win32print.StartDocPrinter(printer_handle, 1, ("ZPL Job", None, "RAW"))
        win32print.StartPagePrinter(printer_handle)
        win32print.WritePrinter(printer_handle, zpl_command.encode('utf-8'))
        win32print.EndPagePrinter(printer_handle)
        win32print.EndDocPrinter(printer_handle)
        win32print.ClosePrinter(printer_handle)
        messagebox.showinfo("Sucesso", "Etiqueta(s) enviada(s) com sucesso.")
    except Exception as e:
        messagebox.showerror("Erro", f"Erro ao imprimir: {e}")

# Opção 1: CSV para ZPL
def gerar_zpl_de_csv(caminho_csv):
    try:
        with open(caminho_csv, newline='', encoding='utf-8') as ficheiro:
            leitor = csv.DictReader(ficheiro)
            etiquetas = []
            for linha in leitor:
                nome = linha['Nome']
                codigo = linha['Codigo']
                zpl = f"^XA^F050,50^ADN,36,20^FDNome: {nome}^FS^F050,100^FDCódigo: {codigo}^FS^XZ"
                etiquetas.append(zpl)
            return ''.join(etiquetas)
    except Exception as e:
        messagebox.showerror("Erro", f"Erro ao ler CSV: {e}")
        return ""
```

```

# Opção 2: QR + Código de Barras
def gerar_zpl_com_codigos(dado):
    return f"""
^XA
^F050,50^BQN,2,6^FDLA,{dado}^FS
^F050,200^BCN,100,Y,N,N^FD{dado}^FS
^XZ
"""

# Opção 3: Etiquetas múltiplas simples
def gerar_multiplas_etiquetas():
    etiquetas = [
        "^XA^F050,50^FDProduto A^FS^XZ",
        "^XA^F050,50^FDProduto B^FS^XZ",
        "^XA^F050,50^FDProduto C^FS^XZ"
    ]
    return ''.join(etiquetas)

# Ações ligadas aos botões da interface
def acao_csv():
    caminho = filedialog.askopenfilename(filetypes=[("Ficheiros CSV", "*.csv")])
    if caminho:
        zpl = gerar_zpl_de_csv(caminho)
        if zpl:
            enviar_zpl(printer_entry.get(), zpl)

def acao_codigos():
    dado = input_entry.get()
    if dado:
        zpl = gerar_zpl_com_codigos(dado)
        enviar_zpl(printer_entry.get(), zpl)
    else:
        messagebox.showwarning("Atenção", "Preenche o campo de conteúdo.")

def acao_multiplas():
    zpl = gerar_multiplas_etiquetas()
    enviar_zpl(printer_entry.get(), zpl)

# Interface gráfica
root = tk.Tk()
root.title("Impressão de Etiquetas Zebra")
root.geometry("500x300")
root.resizable(False, False)

```



```
# Campo para nome da impressora
tk.Label(root, text="Nome da Impressora:").pack(pady=5)
printer_entry = tk.Entry(root, width=50)
printer_entry.pack()
printer_entry.insert(0, win32print.GetDefaultPrinter())

# Campo para conteúdo do QR/Código de barras
tk.Label(root, text="Conteúdo (QR/Código de Barras):").pack(pady=5)
input_entry = tk.Entry(root, width=50)
input_entry.pack()

# Botões de ação
tk.Button(root, text="1. Imprimir etiquetas de ficheiro CSV", command=acao_csv, width=40).pack(pady=10)
tk.Button(root, text="2. Imprimir etiqueta com QR + Código", command=acao_codigos, width=40).pack(pady=5)
tk.Button(root, text="3. Imprimir múltiplas etiquetas simples", command=acao_multiplas, width=40).pack(pady=5)

# Iniciar aplicação
root.mainloop()
```

Guarda o código num ficheiro chamdo etiquetas\_zebra\_gui.py. Testa o programa.

6. Vamos agora modificar o código anterior para elaborarmos uma **interface gráfica mais avançada em Tkinter**, com:

- **Botões para escolher ficheiros CSV e pasta de destino**
- **Configuração de impressora (IP, Porta)**
- **Campo para ajustar tamanho da etiqueta**
- **Pré-visualização da etiqueta gerada** (como na imagem)

Para tal vamos efetuar os seguintes passos:

### Adição de um Widget de Pré-visualização

- Agora, ao gerar etiquetas (CSV, QR Code, múltiplas), a imagem da etiqueta aparece antes da impressão.
- Para isso, foi criada a função `previsualizar_zpl()` que usa a API **Labelary** para converter ZPL em imagem.

### Separação da Geração e Impressão

- Antes, ao clicar num botão, o programa imprimia diretamente.
- Agora, primeiro mostra a **pré-visualização**, e só depois permite imprimir.

### Novo Layout e Configurações

- **Configuração do IP e Porta da Impressora** para impressoras conectadas via rede.
- **Opção para ajustar o tamanho da etiqueta.**



## Correção para aceitar qualquer ficheiro CSV

- **Aceita qualquer ficheiro CSV**, independentemente das colunas.
- **Mostra as colunas disponíveis** e permite escolher quais usar.
- **Se não houver colunas Nome e Código**, permite usar qualquer outra.
- Gera a pré-visualização em PDF
- Gera o resultado final em PDF após a impressão
- Abre os PDFs e verifica o que será impresso e o que foi impresso

```
import tkinter as tk
from tkinter import filedialog, messagebox, ttk
import csv
import os
from fpdf import FPDF
import webbrowser

# Variáveis globais
zpl_atual = ""
csv_dados = []
csv_colunas = []
num_etiquetas = 1 # Número de etiquetas a gerar

# Função para selecionar e processar qualquer CSV
def selecionar_csv():
    global csv_dados, csv_colunas
    caminho_csv = filedialog.askopenfilename(filetypes=[("Ficheiros CSV", "*.csv")])

    if not caminho_csv:
        return

    try:
        with open(caminho_csv, newline='', encoding='utf-8') as ficheiro:
            leitor = csv.DictReader(ficheiro)
            csv_dados = list(leitor)
            if not csv_dados:
                messagebox.showwarning("Aviso", "O ficheiro CSV está vazio.")
                return

            csv_colunas = list(csv_dados[0].keys())
            atualizar_dropdowns(csv_colunas)

    except Exception as e:
        messagebox.showerror("Erro", f"Erro ao ler CSV: {e}")
```

```
# Atualizar os dropdowns para escolher colunas
def atualizar_dropdowns(colunas):
    coluna_nome['values'] = colunas
    coluna_codigo['values'] = colunas
    coluna_nome.set(colunas[0] if colunas else "")
    coluna_codigo.set(colunas[1] if len(colunas) > 1 else colunas[0] if colunas else "")

# Gerar múltiplas etiquetas
def gerar_multiplas_etiquetas():
    global zpl_atual, num_etiquetas
    try:
        num_etiquetas = int(num_etiquetas_entry.get())
    except ValueError:
        messagebox.showwarning("Aviso", "Por favor, insira um número válido de etiquetas.")
        return

    etiquetas = []
    for i in range(1, num_etiquetas + 1):
        zpl = f"Etiqueta {i}\nNome: Produto {i}\nCódigo: {1000 + i}"
        etiquetas.append(zpl)

    if etiquetas:
        zpl_atual = "\n\n".join(etiquetas)
        gerar_pdf_previsualizacao(zpl_atual)

# Gerar etiqueta a partir do CSV selecionado
def gerar_zpl_de_csv():
    global zpl_atual
    if not csv_dados:
        messagebox.showwarning("Aviso", "Nenhum ficheiro CSV selecionado.")
        return

    nome_coluna = coluna_nome.get()
    codigo_coluna = coluna_codigo.get()

    etiquetas = []
    for linha in csv_dados:
        nome = linha.get(nome_coluna, "Desconhecido")
        codigo = linha.get(codigo_coluna, "000000")
        zpl = f"Nome: {nome}\nCódigo: {codigo}"
        etiquetas.append(zpl)

    if etiquetas:
        zpl_atual = etiquetas[0]
        gerar_pdf_previsualizacao(zpl_atual)
```

```
# Função para gerar etiqueta com QR Code e Código de Barras
def gerar_zpl_com_codigos():
    global zpl_atual
    dado = input_qr_entry.get().strip()

    if not dado:
        messagebox.showwarning("Atenção", "Preencha o campo de conteúdo para QR Code/Código de Barras.")
        return

    zpl_atual = f"Conteúdo: {dado} (QR Code + Código de Barras)"
    gerar_pdf_previsualizacao(zpl_atual)

# Função para gerar o PDF da pré-visualização antes da impressão
def gerar_pdf_previsualizacao(texto):
    pdf = FPDF()
    pdf.add_page()
    pdf.set_font("Arial", size=12)
    pdf.cell(200, 10, txt="Pré-visualização da Etiqueta", ln=True, align='C')
    pdf.multi_cell(0, 10, texto)

    caminho_pdf = os.path.join(os.getcwd(), "etiqueta_previsualizacao.pdf")
    pdf.output(caminho_pdf)

    messagebox.showinfo("Pré-visualização", f"O PDF foi gerado e será aberto: {caminho_pdf}")
    webbrowser.open(caminho_pdf) # Abre o PDF automaticamente

# Função para gerar o PDF do resultado final após a impressão
def gerar_pdf_resultado_final():
    pdf = FPDF()
    pdf.add_page()
    pdf.set_font("Arial", size=12)
    pdf.cell(200, 10, txt="Resultado Final da Impressão", ln=True, align='C')
    pdf.multi_cell(0, 10, f"Etiqueta final gerada:\n\n{zpl_atual}")

    caminho_pdf = os.path.join(os.getcwd(), "resultado_final_etiqueta.pdf")
    pdf.output(caminho_pdf)

    messagebox.showinfo("Resultado Final", f"O PDF final foi gerado e será aberto: {caminho_pdf}")
    webbrowser.open(caminho_pdf) # Abre o PDF automaticamente

# Interface gráfica
root = tk.Tk()
root.title("Gerador de Etiquetas Zebra")
root.geometry("900x750") # Ajustado para mais espaço
root.resizable(False, False)
```

```
# Configuração do tamanho da etiqueta
tk.Label(root, text="Largura (pol):").pack()
largura_entry = tk.Entry(root, width=10)
largura_entry.pack()
largura_entry.insert(0, "4")

tk.Label(root, text="Altura (pol):").pack()
altura_entry = tk.Entry(root, width=10)
altura_entry.pack()
altura_entry.insert(0, "6")

# **Campo para inserir conteúdo do QR Code**
tk.Label(root, text="Conteúdo do QR Code / Código de Barras:").pack(pady=5)
input_qr_entry = tk.Entry(root, width=50)
input_qr_entry.pack()

# Campo para definir o número de etiquetas a gerar
tk.Label(root, text="Número de etiquetas:").pack(pady=5)
num_etiquetas_entry = tk.Entry(root, width=10)
num_etiquetas_entry.pack()
num_etiquetas_entry.insert(0, "1")

# Botão para selecionar CSV
tk.Button(root, text="Selecionar Ficheiro CSV", command=selecionar_csv, width=40).pack(pady=10)

# Dropdowns para selecionar colunas
tk.Label(root, text="Escolha a coluna do Nome:").pack()
coluna_nome = ttk.Combobox(root, state="readonly", width=30)
coluna_nome.pack()

tk.Label(root, text="Escolha a coluna do Código:").pack()
coluna_codigo = ttk.Combobox(root, state="readonly", width=30)
coluna_codigo.pack()

# Botões para gerar etiquetas
tk.Button(root, text="Gerar Etiqueta do CSV", command=gerar_zpl_de_csv, width=40).pack(pady=5)
tk.Button(root, text="Gerar QR + Código de Barras", command=gerar_zpl_com_codigos, width=40).pack(pady=5)
tk.Button(root, text="Gerar Múltiplas Etiquetas", command=gerar_multiplas_etiquetas, width=40).pack(pady=5)

# Botão de visualizar PDF pré-impressão
tk.Button(root, text="Pré-visualizar em PDF", command=lambda: gerar_pdf_previsualizacao(zpl_atual),
width=40).pack(pady=5)

# Botão para gerar o PDF do resultado final pós-impressão
tk.Button(root, text="Ver Resultado Final em PDF", command=gerar_pdf_resultado_final, width=40, bg="blue",
fg="white").pack(pady=5)
```

```
# Iniciar aplicação  
root.mainloop()
```

Guarda o código anterior num ficheiro com o nome etiquetas\_zebra\_gui\_v2.py

7. Gera um executável do teu programa 😊