```
# ========================
# 📦 IMPORTAÇÕES E CONFIG
# ========================
import os, io
import numpy as np
import pandas as pd
from PIL import Image as PILImage
from tqdm import tqdm
import tensorflow as tf
import tensorflow_hub as hub
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns
import ipywidgets as widgets
from IPython.display import display, clear_output
from google.colab import files

!pip install ipywidgets
from google.colab import output
output.enable_custom_widget_manager()


# Caminhos principais
IMAGE_DIR = "/content/imagens"
CACHE_FILE = "/content/catalogo_cache.npz"
os.makedirs(IMAGE_DIR, exist_ok=True)
```

```
Requirement already satisfied: parso<0.9.0,>=0.8.4 in /usr/local/lib/python3.12/dist-packages (from jedi>=0.16->ipython>=4.0.0->ip
Requirement already satisfied: entrypoints in /usr/local/lib/python3.12/dist-packages (from jupyter-client>=6.1.12->ipykernel>=4.5
Requirement already satisfied: jupyter-core>=4.9.2 in /usr/local/lib/python3.12/dist-packages (from jupyter-client>=6.1.12->ipyker
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from jupyter-client>=6.1.12->ipy
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.0-
Requirement already satisfied: argon2-cffi in /usr/local/lib/python3.12/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3
Requirement already satisfied: nbformat in /usr/local/lib/python3.12/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.6.
Requirement already satisfied: nbconvert>=5 in /usr/local/lib/python3.12/dist-packages (from notebook>=4.4.1->widgetsnbextension~=
Requirement already satisfied: Send2Trash>=1.8.0 in /usr/local/lib/python3.12/dist-packages (from notebook>=4.4.1->widgetsnbextens
Requirement already satisfied: terminado>=0.8.3 in /usr/local/lib/python3.12/dist-packages (from notebook>=4.4.1->widgetsnbextensi
Requirement already satisfied: prometheus-client in /usr/local/lib/python3.12/dist-packages (from notebook>=4.4.1->widgetsnbextens
Requirement already satisfied: nbclassic>=0.4.7 in /usr/local/lib/python3.12/dist-packages (from notebook>=4.4.1->widgetsnbextensi
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.12/dist-packages (from pexpect>4.3->ipython>=4.0.0->ipywi
Requirement already satisfied: wcwidth in /usr/local/lib/python3.12/dist-packages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.12/dist-packages (from jupyter-core>=4.9.2->jupyter-cli
Requirement already satisfied: notebook-shim>=0.2.3 in /usr/local/lib/python3.12/dist-packages (from nbclassic>=0.4.7->notebook>=4
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.12/dist-packages (from nbconvert>=5->notebook>=4.4.1->widg
Requirement already satisfied: bleach!=5.0.0 in /usr/local/lib/python3.12/dist-packages (from bleach[css]!=5.0.0->nbconvert>=5->no
Requirement already satisfied: defusedxml in /usr/local/lib/python3.12/dist-packages (from nbconvert>=5->notebook>=4.4.1->widgetsn
Requirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.12/dist-packages (from nbconvert>=5->notebook>=4.4.1-
Requirement already satisfied: markupsafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from nbconvert>=5->notebook>=4.4.1->wid
Requirement already satisfied: mistune<4,>=2.0.3 in /usr/local/lib/python3.12/dist-packages (from nbconvert>=5->notebook>=4.4.1->w
Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.12/dist-packages (from nbconvert>=5->notebook>=4.4.1->wid
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.12/dist-packages (from nbconvert>=5->notebook>=4.4.1
Requirement already satisfied: fastjsonschema>=2.15 in /usr/local/lib/python3.12/dist-packages (from nbformat->notebook>=4.4.1->wi
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.12/dist-packages (from nbformat->notebook>=4.4.1->widgets
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->jupyter-client>=6
Requirement already satisfied: argon2-cffi-bindings in /usr/local/lib/python3.12/dist-packages (from argon2-cffi->notebook>=4.4.1-
Requirement already satisfied: webencodings in /usr/local/lib/python3.12/dist-packages (from bleach!=5.0.0->bleach[css]!=5.0.0->nb
Requirement already satisfied: tinycss2<1.5,>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from bleach[css]!=5.0.0->nbconvert
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.12/dist-packages (from jsonschema>=2.6->nbformat->notebook>
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.12/dist-packages (from jsonschema>=2
Requirement already satisfied: arrow>=0.15.0 in /usr/local/lib/python3.12/dist-packages (from isoduration->jsonschema[format-nongp
```

```
# from google.colab import files
# uploaded = files.upload()  # Upload manual do CSV (desativado)


# ========================
# 📂 UPLOAD DE IMAGENS
# ========================
uploaded = files.upload()
for fname in uploaded.keys():
    with open(os.path.join(IMAGE_DIR, fname), 'wb') as f:
        f.write(uploaded[fname])
print(f"🧢 {len(uploaded)} imagens enviadas.")
```

Escolher arquivos 25 arquivos

- **Sapato7.png**(image/png) - 100472 bytes, last modified: 05/09/2025 - 100% done
- **Sapato3.jpg**(image/jpeg) - 94575 bytes, last modified: 05/09/2025 - 100% done
- **Sapato2.jpg**(image/jpeg) - 53136 bytes, last modified: 05/09/2025 - 100% done
- **Sapato1.png**(image/png) - 141983 bytes, last modified: 05/09/2025 - 100% done
- **Sapato.png**(image/png) - 55760 bytes, last modified: 05/09/2025 - 100% done
- **Sapato.jpg**(image/jpeg) - 73632 bytes, last modified: 05/09/2025 - 100% done
- **Relogio6.png**(image/png) - 44740 bytes, last modified: 05/09/2025 - 100% done
- **Relogio5.jpg**(image/jpeg) - 105084 bytes, last modified: 05/09/2025 - 100% done
- **Relogio3.jpeg**(image/jpeg) - 92427 bytes, last modified: 05/09/2025 - 100% done
- **Relogio2.jpeg**(image/jpeg) - 185584 bytes, last modified: 05/09/2025 - 100% done
- **Relogio1.png**(image/png) - 283992 bytes, last modified: 05/09/2025 - 100% done
- **Relogio.jpg**(image/jpeg) - 48289 bytes, last modified: 05/09/2025 - 100% done
- **Perfume8.jpg**(image/jpeg) - 68794 bytes, last modified: 05/09/2025 - 100% done
- **Perfume5.png**(image/png) - 111418 bytes, last modified: 05/09/2025 - 100% done
- **Perfume3.png**(image/png) - 8550 bytes, last modified: 05/09/2025 - 100% done
- **Perfume2.png**(image/png) - 27104 bytes, last modified: 05/09/2025 - 100% done
- **Perfume1.png**(image/png) - 18192 bytes, last modified: 05/09/2025 - 100% done
- **Perfume.jpg**(image/jpeg) - 137685 bytes, last modified: 05/09/2025 - 100% done
- **Jaqueta3.jpg**(image/jpeg) - 86937 bytes, last modified: 05/09/2025 - 100% done
- **Jaqueta2.jpg**(image/jpeg) - 20020 bytes, last modified: 05/09/2025 - 100% done
- **Jaqueta1.jpg**(image/jpeg) - 24433 bytes, last modified: 05/09/2025 - 100% done
- **Jaqueta.jpg**(image/jpeg) - 39000 bytes, last modified: 05/09/2025 - 100% done
- **Blazer2.jpg**(image/jpeg) - 67639 bytes, last modified: 05/09/2025 - 100% done
- **Blazer1.jpeg**(image/jpeg) - 87064 bytes, last modified: 05/09/2025 - 100% done
- **Blazer.jpg**(image/jpeg) - 26918 bytes, last modified: 05/09/2025 - 100% done

```
Saving Sapato7.png to Sapato7.png
Saving Sapato3.jpg to Sapato3.jpg
Saving Sapato2.jpg to Sapato2.jpg
Saving Sapato1.png to Sapato1.png
Saving Sapato.png to Sapato.png
Saving Sapato.jpg to Sapato.jpg
Saving Relogio6.png to Relogio6.png
Saving Relogio5.jpg to Relogio5.jpg
Saving Relogio3.jpeg to Relogio3.jpeg
Saving Relogio2.jpeg to Relogio2.jpeg
Saving Relogio1.png to Relogio1.png
Saving Relogio.jpg to Relogio.jpg
Saving Perfume8.jpg to Perfume8.jpg
Saving Perfume5.png to Perfume5.png
Saving Perfume3.png to Perfume3.png
Saving Perfume2.png to Perfume2.png
Saving Perfume1.png to Perfume1.png
Saving Perfume.jpg to Perfume.jpg
Saving Jaqueta3.jpg to Jaqueta3.jpg
Saving Jaqueta2.jpg to Jaqueta2.jpg
Saving Jaqueta1.jpg to Jaqueta1.jpg
Saving Jaqueta.jpg to Jaqueta.jpg
Saving Blazer2.jpg to Blazer2.jpg
Saving Blazer1.jpeg to Blazer1.jpeg
Saving Blazer.jpg to Blazer.jpg
🧢 25 imagens enviadas.
```

```
# ========================
# 🛠️ FUNÇÕES AUXILIARES
# ========================
def load_and_preprocess_image(path, size=(224, 224)):
    img = PILImage.open(path).convert('RGB')
    img = img.resize(size)
    return np.array(img) / 255.0

# Carrega modelo de extração de features
model_url = "https://tfhub.dev/google/bit/m-r50x1/1"
encoder = hub.load(model_url)

def extract_features(img_array):
    img_tensor = tf.convert_to_tensor([img_array], dtype=tf.float32)
    features = encoder(img_tensor)
    return np.array(features)[0]
```

```
# ========================
# 💾 CACHE AUTOMÁTICO
# ========================
if os.path.exists(CACHE_FILE):
    cache = np.load(CACHE_FILE, allow_pickle=True)
    vectors = list(cache['vectors'])
    names = list(cache['names'])
    print(f"🔄 Cache carregado com {len(names)} itens.")
else:
    vectors, names = [], []
    print("📂 Nenhum cache encontrado, criando do zero.")

all_images = sorted(os.listdir(IMAGE_DIR))
new_images = [img for img in all_images if img not in names]

if new_images:
    print(f"➕ {len(new_images)} imagens novas encontradas.")
    for img_name in tqdm(new_images):
        img_array = load_and_preprocess_image(os.path.join(IMAGE_DIR, img_name))
        vec = extract_features(img_array)
        vectors.append(vec)
        names.append(img_name)
    np.savez(CACHE_FILE, vectors=np.array(vectors), names=np.array(names))
    print("💾 Cache atualizado.")
else:
    print("✅ Nenhuma imagem nova.")

df = pd.DataFrame({"name": names})
images = [load_and_preprocess_image(os.path.join(IMAGE_DIR, n)) for n in df['name']]
vectors = np.array(vectors)
```

```
📂 Nenhum cache encontrado, criando do zero.
➕ 25 imagens novas encontradas.
100%|██████████| 25/25 [00:13<00:00,  1.83it/s]
💾 Cache atualizado.
```

```
# ========================
# 🗑️ HISTÓRICO DE EXCLUSÕES
# ========================
deleted_items = []

def restore_item(name):
    global vectors, df, images, deleted_items
    item = next((x for x in deleted_items if x['name'] == name), None)
    if item:
        idx = min(item['idx'], len(df))
        df = pd.concat([df.iloc[:idx], pd.DataFrame({"name": [item['name']]}), df.iloc[idx:]]).reset_index(drop=True)
        images.insert(idx, item['image'])
        vectors = np.insert(vectors, idx, [item['vector']], axis=0)
        PILImage.fromarray((item['image'] * 255).astype(np.uint8)).save(os.path.join(IMAGE_DIR, item['name']))
        np.savez(CACHE_FILE, vectors=vectors, names=df['name'].values)
        deleted_items = [x for x in deleted_items if x['name'] != name]
        show_recommendations(idx)

def delete_item(idx):
    global vectors, df, images, deleted_items
    del_name = df.loc[idx, 'name']
    deleted_items.append({'idx': idx, 'name': del_name, 'image': images[idx], 'vector': vectors[idx]})
    df.drop(idx, inplace=True)
    df.reset_index(drop=True, inplace=True)
    images.pop(idx)
    vectors = np.delete(vectors, idx, axis=0)
    img_path = os.path.join(IMAGE_DIR, del_name)
    if os.path.exists(img_path):
        os.remove(img_path)
    np.savez(CACHE_FILE, vectors=vectors, names=df['name'].values)
    show_restore_options()

def show_restore_options():
    if deleted_items:
        undo_btn = widgets.Button(description="↩️ Desfazer último", button_style='info')
        undo_btn.on_click(lambda b: restore_item(deleted_items[-1]['name']))
        dropdown = widgets.Dropdown(options=[x['name'] for x in deleted_items], description='Restaurar:')
        restore_btn = widgets.Button(description="Restaurar selecionado", button_style='success')
        restore_btn.on_click(lambda b: restore_item(dropdown.value))
        display(widgets.HBox([undo_btn, dropdown, restore_btn]))


# ========================
# 🖼️ VITRINE INTERATIVA
# ========================
def image_card_with_delete(img_array, label, idx, on_click_callback):
```

```
    buf = io.BytesIO()
    PILImage.fromarray((img_array * 255).astype(np.uint8)).save(buf, format='PNG')
    buf.seek(0)
    img_widget = widgets.Image(value=buf.getvalue(), format='png', width=150, height=150)
    btn_ref = widgets.Button(description=label, layout=widgets.Layout(width='150px'))
    btn_ref.on_click(lambda b: on_click_callback(idx))
    btn_del = widgets.Button(description="🗑️", layout=widgets.Layout(width='40px'))
    btn_del.on_click(lambda b: confirm_delete(idx))
    return widgets.VBox([img_widget, btn_ref, btn_del])

def confirm_delete(idx):
    del_name = df.loc[idx, 'name']
    confirm_label = widgets.Label(f"Tem certeza que deseja excluir '{del_name}'?")
    btn_yes = widgets.Button(description="Sim", button_style='danger')
    btn_no = widgets.Button(description="Não", button_style='success')
    btn_yes.on_click(lambda b: (clear_output(wait=True), delete_item(idx)))
    btn_no.on_click(lambda b: (clear_output(wait=True), show_recommendations(idx)))
    display(widgets.VBox([confirm_label, widgets.HBox([btn_yes, btn_no])]))

def show_recommendations(ref_index, top_k=5):
    clear_output(wait=True)
    if len(df) == 0:
        print("📂 Catálogo vazio.")
        show_restore_options()
        return
    ref_vector = vectors[ref_index]
    similarities = cosine_similarity([ref_vector], vectors)[0]
    top_indices = np.argsort(similarities)[::-1][1:top_k+1]
    print(f"Imagem de referência: {df['name'][ref_index]}")
    display(image_card_with_delete(images[ref_index], f"Referência\n(100%)", ref_index, show_recommendations))
    print("\nMais parecidas:")
    buttons = [image_card_with_delete(images[idx], f"{df['name'][idx]}\n({similarities[idx]*100:.1f}%)", idx, show_recommendations) for
    display(widgets.HBox(buttons))
    display(search_box)
    show_restore_options()

def on_search_change(change):
    query = change['new'].strip().lower()
    matches = df[df['name'].str.lower().str.contains(query)]
    if not matches.empty:
        show_recommendations(matches.index[0], top_k=5)

search_box = widgets.Text(placeholder='Digite parte do nome...', description='Buscar:', continuous_update=False)
search_box.observe(on_search_change, names='value')


# ========================
# 🚀 INICIAR SISTEMA
# ========================
show_recommendations(0, top_k=5)
```

```
Imagem de referência: Blazer.jpg
```

Referência (100%)

🗑️

Mais parecidas:

Blazer1.jpeg (89.6%)

🗑️

Blazer2.jpg (86.5%)

🗑️

Jaqueta.jpg (69.4%)

🗑️

Jaqueta1.jpg (64.1%)

🗑️

Jaqueta3.jpg (61.8%)

🗑️

Buscar: Digite parte do nome...

```
# ========================
# 🔎 BUSCA POR CATEGORIA (MESMA CATEGORIA)
# ========================
def show_recommendations_filtered(ref_index, candidate_indices, top_k=5):
    """
    Mostra 1 referência e as 5 mais parecidas, calculadas SOMENTE dentro
    do subconjunto candidate_indices (mesma categoria/termo).
    """
    clear_output(wait=True)
    if len(candidate_indices) == 0:
        print("Nenhum candidato encontrado para a categoria.")
        display(search_box)
        show_restore_options()
        return

    # Vetor de referência
    ref_vector = vectors[ref_index]
    # Vetores apenas dos candidatos
    cand_vectors = vectors[candidate_indices]
    sims = cosine_similarity([ref_vector], cand_vectors)[0]

    # Ordena por similaridade dentro do subconjunto e remove o próprio ref_index
    order = np.argsort(sims)[::-1]
    ordered_indices = [candidate_indices[i] for i in order if candidate_indices[i] != ref_index]
    top_indices = ordered_indices[:top_k]

    print(f"Imagem de referência: {df['name'][ref_index]}")
    display(image_card_with_delete(images[ref_index], f"Referência\n(100%)", ref_index, lambda idx: show_recommendations_filtered(idx, (

    print("\nMais 5 parecidas (mesma categoria):")
    buttons = []
    for idx in top_indices:
        sim_percent = cosine_similarity([ref_vector], [vectors[idx]])[0][0] * 100
        buttons.append(image_card_with_delete(images[idx], f"{df['name'][idx]}\n({sim_percent:.1f}%)", idx, lambda i: show_recommendatio

    if buttons:
        display(widgets.HBox(buttons))
    else:
        print("Sem similares suficientes nessa categoria.")

    display(search_box)
    show_restore_options()

def on_search_change_same_category(change):
    """
    Busca por termo (ex.: 'relogio', 'sapato', 'blazer', 'perfume', 'jaqueta')
    e mostra 1 referência + 5 similares dentro do mesmo grupo encontrado.
    """
    query = change['new'].strip().lower()
    if not query:
        return
    matches = df[df['name'].str.lower().str.contains(query)]
    if matches.empty:
        clear_output(wait=True)
        print(f"Nenhum resultado encontrado para: {query}")
        display(search_box)
        show_restore_options()
        return

    # Índice da referência (primeira ocorrência) e subconjunto candidato
    ref_index = matches.index[0]
    candidate_indices = matches.index.tolist()
    show_recommendations_filtered(ref_index, candidate_indices, top_k=5)

# Substitui o handler antigo pelo novo (mesma categoria)
try:
    search_box.unobserve_all('value')
except Exception:
    pass
search_box.observe(on_search_change_same_category, names='value')

print("Busca atualizada: referência + 5 similares dentro da MESMA categoria digitada.")
```

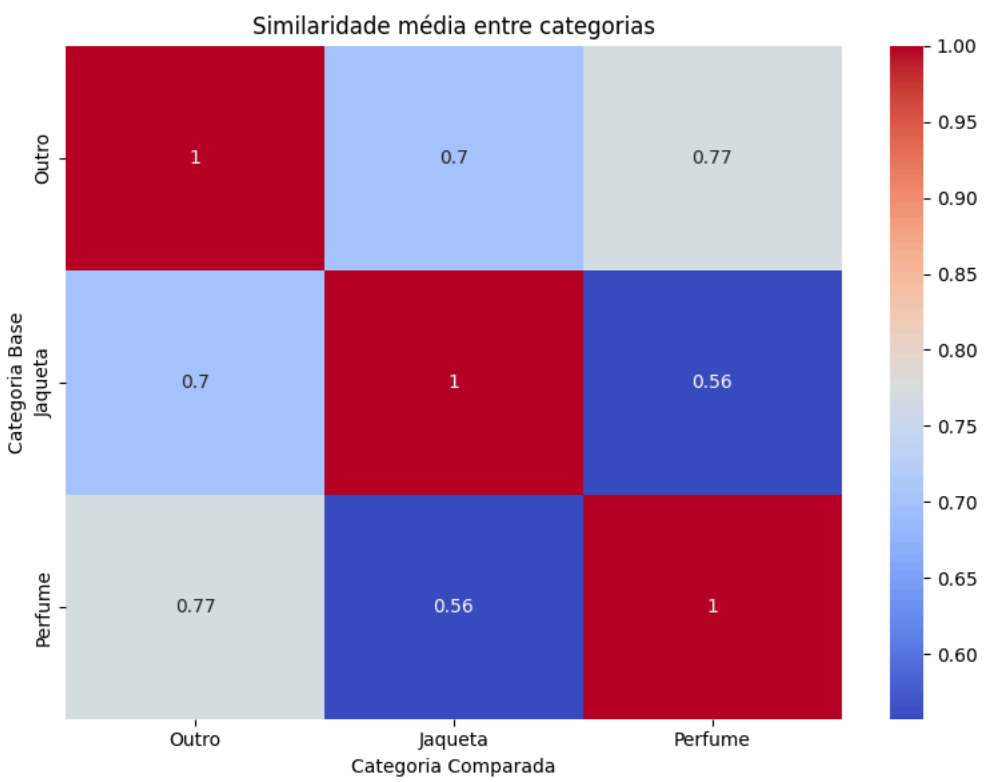
```
Busca atualizada: referência + 5 similares dentro da MESMA categoria digitada.
```

```
# ========================
# 🧰 UTILIDADES
# ========================

def salvar_cache():
    np.savez(CACHE_FILE, vectors=vectors, names=df['name'].values)
    print(f"Cache salvo: {CACHE_FILE}")
```

```
def rebuild_from_folder():
    """
    Recalcula TUDO a partir da pasta de imagens (útil se o cache corromper ou se trocar o encoder).
    """
    global vectors, names, df, images
    file_list = sorted(os.listdir(IMAGE_DIR))
    new_vectors, new_names = [], []
    for img_name in tqdm(file_list):
        img_array = load_and_preprocess_image(os.path.join(IMAGE_DIR, img_name))
        vec = extract_features(img_array)
        new_vectors.append(vec)
        new_names.append(img_name)

    vectors = np.array(new_vectors)
    names = new_names
    df = pd.DataFrame({"name": names})
    images = [load_and_preprocess_image(os.path.join(IMAGE_DIR, n)) for n in df['name']]
    salvar_cache()
    print("Reconstrução concluída.")

print("Utilidades carregadas: salvar_cache(), rebuild_from_folder()")
```

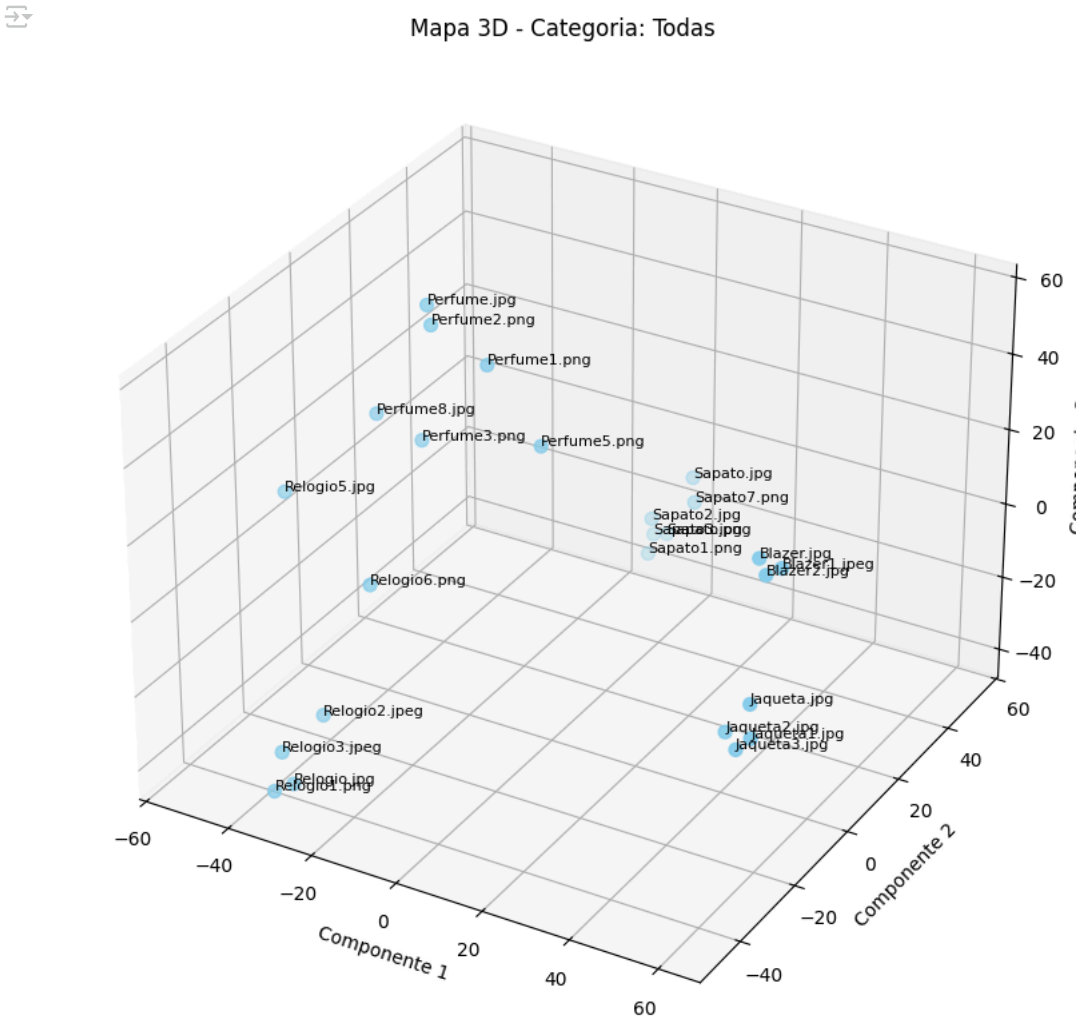
```
Utilidades carregadas: salvar_cache(), rebuild_from_folder()
```

```
# ========================
# 📊 BLOCO 12 - MATRIZ DE SIMILARIDADE ENTRE CATEGORIAS
# ========================

# 🔧 Atribuição automática de categorias com base no nome do arquivo
df['categoria'] = df['name'].apply(lambda x:
    'Tênis' if 'tenis' in x.lower() else
    'Perfume' if 'perfume' in x.lower() else
    'Jaqueta' if 'jaqueta' in x.lower() else
    'Outro'
)

# 📊 Geração da matriz de similaridade entre categorias
categorias = df['categoria'].unique()
matriz_cat = np.zeros((len(categorias), len(categorias)))

for i, cat1 in enumerate(categorias):
    for j, cat2 in enumerate(categorias):
        idx1 = df[df['categoria'] == cat1].index
        idx2 = df[df['categoria'] == cat2].index
        matriz_cat[i, j] = cosine_similarity(
            vectors[idx1].mean(axis=0).reshape(1, -1),
            vectors[idx2].mean(axis=0).reshape(1, -1)
        )[0][0]

# 🎨 Visualização com heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(matriz_cat, xticklabels=categorias, yticklabels=categorias, annot=True, cmap="coolwarm")
plt.title("Similaridade média entre categorias")
plt.xlabel("Categoria Comparada")
plt.ylabel("Categoria Base")
plt.tight_layout()
plt.show()
```




```
# ========================
# 🎯 PCA COM FILTRO POR CATEGORIA
# ========================

def plot_pca_filtrado(vectors, labels, categorias, modo='2D', categoria_selecionada=None):
    if categoria_selecionada:
        mask = df['categoria'] == categoria_selecionada
        vectors = vectors[mask]
        labels = df['name'][mask]
        categorias = df['categoria'][mask]

    if modo == '2D':
        pca = PCA(n_components=2)
        coords = pca.fit_transform(vectors)

        plt.figure(figsize=(10, 8))
        plt.scatter(coords[:, 0], coords[:, 1], c='skyblue', s=50)
        for i, name in enumerate(labels):
            plt.text(coords[i, 0], coords[i, 1], name, fontsize=8)

        plt.title(f"Mapa 2D - Categoria: {categoria_selecionada or 'Todas'}")
        plt.xlabel("Componente 1")
        plt.ylabel("Componente 2")
        plt.tight_layout()
        plt.show()

    else:
        pca = PCA(n_components=3)
        coords = pca.fit_transform(vectors)

        fig = plt.figure(figsize=(10, 8))
        ax = fig.add_subplot(111, projection='3d')
        ax.scatter(coords[:, 0], coords[:, 1], coords[:, 2], c='skyblue', s=50)
        for i, name in enumerate(labels):
            ax.text(coords[i, 0], coords[i, 1], coords[i, 2], name, fontsize=8)

        ax.set_title(f"Mapa 3D - Categoria: {categoria_selecionada or 'Todas'}")
        ax.set_xlabel("Componente 1")
        ax.set_ylabel("Componente 2")
        ax.set_zlabel("Componente 3")
        plt.tight_layout()
        plt.show()

# Widgets de controle
modo_pca = widgets.ToggleButtons(
    options=['2D', '3D'],
    description='Visualização:',
    button_style='info'
)
```

```
categoria_dropdown = widgets.Dropdown(
    options=['Todas'] + sorted(df['categoria'].unique()),
    description='Categoria:',
    style={'description_width': 'initial'}
)

def atualizar_visualizacao(change=None):
    clear_output(wait=True)
    modo = modo_pca.value
    categoria = categoria_dropdown.value
    cat = None if categoria == 'Todas' else categoria
    plot_pca_filtrado(vectors, df['name'], df['categoria'], modo=modo, categoria_selecionada=cat)
    display(widgets.HBox([modo_pca, categoria_dropdown]))

modo_pca.observe(atualizar_visualizacao, names='value')
categoria_dropdown.observe(atualizar_visualizacao, names='value')

# Exibe os controles e inicia com visualização padrão
atualizar_visualizacao()
```



Visualização: Categoria: Todas