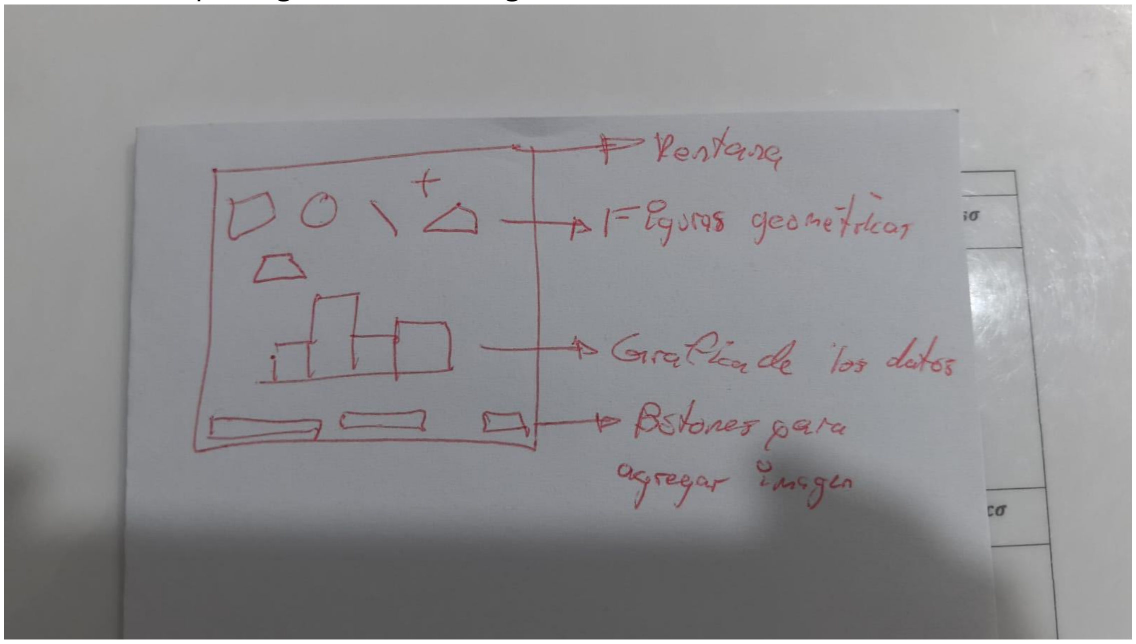
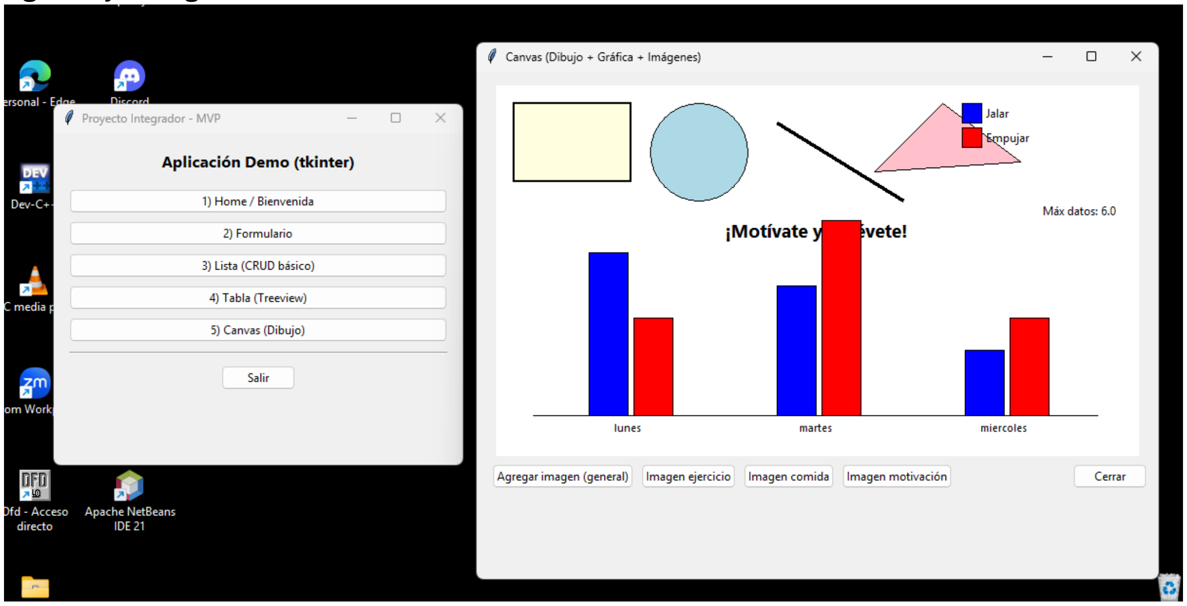
Angel Alexis Torres Viurquiz
Ventana: win_canvas.py

Los cambios que le generé son los siguientes:



Como se muestra en la foto, intenté añadir botones para añadir imágenes, más figuras y una grafica con los datos dado en la ventana de tabla.



Y estos cambios apoyan a crear este proyecto de una manera más dinámica, fácil de entender y ayuda a todas aquellas personas que quieren comenzar a tener un mejor físico y mejorar su salud a tomar en cuenta todos estos aspectos que tiene la app para organizar, crear y llevar a cabo cada ejercicio

Codigo:

```
import tkinter as tk
from tkinter import ttk, filedialog, messagebox
from pathlib import Path
import csv
import unicodedata

# Pillow es opcional: si está instalado permite JPG/JPEG. Si no, solo PNG/GIF con tk.PhotoImage.
try:
    from PIL import Image, ImageTk
    PIL_AVAILABLE = True
except Exception:
    PIL_AVAILABLE = False

def normalize_header(s):
    """Quita tildes, espacios y pone en minúscula para comparar encabezados."""
    if s is None:
        return ""
    s = s.strip()
    s = unicodedata.normalize("NFKD", s)
    s = "".join(ch for ch in s if not unicodedata.combining(ch))
    s = s.lower()
    s = "".join(ch for ch in s if ch.isalnum() or ch.isspace())
    s = s.replace(" ", "")
    return s

def parse_number(s):
    """Convierte cadenas tipo '1.234,56' o '1234' o '12,34' a float; devuelve 0.0 si no válido."""
    if s is None:
        return 0.0
    s = str(s).strip()
    if s == "":
        return 0.0
    s = s.replace("%", "").replace("$", "").replace(" ", "")
    try:
        # manejo mixto de separadores
        if s.count(",") > 0 and s.count(".") > 0:
            # suponemos que el último separador es decimal
            if s.rfind(",") > s.rfind("."):
                s = s.replace(".", "").replace(",", ".")
            else:
                s = s.replace(",", "")
        elif s.count(",") > 0 and s.count(".") == 0:
            s = s.replace(",", ".")
        cleaned = "".join(ch for ch in s if ch.isdigit() or ch in ".-")
        if cleaned in ("", "-", ".", "-."):
            return 0.0
        return float(cleaned)
    except Exception:
        return 0.0

def open_win_canvas(parent: tk.Tk):
    win = tk.Toplevel(parent)
    win.title("Canvas (Dibujo + Gráfica + Imágenes)")
    win.geometry("700x520")
    frm = ttk.Frame(win, padding=12)
    frm.pack(fill="both", expand=True)

    canvas_w, canvas_h = 660, 380
```

```
canvas = tk.Canvas(frm, width=canvas_w, height=canvas_h, bg="white")
canvas.pack()

# --- Figuras geométricas de ejemplo ---
canvas.create_rectangle(20, 20, 140, 100, outline="black", width=2, fill="lightyellow")
canvas.create_oval(160, 20, 260, 120, fill="lightblue", outline="black")
canvas.create_line(290, 40, 420, 120, width=4, smooth=True)
canvas.create_polygon(460, 20, 540, 80, 390, 90, fill="pink", outline="black")
canvas.create_text(330, 150, text="¡Motívate y muévete!", font=("Segoe UI", 14, "bold"))

# --- Leer CSV de forma robusta (varias codificaciones y normalización de encabezados) ---
ruta = Path(__file__).resolve().parents[1] / "data" / "sample2.csv"
rows = []
header_map = {}
if ruta.exists():
    encodings = ["utf-8", "latin-1", "cp1252"]
    read_success = False
    last_exc = None
    for enc in encodings:
        try:
            with open(ruta, "r", encoding=enc, newline="") as f:
                reader = csv.DictReader(f)
                fieldnames = reader.fieldnames
                if fieldnames:
                    header_map = {normalize_header(n): n for n in fieldnames if n is not None}
                    for row in reader:
                        rows.append(row)
                    read_success = True
                    break
                else:
                    # Si no hay encabezado, leemos filas simples y asumimos orden (Dia,Jalar,Empujar)
                    f.seek(0)
                    simple = list(csv.reader(f))
                    if simple:
                        for r in simple:
                            if len(r) >= 3:
                                rows.append({"Dia": r[0], "Jalar": r[1], "Empujar": r[2]})
                        read_success = True
                        break
        except UnicodeDecodeError as e:
            last_exc = e
            continue
        except Exception as e:
            last_exc = e
            break
    if not read_success:
        canvas.create_text(canvas_w / 2, canvas_h / 2, text=f"Error leyendo CSV: {last_exc}", fill="red")
else:
    canvas.create_text(canvas_w / 2, canvas_h / 2, text="No se encontró sample2.csv", fill="red")

# --- Normalizar campos y extraer números ---
data = []
if rows:
    def pick_field(candidates):
        for c in candidates:
            k = normalize_header(c)
            if k in header_map:
                return header_map[k]
        # fallback: comparar con las llaves de la primera fila
        sample = rows[0]
        for k in sample.keys():
            if normalize_header(k) in [normalize_header(x) for x in candidates]:
                return k
        return None

    dia_field = pick_field(["Día", "Dia", "day"])
    jalar_field = pick_field(["Jalar", "jalado", "pull", "jalar"])
    empujar_field = pick_field(["Empujar", "push", "empujar"])

    for r in rows:
        dia = r.get(dia_field) if dia_field else (next(iter(r.values())) if r else "")
        jalar = parse_number(r.get(jalar_field) if jalar_field else (list(r.values())[1] if len(r.values()) > 1 else 0))
        empujar = parse_number(r.get(empujar_field) if empujar_field else (list(r.values())[2] if len(r.values()) > 2 else
0))
        data.append((str(dia), jalar, empujar))

  # --- Graficar (barra agrupada) ---
  if data:
    max_items = 8
    data = data[:max_items]
    max_val = max((max(j, e) for _, j, e in data), default=1)
    max_bar_h = 200
    scale = max_bar_h / max_val if max_val > 0 else 1
    left = 40
    right = canvas_w - 40
    group_w = (right - left) / len(data)
    bar_w = min(40, group_w * 0.35)
    base_y = canvas_h - 40

    for i, (dia, jalar, empujar) in enumerate(data):
        gx = left + i * group_w
        x1 = gx + (group_w - 2 * bar_w) / 2
        y_top_j = base_y - jalar * scale
        canvas.create_rectangle(x1, y_top_j, x1 + bar_w, base_y, fill="blue")
        x2 = x1 + bar_w + 6
        y_top_e = base_y - empujar * scale
        canvas.create_rectangle(x2, y_top_e, x2 + bar_w, base_y, fill="red")
        canvas.create_text(gx + group_w / 2, base_y + 12, text=dia, font=("Segoe UI", 8))

    canvas.create_line(left, base_y, right, base_y)
    canvas.create_text(right - 20, base_y - max_bar_h - 10,
                 text=f"Máx datos: {round(max_val,2)}")
    # Leyenda
    canvas.create_rectangle(right - 140, 20, right - 120, 40, fill="blue")
    canvas.create_text(right - 115, 30, text="Jalar", anchor="w")
    canvas.create_rectangle(right - 140, 45, right - 120, 65, fill="red")
    canvas.create_text(right - 115, 55, text="Empujar", anchor="w")
  else:
    canvas.create_text(canvas_w / 2, canvas_h / 2 + 30, text="No hay datos para graficar", fill="gray")

  # --- Manejo de imágenes (soporte PNG/GIF directo; JPG si Pillow está instalado) ---
  images_refs = []

  def add_image_at(path, x, y, max_size=(120, 120)):
    if not path:
        return
    try:
        if PIL_AVAILABLE:
            img = Image.open(path)
            img.thumbnail(max_size, Image.LANCZOS)
            photo = ImageTk.PhotoImage(img)
        else:
            if path.lower().endswith((".png", ".gif")):
                photo = tk.PhotoImage(file=path)
            else:
                messagebox.showinfo("Formato no soportado",
                                    "Para JPG/JPEG es necesario instalar Pillow:\n\npip install pillow")
                return
        images_refs.append(photo)
        canvas.create_image(x, y, image=photo)
    except Exception as e:
        messagebox.showerror("Error al cargar imagen", str(e))

  def cargar_imagen_general():
    path = filedialog.askopenfilename(title="Selecciona imagen",
                                      filetypes=[("Imágenes", "*.png;*.jpg;*.jpeg;*.gif;*.bmp")])
    if path:
        count = len(images_refs)
        add_image_at(path, 80 + (count % 3) * 200, canvas_h - 80 - (count // 3) * 120)

  def cargar_ejercicio():
    path = filedialog.askopenfilename(title="Selecciona imagen de ejercicio",
                                      filetypes=[("Imágenes", "*.png;*.jpg;*.jpeg;*.gif")])
    add_image_at(path, 120, canvas_h - 160)

  def cargar_comida():
    path = filedialog.askopenfilename(title="Selecciona imagen de comida",
                                      filetypes=[("Imágenes", "*.png;*.jpg;*.jpeg;*.gif")])
    add_image_at(path, 330, canvas_h - 160)

  def cargar_motivacion():
    path = filedialog.askopenfilename(title="Selecciona imagen de motivación",
                                      filetypes=[("Imágenes", "*.png;*.jpg;*.jpeg;*.gif")])
    add_image_at(path, 540, canvas_h - 160)

  btn_frm = ttk.Frame(frm)
  btn_frm.pack(fill="x", pady=6)
  ttk.Button(btn_frm, text="Agregar imagen (general)", command=cargar_imagen_general).pack(side="left",
padx=4)
  ttk.Button(btn_frm, text="Imagen ejercicio", command=cargar_ejercicio).pack(side="left", padx=4)
  ttk.Button(btn_frm, text="Imagen comida", command=cargar_comida).pack(side="left", padx=4)
  ttk.Button(btn_frm, text="Imagen motivación", command=cargar_motivacion).pack(side="left", padx=4)
  ttk.Button(btn_frm, text="Cerrar", command=win.destroy).pack(side="right")
```