Código de la app — Revive (ODS 12)

Versión sin PIL / imágenes (solo stdlib)

Generado 2025-10-14 05:45

Este documento es evidencia del código fuente.

```
# -------------------------------------------------------------
# Revive (ODS 12) - versión sin PIL/imagenes (solo stdlib)
# Ejecuta: python revive_app_sin_imagen.py
# -------------------------------------------------------------
import tkinter as tk
from tkinter import ttk, filedialog, messagebox
from tkinter import StringVar
import io, base64, os, json, datetime, webbrowser

# ============= ARCHIVO DE DATOS =============
DATA_FILE = "revive_data.json"

def load_data():
    if not os.path.exists(DATA_FILE):
        data = {
            "users": {},           # email -> {name, city, created_at}
            "items": [],           # listado de artículos
            "stats": {             # acumulados globales (por simplicidad)
                "sold": 0,
                "bought": 0,
                "co2_saved_kg": 0.0
            }
        }
        save_data(data)
        return data
    with open(DATA_FILE, "r", encoding="utf-8") as f:
        return json.load(f)

def save_data(data):
    with open(DATA_FILE, "w", encoding="utf-8") as f:
        json.dump(data, f, ensure_ascii=False, indent=2)

# ============= COEFICIENTES CO2 (estimados sencillos) =============
CO2_KG_BY_CATEGORY = {
    "Electrónica": 50.0,
    "Ropa": 3.0,
    "Muebles": 25.0,
    "Libros": 1.0,
    "Deporte": 8.0,
    "Hogar": 5.0,
    "Otro": 4.0
}
ANNUAL_GOAL_KG = 100.0  # objetivo personal para la barra de progreso

# ============= REGLAS DE SUGERENCIA (texto -> vender/reparar/donar) =============
NEGATIVE_KEYS = ["roto", "quebrado", "no sirve", "descompuesto", "fallas", "dañado", "manchado"]
REPAIR_KEYS   = ["pantalla", "batería", "costura", "cierre", "soldar", "coser", "tornillo", "pieza"]
DONATE_KEYS   = ["regalar", "donar", "gratis", "necesitado", "escuela", "comunidad"]

def suggest_action(description: str) -> str:
    t = description.lower()
    score = {"Vender": 0, "Reparar": 0, "Donar": 0}
    if any(k in t for k in DONATE_KEYS): score["Donar"] += 2
    if any(k in t for k in REPAIR_KEYS): score["Reparar"] += 2
    if any(k in t for k in NEGATIVE_KEYS): score["Reparar"] += 1; score["Donar"] += 1
    if all(v == 0 for v in score.values()):
        return "Vender"
    return max(score, key=score.get)

# ============= UI: App con Frames de navegación =============
class ReviveApp(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Revive – ODS 12 (sin imágenes)")
        self.geometry("980x650")
        self.minsize(900, 620)
        self.configure(bg="white")
        self.style = ttk.Style(self)
        self.style.theme_use('clam')
        self.style.configure("TButton", padding=10, font=("Segoe UI", 11, "bold"))
        self.style.configure("TLabel", font=("Segoe UI", 11))
        self.style.configure("H.TLabel", font=("Segoe UI", 18, "bold"))
        self.current_user = None
        self.data = load_data()

        container = ttk.Frame(self, padding=10)
        container.pack(fill="both", expand=True)

        self.frames = {}
        for F in (Landing, Auth, Dashboard, SearchItems, UploadItem, Impact):
            page = F(parent=container, app=self)
            self.frames[F.__name__] = page
            page.grid(row=0, column=0, sticky="nsew")

        self.show("Landing")

    def show(self, name):
        frame = self.frames[name]
        frame.event_generate("<<ShowFrame>>", when="tail")
        frame.tkraise()

    # Helpers para stats
    def add_co2(self, kg):
        self.data["stats"]["co2_saved_kg"] = round(self.data["stats"]["co2_saved_kg"] + float(kg),
2)
        save_data(self.data)

    def inc_sold(self):
        self.data["stats"]["sold"] += 1; save_data(self.data)

    def inc_bought(self):
        self.data["stats"]["bought"] += 1; save_data(self.data)



# ============= Landing (presentación) =============
class Landing(ttk.Frame):
    def __init__(self, parent, app: ReviveApp):
        super().__init__(parent)
        self.app = app
        self.columnconfigure(0, weight=1)
        card = ttk.Frame(self, padding=30)
        card.grid(row=0, column=0, sticky="nsew", padx=40, pady=30)
        card.columnconfigure(0, weight=1)

        ttk.Label(card, text="Revive", style="H.TLabel", foreground="#0A4D8C").grid(row=0, column=0,
 pady=10)
        ttk.Label(card, text="Consumo y producción responsables (ODS 12).",
                  style="H.TLabel").grid(row=1, column=0, pady=(0, 10))
        ttk.Label(card, wraplength=780, justify="center",
                  text="Publica lo que ya no usas. Compra de segunda. Repara con talleres locales. "
                       "Cada acción suma y evita CO₂.").grid(row=2, column=0, pady=(0, 20))

        ttk.Button(card, text="Iniciar sesión / Registrarme", command=lambda:
self.app.show("Auth")).grid(row=3, column=0)

        ttk.Label(card, foreground="#64748B",
                  text="Demo educativa. Datos locales en revive_data.json").grid(row=4, column=0,
pady=(30,0))



# ============= Auth (registro simple) =============
class Auth(ttk.Frame):
    def __init__(self, parent, app: ReviveApp):
        super().__init__(parent)
        self.app = app
        self.email = StringVar(); self.name = StringVar(); self.city = StringVar()

        wrap = ttk.Frame(self, padding=20)
        wrap.pack(expand=True)

        ttk.Label(wrap, text="Bienvenida/o a Revive", style="H.TLabel").grid(row=0, column=0,
columnspan=2, pady=(10,20))

        form = ttk.Frame(wrap)
        form.grid(row=1, column=0, sticky="w")

        ttk.Label(form, text="Correo").grid(row=0, column=0, sticky="w")
        ttk.Entry(form, width=38, textvariable=self.email).grid(row=1, column=0, pady=(0,10))

        ttk.Label(form, text="Nombre").grid(row=2, column=0, sticky="w")
        ttk.Entry(form, width=38, textvariable=self.name).grid(row=3, column=0, pady=(0,10))

        ttk.Label(form, text="Ciudad").grid(row=4, column=0, sticky="w")
        ttk.Entry(form, width=38, textvariable=self.city).grid(row=5, column=0, pady=(0,20))

        ttk.Button(form, text="Continuar", command=self.register).grid(row=6, column=0, sticky="ew")

    def register(self):
        email = self.email.get().strip().lower()
        name = self.name.get().strip()
        city = self.city.get().strip()
        if not email or "@" not in email:
            messagebox.showerror("Ups", "Escribe un correo válido.")
            return
        if not name or not city:
            messagebox.showerror("Ups", "Nombre y ciudad son obligatorios.")
            return

        users = self.app.data["users"]
        if email not in users:
            users[email] = {"name": name, "city": city, "created_at":
datetime.datetime.now().isoformat()}
            save_data(self.app.data)

        self.app.current_user = {"email": email, "name": users[email]["name"], "city":
users[email]["city"]}
        messagebox.showinfo("Listo", f"Hola, {users[email]['name']} de {users[email]['city']} □")
        self.app.show("Dashboard")



# ============= Dashboard =============
class Dashboard(ttk.Frame):
    def __init__(self, parent, app: ReviveApp):
        super().__init__(parent)
        self.app = app
        self.bind("<<ShowFrame>>", self.refresh)

        top = ttk.Frame(self, padding=10)
        top.pack(fill="x")
        self.greet = ttk.Label(top, text="", style="H.TLabel")
        self.greet.pack(side="left", padx=(0,10))

        ttk.Button(top, text="Buscar artículos", command=lambda:
app.show("SearchItems")).pack(side="left", padx=5)
        ttk.Button(top, text="Subir artículo", command=lambda:
app.show("UploadItem")).pack(side="left", padx=5)
        ttk.Button(top, text="Indicador de CO₂", command=lambda:
app.show("Impact")).pack(side="left", padx=5)

        main = ttk.Frame(self, padding=18)
        main.pack(fill="both", expand=True)

        intro = ("Aquí puedes:\n"
                 "• Buscar y comprar/intercambiar usados\n"
                 "• Subir lo que ya no usas (con foto y descripción)\n"
                 "• Ver el CO₂ evitado y tu progreso\n"
                 "Tip: describe honestamente el estado; si está roto, sugiere reparar o donar.")
        ttk.Label(main, text=intro, justify="left").pack(anchor="w")

    def refresh(self, *_):
        u = self.app.current_user
        if u:
            self.greet.config(text=f"Revive — Dashboard  |  {u['name']} · {u['city']}")



# ============= Buscar artículos =============
class SearchItems(ttk.Frame):
    def __init__(self, parent, app: ReviveApp):
        super().__init__(parent)
        self.app = app
        self.query = StringVar()
        self.city_filter = StringVar(value="Todas")
        self.category_filter = StringVar(value="Todas")

        hdr = ttk.Frame(self, padding=10); hdr.pack(fill="x")
        ttk.Label(hdr, text="Buscar artículos", style="H.TLabel").pack(side="left")
        ttk.Button(hdr, text="Volver", command=lambda: app.show("Dashboard")).pack(side="right")

        filt = ttk.Frame(self, padding=10); filt.pack(fill="x")
        ttk.Entry(filt, width=40, textvariable=self.query).pack(side="left", padx=(0,8))
        ttk.Combobox(filt, width=14, state="readonly", textvariable=self.city_filter,
                     values=["Todas"] + sorted({u["city"] for u in
app.data["users"].values()})).pack(side="left", padx=5)
        ttk.Combobox(filt, width=16, state="readonly", textvariable=self.category_filter,
                     values=["Todas"] + list(CO2_KG_BY_CATEGORY.keys())).pack(side="left", padx=5)
        ttk.Button(filt, text="Buscar", command=self.update_list).pack(side="left", padx=8)

        self.tree = ttk.Treeview(self, columns=("titulo","ciudad","categoria","sug","co2"),
show="headings", height=16)
        for i, t in enumerate(("Título","Ciudad","Categoría","Sugerencia","CO₂ (kg pot.)")):
            self.tree.heading(self.tree["columns"][i], text=t)
            self.tree.column(self.tree["columns"][i], width=[280,120,120,120,120][i], anchor="w")
        self.tree.pack(fill="both", expand=True, padx=10, pady=(0,10))

        bar = ttk.Frame(self, padding=10); bar.pack(fill="x")
        ttk.Button(bar, text="Ver detalle / Comprar", command=self.buy_selected).pack(side="left")
        ttk.Button(bar, text="Abrir talleres locales (Google Maps)",
command=self.open_maps).pack(side="left", padx=8)

        self.bind("<<ShowFrame>>", lambda *_: self.update_list())

    def filtered_items(self):
        q = self.query.get().strip().lower()
        cf = self.city_filter.get()
        cat = self.category_filter.get()
        out = []
        for it in self.app.data["items"]:
            if q and (q not in it["title"].lower() and q not in it["description"].lower()):
                continue
            if cf != "Todas" and it.get("city","") != cf:
                continue
            if cat != "Todas" and it.get("category","Otro") != cat:
                continue
            out.append(it)
        return out

    def update_list(self):
        for r in self.tree.get_children(): self.tree.delete(r)
        for it in self.filtered_items():
            self.tree.insert("", "end", iid=it["id"],
                             values=(it["title"], it.get("city",""), it.get("category","Otro"),
                                     it.get("suggestion",""),
CO2_KG_BY_CATEGORY.get(it.get("category","Otro"),4.0)))
```

```
    def buy_selected(self):
        sel = self.tree.selection()
        if not sel:
            messagebox.showwarning("Nada seleccionado", "Elige un artículo.")
            return
        item_id = sel[0]
        item = next((x for x in self.app.data["items"] if x["id"] == item_id), None)
        if not item:
            return
        cat = item.get("category","Otro")
        kg = CO2_KG_BY_CATEGORY.get(cat, 4.0)
        self.app.inc_bought()
        self.app.add_co2(kg)
        messagebox.showinfo("¡Hecho!", f"Marcaste como comprado.\nCO₂ evitado estimado: {kg:.1f}
kg")
        self.update_list()

    def open_maps(self):
        city = (self.app.current_user or {}).get("city","")
        q = f"https://www.google.com/maps/search/taller+reparación+{city}".replace(" ", "+")
        webbrowser.open(q)


# ============= Subir artículo =============
class UploadItem(ttk.Frame):
    def __init__(self, parent, app: ReviveApp):
        super().__init__(parent)
        self.app = app
        self.title_var = StringVar()
        self.desc_var = StringVar()
        self.cat_var = StringVar(value="Otro")
        self.suggestion_var = StringVar(value="-")
        self.image_bytes_b64 = None
        self.image_path = StringVar(value="(sin foto)")

        hdr = ttk.Frame(self, padding=10); hdr.pack(fill="x")
        ttk.Label(hdr, text="Subir artículo", style="H.TLabel").pack(side="left")
        ttk.Button(hdr, text="Volver", command=lambda: app.show("Dashboard")).pack(side="right")

        form = ttk.Frame(self, padding=16); form.pack(fill="x")
        ttk.Label(form, text="Título").grid(row=0, column=0, sticky="w")
        ttk.Entry(form, width=48, textvariable=self.title_var).grid(row=1, column=0, sticky="w",
pady=(0,10))

        ttk.Label(form, text="Descripción (estado, fallas, uso, etc.)").grid(row=2, column=0,
sticky="w")
        ttk.Entry(form, width=70, textvariable=self.desc_var).grid(row=3, column=0, sticky="w",
pady=(0,10))

        ttk.Label(form, text="Categoría").grid(row=4, column=0, sticky="w")
        ttk.Combobox(form, width=20, state="readonly", textvariable=self.cat_var,
                     values=list(CO2_KG_BY_CATEGORY.keys())).grid(row=5, column=0, sticky="w",
pady=(0,10))

        imgbox = ttk.Frame(self, padding=10); imgbox.pack(fill="x")
        ttk.Button(imgbox, text="Elegir foto…", command=self.pick_image).pack(side="left")
        ttk.Label(imgbox, textvariable=self.image_path).pack(side="left", padx=12)

        sug = ttk.Frame(self, padding=10); sug.pack(fill="x")
        ttk.Label(sug, text="Sugerencia del sistema:").pack(side="left")
        ttk.Label(sug, textvariable=self.suggestion_var, foreground="#0A4D8C", font=("Segoe UI", 12,
 "bold")).pack(side="left", padx=6)

        actions = ttk.Frame(self, padding=10); actions.pack(fill="x")
        ttk.Button(actions, text="Calcular sugerencia",
command=self.run_suggestion).pack(side="left")
        ttk.Button(actions, text="Publicar", command=self.publish).pack(side="left", padx=8)

        note = ("La sugerencia usa reglas simples por palabras clave (ej. 'roto', 'batería',
'donar'). "
                "Es apoyo; la decisión final es tuya.")
        ttk.Label(self, text=note, foreground="#64748B", wraplength=800).pack(anchor="w", padx=12,
pady=(0,10))

    def pick_image(self):
        path = filedialog.askopenfilename(title="Elige una imagen", filetypes=[("Imágenes",
"*.png;*.jpg;*.jpeg")])
        if not path: return
        self.image_path.set(os.path.basename(path))
        with open(path, "rb") as f:
            data = f.read()
        self.image_bytes_b64 = base64.b64encode(data).decode("ascii")

    def run_suggestion(self):
        s = suggest_action(self.desc_var.get())
        self.suggestion_var.set(s)

    def publish(self):
        if not self.app.current_user:
            messagebox.showerror("Ups", "Primero inicia sesión.")
            return
        t = self.title_var.get().strip()
        d = self.desc_var.get().strip()
        c = self.cat_var.get()
        if not t or not d:
            messagebox.showerror("Falta info", "Título y descripción son necesarios.")
            return
        s = self.suggestion_var.get()
        if s == "-": s = suggest_action(d)

        item = {
            "id": f"it_{len(self.app.data['items'])+1}_{int(datetime.datetime.now().timestamp())}",
            "title": t,
            "description": d,
            "category": c,
            "suggestion": s,
            "city": self.app.current_user["city"],
            "owner": self.app.current_user["email"],
            "created_at": datetime.datetime.now().isoformat(),
            "image_b64": self.image_bytes_b64,
            "image_name": self.image_path.get()
        }
        self.app.data["items"].append(item)
        self.app.inc_sold()  # simplificado
        save_data(self.app.data)

        kg = CO2_KG_BY_CATEGORY.get(c, 4.0)
        self.app.add_co2(kg * 0.5)  # publicar: 50% del potencial
        messagebox.showinfo("Publicado", f"Artículo subido.\nImpacto inicial estimado: {kg*0.5:.1f}
kg CO₂")

        # reset
        self.title_var.set(""); self.desc_var.set(""); self.suggestion_var.set("-")
        self.image_path.set("(sin foto)"); self.image_bytes_b64 = None



# ============= Indicador de CO₂ (y Progreso) =============
class Impact(ttk.Frame):
    def __init__(self, parent, app: ReviveApp):
        super().__init__(parent)
        self.app = app
        self.bind("<<ShowFrame>>", self.refresh)

        hdr = ttk.Frame(self, padding=10); hdr.pack(fill="x")
        ttk.Label(hdr, text="Indicador de CO₂", style="H.TLabel").pack(side="left")
        ttk.Button(hdr, text="Volver", command=lambda: app.show("Dashboard")).pack(side="right")

        self.info = ttk.Label(self, text="", font=("Segoe UI", 12))
        self.info.pack(anchor="w", padx=16, pady=(8, 0))

        pr = ttk.Frame(self, padding=14); pr.pack(fill="x")
        ttk.Label(pr, text="Progreso vs objetivo anual (100 kg):").pack(anchor="w")
        self.pb = ttk.Progressbar(pr, length=520, mode="determinate", maximum=ANNUAL_GOAL_KG)
        self.pb.pack(anchor="w", pady=(4,0))

        ttk.Button(self, text="Revisar progreso", command=self.show_progress).pack(anchor="w",
padx=16, pady=12)
        self.prog_lbl = ttk.Label(self, text="", font=("Segoe UI", 11))
        self.prog_lbl.pack(anchor="w", padx=16)

        tip = ("ODS 12: Reducimos residuos, prolongamos vida útil y apoyamos economía local. "
               "Comprar usado o reparar evita emisiones de fabricación y transporte.")
        ttk.Label(self, text=tip, wraplength=800, foreground="#64748B").pack(anchor="w", padx=16,
pady=(14,0))

    def refresh(self, *_):
        kg = self.app.data["stats"]["co2_saved_kg"]
        self.info.config(text=f"CO₂ evitado acumulado: {kg:.1f} kg")
        self.pb['value'] = min(kg, ANNUAL_GOAL_KG)

    def show_progress(self):
        s = self.app.data["stats"]["sold"]
        b = self.app.data["stats"]["bought"]
        kg = self.app.data["stats"]["co2_saved_kg"]
        pct = min(kg / ANNUAL_GOAL_KG * 100, 100.0)
        self.prog_lbl.config(text=f"Productos vendidos: {s}   |   comprados: {b}   |   Progreso CO₂:
 {pct:.1f}%")

# ============= main =============
if __name__ == "__main__":
    app = ReviveApp()
    app.mainloop()
```