Café “El Economista”

Integrantes

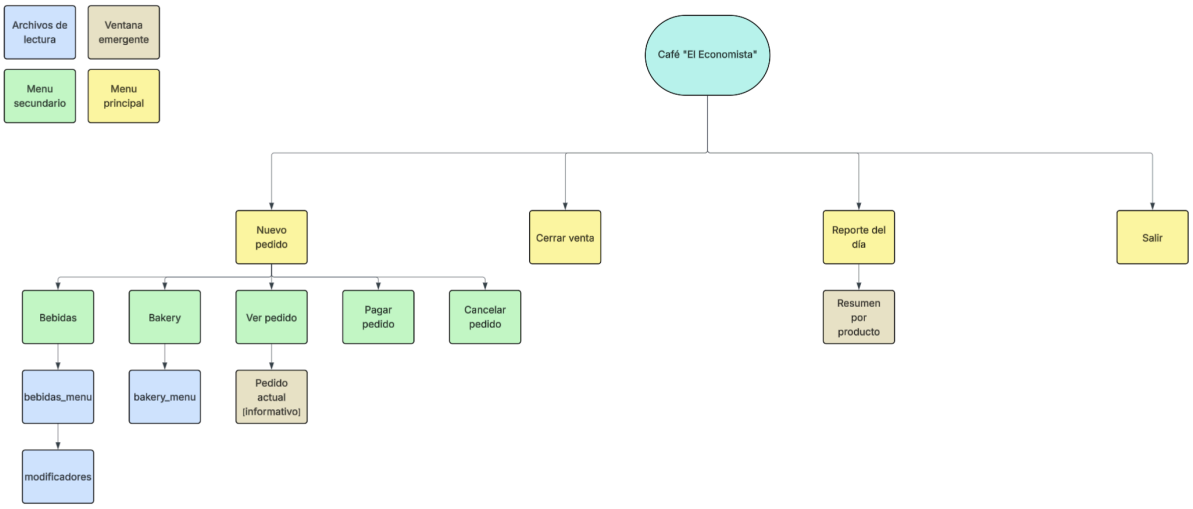
- Chuvac González, Berni Alejandro
- Pérez Romero, Rita Guadalupe

Descripción del proyecto

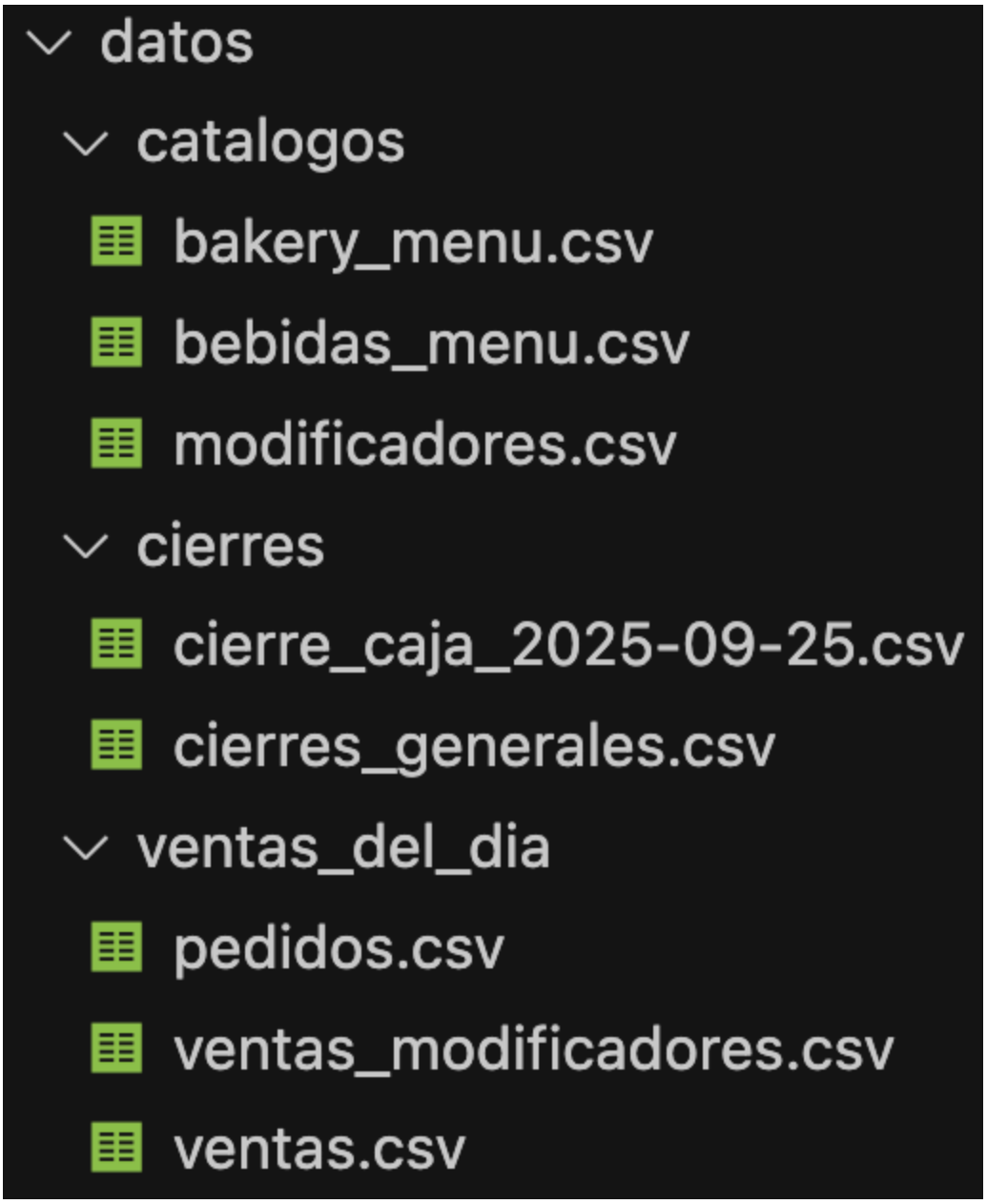
Sistema de caja registradora para un café ("El Economista"), donde el colaborador puede elegir entre diversos productos (bebidas, productos de bakery, extras, etc). El programa lleva un inventario que se actualiza automáticamente cuando se venden productos, y al agotarse aparecerá como "agotado". También calcula el total a pagar. Se podrá generar un reporte con las ventas y el inventario. La interacción es a través de una interfaz gráfica desarrollada con Tkinter.

Diagramas

▼ 1. Diagrama de navegación



▼ 2. Esquema de carpetas y archivos


datos
catalogos
bakery_menu.csv
bebidas_menu.csv
modificadores.csv
cierres
cierre_caja_2025-09-25.csv
cierres_generales.csv
ventas_del_dia
pedidos.csv
ventas_modificadores.csv
ventas.csv

Proyecto (código + interfaz)

Diagramas

Explicación:

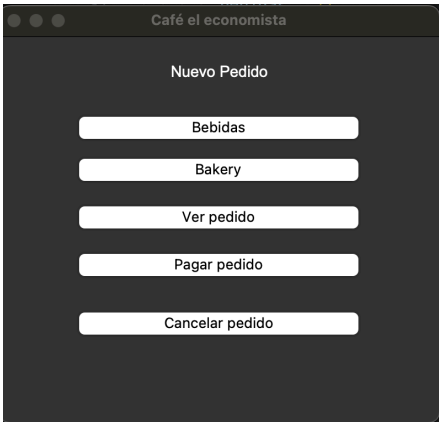
1. Se arma la ruta del archivo bebidas_menu.csv que está en la carpeta de catálogos.
2. Se abre el archivo con utf-8-sig para no tener problemas con acentos.
3. Se usa csv.DictReader para leer cada fila como un diccionario.
4. Se limpia cada valor (.strip()) y se convierten tipos: precios a float, activo a int.
5. Se devuelve una lista de diccionarios, cada uno representando una bebida.

Notas:

Las funciones leer_bakery() y leer_modificadores() siguen exactamente el mismo patron, cambiando solo las columnas especificas de cada catalogo.

```
carpeta_catalogo = Path("datos/catalogos")

#Definir la primera función para leer catálogo de bebidas
def leer_bebidas():

# Creamos la ruta completa al archivo CSV
    ruta = carpeta_catalogo / "bebidas_menu.csv"

# Abrimos el archivo y leemos su contenido y lo almacenamos en un dicciona
rio en f
    with open(ruta, newline="", encoding='utf-8-sig') as f:
        # leer el archivo CSV y indicar que el delimitador es una coma
        archivo = csv.DictReader(f, delimiter=',')
        bebidas = []
```

```
        for fila in archivo:
# Agregamos cada fila del archivo CSV a la lista bebidas
            bebidas.append({"id_bebida": fila["id_bebida"].strip(),"nombre": fila["n
ombre"].strip(), "precios_base": float(fila["precios_base"]),"categoria": fila.get
("categoria", "").strip(),"activo": int(fila["activo"])})
    return bebidas

# Probamos la función
if __name__ == "__main__":
    print(leer_bebidas())
```

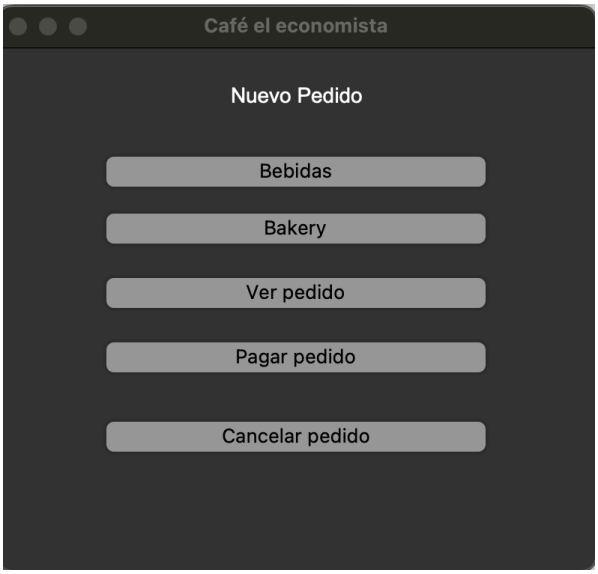



```
import tkinter as tk # importar librería para utilizar métodos de Tkinter
from tkinter import messagebox # mensajes emergentes
import datetime
import data_io
```

```
pedido_actual = {
    "fecha_hora": datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%
S"),
    "lineas": [],
    "total": 0.0}
id_linea_actual = 0
```

```
def abrir_ventana_pedido():
    """Implementa la lógica para abrir la venta de nuevo pedido"""
    ventana_pedido = tk.Toplevel(ventana)
    ventana_pedido.title("Café el economista")
    ventana_pedido.geometry("400×350")

    tk.Label(ventana_pedido, text="Nuevo Pedido", font=("Arial", 14)).pack(pad
y=20)
    def cancelar_pedido():
        confirmar = messagebox.askyesno("Cancelar pedido", "¿Realmente dese
a cancelar el pedido?")
        if confirmar:
            pedido_actual["lineas"].clear()
            pedido_actual["total"] = 0.0
            pedido_actual["fecha_hora"] = datetime.datetime.now().strftime("%Y-
%m-%d %H:%M:%S")
            global id_linea_actual
            id_linea_actual = 0
            messagebox.showinfo("Pedido cancelado", "El pedido fue cancelado c
orrectamente.")
            ventana_pedido.destroy()

    # Botones del menú de pedido
    tk.Button(ventana_pedido, text="Bebidas", width=25, command=mostrar_b
ebidas).pack(pady=5)
    tk.Button(ventana_pedido, text="Bakery", width=25, command=mostrar_ba
kery).pack(pady=5)
    tk.Button(ventana_pedido, text="Ver pedido",width=25, command=ver_pedi
do).pack(pady=10)
    tk.Button(ventana_pedido, text="Pagar pedido", width=25, command=paga
r_pedido).pack(pady=5)
    tk.Button(ventana_pedido, text="Cancelar pedido", width=25, command=ca
ncelar_pedido).pack(pady=20)
```

```
BEBIDAS = data_io.leer_bebidas()
BAKERY = data_io.leer_bakery()
MODIFICADORES = data_io.leer_modificadores()

def mostrar_bakery():
    ventana_bakery = tk.Toplevel()
    ventana_bakery.title("Café el economista")

    tk.Label(ventana_bakery, text="Productos de Bakery", font=("Arial", 14)).gri
d(row=0, column=0, columnspan=4, pady=10)

    productos_activos = [p for p in BAKERY if p["activo"] == 1]
```



Base de productos

Bebidas

Nombre Bebida	Precio base, Q	Categoría	Activo
guatemalteco	19	Caliente	1
Cappuccino	26	Caliente	1
Latte	25	Caliente	1
Mocca Latte	33	Caliente	1
Caramel Macchiato	33	Caliente	1
Bianca Latte	33	Caliente	1
Chocolate caliente	29	Caliente	1
Flat White	29	Caliente	1
Te Chai	35	Caliente	1
Te Matcha	35	Caliente	1
Clasico	33	Frappé	1
Caramel Frappe	33	Frappé	1
Mocca Frappe	33	Frappé	1
Bianca Mocca Frappe	33	Frappé	1
Chocomaniaco	36	Frappé	1
Vainila Frappe	36	Frappé	1

Nombre Bebida	Precio base, Q	Categoría	Activo
White Chocolate Snow	36	Frappé	1
Te Chai FrappeTe	35	Frappé	1
Cold brew coffee	22	Fría	1
Cold brew Latte	26	Fría	1
Iced Coffee	19	Fría	1

Bakery

Nombre Cakes	Precio base, Q	Categoría	Activo
Muffin de vainilla	17	Muffin	1
Pound cake de limon	18	Cake	1
Brownie de chocolate	22	Cake	1
Muffin de chocolate	22	Muffin	1
Tartaleta de pecanas	22	Cake	1
Pastel de chocolate y cajeta	30	Cake	1

Modificadores

Extras	Precio base, Q	Tipo	Activo
shot de espresso	5	cafe	1
Jarabe de vainilla	3	Jarabe	1
Jarabe caramelo	3	Jarabe	1
Leche entera	0	Leche	1

Extras	Precio base, Q	Tipo	Activo
Leche deslactosada	0	Leche	1
Leche descremada	0	Leche	1
Leche de soya	3	Leche	1
Leche de almendras	3	Leche	1
Crema batida	5	Crema	1
Cambio a tamaño mediano	3	Size	1
Cambio a tamaño grande	5	Size	1

Notas adicionales

Problemas que se enfrentaron

Para la mayoría logramos encontrar una solución a través de investigación o porque se ha tenido experiencia trabajando con personas que desarrollan software.

- ¿Cómo trabajar con interfaces gráficas? a nivel de flujo de trabajo, como equipo donde los involucrados deben avanzar en el proyecto pero no deben existir conflictos.
  - Usar dos archivos diferentes, uno dedicado a la gestión de datos (lectura y generación de archivos) y otro con la funcionalidad de la interfaz gráfica.
  - Avanzar en el proyecto y tener reuniones diarias para hablar de lo trabajado, además de avisar cuando estemos realizando cambios.
- ¿Qué es lo primero que se debe abordar en este tipo de proyectos?
  - A través de investigación y observación (de trabajos previos), se llegó a la conclusión que lo primero era hacer un diagrama con los menús que se iban a tener, esto ayudó a aterrizar lo que se necesitaría.

Herramientas utilizadas

Tkinter: para la generación de la interfaz gráfica.

> El paquete `tkinter` («interfaz Tk») es la interfaz por defecto de Python para el kit de herramientas de GUI Tk. - Python

- GUI: Interfaces gráficas de usuario. Parte visual de un programa con la que interactúa un usuario mediante elementos gráficos en lugar de comandos.

- Tkinter es como un puente entre Python y Tk, este último es un kit de herramientas para generar ventanas, botones, menús, etc.

▼ **ChatGPT**

- Detección de errores que a simple vista o que durante las jornadas de generación de código se pasan por alto.

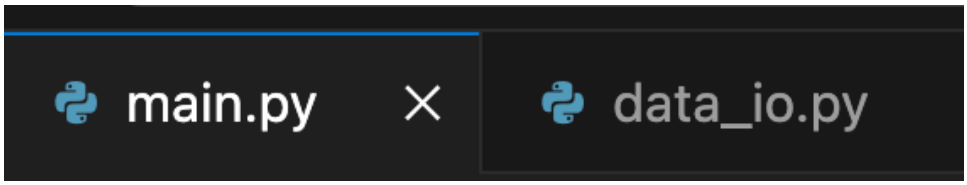
  ▼ Ejemplo: que falte un “guión” que es parte del nombre de una función.

- Recomendaciones de cómo abordar el flujo de trabajo.

  Ejemplos:

  - Trabajar en dos archivos separados (uno de datos y otro de funcionalidad de interfaz gráfica).

    ▼ Ejemplo



  - Trabajar main con datos de prueba y cuando ya están las funciones en datos solo se reemplaza (de esta manera ya no es necesario esperar a que la otra persona finalice)

    ▼ Ejemplo

```
# ======== Datos de prueba, luego se cambia a leer des
de CSV
# BEBIDAS = [
#   {"id_bebida": "LATTE-12", "nombre": "Latte 12oz", "prec
ios_base": 22.0, "categoría": "espresso", "activo": 1},
#   {"id_bebida": "ESP-DBL", "nombre": "Espresso doble",
"precios_base": 15.0, "categoría": "espresso", "activo": 1},
#   {"id_bebida": "CAP-12", "nombre": "Cappuccino 12oz",
"precios_base": 22.0, "categoría": "espresso", "activo": 1},
# ]

# MODIFICADORES = [
#   {"id_modificador": "LECHE-ENT", "nombre": "Leche ent
era", "tipo": "leche", "ajuste_precio": 0.0,"activo": 1},
#   {"id_modificador": "LECHE-COCO", "nombre": "Leche d
e coco", "tipo": "leche", "ajuste_precio": 3.0, "activo": 1},
#   {"id_modificador": "SHOT-DBL", "nombre": "Doble sho
t", "tipo": "shot", "ajuste_precio": 3.0, "activo": 1},
#   {"id_modificador": "JAR-CAR", "nombre": "Jarabe caram
elo", "tipo": "jarabe", "ajuste_precio": 2.0,"activo": 1},
# ]

# BAKERY = [
#   {"id_producto": "PROD-001", "nombre": "Pastelito 1", "c
ategoría": "pastel azul", "existencias": 48, "precio_unitari
o": 6.0, "activo": 1},
#   {"id_producto": "PROD-001", "nombre": "Pastelito 2", "c
ategoría": "pastel rojo", "existencias": 12, "precio_unitario":
8.0, "activo": 1},
# ]
# ======== Fin de datos de prueba

pedido_actual = {
    "fecha_hora": datetime.datetime.now().strftime("%Y-%
m-%d %H:%M:%S"),
```

```
    "lineas": [],
    "total": 0.0}
id_linea_actual = 0

BEBIDAS = data_io.leer_bebidas()
BAKERY = data_io.leer_bakery()
MODIFICADORES = data_io.leer_modificadores()
```

- **Python:** lenguaje de programación utilizado para el proyecto.
  - **Visual studio code:** entorno de desarrollo (software que integra herramientas para escribir, ejecutar y depurar código).
- **Github y git:** gestión del repositorio (y trabajo colaborativo) y control de versiones.
- **Notion:** documentación/presentación del proyecto.
- **Zoom:** reuniones virtuales fuera del Banco.

▼ **Fases de trabajo (generalizado)**

1. Investigación sobre cómo abordar proyectos con interfaces gráficas
2. Diagrama de navegación (menús)
   a. Una forma de aterrizar lo que se requiere a nivel de archivos y navegación a través de los menús
   b. Sirve para el esquema de carpetas y archivos
3. Esquema de carpetas y archivos (con base al diagrama de navegación)
4. Asignación de objetivos por integrante
   - Berni - gestión de datos [**archivo:** data_io.py]
     - io = input/output
   - Rita - funcionalidad con la interfaz gráfica [**archivo:** main.py]
5. En forma paralela se trabajaron ambos objetivos, utilizando listas con datos de prueba en main, y se reemplazaron posteriormente con las

funciones definidas en data_io

6. Se realizaron pruebas con cada fase de la interfaz (menú principal, menú bebidas, etc)
7. Integración de ambos objetivos (reemplazar datos de prueba, con las funciones de data_io)
8. Pruebas finales
9. Documentación para presentación de proyecto

Guía Tkinter

Guía Tkinter

Glosario de Tkinter usado en el proyecto

Elemento	Qué es / Qué hace	Dónde se usa en tu código
`tk.Tk()`	Crea la **ventana principal** de la app.	Al final: `ventana = tk.Tk()`
`tk.Toplevel()`	Crea **ventanas secundarias** (modales o auxiliares) sobre la principal.	`abrir_ventana_pedido`, `mostrar_bebidas`, `ventana_bebida_con_modificadores`, `mostrar_bakery`, `ver_pedido`, `pagar_pedido`, `mostrar_reporte_dia`
`.title(str)`	Cambia el **título** de la ventana.	En todas las ventanas (principal y toplevels).
`.geometry("WxH")`	Define **tamaño** de la ventana (opcional).	En principal y `abrir_ventana_pedido`.
`tk.Label(parent, text=..., font=...)`	**Texto** en pantalla (etiquetas).	En todas las pantallas para títulos y descripciones.
`tk.Button(parent, text=..., command=...)`	**Botón** que ejecuta una función al hacer clic.	Menú principal y todos los flujos (agregar, pagar, cancelar, etc.).
`tk.Checkbutton(parent, text=..., variable=..., command=...)`	Casilla de verificación para **activar/desactivar** opciones.	En `ventana_bebida_con_modificadores` para seleccionar modificadores.
`tk.IntVar()`	**Variable entera** ligada a un widget (0/1 en checkbuttons).	Para saber si un modificador está seleccionado.
`tk.DoubleVar(value=...)`	**Variable float** ligada a widgets; permite	`subtotal_var` para recalcular y mostrar el subtotal de la bebida.

Elemento	Qué es / Qué hace	Dónde se usa en tu código
	refrescar valores al vuelo.	
`.pack(...)`	**Administrador de geometría** simple; apila widgets verticalmente.	En la mayoría de ventanas (títulos, botones simples).
`.grid(row=..., column=...)`	Administrador de geometría por **rejilla** (filas/columnas).	En `mostrar_bebidas` y `mostrar_bakery` para botones en 4 columnas.
`messagebox.showinfo(...)`	**Diálogo informativo**.	Confirmaciones (“Producto agregado”, “Pago exitoso”, etc.).
`messagebox.showwarning(...)`	**Advertencia**.	Cuando no hay productos en el pedido.
`messagebox.askyesno(...)`	**Pregunta Sí/No** que devuelve `True/False`.	`cancelar_pedido()` y `cerrar_venta()` para confirmar acciones.
`.destroy()`	**Cierra** una ventana toplevel.	Tras confirmar acciones (p. ej., al agregar bebida o pagar).
`.quit()`	**Cierra** el loop de la app (sale del programa).	`salir()` en el botón “Salir”.
`.mainloop()`	Inicia el **bucle de eventos** de Tkinter.	Última línea del archivo.

Cómo se importa Tkinter y qué hace cada parte

Línea de importación	¿Qué hace?	¿Cómo se usa en el código?	Cuándo conviene
`import tkinter as tk`	Importa **todo el paquete Tkinter** y le pone el alias **`tk`** para escribir corto.	`ventana = tk.Tk()`, `tk.Label(...)`, `tk.Button(...)`, `tk.Toplevel(...)`, `tk.IntVar()`, `tk.DoubleVar()`	**Recomendado** como import principal. Mantiene el **namespace claro** (`tk.`) y evita choques de nombres.
`from tkinter import messagebox`	Importa **solo** el submódulo de **cuadros de diálogo** (info, warning, yes/no).	`messagebox.showinfo("Título","Texto")`, `messagebox.askyesno(...)`, `messagebox.showwarning(...)`	Cuando usarás diálogos. Más limpio que `tk.messagebox...` y fácil de leer.