UNIVERSIDAD ANÁHUAC QUERÉTARO

PI03: Tercera Entrega Intersemestral

Estructura de datos

Profesor:

Roberto Trejo

***Estudiante*:**

David Ceballos Mata	*00476577*
Luis Alberto Arias Llaguno	*00483701*
José Yael Bejar Jiménez	*00482719*

Fecha de entrega: 26/02/2025

Instrucciones de navegación dentro del programa:

1. Presiona el botón 'INSTRUCCIONES' para ver los detalles.
2. Presiona el botón 'CÓDIGO' para ver el código fuente.
3. Presiona 'SALIR' para cerrar la aplicación.
4. Puedes hacer scroll si hay más contenido abajo.

Introducción:

La lotería nacional es un juego tradicional que ha formado parte de nuestra cultura durante varias generaciones. En este proyecto, desarrollaremos en conjunto un program que simule dicho juego usando Python, implementado con una interfaz gráfica sencilla de usar con el uso de la biblioteca Tkinter y usando nuestros conocimientos de estructuras de datos para darle una funcionalidad correcta, así como óptima.

Como ya mencionamos, uno de los objetivos es integrar estructuras como listas, diccionarios y colas, para manejar el mazo ded 54 cartas de manera aleatoria, generar plantillas e ir descartando las cartas de las plantillas cuando salen del mazo principal. A su vez, emplearemos nuestros conocimientos en Programación Orientada a Objetos, para que la programación sea modular, reutilizable y facilite la colaboración del equipo.

Historia:

La lotería mexicana es un símbolo de identidad, una tradición que ha unido a familias y amigos durante generaciones, con sus carismáticas cartas y su divertida dinámica. El juego es un espectáculo de emoción, que requiere de la rapidez de las personas, provocando risas e increíbles momentos. Toda esta dinámica es acompañada del inconfundible canto del gritón, quien anuncia cada carta con un ingenio poético y picardía que le da vida a la partida.

Originaria del siglo XVIII, la lotería mexicana ha perdurado como parte esencial de la cultura popular mexicana. En fiestas, ferias y reuniones, este juego es capaz de crear momentos inolvidables, donde la estrategia, la suerte y la diversión se entrelazan en cada ronda para poder determinar a un ganador.

¿Cómo se juega?

1. Preparar el juego

- Se juega con un **mazo de 54 cartas**, cada una con una imagen diferente.

- Cada jugador recibe un **tablero (o tabla)** con una selección de 16 imágenes diferentes.
- Se necesitan fichas, frijoles o cualquier objeto pequeño para marcar las cartas en la tabla.

2. Dinámica del Juego

- En este caso, por ser una versión electrónica de la lotería, quien se encargará de sacar y anunciar cada carta del mazo será la propia computadora.
- Los jugadores deberán buscar la carta anunciada en sus tableros y, si la tienen, la marcan con una ficha.

3. Objetivo y Condiciones de Victoria

- El primer jugador en completar un patrón específico gana la partida.
- Los patrones pueden variar según las reglas establecidas antes de comenzar el juego. Algunos ejemplos comunes son:
    - **Línea:** completar una fila, columna o diagonal.
    - **Cuatro Esquinas:** marcar las cuatro esquinas del tablero.
    - **Tablero Completo:** marcar todas las cartas de la tabla.

¿Por qué 54 cartas?

Consideramos que este número de cartas permite tener un abanico variado de combinaciones para que el juego sea dinámico, sin volverlo tan complejo. Existen varias versiones de la baraja, sin embargo, la más popular sigue siendo la original.

Programa Fuente

Código Portada

```
import tkinter as tk

from PIL import Image, ImageTk

import Instrucciones

import Programa_principal

from pathlib import Path
```

```
   #Este es el main

   class Portada:

       def __init__(self, ventana):

           self.ventana = ventana

           self.ventana.title("Portada")

           self.ventana.geometry("1920x1080+200+1") #Ajustar la posición
de la ventana (lo que dice +300+1 es dónde aparecerá la ventana)

           self.ventana.state("zoomed")

           self.raiz = Path(__file__).parent.resolve()



           self.canvas = tk.Canvas(self.ventana, width=1920, height=1080)

           self.canvas.pack()



           self.portada = Image.open(self.raiz / "imagenes" /
"PortadaLot.png")

           self.portadaResized = self.portada.resize((1545, 800))

           self.portadaTK = ImageTk.PhotoImage(self.portadaResized)



           self.canvas.create_image(0, 0, anchor="nw",
image=self.portadaTK)




           #Creación de textos en el canvas

           textos = [
```

```
            (770, 270, "Proyecto Intersemestral", 18),
            (770, 300, "Fecha de entrega: 10 de marzo 2025", 18),
            (770, 330, "Universidad Anáhuac Querétaro", 18),
            (770, 360, "Portada Principal", 18),
            (770, 390, "Integrantes del equipo:", 14),
            (770, 420, "Luis Alberto Arias Llaguno - ID: 00483701",
12),
            (770, 450, "Jose Yael Bejar Jiménez - ID: 00482719", 12),
            (770, 480, "David Ceballos Mata - ID: 00476577", 12),
            (770, 510, "Estructura de datos", 18),
            (770, 540, "Profesor: Roberto Trejo", 18),
            (770, 570, "Lotería Mexicana", 14),
        ]

        for x, y, text, font_size in textos:
            self.canvas.create_text(x, y, text=text, font=("Arial",
font_size), anchor="center")

        #Imagen de los integrantes
        self.foto_original = Image.open(self.raiz / "imagenes" /
"fotointegrantes.png")
        self.foto_resized = self.foto_original.resize((300, 200))
        self.foto_Tk = ImageTk.PhotoImage(self.foto_resized)
```

```
            self.label_foto = tk.Label(self.ventana, image=self.foto_Tk,
bg="white")

            self.label_foto.image = self.foto_Tk

            self.label_foto.place(x=1100, y=340)



            #Botones

            instr_path = self.raiz / "imagenes" / "Instrucciones.png"

            self.img_instr = tk.PhotoImage(file=str(instr_path))

            self.img_instr_resized = self.img_instr.subsample(10, 10)

            self.btn_instrucciones = tk.Button(self.ventana,
image=self.img_instr_resized, command=self.abrir_instrucciones,
bg="#46ADF4", fg="white")

            self.btn_instrucciones.place(x=1200, y=720)



            jugar_path = self.raiz / "imagenes" / "Jugar.png"

            self.img_jugar = tk.PhotoImage(file=str(jugar_path))

            self.img_jugar_resized = self.img_jugar.subsample(10, 10)

            self.btn_codigo = tk.Button(self.ventana,
image=self.img_jugar_resized, command=self.abrir_programa, bg="#00CC47",
fg="white")

            self.btn_codigo.place(x=1270, y=720)



            salir_path = self.raiz / "imagenes" / "Salir.png"

            self.img_salir = tk.PhotoImage(file=str(salir_path))

            self.img_salir_resized = self.img_salir.subsample(10, 10)
```

```
            self.img_salir = tk.PhotoImage(file=str(salir_path))

            self.btn_salir = tk.Button(self.ventana,
image=self.img_salir_resized, command=lambda: ventana.quit(),
bg="#F55757", fg="white")

            self.btn_salir.place(x=1360, y=720)



        def abrir_instrucciones(self):

            self.ventana.withdraw()

            n_ventana = tk.Toplevel(self.ventana)

            Instrucciones.Instrucciones(n_ventana)



        def abrir_programa(self):

            self.ventana.withdraw()

            n_ventana = tk.Toplevel(self.ventana)

            Programa_principal.Loteria(n_ventana)



    if __name__ == "__main__":

        ventana = tk.Tk()

        app = Portada(ventana)

        ventana.mainloop()
```

Código Instrucciones

```
import tkinter as tk

from PIL import Image, ImageTk

import Portada

import Programa_principal

from pathlib import Path



class Instrucciones:

    def __init__(self, ventana):

        self.ventana = ventana

        self.ventana.title("Instrucciones")

        self.ventana.geometry("1920x1080+200+1")

        self.ventana.state("zoomed")

        self.raiz = Path(__file__).parent.resolve()



        # Canvas grande con la imagen de fondo

        self.canvasFondo = tk.Canvas(self.ventana, width=1920,
height=1080, bg="grey")

        self.canvasFondo.pack(fill="both", expand=True)



        self.fondo = Image.open(self.raiz / "imagenes" /
"FondoLot.png")

        self.fondoResized = self.fondo.resize((1545, 800))

        self.fondoTK = ImageTk.PhotoImage(self.fondoResized)
```

```
            self.canvasFondo.create_image(0, 0, anchor="nw",
image=self.fondoTK)



            # Canvas blanco (800x400) con Scrollbar

            self.frame_canvas = tk.Frame(self.ventana, bg="white",
width=800, height=400)

            self.frame_canvas.place(x=380, y=200)  # Centrar el Canvas en
la pantalla



            # Crear Canvas dentro del frame para que tenga scrollbar

            self.canvas = tk.Canvas(self.frame_canvas, width=800,
height=400, bg="white")

            self.scrollbar = tk.Scrollbar(self.frame_canvas,
orient="vertical", command=self.canvas.yview)



            # Configurar el canvas y la scrollbar

            self.canvas.configure(yscrollcommand=self.scrollbar.set)

            self.canvas.pack(side="left", fill="both", expand=True)

            self.scrollbar.pack(side="right", fill="y")



            # Frame que contendrá el contenido desplazable

            self.frame_contenido = tk.Frame(self.canvas, bg="white")

            self.canvas.create_window((415, 0),
window=self.frame_contenido, anchor="n")
```

```
            self.instrucciones = Image.open(self.raiz / "imagenes" /
"InstruccionesLotFinal.jpg")

            self.instruccionesTk = ImageTk.PhotoImage(self.instrucciones)



            self.labelImagen = tk.Label(self.frame_contenido,
image=self.instruccionesTk, bg="white")

            self.labelImagen.pack(pady=10)



            # Agregar contenido al frame_contenido

            '''for i in range(30):  # Simulación de muchos textos para
probar el scroll

                tk.Label(self.frame_contenido, text=f"Instrucción {i+1}",
bg="white", font=("Arial", 12)).pack(pady=5)'''



            # Actualizar el área de desplazamiento

            self.frame_contenido.update_idletasks()

            self.canvas.config(scrollregion=self.canvas.bbox("all"))



            #  Botones

            port_path = self.raiz / "imagenes" / "Portada.png"

            self.img_port = tk.PhotoImage(file=str(port_path))

            self.img_port_resized = self.img_port.subsample(10, 10)

            self.btn_portada = tk.Button(self.ventana,
image=self.img_port_resized, command=self.abrir_portada, bg = "#B2B2B2",
fg = "white")

            self.btn_portada.place(x=1200, y=720)
```

```
            jugar_path = self.raiz / "imagenes" / "Jugar.png"

            self.img_jugar = tk.PhotoImage(file=str(jugar_path))

            self.img_jugar_resized = self.img_jugar.subsample(10, 10)

            self.btn_codigo = tk.Button(self.ventana,
image=self.img_jugar_resized, command=self.abrir_programa, bg="#00CC47",
fg="white")

            self.btn_codigo.place(x=1275, y=720)



            salir_path = self.raiz / "imagenes" / "Salir.png"

            self.img_salir = tk.PhotoImage(file=str(salir_path))

            self.img_salir_resized = self.img_salir.subsample(10, 10)

            self.img_salir = tk.PhotoImage(file=str(salir_path))

            self.btn_salir = tk.Button(self.ventana,
image=self.img_salir_resized, command=lambda: ventana.quit(),
bg="#F55757", fg="white")

            self.btn_salir.place(x=1360, y=720)



            # Configurar el scroll con la rueda del mouse

            self.canvas.bind("<Configure>", self.actualizar_scroll)

            self.canvas.bind_all("<MouseWheel>", self.scroll_con_rueda)



        def actualizar_scroll(self, event):

            self.canvas.config(scrollregion=self.canvas.bbox("all"))
```

```
    def scroll_con_rueda(self, event):
        self.canvas.yview_scroll(-1 * (event.delta // 120), "units")


    def abrir_portada(self):
        self.ventana.destroy()
        n_ventana = tk.Toplevel()
        Portada.Portada(n_ventana)


    def abrir_programa(self):
        self.ventana.destroy()
        n_ventana = tk.Toplevel()
        Programa_principal.Loteria(n_ventana)
```

Código Principal (Avance)

```
import tkinter as tk
from PIL import Image, ImageTk
import tkinter.messagebox as messagebox
import time
import random
import winsound
import Portada
import Instrucciones
```

```
from pathlib import Path


class Carta:

    def __init__(self, nombre, imagen):

        self.nombre = nombre

        self.imagen = imagen


class Jugador:

    def __init__(self, nombre, cartas_disponibles):

        self.nombre = nombre

        self.tablero = random.sample(cartas_disponibles, 16)

        self.cartas_sacadas = 0

        self.imagenes = []

        self.victorias = 0


class Loteria:

    def __init__(self, ventana):

        self.ventana = ventana

        self.ventana.title("Lotería Nacional")

        self.ventana.geometry("1920x1080+200+1")

        self.ventana.state("zoomed")

        self.raiz = Path(__file__).parent.resolve()
```

```
            # Fondo
            self.canvasFondo = tk.Canvas(self.ventana, width=1920,
height=1080, bg="grey")
            self.canvasFondo.pack(fill="both", expand=True)


            self.fondo = Image.open(self.raiz / "imagenes" /
"PortadaLot.png")
            self.fondoResized = self.fondo.resize((1545, 800))
            self.fondoTK = ImageTk.PhotoImage(self.fondoResized)
            self.canvasFondo.create_image(0, 0, anchor="nw",
image=self.fondoTK)


            # Área de juego
            self.frame_canvas = tk.Frame(self.ventana, bg="white",
width=1480, height=650)
            self.frame_canvas.place(x=30, y=60)  # Posición en la
ventana


            self.canvas = tk.Canvas(self.frame_canvas, width=1480,
height=650, bg="white")
            self.canvas.pack()


            # Frame para los elementos de jugadores
            self.frame_jugadores = tk.Frame(self.canvas, bg="white")
            self.canvas.create_window((740, 50),
window=self.frame_jugadores, anchor="n")
```

```
            # Variables
            self.num_jugadores = 2
            self.max_jugadores = 5
            self.min_jugadores = 2
            self.entradas_jugadores = []

            # UI de selección de jugadores
            self.crear_ui_jugadores()

            # Botones
            port_path = self.raiz / "imagenes" / "Portada.png"
            self.img_port = tk.PhotoImage(file=str(port_path))
            self.img_port_resized = self.img_port.subsample(10, 10)
            self.btn_portada = tk.Button(self.ventana,
image=self.img_port_resized, command=self.abrir_portada, bg = "#B2B2B2",
fg = "white")
            self.btn_portada.place(x=1200, y=720)

            instr_path = self.raiz / "imagenes" / "Instrucciones.png"
            self.img_instr = tk.PhotoImage(file=str(instr_path))
            self.img_instr_resized = self.img_instr.subsample(10, 10)
            self.btn_instrucciones = tk.Button(self.ventana,
image=self.img_instr_resized, command=self.abrir_instrucciones,
bg="#46ADF4", fg="white")
```

```
            self.btn_instrucciones.place(x=1285, y=720)



            salir_path = self.raiz / "imagenes" / "Salir.png"

            self.img_salir = tk.PhotoImage(file=str(salir_path))

            self.img_salir_resized = self.img_salir.subsample(10, 10)

            self.img_salir = tk.PhotoImage(file=str(salir_path))

            self.btn_salir = tk.Button(self.ventana,
image=self.img_salir_resized, command=lambda: ventana.quit(),
bg="#F55757", fg="white")

            self.btn_salir.place(x=1360, y=720)



            # Cargar la imagen del "frijol"

            self.img_frijol = Image.open(self.raiz / "imagenes" /
"frijol.png").resize((45, 45))

            self.img_frijol_tk = ImageTk.PhotoImage(self.img_frijol)



            self.cartas = [

                Carta("El gallo", self.raiz / "imagenes" /
"01Gallo.jpg"),

                Carta("El diablito", self.raiz / "imagenes" /
"02Diablito.jpg"),

                Carta("La dama", self.raiz / "imagenes" / "03Dama.jpg"),

                Carta("El catrín", self.raiz / "imagenes" /
"04Catrin.jpg"),

                Carta("El paraguas", self.raiz / "imagenes" /
"05Paraguas.jpg"),
```

```
                Carta("La sirena", self.raiz / "imagenes" /
"06Sirena.jpg"),

                Carta("La escalera", self.raiz / "imagenes" /
"07Escalera.jpg"),

                Carta("La botella", self.raiz / "imagenes" /
"08Botella.jpg"),

                Carta("El barril", self.raiz / "imagenes" /
"09Barril.jpg"),

                Carta("El árbol", self.raiz / "imagenes" /
"10Arbol.jpg"),

                Carta("El melón", self.raiz / "imagenes" /
"11Melon.jpg"),

                Carta("El valiente", self.raiz / "imagenes" /
"12Valiente.jpg"),

                Carta("El gorrito", self.raiz / "imagenes" /
"13Gorrito.jpg"),

                Carta("La muerte", self.raiz / "imagenes" /
"14Muerte.jpg"),

                Carta("La pera", self.raiz / "imagenes" / "15Pera.jpg"),

                Carta("La bandera", self.raiz / "imagenes" /
"16Bandera.jpg"),

                Carta("El bandolón", self.raiz / "imagenes" /
"17Bandolon.jpg"),

                Carta("El violoncello", self.raiz / "imagenes" /
"18Violoncello.jpg"),

                Carta("La garza", self.raiz / "imagenes" /
"19Garza.jpg"),

                Carta("El pájaro", self.raiz / "imagenes" /
"20Pajaro.jpg"),
```

```
                Carta("La mano", self.raiz / "imagenes" / "21Mano.jpg"),
                Carta("La bota", self.raiz / "imagenes" / "22Bota.jpg"),
                Carta("La luna", self.raiz / "imagenes" / "23Luna.jpg"),
                Carta("El cotorro", self.raiz / "imagenes" /
"24Cotorro.jpg"),
                Carta("El borracho", self.raiz / "imagenes" /
"25Borracho.jpg"),
                Carta("El negrito", self.raiz / "imagenes" /
"26Negrito.jpg"),
                Carta("El corazón", self.raiz / "imagenes" /
"27Corazon.jpg"),
                Carta("La sandía", self.raiz / "imagenes" /
"28Sandia.jpg"),
                Carta("El tambor", self.raiz / "imagenes" /
"29Tambor.jpg"),
                Carta("El camarón", self.raiz / "imagenes" /
"30Camaron.jpg"),
                Carta("Las jaras", self.raiz / "imagenes" /
"31Jaras.jpg"),
                Carta("El músico", self.raiz / "imagenes" /
"32Musico.jpg"),
                Carta("La araña", self.raiz / "imagenes" /
"33Arana.jpg"),
                Carta("El soldado", self.raiz / "imagenes" /
"34Soldado.jpg"),
                Carta("La estrella", self.raiz / "imagenes" /
"35Estrella.jpg"),
                Carta("El cazo", self.raiz / "imagenes" / "36Cazo.jpg"),
```

```
                Carta("El mundo", self.raiz / "imagenes" /
"37Mundo.jpg"),

                Carta("El apache", self.raiz / "imagenes" /
"38Apache.jpg"),

                Carta("El nopal", self.raiz / "imagenes" /
"39Nopal.jpg"),

                Carta("El alacrán", self.raiz / "imagenes" /
"40Alacran.jpg"),

                Carta("La rosa", self.raiz / "imagenes" / "41Rosa.jpg"),

                Carta("La calavera", self.raiz / "imagenes" /
"42Calavera.jpg"),

                Carta("La campana", self.raiz / "imagenes" /
"43Campana.jpg"),

                Carta("El cantarito", self.raiz / "imagenes" /
"44Cantarito.jpg"),

                Carta("El venado", self.raiz / "imagenes" /
"45Venado.jpg"),

                Carta("El sol", self.raiz / "imagenes" / "46Sol.jpg"),

                Carta("La corona", self.raiz / "imagenes" /
"47Corona.jpg"),

                Carta("La chalupa", self.raiz / "imagenes" /
"48Chalupa.jpg"),

                Carta("El pino", self.raiz / "imagenes" / "49Pino.jpg"),

                Carta("El pescado", self.raiz / "imagenes" /
"50Pescado.jpg"),

                Carta("La palma", self.raiz / "imagenes" /
"51Palma.jpg"),

                Carta("La maceta", self.raiz / "imagenes" /
"52Maceta.jpg"),
```

```
                Carta("El arpa", self.raiz / "imagenes" / "53Arpa.jpg"),
                Carta("La rana", self.raiz / "imagenes" / "54Rana.jpg"),
            ]



        def crear_ui_jugadores(self):
            """Crea o actualiza la interfaz de selección de jugadores."""
            for widget in self.frame_jugadores.winfo_children():
                widget.destroy()

            # Label de número de jugadores
            tk.Label(self.frame_jugadores, text="Número de jugadores:", font=("Arial", 14), bg="white").grid(row=0, column=0, columnspan=3, pady=10)

            # Botón para disminuir
            btn_menos = tk.Button(self.frame_jugadores, text="-", font=("Arial", 14), width=3, command=self.disminuir_jugadores)
            btn_menos.grid(row=1, column=0)

            # Mostrar número de jugadores
            self.label_num_jugadores = tk.Label(self.frame_jugadores, text=str(self.num_jugadores), font=("Arial", 14), bg="white")
            self.label_num_jugadores.grid(row=1, column=1, padx=10)
```

```
            # Botón para aumentar
            btn_mas = tk.Button(self.frame_jugadores, text="+",
font=("Arial", 14), width=3, command=self.aumentar_jugadores)
            btn_mas.grid(row=1, column=2)

            # Campos para los nombres
            self.entradas_jugadores = []
            for i in range(self.num_jugadores):
                tk.Label(self.frame_jugadores, text=f"Jugador {i + 1}:",
font=("Arial", 12), bg="white").grid(row=i + 2, column=0, padx=10, pady=5,
sticky="w")
                entrada = tk.Entry(self.frame_jugadores, font=("Arial",
12), width=20)
                entrada.grid(row=i + 2, column=1, columnspan=2, padx=10,
pady=5)
                self.entradas_jugadores.append(entrada)

            # Botón "JUGAR"
            self.btn_jugar = tk.Button(self.frame_jugadores,
text="JUGAR", font=("Arial", 14), bg="green", fg="white",
command=self.iniciar_juego)
            self.btn_jugar.grid(row=self.num_jugadores + 2, column=0,
columnspan=3, pady=20)

        def iniciar_juego(self):
            # Crear jugadores con tableros
```

```
            nombres = [entrada.get() if entrada.get() else f"Jugador {i+1}" for i, entrada in enumerate(self.entradas_jugadores)]
            self.jugadores = [Jugador(nombre, self.cartas) for nombre in nombres]

            for jugador in self.jugadores:
                print(f"Tablero de {jugador.nombre}:")
                for i, carta in enumerate(jugador.tablero):
                    print(f"{i+1}. {carta.nombre}")
                print("-" * 30)

            """Oculta la interfaz de selección de jugadores y prepara los tableros."""
            self.frame_jugadores.destroy()
            self.dibujar_tableros()

        def dibujar_tableros(self):
            self.canvas.delete("victoria_texto")
            self.canvas.delete("ganador")

            """Distribuye y dibuja los tableros de los jugadores en el canvas."""
            posiciones = {
                2: [(320, 320), (890, 320)],
```

```
                3: [(200, 320), (550, 320), (900, 320)],
                4: [(200, 335), (450, 335), (700, 335), (950, 335)],
                5: [(320, 180), (900, 180), (620, 320), (320, 490),
(900, 490)],
            }

            # Tamaño dinámico de cartas
            if self.num_jugadores in [2,3]:
                carta_ancho, carta_alto = 75, 112
                resta_y = 265
                suma_y = 255
            elif self.num_jugadores in [4]:
                carta_ancho, carta_alto = 50, 75
                resta_y = 190
                suma_y = 180
            else:
                carta_ancho, carta_alto = 35, 52
                resta_y = 135
                suma_y = 130

            espacio_x = carta_ancho + 5
            espacio_y = carta_alto + 5
```

```
            for i, jugador in enumerate(self.jugadores):
                x_centro, y_centro = posiciones[self.num_jugadores][i]

                # Dibujar nombre
                self.canvas.create_text(x_centro, y_centro - resta_y,
text=jugador.nombre, font=("Arial", 14, "bold"), fill="black")

                # Dibujar victorias
                self.canvas.create_text(x_centro, y_centro + suma_y,
text=f"Victorias: {jugador.victorias}", font=("Arial", 14, "bold"),
fill="black", tags="victoria_texto")

                # Dibujar tablero 4x4
                x_inicio = x_centro - (2 * espacio_x)
                y_inicio = y_centro - (2 * espacio_y)

                # Almacenar las referencias de las imágenes en una lista
dentro del jugador
                jugador.imagenes = []

                for fila in range(4):
                    for col in range(4):
                        carta = jugador.tablero[fila * 4 + col]
                        img =
Image.open(carta.imagen).resize((carta_ancho, carta_alto))
```

```
                        img_tk = ImageTk.PhotoImage(img)

                        # Guardar la imagen en la lista del jugador
                        jugador.imagenes.append(img_tk)

                        # Dibujar la imagen en el canvas
                        self.canvas.create_image(x_inicio + col * espacio_x, y_inicio + fila * espacio_y, image=img_tk, anchor="nw")

            self.img_carta_atras = Image.open(self.raiz / "imagenes" / "cartaAtras.jpg").resize((200, 300))
            self.img_carta_atras_tk = ImageTk.PhotoImage(self.img_carta_atras)

            # Dibujar la imagen en el canvas
            self.canvas.create_image(1275, 325, image=self.img_carta_atras_tk, anchor="center")

            # Crear un Frame para los botones
            frame_botones = tk.Frame(self.canvas, bg="white")

            # Botón "CARTA"
            self.btn_carta = tk.Button(frame_botones, text="CARTA", font=("Arial", 14), bg="green", fg="white", command=self.sacar_carta)
            self.btn_carta.grid(row=0, column=0, pady=10)
```

```
            # Botón "JUGAR DE NUEVO"
            self.btn_jugar_nuevo = tk.Button(frame_botones, text="JUGAR DE NUEVO", font=("Arial", 14), bg="blue", fg="white", command=self.jugar_de_nuevo)
            self.btn_jugar_nuevo.grid(row=1, column=0, pady=5)

            # Insertar el Frame con los botones en el canvas
            self.canvas.create_window(1275, 535, window=frame_botones)

        def sacar_carta(self):
            """Saca una carta aleatoria del mazo y la muestra en el área de cartas."""
            if not self.cartas:  # Si ya no hay cartas disponibles
                print("No quedan más cartas en el mazo.")
                return

            # Elegir una carta aleatoria y eliminarla del mazo
            carta = random.choice(self.cartas)
            self.cartas.remove(carta)

            # Cargar la nueva imagen con el mismo tamaño
            img_carta = Image.open(carta.imagen).resize((200, 300))
            self.img_carta_actual_tk = ImageTk.PhotoImage(img_carta)  # Guardar referencia
```

```
            # Eliminar la imagen anterior (carta volteada) y dibujar la
nueva carta
            self.canvas.delete("carta_actual")  # Elimina cualquier
carta anterior
            self.canvas.create_image(1275, 325,
image=self.img_carta_actual_tk, anchor="center", tags="carta_actual")


            if self.num_jugadores in [2,3]:
                carta_ancho, carta_alto = 75, 112
            elif self.num_jugadores in [4]:
                carta_ancho, carta_alto = 50, 75
            else:
                carta_ancho, carta_alto = 35, 52


            espacio_x = carta_ancho + 5
            espacio_y = carta_alto + 5


            ganadores = []


            # Comprobar si la carta sacada está en los tableros de los
jugadores
            for jugador in self.jugadores:
                for fila in range(4):
                    for col in range(4):
```

```
                        carta_tablero = jugador.tablero[fila * 4 + col]
                        if carta_tablero.nombre == carta.nombre:
                            # Si la carta está en el tablero, colocar el
"frijol"
                            x_centro, y_centro =
self.obtener_posicion_tablero(jugador)
                            x_inicio = x_centro - (2 * espacio_x)
                            y_inicio = y_centro - (2 * espacio_y)
                            self.canvas.create_image(x_inicio + col *
espacio_x + carta_ancho / 2,
                                y_inicio + fila * espacio_y + carta_alto
/ 2,
                                image=self.img_frijol_tk,
                                anchor="center")

                            # Incrementar el conteo de cartas sacadas
del jugador
                            jugador.cartas_sacadas += 1

                            # Verificar si el jugador ha ganado (tiene
todas las 16 cartas)
                            if jugador.cartas_sacadas == 16:
                                ganadores.append(jugador)
                                self.btn_carta.config(state="disabled")
# Deshabilitar el botón de sacar carta
                                #return
```

```
            if ganadores:
                if len(ganadores) > 1:
                    self.canvas.create_text(1275, 130, text=f"¡Hay
empate!", font=("Arial", 20, "bold"), fill="green", tags="ganador")
                else:
                    self.canvas.create_text(1275, 130,
text=f"¡{ganadores[0].nombre} ha ganado!", font=("Arial", 20, "bold"),
fill="green", tags="ganador")

                for jugador in ganadores:
                    jugador.victorias += 1  # Aumentar victorias de los
ganadores

        def jugar_de_nuevo(self):
            """Reinicia el juego sin borrar las victorias."""
            # Reiniciar el estado de los tableros y la baraja
            self.cartas = [
                Carta("El gallo", self.raiz / "imagenes" /
"01Gallo.jpg"),
                Carta("El diablito", self.raiz / "imagenes" /
"02Diablito.jpg"),
                Carta("La dama", self.raiz / "imagenes" / "03Dama.jpg"),
                Carta("El catrín", self.raiz / "imagenes" /
"04Catrin.jpg"),
                Carta("El paraguas", self.raiz / "imagenes" /
"05Paraguas.jpg"),
```

```
                Carta("La sirena", self.raiz / "imagenes" /
"06Sirena.jpg"),

                Carta("La escalera", self.raiz / "imagenes" /
"07Escalera.jpg"),

                Carta("La botella", self.raiz / "imagenes" /
"08Botella.jpg"),

                Carta("El barril", self.raiz / "imagenes" /
"09Barril.jpg"),

                Carta("El árbol", self.raiz / "imagenes" /
"10Arbol.jpg"),

                Carta("El melón", self.raiz / "imagenes" /
"11Melon.jpg"),

                Carta("El valiente", self.raiz / "imagenes" /
"12Valiente.jpg"),

                Carta("El gorrito", self.raiz / "imagenes" /
"13Gorrito.jpg"),

                Carta("La muerte", self.raiz / "imagenes" /
"14Muerte.jpg"),

                Carta("La pera", self.raiz / "imagenes" / "15Pera.jpg"),

                Carta("La bandera", self.raiz / "imagenes" /
"16Bandera.jpg"),

                Carta("El bandolón", self.raiz / "imagenes" /
"17Bandolon.jpg"),

                Carta("El violoncello", self.raiz / "imagenes" /
"18Violoncello.jpg"),

                Carta("La garza", self.raiz / "imagenes" /
"19Garza.jpg"),

                Carta("El pájaro", self.raiz / "imagenes" /
"20Pajaro.jpg"),
```

```
                Carta("La mano", self.raiz / "imagenes" / "21Mano.jpg"),
                Carta("La bota", self.raiz / "imagenes" / "22Bota.jpg"),
                Carta("La luna", self.raiz / "imagenes" / "23Luna.jpg"),
                Carta("El cotorro", self.raiz / "imagenes" /
"24Cotorro.jpg"),
                Carta("El borracho", self.raiz / "imagenes" /
"25Borracho.jpg"),
                Carta("El negrito", self.raiz / "imagenes" /
"26Negrito.jpg"),
                Carta("El corazón", self.raiz / "imagenes" /
"27Corazon.jpg"),
                Carta("La sandía", self.raiz / "imagenes" /
"28Sandia.jpg"),
                Carta("El tambor", self.raiz / "imagenes" /
"29Tambor.jpg"),
                Carta("El camarón", self.raiz / "imagenes" /
"30Camaron.jpg"),
                Carta("Las jaras", self.raiz / "imagenes" /
"31Jaras.jpg"),
                Carta("El músico", self.raiz / "imagenes" /
"32Musico.jpg"),
                Carta("La araña", self.raiz / "imagenes" /
"33Arana.jpg"),
                Carta("El soldado", self.raiz / "imagenes" /
"34Soldado.jpg"),
                Carta("La estrella", self.raiz / "imagenes" /
"35Estrella.jpg"),
                Carta("El cazo", self.raiz / "imagenes" / "36Cazo.jpg"),
```

```
                Carta("El mundo", self.raiz / "imagenes" /
"37Mundo.jpg"),

                Carta("El apache", self.raiz / "imagenes" /
"38Apache.jpg"),

                Carta("El nopal", self.raiz / "imagenes" /
"39Nopal.jpg"),

                Carta("El alacrán", self.raiz / "imagenes" /
"40Alacran.jpg"),

                Carta("La rosa", self.raiz / "imagenes" / "41Rosa.jpg"),

                Carta("La calavera", self.raiz / "imagenes" /
"42Calavera.jpg"),

                Carta("La campana", self.raiz / "imagenes" /
"43Campana.jpg"),

                Carta("El cantarito", self.raiz / "imagenes" /
"44Cantarito.jpg"),

                Carta("El venado", self.raiz / "imagenes" /
"45Venado.jpg"),

                Carta("El sol", self.raiz / "imagenes" / "46Sol.jpg"),

                Carta("La corona", self.raiz / "imagenes" /
"47Corona.jpg"),

                Carta("La chalupa", self.raiz / "imagenes" /
"48Chalupa.jpg"),

                Carta("El pino", self.raiz / "imagenes" / "49Pino.jpg"),

                Carta("El pescado", self.raiz / "imagenes" /
"50Pescado.jpg"),

                Carta("La palma", self.raiz / "imagenes" /
"51Palma.jpg"),

                Carta("La maceta", self.raiz / "imagenes" /
"52Maceta.jpg"),
```

```
                Carta("El arpa", self.raiz / "imagenes" / "53Arpa.jpg"),
                Carta("La rana", self.raiz / "imagenes" / "54Rana.jpg"),
            ]

            for jugador in self.jugadores:
                jugador.cartas_sacadas = 0
                jugador.tablero = random.sample(self.cartas, 16)

            # Eliminar las imágenes de las cartas actuales en el canvas
            self.canvas.delete("carta_actual")

            # Redibujar los tableros
            self.dibujar_tableros()

            # Rehabilitar el botón "CARTA" si es necesario
            self.btn_carta.config(state="normal")


        def obtener_posicion_tablero(self, jugador):
            """Devuelve la posición (x, y) del tablero de un jugador."""
            posiciones = {
                2: [(320, 320), (890, 320)],
                3: [(200, 350), (550, 350), (900, 350)],
```

```
                    4: [(200, 335), (450, 335), (700, 335), (950, 335)],
                    5: [(320, 180), (900, 180), (620, 320), (320, 490),
(900, 490)],
                }
                x_centro, y_centro =
posiciones[self.num_jugadores][self.jugadores.index(jugador)]
                return x_centro, y_centro


        '''def declarar_ganador(self, jugador):
            """Declara al jugador como ganador y muestra un mensaje en
el canvas."""
            self.canvas.create_text(1275, 130, text=f"¡{jugador.nombre}
ha ganado!", font=("Arial", 20, "bold"), fill="green", tags="ganador")'''


        def aumentar_jugadores(self):
            """Aumenta el número de jugadores (máx. 5)."""
            if self.num_jugadores < self.max_jugadores:
                self.num_jugadores += 1

self.label_num_jugadores.config(text=str(self.num_jugadores))
                self.crear_ui_jugadores()


        def disminuir_jugadores(self):
            """Disminuye el número de jugadores (mín. 2)."""
            if self.num_jugadores > self.min_jugadores:
```

```
                self.num_jugadores -= 1

self.label_num_jugadores.config(text=str(self.num_jugadores))
                self.crear_ui_jugadores()

        # (No se elimina) Este no debería de eliminarse, es para abrir la portada
        def abrir_portada(self):
            #Cerrar la ventana actual y abrir la portada
            #self.jugar_de_nuevo()
            self.ventana.withdraw()
            n_ventana = tk.Toplevel(self.ventana)
            Portada.Portada(n_ventana)

        # (No se elimina) Este no debería de eliminarse, es para abrir las instrucciones
        def abrir_instrucciones(self):
            #self.jugar_de_nuevo()
            self.ventana.withdraw()
            n_ventana = tk.Toplevel(self.ventana)
            Instrucciones.Instrucciones(n_ventana)

        def cerrar_programa(self):
            self.ventana.quit()
```

```
            self.ventana.destroy()
```

¿Qué estructuras de datos estamos usando y por qué?

Las principales estructuras de datos que se van a usar en este proyecto son listas y matrices, debido a su gran potencial y flexibilidad en cuatro se habla de agregar nuevas cartas a un tablero. También creemos que es la mejor estructura de datos debido a la estructura de las plantillas de la lotería como tal, en el que se puede asignar un objeto de carta a cada uno de los espacios de la matriz y de esa forma hacerlo más eficiente. Aparte, para relacionar imágenes, audios y número de cartas usaremos diccionarios.

Conclusión

Conclusión Yael

Esta parte del proyecto pudimos aprender un poco más del uso del frontend de un programa, usando los labels, para las imágenes, al igual que pudimos usar botones para hacer navegación entre ventanas, esto nos ayuda a poder tener un programa más estético y dejar de lado el uso de la terminal. Le dimos una estética más colorida con imágenes hechas en canvas para poder agregarle paz estética y mejor presentación, al igual que agregamos instrucciones de uso del juego para que quien corra el programa pueda comprender el juego.

Con esta penúltima entrega le dimos un formato distinto y más estético a la aplicación, le cambiamos los botones por un formato sin texto, usando imágenes para poder hacerlo un poco más intuitivo, al igual que cambiamos algunos detalles con el juego y con un problema que teníamos al momento de pasar el archivo a rar, esto se solucionó usando un método de raíz para evitar problemas con las imágenes.

Conclusión Luis

Personalmente, puedo concluir que gracias al proyecto estoy aprendiendo cada vez más sobre tkinter, así como las diferentes maneras en las cuales uno puede estructurar los datos y los beneficios que cada uno de estos trae a la mesa dependiendo del código que se esté creando. Aunque todavía hay algunos conceptos en los cuales me siento un poco menos conocedor, puedo

decir con certeza que estoy aprendiendo a manejar los datos de forma más eficiente y más clara tanto para el programador como para el usuario.

En esta nueva tercera entrega, pudimos jugar un poco más con las estructuras de datos, de manera en que pudimos resolver el problema de la lotería en términos técnicos. Y, aunque falten algunos detalles para terminar, la verdad es que estoy bastante satisfecho con cómo es que se está formando este programa y verlo en acción definitivamente es algo que voy a recordar con cariño, así como los aprendizajes obtenidos sobre el manejo de datos y cómo es que este funciona.

Conclusión David

En la primer entrega, pudimos meternos más en profundidad con tkinter, usando scrolls, manejo de textos, además de la navegación entre distintas ventanas. Ahora, tenemos un esqueleto muy completo y funcional para nuestras futuras entregas con portada, instrucciones y el programa pirncipal. También aprendimos a controlar con qué ventana arrancará el programa inicialmente. En esta segunda entrega, aprendimos a trabajar con imágenes sobre todo, buscamos los recursos necesarios para arrancar con el desarrollo, a su vez, finalizamos con la investigación y documentación pertinentes.

En la tercera entrega, decidimos que había que cambiar los botones para que el programa fuera más dinámico, además, adelantamos el funcionamiento, logramos colocar las plantillas ordenadamente y cambiar los tamaños dependiendo del número de jugadores. También, implementamos nuevas clases como Jugador y Carta, para que al sacar una carta, podamos saber la ubicación exacta de la cartas para marcarlas con un frijol y determinar ganadores o empate. Todo esto, llevó indagar en documentació para poder comprender cómo funciona cada método de Tkinter, así como el uso de conceptos como POO, manejo de listas y considerar todas las validaciones pertinentes para evitar fallos en el funcionamiento del juego.

Referencias

Saavedra, T. A. (2022, 31 agosto). Historia del juego de la lotería mexicana y los 54 versos para cantarla - México Desconocido. México Desconocido. https://www.mexicodesconocido.com.mx/historia-del-juego-de-la-loteria-y-los-54-versos-para-cantarla.html

Reglas e instrucciones de la lotería - TheLotter México. (s.f.). TheLotter México. https://www.thelotter.mx/juego-loteria-mexicana/

Python, R. (2023, 15 abril). Barra de desplazamiento (Scrollbar) en Tk (tkinter) - Recursos Python. Recursos Python. https://recursospython.com/guias-y-manuales/barra-de-desplazamiento-scrollbar-en-tk-tkinter/

OpenAI. (2023). ChatGPT (21 Feb version) [Large language model]. https://chat.openai.com/chat

GeeksforGeeks. (2021, 31 agosto). *Python Tkinter Canvas Widget*. GeeksforGeeks. Recuperado 3 de marzo de 2025, de https://www-geeksforgeeks-org.translate.goog/python-tkinter-canvas-widget/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

How to insert an image in a Tkinter canvas item? (s. f.). Recuperado 3 de marzo de 2025, de https://www.tutorialspoint.com/how-to-insert-an-image-in-a-tkinter-canvas-item

GeeksforGeeks. (2022, 22 febrero). *Python | pack() method in Tkinter*. GeeksforGeeks. Recuperado 3 de marzo de 2025, de https://www-geeksforgeeks-org.translate.goog/python-pack-method-in-tkinter/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

GeeksforGeeks. (2022b, diciembre 22). *Python | place() method in Tkinter*. GeeksforGeeks. Recuperado 3 de marzo de 2025, de https://www-geeksforgeeks-org.translate.goog/python-place-method-in-tkinter/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

destroy() method in Tkinter - Python. (s. f.). Recuperado 3 de marzo de 2025, de https://www-tutorialspoint-com.translate.goog/destroy-method-in-tkinter-python?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

GeeksforGeeks. (2024a, julio 24). *How to Hide, Recover and Delete Tkinter Widgets?* GeeksforGeeks. Recuperado 3 de marzo de 2025, de https://www.geeksforgeeks.org/how-to-hide-recover-and-delete-tkinter-widgets/

How to add text inside a Tkinter Canvas? (s. f.). Recuperado 3 de marzo de 2025, de https://www.tutorialspoint.com/how-to-add-text-inside-a-tkinter-canvas

GeeksforGeeks. (2021a, agosto 31). *How to resize Image in Python Tkinter?* GeeksforGeeks. Recuperado 3 de marzo de 2025, de https://www.geeksforgeeks.org/how-to-resize-image-in-python-tkinter/

ImageTK module. (s. f.). Pillow (PIL Fork). Recuperado 3 de marzo de 2025, de https://pillow.readthedocs.io/en/stable/reference/ImageTk.html

GeeksforGeeks. (2024a, junio 5). *How to Disable / Enable a button in Tkinter?* GeeksforGeeks. Recuperado 3 de marzo de 2025, de https://www.geeksforgeeks.org/how-to-disable-enable-a-button-in-tkinter/

W3Schools.com. (s. f.). Recuperado 3 de marzo de 2025, de https://www.w3schools.com/python/ref_random_sample.asp

GeeksforGeeks. (2021a, julio 4). *Hide and Unhide The Window in Tkinter Python*. GeeksforGeeks. Recuperado 3 de marzo de 2025, de https://www.geeksforgeeks.org/hide-and-unhide-the-window-in-tkinter-python/

GeeksforGeeks. (2021b, agosto 11). *Python Tkinter Toplevel Widget*. GeeksforGeeks. Recuperado 3 de marzo de 2025, de https://www.geeksforgeeks.org/python-tkinter-toplevel-widget/