TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN A DISTANCIA

Trabajo Integrador Final

Alumnos:

María Verónica Biskupski – veronicabiskupski79@gmail.com - Comisión 7

Leylén Magalí Demarlenge - ldemarlenge@gmail.com - Comisión 4

Materia: Programación I

Profesor: Cinthia Rigoni y Sebastían Bruselario

Fecha de Entrega: 08 de noviembre de 2025

Índice

1. Objetivo

Desarrollar una aplicación en Python que permita gestionar información sobre países, aplicando listas, diccionarios, funciones, estructuras condicionales y repetitivas, ordenamientos y estadísticas. El sistema debe ser capaz de leer datos desde un archivo CSV, realizar consultas y generar indicadores clave a partir del dataset.

El objetivo principal es afianzar el uso de estructuras de datos, modularización con funciones y técnicas de filtrado/ordenamiento, aplicando los conceptos aprendidos en Programación 1.

2. Diseño

Conceptos aplicados:

Listas:

El programa utiliza listas para almacenar los registros de países. Cada elemento de la lista representa un país con sus respectivos datos.

Diccionarios:

Cada país se guarda como un diccionario con las claves: 'nombre', 'poblacion', 'superficie' y 'continente'. Estos diccionarios se agrupan dentro de una lista principal.

Funciones:

El código está estructurado en funciones que ejecutan tareas específicas como agregar, actualizar, buscar, filtrar y ordenar países, además de mostrar estadísticas. Esto permite una organización modular y clara del programa.

Condicionales:

Se emplean condicionales (if, while, match, case) para validar datos ingresados, controlar las opciones del menú, verificar si un país existe y asegurar que los valores sean correctos.

Ordenamientos:

El programa implementa ordenamientos mediante la función sorted() y expresiones lambda, permitiendo ordenar los países por nombre, población o superficie de forma ascendente o descendente.

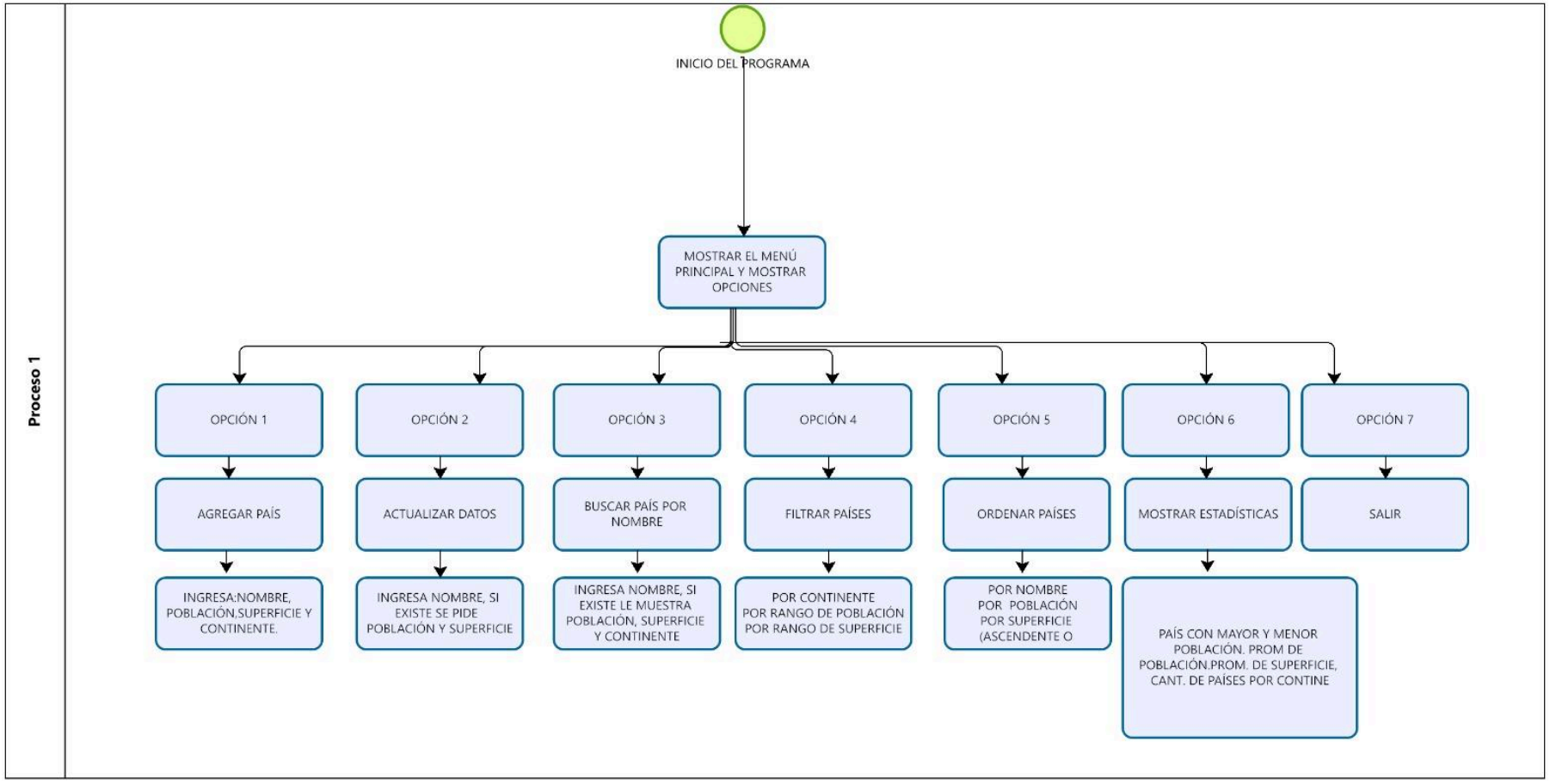
Estadísticas básicas:

Se calculan estadísticas simples como el país con mayor y menor población, el promedio de población y superficie, y la cantidad de países por continente utilizando funciones como max(), min(), sum() y len().

Archivos CSV:

El programa utiliza archivos CSV (datos_paises.csv) para almacenar y leer la información de manera persistente. Emplea csv.DictReader y csv.DictWriter para manejar los datos en formato de diccionario, lo que facilita la manipulación estructurada de la información.

• Definir el flujo de operaciones principales en un diagrama o esquema.


Proceso 1
INICIO DEL PROGRAMA
MOSTRAR EL MENÚ PRINCIPAL Y MOSTRAR OPCIONES
OPCIÓN 1
OPCIÓN 2
OPCIÓN 3
OPCIÓN 4
OPCIÓN 5
OPCIÓN 6
OPCIÓN 7
AGREGAR PAÍS
ACTUALIZAR DATOS
BUSCAR PAÍS POR NOMBRE
FILTRAR PAÍSES
ORDENAR PAÍSES
MOSTRAR ESTADÍSTICAS
SALIR
INGRESA:NOMBRE, POBLACIÓN,SUPERFICIE Y CONTINENTE.
INGRESA NOMBRE, SI EXISTE SE PIDE POBLACIÓN Y SUPERFICIE
INGRESA NOMBRE, SI EXISTE LE MUESTRA POBLACIÓN, SUPERFICIE Y CONTINENTE
POR CONTINENTE
POR RANGO DE POBLACIÓN
POR RANGO DE SUPERFICIE
POR NOMBRE
POR POBLACIÓN
POR SUPERFICIE
(ASCENDENTE O
PAÍS CON MAYOR Y MENOR POBLACIÓN. PROM DE POBLACIÓN.PROM. DE SUPERFICIE, CANT. DE PAÍSES POR CONTINE

3. Funcionalidades

El programa creado ofrece un menú de opciones en consola que permite al usuario:

1. Agregar un país con todos los datos necesarios para almacenarse
2. Actualizar los datos de Población y Superficie de un País
3. Buscar un país por nombre
4. Filtrar países por:
    a. Continente
    b. Rango de población
    c. Rango de superficie
5. Ordenar países por:
    a. Nombre
    b. Población
    c. Superficie (ascendente o descendente)
6. Mostrar estadísticas:
    a. País con mayor y menor población
    b. Promedio de población
    c. Promedio de superficie
    d. Cantidad de países por continente

Planteamos como objetivo del programa que el mismo devuelva información de manera clara y legible para el usuario, en este punto investigamos cómo podíamos mostrar la información de mejor manera y encontramos la opción de mostrarla en formato tabla para los puntos 4 y 5 del menú.

El código creado ofrece la posibilidad de seguir utilizando el programa luego de elegir un punto del menú y sólo salir en caso de que el usuario elija hacerlo tanto en el menú principal como en los sub-menús de los puntos 4, 5 y 6. Esto otorga fluidez y una mejor conexión entre el programa y el usuario, haciendo que este pueda seguir utilizándolo una y otra vez las veces que desee antes de salir.

4. Código

A continuación presentamos el código que se ha desarrollado:

```
#Importamos paquetes que nos permitirán trabajar con un archivo csv que contenga la información
import csv
import os

#Para evitar poner el nombre del archivo cada vez que lo llamemos, lo guardamos en una variable
```

```
NOMBRE_ARCHIVO = "datos_paises.csv"

#Definimos la función que permitirá extraer los datos del archivo en una lista
def obtener_paises():
    paises = []

    #Validamos si el archivo existe, si no ingresa en el if:
    if not os.path.exists(NOMBRE_ARCHIVO):
        print("Archivo no existe")
        #Abrimos el archivo en modo escritura con el w
            with open(NOMBRE_ARCHIVO, "w", newline="", encoding="utf-8") as
archivo:
              #Escribimos dentro el csv. con el escritor, mandamos el archivo
con los encabezados: nombre,poblacion,superficie,continente
                    escritor = csv.DictWriter(archivo, fieldnames=["nombre",
"poblacion", "superficie", "continente"])
            #Se escriben los encabezados
            escritor.writeheader()
        return paises

    #Abrimos el archivo datos_paises.csv; el newline hace que no agregue
    #un enter despues de cada linea, el encoding utf-8 es el más comun
    with open(NOMBRE_ARCHIVO, newline="", encoding="utf-8") as archivo:
         #lector(.dictreader empieza a leer el archivo que se envia, devuelve
un iterador que lee cada una de las lineas
        #un diccionario (nombre,poblacion, superficie, continente)
        lector = csv.DictReader(archivo)

        #Con for recorremos cada uno de los items de lector
        for fila in lector:
            #Agregamos un diccionario, el valor de la key nombre tiene que ser
un string
            #el valor de la key poblacion tiene que ser int por lo que hay que
convertirlo
             #el valor de la key superficie también tenemos que convertirlo a
int
            #el valor de la key continente es un string
```

```
                    paises.append({"nombre": fila["nombre"], "poblacion":
float(fila["poblacion"]),      "superficie":      float(fila["superficie"]),
"continente": fila["continente"]})
    return paises

#Definimos la funcion agregar paises
def agregar_paises(pais):
    #Abrimos el archivo para agregar con a
    with open(NOMBRE_ARCHIVO, "a", newline="", encoding="utf-8") as archivo:
         escritor = csv.DictWriter(archivo, fieldnames=["nombre", "poblacion",
"superficie", "continente"])#abro para escritura con los encabezados
        escritor.writerow(pais)#guardamos una fila

#Definimos la función que nos permitirá validar si el pais ya está en el
archivo
def existe_pais(nombre):
    paises = obtener_paises()
    #Recorremos los paises para ver si ya existe
    for pais in paises:
        #Validamos si ese pais esta, lo escribe como título.
        #Si existe devuelve True y sino False
        if pais["nombre"].title() == nombre.strip().title():
            return True
    return False

#En la siguiente función validamos que la población sea un n° positivo
def validar_numero_positivo(poblacion):
    if poblacion.count(".") > 1:
        return False
      if not poblacion.replace(".", "").isdigit():#reemplaza los puntos por
nada, isdigit si todos no son digitos retorna un false
        return False
    return True

#En la siguiente función validamos que la superficie sea un n° positivo
def validar_numero_positivo(superficie):
    if superficie.count(".") > 1:
        return False
```

```
      if not superficie.replace(".", "").isdigit():#reemplaza los puntos por
nada, isdigit si todos no son digitos retorna un false
        return False
    return True


#Se crea la función que será llamada al elegir la OPCIÓN 1:
def agregar_pais():
    #Solicitamos al usuario el nombre del nuevo país a ingresar
    #Usamos title() para que quede en el mismo formato que los ya ingresados y
strip() para que elimine espacios indeseados
    nombre = input("Ingrese el nombre del país: ").title().strip()
    #Si no se ingresó ningún nombre da un mensaje y vuelve al menú principal:
    if not nombre:
        print("\nEl nombre no puede ser un string vacío\n")
        return
     #Verificamos que no se haya ingresado un número como nombre, no podemos
utilizar .isalpha()
    #porque con un nombre como "Estados Unidos" fallaría.
     #Como solución verificamos si al quitar los espacios queda un dígito o un
número positivo.
    #Y verificamos si tiene al menos un carácter alfabético.
    if not any(char.isalpha() for char in nombre):
        print("\nNombre del país inválido\n")
        return

    #Validamos si el país ya existe, de ser así da un mensaje y vuelve al menú
anterior
    if existe_pais(nombre):
        print("\nEl país que desea ingresar ya existe en la base de datos\n")
        return

    #Solicitamos al usuario la población del nuevo pais.
    #Con strip() eliminamos espacios indeseados
     poblacion = input("Ingrese la población del nuevo pais (sin puntos ni
comas): ").strip()

    #Valida si la población es un número positivo, si no lo es vuelve al menú
anterior
```

```
    if not validar_numero_positivo(poblacion):
        print("\nEl valor de población ingresado no no es válido\n")
        return
    #Si está bien ingresado el valor, lo guarda como entero
    poblacion = float(poblacion)

    #Solicitamos al usuario la superficie del nuevo pais.
    #Con strip() eliminamos espacios indeseados
     superficie = input("Ingrese la superficie del nuevo país (en km2, sin puntos ni comas): ").strip()

    #Valida si la superficie es un número positivo, si no lo es vuelve al menú anterior
    if not validar_numero_positivo(superficie):
        print("\nEl valor de superficie ingresado no no es válido\n")
        return
    #Si está bien ingresado el valor, lo guarda como entero
    superficie = float(superficie)

    #Solicitamos al usuario el nombre del continende del país a ingresado
    #Usamos title() para que quede en el mismo formato que los ya ingresados y strip() para que elimine espacios indeseados
     continente = input("Ingrese el continente al que pertenece el nuevo país: ").strip().title()
    #Si no se ingresó ningún nombre da un mensaje y vuelve al menú principal:
    if not continente:
        print("\nEl continente no puede ser un string vacío\n")
        return
    #Al igual que con el nombre del país, verificamos que no se haya ingresado un número como continente
    if not any(char.isalpha() for char in continente):
        print("\nNombre del continente inválido\n")
        return

    #Agrega el nuevo país con todos sus datos
     agregar_paises({"nombre": nombre, "poblacion": poblacion, "superficie": superficie, "continente": continente})
    print("\nSe agregó el nuevo país correctamente\n")
```

```
#Definimos la función que nos permitirá actualizar el archivo con el nuevo país ingresado
def guardar_paises(paises):
     #Abrimos el archivo para ecsritura con w, sobreescribe el ya existente pero nos asegura mantener actualizado el .csv
    with open(NOMBRE_ARCHIVO, "w", newline="", encoding="utf-8") as archivo:
         escritor = csv.DictWriter(archivo, fieldnames=["nombre", "poblacion", "superficie", "continente"])
        escritor.writeheader()#escribe los encabezados
        escritor.writerows(paises)#crea varias columnas


#Se crea la función que será llamada al elegir la OPCIÓN 2:
def actualizar_datos():
    #Solicitamos al usuario el nombre del país a modificar
    #Usamos title() para que quede en el mismo formato que los ya ingresados y strip() para que elimine espacios indeseados
    nombre = input("Ingrese el país a modificar: ").strip().title()
    #Si no se ingresó ningún nombre:
    if not nombre:
        print("\nEl nombre no puede ser un string vacío\n")
        return

       #Llamamos a la función obtener_paises() creada anteriormente y la guardamos en la variable paises
    paises = obtener_paises()
    #Bandera que se utilizará para validar, se inicia como falsa
    encontrado = False
    #Recorremos la variable paises (es una lista)
    for pais in paises:
        #si el país ingresado coincide con uno ya existente:
        if pais["nombre"].title() == nombre.title():
            #Solicitamos el nuevo valor de población
            poblacion = input("Ingrese nueva cantidad de población: ").strip()
            #Validamos si el valor ingresado no es positivo
            if not validar_numero_positivo(poblacion):
                print("\nEl valor ingresado como población no es válido\n")
                return
            #Convertimos el valor de población a flotante
```

```
            pais["poblacion"] = float(poblacion)

            #Solicitamos el nuevo valor de superficie
            superficie = input("Ingrese nuevo valor de superficie: ").strip()
            #Validamos si el valor ingresado no es positivo
            if not validar_numero_positivo(superficie):
                print("\nEl valor ingresado como superficie no es válido\n")
                return
            #Convertimos el valor de superficie a flotante
            pais["superficie"] = float(superficie)

            #Llamamos a la función guardar_paises() para guardar los cambios
            guardar_paises(paises)
            #La bandera cambia de valor ya que el país fue encontrado
            encontrado = True
            #Informamos al usuario
            print("\nPaís modificado correctamente\n")
            break

    #En caso de que el país ingresado no esté guardado en el archivo
    if not encontrado:#aca la bandera queda en false e informa al usuario
        print("\nNo se encuentra el país en el archivo\n")


#Se crea la función que será llamada al elegir la OPCIÓN 3:
def buscar_pornombre():
    #Solicitamos el país a buscar
    nombre = input("Ingrese el nombre del país a buscar: ").title().strip()

    #paises llama a la funcion obtener paises
    paises = obtener_paises()
    #Colocamos una bandera encontrado que iniciamos en false
    encontrado = False

    #Buscamos con un ciclo for
    #Recorremos cada país del listado paises
    for pais in paises:
        #Compara el nombre del país que está leyendo (pais["nombre"]) con el
nombre que ingresó el usuario (nombre).
```

```
        if pais["nombre"] == nombre:
            #Si encuentra coincidencia informa todos los datos de ese país
            print("\n-País encontrado-")
            print(f"Nombre: {pais['nombre']}")
            print(f"Población: {pais['poblacion']}")
            print(f"Superficie: {pais['superficie']}")
            print(f"Continente: {pais['continente']}\n")
            #Si encuentra el país cambia la variable a True
            encontrado = True
            break #sale del ciclo for
    #Si no encuentra el país dentro de paises imprime mensaje
    if not encontrado:
        print("\nNo se encontró ningún país con ese nombre\n")


#Se crea la función que será llamada al elegir la OPCIÓN 4:
#Primero necesitamos una función que muestre los paises de forma ordenada
#Investigamos con IA para ver cómo mostrar los paises de manera más legible
def mostrar_paises(paises):
    #En caso de que el archivo esté vacío:
    if not paises:
        print("\nNo hay países para mostrar\n")
        return

    #Encabezado de la tabla
    print("-" * 70) #separadores horizontales de la tabla
        print(f"{'Nombre':<20}  |  {'Continente':<12}  |  {'Población':<15}  |
{'Superficie':<15}") #separadores verticales
    print("-" * 70) #separadores horizontales de la tabla

    #Recorre cada país para formatear los datos de población y superficie
    for pais in paises:
        #Formatear la población y superficie con separadores de miles (depende
de la configuración regional, pero usaremos el f-string genérico)
                                                    poblacion_formato       =
f"{pais['poblacion']:,}".split('.')[0].replace(",", ".")
                                                   superficie_formato       =
f"{pais['superficie']:,}".split('.')[0].replace(",", ".")
        #Reemplaza en la tabla:
```

```
                print(f"{pais['nombre']:<20}  |  {pais['continente']:<12}  |
{poblacion_formato:<15} | {superficie_formato:<15}")
    print("-" * 70)
    print("\n")

def filtrar_paises():
    #Llamamos a la función obtener_paises()
    paises = obtener_paises()

    #En caso de que el archivo esté vacío
    if not paises:
        print("\nNo hay países en el archivo para filtrar\n")
        return

     #Creamos un sub-menú para consultar al usuario cómo quiere filtrar la
información
     #El ciclo while nos asegura que vuelva a consultar si desea hacer otro
filtro y no volverá al
    # menú anterior a menos que el usuario elija la opción "d"
    while True:
        print("a. Filtrar por Continente")
        print("b. Filtrar por Rango de población")
        print("c. Filtrar por Superficie")
        print("d. Volver al menú anterior")
        opcion_filtro = input("\nElija una opción: ").strip().lower()

        #Creamos una lista vacía que iremos llenando según el filtro elegido
        paises_filtrados = []

        #Con match recorremos las opciones del sub-menú
        match opcion_filtro:
            case "a":
                        print("\n---Seleccionó  la  opciónd  de  filtrar  por
Continente---\n")
                #Solicitamos que ingrese el continente
                        continente_a_filtrar  =  input("Ingrese  el  nombre  del
continente: ").strip().title()

                #Filtrar por Continente, recorremos con un ciclo for
```

```
                for pais in paises:
                            #Si encuentra coincidencia, agrega a la lista paises_filtrados
                    if pais["continente"] == continente_a_filtrar:
                        paises_filtrados.append(pais)

                        print(f"\nResultados del filtro para el continente: {continente_a_filtrar}")
                mostrar_paises(paises_filtrados)

            case "b":
                    print("\n---Seleccionó la opción de filtrar por Rango de población---\n")
                #Solicitamos y validamos un rango mínimo
                while True:
                    pob_min_str = input("Ingrese Población Mínima: ").strip()
                     if validar_numero_positivo(pob_min_str) or pob_min_str == "":
                        pob_min = float(pob_min_str) if pob_min_str else 0
                        break
                    print("\nPoblación mínima no válida. Intente de nuevo.\n")

                # Solicitamos y validamos un rango máximo
                while True:
                     pob_max_str = input("Ingrese Población Máxima (deje vacío para no limitar): ").strip()
                     if validar_numero_positivo(pob_max_str) or pob_max_str == "":
                         #Usamos un número muy grande (infinito) si el usuario deja vacío el máximo
                             pob_max = float(pob_max_str) if pob_max_str else float('inf')
                        break
                    print("\nPoblación máxima no válida. Intente de nuevo.\n")

                #Verificamos que el mínimo no sea mayor que el máximo, a menos que el máximo sea 'inf'
                if pob_min > pob_max:
```

```
                    print("\nError: La población mínima no puede ser mayor que
la población máxima.\n")
                    continue

                #Filtramos por Rango de Población
                for pais in paises:
                    if pob_min <= pais["poblacion"] <= pob_max:
                        paises_filtrados.append(pais)

                 print(f"\nResultados del filtro por Población (de {pob_min:,}
a {pob_max:,})")
                mostrar_paises(paises_filtrados)

            case "c":
                            print("\n---Seleccionó la opción de filtrar por
Superficie---\n")
                #Solicitamos y validamos la superficie mínima
                while True:
                    sup_min_str = input("Ingrese Superficie Mínima: ").strip()
                    if validar_numero_positivo(sup_min_str):
                        sup_min = float(sup_min_str)
                        break
                            print("\nSuperficie mínima no válida. Intente de
nuevo.\n")

                #Filtramos por Superficie Mínima
                for pais in paises:
                    if pais["superficie"] >= sup_min:
                        paises_filtrados.append(pais)

                    print(f"\nResultados del filtro por Superficie (mínima de
{sup_min:,})")
                mostrar_paises(paises_filtrados)

            case "d":
                print("\n---Volver al menú principal---\n")
                break
            case _:
                print("\nOpción incorrecta\n")
```

```
#Se crea la función que será llamada al elegir la OPCIÓN 5:
def ordenar_paises():
     #Creamos un sub-menú para consultar al usuario cómo quiere ordenar la información
     #El ciclo while nos asegura que vuelva a consultar si desea hacer otro filtro y no volverá al
    # menú anterior a menos que el usuario elija la opción "d"
    while True:
        print("a. Ordenar por Nombre")
        print("b. Ordenar por Población")
        print("c. Ordenar por Superficie")
        print("d. Volver al menú anterior")
        opcion_ordenar = input("\nElija una opción: ").strip().lower()

            #Obtenemos la lista de paises desde el archivo con la función obtener_paises() que guardamos en la variable paises
        paises = obtener_paises()

        #En caso de que el archivo esté vacío:
        if not paises:
            print("\nNo hay países para ordenar\n")
            continue #para volver al menú anterior

         #Asignamos variables que nos permitirán determinar si se ordena por: nombre, población o superficie
        clave_ordenamiento = ""
        titulo = ""

        #Con match recorremos el sub-menú
        match opcion_ordenar:
            case "a":
                print("\n---Seleccionó la opción de ordenar por Nombre---\n")
                #Pedimos orden ascendente o descendente
                while True:
                      orden = input("¿Desea ordenar de forma Ascendente (A) o Descendente (D)? ").strip().lower()
                    #Verificamos que hayan ingresado una opción válida
```

```
                    if orden in ["a", "d"]:
                        break
                    else:
                        print("\nOpción no válida. Ingrese 'A' o 'D'.\n")
                #Definimos si es orden inverso (descendente)
                inverso = True if orden == "d" else False

                clave_ordenamiento = 'nombre'
                titulo = "Nombre"

            case "b":
                        print("\n---Seleccionó la opción de ordenar por
Población---\n")
                #Pedimos orden ascendente o descendente
                while True:
                      orden = input("¿Desea ordenar de forma Ascendente (A) o
Descendente (D)? ").strip().lower()
                    #Verificamos que hayan ingresado una opción válida
                    if orden in ["a", "d"]:
                        break
                    else:
                        print("\nOpción no válida. Ingrese 'A' o 'D'.\n")
                #Definimos si es orden inverso (descendente)
                inverso = True if orden == "d" else False

                clave_ordenamiento = 'poblacion'
                titulo = "Población"

            case "c":
                        print("\n---Seleccionó la opción de ordenar por
Superficie---\n")
                #Pedimos orden ascendente o descendente
                while True:
                      orden = input("¿Desea ordenar de forma Ascendente (A) o
Descendente (D)? ").strip().lower()
                    #Verificamos que hayan ingresado una opción válida
                    if orden in ["a", "d"]:
                        break
                    else:
```

```
                print("\nOpción no válida. Ingrese 'A' o 'D'.\n")
            #Definimos si es orden inverso (descendente)
            inverso = True if orden == "d" else False

            clave_ordenamiento = 'superficie'
            titulo = "Superficie"

        case "d":
            print("\n---Volver al menú principal---\n")
            break
        case _:
            print("\nOpción incorrecta\n")
            continue

              #Usamos sorted() ordena la lista 'paises', usando la
'clave_ordenamiento' (nombre, poblacion, o superficie)
    # y aplicando el orden inverso si corresponde.
                paises_ordenados = sorted(paises, key=lambda pais:
pais[clave_ordenamiento], reverse=inverso)

    #Mostramos los países ordenados
      print(f"\nLista de Países Ordenados por {titulo} ({'Descendente' if
inverso else 'Ascendente'}):")
         mostrar_paises(paises_ordenados) #llamamos a la función creada
anteriormente, que muestra los datos en forma de tabla
    continue #Continuar en el menú de ordenar hasta que elijan 'd'


#Se crea la función que será llamada al elegir la OPCIÓN 6:
def mostrar_estadisticas():
    paises = obtener_paises()
    #En caso de que el archivo esté vacío:
    if not paises:
        print("\nNo hay datos para calcular estadísticas.\n")
        return
    #Creamos un sub-menú para consultar al usuario que estadística quiere ver
     #El ciclo while nos asegura que vuelva a consultar si desea hacer otro
filtro y no volverá al
    # menú anterior a menos que el usuario elija la opción "e"
```

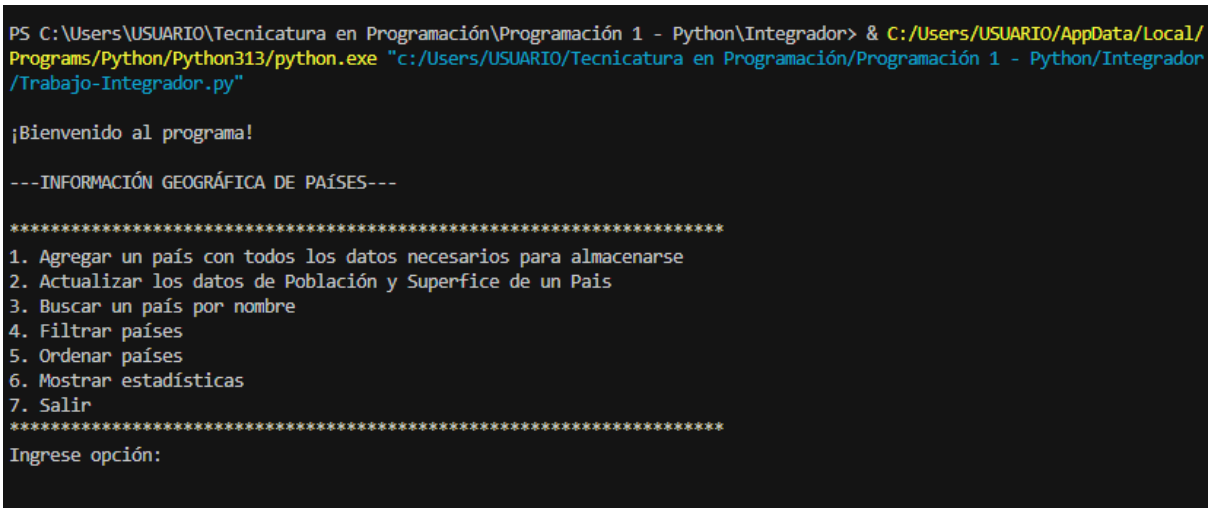
```
    while True:
        print("a. País con mayor y menor población")
        print("b. Promedio de población")
        print("c. Promedio de superficie")
        print("d. Cantidad de países por continente")
        print("e. Volver al menú anterior")

        opcion = input("\nElija una opción: ").lower().strip()

        #Con match recorremos el sub-menú:
        match opcion:
            case "a":
                    print("\n---Seleccionó la opción para mostrar el país con
mayor y menor población---\n")
                  #max() y min() con una función lambda para encontrar el país
con la población más grande y más chica.
                  mayor = max(paises, key=lambda p: p["poblacion"]) #key=lambda
define una función que recibe como parametro población
                menor = min(paises, key=lambda p: p["poblacion"])
                #Muestra los nombres y valores con formato de miles :,.0f
                      print(f"\n País con mayor población: {mayor['nombre']}
({mayor['poblacion']:,.0f} habitantes)")
                        print(f" País con menor población: {menor['nombre']}
({menor['poblacion']:,.0f} habitantes)\n")

            case "b":
                  print("\n---Seleccionó la opción para mostrar el promedio de
población de los países guardados---\n")
                   promedio_poblacion = sum(p["poblacion"] for p in paises) /
len(paises) #Divide entre la cantidad total de países (len(paises))
                #para obtener el promedio.
                  print(f"\n  Promedio de población: {promedio_poblacion:,.2f}
habitantes\n") #Muestra el resultado con dos decimales (:,.2f).

            case "c":
                  print("\n---Seleccionó la opción para mostrar el promedio de
superficie de los países guardados---\n")
                  promedio_superficie = sum(p["superficie"] for p in paises) /
len(paises)
```

```
                print(f"\n Promedio de superficie: {promedio_superficie:,.2f} km²\n")

            case "d":
                print("\n---Seleccionó la opción para mostrar la cantidad de países guardados por continente---\n")
                #Crea un diccionario vacío "conteo" para contar cuántos países hay por continente.
                conteo = {}
                #recorre todos los paises y
                for p in paises:
                    cont = p["continente"]
                        conteo[cont] = conteo.get(cont, 0) + 1 #suma uno al contador correspondiente.
                print("\n Cantidad de países por continente:")
                for cont, cantidad in conteo.items():
                    print(f" {cont}: {cantidad}")
                print()

            case "e":
                print("\n---Volver al menú principal---\n")
                break

            case _:
                print("\nOpción inválida. Intente nuevamente.\n")


#Se crea una función que muestra las opciones del menú y permite al usuario seleccionar una
def mostrar_menu():
    print("\n¡Bienvenido al programa!")
    print("\n---INFORMACIÓN GEOGRÁFICA DE PAíSES---\n")
    while True:
        print("*" * 70)
            print("1. Agregar un país con todos los datos necesarios para almacenarse")
        print("2. Actualizar los datos de Población y Superfice de un Pais")
        print("3. Buscar un país por nombre")
        print("4. Filtrar países")
```

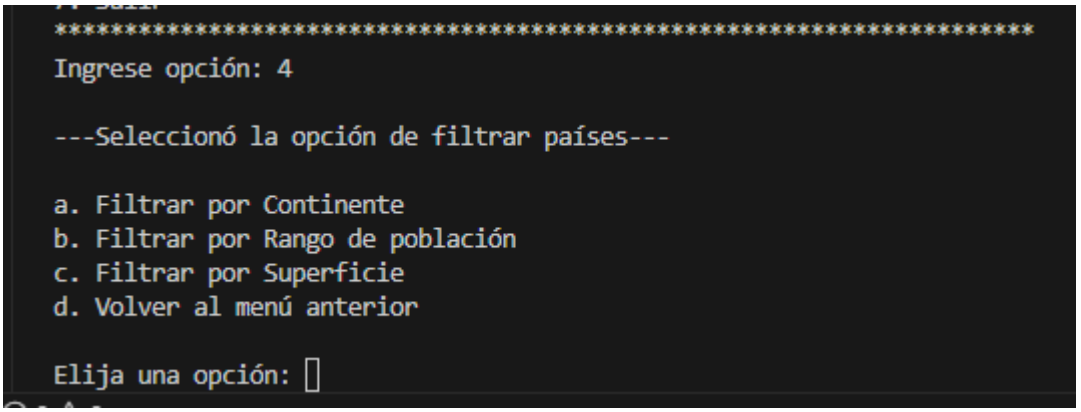
```
        print("5. Ordenar países")
        print("6. Mostrar estadísticas")
        print("7. Salir")
        print("*" * 70)
        opcion = input("Ingrese opción: ").strip()

        match opcion:
            case "1":
                        print("\n---Seleccionó la opción de agregar un nuevo país---\n")
                agregar_pais()
            case "2":
                    print("\n---Seleccionó la opción para actualizar datos de población y superficie---\n")
                actualizar_datos()
            case "3":
                      print("\n---Seleccionó la opción de buscar un país por nombre---\n")
                buscar_pornombre()
            case "4":
                print("\n---Seleccionó la opción de filtrar países---\n")
                filtrar_paises()
            case "5":
                print("\n---Seleccionó la opción de ordenar países---\n")
                ordenar_paises()
            case "6":
                                print("\n---Seleccionó la opción de mostrar estadísticas---\n")
                mostrar_estadisticas()
            case "7":
                print("\n---Usted eligió la opción salir---\n")
                break
            case _:
                print("\nOpción ingresada no válida\n")

#Llamamos a la función mostrar_menu:
mostrar_menu()
```

5. Capturas de pantalla

- Vista del menú principal:



- Vista del sub-menú para el punto 4:



  - Ejemplo de filtrado:

```
Elija una opción: a

---Seleccionó la opciónd de filtrar por Continente---

Ingrese el nombre del continente: asia

Resultados del filtro para el continente: Asia
-------------------------------------------------------------------------
Nombre              | Continente   | Población       | Superficie
-------------------------------------------------------------------------
Japón               | Asia         | 125.800.000     | 377.975
China               | Asia         | 1.408.280.000   | 9.562.910
-------------------------------------------------------------------------


a. Filtrar por Continente
b. Filtrar por Rango de población
c. Filtrar por Superficie
d. Volver al menú anterior

Elija una opción:
```

- Vista del sub-menú para el punto 5:

```
Ingrese opción: 5

---Seleccionó la opción de ordenar países---

a. Ordenar por Nombre
b. Ordenar por Población
c. Ordenar por Superficie
d. Volver al menú anterior

Elija una opción:
```

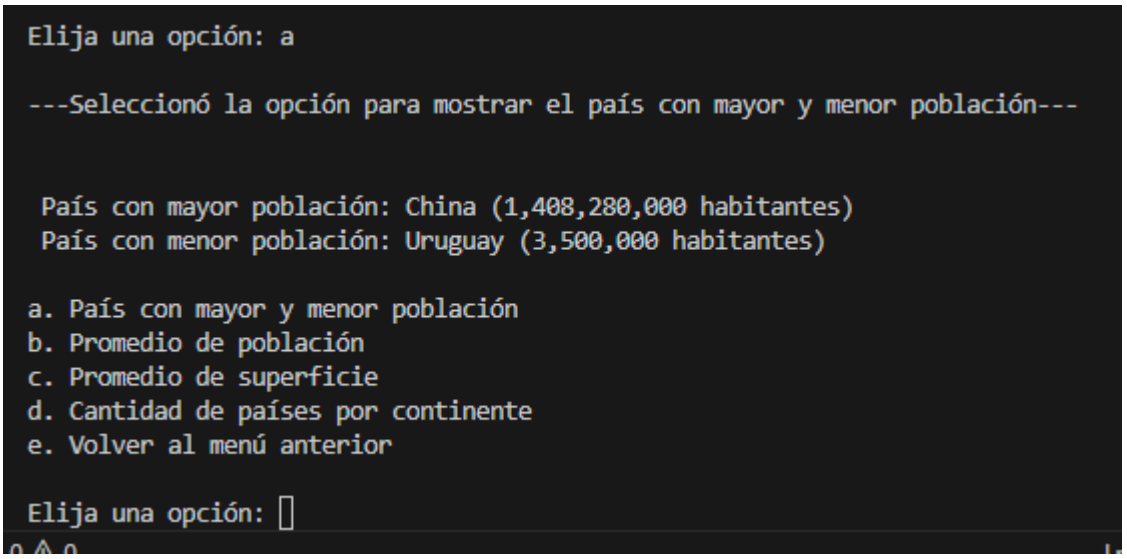
  - Ejemplo de ordenamiento:

```
Elija una opción: b

---Seleccionó la opción de ordenar por Población---

¿Desea ordenar de forma Ascendente (A) o Descendente (D)? a

Lista de Países Ordenados por Población (Ascendente):
------------------------------------------------------------------------
Nombre              | Continente    | Población       | Superficie
------------------------------------------------------------------------
Uruguay             | América       | 3.500.000       | 172.600
Angola              | Africa        | 39.000.000      | 1.246.700
Argentina           | América       | 45.376.763      | 2.780.400
España              | Europa        | 48.619.695      | 505.990
Alemania            | Europa        | 83.149.300      | 357.022
Japón               | Asia          | 125.800.000     | 377.975
Brasil              | América       | 213.993.437     | 8.515.767
China               | Asia          | 1.408.280.000   | 9.562.910
------------------------------------------------------------------------


a. Ordenar por Nombre
b. Ordenar por Población
c. Ordenar por Superficie
d. Volver al menú anterior

Elija una opción:
```

- Vista del sub-menú para el punto 6:

```
************************************************************************
Ingrese opción: 6

---Seleccionó la opción de mostrar estadísticas---

a. País con mayor y menor población
b. Promedio de población
c. Promedio de superficie
d. Cantidad de países por continente
e. Volver al menú anterior

Elija una opción:
```

  - Ejemplo de estadística:



6. Conclusiones

Con este trabajo hemos podido afianzar y aplicar los conceptos que aprendimos a lo largo de la materia. Hemos podido crear un programa, a nuestro criterio, completo robusto y que cumple con lo solicitado en las consignas.

De lo visto en las unidades 4 (Listas) y 7 (Datos Complejos) pusimos en práctica las estructuras de: listas, la cual resultó ideal para la gestión de los registros; y diccionarios, que permitió guardar los atributos de cada país (nombre, población, superficie y continente) facilitando su acceso y manipulación.

Por medio de funciones (unidad 6) hemos podido separar el código en pequeños módulos, esto no sólo nos permitió a nosotras como desarrolladoras dividir el problema en partes pequeñas y resolverlos de a uno; sino que hizo que el código quede más legible y prolijo. Entendemos que esto también favorece al mantenimiento del mismo, si fuese necesario.

Mediante un archivo .CSV logramos guardar la información de los países aún cuando cerramos el código o apagamos la PC. Gracias a lo visto en la unidad 8 (Manejo de archivos) pudimos hacer que la información se mantenga actualizada luego de cada ejecución.

Parte esencial del código fueron las estructuras repetitivas y condicionales (vistas en las unidades 3 y 4) que no sólo nos permitieron validaciones sino que nos permitieron crear menús interactivos y cíclicos, mejorando la experiencia del usuario y dando fluidez al programa.

El trabajo colaborativo fue parte esencial de este trabajo, cada una pudimos trabajar en el código, mejorarlo, probarlo y corregirlo hasta lograr el resultado final.

7. Bibliografía

Python Software Foundation. (2024). *Referencia del Lenguaje Python* (versión 3.14.0) [Documentación en línea]. https://docs.python.org/es/3/reference/index.htm

Universidad Tecnológica Nacional. (2025). *Teoría de listas, tuplas y diccionarios* [Archivo Jupyter]. Tecnicatura Universitaria en Programación.

Google LLC. (2025). *Google Gemini* [Sitio web]. Recuperado de https://gemini.google.com/