Trabajo Practico Integrador

Gestion de países en Python

UTN-Tecnicatura Universitaria en Programación

Prof. **Ariel Edgardo Enferrel**

Integrantes: **Martinez Enzo**
Puca Kevin

Marco Teórico

Listas

Una lista en Python es una colección ordenada y mutable de elementos que pueden ser de distintos tipos. Permite almacenar varios valores en una sola variable, manteniendo el orden en que se agregan. Se pueden modificar, agregar o eliminar elementos en cualquier momento.
Las listas son muy útiles cuando se necesita trabajar con conjuntos de datos, recorrer elementos o aplicar operaciones repetitivas.
En el proyecto, las listas se utilizaron para guardar la colección completa de países cargados desde el archivo CSV.

Diccionarios

Un diccionario es una estructura de datos que almacena pares de clave y valor. Cada clave es única y se asocia directamente a un valor.
Esta estructura permite acceder a los datos de manera rápida y organizada, facilitando el trabajo con información que tiene distintos atributos.
En el trabajo práctico, se usaron diccionarios para representar cada país con sus características, como nombre, población, superficie y continente. Así, cada elemento de la lista de países fue un diccionario con sus propios datos.

Funciones

Las funciones son bloques de código reutilizables que ejecutan una tarea específica. Ayudan a dividir un programa en partes más pequeñas, facilitando su comprensión, mantenimiento y reutilización.
En el programa, las funciones se usaron para acciones como buscar países, filtrar por continente, calcular promedios o mostrar estadísticas. Cada función tiene un propósito definido y puede ser llamada desde cualquier parte del código.

Condicionales

Los condicionales permiten ejecutar diferentes bloques de código según se cumplan o no ciertas condiciones. En Python, se utilizan las palabras clave *if*, *elif* y *else*.
Gracias a los condicionales, el programa puede tomar decisiones: por ejemplo, verificar si una opción del menú es válida o si el usuario ingresa datos correctos.
También se usaron junto con *try / except* para manejar errores y evitar que el programa se detenga cuando hay entradas inválidas.

Ordenamientos

El ordenamiento de datos permite organizar la información de forma ascendente o descendente según un criterio determinado, como el nombre, la población o la superficie de un país.
En Python existen distintas formas de ordenar listas, lo que facilita presentar los resultados de manera más clara para el usuario.
En el proyecto, se aplicaron ordenamientos para mostrar los países ordenados por distintos atributos.

Estadísticas

Las estadísticas permiten obtener información resumida a partir de un conjunto de datos, como máximos, mínimos, promedios o conteos.
En el proyecto, se calcularon estadísticas como el país con mayor y menor población, el promedio de población y superficie, y la cantidad de países por continente.
Estos cálculos permiten analizar los datos de forma más general y comprender mejor la información almacenada.

Archivos CSV

Los archivos CSV (*Comma-Separated Values*) son archivos de texto que almacenan datos en forma de tabla, separando los valores por comas.
Son muy utilizados para importar y exportar información entre programas, ya que pueden abrirse fácilmente con hojas de cálculo o procesarse con código.
En Python, se pueden leer y escribir archivos CSV con el módulo *csv*, permitiendo convertir los datos en listas o diccionarios.
En el sistema desarrollado, el archivo CSV fue la base de datos del programa: allí se guardaron todos los países y sus datos.

En resumen:

El uso combinado de listas, diccionarios, funciones, condicionales, ordenamientos, estadísticas y archivos CSV permite construir programas completos y bien estructurados.
Cada uno de estos elementos cumple una función clave: las listas almacenan colecciones, los diccionarios organizan la información, las funciones separan tareas, los condicionales controlan el flujo, los ordenamientos y las estadísticas procesan los datos, y los archivos CSV facilitan el almacenamiento externo.
Gracias a la integración de todos estos conceptos, se pudo desarrollar un sistema práctico, modular y funcional para gestionar información sobre países.

Fuentes consultadas

- Programiz. *Python List (With Examples)*. Disponible en: https://www.programiz.com/python-programming/list
- Analytics Vidhya. *Working with Lists and Dictionaries in Python*. Disponible en: https://www.analyticsvidhya.com/blog/2021/06/working-with-lists-dictionaries-in-python/
- Real Python. *Using Python for Data Analysis*. Disponible en: https://realpython.com/python-for-data-analysis/

- KDnuggets. *Mastering Python's Built-in Statistics Module*. Disponible en: https://www.kdnuggets.com/mastering-pythons-built-in-statistics-module

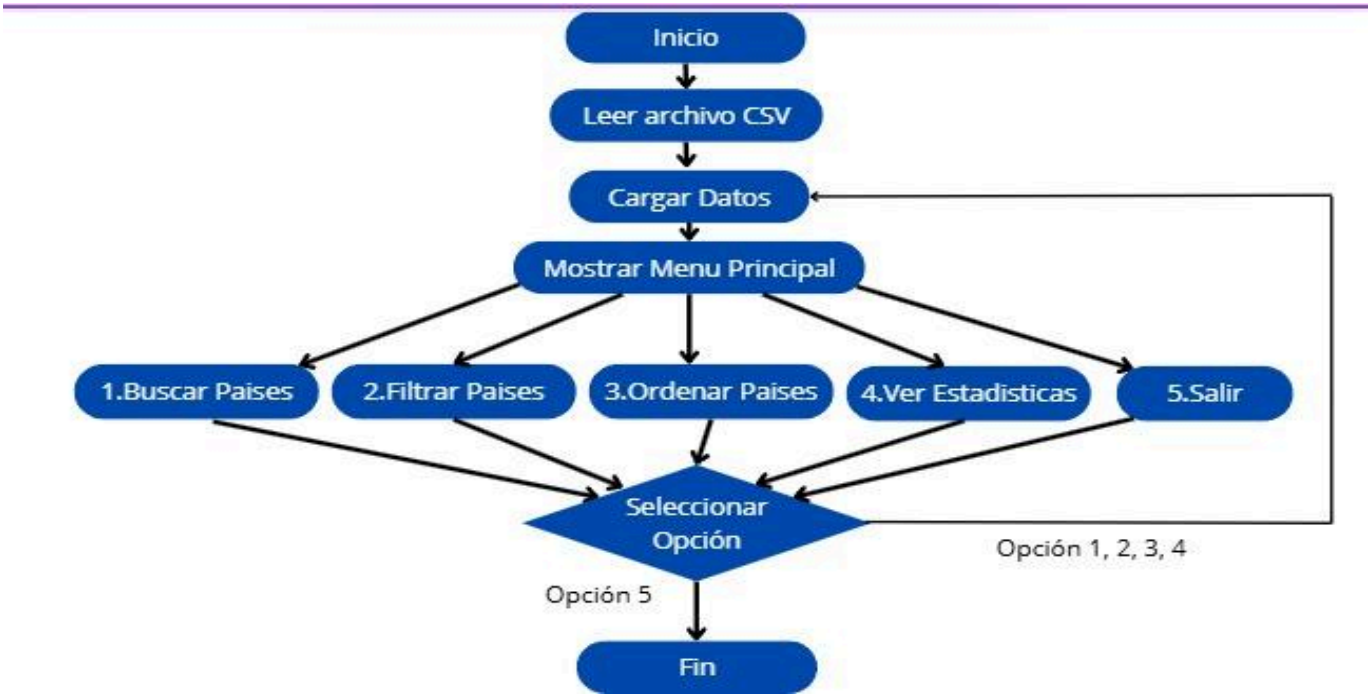
- Documentación oficial de Python https://docs.python.org/3/

- Estructura de datos en Python: https://docs.python.org/3/tutorial/datastructures.html

Objetivo

Desarrollar una aplicación en Python que permita gestionar información sobre países, aplicando estructuras de datos como listas y diccionarios, junto con el uso de funciones modulares, condicionales y estructuras repetitivas.
El sistema debe ser capaz de obtener los datos desde una API externa, generar y actualizar un archivo CSV de respaldo, y ofrecer un menú interactivo que permita realizar búsquedas, filtrados, ordenamientos y estadísticas.
Además, la aplicación deberá poder ejecutarse dentro de un contenedor Docker, garantizando su portabilidad, despliegue sencillo y correcto funcionamiento en cualquier entorno.

Flujo Principal del Programa



Para el desarrollo del sistema de gestión de países se utilizó una metodología modular:
Analizamos el problema, identificando los requerimientos del sistema y definiendo las funcionalidades principales. Luego, se procedió al diseño del programa, estableciendo la estructura general y el uso de listas y diccionarios para almacenar los datos de los países.

Durante la etapa de implementación, se programó en Python aplicando funciones, estructuras condicionales y repetitivas, y manejo de archivos CSV para garantizar la persistencia de la información.

Posteriormente, se llevaron a cabo pruebas y validaciones para comprobar el correcto funcionamiento del sistema, la coherencia de los datos y la fiabilidad de las estadísticas generadas.

Finalmente, se realizó la documentación del proyecto, incluyendo el código fuente, el diagrama y las conclusiones obtenidas.

Evaluaciones y Pruebas

ARCHIVO CSV

Empezamos con el código CSV:
Primero creamos el archivo **"paises.csv**" con 100 países para empezar la prueba

- Usamos csv.DictReader para leer cada fila del CSV y convertirla automáticamente en un diccionario con las claves de la primera fila (nombre, poblacion, superficie, continente).

- Convertimos la población y la superficie a enteros para poder hacer cálculos y estadísticas más adelante.

- Guardamos todos los diccionarios en una lista llamada paises.

- Incluimos manejo de errores con try / except para controlar:
  Que el archivo exista.
  Que no ocurran errores al convertir los datos.

```python
import csv

# --- Funcion para cargar paises desde CSV ---
def cargar_paises(nombre_archivo):
    paises = []
    try:
        with open(nombre_archivo, "r", encoding="utf-8") as archivo:
            lector = csv.DictReader(archivo)
            for fila in lector:
                # Convertimos poblacion y superficie a enteros
                pais = {
                    "nombre": fila["nombre"],
                    "poblacion": int(fila["poblacion"]),
                    "superficie": int(fila["superficie"]),
                    "continente": fila["continente"]
                }
                paises.append(pais)
        return paises
    except FileNotFoundError:
        print("No se encontro el archivo:", nombre_archivo)
        return []
    except Exception as e:
        print("Ocurri un error al leer el archivo:", e)
        return []
```

Luego creamos un código de prueba poner en funcionamiento el archivo.csv

```python
# --- Bloque de prueba ---
if __name__ == "__main__":
    paises = cargar_paises("paises.csv")
    if paises:
        print(f"Se cargaron {len(paises)} paises correctamente:")
        for pais in paises[:5]:  # mostramos solo los primeros 5 para probar
            print(pais)
    else:
        print("No se pudieron cargar los paises.")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Error en los datos de Vaticano, se omite este pais.
Se cargaron 95 paises correctamente:
{'nombre': 'Argentina', 'poblacion': 45376763, 'superficie': 2780400, 'continente': 'América'}
{'nombre': 'Brasil', 'poblacion': 213993437, 'superficie': 8515767, 'continente': 'América'}
{'nombre': 'México', 'poblacion': 130262216, 'superficie': 1964375, 'continente': 'América'}
{'nombre': 'Canadá', 'poblacion': 38005238, 'superficie': 9984670, 'continente': 'América'}
{'nombre': 'Estados Unidos', 'poblacion': 331893745, 'superficie': 9833520, 'continente': 'América'}
```

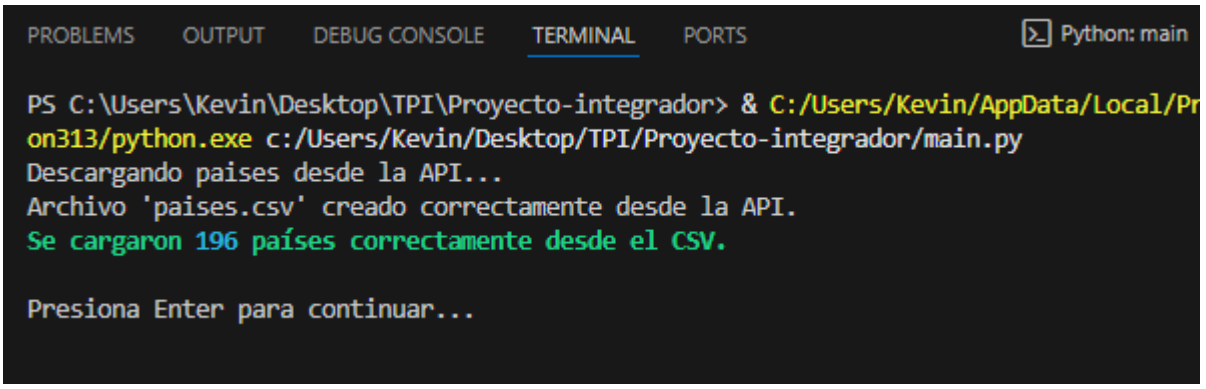
El código carga los países correctamente una vez finalizado, se realizaron cambios como agregar un try/except dentro del for y mejoras de visualización y errores ortográficos.(El código no se encuentra en el programa principal, ya que fue creado para probar la carga del archivo CSV)

API

Además de la carga de datos desde un archivo CSV, el sistema incorpora la conexión con una **API externa** para obtener información actualizada sobre los países.
La API utilizada es:
🔗 **https://api-paises-zilz.onrender.com/paises**

Esta integración permite **automatizar la generación del archivo CSV** utilizado por el programa, garantizando que los datos estén actualizados y provengan de una fuente remota confiable.

```python
import requests
import csv
import os

def actualizar_csv_desde_api():
    archivo_csv = "paises.csv"

    # Si ya existe, no se descarga de nuevo
    if os.path.exists(archivo_csv):
        print("El archivo 'paises.csv' ya existe. No se descargara de nuevo.")
        return

    print("Descargando paises desde la API...")
    url = "https://api-paises-zilz.onrender.com/paises"
```



Buscar Paises

- Convierte nombre_buscar a minúsculas y elimina espacios al principio o al final con .lower().strip().
- Recorre cada país en la lista paises.
- Compara si nombre_buscar está contenido dentro de pais["nombre"].lower().

- Si coincide, agrega ese país a la lista resultados.
- Devuelve la lista resultados, que puede tener uno, varios o ningún país.

```python
def buscar_pais(paises, nombre_buscar):
    """
    Busca paises que contengan 'nombre_buscar' en su nombre (coincidencia parcial o exacta)
    Devuelve una lista de diccionarios con los resultados
    paises proviene del csv prueba que creamos nosotros(paises.csv)
    """
    resultados = []
    nombre_buscar = nombre_buscar.lower().strip()

    for pais in paises:
        if nombre_buscar in pais["nombre"].lower():
            resultados.append(pais)
    return resultados
```

Luego hicimos un código para probar su funcionamiento (el código fue creado para probar su funcionamiento, luego se descartó)

```python
from cargar import cargar_paises
# --- Bloque de prueba rapido para buscar nombres ---
if __name__ == "__main__":
    paises = cargar_paises("paises.csv")  # cargamos el CSV

    if not paises:
        print("No se pudieron cargar los paises.")
    else:
        # Buscamos un par de nombres de prueba
        pruebas = ["argentina", "brasil", "uni"]  # 'uni' para que coincida con "Reino Unido" o "Estados Ur

        for nombre in pruebas:
            resultados = buscar_pais(paises, nombre)
            print(f"\nBuscando: {nombre}")
            if resultados:
                for pais in resultados:
                    print(pais["nombre"])
            else:
                print("No se encontraron coincidencias.")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                Python

Error en los datos de Vaticano, se omite este pais.

Buscando: argentina
Argentina

Buscando: brasil
Brasil

Buscando: uni
Estados Unidos
```

Filtrar Países

Por continente:

- Convierte el nombre del continente ingresado a minúsculas y elimina espacios al inicio y final.

-Valida que el usuario haya ingresado un valor; si no, lanza un error indicando que falta el continente.

-Recorre la lista de países y compara el continente de cada país con el valor ingresado.

-Si coinciden, agrega ese país a la lista de resultados.

-Maneja posibles errores utilizando bloques try / except para mostrar mensajes claros en caso de problemas.

```python
def filtrar_por_continente(paises, continente):
    """
    Filtra los paises que pertenezcan al continente indicado
    Devuelve una lista de diccionarios con los resultados
    """
    resultados = []
    try:
        # Limpiamos espacios y pasamos a minusculas
        continente = continente.strip().lower()

        if not continente:
            raise ValueError("No ingresaste ningun continente.")

        # Recorremos la lista de paises
        for pais in paises:
            if pais["continente"].lower() == continente:
                resultados.append(pais)

    except ValueError as e:
        print("Error:", e)
    except Exception as e:
        print("ERROR, Ocurrio un error inesperado:", e)

    return resultados
```

Por población:

-Convierte los valores de población mínima y máxima a enteros para poder compararlos

.

-Valida que la población mínima no sea mayor que la máxima; si lo es, lanza un error indicando el problema.

-Recorre la lista de países y selecciona aquellos cuya población esté dentro del rango indicado.

-Maneja errores con bloques try / except y muestra mensajes claros si hay problemas.

```python
def filtrar_por_poblacion(paises, min_poblacion, max_poblacion):
    """
    Filtra los paises cuya poblacion este entre min_poblacion y max_poblacion
    Devuelve una lista de diccionarios con los resultados
    """
    resultados = []

    try:
        # Convertimos los valores a enteros
        min_poblacion = int(min_poblacion)
        max_poblacion = int(max_poblacion)

        if min_poblacion > max_poblacion:
            raise ValueError("El valor minimo no puede ser mayor que el maximo.")

        # Recorremos la lista de paises
        for pais in paises:
            if pais["poblacion"] >= min_poblacion and pais["poblacion"] <= max_poblacion:
                resultados.append(pais)

    except ValueError as e:
        print("Error:", e)
    except Exception as e:
        print("ERROR, Ocurrio un error inesperado:", e)

    return resultados
```

Por superficie:

-Convierte los valores de superficie mínima y máxima a enteros.

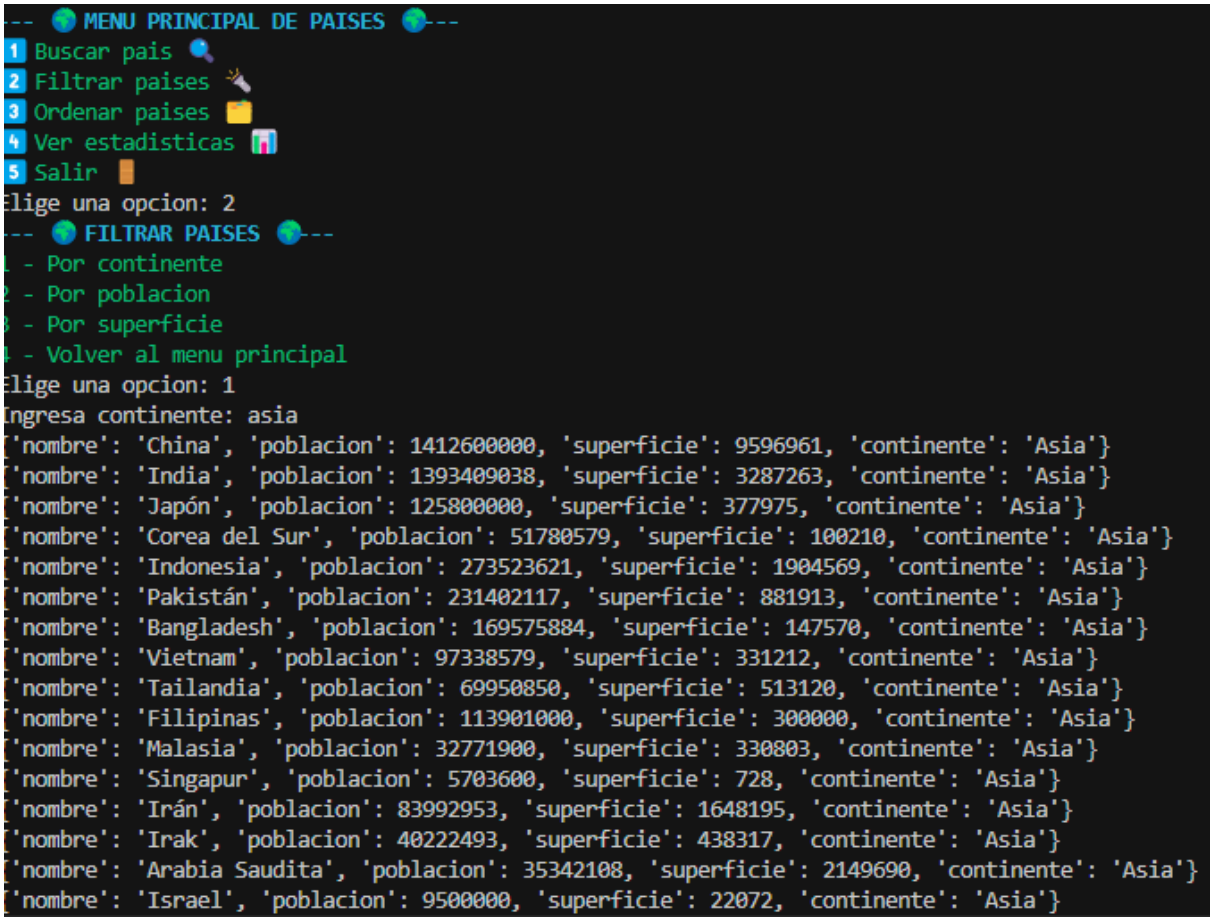
-Valida que la superficie mínima no sea mayor que la máxima; si lo es, lanza un error indicando el problema.

-Recorre la lista de países y selecciona aquellos cuya superficie esté dentro del rango indicado.

-Captura posibles errores con try / except para mostrar mensajes claros en caso de problemas.

```python
def filtrar_por_superficie(paises, min_superficie, max_superficie):

    """Filtra los paises cuya superficie este entre min_superficie y max_superficie
    Devuelve una lista de diccionarios con los resultados
    """
    resultados = []

    try:
        # Convertimos los valores a enteros
        min_superficie = int(min_superficie)
        max_superficie = int(max_superficie)

        if min_superficie > max_superficie:
            raise ValueError("El valor minimo no puede ser mayor que el maximo.")

        # Recorremos la lista de paises
        for pais in paises:
            sup = pais["superficie"]
            if sup >= min_superficie and sup <= max_superficie:
                resultados.append(pais)

    except ValueError as e:
        print("Error:", e)
    except Exception as e:
        print("ERROR, Ocurrio un error inesperado:", e)

    return resultados
```

Luego de crear las funciones probamos el código.
(A su vez, no utilizo códigos de prueba ya que mi compañero pudo realizar el menu y submenu con su correcto funcionamiento)


--- 🌎 MENU PRINCIPAL DE PAISES 🌎---
1 Buscar pais 🔍
2 Filtrar paises 🍾
3 Ordenar paises 🗂️
4 Ver estadisticas 📊
5 Salir 🚪
Elige una opcion: 2
--- 🌎 FILTRAR PAISES 🌎---
1 - Por continente
2 - Por poblacion
3 - Por superficie
4 - Volver al menu principal
Elige una opcion: 1
Ingresa continente: asia
{'nombre': 'China', 'poblacion': 1412600000, 'superficie': 9596961, 'continente': 'Asia'}
{'nombre': 'India', 'poblacion': 1393409038, 'superficie': 3287263, 'continente': 'Asia'}
{'nombre': 'Japón', 'poblacion': 125800000, 'superficie': 377975, 'continente': 'Asia'}
{'nombre': 'Corea del Sur', 'poblacion': 51780579, 'superficie': 100210, 'continente': 'Asia'}
{'nombre': 'Indonesia', 'poblacion': 273523621, 'superficie': 1904569, 'continente': 'Asia'}
{'nombre': 'Pakistán', 'poblacion': 231402117, 'superficie': 881913, 'continente': 'Asia'}
{'nombre': 'Bangladesh', 'poblacion': 169575884, 'superficie': 147570, 'continente': 'Asia'}
{'nombre': 'Vietnam', 'poblacion': 97338579, 'superficie': 331212, 'continente': 'Asia'}
{'nombre': 'Tailandia', 'poblacion': 69950850, 'superficie': 513120, 'continente': 'Asia'}
{'nombre': 'Filipinas', 'poblacion': 113901000, 'superficie': 300000, 'continente': 'Asia'}
{'nombre': 'Malasia', 'poblacion': 32771900, 'superficie': 330803, 'continente': 'Asia'}
{'nombre': 'Singapur', 'poblacion': 5703600, 'superficie': 728, 'continente': 'Asia'}
{'nombre': 'Irán', 'poblacion': 83992953, 'superficie': 1648195, 'continente': 'Asia'}
{'nombre': 'Irak', 'poblacion': 40222493, 'superficie': 438317, 'continente': 'Asia'}
{'nombre': 'Arabia Saudita', 'poblacion': 35342108, 'superficie': 2149690, 'continente': 'Asia'}
{'nombre': 'Israel', 'poblacion': 9500000, 'superficie': 22072, 'continente': 'Asia'}

Ordenar países

```python
def obtener_nombre(pais):
    return pais["nombre"]

def obtener_poblacion(pais):
    return pais["poblacion"]

def obtener_superficie(pais):
    return pais["superficie"]

def ordenar_paises(paises):
    print("--- ORDENAR PAISES ---")

    # validacion de la opcion de criterio
    try:
        print("1 - Por nombre")
        print("2 - Por poblacion")
        print("3 - Por superficie")
        opcion = int(input("Elige una opcion: "))
    except ValueError:
        print("Debe ingresar un numero valido")
        return []
```

```python
# validacion de orden ascendente/descendente
try:
    print("4 - Ascendente")
    print("5 - Descendente")
    orden = int(input("Elige el orden: "))
    if orden == 5:
        reversa = True
    elif orden == 4:
        reversa = False
    else:
        print("ORDEN NO VALIDO. Se usara ascendente por defecto.")
        reversa = False
except ValueError:
    print("ENTRADA NO VALIDA. Se usara ascendente por defecto.")
    reversa = False

# Ordena segun la opcion
if opcion == 1:
    paises_ordenados = sorted(paises, key=obtener_nombre, reverse=reversa)
elif opcion == 2:
    paises_ordenados = sorted(paises, key=obtener_poblacion, reverse=reversa)
elif opcion == 3:
    paises_ordenados = sorted(paises, key=obtener_superficie, reverse=reversa)
else:
    print("OPCION INVALIDA")
    return []

# Mostrar resultados
print("Paises ordenados:")
for pais in paises_ordenados:
    print(f"{pais['nombre']} - Poblacion: {pais['poblacion']:,} - Superficie: {pais['superficie']:,}")

return paises_ordenados
```

El usuario elige entre tres opciones para decidir cómo ordenar los países: Por nombre, Por población Por superficie

Se utiliza un bloque de control de errores para asegurarse de que el usuario ingrese un número válido. Si el valor ingresado no es correcto, se muestra un mensaje de error y la función termina, devolviendo una lista vacía.

Luego, el usuario debe elegir si el orden será ascendente o descendente.

Ascendente: los países se muestran de menor a mayor según el criterio elegido.
Descendente: los países se muestran de mayor a menor según el criterio elegido.

Si el usuario ingresa un valor incorrecto o no numérico, se asigna automáticamente el orden ascendente por defecto y se muestra un mensaje indicándolo.

Dependiendo de la opción seleccionada, la lista de países se ordena según el atributo correspondiente:

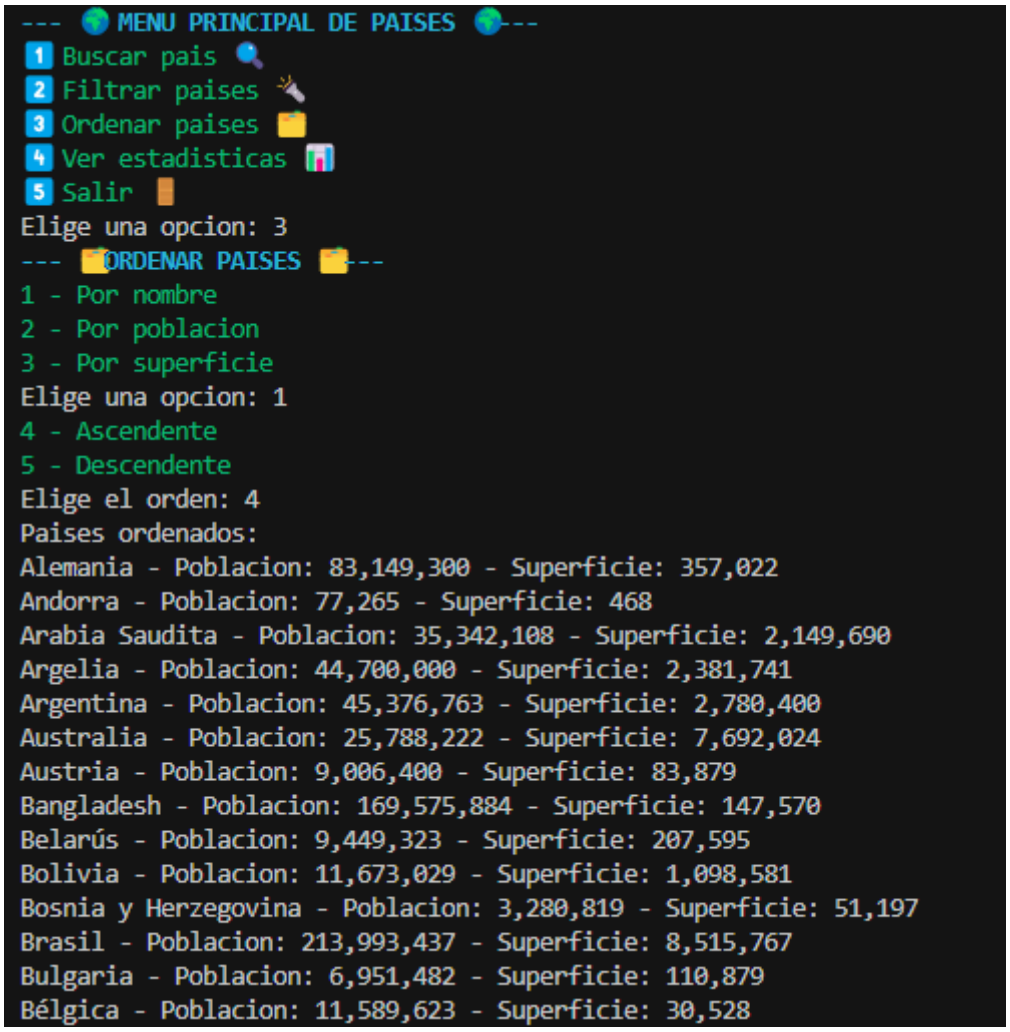
- Por nombre
- Por población

- Por superficie

El ordenamiento se realiza internamente mediante funciones auxiliares que indican qué atributo usar para cada país.

Si el usuario ingresa un número fuera de las opciones permitidas, la función muestra un mensaje de error y termina devolviendo una lista vacía, evitando que el programa se detenga inesperadamente.

Una vez desarrollado se utilizó un submenú dentro de la función diferente a lo realizado anteriormente.



Expuesto a lo anteriormente descrito en la función filtrar, se puso a prueba ya con el menú realizado, sin utilizar un código de prueba para su funcionamiento.

Mostrar estadísticas

```python
# Funcion auxiliar para obtener poblacion de un pai

def obtener_poblacion(pais):
    return pais["poblacion"]

#   Pais con mayor poblacion

def pais_mayor_poblacion(paises):
    if not paises:
        return None

    mayor = paises[0]
    for p in paises:
        if p["poblacion"] > mayor["poblacion"]:
            mayor = p
    return mayor

# Pais con menor poblacion
def pais_menor_poblacion(paises):
    if not paises:
        return None

    menor = paises[0]
    for p in paises:
        if p["poblacion"] < menor["poblacion"]:
            menor = p
    return menor
```

```python
# Promedio de poblacion
def promedio_poblacion(paises):
    if not paises:
        return 0
    total = 0

    for p in paises:
        total += p["poblacion"]
    return total / len(paises)

# Promedio de superficie
def promedio_superficie(paises):
    if not paises:
        return 0
    total = 0

    for p in paises:
        total += p["superficie"]
    return total / len(paises)

# Cantidad de paises por continente

def cantidad_por_continente(paises):
    continentes = {}

    for p in paises:
        cont = p["continente"]
        if cont in continentes:
            continentes[cont] += 1
        else:
            continentes[cont] = 1
    return continentes
```

Función para obtener la población

Esta función auxiliar devuelve el valor de la población de un país específico. Se utiliza en otras funciones para acceder rápidamente al dato de población sin tener que repetir código.

País con mayor población

Esta función recorre la lista de países y compara las poblaciones para determinar cuál tiene la cantidad más alta de habitantes.
Primero verifica si la lista no está vacía para evitar errores. Luego inicializa una variable con el primer país y, a medida que recorre el resto, va reemplazando el valor si encuentra uno con una población mayor.
Finalmente, devuelve el país con la población más grande.

País con menor población

Su funcionamiento es similar al de la función anterior, pero busca el país con la cantidad más baja de habitantes.

También compara cada país de la lista y guarda aquel que tenga una población menor.
Si la lista está vacía, la función devuelve “None”, lo que indica que no hay datos para analizar.

Promedio de población

Esta función calcula el promedio general de población de todos los países registrados.
Primero verifica si la lista está vacía. Si no lo está, recorre todos los países sumando sus poblaciones y luego divide el total entre la cantidad de países.

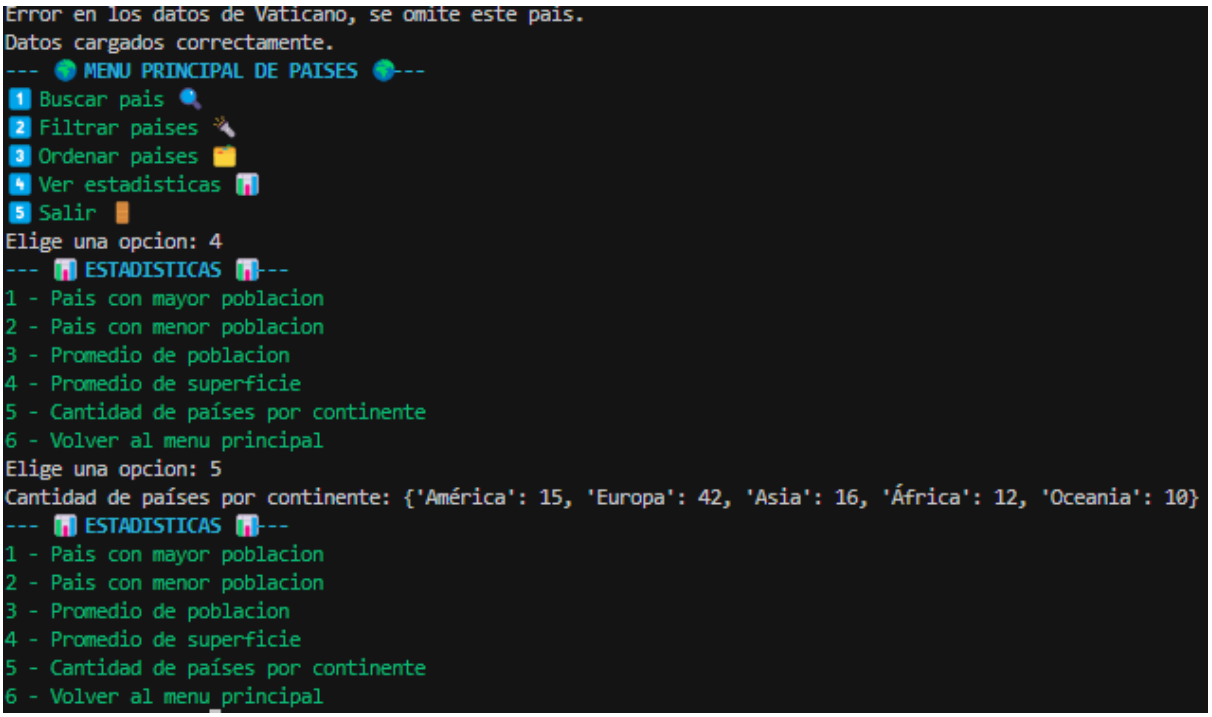
El resultado obtenido representa el promedio de habitantes por país dentro del archivo.

Promedio de superficie

El objetivo de esta función es determinar la media de superficie de los países.
Su estructura es la misma que la función anterior, pero en lugar de sumar poblaciones, acumula el valor de la superficie de cada país.
El total se divide por la cantidad de países para obtener el promedio de extensión territorial.

Cantidad de países por continente

Esta función cuenta cuántos países pertenecen a cada continente.
Para lograrlo, utiliza un diccionario donde la clave es el nombre del continente y el valor es la cantidad de países que tiene.
Cada vez que encuentra un país de un continente ya existente, incrementa su contador. Si el continente aparece por primera vez, lo agrega al diccionario con un valor inicial de 1.
Al finalizar, devuelve un diccionario con el número total de países agrupados por continente.



Elegimos una opción para la prueba de su funcionamiento en el menú

Menu y Submenu (Main)

Estructura general

El programa se organiza en tres niveles principales:

Función main()
Es la función principal del programa. Se encarga de cargar los datos desde el archivo paises.csv mediante la función cargar_paises.
Si los datos se cargan correctamente, muestra un mensaje de confirmación y llama a la función menu_principal.
En caso de que no se pueda acceder al archivo o esté vacío, se informa al usuario y el programa se detiene.

Función menu_principal()
Es la encargada de mostrar el menú principal con las diferentes opciones que el usuario puede elegir.
El menú incluye las siguientes funcionalidades:

- Buscar país
- Filtrar países
- Ordenar países
- Ver estadísticas

- Salir del programa

Cada opción se selecciona escribiendo el número correspondiente.
Si el usuario ingresa una opción inválida, se muestra un mensaje de error y se vuelve a pedir una entrada válida.

Submenú de búsqueda

Cuando el usuario elige la opción **1**, se activa la función buscar_pais, que permite buscar países por nombre, ya sea una coincidencia exacta o parcial.
El programa solicita al usuario el nombre a buscar y muestra todos los países que coincidan con la palabra ingresada.
Si no se encuentra ningún resultado, se muestra un mensaje indicándolo.

Submenú de filtrado

La opción **2** del menú principal llama a la función menu_filtrar, que contiene tres tipos de filtros:

- **Por continente**
  Permite mostrar todos los países pertenecientes al continente ingresado.

- **Por población**
  Muestra los países cuya población se encuentra entre un valor mínimo y máximo ingresado por el usuario.

- **Por superficie**
  Filtra los países según su extensión territorial en kilómetros cuadrados, dentro de un rango especificado.

El submenú se mantiene activo hasta que el usuario elige la opción “Volver al menú principal”.
Además, se incluye manejo de errores para entradas vacías o no numéricas, garantizando que el programa no se interrumpa ante datos incorrectos.

Submenú de ordenamiento

En la opción **3**, el usuario puede acceder a la función ordenar_paises, que permite organizar los datos según tres criterios:

- Nombre del país

- Población

- Superficie

También se puede elegir si el orden será **ascendente** o **descendente**.

Una vez ordenados los países, el programa muestra la lista con el nombre, la población y la superficie de cada uno.

Submenú de estadísticas

La opción **4** del menú principal abre el submenú menu_estadisticas.
Desde este apartado, el usuario puede consultar distintos valores estadísticos relacionados con los países cargados:

1. País con mayor población
2. País con menor población
3. Promedio de población
4. Promedio de superficie
5. Cantidad de países por continente

Cada opción muestra el resultado directamente en pantalla.
El submenú también permite regresar al menú principal cuando el usuario selecciona la opción correspondiente.

Salida del programa

La opción **5** del menú principal permite finalizar la ejecución del programa.
Antes de salir, se muestra un mensaje de confirmación y se interrumpe el ciclo principal, cerrando la aplicación de forma ordenada.

```python
from cargar import cargar_paises
from buscar import buscar_pais
from filtrar import filtrar_por_continente, filtrar_por_poblacion, filtrar_por_superficie
from ordenar import ordenar_paises
from estadisticas import pais_mayor_poblacion, pais_menor_poblacion, promedio_poblacion, promedio_superficie, cantidad_por_continente


# --- SUBMENU FILTRAR ---

def menu_filtrar(paises):

    while True:
        print(Fore.CYAN + Style.BRIGHT + "--- 🌎 FILTRAR PAISES 🌎---")
        print(Fore.GREEN + "1 - Por continente")
        print(Fore.GREEN + "2 - Por poblacion")
        print(Fore.GREEN + "3 - Por superficie")
        print(Fore.GREEN + "4 - Volver al menu principal")

        opcion = input("Elige una opcion: ").strip()

        if opcion == "1":
            continente = input("Ingresa continente: ")
            resultados = filtrar_por_continente(paises, continente)
            if resultados:
                for pais in resultados:
                    print(pais)
            else:
                print(Fore.YELLOW + "No se encontraron resultados.")
```

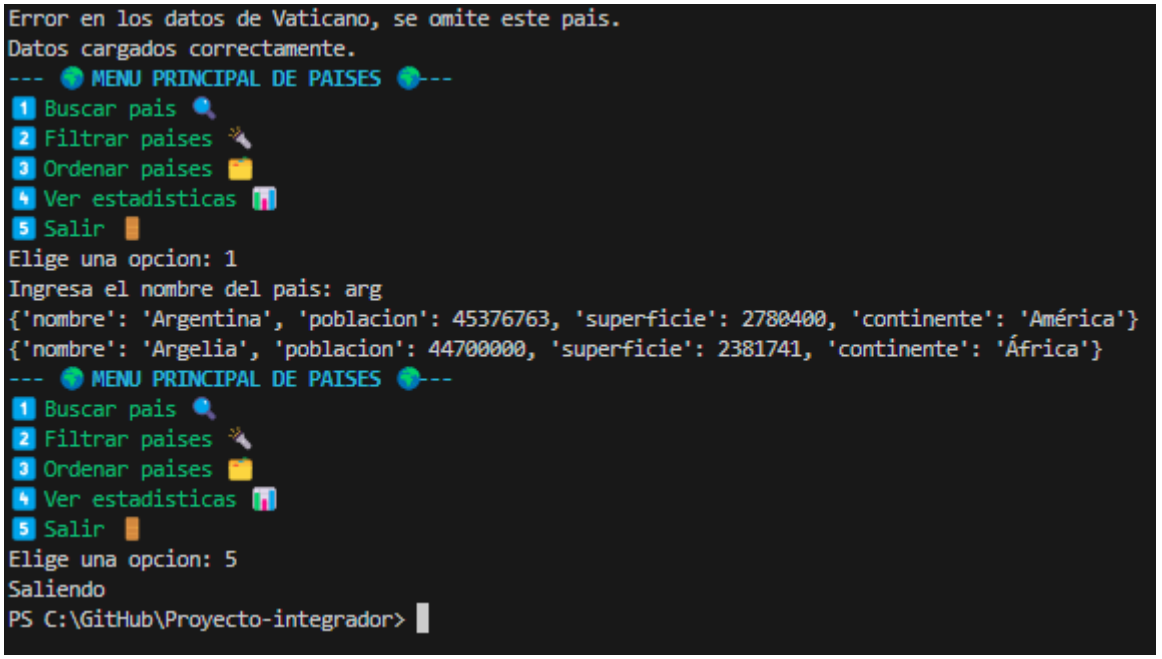
```python
        elif opcion == "2":
            min_pob = input("Poblacion minima: ")
            max_pob = input("Poblacion maxima: ")
            resultados = filtrar_por_poblacion(paises, min_pob, max_pob)
            if resultados:
                for pais in resultados:
                    print(pais)
            else:
                print(Fore.YELLOW + "No se encontraron resultados.")

        elif opcion == "3":
            min_sup = input("Superficie minima: ")
            max_sup = input("Superficie maxima: ")
            resultados = filtrar_por_superficie(paises, min_sup, max_sup)
            if resultados:
                for pais in resultados:
                    print(pais)
            else:
                print(Fore.YELLOW + "No se encontraron resultados.")

        elif opcion == "4":
            break  # aca vuelvee al menu principal

        else:
            print(Fore.RED + "OPCION INVALIDA. INTENTE DE NUEVO ❌")
```

```python
# --- SUBMENU ESTADISTICAS ---

def menu_estadisticas(paises):
    while True:
        print(Fore.CYAN + Style.BRIGHT + "--- 📊 ESTADISTICAS 📊---")
        print(Fore.GREEN + "1 - Pais con mayor poblacion")
        print(Fore.GREEN + "2 - Pais con menor poblacion")
        print(Fore.GREEN + "3 - Promedio de poblacion")
        print(Fore.GREEN + "4 - Promedio de superficie")
        print(Fore.GREEN + "5 - Cantidad de países por continente")
        print(Fore.GREEN + "6 - Volver al menu principal")

        opcion = input("Elige una opcion: ").strip()

        if opcion == "1":
            mayor = pais_mayor_poblacion(paises)
            if mayor:
                print("Paias con mayor poblacion:", mayor["nombre"], "-", mayor["poblacion"])

        elif opcion == "2":
            menor = pais_menor_poblacion(paises)
            if menor:
                print("Pais con menor poblacion:", menor["nombre"], "-", menor["poblacion"])

        elif opcion == "3":
            print("Promedio de poblacion:", round(promedio_poblacion(paises)))

        elif opcion == "4":
            print("Promedio de superficie:", round(promedio_superficie(paises)))

        elif opcion == "5":
            print("Cantidad de países por continente:", cantidad_por_continente(paises))

        elif opcion == "6":
            break  # se vuelve al menu principal

        else:
            print(Fore.RED + "OPCION INVALIDA. INTENTE DE NUEVO ❌")
```

```python
# -- MENU PRINCIPAL ---

def menu_principal(paises):

    while True:
        print(Fore.CYAN + Style.BRIGHT + "--- 🌎 MENU PRINCIPAL DE PAISES 🌎---")
        print(Fore.GREEN + "1️⃣  Buscar pais 🔍")
        print(Fore.GREEN + "2️⃣  Filtrar paises 🔌")
        print(Fore.GREEN + "3️⃣  Ordenar paises 🗂️")
        print(Fore.GREEN + "4️⃣  Ver estadisticas 📊")
        print(Fore.GREEN + "5️⃣  Salir 🚪")

        opcion = input("Elige una opcion: ").strip()

        if opcion == "1":
            nombre = input("Ingresa el nombre del pais: ")
            resultados = buscar_pais(paises, nombre)
            if resultados:
                for pais in resultados:
                    print(pais)
            else:
                print(Fore.YELLOW + "No se encontraron resultados.")

        elif opcion == "2":
            menu_filtrar(paises)  # aca llamamos al submenu

        elif opcion == "3":
            ordenar_paises(paises)

        elif opcion == "4":
            menu_estadisticas(paises)  # aca llamamos al sub menu

        elif opcion == "5":
            print("Saliendo")
            break

        else:
            print(Fore.RED + "OPCION INVALIDA. INTENTE DE NUEVO ❌")
```

```python
# --- FUNCION DEL MAIN PRINCNIPAL ---

def main():
    paises = cargar_paises("paises.csv")
    if not paises:
        print(Fore.YELLOW + "No se pudieron cargar los datos.")
        return
    print("Datos cargados correctamente.")
    menu_principal(paises)

if __name__ == "__main__":
    main()
```

Imagen del menú en funcionamiento



Docker

Gracias a Docker, la aplicación puede ejecutarse en **cualquier computadora o servidor**, sin necesidad de instalar manualmente Python ni las dependencias externas

El “Dockerfile” define las instrucciones necesarias para construir la imagen del proyecto.

Para crear la imagen de Docker, se utiliza el siguiente comando en la terminal

```
PS C:\Users\Kevin\Desktop\TPI\Proyecto-integrador> Docker build -t tpi-python .
[+] Building 2.7s (11/11) FINISHED                                  docker:desktop-linux
 => [internal] load build definition from Dockerfile                                0.0s
 => => transferring dockerfile: 198B                                                0.0s
 => [internal] load metadata for docker.io/library/python:3.11                      1.8s
 => [auth] library/python:pull token for registry-1.docker.io                       0.0s
 => [internal] load .dockerignore                                                   0.0s
 => => transferring context: 2B                                                     0.0s
 => [internal] load build context                                                   0.1s
 => => transferring context: 19.51kB                                                0.0s
 => [1/5] FROM docker.io/library/python:3.11@sha256:df64b825ea3eff541ad1fda26620222d68824e8f3a6  0.0s
 => => resolve docker.io/library/python:3.11@sha256:df64b825ea3eff541ad1fda26620222d68824e8f3a6  0.0s
 => CACHED [2/5] WORKDIR /usr/src/app                                               0.0s
 => CACHED [3/5] COPY requirements.txt ./                                           0.0s
 => CACHED [4/5] RUN python -m pip install --no-cache-dir -r requirements.txt       0.0s
 => [5/5] COPY . .                                                                  0.1s
 => exporting to image                                                              0.4s
 => => exporting layers                                                             0.2s
 => => exporting manifest sha256:8aed51a3c4eceaf4ce0568bcc51c170979790a1968102187546c0bbe261ae5  0.0s
 => => exporting config sha256:dcda6ab901502384f05573a449838730c206770a34825bdf32cb85f407e49568  0.0s
 => => exporting attestation manifest sha256:3a43c96b494b23a740bcfb9832d35d601c4d5e3dd026d99253  0.0s
 => => exporting manifest list sha256:e66302f5c042da2199093bbece39f1f709e3e8a23b3da83fcb4a4dbeb  0.0s
 => => naming to docker.io/library/tpi-python:latest                                0.0s
 => => unpacking to docker.io/library/tpi-python:latest                             0.1s
PS C:\Users\Kevin\Desktop\TPI\Proyecto-integrador>
```

Una vez construida la imagen el programa puede ejecutarse con:

```
\TPI\Proyecto-integrador> docker run -it --rm tpi-python
```

```
PS C:\Users\Kevin\Desktop\TPI\Proyecto-integrador> docker run -it --rm tpi-python
El archivo 'paises.csv' ya existe. No se descargara de nuevo.
Se cargaron 196 países correctamente desde el CSV.

Presiona Enter para continuar...
```

Para comprobar los contenedores en ejecución:

```
\TPI\Proyecto-integrador> docker ps
```

```
PS C:\Users\Kevin\Desktop\TPI\Proyecto-integrador> docker ps
CONTAINER ID   IMAGE          COMMAND            CREATED        STATUS        PORTS     NAMES
bef5126845e9   da95cec8300c   "python main.py"   15 hours ago   Up 15 hours             agitated_grothendieck
PS C:\Users\Kevin\Desktop\TPI\Proyecto-integrador>
```

Resultados Obtenidos

Como resultado del desarrollo del Trabajo Práctico Integrador, el sistema desarrollado cumplió con los objetivos planteados, permitiendo gestionar información sobre países de forma eficiente y modular.
Durante las pruebas, se verificó el correcto funcionamiento de todas las funcionalidades principales:

- **Búsqueda de países** por nombre (exacta o parcial).

- **Filtrado** por continente, rango de población y superficie.
- **Ordenamiento** de los países por nombre, población o superficie.
- **Cálculo de estadísticas**, incluyendo país con mayor y menor población, promedios y cantidad de países por continente.
- **Lectura y escritura en archivos CSV**, garantizando la persistencia de los datos.

Además, se implementó la conexión con una API externa, lo que permitió descargar automáticamente la información actualizada de los países y generar el archivo paises.csv sin necesidad de carga manual.
Finalmente, el proyecto fue ejecutado dentro de un contenedor Docker, confirmando su portabilidad y correcta instalación en cualquier entorno.

Conclusiones

Durante el desarrollo de este trabajo integrador, pudimos aplicar los principales conceptos de la materia Programación 1, utilizando el lenguaje Python para crear un sistema que gestiona información sobre países. Sin embargo, tras la devolución de la primera entrega tuvimos que expandir nuestro conocimiento sobre API y mejoras visuales de la consola.
En conclusión, este proyecto fue una experiencia muy útil para fortalecer nuestra lógica de programación, mejorar el trabajo en equipo y aprender a resolver problemas reales.

Repositorio GitHub

https://github.com/EnzoMartinez25/

Proyecto-integrador Video:

https://youtu.be/n3mniGPDdCQ