**Trabajo Practico Integrador**

**Gestión de Datos de Países en Python**

Pablo Nogueira

Bruno Tello

**Objetivo**

El objetivo principal del proyecto es desarrollar un programa modular en Python que permita gestionar información de países obtenida desde una API pública (RestCountries).  
El sistema ofrece un menú interactivo donde el usuario puede buscar países, filtrados por diferentes criterios (continente, población, superficie), ordenarlos y obtener estadísticas generales.  
Además, el programa genera automáticamente un archivo CSV con los datos obtenidos desde la API y muestra los resultados de forma paginada, mejorando la legibilidad cuando la cantidad de países es extensa.

El proyecto implementa buenas prácticas de programación estructurada, manejo de excepciones, modularizarían y uso de Docker para garantizar portabilidad, independencia del entorno y fácil ejecución.

**Elementos del código**

* **Listas**
  + Estructuras ordenadas que almacenan múltiples elementos.
  + Se usan para guardar el conjunto de países descargados de la API.

Las listas permiten recorrer y manipular fácilmente grandes volúmenes de datos obtenidos desde la API.

Gracias a ellas se pueden aplicar filtros, ordenamientos y estadísticas de manera eficiente utilizando bucles y funciones integradas en Pyhton.

* **Diccionarios**
  + Estructuras con pares clave-valor, ideales para representar información compleja.

Cada país contiene distintos atributos (nombre, población, superficie, continente), por lo que un diccionario permite acceder directamente a la información usando claves descriptivas.

Esto simplifica el código y mejora la legibilidad al evitar el uso de índices numéricos.

* **Funciones**
  + Permiten dividir el código en bloques reutilizables.
  + Cada modulo contiene funciones con una única responsabilidad (por ejemplo, buscar, filtrar, ordenar o calcular estadísticas).

El uso de funciones mejora la organización del código y evita la repetición.

Además, facilita el mantenimiento del sistema y permite reutilizar la lógica de forma independiente en otros contextos o programas.

* **Condicionales**
  + Controlan el flujo del programa y permiten responder a las selecciones del usuario en el menú principal.

Las estructuras if, elif y else hacen posible que el menú principal responda a las opciones elegidas por el usuario y ejecute la función correspondiente.

También se usan para validar datos y manejar casos especiales, como entradas fuera de rango o resultados vacíos.

* **Ordenamientos**
  + Se aplican con funciones y expresiones para organizar los países según nombre, población o superficie (de forma ascendente o descendente).

El ordenamiento mejora la presentación y el análisis de los datos.

Permite que el usuario observe la información de manera lógica, comparando países según distintos criterios sin modificar la estructura original del dataset.

* **Estadísticas**
  + Se utilizan funciones como max(), min() y operaciones aritméticas para obtener indicadores como país con mayor o menor población, promedios de población y superficie, y cantidad de países por continente.

Las estadísticas proporcionan información resumida y útil para analizar tendencias generales del conjunto de países.

Ayudan a comprender las diferencias demográficas y geográficas sin necesidad de examinar todos los datos uno por uno.

* **Archivos CSV**
  + El sistema guarda automáticamente todos los datos obtenidos en un archivo países.csv, asegurando persistencia local y cumpliendo con el manejo de archivos estructurados.

El guardado automático en CSV permite conservar los datos obtenidos de la API incluso después de cerrar el programa.

Además, el formato CSV facilita la reutilización de la información en otros programas herramientas de análisis, como Exel o Pandas.

* **Paginación**
* Se implementa una función que muestra los resultados en bloques de diez países por pantalla, con opciones para avanzar, retroceder o salir.
* Esto mejora la experiencia de usuario y evita desbordar la consola con listados extensos.

La paginación permite visualizar grandes listados de forma ordenada y comprensible.

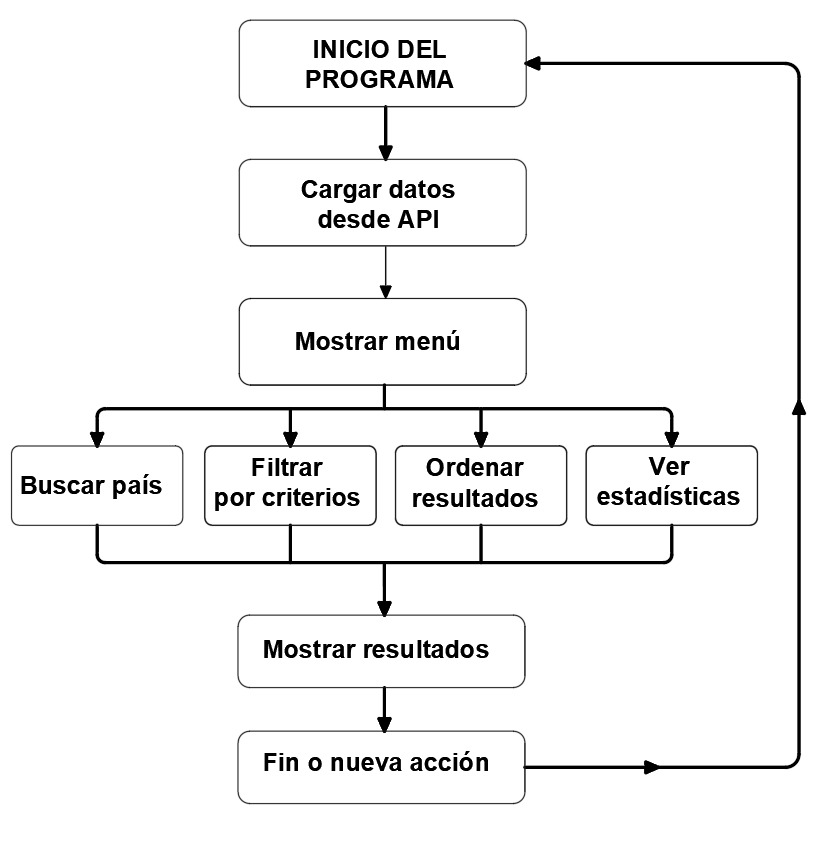
Evita que la consola se satura de información y mejora la interacción del usuario al permitir navegar entre resultados.

* **Manejo de errores**
  + A través de estructuras try/except, se controlan errores comunes como valores inválidos o fallos de conexión con la API.

El manejo de excepciones garantiza que el programa no se interrumpa ante errores previsibles.

Esto mejora la estabilidad del sistema y la experiencia del usuario al ofrecer mensajes claros y seguros ante fallos o datos incorrectos.

**Estructura del flujo del programa**



**Conclusión**

El resultado final es un sistema organizado y completamente funcional, que emplea Python para gestionar información real de forma clara y eficiente.

En el ámbito académico, el proyecto nos permitió poner en práctica los conceptos fundamentales aprendidos: estructuras de datos, modularizarían, validaciones, manejo de archivos externos y análisis de datos.

Desde una perspectiva profesional, nos dejó una enseñanza clave: programar no consiste solo en escribir líneas de código, sino en diseñar soluciones bien planificadas, mantenibles y escalables.

Gracias a la modularidad, el sistema puede ampliarse o modificarse con facilidad, reflejando las buenas practicas utilizadas en el desarrollo de software en entornos reales.