Informe del Programa
Visor de Datos de Países Globales

PYTHON

UTN FACULTAD REGIONAL MENDOZA UNIVERSIDAD TECNOLÓGICA NACIONAL	**Senatore Genaro** **Castro Exequiel Hernán**	**Trabajo Integrador** **Programación 1**

Índice:

1. Marco Teórico

¿Qué es Python?

Python es un lenguaje de programación orientado a objetos de alto nivel y fácil de interpretar con una sintaxis fácil de leer. Ideal para la creación de prototipos y tareas, Python tiene un amplio uso en computación científica, desarrollo web y automatización.
Es importante entender todos los aspectos de Python como un lenguaje de programación de código abierto.

Breve historia de Python

En 1989, Guido van Rossum trabajaba en un laboratorio de informática en los Países Bajos. Decidió escribir un lenguaje que mejorara las fallas de los más populares de su tiempo. Cinco años y medio más tarde lo lanzó, llamándolo "Python".
Al principio fue lento, pero Python tuvo una gran oportunidad cuando un joven Google anunció que usaba Python para muchos procesos internos.
En 2005 Python lanzó Django, un marco para la creación de aplicaciones web. Django explotó en popularidad y llegó a desafiar el entonces dominante marco de Ruby on Rails.
En 2011, Python era el lenguaje más enseñado en programas de ciencias de la computación en todo el país. Unos años más tarde se convirtió en el estándar de facto para los investigadores en los campos de Machine Learning e Inteligencia Artificial, probablemente debido a su facilidad de uso y legibilidad.
El desarrollo de Python ahora es manejado por la Python Software Foundation.

¿Por qué los desarrolladores deben aprender Python?

Python es el primer lenguaje de muchos programadores nuevos. En un curso de informática universitario, generalmente es el primero introducido. ¿Por qué?
Por un lado, Python es legible. A veces, con un poco de conocimiento del idioma (inglés), puedes mirar el código y tener una idea de lo que está tratando de hacer. Esto contrasta con otros lenguajes, lo que podría ser más confuso. Ejemplo de Python para imprimir "¡Hola, mundo!" en la consola de un ordenador**.**

```
print("Hola Mundo!")
print("Hellow World!")
```

Python te obliga a escribir código limpio

A diferencia de la mayoría de los otros lenguajes de desarrollo, el código Python depende de la sangría. En resumen, debe agregar espacios a las líneas de código dentro de un bloque de código determinado de una manera específica y predecible. De lo contrario, el código no se ejecutará.

¿Qué puedes construir con Python?

Como lenguaje de propósito general, la respuesta es: ¡casi cualquier cosa! Python sobresale cuando tienes una tarea compleja que necesitas simplificar, un script corto para ejecutar o un conjunto de datos grande que necesitas manipular.

- Scripts sencillos para automatizar los flujos de trabajo
- Raspadores web para recopilar datos de Internet
- Binarios independientes (es decir, aplicaciones) con Py Installer
- Aplicaciones web de alta potencia y API de REST con Flask o Django
- Programas de aprendizaje automático e inteligencia artificial
- "Código azul" que conecta el software en diferentes idiomas
- Aplicaciones de trading que administran dinero
- Desarrollo de software de código abierto

El lenguaje de programación Python se utiliza activamente en todos los dominios de la informática contemporánea. Dado que el desarrollo de Python es más eficiente que la mayoría del resto de lenguajes, es una opción popular entre las empresas emergentes, donde los cambios en la base de código deben realizarse de forma rápida y barata.
También es un éxito entre científicos y matemáticos, con una serie de potentes bibliotecas internas para obtener estadísticas y realizar cálculos matemáticos complejos.

Bibliotecas populares de Python

Las bibliotecas de paquetes son un servicio esencial para cualquier lenguaje de programación moderno. Sin ellos, cada codificador tendría que escribir su propio software para las tareas más rutinarias y comunes.
Afortunadamente, Python tiene una gran selección de paquetes para todas sus necesidades de programación. Algunas bibliotecas de estrellas escritas en Python se enumeran a continuación:

- **Beautiful Soup** es un raspador de HTML súper cargado, lo que permite a un desarrollador extraer datos de la web a escala
- **Flask** y **Django**, mencionados brevemente anteriormente, proporcionan un desarrollo web increíblemente rápido para casos de uso simples y complejos
- **Tkinter** es el paquete estándar de Python para crear Interfaces Gráficas de Usuario (GUI)
- **NumPy** y **Matplotlib** permiten visualizaciones de datos sencillas e impresionantes
- **PyTorch** para el aprendizaje automático de primera clase

¿Qué es Tkinter?

Tkinter es el paquete de interfaz gráfica de usuario (GUI) predeterminado de Python y está incluido en la biblioteca estándar. Es una capa orientada a objetos que se basa en el conjunto de herramientas de widgets Tcl/Tk de código abierto . No es necesario comprender Tcl/Tk para usarlo.

Aunque existen frameworks de interfaz gráfica de usuario (GUI) más completos (como Qt), Tkinter sigue siendo popular para proyectos pequeños y para principiantes, debido a su sencillez y fácil acceso. Si simplemente quieres probar la programación de interfaces gráficas de usuario, es un excelente punto de partida.

Sintaxis en Python

La sintaxis de Python se diseñó con el objetivo de ser intuitiva y promover un código limpio y legible. Esto se refleja en la forma en que se escribe y estructura el código en Python, facilitando la comprensión y el mantenimiento del mismo.

Espacios en blanco significativos

A diferencia de otros lenguajes, Python utiliza los espacios en blanco (indentaciones) para definir la estructura del código en lugar de llaves o palabras clave específicas.

Esto significa que el nivel de indentación de una línea de código indica su relación con las líneas anteriores, estableciendo así bloques de código.

Comentarios

Los comentarios en Python se realizan usando el símbolo **#** para comentarios de una sola línea, y triple comillas dobles **“ ”** para comentarios de múltiples líneas o docstrings, los cuales son especialmente útiles para documentar el propósito de funciones y clases.

```
# Ejemplo Comentado
"""Ejemplo Comentado"""
```

Variables y tipos de datos

En Python, las variables son como cajas mágicas donde puedes guardar casi cualquier cosa que desees, sin tener que etiquetarlas con tipos específicos.

Esta flexibilidad proviene de ser un lenguaje de tipado dinámico, lo que significa que Python entiende qué tipo de objeto estás almacenando al momento que lo almacenas.

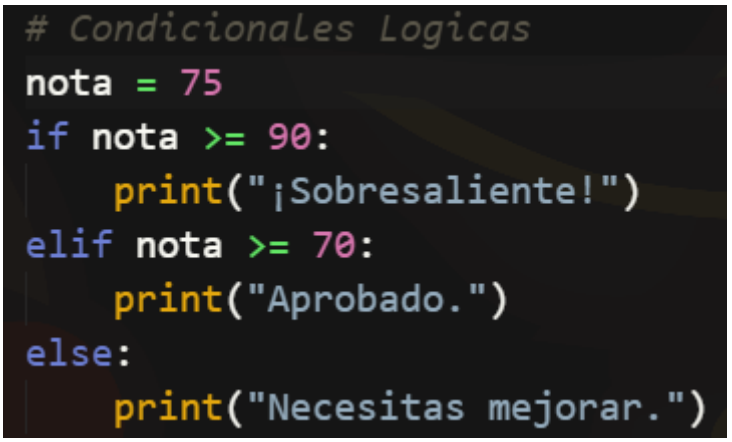
Esto es porque Python infiere el tipo de dato, permitiendo una sintaxis más ágil y menos verbosa. Por ejemplo, para asignar un valor a una variable, simplemente utilizas el signo de igual (=) sin necesidad de especificar qué va a almacenar:

```
mi_variable= "hola"
mi_variable2= 10
mi_variable3= 20.34
```

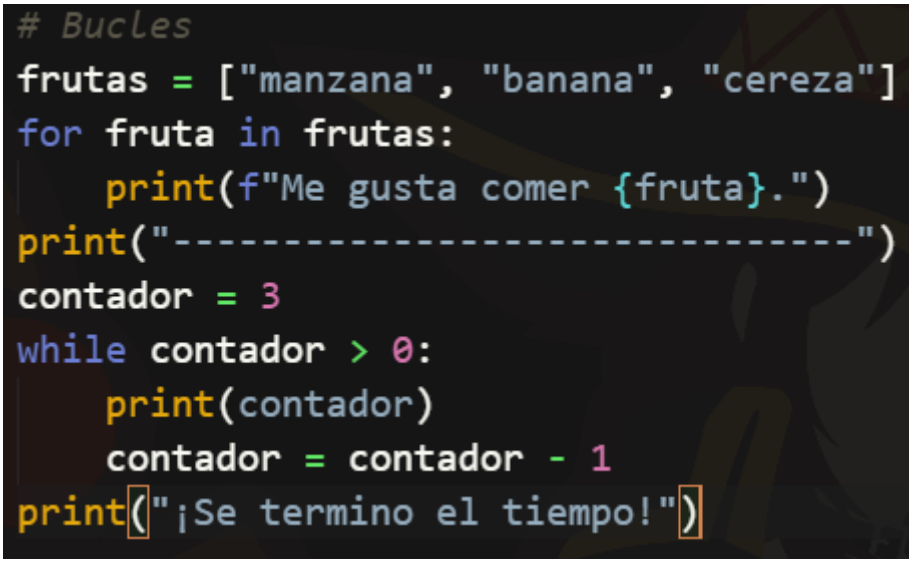
Estructuras de control

Las estructuras de control en Python incluye **Condicionales Lógicos (if, elif, else)** y **Bucles (for, while)**. Estas estructuras permiten dirigir el flujo de ejecución del programa en función de condiciones específicas o repetir un bloque de código múltiples veces.

Estas estructuras proveen formas simplificadas de escribir ciertos fragmentos del código y generar instrucciones que puedan ser ejecutadas dependiendo de ciertas condiciones y un número controlado de veces.



```python
# Condicionales Logicas
nota = 75
if nota >= 90:
    print("¡Sobresaliente!")
elif nota >= 70:
    print("Aprobado.")
else:
    print("Necesitas mejorar.")
```



```python
# Bucles
frutas = ["manzana", "banana", "cereza"]
for fruta in frutas:
    print(f"Me gusta comer {fruta}.")
print("------------------------------")
contador = 3
while contador > 0:
    print(contador)
    contador = contador - 1
print("¡Se termino el tiempo!")
```

Tipos de datos

En Python, los tipos de datos fundamentales definen la categoría de valores que una variable puede almacenar y cómo el intérprete maneja dicha variable. Comprender estos tipos es crucial para manipular datos de manera efectiva en Python. Veamos los más comunes:

Números

Python admite varios tipos numéricos, incluidos enteros **(int)**, números de punto flotante (float). Los enteros pueden ser de cualquier longitud y los de punto flotante tienen precisión doble.

```
mi_variable= int(5)
mi_variable2= float(10.34)
```

Cadenas de texto (Strings)

Las cadenas de texto en Python se definen entre comillas simples (') o dobles ("), y pueden contener caracteres alfanuméricos y símbolos.

```
mi_variable= str("Hola Mundo!")
```

Booleanos

Los booleanos representan dos valores: Verdadero **(True)** y Falso **(False)**. Son muy utilizados en expresiones condicionales y bucles.

```
mi_variable= bool(True)
mi_variable1= bool(False)
```

Listas

Las listas son estructuras de datos que permiten almacenar una colección ordenada y mutable de elementos. Los elementos pueden ser de diferentes tipos, incluyendo otra lista.

```
mi_lista=[1, 2.5, "Hola Mundo!", [3, 4]]
```

Tuplas

Las tuplas son similares a las listas, pero son inmutables. Una vez creada una tupla, no se pueden modificar sus elementos. Es un tipo más ligero ya que su manipulación es mínima.

```
mi_tupla = (1, 2.5, "Hola Mundo!")
```

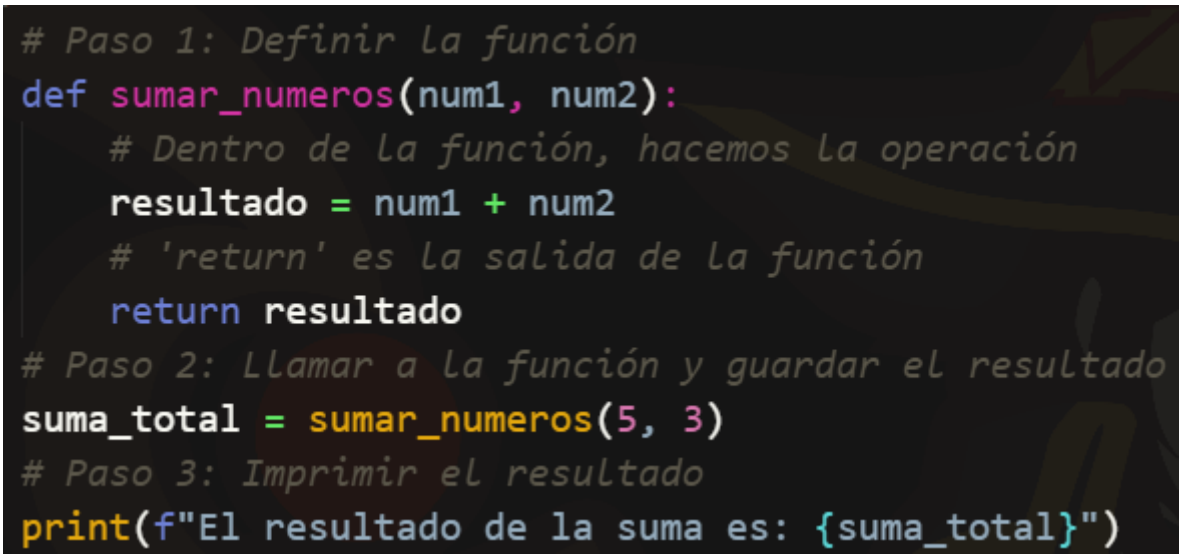
Diccionarios

Los diccionarios almacenan pares de clave-valor, siendo una estructura mutable. Las claves deben ser únicas y pueden ser de cualquier tipo inmutable.

```
mi_diccionario={'nombre': 'Juan', 'edad': 30, 'lenguajes': ['Python', 'JavaScript']}
```

Función

Una función en Python es un bloque de código reutilizable que agrupa instrucciones para realizar una tarea específica. Sirven para organizar el código en partes más simples, evitar la repetición y hacerlo más fácil de leer y mantener. Las funciones pueden recibir datos de entrada (argumentos), procesarlos y, opcionalmente, devolver un resultado.



```python
# Paso 1: Definir la función
def sumar_numeros(num1, num2):
    # Dentro de la función, hacemos la operación
    resultado = num1 + num2
    # 'return' es la salida de la función
    return resultado
# Paso 2: Llamar a la función y guardar el resultado
suma_total = sumar_numeros(5, 3)
# Paso 3: Imprimir el resultado
print(f"El resultado de la suma es: {suma_total}")
```

¿Qué es GitHub?

GitHub es una plataforma basada en la nube donde puedes almacenar, compartir y trabajar junto con otros usuarios para escribir código.
Almacenar tu código en un "repositorio" en GitHub te permite lo siguiente:

- Presentar o compartir el trabajo.
- Seguir y administrar los cambios en el código a lo largo del tiempo.
- Dejar que otros usuarios revisen el código y realicen sugerencias para mejorarlo.
- Colaborar en un proyecto compartido, sin preocuparse de que los cambios afectarán al trabajo de los colaboradores antes de que esté listo para integrarlos.

El trabajo colaborativo, una de las características fundamentales de GitHub, es posible gracias al software de código abierto Git, en el que se basa GitHub.

¿Qué es Docker?

Un contenedor es una unidad estándar de software que empaqueta el código y todas sus dependencias para que la aplicación se ejecute de forma rápida y fiable en cualquier entorno informático.

¿Qué es una imagen de contenedor Docker?

Es un paquete de software ligero, independiente y ejecutable que incluye todo lo necesario para ejecutar una aplicación: código, entorno de ejecución, herramientas del sistema, bibliotecas del sistema y configuración.

Las imágenes de contenedor se convierten en contenedores en tiempo de ejecución y, en el caso de los contenedores Docker, se convierten en contenedores al ejecutarse en Docker Engine . Disponible para aplicaciones basadas en Linux y Windows, el software en contenedores siempre se ejecutará de la misma manera, independientemente de la infraestructura. Los contenedores aíslan el software de su entorno y garantizan un funcionamiento uniforme a pesar de las diferencias, por ejemplo, entre entornos de desarrollo y de pruebas.

¿Qué es CSV?

La abreviatura "CSV" significa "valores separados por comas" (del inglés ‘comma separated value’) y describe la estructura de un archivo de texto con el que se intercambian y almacenan datos estructurados de forma sencilla. La información (valores) en los archivos CSV está separada por comas en lugar de almacenarse en columnas. Dependiendo del software y de la configuración del usuario, también son posibles puntos y comas, dos puntos, espacios, tabulaciones u otros caracteres. El formato CSV permite guardar listas y tablas de cualquier longitud, que también pueden leerse en Excel, entre otras cosas.

2. Objetivo del Trabajo

El proyecto es un Visor de Datos Geográficos y Demográficos de Países implementado en Python, utilizando la librería **tkinter** para la Interfaz Gráfica de Usuario (GUI). Sus objetivos principales pueden detallarse de la siguiente manera:

1. Obtener Datos Actualizados y Globales de una Fuente Externa

- ¿Qué hace? El programa utiliza la librería **requests** para conectarse a la API pública **restcountries.com**. Esta API sirve como la única fuente de verdad para toda la información.
- ¿Por qué es importante? Al depender de una API en vivo, el programa se asegura de que los datos de los países (como población, fronteras o nombres) sean lo más actualizados y precisos posible en el momento de la ejecución, en lugar de depender de archivos locales estáticos que podrían quedar obsoletos.
- Detalle: El módulo **generarPaises.py** gestiona las llamadas a esta API, solicitando la información región por región para asegurarse de cubrir todo el globo.

2. Consolidar y Preparar los Datos para el Análisis

Una vez obtenidos los datos en formato JSON, deben ser procesados para que sean útiles en la interfaz:

- Extracción y Selección: La API proporciona muchísima información. El programa solo extrae los campos que son relevantes para el usuario final, como el nombre en español, la población, el área y la capital, desechando el ruido.
- Normalización y Unificación: Los datos se descargan en archivos CSV separados por continente. La función **unir_csvs_en_uno** combina todos estos archivos en un único maestro llamado **Todos.csv**.
- Enriquecimiento de Datos: Durante la unificación, se realiza una transformación clave: se añade una columna llamada **continente** a cada registro. Este campo no viene directamente de la API, sino que se infiere del nombre del archivo original. Esto permite a la interfaz realizar filtros por continente y calcular estadísticas por región.

3. Presentar los Datos de Forma Interactiva y Usable (GUI)

El objetivo final es hacer que el gran volumen de datos sea accesible y fácil de explorar, lo cual se logra mediante el módulo **interfaz.py** que usa **tkinter**:

- Visualización Clara: Los datos son presentados en una tabla Treeview, que es el estándar para mostrar grandes conjuntos de datos de forma estructurada (Nombre, Población, Superficie).
- Interacción sin Programación: La GUI dota al usuario de herramientas de análisis inmediato que serían imposibles con un archivo de texto o CSV simple:
    - Búsqueda Rápida: Filtra los resultados mientras el usuario escribe.
    - Ordenamiento Dinámico: Permite hacer clic en un encabezado (como "Población") para reordenar instantáneamente la tabla.
    - Filtrado Compuesto: Facilita la aplicación de múltiples condiciones (ejemplo: "Países de Asia con Población entre 10 y 50 millones").

En resumen, el programa transforma un servicio web externo y complejo en una herramienta local, estructurada y potente para la consulta de información geográfica.

3. Diseño del caso práctico

El diseño del trabajo está dividido en tres fases principales, con responsabilidades bien definidas para cada módulo de Python:

Fase 1: Extracción y Almacenamiento
Esta fase se centra en obtener los datos crudos y guardarlos de forma temporal.

Módulo Principal	Responsabilidad	Detalle del Trabajo
main.py	Orquestación de Extracción	Define la lista de continentes y, mediante un bucle, le indica a **generarPaises.py** qué región debe procesar a continuación.
generarPaises.py	Orquestación de Extracción	Llama a la API (**restcountries.com**) y utiliza la librería **requests** para obtener los datos JSON de cada continente.
generarPaises.py	Almacenamiento Temporal	Filtra los campos JSON a sólo los relevantes (**Población, Área, Nombres, Capital**) y los guarda en archivos CSV individuales (ej. **Asia.csv**) en la carpeta **Continentes**.

Fase 2: Transformación
Esta es la fase de limpieza, enriquecimiento y consolidación del conjunto de datos.

Módulo Principal	Responsabilidad	Detalle del Trabajo
main.py	Orquestación de Transformación	Una vez completada la descarga de todos los continentes, llama a la función de unificación.
generarPaises.py	Unificación	Lee todos los CSV individuales de la carpeta **Continentes** y los concatena fila por fila.
generarPaises.py	Enriquecimiento de Datos	**Acción Clave**: Durante la unificación, añade la columna de metadato continente a cada fila, tomando el valor del nombre del archivo de origen. Esto es vital para las funcionalidades de filtrado de la GUI.
Archivo Final	Conjunto de Datos Maestro	Se genera el archivo **Todos.csv**, que es la única fuente de datos para la visualización.

Fase 3: Presentación e Interacción (GUI)
Esta fase se enfoca en la interfaz de usuario y la lógica de interacción para el análisis.

Módulo Principal	Responsabilidad	Detalle del Trabajo
main.py	Lanzamiento de la GUI	Inicia el bucle principal de **tkinter**, pasando el control a **interfaz.py**.
interfaz.py	Gestión de Datos en Memoria	Lee **Todos.csv** y carga el conjunto de datos completo en una lista de diccionarios (dataset_paises) para un acceso rápido.
interfaz.py	Visualización	Crea la ventana principal de **tkinter** y muestra los datos en el widget Treeview (la tabla).
interfaz.py	Lógica de Interacción	Implementa las funciones que permiten al usuario: Ordenar al hacer clic en las cabeceras, Buscar rápidamente por nombre, y aplicar Filtros Avanzados (población, área, continente) sobre el **dataset_paises**.

4. Metodología utilizada

1. Metodología de Procesamiento de Datos: ETL
La metodología central que define el flujo del programa es ETL (por sus siglas en inglés: Extract, Transform, Load), que en español significa Extracción, Transformación y Carga.
Este es un modelo estándar en el manejo de datos, donde cada fase se asigna a un módulo específico del código:

Fase ETL	Propósito	Módulo/Archivo Responsable	Detalles por Fase
Extracción	Obtener los datos crudos de la fuente	generarPaises.py	Se realiza una extracción activa al conectarse directamente a una fuente de datos externa (la API restcountries.com) mediante peticiones HTTP.
Transformación	Limpiar, filtrar y enriquecer los datos.	generarPaises.py	Esta es la fase más crítica. El programa no solo filtra los campos irrelevantes, sino que realiza un enriquecimiento de datos crucial: agrega la columna continente a cada registro, infiriendo el valor del nombre del archivo temporal.
Carga (Load)	Mover los datos transformados al destino final para su uso.	main.py / interfaz.py	La carga tiene dos etapas: 1.Carga inicial de los datos transformados al archivo maestro Todos.csv. 2.Carga final del Todos.csv en la memoria de la aplicación (dataset_paises) para su manipulación en la Interfaz Gráfica.

2. Metodología de Diseño de Código: Modular

La arquitectura del código sigue una metodología Modular, donde las responsabilidades del sistema se dividen claramente en archivos separados:

- **main.py** actúa como el orquestador (responsabilidad de control).
- **generarPaises.py** se encarga de la lógica de datos y archivos (responsabilidad de servicios).
- **interfaz.py** se encarga de la lógica de presentación y visualización (responsabilidad de la interfaz de usuario).

3. Metodología de Despliegue: Contenerización

El uso del archivo **dockerfile** implementa una metodología de Contenerización.
Objetivo: Garantizar que el programa se ejecute de manera idéntica y reproducible en cualquier entorno (**desarrollo, prueba o producción**), aislando las dependencias de Python (**requirements.txt**) y de sistema operativo (**tk para tkinter**).
Ventaja: Permite que la aplicación sea portable y minimiza los problemas de "en mi máquina funciona".

5. Resultados Obtenidos

El resultado principal del trabajo se divide en dos categorías: El **Producto de Datos Consolidado** y la **Herramienta de Análisis Operativo.**

1. Producto de Datos Consolidado
El resultado más fundamental y permanente del trabajo es la generación de un conjunto de datos limpio y listo para el análisis:
Archivo Maestro Todos.csv

- **Descripción:** Un único archivo CSV que contiene la información esencial de todos los países extraídos de la API.
- **Valor Añadido:**
    - **Consolidación:** Reemplaza múltiples archivos individuales (ej. **Africa.csv, Europe.csv**) con una fuente de datos única y confiable.
    - **Enriquecimiento:** Incluye la columna continente, que no estaba disponible en el formato original de la API. Esta columna es crucial para las funcionalidades de filtrado y estadísticas de la aplicación.
    - **Limpieza:** Los datos están filtrados, conteniendo solo los campos relevantes para el usuario final (Nombre, Población, Superficie, Capital), eliminando la información redundante de la fuente original.

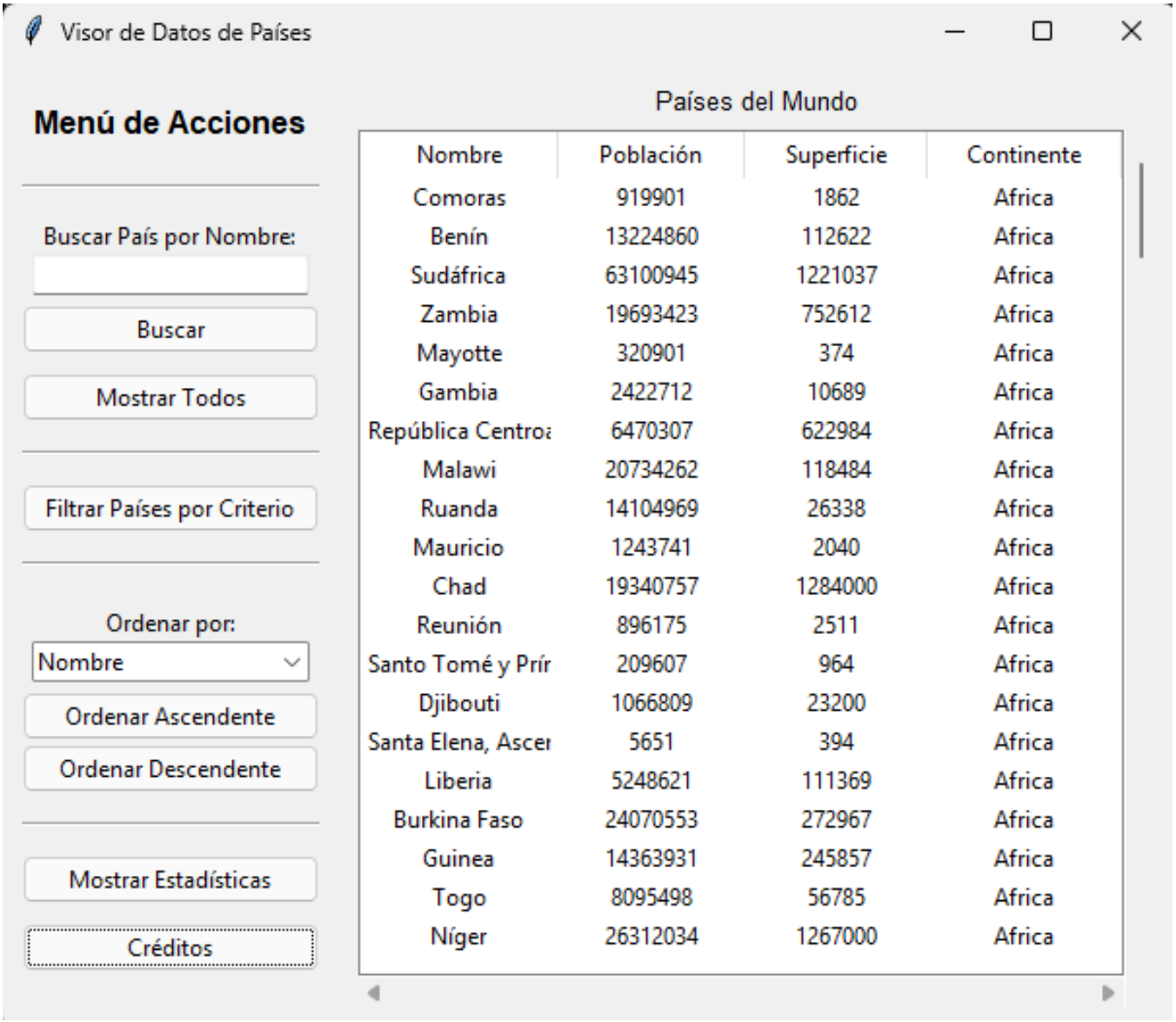
2. Herramienta de Análisis Operativa
El resultado final, visto por el usuario, es una aplicación completa de escritorio que permite interactuar con el producto de datos consolidado:
Aplicación de Escritorio con GUI (Interfaz Gráfica)
El programa es una herramienta funcional que ofrece análisis y navegación de datos de manera interactiva:

Funcionalidad Clave	Impacto en el Usuario
Visualización Centralizada	Presenta los cientos de registros en una tabla Treeview, facilitando la navegación y lectura.
Búsqueda y Ordenamiento Dinámico	Permite a los usuarios encontrar países de forma rápida (escribiendo) u ordenar la lista instantáneamente por métricas (Población, Superficie) con un solo clic.
Filtrado Analítico Avanzado	Permite realizar consultas complejas mediante la combinación de tres criterios (continente, rango de población y rango de superficie), lo cual permite segmentar y analizar subconjuntos de datos específicos.
Estadísticas Automáticas	Genera y muestra métricas globales (promedios, máximos, conteos) que resumen el conjunto de datos de manera inmediata, eliminando la necesidad de cálculos manuales.

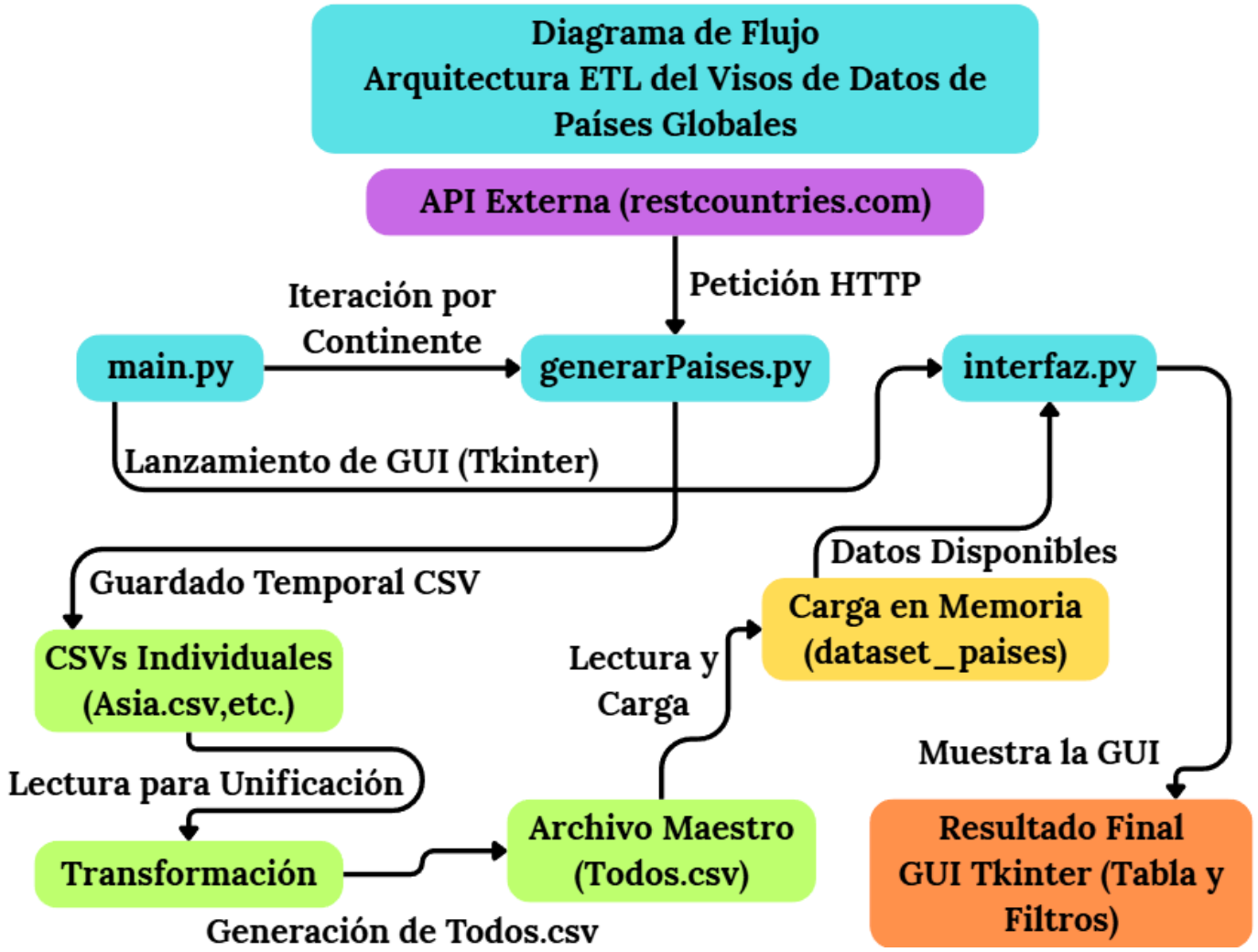
3. Resultado Metodológico

Además del producto final, se obtiene un resultado de desarrollo de alta calidad:

- **Arquitectura Modular y Reutilizable:** El trabajo resultó en un código estructurado donde las responsabilidades están separadas (**generarPaises.py** para datos, **interfaz.py** para GUI). Esto facilita el mantenimiento, la prueba y la futura expansión del programa.
- **Entorno Contenedorizado:** El uso de dockerfile garantiza que el programa pueda ser ejecutado en cualquier sistema operativo sin fallos de dependencia, asegurando la portabilidad y reproducibilidad del trabajo.



6. Diagrama de Flujo



7. Conclusión

Nuestra conclusión es que el trabajo es una demostración práctica de cómo Python sirve perfectamente para conectar servicios de datos externos con aplicaciones de escritorio amigables. La portabilidad del proyecto confirma que Python es la herramienta ideal para construir soluciones de software modulares, mantenibles y preparadas para el despliegue en el ámbito del procesamiento y visualización de datos.

Bibliografía

https://www.oracle.com/latam/developer/what-is-python-for-developers

https://openwebinars.net/blog/fundamentos-de-python-sintaxis-variables-y-estructuras-de-control/#:~:text=Las%20estructuras%20de%20control%20en%20Python%20incluyen,repetir%20un%20bloque%20de%20código%20múltiples%20veces.

https://docs.github.com/es/get-started/start-your-journey/about-github-and-git

https://www-docker-com.translate.goog/resources/what-container/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

https://www.adobe.com/es/acrobat/resources/document-files/text-files/csv-file.html