# Trabajo Practico Integrador

Gestión de Datos de Países en Python

Ignacio Beltramino Martinez – Cristhian Santiago Gimenez
Grupo 130
PROGRAMACIÓN I

UNIVERSIDAD TECNOLOGICA NACIONAL

**Índice**

## Marco Teórico

### Estructuras de datos

Nuestro programa cuenta con una lista principal llamada “lista_paises” que esta compuesta por diccionarios, que cada uno de estos sería un país en nuestro sistema. Cada diccionario incluye cuatro claves fundamentales:

- Nombre: representa el nombre del país (por ejemplo: Argentina, Italia)
- Población: indica la cantidad de habitantes del país, es un entero positivo.
- Superficie: Representa la cantidad de territorio que tiene un país, también es un entero positivo.
- Continente: nos indica el continente al que pertenece el país. (por ejemplo: América, África)

Estas estructuras nos permiten resolver todas las problemáticas que nos vamos encontrando en el trabajo, ya que podemos ordenar en el CSV de forma alfabética, realizar estadísticas, realizar altas bajas y modificaciones sobre la información de los países.

### Lectura del archivo CSV:

A partir de la experiencia adquirida durante la realización del segundo parcial de la materia, comprendimos con mayor claridad cómo abordar el manejo correcto de archivos CSV dentro de nuestro programa.

Para lograrlo, desarrollamos dos funciones específicas (que serán detalladas más adelante), las cuales se encargan de la lectura y escritura del archivo CSV de forma controlada. Gracias a esta implementación, el archivo se mantiene actualizado y

no se pierde información entre sesiones, lo que asegura la integridad del sistema y mejora la experiencia del usuario.

**Funciones creadas:**

Para poder desarrollar todo el trabajo, creamos las siguientes funciones, todas con un uso especifico que nos permite avanzar de forma más ordena y concisa en las consignas.

- **“obtenerPaises”:** esta función se encarga de cargar en la lista principal los países, esto lo hace directamente desde el CSV, en caso de que el mismo no exista, lo crea, entonces es funcional siempre, ya sea que tengamos un archivo previo o no.
- **“guardarPaises”:** esta función se llama cada vez que se hace un cambio en la lista principal, por ejemplo, cuando se agrega un país a la lista, llamamos la función para efectuar el cambio en el archivo CSV.
- **“normalizarString”:** el uso que se le da a esta función, es llamarla cada ves que se le pide al usuario que ingrese un dato que debe ser del tipo STRING, lo que hace la misma es validad que efectivamente se haya ingresado un string y no un número (ya que son nombres de países y no tienen que tener números) además de esto, la función convierte todo en minúscula para un mayor control de los datos.
- **“normalizarInt”:** esta función hace exactamente lo mismo que la anterior solo que es para los datos que deben ser del tipo INT, valida que no sean

números menores que 0, ya que en el contexto que estamos trabajando, no puede haber

de ninguna forma números negativos y además valida que no se le ingresen strings.

- **“paisYaExiste”:** se le pasa como parámetro el nombre de un país, valida en toda la lista si este existe, en caso de que no, devuelve “False” y en caso contrario devuelte “True”.
- **“buscarCoincidenciasPorNombre”:** esta función la creamos para poder conseguir una lista de coincidencias, es decir, se la pasa por parámetro el nombre del país buscado, la lista de países de esta forma, se lista todos los países que tengan una coincidencia parcial el nombre ingresado.
- **“filtrarPorContinente(continente, lista_paises)”:** Esta función se encarga de **filtrar los países que pertenecen a un continente determinado**. Primero normaliza el nombre del continente para evitar errores de mayúsculas o espacios, y luego recorre la lista principal agregando a una nueva lista todos los países cuyo campo "CONTINENTE" coincide con el ingresado. Finalmente, devuelve esa nueva lista filtrada.
- **“filtroPorRangoPoblacion(lista_paises,rango_minimo, rango_maximo)”:** Su función es **filtrar los países cuya población se encuentre dentro de un rango específico**, es decir, entre un valor mínimo y máximo definidos por el usuario.

  Recorre la lista principal y va agregando solo los países que cumplen la condición:

rango_minimo <= POBLACION <= rango_maximo. Devuelve una lista con todos los países que se encuentran dentro de ese rango.

- **“filtoPorRangoSuperficie(lista_paises, rango_minimo, rango_maximo)”**: Cumple la misma lógica que la anterior, pero aplicada al campo "SUPERFICIE".

  Filtra los países que tienen una superficie comprendida entre un mínimo y un máximo ingresado, devolviendo una nueva lista con los resultados.
- **“obtenerNombre(pais)”:** Devuelve el nombre del país pasado por parámetro. Se utiliza como función auxiliar dentro de sorted() para poder ordenar la lista de países según su nombre.
- **“obtenerPoblacion(pais)”:** Devuelve la población del país recibido como parámetro.

  También es una función auxiliar usada como clave (key) para ordenar los países por cantidad de habitantes.
- **“ordenoPaisPorNombre(lista_paises)”:** Ordena la lista principal de países alfabéticamente por nombre de forma ascendente (de la A a la Z). Para ello utiliza la función sorted() junto con key=obtenerNombre.
- **“ordenoPaisPorPoblacion(lista_paises, opcion)”:** Ordena la lista de países según su población.

  Si la opción es 1, los ordena de forma ascendente (de menor a mayor población).

  Si la opción es 2, los ordena de forma descendente (de mayor a menor población).

  El ordenamiento se realiza usando sorted() con la función auxiliar obtenerPoblacion.

- **“ordenPaisPorSuperficie(lista_paises, opcion)”:** Ordena los países por su superficie total:

  Si la opción es 1, en orden ascendente (de menor a mayor superficie).

  Si la opción es 2, en orden descendente (de mayor a menor superficie). La lógica es la misma que la del ordenamiento por población, pero aplicada al campo "SUPERFICIE".

- **“mayorPoblacion(lista_paises)”:** Recorre la lista de países y devuelve el país con mayor cantidad de habitantes.
  Para lograrlo, compara el campo "POBLACION" de cada país con el valor actual más alto, actualizándolo si encuentra uno mayor.

- **“menorPoblacion(lista_paises)”:** Hace el proceso inverso: busca y devuelve el país con menor cantidad de población.
  Compara todos los valores del campo "POBLACION" y se queda con el más pequeño.

- **“promedioPoblacion(lista_paises)”:** Calcula el promedio general de población entre todos los países de la lista.
  Para ello, acumula la cantidad total de habitantes (acum) y cuenta la cantidad de países (cont), luego divide ambos valores: promedio = acum / cont.

- **“promedioSuperficie(lista_paises)”:** Calcula el promedio de superficie de todos los países.
  Funciona igual que la anterior, pero tomando el campo "SUPERFICIE" en lugar de "POBLACION".

- **“cantPaisesPorContinente(lista_paises)”:** Cuenta cuántos países hay en cada continente.

Inicializa contadores separados para América, Europa, Asia, Oceanía y África.

Por cada país, incrementa el contador correspondiente según el valor de "CONTINENTE".

Al final, devuelve una lista con los resultados, donde cada elemento representa la cantidad de países por continente**.**

**Ejemplo de entrada y salida**

A continuación, se presentan algunos ejemplos de cómo funciona el programa en diferentes situaciones:

Ejemplo 1 – Filtrado por continente

- **Entrada:** América
- **Salida:**

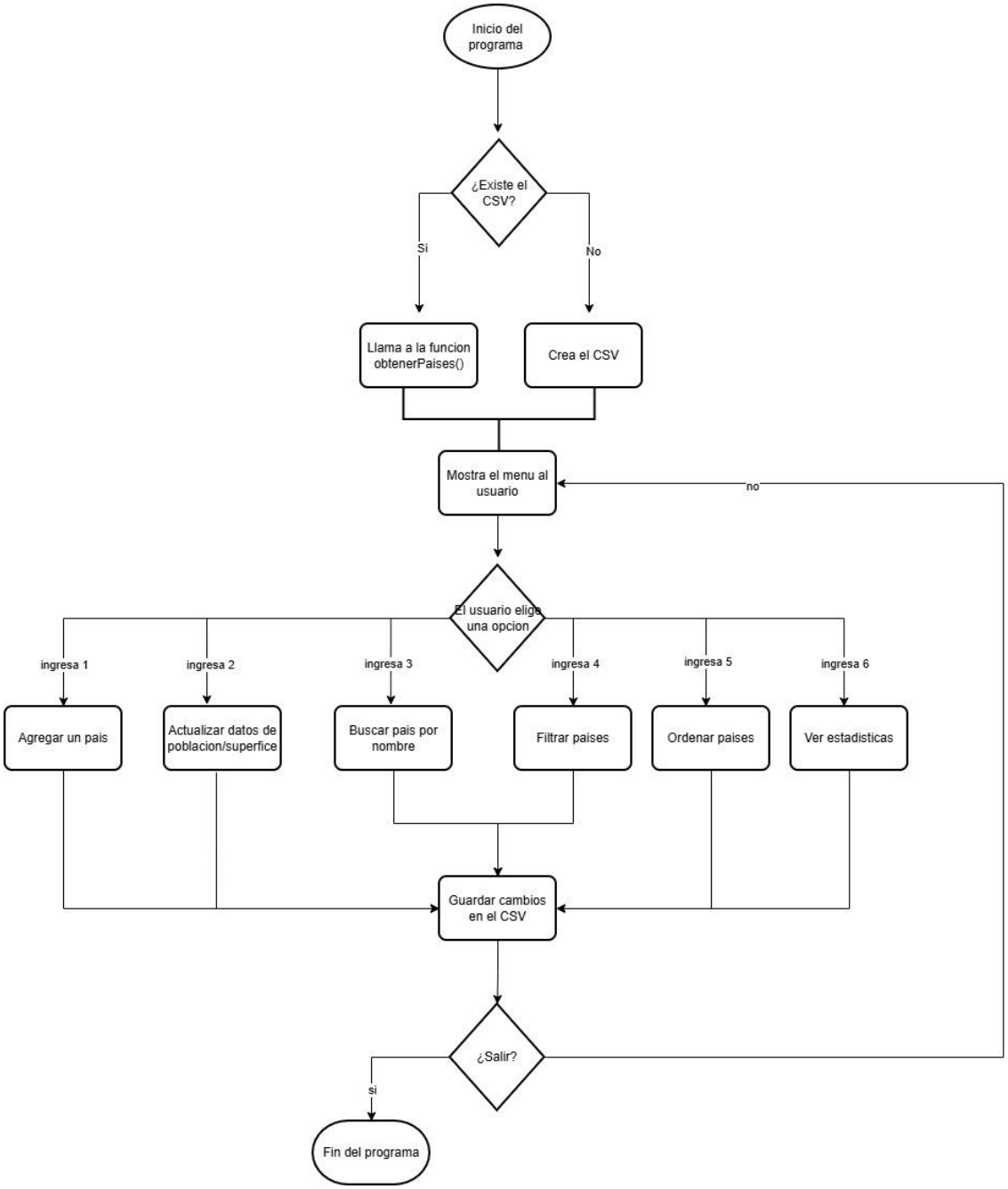
Países en América

- Argentina | Población: 45.000.000 | Superficie: 2.780.400 km²

- Brasil | Población: 212.000.000 | Superficie: 8.516.000 km²

- Chile | Población: 19.000.000 | Superficie: 756.000 km²

**Definir el flujo de operaciones principales diagrama:**



**Objetivo del trabajo:**

El objetivo de este trabajo fue crear un programa que nos permita gestionar información sobre distintos países: cargarlos, modificarlos, buscarlos y guardarlos de forma segura. Para eso usamos una lista principal con diccionarios, y trabajamos con archivos CSV para que los datos no se pierdan, aunque cerremos el programa.

Además, aprovechamos lo aprendido en la cursada para aplicar validaciones, organizar el código en funciones y hacer que el sistema sea fácil de usar y mantener. La idea fue resolver todas las consignas de forma ordenada, usando buenas prácticas y asegurando que el programa funcione bien en distintos escenarios.

**Diseño del caso práctico y metodología utilizada:**

El desarrollo lo fuimos armando paso a paso. Primero nos enfocamos en que el programa pudiera leer y escribir correctamente en el archivo CSV, algo que ya habíamos practicado en el segundo parcial. Después empezamos a crear funciones específicas para validar los datos que se ingresan, como nombres, números enteros positivos, y evitar errores comunes.

A medida que avanzábamos, fuimos modularizando el código: cada función cumple una tarea puntual, lo que nos permitió mantener el orden y facilitar la lectura. También agregamos funciones para buscar países por coincidencia parcial, verificar si ya existen en la lista, y mostrar los resultados de forma clara para que el usuario pueda elegir.

La metodología fue bastante práctica: probamos cada parte del programa a medida que la íbamos desarrollando, ajustando lo que no funcionaba y mejorando la interacción con el

usuario. El objetivo fue que el sistema sea fácil de usar, que los datos estén bien controlados, y que el archivo se mantenga siempre actualizado.

**Resultados obtenidos y conclusión:**

El trabajo nos dejó una experiencia muy positiva. Pudimos cumplir con todas las consignas y validaciones que se pedían, y además logramos organizar el código de forma correcta, separando las funciones en un archivo y el programa principal en otro, lo que nos ayudó a mantener todo más ordenado.

Más allá de lo técnico, fue un proyecto que disfrutamos mucho. Nos permitió aplicar todo lo que fuimos aprendiendo durante la cursada: desde el manejo de archivos hasta la validación de datos y la estructura del programa. También nos dejó varias enseñanzas sobre cómo planificar mejor, trabajar con funciones reutilizables y pensar en cómo hacer que el sistema sea más claro para el usuario.

En resumen, fue un trabajo completo que nos ayudó a reforzar conocimientos, ganar confianza y entender mejor cómo aplicar la programación en casos prácticos**.**