

## **Trabajo Práctico Integrador (TPI)**

**Gestión de Datos de Países en Python: filtros, ordenamientos y estadísticas**

### **Alumnos - Grupo n° 6**

Avalo, Pablo – Suárez, Ismael.

**Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional.**

**Programación 1 - 2025**

### **Docente Titular**

Cintia, Rigoni

### **Docente Tutor**

Ramiro, Hualpa

3 de noviembre de 2025

## Tabla de contenido

Trabajo Práctico Integrador (TPI) .....	1
Tabla de contenido .....	2
Introducción.....	4
Objetivos.....	5
Consignas generales.....	5
Dominio (dataset de países).....	5
Requerimientos técnicos.....	6
1) Diseño (previo al código).....	6
2) Funcionalidades mínimas del sistema .....	6
3) Validaciones .....	7
Entregables (obligatorios).....	7
1. Carpeta digital .....	7
2. Repositorio en GitHub.....	7
3. Video tutorial (10–15 minutos).....	8
Criterios de evaluación.....	8
Desarrollo .....	9
1-Diseño (previo al código).....	9
2-Funcionalidades mínimas del sistema .....	10
Estructura:.....	10

Selección de servidor .....	10
Menú .....	11
3-Validaciones .....	11
Funcionamiento .....	12
Conclusión.....	14
Referencias .....	15
-Video Presentación YouTube: .....	15
<a href="https://www.youtube.com/watch?v=Yjla5Ayn3vM">https://www.youtube.com/watch?v=Yjla5Ayn3vM</a> .....	15

## Trabajo Integrador Programación 1 -2025

### Introducción

En el marco de la asignatura Programación 1, desarrollamos una aplicación de consola en Python para la consulta y administración de datos de países. Nuestra solución integra un menú principal y dos modos de operación —local (archivo CSV) y remoto (API REST)— que nos permitieron ejercitarn la persistencia en archivos y el consumo de servicios HTTP. Implementamos búsquedas, filtros, ordenamientos, estadísticas y un CRUD para altas, bajas y modificaciones (ABM). Priorizamos una arquitectura modular con funciones en español y documentamos el código mediante docstrings estilo Google, favoreciendo la legibilidad y la mantenibilidad. Si bien el desarrollo responde a los objetivos iniciales del cursado, dejamos una base sólida para incorporar validaciones, pruebas automatizadas y mejoras evolutivas en futuras iteraciones.

## Objetivos

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.

## Consignas generales

- Lenguaje: Python 3.x
- Estructuras: listas, diccionarios, funciones.
- Archivos: lectura desde CSV.
- Código claro, comentado y modularizado (una función = una responsabilidad).
- Validaciones de entradas y manejo básico de errores.
- Trabajo en equipos de 2 personas.

## Dominio (dataset de países)

Cada país estará representado con los siguientes datos:

- Nombre (string)
- Población (int) • Superficie en km<sup>2</sup> (int)
- Continente (string) Ejemplo de registro CSV:

nombre, población, superficie, continente  
Argentina, 45376763, 2780400, América  
Japón, 125800000, 377975, Asia  
Brasil, 213993437, 8515767, América  
Alemania, 83149300, 357022, Europa

## Requerimientos técnicos

### 1) Diseño (previo al código)

Explicar en un informe teórico los conceptos aplicados:

- o Listas
- o Diccionarios
- o Funciones
- o Condicionales
- o Ordenamientos
- o Estadísticas básicas
- o Archivos CSV • Definir el flujo de operaciones principales en un diagrama o esquema.

### 2) Funcionalidades mínimas del sistema

El programa debe ofrecer un menú de opciones en consola que permita:

- Buscar un país por nombre (coincidencia parcial o exacta).
- Filtrar países por: o Continente o Rango de población o Rango de superficie
- Ordenar países por:
  - o Nombre
  - o Población
  - o Superficie (ascendente o descendente)
- Mostrar estadísticas:
  - o País con mayor y menor población
  - o Promedio de población

- o Promedio de superficie
- o Cantidad de países por continente

### **3) *Validaciones***

- Controlar errores de formato en el CSV.
- Evitar fallos al ingresar filtros inválidos o búsquedas sin resultados.
- Mensajes claros de éxito/error.

### **Entregables (obligatorios)**

#### **1. *Carpeta digital***

- Marco teórico con fuentes bibliográficas.
- Código Python funcional, modular y comentado.
- Capturas de pantalla de ejecución de ejemplos.
- Conclusiones grupales sobre los aprendizajes.

#### **2. *Repositorio en GitHub***

Debe incluir:

- Proyecto completo en Python.
- README.md con:
  - o Descripción del programa.
  - o Instrucciones de uso.
  - o Ejemplos de entradas y salidas.
  - o Participación de los integrantes.
- Archivo CSV con el dataset base.

### ***3. Video tutorial (10–15 minutos)***

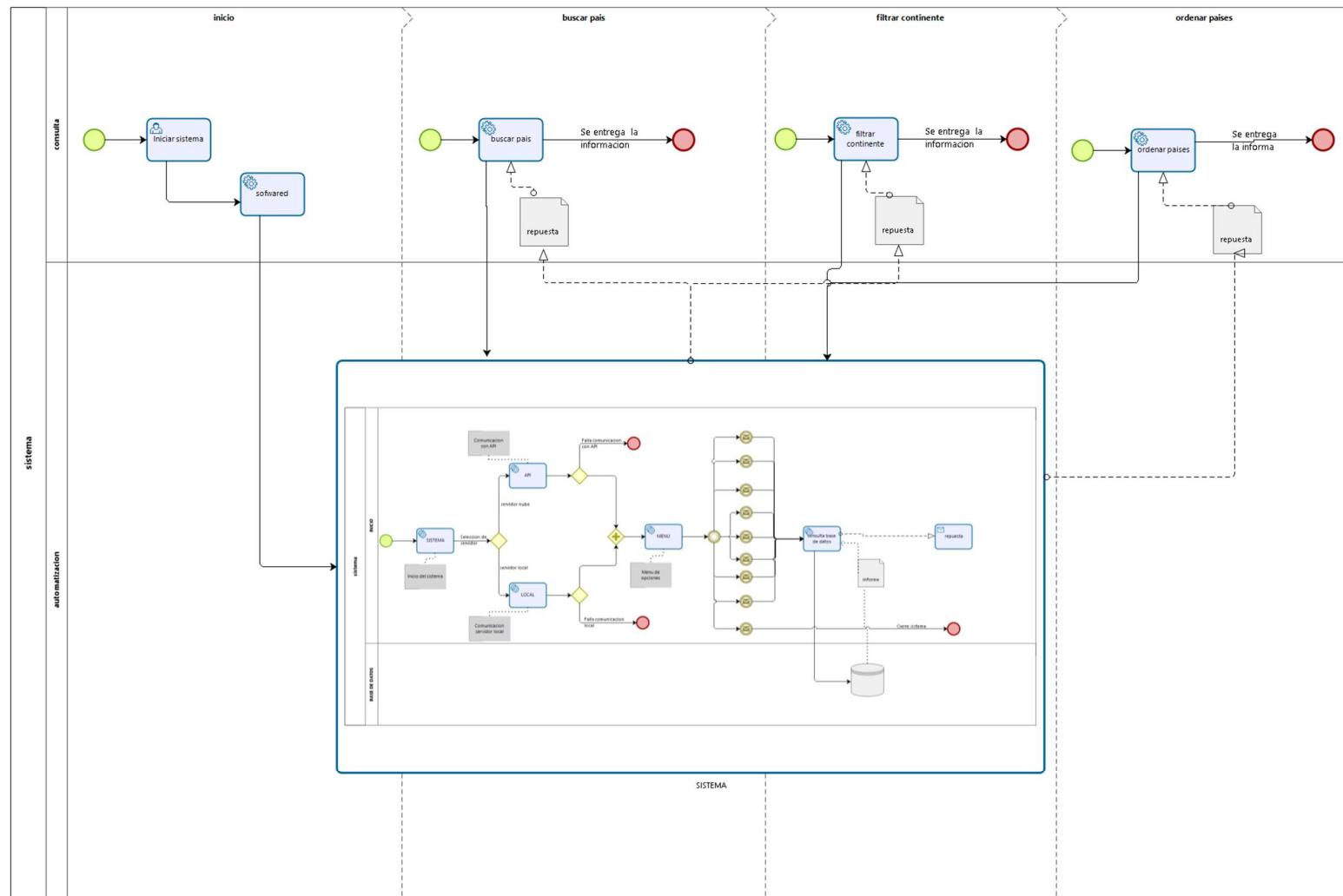
- Explicación del problema planteado.
- Presentación de la estructura de datos utilizada.
- Demostración del programa funcionando.
- Reflexión final sobre el desarrollo del proyecto.

### ***Criterios de evaluación***

- Correcta funcionalidad (búsquedas, filtros, ordenamientos, estadísticas).
- Uso correcto de estructuras de datos (listas y diccionarios).
- Calidad del código (modularización, legibilidad, comentarios).
- Documentación (README claro, informe teórico coherente).
- Presentación en video (tiempo adecuado, explicación técnica, participación equitativa).
- Entrega completa en GitHub con código, informe y CSV.

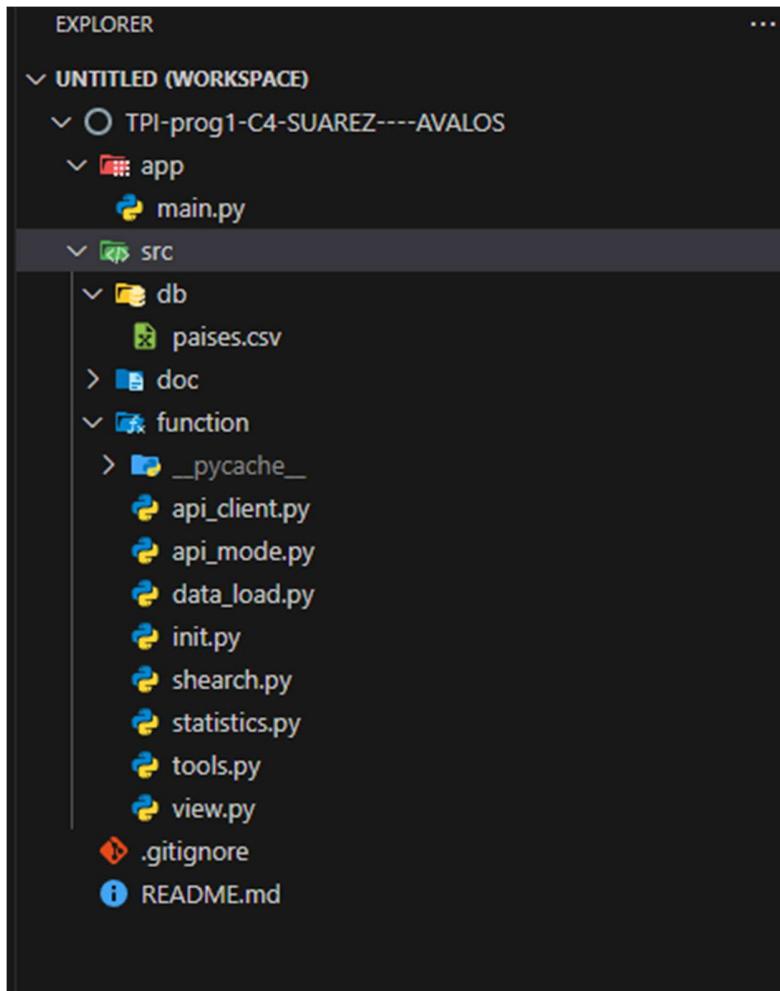
## Desarrollo

### 1-Diseño (previo al código)



## 2-Funcionalidades mínimas del sistema

### Estructura:



### Selección de servidor

```
*****  
○ [UP] Iniciando aplicacion...  
✓ Sistema cargado  
****Seleccione el servidor****  
1. CSV local [ ]  
2. CSV API [ ]  
3. Salir [ ]  
Elegí 1 o 2 : [ ]
```

## Menú

```
*****INFO GEOGRAFICO*****
1. Buscar pais por nombre
2. Filtrar por continente
3. Filtrar por rango de poblacion
4. Filtrar por rango de superficie
5. Ordenar paises
6. Mostrar estadisticas
7. Agregar un pais
8. Editar poblacion y superficie de un pais
9. Borrar pais
10. Cambiar modo de servidor
11. Salir
Ingresar una opcion 1-11: 
```

## 3-Validaciones

```
def buscar_pais(paises, nombre):
    encontrado=[p for p in paises if nombre in p["nombre"].lower()]
    continentes= set(p["continente"] for p in paises)
    print("Continentes disponibles:",", ".join(continentes))

    if encontrado:
        for p in encontrado:
            print(p)
    else:
        print("No se encontró país con ese nombre.")

def filtrar_continente(paises, continente):
    continente_normalizado = normalizar(continente)
    resultados = [p for p in paises if continente_normalizado in normalizar(p["continente"])]
    if resultados:
        print(f"\n Paises en el continente '{continente}':")
        mostrar_paises(resultados)
    else:
        print(f"\n No se encontraron países en el continente '{continente}'.")

def filtrar_poblacion(paises):
    minimo = _leer_entero_no_negativo("Ingrese la población mínima: ")
    if minimo is None:
        return
    maximo = _leer_entero_no_negativo("Ingrese la población máxima: ")
    if maximo is None:
        return

    if minimo > maximo:
        print("El mínimo no puede ser mayor que el máximo.")
    return
```

## Funcionamiento

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

○ *****Estadísticas generales*****
  □ País con mayor población: China (1,412,600,000 hab.)
  □ País con menor población: Mónaco (39,242 hab.)
  □ Promedio de población: 54,214,735 hab.
  □ Promedio de superficie: 885,762.6043165468 km²

*****Cantidad de países por continente*****
  - Asia: 35
  - Europa: 39
  - África: 34
  - América: 27
  - Oceanía: 2
  - Europa/Asia: 1
  - Asia/Europa: 1
*****INFO GEOGRAFICO*****
1. Buscar pais por nombre
2. Filtrar por continente
3. Filtrar por rango de poblacion
4. Filtrar por rango de superficie
5. Ordenar paises
6. Mostrar estadisticas
7. Agregar un pais
8. Editar poblacion y superficie de un pais
9. Borrar país
10. Cambiar modo de servidor
11. Salir
Ingrese una opcion 1-11: []

```

- Se realiza la búsqueda de la estadística de los países cargados en el csv

Código:

```

def mostrar_estadisticas(paises):
    """Imprime estadísticas generales de una lista de países.

Calcula y muestra:
    - País con mayor población.
    - País con menor población.
    - Promedio de población y de superficie.
    - Cantidad de países por continente (agrupando por nombre normalizado).

```

La función imprime por consola y no modifica la lista recibida.

Args:

```

paises (list[dict]): Lista de países. Cada país debe contener las
                     claves:

```

```
- 'nombre' (str)
- 'poblacion' (int)
- 'superficie' (float | int)
- 'continente' (str)

Returns:
    None
"""
if not paises:
    print(" No hay datos disponibles para mostrar estadísticas.")
    return
pais_mayor = max(paises, key=lambda x: x["poblacion"])
pais_menor = min(paises, key=lambda x: x["poblacion"])

promedio_poblacion = sum(p["poblacion"] for p in paises) / len(paises)
promedio_superficie = sum(p["superficie"] for p in paises) / len(paises)
paises_por_continente = {}
nombre_muestra_por_normalizado = {}

for p in paises:
    cont_norm = normalizar(p["continente"])
    paises_por_continente[cont_norm] = paises_por_continente.get(cont_norm,
0) + 1
    if cont_norm not in nombre_muestra_por_normalizado:
        nombre_muestra_por_normalizado[cont_norm] = p["continente"]

print("*****Estadísticas generales*****")
print(f"  País con mayor población: {pais_mayor['nombre']}")
({pais_mayor['poblacion']:,} hab.)")
print(f"  País con menor población: {pais_menor['nombre']}")
({pais_menor['poblacion']:,} hab.)")
print(f"  🌎 Promedio de población: {int(promedio_poblacion):,} hab.")
print(f"  🌎 Promedio de superficie: {float(promedio_superficie):,} km²")
print("")
print("*****Cantidad de países por continente*****")
for cont_norm, cantidad in paises_por_continente.items():
    cont_muestra = nombre_muestra_por_normalizado.get(cont_norm, cont_norm)
    print(f"      - {cont_muestra}: {cantidad}")
print("*****")
```

## Conclusión

Como equipo, integramos los conceptos vistos en clase en una solución coherente y funcional. Leímos y persistimos datos en CSV, replicamos las mismas operaciones contra una API remota y unificamos la experiencia en una interfaz de consola simple. Este recorrido nos permitió consolidar el manejo de tipos de datos, control de flujo, funciones y colecciones, además de normalizar entradas y separar responsabilidades (vista, lógica y utilidades). Como próximos pasos, proponemos elevar la calidad con pruebas unitarias y automatizadas, validaciones más estrictas (límites y formatos), manejo de errores más detallado, registro de eventos (logging) y empaquetado para distribución. Consideramos que el trabajo cumplió los objetivos de aprendizaje y nos deja una base firme para seguir avanzando.

### Referencias

-Tecnicatura Universitaria en Programación.

Programación 1-2025. Unidad 5 Universidad Tecnológica Nacional.

### -Video Presentación YouTube:

<https://www.youtube.com/watch?v=Yjla5Ayn3vM>