

TRABAJO INTEGRADOR FINAL - PROGRAMACIÓN

“Gestión de Datos de Países en Python”

Tecnicatura Universitaria en Programación – UTN

Integrantes:

-Casto Ricardo Gil Cañizalez

Comisión 5

Correo: casto782@gmail.com

-Alejo Almada

Comisión 1

Correo: alejonahuelalmada@gmail.com

Fecha de entrega: noviembre de 2025

Índice

1. Introducción
2. Objetivos del Trabajo
3. Marco Teórico
4. Diseño del Caso Práctico
5. Metodología de Desarrollo
6. Resultados Obtenidos
7. Conclusiones Grupales
8. Fuentes Bibliográficas, Repositorio y Recursos Digitales

1. Introducción

El presente Trabajo Práctico Integrador tiene como finalidad aplicar los conocimientos adquiridos a lo largo del cursado de Programación 1, mediante el desarrollo de una aplicación en Python que permita gestionar información de países de manera modular, validada y persistente.

El proyecto propone la resolución de un caso práctico de tipo administrativo, simulando un sistema que almacena, consulta y analiza datos de una colección de países. Para ello se utilizaron estructuras de datos como listas y diccionarios, el manejo de archivos CSV para la persistencia de la información, y la implementación de un menú de opciones en consola para interactuar con el usuario.

Además, se incorporaron buenas prácticas de programación: modularización del código, validaciones bloqueantes y control de errores mediante verificaciones manuales, evitando el uso de try/except, en línea con las pautas establecidas por la cátedra.

Este trabajo fue realizado de forma colaborativa por **Casto Gil (Comisión 5)** y **Alejo Almada (Comisión 1)**, quienes distribuyeron las funciones por módulos y gestionaron el código mediante *GitHub*, empleando ramas individuales y revisiones cruzadas (*peer review*) para asegurar la integración y la calidad del proyecto.

2. *Objetivos del Trabajo*

- Desarrollar una aplicación modular en Python que permita administrar información de países mediante un menú interactivo en consola.
- Implementar funciones específicas para agregar, modificar, buscar, filtrar, ordenar y generar estadísticas sobre los datos.
- Aplicar estructuras de control (if, while) con loops bloqueantes, garantizando que las entradas del usuario sean válidas antes de continuar la ejecución.
- Utilizar el manejo de archivos CSV para lograr la persistencia, actualización y almacenamiento permanente de los registros.
- Implementar control de errores mediante validaciones manuales, ofreciendo mensajes claros y comprensibles al usuario final.
- Fomentar el trabajo colaborativo con Git y GitHub, aplicando control de versiones a través de ramas independientes y revisiones cruzadas (*peer review*).
- Cumplir con los criterios de evaluación establecidos por la cátedra, priorizando la modularización, la validación de datos, la legibilidad del código y una documentación técnica clara y completa.

3. Marco Teórico

El proyecto se fundamenta en los conceptos de programación estructurada, modularización y validación de datos, pilares centrales del enfoque de enseñanza de Programación 1 en la UTN.

Estructuras de datos utilizadas

Se emplearon listas y diccionarios como estructuras principales:

- *Las listas* permitieron almacenar el conjunto de países.
- *Los diccionarios* representaron cada país, con claves como nombre, población, superficie y continente.

Modularización

El código se organizó en múltiples módulos para cumplir con la rúbrica de legibilidad y separación lógica:

- **datos.py**: Lectura, escritura y actualización del CSV.
- **validaciones.py**: Control de entradas del usuario.
- **ordenamiento.py**: Ordenamientos manuales (alfabético y numérico).
- **busquedas.py y filtros.py**: Consultas dinámicas sobre los datos.
- **estadisticas.py**: Cálculos de promedios, máximos y mínimos.
- **main.py**: Menú principal y flujo general del programa.

Control de flujo y validaciones:

El código utiliza bucles bloqueantes (while) y verificaciones condicionales (isdigit(), isalpha(), strip()) para garantizar entradas válidas, evitando interrupciones o comportamientos inesperados.

No se emplea try/except, ya que la validación se realiza de forma estructurada y controlada.

Control de versiones

El proyecto se gestionó con GitHub, aplicando un flujo de trabajo basado en ramas (rama-casto, rama-almada), commits descriptivos y revisiones por pares antes de la fusión al branch principal (main).

4. Diseño del Caso Práctico

El sistema desarrollado permite al usuario gestionar un listado de países mediante un **menú interactivo en consola**, aplicando una estructura modular que separa las responsabilidades de cada componente.

Cada opción del menú está asociada a una **función independiente**, lo que facilita la lectura, el mantenimiento y la escalabilidad del programa.

Opciones principales del sistema:

1. **Agregar país:** permite ingresar un nuevo país validando que no exista y que los campos sean correctos.
2. **Actualizar país:** modifica la población o superficie de un país existente.
3. **Buscar país:** realiza coincidencias exactas o parciales según el nombre ingresado.
4. **Filtrar países:** aplica filtros por continente o por rangos de población y superficie.
5. **Ordenar países:** ordena los registros de forma ascendente o descendente según distintos criterios (nombre, población o superficie).
6. **Mostrar estadísticas:** calcula y muestra máximos, mínimos, promedios y conteos por continente.
7. **Guardar cambios:** persiste los datos actualizados en el archivo CSV.
8. **Salir:** finaliza el programa mostrando un mensaje de cierre.

El programa utiliza validaciones bloqueantes y un esquema de mensajes de error claros, garantizando que las entradas del usuario sean válidas antes de continuar. Asimismo, se emplea un formato tabulado para mostrar los datos de manera ordenada, y se asegura la integridad del archivo CSV mediante verificaciones previas de lectura y escritura.

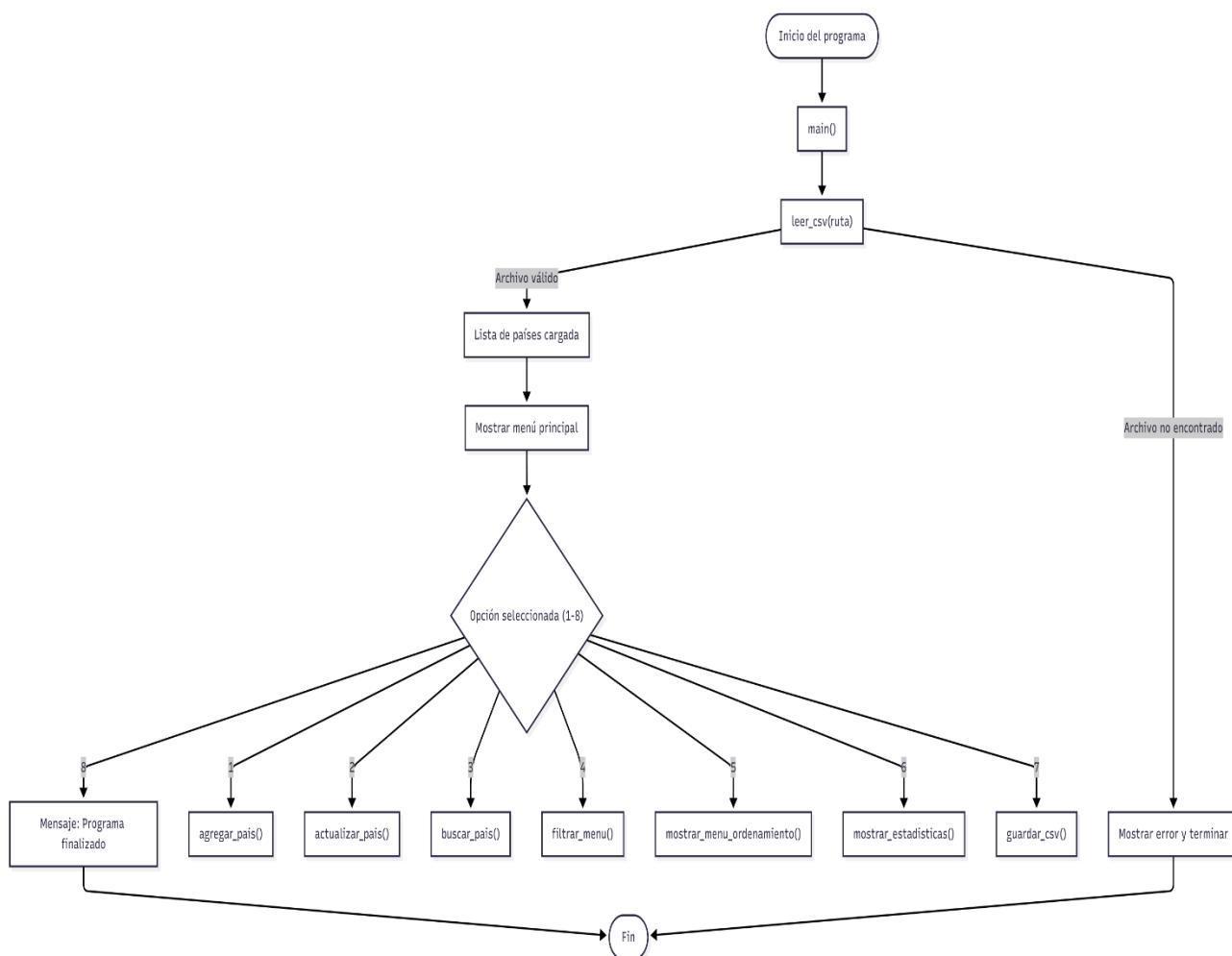
La **estructura modular** del código contribuye a una alta legibilidad y mantenimiento, cumpliendo con los principios de la programación estructurada promovidos por la cátedra.

Diagramas de Flujo de los Módulos Principales

A continuación, se presentan los diagramas de flujo correspondientes a los módulos principales del programa, los cuales ilustran el funcionamiento interno y la lógica estructurada de cada componente.

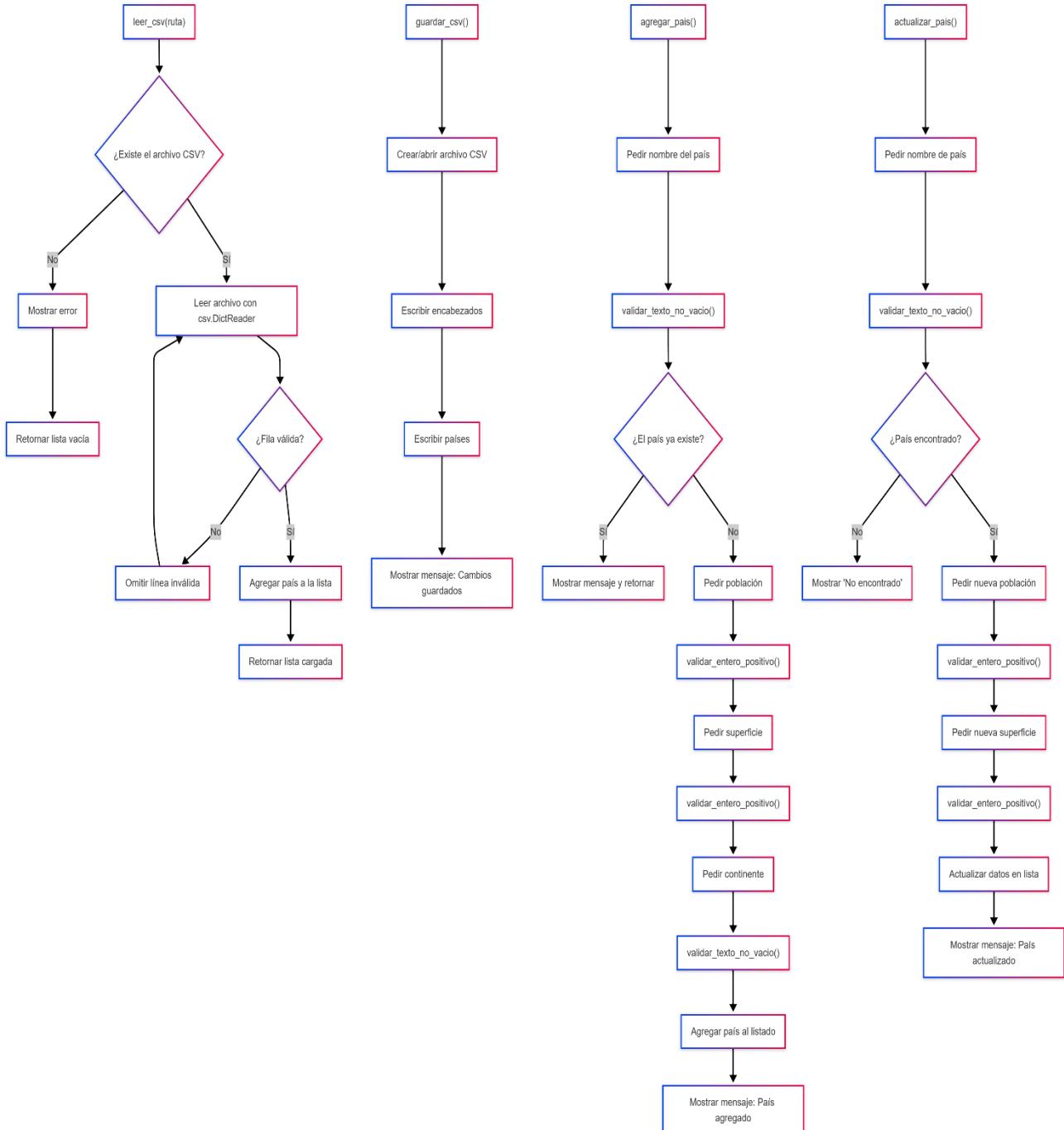
1. Módulo main.py — Flujo Principal del Programa:

Este diagrama representa el flujo general de ejecución del programa, desde la carga del archivo CSV hasta la interacción del usuario con el menú principal y las funciones asociadas.



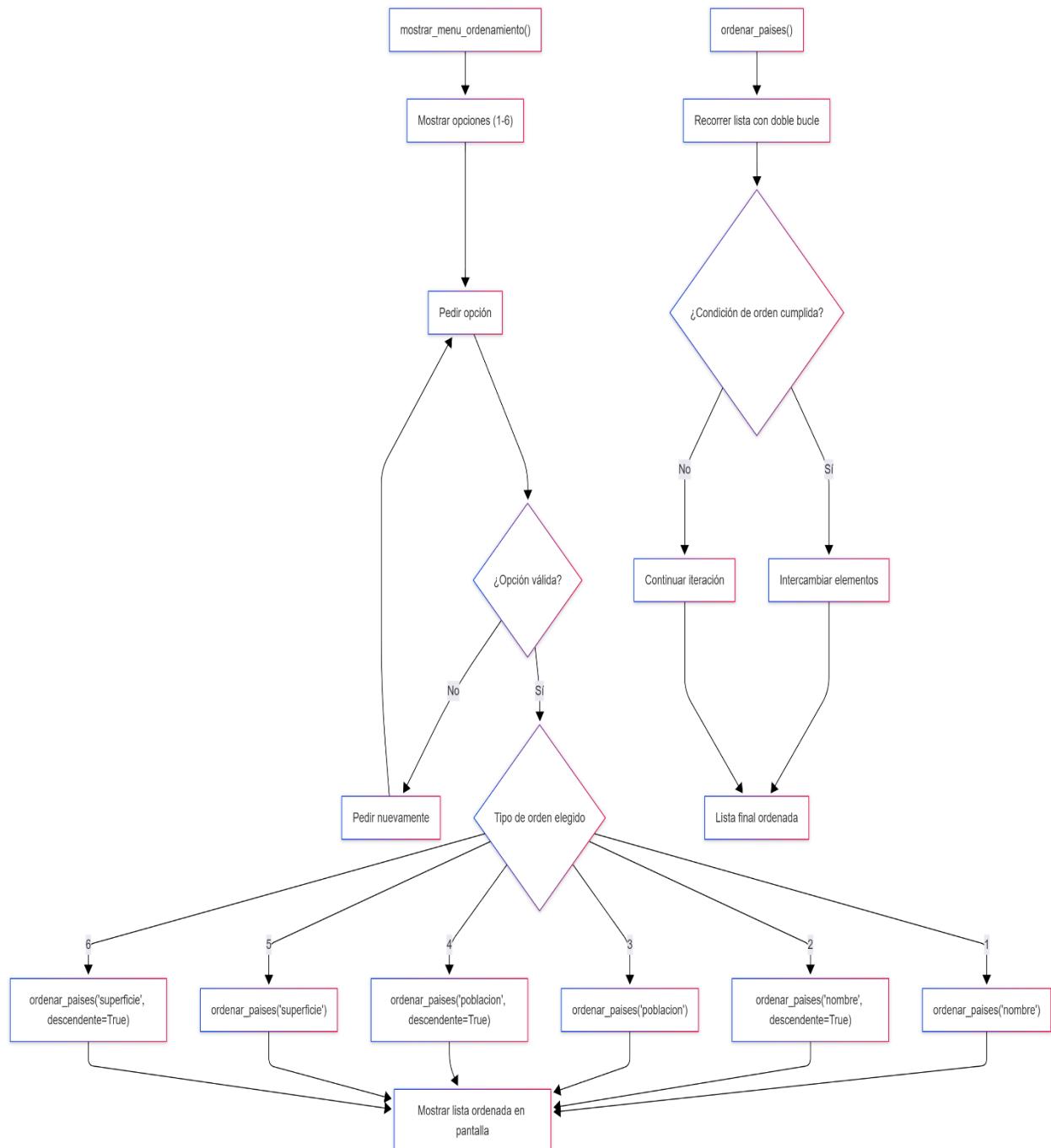
2. Módulo datos.py — Gestión de Datos:

Este diagrama muestra las operaciones de lectura, escritura y actualización de los datos del archivo CSV, así como las validaciones aplicadas en las funciones de carga y modificación.



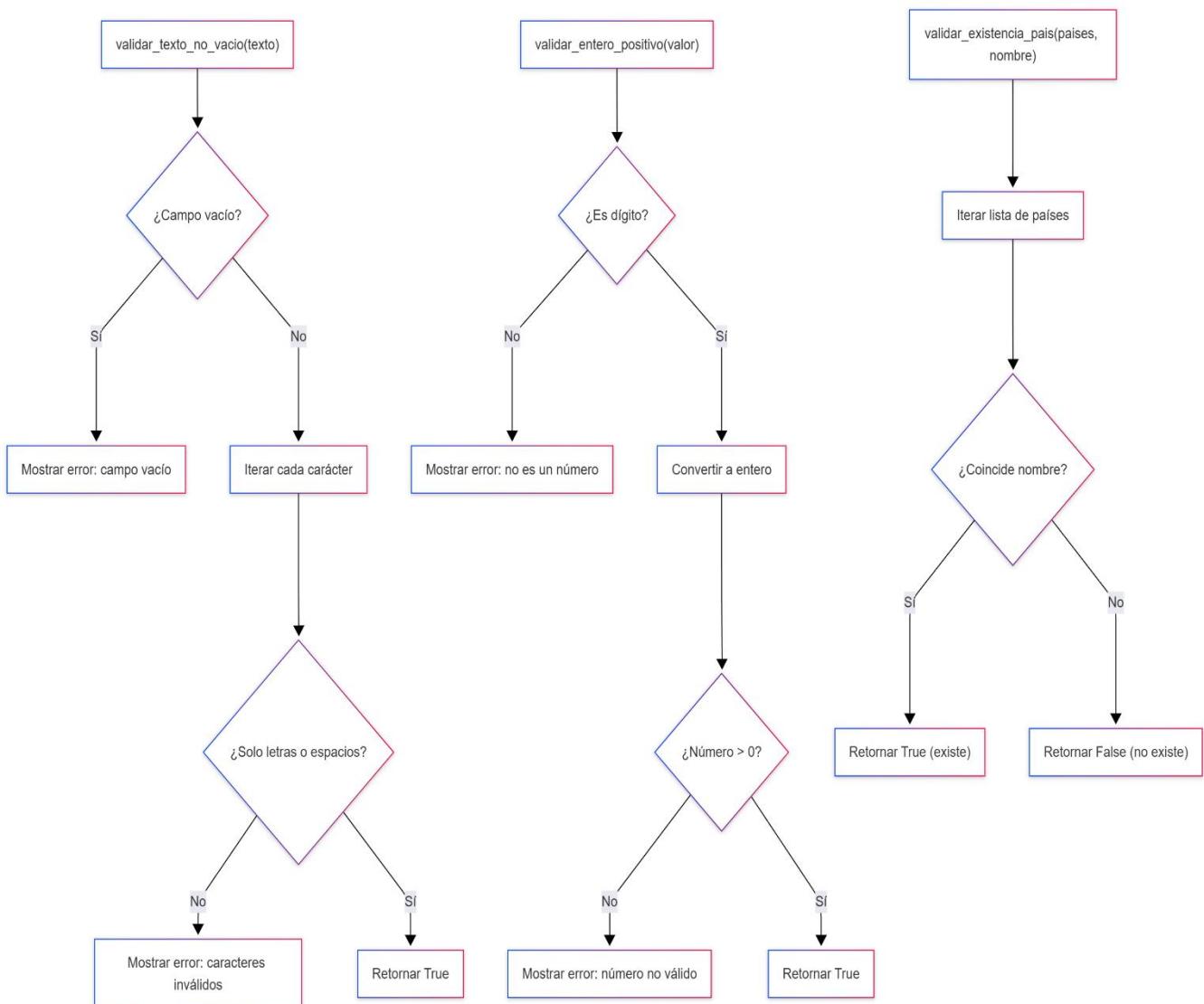
3. Módulo ordenamiento.py — Ordenamiento y Submenú:

El siguiente diagrama detalla la lógica del submenú de ordenamiento y el algoritmo manual implementado para organizar la lista de países según diferentes criterios.



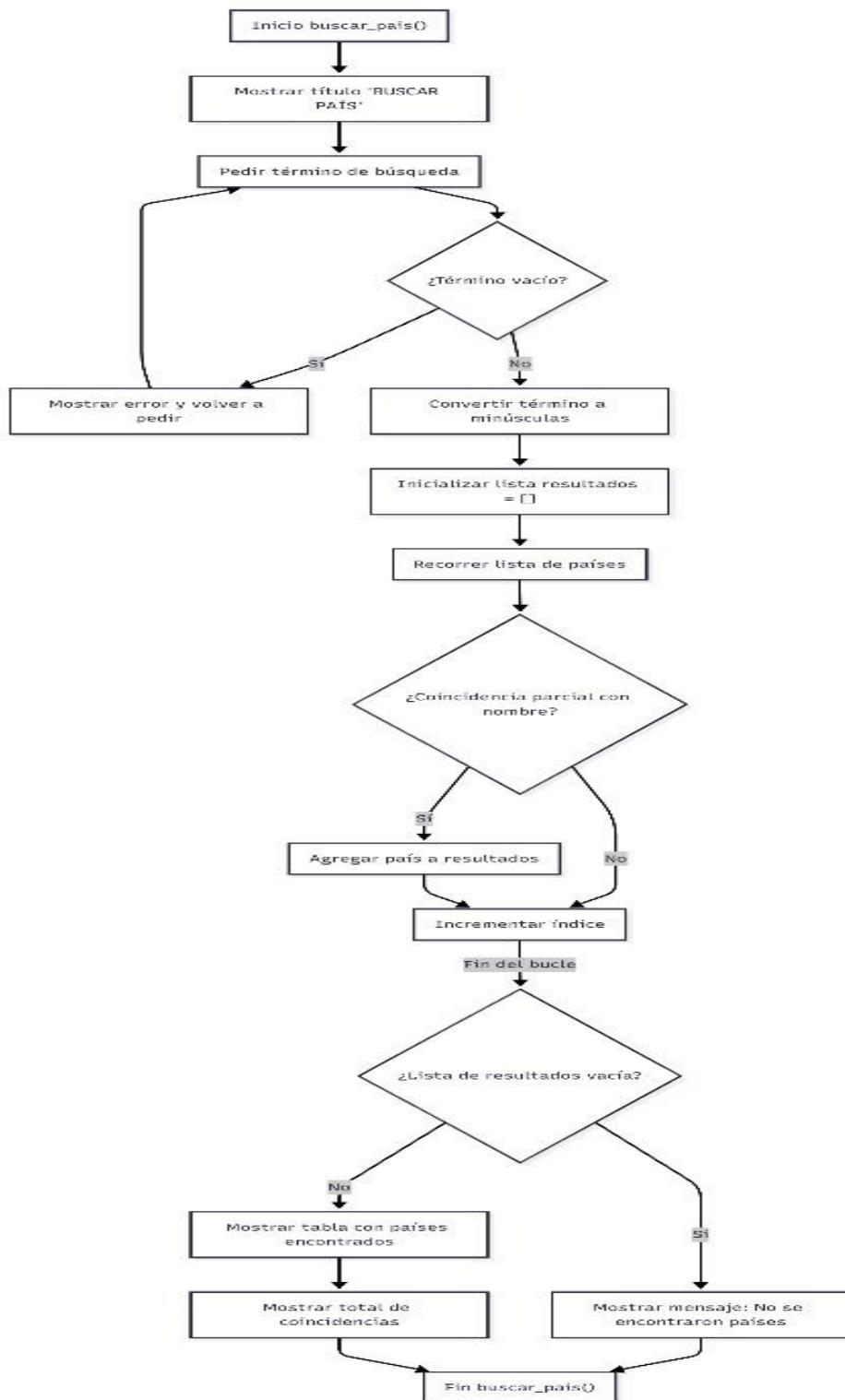
4. Módulo validaciones.py — Control de Entradas:

Este módulo garantiza la validez de los datos antes de su procesamiento. Implementa verificaciones manuales para prevenir errores de tipo y formato sin utilizar excepciones (try/except).



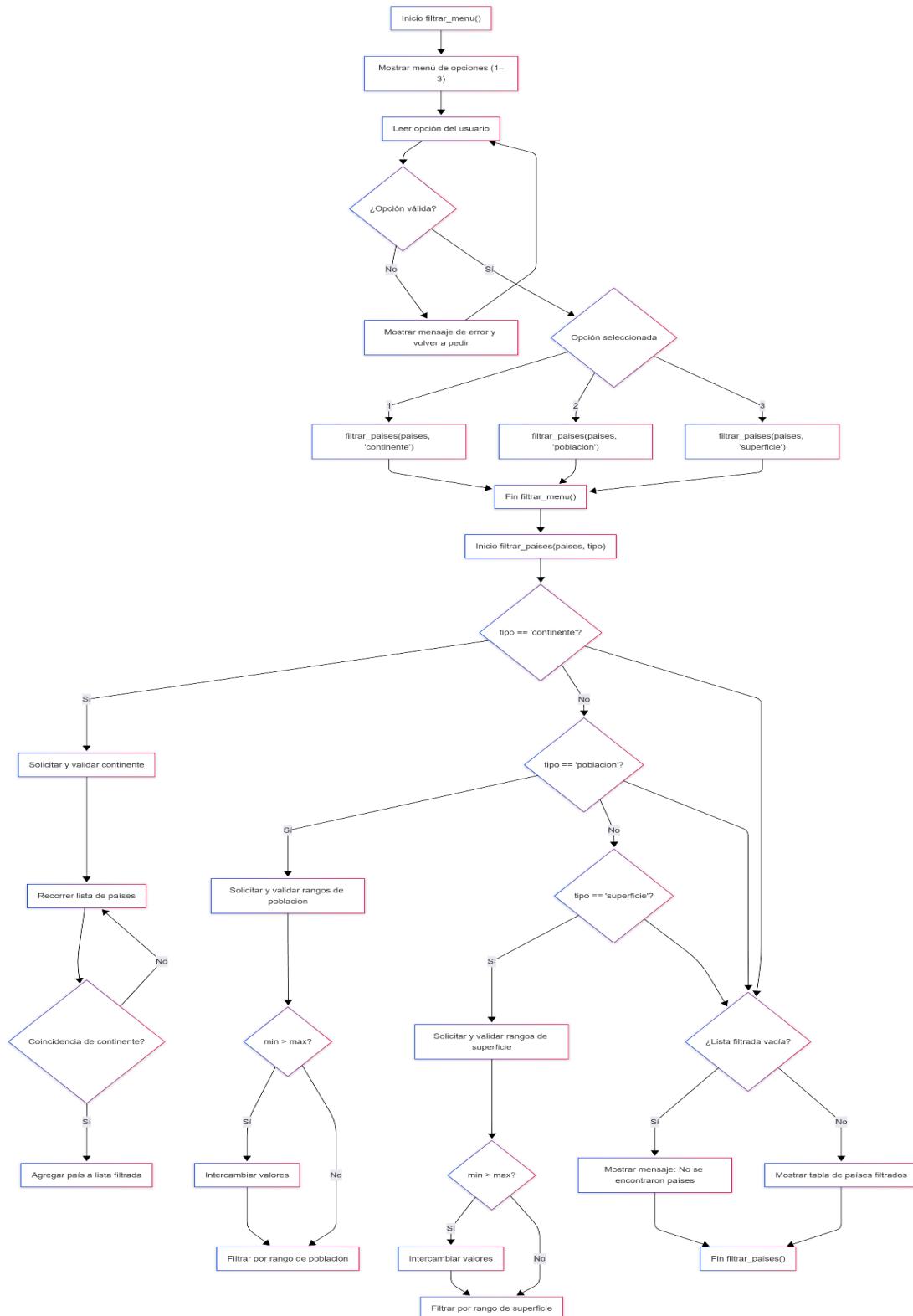
5. Modulo búsquedas.py — Búsqueda de Países:

Este módulo permite localizar uno o varios países dentro del listado general, utilizando coincidencias exactas o parciales según el texto ingresado por el usuario.



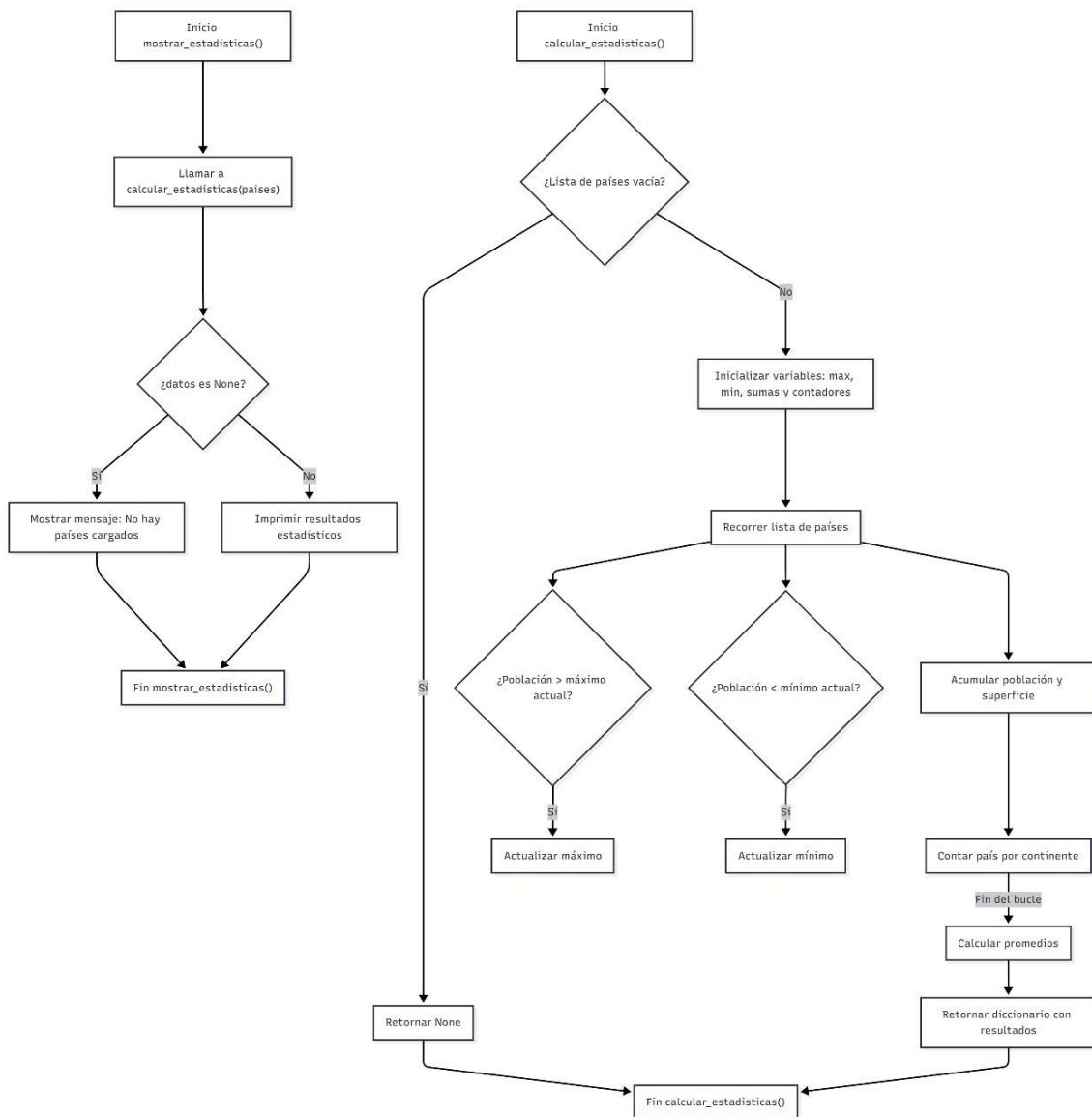
6. Módulo filtros.py — Filtrado de Países:

El módulo de filtros brinda al usuario la posibilidad de aplicar distintos criterios de selección sobre la lista de países. Presenta un submenú con tres opciones principales: filtrar por continente, por rango de población o por rango de superficie.



7. Módulo estadisticas.py — Cálculo de Estadísticas:

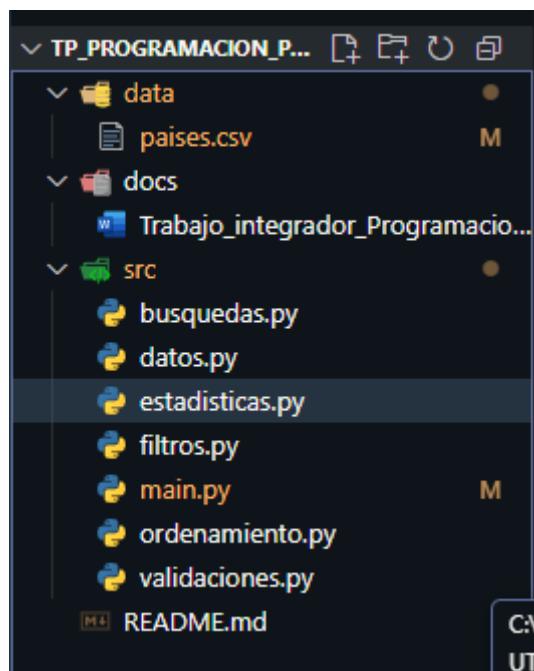
El módulo de estadísticas se encarga de analizar los datos de los países y generar indicadores generales sobre el conjunto. Implementa dos funciones principales: `calcular_estadisticas(paises)` y `mostrar_estadisticas(paises)`, que trabajan en conjunto.



8. Metodología de Desarrollo

El desarrollo se organizó en tres fases principales: **planificación, implementación y validación**.

- Durante la planificación, se definió la **estructura modular del proyecto**, distribuyendo las carpetas principales (src/, data/, docs/) y asignando funciones a cada integrante.



- En la fase de implementación, se aplicaron **bucles bloqueantes** para validar entradas erróneas, evitando retornos prematuros de funciones y garantizando un flujo lógico controlado.
- Se utilizaron **validaciones manuales** en lugar de excepciones automáticas, asegurando la correcta captura de datos por parte del usuario.
- Cada módulo fue **testeado individualmente** con casos de prueba en consola antes de la integración general.

- La comunicación y colaboración se realizaron mediante **GitHub** y **mensajería colaborativa**, aplicando revisiones cruzadas de código (*peer review*).

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks or learn more about diff comparisons.

base: rama-casto compare: main Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#) [Create pull request](#)

-o 14 commits 15 files changed 3 contributors

Commits on Nov 3, 2025

- Merge pull request #3 from CastoGil/rama-casto ...
CastoGil authored last week

Commits on Nov 4, 2025

- Merge pull request #4 from CastoGil/rama-casto ...
CastoGil authored last week
- Merge pull request #5 from CastoGil/rama-casto ...
CastoGil authored last week
- Merge pull request #6 from CastoGil/rama-casto ...
CastoGil authored last week

Commits on Nov 5, 2025

Rama alejo #10

Merged CastoGil merged 3 commits into main from rama-alejo 13 hours ago

Conversation 0 Commits 3 Checks 0 Files changed 10 +183 -8

Commits on Nov 8, 2025

- Se completo funcionalidad busqueda y filtros.
alealmada committed 2 days ago
- Se creo la funcionalidad de estadisticas
alealmada committed yesterday
- Configuraciones en README
alealmada committed yesterday

Rama alejo #10

Merged CastoGil merged 3 commits into main from rama-alejo 13 hours ago

Conversation 0 Commits 3 Checks 0 Files changed 10

CastoGil commented 13 hours ago

No description provided.

alealmada added 3 commits 2 days ago

- Se completo funcionalidad busqueda y filtros.
680c269
- Se creo la funcionalidad de estadisticas
8b53aa6
- Configuraciones en README
7167d82

CastoGil merged commit d161697 into main 13 hours ago Revert

- Se realizaron **commits frecuentes y descriptivos** en las ramas individuales, que posteriormente se integraron al branch principal main tras verificación mutua.

Branch	Updated	Check status	Behind Ahead	Pull request
main	10 hours ago		Default	...
rama-casto	13 hours ago		14 1	...
rama-alejo	yesterday		3 0	(+ 10)

- Finalmente, se elaboró el archivo **README.md** con las instrucciones de ejecución, y el **informe teórico** conforme a la rúbrica de la cátedra, garantizando documentación completa, coherente y profesional.

```

TP_Programacion_Paises/
├── data/
│   └── paises.csv
├── docs/
│   └── informe_teorico.pdf
└── src/
    ├── datos.py
    ├── validaciones.py
    ├── ordenamiento.py
    ├── estadisticas.py
    ├── busquedas.py
    ├── filtros.py
    └── main.py

```

README

Integrantes y Roles

Integrante	Comisión	Rama GitHub	Rol principal
Casto Gil	5	rama-gil	Gestión de datos, validaciones, ordenamientos y menú principal
Alejo Almada	1	rama-almada	Búsquedas, filtros y estadísticas

Distribución de Responsabilidades

Categoría de la rúbrica	Casto Gil (Com. 5)	Alejo Almada (Com. 1)
Gestión de datos (alta, modificación, guardado)	datos.py	—
Lectura CSV y validaciones	validaciones.py	Validaciones en busquedas.py / filtros.py
Ordenamientos manuales	ordenamiento.py	—
Estadísticas (promedios, min, max, conteo)	—	estadisticas.py
Búsquedas (exacta/parcial)	—	busquedas.py
Filtros (continente, población, superficie)	—	filtros.py
Menú e integración general	main.py (Casto)	estadisticas.py (consultas y reportes)
Persistencia (guardar CSV)	datos.py	—
Pruebas y presentación	Pruebas de integración y validación de flujo completo del sistema.	—
Trabajo en equipo / documentación	Coordinación Git / README	Ejemplos y documentación interna

README

⏱ Flujo del Programa (Menú Principal)

```
=====
GESTIÓN DE PAÍSES EN PYTHON
=====
1. Agregar país      → datos.py (Casto)
2. Actualizar país   → datos.py (Casto)
3. Buscar país       → busquedas.py (Alejo)
4. Filtrar países    → filtros.py (Alejo)
5. Ordenar países    → ordenamiento.py (Casto)
6. Mostrar estadísticas → estadisticas.py (Alejo)
7. Guardar cambios   → datos.py (Casto)
8. Salir             → main.py (Casto)

-----
Seleccione una opción:
```

⚙ Ejecución del Programa

Desde la carpeta `src`:

```
python main.py
```

📊 Ejemplo de Salida

- Ejemplo de resultados obtenidos al ejecutar las funciones de estadísticas:

```
🌐 País con mayor población: China (1,411,778,724)
🚩 País con menor población: Uruguay (3,423,108)
📊 Promedio de población: 394,357,149
📐 Promedio de superficie: 4,632,298 km²

Cantidad de países por continente:
- América: 5
```

```
 README
 PROMedio de superficie: 4,632,298 KM²

 Cantidad de países por continente:
 - América: 5
 - Europa: 3
 - Asia: 2
```

💡 Metodología de Trabajo

- **Control de versiones:** Flujo colaborativo con **Git** y **GitHub**, mediante ramas paralelas y Pull Requests.
- **Ramas de desarrollo:** `rama-casto` y `rama-almada`.
- **Validaciones bloqueantes:**
 - Cada ingreso de dato se controla con bucles while que solicitan nuevamente hasta ser válido.
 - No se utilizan estructuras avanzadas como try/except.
- **Estructuras de control:**
 - Uso de while con contadores manuales en lugar de for.
 - Validación de opciones incorrectas del menú.
- **Estructura modular:** Funciones pequeñas, reutilizables y documentadas.
- **Validación de CSV:** Se omiten líneas con errores mostrando mensajes claros.
- **Pruebas unitarias:** Cada módulo probado antes de fusionarse a `main`.
- No se utilizaron librerías externas; únicamente módulos nativos de Python.

📁 Repositorio y Documentación

- 📁 **Repositorio GitHub:**
 https://github.com/CastoGil/Tp_Programacion_Paises
- 📘 **Informe teórico:**
 /docs/informe_teorico.pdf
- 🌐 **Datos:**
 </data/paises.csv>

```
 README
 📁 Repositorio GitHub:
 ⚡ https://github.com/CastoGil/Tp\_Programacion\_Paises

 📖 Informe teórico:
 /docs/informe_teorico.pdf

 🌐 Datos:
 /data/paises.csv
```

🎥 Video de Exposición

Duración estimada: 10 – 15 minutos

Estructura sugerida:

1. 🎬 Introducción y objetivos — *ambos integrantes*
2. 📊 Gestión de datos, validaciones y ordenamientos — *Casto Gil*
3. 🔎 Búsquedas, filtros y estadísticas — *Alejo Almada*
4. 🎯 Conclusiones y reflexión final — *ambos integrantes*

🏫 Cátedra

Tecnicatura Universitaria en Programación — UTN

Materia: Programación 1

Tutores: Martín García y Matías Torres

Comisiones: 1 y 5 — 2º Cuatrimestre 2025

- Las pruebas funcionales se realizaron con **archivos CSV de prueba**, conteniendo registros válidos e inválidos, para verificar la respuesta de las

funciones de validación, la actualización de datos y el correcto comportamiento del menú principal.

```
PS C:\Users\Admin\OneDrive - TUPAD UTN\Escritorio\TP_Programacion\TP_Programacion_Paises\src> python main.py
- Se cargaron 13 países correctamente.
=====
GESTIÓN DE PAÍSES EN PYTHON
=====

1. Agregar país
2. Actualizar país
3. Buscar país
4. Filtrar países
5. Ordenar países
6. Mostrar estadísticas
7. Guardar cambios
8. Salir

-----
Seleccione una opción (1-8): [
```

6. Resultados Obtenidos

El programa desarrollado logró cumplir con todos los requerimientos del enunciado y los criterios de evaluación definidos.

Se alcanzaron los siguientes resultados:

- **Funcionalidad completa:** búsquedas, filtros, ordenamientos, estadísticas y persistencia de datos.
- **Robustez:** manejo de errores de entrada y validaciones consistentes.
- **Eficiencia:** uso de estructuras nativas de Python con bajo costo computacional.
- **Claridad y legibilidad:** código modular, comentado y de fácil mantenimiento.
- **Trabajo colaborativo efectivo:** integración exitosa mediante control de versiones y coordinación entre integrantes.

El principal desafío fue integrar los módulos de filtrado y ordenamiento sin usar funciones automáticas como sorted(), resolviéndolo con comparaciones manuales y ciclos anidados.

El programa fue probado con distintos escenarios y archivos CSV, confirmando su correcto funcionamiento ante datos válidos e inválidos.

Se incluyen capturas representativas del sistema en ejecución que demuestran el cumplimiento de las principales funcionalidades:

- Menú principal:

```
- Se cargaron 11 países correctamente.
```

```
=====
```

```
GESTIÓN DE PAÍSES EN PYTHON
```

```
=====
```

1. Agregar país
2. Actualizar país
3. Buscar país
4. Filtrar países
5. Ordenar países
6. Mostrar estadísticas
7. Guardar cambios
8. Salir

```
=====
```

```
Seleccione una opción (1-8): |
```

- Agregar país:

```
Seleccione una opción (1-8): 1
```

```
=====
```

```
AGREGAR PAÍS
```

```
=====
```

```
Ingrese nombre del país: Estados Unidos
```

```
Ingrese población: 70258462258
```

```
Ingrese superficie (km²): 98752200455
```

```
Ingrese continente: America
```

```
- Estados Unidos agregado correctamente.
```

- Buscar país:

```
Seleccione una opción (1-8): 3
=====
          BUSCAR PAÍS
=====
Ingrese el nombre o parte del país a buscar: Argentina

Resultados encontrados:

Nombre      | Población | Superficie (km²) | Continente
-----
Argentina   |    45376763 |        2780400 |     América
-----
Total de países encontrados: 1
```

```
=====
          BUSCAR PAÍS
=====
Ingrese el nombre o parte del país a buscar: Ven

Resultados encontrados:

Nombre      | Población | Superficie (km²) | Continente
-----
Venezuela   |      70000 |           44 |     America
-----
Total de países encontrados: 1
```

- Filtrar países:

```
=====
          FILTRAR PAÍSES
=====

1. Por continente
2. Por rango de población
3. Por rango de superficie

Seleccione una opción (1-3): 1
Elija por cuál continente quiere filtrar (América, Europa, Asia, Oceanía o África): America

Nombre      | Población | Superficie (km²) | Continente
-----
Canadá      |    38005238 |        9984670 |     América
Argentina   |    45376763 |        2780400 |     América
Venezuela   |      70000 |           44 |     America
Brasil      |  213993437 |        8515767 |     América
Estados Unidos | 70258462258 |  98752200455 |     America
Chile       |        47 |        28464523546 |     America
-----
Total de países encontrados: 6
```

- Estadísticas generales:

```
=====
ESTADÍSTICAS GENERALES
=====
País con mayor población: Estados Unidos (70258462258)
País con menor población: Venezuela (70000)
Promedio de población: 6154014822.58
Promedio de superficie: 8233074376.58 km2

Cantidad de países por continente:
- América: 5
- Europa: 2
- Asia: 2
- Oceanía: 1
- África: 2
```

- Fin del programa – Guardar datos:

```
=====
GESTIÓN DE PAÍSES EN PYTHON
=====

1. Agregar país
2. Actualizar país
3. Buscar país
4. Filtrar países
5. Ordenar países
6. Mostrar estadísticas
7. Guardar cambios
8. Salir

-----
Seleccione una opción (1-8): 8

¿Desea guardar los cambios antes de salir? (s/n)
→ No

Gracias por utilizar el sistema de gestión de países.
```

Estas evidencias son suficientes para verificar el correcto funcionamiento y la interacción del usuario con el sistema, sin necesidad de mostrar la totalidad de los procesos.

7. Conclusiones Grupales

Este trabajo permitió consolidar los conceptos teóricos de la materia Programación 1 a través de su aplicación práctica en un proyecto real, integrando estructuras de datos, validaciones y modularización en Python.

Entre los principales aprendizajes se destacan:

- **La modularización** y la separación de responsabilidades como base para un código ordenado, legible y escalable.
- **Las validaciones bloqueantes** y el control de errores, fundamentales para garantizar la integridad de los datos y la estabilidad del sistema.
- **El manejo de archivos CSV** como mecanismo de persistencia simple y eficiente para proyectos de administración de datos.
- **El uso de GitHub** como herramienta de colaboración, control de versiones y organización del flujo de trabajo entre los integrantes.
- **La experiencia de trabajo en equipo**, que permitió coordinar tareas, resolver conflictos y alcanzar un resultado profesional y funcional.

En conclusión, el desarrollo de este proyecto integrador nos permitió **fortalecer nuestras habilidades técnicas y de colaboración**, aplicando los conocimientos adquiridos a un caso concreto que refleja los principios de la programación estructurada y el trabajo cooperativo en entornos reales.

8. Fuentes Bibliográficas, Repositorio y Recursos Digitales

- Matthes, Eric. *Python Crash Course*. No Starch Press, 2019.
- Documentación oficial de Python: <https://docs.python.org/es/3/>
- Apuntes de Cátedra — UTN Programación 1 (2025).
- Material del Campus Virtual UTN.
- GitHub Docs — *Collaborative workflows and branching strategies*, 2024.
- **Repositorio del Proyecto:**

 https://github.com/CastoGil/TP_Programacion_Paises

El repositorio incluye el código fuente completo del sistema, el archivo CSV de datos, el informe teórico, las capturas de pantalla, y el archivo README con las instrucciones de uso y documentación técnica.

- **Video de Presentación (TPI – Programación 1, UTN 2025):**

 https://www.youtube.com/watch?v=yu6_il5qRdo

El video explicativo tiene una duración total de **13–15 minutos**, y presenta a ambos integrantes del equipo.

En él se detallan el diseño del caso práctico, el funcionamiento del programa en Visual Studio Code, las validaciones implementadas, y una reflexión final sobre la experiencia del trabajo colaborativo.