Trabajo Práctico Integrador

Programación

Alumnos

Máximo Quiroga y Santino barone

Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional.

Docente Titular

Cinthia Rigoni

Ramiro Hualpa

2025

ÍNDICE

Introducción.....**3**
Objetivos.....**3**
Desarrollo.....**4**
Puntos clave del código.....4
Decisiones Clave de Diseño.....6
Conceptos Aplicados.....8
Conclusión.....**10**
Referencias.....**10**

Introducción

El presente informe detalla la estructura y funcionamiento de un sistema de software desarrollado en Python. El programa consiste en una aplicación de consola (CLI) diseñada para gestionar información geográfica y demográfica de países.

El sistema permite a un usuario interactuar con un conjunto de datos almacenado en un archivo CSV. Las funcionalidades principales incluyen la capacidad de buscar, filtrar, ordenar y realizar operaciones de alta, baja y modificación (ABM) sobre los datos de los países, así como visualizar estadísticas relevantes.

La arquitectura del proyecto está dividida en tres módulos principales: un archivo de arranque (`main.py`), un módulo para la lógica de negocio (`funciones.py`) y un módulo de herramientas auxiliares (`utilidades.py`), lo que facilita su mantenimiento y escalabilidad.

Objetivos

- **Analizar la estructura de un programa modular en Python.**
- **Identificar las responsabilidades de cada componente del software.**
- **Comprender las decisiones de diseño que sustentan la funcionalidad del sistema.**
- **Evaluar la interacción entre la lógica del programa, la gestión de datos y la interfaz de usuario.**
- **Reconocer los conceptos fundamentales de programación aplicados en una solución práctica.**

Desarrollo

Puntos clave del código

El código se organiza en tres archivos distintos, cada uno con una responsabilidad bien definida.

1. Archivo `main.py` (Controlador Principal) Este archivo es el punto de entrada de la aplicación.

- **Orquestación: No contiene lógica de negocio. Su única función es mostrar el menú principal al usuario y, según la opción seleccionada, invocar a las funciones correspondientes de los otros módulos.**
- **Bucle Principal: Utiliza un bucle `while True` para mantener la aplicación en ejecución hasta que el usuario decida salir.**
- **Manejo de Menú: Emplea una estructura `match-case` para gestionar las opciones del menú, lo cual es una sintaxis moderna y legible para manejar flujos de control.**
- **Carga Inicial de Datos: Al iniciar, invoca a `utilidades.cargar_archivo_csv()` para leer los datos del archivo y cargarlos en la memoria.**

Python

```
# main.py: Carga inicial y bucle principal del menú

if __name__=="__main__":

   lista_paises = utilidades.cargar_archivo_csv() # Carga los datos una sola vez

   while True:

      opcion=input("""...""") # Muestra el menú
```

```
match opcion: # Delega la acción a otros módulos
    case "1":
        funciones.BusquedaPais(lista_paises)
    case "2":
        funciones.manejar_submenu_filtros(lista_paises)
    # ... resto de las opciones
```

2. Archivo `utilidades.py` (Caja de Herramientas) Este módulo agrupa funciones genéricas y reutilizables que dan soporte al resto del programa.

- **Interacción con Archivos CSV: Contiene las funciones `cargar_archivo_csv()` y `actualizar_csv()`. Utiliza el módulo `csv` de Python, específicamente `DictReader` y `DictWriter`, para leer y escribir los datos de manera estructurada y robusta.**
- **Validación de Entradas: Provee funciones como `leer_entero()`, `pedir_string()` y `pedir_int()`, que aseguran que el usuario ingrese datos del tipo y formato correctos, evitando errores en tiempo de ejecución.**
- **Normalización de Texto: La función `quitar_tildes()` es crucial para que las búsquedas y comparaciones no distingan entre mayúsculas, minúsculas o acentos (ej: buscar "america" y encontrar "América").**
- **Presentación de Datos: La función `mostrar_lista_paises()` se encarga de formatear y mostrar los datos en una tabla prolija en la consola, centralizando la lógica de visualización.**

3. Archivo `funciones.py` (Lógica de Negocio) Este es el núcleo de la aplicación, donde reside la lógica para manipular los datos.

- **Operaciones ABM (CRUD): Contiene las funciones `crear_pais()`, `editar_pais()` y `eliminar_pais()`. Estas gestionan el flujo completo de la operación: solicitan los datos al usuario, validan la información (ej: que no se repitan países) y aplican los cambios a la lista en memoria.**
- **Búsqueda y Ordenamiento: La función `BusquedaPais()` implementa una búsqueda flexible por subcadena. La función `Ordenar()` reorganiza la lista de datos según el criterio seleccionado (nombre, población, etc.) y la reescribe en el archivo CSV.**
- **Filtrado y Estadísticas: Proporciona menús y funciones dedicadas para `manejar_submenu_filtros()` y `menu_estadisticas()`, separando estas funcionalidades complejas del menú principal.**

Decisiones Clave de Diseño

1. **Arquitectura Modular: La decisión más importante fue separar el código en tres archivos.**
    - **Beneficio: Esta división sigue el principio de separación de responsabilidades. `main.py` se ocupa del "qué" hacer (la navegación), `funciones.py` del "cómo" se hace (la lógica) y `utilidades.py` de las "herramientas" para hacerlo. Esto hace que el código sea más fácil de leer, depurar y extender.**
2. **Manejo de Datos en Memoria y Persistencia:**
    - **Decisión: El sistema carga todo el archivo CSV en una lista de diccionarios al arrancar. Cada operación (crear, editar, ordenar) modifica esta lista en memoria y reescribe el archivo CSV completo inmediatamente a través de `actualizar_csv()`.**

      Análisis:

- **Ventaja: Es un enfoque simple y seguro. Garantiza que el archivo CSV siempre esté sincronizado con el estado actual del programa, evitando la pérdida de datos si el programa se cierra inesperadamente.**
- **Desventaja: Para archivos de datos muy grandes (cientos de miles de registros), reescribir el archivo entero en cada cambio sería ineficiente. Sin embargo, para la escala de este proyecto, es una solución perfectamente adecuada.**

3. **Interfaz de Usuario y Experiencia :**
   - **Decisión: Se implementó una interfaz de línea de comandos basada en menús anidados. Para las búsquedas y filtros, se normalizan las entradas del usuario (se quitan tildes y se convierte a minúsculas).**
   - **Beneficio: Aunque es una interfaz sencilla, las validaciones de entrada (`pedir_int`, `pedir_string`) y la normalización del texto la hacen robusta y amigable, ya que previene errores y flexibiliza las búsquedas.**
4. **Representación de Datos:**
   - **Decisión: Se eligió una lista de diccionarios (ej: `[{'nombre': 'Argentina', 'poblacion': 47000000, ...}]`) para representar los datos en memoria.**
   - **Beneficio: Esta estructura es muy legible, ya que se accede a los datos por claves con nombre (`pais['poblacion']`) en lugar de por índices numéricos (`pais[1]`). Esto reduce errores y hace el código más auto-documentado. Además, se integra de forma natural con el módulo `csv.DictReader`.**

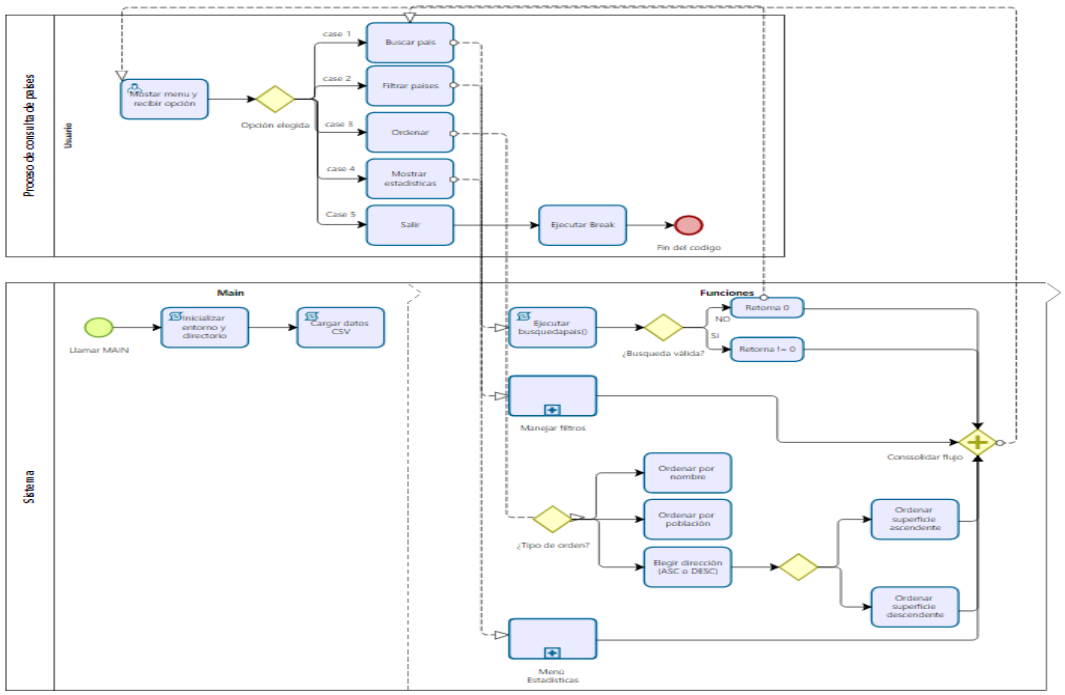
Conceptos Aplicados

El proyecto demuestra la aplicación práctica de varios conceptos fundamentales de la programación:

- **Listas: Es la estructura de datos principal (`lista_paises`). Se utiliza para almacenar el conjunto de países en memoria, permitiendo iterar sobre ellos, agregar nuevos elementos (`.append()`) y eliminarlos (`del`).**
- **Diccionarios: Cada país dentro de la `lista_paises` se representa como un diccionario. Esto permite almacenar datos de forma estructurada con pares clave-valor (ej: `{'nombre': ..., 'poblacion': ...}`), facilitando el acceso y la modificación de atributos específicos de cada país.**
- **Funciones: El código está completamente modularizado. Se definen funciones con responsabilidades únicas (ej: `crear_pais`, `leer_entero`, `calcular_promedios`), lo que promueve la reutilización de código y facilita la lectura.**
- **Condicionales: Se utilizan extensivamente (con `if`, `elif`, `else` y `match-case`) para controlar el flujo del programa, validar entradas del usuario (ej: `if n<0:`) y tomar decisiones lógicas (ej: `if busqueda in i["nombre"]:`).**
- **Ordenamientos: Se implementa una funcionalidad de ordenamiento explícita en `funciones.Ordenar()`, que utiliza la función nativa de Python `sorted()` junto con una función `lambda` como clave (`key=lambda x: x[parte]`) para ordenar la lista de países según diferentes criterios.**
- **Estadísticas básicas: El módulo de estadísticas (`menu_estadisticas`) aplica lógica para el análisis de datos, utilizando funciones como `max()`, `min()` y `sum()` sobre la lista de países para calcular valores relevantes (mayor/menor población, promedios).**

- **Archivos CSV: El programa utiliza el módulo `csv` para gestionar la persistencia de datos. `csv.DictReader` (`cargar_archivo_csv`) se usa para leer el archivo y cargarlo en la lista de diccionarios, y `csv.DictWriter` (`actualizar_csv`) para guardar los cambios de vuelta en el archivo.**
- **Funciones CRUD: El núcleo del menú de edición (opción 5) implementa las cuatro operaciones básicas de gestión de datos (CRUD: Create, Read, Update, Delete), que en el código se corresponden con las funciones:**
    - **Create (Crear): `funciones.crear_pais()`**
    - **Read (Leer): `funciones.BusquedaPais()` y `utilidades.mostrar_lista_paises()`**
    - **Update (Actualizar): `funciones.editar_pais()`**
    - **Delete (Borrar): `funciones.eliminar_pais()`**

DIAGRAMA DE FLUJO (Adjunto en el repositorio)



Conclusión

El desarrollo de este sistema de gestión ha servido como una demostración práctica y exitosa de los conceptos clave de la programación. El proyecto no solo cumple con todas sus metas funcionales, sino que también valida la importancia de una buena arquitectura de software.

La modularidad del código, separando responsabilidades en distintos archivos, fue la decisión de diseño más impactante, facilitando la implementación de lógicas complejas como el patrón CRUD (Crear, Leer, Actualizar, Borrar) y el módulo de estadísticas.

Se concluye que el uso correcto de estructuras de datos (listas y diccionarios) y una gestión de archivos resiliente son pilares fundamentales para crear software fiable. El proyecto final es una base de código estable.

Referencias

- **Lutz, M. (2013). *Aprendiendo Python (5ª ed.)*. O'Reilly Media.**
- **Sweigart, A. (2020). *Automatiza tareas aburridas con Python*. No Starch Press.**
- **UTN (2025). *Apuntes de Programación I Campus*. Universidad Tecnológica Nacional.**
- **Pressman, R. S., & Maxim, B. R. (2020). *Software Engineering: A Practitioner's Approach (9th ed.)*. McGraw-Hill.**
- **Downey, A. (2015). *Think Python: How to Think Like a Computer Scientist*. Green Tea Press.**