UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

CENTRO UNIVERSITARIO DE OCCIDENTE

DIVISIÓN CIENCIAS DE LA INGENIERIA

ESTRUCTURA DE DATOS

BIBLIOTECA MAGICA

BRANDON GUSTAVO GÜINAC ROMÁN

201931217

FECHA DE ENTREGA: 02/11/2025

1. Introducción

El presente documento describe el funcionamiento interno, diseño arquitectónico, componentes técnicos y dependencias del sistema **Biblioteca Mágica Alrededor del Mundo**, desarrollado en **Python** como parte del curso **Laboratorio de Estructura de Datos**.

El sistema gestiona una red de bibliotecas interconectadas que permiten el registro, búsqueda, organización y transferencia de libros utilizando **estructuras de datos implementadas desde cero**. Las estructuras son visualizadas dinámicamente mediante **Graphviz**, y el sistema integra una **interfaz gráfica interactiva** que facilita la administración y simulación de operaciones.

2. Arquitectura General del Sistema

El sistema está diseñado bajo una **arquitectura modular**, organizada en los siguientes componentes principales:

Módulo	**Descripción**	**Ejemplo de Clases**
Interfaz Gráfica (GUI)	Gestiona la interacción con el usuario. Permite agregar libros, ejecutar búsquedas, iniciar simulaciones y mostrar gráficos.	VentanaPrincipal, PanelBibliotecas, PanelLibros
Controladores	Enlazan la lógica del programa con las estructuras de datos. Administran operaciones complejas y coordinan los módulos.	ControladorBiblioteca, ControladorLibro
Estructuras de Datos	Contienen las implementaciones de listas, árboles, tablas, colas, pilas y grafos.	ArbolAVL, ArbolB, TablaHash, Grafo
(Graphviz)	Genera archivos .dot y renderiza imágenes de las estructuras.	ReporteManager, graficar()
Módulo de Archivos	Maneja la carga y validación de archivos CSV (libros, bibliotecas, conexiones).	CargadorCSV, GestorArchivos
Reportes	Crea estadísticas, comparativas y reportes visuales de rendimiento.	ReporteTiempos, ReporteEstructuras

3. Tecnologías Utilizadas

Tecnología	Descripción	Uso en el sistema
Python 3.12	Lenguaje principal del proyecto.	Implementación de estructuras y GUI.
Graphviz	Motor de renderizado gráfico.	Visualización de árboles, tablas y grafos.
Tkinter / ttkbootstrap	Framework de interfaz gráfica.	Menús, paneles y botones.
CSV /	Carga de archivos de datos.	Lectura del catálogo y conexiones.
VS Code / Terminal	Entorno de desarrollo.	Ejecución y depuración.

4. Estructura del Proyecto

BibliotecaMagica/

│

├── interfaz/

│ ├── VentanaPrincipal.py

│ ├── PanelBibliotecas.py

│ ├── PanelLibros.py

│

├── controladores/

│ ├── ControladorBiblioteca.py

│ ├── ControladorLibro.py

│

├── estructuras/

│ ├── ListaEnlazada.py

│ ├── Pila.py

│ ├── Cola.py

│ ├── ArbolAVL.py

│ ├── ArbolB.py

│ ├── ArbolBMas.py

│ ├── TablaHash.py

│ ── Grafo.py

│

── reportes/

│ ── ReporteManager.py

│ ── ReporteTiempos.py

│

|── datos/

│ ── libros.csv

│ ── bibliotecas.csv

│ ── conexiones.csv

│

| -- main.py

5. Lógica de Funcionamiento

1. **Inicio del sistema:**
   El usuario ejecuta main.py, lo que inicializa la interfaz gráfica (VentanaPrincipal).
2. **Carga de datos:**
   Los archivos CSV son leídos por el módulo de archivos, validando formato y registros.
3. **Inicialización de estructuras:**
   Los libros son insertados en las estructuras principales:
   - Árbol AVL (por título)
   - Árbol B (por año)
   - Árbol B+ (por género)
   - Tabla Hash (por ISBN)
   - Listas enlazadas (por colección)

4. **Gestión de bibliotecas:**
   Cada biblioteca se registra como un nodo dentro del grafo ponderado, con sus tiempos de ingreso, traspaso y despacho.
5. **Transferencia de libros:**
   Cuando un libro se envía de una biblioteca a otra:
   - Se calcula la ruta óptima usando Dijkstra.
   - Se simula el tránsito en colas FIFO.
   - Se actualizan estados (“en tránsito”, “disponible”, “agotado”).
6. **Visualización gráfica:**
   Las estructuras se exportan a .dot mediante ReporteManager y se renderizan con **Graphviz** en formato .png.
7. **Comparación de rendimiento:**
   El sistema mide los tiempos de búsqueda y ordenamiento, registrando estadísticas en reportes.

6. Dependencias y Librerías

Para ejecutar el proyecto correctamente, deben instalarse las siguientes dependencias:

pip install graphviz

pip install ttkbootstrap

pip install pandas

Además, debe estar instalado **Graphviz** en el sistema operativo.
Verificar con el comando:

dot -V

7. Compilación y Ejecución

Ejecución directa desde consola:

python main.py

Alternativa (Windows):
Hacer doble clic sobre main.py o usar el acceso directo creado en la interfaz.

Recomendaciones:

- Mantener los archivos CSV en la carpeta /datos.
- Verificar permisos de escritura en /reportes.
- Usar Python 3.10 o superior.

8. Manejo de Archivos y Rutas

El sistema carga tres archivos base:

Archivo	**Propósito**
libros.csv	Catálogo general de libros.
bibliotecas.csv	Lista de bibliotecas registradas.
conexiones.csv	Rutas ponderadas entre bibliotecas.

Durante la ejecución, se generan:

- Archivos .dot en /reportes/estructuras/
- Imágenes .png en /reportes/graficos/
- Reportes de texto en /reportes/estadisticas/

9. Métodos Principales

Clase	Método	Descripción
ArbolAVL	insertar(), eliminar(), buscar()	Gestiona libros por título.
TablaHash	insertar(), buscar(), eliminar()	Indexa libros por ISBN.
Grafo	ruta_optima()	Calcula ruta entre bibliotecas.
Cola	enqueue(), dequeue()	Controla despacho de libros.
ReporteManager	generar_dot(), renderizar()	Exporta estructuras con Graphviz.
VentanaPrincipal	mostrar_estructura()	Visualiza resultados gráficos.

10. Consideraciones de Diseño

- Todos los TADs fueron implementados **sin usar librerías de estructuras estándar** (como dict, set o heapq).
- Se priorizó la **modularidad y la reutilización de código.**
- Cada estructura cuenta con su propio método graficar() para mantener independencia.
- Los algoritmos de recorrido (inorden, preorden, BFS, DFS) se implementaron manualmente.

11. Análisis de Complejidad (Big O)

Estructura	Inserción	Eliminación	Búsqueda
Lista Enlazada	O(1)	O(n)	O(n)
Pila / Cola	O(1)	O(1)	O(1)
Árbol AVL	O(log n)	O(log n)	O(log n)
Árbol B / B+	O(log n)	O(log n)	O(log n)
Tabla Hash	O(1)*	O(1)*	O(1)*
Grafo	O(V + E)	O(V + E)	$O(V^2)$ (rutas)

En promedio, dependiendo del factor de carga.

12. Errores Comunes y Soluciones

Error	Causa	Solución
“graphviz.backend.ExecutableNotFound”	Graphviz no instalado o no agregado al PATH.	Reinstalar Graphviz y agregarlo a variables de entorno.
“Archivo no encontrado”	CSV en ubicación incorrecta.	Verificar rutas en /datos/.
“Error de tipo”	Dato mal formateado en CSV.	Revisar estructura del archivo o agregar validación.
“Colisión en Tabla Hash”	Demasiados registros por índice.	Aumentar tamaño inicial de la tabla.

13. Seguridad y Validaciones

- Validación de ISBN antes de inserción.
- Detección de duplicados en bibliotecas.
- Manejo de errores al cargar archivos.
- Control de rangos válidos de años.
- Bloqueo de inserciones vacías.

14. Conclusión

El **Manual Técnico** documenta la implementación completa del sistema, detallando su estructura modular, dependencias, diseño y funcionamiento interno.
El uso de **Python** junto a **Graphviz** permitió desarrollar una solución potente, intuitiva y visualmente atractiva, capaz de representar el comportamiento dinámico de múltiples estructuras de datos avanzadas de manera eficiente y didáctica.