MANUAL TECNICO

Madelayne Ana MAria Perez Perez

202130171

MANUAL TÉCNICO DEL SISTEMA DE BIBLIOTECA

---

1. Introducción Técnica

El Sistema de Biblioteca fue desarrollado en C++20 utilizando estructuras de datos avanzadas para optimizar la gestión, búsqueda y eliminación de libros.
El sistema emplea tres estructuras principales:

Árbol AVL → organiza libros por título, garantizando búsquedas rápidas y balanceadas.

Árbol B → organiza libros por año de publicación, ideal para consultas por rango.

Árbol B+ → (opcional, si lo usas) extiende el árbol B para un acceso más secuencial.

Cada estructura fue diseñada desde cero, con control de memoria manual, validación de datos desde CSV y soporte para visualización con Graphviz.

2. Requisitos Técnicos

2.1 Hardware

- Procesador Intel/AMD 64 bits
- Memoria RAM mínima: **4 GB**
- Espacio en disco: **200 MB** libres (para binarios y gráficos)

2.2 Software

- **Windows 10/11**, macOS o Linux
- **C++ Compiler** compatible con C++20
- **CMake 3.15 o superior**
- **Graphviz 14.0 o superior**
- **CLion o Visual Studio Code** (opcional, para entorno de desarrollo)

---

3. Instalación y Configuración

3.1 Instalación del Compilador C++

🔹 *Opción 1: MSVC (Windows)*

1. Instalar **Visual Studio Build Tools** desde https://visualstudio.microsoft.com/es/downloads/
2. Durante la instalación, marcar:
   - "C++ Build Tools"
   - "Windows 10 SDK"

🔹 *Opción 2: MinGW (Windows)*

1. Descargar **MinGW-w64** desde https://www.mingw-w64.org/downloads/
2. Instalar y agregar al PATH: setx PATH "%PATH%;C:\MinGW\bin"

Luego verificar: g++ --version

3.2 Instalación de CMake

1. Descargar desde https://cmake.org/download/.
2. Durante la instalación, activar la opción **"Add CMake to PATH"**.
3. Verificar con: cmke—version

3.3 Instalación de Graphviz

1. Descargar **Graphviz** desde https://graphviz.org/download/
2. Instalar y marcar "**Add Graphviz to PATH**".
3. Verificar instalación:

```
4. dot -V
```

   Resultado esperado:

```
dot - graphviz version 14.0.1 (2025-02-10)
```

Si Graphviz no se agrega automáticamente al PATH, se puede hacer manualmente:

```
setx PATH "%PATH%;C:\Program Files\Graphviz\bin"
```

3.4 Configuración de CLion

1. Abrir CLion → **File → Settings → Toolchains**
2. Verificar:
   - C Compiler: `gcc.exe` o `cl.exe`
   - C++ Compiler: `g++.exe` o `cl.exe`
   - CMake: detectado automáticamente
3. En **Build Type**, seleccionar:
   - `Debug` para pruebas
   - `Release` para entregar la versión final

3.5 Compilación del Proyecto

1. Abre la terminal en la raíz del proyecto (donde está el `CMakeLists.txt`):

- `mkdir build`
- `cd build`
- `cmake ..`
- `cmake --build .`

2. El ejecutable aparecerá en:

- `cmake-build-debug/untitled.exe`

3. Para volver a compilar después de cambios:

- `cmake --build . --target clean`
- `cmake --build .`

4. Arquitectura del Sistema

Módulo	Descripción
`main.cpp`	Control del menú principal e interacción con el usuario.
`LeerArchivo.cpp/.h`	Carga y validación del CSV.
`Libro.cpp/.h`	Estructura base de datos del libro.
`ArbolAVL.cpp/.h`	Árbol AVL para gestión de títulos.
`ArbolB.cpp/.h`	Árbol B para búsqueda por año.
`ArbolBmas.cpp/.h`	Árbol B+ (si aplica) para lectura secuencial.
`ListaNoOrdenada.cpp/.h`	Apoyo para almacenamiento temporal.

5. Complejidad Algorítmica (Big O)

5.1 Lectura y Validación de CSV

Operaciones:

- Leer línea por línea.
- Validar duplicados por ISBN.
- Ignorar errores de formato.

Complejidad:

- Lectura: `O(n)`
- Validación duplicados (usando `std::set`): `O(log n)` por inserción → total `O(n log n)`
  Total: `O(n log n)`

5.2 Árbol AVL

Operación	Descripción	Complejidad	Justificación
Insertar	Inserta un libro por título	`O(log n)`	Altura logarítmica y máximo una rotación simple o doble.
Eliminar	Elimina un nodo y rebalancea	`O(log n)`	Rebalanceo local en recorrido de altura `O(log n)`.
Buscar	Busca por título	`O(log n)`	Comparación descendiendo a través de niveles del árbol.
InOrden	Lista ordenada de libros	`O(n)`	Recorrido completo del árbol.

Justificación formal:
El árbol AVL garantiza que `|altura_izq - altura_der| ≤ 1`, manteniendo altura `h = O(log n)` en todo momento.
Cada inserción y eliminación afecta como máximo `O(log n)` nodos con operaciones `O(1)` (rotaciones).

5.3 Árbol B

Operación	Descripción	Complejidad	Justificación
Insertar	Inserta por año, divide nodos si es necesario	`O(log n)`	Cada división ocurre en un nivel del árbol.
Eliminar	Fusiona o presta claves de hijos	`O(log n)`	Requiere rebalancear solo en el camino recorrido.
Buscar	Localiza libro por año	`O(log n)`	En cada nivel hay hasta `2T-1` comparaciones (constante).
Buscar por rango	Encuentra todos los libros entre años `[a,b]`	`O(log n + k)`	Se localiza el nodo inicial (`O(log n)`) y se recorren `k` resultados.
Graficar	Genera archivo DOT recursivamente	`O(n)`	Cada nodo se procesa una sola vez.

Altura del árbol:

h=O(logTn)h = O(\log_T n)h=O(logTn)

donde `T` es el grado mínimo. Dado que `T` es constante, `O(\log_T n) = O(\log n)`.

5.4 Árbol B+

Operación	Descripción	Complejidad	Justificación
Insertar / Eliminar	Similar al Árbol B	`O(log n)`	Mismo balance y rebalanceo.
Búsqueda Secuencial	Recorre hojas enlazadas	`O(k)`	Acceso directo desde la primera hoja.
Búsqueda por rango	Encuentra `[a,b]` en hojas contiguas	`O(log n + k)`	Localización + recorrido secuencial.

5.5 Comparación de Búsquedas

Durante la ejecución, se comparan tiempos entre AVL y B con `std::chrono::high_resolution_clock`.

- AVL (por título): `O(log n)`
- B (por año): `O(log n)`

- Resultado esperado: tiempos similares, diferencias menores a milisegundos para `n < 10^5`.

---

6. Generación de Gráficos con Graphviz

Cada árbol puede visualizarse gracias al archivo `.dot` generado automáticamente.

Ejemplo de generación:

```
std::string comando =
    "\"C:\\Progra~2\\Graphviz\\bin\\dot.exe\" -Tpng \"" + rutaDot + "\" -
o \"" + rutaPng + "\"";
int ret = system(comando.c_str());
```

Complejidad:

- Generar archivo DOT → `O(n)`
- Renderizado con Graphviz → depende del tamaño (`O(n)` típico)

Salida:

- `arbolAVL.dot` → `arbolAVL.png`
- `arbolB.dot` → `arbolB.png`
  Ubicados en `C:\Users\Ana\Desktop\BibliotecaEdd\`

7. Manejo de Errores

Situación	Mensaje
Archivo CSV no encontrado	`"No se pudo abrir el archivo"`
Año inválido	`"Línea ignorada: Año inválido"`
ISBN duplicado	`"Línea ignorada: ISBN duplicado"`
Error al ejecutar Graphviz	`"Error al ejecutar Graphviz. Código: 1"`

---

8. Mantenimiento y Extensión

- **Agregar campos adicionales:** Modificar la clase `Libro`.
- **Cambiar estructura:** Sustituir AVL/B por árboles Red-Black o HashMaps.
- **Migración a GUI:** Integrar con Qt o JavaFX conectando la capa lógica actual.
- **Internacionalización:** Cambiar mensajes a inglés/español desde una clase `Mensajes.h`.

Conclusión

El sistema fue diseñado bajo principios de **eficiencia algorítmica**, **modularidad**, y **portabilidad**.
Las estructuras implementadas (AVL, B, B+) garantizan rendimiento óptimo incluso con grandes volúmenes de datos.

DIAGRAMAS

DIAGRAMA DEL PROYECTO

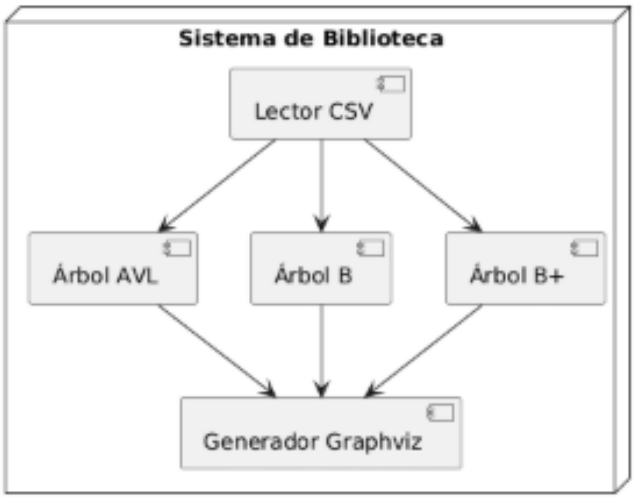

DIAGRAMA DE SECUENCIA

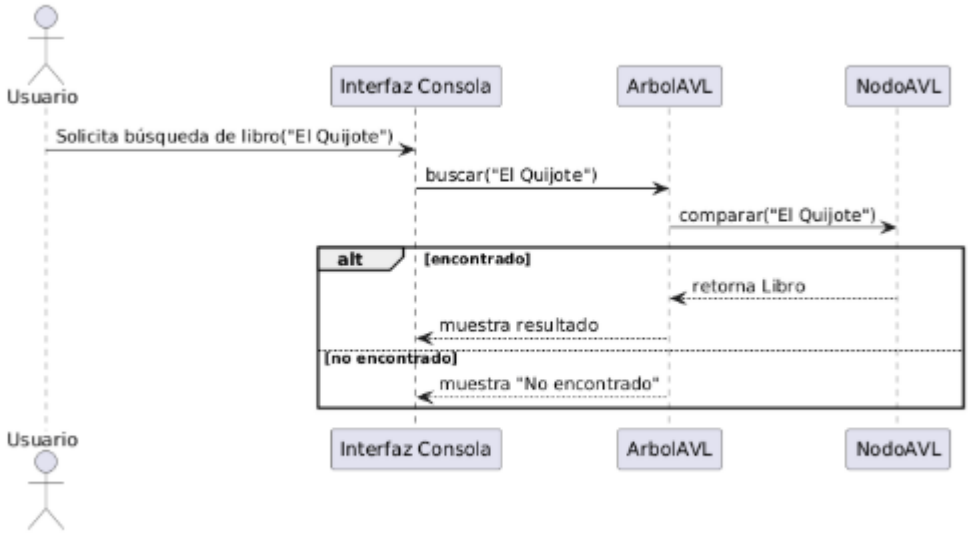

DIAGRAMA DE TABLAS

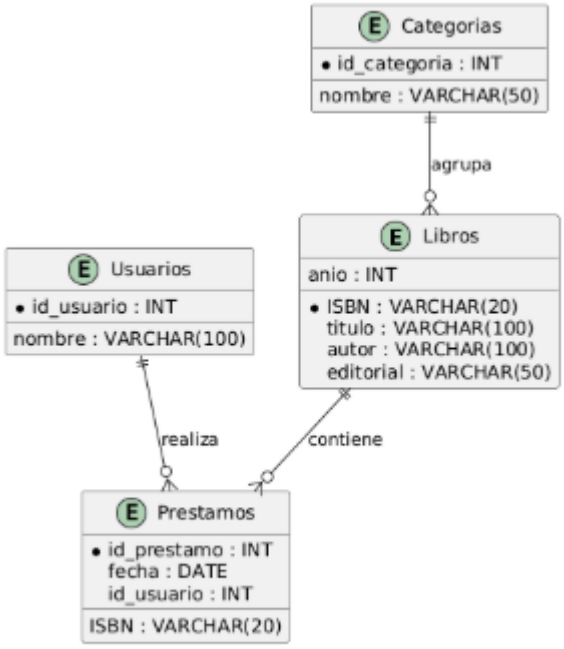

DIAGRAMA E-R

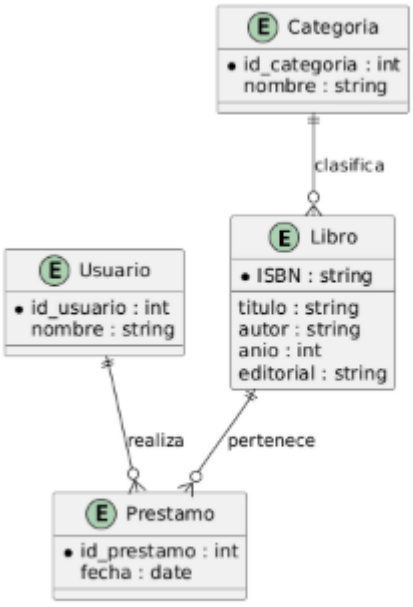

DIAGRAMA DE CLASES