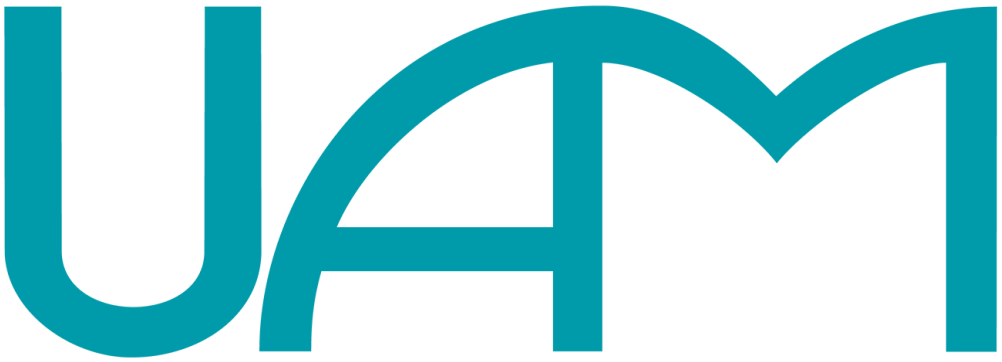

Programación Estructurada

Grupo 1

Biblioteca Digital Estudiantil

Andy Steven Diaz Montenegro

24 de Noviembre de 2025

I. INTRODUCCIÓN

En la era digital actual, la cantidad de información disponible crece exponencialmente, haciendo que la capacidad de recuperar datos específicos de manera eficiente sea una necesidad crítica en cualquier sistema de software. El presente proyecto, titulado "Sistema de Búsqueda para una Biblioteca Digital Estudiantil", aborda esta problemática mediante el desarrollo de un prototipo funcional en C# con interfaz gráfica (Windows Forms).

El objetivo principal de este trabajo es aplicar y demostrar el funcionamiento de algoritmos de búsqueda fundamentales: la búsqueda lineal y la búsqueda binaria. A través de la implementación de un catálogo de libros virtual, se explora cómo diferentes estructuras de datos y algoritmos impactan en la recuperación de información, desde la búsqueda exacta de un autor hasta el hallazgo de palabras clave dentro de una descripción textual. Este informe detalla el marco teórico que sustenta estos algoritmos, la metodología de implementación utilizada y los resultados obtenidos.

II. DESARROLLO

2.1 Algoritmos de Búsqueda

Un algoritmo de búsqueda es un conjunto de instrucciones diseñadas para localizar un elemento específico dentro de una estructura de datos (Cormen et al., 2009). La eficiencia de estos algoritmos es vital para el rendimiento de bases de datos y motores de búsqueda modernos.

2.1.1 Búsqueda Lineal (Secuencial)

Es el método más simple de búsqueda. Consiste en recorrer la estructura de datos elemento por elemento, desde el inicio hasta el final, comparando cada ítem con el valor buscado.

Ventaja: Funciona en listas desordenadas.

Desventaja: Es ineficiente en grandes volúmenes de datos, con una complejidad temporal de O(n) (Knuth, 1998).

2.1.2 Búsqueda Binaria

Es un algoritmo más eficiente que sigue el enfoque de "divide y vencerás". Requiere que la lista esté previamente ordenada. El algoritmo compara el valor buscado con el elemento central de la lista; si no son iguales, determina si el valor es mayor o menor y descarta la mitad correspondiente, repitiendo el proceso.

Ventaja: Es extremadamente rápida, con una complejidad de O(log n). * Desventaja: Requiere un costo computacional previo de ordenamiento (Sedgewick & Wayne, 2011).

2.2 Explicación de Algoritmos Implementados

Para el desarrollo del prototipo, se creó una clase `Libro` con propiedades como Título, Autor, Año y Descripción. Se implementaron cuatro lógicas de búsqueda distintas:

1. Búsqueda Lineal (Por Título): Se implementó un recorrido secuencial (`foreach`) sobre la lista de libros. Se utilizó el método `IndexOf` con la propiedad `StringComparison.OrdinalIgnoreCase` para permitir coincidencias parciales y omitir la distinción entre mayúsculas y minúsculas, mejorando la experiencia de usuario frente a una comparación estricta.

2. Búsqueda Binaria (Por Autor): Para cumplir con el requisito de ordenamiento, se aplicó primero el método `Sort` a la lista, utilizando el nombre del autor como criterio. Posteriormente, se implementó el ciclo `while` clásico de la búsqueda binaria, ajustando los índices `izquierda` y `derecha` según la comparación alfabética de las cadenas de texto.

3. Búsqueda de Texto (En Descripción): Este algoritmo realiza un escaneo completo de las descripciones de los libros. A diferencia de la búsqueda por título, aquí se busca si una subcadena (palabra clave) está contenida dentro de una cadena mayor, permitiendo encontrar libros por su temática (ej. "piloto", "distopía").

4. Búsqueda de Extremos (Mínimo y Máximo): Se implementó un algoritmo de recorrido simple que almacena temporalmente el libro con el año menor y mayor encontrados hasta el momento, actualizando estas variables tras comparar con cada elemento de la colección.

III. CONCLUSIONES

El desarrollo de este sistema de biblioteca digital ha permitido comprobar en la práctica las diferencias teóricas entre los algoritmos de búsqueda. Se concluye que:

1. La Búsqueda Lineal es versátil y fácil de implementar para búsquedas de texto flexible (como títulos parciales o descripciones), donde el ordenamiento previo es costoso o innecesario.

2. La Búsqueda Binaria es superior en velocidad para datos exactos (como nombres de autores), pero su dependencia del ordenamiento previo añade una capa de complejidad en la gestión de la lista.

3. La implementación en Windows Forms facilita la interacción del usuario con estos algoritmos, demostrando que la lógica de programación backend es inseparable de una buena interfaz de usuario. En resumen, no existe un "mejor" algoritmo universal; la elección depende de la naturaleza de los datos (ordenados vs. desordenados) y del tipo de consulta que el usuario necesita realizar.

ANEXOS

Imagen #1: Interfaz con todos los libros de la biblioteca

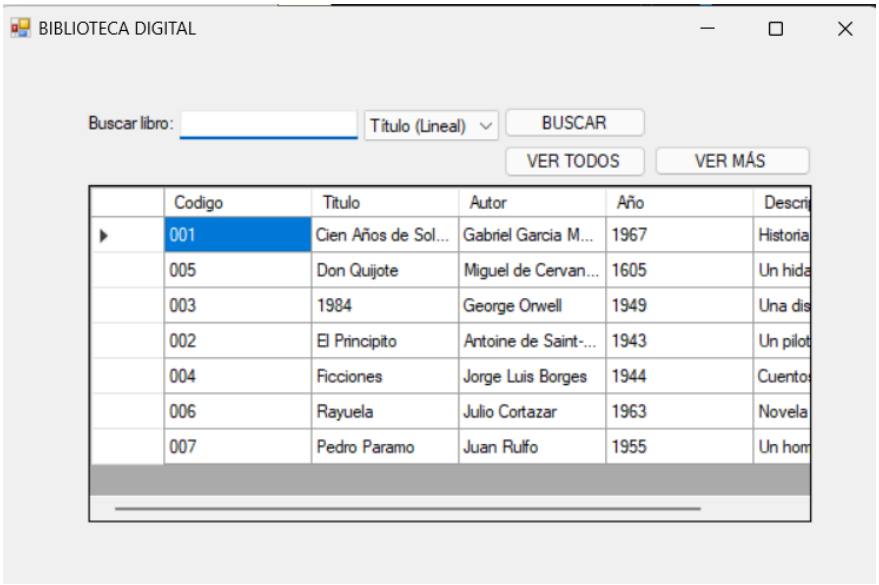

Imagen #2: Resultado de una búsqueda binaria por autor

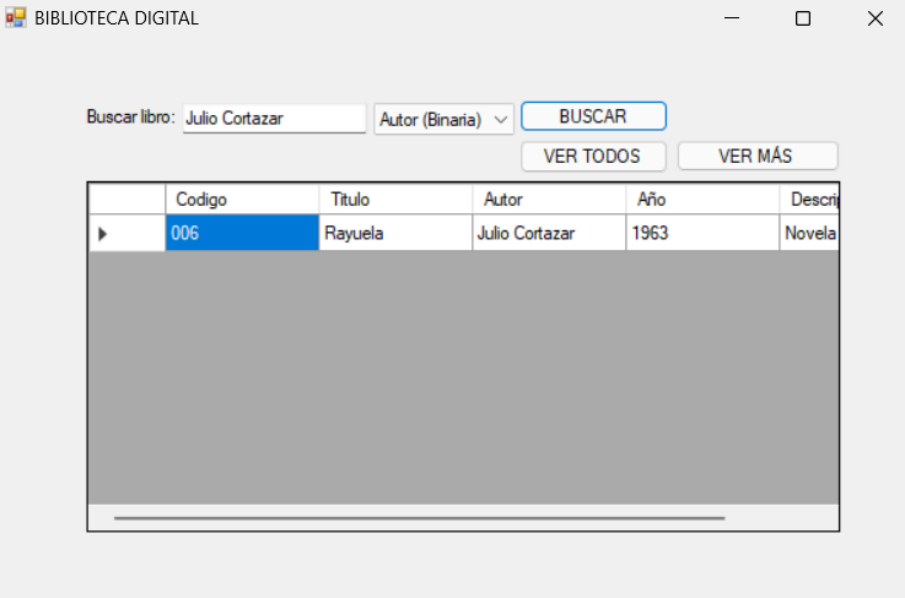

Imagen #3: Repositorio GitHub

REFERENCIAS

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.

Knuth, D. E. (1998). *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley.

Microsoft. (2023). *Documentación de C# - List<T>.Sort Método*. Microsoft Learn.

Sedgewick, R., & Wayne, K. (2011). *Algorithms* (4th ed.). Addison-Wesley Professional.