Programación Estructurada

Informe de Caso de Estudio:

"Sistemas de Búsqueda para Biblioteca Digital"

Autor:

Rodolfo Alfredo Ramírez Collado (22010929)

Docente: Msc. José Durán

Miércoles 26 de noviembre, 2025 Managua, Nicaragua

1. Introducción

En el contexto actual de la sociedad de la información, el volumen de datos que manejan las instituciones educativas crece de manera exponencial. La Universidad, en su esfuerzo por modernizar sus servicios académicos, ha planteado la necesidad de desarrollar una **Biblioteca Digital Estudiantil**. Sin embargo, almacenar información es inútil si no se dispone de mecanismos efectivos para recuperarla.

La capacidad de localizar un libro, un autor o un código específico de manera instantánea es lo que define la usabilidad y la calidad de un sistema de software. Este documento presenta la investigación teórica, el análisis técnico y el diseño de algoritmos fundamentales necesarios para construir el módulo de búsqueda de dicho sistema. Se examinan a profundidad los paradigmas de búsqueda lineal y binaria, así como su aplicación en estructuras de datos, contrastando su eficiencia computacional (Big O) y su pertinencia para distintos escenarios de volumen de datos. El objetivo final es sentar una base sólida de código en C# que sea escalable y eficiente.

2. Marco Teórico: Algoritmos de Búsqueda

2.1. Definición Conceptual

Un algoritmo de búsqueda es un procedimiento computacional diseñado para localizar un elemento específico (conocido como *target* o clave) dentro de una colección de datos. El éxito del algoritmo se define por dos resultados posibles: devolver la ubicación exacta del dato o determinar con certeza su inexistencia. La

eficiencia de estos algoritmos no solo ahorra tiempo de cómputo, sino que es crítica para la experiencia del usuario final (UX).

2.2. Análisis Detallado de las Técnicas

A. Búsqueda Lineal (Sequential Search)

Es el enfoque más intuitivo y directo. Metafóricamente, es comparable a buscar una carta específica en una baraja desordenada volteando una por una.

- **Mecánica:** El algoritmo recorre la estructura de datos secuencialmente, comenzando por el índice 0 hasta el índice n-1. En cada paso, compara el valor almacenado con el valor buscado.
- **Precondiciones:** Es el algoritmo más versátil, ya que **no requiere** que los datos estén ordenados previamente.
- **Análisis de Complejidad (Big O):**
    - *Mejor caso (O(1)):* El elemento buscado está en la primera posición.
    - *Peor caso (O(n)):* El elemento está al final de la lista o no existe.
    - *Promedio:* Requiere recorrer la mitad de la lista (n/2).

B. Búsqueda Binaria (Binary Search)

Es un algoritmo de alta eficiencia basado en el paradigma "divide y vencerás". Es análogo a buscar una palabra en un diccionario físico: no se lee hoja por hoja, sino que se abre a la mitad y se descarta la sección donde alfabéticamente no puede estar la palabra.

- **Mecánica:** Se calcula el índice central de la colección.
    - Si el valor central es el buscado, termina el proceso.

    - Si el valor buscado es menor que el central, se descarta la mitad derecha y se repite el proceso con la mitad izquierda.
    - Si es mayor, se descarta la mitad izquierda.
- **Precondiciones:** Es **estrictamente obligatorio** que la lista esté ordenada (numérica o alfabéticamente).
- **Análisis de Complejidad (Big O):**
    - *Complejidad:* O(log n). Esto significa que si duplicamos la cantidad de datos, el algoritmo solo necesita **un paso adicional** para encontrar el valor. Es exponencialmente superior a la búsqueda lineal en grandes volúmenes.

C. Búsqueda en Estructuras No Lineales

La información en una biblioteca no siempre es una lista simple; a menudo es jerárquica (Categoría -> Subcategoría -> Libro). Para esto se usan estructuras como Árboles (Trees) y Grafos.

- **Búsqueda en Profundidad (DFS - Depth First Search):** Explora una rama del árbol completa hacia abajo antes de retroceder. Útil para resolver laberintos o simulaciones de juegos.
- **Búsqueda en Anchura (BFS - Breadth First Search):** Explora todos los nodos vecinos al nivel actual antes de bajar al siguiente nivel. Es el estándar para encontrar "el camino más corto" en mapas o redes sociales (conexiones de amigos).

3. Aplicación en el Mundo Profesional y Casos Reales

La teoría de búsqueda es la columna vertebral de la industria tecnológica moderna. Su aplicación va más allá de simples listas:

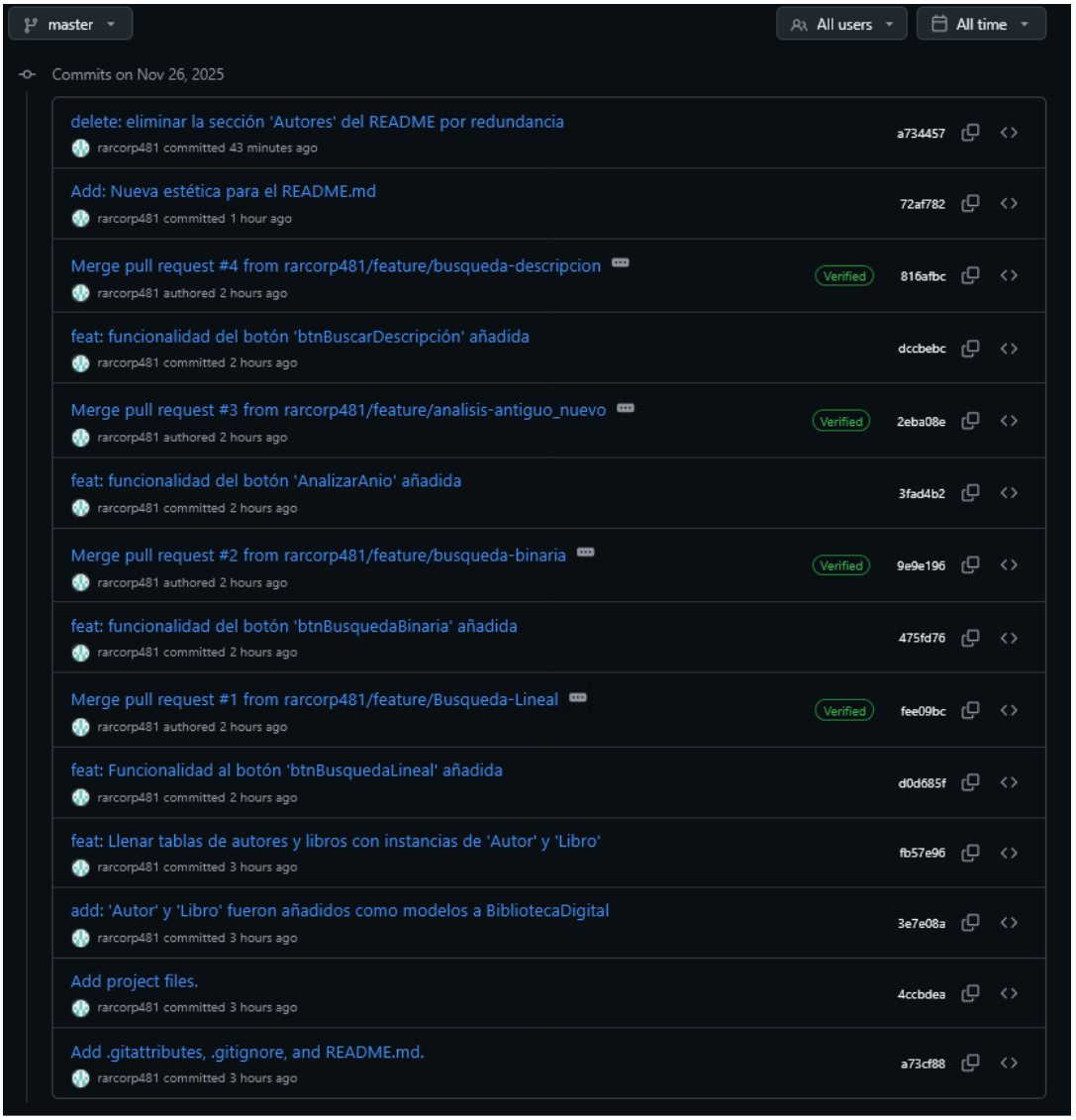
1. **Sistemas de Bases de Datos (SQL y NoSQL):** Cuando un estudiante busca un libro por su ID en el sistema universitario, la base de datos no realiza un escaneo completo (Full Table Scan) a menos que sea inevitable. Utiliza **Índices** (estructuras basadas en B-Trees o Hash Maps) que permiten saltos directos a la ubicación de la memoria donde reside el dato, comportándose de manera similar a una búsqueda binaria optimizada.
2. **Plataformas de Streaming y E-Commerce:** Empresas como Spotify o Amazon manejan catálogos de millones de ítems. Utilizan algoritmos de búsqueda híbridos y motores como *Elasticsearch*. Estos permiten búsquedas difusas (encontrar resultados aunque el usuario escriba mal una palabra) y búsquedas facetadas (filtrar por género, precio y valoración simultáneamente) en milisegundos.
3. **Inteligencia Artificial y Pathfinding:** En videojuegos o sistemas de GPS (como Waze o Google Maps), se utilizan algoritmos de búsqueda heurística como *A (A-Star)*. Estos algoritmos buscan el camino más eficiente entre dos puntos en un grafo ponderado (el mapa), evaluando el costo del tráfico y la distancia.

4. Comparativa Técnica: Ventajas y Desventajas

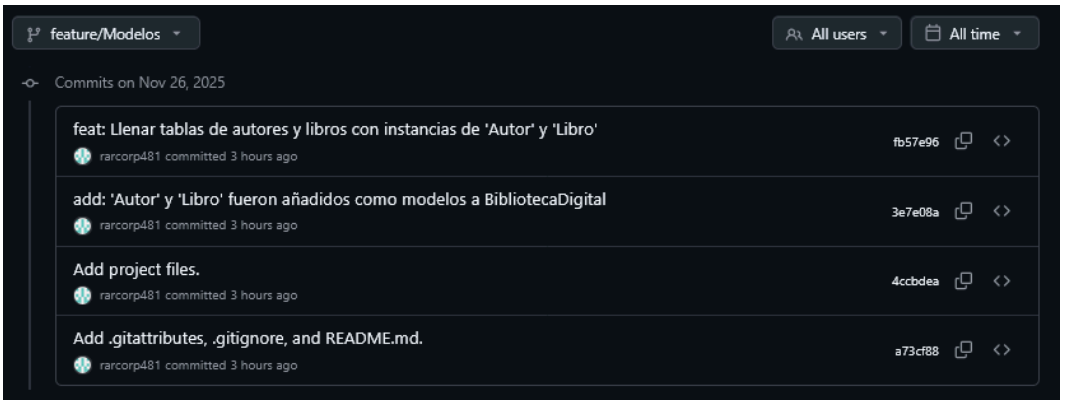
Característica	Búsqueda Lineal	Búsqueda Binaria
Complejidad Temporal	O(n) - Lineal (Lenta en grandes volúmenes)	O(log n) - Logarítmica (Muy rápida)
Requisito de Orden	Ninguno. Funciona en datos caóticos.	Estricto. Los datos deben estar ordenados.
Inserción de Datos	Rápida. Se agrega al final y listo.	Lenta. Al insertar, se debe reordenar la lista para mantener la integridad.
Implementación	Sencilla, pocas líneas de código.	Moderada, requiere lógica de índices y bucles `while`.
Uso Ideal	Listas pequeñas (<100 elementos) o datos no ordenados.	Bases de datos grandes, arrays estáticos masivos.

5. Evidencias del uso de GitHub

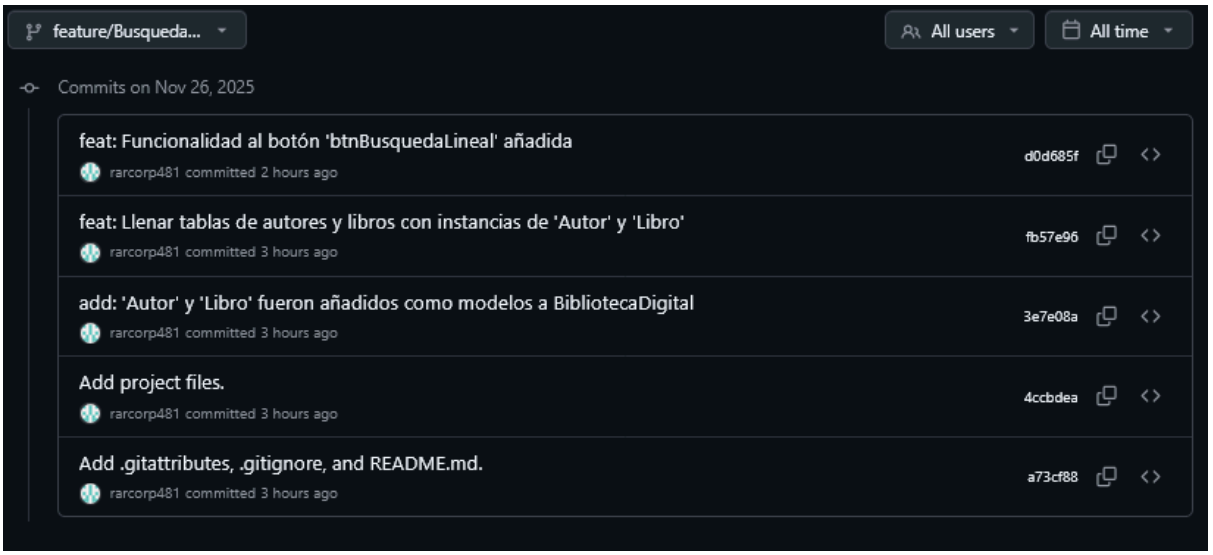
1. **Captura de la estructura de commits en la rama ‘master’:**



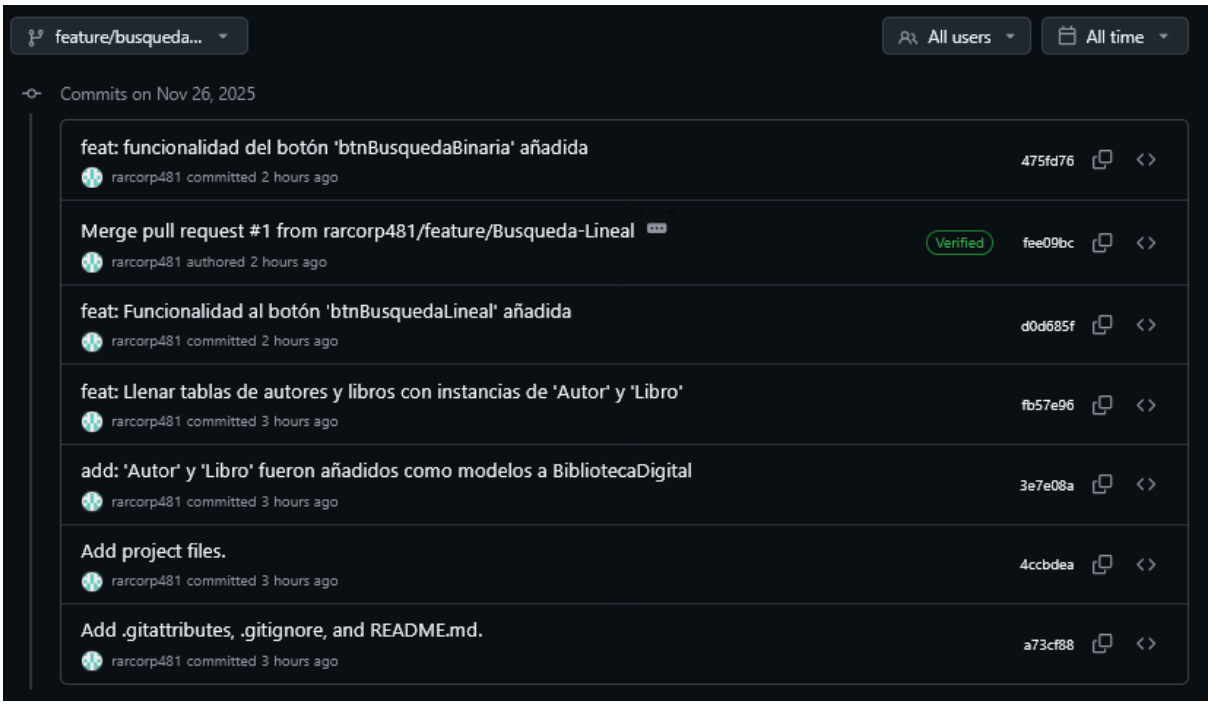
2. **Captura de los commits realizados en la rama ‘feature/Modelos:**



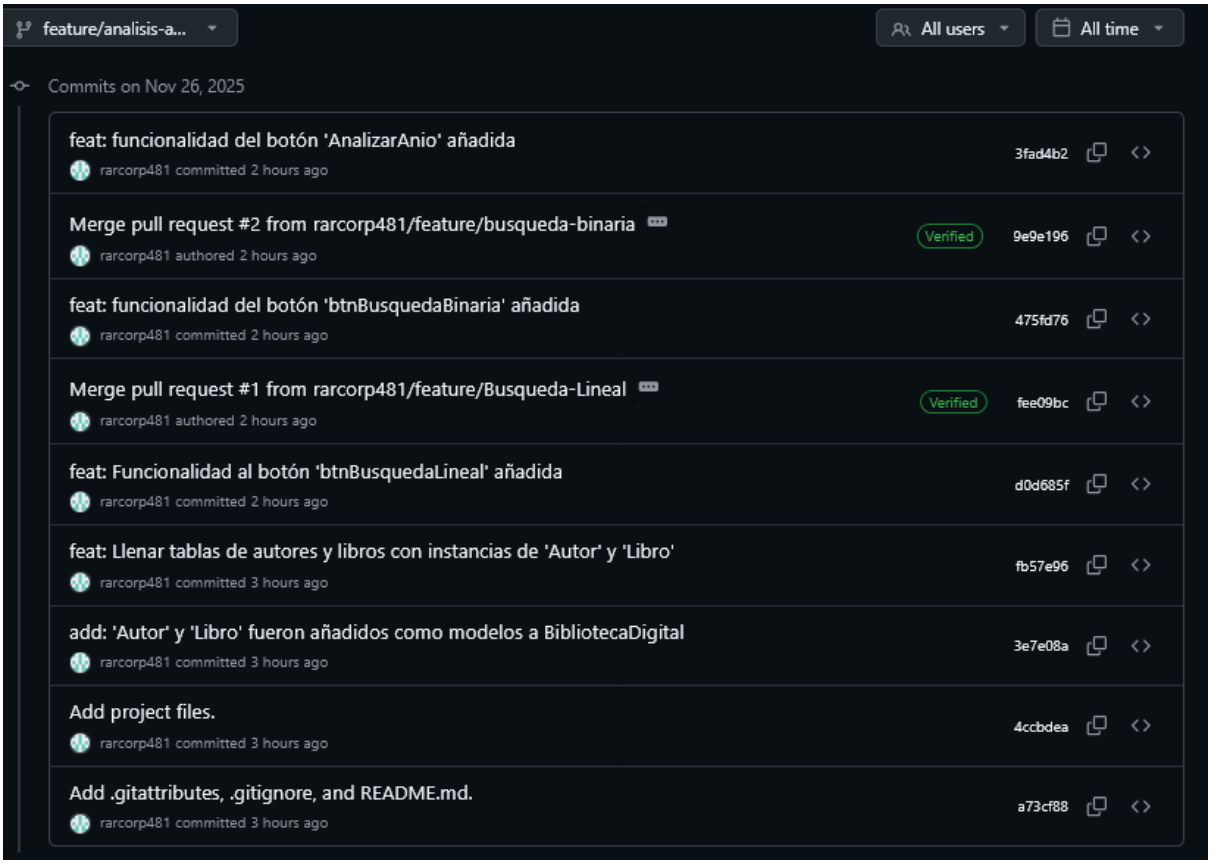
3. **Captura de los commits realizados en la rama ‘feature/Busqueda-Lineal’:**



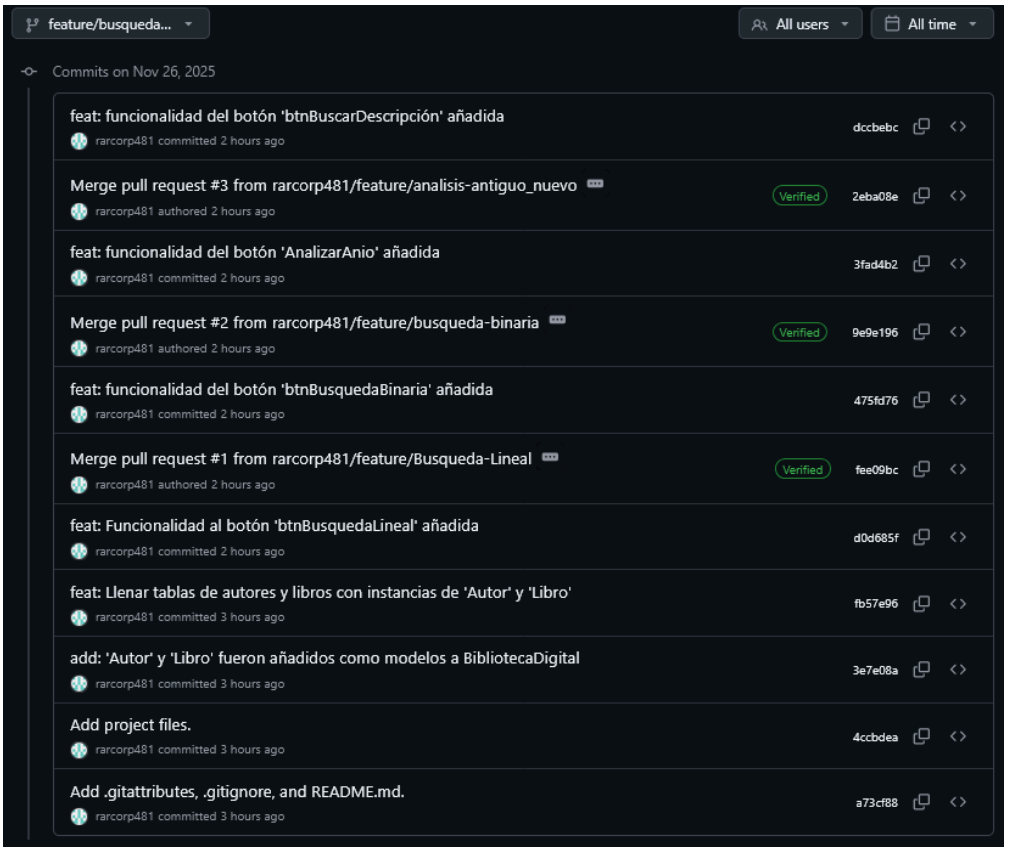
4. **Captura de los commits realizados en la rama ‘feature/busqueda-binaria’:**



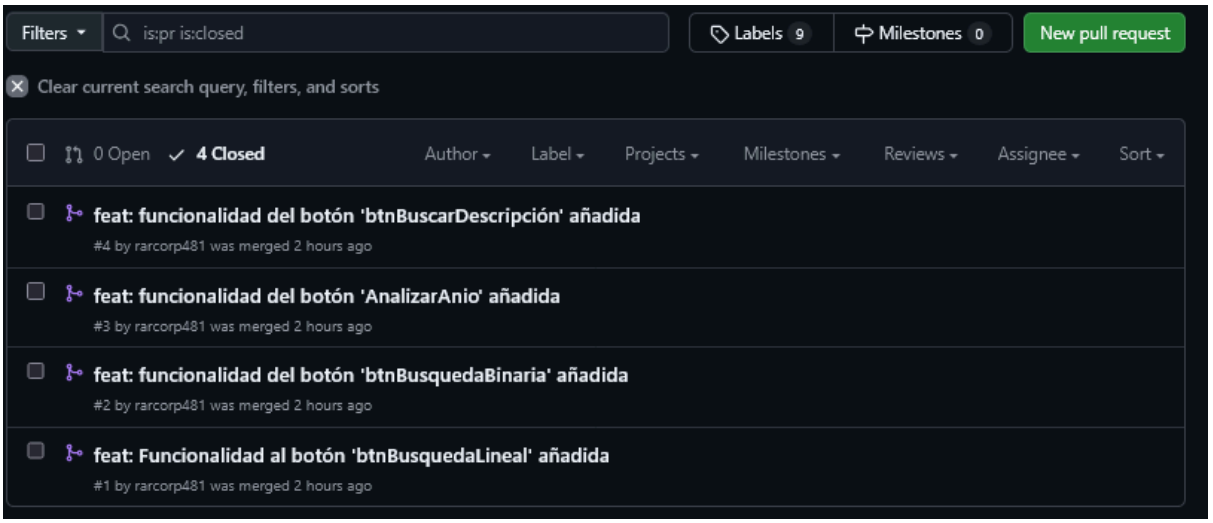
5. **Captura de los commits realizados en la rama ‘feature/analisis-antiguo_nuevo’:**



6. **Captura de los commits realizados en la rama ‘feature/busqueda-descripción’:**



7. **Pull requests del proyecto:**



8. **Archivo README.md:**

🚀 Funcionalidades Principales

La aplicación cuenta con una interfaz gráfica basada en pestañas (`TabControl`) que organiza las distintas operaciones:

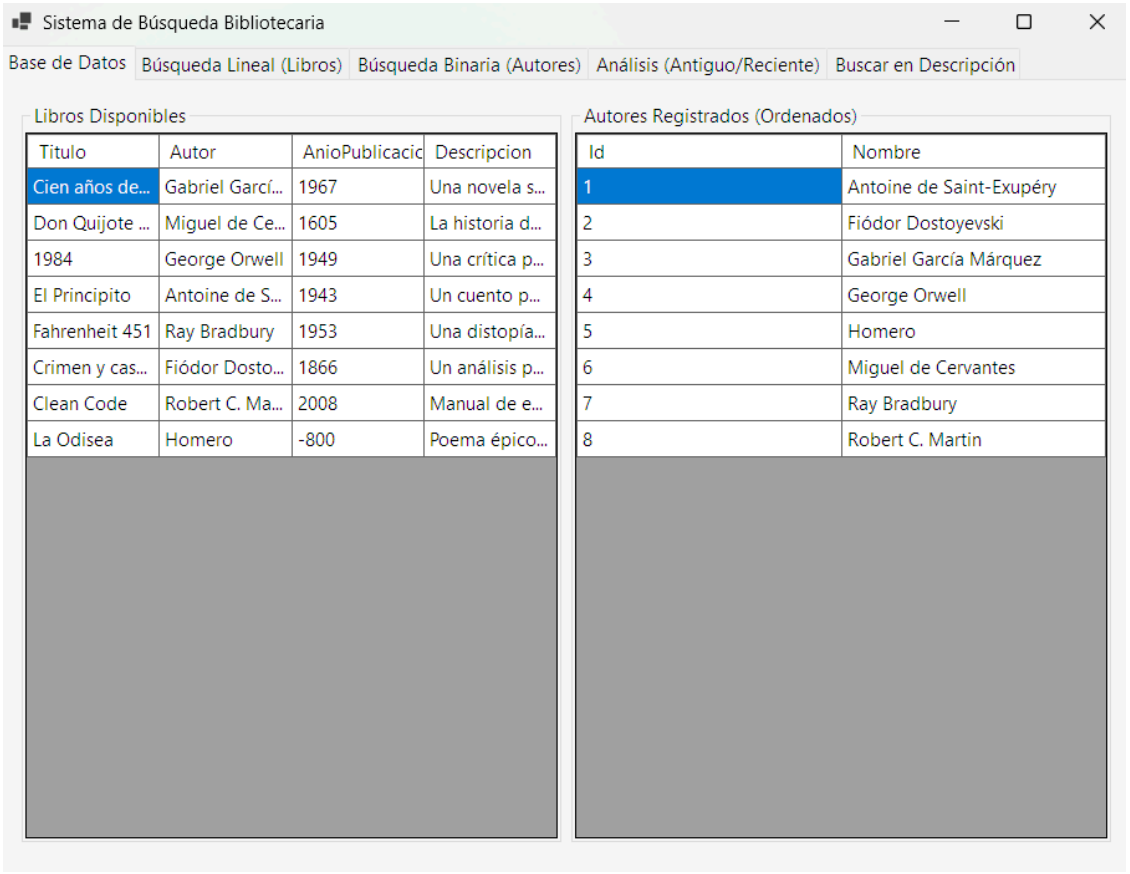
1. **Base de Datos Visual:** Vista general de todos los libros y autores disponibles en el sistema.
2. **Búsqueda de Libros:** Localización exacta por título.
3. **Búsqueda de Autores:** Localización rápida en listas ordenadas.
4. **Análisis de Colección:** Identificación automática de obras por antigüedad.
5. **Búsqueda Contextual:** Rastreo de palabras clave dentro de las descripciones de las obras.

🧠 Algoritmos Implementados

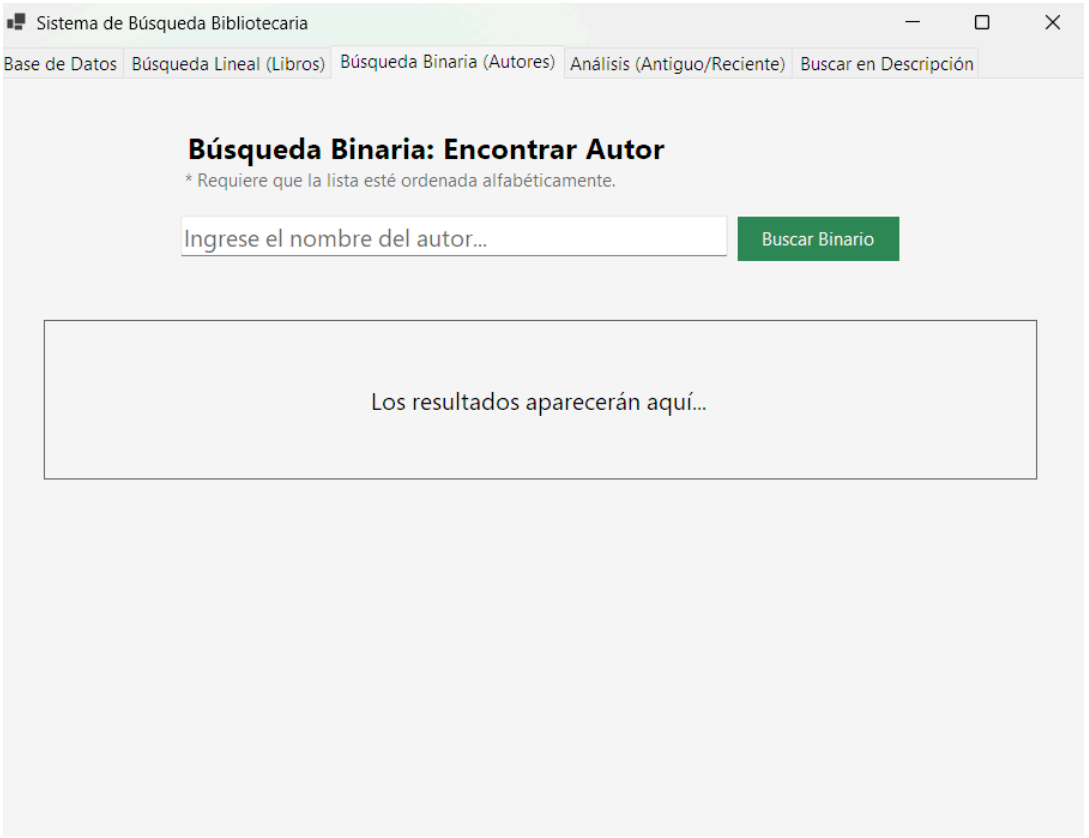
Este proyecto implementa manualmente los siguientes algoritmos para fines educativos:

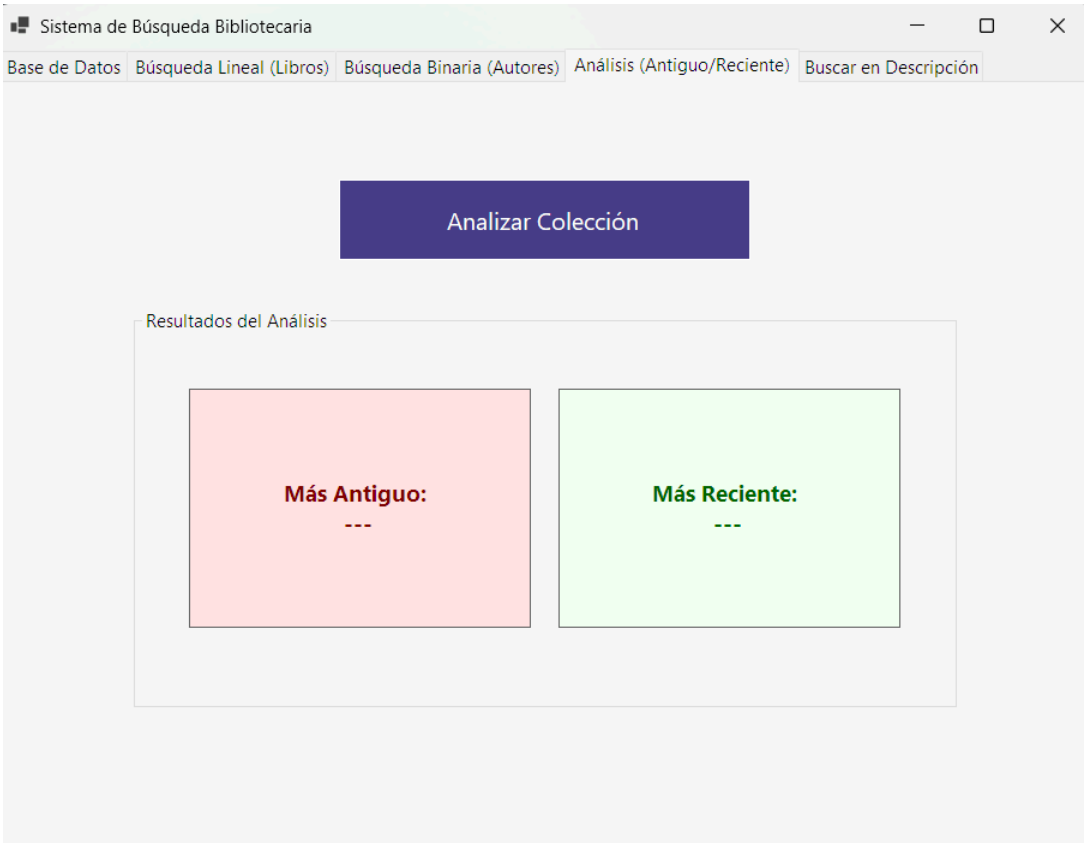
Algoritmo	Tipo	Complejidad (Big O)	Descripción
Búsqueda Lineal	Secuencial	$O(n)$	Recorre la lista de libros uno por uno hasta encontrar el título exacto.
Búsqueda Binaria	Divide y Vencerás	$O(\log n)$	Algoritmo optimizado para buscar autores en una lista previamente ordenada.
Recorrido Min/Max	Lineal	$O(n)$	Itera sobre la colección para encontrar el libro más antiguo y el más reciente simultáneamente.
Búsqueda de Patrones	String Matching	$O(n \cdot m)$	Busca coincidencias parciales de texto dentro de las descripciones de los libros.

9. **Capturas de pantalla del proyecto en ejecución:**


Sistema de Búsqueda Bibliotecaria
Base de Datos
Búsqueda Lineal (Libros)
Búsqueda Binaria (Autores)
Análisis (Antiguo/Reciente)
Buscar en Descripción
Libros Disponibles
Titulo
Autor
AnioPublicacic
Descripcion
Cien años de...
Gabriel Garcí...
1967
Una novela s...
Don Quijote ...
Miguel de Ce...
1605
La historia d...
1984
George Orwell
1949
Una crítica p...
El Principito
Antoine de S...
1943
Un cuento p...
Fahrenheit 451
Ray Bradbury
1953
Una distopía...
Crimen y cas...
Fiódor Dosto...
1866
Un análisis p...
Clean Code
Robert C. Ma...
2008
Manual de e...
La Odisea
Homero
-800
Poema épico...
Autores Registrados (Ordenados)
Id
Nombre
1
Antoine de Saint-Exupéry
2
Fiódor Dostoyevski
3
Gabriel García Márquez
4
George Orwell
5
Homero
6
Miguel de Cervantes
7
Ray Bradbury
8
Robert C. Martin


Sistema de Búsqueda Bibliotecaria
Base de Datos
Búsqueda Lineal (Libros)
Búsqueda Binaria (Autores)
Análisis (Antiguo/Reciente)
Buscar en Descripción
Búsqueda Lineal: Encontrar Libro
Ingrese el nombre del libro exacto...
Buscar
Los resultados aparecerán aquí...


Sistema de Búsqueda Bibliotecaria
Base de Datos
Búsqueda Lineal (Libros)
Búsqueda Binaria (Autores)
Análisis (Antiguo/Reciente)
Buscar en Descripción
Búsqueda Binaria: Encontrar Autor
* Requiere que la lista esté ordenada alfabéticamente.
Ingrese el nombre del autor...
Buscar Binario
Los resultados aparecerán aquí...


Sistema de Búsqueda Bibliotecaria
Base de Datos
Búsqueda Lineal (Libros)
Búsqueda Binaria (Autores)
Análisis (Antiguo/Reciente)
Buscar en Descripción
Analizar Colección
Resultados del Análisis
Más Antiguo:

Más Reciente:

6. Reflexión Final

¿Por qué es importante la búsqueda en la ciencia de la computación y en el desarrollo de software?

La búsqueda de información es el proceso fundamental que da sentido al almacenamiento de datos. En la ciencia de la computación, la importancia de estos algoritmos no reside simplemente en "encontrar algo", sino en la **escalabilidad y la eficiencia de los recursos**.

En el contexto de las **estructuras de datos**, comprender los algoritmos de búsqueda dicta la arquitectura del software. Si sabemos que un sistema requerirá miles de búsquedas por segundo (como un buscador de biblioteca), estamos obligados a diseñar estructuras que soporten Búsqueda Binaria o Índices Hash, sacrificando quizás un poco de velocidad en la escritura para ganar velocidad extrema en la lectura.

El **impacto en sistemas reales** es absoluto. Vivimos en la era del "Big Data"; sin algoritmos de búsqueda eficientes (O(log n) u O(1)), servicios como Google, las transacciones bancarias o las consultas médicas digitales serían funcionalmente imposibles debido a la latencia.

Finalmente, las **consecuencias de un algoritmo ineficiente** son severas. Un algoritmo de orden O(n) u O(n^2) aplicado a una base de datos de millones de registros puede congelar un servidor (bloqueo de CPU), aumentar drásticamente los costos de energía y hardware, y

causar el abandono de los usuarios debido a la lentitud de la interfaz. La optimización de la búsqueda es, en esencia, el respeto por el tiempo del usuario y los recursos de la máquina.

7. Referencias Bibliográficas

- Alvarez, A. (2023, 27 de febrero). *Estructuras de datos: Qué son y los 7 tipos más usados*. EDteam. https://ed.team/blog/estructuras-de-datos
- Khan Academy. (s.f.). *Búsqueda binaria*. Khan Academy Computación. https://es.khanacademy.org/computing/computer-science/algorithms/binary-search/a/binary-search
- Microsoft. (2023, 15 de septiembre). *Colecciones y estructuras de datos en .NET*. Microsoft Learn. https://learn.microsoft.com/es-es/dotnet/standard/collections/
- Pelayo, G. (2024, 1 de febrero). *Guía: Notación Big O - Gráfico de complejidad de tiempo*. freeCodeCamp.org. https://www.freecodecamp.org/espanol/news/hoja-de-trucos-big-o/
- Q2B Studio. (2025, 21 de septiembre). *Todo sobre la Búsqueda Binaria*. Q2B Studio Blog. https://www.q2bstudio.com/nuestro-blog/27263/todo-sobre-la-busqueda-binaria
- StudySmarter. (2024, 10 de junio). *Búsqueda Lineal: Explicación y ejemplos*. StudySmarter. https://www.studysmarter.es/resumenes/ciencias-de-la-computacion/algoritmos-en-ciencias-de-la-computacion/busqueda-lineal/