Juan David Ardila, Juan esteban Cardenas, Juan Esteban Pena, Julian Gomez, Nestor Piraquive

Equipo Trabajo: TuneTrace

I. [1] INTRODUCCIÓN

La música es una parte fundamental de la vida de muchas personas, y su diversidad es asombrosa. A menudo, los amantes de la música se encuentran explorando más allá de sus géneros y artistas favoritos, en busca de canciones que resuenen con aspectos específicos como el género, el autor, la duración o el tema. Sin embargo, encontrar nuevas canciones que coincidan con estos gustos particulares puede ser un desafío. Para abordar esta necesidad, proponemos el desarrollo de una aplicación de música con un enfoque diferente en la recomendación de canciones.

II. DESCRIPCIÓN DEL PROBLEMA A RESOLVER

El dilema que enfrentan los amantes de la música es la limitación de las plataformas de streaming actuales para ofrecer recomendaciones que a veces no tienen coherencia con lo solicitado. Esto resulta en una experiencia de búsqueda y descubrimiento limitada para aquellos que desean explorar aspectos más específicos de una canción. Nuestra aplicación busca resolver este problema al permitir a los usuarios definir sus gustos musicales en términos de elementos específicos de las canciones.

III. USUARIO DEL PRODUCTO DE SOFTWARE

En nuestra aplicación, existen dos tipos principales de clasificación: los usuarios Premium y los usuarios gratuitos. Los usuarios Premium disfrutan de todas las funcionalidades que la aplicación ofrece, sin restricciones en su uso, lo que les garantiza la mejor experiencia que nuestra aplicación puede proporcionar.

Por otro lado, los usuarios gratuitos tienen acceso a un conjunto más limitado de funcionalidades en comparación con los usuarios Premium. Algunas de estas funcionalidades pueden estar sujetas a ciertas restricciones, lo que significa que no experimentarán la misma calidad de experiencia que ofrece nuestra aplicación.

Todos los usuarios comienzan en nuestra aplicación como usuarios gratuitos, pero tienen la opción de convertirse en usuarios Premium al pagar un valor monetario determinado. Esta inversión les brinda acceso a todos los beneficios asociados con este rol.

IV. REQUERIMIENTOS FUNCIONALES DEL SOFTWARE

- *Ingreso de Canciones por Usuario:*

El usuario introduce una canción o una lista de canciones de su preferencia en una sección específica de la aplicación.

Requerimientos funcionales:

La aplicación almacena la lista de canciones introducidas por el usuario, junto con sus características específicas, como la duración, el género, el autor, entre otros. Si la aplicación no puede ejecutar este proceso con éxito, mostrará un mensaje de error al usuario, advirtiendo sobre la posible entrada incorrecta de datos.

- *Filtración de canciones personalizado:*

Después de ingresar los datos, el usuario podrá definir ciertas preferencias en una sección específica de la aplicación, lo que afectará la generación de recomendaciones de canciones de acuerdo con las preferencias seleccionadas.

Requerimientos funcionales:

La aplicación almacena la lista de preferencias seleccionadas por el usuario (puede estar vacía).

- *Recomendaciones de canciones:*

La aplicación proporciona una lista de canciones sugeridas que comparten características similares con la lista de canciones ingresada por el usuario y cumplen con las preferencias seleccionadas en la sección de filtración.

Requerimientos funcionales:

La aplicación realiza una consulta exhaustiva en la biblioteca de canciones (un repositorio con una gran cantidad de canciones disponibles para la aplicación) y luego efectúa una búsqueda parcial de canciones que compartan dos o más características con la lista de canciones ingresadas por el usuario, además de cumplir con las preferencias seleccionadas.

Posteriormente, la aplicación mostrará como respuesta al usuario una lista de canciones resultantes de la búsqueda parcial, ordenadas de mayor a menor en función de la cantidad de características coincidentes encontradas. Si no se encuentra ninguna canción que cumpla con los requisitos mencionados anteriormente, se mostrará un

[1]

mensaje notificando al usuario que no se han encontrado recomendaciones.

- *Guardado de lista de canciones escogidas por el usuario:*

El usuario tiene la opción de guardar la lista de canciones recomendadas por el sistema, ya sea como una nueva lista o en una lista existente. Estas listas guardadas se muestran en una sección específica de la aplicación, y el usuario también puede modificar el nombre de la lista, agregar nuevas canciones o eliminar las existentes.

Requerimientos funcionales:

La aplicación almacena la lista de canciones generada durante la búsqueda parcial y la muestra en la sección correspondiente. Al guardar la lista, la aplicación debe considerar si se trata de la creación de una nueva lista en la sección adecuada o la actualización de una lista preexistente en esa sección.

Las listas ubicadas en esta sección deben ser fácilmente modificables, lo que significa que se pueden realizar inserciones o eliminaciones de datos. Si la lista que se intenta guardar ya existe, se notificará al usuario sobre la posible duplicación de datos y se le pedirá que cambie el nombre de la lista. De manera similar, si durante la inserción de canciones en una lista existente se encuentran canciones duplicadas, estas no se agregarán a la lista.

V. DESCRIPCIÓN DE LA INTERFAZ DE USUARIO PRELIMINAR

Para esta primera versión preliminar del aplicativo se presentan los siguientes mockups que darán una idea del concepto que se tiene sobre Tune Trace:

Para la primera vista, se tiene principalmente el logo de la aplicación junto con un botón que permitirá al usuario ingresar a la segunda vista donde podrá ingresar canciones de su preferencia.

En la segunda vista, se encuentra un menú donde el usuario podrá por un lado filtrar los resultados por diferentes categorías como género, artista, tema, etc. En otro apartado también podrá consultar las listas que ya tenga guardadas para su posterior uso. Por último, se tiene una barra de búsqueda en donde el usuario puede ingresar canciones que sean de su agrado para que el aplicativo se base en ellas para realizar una recomendación exitosa.

En última vista se puede observar como después de haber ingresado las canciones que se usarán como referencia se mostrará en forma de lista las canciones que la aplicación recomienda al usuario, cumpliendo así su función.

Cabe mencionar que estas imágenes solo representan una idea de como se desea que sea el apartado visual de Tune Trace, no significa que la aplicación vaya a ser lanzada solamente en

Móvil, sino que aún se contempla la mejor plataforma para su lanzamiento.

VI. ENTORNOS DE DESARROLLO Y DE OPERACIÓN

Por el momento el proyecto se lanzará de manera local, en un PC portátil con OS windows 11 y será provisionalmente compatible solo con este. El desarrollo del mismo se realizará en el lenguaje de programación Java usando el IDE NetBeans, esto para el Backend del aplicativo.

Al momento del lanzamiento del programa, este será corrido provisionalmente en una VivoBook ASUS Laptop con un procesador AMD Ryzen 5 3500U con Radeon Vega Mobile Gfx con 8 núcleos a 2.1 GHz, 8 GB de RAM.

VII. PROTOTIPO DE SOFTWARE INICIAL

El aplicativo de Tune Trace se alojó en el siguiente repositorio de GitHub siguiendo las recomendaciones para su acceso: https://github.com/J-HuertaS/TuneTrace.git

VIII. IMPLEMENTACIÓN Y APLICACIÓN DE LAS ESTRUCTURAS DE DATOS

Dentro de las estructuras que se implementaron para el análisis de tiempo y complejidad, se tomaron en cuenta las siguientes estructuras de datos secuenciales.

Arreglo dinámico: Implementado cómo un arreglo que permite añadir elementos sin conocer el tamaño previo del arreglo. Dentro del proyecto su funcionalidad se plantea para el almacenamiento de todas las canciones existentes, en este arreglo se guardan previamente los datos de canciones obtenidos de distintas fuentes, para que al momento de que el usuario interactúe con la aplicación, se verifique dentro de este arreglo si la canción que desea añadir a su lista de intereses es realmente una canción existente.

Arreglo dinámico ordenado: Implementado cómo un arreglo que permite añadir elementos sin conocer el tamaño previo del arreglo, y en un orden específico. Dentro del proyecto su funcionalidad también está enfocada en el almacenamiento de las canciones existentes, al ser un arreglo ordenado resulta útil en la organización de datos numéricos cómo es el tiempo de duración o el año de lanzamiento de una canción.

Pilas y Colas: Se implementan en ambos casos cómo clases que heredan los métodos y atributos de una clase Linkedlist, haciendo uso de los métodos respectivos para cada estructura. Dentro del proyecto su utilidad se refleja en el almacenamiento de las canciones que ingresa el usuario cómo su lista de intereses, la cuál va a ser analizada por la aplicación para retornar sus recomendaciones, las cuales también se encuentran almacenadas a través de esta estructura, con las cuales el usuario puede interactuar.

Lista doblemente enlazada: Dentro del proyecto su implementación tomará las referencias del objeto canción, donde cada objeto toma los atributos de nombre, artista, año de lanzamiento, duración y género por cada canción que se encuentre guardada dentro del sistema. La utilidad que genera se encuentra en permitirle al usuario guardar las recomendaciones que le interesen, interactuar con ellas, filtrar las canciones de acuerdo a ciertos atributos de interés, cómo por ejemplo filtrar por género, filtrar por duración, entre otros.

IX. PRUEBAS DEL PROTOTIPO Y ANÁLISIS COMPARATIVO

Con el fin de comparar las diferentes estructuras implementadas, se desarrollaron casos de pruebas con cantidades diferentes de datos para cuatro funcionalidades que serán fundamentales a lo largo del desarrollo de nuestro proyecto, dentro de las cuales encontramos: Inserción de n datos, búsqueda de un dato, eliminación de un dato y consulta de todos los datos (retornado en arreglo estático). Lo anterior puede ser revisado en el repositorio, dentro del archivo Prototipo ubicado en la carpeta src. Sin embargo, a manera de resumen se presenta la siguiente información:

- Inserción de n elementos:

Estructura de datos	Complejidad Big-O
Doubly-linkedlist	O(n)
Stack	O(n)
Queue	O(n)
Arreglo dinámico	O(n)
Arreglo dinámico ordenado	O(n^2)

- Eliminación de un elemento:

Estructura de datos	Complejidad Big-O
Doubly-linkedlist	O(n)
Stack	O(n)
Queue	O(n)
Arreglo dinámico	O(n)
Arreglo dinámico ordenado	O(log(n))

- Búsqueda de un elemento:

Estructura de datos	Complejidad Big-O

Doubly-linkedlist	O(n)
Stack	O(n)
Queue	O(n)
Arreglo dinámico	O(n)
Arreglo dinámico ordenado	O(log(n))

- Consulta de todos los datos:

Estructura de datos	Complejidad Big-O
Doubly-linkedlist	O(n)
Stack	O(n)
Queue	O(n)
Arreglo dinámico	O(n)
Arreglo dinámico ordenado	O(n)

Sin embargo, para un mayor detalle se puede consultar los anexos donde se tienen tablas comparativas de los tiempos de cada estructura para cada una de las funcionalidades implementadas y, para un análisis más completo, el gráfico del análisis asintótico para cada caso.

X. ROLES Y ACTIVIDADES

Los siguientes son los roles asignados para esta entrega entre los integrantes:

ROL	Actividades fundamentales
Líder(esa)	Consultar a los otros miembros del equipo, atento que la información sea constante para todos. Aportar con la organización y plan de trabajo.
Coordinador(a)	Mantener el contacto entre todos, Programar y agendar reuniones; ser facilitador para el acceso a los recursos.
Experto(a)	Líder técnico que propende por coordinar las funciones y actividades operativas.
Investigador(a)	Consultar otras fuentes. Propender por resolver inquietudes comunes para todo el equipo.
Observador(a)	Siempre está atento en el desarrollo del proyecto y aporta en el momento apropiado cuando se requiera apoyo adicional por parte del equipo.
Animador(a)	Energía positiva, motivador en el grupo.
Secretario(a)	Se convierte en un facilitador de la comunicación en el grupo. Documenta (actas) de los acuerdos/compromisos realizados en las reuniones del equipo.
Técnico(a)	Aporta técnicamente en el desarrollo del proyecto.

En la siguiente tabla se registraron los roles como las actividades realizadas por cada miembro del equipo:

INTEGRANTE	ROL(ES)	ACTIVIDADES REALIZADAS (Listado)
Juan David Ardila	Secretario	Facilitador y documentador de las propuestas discutidas para la consolidación del proyecto
	Observador	Colaborador en el documento escrito
Juan Esteban Cardenas	Experto	Implementación del proyecto
		Toma de tiempos y creación de tablas para muestra.
	Investigador	Búsqueda de los métodos para poder implementar archivos JSON en el proyecto
	Técnico	Creación del repositorio y mantenimiento del mismo.
		Desarrollo del análisis asintótico comparativo entre las funcionalidades implementadas.
Juan Esteban Peña	Técnico	Implementación del proyecto
		Colaboración en la creación de los Mockups

	Investigador	Consulta de la implementación
Julian Gomez	Líder	Coordinar las reuniones pertinentes
		Colaborador en el documento escrito
	Coordinador	Estar en constante comunicación con los integrantes
		Creación del documento de trabajo
		Colaboración en la creación de los Mockups
Nestor Piraquive	Animador	Preocupación por el ánimo del grupo
	Observador	Colaborador en el documento escrito

XI. DIFICULTADES Y LECCIONES APRENDIDAS

Dentro de las dificultades presentadas dentro del desarrollo del proyecto nos encontramos con qué el análisis de tiempos para la ejecución de funciones en distintas estructuras de datos secuenciales, dependiendo de las capacidades de hardware del equipo en que se estén realizando, pueden llegar a tomar grandes intervalos de tiempo, por lo cuál se concluye que la notación asintótica nos permite realizar este tipo de comparaciones sin necesidad de realizar ese cálculo exhaustivo de tiempos entre cada estructura.

También se concluye que para un gran volumen de datos dinámicos, como el que se maneja en el proyecto, dado que el usuario tiene que estar interactuando todo el tiempo con las estructuras en diferentes posiciones, no resulta tan eficiente utilizar pilas y colas, por el contrario, es más recomendable hacer uso de los arreglos dinámicos y de las listas doblemente enlazadas para su implementación.

Adicionalmente, también se encontró que de las diferentes estructuras de datos que se han presentado dentro del programa, las estructuras secuenciales no son las más llamativas, dado que al utilizar un objeto más abstracto cómo una canción, que no puede ordenarse de una manera completamente secuencial, es mejor utilizar otro tipo de estructuras que permitan hacer balanceos, o generar prioridades dentro de la consulta de los datos.

XII. REFERENCIAS BIBLIOGRÁFICAS

[1] Weiss, M.A.: *Data Structures and Algorithm Analysis in C++, 4th Edition*, Pearson/Addison Wesley, 2014.

[2] Hernández, Z.J. y otros: *Fundamentos de Estructuras de Datos. Soluciones en Ada, Java y C++*, Thomson, 2005.

[3] Shaffer, Clifford A.: *Data Structures and Algorithm Analysis in C++, Third Edition*, Dover Publications, 2013. (En línea.)

[4] Campos Laclaustra, J.: *Apuntes de Estructuras de Datos y Algoritmos*, segunda edición, 2018. (En línea.)

[5] Martí Oliet, N., Ortega Mallén, Y., Verdejo López, J.A.: Estructuras de datos y métodos algorítmicos: 213 ejercicios resueltos. 2ª Edición, Ed. Garceta, 2013.

[6] Joyanes, L., Zahonero, I., Fernández, M. y Sánchez, L.: Estructura de datos. Libro de problemas, McGraw Hill, 1999.

ANEXO 1: Gráficas de tiempo de ejecución

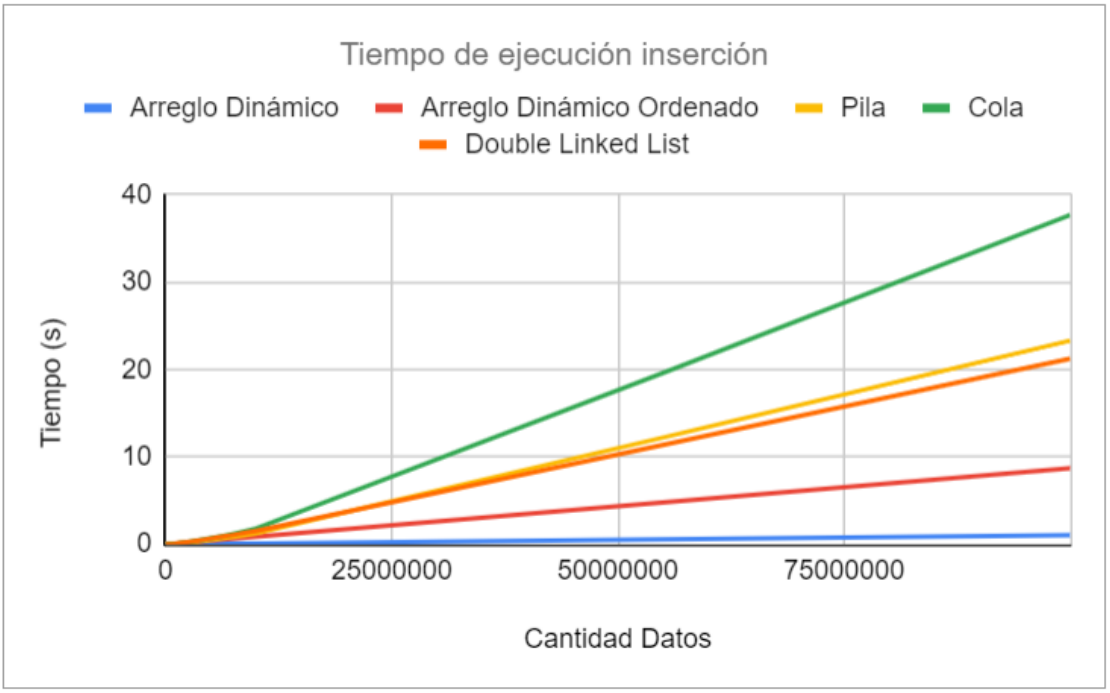

Imagen 1: Análisis de tiempo para insertar n elementos

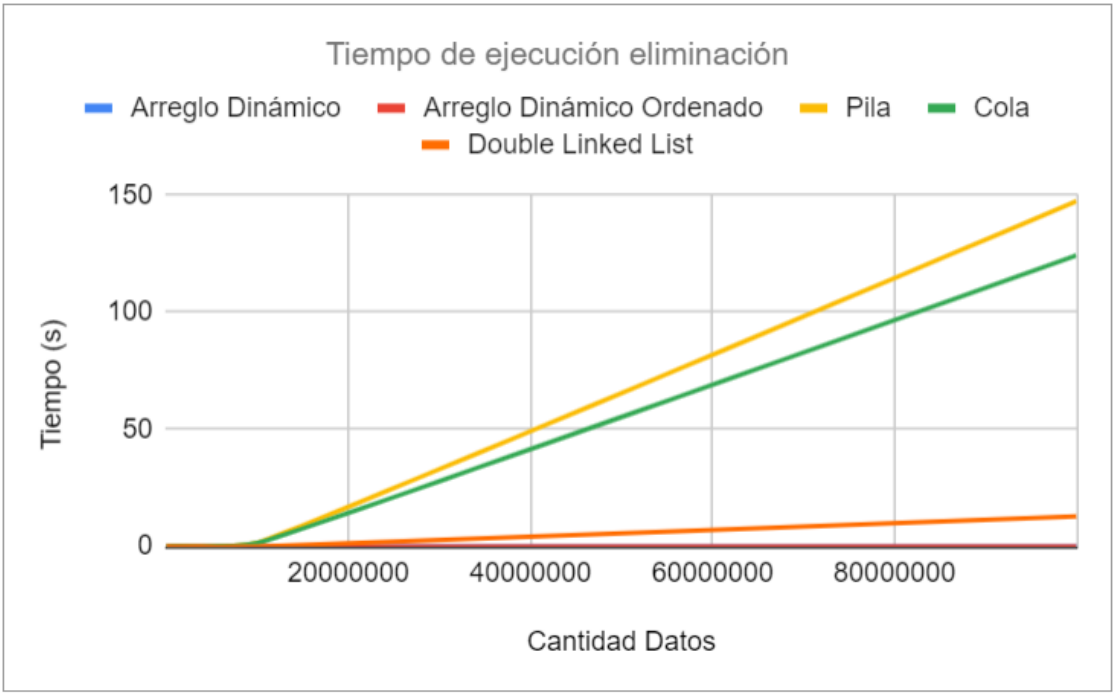

Imagen 2: Análisis de tiempo para eliminar un elemento

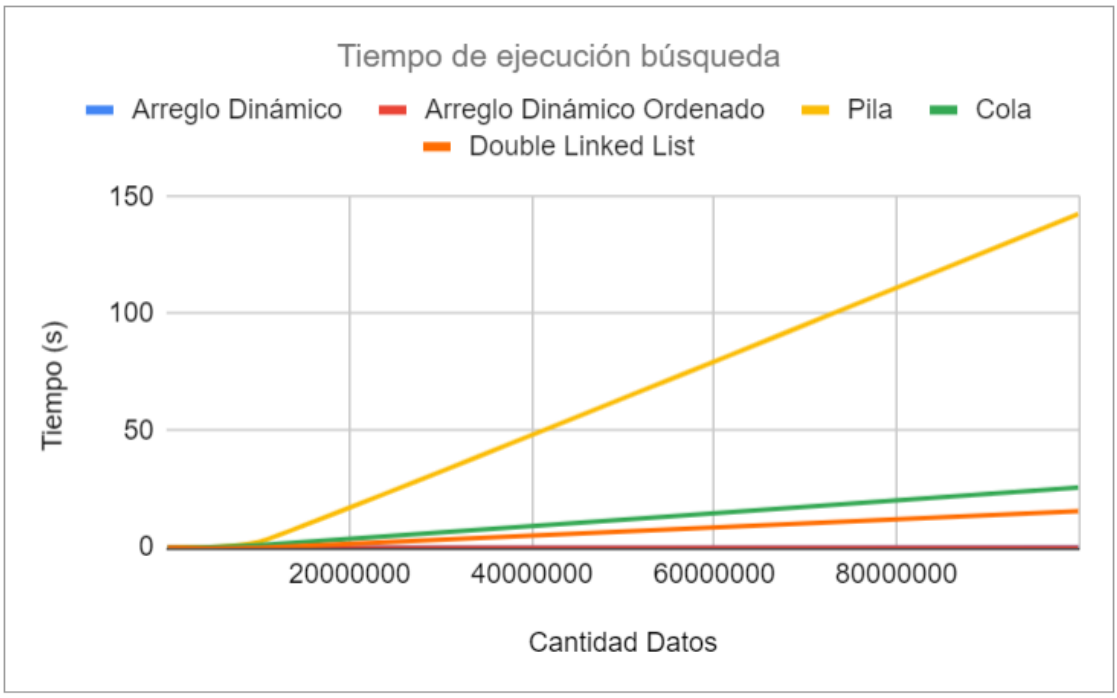

Imagen 3: Análisis de tiempo para buscar un elemento

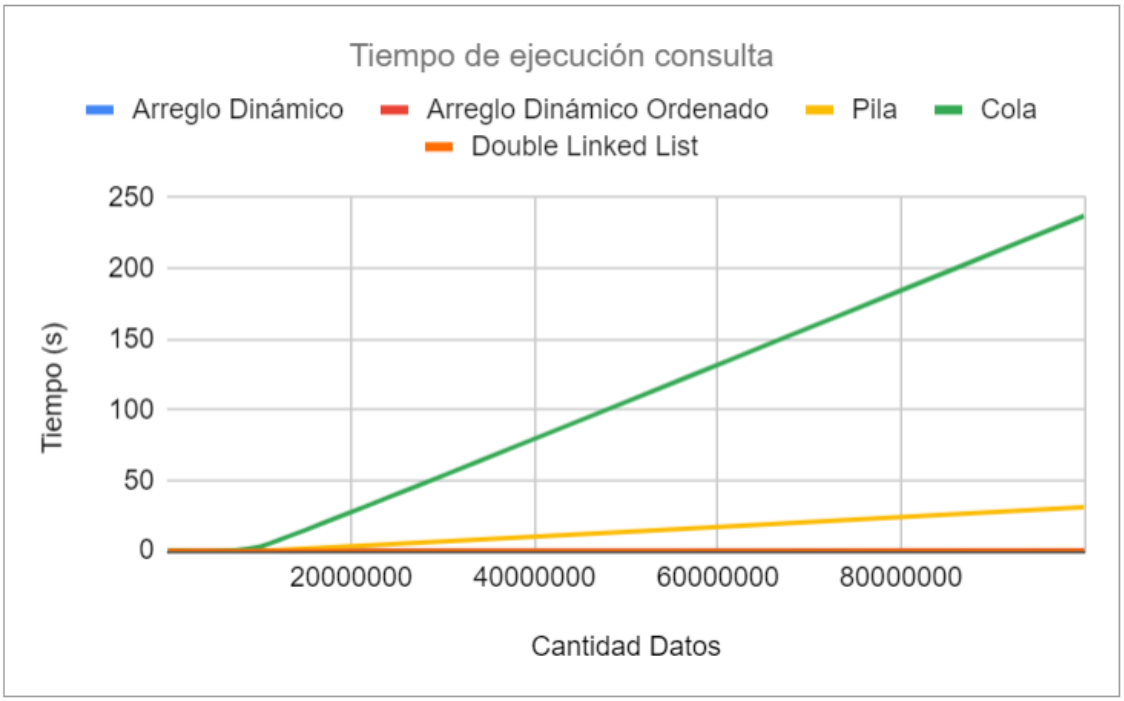

Imagen 4: Análisis de tiempo para consultar los elementos

ANEXO 2: Tablas comparativas de tiempo de ejecución (tiempo en segundos)

Insertar					
Cantidad Datos	Arreglo Dinámico	Arreglo Dinámico Ordenado	Pila	Cola	Double Linked List
10000	0,001383	0,002557	0,001106	0,000816	0,001469
100000	0,00903	0,01426	0,00657	0,005454	0,006919
1000000	0,021739	0,082174	0,079881	0,05527	0,058397
10000000	0,117233	0,87824	1,267902	1,826616	1,549223
100000000	1,073659	8,73023	23,323993	37,675271	21,258013

Buscar					
Cantidad Datos	Arreglo Dinámico	Arreglo Dinámico Ordenado	Pila	Cola	Double Linked List
10000	0,000275	0,000006	0,001923	0,000391	0,000628
100000	0,00034	0,000011	0,015709	0,003117	0,003741
1000000	0,006726	0,00001	0,125942	0,038536	0,007428
10000000	0,010268	0,00001	2,477291	1,222411	0,086481
100000000	0,145288	0,000021	142,753979	25,778637	15,776628

Imagen 5: Análisis de tiempo para insertar n elementos y buscar un elemento

Eliminar					
Cantidad Datos	Arreglo Dinámico	Arreglo Dinámico Ordenado	Pila	Cola	Double Linked List
10000	0,000263	0,000019	0,001665	0,001665	0,000641
100000	0,002189	0,000019	0,020741	0,013271	0,00104
1000000	0,004578	0,000021	0,08546	0,219383	0,004061
10000000	0,009197	0,000018	1,747813	1,560564	0,096294
100000000	0,089397	0,00002	147,519157	124,39674	12,930139

Consultar					
Cantidad Datos	Arreglo Dinámico	Arreglo Dinámico Ordenado	Pila	Cola	Double Linked List
10000	0,000684	0,000502	0,000358	0,001894	0,000004
100000	0,006038	0,006647	0,012361	0,014129	0,000011
1000000	0,031478	0,005417	0,022053	0,051918	0,000005
10000000	0,074472	0,090995	0,174007	3,079845	0,000005
100000000	0,287425	0,265065	31,027193	237,02903	0,000015

Imagen 6: Análisis de tiempo para eliminar un elemento y consultar todos los elementos

ANEXO 3: Gráficas de análisis asintótico

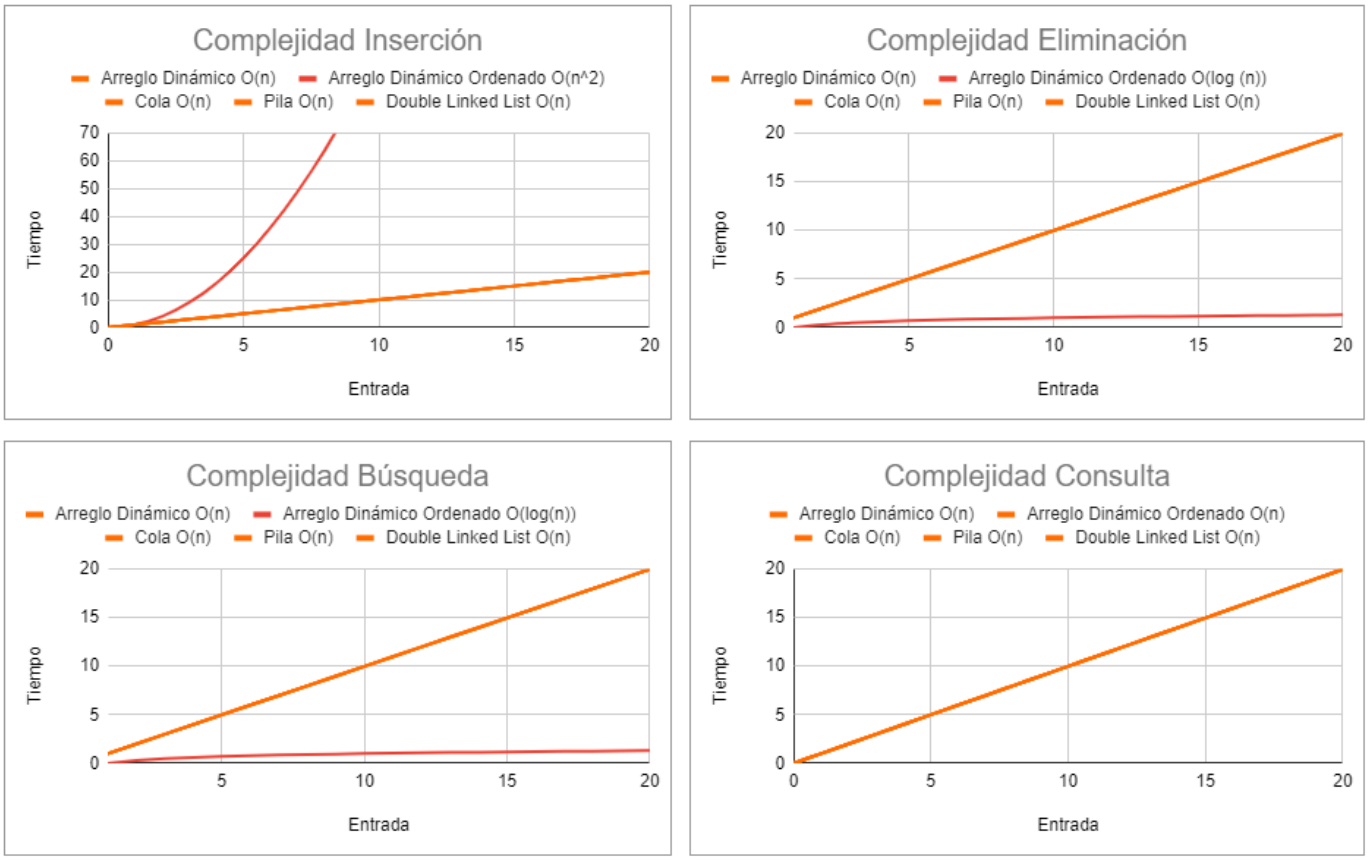

Imagen 7: Análisis asintótico para la inserción, eliminación, búsqueda y consulta de elementos