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

Equipo Trabajo: TuneTrace

I. 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 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:

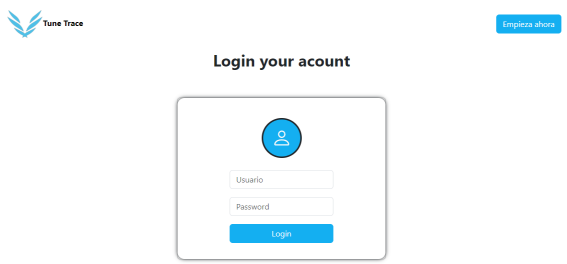
La aplicación almacena la lista de canciones generada durante la búsqueda parcial y la muestra en la sección correspondiente. Al guardar la lista, la aplicación debe considerar si se trata de la creación de una nueva lista en la sección adecuada o la actualización de una lista preexistente en esa sección.

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

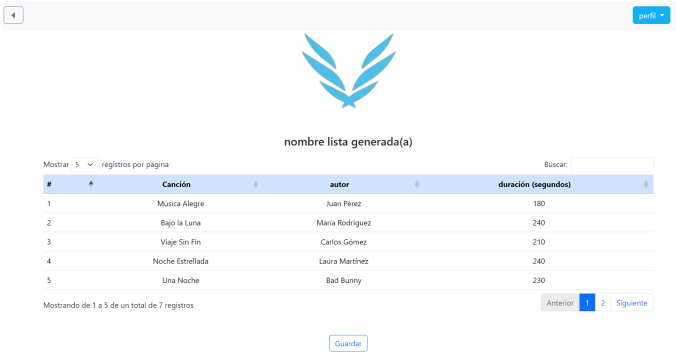
En esta segunda entrega se programó el Frontend usando HTML, CSS, JavaScript y la librería de Bootstrap para algunos estilos en las pantallas. Se hicieron 5 de estas.

Primero se tiene el login donde se tiene principalmente un formulario que solicita un nombre de usuario y una contraseña y por medio de un script en JS válida si se encuentra autorizado, si es así lo redirige al buscador de canciones; si el usuario o la contraseña no son correctos la pagina lanza una alerta que indica que las credenciales ingresadas no son correctas.



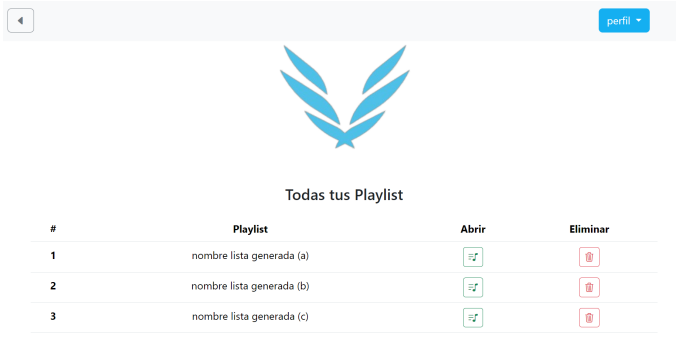
Como ya se mencionó, si se ingresan las credenciales correctas el usuario tiene acceso al buscador de canciones donde se tiene un campo de texto para que se pueda ingresar el nombre de la canción la cual se desea generar una playlist con el botón “generar playlist”. Se tiene también otro botón por el cual se despliega un popup donde se pueden agregar más filtros como lo son género, artista, duración o tema. Este apartado se encuentra en desarrollo. Por último en la esquina superior derecha se encuentran los apartados de “tus playlists” donde se redirige al usuario a una pantalla que contiene las playlist almacenadas; al lado de este se encuentra un botón llamado “tu perfil” donde la única opción habilitada hasta el momento es cerrar la la sesión actual, redirigiendo al usuario al login.

Cuando se genera la playlist pasamos a la siguiente pantalla, donde se ve en una tabla la lista de todas las canciones recomendadas, a su vez se puede buscar en la tabla y editar la longitud de la tabla. Por otro lado, hay un botón para guardar la playlist y que se pueda acceder a esta fácilmente en el apartado de playlist. Como las demás pantallas en la esquina superior derecha se encuentra un botón llamado “tu perfil” donde la única opción habilitada hasta el momento es cerrar la sesión actual, redirigiendo al usuario al login. Aparte de un botón en la esquina superior izquierda para devolverse al

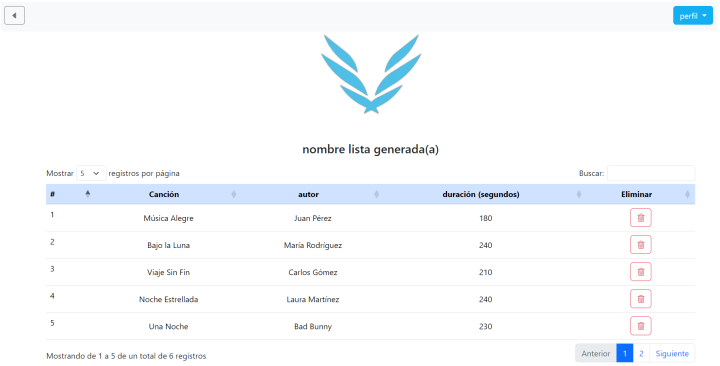


recomendador de canciones y hacer otras peticiones.

La siguiente pantalla, es el apartado donde se muestran todas las playlist guardadas por el usuario, las cuales genera la aplicación. Se puede acceder a este apartado desde el recomendador en el botón de "tus playlist". En esta pantalla se tienen dos botones, uno para eliminar la playlist de las lista de guardados (icono de bote de basura y color rojo), por otro lado, un botón para poder acceder a la lista de canciones de cada playlist. Se tiene previsto que para la siguiente entrega este botón aparte de cumplir la funcionalidad antes mencionada, también redireccione a la misma playlist pero en la plataforma de Spotify para facilitarle al usuario la reproducción de la misma. Como las demás pantallas en la esquina superior derecha se encuentra un botón llamado "tu perfil" donde la única opción habilitada hasta el momento es cerrar la sesión actual, redirigiendo al usuario al login. Aparte de un botón en la esquina superior izquierda para devolverse al recomendador de canciones y hacer otras peticiones.



La siguiente pantalla y la última de nuestro proyecto, es la lista de canciones que contienen las playlist que están en el apartado de guardado, Se accede a esta pantalla desde el menú de playlist guardas. Por otro lado, hay un botón con el que se puede eliminar la canción correspondiente a la fila en el que está en botón (icono de bote de basura y color rojo).Como las demás pantallas en la esquina superior derecha se encuentra un botón llamado "tu perfil" donde la única opción habilitada hasta el momento es cerrar la sesión actual, redirigiendo al usuario al login. Aparte de un botón en la esquina superior izquierda para devolverse al menú de playlist guardadas y hacer otras peticiones.



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, especialmente para esta entrega, su uso es fundamental para la implementación de estructuras como heaps y conjuntos disjuntos, ya que son árboles completos representados en arreglos pero, como no sabemos la cantidad de canciones a filtrar, es necesario un arreglo dinámico.

Árboles de búsqueda binarios y árboles AVL: Se implementan en ambos casos con el fin de mantener ordenadas las clasificaciones que serán usadas por el algoritmo de recomendación. En este caso, cada árbol representa una posibilidad según el parámetro a filtrar.

Lista doblemente enlazada: Dentro del proyecto su implementación tomará las diferentes posibilidades según el

parámetro a filtrar, en este caso, cada nodo serán árboles o heaps para almacenar las canciones que cumplan con ese atributo.

Heaps: En esta entrega, se implementó un min-heap, y al igual que los árboles mantiene las canciones que cumplen con el atributo del parámetro dado. De esta manera, se almacena ese conjunto de datos teniendo la canción con identificador menor en la parte superior.

Conjuntos disjuntos: En este caso, los conjuntos disjuntos ayudan a representar las relaciones entre canciones. Del mismo modo que las listas enlazadas con ayuda de los árboles, pero en este caso, con una sola estructura de datos. A diferencia de los árboles o los heaps, no almacena los datos con un orden específico, sin embargo, funciona para la clasificación de las canciones ya que podemos generar conjunto de datos a partir de un parámetro (filtrar datos).

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 dos funcionalidades que tienen un enfoque en filtrar las canciones, lo cual será fundamental para el algoritmo de recomendación el cual es nuestro producto final, dentro de las cuales encontramos: Inserción y clasificación de n datos y consulta de canciones dado un parámetro (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:

m: cantidad de categorías
n: cantidad de datos

- Inserción y clasificación de n elementos:

Estructura de datos	Complejidad Big-O
BST	$O(m*n^2)$
AVL	O(m*n*log(n))
Min-Heap	O(m*n*log(n))
Conjunto disjunto	O(n)

- Consulta de canciones dado un parámetro:

Estructura de datos	Complejidad Big-O
BST	O(m*n)
AVL	O(m*n)
Min-Heap	O(m*n)
Conjunto disjunto	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 coord 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
	Técnico	Implementación de las estructuras para las funcionalidades de la entrega.
	Observador	Colaborador en el documento escrito
Juan Esteban Cardenas	Experto	Implementación del proyecto
		Toma de tiempos y creación de tablas para muestra.
	Investigador	Consulta de APIs funcionales para la obtención de datos reales.
	Técnico	Creación del repositorio y mantenimiento del mismo.
		Desarrollo del análisis asintótico comparativo entre las funcionalidades implementadas.
		Implementación de las estructuras para las funcionalidades de la entrega.
Juan Esteban Peña	Técnico	Implementación del proyecto
		Colaboración en la creación de los Mockups
	Investigador	Consulta de la conexión con la API de Spotify
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, 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, es importante considerar estructuras que sean escalables, tal como los árboles. Sin embargo, un problema que no se tuvo en cuenta inicialmente fue la repetición de datos.

Adicionalmente, también se encontró que de las diferentes estructuras de datos que se han presentado dentro del programa, los árboles son útiles para el producto final de un algoritmo de recomendación pero no es suficiente, dado que aún hay un problema con la consulta de datos en una base de datos amplia.

Una de las dificultades que tuvimos fue generar un planteamiento adecuado para poder comparar las estructuras entre sí, puesto que a pesar de ser similares, en cuánto a

representar árboles conceptualmente, pueden implementarse para funcionalidades completamente distintas, que si bien pueden ser útiles dentro del proyecto, no son completamente comparables entre sí.

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

inserción y clasificacion (genero)				
Datos	BST	AVL	Heap	Disjoin Set
10000	0,070034	0,067763	0,041695	0,036106
100000	0,378273	0,398341	0,166906	0,143879
1000000	3,375856	3,592634	0,932104	1,003683
10000000	43,743937	59,214729	14,779861	18,835939
20000000	112,370272	117,952421	29,751699	66,015437

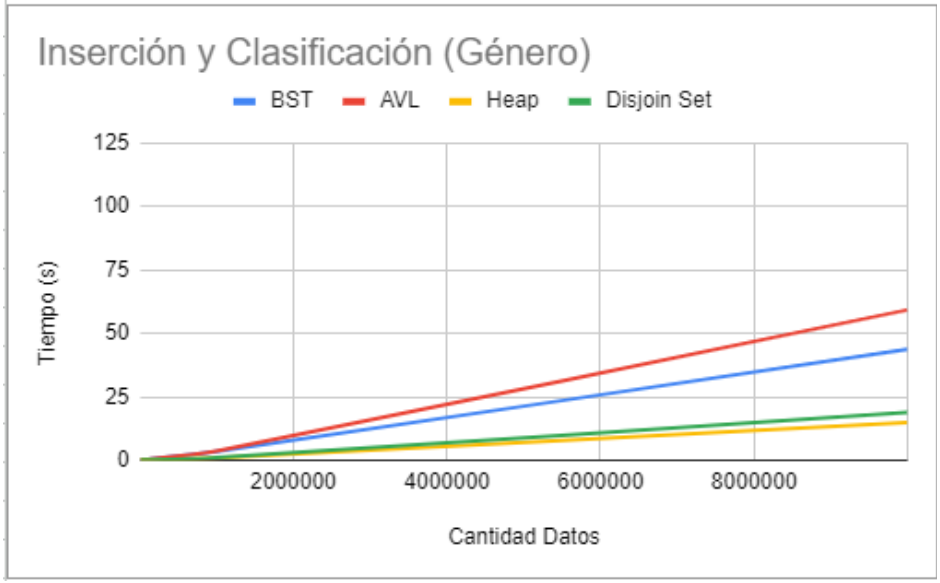


Imagen 1: Análisis de tiempo para insertar y clasificar por género

consulta por genero				
Datos	BST	AVL	Heap	Disjoin Set
10000	0,000282	0,000198	0,000898	0,000149
100000	0,001327	0,000580	0,003504	0,006563
1000000	0,004692	0,002460	0,037395	0,015756
10000000	0,238761	0,024829	1,091465	0,110069
20000000	0,161500	0,889421	2,042035	0,649700

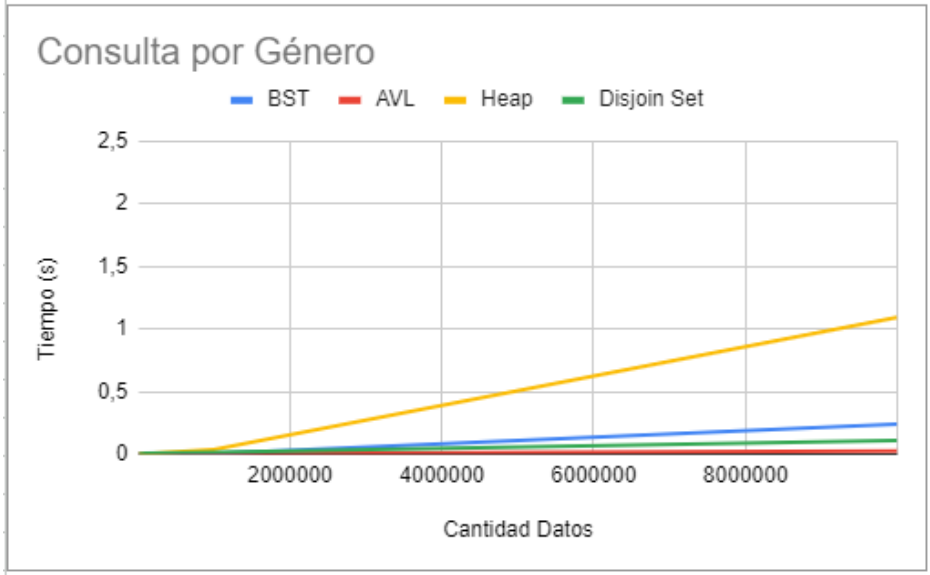


Imagen 2: Análisis de tiempo para consultar por género

insercion y clasificacion (autor)				
Datos	BST	AVL	Heap	Disjoin Set
10000	0,070332	0,060690	0,041340	0,037686
100000	0,427631	0,448347	0,195352	0,146145
1000000	3,630667	3,497477	1,010616	1,087970
10000000	55,161048	44,479399	14,873677	19,350435
20000000	130,218601	121,405864	32,140036s	81,016400

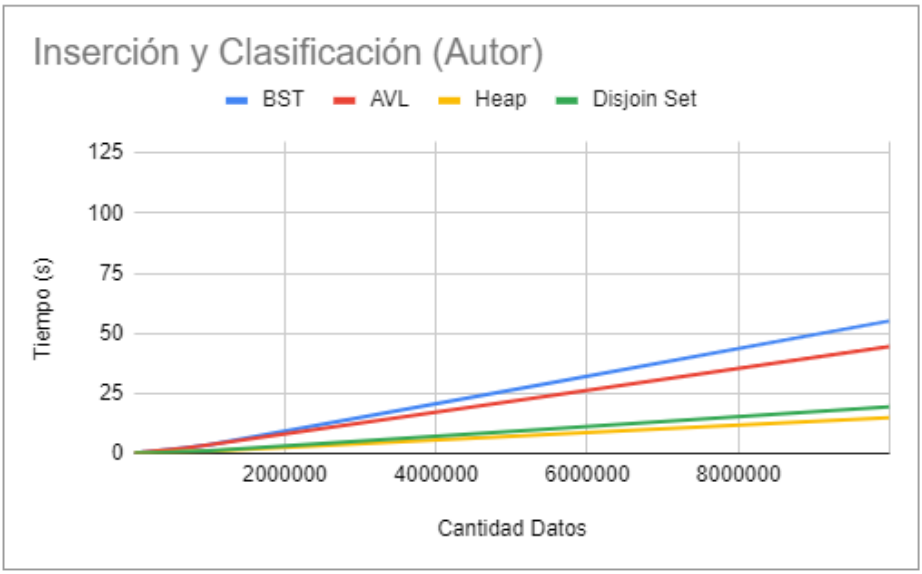


Imagen 3: Análisis de tiempo para insertar y clasificar por autor

consulta por autor				
Datos	BST	AVL	Heap	Disjoin Set
10000	0,000103	0,000230	0,000383	0,000692
100000	0,000407	0,000959	0,003282	0,003478
1000000	0,002774	0,006755	0,012244	0,015467
10000000	0,020571	0,176372	0,176528	0,079708
20000000	1,598359	0,144533	0,291655	0,685982

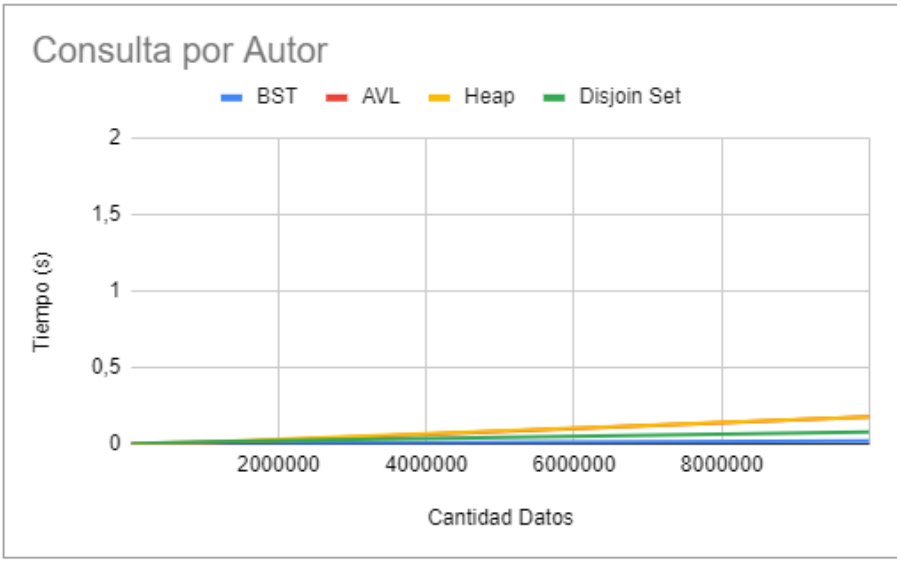


Imagen 4: Análisis de tiempo para consultar por autor

ANEXO 3: Gráficas de análisis asintótico

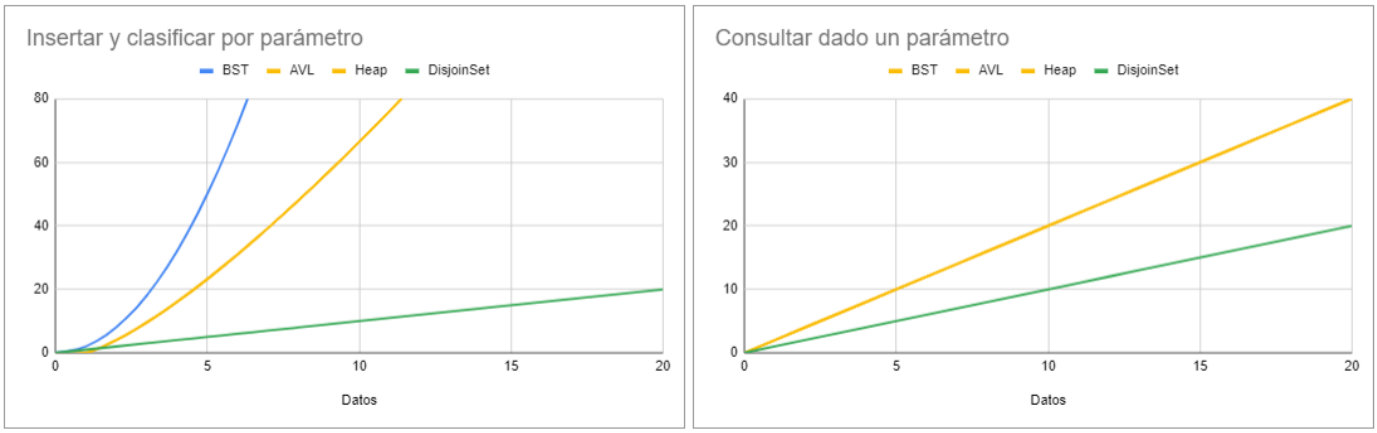


Imagen 7: Análisis asintótico para la inserción, clasificación y consulta por parámetros