Inventory system

John Alejandro Pastor Sandoval, Brallan Ferney Mendoza Delgado, Santiago Bejarano Ariza, Marie Virazaels, Juan David Loaiza Reyes

Estructuras de datos - Grupo 1. Grupo de trabajo 6

I. [1] INTRODUCCIÓN

El objeto del segundo informe del proyecto, es presentar los avances con las funcionalidades que se presentaron en el primer prototipo, optimizar y mejorar sus procesos, mejorar la base de datos que se usa para almacenar toda la información y dar una primera vista sobre la interfaz gráfica que se implementará.

II. DESCRIPCIÓN DEL PROBLEMA A RESOLVER

En el proyecto que propone un sistema de inventario que simula una venta, devolución o solicitud de productos, se encontró que en el primer prototipo las estructuras de datos utilizadas no eran muy eficientes en cuanto al tiempo de respuesta al momento de ejecutar las funcionalidades, por lo tanto el principal problema a resolver en esta entrega es una optimización de todo el sistema así como usar distintas estructuras para mejorar la eficiencia del programa.

III. USUARIOS DEL PRODUCTO DE SOFTWARE

En el software existen dos tipos de usuario, un vendedor y un comprador, ambos usuarios dependen del acceso a un correo electrónico y una contraseña para poder usar el software. Además, en una base de datos se guardarán cada uno de los usuarios con su información. En esta base de datos los usuarios tendrán dos tipos de identificadores, que permitirán definir qué funcionalidades tendrá acceso cada tipo de usuario, por ejemplo, el vendedor tendrá la funcionalidad de crear, borrar, editar y leer productos, mientras que el comprador tendrá acceso solo a las funcionalidades de: comprar, devolver, o pedir un producto específico (Funciones que un vendedor también puede realizar).

IV. REQUERIMIENTOS FUNCIONALES DEL SOFTWARE

Funciones Habilitadas para ambos usuarios:

Autenticación de Usuarios:

La autenticación de usuarios da acceso a estos mismos al software dependiendo si se encuentran en la base de datos, además también permite conocer el rol de cada usuario y de esta manera permitirles el acceso a las distintas funcionalidades.

El software le pedirá al usuario que digite su correo electrónico y contraseña, el software realizará una verificación y permitirá el acceso al usuario al programa y a las respectivas funciones. En caso de que el usuario no se haya registrado en la base de datos puede crear un nuevo usuario para acceder al programa.

Compra de Productos:
La compra de productos permitirá al usuario ver los productos disponibles en el inventario y podrá comprar artículos. El software le preguntará al usuario por el producto que desea comprar y la cantidad y dependiendo del stock que hay determinará si es posible la venta o si toca comprar una cantidad menor.

Pedir producto:
Si el usuario no encuentra el producto que desea en el inventario, tiene la posibilidad de pedir este producto por encargo, de tal manera que se incluya en el inventario para su respectiva venta. El software le permitirá al usuario digitar el producto que desea pedir y la cantidad que este desea. En este prototipo el pedir producto crea un item con precio nulo, de manera que se busca que el vendedor después de pedir el producto le establezca el precio pertinente.

Devolver producto:
En caso de que el usuario desee devolver un producto, puede hacerlo y el producto se devolverá al inventario con su respectiva cantidad. El programa preguntará al usuario el producto que desea devolver y hará el respectivo proceso para devolver el producto al inventario.

Filtrado de productos
En caso de querer, el usuario puede filtrar los productos según su stock y precio principalmente, y buscar también por nombre. Esto devolverá en pantalla los datos que satisfacen el tipo de filtrado, por ejemplo con un precio menor a 10, devolverá todos los productos con un precio menor que 10. Funciona igual con stock.

Funcionalidades habilitadas solo para el Usuario con rol de vendedor.

Crear producto:
Esta funcionalidad permitirá al vendedor agregar un nuevo producto al sistema de inventario con sus respectivas propiedades. El software le pedirá al usuario el nombre, precio y stock inicial del producto para agregarlo al inventario.

Borrar producto:
Esta funcionalidad permitirá que el vendedor pueda borrar un producto del sistema de inventario. El software sólo pedirá el id del producto y lo ubicará para respectivamente eliminar el producto del inventario

Editar producto:
Si el usuario desea cambiar las propiedades de un producto el software le pedirá el id del producto para después actualizar los datos que el usuario desee.

[1]

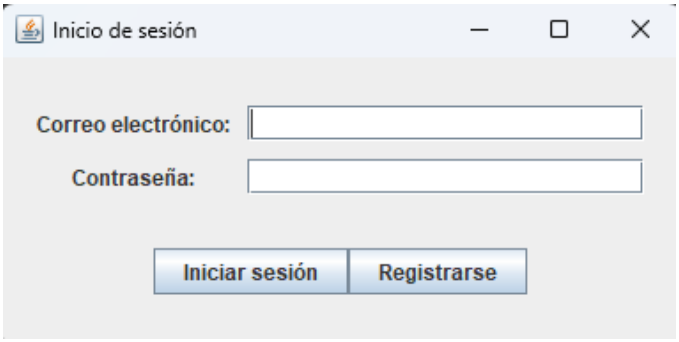
Leer producto:
En caso de que el usuario solo desee verificar la existencia del producto o quiera conocer la existencia de este, puede acceder esta funcionalidad donde puede dar el nombre del producto y el software le dirá si el producto se encuentra o no en el inventario.

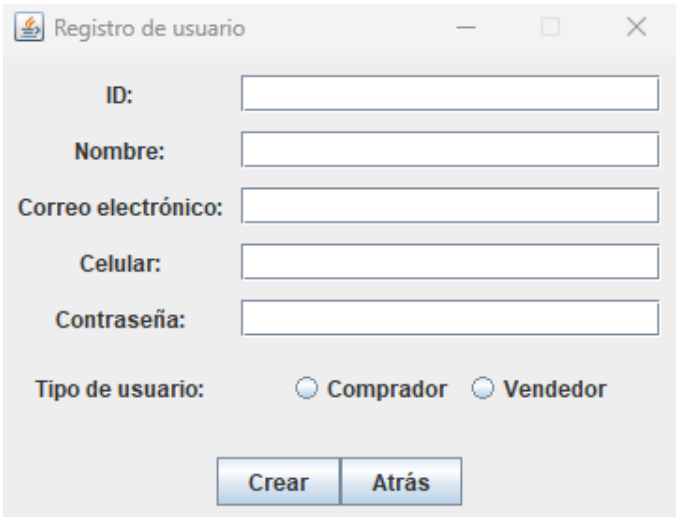
V. AVANCE EN LA IMPLEMENTACIÓN DE LA INTERFAZ DE USUARIO

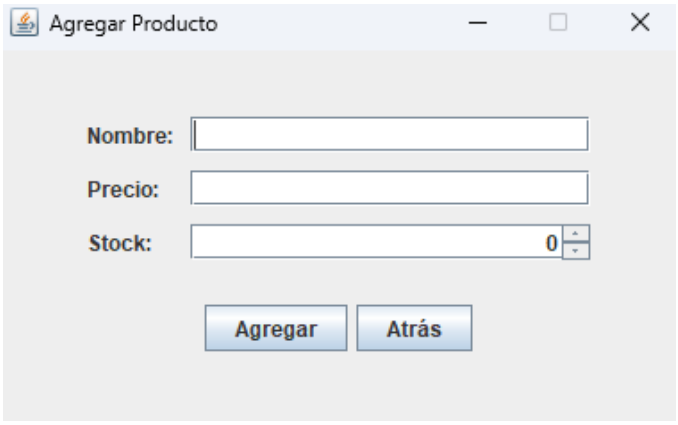
La interfaz gráfica del sistema cuenta con siete interfaces divididas por las diferentes funciones que realiza la aplicación.

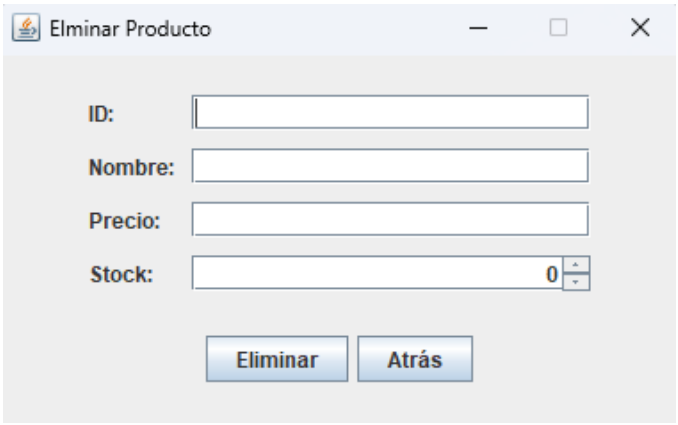
Cuenta con una interfaz para login, una para sign up, update products, add products y delete produtcs. Además cuenta con una interfaz para el vendedor que va a ser quien administre el sistema y una para el usuario común que quiera buscar información en el sistema.

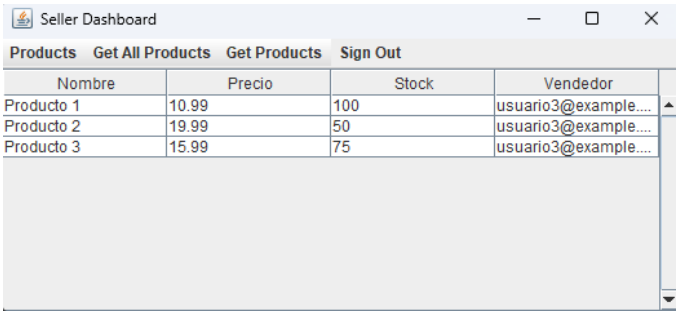
Cabe aclarar que, la interfaz al aún seguir en desarrollo se encuentra en la rama “brallan” mas no en main.

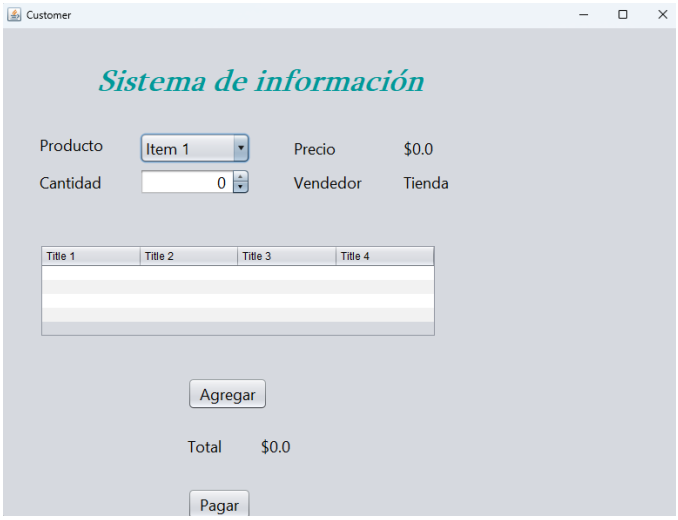












VI. ENTORNOS DE DESARROLLO Y DE OPERACIÓN

En el entorno de desarrollo se usó el lenguaje de programación Java, en este lenguaje usaron principalmente las librerías: Java Utility Library, Java Database Connectivity API (JDBC) para el manejo de las bases de datos y de estructuras de datos secuenciales. Como herramientas de desarrollo se utilizó el editor de código Visual Studio Code, en este editor trabajamos los 6 integrantes del proyecto donde cada uno se enfocó en desarrollar funcionalidades específicas para el funcionamiento del software. También usamos herramientas como el cliente MySQL para el manejo de la base de datos y el repositorio online GitHub para facilitar el manejo compartido del código con el grupo de trabajo.

En cuanto a especificaciones que necesitará un dispositivo para usar el software. El software está hecho para ser usado en los sistemas operativos: Windows 8 x64, Ubuntu Linux 18.04 LTS x64 y OS X 10.11+ x64, como requisitos mínimos para que el software funcione correctamente se recomienda el uso de un dispositivo que tenga un procesador de mínimo 1GHZ puede ser Ryzen o Intel de quinta generación por ejemplo, en cuanto a memoria Ram lo recomendado es que sea de 1 GB mínimo y 4 GB de espacio libre en el disco duro, también es necesario tener instalado tanto el JDK como MySQL.

VII. DESCRIPCIÓN DEL PROTOTIPO DE SOFTWARE

En esta segunda versión del prototipo se han mejorado algunas funcionalidades, optimizando las y cambiando algunas estructuras para una rápida ejecución, también los archivos poseen un código más limpio y eliminando funcionalidades o condiciones innecesarias. Además de agregar nuevas clases y mejorar la base de datos donde se maneja toda la información

Cabe aclarar que el archivo dentro de dist apunta a una base de datos con el usuario y contraseña de Juan David Loaiza Reyes esto porque fue quien compilo el proyecto. El código fuente es encontrado

https://github.com/juanloaiza21/dtastr-inventory-system

VIII. DISEÑO, IMPLEMENTACIÓN Y APLICACIÓN DE LAS ESTRUCTURAS DE DATOS

Class Item: En esta clase se implementan métodos para obtener todos los productos, donde usando un método para obtener todos los productos de la base de datos los agrega a una lista enlazada para después implementar otro método que al recibir la lista donde fueron almacenados los productos y un id, donde devolverá el producto si se encuentra en la lista, si no devolverá un valor nulo. a diferencia de el primer prototipo esta clase funciona más como una clase para usar metodos de obtencion y cambio de items.

Class ItemA: la clase permite determinar las características que tendrá cada producto, es decir el id, el nombre, el precio y la cantidad existente.

Class ItemFilter: Usa arboles binarios para hacer un filtrado de los ítems, permitiendo una busqueda mas rapida y eficaz de los ítems, es una primera implementación y falta una mayor optimización pero por su estructura un árbol binario será una solución bastante eficaz para mejorar el tiempo de busqueda de items.Además permite filtrar los ítems por nombre, precios y stock haciéndola más versátil.

Clase Users: Esta clase funciona como una interfaz para métodos que se implementaran en la clase User, ademas de usar listas enlazadas para almacenar los datos de los usuarios recibiendo como parámetro una lista con uno o varios datos, también implementa métodos para validar el rol y el id de cada usuario para verificar que si existen en la base de datos.

Clase User: La clase User implementa los métodos previamente creados en la clase Users para desarrollarlos y crear las funcionalidades específicas para el funcionamiento del prototipo.

Como primera medida implementa el método login que devuelve en tipo de dato booleano, donde al recibir un email y una contraseña verifique si existe el usuario. Los métodos de update son funciones que usando listas llenan cada espacio determinado por un índice que define el tipo de dato que debe existir, entonces así se actualizan nombres, correos y otros tipos de datos usando listas. También se implementa un método que permite encriptar las contraseñas haciendo cambios entre sistemas hexadecimales y binarios. Finalmente existe un método para crear usuarios donde usando métodos previamente hechos en la clase Users permite agregar el usuario y los datos necesarios a la base de datos.

Interfaz Query: La interfaz Query crea los métodos sin implementar las funcionalidades que se usarán en la clase Connector.

Class Conector: La clase Conector se encarga principalmente de crear las conexiones del software con la base de datos para obtener los datos necesarios de los usuarios y los productos,además de implementar los métodos necesarios para la manipulación de de productos y usuarios, por ejemplo los métodos de inserción de datos reciben los datos y los campos que se buscan añadir, entonces para esto genera secuencias SQL con los datos que se quieren insertar, de esta manera luego usa métodos creados específicamente para ejecutar las secuencias SQL para alterar la base de datos y generar los cambios.

Los métodos para eliminar uno o todos los datos funcionan de una manera similar al método para insertar, debido a que también genera secuencias SQL específicas que permiten eliminar los datos de la base de datos.

También existen métodos que permiten mostrar en consola los ítems o usuarios de manera ordenada con sus respectivos atributos y valores, donde solo usando operaciones de cadenas de texto y obteniendo los valores actuales de la base de datos los muestra al usuario.

Existen varios métodos “get” que son específicamente para mostrar o devolver los datos de las bases de datos, primero se conectan a la base de datos y preparan instrucciones específicas para que el usuario añada los datos que quiere obtener y así generar la secuencia SQL que muestra los datos que pida.

Finalmente los métodos que permiten usar de manera óptima las secuencias SQL y crearlas de una manera simple se conocen como los métodos query, estos métodos generan nuevas cadenas de texto y dependiendo los datos que el usuario proporcione genera la secuencia que será usada luego para ejecutarla y realizar el proceso que necesite.

Las estructuras de datos que implementa este módulo son las linked list y los arrays, estos proviniendo de la clase de java.util.

Clase Sell Items: Esta clase permite la interacción entre el usuario y el software para realizar la respectiva compra así como los métodos para realizar la actualización de stock de los datos ya fueron implementados en otras clases, se procede a importar estas clases y sus métodos de manera que solo se necesitan entradas por consola para su funcionamiento. Por lo tanto solo se usa un método para vender que depende de una lista enlazada para mostrar todos los productos permitiendo su búsqueda rápida y su actualización dependiendo de la operación que realice el usuario.

Class AskProducts: La idea de pedir un producto se encontraba basada en ordenar un producto y una cantidad por encargo si el usuario no encontraba este mismo en el inventario, primero se usó una cola para almacenar estos pedidos, sin embargo el tiempo de respuesta era bastante lento. Por tal motivo el grupo cambia la cola por una lista enlazada y además creó una nueva tabla en la base de datos donde se almacenará esta información.

Class Refund: Al momento de devolver un producto se agregó una nueva verificación de si el ítem se encuentra o no en el sistema, se maneja una lista enlazada para manejar los ítems que existen y permitir un acceso rápido a estos.

Class Main: esta clase implementa métodos para actualizar stock, borrar items o actualizar precios, además de ser el archivo que permite experimentar cómo funciona el programa llamando las clases mencionadas anteriormente para probar sus funcionalidades.

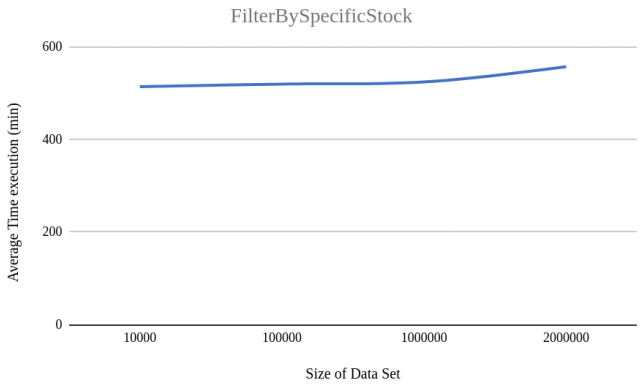
IX. PRUEBAS DEL PROTOTIPO DE SOFTWARE

Para las pruebas se eligen 3 funcionalidades nuevas, ya que las que fueron usadas para el reporte anterior fueron optimizadas. Las funcionalidades que se eligen someter a pruebas con datos masivos son las relativas al filtrado de ítems en árboles. *StockLessThan* se encarga de encontrar los productos con un stock menor que un input e imprimirlos en pantalla para este prototipo. *FilterBySpecificStock* busca encontrar datos con un stock específico como input e imprimirlos en pantalla. *StockGreatherThan* busca el ítem directamente siguiente con un stock mayor que el input. Cabe aclarar que estos métodos tienen otros 2 filtros hermanos, los cuales hacen referencia a precio y nombre, entonces el testeo de estas funciones resulta siendo lo mismo que testear estas 6 funcionalidades, ya que la implementación solo varia en que tienen como key estos filtros.

Las pruebas son hechas con el input de estas, que es un array de n objetos item, donde n representa la cantidad de datos que entran, luego se hacen varias peticiones y se saca el tiempo promedio en nanosegundos (explicado en el archivo anterior). Con este tiempo promedio se hacen gráficas y tablas. Los datos son completamente aleatorios generados para hacer las pruebas al método.

FilterBySpecificStock

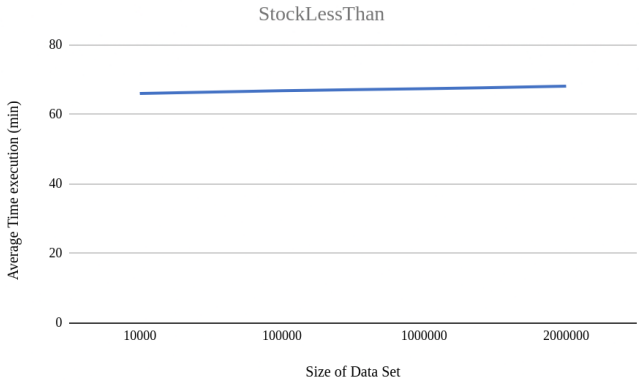
FilterBySpecificStock		
n	Average Time (nanoseconds)	Average Time (minutos)
10000	3083364314915	513,8940525
100000	3117854498549	519,6424164
1000000	3144543369077	524,0905615
2000000	3342436079523	557,0726799



Vemos como la gráfica se asemeja a un O(1) con leves variaciones. En el análisis asintótico de la función se ve que esta es de tipo O(nlogn) pero el hecho de que los datos sean tantos, hace que empiece a verse una tendencia a normalizarse como con un tiempo constante casi.

StockLessThan

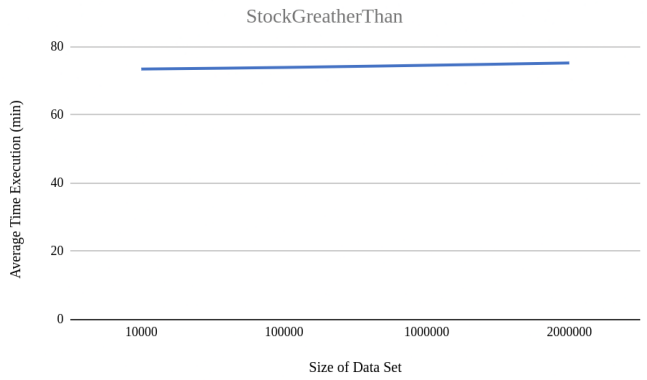
StockLessThan		
n	Average Time (nanoseconds)	Average Time (minutos)
10000	3958786162749	65,97976938
100000	4008522978103	66,8087163
1000000	4041074201741	67,3512367
2000000	4087460459802	68,124341



Nuevamente, la gráfica parece que el método funcionará en tiempo O(1), pero la implementación con treemaps no da un O(nlogn). Nuevamente le atribuimos esta eficiencia y constancia a como funciona una función logarítmica y su tendencia a infinito, como en ciertos tramos se ve estabilizada. Además de que la estructura de datos (treemap) de java es mucho mejor para buscar datos menores que el input a buscar datos mayores que este.

StockGreatherThan

StockGreatherThan		
n	Average Time (nanoseconds)	Average Time (minutos)
10000	4404925246392	73,41542077
100000	4433015842412	73,88359737
1000000	4471928656052	74,53214427
2000000	4511237773173	75,18729622



Vemos un comportamiento similar al anterior método, la gráfica en este caso nos señala un comportamiento más similar a O(logn) el análisis asintótico nos dice que el comportamiento es O(nlogn), y es que este método para datos mayores que n no es tan óptimo con la estructura de datos usada, y eso hace que solo encuentre uno de los datos, para encontrar el n siguiente y asi sucesivamente llega a ser O(n) entonces se ha de buscar una estructura que permita encontrar máximos sin complicación.

X. ACCESO AL REPOSITORIO DE SOFTWARE

https://github.com/juanloaiza21/dtastr-inventory-system

XI. ACCESO AL VIDEO DEMOSTRATIVO DE SOFTWARE

https://drive.google.com/file/d/1ECIItaywQFNSDsM257qWHVf972CcvJ42/view?usp=sharing

El video demostrativo ahonda en las nuevas funcionalidades agregadas, es decir el filtrado de datos.

XII. ROLES Y ACTIVIDADES

INTEGRANTE	ROL(ES)	ACTIVIDADES REALIZADAS (LISTADO)
JUAN LOAIZA	Líder, coordinador	OPTIMIZACIÓN ÍTEMS,PRUEBAS PROTOTIPO
JOHN PASTOR	Animador, técnico	PEDIDO DE PRODUCTOS, CRUD TABLAS
BRALLAN MENDOZA	Técnico, QA	INTERFAZ GRÁFICA, FILTRADO DE ÍTEMS
SANTIAGO BEJARANO	Tecnico	INTERFAZ GRÁFICA, QA
MARIE VIRAZAELS	Técnica, Tester	TESTER, QA

XIII. DIFICULTADES Y LECCIONES APRENDIDAS

Durante el proceso para desarrollar la segunda parte del prototipo se encontró una dificultad al momento de realizar el trabajo en equipo, por distintos motivos que impide un avance apropiado y una entrega responsable en cuanto a las actividades establecidas por el grupo.

En esta versión se mejoró la forma de compartir los archivos por el repositorio GitHub haciendo más ameno el proceso de entender y unir el código de los participantes.

Finalmente la mayor dificultad presentada en el desarrollo fue la optimización de los métodos y funcionalidades del prototipo puesto que tomaba mucho tiempo idear una idea que mejorará la eficacia y rapidez de ejecución.

XIV. REFERENCIAS BIBLIOGRÁFICAS

[1] *MySQL :: MySQL Workbench Manual :: 8.1.1 SQL Query Tab*. (s. f.). https://dev.mysql.com/doc/workbench/en/wb-sql-editor-query-panel.html

[2] *Java Iterator*. (s. f.). https://www.w3schools.com/java/java_iterator.asp

[3] *Java Exceptions (Try. . .Catch). (s. f.).* https://www.w3schools.com/java/java_try_catch.asp

[4] GeeksforGeeks. (2023). TreeMap in Java. GeeksforGeeks. https://www.geeksforgeeks.org/treemap-in-java/