VetAnimalData

Juan Andrés Barrera Rodríguez, Sara Valentina Cardona Mejía, Iván Alexander Morales Muñoz, Nelson David Ramírez Marín, Santiago Dleón Sánchez Romero

No. grupo del curso: {*5*} y No. de Equipo de Trabajo: {*4*}

I. INTRODUCCIÓN

La Fundación Huellitas Casanare es una entidad sin ánimo de lucro que realiza labores de rescate, recuperación y reubicación de perros y gatos en estado de vulnerabilidad y abandono, en la ciudad de Yopal y en los demás municipios del departamento de Casanare, así que colaborar con las tareas que desempeñan los funcionarios y contribuir al bienestar/ética de dichos animales de compañía de los que se ocupa la fundación, motivan la ejecución de este proyecto.

Ahora bien, aplicando los conocimientos adquiridos en la asignatura *Estructuras de Datos*, se pretende desarrollar un sistema que permita la digitalización de la información de manera clara y ordenada, que permita acceder a la misma fácil y rápidamente, considerando que dentro de las actividades de la fundación están dar a los animales en adopción además de otros cuidados diarios como medicación y seguimiento.

Este proyecto le provee a la fundación una herramienta práctica, que facilita el manejo de documentación, ahorra gastos, tiempo y esfuerzos a la hora del manejo interno de los datos.

II. DESCRIPCIÓN DEL PROBLEMA

El rescate de animales que se encuentran en estado de abandono, es una actividad que se ha incrementado en los últimos años. Gracias a la labor realizada por las fundaciones que se dedican a los animales, muchos de estos logran ser adoptados por familias que les pueden brindar un hogar. Sin embargo, muchas de estas fundaciones carecen de buenos mecanismos para el almacenamiento y gestión de la información de los animales, generando que muchos de los establecimientos de rescate y adopción tengan inconvenientes a la hora de manejar la información sobre los animales (en este caso perros y gatos) que llegan y que se van del establecimiento, causando que esta actividad se torne poco eficaz.

Por esta razón, se ideó un proyecto con el cual se propone crear y desarrollar una herramienta práctica y eficaz que permita que estas entidades sean capaces de administrar, de una manera más óptima, toda la información que manejan sobre los animales que allí se encuentran.

III. USUARIOS DEL PRODUCTO DE SOFTWARE

La aplicación en primer lugar será usada por la Fundación “Huellitas Casanare” permitiendo a ésta una actualización de sus archivos de manera diaria y más específicamente, será utilizada por las personas encargadas de registrar la información de los nuevos perros y gatos que llegan a la fundación, de los que aún permanecen allí y de los que han sido adoptados. Además, las personas encargadas del cuidado de estos animales también tendrán acceso a la aplicación para estar al tanto del estado de salud de los perros y de los gatos en general. Y por otro lado, se dará prioridad a un grupo de personas con años de experiencia en la fundación para que aprendan a manejar la aplicación y más adelante capaciten a los nuevos integrantes. En adición, de llegar a cumplir muy bien con todas las expectativas, es posible que la aplicación se dé a conocer a otras fundaciones, por ser ésta una referencia a nivel departamental.

IV. REQUERIMIENTOS FUNCIONALES DEL SOFTWARE

1)

Nombre de la funcionalidad: Creación de la hoja de vida de un animal.

Descripción: Permite al usuario digitalizar información asociada a un animal que ingresa a la fundación y almacenarla.

Acciones iniciadoras y comportamiento esperado: El programa le va pidiendo al usuario ingresar dato por dato del animal. Al finalizar se imprimirá un mensaje de creación exitosa o de error si un dato ingresado no corresponde al solicitado.

Requerimientos funcionales: Asegurar que el dato ingresado del animal, corresponda al solicitado. Los datos por cada animal son: nombre, si es perro o gato, género, la fecha de ingreso, el estado de ingreso, si es adoptable y la fecha de salida si ha sido adoptado. En “el estado de ingreso” los posibles valores son: Muy malo, malo, bueno y muy bueno.

La hoja de vida del animal se almacena con serialización.

2)

Nombre de la funcionalidad: Actualización de la hoja de vida de un animal.

Descripción: Permite al usuario actualizar la información asociada a un animal de la fundación.

Acciones iniciadoras y comportamiento esperado: Se realiza la actualización de forma manual de la hoja de vida de un animal. El usuario busca el animal por medio del ID, o por el nombre del animal. Cuando ya tenga certeza de la hoja de vida que quiere actualizar se le pedirá el nuevo dato y a qué campo pertenece. Al final se muestra un mensaje indicando que la actualización se ha completado satisfactoriamente o si ha ocurrido algún problema.

Requerimientos funcionales: Se necesita mostrar al usuario la información asociada al animal del que se quiere actualizar su información. El software comprueba que el nuevo valor corresponde al tipo de dato que requiere ser actualizado. En caso de que no, el software NO debe permitir la modificación.

3)
Nombre de la funcionalidad: Eliminación de la hoja de vida de animales adoptados.

Descripción: Permite de forma manual la eliminación de la hoja de vida de cualquier animal cuando se confirma la adopción, es decir, la salida de éste de la fundación.

Acciones iniciadoras y comportamiento esperado: El usuario registra la adopción y confirma la salida del animal adoptado en la aplicación que da la opción de eliminar la hoja de vida de dicho animal. Al final se muestra un mensaje de confirmación sobre la eliminación de la hoja de vida con el nombre del animal.

Requerimientos funcionales: Confirmar con un sí o un no sobre la adopción de un animal, y además, confirmar de la misma manera si se desea eliminar la hoja de vida con el nombre del animal. Si se colocan entradas diferentes a las expuestas anteriormente el software dará un mensaje de error con la frase "entrada no válida".

4)
Nombre de la funcionalidad: Búsqueda y filtro de las hojas de vida de los animales.

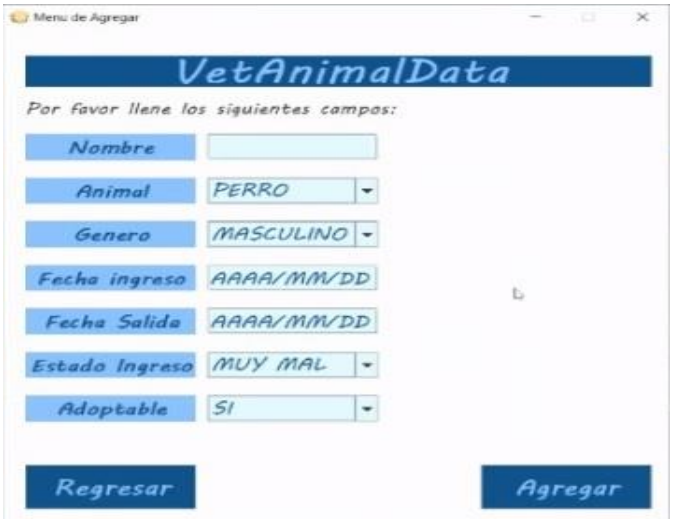
Descripción: Permite al usuario realizar la búsqueda de hojas de vida que cumplan o no parámetros dados por él.

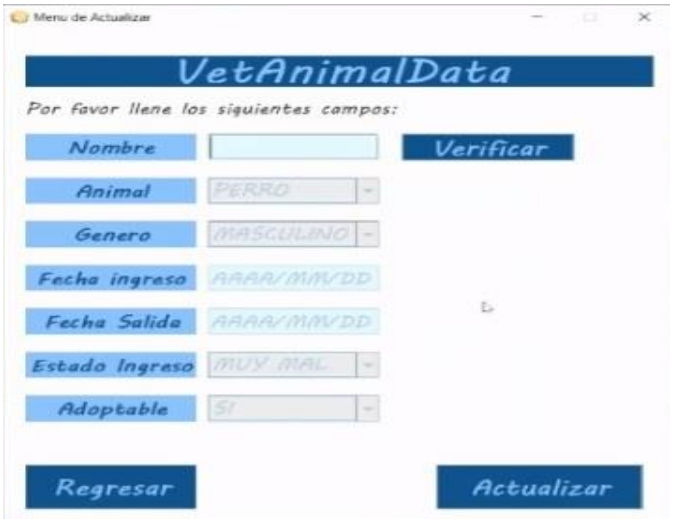
Acciones iniciadoras y comportamiento esperado: Se le brinda la posibilidad al usuario de aplicar filtros a su búsqueda, es decir, si quiere obtener hojas de vida que cumplan cierta condición. El software debe ser capaz de mostrar los resultados solicitados por el usuario.

Requerimientos funcionales: Se necesita preguntar al usuario si desea aplicar filtros. Si es el caso, el usuario podrá aplicar los siguientes filtros: Si es perro o gato y si es adoptable o no.
Los filtros se pueden usar en conjunto, o uno sin el otro. En caso de que el usuario no necesite una búsqueda parcial, simplemente se muestran las hojas de vida de todos los animales.

V. IMPLEMENTACIÓN DE LA INTERFAZ DE USUARIO

A partir de los mock-ups realizados en la versión preliminar, se decidió elaborar la interfaz de usuario del software, en dicha interfaz se implementaron, inicialmente el logo del proyecto y en el menú principal, los botones de "historias clínicas", "agregar", "actualizar", "eliminar" y "búsqueda por filtros", que se emplean respectivamente para buscar una historia clínica, agregar una historia clínica, actualizar una historia clínica, eliminar una historia clínica y realizar una búsqueda por filtro de una o varias historias clínicas, considerando los datos requeridos para llevar a cabo cada función. Estos botones están distribuidos en la interfaz desarrollada como se indica a continuación.





VI. ENTORNOS DE DESARROLLO Y DE OPERACIÓN

El lenguaje de programación Java será el que se utilice en este proyecto, que se desarrollará en la IDE de NetBeans ya que es de código abierto (open source) y ofrece un conjunto de módulos para desarrollar aplicaciones con Java. Una de las características más destacadas de este IDE es la de resaltar el código Java sintáctica y semánticamente. Además, cuenta con extensiones para trabajar con otros lenguajes de programación y se puede usar en distintos Sistemas Operativos como Windows, Linux, macOS y Solaris. Para cuando se ponga en operación la aplicación se correrá en un computador que cuente con Windows 10 e Intel 10th Gen de 12 Gb con ddr4 ram.

VII. DESCRIPCIÓN DEL PROTOTIPO DE SOFTWARE

Para dar continuidad a los lineamientos del proyecto, se realizó un prototipo funcional del software, el cual debía cumplir con algunos de los requisitos propuestos al inicio del proyecto. Para realizar esta actividad se hizo uso de la herramienta Github, con la cual se creó un repositorio para tener un fácil acceso al proyecto, que está disponible en el siguiente link:

https://github.com/Jbarreraro/ProyectoEEDD

El uso de esta herramienta facilitó la creación y el desarrollo de la aplicación al permitir la colaboración de los miembros del grupo de una manera unificada y sencilla.

El prototipo final de la aplicación, consiste en una interfaz que le permite al usuario interactuar de diferentes maneras con los datos que son almacenados en las estructuras de datos, en ellas es posible agregar, buscar, eliminar y mostrar los datos de la estructura. Para realizar la función de almacenar datos, se usaron árboles, heap y map. Su funcionalidad será explicada detalladamente más adelante.

VIII. DESCRIPCIÓN GENERAL DEL PROTOTIPO

A partir de los parámetros y funcionalidades establecidas para el prototipo desde el inicio del proyecto, fue posible el desarrollo de la aplicación. La aplicación “Vet Animal Data” permite la creación y el almacenamiento de registros que contienen la información que necesita la fundación acerca de cada animal, estos registros contienen datos específicos de cada animal tales cómo nombre, estado en el que se encontraba al momento en el que ingresó a la fundación, la fecha de ingreso y la fecha de salida si es que el animal ya fue adoptado.
Para que todos estos registros puedan cumplir su función de optimizar el manejo de los datos de los animales, se debieron añadir ciertas funcionalidades más específicas, como la adición de un nuevo registro, la eliminación o actualización de un determinado registro, la búsqueda y el filtro de los registros y la búsqueda por rango. Todas estas funciones fueron realizadas mediante el uso de diferentes estructuras de datos, como arrays dinámicos, árboles binarios de búsqueda y “heaps”.

IX. IMPLEMENTACIÓN Y APLICACIÓN DE LAS ESTRUCTURAS DE DATOS

Para el desarrollo del proyecto se realizó un análisis de las funcionalidades que éste debía tener para cumplir con los objetivos propuestos, a partir de esto se concluye que las estructuras de datos que cumplieran mejor estos requerimientos serán implementadas en el prototipo final del proyecto. Tales estructuras fueron árboles (AVL y BST), Heap (Max-Heap y Min-Heap) y Map (Tablas Hash). Estas estructuras se emplean para almacenar registros de los animales que han ingresado a la fundación y cuya información comprende datos específicos del

ingreso del animal como fecha de ingreso, fecha de salida, género, estado de ingreso y condición de adopción.

En cuanto a los árboles BST y AVL, estas estructuras fueron implementadas con el propósito de lograr una búsqueda por filtro y rango de cualquier animal o grupo de animales de interés para el usuario; por otro lado, el Max-Heap fue implementado con el fin de tener la posibilidad de acceder a la hoja de vida de un animal que se busca por la prioridad de una característica específica como podría ser la fecha de ingreso del animal al refugio y, por último, el HashMap se implementó con la finalidad de acceder a la información de la hoja de vida de un animal de acuerdo al nombre del mismo.

Para la implementación del árbol BST se utilizó la implementación previamente desarrollada de arreglos dinámicos y, a grandes rasgos, se usaron métodos que tenían las siguientes características:

1. Retornar el valor máximo almacenado en el árbol.
2. Retornar la referencia del nodo que contiene el valor del key.
3. Regresar los nodos que contienen keys con valores.
4. Insertar un nodo en el árbol.
5. Eliminar un nodo del árbol y mantenerlo invariante.

Para la implementación del árbol AVL, en términos generales, se emplearon métodos que tenían las siguientes características:

1. Rebalancear el árbol.
2. Mantener el árbol balanceado.

Para la implementación del Heap se recurrió a la implementación previamente desarrollada de arreglos dinámicos y, a grosso modo, se usaron métodos que tenían las siguientes características:

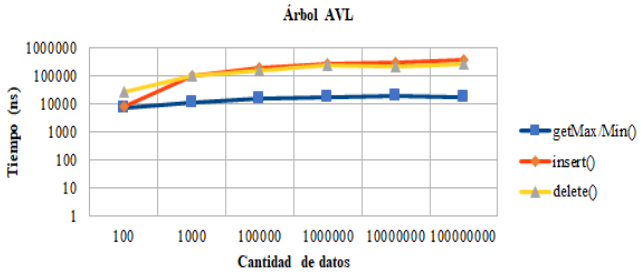
1. Aceptar un valor booleano que decida si se construye un max-heap o un min-heap.
2. Retornar el valor almacenado en la raíz del árbol.
3. Extraer el valor almacenado en la raíz del árbol.
4. Insertar un elemento en el heap.
5. Eliminar un elemento del heap.

Para la implementación del HashMap se empleó la implementación previamente desarrollada de listas enlazadas y, básicamente, se usaron métodos que tenían las siguientes características:

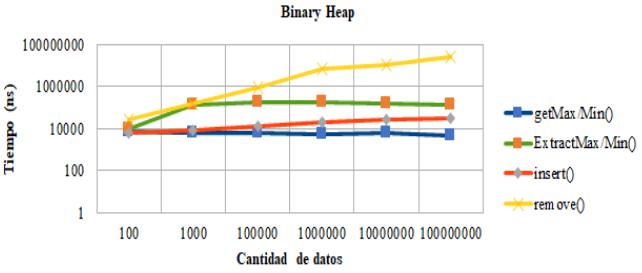
1. Hash para enteros.
2. Hash polinomial para cadenas String.
3. Añadir n elementos a cada hashmap.
4. Obtener el índice dentro del hashmap.
5. Eliminar un elemento del hashmap dada una llave.
6. Retornar un booleano indicando si el hashmap contiene el key del parámetro.

X. PRUEBAS DEL PROTOTIPO Y ANÁLISIS COMPARATIVO

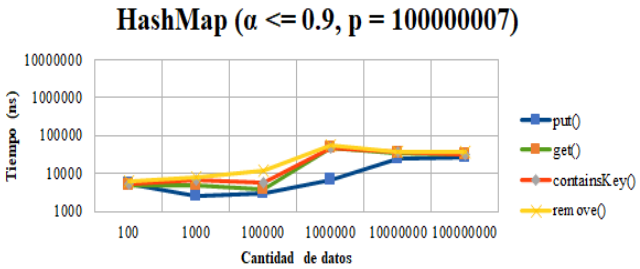
Árbol AVL			
	O(log(n))	O(log(n))	O(log(n))
n	getMax/Min()	insert()	delete()
100	6800	7600	27600
1000	11400	100700	98700
100000	15000	190600	149500
1000000	16400	266400	237100
10000000	20100	291900	204500
100000000	17000	362300	267200



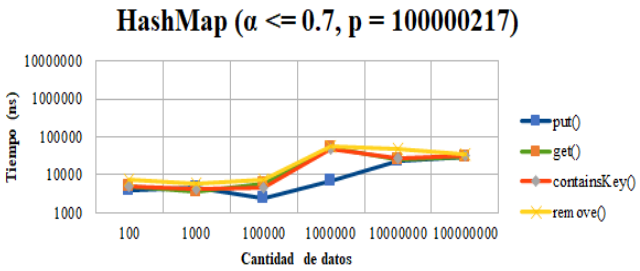
Binary Heap				
	O(1)	O(log(n))	O(log(n))	O(n)
n	getMax/Min()	ExtractMax/Min()	insert()	remove()
100	7100	10302	6400	26800
1000	6100	145500	9000	150700
100000	6400	171800	13500	973500
1000000	5300	175000	21200	7093200
10000000	6100	168000	25789	11483500
100000000	4900	143300	30231	27522200



HashMap (α <= 0.9, p = 100000007)				
	O(1)	O(1)	O(1)	O(1)
n	put()	get()	containsKey()	remove()
100	5200	4800	5300	6100
1000	2500	4700	6800	7800
100000	2900	3800	5600	11700
1000000	6500	48100	47900	55000
10000000	24800	35200	35600	37200
100000000	26500	30800	31600	37600



HashMap (α <= 0.7, p = 100000217)				
	O(1)	O(1)	O(1)	O(1)
n	put()	get()	containsKey()	remove()
100	3900	5000	5200	7500
1000	4600	3500	4400	6100
100000	2500	5800	4600	7800
1000000	6700	51400	48600	55300
10000000	24000	25400	27800	48300
100000000	28900	29900	32900	35600



Según el análisis de la entrega anterior se puede concluir que en las operaciones de búsqueda, eliminación e inserción, los arreglos dinámicos toman un menor tiempo para desarrollarlas en comparación con las listas enlazadas, pero en contraste, las listas enlazadas sólo tienen tiempos similares en la acción de inserción, ya que en las demás difiere con los arreglos dinámicos.

Teniendo en cuenta lo anterior y los nuevos datos obtenidos a partir de las pruebas realizadas a las estructuras árboles, heaps y hash (tablas y gráficas anteriores), se puede observar cómo estos últimos son capaces de realizar cualquier operación en un tiempo constante, es decir O(1), esto significa que son increíblemente eficientes; por otro lado, es posible evidenciar cómo los árboles y los heaps tienen complejidades mayores, a saber O(log(n)) y O(n), acercándose más a los resultados obtenidos por estructuras previamente utilizadas como las listas enlazadas y los arreglos dinámicos.

XI. ACCESO AL VIDEO DEMOSTRATIVO DE SOFTWARE

Se realizó un video demostrativo del prototipo final de software implementado en el desarrollo de la aplicación, disponible en el siguiente enlace:

https://youtu.be/d1H5jk4_ESY

XII. ROLES Y ACTIVIDADES

Para el cumplimiento de los requerimientos establecidos inicialmente fue necesario asignar roles dentro del equipo, para que así cada integrante del mismo pudiera contribuir a partir de sus fortalezas, a la vez que se permitía una constante retroalimentación que ayudaba a la familiarización con nuevos conceptos. Teniendo en cuenta lo anteriormente mencionado, en este equipo, los roles existentes y los integrantes que los desempeñaron, son los siguientes:

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 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.

INTEGRANTE	ROLE(ES)	ACTIVIDADES REALIZADAS
Juan Barrera	Investigador	Documento
	Técnico	Diapositivas
Sara Cardona	Coordinadora	Código
	Observadora	Documento
Iván Morales	Líder	Código
	Experto	Documento

Nelson Ramírez	Investigador	Documento
	Técnico	Código
Santiago Sánchez	Observador	Código
	Investigador	Documento

XIII. DIFICULTADES Y LECCIONES APRENDIDAS

Las principales dificultades encontradas durante el desarrollo del proyecto tuvieron que ver con: los encuentros virtuales y el tiempo dedicado al proyecto, debido a que los integrantes del grupo manejan horarios diferentes; la aplicación de las estructuras empleadas, ya que requerían de bastantes métodos por lo que eran susceptibles a varios errores; las pruebas del prototipo y el análisis comparativo, puesto que la cantidad de datos fue difícil de conseguir, su ejecución tomó bastante tiempo y además, en la entrega pasada el procedimiento utilizado no fue el correcto.

Las principales lecciones aprendidas fueron la importancia del trabajo en equipo y la comunicación respetuosa entre los integrantes, que permitieron un ambiente agradable y el cumplimiento de la entrega. Además, la puntualidad y la responsabilidad fueron factores destacables.

XIV. REFERENCIAS BIBLIOGRÁFICAS

[1] Streib, J & Soma, T.: Guide to *Data Structures A Concise Introduction Using Java*, Springer, 2017.

[2] Universidad de California en San Diego & HSE University: Estructuras de Datos, Coursera, 2021.

[3] ChuWiki: lenguajes de programación, tecnologías y programación web, herramientas y aplicaciones, 2018. (En línea.) Disponible en: http://chuwiki.chuidiang.org/index.php?title=Chuwiki.