Felipe Gutiérrez, Daniel Mahecha, Alejandro Camargo

No. de Equipo Trabajo: 4

Plataforma ECO-LOMBIA

I. [1] INTRODUCCIÓN

El proyecto ECO-LOMBIA se basa en una plataforma colaborativa que alimenta una base de datos sobre la gran riqueza biológica que tiene nuestro país. Destinada para el público en general, pero en especial para los amantes de la biodiversidad en Colombia y los interesados a nivel internacional sobre el tema. En las siguientes secciones se hacen las especificaciones sobre el proyecto.

II. DESCRIPCIÓN DEL PROBLEMA A RESOLVER

Colombia es el segundo país más biodiverso del mundo, cuenta con 56.343 especies [1], sin embargo, el desconocimiento de la mayoría de la población es evidente. Esto tiene diversos factores, pero el principal es el poco acceso a la información. El objetivo de ECO-LOMBIA es facilitar ese acceso a la mayoría de la población colombiana, en especial a quienes cohabitan con esta biodiversidad e ignoran su importancia biológica. También el proyecto busca informar sobre los diversos parques ecológicos que existen en la nación, para fomentar su visita, mejora y preservación.

III. USUARIOS DEL PRODUCTO DE SOFTWARE

Al ser una red colaborativa, sus roles no son tan específicos. Pero los principales tipos de usuarios son:

- Investigadores colaboradores: Son las personas que tienen acceso a las bases de datos, ellos agregan y verifican la información que estará en la plataforma. Son los encargados de agregar especies, filos, características de cada especie o información relativa a eco-parques
- Visitantes del sitio: Personas que tienen acceso a observar la información, más no modificarla, son la población objetivo, por quienes se hace esta plataforma.

IV. REQUERIMIENTOS FUNCIONALES DEL SOFTWARE

- *Inserción de datos*

- Descripción: Es la acción que realiza el usuario registrado al momento de querer añadir información de una especie o ecoparque al sistema, especificando los datos en sus respectivos campos.
- Acciones iniciadas y comportamiento esperado:
  Al llenar los campos, cada valor de dato, corresponderá a un atributo del objeto seleccionado previamente por el usuario y se añadirá al final de una lista enlazada del objeto.

  - Requerimientos funcionales

  - Creación:
    Al llenar los datos y guardar, implícitamente el usuario, crea un objeto y lo añade a la lista enlazada, ya mencionada, todos los datos añadidos serán tomados como un String.

  - *Actualización o eliminación de datos*
    Para la actualización de un dato el usuario deberá indicar el tipo de dato que desea actualizar, ya sea fauna ,flora o ecoparque y así mismo deberá indicar el parámetro de búsqueda que desea actualizar, posteriormente indicará el dato antiguo y el dato nuevo.
    Para la eliminación de dato es un proceso bastante similar el usuario deberá escoger ya se fauna, flora o ecoparque, posteriormente el atributo que desea pasar como búsqueda y luego el dato que desea eliminar.

  - Descripción: Es el proceso mediante el cual el usuario actualiza la información de un elemento ya existente o lo elimina por completo
  - Acciones iniciadas y comportamiento esperado:
    El usuario selecciona un elemento específico, si existe, se mostrará y este podrá modificar su información o eliminarlo, en caso de no existir se le indicará al usuario que el dato insertado no fue encontrado y se volverá al menú secundario ya se de fauna flora o ecoparque.
  - Requerimientos funcionales
    Actualizacion y eliminacion:
    El elemento seleccionado por el usuario le dara la opcion de modificar campos o eliminar el objeto, en caso de no realizar ninguna acción se mantendrá el menú hasta que el usuario pase el dato a eliminar o actualizar, y en caso de ser eliminado, este será eliminado permanentemente.

- *Registro de Usuarios del sistema*

[1]

- *Descripción:* Procedimiento en el cual un usuario no registrado puede solicitar registrarse en la aplicación.

- *Acciones iniciadoras y comportamiento esperado*:

El usuario no registrado accede a la ventana de registro y llena los datos correspondientes a su información personal y profesional para registrarse mediante el aporte de la información anterior y un clic en el botón "Enviar Solicitud", dado que en el caso real una persona verificaría los datos para aprobar el ingreso como usuario registrado en la aplicación pero por ahora es una opción que siempre será True para todos los usuarios y podrán quedar inscritos de manera directa.

- Creación: se requiere que primero al iniciar la aplicación se cree una estructura que almacene a todos los usuarios que estén registrados mediante la lectura de un archivo de texto plano que los contendrá.
- Actualización: Cuando el usuario solicita registrarse el proceso debe ser capaz de añadir al objeto usuario con sus atributos ingresados por él mismo un momento antes, e indicarle mediante un cuadro de diálogo que esto se realizó de manera correcta, tal que si ocurre algún error , de la misma manera se le informe al usuario que debe intentarlo de nuevo o comunicarse con soporte.

Una vez el usuario desee cerrar la aplicación el programa debe almacenar los datos para poder continuar su manipulación en la próxima utilización de la aplicación.

- Eliminación: Una vez creado un usuario este jamás será eliminado por este mismo u otro usuario.
- Consulta total de los datos: No se requiere que en ningún procedimiento de la aplicación que se consulten todos los datos de los usuarios .
- Ordenamiento: se deben ordenar los números de identificación de menor a mayor con el fin de garantizar resultados de búsqueda mucho más eficaces cuando se está creando la estructura al ejecutar la aplicación y antes de que aparezca la interfaz.
- Almacenamiento: Los datos son manejados en tiempo de ejecución por lo tanto pueden cambiar de su versión inicial entonces antes de terminar la ejecución se deben reescribir una vez el usuario solicite cerrar la aplicación de tal manera que facilite la lectura en el orden indicado anteriormente

- *Inicio de sesión de Usuarios del sistema*

- *Descripción:* Procedimiento en el cual un usuario no inicia sesion en la aplicación para acceder a su perfil y realizar operaciones de aporte o edición de sus aportes.

- *Acciones iniciadoras y comportamiento esperado*:

Para iniciar sesión el usuario puede acceder a la ventana de iniciar sesión e ingresar su número de identificación y contraseña para acceder a su perfil mediante el clic una vez finalizado el ingreso de datos. el software debe buscar por el número de identificación al usuario que se desea consultar para obtener los datos y compararlos con los ingresados para lograr el procedimiento.

- *Cambio de contraseña de Usuarios del sistema*

- *Descripción:* Procedimiento en el cual un usuario cambia su contraseña con una actual ya sea temporal o la suya propia .

- *Acciones iniciadoras y comportamiento esperado*:

Una vez en su perfil el usuario puede cambiar su contraseña si lo desea mediante un clic en un enunciado resatado y una ventana emergente se abre para que con su contraseña actual podrá ingresar una nueva.

- Actualización:

En el caso de actualizar la contraseña , una vez el usuario haya ejecutado el cambio debe garantizar mediante un cuadro de diálogo que se realizó de manera correcta y que internamente haya modificado este atributo del objeto usuario.

Si los datos ingresados en el registro no son validos, como por ejemplo que el numero de cedula no sea un entero se le indicará al al usuario mediante un cuadro de diálogo que un dato de los suministrados no es correcto para este lo cambie y pueda completar el procedimiento. Al igual al iniciar sesión puede que el número de identificación ingresado sea mal digitado y no concuerde con ningún usuario o digite incorrectamente la contraseña para lo cual el software mediante un cuadro de diálogo le indicará que el usuario con ese ID no está registrado o que la contraseña es incorrecta respectivamente indicando que intente de nuevo llenar los campos y solicitar el acceso haciendo clic en el botón correspondiente.

- *Paginación de resultados de búsqueda:*

- *Descripción:* Mostrar un número determinado de resultados en el canvas de la aplicación, organizando los datos por página

- *Acciones iniciadoras:* Al realizar una búsqueda, aparecerán los primeros n datos, n es el número de resultados por página y p, el número de datos por página, y al dar "siguiente" aparezcan los resultados del n+1 al resultado 2n, y al dar anterior, se muestren los datos 1 al n.

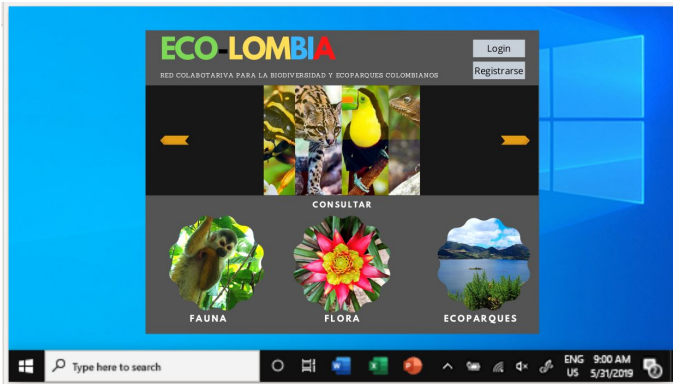
También se puede determinar la variable n, Siendo n=10 el valor por defecto, se puede cambiar a 20, a

50, o al valor que desee el usuario para mayor comodidad.

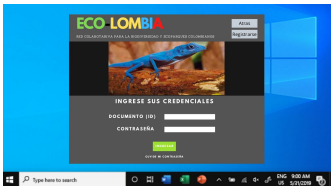
- Almacenamiento: Se necesita una estructura que almacene y ordene los resultados que están dentro de las bases de datos de la aplicación, para que sean impresos según lo indique el usuario.

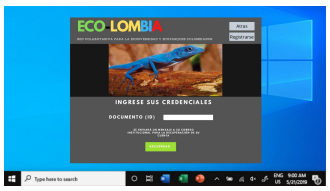
V. DESCRIPCIÓN DE LA INTERFAZ DE USUARIO PRELIMINAR

La aplicación de escritorio está pensada masivamente para usuarios no registrados de tal manera que su principal objetivo es buscar algún ser vivo o ecoparque de la base de datos, de tal manera en el inicio la apariencia es la siguiente.



un inicio donde puede iniciar sesión un usuario registrado o solicitar el registro en la aplicación mediante dos botones. la zona con fondo negro hace referencia a la función de visualización de noticias, y en las imágenes de abajo se accede a la búsqueda.



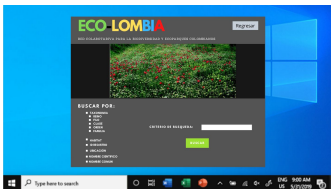


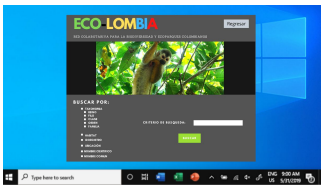
Anteriormente se visualiza las interfaces para el inicio de sesión y en caso de olvidar la contraseña. podrán ser visualizadas como anexo en caso de que desee verlo con más detalle.

La anterior imagen muestra la visualización del perfil de una persona registrada, aquí permite visualizar y editar los aportes realizados por un usuario,y la opción para generar un aporte nuevo.

VI. ENTORNOS DE DESARROLLO Y DE OPERACIÓN

En esta sección se debe describe el entorno en el que se desarrollará el software, así como el entorno en el que operará el software en tiempo de ejecución, específicamente el sistema operativo, recursos y el hardware sobre el que operará el software.





Siguiendo con la descripción de la interfaz en la figura anterior se puede visualizar las interfaces de filtrado y parámetros de búsqueda para la información de el animal, la planta o el ecoparque.

Finalmente la idea es mostrar los datos mediante tablas con una determinada cantidad de elementos para la visualización y un paginador que permite avanzar hacia más resultados.

VII. PROTOTIPO DE SOFTWARE INICIAL

A. Repositorio de github : https://github.com/lfgutierrezg/Ecolombia

B. Link video https://drive.google.com/open?id=12t8YH5I2yapDPEhi9w4K5Ww0Zdgwdk7L

VIII. PRUEBAS DEL PROTOTIPO

- Registro de Usuarios

Para las pruebas de esta funcionalidad se realizó la implementación mediante arreglo dinámico y listas enlazadas los resultados se muestran a continuación en tablas y se realizará un análisis comparativo en la siguiente sección. para calibrar el tiempo se utilizó un dato long para medir los milisegundos antes de iniciar el método y posterior a terminar el método, la diferencia entre estos valores genera el tiempo en milisegundos que tomo la operación. Para más de 3000000 datos en la funcionalidad de Registro de Usuarios se presenta un error de space heap en el ordenador en que se ejecuta por lo cual es el valor máximo de trabajo de datos. Los datos se obtuvieron de un generador de datos online de 10000 datos y posteriormente fueron duplicados hasta el millón de datos en excel donde se le agregó un id que identifica el número de dato que es es decir hay id con valores hasta de 999999.

Registro de usuarios			
DinamicArray			
Número de datos	Tiempo cronometrado [ms]		
	insertado masivo	Insertado de un dato	Actualización
10000	96	0	12
100000	344	0	33
1000000	6626	0	67
3000000	12572	0	102

Registro de usuarios			
DinamicArray			
Número de datos	Tiempo cronometrado [ms]		
	Buscar un dato	Acceder a todos los datos	Almacenar
10000	11	2	80
100000	31	7	418
1000000	61	13	3672
3000000	91	14	8728

Registro de usuarios			
Linked-List			
Número de datos	Tiempo cronometrado [ms]		
	insertado masivo	Insertado de un datos	Actualización
10000	92	0	7
100000	355	0	21
1000000	6543	0	118
3000000	12727	0	215

Registro de usuarios			
Linked-List			
Número de datos	Tiempo cronometrado [ms]		
	Buscar un dato	Acceder a todos los datos	Almacenar
10000	5	3	88
100000	20	12	393
1000000	115	115	3122
3000000	213	239	8646

Linked List Especies y Ecoparques

Numero de datos	Tiempo en ms				
	Insertado masivo	Insertado de un dato	Buscar un dato	Actualizar un dato	Eliminar un dato
10,000	283	0	13	7	7
100,000	2258	0	70	40	47
500,000	7106	0	155	261	246
1,000,000	12069	0	202	203	285

Dynamic Array Especies y Ecoparques

Numero de datos	Tiempo en ms				
	Insertado masivo	Insertado de un dato	Buscar un dato	Actualizar un dato	Eliminar un dato
10,000	157	0	17	6	7
100,000	905	0	66	34	35
500,000	4511	0	134	125	134
1,000,000	8400	0	251	450	250

- Paginación de resultados: se implementó el arreglo dinámico utilizado en la inserción masiva de datos, ya con esta estructura de base, se creó una clase para cada tipo de base de datos, una para la búsqueda de flora, una para búsqueda de fauna y una para búsqueda de ecoparques.

  Al insertar los datos en el arreglo dinámico, se crearon dos variables enteras, estas controlaban el número de datos impresos en la consola y cuales datos se imprimen en pantalla.

  En esta versión del programa se muestran todos los datos registrados en la base y se imprimen en el mismo orden en el que los datos están en el archivo. Para la impresión de los datos se usó un ciclo *for*, donde se compara el número de página con la posición del dato en el arreglo, donde el conector de esta comparación es el número de resultados por página. También existe un contador de datos, el cual viene intrínseco en el arreglo dinámico utilizado para la implementación de la funcionalidad.

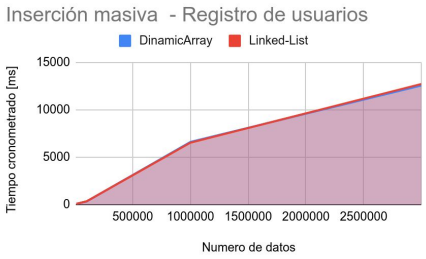
I. ANÁLISIS DE COMPARATIVO

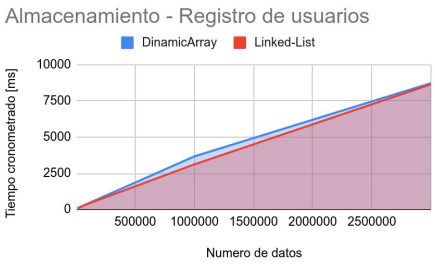
Realice una tabla comparativa entre los tiempos que toma realizar las funcionalidades consideradas para los diferentes tamaños y estructuras implementadas durante pruebas. Grafique el correspondiente análisis comparativo y su respectiva complejidad, recuerde usar la notación Big O para ello.

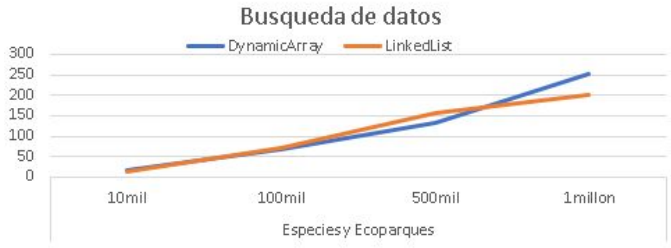
- Registro de Usuarios

La diferencia del tiempo de inserción masiva de datos tanto en el arreglo dinámico como en la lista enlazada es mínima lo que se puede explicar mediante el análisis de amortización en el cual se concluye que al comparar los métodos de inserción de ambas estructuras el orden daba lineal por lo tanto era igual de complejo insertar datos de manera masiva en una o en la otra. Es decir aunque en notación big O la inserción de un dato en la lista enlazada es de O(1) y en un arreglo dinámico es de

O(n) para un conjunto de datos n la complejidad es la misma si para el arreglo dinámico se va ahorrando para cuando llegue el momento del resize.



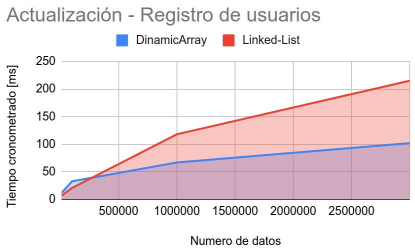


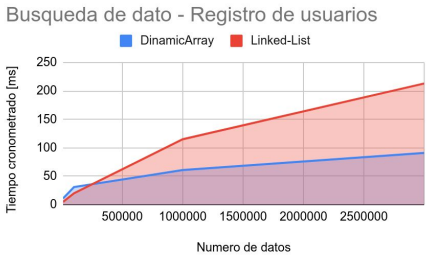


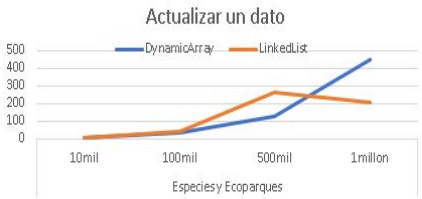
En el método de almacenamiento al momento de terminar la ejecución de la estructura se ve de manera muy definida la complejidad lineal para n datos y la similitud entre las estructuras a pesar de que en notación bigO las complejidades sean diferentes.

No se realiza gráfica comparativa para la inserción de un dato dado que el orden es constante(O(1)) para ambas estructuras y mantiene un valor inferior a 1ms.

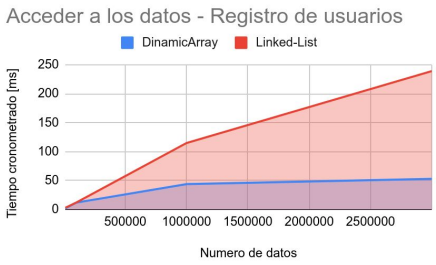
Para la actualización de datos y la búsqueda de datos los tiempos son similares en cada estructura dado que se requiere primero buscar el dato para poder actualizarlo sin embargo la pendiente de crecimiento es más grande en las listas enlazadas dado que se debe definir en cada paso del ciclo un elemento nodo que me permita buscar si el objeto (key) de ese nodo es el buscado es decir se requieren más instrucciones en el ciclo para poder hacer el método.







En la gráfica de acceso a todos los datos mostrada a continuación se ve una diferencia significativa en los tiempos de ejecución del método por la misma razón de los métodos anteriores dado el número de instrucciones en el ciclo y la complejidad menor en el arreglo de tener sus posiciones contiguas hace que la brecha entre las estructuras sea muy marcada.



Mediante el análisis anterior se puede inferir que en caso de que se fuera a utilizar una de estas dos estructuras el arreglo dinámico es más eficiente que en la búsqueda y acceso a los datos.

- Paginador de resultados: Se utilizaron las clases descritas anteriormente, por lo tanto tiene el mismo comportamiento del arreglo dinámico en la inserción masiva de datos. No se hizo necesaria la implementación de la lista enlazada, pues, como observamos, tiene menor eficiencia que el arreglo dinámico. El tiempo de respuesta de la funcionalidad oscila entre 500 ms y 1.7 s, dependiendo claramente de los datos. y la impresión es casi que inmediata.

II. ROLES Y ACTIVIDADES

Luis Felipe Gutierrez : Rol de líder, coordinador y técnico.
Realizó:La interfaz del proyecto, la implementación de registro de usuarios, la clase de Arreglos dinámicos, parte del video y parte de la presentación PDF y PowerPoint.

Hugo Alejandro Barrera: Rol de investigador, moderador y técnico.
Realizó: Implementacion de el paginador mediante Arreglos Dinamicos, parte del video, parte de la presentación PDF y PowerPoint.

Daniel Esteban Perez : Rol de investigador, moderador y observador.
Realizó: La implementación de ecoparques, fauna y flora, la clase de lista enlazada, parte del video y parte de la presentación PDF y PowerPoint

III. DIFICULTADES Y LECCIONES APRENDIDAS

- En la lista enlazada implementada se utilizó el tail, lo cual nos hacía tenerlo en cuenta en todos los métodos, así en algunos textos no tuvieran tail, tuvimos que adaptarlos. Esto generó algunos errores
- Al recorrer una lista enlazada es importante tener en cuenta que al acercarse al final, el next del último nodo es null, lo que puede generar algunos errores si estamos realizando comparaciones
- El uso de las variables tipo *Generic* requieren especial atención, pues puede que genere errores de Casteo a la hora de implementar las estructuras. Estos errores impidieron la impresión de los datos y su solución constaba de dos líneas de código.
- En algunos casos, la falta de experiencia en el uso de IDEs retrasaron el trabajo e implicaron un reto adicional al momento de programar. Gracias al trabajo en equipo se pudo superar este impedimento.
- El tiempo y la limitante de conocimiento de estructuras frenaron ideas para reforzar el proyecto, se intentarán incorporar en las siguientes etapas de programación, aún así el uso de pilas, colas, listas enlazadas y arreglos dinámicos son una base rígida para el desarrollo de la aplicación.
- Al momento de implementar múltiples switch case es importante llevar un orden y control de los breaks, para evitar que el programa entre en casos que no deseamos.
- No teníamos un norte claro al inicio lo que hizo que se perdieran esfuerzo hechos en la etapa inicial del proyecto, el aprendizaje a raíz de esto radica en que se entendió la importancia de no sectorizar el trabajo por grupos independientes dado que nos desconectamos, en lugar de esto los trabajos independientes se deben seccionar de tal manera que todos tenemos conocimiento e incidencia en todo y esto nos permite unir fuerzas y apoyarnos en dificultades individuales.
- La virtualidad debe de entenderse como un impedimento que nos lleva a tener menos empatía entre los integrantes de grupo si no se hacen espacios seguidos de compartir los avances, el aprendizaje es a tener un espacio que nos permita compartir los avances y también los problemas, dado que puede que alguno se haya enfrentado a ese mismo problema y ya lo haya podido solucionar, crecimos como equipo y ahora tendremos más éxito para unir esfuerzos y hacer nuestro trabajo mucho más productivo.