Red Social Personal

Diego Alejandro Alvarado Chaparro, Luis Angel Rodriguez Valdelamar

I. INTRODUCCIÓN

Pensando en la necesidad de crean una red social personal y privada para manejar las amistades se desarrolla este proyecto, una red social personal con la cual se podrán organizar las relaciones con amigos sin necesidad de usar alguna red social publica en internet.

II. DESCRIPCIÓN DEL PROBLEMA A RESOLVER

Cuando se necesita o se quiere organizar las amistades de manera privada sin que nadie más tenga acceso a esta información (incluyendo las empresas dueñas de las redes sociales) se podrá lograr usando una red social personal.

III. USUARIOS DEL PRODUCTO DE SOFTWARE

Personas que quieran o requieran un manejo privado de sus relaciones con otras personas.
Se requiere poca experiencia con el uso de dispositivos tecnológicos.
Conocimiento mínimo acerca de que es una red social.
.

IV. REQUERIMIENTOS FUNCIONALES DEL SOFTWARE

La información se almacena en Perfiles, los cuales tienen los siguientes atributos:

- Nombre (String)
- Apellido (String)
- Correo (String)
- Contraseña (String)
- Edad (int)
- Genero (String)

La información de los perfiles se almacena en un archivo ".txt" del cual se escribe o cargan los datos. Para la versión de prueba se manejarán dos archivos de texto: "usuarios10K.txt" y "usuarios100K.txt", los cuales tienen 10 mil perfiles y 100 mil perfiles guardados, respectivamente.
Para organizar los perfiles se usó una lista doblemente enlazada (Double Linked List) la cual maneja nodos que almacenan un Perfil cada uno.
Además, cada perfil tiene 3 listas doblemente enlazadas que guardan Strings:

- Amigos
- Muro
- Solicitudes

Por el momento no se guarda la información de estas listas pero durante la ejecución de la aplicación son usadas.

Estas listas tienen las funciones mínimas requeridas de una lista doblemente enlazada: PushFront, PopFront, TopFront, PushBack, PopBack, TopBack, AddAfter, AddBefore, EmptyList, Erase, Find.
Además de las funciones básicas vistas en clase, se manejan:

- *Función getItem (i)*
- *Descripción:*
  - Dado un int i se itera desde la cabeza de la lista al siguiente nodo i veces, una vez termina retorna la llave del nodo i de la lista.
- *Acciones iniciadoras y comportamiento esperado*: Al tener un objeto tipo DoubleLinkedList<T> se puede conseguir el ítem en la posición n de la lista llamando al método de la lista getItem(n). Si n es mayor al tamaño de la lista se retorna null.

Requerimientos funcionales: Se debe tener una lista tipo DoubleLinkedList<T> para poder llamar el método.

- *Crear nuevo Perfil*
- *Descripción:*
  - Desde la interfaz de la aplicación se podrá crear un nuevo perfil para ingresar a la red social mediante esta funcionalidad.
- *Acciones iniciadoras y comportamiento esperado*:

Una vez abierta la aplicación se podrá crear un nuevo perfil dando clic en el botón registro. Este activara la sección para crear un nuevo perfil permitiendo escribir en los campos correspondientes a los datos necesarios para crearlo y cada campo tiene requisitos. Nombre, Apellido, Contraseña, Correo y Genero deben ser tipo String, Edad debe ser tipo int y Correo debe seguir el siguiente formato: texto1@texto2.texto3 siendo texto1, texto2 y texto3 cadenas de caracteres y al ingresar un correo se corrobora que el correo no en uso recorriendo la lista desde la cabeza buscando perfil por perfil que no exista. Además de esto, ningún campo puede tener "," dentro. Se agregará el perfil a una lista doblemente enlazada con PushBack.

Requerimientos funcionales: Se debe tener la aplicación abierta y con la sección de creación de perfiles activa.

- *Iniciar Sesión.*

- *Descripción:*

  - Al ingresar un correo que ya este registrado con su respectiva contraseña, la aplicación permite ver y modificar información del perfil.

- *Acciones iniciadoras y comportamiento esperado*:

Se requiere que se ingrese en el campo Correo un correo que este guardado en el archivo de texto que se este utilizando (usuarios10K.txt o usuarios100K.txt) y se requiere ingresar en el campo la contraseña que esta en el perfil que tiene el correo ingresado. La aplicación buscara desde la cabeza hasta la cola de la lista para ver cual nodo tiene el perfil con el correo que se ingresó. Si el correo no esta en alguno de los perfiles de la lista se muestra un mensaje de error. Si el correo si existe dentro de alguno de los perfiles pero la contraseña no es la correspondiente se muestra un mensaje de error. Si es correcto el Correo y la contraseña entonces la aplicación creara una referencia al Perfil y se podrá ver o modificar la información del mismo.

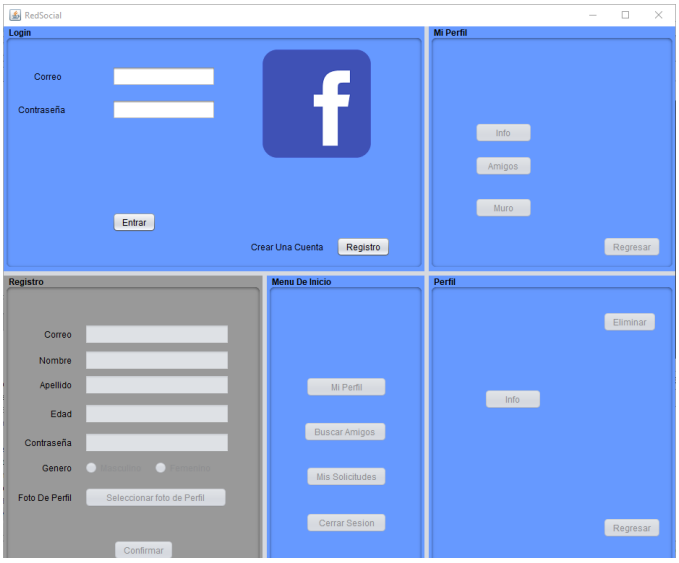
- *Publicar en el Muro.*

- *Descripción:*

  - Al haber iniciado sesión se podrá “publicar” en un muro personal el texto que se desee.

- *Acciones iniciadoras y comportamiento esperado*: al dar clic en el botón “muro” se abrirá una pequeña ventana en la cual se podrán agregar nuevas entradas en el muro. Para poder acceder al muro ya se debió haber iniciado sesión y por lo tanto se tiene acceso al perfil del usuario que ingreso y con ello se tiene acceso directo al muro.

V.DESCRIPCIÓN DE LA INTERFAZ DE USUARIO PRELIMINAR



La interfaz se divide en 5 secciones, cuando una sección esta activa las demás no lo son.

VI. ENTORNOS DE DESARROLLO Y DE OPERACIÓN

La aplicación se esta desarrollando en NetBeans usando el lenguaje de programación Java. La aplicación esta pensada para funcionar en dispositivos Windows versiones Windows 7 o más recientes con java instalado, 32 o 64 bits, memoria RAM recomendada 1GB.

VII. PROTOTIPO DE SOFTWARE INICIAL

https://github.com/dalvaradoc/ProyectoED

VIII. PRUEBAS DEL PROTOTIPO

Pruebas con 10 mil usuarios registrados :

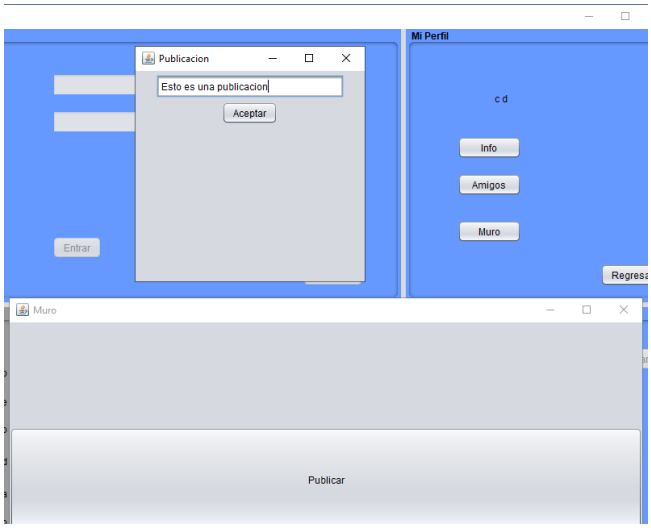
- Crear nuevo Perfil.

Tiempo prom : 0.9023s

- Iniciar Sesión:

Tiempo prom : 1.3117s

- Publicar en el muro



Tiempo prom: 0.144s

Pruebas con 100mil usuarios:

- Crear nuevo Perfil
  Tiempo prom : > 5min
- Iniciar Sesión
  Tiempo prom : > 5min
  La implementación actual usa la función getItem para conseguir el i-esimo elemento de la lista. Para lograrlo getItem avanza i veces los nodos de la lista. Para crear un nuevo perfil se busca si el correo ya existe, para esto se hace un for por cada perfil que haya (100 mil) y se llama getItem(i), por lo tanto al buscar se esta iterando dos veces volviendo la funcionalidad de $O(n^2)$, mas específicamente seria $n(n+1)$.
  Al usarse arreglos conseguir el i-esimo elemento de la lista seria tipo O(1) y volvería la búsqueda O(n).

- Publicar en el muro
  Tiempo prom : 0.144