Task It

Adrian Ramirez, Mateo Bustos, Santiago Avila

No. de Equipo Trabajo: {*5*}

I. INTRODUCCIÓN

En este documento, presentaremos una descripción detallada de cómo nuestro equipo abordó el diseño y desarrollo de un programa que permita asignar tareas de manera efectiva y eficiente dentro de un equipo de trabajo.

En este proyecto, nos enfocamos en el diseño y la implementación de diversas estructuras de datos unidimensionales, como colas, pilas y listas enlazadas, para manejar y optimizar la asignación de tareas dentro del equipo. A través de este documento, explicaremos en detalle cómo funcionan estas estructuras de datos y cómo se integran en la aplicación para mejorar la gestión de tareas y la productividad del equipo de trabajo. También presentaremos una serie de pruebas y análisis de rendimiento con las diferentes soluciones implementadas con las estructuras de datos antes mencionadas, de forma que se logre evidenciar cuál de estas es la más óptima para el fin requerido.

En resumen, este documento es una guía completa para entender cómo nuestro equipo abordó el desafío de implementar estructuras de datos en un programa de distribución de tareas para equipos de trabajo, que logre crear mayor compenetración para la culminación de tareas, en especial para equipos de trabajo numerosos, siempre buscando la mayor optimización posible en términos de manejo y gestión de gran número de datos.

II. DESCRIPCIÓN DEL PROBLEMA A RESOLVER

Tenemos claro que la repartición de tareas en un equipo de trabajo es de las cosas que más problemas puede traer, ya sea por falta de comunicación asertiva, por inequidad en la repartición de las mismas, o muchas otros factores que pueden afectar de cierta manera dicha repartición, y que al ser tan determinante en la realización exitosa de un proyecto pueda llevar a malos resultados si no se tiene sumo cuidado con este tema.

III. USUARIOS DEL PRODUCTO DE SOFTWARE

Los usuarios esperados de nuestro producto de software serían cualquier grupo de trabajo que necesite clasificar y repartir diferentes tareas para una mayor eficiencia, ya sean grupos de trabajo colegiales, universitarios, laborales, entre otros.

Habría un único rol, el de integrante del grupo, pues no habrían privilegios de acceso ni de seguridad, ya que todos podrían en todo momento visualizar, agregar y finalizar las tareas que ellos vean convenientes.

IV. REQUERIMIENTOS FUNCIONALES DEL SOFTWARE

- ***Ingresar nueva tarea***

- ***Descripción:***

En esta sección el integrante del grupo podrá ingresar una nueva tarea a la cola de tareas pendientes. En esta función el integrante podrá definir a que subdivisión específica será encolada la tarea.

- ***Acciones iniciadoras y comportamiento esperado*:**

El integrante deberá suministrar una descripción de la tarea además de especificar la subdivisión a la que está pertenecera. Si la descripción es válida, el programa le asignará un ID único a la tareas y la guardará en la cola de tareas relacionada con la subdivisión escogida (cada subdivisión cuenta con su cola única de tareas) . En caso de que el integrante no proporcione una descripción, el programa no permitirá el ingreso de la tarea.

- ***Requerimientos funcionales*:**

    - Se podrán ver todas las subdivisiones registradas actualmente.
    - El usuario podrá describir la tarea de la manera en que lo crea conveniente. En caso de que la descripción de la tarea esté vacía, no se permitirá el ingreso de la tarea.
    - El usuario podrá elegir libremente la subdivisión en la que está interesado.
    - Todas las tareas válidas que se ingresen tendrán asignada una identificación única.

- ***Notificar tarea finalizada***

- ***Descripción:***

Lo que hace la funcionalidad “Notificar tarea finalizada” es que poniendo la ID de la tarea que se realizó, reporte la tarea como finalizada y lo lleve al inicio.

- ***Acciones iniciadoras y comportamiento esperado*:**

Se espera que el usuario digite el ID de la tarea que realizó y que el programa haga comprobación, primero de que lo que ingresó el usuario sea un número, y luego que verifique si la

presunta ID que ingresó el usuario exista en los archivos guardados, y si es así de a la tarea como finalizada.

- ***Requerimientos funcionales:***

Para que el usuario pueda desempeñar correctamente la tarea:

- El software debe tener las comprobaciones anteriormente mencionadas, para que el programa no se quiebre luego de recibir un ID incorrecto o inexistente.
- Ser capaz de buscar en la lista doblemente enlazada en la que se guardan las subdivisiones, la tarea, y que cuando la encuentre, pase de la lista doblemente enlazada con cola de "Tareas en progreso" a la pila de "Tareas finalizadas".
- Debe habilitar correctamente el resto de funcionalidades para el correcto funcionamiento del programa.

- **Solicitar tarea**

- ***Descripción:***

Cuando el usuario requiera una nueva tarea, podrá pedirla en esta sección, únicamente tendrá que seleccionar la subdivisión deseada y se le otorgará la descripción y la ID de su nueva tarea.

- ***Acciones iniciadoras y comportamiento esperado*:**

Se espera que el usuario seleccione la subdivisión deseada en un combobox y presione el botón de "Pedir Tarea", luego el programa buscará en los archivos guardados la subdivisión en la cola de "Tareas" y le suministre al usuario la descripción y la ID de la más antigua de ellas.

- ***Requerimientos funcionales*:**

Los requerimientos asociados a esta funcionalidad son:

- El programa debe verificar que existan tareas sin asignar en la subdivisión deseada para posteriormente notificar al usuario.
- Debe otorgar correctamente las subdivisiones que hay actualmente en los archivos guardados.

- ***Agregar nueva subdivisión***

- ***Descripción:***

El programa permitirá crear nuevas subdivisiones al interior del grupo donde cada una contará con sus respectivas listas de tareas (Sin asignar, en proceso, finalizado).

- ***Acciones iniciadoras y comportamiento esperado*:**

Secuencia de acciones del usuario y respuestas esperadas del programa para esta funcionalidad.

En esta función el usuario únicamente escribirá el nombre de la subdivisión que quiere crear. Si la descripción no está vacía, el programa procederá a instanciar una nueva subdivisión con sus diferentes listas de tareas mencionadas anteriormente.

- ***Requerimientos funcionales*:**

- Permitir al usuario ingresar el nombre de la subdivisión y hacer la verificación de que dicho nombre no esté vacío.
- Si el nombre es valido, se instancian una nueva subdivisión la cual contará contará con los siguientes atributos:
  - Nombre de la subdivisión.
  - Cola para guardar tareas sin asignar.
  - Lista doblemente enlazada con cola para guardar tareas en progreso.
  - Pila para guardar tareas finalizadas.
- La subdivisión se guardará en una lista doblemente enlazada con cola junto con el resto de subdivisiones

- ***Visualizar estado de tareas***

- ***Descripción:***

Permite visualizar en tablas los estados de cada una de las tareas de una subdivisión determinada.

- ***Acciones iniciadoras y comportamiento esperado*:**

El usuario podrá escoger cualquiera de las subdivisiones que se encuentren registradas en el programa actualmente y, al presionar el botón "consultar" la información de las tareas será cargada.

- ***Requerimientos funcionales*:**

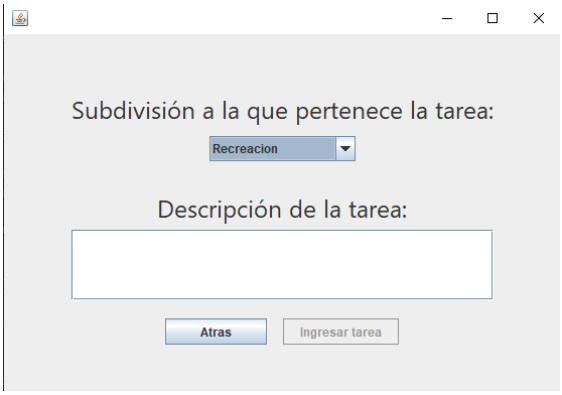
- Se proveerá una lista con todas las subdivisiones que se encuentren registradas al momento de usar la función.
- Al momento de confirmar la subdivisión en la que se está interesado por medio del botón "consultar".
- Se mostrarán 3 tablas donde en cada una se guardarán las tareas asociadas con cada uno de los estados definidos en el programa (Sin asignar, En proceso, Finalizado).

V. DESCRIPCIÓN DE LA INTERFAZ DE USUARIO PRELIMINAR

- **Pestaña-Inicio**

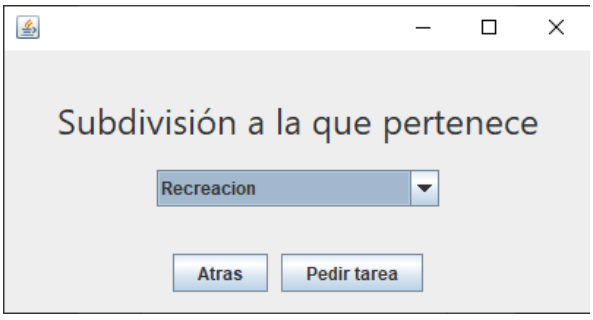
Pestaña principal del programa, la cual da acceso a las diferentes funcionalidades del mismo.

- **Pestaña-Ingresar nueva tarea**



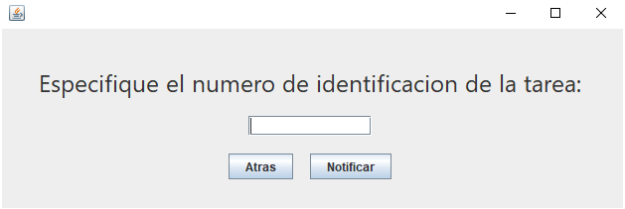
Pestaña que permite al usuario agregar una nueva tarea, almacenando y asignando a su respectiva subdivisión.

- **Pestaña-Solicitar Tarea**



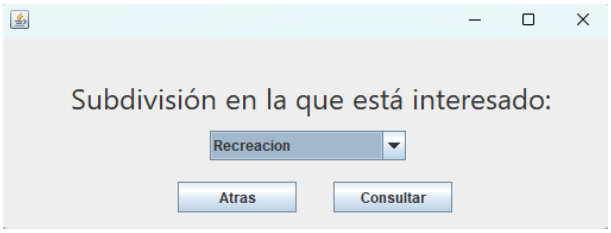
Pestaña en que el usuario puede solicitar que se le asigne una tarea, de acuerdo a la subdivisión a la que se pertenezca y al tiempo que esta tarea lleva agregada y pendiente.

- **Pestaña-Notificar tarea finalizada**



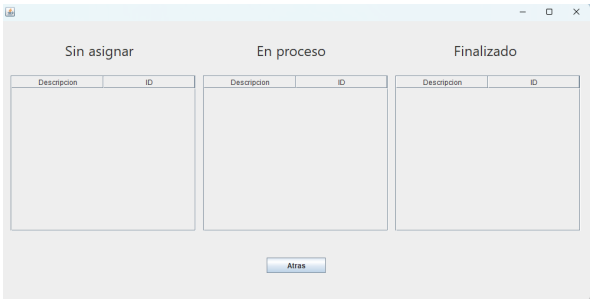
Esta pestaña permite notificar la culminación de una tarea, permite ingresar el ID único para cada tarea para notificar que esta ha sido terminada

- **Pestaña-Ver estado tarea**



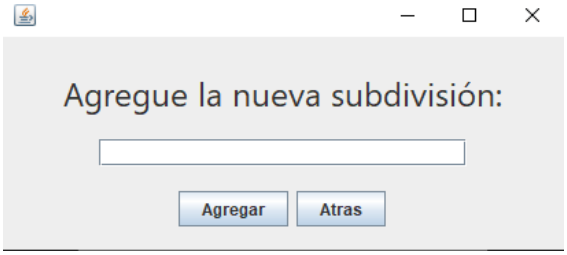
Pestaña para indicar la subdivisión la cual queremos conocer el estado de sus tareas.

- **Pestaña-Tablas con los estados de las tareas**



Pestaña donde se mostrarán en tablas el estado de las tareas de una subdivisión específica.

- **Pestaña-Agregar subdivisiones**



Pestaña que permite agregar una nueva subdivisión de trabajo dentro del equipo, al cual se le pueden asignar sus propias tareas.

VI. Entornos de desarrollo y de operación

El software se ha desarrollado en Java con IDE NetBeans en sistemas operativos de Windows.
El proyecto puede ser ejecutable en cualquier plataforma que soporte java, con un hardware básico.

VII. PROTOTIPO DE SOFTWARE INICIAL

Link del repositorio del proyecto en github: https://github.com/adrianram21/Task-It

VIII. IMPLEMENTACIÓN Y APLICACIÓN DE LAS ESTRUCTURAS DE DATOS

Para el prototipo del programa se usaron 4 diferentes tipos de estructuras de datos abstractas para el buen funcionamiento del algoritmo, en una lista doblemente enlazada con cola se guardaban los objetos de la clase "Subdivisiones", estos objetos a su vez tienen tres estructuras como atributos, los tres estados posibles de las tareas, las tareas sin asignar las guarda en una cola llamada "Tareas", para que la tarea que más lleve esperando sea la primera que vaya siendo asignada, las "Tareas en progreso", están guardadas en una lista doblemente enlazada con cola, pues estas no tienen prioridad de salida, solo va saliendo la que el usuario haya notificado como finalizada, y por último las "Tareas finalizadas" se van apilando en una pila. Adicionalmente, para visualizar el estado de las tareas en tablas se hizo uso de Static array lists para ingresar las filas dentro de las tablas creadas en la pestaña correspondiente..

IX. PRUEBAS DEL PROTOTIPO Y ANÁLISIS COMPARATIVO

- Asignar tarea:

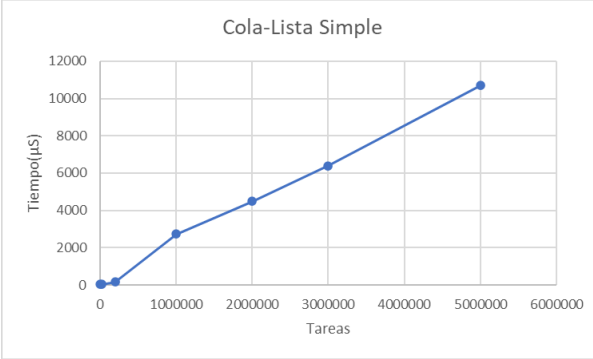
En esta funcionalidad se realiza dequeue en la cola y PushBack en la lista

- Primera implementación-Cola(lista enlazada) y Lista Doblemente enlazada con cola
  Se espera una complejidad constante O(1)

Cola-Lista doble con cola	
Tareas	Tiempo(µS)
2000	19.9
20000	17.7
200000	22.8
2000000	22.6
6000000	21.4
10000000	18.1
20000000	21.8
60000000	20.5
100000000	22.6

- Segunda implementación-Cola(lista enlazada) y Lista simplemente enlazada
  Se espera una complejidad lineal O(n) gracias al pushback de la lista simplemente enlazada

Cola-Lista Simple	
Tareas	Tiempo(µS)
2000	24.8
20000	32.3
200000	153.8
1000000	2733.3
2000000	4481
3000000	6381.9
5000000	10686

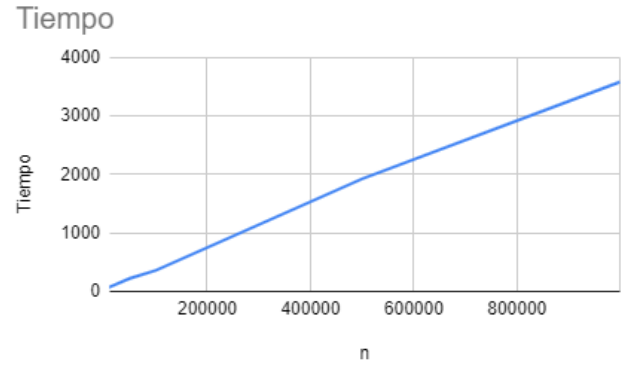


- Tercer implementación-Cola(Arreglo circular) y Lista Doblemente enlazada con cola
  Se espera una complejidad constante O(1)

Cola-Lista doble con cola	
Tareas	Tiempo(µS)
2000	11.4
20000	22.1
200000	19.9
2000000	13.9
6000000	12.8
10000000	11.8
20000000	16.6
60000000	14.8
100000000	15.4

- Ingresar nueva tarea:
  Implementación principal, cola en una lista simplemente enlazada.
  Complejidad esperada O(n).

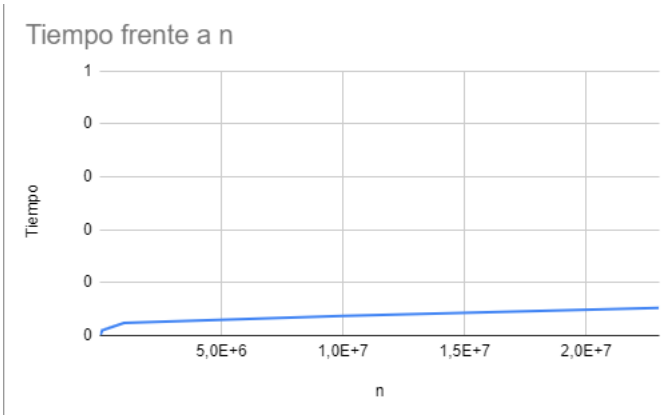
Cola Lista Simplemente Enlazada	
Tareas	Tiempo(µS)
5000	8
10000	76
50000	226.1
100000	363.7
500000	1924.8
1000000	3580.5



Podemos observar que la gráfica tiende a comportarse linealmente por lo que la complejidad esperada de O(n) se cumple.

Implementación de la cola en una lista doblemente enlazada con cola.
Complejidad esperada O(1).

Cola en una lista doblemente enlazada con cola	
Tareas	Tiempo(µS)
10000	1.20
100000	9.30
1000000	23.30
10000000	37.4
23000000	51.8



En la gráfica se puede observar como la gráfica tiende a ser constante con diferentes cantidades de datos, haciendo que la teoría se cumpla en la práctica.

- **Notificar tareas finalizadas**

Esta funcionalidad se divide en dos partes: Borrado de tareas en la lista de tareas en progreso e ingreso de tareas en la lista de tareas finalizadas. Teniendo esto en cuenta, se realiza un análisis de complejidad para cada parte.

1. Borrado de tareas en la lista de tareas progreso

Implementación principal: Lista doblemente enlazada con cola.

Complejidad esperada: O(n)

Lista doblemente enlazada con cola (Erase)	
Tareas	Tiempo (µS)
10000	1125,4
100000	8159,2
1000000	10710,9
10000000	81207,2

Implementación alternativa: Array list estático

Complejidad esperada: O(n)

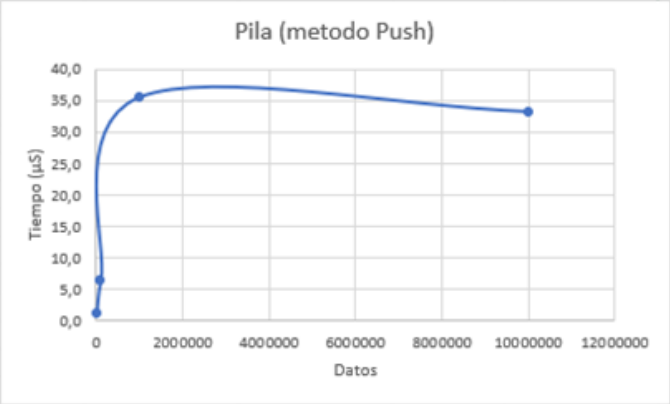
Array estático (Delete)	
Tareas	Tiempo (µS)
10000	452
100000	4059
1000000	11397,7
10000000	24813,9
10000000	256976,8

2. Ingreso de tareas en la lista de tareas finalizadas

Implementación principal: Pila implementada con lista simple enlazada

Complejidad esperada: O(1)

Pila (Push)	
Tareas	Tiempo (µS)
10000	1,2
100000	6,3
1000000	35,5
10000000	33,2



Implementación alterna: Array list estático

Complejidad esperada: O(n)

Array list estatico (Push Front)	
Tareas	Tiempo (µS)
10000	280,6
100000	1708,4
1000000	6043,3
10000000	32274,4
100000000	259358,9

X. ROLES Y ACTIVIDADES

INTEGRANTE	ROL(ES)	ACTIVIDADES REALIZADAS (Listado)
Adrian Ramirez	Líder	Distribución de trabajo.
	Tecnico	Diseño de interfaz.
		Implementación de estructuras de datos.
Santiago Avila	Coordinador	Programar funciones.
		Redactar documento.
	Programador	
Mateo Bustos	Investigador	Buscar cómo solucionar los nuevos desafíos encontrados.
		Lectura de la bibliografía en busca de información útil.
	Observador	Analizar el trabajo en equipo y ver los aspectos en los que flaquea.
		Comprobación del código base.

XI. DIFICULTADES Y LECCIONES APRENDIDAS

Las dificultades a las que nos enfrentamos en primer lugar, se dieron a la hora de unir las funciones creadas por cada persona, puesto que cada uno tiene una forma diferente de analizar y realizar la solución de dicha funcionalidad, entonces a cada uno tener una visión diferente sobre cómo realizar X o Y tarea, desemboca en que el manejo y construcción de las funciones de inicio sean diferentes, tanto como que retornen datos que algunos compañeros no esperaban o que las funciones reciban ciertos argumentos que otros no tenían ya estipulados en sus códigos.

Las lecciones aprendidas es que la comunicación en el equipo de trabajo es muy importante para tener claridad en cuáles

funciones desarrollará cada integrante del equipo y cómo las hará, por ello se realizarán reuniones periódicas para la socialización de cada avance realizado.

XII. REFERENCIAS BIBLIOGRÁFICAS

[1] J. T. Streib y T. Soma, “Guide to Data Structures: A Concise Introduction Using Java ” . Suiza: Springer International Publishing, 2017.