Task It

Adrian Ramirez, Mateo Bustos, Santiago Avila

No. grupo del curso: {*2*}

No. de Equipo de Trabajo: {*5*}

I. INTRODUCCIÓN

En este documento se presentarán los avances finales del proyecto Task It para la materia de estructuras de datos. Para este último avance, además de implementar algunas de las más recientes estructuras de datos vistas en la materia (conjuntos disjuntos y hash) y mejorar el diseño del programa, también se un sistema de logeo y la redefinición algunas funciones, lo cual permite que el usuario tenga una mejor experiencia al momento de usar el programa. A lo largo de este documento se realizará la respectiva definición de cada una de las funciones además de realizar el análisis de complejidad de algunas de estas para ver como la implementación de las nuevas estructuras nos permite mejorar el desempeño y la seguridad de la información que maneja el programa.

II. DESCRIPCIÓN DEL PROBLEMA A RESOLVER

Uno de los más grandes beneficios de trabajar en equipo, ya sea en un ambiente laboral o académico, es la posibilidad de poder repartir el trabajo entre cada uno de los integrantes del grupo. Sin embargo, se necesita que el equipo tenga una gran comunicación y compenetración entre ellos para poder realizar la repartición de tareas de manera efectiva, pero, debido a distintos factores, puede llegar a ser difícil conseguir estas condiciones. El proyecto Task It pretende servir como un gestor de tareas para equipos de trabajo en el cual se pueda registrar, asignar y notificar la finalización de las tareas dentro de un equipo de trabajo y asimismo poder tener un registro del desempeño de cada una de las subdivisiones (o subgrupos) que compone a un equipo. Esto permitirá que el flujo de trabajo dentro de un equipo se vea lo menos afectado posible por problemas de comunicación o entendimiento que puedan existir dentro del mismo, logrando así que los resultados ofrecidos por cualquier grupo mejoren y sean conseguidos en un tiempo más corto.

III. USUARIOS DEL PRODUCTO DE SOFTWARE

Para el proyecto entendemos que dentro de cualquier equipo de trabajo existen subgrupos o subdivisiones que se encargan de partes específicas del proyecto o trabajo que se está desarrollando. Además, también sabemos que hay información relacionada con el flujo de trabajo que puede ser más importante para unos usuarios que para otros. Teniendo esto en cuenta, identificamos dos roles dentro de los usuarios:

- **Integrante:** Los integrantes son cada una de las personas que se encuentran dentro de una subdivisión trabajando. No existe un límite para la cantidad de integrantes que pueden pertenecer a una subdivisión, pero cada integrante podrá estar relacionado con una única subdivisión. Este rol esencialmente tiene acceso a 3 funciones dentro del programa:
    - Ingresar nuevas tareas dentro del sistema.
    - Solicitar la asignación de una tarea.
    - Notificar la finalización de tareas.

- **Líder:** Estos usuarios son las personas principales a cargo de cada una de las subdivisiones y, para el proyecto, se asume que existe un solo líder para cada subdivisión y además este será siempre el primer integrante y creador de la subdivisión. Este rol, además de tener acceso a las mismas funciones que cualquier integrante también necesita saber cuál es el desempeño de su subdivisión, y por lo tanto dentro del programa serán las únicas personas que podrán acceder a la información relacionada con el flujo de trabajo de la subdivisión, con el fin de que puedan saber constantemente que tan bien se encuentra la subdivisión en términos de productividad y, en caso de ser necesario, que puedan utilizar esta información como referencia para proponer estrategias que mejoren la productividad de la subdivisión.

IV. REQUERIMIENTOS FUNCIONALES DEL SOFTWARE

Sistema de inicio de sesión

- **Descripción**

Para hacer uso del programa y de sus funciones, existirá un procedimiento de inicio de sesión donde se tendrá que presentar un usuario y contraseña válidos antes de poder usar el programa.

- **Acciones iniciadoras y comportamiento esperado**

Se espera que el usuario presente un usuario y contraseña previamente registrados dentro del programa. El programa se encargará de validar dicho usuario y contraseña guardado en un HashTable y, en caso de ser correctos, se le mostrará al usuario el menú principal donde la persona podrá hacer uso de las funciones que estén permitidas para su rol en específico (integrante o líder).

- **Requerimientos funcionales**

1. El sistema deberá ser capaz de validar correctamente la existencia del usuario dentro

del sistema además de también verificar que la contraseña sea correcta.
2. Al iniciar sesión, el sistema debe considerar cuál es el rol del usuario con el fin de restringir el acceso a las funciones que no están destinadas para su uso por parte de un rol específico.
3. Al momento de usar cualquier funcionalidad contenida en el menú principal se deberá garantizar que cualquier consulta o modificación se haga únicamente dentro de la subdivisión a la que pertenece el usuario.

Registrarse

- **Descripción**

Se permitirá que cualquier persona pueda registrarse dentro del programa para que posteriormente pueda iniciar sesión y hacer uso de las funcionalidades del programa.

- **Acciones iniciadoras y comportamiento esperado**

Al momento de que una persona quiera registrarse tendrá que especificar si se quiere registrar como líder o integrante. Si el usuario decide registrarse como integrante, tendrá que crear un usuario, una contraseña y especificar la subdivisión a la que pertenece. Por otro lado, si el usuario quiere registrarse como líder, también tendrá que crear un usuario, una contraseña y además especificar un nombre para la subdivisión que va a crear y de la cual ese usuario será el líder.

- **Requerimientos funcionales**

1. El sistema debe ser capaz de guardar toda la información dada por el usuario para que no existan problemas al momento de iniciar sesión.
2. Las subdivisiones creadas al momento de que un líder se registra deberán estar disponibles para ser escogidas por parte de los integrantes de esa subdivisión que se registren después del líder.

Ingresar nueva tarea

- **Descripción**

El programa permitirá que cualquier usuario pueda ingresar una nueva tarea la cual pueda ser almacenada con base en su nivel de prioridad.

- **Acciones iniciadoras y comportamiento esperado**

Se espera que el usuario indique un nivel de prioridad del 1 al 10 además de que proporcione una breve descripción de la tarea. Si la tarea es válida, será guardada dentro de una cola prioritaria donde se guardan las tareas sin asignar de la subdivisión en específico a la que pertenece el usuario.

- **Requerimientos funcionales**

1. El sistema verificará que la descripción de la tarea no se encuentre vacía.
2. El sistema comprobará que el nivel de prioridad dado por el usuario se encuentre entre los límites definidos (del 1 al 10).
3. El sistema debe guardar la tarea en la cola prioritaria que está relacionada con la subdivisión en específico a la que pertenece el usuario.
4. Si la tarea ingresada es válida, se le asignará una identificación única al momento de ser guardada.

Solicitar tarea

- **Descripción**

Cualquier usuario podrá solicitar que se le asigne una tarea que esté relacionada con la subdivisión a la que pertenece el usuario.

- **Acciones iniciadoras y comportamiento esperado**

Después de que el usuario ya haya iniciado sesión y tenga acceso a las funcionalidades que ofrece el programa, podrá hacer click en la opción relacionada con la solicitud de tarea y el sistema inmediatamente le mostrará la descripción de una tarea relacionada con su subdivisión y además le mostrará la identificación que se le dio a la tarea al momento de ingresarla en el sistema.

- **Requerimientos funcionales**

1. El programa entregará las tareas a los usuarios que lo soliciten según el orden de prioridad.
2. La tarea entregada siempre pertenecera a la subdivisión del usuario.
3. La tarea entregada se moverá de la cola prioritaria donde están guardadas las tareas sin asignar y pasará a estar en un AVL donde se guardan las tareas en progreso de la subdivisión del usuario.

Notificar la finalización de una tarea

- **Descripción**

Cualquier usuario podrá indicarle al programa la finalización de una tarea en progreso haciendo uso del ID suministrado al momento de solicitar la tarea.

- **Acciones iniciadoras y comportamiento esperado**

Se espera que el usuario al terminar de realizar una tarea indique la identificación que se le entregó previamente. Se buscará la identificación dentro del AVL donde se guardan las tareas en progreso de la subdivisión del usuario, con el fin de moverla a la pila donde se guardan las tareas finalizadas de esa misma subdivisión.

- **Requerimientos funcionales**

1.

Ver estado de las tareas

- **Descripción**

Los usuarios que cuenten con el rol de líder podrán ver en qué estado (sin asignar, en proceso o finalizado) se encuentra cada una de las tareas relacionadas con la subdivisión que lidera.

- **Acciones iniciadoras y comportamiento esperado**

Se espera que un usuario que se haya registrado como líder de una subdivisión acuda a esta opción. Si el usuario posee el rol de líder, se mostrarán 3 tablas donde se mostrarán las tareas clasificadas en sus 3 estados posibles (sin asignar, en proceso o finalizado).

- **Requerimientos funcionales**

1. El sistema permitirá visualizar el estado de las tareas únicamente a los líderes de las subdivisiones.
2. Un líder podrá ver únicamente el estado de las tareas de la subdivisión a la que pertenece.

V. INTERFAZ DE USUARIO

- **Inicio de sesión**

Esta es la interfaz inicial presentada a cualquier usuario donde podrá ingresar su usuario y contraseña para iniciar sesión dentro del programa. Al mismo tiempo en la interfaz también se dispone de la opción de registro para los usuarios que no cuenten con un usuario dentro del programa y ahora quieren crear uno.

- **Interfaz para escoger registro según el rol**

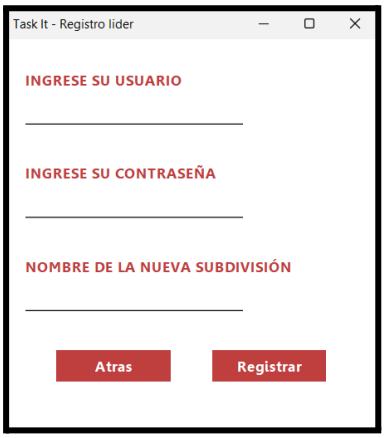
En caso de que en la primera interfaz presentada se escoja la opción “Registrarse”, al usuario le aparecerá el siguiente mensaje donde deberá decidir si quiere ser registrado dentro del programa como integrante o líder de una subdivisión.

- **Registro para integrantes**

Esta interfaz es la que le aparecerá a cualquier persona que decida registrarse como integrante de una subdivisión donde además de especificar un usuario y contraseña, también deberá escoger la subdivisión a la que estará ligado dicha persona.

- **Registro para líderes**

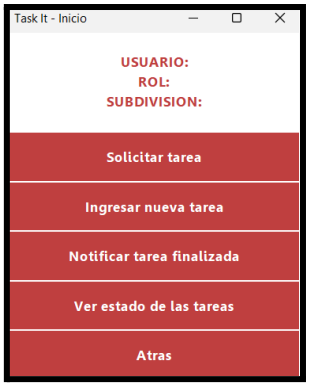
Esta interfaz le aparecerá a cualquier persona que decida registrarse como líder de una subdivisión. Además, teniendo en cuenta que cada subdivisión contará con un único líder, cada vez que una persona decida registrarse como líder de manera simultánea se creará la subdivisión de la que será líder la persona. Es esta la razón por la cual además de solicitarle a la persona que especifique un usuario y una contraseña, también se pide que ingrese un nombre para la subdivisión única que se creará.



- **Menú principal**

Después de que el usuario inicie sesión, se encontrará con el menú principal del programa donde tendrá

acceso a las funciones permitidas para su rol en específico las cuales fueron explicadas en secciones anteriores de este documento. Además, en la parte superior de esta interfaz se encontrará la información básica del usuario (Nombre, rol y subdivisión a la que pertenece) que se encuentra utilizando el programa.



- **Ingresar nueva tarea**

Cuando el usuario quiera ingresar una nueva tarea, se desplegará una interfaz donde el usuario tendrá que elegir un nivel de prioridad de la tarea y además proporcionar una descripción breve de la tarea que se debe realizar para posteriormente dar click al botón "Ingresar tarea" y así el programa procederá a guardar la tarea en el lugar que corresponda teniendo en cuenta siempre la subdivisión del usuario que hizo uso de la función.

- **Solicitar tarea**

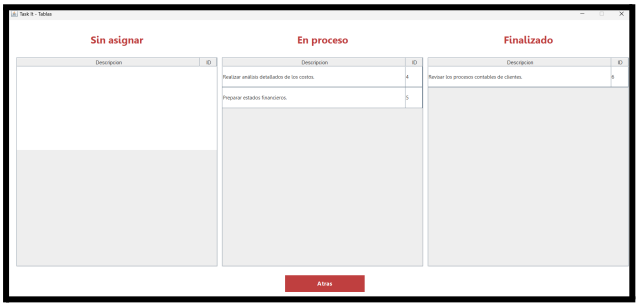
En el momento en que el usuario se encuentra en el menú principal y da click a la opción "Solicitar tarea", inmediatamente aparecerá un mensaje el cual le proporcionará una tarea relacionada con la subdivisión a la que pertenece el usuario. En el mensaje se podrá ver la descripción de la tarea y un número de identificación único para la misma proporcionado por el programa.

- **Notificar la finalización de tareas**

En la interfaz para la notificación de la finalización de las tareas únicamente habrá un espacio donde el usuario deberá escribir el número de identificación de la tarea que se le entregó al momento de solicitar la tarea para que así el sistema pueda ubicar la tarea que se acaba de terminar y pueda darle el manejo adecuado.

- **Ver estado de las tareas**

En esta interfaz hecha únicamente para los usuarios que tienen el rol de líder, se podrán ver todas las tareas que están relacionadas con una subdivisión clasificadas por su estado (Sin asignar, en proceso o finalizada) para que así el líder de la subdivisión pueda llevar un control de que tan bien está trabajando la subdivisión. Para la visualización de las tablas, las tareas sin asignar se muestran en orden de prioridad, las tareas en proceso no tienen un criterio de orden al ser mostradas y las tareas finalizadas se muestran en orden cronológico inverso (mientras más recientemente se haya terminado una tarea, más arriba estará en la tabla).



VI. ENTORNOS DE DESARROLLO Y DE OPERACIÓN

Para la creación del software se usa únicamente el lenguaje Java y se hace uso de la IDE NetBeans para crear todo el programa. Todo el desarrollo del programa se hace dentro del sistema operativo Windows 11 y se espera que el programa pueda ser ejecutado en cualquier sistema que soporte la ejecución de programas hechos en Java. En términos de hardware se espera que el programa pueda ser ejecutado en cualquier computador con especificaciones mínimas.

VII. DESCRIPCIÓN DEL PROTOTIPO DE SOFTWARE

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

Para el programa se crearon 3 clases:

- **Subdivision**

Atributos:

- Nombre (String)
- Tareas sin asignar (Cola prioritaria).
- Tareas en progreso (AVL).
- Tareas finalizadas (Pila).
- Líder (Usuario).

- **Tarea**

Atributos:

- Descripción de la tarea (String).
- Identificación (int).
- Nivel de prioridad del 1 al 10 (int).

- **Usuario**

Atributos:

- Nombre del usuario (String).
- Rol (String).
- Nombre de la subdivisión a la que pertenece (String).

Para la el programas se utilizaron las siguientes estructuras de datos:

- **Colas prioritarias (Binary max heap):** Las tareas sin asignar son guardadas en estas estructuras ya que nos permite modificar el orden de los elementos basados en el nivel de prioridad que le da cada usuario a una tarea al momento de ingresar al sistema y así cada vez que un usuario solicite una tarea siempre se le entrega la tarea que tenga un nivel de prioridad más alto dentro de la cola prioritaria. Después de que la tarea es entregada, desaparece de la cola prioritaria y pasa a estar en el AVL donde se encuentran las tareas en progreso relacionadas con esa subdivisión.

- **AVL:** Está estructuras tiene dos usos principales dentro del programa:
  - Todas las subdivisiones son guardadas dentro de un mismo AVL con el fin de optimizar la búsqueda de estos. El programa verifica el nombre de la subdivisión a la que pertenece el usuario y posteriormente busca la subdivisión dentro de este AVL para poder acceder a sus atributos y poder ejecutar la función en particular que el usuario esté interesado en realizar.
  - Las tareas en progreso son guardadas en un AVL único por cada subdivisión con el fin de optimizar la búsqueda de las tareas. Esta estructura facilita la notificación de la finalización de tareas ya que el usuario proporciona la ID de la tarea que acaba de terminar y el programa busca la tarea dentro de este AVL con el fin de eliminarla de este y moverla hacia la pila donde se encuentran las tareas ya finalizadas relacionadas con la subdivisión.

- **Pila:** Las tareas finalizadas de cada subdivisión son guardadas en pilas ya que gracias a su estructura LIFO nos permite guardar las tareas de manera cronológica inversa y así los líderes de cada subdivisión pueden ver qué tareas han sido terminadas más recientemente.

- **Hash table:** A partir del uso de funciones hash para enteros y strings, se hace uso de un hash table donde se guardan todas las subdivisiones registradas en el programa con el fin de que, gracias a las características de los hash table, la búsqueda de estas subdivisiones tenga en promedio una complejidad de O(1) y así se pueda acceder de manera rápida a la información contenidas dentro de cada subdivisión.

- **Conjuntos disjuntos:** Esta estructura es utilizada para almacenar los usuarios, esto permite que cada usuario quede asignado al conjunto del usuario líder de la subdivisión a la que pertenezca, siendo asi eficiente ubicar a que subdivisión pertenece cada usuario, y también permitiendo la asignación del usuario a una subdivisión al realizar el proceso de unión de conjuntos.

IX. PRUEBAS DEL PROTOTIPO DE SOFTWARE

- **Operación find en la búsqueda de subdivisiones**

A lo largo del desarrollo de este proyecto siempre se ha buscado especialmente que la búsqueda de las subdivisiones dentro del programa sea lo más rápido posible. Por esta razón, para este análisis de complejidad nos interesamos principalmente en ver las diferencias que existen (en términos de tiempos de ejecución) en la operación find de las distintas estructuras de datos que hemos utilizado a lo largo de estas entregas para guardar y realizar la búsqueda de las subdivisiones dentro del programa. Las estructuras a comparar serán las siguientes:

- **Hash table**
  - Usada en la tercera entrega.

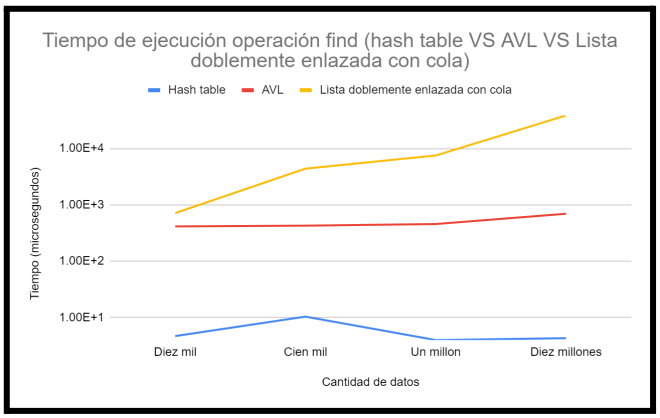
- Complejidad esperada: O(1).

- **AVL**
  - Usada en la segunda entrega.
  - Complejidad esperada: O(log(n)).

- **Lista doblemente enlazada con cola**
  - Usada en la primera entrega.
  - Complejidad esperada: O(n).

Al momento de hacer las respectivas pruebas existieron algunos problemas al usar cien millones de datos en el hash table debido a los valores tan grandes que se asignaban a los parámetros de las funciones hash usadas en el hash table, lo cual hizo que solo pudiéramos hacer pruebas hasta los diez millones de datos. Los resultados de esas pruebas son los siguientes:

	Tiempo (microsegundos)		
Cantidad de datos	**Hash Table**	**AVL**	**Lista doblemente enlazada con cola**
Diez mil	4.70	414.00	711.70
Cien mil	10.40	427.30	4381.00
Un millón	4.00	455.60	7473.80
Diez millones	4.30	692.10	38357.60



Como se puede ver en en la gráfica, la cual tiene al eje y (tiempo) en escala logarítmica, la diferencia en tiempos de ejecución entre estas el hash table y las otras 2 estructuras de datos al momento de ejecutar la operación find es bastante grande, lo cual nos permite corroborar que la decisión de haber usado un hash table para guardar las subdivisiones en esta última entrega fue un completo acierto en miras de mejorar el desempeño del programa.

X. ACCESO AL REPOSITORIO DE SOFTWARE

Link del repositorio de GitHub donde se encuentra el proyecto: https://github.com/adrianram21/Task-It

XI. ACCESO AL VIDEO DEMOSTRATIVO DEL SOFTWARE

Link del video: https://youtu.be/YpG6ckBoX8I

XII. ROLES Y ACTIVIDADES

INTEGRANTE	ROL(ES)	ACTIVIDADES REALIZADAS (Listado)
ADRIAN RAMIREZ GONZALEZ	Líder	Distribución de trabajo
	Técnico	Mejora estética de la interfaz.
		Implementación de estructuras de datos
SANTIAGO AVILA	Coordinador	Redactar documento
	Programador	Programar funciones
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.

XIII. DIFICULTADES Y LECCIONES APRENDIDAS

Por la parte del uso del hash, se tuvieron algunas dificultades a la hora de implementar el hashing como tal, ya que teníamos que tener en cuenta como era el hash para strings, la lección aprendida en este caso fue saber diferenciar entre el hashing para enteros y el hashing para strings.

Se tuvieron dificultades con los disjoints sets, ya que en la implementación vista en clase se nos enseñó a representar a los disjoints sets por medio de arreglos, en ese caso el índice del arreglo era el que reflejaba el objeto, por lo que solo podían ser objetos, y lo que tenía el arreglo reflejaba el conjunto al que pertenecía el elemento, al momento de implementarlo, como tenemos que almacenar el objeto "Usuarios", la solución fue crear otro arreglo para que almacenará que objeto usuario le correspondía cada índice, y la otra lista eran las relaciones y se usaban todos los algoritmos aprendidos.

XIV. REFERENCIAS BIBLIOGRÁFICAS

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