| Computación y estructuras discretas I |
| --- |
| Método de la ingeniería |
| Tarea Integradora I |
| David Dulce, Nicolas Cardona, Juan Esteban Eraso  1-10-2023 |

Tabla de contenido

[Paso 1. Identificación del Problema 2](#_heading=h.gjdgxs)

[**Paso 2. Recopilación de Información** 9](#_heading=h.30j0zll)

[**Paso 3. Búsqueda de Soluciones Creativas** 10](#_heading=h.1fob9te)

[**Paso 4. Transición de las Ideas a los Diseños Preliminares** 10](#_heading=h.3znysh7)

# Paso 1. Identificación del Problema

**-**El usuario requiere almacenar tareas y recordatorios .

-Se requiere una interfaz de usuario que permita a los usuarios agregar, modificar y eliminar tareas y recordatorios. Los usuarios pueden ver una lista de todas las tareas y recordatorios, ordenados por fecha límite o prioridad.

-Se requiere tener dos categorías para tareas,"Prioritaria"y "No prioritaria".

-Se requiere implementar un método que permita deshacer las acciones realizadas por un usuario en el sistema.

para efectuar todo se necesita Filtrar tareas por prioridad,Filtrar tareas por fecha

Ordenar tareas por fecha,Ordenar tareas por prioridad ,Ordenar tareas por estado.

resumiendo las necesidades se tiene que permitir:

-Crear tareas

-Editar tareas

-Eliminar tareas

-Mostrar lista de tareas por orden de llegada.

-Marcar tareas de prioridad

-Deshacer

| Cliente | Marlon |
| --- | --- |
| Usuario | Usuarios de el sistema de gestión de tareas y recordatorios |
| Requerimientos funcionales | RF0 - agregar tarea  RF1- mostrar lista de tareas  RF1.1 por orden de llegada  RF1.2 por orden de prioridad  RF2 – modificar tarea  RF3 - eliminar tarea  RF4 –deshacer acciones |
| Contexto del problema | Se requiere diseñar un sistema de gestión de tareas y recordatorios que permita a los usuarios agregar, organizar y administrar sus tareas pendientes y recordatorios. En este se almacenarán las tareas según su categoría: Prioritaria o no Prioritaria. |
| Requerimientos no funcionales | * Interfaz gráfica * Respuesta Rápida |

| Identificador y nombre | *[RF0-agregar tarea]* | | | |
| --- | --- | --- | --- | --- |
| Resumen | *el sistema permite al usuario agregar una tarea o recordatorio* | | | |
| Entradas | **Nombre entrada** | **Tipo de dato** | | **Condición valores válidos** |
| título | String | |  |
|  | descripción | String | |  |
|  | FechaLimite | Date | |  |
|  | Prioridad | int o char | | 1. No prioritaria  2. Prioritaria.  2.1 high  2.2 medium  2.3 low |
| Resultado o Postcondición | El programa almacena la tarea o recordatorio. | | | |
| Salidas | **Nombre salida** | | **Tipo de dato** | **Formato** |
| msg | | String | Se muestra un mensaje donde se confirma que se agregó la tarea |

| Identificador y nombre | *[RF1-mostrar lista de tareas]* | | | |
| --- | --- | --- | --- | --- |
| Resumen | *El sistema mostrará una lista de todas las tareas y recordatorios, ordenados por fecha límite o prioridad.* | | | |
| Entradas | **Nombre entrada** | **Tipo de dato** | | **Condición valores válidos** |
|  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
| Resultado o Postcondición | El programa muestra la lista de todas las tareas y recordatorios, ordenados por fecha límite o prioridad.  . | | | |
| Salidas | **Nombre salida** | | **Tipo de dato** | **Formato** |
|  | |  |  |

| Identificador y nombre | *[RF1.1-mostrar lista de tareas por orden de llegada]* | | | |
| --- | --- | --- | --- | --- |
| Resumen | *El sistema muestra al usuario las tareas no prioritarias por orden de llegada.* | | | |
| Entradas | **Nombre entrada** | **Tipo de dato** | | **Condición valores válidos** |
|  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
| Resultado o Postcondición | muestra la lista de tareas por orden de llegada. | | | |
| Salidas | **Nombre salida** | | **Tipo de dato** | **Formato** |
| msg | | String | Se muestra un mensaje donde se confirma que se agregó la tarea |

| Identificador y nombre | *[RF1.2-mostrar lista de tareas por orden de prioridad]* | | | |
| --- | --- | --- | --- | --- |
| Resumen | *El sistema muestra al usuario las tareas prioritarias por orden de prioridad..* | | | |
| Entradas | **Nombre entrada** | **Tipo de dato** | | **Condición valores válidos** |
|  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
| Resultado o Postcondición | muestra la lista de tareas por orden de prioridad. | | | |
| Salidas | **Nombre salida** | | **Tipo de dato** | **Formato** |
|  | |  |  |

| Identificador y nombre | *[RF2-modificar tarea]* | | | |
| --- | --- | --- | --- | --- |
| Resumen | *El sistema permitirá al usuario modificar una tarea antes agregada según los criterios que desee.* | | | |
| Entradas | **Nombre entrada** | **Tipo de dato** | | **Condición valores válidos** |
| titulo | String | |  |
|  | descripción | String | |  |
|  | prioridad | Enum | | 1. No prioritaria  2. Prioritaria.  2.1 high  2.2 medium  2.3 low |
|  |  |  | |  |
| Resultado o Postcondición | la tarea quedará modificada.  . | | | |
| Salidas | **Nombre salida** | | **Tipo de dato** | **Formato** |
| msg | | String | Mensaje confirmando la modificación de la tarea. |

| Identificador y nombre | *[RF3-eliminar tarea]* | | | |
| --- | --- | --- | --- | --- |
| Resumen | *El sistema permite al usuario eliminar una tarea.* | | | |
| Entradas | **Nombre entrada** | **Tipo de dato** | | **Condición valores válidos** |
| key | int | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
| Resultado o Postcondición | La tarea quedará eliminada.  . | | | |
| Salidas | **Nombre salida** | | **Tipo de dato** | **Formato** |
| msg | | String | Mensaje de confirmación de que la tarea ha sido eliminada. |

| Identificador y nombre | *[RF4-deshacer acciones]* | | | |
| --- | --- | --- | --- | --- |
| Resumen | *El sistema permite al usuario deshacer acciones* | | | |
| Entradas | **Nombre entrada** | **Tipo de dato** | | **Condición valores válidos** |
|  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
|  |  |  | |  |
| Resultado o Postcondición | El programa deshace las acciones del usuario.  . | | | |
| Salidas | **Nombre salida** | | **Tipo de dato** | **Formato** |
|  | |  |  |

## **Paso 2. Recopilación de Información**

**Para almacenar tareas y recordatorios**: se utiliza una tabla hash. La clave podría ser un identificador único y el valor podría ser la información de la tarea/recordatorio. Cada entrada en la tabla hash podría tener la siguiente información: título, descripción, fecha límite, prioridad, etc.

**Para Interfaz de Usuario**: Los usuarios pueden ver una lista de todas las tareas y recordatorios, ordenados por fecha límite o prioridad. Dado que las tareas y recordatorios estarán en una hash se usará heapsort para ordenar.

**HashTable:** Una Hashtable Java es una estructura de datos que utiliza una función hash para identificar datos mediante una llave. La función hash transforma una llave a un valor índice de un arreglo de elementos. En este caso a una índice de nuestra Hashtable Java.Se utiliza para almacenar y recuperar datos de manera eficiente ya que es muy útil cuando se necesita mapear claves a valores y realizar búsquedas eficientes por clave.

**Función Hash:**Una función hash es una función matemática que toma una entrada (o "clave") y la transforma en un valor numérico, generalmente una dirección en una tabla o un número que se utiliza para indexar en una estructura de datos.

**Heapsort**:Es un algoritmo de ordenamiento basado en comparaciones que se utiliza para ordenar elementos en una lista o arreglo. Es eficiente y tiene un tiempo de ejecución garantizado de O(n log n), lo que lo hace adecuado para grandes conjuntos de datos.

**Stack**:Un Stack (o pila) en Java es una estructura de datos de tipo LIFO (Last-In-First-Out), lo que significa que el último elemento que se añade es el primero en ser eliminado.Al implementar Stack se puede tener el método push para agregar elementos a ella. peek para ver el elemento en la parte superior de la pila sin eliminarlo, y pop para eliminar el elemento en la parte superior de la pila. También podemos verificar el tamaño de la pila con size y verificar si está vacía con isEmpty.

**Queue:** Una cola es una estructura de datos que sigue el principio FIFO (First-In-First-Out), entonces Permite almacenar objetos y luego recuperarlos en el orden en el cual se insertaron.También implementa métodos para agregar elementos a ella offer.peek para ver el elemento en la parte frontal de la cola sin eliminarlo y poll para eliminar el elemento en la parte frontal de la cola. También podemos verificar el tamaño de la cola con size y verificar si está vacía con isEmpty.

**Priorityqueue:** Una variante de una cola, la clase PriorityQueue. Cuando se agregan elementos a la cola se organiza según su valor.

### **Paso 3. Búsqueda de Soluciones Creativas**

**Alternativa 1:**

**HashTable, Stack,queue y Priorityqueue:**

El problema requiere que la búsqueda de tareas debe realizarse de la forma más rápida, por lo que la implementación de la tabla hash sería una de las mejores opciones porque la búsqueda tendría una complejidad de O(1). Con respecto a la priorización de tareas priorityqueue es muy buena alternativa para este problema Las PriorityQueue se organizan de manera que los elementos con la mayor prioridad se encuentren en la parte superior de la cola y sean los primeros en ser eliminados o accedidos. La prioridad generalmente se determina según un criterio específico.Además con la queue se puede trabajar en las no prioritarias que simplemente se hagan en orden de llegada (FIFO).En el caso de la Stack (LIFO) se implementa para el tema de la función de deshacer.

**Alternativa 2:**

Para llevar a cabo este sistema se podría utilizar lo siguiente:

Tabla hash, La tabla hash actúa como un almacén eficiente de datos(tareas).

Para implementar la tabla hash se usarán arrays de tamaño m bastante grande como para evitar colisiones en caso de presentarse colisiones se implementan linkedlist en el slot donde ocurrió la colisión.

También se podría utilizar arraylist para hacer la tabla hash.

Al implementar tabla hash podemos implementar heapsort Si se desea ordenar las tareas y recordatorios por fecha límite o prioridad este es excelente para reorganizar la lista de tareas en función de los criterios de orden deseados.

aparte de la tabla hash se utilizan estructuras auxiliares como lo son las pilas y las colas:

las pilas se usan para resolver el problema de deshacerse las acciones

las pilas, para mostrar tareas por orden de llegada

las priorityqueues para orden de prioridad.

#### **Paso 4. Transición de las Ideas a los Diseños Preliminares**

**Paso 5. Evaluación y Selección de la Mejor Solución.**

**Paso 6. Preparación de Informes y Especificaciones.**

**Paso 7. Implementación del Diseño.**