metodo de la ingenieria

**fase 1: identificacion del problema**

identificacion de necesidades y sintomas

* los usuarios del sistema de gestion de tareas y recordatorios requieren agregar, organizar y administrar sus tareas pendientes y recordatorios.
* una condicion bajo la cual debe ser resuelta es que a la hora de almacenar tareas y recordatorios se debe hacer por medio de tablas hash
* los usuarios podran ver una lista de todas las tareas y recordatorios y estas deben ser ordenadas por fecha de limite o prioridad
* se debe tener una funcionalidad de ordenar para utilizar un heapsort
* la solucion debe tener dos categorias para que el usuario realice las tareas: prioritaria que es para que realice las tarea segun su nivel de importancia permitiendo que las tareas importantes se realicen primero y tarea no prioritarias que permita manejar las tareas a las que no se les ha dado prioridad
* el usuario podra deshacer las acciones realizadas por el en el sistema, esto debe ser resuelto por medio de pilas.

**paso 2. recopilacion de informacion**

se hace una busqueda de las definiciones de los terminos que vamos a implementar en el problema planteado.

**complejidad temporal**

La complejidad temporal es el número de operaciones que realiza un algoritmo para completar su tarea (considerando que cada operación dura el mismo tiempo). El algoritmo que realiza la tarea en el menor número de operaciones se considera el más eficiente en términos de complejidad temporal. Sin embargo, la complejidad espacial y temporal se ve afectada por factores como el sistema operativo y el hardware, pero no los incluiremos en discusión.

<https://www.freecodecamp.org/espanol/news/introduccion-a-la-complejidad-temporal-de-los-algoritmos/>

**complejidad espacial**

Memoria que utiliza un programa para su ejecución, indica la cantidad de espacio requerido para ejecutar el algoritmo; es decir, el espacio en memoria que ocupan todas las variables propias al algoritmo.

<http://aniei.org.mx/paginas/uam/CursoAA/curso_aa_02.html#:~:text=Complejidad%20Espacial%3A%20Memoria%20que%20utiliza,las%20variables%20propias%20al%20algoritmo>.

**pilas**

Una pila (stack) es una colecciona ordenada de elementos en la cual los datos se insertan o se retiran por el mismo extremo llamado “parte superior” de la pila.

<https://ccc.inaoep.mx/ingreso/programacion/corto2015/Curso-PROPE-PyED-5-Pilas-Colas.pdf>

**colas**

Una cola (también llamada fila) es un tipo de dato abstracto, caracterizada por ser una secuencia de elementos en la que la operación de inserción push se realiza por un extremo y la operación de extracción pull por el otro. También se le llama estructura FIFO (del inglés First In First Out), debido a que el primer elemento en entrar será también el primero en salir.

<https://es.wikipedia.org/wiki/Cola_(inform%C3%A1tica)>

**hash tables**

Hash Table is a data structure which stores data in an associative manner. In a hash table, data is stored in an array format, where each data value has its own unique index value. Access of data becomes very fast if we know the index of the desired data.

<https://www.tutorialspoint.com/data_structures_algorithms/hash_data_structure.htm>

**big O**

Big O notation is one of the most fundamental tools for computer scientists to analyze the cost of an algorithm. It is a good practice for software engineers to understand in-depth as well.

<https://www.freecodecamp.org/news/big-o-notation-why-it-matters-and-why-it-doesnt-1674cfa8a23c/>

**parameterization**

Parameterization is the process of taking values or objects defined within a function or a method, and making them parameters to that function or method, in order to generalize the code. This process is also known as the “extract parameter” refactoring.

<https://www.toptal.com/python/python-parameterized-design-patterns#:~:text=Parameterization%20is%20the%20process%20of,the%20%E2%80%9Cextract%20parameter%E2%80%9D%20refactoring>.

**unit testing**

Unit testing is an essential instrument in the toolbox of any serious software developer. However, it can sometimes be quite difficult to know how to write unit tests for a particular piece of code. Having difficulty testing their own or someone else’s code, developers often think that their struggles are caused by a lack of some fundamental testing knowledge or secret unit testing techniques.

<https://www.toptal.com/qa/how-to-write-testable-code-and-why-it-matters>

**arrays**

An array is an arrangement of numbers, pictures or objects formatted into rows and columns according to their type. In coding and programming, an array is a collection of items, or data, stored in contiguous memory locations, also known as database systems. The purpose of an array is to store multiple pieces of data of the same type together. You can use an array to demonstrate a mathematical property known as the commutative property of multiplication, which illustrates that you can change the order of the factors or elements, and the product of the elements remains the same.

In computer programming, an array can help you locate and identify where you stored each piece of data, or element, by adding an offset to each value. An offset is a number that represents the difference between the two indexes. Similar to an index in a book, an index in computer programming contains a record of entries with the names of the data items and their locations. Thus, you can identify each element and its location by referring to the index within the array.

**generics**

Generic programming is a style of computer programming in which algorithms are written in terms of types to-be-specified-later that are then instantiated when needed for specific types provided as parameters.

**arraylist**

The ArrayList class is a resizable array, which can be found in the java.util package. The difference between a built-in array and an ArrayList in Java, is that the size of an array cannot be modified (if you want to add or remove elements to/from an array, you have to create a new one).

<https://www.w3schools.com/java/java_arraylist.asp#:~:text=The%20ArrayList%20class%20is%20a,to%20create%20a%20new%20one>).

**heapsort**

Heap sort is a comparison-based sorting technique based on Binary Heap data structure. It is similar to the selection sort where we first find the minimum element and place the minimum element at the beginning. Repeat the same process for the remaining elements.

<https://www.geeksforgeeks.org/heap-sort/>

**Paso 3. Búsqueda de Soluciones Creativas**

dividirlo en subsistemas y para cada subsistema vamos a considerar diferentes formas de abordarlo por ejemplo que tipo de estructuras, y metodos que serian necesarios.

vamos a abordar el problema dividendo en tres partes que serian las principales, tenemos lo siguiente:

* **almacenar las tareas**

tablas hash

array list

array

lista enlazada

fila

cola

* **gestion de tareas prioritarias**

cola

pilas

tablas hash

pilas fifo

* **gestion de acciones del usuario**

pilas lifo

pilas

colas

listas enlazadas

arboles

para la parte del almacenamiento de las tareas se sobreentiende que necesitamos una estructura de almacenamiento la cual puede ser arreglos, arraylist, listas enlazadas y hash tables.

para la getion de tareas prioritarias podriamos utilizar arraylist

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

aqui vamos a dividir en secciones y con base a esto eliminar las ideas menos factibles.

1. **Almacenar las tareas:**

Tablas Hash: Ofrecen acceso rápido y eficiente a las tareas si se pueden asociar con claves únicas, pero pueden consumir más memoria.

ArrayList: Una lista dinámica que proporciona acceso rápido a los elementos por índice, pero no es tan eficiente para eliminar o insertar elementos en posiciones intermedias.

Array: Similar a ArrayList, pero con tamaño fijo. Puede ser adecuado si se conoce el número máximo de tareas por adelantado.

Lista Enlazada: Eficiente para insertar y eliminar tareas en cualquier posición, pero puede ser menos eficiente para acceder a elementos en posiciones aleatorias.

Fila (Queue): Útil si se necesita mantener un orden estricto en el procesamiento de tareas y seguir el principio FIFO (primero en entrar, primero en salir).

Cola de prioridad: Si las tareas tienen prioridades, una cola de prioridad podría ser adecuada para asegurar que las tareas prioritarias se manejen primero.

2. **Gestión de tareas prioritarias:**

Cola (Queue): Se puede usar una cola para gestionar las tareas prioritarias en base al principio FIFO.

Pilas (Stack): Si se necesita manejar tareas prioritarias con el principio LIFO (último en entrar, primero en salir), una pila sería apropiada.

Tablas Hash: Se puede usar una tabla hash si se necesita acceso rápido a las tareas prioritarias según sus claves.

3. **Gestión de acciones del usuario:**

Pilas (Stack): Si es necesario rastrear las acciones del usuario en orden inverso (LIFO), una pila LIFO podría ser útil.

Colas (Queue): Si necesitamos registrar las acciones del usuario en el orden exacto en que ocurrieron (FIFO), una cola sería apropiada.

Listas Enlazadas: se pueden utilizar listas enlazadas para mantener un registro ordenado de las acciones del usuario.

Árboles: Si se necesitauna estructura de datos más compleja para representar la secuencia de acciones del usuario, como un árbol para navegar entre acciones.

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

este paso se hara con el fin de encontrar la mejor solucion para implementar, crearemos un sistema de evaluacion numerico basandonos en los siguientes criterios:

1. eficiencia
2. usabilidad
3. mantenibilidad
4. escalabilidad

enumerando cada punto del 1 al 5, siendo 1 muy malo y 5 muy bueno. al final de cada caso se hara una suma y entre mas alto sea el valor mas conveniente de utilizar sera.

**Para el subsistema de Almacenar las Tareas:**

Tablas Hash:

Eficiencia: Ofrece un acceso rápido y eficiente a las tareas. 5

Usabilidad: Puede requerir claves únicas y consume más memoria. 4

Mantenibilidad: Mantenimiento razonable. 4

Escalabilidad: Escalable para un gran número de tareas. 5

total: 18

ArrayList:

Eficiencia: Acceso rápido por índice, menos eficiente para inserciones y eliminaciones intermedias. 3

Usabilidad: Fácil de usar y comprender. 4

Mantenibilidad: Mantenimiento razonable. 4

Escalabilidad: Escalable, pero no es la mejor opción para grandes cantidades de datos en constante cambio. 3

total: 14

Array:

Eficiencia: Acceso rápido, tamaño fijo. 3

Usabilidad: Limitado por el tamaño fijo. 2

Mantenibilidad: Mantenimiento razonable. 4

Escalabilidad: No es adecuado para un número variable de tareas. 2

total: 12

Lista Enlazada:

Eficiencia: Eficiente para inserciones y eliminaciones en cualquier posición. 4

Usabilidad: Un poco más compleja de usar que ArrayList. 3

Mantenibilidad: Mantenimiento razonable. 4

Escalabilidad: Adecuada para un número variable de tareas. 4

total: 15

Fila (Queue):

Eficiencia: Útil para mantener un orden estricto (FIFO). 4

Usabilidad: Útil para tareas que deben procesarse en orden de llegada. 4

Mantenibilidad: Mantenimiento razonable. 4

Escalabilidad: Escalable para gestionar tareas en cola. 4

total: 16

Cola de Prioridad:

Eficiencia: Eficiente para gestionar tareas prioritarias. 4

Usabilidad: Excelente para tareas con prioridades. 5

Mantenibilidad: Mantenimiento razonable. 4

Escalabilidad: Escalable y adecuada para priorizar tareas. 4

total: 17

**Para el subsistema de Gestión de Tareas Prioritarias:**

Cola (Queue):

Eficiencia: Útil para gestionar tareas prioritarias en base a FIFO. 5

Usabilidad: Fácil de usar. 4

Mantenibilidad: Mantenimiento razonable. 4

Escalabilidad: Escalable para gestionar tareas prioritarias. 4

total: 17

Pilas (Stack):

Eficiencia: Útil para gestionar tareas prioritarias en base a LIFO. 4

Usabilidad: Fácil de usar. 4

Mantenibilidad: Mantenimiento razonable. 4

Escalabilidad: Escalable para gestionar tareas prioritarias. 4

total 16

Tablas Hash:

Eficiencia: Ofrece acceso rápido a tareas prioritarias según claves. 4

Usabilidad: Puede requerir claves únicas. 3

Mantenibilidad: Mantenimiento razonable. 4

Escalabilidad: Escalable para gestionar tareas prioritarias. 4

total: 15

Cola de Prioridad:

Eficiencia: Eficiente para gestionar tareas prioritarias. 4

Usabilidad: Excelente para tareas con prioridades. 5

Mantenibilidad: Mantenimiento razonable. 4

Escalabilidad: Escalable y adecuada para priorizar tareas. 4

total: 17

**Para el subsistema de Gestión de Acciones del Usuario:**

Pilas (Stack):

Eficiencia: Útil para rastrear acciones del usuario en orden inverso (LIFO). 4

Usabilidad: Fácil de usar. 4

Mantenibilidad: Mantenimiento razonable. 4

Escalabilidad: Escalable para registrar acciones del usuario. 4

total: 16

Colas (Queue):

Eficiencia: Útil para registrar acciones del usuario en el orden exacto en que ocurrieron (FIFO). 4

Usabilidad: Fácil de usar. 4

Mantenibilidad: Mantenimiento razonable. 4

Escalabilidad: Escalable para registrar acciones del usuario. 4

total: 16

Listas Enlazadas:

Eficiencia: Eficiente para mantener un registro ordenado de acciones del usuario. 4

Usabilidad: Razonablemente fácil de usar. 3

Mantenibilidad: Mantenimiento razonable. 4

Escalabilidad: Adecuada para registrar acciones del usuario. 4

total: 15

Árboles:

Eficiencia: Potencialmente eficiente para representar la secuencia de acciones del usuario. 3

Usabilidad: Más complejo de implementar y usar en comparación con otras opciones. 2

Mantenibilidad: Mantenimiento razonable. 4

Escalabilidad: Escalable para estructuras de acciones más complejas. 3

total. 13

con base en la evaluacion anterior, determinamos que para:

**Almacenar las Tareas:** utilizaremos tablas hash ya que con esto podemos utilizar claves y con estas desbloquear un valor, es decir podemos utilizar una clave con un identificador unico, y con esta ver la informacion de la tarea/ recordatorio.

**Gestión de Tareas Prioritarias:** para este punto utilizaremos colas y colas de prioridad esto con el fin de organizar las tareas segun su nivel de importancia y las colas con el fin de tratar las tareas no prioritarias y gestionarlas de acuerdo al orden de llegada con la funcion FIFO.

**Gestión de Acciones del Usuario:** para este paso utilizaremos pilas y quiza tambien colas, debido a que al usar una pila podremos utilizar la funcion LIFO para realizar un seguimiento de las acciones que ya se han hecho, al igual que para implementar la funcion deshacer

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

**Paso 7. Implementación del Diseño**