EL MÉTODO DE DISEÑO EN INGENIERÍA

FASE 1: IDENTIFICACIÓN DEL PROBLEMA

Definición del problema

//Se requiere un sistema de gestión de tareas y recordatorios que permita a los usuarios agregar, organizar y administrar sus tareas pendientes y recordatorios.//

Se identificó la necesidad de un sistema de gestión de tareas y recordatorios para ayudar a los usuarios a organizar sus responsabilidades diarias. Esta necesidad se basa en la dificultad que muchos enfrentan para llevar un registro efectivo de sus tareas pendientes y recordatorios importantes.

Identificación de necesidades y síntomas

El sistema de gestión debe contener los siguientes componentes y funcionalidades:

- Los usuarios podrán al almacenar tareas y recordatorios
- La interfaz de usuario debe permitir mostrar a los usuarios las acciones de agregar, modificar y eliminar tareas y recordatorios
- Las tareas se deben clasificar en dos categorías: “prioritarias”,”no prioritarias”.
- Crear pilas que contenga las acciones realizadas por el usuario para hacer seguimiento al usuario
- El usuario debe estar en la capacidad de deshacer una acción

FASE 2: RECOPILACIÓN DE LA INFORMACIÓN NECESARIA
BÚSQUEDA

Tras investigar los sistemas de gestión de tareas existentes, hemos identificado varios tipos. Los más comunes son las aplicaciones de listas de tareas, como Todoist y Wunderlist, que permiten a los usuarios crear, organizar y completar tareas. También existen aplicaciones de tablero Kanban, como Trello, que se centran en la organización visual de tareas en columnas. Asimismo, aplicaciones más avanzadas, como Asana y Jira, ofrecen características para equipos y proyectos colaborativos.

Mejores Prácticas en el Campo de gestión de tareas:

Durante nuestra investigación, encontramos algunas mejores prácticas clave en el campo de sistemas de gestión de tareas:

Sencillez y Usabilidad: Los sistemas deben ser intuitivos y fáciles de usar para los usuarios.
Priorización: La capacidad de priorizar tareas es fundamental para ayudar a los usuarios a centrarse en lo más importante.
Recordatorios: La función de recordatorios es esencial para no olvidar las tareas pendientes.

Colaboración: Si bien nuestro sistema inicialmente se centra en usuarios individuales, la capacidad de colaboración podría ser una característica a considerar a futuro.

Herramientas y Tecnologías Disponibles:

En cuanto a las herramientas y tecnologías disponibles para implementar nuestro sistema, hemos investigado y considerado varias opciones:

Base de Datos: Para almacenar las tareas y recordatorios, podríamos utilizar bases de datos relacionales como MySQL o sistemas NoSQL como MongoDB.

Desarrollo Frontend: Para la interfaz de usuario, tecnologías web como HTML, CSS y JavaScript son esenciales. Frameworks como React o Vue.js pueden acelerar el desarrollo(Claro está que para este proyecto no usaremos interfaz gráfica avanzada).

Desarrollo Backend: Para la lógica del servidor, podríamos utilizar lenguajes como Python (con Flask o Django), Node.js (con Express.js), o Ruby on Rails.

Seguridad: La seguridad de los datos es crucial. Implementaremos prácticas de seguridad, como encriptación y autenticación de usuarios.

Método de "Deshacer": Implementaremos la funcionalidad de "deshacer" utilizando una pila (LIFO) para rastrear las acciones del usuario.

FASE 3: BÚSQUEDA DE SOLUCIONES CREATIVAS

1. Almacenamiento de Tareas y Recordatorios en una Tabla Hash:

Para almacenar las tareas y recordatorios en Java, utilizamos una estructura de datos `HashMap`. Cada tarea se puede representar como un objeto de la clase `Task`. La clave en el mapa podría ser el título de la tarea, que debe ser único para cada tarea. La información detallada de la tarea se almacena como el valor correspondiente en el mapa.

```
import java.util.HashMap;

// Crear una tabla hash para almacenar tareas
HashMap<String, Task> taskMap = new HashMap<>();

// Agregar una tarea
Task newTask = new Task("Título", "Descripción", "Fecha límite", "Prioridad");
taskMap.put("Título", newTask);

// Para acceder a una tarea específica
Task task = taskMap.get("Título");
```

2. Cola de Prioridades en Java:

Para organizar tareas prioritarias, utilizamos una “PriorityQueue” en Java. La “PriorityQueue” se basa en la prioridad especificada al crear una tarea. Las tareas se ordenarán automáticamente en función de esta prioridad.

```java
import java.util.PriorityQueue;

// Crear una cola de prioridades para tareas prioritarias
PriorityQueue<Task> priorityQueue = new PriorityQueue<>();

// Agregar una tarea prioritaria
Task highPriorityTask = new Task("Alta Prioridad", ...);
priorityQueue.add(highPriorityTask);

// Para obtener la tarea más prioritaria
Task nextTask = priorityQueue.poll();
```

3. Gestión FIFO en Java:

Para las tareas no prioritarias, puedes utilizar una `LinkedList`. Las tareas se agregan al final de la lista y se manejan en el orden en que se agregaron.

```
import java.util.LinkedList;

// Crear una lista enlazada para tareas no prioritarias
LinkedList<Task> fifoList = new LinkedList<>();

// Agregar una tarea no prioritaria
Task nonPriorityTask = new Task("Tarea No Prioritaria", ...);
fifoList.add(nonPriorityTask);

// Para obtener y eliminar la siguiente tarea no prioritaria (FIFO)
Task nextTask = fifoList.poll();
```

4. Implementar la Función "Deshacer" en Java:

Para implementar la función de "Deshacer," utilizamos una pila (`Stack`) para realizar un seguimiento de las acciones del usuario. Cada vez que el usuario realice una acción (agregar, modificar o eliminar una tarea), registramos la acción en la pila. Luego, implementamos un método para deshacer la última acción realizada.

```
import java.util.Stack;

// Crear una pila para seguimiento de acciones
```

```
Stack<UserAction> actionStack = new Stack<>();

// Registra una acción (por ejemplo, agregar tarea)
UserAction action = new UserAction("Agregar Tarea", taskDetails);
actionStack.push(action);

// Implementar un método para deshacer
public void undoLastAction() {
    if (!actionStack.isEmpty()) {
        UserAction lastAction = actionStack.pop();
        lastAction.undo(); // Revertir la acción, por ejemplo, eliminar la tarea recién agregada
    }
```

FASE 4: TRANSICIÓN DE LA FORMULACIÓN DE IDEAS A LOS DISEÑOS PRELIMINARES

1. Almacenamiento de Tareas y Recordatorios con Tabla Hash:
Hemos decidido utilizar una tabla hash (HashMap en Java) para almacenar tareas y recordatorios. Esto proporciona una manera eficiente de acceder y administrar estas tareas. Cada tarea se almacena con un título único como clave y la información detallada como valor.

2. Organización de Tareas Prioritarias con Cola de Prioridades:
Para las tareas prioritarias, hemos optado por una cola de prioridades (PriorityQueue en Java). Esto permite que las tareas importantes se manejen primero según su nivel de importancia. Las tareas se ordenarán automáticamente en función de su prioridad.

3. Funcionalidad de Ordenar Utilizando Heapsort:
Hemos decidido agregar una funcionalidad de ordenar utilizando el algoritmo Heapsort. Esto permitirá a los usuarios ordenar la lista de tareas y recordatorios almacenados según su fecha límite o prioridad.

FASE 5: EVALUACIÓN Y SELECCIÓN DE LA MEJOR SOLUCIÓN

Usabilidad: La usabilidad es esencial para que nuestros usuarios se sientan cómodos y eficaces al utilizar la aplicación. Nuestra interfaz de usuario permitirá a los usuarios agregar, modificar y eliminar tareas de manera intuitiva. Además, hemos incorporado la funcionalidad de "deshacer" para que los usuarios puedan corregir errores de manera sencilla.

Gestión de Tareas Prioritarias y No Prioritarias: Hemos abordado la necesidad de diferenciar entre tareas prioritarias y no prioritarias. Nuestra cola de prioridades garantiza que las tareas importantes se gestionen primero, mientras que las no prioritarias siguen el principio de gestión FIFO.

Método de "Deshacer" Utilizando una Pila (LIFO): Hemos implementado un sólido método de "deshacer" que utiliza una pila (LIFO) para rastrear las acciones de los usuarios. Cada

acción realizada se registra en la pila, lo que permite una reversión efectiva de las acciones en caso de errores. Cuando un usuario selecciona la opción de "Deshacer", el sistema desapila la última acción y revierte la acción correspondiente.

FASE 6: PREPARACIÓN DE INFORMES Y ESPECIFICACIONES

Arquitectura del Sistema:

Nuestro sistema de gestión de tareas y recordatorios está diseñado de manera eficiente y consta de varios componentes clave:

Almacenamiento de Datos: Hemos implementado una tabla hash para almacenar tareas y recordatorios. Cada entrada en la tabla hash contiene información esencial, como título, descripción, fecha límite y prioridad. Utilizamos una cola de prioridades para gestionar tareas prioritarias y no prioritarias.

Interfaz de Usuario: La interfaz de usuario es intuitiva y permite a los usuarios realizar las siguientes acciones:

Agregar nuevas tareas y recordatorios.
Modificar detalles de tareas y recordatorios existentes.
Eliminar tareas y recordatorios.
Ver una lista de todas las tareas y recordatorios, ordenados por fecha límite o prioridad.
Ordenar tareas y recordatorios utilizando el algoritmo Heapsort.
Gestión de Prioridades: Hemos diferenciado tareas en dos categorías: "Prioritarias" y "No prioritarias." Las tareas prioritarias se manejan mediante una cola de prioridades, garantizando que las más importantes se gestionen primero. Las tareas no prioritarias siguen el principio FIFO.

Función de "Deshacer": Hemos implementado una funcionalidad de "deshacer" que permite a los usuarios revertir acciones no deseadas. Cada acción realizada se registra en una pila (LIFO). Cuando se selecciona la opción de "Deshacer," el sistema desapila la última acción y revierte la acción correspondiente.

Especificaciones Técnicas:

Lenguaje de Programación: Hemos desarrollado el sistema en Java, aprovechando su eficiencia y capacidades en la manipulación de estructuras de datos.

Almacenamiento de Datos: Utilizamos una tabla hash para el almacenamiento de tareas y recordatorios, garantizando un acceso eficiente a los elementos. La cola de prioridades se ha implementado para gestionar tareas prioritarias.

Ordenamiento: Para ordenar las tareas y recordatorios, hemos aplicado el algoritmo Heapsort, que es altamente eficiente en la clasificación de grandes conjuntos de datos.

Interfaz de Usuario:

La interfaz de usuario es amigable y permite a los usuarios interactuar con el sistema de manera sencilla. Las funciones clave se presentan de manera intuitiva para garantizar una experiencia de usuario satisfactoria.

Función de "Deshacer":

Los usuarios pueden deshacer acciones no deseadas de la siguiente manera:

En la interfaz de usuario, seleccionan la opción "Deshacer."
El sistema desapila la última acción registrada en la pila (LIFO).
La acción se revierte, lo que restaura el estado previo de la tarea o recordatorio afectado.
Esto garantiza que los usuarios tengan un control completo sobre las acciones que realizan y puedan corregir rápidamente cualquier error.

Conclusiones:

Nuestro sistema de gestión de tareas y recordatorios ofrece una solución eficiente y versátil para ayudar a los usuarios a organizar sus actividades. La combinación de una tabla hash para el almacenamiento de datos, una cola de prioridades para la gestión de tareas, el algoritmo Heapsort para el ordenamiento y la función de "deshacer" proporciona una experiencia de usuario robusta y satisfactoria. Este informe técnico proporciona una visión general de la arquitectura y las características clave del sistema.

FASE 7: IMPLEMENTACIÓN DEL DISEÑO

En esta etapa de implementación, estamos construyendo nuestro Sistema de Gestión de Tareas y Recordatorios, basándonos en los diseños y especificaciones que hemos definido anteriormente.

Hemos trabajado duro en la estructura subyacente del sistema. Hemos utilizado una tabla hash para almacenar las tareas y los recordatorios, y creamos una cola de prioridades para las tareas importantes. También implementamos la función de "deshacer" utilizando una pila (LIFO), lo que permitirá a los usuarios revertir acciones si cometen errores.

En esta fase, estamos enfocados en escribir el código que permitirá a los usuarios realizar acciones clave, como agregar, modificar y eliminar tareas y recordatorios. También estamos trabajando en la funcionalidad de ordenar utilizando el algoritmo Heapsort.

Estamos realizando pruebas exhaustivas para asegurarnos de que el sistema funcione sin problemas y sea fácil de usar. Queremos que sea intuitivo para nuestros usuarios y que cumpla con sus expectativas. Estamos verificando cómo se almacenan y recuperan los datos, que las tareas prioritarias se gestionen correctamente y que la función de "deshacer" funcione de manera precisa.

Nuestra meta es proporcionar a nuestros compañeros estudiantes una herramienta efectiva para administrar sus tareas y recordatorios diarios. Queremos que este sistema mejore su productividad y organización personal. Continuaremos trabajando arduamente en el desarrollo y las pruebas para lograr esto.

Además, recordemos que después de la implementación, continuaremos manteniendo y mejorando el sistema en el futuro para asegurarnos de que siga siendo útil y eficaz. Estamos comprometidos con este proyecto y con brindar la mejor solución posible.