Paso 1. identificación del problema (Dar un contexto, Plantear las necesidades y síntomas, definición del problema. )

La empresa TaskMaster busca nuestra colaboración en el desarrollo de una aplicación diseñada para ayudar a las personas en la gestión de sus tareas y actividades cotidianas. La iniciativa surge de la observación de que muchas personas experimentan dificultades al priorizar sus responsabilidades diarias, lo que puede llevar a la negligencia de eventos importantes, como citas, reuniones de negocios o cumpleaños.

En este contexto, la gestión efectiva de las actividades se basa en dos aspectos fundamentales: el orden cronológico de los eventos y su nivel de prioridad. Además, es esencial considerar el perfil de los usuarios a quienes se dirige la aplicación, ya que las necesidades de un estudiante universitario pueden diferir significativamente de las de un profesional que debe administrar sus tareas laborales diarias.

Para abordar estas necesidades, hemos identificado una serie de características y funcionalidades esenciales que deben formar parte de la aplicación:

**Almacenamiento de tareas y recordatorios:** La capacidad de registrar y almacenar tareas y recordatorios, con información detallada, como título, descripción, fecha límite y nivel de prioridad. Para ello, consideramos la utilización de una tabla hash que permita una rápida recuperación de datos.

**Interfaz de usuario intuitiva:** Una interfaz de usuario amigable que permita a los usuarios agregar, modificar o eliminar tareas y recordatorios de manera sencilla. Esta interfaz también deberá proporcionar una vista ordenada de las tareas, permitiendo la clasificación por fecha límite o prioridad, y se incluirá la funcionalidad de ordenar mediante el algoritmo de heapsort.

**Gestión de prioridades:** La capacidad de categorizar las tareas en "Prioritarias" y "No prioritarias" para ayudar a los usuarios a enfocarse en las tareas más importantes. Para ello, se implementará una cola de prioridades.

**Enfoque en tareas no prioritarias:** La inclusión de una categoría para tareas no prioritarias, que se gestionan en función de su orden de llegada (FIFO), para evitar la acumulación de responsabilidades de menor urgencia.

**Método de deshacer acciones:** La implementación de un método que permita a los usuarios deshacer la última acción realizada en el sistema. Esto se logrará mediante el uso de una pila (LIFO) que registre las acciones del usuario y facilite la reversión de acciones no deseadas.

Este conjunto de características es fundamental para abordar las necesidades iniciales de la aplicación y sentar las bases para una gestión efectiva de tareas y recordatorios. A medida que avancemos en el desarrollo, consideraremos cómo adaptar la interfaz para diferentes perfiles de usuarios, garantizando así una experiencia personalizada y eficiente.

**Necesidades y Síntomas identificados:**

**Acumulación de Tareas:** Los individuos a menudo enfrentan una acumulación constante de tareas y recordatorios a medida que avanzan en su día. Esto puede dar lugar a una sensación de desorden y a la posibilidad de que se pasen por alto tareas importantes.

**Priorización Difícil:** Distinguir entre tareas prioritarias y no prioritarias puede ser un desafío, lo que lleva a una gestión ineficiente del tiempo.

**Falta de Seguimiento:** Las personas pueden olvidar realizar tareas importantes o asistir a eventos programados debido a la falta de un sistema de recordatorios eficaz.

**Errores de Gestión:** Los usuarios pueden cometer errores al agregar, modificar o eliminar tareas, lo que puede resultar en la pérdida de datos o la confusión en la gestión de tareas.

**Necesidad de Flexibilidad:** Los usuarios necesitan la capacidad de deshacer acciones en caso de errores o cambios en la gestión de una tarea o recordatorio.

**Definición del Problema:**

El problema fundamental radica en la falta de una herramienta efectiva de gestión de tareas y recordatorios que satisfaga las necesidades de organización y priorización de los usuarios. Las soluciones actuales carecen de una interfaz intuitiva, una forma efectiva de priorizar tareas y recordatorios, y la capacidad de deshacer acciones en caso de errores. Además, no proporcionan una forma sistemática de almacenar, organizar y recuperar tareas y recordatorios. Por lo tanto, se requiere el desarrollo de un sistema de gestión de tareas y recordatorios que aborde estos desafíos y proporcione una solución integral para la organización y administración de tareas y recordatorios de manera eficiente.

**Requisitos Funcionales:**

**Registro de Tareas y Recordatorios:**

* El sistema debe permitir a los usuarios agregar nuevas tareas y recordatorios.
* Cada tarea debe tener al menos un título y una descripción.
* Los usuarios pueden establecer una fecha límite y hora para las tareas y recordatorios.
* Debe ser posible asignar un nivel de prioridad (alta, media, baja) a cada tarea.

**Interfaz de Usuario Intuitiva:**

* La interfaz de usuario debe ser fácil de usar y accesible tanto en dispositivos móviles como en la web.
* Los usuarios deben poder ver una lista de tareas y recordatorios en una vista organizada y ordenada.
* La interfaz debe permitir a los usuarios modificar, marcar como completadas o eliminar tareas y recordatorios de manera sencilla.

**Gestión de Prioridades:**

* Las tareas se deben categorizar por el usuario como "Prioritarias" o "No prioritarias" en función de su nivel de prioridad.
* Los usuarios pueden cambiar la categorización de una tarea si lo desean.

**Tareas No Prioritarias:**

* Debe existir una categoría especial para las tareas no prioritarias.
* Las tareas no prioritarias se gestionan en función de su orden de llegada (FIFO).

**Ordenamiento y Búsqueda:**

* Los usuarios pueden ordenar la lista de tareas y recordatorios por fecha límite, prioridad o título.
* Debe existir una función de búsqueda que permita a los usuarios encontrar tareas específicas utilizando palabras clave.

**Método de Deshacer Acciones:**

* Los usuarios pueden deshacer la última acción realizada en el sistema, como agregar, modificar o eliminar una tarea.
* La función de deshacer debe ser fácilmente accesible desde la interfaz.

**Notificaciones:**

* Los usuarios deben recibir notificaciones o alertas por correo electrónico, mensajes de texto o notificaciones push en la aplicación para recordarles las tareas pendientes antes de su fecha límite.

**Requisitos No Funcionales:**

**Rendimiento:**

La aplicación debe ser rápida y eficiente en la recuperación y visualización de tareas y recordatorios, incluso cuando la lista es extensa.

**Disponibilidad:**

La aplicación debe estar disponible para los usuarios en todo momento.

**Compatibilidad:**

La aplicación debe ser compatible con una variedad de navegadores web y dispositivos móviles populares.

**Escalabilidad:**

La aplicación debe ser escalable para manejar un crecimiento en la cantidad de usuarios y datos.

**Documentación:**

Debe proporcionarse documentación clara para usuarios finales sobre cómo utilizar la aplicación.

**Desacoplamiento o facilidad de mantenimiento para futuros cambios:**

Debe ser fácil para el equipo de desarrollo realizar actualizaciones y mejoras en la aplicación.

Paso 2. recopilacion de informacion (Recopilar información de la estructura de datos que se va a emplear)

**Tabla Hash:** Una tabla hash es una estructura de datos que asocia claves (identificadores únicos) con valores (información relacionada). Permite un acceso rápido a los datos mediante una función de hash que transforma la clave en una dirección en la tabla donde se almacena el valor correspondiente. Al implementar la tabla hash, se debe considerar el tamaño inicial y la capacidad de expansión de la tabla. En este caso es utilizar una tabla de tamaño variable que se expande automáticamente cuando se alcanza cierta carga. Sobre las colisiones de la tabla Hash se tiene pensado de dos formas, como encadenamiento o direccionamiento abierto.

**Algoritmo de Heapsort:** Heapsort es un algoritmo de ordenación que utiliza una estructura de datos llamada montículo (heap) para organizar los elementos y luego extraerlos en orden ascendente o descendente. Es eficiente y adecuado para ordenar grandes conjuntos de datos. En este caso, implementar el algoritmo de heapsort de manera eficiente. Asegurando que la operación de extracción de elementos del montículo (heap) se realice correctamente para obtener tareas ordenadas por fecha límite o prioridad.

**Cola de Prioridades:**  Utilizar una cola de prioridades mínima para tareas prioritarias y una cola de prioridades máxima para tareas no prioritarias. Ya que una cola de prioridades es una estructura de datos que almacena elementos con un valor de prioridad asociado. Permite insertar elementos en función de su prioridad y recuperar el elemento de mayor o menor prioridad de manera eficiente.

**FIFO** (First-In, First-Out): FIFO es un principio de organización en el que el primer elemento que entra en una estructura o cola es el primero en ser eliminado o procesado. Se utiliza comúnmente para gestionar tareas en el orden en que se reciben. Las tareas se deben encolar en el orden en que llegan y desencolar de manera similar.

**LIFO** (Last-In, First-Out): Cuando los usuarios deseen deshacer una acción, revierte la última acción realizada mediante la pila (LIFO) de acciones. Asegurando que las acciones se deshagan de manera completa y precisa. LIFO es otro principio de organización en el que el último elemento que entra en una estructura o pila es el primero en ser eliminado o procesado. Se planea utilizar para implementar la funcionalidad de "deshacer".

**Interfaz de Usuario** (UI): La interfaz de usuario se refiere a la parte de una aplicación o sistema que permite la interacción entre el usuario y la máquina. Incluye elementos visuales y funcionales que facilitan la comunicación y la realización de acciones por parte del usuario. Para el funcionamiento pensado de la aplicación se espera que las personas que la usen tenga la capacidad de comprender el sistema debido a la intuición del módulo.

**Notificación Push:** Una notificación push es un mensaje que se envía de manera instantánea a través de la aplicación a un dispositivo móvil o una computadora. Se utiliza para informar a los usuarios sobre eventos o actualizaciones relevantes de la aplicación, y se podría representar como la opción de configurar sus preferencias de notificación, como la frecuencia y el tipo de notificaciones que desean recibir según sus fechas y prioridad.

Paso 3. Búsqueda de soluciones (Por cada estructura de datos se plantea una solución para el problema en general)

**Tabla Hash:**

* **Implementación de la Tabla Hash:** Utiliza una tabla hash para almacenar tareas y recordatorios. Cada entrada en la tabla hash podría tener la siguiente información: título, descripción, fecha límite, prioridad, etc.
* **Función de Hashing:** Diseña una función de hash eficiente que pueda convertir el identificador único de cada tarea en una dirección en la tabla hash.
* **Colisión de Hash:** Implementa una estrategia para manejar colisiones de hash, como la resolución por encadenamiento o el rehashing.

**Algoritmo de Heapsort:**

* **Aplicación de Heapsort:** Utiliza el algoritmo de heapsort para ordenar las tareas por fecha límite o prioridad. Construye un montículo (heap) de tareas y extrae elementos en el orden deseado.
* **Mantenimiento del Heap:** Aseguramiento de que el montículo se mantenga correctamente mientras los usuarios agregan, modifican o eliminan tareas.

**Cola de Prioridades:**

* **Categorización de Tareas:** Implementa una cola de prioridades para categorizar las tareas como "Prioritarias" o "No prioritarias" en función de su nivel de importancia.
* **Gestión de Cambios de Prioridad:** Permite a los usuarios cambiar la categorización de una tarea si su nivel de prioridad cambia.

**FIFO (First-In, First-Out):**

* **Tareas No Prioritarias:** Utiliza el principio FIFO para gestionar las tareas no prioritarias. A medida que los usuarios agregan tareas no prioritarias, colócalas en una cola simple y procesa las tareas en el orden en que llegaron.

**LIFO (Last-In, First-Out):**

* **Método de Deshacer:** Implementa una pila (LIFO) para realizar un seguimiento de las acciones realizadas por los usuarios. Cada vez que un usuario agrega, modifica o elimina una tarea, registra la acción en la pila.

**Interfaz de Usuario (UI):**

* **Diseño Intuitivo:** Crea una interfaz de usuario amigable que permita a los usuarios agregar, modificar y eliminar tareas y recordatorios de manera sencilla.
* **Visualización de Tareas:** Proporciona una vista ordenada de las tareas, permitiendo a los usuarios ordenarlas por fecha límite o prioridad. Implementa la funcionalidad de ordenar utilizando el algoritmo de heapsort.

**Arraylist:**

* Implementa un arraylist para guardar la información de los recordatorios y actividades.

**Notificación Push:**

* Integración de Notificaciones: Diversos formatos en los que se podría presentar un mensaje emergente o avisos según el editor de código implementado.

Paso 4. Diseños preliminares(Descartar las soluciones que no están aptas con el modelo)

**Diseño de la Tabla Hash:**

* **Implementación de la Tabla Hash:** Con base a una clase o estructura de datos que represente la tabla hash. Se debería incluir métodos para agregar, eliminar y buscar tareas y recordatorios utilizando la función de hash.
* **Manejo de Colisiones:** Implementación de la resolución de colisiones según la estrategia (encadenamiento o direccionamiento abierto).
* **Optimización de la Función de Hash:** Aseguramiento de que la función de hash sea eficiente y distribuya uniformemente las claves (Validación).

**Diseño del Algoritmo de Heapsort:**

* **Montículo (Heap):** Con base a una estructura de datos de montículo (heap), se debe permitir la inserción eficiente de tareas y recordatorios, así como la extracción de elementos según la prioridad.
* **Ordenación:** Implementar el algoritmo de heapsort de manera que las tareas se puedan ordenar por fecha límite o prioridad según las necesidades del usuario.

**Diseño de la Cola de Prioridades:**

* **Clasificación de Tareas:** Una cola de prioridades mínima (min heap) para tareas prioritarias y una cola de prioridades máxima (max heap) para tareas no prioritarias.
* **Actualización de Prioridades:** Permitir a los usuarios cambiar la prioridad de una tarea y ajustar su posición en la cola de prioridades.

**Diseño de la Interfaz de Usuario (UI):**

* **Flujo de Usuario:** Los usuarios se podrán relacionar con un menú para diferentes actividades que se implementen en el sistema

Welcome to:

/—------**ALWAYS NOTE**—--------/

1. Agregar una nueva tarea.

2. Modificar una tarea existente.

3. Eliminar una tarea.

4. Ver lista de tareas y recordatorios.

5. Filtrar tareas por prioridad.

6. Deshacer la última acción.

7. Configuración de notificaciones.

8. Salir del sistema.

**Diseño de Notificaciones Push:**

* **Integración de Notificaciones:** Serán los mensajes por computadora estandarizados de la aplicación donde se genere el código. Esto incluye la configuración de las notificaciones y la gestión de eventos que activarán las notificaciones o el formato en el que se le presentarán al usuario.Ya no dependerá de algún formato en especifico o aviso, solo del editor de código que se emplee.

**Diseño del Sistema de "Deshacer":**

* **Pila de Acciones:** Diseño de una pila (LIFO) que registre las acciones del usuario junto con los detalles de las tareas afectadas. Asegúrate de que la pila sea eficiente en cuanto a uso de memoria.

**Arraylist:**

* No se relaciona de forma práctica con el ejercicio. Por lo tanto, se descarta esta opción.

Paso 5. Pruebas y selección de solución (Evaluar Precisión, eficiencia, complejidad, facilidad de implementación algorítmica)

**Criterio A.** Precisión de la solución. La alternativa entrega una solución:

­ [2] Exacta (se prefiere una solución exacta)

­ [1] Aproximada

­ **Criterio B.** Eficiencia. Se prefiere una solución con mejor eficiencia que las otras consideradas. La

eficiencia puede ser:

­ [4] Constante

­ [3] Mayor a constante

­ [2] Logarítmica

­ [1] Lineal

­ **Criterio C.** Completitud. Se prefiere una solución que encuentre todas las soluciones. Cuántas

soluciones entrega:

­ [3] Todas

­ [2] Más de una si las hay, aunque no todas

­ [1] Solo una o ninguna

­ **Criterio D.** Facilidad en implementación algorítmica:

­ [2] Compatible con las operaciones aritméticas básicas de un equipo de cómputo moderno

­ [1] No compatible completamente con las operaciones aritméticas básicas de un equipo de

cómputo moderno

|  | **Criterio A** | **Criterio B** | **Criterio C** | **Criterio D** | **TOTAL** |
| --- | --- | --- | --- | --- | --- |
| **Tabla Hash** |  |  |  |  |  |
| **Algoritmo de Heapsort** |  |  |  |  |  |
| **Cola de Prioridades** |  |  |  |  |  |
| **Interfaz de Usuario (UI)** |  |  |  |  |  |
| **Notificaciones Push** |  |  |  |  |  |
| **Sistema de "Deshacer"** |  |  |  |  |  |
| **Arraylist** |  |  |  |  |  |

La elección de soluciones es la siguiente:

Puntos que desarrollan después de todo el proceso investigativo

6. informes y especificaciones

7. Implementación