Informe Proyecto Final Programación en C/C++

Osmany Gama Morell 2do CC

Problemática: imagínate que tienes muchas tareas que debes realizar, algunas son tareas de pago y otras son tareas de mantenimiento. Es complicado llevar un control de todas ellas y determinar cuáles son más importantes o cuáles ya han vencido el plazo. El proyecto del Gestor de Tareas resuelve esta problemática al proporcionar una forma organizada de administrar estas tareas.

¿Cómo se resuelve?

En este proyecto, se ha creado un software llamado Gestor de Tareas que ayuda a organizar diferentes tipos de tareas de manera eficiente. El objetivo principal es facilitar la gestión de tareas como pago y mantenimiento, manteniendo un registro ordenado y calculando pagos asociados.

Se han utilizado las clases (Ver diagrama de clases UML):

* Tarea: esta es una clase genérica que se creó como una base para todas las tareas con información básica como plazo, ID y descripción.
* Tarea de mantenimiento: se desarrollo para almacenar las tareas por las cuales el usuario debe pagar para realizar el mantenimiento o reparación.
* Tarea de pago: son las tareas las cuales son solicitadas por clientes y generan un pago.
* Gestor de tareas: esta herramienta centraliza todas las tareas, permitiendo añadir, eliminar, ordenar y calcular ganancias relacionadas con las tareas.

¿Cómo funciona?

Este software simplifica la gestión de tareas, asegurando que cada tarea se maneje de forma correcta y eficaz, permitiendo una mejor organización y control de las actividades a realizar. Para ello se definieron las siguientes funcionalidades:

* Guardar en Archivo: Permite guardar todas las tareas en un archivo para no perder la información.
* Cargar desde archivo: Carga las tareas desde un archivo binario con el objetivo de recuperar la información.
* Ordenar: Organiza las tareas para una mejor visualización y gestión.
* Agregar Tarea: Permite añadir nuevas tareas al sistema según su tipo.
* Eliminar Tarea: Facilita la eliminación de tareas que ya no son necesarias.
* Calcular Ganancias: Determina cuánto se gana una vez realizadas todas las tareas.
* Eliminar Tareas Expiradas: Elimina las tareas que ya han vencido su plazo.

Implementación de las funcionalidades:

* Guardar\_en\_archivo(const std::string& fileName): Este método permite guardar todas las tareas almacenadas en el gestor en un archivo binario especificado por fileName. Es útil para preservar la información de las tareas de manera persistente.
* Cargar\_desde\_archivo(const std::string& fileName): Permite cargar las tareas almacenadas previamente en un archivo binario identificado por fileName. Esta función es útil para recuperar datos guardados anteriormente.
* Ordenar(): Se encarga de organizar las tareas dentro del gestor según el plazo definido para cada tarea. Esto facilita la visualización y gestión de las tareas.
* Agregar\_tarea(Tarea\* t): Permite añadir una nueva tarea al gestor de tareas. Puede recibir un puntero a un objeto de tipo Tarea, lo que posibilita la inclusión de distintos tipos de tareas en el gestor.
* Eliminar\_tarea(std::string id\_tarea): Se elimina la tarea correspondiente al identificador (id\_tarea) proporcionado como parámetro. Esto simplifica la gestión al permitir la eliminación de tareas específicas.
* Ganancia\_total(): Calcula la ganancia total generada por todas las tareas. Proporciona una visión global de los pagos asociados a las tareas del gestor.
* Eliminar\_expiradas(): Se eliminan de la lista del gestor las tareas que han superado su plazo. Esta función ayuda a mantener actualizada la lista de tareas vigentes.
* Cant(): Devuelve la cantidad total de tareas almacenadas en el gestor, ofreciendo información sobre el volumen de tareas gestionadas en el sistema.
* Mostrar\_descripcion(int x): Se muestra la descripción de la tarea situada en la posición x dentro del gestor. Facilita la visualización de detalles específicos de una tarea concreta.
* Mostrar\_todas(): Muestra todas las tareas almacenadas en el gestor, ofreciendo una visión completa de todas las tareas gestionadas en el sistema.

|  |
| --- |
| Class Gestor\_de\_tareas |
| -tareas: std::vector<Tarea\*> |
| +Guardar\_en\_archivo(fileName: std::string)  +ordenar()  +Cargar\_desde\_archivo(fileName: std::string)  +agregar\_tarea(t: Tarea\*)  +eliminar\_tarea(id\_tarea: std::string)  +ganancia\_total()  +eliminar\_expiradas()  +cant()  +mostrar\_descripcion(x: int)  +mostrar\_todas() |

|  |
| --- |
| Class Tarea |
| -plazo: int  -id: std::string  -descripcion: std::string |
| +Tarea(plazo: int, id: std::string, descripcion: std::string)  +expiro(): bool  +pago(): double  +operator==(otra: Tarea): bool  +operator<(otra: Tarea): bool  +getID(): std::string  +getDescripcion(): std::string |

|  |
| --- |
| Class Tarea\_de\_pago |
| -dificultad: int  -cliente: std::string |
| +Tarea\_de\_pago(dificultad: int, cliente: std::string, plazo: int, id: std::string, descripcion: std::string)  +pago(): double |

|  |
| --- |
| Class Tarea\_de\_mantenimiento |
| -tipo\_de\_falla: std::string  -gravedad: int |
| +Tarea\_de\_mantenimiento(tipo\_de\_falla: std::string, gravedad: int, id: std::string, descripcion: std::string, plazo: int)  +pago(): double |