Gestor de tareas

Hecho por: Juan Garzón

Objetivo

El objetivo del proyecto fue desarrollar un programa que le permita permitiera a distintos usuarios gestionar sus tareas desde la consola. Utilizando conceptos como: listas, para guardar la información de los usuarios y de las tareas; estructuras de control y control repetitivas, para validar condiciones que ocurran a lo largo del programa; vectores, para guardar información de una tarea; modularidad, para escribir un código más legible.

Funciones principales

- **Gestión de usuarios:**
  - Registro_nuevo_usuario: Guardar el nombre, correo y un ID generado automáticamente en la lista de usuarios. Y el ID generado en su lista de IDs respectiva.

  - Revisar_correo: Validar si un correo existe o no existe.

- **Gestión de tareas:**
  - Crear_tareas: Pide el correo del usuario, y va configurando los valores para guardar su tarea como, descripción de la tarea, estado de la tarea,
  - Actualizar_tareas: Cambiar el estado de tarea actual a las otras 3 opciones restantes, mostrando un pequeño menú.

  - Busqueda_tareas: Realiza la búsqueda de una tarea dependiendo si el usuario decide buscar por la descripción o ID de su tarea. Y retorna la posición de esta en la lista de tareas general.
- **Mostrar tus tareas:**
  - Imprimir_tareas: Mostrar al usuario las tareas que tiene con su descripción, estado actual de la tarea y fecha límite.

  - Tarea_atrasada: Mostrar al usuario cuantos días faltan para la fecha límite de una tarea.

- Cantidad_tareas_estado: Mostrar al usuario cuantas tareas tiene en un estado.

Problemas encontrados

1. **Gestión de usuarios:** Al momento de un usuario querer registrarse, mostraba que su correo ya existía o cuando un usuario ingresaba su correo, mostraba que su correo no existía. Esto se corrigió modificando la lógica, y colocando 2 casos en la función "revisar_correo", cuando se quiere saber si un correo ya existe y cuando un correo no existe

2. **Búsqueda de tareas:** Modificar la función de búsqueda tareas, para que no sean 2 funciones por separado, y sea solo una. Esto se hace para eliminar líneas de código innecesarias y aplicar el concepto de modularidad.

3. **Condiciones en las estructuras de control repetitivas:** Cuando se quería validar en algunas funciones. Por ejemplo, que el número de opción que ingresaba el usuario esté dentro del rango posible, no validaba porque la condición estaba escrita con el operador incorrecto. En vez de "or" estaba escrito con "and".

Conclusiones

Este proyecto fue un reto interesante, ya que se tuvo que realizar una ligera investigación sobre las librerias random para la generación de IDs, y datetime para las fechas. Además, de una dificultad moderada-alta porque teníamos que usar los conceptos vistos a lo largo del semestre y de manera particular la modularidad.

Tambien, este proyecto nos ayudó a mejorar nuestra lógica de programación, y la capacidad de resolución de problemas. Asimismo, corregir errores que vayan surgiendo de nuestro código, que es una parte fundamental en un contexto laboral. Y así, ver oportunidades de mejores en el mismo o de manera general, la depuración de código.

Por último, este proyecto nos incentiva a crear proyectos cada vez más complejos y con mayores funcionalidades como la creación de interfaces graficas, trabajar con bases de datos, utilizar diferentes lenguajes de programación, entre otros.