PROYECTO SOBRE HORARIOS DE UNA FARMACIA

Este proyecto es una aplicación para gestionar los turnos de trabajo de empleados en una farmacia, utilizando una interfaz gráfica desarrollada con Tkinter en Python. Los empleados tienen turnos predefinidos para cada día de la semana, distribuidos en bloques de 6 horas (por ejemplo, de 00:00 a 06:00, de 06:00 a 12:00, etc.).

La interfaz de usuario permite a los empleados consultar su turno para un día específico, ingresando su nombre y el día de la semana. También pueden consultar la tabla completa de horarios para todos los empleados en una vista semanal. Además, se puede ver quién está trabajando en un horario específico (por rapidez), introduciendo la hora.

El sistema utiliza un algoritmo de búsqueda binaria para encontrar el turno de un empleado de manera eficiente, en lugar de recorrer toda la lista de empleados. Esto es posible porque los empleados están ordenados alfabéticamente en la lista por el algoritmo de ordenación Bubble Sort.

El proyecto tiene como objetivo facilitar la gestión de turnos de los empleados de la farmacia, asegurando que siempre haya cobertura en todo momento. Además, permite a los empleados consultar fácilmente su horario y el de sus compañeros de trabajo.

El código tiene varias funciones que gestionan los turnos de los empleados. La función consultar_turno permite consultar el turno de un empleado en un día específico, validando los datos ingresados. mostrar_tabla_horarios muestra todos los turnos de los empleados ordenados alfabéticamente para cada día de la semana. consultar_empleado_a_hora permite saber qué empleados están trabajando a una hora específica. ordenar_empleados organiza los nombres de los empleados alfabéticamente, y busqueda_binaria busca un empleado en una lista ordenada.

He elegido Bubble Sort y Búsqueda Binaria por las siguientes razones:

Bubble Sort: Este algoritmo es fácil de implementar y adecuado para una lista pequeña. Aunque no es el más eficiente en grandes cantidades de datos, al ser simple lo hace ideal para este contexto. Su eficiencia temporal en el peor caso es O(n^2), que ocurre si la lista está completamente desordenada, en el mejor caso, si la lista ya está ordenada, es O(n). Si se escalara a cientos de empleados, sería recomendable reemplazar Bubble Sort por un algoritmo más eficiente como Merge Sort, que tiene una complejidad O(nlogn).

Búsqueda Binaria: He elegido este algoritmo para la consulta de turnos, ya que una vez que los empleados estén ordenados, la búsqueda binaria permite encontrar el turno de un empleado de manera más eficiente en comparación con una búsqueda secuencial. Esto mejora la velocidad de la búsqueda y permite consultas rápidas. Su eficiencia temporal es siempre O(logn), ya que reduce el espacio de búsqueda a la mitad en cada paso.