Equipo #3 - IDFT301

Formato de muestra en consola

> [id]:[Nombre de la Tarea](i:[importancia], p:[prioridad]) end [fecha de fin]

Cola con Prioridad

Para mantener un orden según las prioridades de las tareas almacenadas y mostrarlas según ese orden, se ha implementado una cola de prioridad max-heap. La prioridad de cada tarea es calculada teniendo en cuenta los días que quedan para la culminación de la misma, la importancia establecida por el propio usuario y si este se ha otorgado la tarea a sí mismo o lo ha hecho su jefe. Luego de haber realizado el cálculo entra en la cola de prioridad quedando ordenada. A continuación se mostraría en la pantalla destinada a mostrar las tareas, primeramente la raíz de esta cola, la cual se eliminaría una vez terminada su fecha de culminación o cuando se haya realizado con éxito la tarea y pasaría a estar en la raíz la siguiente tarea con mayor prioridad de las restantes.

Entrada desordenada

> 3: Data Structure(i:high, p:35) end 2022-10-11 00:00:00.000
> 2: MER(i:medium, p:20) end 2022-10-09 00:00:00.000
> 1: DER(i:high, p:25) end 2022-10-10 00:00:00.000

Resultado ordenado por la prioridad

> 3: Data Structure(i:high, p:35) end 2022-10-11 00:00:00.000
> 1: DER(i:high, p:25) end 2022-10-10 00:00:00.000
> 2: MER(i:medium, p:20) end 2022-10-09 00:00:00.000

Ordenamiento

Para mostrar las tareas pendientes al usuario según la fecha de culminación en la aplicación móvil se implementó el algoritmo de ordenamiento “Quick Sort” sobre las tareas. Cuando se implementa bien este algoritmo puede ser un poco más rápido que el “MergeSort” y unas dos o tres veces más rápido que el “Heapsort”.

Algoritmo para obtener las tareas según la fecha (QuickSort):

```
List<Task> getOrderTasksByEndDate() {
List<Task> copyArray =  List.from(_array);
_orderByPriority(copyArray, 0, copyArray.length -  1);
return copyArray;
}

void  _orderByPriority(List<Task> tasks, int first, int last) {
int i, j, center;
center = (first + last) ~/  2;
i = first;
j = last;
var pivot = tasks[center];
do {
while (tasks[i].endDateTime.difference(DateTime.now()).inDays <
pivot.endDateTime.difference(DateTime.now()).inDays) i++;
while (tasks[j].endDateTime.difference(DateTime.now()).inDays >
pivot.endDateTime.difference(DateTime.now()).inDays) j--;
if (i <= j) {
Task temp = tasks[i];
tasks[i] = tasks[j];
tasks[j] = temp;
i++;
j--;
}
} while (i <= j);
if (j > first) _orderByPriority(tasks, first, j);
if (i < last) _orderByPriority(tasks, i, last);
}
```

Entrada desordenada

> 3: Data Structure(i:high, p:35) end 2022-10-11 00:00:00.000
> 2: MER(i:medium, p:20) end 2022-10-09 00:00:00.000
> 1: DER(i:high, p:25) end 2022-10-10 00:00:00.000

Resultado ordenado teniendo en cuenta la fecha de culminación

> 2: MER(i:medium, p:20) end 2022-10-09 00:00:00.000
> 1: DER(i:high, p:25) end 2022-10-10 00:00:00.000
> 3: Data Structure(i:high, p:35) end 2022-10-11 00:00:00.000