OBSERVACIONES DEL RETO 1

Juan Pablo Rodríguez Briceño Cod 202022764

Nicolas Pérez Terán Cod 202116903

Ambientes de pruebas

	Máquina 1	Máquina 2
Procesadores	AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10 GHz	M1
Memoria RAM (GB)	12 GB (9,95 utilizables)	8 GB
Sistema Operativo	Windows 10 Home 64-bits	MacOs Big Sur

Maquina 1

Resultados

Porcentaje de la muestra [pct]	Req – 1	Req - 2	Req - 3	Req - 4	Req - 5
Small (768)	109.375 ms	46.875 ms	15.625 ms	609.375 ms	62.5 ms
10.00% (15008)	343.75 ms	890.625 ms	62.5 ms	413250 ms	750 ms

Maquina 2

Resultados

Porcentaje de la muestra [pct]	Req - 1	Req - 2	Req - 3	Req - 4	Req - 5
Small (768)	91.85699999997 ms	60.791000000 00004 ms	22.357000000 00007 ms	405.18000000 000001 ms	45.211200000 0000003 ms
10.00% (15008)	222.46600000 0000072 ms	571.27000000 00002 ms	67.570999999 99999916 ms	243805.655	489.11000000 000001 ms

Análisis de complejidad por cada requerimiento.

PD: La extracción de nombres tiene una complejidad de O(1), porque al ser un diccionario basta con que se introduzca el id como llave.

- **Req 1: Listar cronológicamente los artistas (Grupal)**
  Se compone de:
  - sortArtistByDate : Que tiene una complejidad de O(nLog(n)) porque es un ordenamiento con Mergesort.
  - getYearRange : Que se divide en las siguientes secciones:
    - Un algoritmo con complejidad O(n), que busca la posicion del primer elemento del rango.
    - Un algoritmo con complejidad O(n), que busca la posicion del último elemento del rango.
    - Dos algoritmos con complejidad O(3), que buscan los 3 primeros elementos y los 3 últimos, respectivamente

- **Req 2: Listar cronológicamente las adquisiciones (Grupal)**
  Se compone de:
  - sortByDate(): Con una complejidad de O(nLog(n)) porque solo aplica un MergeSort.
  - getArtworksByDate: Que se divide en las siguientes secciones:
    - Un algoritmo con complejidad O(n), que busca la posicion del primer elemento del rango.
    - Un algoritmo con complejidad O(n), que busca la posición del último elemento del rango.
    - Dos algoritmos con complejidad O(3), que buscan los 3 primeros elementos y los 3 últimos, respectivamente.
      (Se trabajan en un Array_List)
    - Un algoritmo de complejidad de O(n) que se encargara de contar las obras que fueron compradas.

- **Req 3: Clasificar las obras de un artista por tecnica (Individual)**

  Se compone de:

  - getArtistID: El cual tiene una complejidad de O(n) ya que recorre cada registro obteniendo el ID del autor o los autores de la obra

  - getArtworksByArtistsTechnique: El cual se divide en X secciones:

    - Un algoritmo de complejidad O(n) que extrae las obras que coincidan con el artista que se está buscando
    - Otro algoritmo de complejidad O(n) que ordenara, de mayor a menor, las técnicas según el número de obras que hayan sido realizadas con esa técnica
    - Un algoritmo de complejidad O(5) que extraerá el top 5 de técnicas más usadas por el artista
    - Un algoritmo de complejidad O(3) que extraerá tres obras que coincidan con la técnica más usada

- **Req 4: Clasificar las obras por la nacionalidad de sus creadores (Individual).**

  Se compone de:

  - addConstituentId(): Que tiene una complejidad de O(n) porque tiene es necesario que revise todos los elementos de la lista y extraiga sus ID.
    PD: n vendria siendo el tamaño de la lista

  - getNatInfo(): Que tiene una complejidad de o(n) porque tiene que recorrer todos los elementos para sacar la información de los artistas.

  - sortNar(): Que se divide en las siguientes secciones.

    - Un algoritmo que con complejidad de O(n), porque recorre toda la lista para agregar los valores a la subList.
    - - Un MergeSort, que tiene complejidad de O(nLog(n))
    - - Un algoritmo para imprimir en pantalla con complejidad O(10) porque solo agarra los primeros 10 elementos.

  - - getArtworksbyArtists(): Que se divide en las siguientes secciones.

    - - Un algoritmo con complejidad de O(n^2) porque necesita revisar n-simas veces n para poder saber cuáles son los artistas y luego buscar sus nombres.

      (La complejidad de conseguir el nombre también seria O(n^2), es decir, la complejidad final podría ser O(n^4))
    - - Un MergeSort de complejidad O(nLog(n))

- - Un algoritmo para mostrar en pantalla con complejidad O(n), porque revisa todos los elementos.

- **Req 5: Transportar obras de un departamento (Grupal)**

  Se compone de:
  - getDepArtworks: El cual tiene una complejidad de O(n) ya que recorre todos los registros buscando si coinciden en el departamento.
  - calculatePrice: El cual tiene una complejidad de O(n) ya que recorre todos los registros para sacar el precio de cada uno
  - showPrice: El cual se divide en 4 secciones:
    - Un algoritmo de complejidad O(nLog(n)) que organiza los datos por fecha de creación usando MergeSort
    - Un algoritmo de complejidad O(5) que extrae las 5 obras más antiguas
    - Otro algoritmo de complejidad O(nLog(n)) ya que organiza los datos por precio (de mayor a menor) usando MergeSort
    - Un algoritmo de complejidad O(5) que extrae los 5 elementos más costosos de transportar