# Reto 1: Documento de análisis

## Integrantes del grupo

Grupo 11: **Federico Melo Barrero.**

202021525.

[f.melo@uniandes.edu.co](mailto:f.melo@uniandes.edu.co)

Implementa el requerimiento individual 3.

**Juan Camilo Prieto Avella.**

201814815.

[jc.prietoa@uniandes.edu.co](mailto:jc.prietoa@uniandes.edu.co)

Implementa el requerimiento individual 4.

## Análisis de complejidad de los requerimientos

### Carga de los datos

Al cargar los datos en el catálogo, se crean TANTAS listas ordenadas:

* artists\_BeginDate tiene a los artistas ordenados por su fecha de nacimiento, BeginDate.
* artworks\_DateAcquired tiene a las obras ordenadas por su fecha de adquisición, DateAcquired.
* artists\_artworks ...
* nationality: Tiene un array list con la llave de nacionalidad y otra llave con un array de las obras que pertenecen a esa nacionalidad.

Como las listas se ordenan al cargar los datos, sólo se ordenan una vez y luego se pueden usar los requisitos libremente.

Para ordenar las listas con las funciones de ordenamiento se eligió el algoritmo **mergesort**, que tiene como complejidad temporal O(n log(n)) en su peor caso, mejor caso y caso promedio, lo que lo hace el algoritmo más veloz de los vistos en clase.

Como estructura de datos se eligió el **arreglo** (ARRAY\_LIST) para cada una de las listas, porque es la más rápida para el propósito de buscar en ella y ordenarla, tanto teóricamente como prácticamente.

### Requerimiento 1

Para el requerimiento 1, se usa la lista de artistas artists\_BeginDate que está ordenada cronológicamente por la fecha de nacimiento.

Cuando se ejecuta el requerimiento, se toma una sublista con artistas cuyas fechas de nacimiento están en el intervalo de años dados. Esto se hace con la función rangoArtists que tiene una complejidad temporal de O(n), porque el primer ciclo acaba en la posición start de la lista y allí inicia el segundo. En el peor de los casos, que la posición inicial no está en la lista, corre el primer ciclo pero no el segundo.

En el view, se usa la función printReq1() para mostrar los resultados del requerimiento uno en una tabla, hecha con la librería prettytable. Esta función no interviene en el funcionamiento del requerimiento.

### Requerimiento 2

Para el requerimiento 1, se usa la lista de obras artworks\_DateAcquired que está ordenada cronológicamente por la fecha de adquisición.

Se implementó de forma muy similar al requerimiento 1. Cuando se ejecuta el requerimiento, se toma una sublista con obras cuyas fechas de adquisición están en el intervalo de fechas dadas. Esto se hace con la función rangoArtworks que tiene una complejidad temporal de O(n), porque el primer ciclo acaba en la posición start de la lista y allí inicia el segundo. En el peor de los casos, que la posición inicial no está en la lista, corre el primer ciclo pero no el segundo.

Falta la relación con los artistas

En el view, se usa la función printReq2() para mostrar los resultados del requerimiento dos en una tabla, hecha con la librería prettytable. Esta función no interviene en el funcionamiento del requerimiento.

### Requerimiento 3

Implementado por Federico Melo Barrero.

Se busca el ID del artista con base en el nombre dado, para lo que se usa la función id\_artist que tiene complejidad temporal de N, pues tiene que recorrer cada elemento de la lista de artistas buscando. Se obtiene una lista de obras del artista, que tiene complejidad temporal de N porque tiene que recorre cada elemento de la lista de obras y mete las obras del artista buscado en un arreglo (ARRAY\_LIST).

En el view, se usa la función printReq3(). Esta función no interviene en el funcionamiento del requerimiento.

En el view, se usa la función printReq3(). Esta función no interviene en el funcionamiento del requerimiento.

### Requerimiento 4

Implementado por Juan Camilo Prieto Avella.

En el view, se usa la función printReq4(). Esta función no interviene en el funcionamiento del requerimiento.

### Requerimiento 5

En el view, se usa la función printReq5(). Esta función no interviene en el funcionamiento del requerimiento.

### Requerimiento 6

En el view, se usa la función printReq6(). Esta función no interviene en el funcionamiento del requerimiento.

## Pruebas de tiempos de ejecución de los requerimientos

### Máquina utilizada:

|  |  |
| --- | --- |
|  | Máquina |
| Procesadores | 1,6 GHz Intel Core i5 de dos núcleos, |
| Memoria RAM (GB) | 8.00 GB. |
| Sistema Operativo | MacOS Big Sur 64-bits |

Quitar:

Mía: Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz, 2401 Mhz, 2 procesadores principales, 4 procesadores lógicos, 4.00 GB, Microsoft Windows 10 Home Single Language.

Tuya: 1,6 GHz Intel Core i5 de dos núcleos, 8.00 GB, MacOS Big Sur 64-bits.

### Requerimiento 1:

#### Tabla:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (ARRAYLIST)** | **Insertion Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| small | 1948 | 10.308 | 8.907 | 7.783 |  |
| 50% |  | 33.522 | 40.008 | 32.497 |  |
| 100.00% | 15223 | 40.549 | 48.201 | 46.772 |  |

#### Gráfica:

### Requerimiento 2

#### Tabla:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (ARRAYLIST)** | **Insertion Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| small | 769 | 26.426 | 27.435 | 26.932 |  |
| 50% |  | 1451.382 | 1483.701 | 1530.945 |  |
| 100.00% | 138112 | 2699.743 | 2740.508 | 2768.036 |  |

#### Gráfica:

### Requerimiento 3

#### Tabla:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (ARRAYLIST)** | **Insertion Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| small | 769 |  |  |  |  |
| 50% |  |  |  |  |  |
| 100.00% | 138112 |  |  |  |  |

#### Gráfica:

### Requerimiento 4

#### Tabla:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (ARRAYLIST)** | **Insertion Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
|  |  |  |  |  |  |
| small | 769 | 8.537 | 9.611 |  |  |
| 50% |  | 7.665 | 5.458 | 7.236 |  |
| 100.00% | 138112 | 7.263 | 6.236 | 8.076 |  |

#### Gráfica:

### Requerimiento 5

#### Tabla:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Porcentaje de la muestra [pct]** | **Tamaño de la muestra (ARRAYLIST)** | **Insertion Sort [ms]** | **Shell Sort [ms]** | **Quick Sort [ms]** | **Merge Sort [ms]** |
| small | 769 |  |  |  |  |
| 100.00% | 138112 |  |  |  |  |

#### Gráfica:

### Requerimiento 6