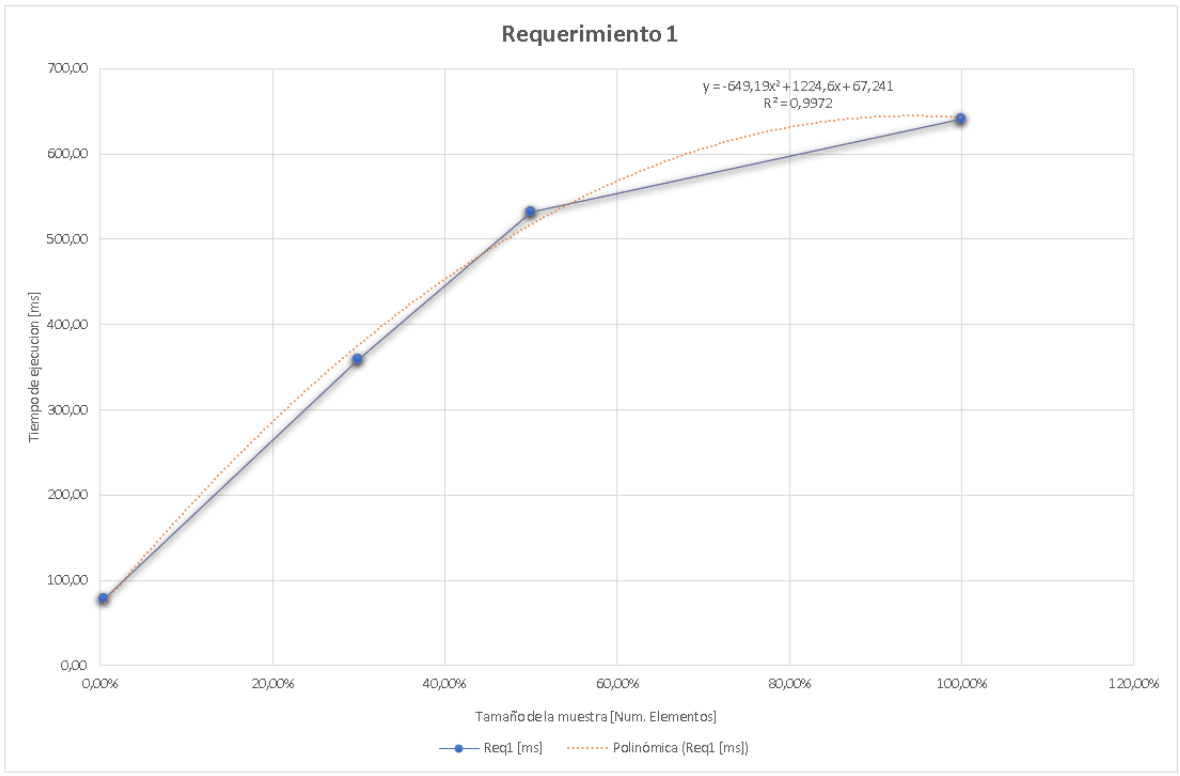
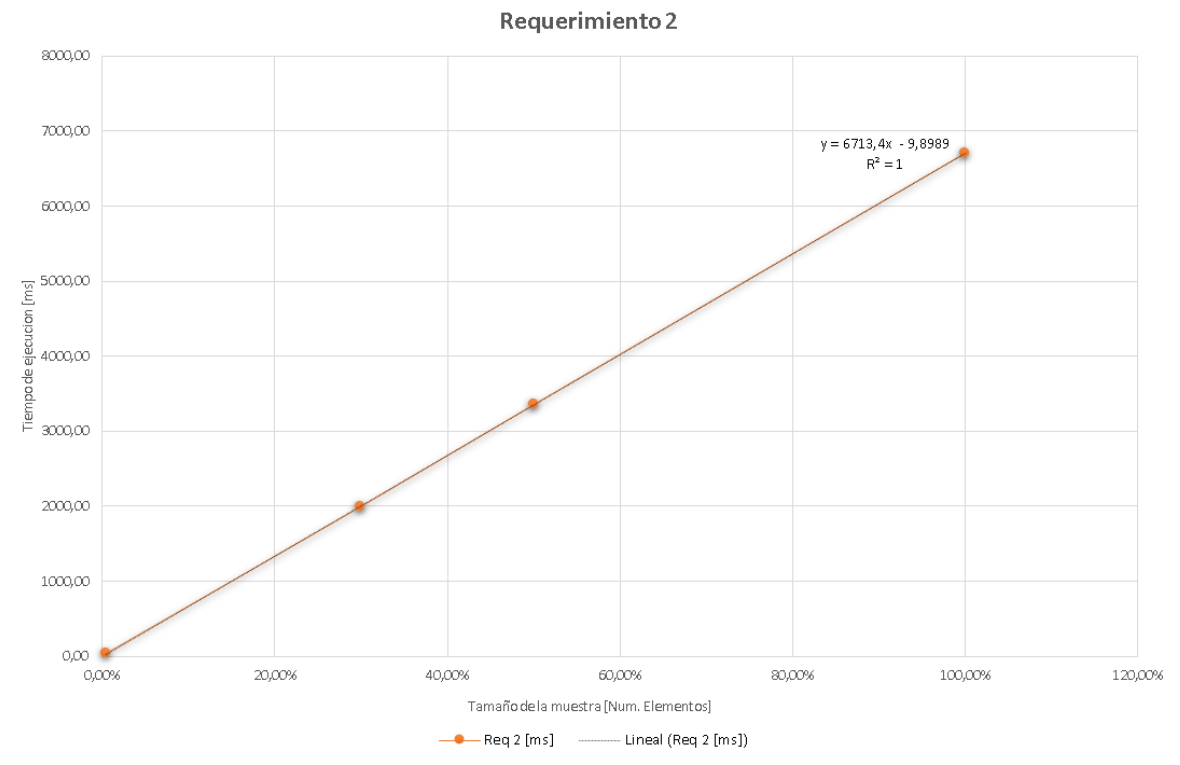
Analisis Resultados

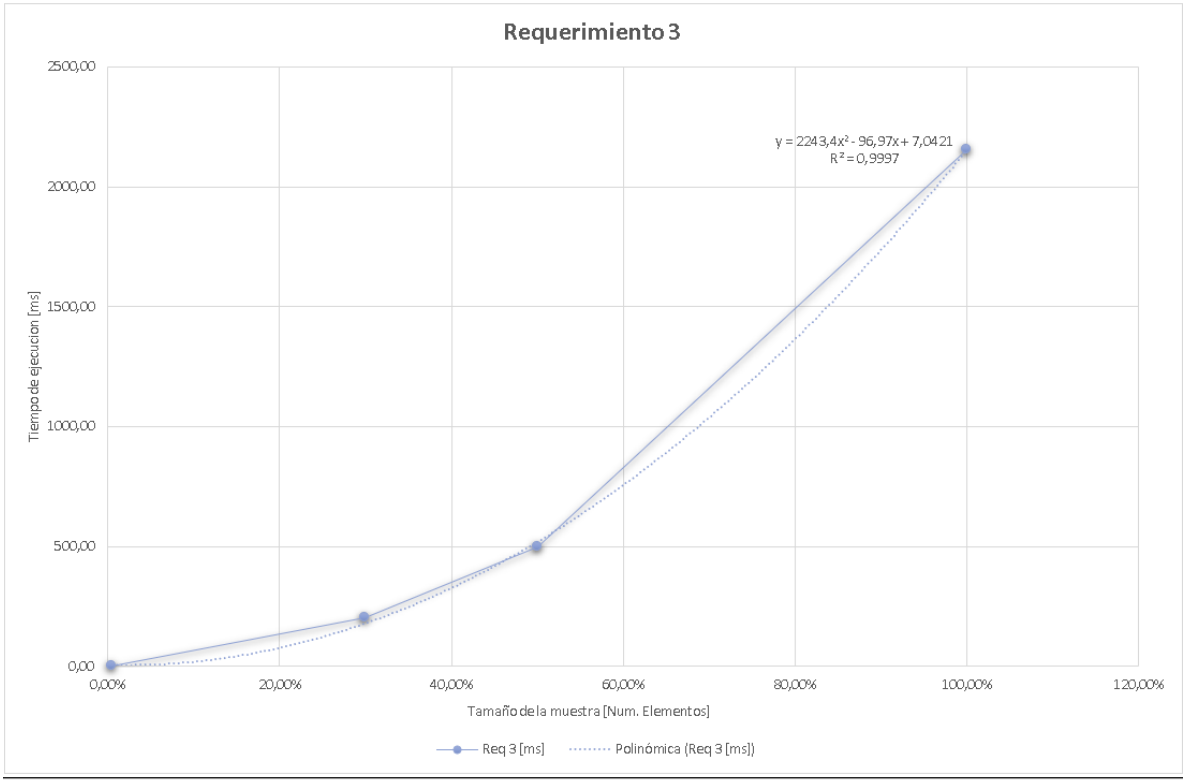
Juan Camilo Nieves - j.nievesh@uniandes.edu.co - 202116708

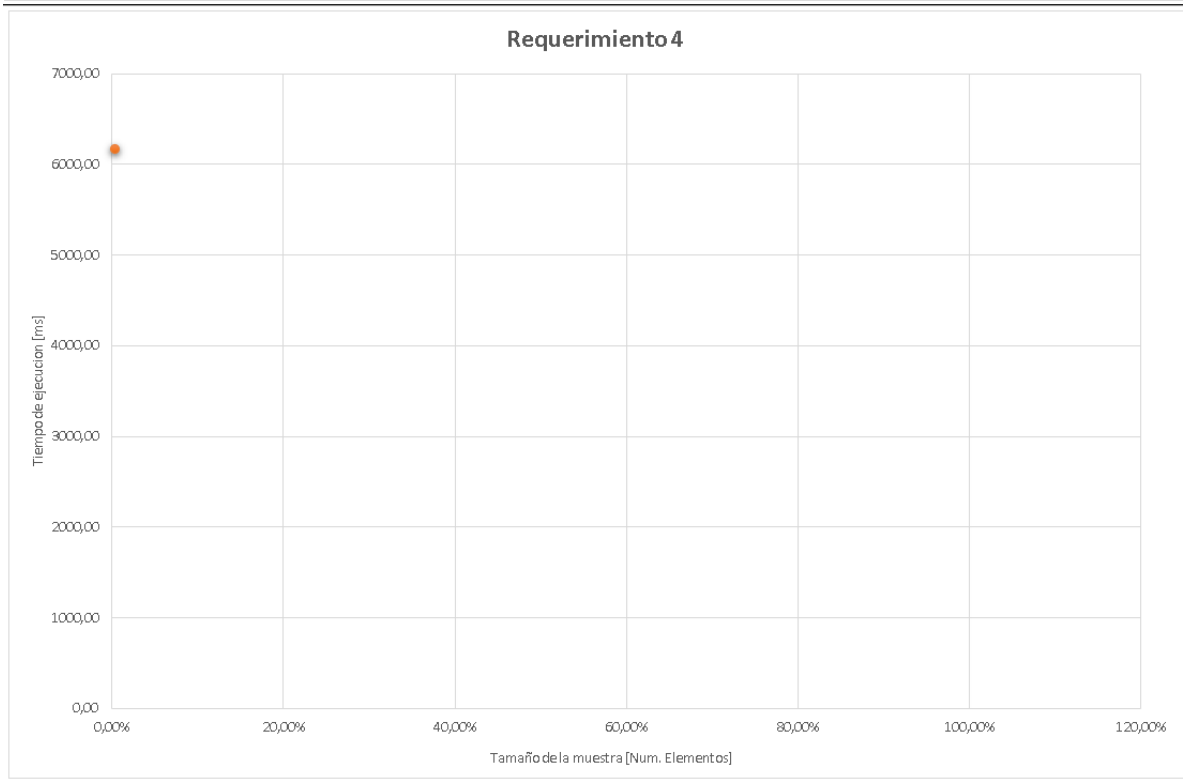
Gabriel Dicelis - g.dicelis@uniandes.edu.co – 201920847

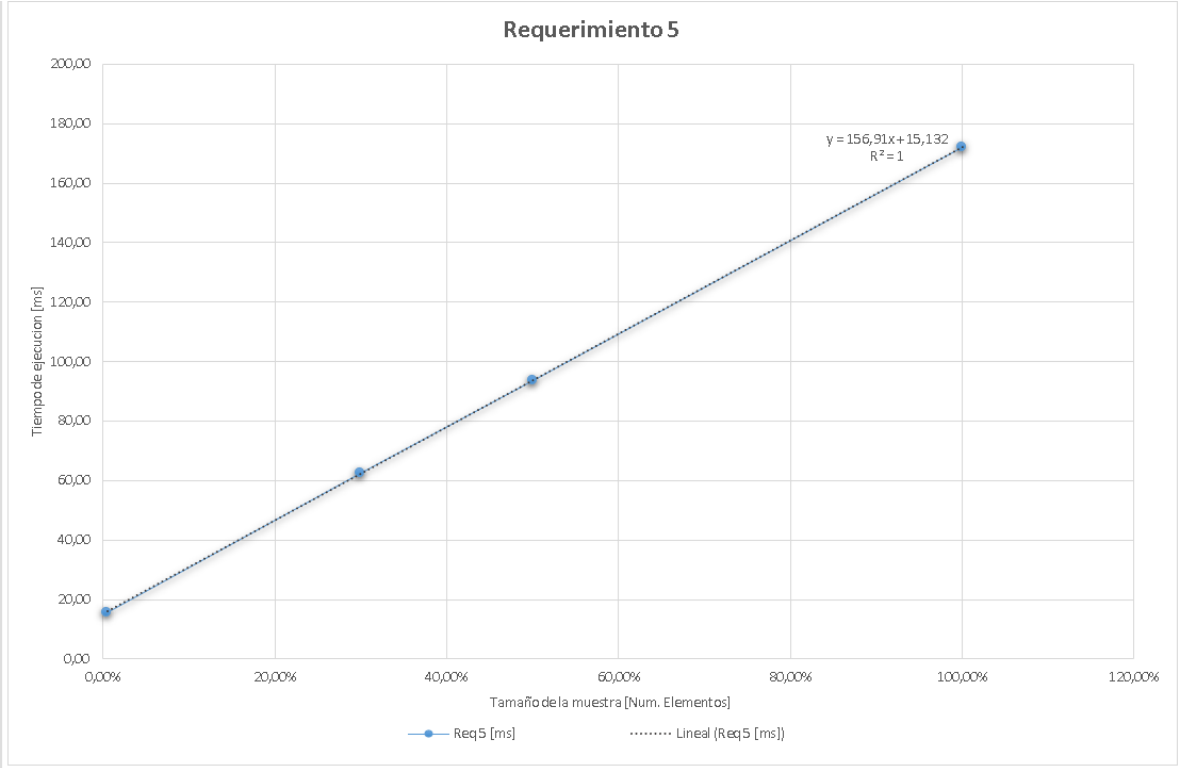
Porcentaje de la muestra [pct]	Req1 [ms]	Req 2 [ms]	Req 3 [ms]	Req 4 [ms]	Req 5 [ms]
0,50%	78,13	31,25	0,00	6156,25	15,63
30,00%	359,38	1984,38	203,13	Supera tiempo estimado	62,50
50,00%	531,25	3359,38	500,00	Supera tiempo estimado	93,75
100,00%	640,63	6703,13	2156,25	Supera tiempo estimado	171,88


Requerimiento 1
y = -649,19x² + 1224,6x + 67,241
R² = 0,9972
700,00
600,00
500,00
400,00
300,00
200,00
100,00
0,00
Tiempo de ejecucion [ms]
0,00%
20,00%
40,00%
60,00%
80,00%
100,00%
120,00%
Tamaño de la muestra [Num. Elementos]
Req1 [ms]
Polinómica (Req1 [ms])


Requerimiento 2
y = 6713,4x - 9,8989
R² = 1
8000,00
7000,00
6000,00
5000,00
4000,00
3000,00
2000,00
1000,00
0,00
Tiempo de ejecucion [ms]
0,00%
20,00%
40,00%
60,00%
80,00%
100,00%
120,00%
Tamaño de la muestra [Num. Elementos]
Req 2 [ms]
Lineal (Req 2 [ms])


Requerimiento 3
2500,00
2000,00
1500,00
1000,00
500,00
0,00
Tiempo de ejecucion [ms]
y = 2243,4x² - 96,97x + 7,0421
R² = 0,9997
0,00%
20,00%
40,00%
60,00%
80,00%
100,00%
120,00%
Tamaño de la muestra [Num. Elementos]
Req 3 [ms]
Polinómica (Req 3 [ms])


Requerimiento 4
7000,00
6000,00
5000,00
4000,00
3000,00
2000,00
1000,00
0,00
Tiempo de ejecucion [ms]
0,00%
20,00%
40,00%
60,00%
80,00%
100,00%
120,00%
Tamaño de la muestra [Num. Elementos]



Requerimientos Individuales	Implementado por
Req 3	Juan Camilo Nieves
Req 4	Gabriel Dicelis

Ordenes de complejidad temporal

Requerimiento 1:

```
def listarArtistas(catalog, inicio, fin):

    inicio = time.process_time()
    model.ordenarArtistas(catalog["artists"])

    rango_artistas = lt.newList(datastructure="SINGLE_LINKED")

    i = 1
    c = False
    while i<=lt.size(catalog["artists"]) and not c:

        artista = lt.getElement(catalog["artists"], i)
        if int(artista["BeginDate"]) > fin:
            c = True

        if int(artista["BeginDate"]) >= inicio and int(artista["BeginDate"]) <= fin:
            lt.addLast(rango_artistas, artista)
        i+=1
    fin = time.process_time()
    tiempo =(fin - inicio)*1000
    print("el t es ", tiempo)
    return rango_artistas
```

Complejidad espacial

Instrucción	**Frecuencia (#veces que se repite)**
Asignación	4+2n
Incremento	n
Comparación	2n
Operador **in**	0

Complejidad espacial = 4+5n

Complejidad temporal = O(N)

Requerimiento 2

```
def sortByDate(catalog, alg):

    sub_list = lt.subList(catalog["artworks"], 1, (lt.size(catalog["artworks"])))
    sub_list = sub_list.copy()
    elapsedtime = 00
    start_time = time.process_time()
    sorted = mg.sort(sub_list, cmpArtWorkByDateAcquired)
    stop_time = time.process_time()
    elapsedtime += (stop_time - start_time)*1000
    print("el t es ", elapsedtime)
    return sorted
```

Instrucción	**Frecuencia (#veces que se repite)**
Asignación	6
Incremento	1
Comparación	0
Operador **in**	0

Complejidad espacial = 7

Complejidad temporal = O(Nlogn)(Mergesort)

Requerimiento 3:

```
def FindIDArtist(catalog, nombre):

    artistas = catalog["artists"]
    c = False
    while c == False:
        for i in range(lt.size(artistas)):
            x = lt.getElement(artistas, i)
            if x['DisplayName'] == nombre:
                IDArtista = x['ConstituentID']
                c = True

    return IDArtista
```

```python
def tecnica_mas_usada(obrasArtista):

    tecnica_mas_usada = lt.newList(datastructure="ARRAY_LIST")
    obras = obrasArtista["elements"]
    dic = {}
    x = ""
    for obra in range(len(obras)):

        iguales = 0
        #(obras[obra]["Medium"])
        for obra1 in range(len(obras)):
            if obras[obra]["Medium"] == obras[obra1]["Medium"]:
                iguales += 1
        dic[obras[obra]["Medium"]] = iguales
    mayor = 0
    obramayor = ""
    for llave in dic:
        if dic[llave] > mayor:
           mayor = dic[llave]
           obramayor = llave
    letra = {obramayor:mayor}

    return obramayor
```

```python
def contar_tecnicas(obrasArtista):

    tecnicas = lt.newList(datastructure="ARRAY_LIST")
    for i in range(lt.size(obrasArtista)):
        artwork = lt.getElement(obrasArtista, i)
        if lt.isPresent(tecnicas, artwork["Medium"]) == 0:
            lt.addLast(tecnicas, artwork["Medium"])
    tecnicas = lt.size(tecnicas)
```

```python
def obras_tecnicaUsada(obrasArtista, obramayor):
    obrasA = obrasArtista["elements"]
    obras = lt.newList(datastructure='ARRAY_LIST')

    for i in range(len(obrasA)):
        if obrasA[i]["Medium"] == obramayor:
            lt.addLast(obras, obrasA[i])
    obrasTecnica = obras["elements"]
    long = len(obrasTecnica)
    return obrasTecnica, long
```

Instrucción	**Frecuencia (#veces que se repite)**
Asignación	15+3n^2+4n
Incremento	n^2
Comparación	$2n^2 + 3n$
Operador **in**	0

Orden de tiempo = O($2n^2$)

Requerimiento 4:

Instrucción	**Frecuencia (#veces que se repite)**
Asignación	6+n+$2n^2 + n^3$
Incremento	$2n^3 + 2n^2 + 2n$
Comparación	$2n + 2n^2 + 5n^3$
Operador **in**	0

Orden de tiempo = O(N^3)

Requerimiento 5

Complejidad Temporal O(N)