|  |
| --- |
| ESCUELA POLITECNICA DEL LITORAL |
| Proyecto de Estructura de datos |
| Grupo:  Richard Vivanco Granda  Luis F Zuniga Rosado  Jamil Nagua Campi |

|  |
| --- |
| .  . |

Contenido

[Diseño 2](#_Toc438062510)

[Identificación de los TDA’s (Tipo de datos abstractos) 2](#_Toc438062511)

[TDA NotaDeUsuario 2](#_Toc438062512)

[TDA Usuario 2](#_Toc438062513)

[TDA LeerArchivos 3](#_Toc438062514)

[TDA Reportes 3](#_Toc438062515)

[TDA Estrellas 4](#_Toc438062516)

[Abstracción – Descripción de los TDA’s 5](#_Toc438062517)

[Marco Teórico (Estructura de datos Utilizados) 6](#_Toc438062518)

[Objetos 6](#_Toc438062519)

[Comparable y Comparator 6](#_Toc438062520)

[Collections.sort 6](#_Toc438062521)

[ArrayList 7](#_Toc438062522)

[Que es una ArrayList? 7](#_Toc438062523)

[Optimización del ArrayList 8](#_Toc438062524)

[Errores encontrados 9](#_Toc438062525)

[Uso de la Aplicación 10](#_Toc438062526)

[Comparación de Rendimiento (ArrayList vs LinkedList) 14](#_Toc438062527)

# Diseño

## Identificación de los TDA’s (Tipo de datos abstractos)

TDA NotaDeUsuario

(es la nota que dio un usuario a una película en una determinada fecha )

**Atributos**

public int idPelicula

public int rating

public String fecha

**Metodos**

NotaDeUsuario()

es el constructor vacío

NotaDeUsuario(int idPelicula, int rating, String fecha)

es el constructor con parametros

toString()

retorna un string

notaDePelicula(int idusuario,int idpeli,ArrayList<Usuario> notasPel )

retorna una lista de todas las películas que ha calificado un usuario

compareTo(Object o)

retorna un int

TDA Usuario

(es la id del usuario la nota que dio a una película y la fecha )

**Atributos**

public int idUser

public int rating

public String date

**Metodos**

Usuario()

es el constructor vacío

Usuario(int idUser, int rating, String date)

es el constructor con parametros

toString()

retorna un string

compareTo(Object o)

retorna un int

TDA Pelicula(es la id de la pelicula el año en que salió y su titulo en ingles )

Atributos

public int id;

public int año;

public String titulo;

Metodos

Pelicula()

es el constructor vacío

Pelicula(int id, int año, String titulo)

es el constructor con parametros

toString()

retorna un string

compareTo(Object o)

retorna un int

TDA LeerArchivos

**Metodos**

movies( String archivo)

retorna un ArrrayList<Pelicula>

cargarUsuarios(String archivo)

retorna un Arraylist<Usuario>

buscarpalabra(ArrayList<Pelicula> listas,String palabra)

retorna ArrayList<Pelicula>

TDA Reportes

(genera distintos reportes html)

**Metodos**

reporteCalificacionPorUsuario(ArrayList<NotaDeUsuario> peliculasCalf, int id, ArrayList<Pelicula> pelicula)

reporteTopN(ArrayList<Estrellas> listaDeEstrellas, ArrayList<Pelicula> pelicula, int nEstr)

reportePeliPorAnio(ArrayList<Pelicula> pelicula)

reportePorCalificaicon(ArrayList<Estrellas> listaDeEstrellas, ArrayList<Pelicula> pelicula)

reportePorCoincidencia(ArrayList<Pelicula> concidencia, String palabra)

abrirHTML(String archivo)

TDA Estrellas

**Atributos**

int pelicula;

int estr5;

int estr4;

int estr3;

int estr2;

int estr1;

float promedio;

**Metodos**

listaEstrellas(int x, ArrayList<Estrellas> rating, ArrayList<Usuario> usuarios){

toString()

retorna String

compareTo(Estrellas t)

retorna int

getEstr4()

retorna int

getEstr3()

retorna int

getEstr2()

retorna int

getEstr1()

retorna int

getPromedio()

retorna float

GenerarTop(int top,ArrayList<Estrellas> rating)

Retorna ArrayList<Estrellas>

## Abstracción – Descripción de los TDA’s

|  |  |
| --- | --- |
| Tipo de Dato Abstracto | Funcionalidad |
| NotaDeUsuario | El objeto NotaDeUsuario nos permite guardar la información de cada nota por usuario que es asignada a una película en específico, nos ayuda a llenar la lista de películas y obtener la calificación que le dio un usuario en un fecha especifica |
| Usuario | El dato Usuario nos permite tratar con cada usuario y que película ha calificado, a diferencia de NotaDeUsuario, aquí guardamos el Id del Usuario |
| LeerArchivo | LeerArchivo funciona como el lector que carga la cantidad de archivos fijos y llena listas para su uso posterior |
| GenerarReporte | GenerarReporte nos permite mostrar un reporte HTML, este tipo de dato se encarga del manejo del lenguaje HTML, donde va a ser guardado y lo ejecuta al instante |
| Estrellas | El objeto estrellas me permite contabilizar las calificaciones dadas por películas y generar las listas de Top5, Top4, etc, asi mismo como su nota promedio |

# Marco Teórico (Estructura de datos Utilizados)

## Objetos

Durante todo el proyecto, se analizó con el paradigma orientado a objeto, creando múltiples objetos para la distribución de tareas, tales como NotaDeUsario, Usuario, Estrellas, etc.

Estos objetos se le asignaron las varios atributos y métodos creados por lo programadores y así distribuir la tareas y optimizar el tiempo. Esto se definió al momento del diseño de la solución, la cantidad de objetos necesarios para la correcta distribución de tareas.

## Comparable y Comparator

Para el ordenamiento de las listas se utiliza la comparación implementada por dos interfaces, Comparable y Comparator.

Comparable, implementa en un objeto la función int **CompareTo(Object o)** la cual se recibe un objeto y lo compara con el objeto que esta funcionando como instancia

Por ejemplo:

Objetc1.CompareTo(Object2);

En este caso el Objetc1 es comparado con Object2, y dentro del código de la clase se implementa el criterio de comparación.

En cambio Comparator implementa la función int **compare(Object o, Objecto1)** la cual recibe dos objetos y los comparan entre sí. Pero para ello es necesario que sea creado en otra clase. Una clase que servirá para un ordenamiento a través de un Sort

## Collections.sort

Para el ordenamiento de listas, usamos un método implementado en collections, llamado sort el cual esta sobrescrito para recibir la lista a ordenar o recibe la lista y una clase compare, la cual será el criterio de comparación

Collections.sort(listaAOrdenar);

Collections.sort(listaAOrdenad, new ClaseCompare());

A través de este método de ordenamiento usa el comparable o comparator, para comparar los datos y ordenarlos de mayor

## ArrayList

Durante la creación de nuestro proyecto, tuvimos que analizar la cuestión del rendimiento, al tomar en cuenta el enunciado del ejercicio, llegamos a la conclusión que la estructura más óptima es la ArrayList.

Una lista del tipo ArrayList nos permite guardar en memoria grandes listas de información, durante el momento de cargar la lista se comparó el rendimiento entre ArrayList y LinkedList.

Una LinkedList es una lista doblemente enlazada la cual es óptima en momento de insertar un dato en medio del arreglo, ya que gracias a sus dobles enlaces este tiempo es muy corto, pero ArrayList demora mucho más si necesita insertar en medio del arreglo, puesto a que tiene que partir a la mitad el arreglo, copiarlo en otro arreglo, insertar el nuevo elemento y entonces volver a copiar el arreglo.

Pero en el anunciado del proyecto no se pide ingresar o eliminar ningún arreglo, se pide cargar en memoria y buscar, lo que hacemos iterando el archivo. En ese caso ArrayList (una lista simplemente enlazada) sería la solución más óptima.

## Que es una ArrayList?

ArrayList ya se encuentra implementado en Java, con tamaño de una matriz variable. Una arrayList implementa todas las opciones de la clase List y permite la inserción de todos los elementos (incluyendo el null).

Ademas de los métodos implementados en List, la ArrayList tiene métodos propios que permiten controlar el tamaño de la ArrayList, parecido a un vector con la diferencia que a la ArrayList se le considera un objeto sincronizado o sincrónico.

Cada ArrayList tiene una capacidad, la cual es el tamaño de la matriz a utilizarse en la lista de datos. Siempre es al menos tan grande como el tamaño de la lista. A medida que se le agregan elementos en un ArrayList, su capacidad crece automáticamente. **Este crecimiento tiene un costo constante**.

## Optimización del ArrayList

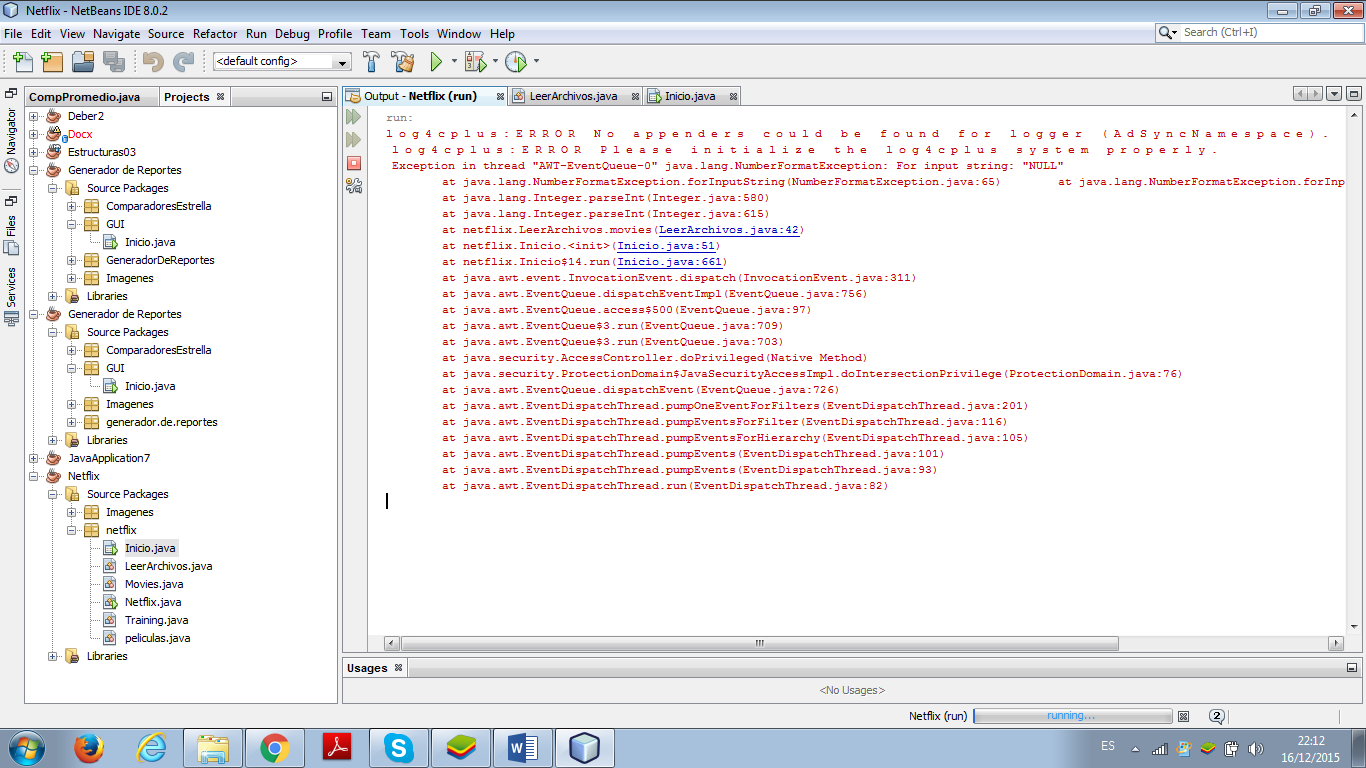
Entendemos que la adición de elementos en un arrayList tiene costo constante, lo cual lo hace óptimo para nuestro programa.

Así mismo algunas de sus funciones principales tienes un costo constante

|  |  |
| --- | --- |
| .size()  .isEmpty()  .get()  .set()  .iterator | Tiempo lineal  O(1) |
| .add(n) | O(n) |

# Errores encontrados

Un error que encontramos fue en el archivo Movies, hay películas sin año, en el año existía la palabra Null y nos producía un error

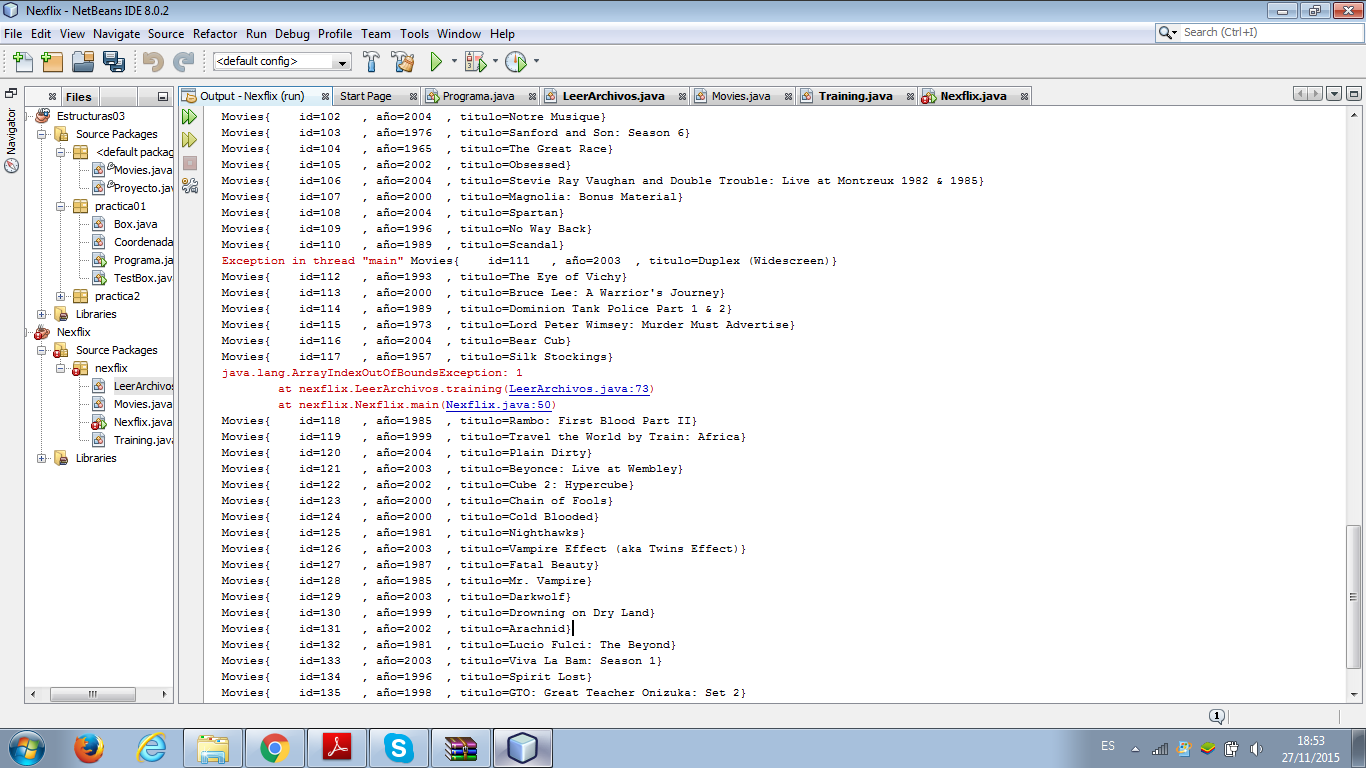


**Solución**

Donde encontramos la palabra Null le agregamos un año estándar

# 

Otro error que nos ocurrió fue el exceso de memoria al querer de guardar todos los archivos en un ArrayList, la solución fue trabajar independientemente cada archivo.

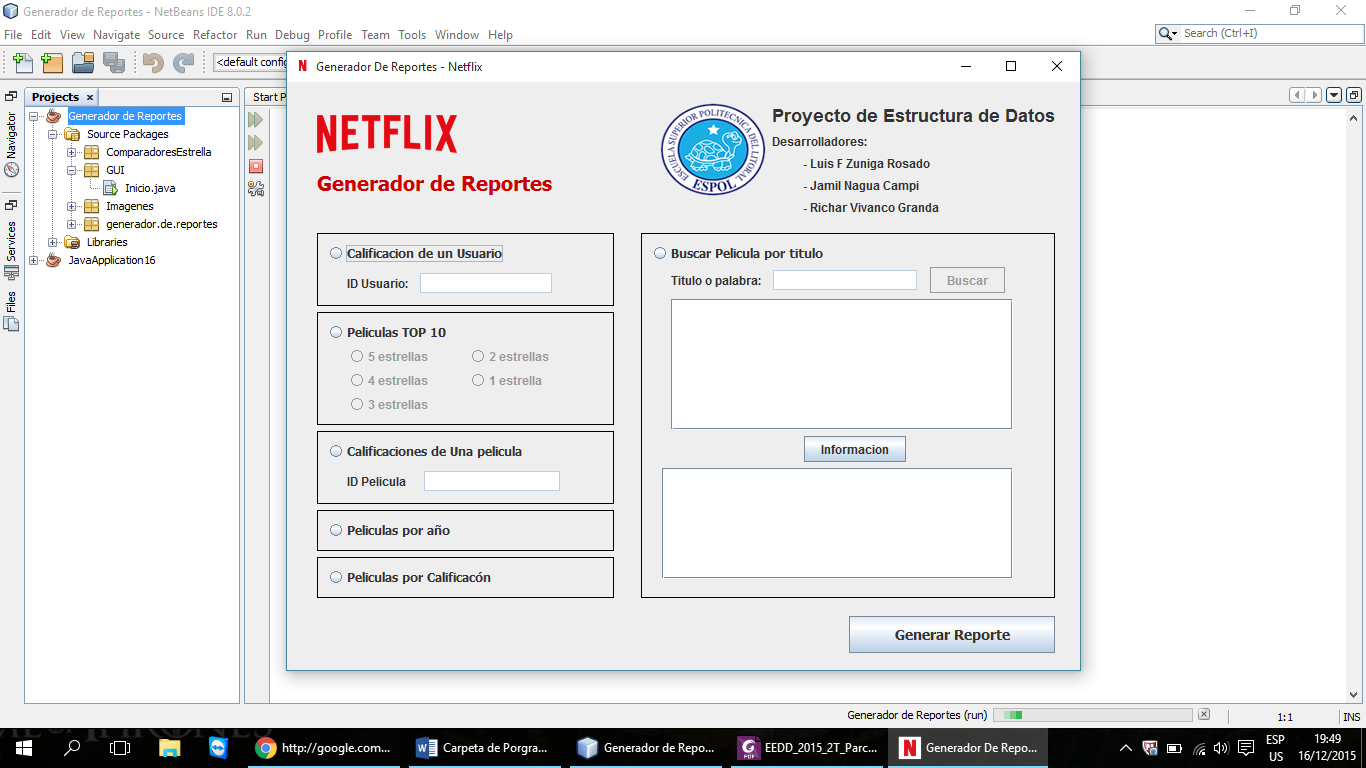


Este error me ocurrió cuando estábamos leyendo el archivo training, los usuarios , existían líneas en blanco, la solución fue tomar en cuenta esas líneas las cuales al momento de hacer Split me arrogaba error

# Uso de la Aplicación

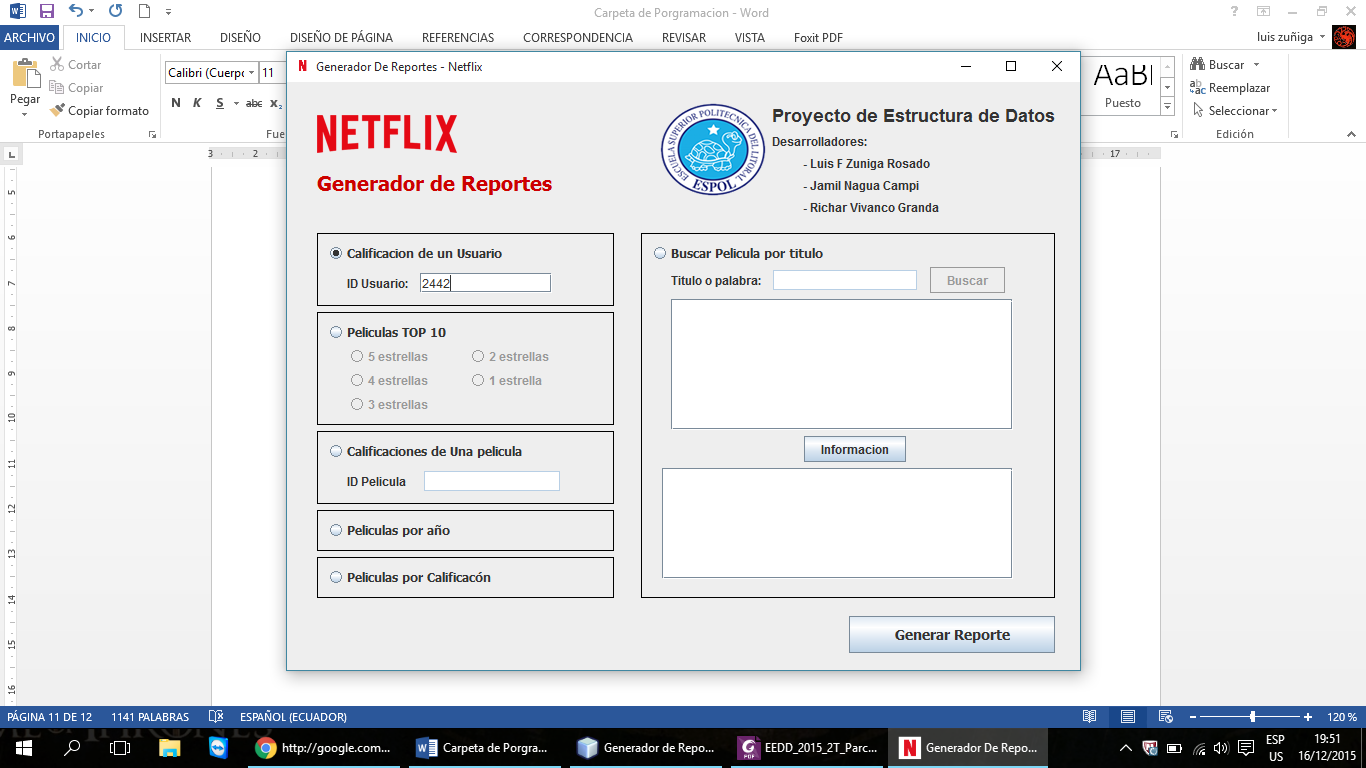
Al iniciar la aplicación es necesaria, la espera mientras se cargan los archivos y la interfaz gráfica (el cual se estima unos 8 minutos en una computadora básica)

Luego aparecerá la interfaz de esta manera:



Se presentan las opciones principales como RadioButtons los cuales son exclusivos, y permiten activar la funcionalidad especifica

Calificación de Usuario

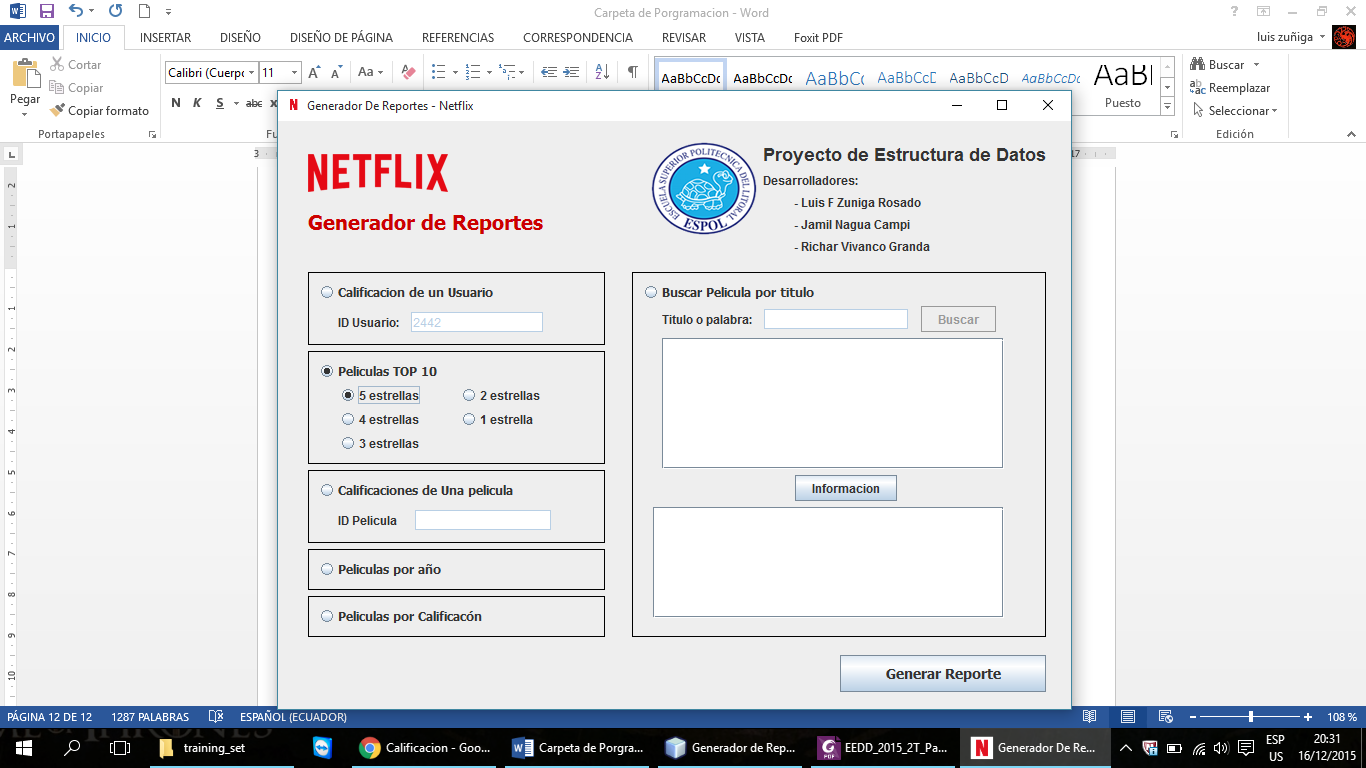


Al seleccionar el radioButton calificación de un Usuario, se pondrá activa el cajón de texto el cual pide un Id de usuario, este es único por lo tanto no es posible una colisión

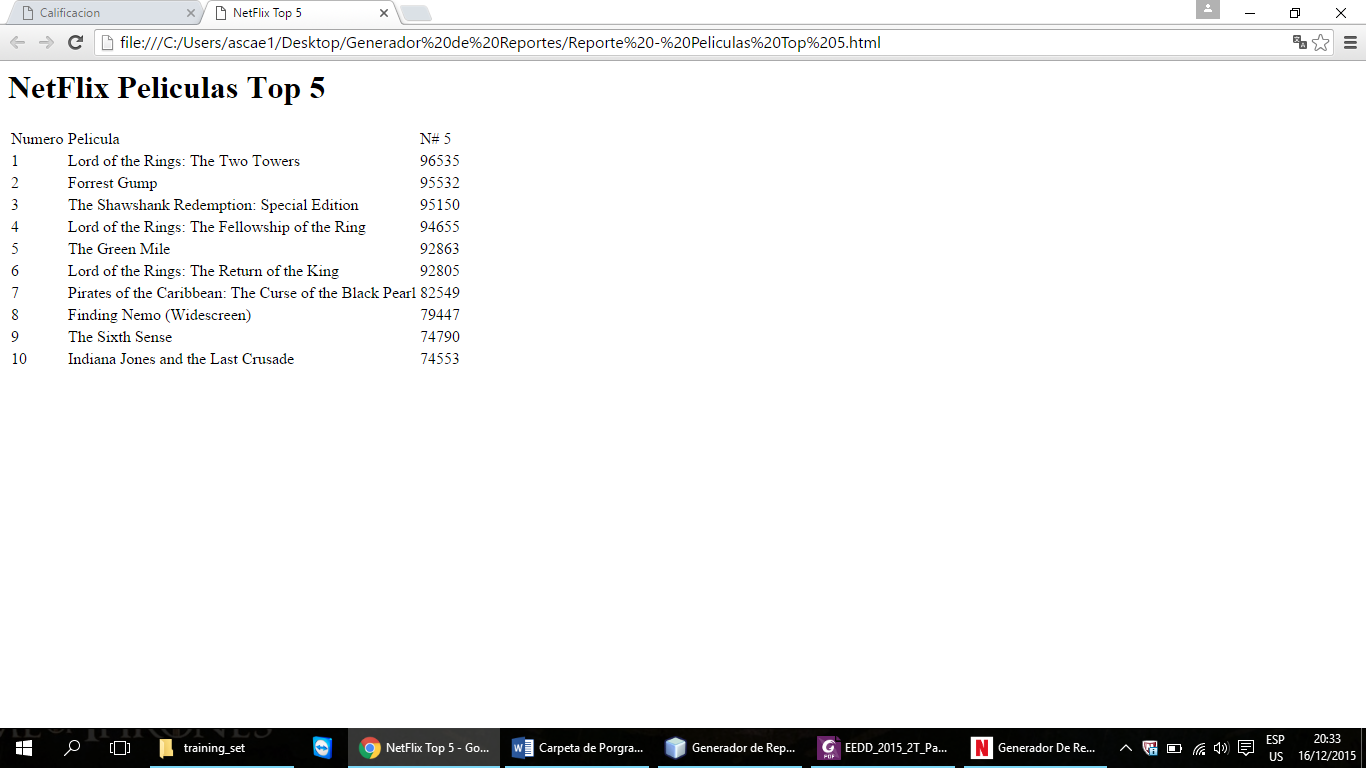


Aparece automáticamente el reporte en el explorador usando lenguaje HTML

Asi mismo la opción que genera la lista top, activa un grupo de radio buton donde se le pide al usuario que seleccione la opción de Top quiere.

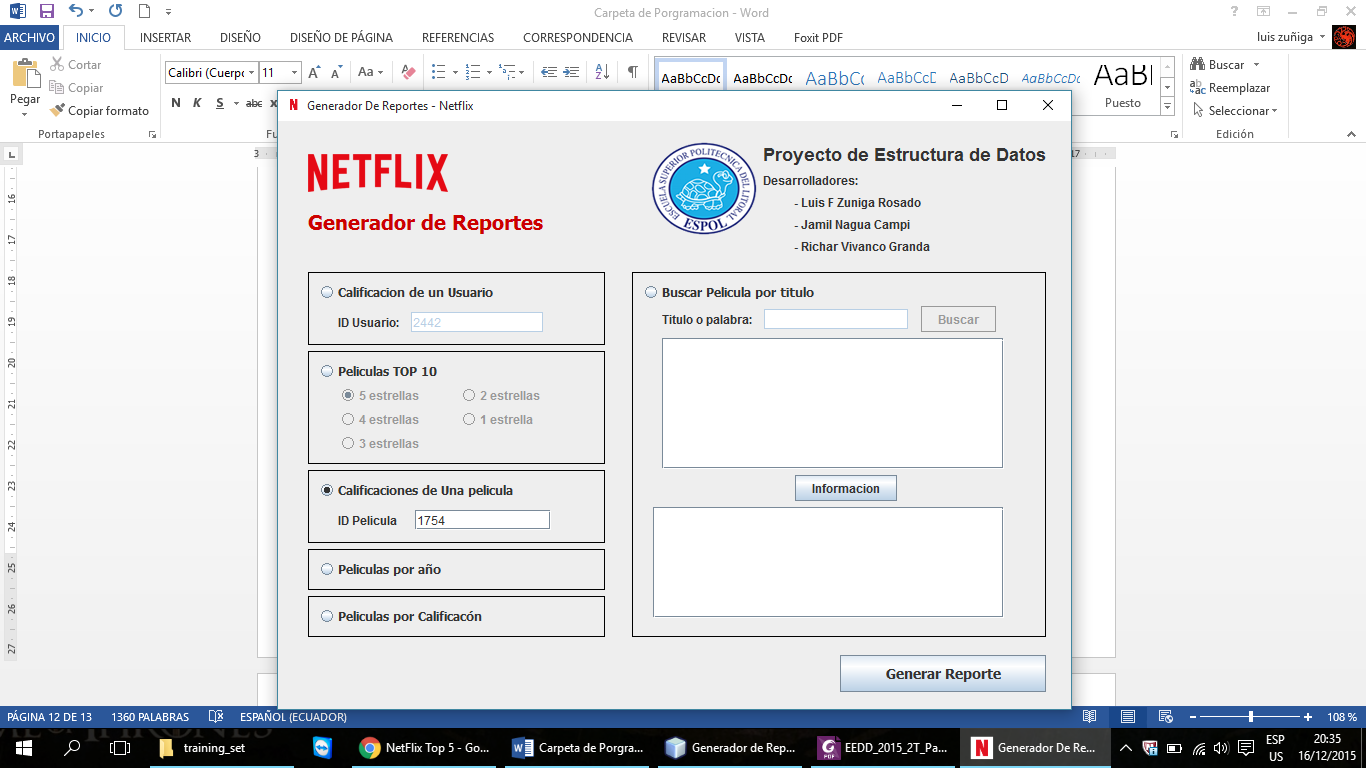


Una vez seleccionado cuantas estrellas, se oprime el botón generar reporte

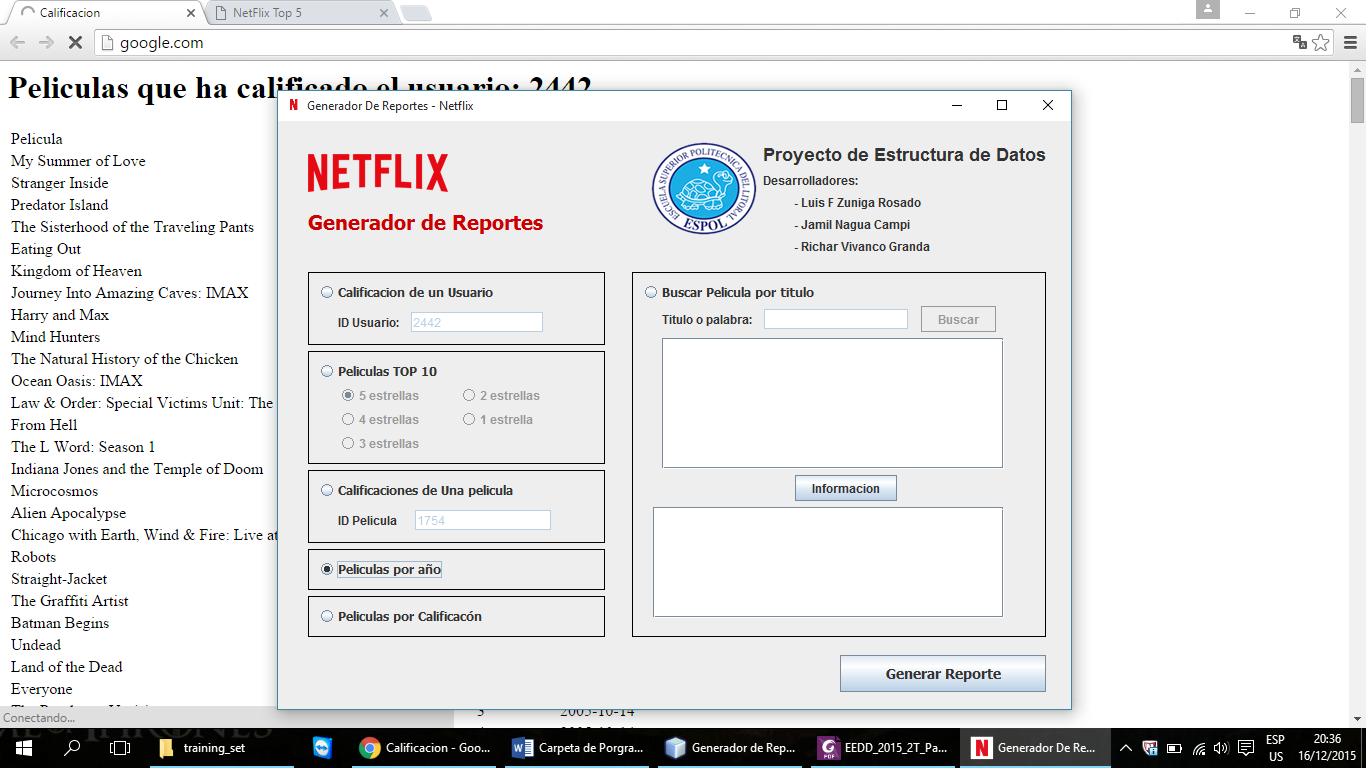


Aparecerá automáticamente en el explorador con tablas de HTML

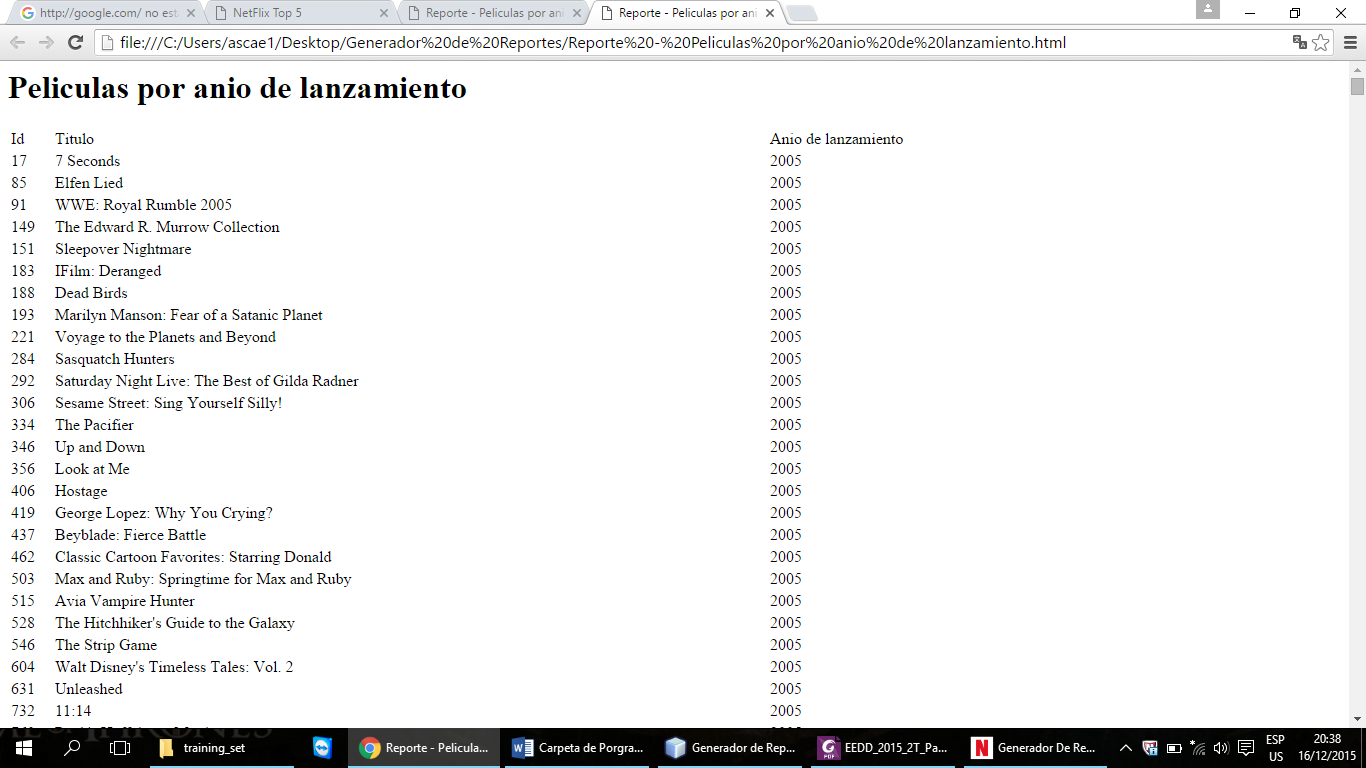
La opción Calificación por película, pide de igual forma el id de la película entre (0-17770) para encontrar la lista de películas y los días que ha sido calificada



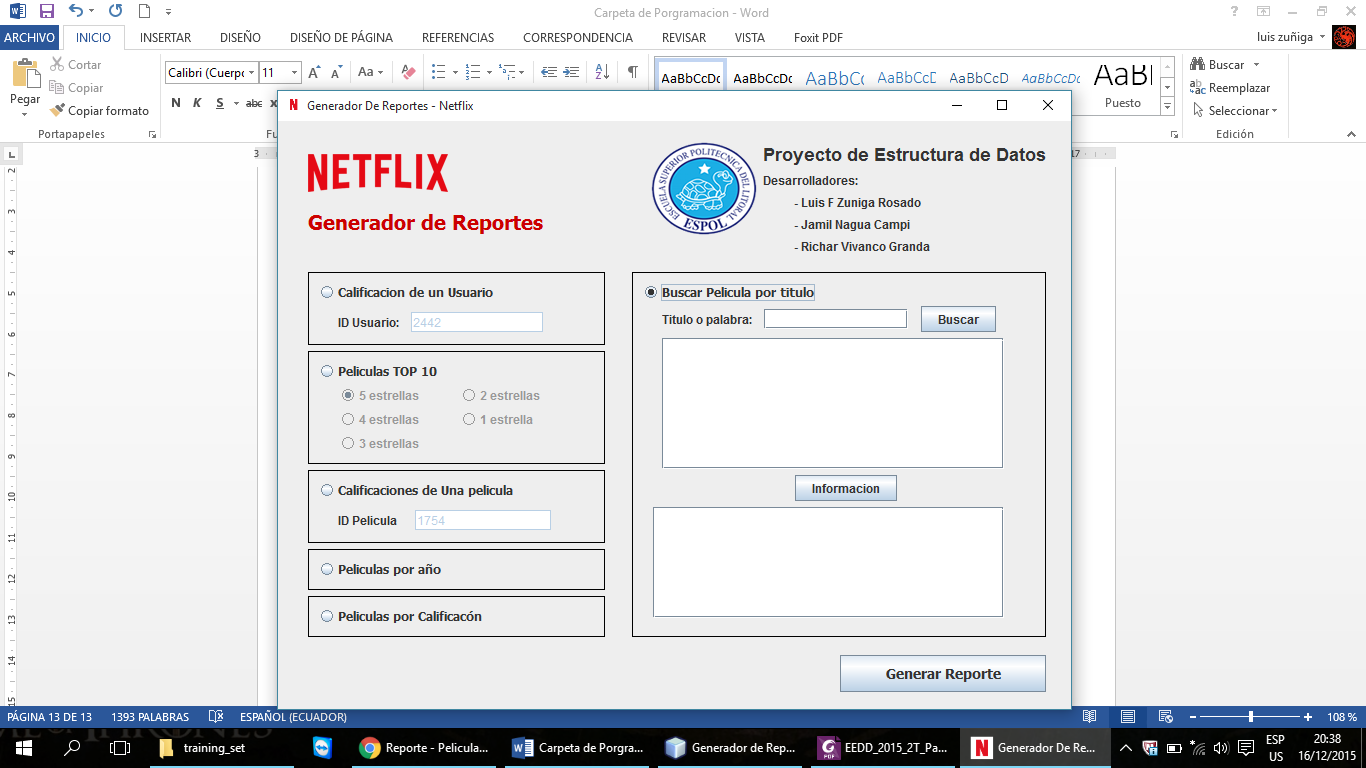
Las opciones películas por año y películas por calificación, obtiene las listas de películas ordenadas por los campos año y calificación



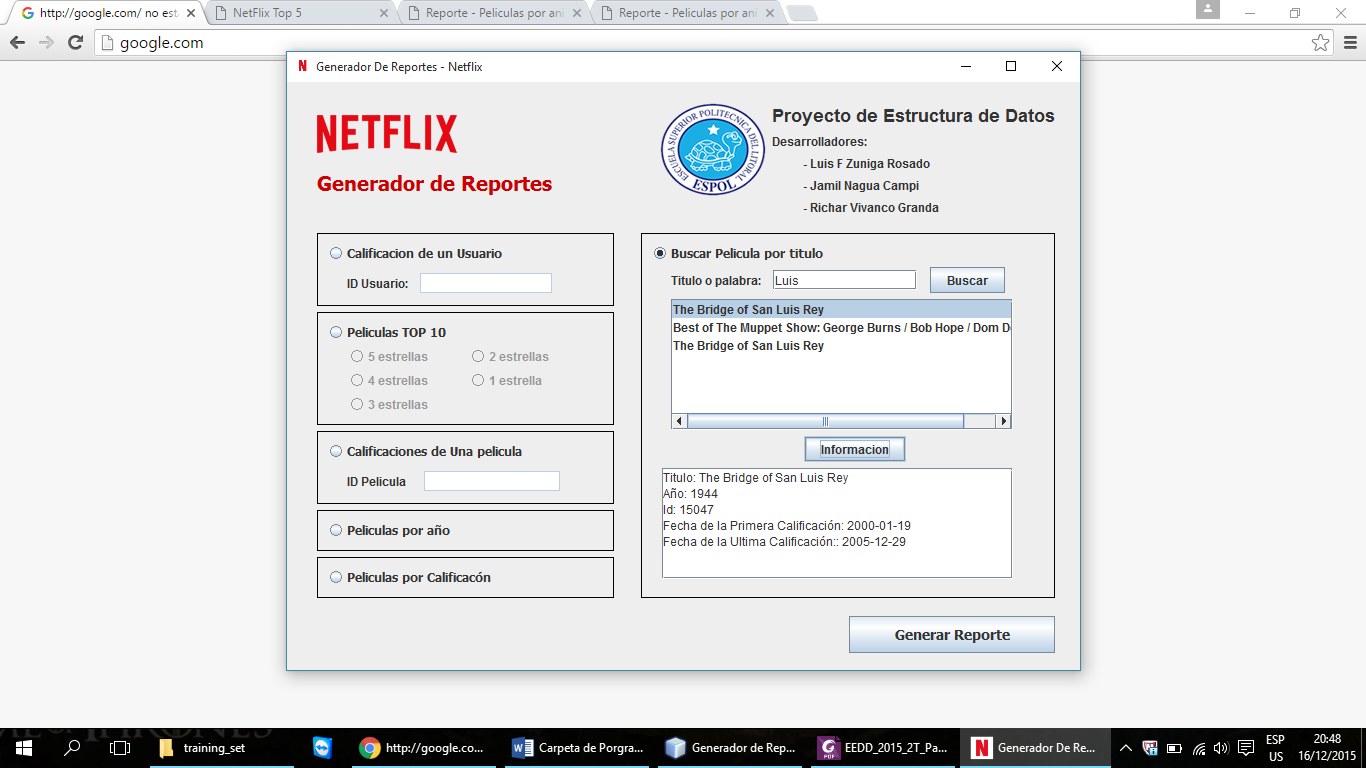
Al generar el reporte aparece automáticamente el reporte en el explorador predeterminado



Finalmente la opción Buscar por coincidencia de palabras, nos muestra primero un preview de nuestra lista encontrada antes de generar el reporte



En la cual ingresamos la palabra para buscar, y nos da la opción de mostrar la información si seleccionamos la película en la lista.



En el preview se puede ver la información que se pide, desde el año, fecha de lanzamientos y la ultima y primera calificación, después de generar reporte aparece de la siguiente manera en el explorador determinado.



El GUI es amigable con el usuario, de manera que la forma de generar los reportes es parecida, solicitando solo la información necesaria y evitando que el programa se detenga.

# Comparación de Rendimiento (ArrayList vs LinkedList)

Durante varias pruebas del programa, comparamos el rendimiento al usar ArrayList y LinkedList por separado, obteniendo los siguientes resultados (ambas pruebas fueron hechas en un Mac con la misma capacidad):

ArrayList: 55 seg

LinkedList: 50 min

# Anexos

