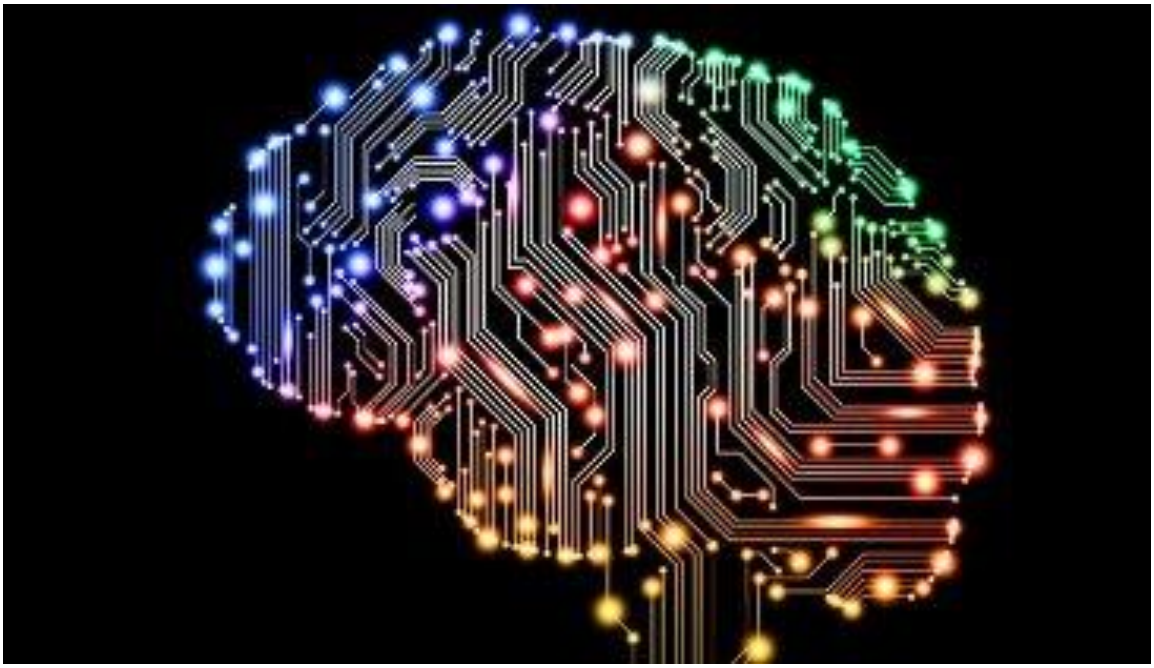
Memoria práctica

Modelos de Computación:



Autor: Tomás Machín Esteve

Grupo: 101

ÍNDICE:

INTRODUCCIÓN:

Esta práctica consiste en la obtención de links o URLs de un codigo HTML. Para conseguir esto se necesita hacer uso de los conocimientos obtenidos en clase e ilustrarlos, o más bien, programarlos para lograr el objetivo. La aplicación a utilizar es 'Apache NetBeans IDE 13' :

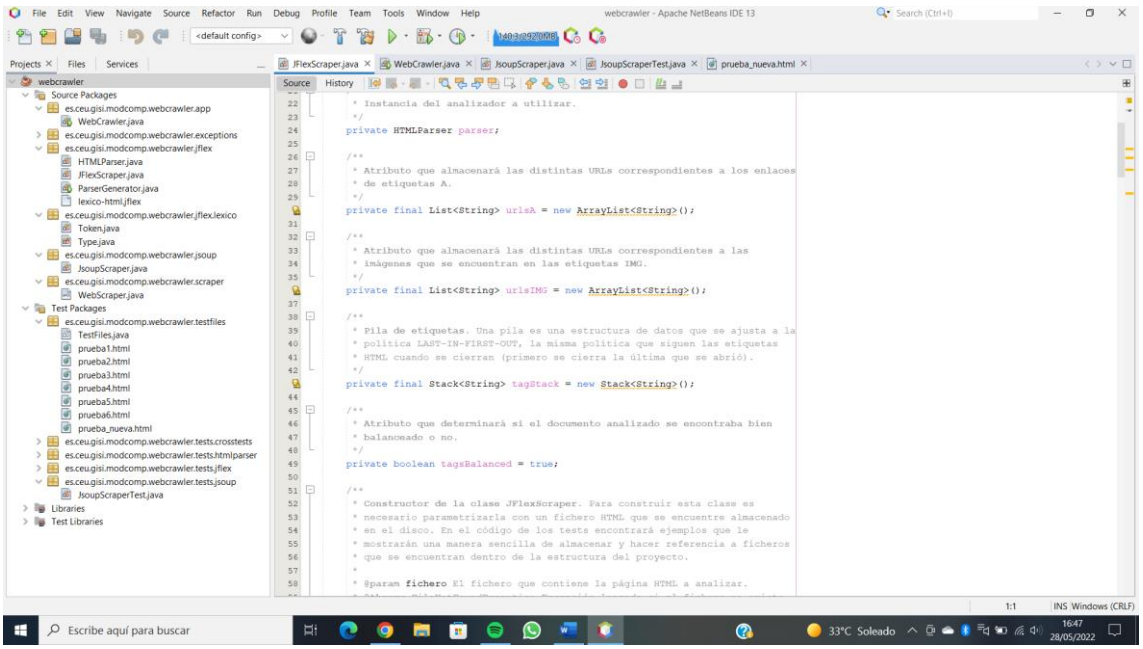
La práctica se centrará en los conocimientos sobre autómatas finitos o expresiones regulares para la elaboracion del código, ademas se utilizarán técnicas de *Web-Scraping.*

Web-Scraping es el término utilizado para distinguir aquellas técnicas que extraen datos de los sitios web. Analizan las páginas web, recopilan información (aquella para la que estén diseñadas) y la almacena para luego pder ser procesada. En este proyecto se simulará el comportamiento de *Google* para descubrir nuevas páginas, es decir, usar técnicas de *web-scraping* para obtener los hiperenlaces de las páginas web además de poder obtener todas las imágenes de esa página.

*Parte 1 - Jflex:

Uso realizado de este:

Para poder lograr el objetivo de obetener los hiperenlaces y las imágenes de una página web, o en este caso de los archivos HTML de prueba ya predetermiandos en el proyecto, ya se nos daban las herramientas necesarias para poder trabajar. Además de ya tener de por sí los conocimientos necesarios para poder completarla. Se nos proporcionaba: arrays ya creados para poder almacenar los hiperenlaces y las imágenes por separado; otro array que funciona como pila para poder comprobar que las etiquetas del código HTML se abren y cierran correctamente y así poder concluir si el código está bien balanceado o no.



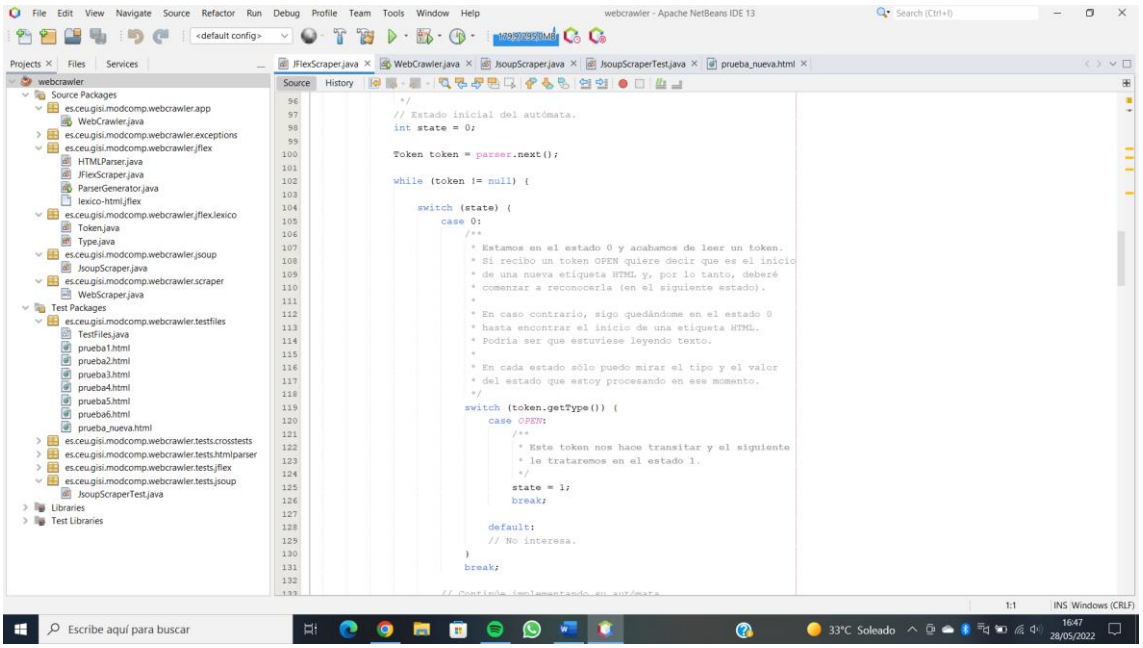
Ejemplos de apertura y cierre de etiquetas y tipos:

1.- <html> (Apertura) </html> (Cierre)

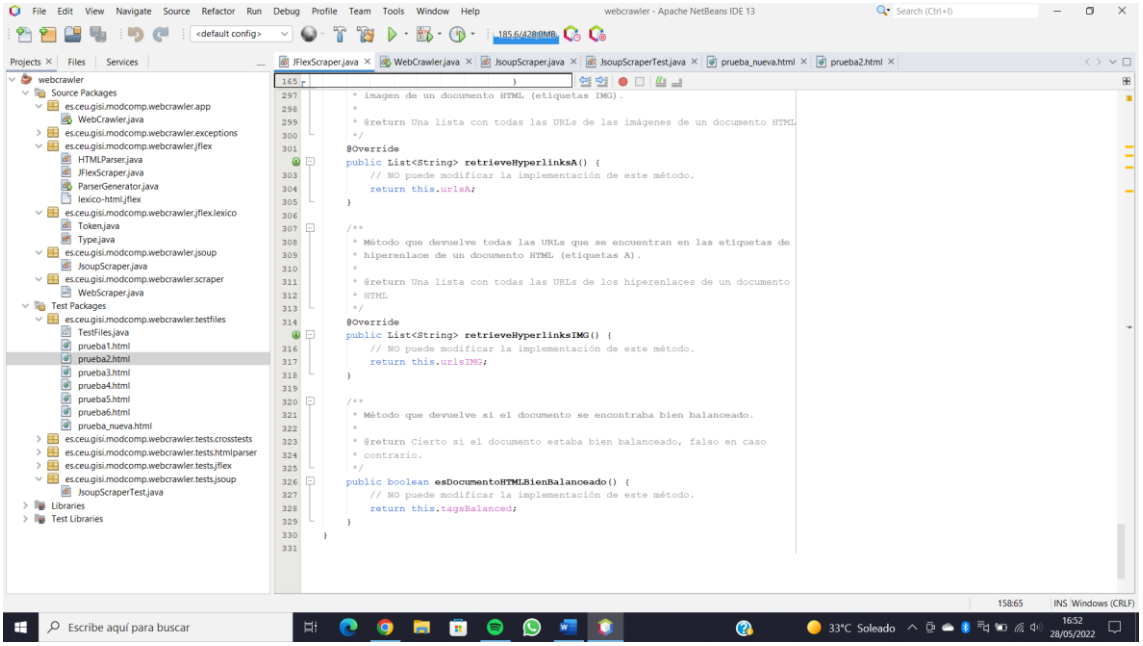
2.- <IMG src="brushedsteel.jpg"/> (Apertura al inicio y cierre al final con la barra)(Ejemplo de etiqueta de tipo imagen también)

3.- <A href="http://www.bbc.co.uk">link</A> (Ejemplo de hiperenlace a ser almacenado)

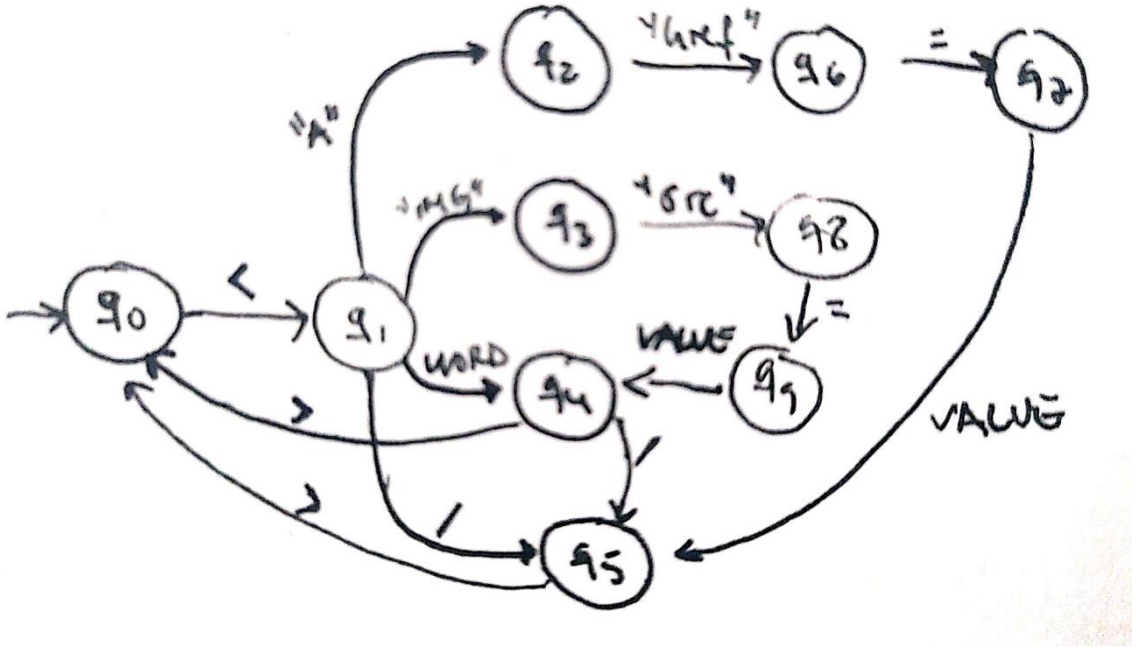
Después se nos proporcionaba el principio del código a completar del autómata. Este autómata se diseñó en papel comprobando bien las transiciones de este para luego poder pasarlo al lenguaje de programacion usado: *Java.*



Finalmente en Jflex se nos proporcionaba los métodos que almacenaban los hiperenlaces, las imágenes y la pila. Estos métodos devuelven lestos hiperenlaces, imágenes y el balance de los códigos de pruba usados.



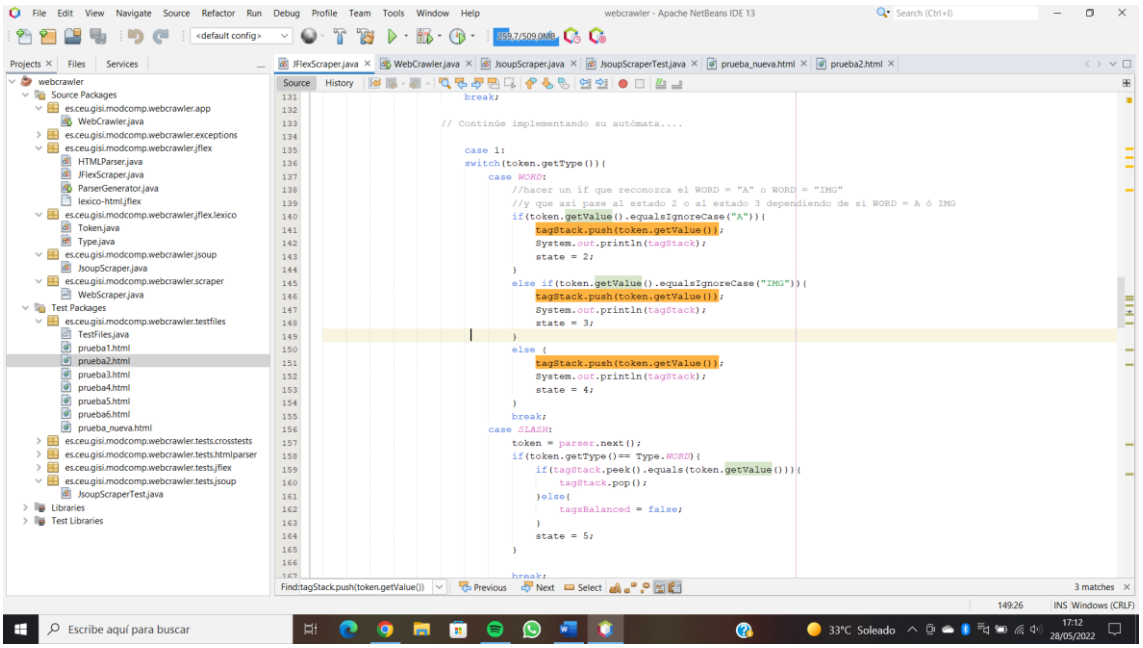
Autómata usado:



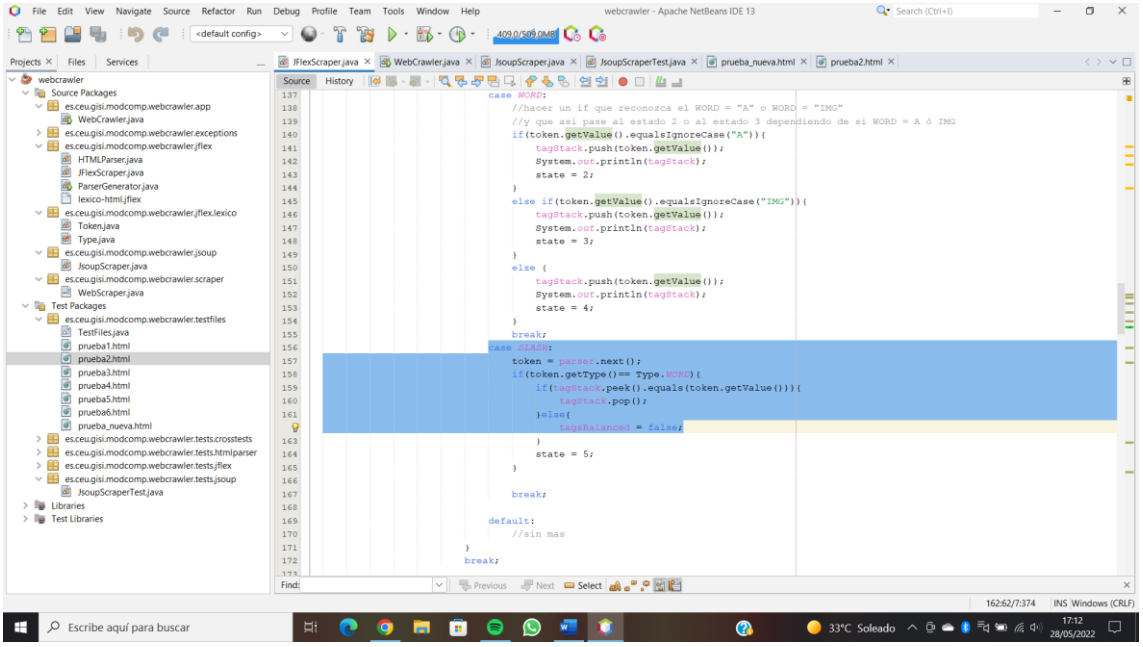
Este autómata representa los diferentes estados en los que se puede estar, se has puesto los casos especiales que son aquellos que nos interesan después de todo (hiperenlaces e imágenes). Para ello hemos creado este autómata para que cuando lea una etiqueta correspondiente con la estuctura de una etiqueta que nos interese almacenar, añada al array correspondiente su respectivo valor (lo que llamamos la URL de la imagen o el link del hiperenlace). Y además hacemos que vaya haciéndolo continuamente para que de una sola lectura del código pueda almacenar todos estos valores.

Decisiones e implementaciones de código:

La primera decisión tomada fue la creación de un autómata que pudiese leer todas las etiquetas en las cuales estábamos interesados. Procesamos el autómata resultante al proyecto mediante código java, haciendo así una primera toma de contacto. Posteriormente nos interesaba que almacenase todos los valores de los que hablábamos antes en sus respectivos arrays, por lo que implemtentamos un par de líneas más de código en aquellos estados en los que al leer el valor los almacenase.



Después decidimos comprobar si verdaderamente almacenaba todos los valores y si lo hacía en sus respectivos arrays con las pruebas proporcionadas por el mismo proyecto. Al ver que todo iba bien y funcionaba, lo siguiente que decidimos hacer fue el balanceado del código HTML, es decir, comprobar si todas las etiquetas que se abrían se cerraban. Pero además de comprobar que hacen eso, comprobar que lo hacían en el orden correcto para poder ir borrando la pila. En caso de que las etiquetas se cierren el orden disparejo la pila no llegaria a borrar todos los elementos almacenados en ella indicando que el código está mal balanceado.



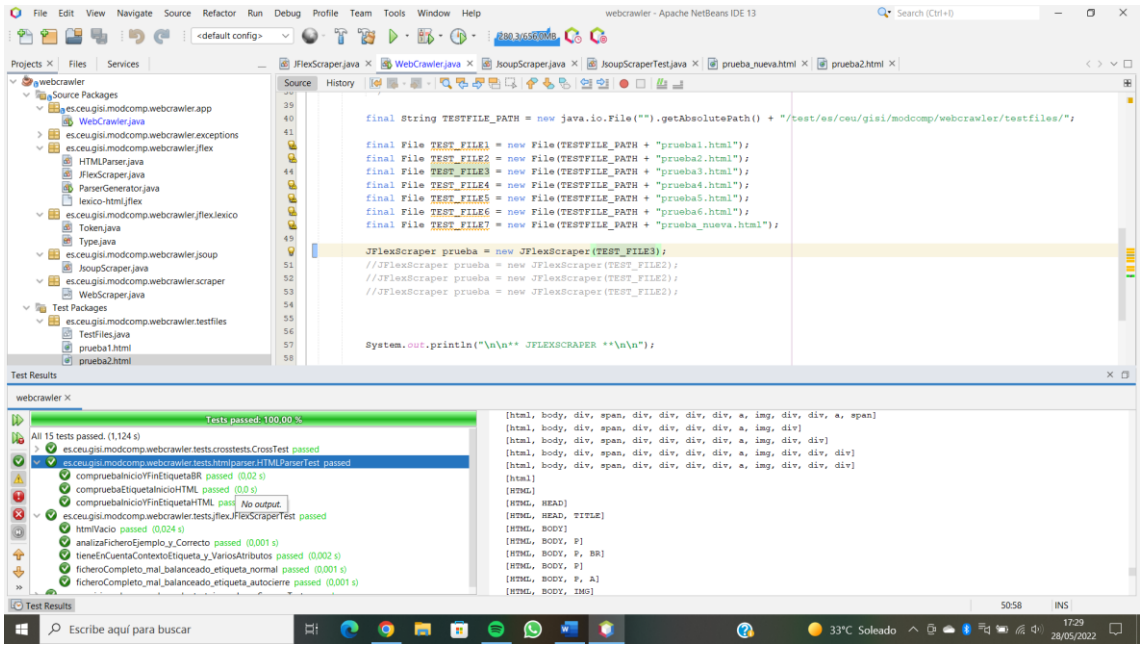
Pruebas realizadas:

Las primeras pruebas realizadas era ver si el autómata compilaba y no había ningun error de código y en caso contrario poder arreglarlo.

Las siguientes pruebas fueron que el código implemetado representando al autómata pudiese leer todas las etiquetas de las pruebas. Ya que al principio el autómata solo queríamos que leyese aquellas que nos interesaban pero para luego hacer el balance del código no podría conseguirse así por lo que hicimos estas pruebas.

Posteriormente queríamos comprobar si al inicializar el autómata, este lograba almacenar todos los valores de imágenes en el array de imágenes y los hiperenlaces en su propio array. Esto lo comprobamos mirando los resultados que nos daba la prueba con los enlaces que aparecian el los codígos predeterminados.

Finalmente nos interesaba comprobar que el codígo de estas pruebas predetermiandas estuvieran balanceadas o no y que en los rsultados de las pruebas no indicase si era así. Por lo que comprobamos mirando el código también al igual que hicimos con las pruebas de los valores para saber si los resultados del balance eran ciertos o algo fallaba. Al principio de estas pruebas salía que todos los códigos estaban balanceados lo cual no era cierto porque la mitad de ellos no lo estaban. Los siguientes resultados que obtuvimos fueron casi contrarios, la primera prueba que salio bien balanceada acertó. En cambio para las pruebas dos y tres nos salía balanceadas y lo estabas y ahora salían no balanceadas por lo que algo fallaba. Además de esto las pruebas que salían balanceadas cuando no lo estaban, salía que ahora no lo estaban lo cual estaba bien. Faltaba algo que pudiera mostrarnos bien el balance de los códigos. Al cabo de más pruebas y ensayo y error logramos que saliese correctamente el balance.

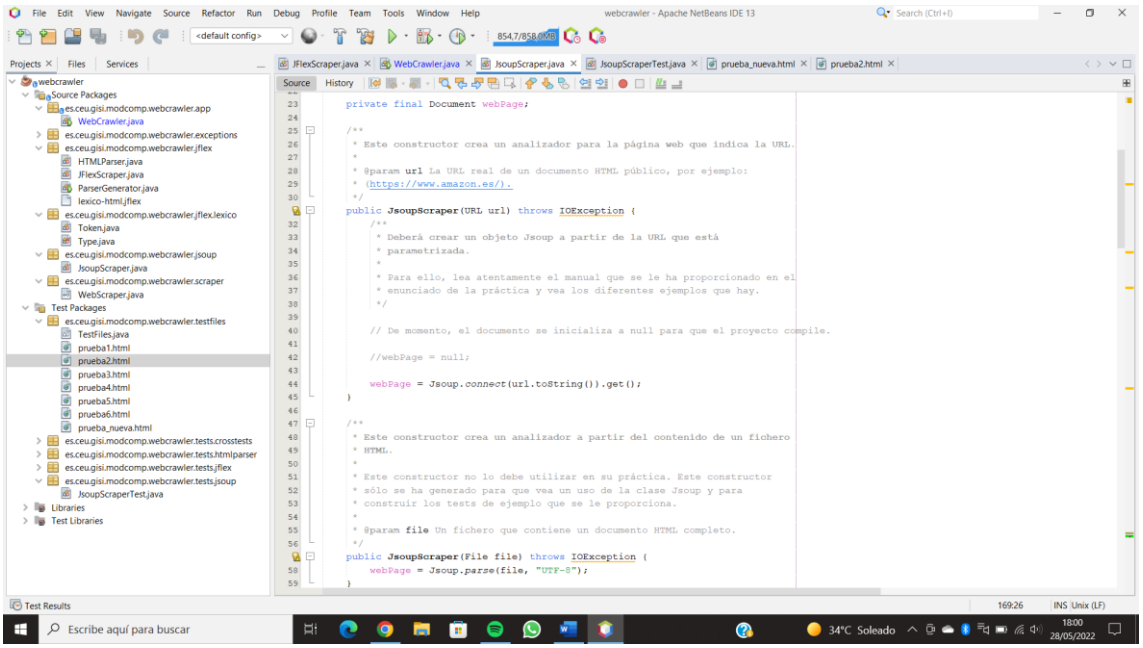


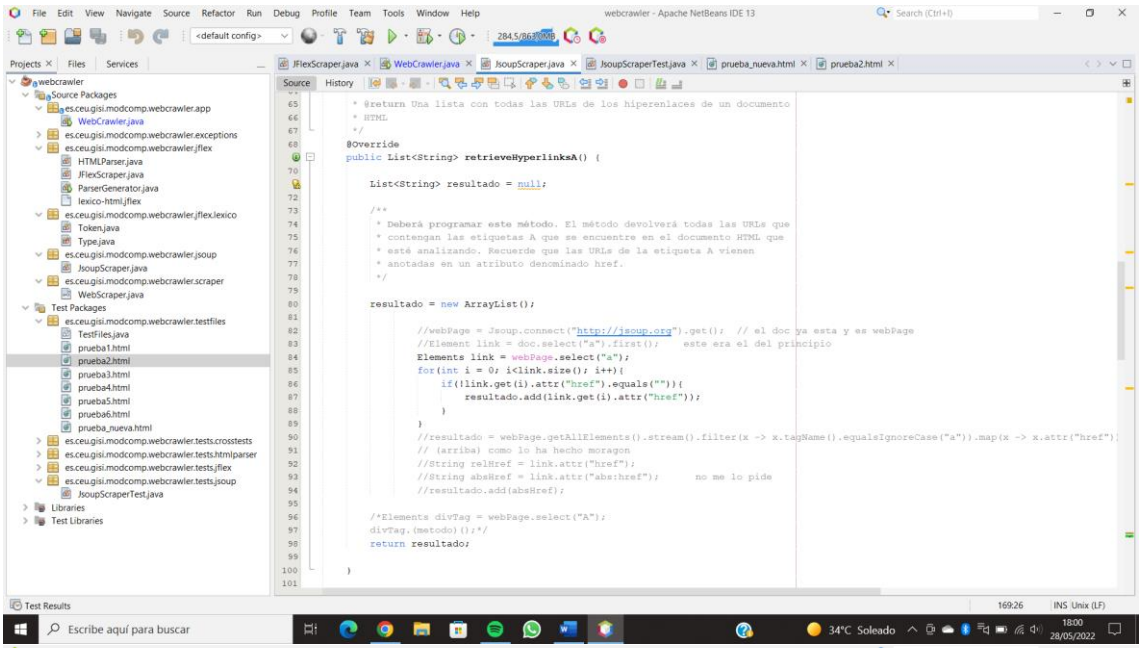
Creo que el conjunto de pruebas realizado es correcto ya que intentamos ver en qué puede llegar a fallar y arreglarlo. Además de que el código implementado responde y se ajusta a los objetivos propuestos.

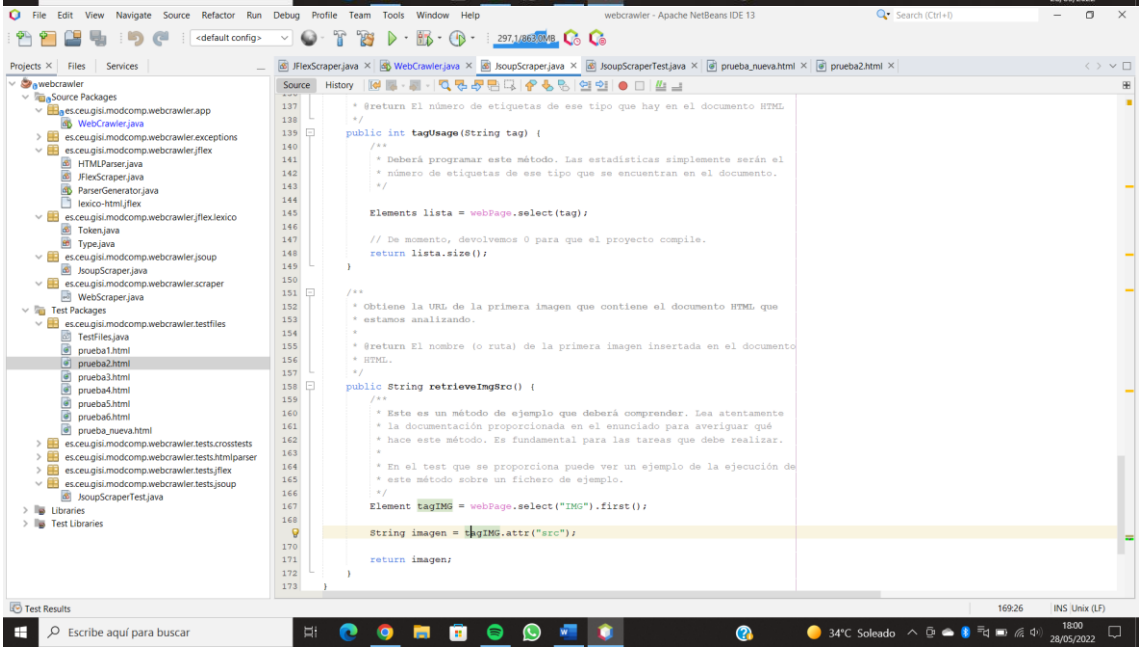
*Parte 2 – Jsoup:

Uso realizado:

Al igual que con Jflex, se nos proporcionaban las herramientas necesarias con las que poder trabajar aunque resultaba más lioso porque era algo que no entendía muy bien. La clase de Jsoup nos proporciona un archivo de tipo documento llamado webPage el cual tendremos que usar para recorrer ese documento y recuperar o coger los valores que necesitamos y añadirlos al array ya creado llamado resultado. Este array esta creado en los métodos para almacenar los hiperenlaces y las imágenes con el mismo nombre. (No hay ningún problema al estar creados en métodos diferentes). Además al final del código de la clase nos proporciona método que debemos usar par poder completar aquellos que tienen que almacenas los valores que queremos.

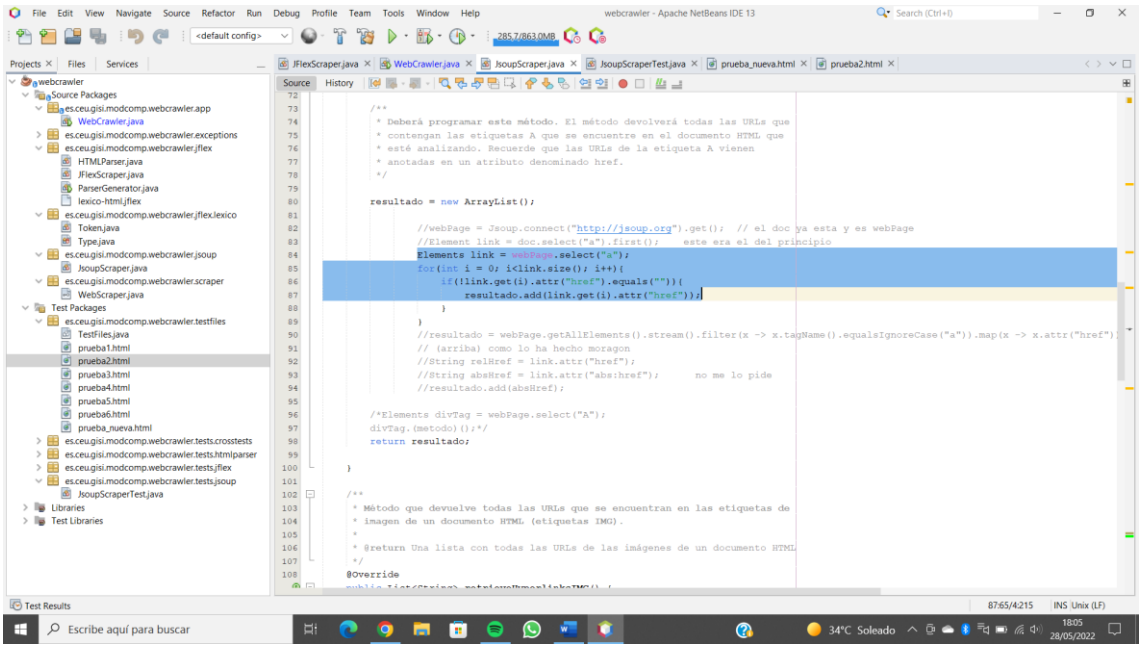






Decisiones e implementaciones de código:

Al no entender mucho sobre Jsoup, lo primero que debía hacer era documentarme sobre ello y me dediqué a buscar y entender qué era y para qué servía. Al mismo tiempo podía ver ejemplos y diferentes métodos para poder lograrlo y buscando multiples ejemplos y probándolos y viendo cual era el más intuitivo y fácil de entender e implementarlo.



Pruebas realizadas:

Las pruebas realizadas con Jsoup fueron pocas ya que de primeras funcionaba perfectamente y me devolvía lo que tenía que devolver pero solo uno de los muchos que debería. Suponiendo que el valor almacenado era el primero de todos loq ue tenía que almacenar, creé un bucle que pudiese recorrer todo el documento:

```
for(int i = 0; i<link.size(); i++){
    if(!link.get(i).attr("href").equals("")){
    resultado.add(link.get(i).attr("href"));
    }
```

Y así pude almacenar todas las etiquetas de un tipo que quería, solo me quedaba implementar el mismo código para el otro método cambiándolo par aque fuera para etiquetas de tipo imagen.

Creo que el conjunto de pruebas realizado es correcto ya que me base en pequeñas pero exhaustivas pruebas de ensayo y error con diferentes métodos para poder intentar logar el objetivo. Además de que el código implementado responde y se ajusta a estos objetivos propuestos.