UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
INGENIERÍA DEL SOFTWARE

ANÁLISIS DE OPINIÓN EN PRENSA

PRESS OPINION ANALYSIS

Realizado por
Sergio Nunes Díaz

Tutorizado por
Eduardo Guzmán de los Riscos

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, JUNIO DE 2019

Fecha defensa: de julio de 2019

Fdo. El/la Secretario/a del Tribunal

Resumen

Hoy en día, el avance tecnológico ha permitido a la sociedad poder obtener y manejar grandes volúmenes de información que hasta hace unos años, era impensable. Esto facilita a su vez la aplicación de la inteligencia artificial para, por ejemplo, a través de una imagen, poder conocer con un alto porcentaje de acierto, si el usuario padece un cáncer de piel, o mismamente, el uso de esta misma inteligencia artificial para poder realizar análisis de opiniones. En este último caso podemos encuadrar este TFG, que, gracias al uso de la programación neurolingüística, nos permite hacer realidad el hecho de poder analizar los comentarios que las personas suben a internet, de manera automatizada y ordenada, para poder realizar análisis y extraer nuestras propias conclusiones. En este caso veremos que podemos extraer información suficiente de los comentarios como para categorizarlos, y así, poder saber cuáles son los ideales políticos que más abundan en las redes y en todo un país en base a la información obtenida de la prensa.

Con este proyecto podremos visualizar en tiempo real (con noticias actuales), cuáles son las opiniones actuales de las personas que comentan noticias a través de internet, en este caso, en la prensa online, con dos de los diarios españoles más influyentes, además de un diario local de Málaga. De esta manera, también será posible analizar cualquier noticia, en general, de cualquiera de estos tres diarios.

Palabras clave: Análisis de Opinión, Spring Boot, Extracción de información web, política

Abstract

Nowadays, the technology advance gave us the opportunity of obtaining and handle a big quantity of information that, until a few years ago was unthinkable. One example can be the use of artificial intelligence to know if a person can suffer a skin cancer with one photo, or the use of the artificial Intelligence to do an opinion analysis. In that last case we can situate this project, that can be real due to neuro-linguistic programming. It permits us the way to make real the analysis of comments the people share on Internet, automatized and ordered to draw our own conclusions; this time, we will see we can get enough Information from comments to categorize them, and then, can know which are the ideal politics that abound in networks and in all the country.

This project will permit us see, in real time (with current news), which are the actual opinions of people who share their point of view in online press, with two of the most influential Spanish newspaper, also one local newspaper from Málaga. In this way, It will be possible analyze any notice from these newspapers.

Keywords: Opinion analysis, Spring Boot, Web Information Retrieval, politic

Agradecimientos

Para empezar, me encantaría poder agradecer a mis padres todo lo que han hecho por mí, tanto a mi madre Luisa Díaz Bermejo, como a mi padre Paulo Ricardo Nunes, ya que sin ellos, sin duda alguna, jamás hubiera llegado donde estoy actualmente. Les agradezco tanto el apoyo moral recibido, como la financiación para poder realizar mis estudios. Ellos han sido el pilar fundamental en mi vida y quienes me han enseñado que los sueños hay que perseguirlos y que no debo rendirme ante ningún obstáculo que me plantee la vida. Desde aquí, MUCHÍSIMAS GRACIAS, OS QUIERO.

También me gustaría aprovechar este momento para agradecer a mis amigos: Álvaro García, Carlos Gamero y Alberto Reyes, por su apoyo y amistad a lo largo de todo el grado, ya que hemos podido aprender muchísimo los unos de los otros en estos cuatro años.

Por último, y no por ello menos importante, gracias a Eduardo Guzmán, por la tutorización del TFG y su gran labor como docente.

Índice

Índice de figuras

1

Introducción

1.1 Motivación

Aunque la programación neurolingüística no está tan avanzada como otras áreas de investigación, pues según múltiples estudios, la mayor tasa de acierto obtenida por un proyecto que hace uso de ella es de, aproximadamente, un 85% (teniendo una base de entrenamiento fiable y amplia), puede darnos mucho juego, ya que aunque no es una herramienta que nos de un 100% de aciertos, nos puede proporcionar una probabilidad de acierto mucho mayor que si realizásemos un análisis de manera aleatoria.

La motivación que podemos encontrar en este proyecto es la de poder analizar la opinión de todo un país, en tiempo real, y al alcance de cualquier persona que tuviera acceso a internet, tanto ejecutando el servidor en local, como accediendo a través de una página web a un servidor externo, es decir, una herramienta al alcance de todos. Para ello se utilizará como fuente de información los comentarios que los usuarios realizan en la prensa on-line sobre noticias sobre política.

De esta manera, una herramienta de estas características facilitaría el poder inferir, de forma indirecta, cierto conocimiento sobre las ideas políticas que abundan en un país sin necesidad de realizar encuestas de opinión.

1.2 Objetivos

La idea principal del TFG es, por tanto, predecir, a través de los comentarios que la gente publica en la prensa online, cuál sería el ganador de las elecciones en un país democrático como es España, pudiendo clasificar los comentarios mediante un análisis de sentimientos.
Para ello, se ha desarrollado el TFG en dos partes fundamentales:

- **Servidor:** En esta parte es donde se realiza el análisis de sentimientos pertinente para poder inferir las cuestiones aclaradas más arriba, además de utilizarse como controlador de la interfaz web.
  Para ello, desde el servidor se hacen las llamadas pertinentes a las páginas web de los periódicos online, recogiendo las noticias, para luego utilizar las APIs (de las que hacen uso para almacenar los comentarios), y así, recogerlos para poder realizar un análisis de opinión sobre estos. Después de realizar el análisis, los resultados se envían al cliente para que pueda visualizarlos.
- **Cliente:** Será la parte visible por el usuario, la cual es una vista agradable y funcional. Por tanto, hablamos de una interfaz adaptativa (*responsive*) y clara, que se puede integrar bien en cualquier dispositivo y que, a su vez, permite que los usuarios puedan entender fácilmente el uso de la aplicación.
  El cliente realizará las llamadas asíncronas al servidor mediante Javascript, para conocer si el número de comentarios/noticias que el usuario ha solicitado analizar están disponibles ya para ser analizados, en caso afirmativo, el servidor realiza el análisis de sentimientos y devuelve los resultados, en otro caso, el servidor recoge más comentarios/noticias y cuando los ha conseguido y los termina de analizar, los envía al cliente. Este último muestra gráficas de los resultados que el servidor le ha proporcionado para que el usuario pueda estudiar de forma visual los resultados.

1.3 Estructura de la memoria

En este apartado se describe brevemente la estructura de la memoria con cada uno de sus capítulos, resumiendo el contenido de cada uno de ellos.

2. Estado del arte. En esta parte se contextualiza el proyecto dentro del marco histórico y tecnológico actual, recogiendo una noticia de interés y el uso que se hace de

las tecnologías de análisis de sentimientos para estudiar a la población.

3. Descripción del proyecto. Aquí se incluye un breve resumen sobre el proyecto y la metodología utilizada para su realización, así como los casos de uso y de prueba y la arquitectura de la aplicación.

4. Librería ScrapAPI. En este capítulo se describe la base en la que se sustenta el proyecto principal, ya que se define la librería que permitirá que la aplicación web final pueda recopilar los comentarios y las noticias de las páginas web de la prensa. Se incluye tanto una introducción sobre su estructura y uso, las librerías externas de las que se hacen uso, así como la implementación, el modelo y las clases de la misma.

5. Proyecto PressOpinionAnalyzer. En esta parte se define el proyecto núcleo principal de este trabajo, donde se hace uso de la librería ScrapAPI. Esta aplicación es la que ejecutaremos y la que permite al usuario poder analizar los periódicos on-line. En este capítulo se describen tanto las librerías utilizadas, el framework, el modelo, sus clases y la implementación.

6. Categorizador de opiniones. Aquí se define la parte de inteligencia artificial del proyecto, en la que se incluye tanto una breve introducción con la librería utilizada para la creación del categorizador, como el entrenamiento llevado a cabo y las pruebas realizadas para conocer su efectividad.

7. Problemas y soluciones. En este capítulo se resumen tanto los errores encontrados a la hora de desarrollar todo el proyecto (la librería ScrapAPI, el proyecto PressOpinionAnalyzer y el categorizador) como las soluciones que se le han dado a cada uno de ellos.

8. Conclusiones. Este capítulo realiza una reflexión personal acerca de todo lo aprendido con todo este trabajo.

9. Trabajo futuro. En esta parte se incluye una breve reflexión sobre las posibles ampliaciones del trabajo, en el caso en el que se tuvieran al alcance tanto las herramientas físicas (hardware), como la mano de obra para poder llevar este software a un nivel internacional y preciso.

Anexo A. Manual de usuario. En este anexo se describe el manual completo que puede seguir el usuario para el uso de la aplicación final, mostrando de manera sencilla y con imágenes los pasos a seguir para que el usuario no pueda tener duda alguna.

Anexo B. Manual de instalación. En este anexo se incluye un manual completo de instalación de la aplicación, mostrando imágenes con los pasos (y los resultados que obtendremos al realizarlos), dando varias opciones de instalación para que el usuario sea libre de elegir la que más se ajuste a él.

2

Estado del arte

Hoy en día, en pleno siglo XXI, con el auge de la minería de datos y del análisis de opinión, todo el mundo quiere poder realizar un estudio acerca de los gustos y opiniones de las personas. Tanto políticos, como empresas tienen el deseo de poder analizar a la gente a través de sus publicaciones en redes sociales, de internet, o en general, pues a través de esta información pueden rectificar su comportamiento y poder llegar a un público en específico, o, simplemente, poder empezar un negocio en concreto dirigido a un público que se ha detectado que es muy grande y no existe un servicio que pueda atenderles dicha necesidad.

Por ello, teniendo en cuenta que hoy en día, saber lo que le gusta a la gente o lo que piensan es algo vital para las empresas y la política de todo el mundo, podemos destacar varios ejemplos de empresas que recopilan o compran información, como Google (a través de las búsquedas en su navegador o del uso que le podemos dar a cualquiera de sus aplicaciones), Facebook (como red social por excelencia, posee los datos de millones de personas alrededor de todo el mundo, tanto números de teléfonos, como gustos, aficiones, relaciones, etc.). Además de Facebook, también recopilan muchísima información otras aplicaciones, como Instagram y WhatsApp. o Amazon recoge información a partir de las múltiples plataformas que ofrecen, además de comprarla, para poder ofrecer a sus clientes productos que sabe que podrían interesarles.
Porque, a quién no se le ha estropeado el portátil, el teléfono o cualquier dispositivo y que, en cuestión de horas, la publicidad dirigida nos ofrezca el dispositivo que

necesitamos o la empresa que pudiera darnos un posible arreglo. Todo esto es gracias a la información que las empresas recopilan, compran y venden sobre nosotros.

En el caso de la política, podemos situar una gran noticia de actualidad, la cual ha tenido una alta repercusión por la polémica que creó. Esta polémica radica en Trump y la empresa Cambridge Analytica, que en el periodo de 2014-2016 dio servicio al candidato republicano Donald Trump, actual presidente de los Estados Unidos de América, proporcionándoles información gracias al análisis que pudieron hacer a través de la red social Facebook, la cual había permitido la extracción de datos al psicólogo Aleksandr Kogan, pensados para el estudio de su disciplina, que más tarde, compartiría con la compañía Cambridge Analytica, de manera ilegal, pues violó las políticas de protección de datos de Facebook, ya que se le habían cedido dichos datos para un fin en concreto. Con todo ello, Cambridge Analytica obtuvo la información de unos cincuenta millones de americanos, y permitiendo así, a través de exhaustivos estudios por parte de la compañía, la realización de una herramienta que permitiera a Trump conocer a sus votantes e influir en sus decisiones, así como poder saber cómo orientar su política para captar el mayor número de votantes.
Con todo ello, hubo una gran polémica, ya que se hizo uso de un análisis de sentimientos a través de los datos de cincuenta millones de americanos, obtenidos de manera totalmente ilícita y para el beneficio de un partido político.

Este es un caso reciente, aunque seguramente haya más casos que no hayan salido a la luz, además de los que se encuentran dentro del marco legal, como pueden ser las empresas que capturan, compran y venden información (las anteriormente mencionadas).

3

Descripción del proyecto

3.1 Metodología

El proyecto constará de una **aplicación web** que sea capaz de ofrecer al usuario, la posibilidad de analizar los comentarios de las noticias en la web, y así, dar a conocer, cuál es la orientación política de la mayoría de las personas que introducen comentarios en ellas.

De esta manera, damos a conocer qué partido político predomina en las redes, o qué perfil político abunda dependiendo del periódico online que estemos analizando.

Se hará uso de la tan conocida metodología **Scrum**, respetando cada una de las fases, ya que obviar la realización de una de ellas puede desembocar en errores posteriores, los cuales, pueden costar mucho de arreglar y pueden hacer que tengamos un código menos desacoplado, y así, no poder reutilizar las clases del modelo. En definitiva, tardar mucho más en realizar el proyecto y que el resultado pueda llegar a no ser satisfactorio.

3.2 Casos de uso

A continuación se mostrarán los casos de uso, en forma de tabla para que sean más visuales, entre los diferentes actores que participan: usuario y sistema.

Para definir los casos de uso, nos basaremos en el estándar del **NIST** (National Institute of Standards and Technology) para realizar buenas prácticas de ingeniería.
Antes de ver todas las tablas, podemos visualizar un diagrama que resume todos los casos de uso del sistema, y la manera en que interactúan.

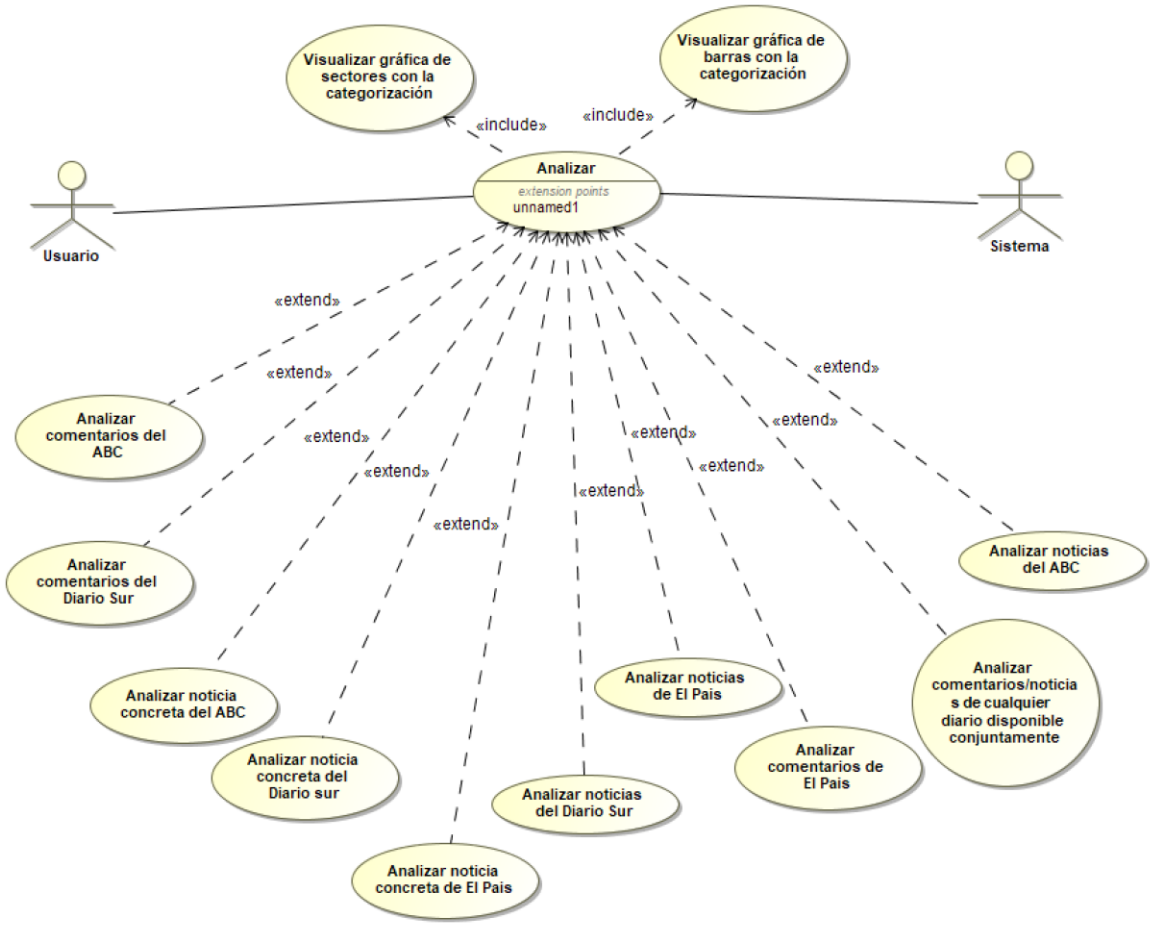

Figura 3.1 Diagrama de casos de uso

3.2.1 El usuario podrá analizar comentarios del ABC

Descripción	El usuario podrá analizar el número de comentarios del ABC que desee.
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Escenario principal	
1. El usuario selecciona la opción de analizar noticias automáticamente. 2. El usuario selecciona la opción de recolectar información del ABC. 3. El usuario selecciona que desea analizar comentarios. 4. El usuario pone el número de comentarios (mayor que 0) que desea analizar. 5. El usuario pulsa el botón "Analizar". 6. El sistema comprueba que tiene ya suficientes comentarios para analizar. 7. El sistema categoriza los comentarios.	
Escenario alternativo	
4.1.1 El número de comentarios especificado es menor o igual a 0. 4.1.2 El sistema mostrará al usuario gráficas vacías. 6.2.1 El sistema comprueba que no tiene suficientes comentarios para analizar. 6.2.2 El sistema obtiene más comentarios actuales del ABC. 6.2.3 El sistema categoriza los comentarios.	
Postcondiciones	1. El usuario será capaz de visualizar las estadísticas que el sistema ha realizado sobre el número de comentarios especificado.

3.2.2 El usuario podrá analizar comentarios del Diario Sur

Descripción	El usuario podrá analizar el número de comentarios del Diario Sur que desee.
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Escenario principal	
1. El usuario selecciona la opción de analizar noticias automáticamente. 2. El usuario selecciona la opción de recolectar información del Diario Sur. 3. El usuario selecciona que desea analizar comentarios. 4. El usuario pone el número de comentarios (mayor que 0) que desea analizar. 5. El usuario pulsa el botón "Analizar". 6. El sistema comprueba que tiene ya suficientes comentarios para analizar. 7. El sistema categoriza los comentarios.	
Escenario alternativo	
4.1.1 El número de comentarios especificado es menor o igual a 0. 4.1.2 El sistema mostrará al usuario gráficas vacías. 6.2.1 El sistema comprueba que no tiene suficientes comentarios para analizar. 6.2.2 El sistema obtiene más comentarios actuales del Diario Sur. 6.2.3 El sistema categoriza los comentarios.	
Postcondiciones	1. El usuario será capaz de visualizar las estadísticas que el sistema ha realizado sobre el número de comentarios especificado.

3.2.3 El usuario podrá analizar comentarios de El Pais

Descripción	El usuario podrá analizar el número de comentarios de El País que desee.
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Escenario principal	
1. El usuario selecciona la opción de analizar noticias automáticamente. 2. El usuario selecciona la opción de recolectar información de El País. 3. El usuario selecciona que desea analizar comentarios. 4. El usuario pone el número de comentarios (mayor que 0) que desea analizar. 5. El usuario pulsa el botón "Analizar". 6. El sistema comprueba que tiene ya suficientes comentarios para analizar. 7. El sistema categoriza los comentarios.	
Escenario alternativo	
4.1.1 El número de comentarios especificado es menor o igual a 0. 4.1.2 El sistema mostrará al usuario gráficas vacías. 6.2.1 El sistema comprueba que no tiene suficientes comentarios para analizar. 6.2.2 El sistema obtiene más comentarios actuales de El País. 6.2.3 El sistema categoriza los comentarios.	
Postcondiciones	1. El usuario será capaz de visualizar las estadísticas que el sistema ha realizado sobre el número de comentarios especificado.

3.2.4 El usuario podrá analizar noticias del ABC

Descripción	El usuario podrá analizar el número de noticias del ABC que desee.
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Escenario principal	
1. El usuario selecciona la opción de analizar noticias automáticamente. 2. El usuario selecciona la opción de recolectar información del ABC. 3. El usuario selecciona que desea analizar noticias. 4. El usuario pone el número de noticias (mayor que 0) que desea analizar. 5. El usuario pulsa el botón "Analizar". 6. El sistema comprueba que tiene ya suficientes noticias para analizar. 7. El sistema categoriza los comentarios de las noticias.	
Escenario alternativo	
4.1.1 El número de noticias especificado es menor o igual a 0. 4.1.2 El sistema mostrará al usuario gráficas vacías. 6.2.1 El sistema comprueba que no tiene suficientes noticias para analizar. 6.2.2 El sistema obtiene más noticias actuales del ABC. 6.2.3 El sistema categoriza los comentarios de las noticias.	
Postcondiciones	1. El usuario será capaz de visualizar las estadísticas que el sistema ha realizado sobre el número de comentarios especificado.

3.2.5 El usuario podrá analizar noticias del Diario Sur

Descripción	El usuario podrá analizar el número de noticias del Diario Sur que desee.
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Escenario principal	
1. El usuario selecciona la opción de analizar noticias automáticamente. 2. El usuario selecciona la opción de recolectar información del Diario Sur. 3. El usuario selecciona que desea analizar noticias. 4. El usuario pone el número de noticias (mayor que 0) que desea analizar. 5. El usuario pulsa el botón "Analizar". 6. El sistema comprueba que tiene ya suficientes noticias para analizar. 7. El sistema categoriza los comentarios de las noticias.	
Escenario alternativo	
4.1.1 El número de noticias especificado es menor o igual a 0. 4.1.2 El sistema mostrará al usuario gráficas vacías. 6.2.1 El sistema comprueba que no tiene suficientes noticias para analizar. 6.2.2 El sistema obtiene más comentarios actuales del Diario Sur. 6.2.3 El sistema categoriza los comentarios de las noticias.	
Postcondiciones	1. El usuario será capaz de visualizar las estadísticas que el sistema ha realizado sobre el número de comentarios especificado.

3.2.6 El usuario podrá analizar noticias de El Pais

Descripción	El usuario podrá analizar el número de comentarios de El País que desee.
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Escenario principal	
1. El usuario selecciona la opción de analizar noticias automáticamente. 2. El usuario selecciona la opción de recolectar información de El País. 3. El usuario selecciona que desea analizar noticias. 4. El usuario pone el número de noticias (mayor que 0) que desea analizar. 5. El usuario pulsa el botón "Analizar". 6. El sistema comprueba que tiene ya suficientes noticias para analizar. 7. El sistema categoriza los comentarios de las noticias.	
Escenario alternativo	
4.1.1 El número de noticias especificado es menor o igual a 0. 4.1.2 El sistema mostrará al usuario gráficas vacías. 6.2.1 El sistema comprueba que no tiene suficientes noticias para analizar. 6.2.2 El sistema obtiene más comentarios actuales de El País. 6.2.3 El sistema categoriza los comentarios de las noticias.	
Postcondiciones	1. El usuario será capaz de visualizar las estadísticas que el sistema ha realizado sobre el número de comentarios especificado.

3.2.7 El usuario podrá analizar una noticia en concreto del ABC

Descripción	El usuario podrá analizar la noticia concreta del ABC que desee.
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Escenario principal	
1. El usuario selecciona la opción de analizar una noticia en concreto. 2. El usuario indica la URL de la noticia del ABC que desea analizar. 3. El usuario pulsa el botón "Analizar" 4. El sistema categoriza los comentarios de la noticia.	
Escenario alternativo	
2.1.1 La URL indicada no existe. 3.1.2 El sistema mostrará al usuario gráficas vacías. 2.2.1 La URL indicada no pertenece a ninguno de los diarios ofrecidos. 2.2.2 El sistema mostrará al usuario gráficas vacías.	
Postcondiciones	1. El usuario será capaz de visualizar las estadísticas que el sistema ha realizado sobre la noticia que se indicó.

3.2.8 El usuario podrá analizar una noticia en concreto del Diario Sur

Descripción	El usuario podrá analizar la noticia concreta del Diario Sur que desee.
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Escenario principal	
1. El usuario selecciona la opción de analizar una noticia en concreto. 2. El usuario indica la URL de la noticia del Diario Sur que desea analizar. 3. El usuario pulsa el botón "Analizar" 4. El sistema categoriza los comentarios de la noticia.	
Escenario alternativo	
2.1.1 La URL indicada no existe. 3.1.2 El sistema mostrará al usuario gráficas vacías. 2.2.1 La URL indicada no pertenece a ninguno de los diarios ofrecidos. 2.2.2 El sistema mostrará al usuario gráficas vacías.	
Postcondiciones	1. El usuario será capaz de visualizar las estadísticas que el sistema ha realizado sobre la noticia que se indicó.

3.2.9 El usuario podrá analizar una noticia en concreto de El Pais

Descripción	El usuario podrá analizar la noticia concreta de El País que desee.
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Escenario principal	
1. El usuario selecciona la opción de analizar una noticia en concreto. 2. El usuario indica la URL de la noticia de El País que desea analizar. 3. El usuario pulsa el botón "Analizar" 4. El sistema categoriza los comentarios de la noticia.	
Escenario alternativo	
2.1.1 La URL indicada no existe. 3.1.2 El sistema mostrará al usuario gráficas vacías. 2.2.1 La URL indicada no pertenece a ninguno de los diarios ofrecidos. 2.2.2 El sistema mostrará al usuario gráficas vacías.	
Postcondiciones	1. El usuario será capaz de visualizar las estadísticas que el sistema ha realizado sobre la noticia que se indicó.

3.2.10 El usuario podrá visualizar una gráfica de barras con la categorización de las noticias y los comentarios

Descripción	El usuario podrá visualizar una gráfica de barras con la categorización realizada.
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Escenario principal	
1. El usuario selecciona el número de noticias/comentarios del periódico que desee o la noticia concreta a analizar. 2. El usuario pulsa el botón "Analizar". 3. El sistema categoriza los comentarios. 4. El sistema muestra al usuario una gráfica de barras con los resultados calculados anteriormente.	
Escenario alternativo	
1.1.1 El número de comentarios especificado es menor o igual a 0. 1.1.2 El sistema mostrará al usuario una gráfica de barras vacía. 1.2.1 La URL indicada no pertenece a ninguno de los diarios ofrecidos o no existe. 1.2.2 El sistema mostrará al usuario una gráfica de barras vacía.	
Postcondiciones	1. El usuario será capaz de visualizar las estadísticas que el sistema ha realizado sobre el número de comentarios especificado, en forma de gráfico de barras.

3.2.11 El usuario podrá visualizar una gráfica de sectores con la categorización de las noticias y los comentarios

Descripción	El usuario podrá visualizar una gráfica de sectores con la categorización realizada.
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Escenario principal	
1. El usuario selecciona el número de noticias/comentarios del periódico que desee o la noticia concreta a analizar. 2. El usuario pulsa el botón "Analizar". 3. El sistema categoriza los comentarios. 4. El sistema muestra al usuario una gráfica de sectores con los resultados calculados anteriormente.	
Escenario alternativo	
1.1.1 El número de comentarios especificado es menor o igual a 0. 1.1.2 El sistema mostrará al usuario una gráfica de sectores vacía. 1.2.1 La URL indicada no pertenece a ninguno de los diarios ofrecidos o no existe. 1.2.2 El sistema mostrará al usuario una gráfica de sectores vacía.	
Postcondiciones	1. El usuario será capaz de visualizar las estadísticas que el sistema ha realizado sobre el número de comentarios especificado, en forma de gráfico de sectores.

3.2.12 El usuario podrá analizar comentarios y/o noticias de todos los periódicos disponibles a la misma vez

Descripción	El usuario podrá analizar el número de comentarios y/o noticias de los diarios ofertados y analizarlos conjuntamente, de la manera que desee.
Prioridad	Alta
Actores	1. Usuario 2. Sistema
Escenario principal	
1. El usuario selecciona la opción de analizar noticias automáticamente. 2. El usuario selecciona la opción de recolectar información de El País, Diario Sur y/o ABC. 3. El usuario selecciona que desea analizar noticias o comentarios, para cada periódico seleccionado anteriormente. 4. El usuario pone el número de noticias o comentarios (mayor que 0) que desea analizar de cada periódico seleccionado en el punto 2. 5. El usuario pulsa el botón "Analizar". 6. El sistema comprueba que tiene ya suficientes noticias y comentarios para analizar. 7. El sistema categoriza los comentarios de las noticias.	
Escenario alternativo	
3.1.1 El número de comentarios y/o noticias de todos los comentarios especificados es menor o igual a 0. 3.1.2 El sistema mostrará al usuario gráficas vacías. 5.2.1 El sistema comprueba que no tiene suficientes comentarios y/o noticias para analizar. 5.2.2 El sistema obtiene más comentarios y/o noticias actuales de los periódicos seleccionados en el punto 2. 5.2.3 El sistema categoriza los comentarios de las noticias.	
Postcondiciones	1. El usuario será capaz de visualizar las estadísticas que el sistema ha realizado sobre el número de comentarios y/o noticias especificado de cada periódico.

3.3 Casos de prueba

Para definir también los casos de prueba, nos basaremos también en el estándar del NIST.

PR-FN-GC	Función "getComments" del sistema	
	Objetivo a evaluar	Comprobar que se reciben los comentarios que se solicitan y que se muestran correctamente en las gráficas, además de ser fáciles de entender.
	GC-CV-1	Validar que el número de comentarios coincide exactamente con los que se solicitaron en la aplicación.
	GC-CV-2	Validar que cada una de las gráficas muestra la cantidad de comentarios esperados, según el periódico que se analiza.
	GC-CV-3	Validar con usuarios que la función de obtener comentarios es sencilla de usar y que son fácilmente interpretables los resultados.
	Herramientas y métodos de evaluación	
	- Pruebas de integración - Entrevista con usuarios no expertos	

PR-FN-GN	Función "getNotices" del sistema	
	Objetivo a evaluar	Comprobar que se reciben los comentarios del número de noticias que se solicitan y que se muestran correctamente en las gráficas, además de ser fáciles de entender.
	GN-CV-1	Validar que el número de comentarios coincide exactamente con la suma de los comentarios de las noticias analizadas que se solicitaron en la aplicación.
	GN-CV-2	Validar que cada una de las gráficas muestra la cantidad de comentarios esperados, según el periódico que se analiza y las comentarios que poseen cada noticia.
	GN-CV-3	Validar con usuarios que la función de obtener noticias es sencilla de utilizar y que son fácilmente interpretables los resultados.
	Herramientas y métodos de evaluación	
	- Pruebas de integración - Entrevista con usuarios no expertos	

PR-FN-GCN	Función "getConcreteNotice" del sistema	
	Objetivo a evaluar	Comprobar que se reciben los comentarios de la noticia que se solicita y que se muestran correctamente en las gráficas, además de ser fáciles de entender.
	GCN-CV-1	Validar que el número de comentarios coincide exactamente con los que posee la noticia en cuestión.
	GCN-CV-2	Validar que cada una de las gráficas muestra la cantidad de comentarios esperados, según el periódico que se analiza y la noticia de la que se trata.
	GCN-CV-3	Validar con usuarios que la función de obtener una noticia concreta es sencilla de usar y que son fácilmente interpretables los resultados.
	Herramientas y métodos de evaluación	
	- Pruebas de integración - Entrevista con usuarios no expertos	

3.4 Arquitectura

La aplicación web hará uso del paradigma Cliente-Servidor como se muestra en la Figura 3.2. En el caso de que la aplicación se desplegara en un servidor externo, el navegador del usuario haría la función de cliente, y el servidor externo, la de servidor. El otro posible caso sería que el usuario ejecutase el servidor en su propia máquina, por lo que el servidor sería su propio ordenador, y el cliente, el navegador web que utilice.

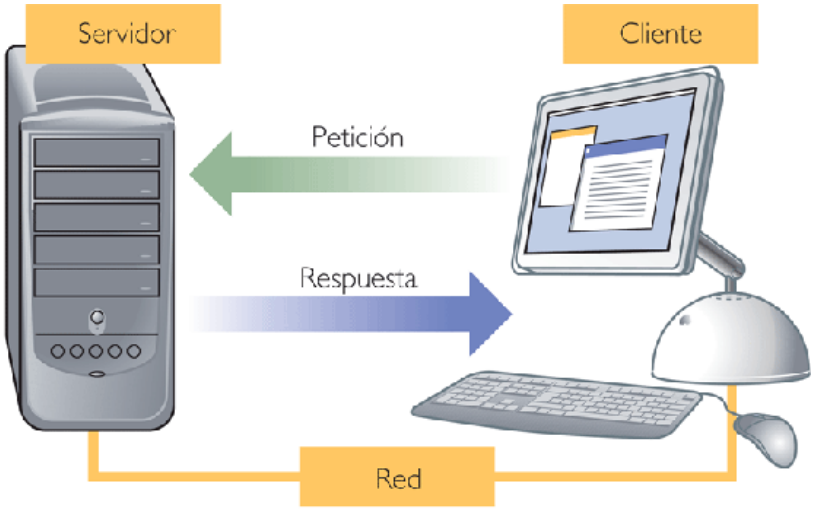

Figura 3.2 Arquitectura cliente servidor

El servidor (la aplicación PressOpinionAnalyzer del capítulo 5) será el encargado de realizar las peticiones y obtener las noticias y los comentarios para poder mandarle los resultados al cliente. PressOpinionAnalyzer hará uso de la librería ScrapAPI (especificada en el capítulo 4) para manejar las noticias y los comentarios, así como para obtenerlos de las páginas de prensa, y del categorizador (capítulo 6) para poder clasificar los comentarios que se hayan recogido previamente. De esta manera, el cliente realizará peticiones JSON al servidor especificando el número de comentarios o noticias de cada diario quiere analizar, y éste le responderá (también en formato JSON) con los resultados obtenidos al analizar lo que el cliente le indicó.

4

Librería ScrapAPI

4.1 Necesidad y uso

Para empezar a describir todo el proyecto, empezaremos por la librería java que nos permitirá realizar, de manera sencilla, labores como la recolección de las noticias y los comentarios.

Esta separación entre el proyecto final "PressOpinionAnalyzer" y esta librería se debe a que, de esta manera, podríamos reutilizar la librería para cualquier otro proyecto que necesitase obtener información de cualquier página de noticias. El modelo se realizó de manera que, pueda ser ampliado, añadiéndole más funcionalidades, e incluso, modificando y añadiendo algunas clases, obtener información de otras páginas, que no tengan por qué ser noticias, aunque fundamentalmente está pensada para obtener noticias y comentarios de cualquier página web.

Se hace uso de Maven para la resolución de dependencias de manera sencilla, además de luego poder exportar la librería. También se usa la librería JSoup para realizar las peticiones HTML a las páginas, y obtener sus respuestas, además de GSON para manejar los JSON (la librería de Google).

Con todo ello, podemos empezar a describir cada una de las partes de esta librería.

4.2 Clases

Empezaremos definiendo las clases que forman el modelo, una a una, indicando su función dentro del mismo. Sólo se incluirán las principales del modelo, y finalmente se mostrarán todas dentro del modelo, ya que muchas de ellas son clases que se implementan heredando de otras.

4.2.1 Clase Notice

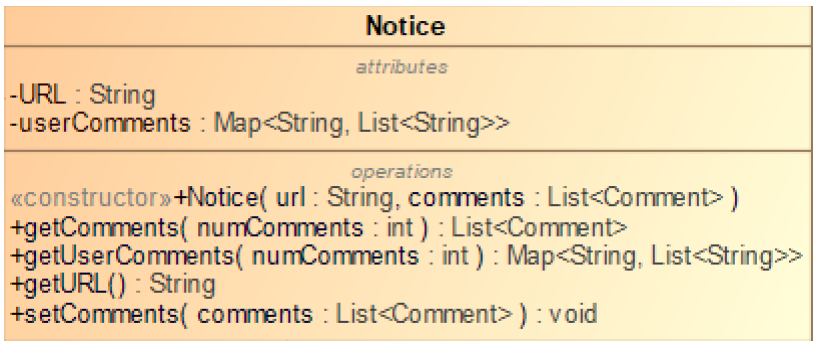

Figura 4.1 Clase Notice

Aquí podemos ver cómo modelamos para este caso, las noticias, como bien indica el nombre de la clase. Guardaremos la URL como nombre de la noticia, y los comentarios de los usuarios (userComments) para el caso en que necesitemos saber los comentarios que han hecho cada usuario. Cada noticia tiene asociada un grupo de comentarios.

4.2.2 Clase Comment

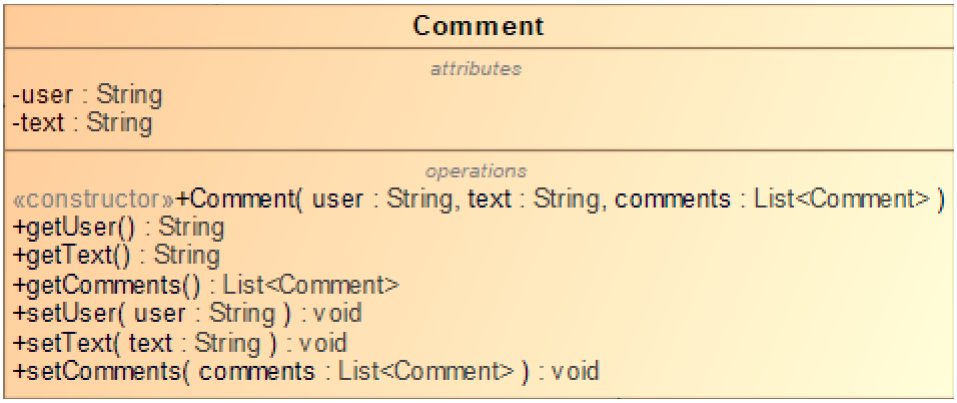

Figura 4.2 Clase Comment

Con esta clase modelaremos los comentarios; esta será la encargada además de almacenar el usuario y el mensaje que ha compartido en la noticia en cuestión.

4.2.3 Clase API

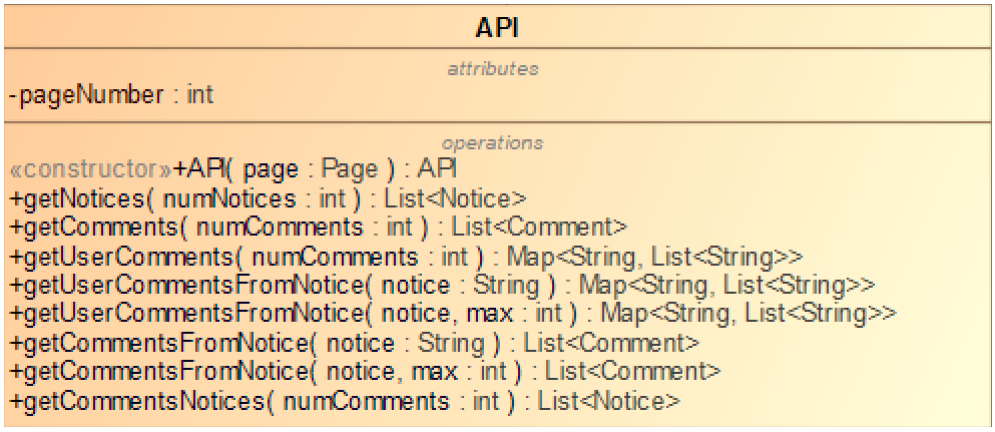

Figura 4.3 Clase API

Aquí podemos encontrar el plato fuerte del modelo. De hecho, será la clase que se utilizará cuando se use la librería, ya que posee todos los métodos necesarios para obtener la información de las noticias, de manera transparente y muy sencilla. Para inicializarla necesitaremos un objeto de la clase Page, que se trata de un enumerado que contiene los posibles periódicos online.
Con esta clase podremos obtener el número de noticias que queramos, con **getNotices**, el número de comentarios que queramos, con **getComments**, los comentarios de cada usuario, con **getUserComments**, los comentarios de los usuarios, de una noticia en concreto, con **getUserCommentsFromNotice**, obtener los comentarios de una noticia, con **getCommentsFromNotice** o recolectar un número de comentarios, pero devolviendo las

noticias de las cuales se han sacado, con **getCommentsNotices.**
Es importante saber, que al instanciar un objeto de la clase API y utilizarla para obtener alguna información, la clase recordará cual fue la última noticia analizada, por lo que, en cuanto se vuelva a utilizar haciéndosele una llamada, se recolectarán las noticias partiendo de la última que se analizó.

4.2.4 Clase APIRequest

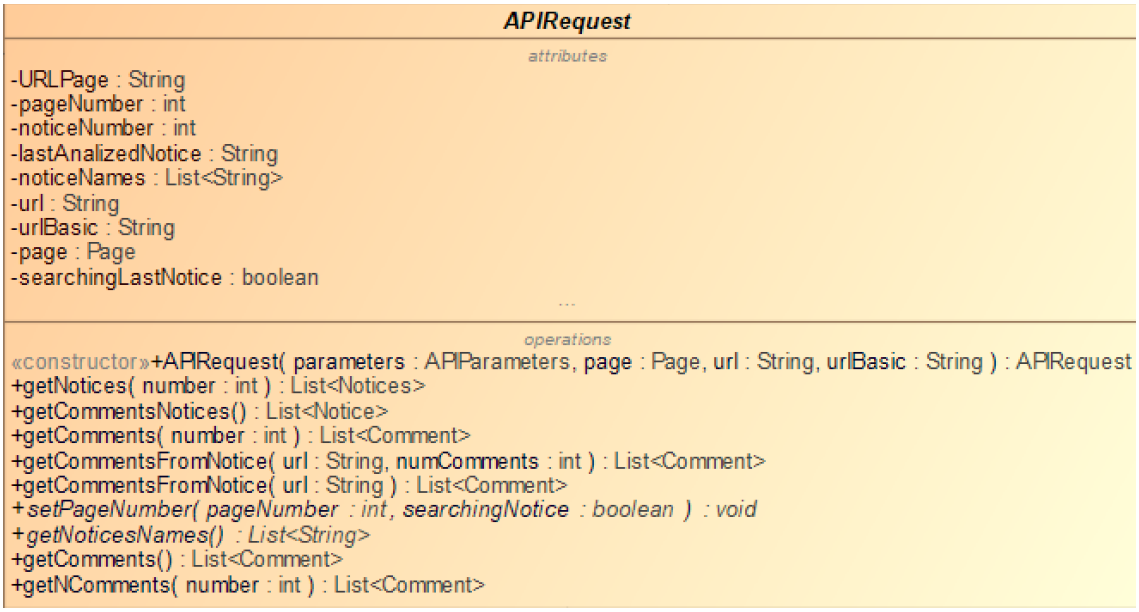

Figura 4.4 Clase APIRequest

Esta clase abstracta será la encargada de recolectar las noticias y los comentarios a bajo nivel. Se hace uso de la librería JSoup, para obtener la información de la página seleccionada. Será la encargada de guardar la última noticia analizada y la página por la que se ha quedado buscando. Además, guardará la URL base de la página, para reconocer si una noticia es del periódico que se está analizando o no.

Hará uso de la clase APIParameters para enviar las peticiones a las APIs donde se guardarán los comentarios de los periódicos. Analizará la respuesta de las APIs con la clase APIResponse. Por tanto, se puede ver que es la clase que se encargará de hacer el papel más importante a la hora de recolectar la información de las noticias. Para poder analizarlas, debemos implementar una clase que herede de ésta (además de otras clases que hereden, una de APIParameters y otra de APIResponse), implementando los métodos **setPageNumber** (que será la encargada de saber cómo cambiar de página dentro de la web de las noticias) y **getNoticesNames** (que se encargará de obtener todas las noticias de la página del periódico online en la que se

encuentre en ese momento). Todas las demás clases, en principio, no será necesaria su implementación, ya que se trata de una implementación de un algoritmo genérico. Sólo deberían implementarse en el caso de que la página tenga una estructura que no tenga nada que ver y sea necesario añadir alguna instrucción intermedia dentro del código, como en el caso de la implementación para el diario El País, que comentaremos más adelante, en el capítulo 4.4 Implementación.

4.2.5 Clase APIParameters

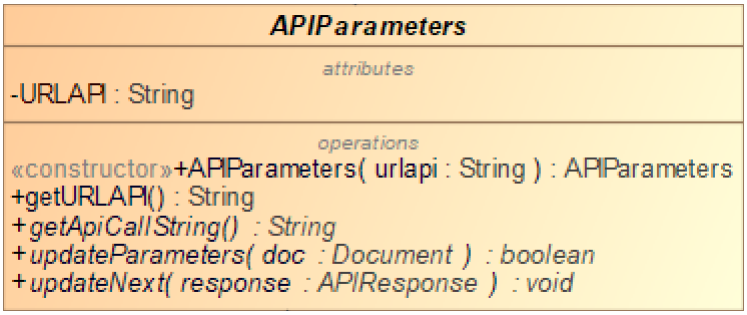

Figura 4.5 Clase APIParameters

Esta clase abstracta será necesaria para juntar los parámetros necesarios para poder hacer la petición a la API de los comentarios. Cada API necesita que se le realicen unas peticiones de una manera específica, y por tanto, para cada periódico online, se necesitará una implementación de esta clase, realizando los métodos abstractos **getApiCallString** (que retornará la URL completa, con los parámetros necesarios para realizar la petición), **updateParameters** (para actualizar los parámetros cada vez que se quiera hacer una petición sobre una noticia diferente) y **updateNext** (en el caso en el que la API de noticias devuelva los comentarios en ciertas cantidades, es decir, poco a poco, teniendo que hacer varias consultas para obtener todos los comentarios de una noticia). Con este último método, se le pasará un objeto de la clase APIResponse, del que hará uso para actualizar los parámetros necesarios para la siguiente petición.

4.2.6 Clase APIResponse

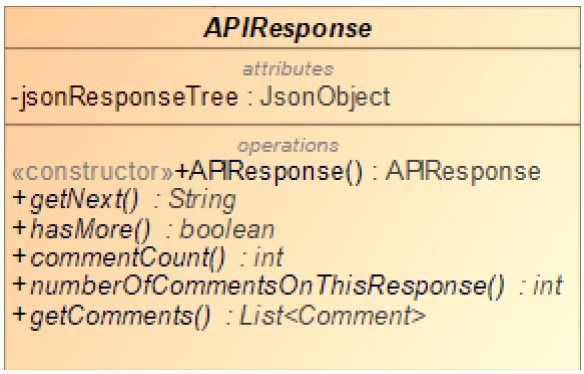

Figura 4.6 Clase APIResponse

Esta clase se encargará de ordenar la respuesta que la clase APIRequest haya hecho con ayuda de los parámetros. Deberemos implementar los métodos abstractos de esta clase, **getNext**, para obtener el siguiente tramo al que haya que hacerle la petición, **hasMore**, para saber si existen más comentarios en esa noticia, **commentCount**, para indicar el número de mensajes totales que tiene la noticia, **numberOfCommentsOnThisResponse**, con el motivo de conocer el número de comentarios que se han obtenido en esa respuesta y **getComments**, para obtener los comentarios que existen dentro de la respuesta.

4.2.7 Clases extras

Para este punto, me gustaría mencionar las clases que han sido necesarias para las tres fuentes de noticias que hemos seleccionado para este TFG: ABC, Diario Sur y El País.

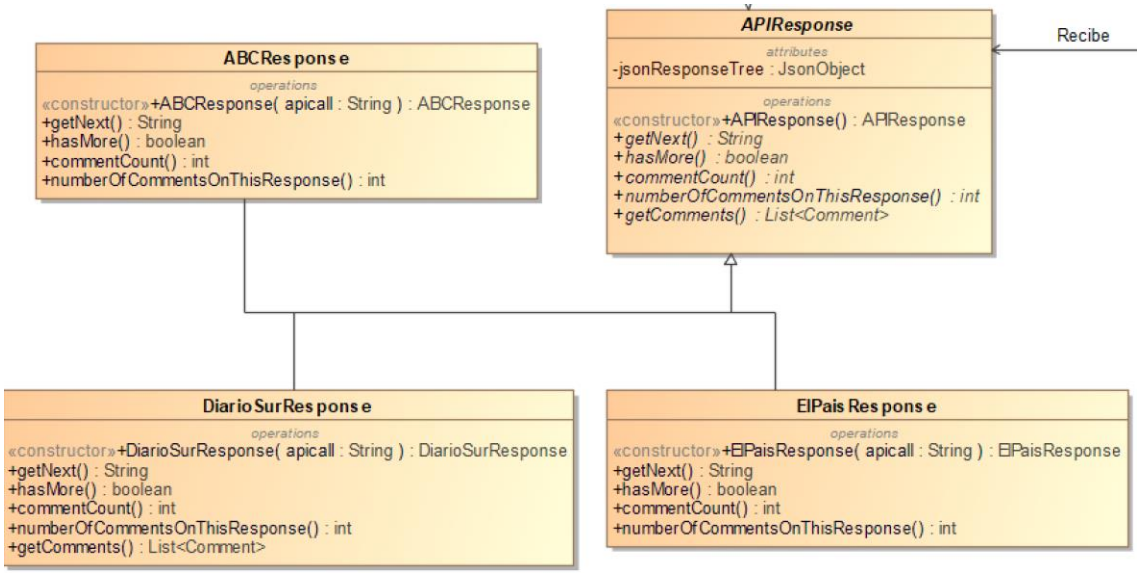

Figura 4.7 ABCResponse, DiarioSurResponse y ElPaisResponse

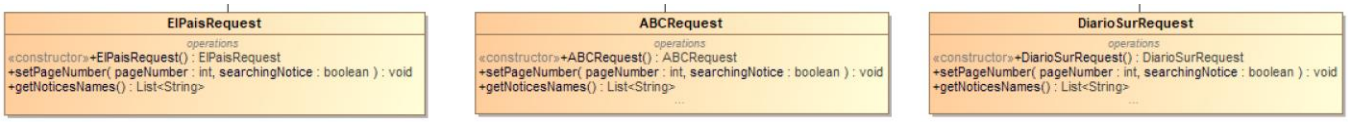

Figura 4.8 ABCRequest, ElPaisRequest y DiarioSurRequest

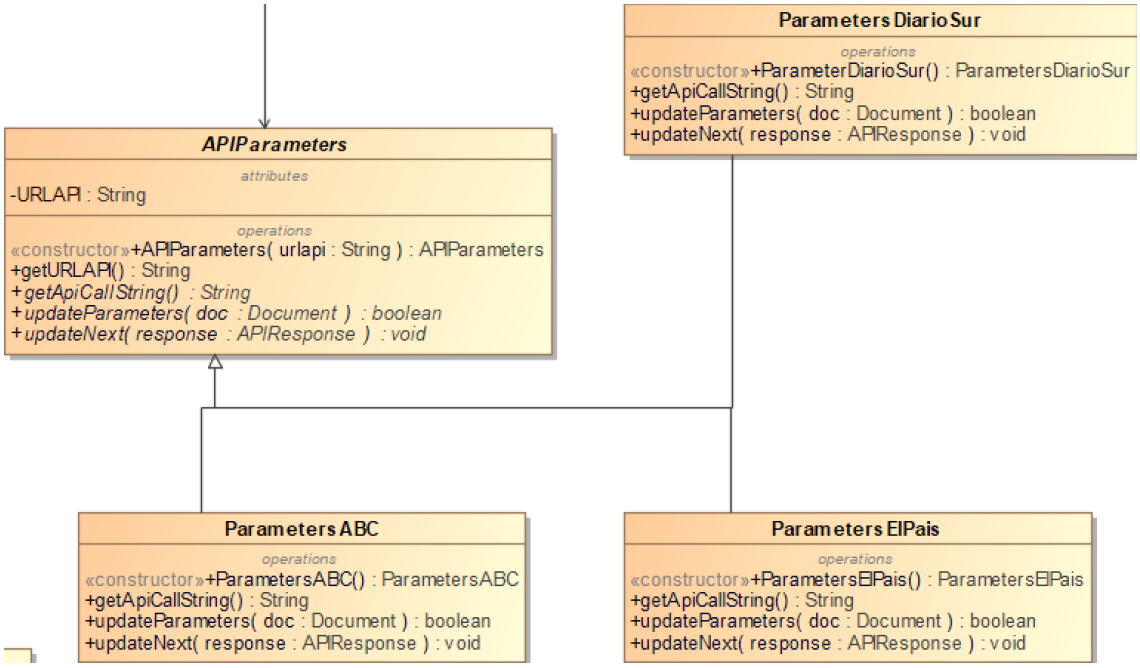

Figura 4.9 ParametersDiarioSur, ParametersElPais y ParametersABC

Como se ha descrito antes, podemos ver que para cada periódico, se ha realizado una clase que herede de APIParameters, una de APIResponse y otra de APIRequest.

Además, una clase que no se esperaba realizar fue ElPaisComment, una clase improvisada, debida a la forma en la que la API de El País mandaba los mensajes en las peticiones. Esta clase hereda de Comment, añadiéndole lo necesario para manejar estos comentarios. Esto lo detallaremos más en el capítulo 4.4 Implementación.

4.3 Modelo

Aquí podemos observar el modelo de clases de la librería, que nos servirá, para visualizar un poco la arquitectura, teniendo en cuenta el uso de cada clase gracias al capítulo anterior (4.2 Clases).

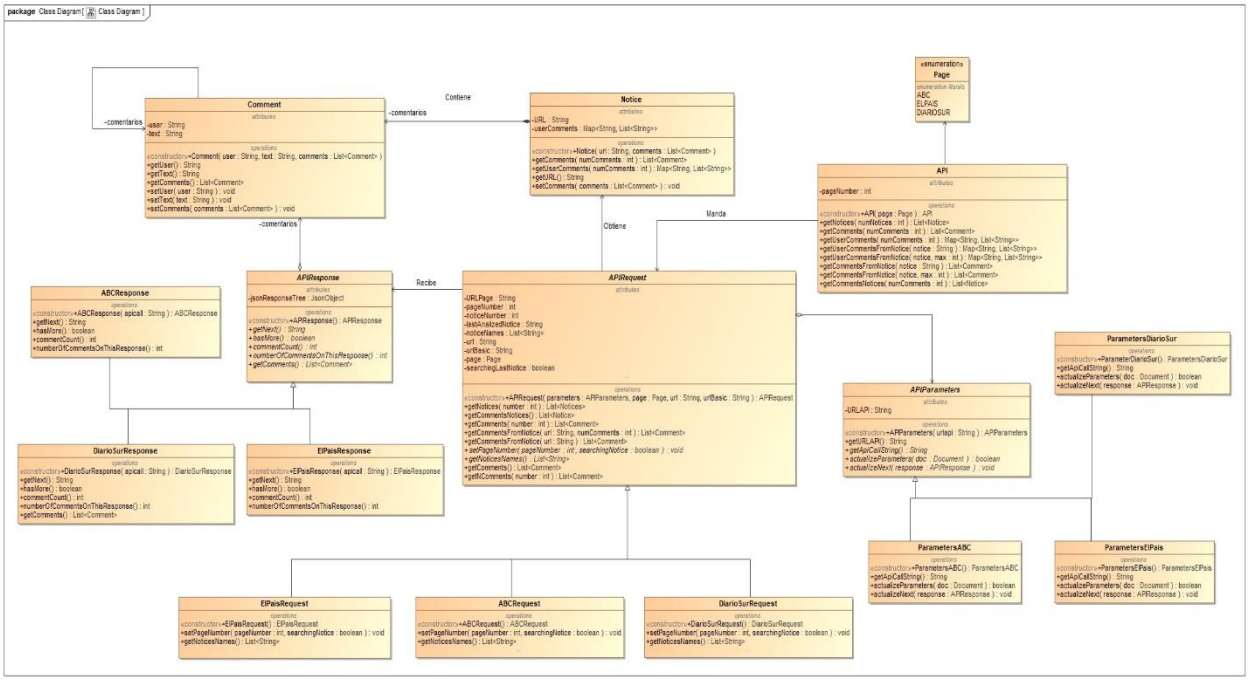

Figura 4.10 Modelo ScrapAPI

4.4 Implementación

Está claro que la implementación siempre es un tema espinoso, ya que es cuando nos encontraremos con la tecnología que hayamos elegido, con sus dependencias y con los errores que surgen durante el desarrollo. Aunque por lo general, no fue difícil implementar la librería, sí que es verdad que tiene un gran estudio detrás, ya que para poder crear cualquiera de los periódicos, se necesita investigar la página, reconocer cómo se puede obtener la página de noticias, cómo poder cambiar de página y analizar el código Javascript de la página para saber cómo se realizan las consultas y a qué URL concreta, además de analizar el comportamiento para saber cuál es el significado de cada uno de los parámetros

y cómo se usan (junto al misterio de dónde salen, ya que en ocasiones puede ser frustrante no encontrar el lugar del cual se sacan los parámetros).

En el caso de la API del País, se encontró un problema cuando se percató que los comentarios se recibían de manera muy diferente a lo esperado, y es que, en el JSON, siempre se recibían los comentarios con identificadores, y haciendo distinción entre hilos, problema que hacía que hasta en la propia página, la manera en la que se veían los comentarios sea muy liosa, además de poder ver que en ocasiones se repiten comentarios cuando la gente se responden los unos a los otros. Esto se arregló añadiendo alguna instrucción más al código de APIRequest para ordenar los mensajes, y creando la clase ElPaisComment, para guardar los identificadores de los mensajes y de las respuestas.

Seguidamente se puede visualizar en un diagrama de secuencia, la manera en que se recopilan las noticias, porque, aunque la librería ScrapAPI incluye muchos más métodos que los que se usan dentro de PressOpinionAnalyzer, getNotices es el que se ha utilizado por su simplicidad.

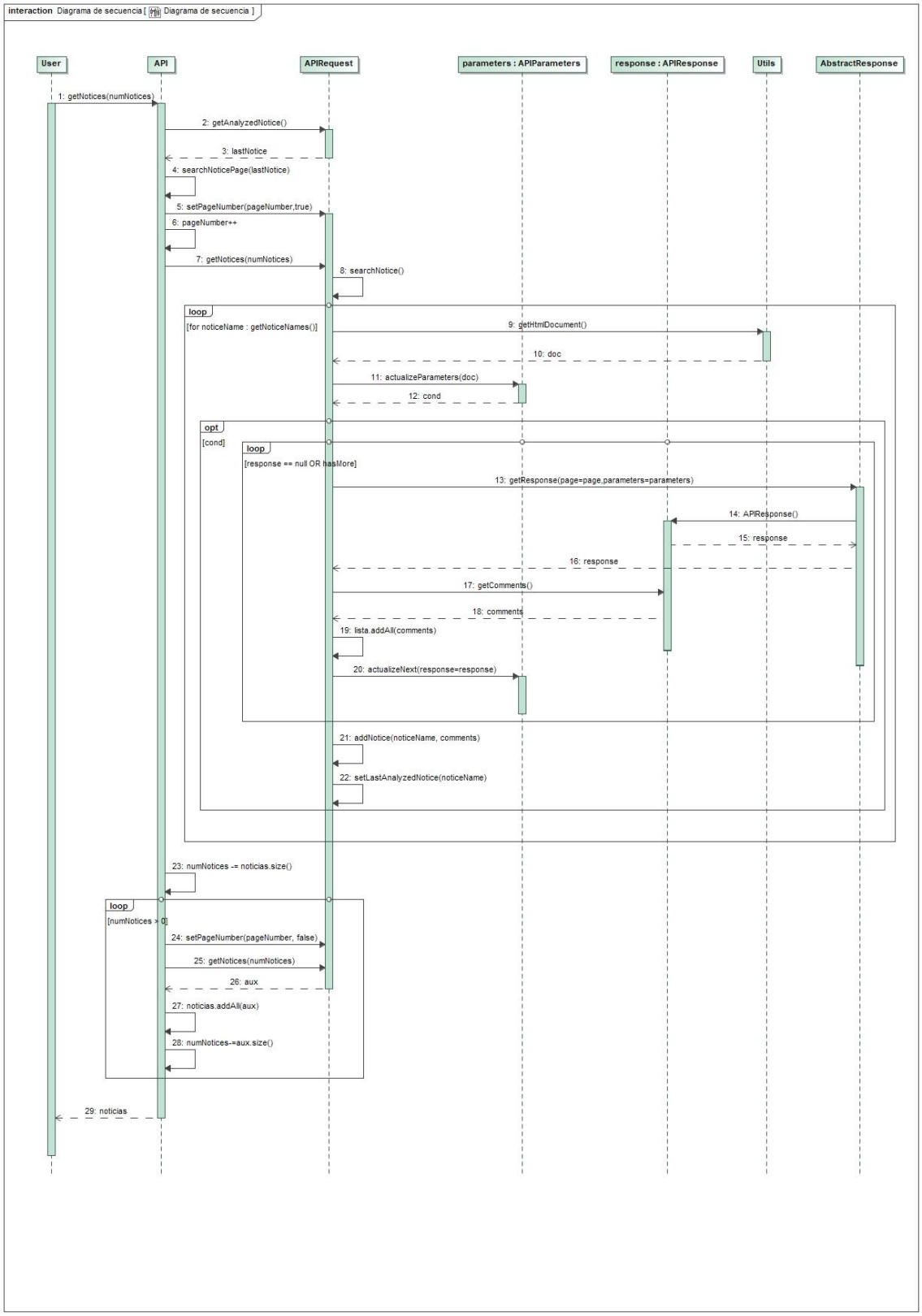

Figura 4.11 Diagrama de secuencia de getNotices

Lo último que quedaría por ver sería el algoritmo obtener las noticias y los comentarios en un diagrama de actividad.

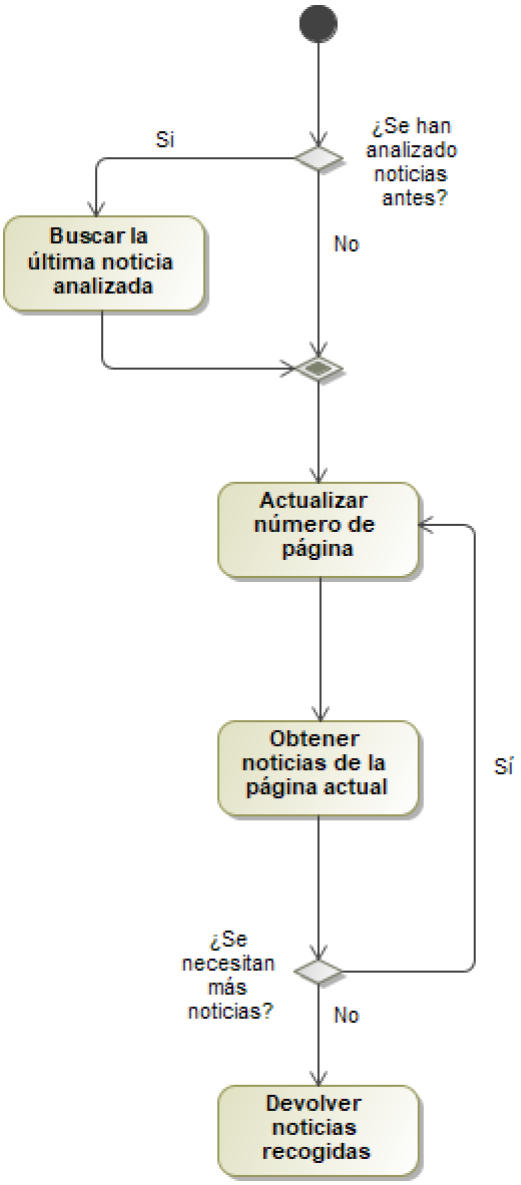

Figura 4.12 Diagrama de actividad de getNotices

Dentro del diagrama anterior se puede visualizar una acción llamada "Obtener noticias de la página actual". Seguidamente podemos visualizar el diagrama de actividad de dicho algoritmo.

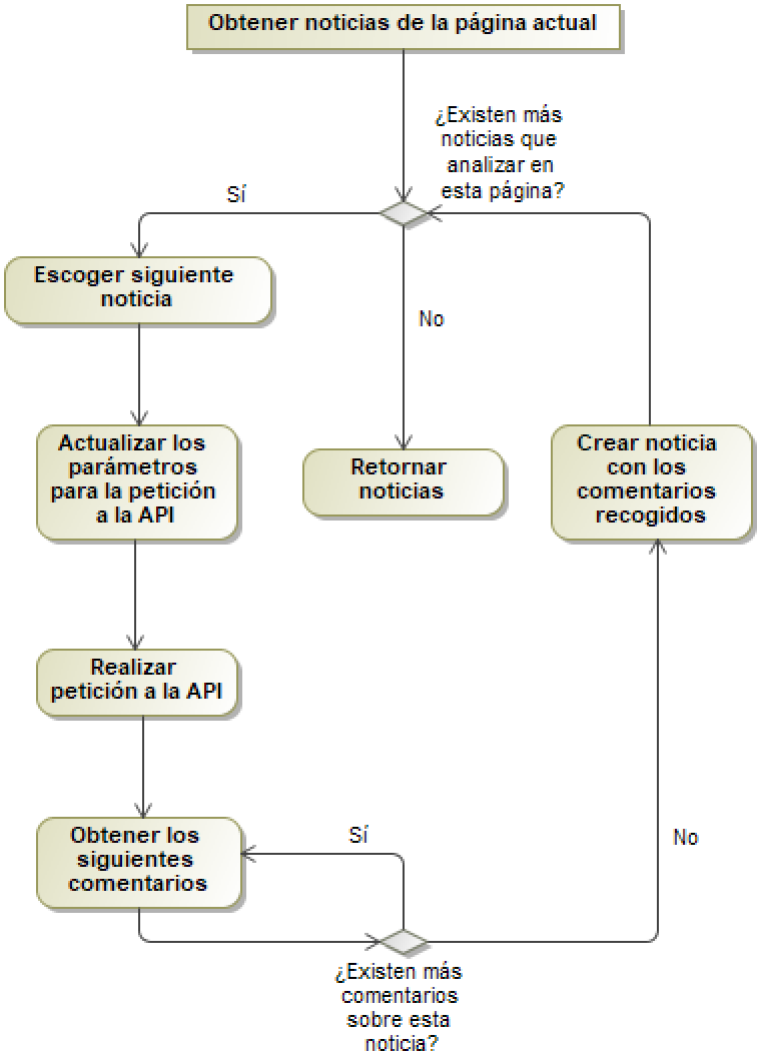

Figura 4.13 Diagrama de actividad de "Obtener noticias de la página actual"

5

Proyecto Press Opinion Analyzer

5.1 Necesidad y uso

En esta capítulo veremos la parte principal del proyecto. Nos encontramos con un servidor hecho en Java, con el framework Spring Boot.

En un principio, debíamos aprender esta tecnología, para luego utilizar la librería ScrapAPI y poder analizar los elementos con el categorizador. La filosofía de este framework se basa en los microservicios, y por tanto, en el desacoplamiento del código. Tendremos separada la lógica de negocio (modelo), de la vista (plantillas) y el controlador multicapa. En este último, existe una separación, entre el controlador, que hará uso de los servicios. Estos últimos, harán usarán los componentes, y finalmente, se utilizarán los repositorios y los manejadores de los repositorios, que serán los encargados de realizar las sentencias SQL al repositorio en concreto.

En este proyecto, no haremos uso de ninguna base de datos, ya que no es necesario guardar nada, por lo que no tendremos ningún paquete de manejadores ni repositorios. Utilizaremos Thymeleaf para el renderizado de plantillas en la vista.

A continuación veremos las clases del modelo y el modelo.

5.2 Clases

A continuación definiremos las clases principales del proyecto, además de definir el modelo y su estructuración dentro del mismo.

5.2.1 Clase ModelBean

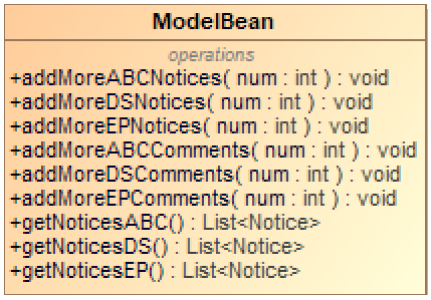

Figura 5.1 Clase ModelBean

Este será el componente encargado de almacenar las noticias y los comentarios recopilados, de obtener la información de los periódicos online y de añadir la información.

Cada uno de los métodos que podemos visualizar dentro de la clase consisten en lo que sus propios nombres indican, para añadir el número de comentarios/noticias de cada periódico. Además, también podemos visualizar tres métodos adicionales, necesarios para obtener la referencia a las listas de noticias que tenemos guardadas.

5.2.2 Clase Controller

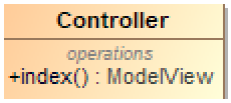

Figura 5.2 Clase Controller

Esta clase se encarga de renderizar las plantillas, en este caso, sólo tenemos una plantilla, ya que la página es dinámica, cambiando su forma con Javascript, además de hacer las peticiones con el mismo, es decir, no necesitaremos otra plantilla además de la principal.

5.2.3 Clase APIController

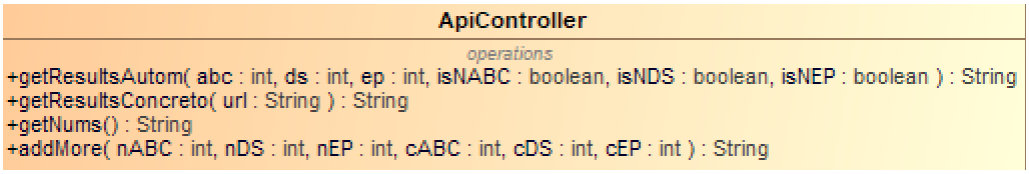

Figura 5.3 Clase APIController

Esta clase es un controlador secundario, utilizado para realizar las llamadas asíncronas desde el cliente, mediante Javascript. Con esta API, podremos obtener los resultados de analizar automáticamente las noticias, analizar una noticia en concreto, obtener el número de noticias/comentarios que hay en el servidor y poder añadir más noticias/comentarios (todo en formato JSON).

Esto permite una comunicación más fluida entre el cliente y el servidor, lo que hace que mejore la experiencia del usuario y no se haga monótona.

5.2.4 Clase Categorizer

Figura 5.4 Interface ICategorizer

Esta será la interfaz servicio de la clase encargada de categorizar los comentarios de las noticias. Para ello, se hará uso de la librería descrita en el capítulo 6.

5.2.5 Clase CommentsGetter

Figura 5.5 Interfaz ICommentsGetter

Esta interfaz servicio nos permitirá recolectar los comentarios necesarios del componente ModelBean.

5.3 Modelo

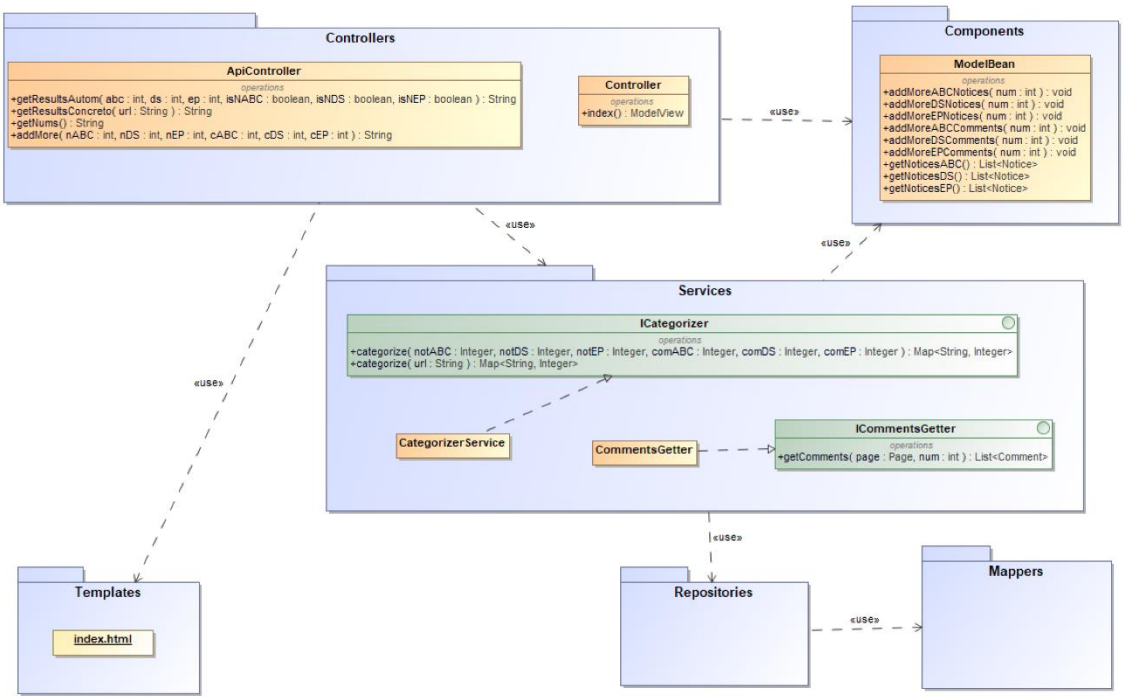

Figura 5.6 Modelo PressOpinionAnalyzer

Aquí podemos visualizar el diagrama completo de la aplicación Spring Boot especificado anteriormente. Podemos encuadrar, dentro de los controladores, la clase APIController y la clase Controller, que harán uso de index.html para renderizar la plantilla con Thymeleaf. A su vez, APIController necesitará hacer uso de ModelBean para obtener las noticias que se han recolectado, los servicios de categorización usarán a su vez las noticias de ModelBean para categorizar sus comentarios, y CommentsGetter obtendrá los comentarios de las noticias guardadas también en ModelBean para que el controlador los recoja de una manera más sencilla.

En el caso de que guardásemos las noticias y los comentarios, tendríamos que crear un nuevo servicio que se encargará de comunicarse con el repositorio que creásemos, y las sentencias SQL las crearíamos dentro de los manejadores.

5.4 Implementación

La implementación de este framework no ha sido complicada ya que, para este punto, tendríamos realizado ya tanto el categorizador como la librería ScrapAPI, de los cuales hace uso este proyecto.

Aquí podemos visualizar el código de los servicios, que es lo más importante dentro de la aplicación, ya que hacen uso del categorizador y la librería de extracción de las noticias, siendo el núcleo de la aplicación.

```
public class Categorizer {

    private DocumentCategorizer categorizer;

    public Categorizer(){
        InputStream is = null;
        try {
            is = getClass().getResourceAsStream( name: "/training/model.bin");
            DoccatModel model = new DoccatModel(is);
            categorizer = new DocumentCategorizerME(model);
        } catch (FileNotFoundException ex) {
            Logger.getLogger(Categorizer.class.getName()).log(Level.SEVERE,  msg: null, ex);
        } catch (IOException ex) {
            Logger.getLogger(Categorizer.class.getName()).log(Level.SEVERE,  msg: null, ex);
        }
    }

    public DocumentCategorizer getCategorizer() { return categorizer; }

}
```

Figura 5.7 Código Categorizer

```java
@Service
public class CategorizerService implements ICategorizer{

    @Override
    public Map<String, Integer> categorize(int numABC, int numDS, int numEP,
                                           int notABC, int notDS, int notEP) {
        ModelBean modelBean = ModelBean.getInstance();
        DocumentCategorizer doccat = new Categorizer().getCategorizer();
        Map<String, Integer> resultados = new TreeMap<>();
        ICommentsGetter commentsGetter = new CommentsGetter();
        List<Notice> noticias = new ArrayList<>();
        resultados.put("PP", 0);resultados.put("PSOE", 0);resultados.put("Vox", 0);
        resultados.put("Podemos", 0);resultados.put("Ciudadanos", 0);resultados.put("NSNC", 0);
        if(notABC == 0)
            noticias.addAll(modelBean.getNoticiasABC().subList(0, numABC));
        else
            noticias.add(new Notice( URL: "ABC", commentsGetter.getComments(Page.ABC, numABC)));
        if(notDS == 0)
            noticias.addAll(modelBean.getNoticiasDS().subList(0, numDS));
        else
            noticias.add(new Notice( URL: "DS", commentsGetter.getComments(Page.DIARIOSUR, numDS)));
        if(notEP == 0)
            noticias.addAll(modelBean.getNoticiasEP().subList(0, numEP));
        else
            noticias.add(new Notice( URL: "EP", commentsGetter.getComments(Page.ELPAIS, numEP)));
        String res;
        for(Notice noticia : noticias) {
            for(Comment comentario : noticia.getComments()) {
                res = doccat.getBestCategory(doccat.categorize(comentario.toString().split( regex: " ")));
                resultados.put(res, resultados.get(res) + 1);
            }
        }
        return resultados;
    }
```

Figura 5.8 Código CategorizerService

```java
@Override
public Map<String, Integer> categorize(String url) {
    DocumentCategorizer doccat = new Categorizer().getCategorizer();
    Map<String, Integer> resultados = new TreeMap<>();
    resultados.put("PP", 0);resultados.put("PSOE", 0);resultados.put("Vox", 0);
    resultados.put("Podemos", 0);resultados.put("Ciudadanos", 0);resultados.put("NSNC", 0);
    String res;
    API api = null;
    if(url.contains("abc.es")) {
        api = new API(Page.ABC);
    }else if(url.contains("diariosur.es")) {
        api = new API(Page.DIARIOSUR);
    }else if(url.contains("elpais.com")) {
        api = new API(Page.ELPAIS);
    }
    if(api != null) {
        for(Comment comentario : Utils.getCommentsOutFromComments(api.getCommentsFromNotice(url))) {
            res = doccat.getBestCategory(doccat.categorize(comentario.toString().split( regex: " ")));
            resultados.put(res, resultados.get(res) + 1);
        }
    }
    return resultados;
}
```

Figura 5.9 Método Categorize

La Figura 5.10 muestra el controlador de la API, con los métodos que se describieron en el capítulo 5.2.3 Clase APIController.

```java
@RequestMapping("/api")
public class ApiController {

    @GetMapping(path = "/getResultsAutom")
    @ResponseBody
    public String getResultsAutom(@RequestParam("abc") int abc, @RequestParam(value="getNoticesABC", defaultValue="0") int getNoticesABC,
                                  @RequestParam("ds") int ds, @RequestParam(value="getNoticesDS", defaultValue="0") int getNoticesDS,
                                  @RequestParam("ep") int ep, @RequestParam(value="getNoticesEP", defaultValue="0") int getNoticesEP) {
        ICategorizer categorizer = new CategorizerService();
        Map<String, Integer> resultados = categorizer.categorize(abc, ds, ep, getNoticesABC, getNoticesDS, getNoticesEP);
        return createJson(resultados);
    }

    @GetMapping(path = "/getResultsConcreto")
    @ResponseBody
    public String getResultsConcreto(@RequestParam("url") String url) {
        ICategorizer categorizer = new CategorizerService();
        Map<String, Integer> resultados = categorizer.categorize(url);
        return createJson(resultados);
    }

    private String createJson(Map<String, Integer> resultados){
        JsonObject json = new JsonObject();
        json.addProperty( property: "PP", resultados.get("PP"));
        json.addProperty( property: "PSOE", resultados.get("PSOE"));
        json.addProperty( property: "Ciudadanos", resultados.get("Ciudadanos"));
        json.addProperty( property: "Vox", resultados.get("Vox"));
        json.addProperty( property: "Podemos", resultados.get("Podemos"));
        json.addProperty( property: "NSNC", resultados.get("NSNC"));
        return json.toString();
    }

    @GetMapping(path = "/getNums")
    @ResponseBody
    public String getNums() {
        ModelBean modelBean = ModelBean.getInstance();
        JsonObject json = new JsonObject();
        json.addProperty( property: "Nabc", modelBean.getNoticiasABC().size());
        json.addProperty( property: "Nds", modelBean.getNoticiasDS().size());
        json.addProperty( property: "Nep", modelBean.getNoticiasEP().size());
        json.addProperty( property: "Cabc", Utils.countAllCommentsFromNotices(modelBean.getNoticiasABC()));
        json.addProperty( property: "Cds", Utils.countAllCommentsFromNotices(modelBean.getNoticiasDS()));
        json.addProperty( property: "Cep", Utils.countAllCommentsFromNotices(modelBean.getNoticiasEP()));
        return json.toString();
    }
```

Figura 5.10 Clase APIController

```java
@GetMapping(path = "/addMore")
@ResponseBody
public String addMore(@RequestParam(value="Nabc", defaultValue="0") int Nabc, @RequestParam(value="Cabc", defaultValue="0") int Cabc,
        @RequestParam(value="Nds", defaultValue="0") int Nds, @RequestParam(value="Cds", defaultValue="0") int Cds,
        @RequestParam(value="Nep", defaultValue="0") int Nep, @RequestParam(value="Cep", defaultValue="0") int Cep) {
    ModelBean modelBean = ModelBean.getInstance();
    modelBean.addMoreABCNotices(Nabc);
    modelBean.addMoreABCComments(Cabc);
    modelBean.addMoreDiarioSurNotices(Nds);
    modelBean.addMoreDiarioSurComments(Cds);
    modelBean.addMoreElPaisNotices(Nep);
    modelBean.addMoreElPaisComments(Cep);
    return "ok";
}
```

Figura 5.11 Método addMore

Como se puede apreciar, gracias a la separación entre la librería de OpenNLP para la categorización, y la librería ScrapAPI para obtener las noticias y los comentarios, tenemos un código limpio, elegante y fácil de interpretar.

Por último, también se puede visualizar un diagrama de actividad con el algoritmo que sigue el cliente para obtener los resultados del análisis del servidor, cuyo código se ha explicado con anterioridad en este mismo capítulo.

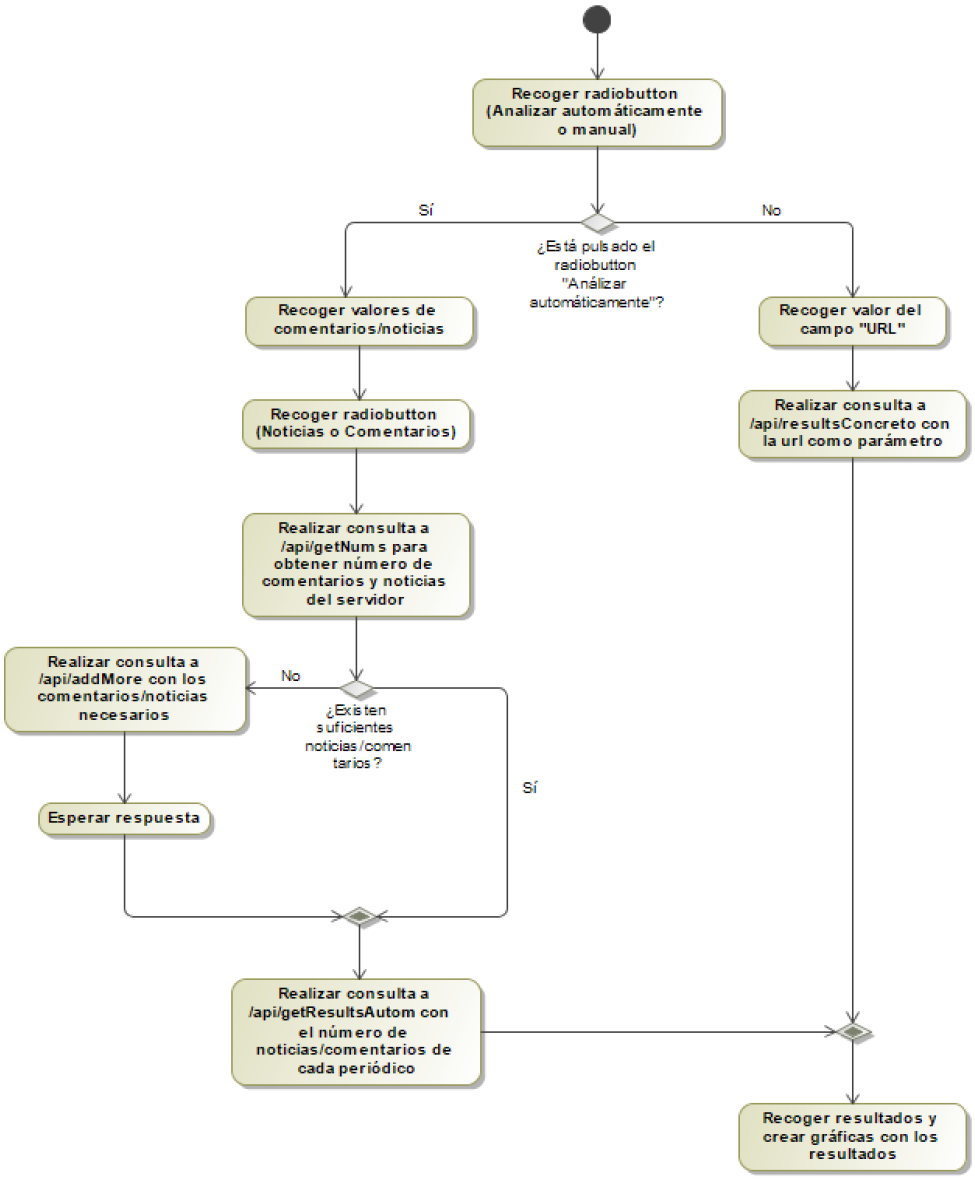

Figura 5.12 Diagrama de actividad del cliente

6

Categorizador de opiniones

6.1 Introducción

Para poder realizar la categorización de los comentarios, se ha hecho uso de una librería de Apache, llamada Apache OpenNLP.
Para poder realizar la categorización, primero debemos construir el modelo, y para el cual es necesario un fichero de entrenamiento, como podemos ver a continuación.

6.2 Entrenamiento

Para realizar el entrenamiento, lo primero que se tuvo que realizar es un script que, con la librería ScrapAPI, obtuviera una base de comentarios para luego, categorizarlos uno a uno manualmente.
Seguidamente podemos visualizar el código del script para conseguir los comentarios, en el formato necesario para que luego se puedan añadir a la librería de OpenNLP.

```java
public static void main(String[] args){
    PrintWriter pw = null;
    try{
        API api = new API(Page.ABC);
        List<Comment> comentarios = api.getComments(1000);
        for(Comment comentario: comentarios)
            System.out.println(comentario.toString());
        comentarios = Utils.getCommentsOutFromComments(comentarios);
        comentarios.sort(new Comparator<Comment>(){
            @Override
            public int compare(Comment o1, Comment o2) {
                return o1.getText().compareTo(o2.getText());
            }
        });
        System.out.println("Total de comentarios: " + comentarios.size());
        pw = new PrintWriter(new FileWriter(new File("src/test/java/comentarios/comentarios.txt")));
        for (Comment comment: comentarios){
            pw.println("Partido " + comment.getText());
        }
    }catch(Exception e){
        System.out.println("Error: " + e.getMessage());
    }finally{
        if(pw != null)
            pw.close();
    }
}
```

Figura 6.1 Código obtener comentarios entrenamiento

Así, como podemos ver en el código, se generaría un archivo con la estructura: "Partido $Comentario" donde $Comentario es uno de los comentarios que se han recopilado.

Después de generar el archivo, lo siguiente sería informarse acerca de la política actual a la vez que se van analizando los comentarios, y estimando a cuál de los 5 principales partidos puede pertenecer.

Este archivo contiene, aproximadamente, unos 1000 comentarios de entrenamiento, que han sido categorizados a mano uno a uno. Dentro de ellos, cabe destacar que esta categorización ha sido llevada a cabo de acuerdo al criterio del proyectista, y que por tanto, podría ser que alguna categorización esté mal hecha.

Aparte de estos 1000 comentarios, fueron categorizados unos 50 más, para hacer una prueba conforme a cuántos comentarios puede acertar. Con ello, se obtuvo un 62% de acierto, algo que no está nada mal, teniendo en cuenta que el análisis de los comentarios no lo ha realizado ningún experto en política.

Después de obtener el fichero de entrenamiento, sólo debemos hacer un algoritmo que sea capaz de crear el modelo a partir de la librería java que nos proporciona OpenNLP. Lo podemos ver seguidamente:

```java
public static void main(String[] args) throws FileNotFoundException{

    DoccatModel model = null;
    InputStreamFactory dataIn = null;

    try{
        dataIn = new MarkableFileInputStreamFactory(new File( pathname: "src/main/resources/training/entrenamiento.txt"));
        ObjectStream<String> lineStream
                = new PlainTextByLineStream(dataIn, StandardCharsets.UTF_8);
        ObjectStream<DocumentSample> sampleStream = new DocumentSampleStream(lineStream);

        TrainingParameters params = new TrainingParameters();
        params.put(TrainingParameters.ITERATIONS_PARAM, 950 + "");
        params.put(TrainingParameters.CUTOFF_PARAM, 0 + "");
        //params.put(AbstractTrainer.ALGORITHM_PARAM, NaiveBayesTrainer.NAIVE_BAYES_VALUE);

        model = DocumentCategorizerME.train( languageCode: "es", sampleStream, params, new DoccatFactory());

        BufferedOutputStream modelOut = new BufferedOutputStream(
                new FileOutputStream( name: "src/main/java/com/pressOpinionAnalysis/categorizer/resources/model.bin"));
        model.serialize(modelOut);

        DocumentCategorizer doccat = new DocumentCategorizerME(model);

        for(int i = 0; i < doccat.getNumberOfCategories(); i++)
            System.out.println(doccat.getCategory(i));

    } catch(IOException ex){
        Logger.getLogger(Trainer.class.getName()).log(Level.SEVERE, msg: null, ex);
    }

}
```

Figura 6.2 Código crear modelo

Como podemos ver, con este script obtenemos un archivo, "model.bin", que se trata del modelo que necesitamos para categorizar los comentarios.

6.3 Pruebas y validación

Para comprobar que funciona, fueron realizadas múltiples pruebas, como la comentada anteriormente para reconocer el número de aciertos del modelo. La más curiosa podemos encontrarla al utilizar la aplicación web y analizar comentarios. Podemos comparar que, si analizamos los comentarios de un periódico o de otro, siempre obtenemos algo principal, y es que, en el ABC hay mucha presencia de comentarios de derechas, mientras que en El País, obtenemos una cuantía superior de mensajes de izquierdas superior, como podemos ver en las siguientes imágenes.

Figura 6.3 Prueba ABC

Figura 6.4 Prueba El Pais

Cabe destacar que, para la tasa de acierto que tiene el modelo, se asemeja bastante a la realidad, aunque, como se puede ver, los comentarios de derechas suelen ser más comunes, o esa percepción también tuve al analizarlos uno a uno a la hora de crear el fichero de entrenamiento.

7

Problemas y soluciones

7.1 ScrapAPI

Aquí se describirán los principales problemas que se tuvieron en la realización de esta librería. También se hará uso de la librería GSON para poder manejar y crear los JSON y Maven para la resolución de dependencias.

7.1.1 Obtener los comentarios

El principal problema que se encontró al realizar la extracción de comentarios fue debido al desconocimiento de la tecnología adecuada. Inicialmente, se intentó utilizar la librería de JSoup para obtener los comentarios directamente de las páginas, pensando que la librería sería capaz de recogerlos, pero no era posible, ya que dichos comentarios no se enviaban junto a la página, sino que se obtienen por parte del cliente, es decir, es el cliente quien realiza las consultas de los comentarios en el momento en el que se necesitan, a través de Javascript. JSoup sólo realiza las consultas HTML, obteniendo el código que se envía desde el servidor, estático, ya que no se ejecuta ninguna de las funciones Javascript que realiza el navegador para darle dinamismo a la página.

Para obtener los comentarios se necesitaba realizar las consultas a una API, y para ello, investigar el funcionamiento a través de la consola y del movimiento de paquetes en la red, todo con ayuda del navegador. Los parámetros que cambiaban respecto a las noticias podíamos encontrarlos ocultos dentro de la página estática (alguno

dentro del código Javascript que se enviaba para ejecutarlo en el cliente), y otros, dentro de alguna etiqueta HTML.

7.1.2 Noticias de El País

En general, la recopilación de la información de El País ha sido complicada. En este caso, las noticias han tenido su pequeña complejidad, ya que al intentar establecer la conexión y hacer las peticiones con Jsoup, no recibía lo que esperaba, sino un error indicando que debo aceptar las cookies para acceder al servicio de buscar las noticias. Se intentó por todos los medios posibles, aceptar las cookies con JSoup, pero no hubo manera alguna de hacerlo, ni siquiera guardando las cookies y mandándolas continuamente con la información necesaria en las peticiones a la página. Finalmente, como no se pudo encontrar ninguna solución, se optó por buscar por otras zonas de la página. Finalmente, se encontró una zona de la página donde no hacía falta aceptar las cookies, y por tanto, fue la que usó, para poder obtener las noticias, ya que de otra manera, era imposible.

Desde la URL encontrada, se realizan las búsquedas de las noticias, pudiendo encontrarse de la misma manera que en las páginas del Diario Sur y ABC.

7.1.3 Comentarios de El País

Con el caso de El País, al ser el último noticiero que se abordó, se pensó que el tratamiento sería similar al del resto, pero al empezar a investigar, averiguamos que los comentarios no se guardaban ni se ordenaban como en las otras dos APIs, y es que, cada JSON recibido de la API, no guardaba los comentarios de respuesta de la gente, y los reconocía por "identificadores" e "identificadores respuesta". Cabe destacar que fue bastante complejo ordenarlos, e incluso parece ni siquiera los desarrolladores de El País lo deben conocer, ya que, al visualizar los comentarios desde la página, se pueden ver algunos repetidos.

Con la clase ElPaisComment y añadiendo alguna funcionalidad extra a la clase ElPaisRequest, ordenamos los comentarios y creamos la lógica que podemos ver en el modelo del capítulo 4.3, y de esta forma, mantener la estructura que tenemos con los otros dos periódicos, y que resulta ser la más sencilla e intuitiva.

7.1.4 Rapidez

Al terminar de realizar las implementaciones, se planteó la posibilidad de aligerarse un poco la recolección de las noticias/comentarios. Para ello, se decidó probar una nueva implementación de un algoritmo que lanzase hebras, y que cada una, obtuviese unos comentarios/noticias. El problema encontrado es que, esto no se pudo llegar a hacer, ya que las APIs no permiten llamadas concurrentes desde el mismo origen, es decir, tiene que pasar un periodo de algo menos de un segundo entre llamada y llamada. Esto se ha inferido tras ejecutar el algoritmo, y observar que, la diferencia del tiempo de recolección entre el algoritmo con hebras y el secuencial era de mucho menos de un segundo.

Todo esto se puede deber a un mecanismo para evitar que se sature el servidor haciéndole múltiples llamadas. Además, evitan la posibilidad de que el servidor recibiera ataques de denegación de servicios. Se podría intentar de otra manera, y sería con una red de ordenadores distribuidos, haciendo las llamadas (cada hebra), y luego, enviando las respuestas al servidor principal. Con esto se podrían mejorar bastante los tiempos, pero carecemos de estos medios.

7.2 PressOpinionAnalyzer

Sin lugar a dudas, los problemas más complejos de resolver han sido aquellos relacionados con la creación del ejecutable java (archivo.jar).

7.2.1 Construcción del Jar : model.bin

El principal problema encontrado a la hora de exportar el proyecto fue el directorio del modelo del categorizador que estaba estructurado de la siguiente manera: "/src/main/resources/categorizer/model.bin". Funcionaba perfectamente al utilizarlo sin haberlo compilado al jar. El problema surgió en el momento en que, al compilarlo, se ejecutaba sin errores aparentes, pero si se utilizaba alguna función que llamase al categorizador, lanzaba una excepción por no encontrar el archivo (FileNotFoundException).
Después de múltiples búsquedas por Internet, encontramos solución a este problema: y es que, al poner la ruta del archivo tal cual, al compilarse, esta se modifica, y no encuentra el archivo donde se había estructurado durante la fase de desarrollo. Para arreglar este problema, sólo había que llamar a la función getClass().getResourceAsStream(path), donde path es la ubicación del archivo. En este caso, como desde un principio estaba ubicado dentro de la carpeta resources, no hubo que

cambiar el archivo de carpeta. Esta función lo que hace es realizar una búsqueda dentro de las carpetas asociadas a la ubicación de las clases (classpath), y la carpeta resources, es una de las que está en dicho classpath.

7.2.2 Construcción del Jar : ScrapAPI

Otro de los problemas al construir el Jar fue al vincular al pom.xml (Maven) la librería local ScrapAPI. El problema surgía cuando, aunque la vinculase dentro del pom.xml y funcionase en desarrollo, al construir el ejecutable, se lanzaba una excepción por no encontrar dicha librería.
Después de múltiples búsquedas por Internet, se pudo llegar a la solución, y es que, para poder vincular librerías locales a Maven, debemos añadir el siguiente código al archivo pom.

```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <configuration>
                <includeSystemScope>true</includeSystemScope>
            </configuration>
            <executions>
                <execution>
                    <goals>
                        <goal>repackage</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

Figura 7.1 Añadido al archivo pom

De esta manera, se solucionaba el error al construir el ejecutable, pero seguía existiendo un error al ejecutar el jar. Este error decía no encontrar la librería.
La manera en que se solucionó este error fue, ejecutando desde consola el comando Maven para poder construir el ejecutable:
mvn -q package
De esta manera, ya sí que se podía ejecutar la aplicación como en desarrollo.

Cabe destacar que ninguno de las IDEs permitía compilar la aplicación de manera efectiva y siempre saltaba algún error. Por esta razón, hubo que recurrir a esta solución.

7.3 Categorizador de opiniones

Dentro de los problemas encontrados en esta parte, están los debidos al entrenamiento, ya que, en un inicio, cualquier categorización que se hiciese daba una mayoría absoluta extrema del PP, es decir, podría obtener como un 94% de los comentarios del PP. Para poder arreglarlo, se borró parte del entrenamiento del PP (ya que era la mayoría) y se buscaron más comentarios de los otros partidos. De esta manera, tendríamos, de forma aproximada, la misma cantidad de comentarios de cada uno, y evitaría el error ya mencionado, que se trataba de un error de *overfitting* (sobreajuste). Esto funcionó bastante bien, pues, combinando diferentes cantidades de iteraciones en el entrenamiento, esta fue la manera en la que se obtuvieron los mejores resultados.

Cabe destacar que, para realizar este entrenamiento, se utilizó el algoritmo MAXENT con un número de 950 iteraciones.

8

Conclusiones

El desarrollo de esta aplicación me ha supuesto numerosos retos, desde el hecho de tener que aprender a manejar un framework como Spring Boot y a entender su funcionamiento; hasta el desarrollo del entrenamiento del categorizador (un proceso muy lento y aburrido) y la realización de la api java (ScrapAPI).
Lo peor sin duda ha podido ser el hecho de encontrar los errores que surgían de manera casi aleatoria, sobre todo al desarrollar ScrapAPI, con errores a la hora de manejar los JSON (que llegaban malformados desde la API de El País), hasta pequeños errores en algunos algoritmos desarrollados. Aún así, aunque haya tardado más de lo que desearía en depurar algunas partes del código, es verdad que, para mí, ha sido un proceso muy enriquecedor.

Gracias a este proyecto he sido capaz de aprender muchísimas cosas, sobre todo, lo complicado que puede llegar a ser a veces una cosa, que, a simple vista parece trivial, y luego puede llegar a dar problemas por no tener en cuenta algún aspecto que no se apreciaba a simple vista. Es muy enriquecedor ver como el proyecto se va realizando y se van obteniendo los resultados que se esperaban, además de poder aprender cosas que hace unos años no pensé que podría llegar a aprender en tan poco tiempo, como puede ser el haber aprendido algo más sobre inteligencia artificial (más concretamente, sobre programación neurolingüística) con la librería de OpenNLP. Además, poder haber aprendido un framework como Spring abre mucho la mente, entender la arquitectura del framework, orientada a los microservicios, y así tener un código mucho más limpio y desacoplado. El hecho de entender la separación entre los controladores, los

servicios, los repositorios y el mapeo de la base de datos puede ayudar mucho a intentar mejorar el código fuera del framework, es decir, intentar crear modelos que sean lo más desacoplados posibles.
Sin duda, para mí ha supuesto un reto, aunque muy gratificante, algo que volvería a repetir, ya que es una pasión el poder aprender cosas nuevas que nunca antes había hecho. La extracción de información de las webs es algo que me ha parecido interesantísimo, que da un potencial muy grande, ya que se puede obtener lo que sea de internet, conseguir la información necesaria de manera automatizada y funcional, por medio de expresiones regulares. Algo muy cautivador y que debería mirar todo ingeniero de software ya que es una herramienta muy potente.
Con todo ello, esto es un recuerdo que me llevo, sin duda algo inolvidable. Espero que esto sea el principio de una gran carrera y de una vida profesional donde pueda aprender y utilizar mis conocimientos.

9

Trabajo futuro

Sin duda, todo trabajo siempre puede ser mejorado y/o ampliado, y este no va a ser una excepción. Este proyecto puede ser totalmente ampliado, se pueden incluir las opciones de realizar más análisis de opinión sobre otros diarios online españoles.
Si quisiéramos ampliar el proyecto y hacerlo internacional, se podrían crear categorizadores para cada país, es decir, escoger los partidos políticos de cada país y analizar los comentarios de las prensas online más famosas e influyentes. Esto incluiría el analizar en diferentes idiomas, lo que implica un mayor trabajo a la hora de la implementación. Adicionalmente, se pueden incluir las opciones de realizar más análisis de opinión sobre otros diarios online españoles.
Además, si consiguiésemos acceder a un sistema o a expertos que fuesen capaces de realizar los ficheros de entrenamiento de manera mucho más fiable, podría darnos unos resultados finales mucho mejores, con una mayor calidad y más fidedigno con la realidad.

Por último, mencionar que la librería de ScrapAPI se diseñó con el objetivo de poder ampliarse de una manera sencilla, es decir, añadir más diarios sólo requeriría de heredar de la clase APIRequest e implementar sus métodos (además de la clase APIParameters y APIResponse, para las cuales sólo sería necesario realizar los métodos que pidiesen ser escritos al heredar de dichas clases). Por lo demás, el algoritmo es lo suficientemente genérico (gracias a la abstracción del modelo) como para no tener que implementar nada más. El único caso en que podría cambiar algo más, sería si los comentarios/noticias tuvieran que recibirse de alguna manera más complicada que la prevista, para lo que podría vincularse con la clase que heredase de APIRequest (haciendo unos pequeños

arreglos, como hice yo en ElPaisRequest) y cambiando el uso de la clase Notice/Comment por la clase Noticia/Comentario que debamos implementar (haciendo que herede de estas). Para este último caso, también está reflejado dentro del proyecto, al realizar la clase ElPaisComment.

10

Referencias

- Librería GSON, para el manejo de los JSON.
https://github.com/google/gson
- Librería JSoup, para realizar las peticiones y recoger las respuestas HTML.
https://jsoup.org/download
- Foto utilizada como fondo en la página index.html (PressOpinionAnalyzer)
https://www.scoop.it/topic/comunicacion-politica-publicaciones-academicas/p/4087002593/2017/10/17/factores-que-inciden-la-percepcion-de-la-imagen-de-un-politico-en-el-escenario-electoral-sotomayor-pereira
- Java
https://docs.oracle.com/javase/7/docs/api/
- Spring Boot
https://spring.io/projects/spring-boot
- Apache OpenNLP
https://opennlp.apache.org/
- Maven
https://maven.apache.org/
- Thymeleaf
https://www.thymeleaf.org/
- Junit 5
https://junit.org/junit5/
- Bootstrap
https://getbootstrap.com/
- Stackoverflow
https://stackoverflow.com/

Anexo A

Manual de usuario

En este manual se enseñará al usuario cómo utilizar la aplicación. Se ilustrará de manera gráfica y sencilla, cómo poder realizar cada una de las acciones pertinentes.

Empezaremos indicando cómo acceder a la aplicación web. Para ello, debemos introducir la dirección URL del servidor (en el caso de que esté desplegada externamente), o "localhost", en el caso de que queramos que el servidor se ejecute localmente. Para poder ejecutar el servidor localmente, diríjase al Anexo B, "Manual de Instalación".

Seguidamente, tras acceder al navegador y a la dirección de la aplicación web, si deseamos analizar los últimos comentarios o noticias de uno de los periódicos online (ABC, Diario Sur o El País), debemos seguir las instrucciones siguientes:

1. Clicar en "Analizar noticias automáticamente".

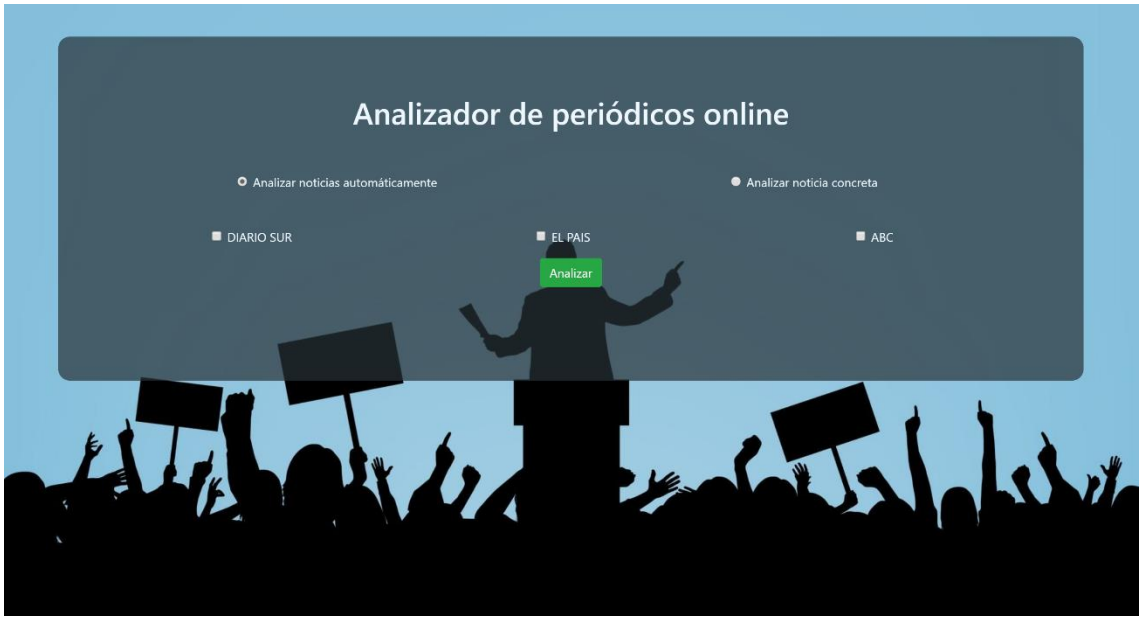

Figura 12.1 Paso 1 Analizar automáticamente

2. Clicar en los periódicos que queramos analizar, en este caso elegiremos El País y ABC.

Figura 12.2 Paso 2 Analizar automáticamente

3. Clicar, dentro de cada uno de los periódicos seleccionados anteriormente, en "Noticias" o en "Comentarios". En este caso se ha elegido "Noticias" para El País y "Comentarios" para el ABC.

Figura 12.3 Paso 3 Analizar automáticamente

4. Rellenamos los nuevos campos de texto que han aparecido debajo de cada periódico seleccionado, indicando el número de comentarios y/o noticias que deseemos. En este caso, se han seleccionado 20 noticias de El País y 100 comentarios del ABC.

Figura 12.4 Paso 4 Analizar automáticamente

5. Seguidamente, presionamos el botón verde "Analizar" y esperamos a que pase la pantalla de carga.

Figura 12.5 Paso 5 Analizar automáticamente

6. Se mostrarán las gráficas pertinentes, con los comentarios a favor de cada partido político, como se muestra a continuación.

Figura 12.6 Paso 6 Analizar automáticamente

En el caso de que deseásemos analizar una noticia en concreto, de cualquiera de los tres diarios, debemos seguir los siguientes pasos:

1. Clicar en "Analizar noticia concreta".

Figura 12.7 Paso 1 Analizar manualmente

2.Clicar en el campo de texto que se abre, y rellenarlo indicando la URL completa de la noticia que deseemos analizar.

Figura 12.8 Paso 2 Analizar manualmente

3. Clicamos en el botón "Analizar", esperaremos a que finalice la carga y podremos visualizar las gráficas generadas por la noticia que hayamos indicado en el campo de texto del punto 2.

Figura 12.9 Paso 3 Analizar manualmente

Anexo B

Manual de instalación

En el caso en que queramos poder instalar esta aplicación web en nuestro propio ordenador, sólo debemos obtener el disco de instalación e introducirlo dentro de la disquetera de la máquina. Entonces, podemos copiar el archivo "PressOpinionAnalyzer-1.0.jar" a nuestro disco duro. A partir de aquí, podemos seguir dos opciones:

1. Ejecutar el servidor en la consola de comandos:

Para ello, debemos abrir la consola, en Windows podemos hacerlo pulsando la combinación de teclas Windows + R e introduciendo "cmd" en el panel de texto que aparece. Seguidamente pulsamos el botón "intro" y se nos abrirá la consola.

- Para ejecutar el servidor, debemos escribir lo siguiente: cd "carpeta", donde "carpeta" se trata de la carpeta en la que se encuentra el archivo PressOpinionAnalyzer.jar. En nuestro caso, sería tal que así:

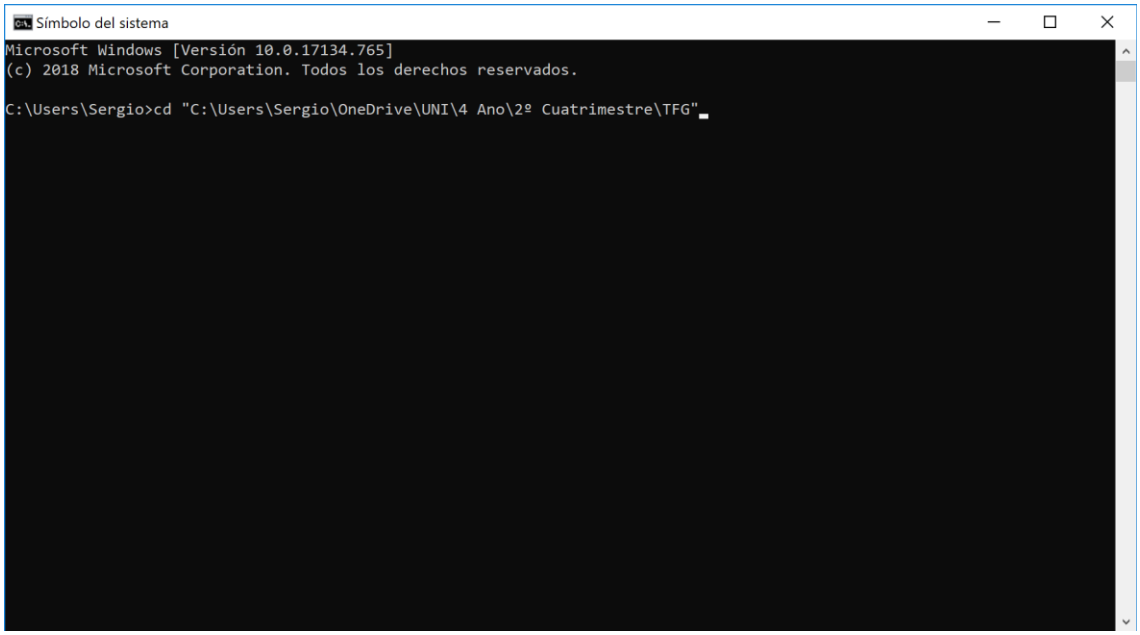

Figura 13.1 Paso 1 Ejecutar en consola

- Seguidamente, escribiremos el siguiente comando:
java -jar PressOpinionAnalyzer.jar
y pulsaremos "intro".

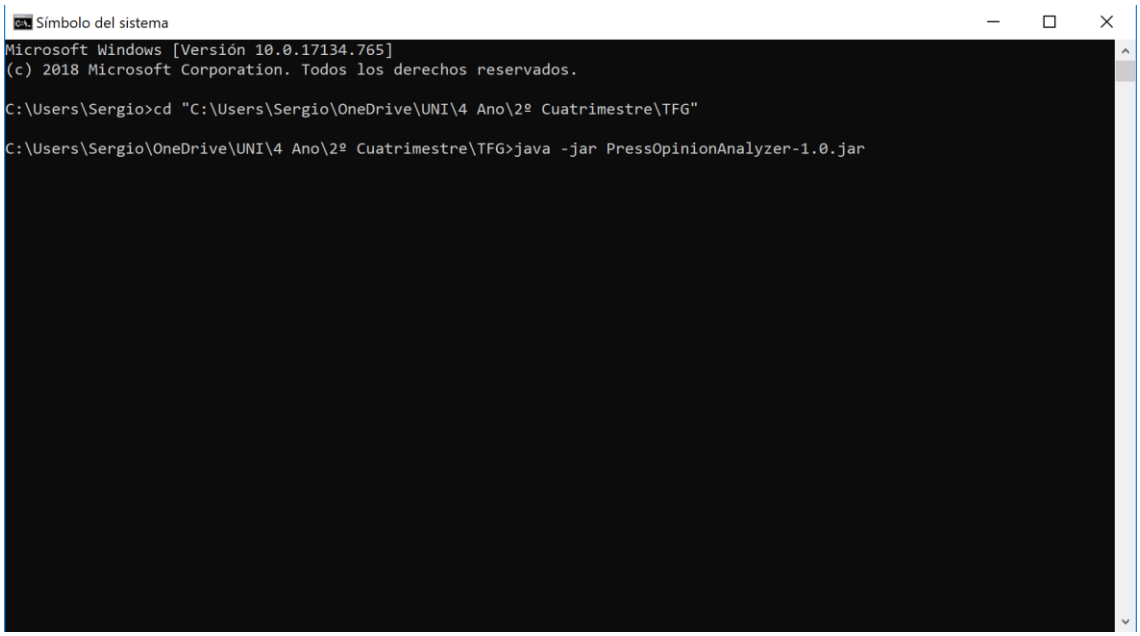

Figura 13.2 Paso 2 Ejecutar en consola

Ahora sólo debemos esperar a que el servidor se despliegue, podremos ver que la consola irá mostrando los avances. En el momento en el que el servidor esté disponible, la consola habrá parado de mostrar texto y se habrá estabilizado, tal y como se muestra en la Figura 13.3:

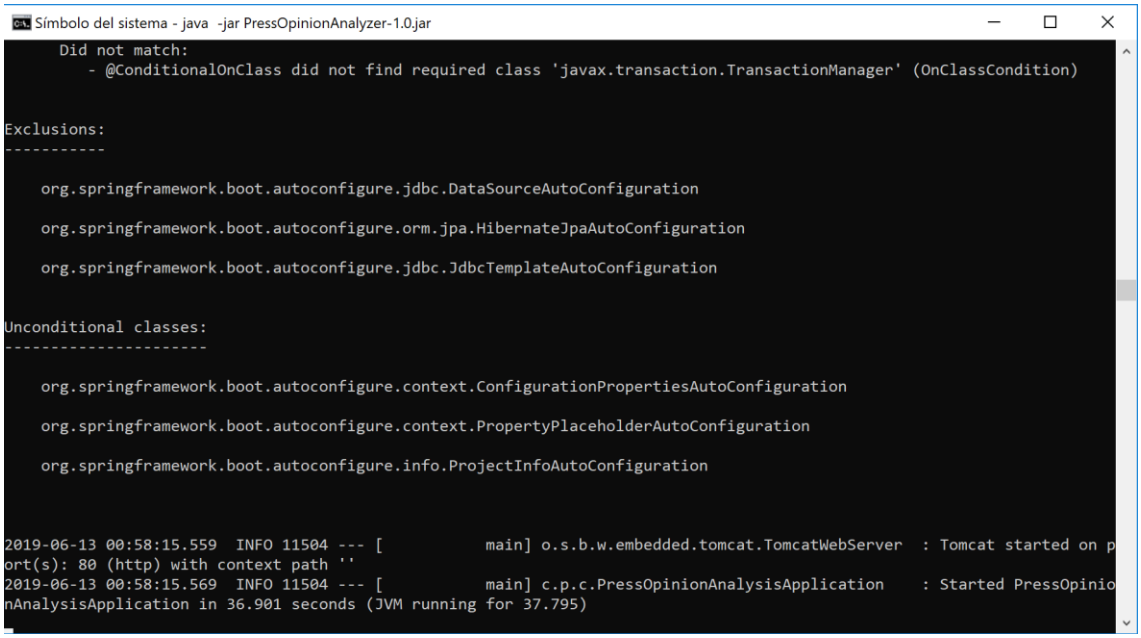

Figura 13.3 Paso 3 Analizar en consola

Podremos leer que en la penúltima y en la última línea pone "Started PressOpinionAnalysisApplication in …". Podremos acceder al servidor desde un navegador, indicando en la URL "localhost".

Para apagar el servidor sólo debemos cerrar la consola dándole a la X.

2. Ejecutar el servidor de manera anónima:

- Para este caso, sólo debemos clicar dos veces encima del icono de "PressOpinionAnalyzer.jar". Debemos esperar un periodo de, alrededor de un minuto, y entonces tendremos el servidor inicializado. Para comprobarlo, sólo debemos acceder a un navegador, indicando la URL "localhost".

- En este caso, si queremos apagar el servidor, sólo debemos acceder al administrador del sistema en Windows con CTRL + ALT + SUPR y cerrar el proceso Java, como podemos ver en la siguiente imagen.

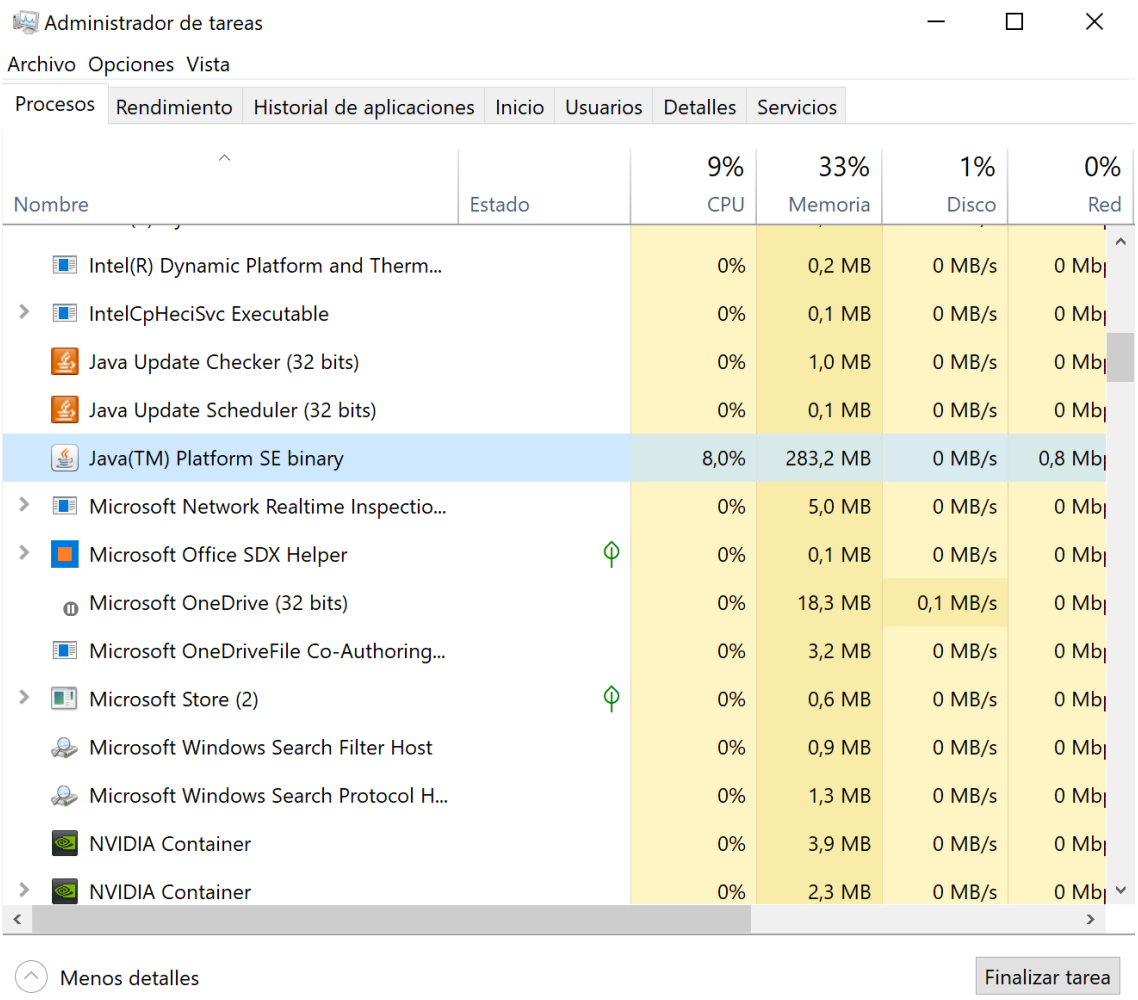

Figura 13.4 Paso 4 Analizar en consola

Requerimientos

Los requerimientos para poder instalar este software son muy simples, de hecho, sólo tiene un requerimiento: tener instalado Java. Lo único necesario para iniciar el servidor es la máquina virtual de java.