

# Trabajo fin de grado

Herramienta de descarga y análisis de datos de Twitter sobre participación ciudadana



José Antonio García del Saz

Escuela Politécnica Superior  
Universidad Autónoma de Madrid  
C\Francisco Tomás y Valiente nº 11



**UNIVERSIDAD AUTÓNOMA DE MADRID**  
**ESCUELA POLITÉCNICA SUPERIOR**



**Grado en Ingeniería Informática y Matemáticas**

**TRABAJO FIN DE GRADO**

**Herramienta de descarga y análisis de datos de  
Twitter sobre participación ciudadana**

**Autor: José Antonio García del Saz**

**Tutor: Iván Cantador Gutiérrez**

**junio 2019**

**Todos los derechos reservados.**

Queda prohibida, salvo excepción prevista en la Ley, cualquier forma de reproducción, distribución comunicación pública y transformación de esta obra sin contar con la autorización de los titulares de la propiedad intelectual.

La infracción de los derechos mencionados puede ser constitutiva de delito contra la propiedad intelectual (*arts. 270 y sgts. del Código Penal*).

DERECHOS RESERVADOS

© 20 de Junio de 2019 por UNIVERSIDAD AUTÓNOMA DE MADRID  
Francisco Tomás y Valiente, nº 1  
Madrid, 28049  
Spain

**José Antonio García del Saz**

*Herramienta de descarga y análisis de datos de Twitter sobre participación ciudadana*

**José Antonio García del Saz**

C\Las Torcas N°1 Portal 3 2ºB

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

*A mi familia, mis amigos, mis compañeros y mis maestros*

*Si quieres aprender, enseña.*

*Cicerón*



# PREFACIO

---

Este estilo de  $\text{\LaTeX} 2_{\varepsilon}$  ha sido diseñado con dos propósitos. El primer propósito es el de facilitar en lo posible la escritura de trabajos de fin de grado y de máster y de tesis doctorales. En ese sentido se han diseñado un conjunto de comandos que simplifican la escritura y diseño de estos trabajos pero que reducen en cierta forma las capacidades de los paquetes de  $\text{\LaTeX}$  utilizados. Sin embargo, dado que los paquetes están incluidos en esta clase, pueden utilizarse directamente y hacer diseños más complejos pero si se hace esto se recomienda mantener una estética coherente con el resto del documento.

El segundo de los propósitos es que estos documentos mantengan una estética uniforme en la Universidad Autónoma de Madrid y fomentar una imagen corporativa en documentos tan relevantes como los trabajos de fin de grado o de máster y las tesis doctorales. Por ese motivo se recomienda mantener una coherencia estética en todo momento. El diseño facilita esa coherencia pero es posible salirse del diseño si se mantiene dicha coherencia.

Como creador de este estilo espero fervientemente que al usar este estilo te sientas cómodo y te facilite la escritura de un documento que es muy relevante en esta etapa de tu vida. Para facilitártela aún más, el código fuente de este documento también está disponible en tu ordenador o en overleaf para que te sirva a modo de ejemplo.

José Antonio García del Saz



# AGRADECIMIENTOS

---

En primer lugar me gustaría agradecer a la Escuela Politécnica Superior por su apoyo para la creación de esta clase y que sea el formato básico para la creación de tesis, trabajos fin de grado y trabajos fin de master.

En particular quiero destacar el trabajo realizado por Fernando López-Colino por su apoyo en la comisión de imagen institucional y por sus comentarios para mejorar este estilo.

También quiero tener un recuerdo para Carmen Navarrete Navarrete dado que este estilo comencé a crearlo a partir de sus necesidades a la hora de escribir la tesis. Y por supuesto a no quiero olvidarme de mi esposa e hijos que han servido de conejillos de indias en sis correspondientes trabajos fin de master y de grado. No quiero olvidar a todos los estudiantes que me pidieron este estilo y lo han usado para presentar sus trabajos pero son muchos y podría olvidarme de alguno, por tanto, mi agradecimiento en general a todos ellos.



# RESUMEN

---

En nuestra Escuela se producen un número considerable de documentos, tanto docentes como investigadores. Nuestros alumnos también contribuyen a esta producción a través de sus trabajos de fin de grado, máster y tesis. El objetivo de este material es facilitar la edición de todos estos documentos y a la vez fomentar nuestra imagen corporativa, facilitando la visibilidad y el reconocimiento de nuestro Centro.

En este sentido se ha intentado diseñar un estilo de  $\text{\LaTeX} 2_{\varepsilon}$  que mantenga una imagen corporativa y con comandos simples que permitan mantener la imagen corporativa con la calidad necesaria sin olvidar las necesidades del autor. Para ello se han creado un conjunto de comandos simples en torno a paquetes complejos. Estos comandos permiten realizar la mayoría de las operaciones que un documento de este tipo pueda necesitar.

Así mismo se puede controlar un poco el diseño del documento a través de las opciones del estilo pero siempre manteniendo la imagen institucional.

## PALABRAS CLAVE

---

Diseño de documento,  $\text{\LaTeX} 2_{\varepsilon}$ , thesis, trabajo fin de grado, trabajo fin de master



# ABSTRACT

---

In our School a considerable number of documents are produced, as many educational as research. Our students also contribute to this production through his final degree, master and thesis projects. The objective of this material is to facilitate the editing of all these documents and at the same time to promote our corporate image, facilitating the visibility and recognition of our center.

In this sense we have tried to design a style of  $\text{\LaTeX}2_{\varepsilon}$  that maintains a corporate image and with simple commands that allow to maintain the corporate image with the necessary quality without forgetting the needs of the author. For this, a set of simple commands have been created around complex packages. These commands allow you to perform most of the operations that a document of this type may need.

Likewise, you can control a little the design of the document through the options of the style but always maintaining the institutional image.

# KEYWORDS

---

Document design,  $\text{\LaTeX}2_{\varepsilon}$ , thesis, final degree project, final master project



# ÍNDICE

---

<b>1 Introducción</b>	<b>1</b>
1.1 Motivación del proyecto .....	1
1.2 Objetivos y enfoque .....	2
<b>2 Participación ciudadana, tecnología y redes sociales</b>	<b>3</b>
2.1 Estado del Arte .....	3
2.2 Tecnologías utilizadas .....	4
<b>3 Análisis de requisitos</b>	<b>5</b>
3.1 Requisitos funcionales .....	5
3.2 Requisitos no funcionales .....	7
<b>4 Concepción y diseño</b>	<b>9</b>
4.1 Modelo de datos .....	11
<b>5 Desarrollo y mantenimiento</b>	<b>19</b>
5.1 Mantenimiento .....	19
<b>6 Resultados y ejemplos</b>	<b>21</b>
<b>7 Conclusiones y trabajo futuro</b>	<b>23</b>
<b>Bibliografía</b>	<b>25</b>
<b>Definiciones</b>	<b>27</b>
<b>Acrónimos</b>	<b>29</b>
<b>Apéndices</b>	<b>31</b>
<b>A Diagramas</b>	<b>33</b>
<b>B Tablas</b>	<b>37</b>
<b>C Imágenes de la interfaz gráfica</b>	<b>39</b>



# LISTAS

---

## **Lista de algoritmos**

## **Lista de códigos**

## **Lista de cuadros**

## **Lista de ecuaciones**

## **Lista de figuras**

2.1	CONSUL en España y el mundo .....	3
3.1	Matriz de compatibilidad reporte-gráfico .....	7
4.1	Diseño del entorno .....	10
4.2	Diagrama UML Principal .....	13
4.3	Ciclo de vida de una tarea asíncrona .....	14
4.4	Funcionamiento general de JFreeChart .....	15
4.5	Diagrama UML Preferencias Gráficos .....	16
4.6	Diagrama UML Análisis Semántico .....	18
A.1	Diagrama UML Tareas Asíncronas .....	34
A.2	Diagrama UML Reportes Análisis .....	35
C.1	Pantalla inicial .....	39
C.2	Registro e inicio de sesión .....	39
C.3	Pantalla de bienvenida .....	40
C.4	Constructor de consultas .....	40
C.5	Edición de credenciales .....	41
C.6	Tarea en segundo plano .....	41

## **Lista de tablas**

2.1	Tecnologías utilizadas .....	4
4.1	Operadores de Twitter API .....	11
B.1	Idiomas soportados por la API de Twitter .....	37

## **Lista de cuadros**

# INTRODUCCIÓN

---

En este Trabajo de Fin de Grado se plantea el desarrollo de una herramienta software que permita la descarga automática desde Twitter de datos sobre participación ciudadana, y el posterior análisis de estos. La herramienta permitirá la configuración de parámetros de entrada para acotar el dominio, temáticas y alcance de los datos a descargar, así como el cálculo y visualización de una serie de gráficas, estadísticas y métricas a partir de los datos descargados. Como caso de uso, se propone evaluar la herramienta con tweets sobre problemáticas, propuestas y discusiones acerca de la ciudad de Madrid y su plataforma electrónica de presupuestos participativos ‘Decide Madrid’.

## 1.1. Motivación del proyecto

Tras los primeros encuentros con el tutor, éste fue introduciéndonos y desarrollándonos una idea sobre la cual él había estado pensando. Consistía en una plataforma o interfaz donde existiese la posibilidad de seleccionar, extraer, analizar y tratar datos (desde distintas fuentes) sobre la participación ciudadana en las ciudades.

La idea sería que cada “módulo” de dicho sistema se dedicase a una de las posibles fuentes y que fuese el propio sistema el que se ocupase de la recopilación, organización y visualización de los resultados.

Entre las fuentes susceptibles de contener información objeto de análisis se encuentran las tan populares redes sociales. Concretamente, en la ciudad de Madrid existe la plataforma Decide Madrid, donde se lanzan y votan propuestas e incluso se vota a qué proyectos se destina el dinero de los presupuestos municipales. Dicha plataforma tiene una cierta integración con Twitter, plataforma en la que los usuarios proponen iniciativas y también opinan y votan.

## 1.2. Objetivos y enfoque

### Lista de objetivos principales

- Desarrollo de herramienta para extraer datos acotables desde Twitter
  - Se usará el lenguaje de programación Java y la API REST de Twitter.
  - Se dotará al sistema de una interfaz gráfica sencilla desde la cual configurar fácilmente los datos a extraer, para después llevar a cabo las extracciones y poder visualizar los datos.
- Tratamiento de los datos
  - Se analizarán los datos extraídos para obtener reportes del volumen de datos, la naturaleza de los datos (analizando hashtags, usuarios, menciones, etc.), la semántica del texto, etc.
  - Se obtendrán representaciones de los reportes obtenidos por medio de gráficos y tablas.
- Extracción de conclusiones
  - Una vez terminado el tratamiento de los datos, se intentarán extraer conclusiones a partir de los resultados.
  - Se analizarán los potenciales resultados que tendría nuestra herramienta en otros escenarios o casos de uso.

# PARTICIPACIÓN CIUDADANA, TECONOLOGÍA Y REDES SOCIALES

---

## 2.1. Estado del Arte

Con la revolución tecnológica ha cambiado de forma sustancial la forma en que los ciudadanos interactuamos con nuestras ciudades y con nuestros vecinos y gobernantes. La explosión de las redes sociales (como Twitter) ha provocado que aparezca la posibilidad de interesar de forma directa (y también pública) a empresas, entidades públicas, individuos, etc, con lo que la diversidad de opiniones públicas y la participación ciudadana abogan por transformar nuestras democracias representativas en democracias más directas como las que ya existieron en Atenas o la República Romana, o como las existen hoy en Suiza.

Es por esto que han nacido proyectos como CONSUL, con los cuales se implementan interfaces que facilitan la participación ciudadana online a través de foros de opinión, de votaciones de propuestas on-line o de herramientas que se adaptan a cada ciudad (a sus distritos, barrios, comunidades, etc.)



(a) Ayuntamientos en España      (b) Ayuntamientos en el mundo

**Figura 2.1:** Ayuntamientos que utilizan el proyecto CONSUL para participación ciudadana.

Este tipo de herramienta ya está siendo utilizada por 33 países, 100 instituciones y 90 millones de ciudadanos en todo el mundo. En concreto en la ciudad de Madrid funciona desde 2015 bajo el nombre de Decide Madrid y está notablemente integrada con la red social Twitter (pueden compartirse a través de ella propuestas, apoyos, opiniones...).

Este tipo de herramientas, junto con las versátiles API de Twitter proporcionan la posibilidad de navegar en un universo de datos de los cuales se pueden formular hipótesis sobre temas tales como qué preocupa a los ciudadanos, cuáles son las medidas más o menos populares, qué usuarios son más activos y cómo se relacionan entre ellos, etc. Y todo esto en tiempo real.

## 2.2. Tecnologías utilizadas

Nombre	Versión	Objetivos
Twitter API	v4.0	Esta API REST nos la proporciona Twitter para acceder a sus datos. Con ella realizaremos consultas que nos permitirán obtener los datos que son sujeto de nuestro análisis
OpenJDK	v12.0.1	La versión OpenSource del más que conocido Java Developpement Kit. Nuestra aplicación se desarrollará en Java y el entorno gráfico lo gestionará JavaFX. El servidor es un proyecto Java EE
Twitter4J	v4.0.7	Una librería Java que ofrece métodos y clases para la explotación de la API de Twitter en el código Java
Spring Framework	v5.1.7	Framework de código abierto para el desarrollo de aplicaciones y contenedor de inversión de control. Nos permitirá compartir recursos entre los diferentes puntos de la aplicación a través de contextos.
Hibernate	v5.4.3	Herramienta de mapeo objeto-relacional que se apoyará sobre el driver jdbc para conectar los módulos de nuestra aplicación al servidor de bases de datos.
PostgreSQL Server	v11.3	Servidor de bases de datos que guardará y gestionará todos nuestros datos.
Apache Tomcat	v8.5.41	Servidor de aplicaciones Java donde estará desplegado el módulo servidor de nuestra aplicación.
Kumo API	v1.17	Librería Java que nos permite crear Word Clouds muy configurables a partir de palabras.
JFreeChart	v1.0.19	Librería para generar gráficos de distintos tipos en código Java. Se usará para mostrar resultados de los análisis.
Apache Lucene	v8.0.0	Librería para la recuperación de información desde texto y web para el código Java. Se usará para la tokenización y tratamiento de textos.
SSL/TLS	v1.2	Protocolo criptográfico que garantiza las conexiones seguras en la red. Se implementará en todas y cada una de las comunicaciones que se realicen entre cada módulo de nuestra aplicación.

**Tabla 2.1:** En esta tabla se citan las tecnologías utilizadas en el proyecto.

# ANÁLISIS DE REQUISITOS

Con el fin de facilitar la concepción y el desarrollo del sistema, se ha procedido a enumerar y clasificar los diferentes requisitos (tanto funcionales como no funcionales) que nuestro sistema debe satisfacer para que la configuración, el funcionamiento y los resultados se desarrollem según lo esperado.

## 3.1. Requisitos funcionales

### Autentificación

**RF-1.**– El acceso al sistema está restringido para usuarios registrados.

**RF-1.1.**– El registro es libre, pudiendo hacerse desde el propio sistema.

**RF-1.2.**– Las credenciales se componen de un nombre de usuario (único) y contraseña

**RF-2.**– Las contraseñas tendrán entre 6-16 caracteres y contendrá al menos una mayúscula, una minúscula y un número.

**RF-3.**– Un usuario que ha iniciado sesión puede cambiar su contraseña desde la GUI para los accesos posteriores.

**RF-4.**– Un usuario que ha iniciado sesión puede eliminar su cuenta, eliminando también todos los datos que hubiere almacenado en la base de datos (extracciones, reportes, gráficos, etc.).

### Extracciones

**RF-5.**– Un usuario puede crear extracciones desde la GUI que serán de su propiedad.

**RF-6.**– El perímetro de una extracción se delimita (durante la creación) a través de filtros, necesarios para la creación de la extracción (cada extracción contiene al menos un filtro).

**RF-7.**– Un usuario puede alimentar una extracción en primer plano (desde la GUI) o en segundo plano (con una tarea asíncrona del servidor).

**RF-8.**– Una extracción podrá alimentarse de forma indefinida en segundo plano.

**RF-9.**– Los tweets extraídos contenidos en una extracción pertenecen a ésta: si se elimina la extracción se eliminan los tweets.

**RF-10.**– Los tweets de las extracciones son consultables en crudo desde la GUI. También pueden eliminarse de forma individual desde la GUI.

**RF-11.**– Los tweets de una extracción se pueden exportar en un fichero XML para su integración externa.

**RF-12.**– Una misma extracción no podrá contener dos veces el mismo tweet.

## Credenciales de la API de Twitter

- RF-13.**– Un usuario puede añadir, modificar y eliminar credenciales para la API de Twitter en su cuenta.
- RF-14.**– El usuario debe tener al menos unos credenciales añadidos para poder comenzar a crear extracciones, tareas y reportes analíticos.

## Tareas asíncronas

- RF-15.**– Un módulo servidor web existirá para gestionar la ejecución de tareas asíncronas en segundo plano.
- RF-16.**– La conexión entre la GUI y el servidor es configurable desde la GUI, y esta configuración se guarda en el registro del sistema operativo.
- RF-17.**– La GUI se comunicará con el módulo servidor a través de servicios web (SOAP sobre SSL/TLS).
- RF-18.**– Las tareas asíncronas tienen un ciclo de vida específico que depende del tipo de tarea. (Ver diagrama 4.3)
- RF-19.**– Con el arranque del servidor, las tareas existentes son cargadas desde la base de datos.
- RF-20.**– Existen tipos de tareas que se ejecutan de forma indefinida si nunca son detenidas
- RF-21.**– El usuario puede crear, eliminar, preparar, lanzar y parar tareas asíncronas desde la GUI.
- RF-22.**– Una tarea puede ser programada para ejecutarse en un momento dado del futuro.
- RF-23.**– Tras un reinicio del servidor, las tareas programadas aún sin caducar deben ser reprogramadas automáticamente, las tareas que se estaban ejecutando deberán volver a ejecutarse automáticamente y las tareas que hayan caducado pasarán a dicho estado.
- RF-24.**– La ejecución de una tarea programada puede ser cancelada pasando dicha tarea al estado "preparada".

## Análisis

- RF-25.**– Un usuario puede crear, modificar y eliminar diversos tipos de reportes analíticos sobre los datos extraídos.
- RF-26.**– Los contenidos (registros) de un reporte pueden ser actualizados para no quedar obsoletos con el tiempo.
- RF-27.**– Un reporte puede ser actualizado en segundo plano con una tarea asíncrona de servidor.
- RF-28.**– Los datos de un reporte se pueden consultar en crudo en la GUI y exportarse en un archivo .csv para su integración externa.
- RF-29.**– Se guardarán en base de datos tanto el instante en que se creó el reporte como el instante en el que se actualizó por última vez.
- RF-30.**– Existen reportes sobre la evolución del volumen de tweets en el tiempo.
- RF-31.**– Existen reportes sobre las tendencias en las extracciones (hashtags, usuarios, menciones, términos...).
- RF-32.**– Existen reportes sobre la clasificación semántica de los tweets.

## Gráficos

- RF-33.**– Desde la GUI, un usuario puede crear, configurar, modificar y eliminar distintos tipos de gráficos que muestran los datos de los reportes disponibles.
- RF-34.**– Cada tipo de gráfico tiene unas configuraciones específicas (tipos de línea, colores, tamaño y estilo de fuentes, etc.) que serán parametrizables desde la GUI durante la creación del gráfico.
- RF-35.**– Las configuraciones de cada tipo de gráfico se guardan en base de datos para ser reutilizadas posteriormente.
- RF-36.**– Los gráficos pueden tanto verse desde la GUI como exportarse a un archivo JPEG.
- RF-37.**– Se podrán generar Word Clouds desde la GUI.
- RF-38.**– Existe una matriz de compatibilidad entre los tipos de reportes y los tipos de gráficos que son compatibles

entre sí:

Este diagrama es una matriz que muestra la compatibilidad entre diferentes tipos de reportes analíticos en las filas y diferentes tipos de gráficos en las columnas. Una diagonal negra de celdas marcadas con un 'X' indica que cada tipo de reporte es compatible consigo mismo. Las filas representan los tipos de reporte y las columnas representan los tipos de gráfico.

	Time Series Chart	XY Bar Chart	Category 3D Bar Chart	3D Pie Chart	Pie Chart	Word Cloud
Timeline Tweet Volume Report	X					
Timeline Top N Hashtags Report	X	X				
Trending Hashtags Reports	X	X	X			
Trending User Mentions Report	X	X	X	X		
Trending Users Report	X	X	X	X	X	
Trending Words Report	X	X	X	X	X	
Tweet Volume by Topics Report	X	X	X	X	X	
Tweet Volume by Named Entities Report	X	X	X	X	X	

**Figura 3.1:** Matriz de compatibilidad entre los tipos de reporte analítico y los tipos de gráficos.

### Procesamiento del lenguaje natural (NLP)

- RF-39.**– Para cada idioma, un usuario puede crear listas de palabras ignorables en ese idioma. De esta manera se permite personalizar las palabras irreverentes en cada contexto y en cada idioma.
- RF-40.**– Las palabras ignorables no se tendrán en cuenta para la medición de frecuencias o la elaboración de reportes relacionados con dichas frecuencias.
- RF-41.**– Para cada idioma, el usuario podrá crear, modificar y eliminar diferentes preferencias para el procesamiento del lenguaje natural desde la GUI. Dichas preferencias contienen una serie de categorías y subcategorías (temas) a través de las cuales luego se podrán clasificar los tweets.
- RF-42.**– Dado un conjunto de extracciones, un usuario puede separar todos sus tweets en palabras para obtener un conjunto de términos presentes en los tweets.
- RF-43.**– Los conjuntos de términos pueden crearse y eliminarse desde la GUI y se almacenan en la base de datos.
- RF-44.**– Dado un conjunto de términos y unas preferencias NLP, el usuario puede clasificar los términos en las diferentes categorías y subcategorías desde la GUI. De este modo se entrena la clasificación semántica de tweets en cada contexto.
- RF-45.**– La clasificación de los términos se puede hacer en paralelo (varios humanos usando la aplicación en varias máquinas) sin que se clasifique la misma palabra dos veces ni en serie ni en paralelo.

## 3.2. Requisitos no funcionales

### Entorno

- RNF-1.**– El sistema es distribuido, siendo sus módulos aislables y orientables.
- RNF-2.**– Las conexiones con la base de datos no se realizan en texto plano. Debe cifrarse punto a punto mediante SSL/TLS.
- RNF-3.**– Los diferentes módulos generan ficheros de log para el mejor trazado de las actividades.
- RNF-4.**– Los logs históricos se comprimen para optimizar el uso de almacenamiento.
- RNF-5.**– El sistema debe proporcionar mensajes de error que sean informativos y orientados a usuario final.
- RNF-6.**– Las tareas largas (acceso a datos, análisis, generación de reportes...) no bloquean la GUI completamente: se ejecutan en otro hilo mientras se muestra una ventana modal que informa de las acciones que ocurren en paralelo.

## Autentificación

**RNF-7.**– Las contraseñas de los usuarios no se pueden guardar en texto plano en la base de datos.

## Extracciones

**RNF-8.**– Una extracción no puede ser alimentada desde dos lugares distintos al mismo tiempo, ni siquiera desde la misma máquina.

**RNF-9.**– Se tendrán en cuenta las restricciones para las credenciales gratuitas que tiene la API de Twitter.

## Tareas asíncronas

**RNF-10.**– Una tarea no puede ejecutarse varias veces en paralelo, sólo se concibe una ejecución a la vez por tarea.

**RNF-11.**– Ningún intercambio de datos entre la GUI y el servidor se puede hacer en texto plano. Siempre se debe usar el cifrado punto a punto sobre SSL/TLS.

# CONCEPCIÓN Y DISEÑO

Dados los requisitos anteriores, llega el momento de decidir la forma en que vamos a satisfacerlos. En primer lugar se enumeraran los distintos módulos que se desarrollarán, la concepción del modelo de datos y las infraestructuras sobre las que se distribuirá el sistema.

## Aplicaciones ejecutables

Como nombre para nuestro sistema se ha elegido **TweetExtractor**, y contendrá dos aplicaciones:

- **TweetExtractorFX:** Es la interfaz gráfica de usuario. Comprende un conjunto de ventanas a través de las cuales se pueden realizar diferentes actos de gestión de la aplicación como gestión de usuarios, extracciones y preferencias; visualización de resultados, reportes y gráficos... Se ha elegido JavaFX (se desarrolla en Java+XML) como librería para la creación de la interfaz gráfica debido a que ya viene integrada con el JDK y también a su sencillez, versatilidad e integración con el back-end en Java. Gracias a Java, podremos ejecutarla en todos los sistemas operativos que lo soporten.
- **TweetExtractor Server:** Es una aplicación web que funciona a modo de servidor. Sigue ejecutándose aunque la interfaz gráfica esté cerrada y se encarga de gestionar las tareas asíncronas (las crea, las modifica, controla su ejecución, las elimina...). Se desarrollará en Java y se desplegará en un servidor de aplicaciones Apache Tomcat. Se comunicará con TweetExtractorFX a través de servicios web Java, que funcionarán sobre la infraestructura de los protocolos SOAP + HTTP + SSL/TLS.

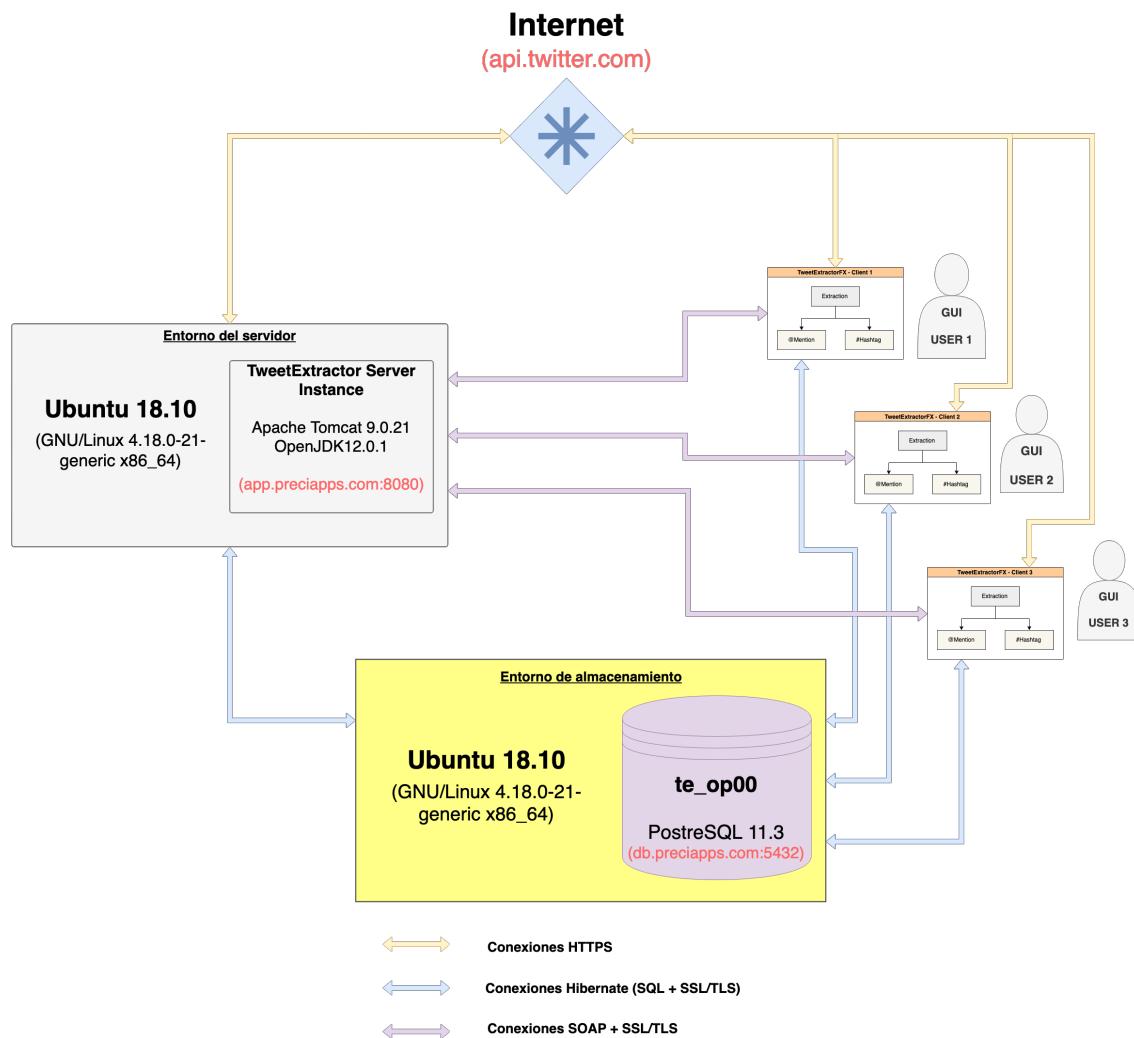
## Entornos

Como podemos observar en el diagrama inferior (4.1), podemos separar los entornos en 3:

- **Entornos de usuario:** Son los entornos (que pueden ser distintos) donde se ejecutaría la interfaz gráfica. Debe tener conexión a Internet para conectarse a Twitter. Se limitarán los entornos compatibles para acotar la fase de pruebas: sólo se probará en Ubuntu (GNU/Linux), macOS y Microsoft Windows con una versión de Java superior a 1.8.
- **Entorno del servidor:** Es el entorno que más carga de memoria soporta. Debe tener conexión a Internet para conectarse a Twitter. Puede ser cualquier entorno donde se pueda desplegar un contenedor de aplicaciones web de Java. En nuestro caso se ha elegido un servidor Linux con una instancia de Apache Tomcat 9.0.21. Se podrá acceder desde Internet a nuestra instancia a través del nombre de dominio app.preciapps.com en el puerto 8080.

- **Entorno de almacenamiento:** Es el entorno que más carga de almacenamiento y conexiones de red soporta. En él se ejecutará una instancia de servidor de base de datos PostgreSQL. En nuestro caso, hemos optado también por un servidor Linux con una instancia PostgreSQL v11.3. Se podrá acceder desde Internet a nuestra instancia a través del nombre de dominio db.preciapps.com en el puerto 5432.

El diagrama siguiente trata de desglosar y hacer más fácilmente comprensible la configuración de los entornos y las interconexiones entre ellos. Nótese que los distintos entornos pueden también fusionarse (incluso funcionar en una única máquina), pero optaremos por no hacerlo por motivos de rendimiento (con los entornos separados la optimización de cada uno de ellos es más sencilla).



**Figura 4.1:** Diagrama que muestra el diseño del entorno, con los diferentes módulos y las conexiones entre ellos.

## 4.1. Modelo de datos

### Extracciones y filtros

Esta es una parte muy importante de la construcción del modelo de datos. Comenzaremos creando la clase Extracción, que pertenecerá al usuario que la creó y contendrá una lista de tweets extraídos desde Twitter. Siguiendo los requisitos funcionales, un usuario debe poder configurar qué condiciones deben cumplir los tweets que desea extraer. En este sentido tenemos que tomar como referencia la API de Twitter.

Con ella, podemos ejecutar consultas utilizando operadores, de los cuales algunos se enumeran en la documentación de la siguiente manera:

Operador	Encuentra Tweets...
viendo ahora	que contienen ambos términos “viendo” y “ahora”. Este es el operador principal.
“estado civil”	que contienen la frase exacta “estado civil”
amor OR odio	que contienen la palabra “amor” o bien la palabra “odio” (o ambas).
cerveza -raíz	que contienen la palabra “cerveza”, pero no contienen la palabra “raíz”
#haiku	que contienen el hashtag #haiku
from:interior	enviados desde la cuenta de Twitter “interior”
list:NASA/astronauts-in-space-now	enviado desde una cuenta de twitter en la lista “astronauts-in-space-now” de NASA
to:NASA	enviados en respuesta a la cuenta de Twitter “NASA”
@NASA	mentionando a la cuenta de Twitter “NASA”
política -filter:safe	que contienen “política” pero no están marcados como potencialmente sensibles
año -filter:retweets	que contienen “año” pero no son retweets.
madrid filter:images	que contienen “madrid” y tienen una imagen adjunta.
batería url:amazon	que contienen la palabra “batería” y una URL con la palabra “amazon” en cualquier lugar de ella.
as since:2015-12-21	que contienen “as” y fueron enviados desde el 21 de diciembre de 2015.
onu until:2015-12-21	que contienen “onu” y fueron enviados hasta el 21 de diciembre de 2015.
película :)	que contienen “película” y tienen una actitud positiva.
vuelo :(	que contienen “vuelo” y tienen una actitud negativa.
tráfico ?	que contienen “tráfico” y hacen una pregunta.

**Tabla 4.1:** En esta tabla se muestran algunos operadores que soportan las consultas a la API de Twitter.

Para modelar estos operadores se ha creado la clase Filtro. Una extracción contiene una lista de filtros (que serán configurables por el usuario desde la GUI), y cada uno de estos filtros se podrá traducir en una cadena de caracteres que se corresponderá con el texto de su operador análogo. Por tanto, al realizar la extracción podremos encadenar la lista de operadores extraída desde la lista de filtros y construir la consulta. Tras un análisis de los operadores distinguimos dos grandes tipos de filtros:

- **Filtros lógicos:** Se corresponden con los operadores lógicos “OR” y “NOT” (el operador “AND” ya equivale a concatenar dos operadores). Son filtros que actúan como operadores entre filtros (“OR” es un operador que actúa sobre dos filtros y “NOT” sobre un filtro).
- **Filtros no lógicos:** Son el resto de filtros que no actúan como operadores sobre otros filtros sino que actúan (metafóricamente) como constantes. Una extracción debe contener al menos uno de estos filtros para poder lanzar una consulta.

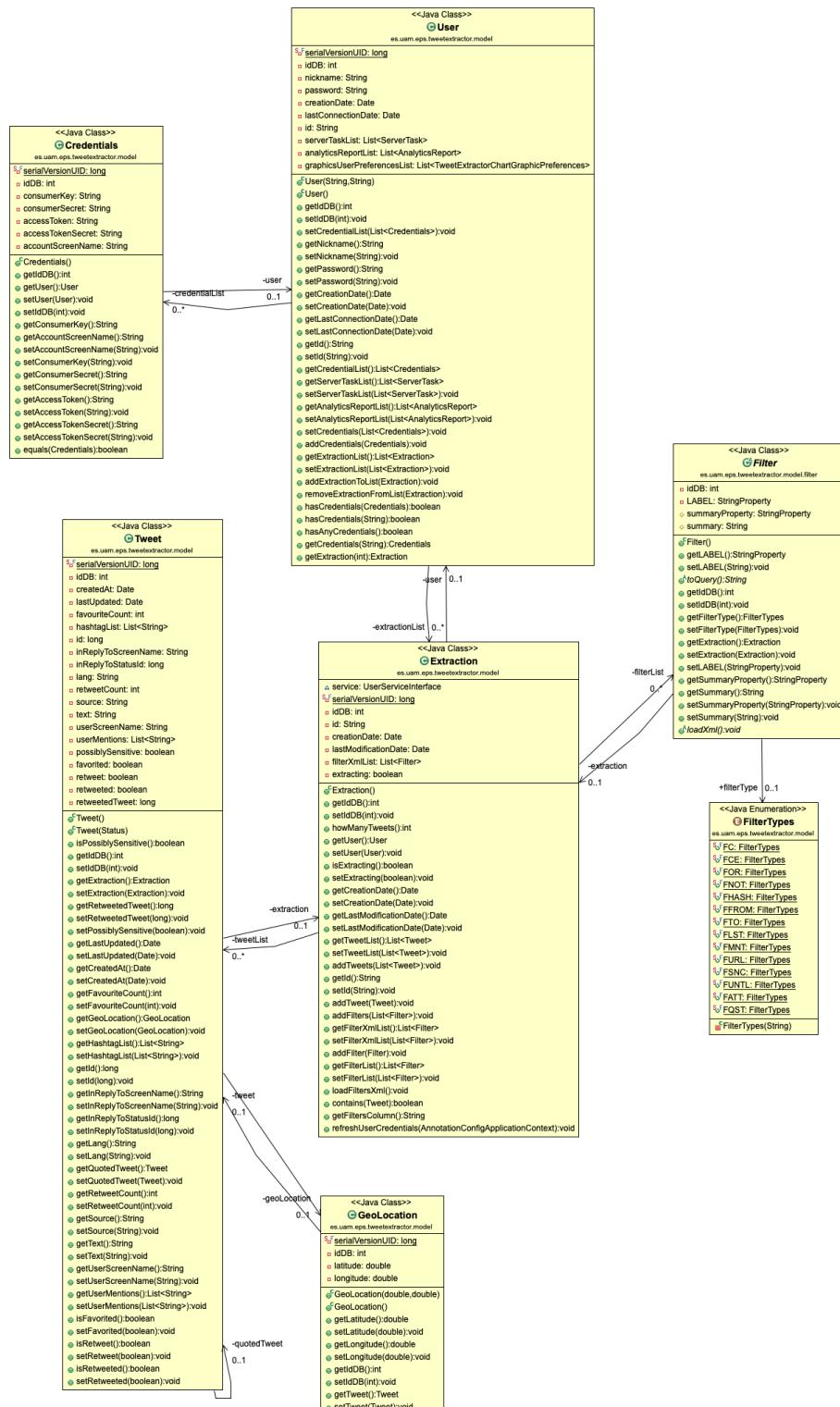
## Usuarios y credenciales

Para cumplir con los requisitos funcionales de autentificación, crearemos clases Java que representen a los distintos usuarios que se pueden registrar y autenticar en nuestro sistema. Cada usuario estará identificado por un nombre único y una contraseña que usará para acceder a la GUI.

## Tweets y Geolocalizaciones

Para modelar un tweet hemos creado la clase Java Tweet. Esta clase contendrá como atributos los datos y metadatos que nos parece interesante guardar para cada tweet extraído (nos hemos basado en la clase Status de la librería Twitter4J, que también encapsula los tweets):

- Fecha de creación
- Recuento de veces marcado como favorito
- Recuento de veces retuiteado
- Lista de hashtags contenidos
- ID de Twitter: es un número entero que identifica inequívocamente un Tweet.
- Cuenta a la que responde el Tweet (si procede)
- Tweet original al que responde el tweet (si procede).
- Idioma del tweet (si se reconoce).
- Origen del tweet (iOS, Android, Twitter Web,etc.)
- Cuerpo del tweet (texto)
- Cuenta desde la que se envía el tweet
- Lista de menciones a otras cuentas
- Marcador de tweets potencialmente sensibles
- Geolocalización desde la cual se envió el tweet (si está disponible). Se encapsula en una clase que contiene la latitud y la longitud como atributos.
- Añadiremos un identificador numérico para nuestra base de datos aparte del que nos proporciona Twitter (por que nosotros podemos guardar el mismo tweet dos veces en dos extracciones diferentes)



**Figura 4.2:** Diagrama UML de las clases principales del modelo de datos. Contiene las clases Usuario, Extracción, Tweet, Filtro, Credenciales, Geolocalización y Filtro

## Credenciales de la API de Twitter

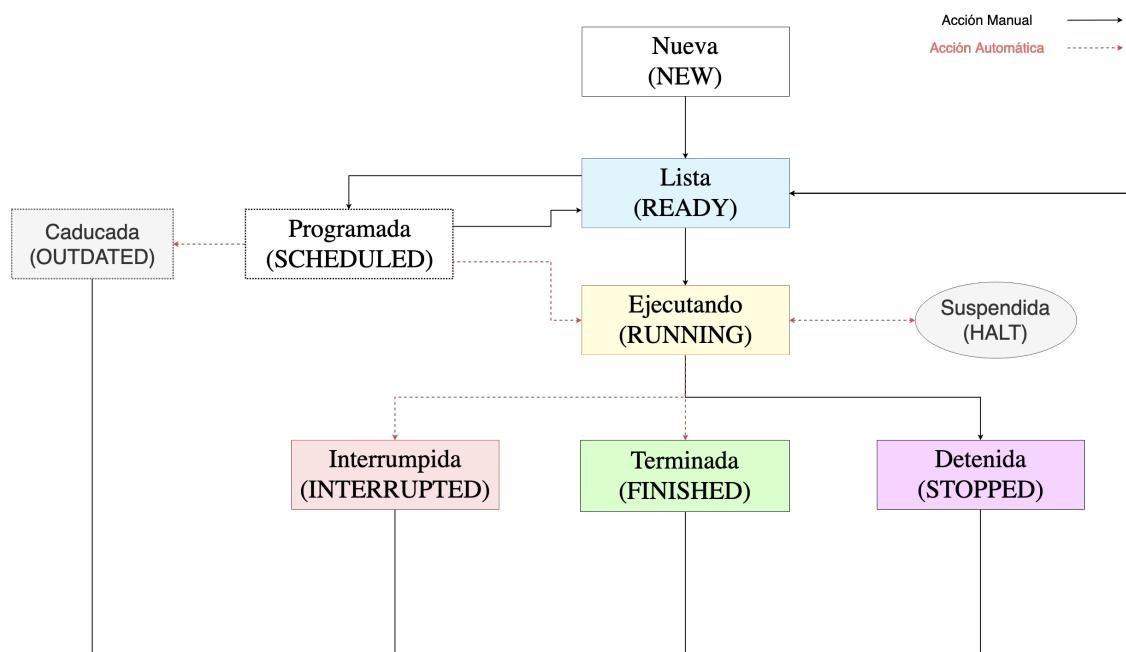
Para modelar unos credenciales de la API de Twitter hemos creado la clase Credenciales, la cual contendrá las cuatro cadenas de caracteres o “tokens” que nos autentifican en el servicio, así como el nombre público de la cuenta.

## Tareas asíncronas del servidor

Para modelar las tareas asíncronas y sus diferentes tipos se creará la clase TareaServidor y todas las clases que heredarán de ella. Según su funcionalidad, se modelarán dos grandes tipos de tareas:

- **Tareas de extracción:** Tareas que realizarán alguna acción concreta sobre una extracción, por ejemplo alimentar una extracción indefinidamente (ver RF-8).
- **Tareas de análisis:** Se encargarán de generar y/o actualizar reportes analíticos sobre los tweets ya extraídos.

El diagrama UML completo conteniendo los tipos de tareas está disponible en el anexo (ver A.1). Las tareas tendrán un ciclo de vida muy concreto que se modelará mediante unos estados en los que puede estar la tarea dependiendo de su tipo, de las acciones manuales de los usuarios y de los eventos que puedan ocurrir durante sus ejecuciones. Este ciclo de vida se muestra en el siguiente diagrama:



**Figura 4.3:** Ciclo de vida de una tarea asíncrona en el servidor. Las líneas continuas marcan acciones manuales (del usuario). Las líneas discontinuas con color marcan acciones automáticas (del servidor)

Una tarea tendrá el estado “Nueva” cuando sea creada y podrá ser eliminada en cualquier estado. Los estados rodeados por línea discontinua son específicos de algún tipo de tarea (el estado “Programada” sólo sera alcanzable para las tareas programables).

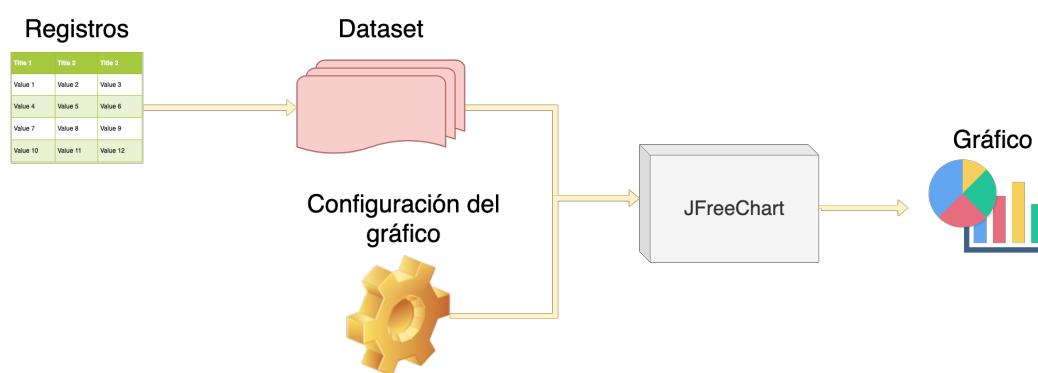
## Reportes Analíticos y Registros

Siguiendo los requisitos funcionales, una vez extraídos y guardados los tweets se podrán crear, modificar y eliminar diferentes tipos de reportes que contendrán los resultados de los análisis que realicemos sobre los datos.

Para modelar todos los tipos de reportes se creará la clase ReporteAnalítico y todas las clases que heredarán de ella. Cada distinto tipo de reporte tendrá unos registros que contendrán los resultados, siendo estos registros también de tipos diversos dependiendo del tipo de reporte al que pertenezcan. El diagrama UML que representa las clases e interfaces más relevantes que modelan estos reportes y registros se encuentra en los apéndices (ver A.2)

## Gráficos

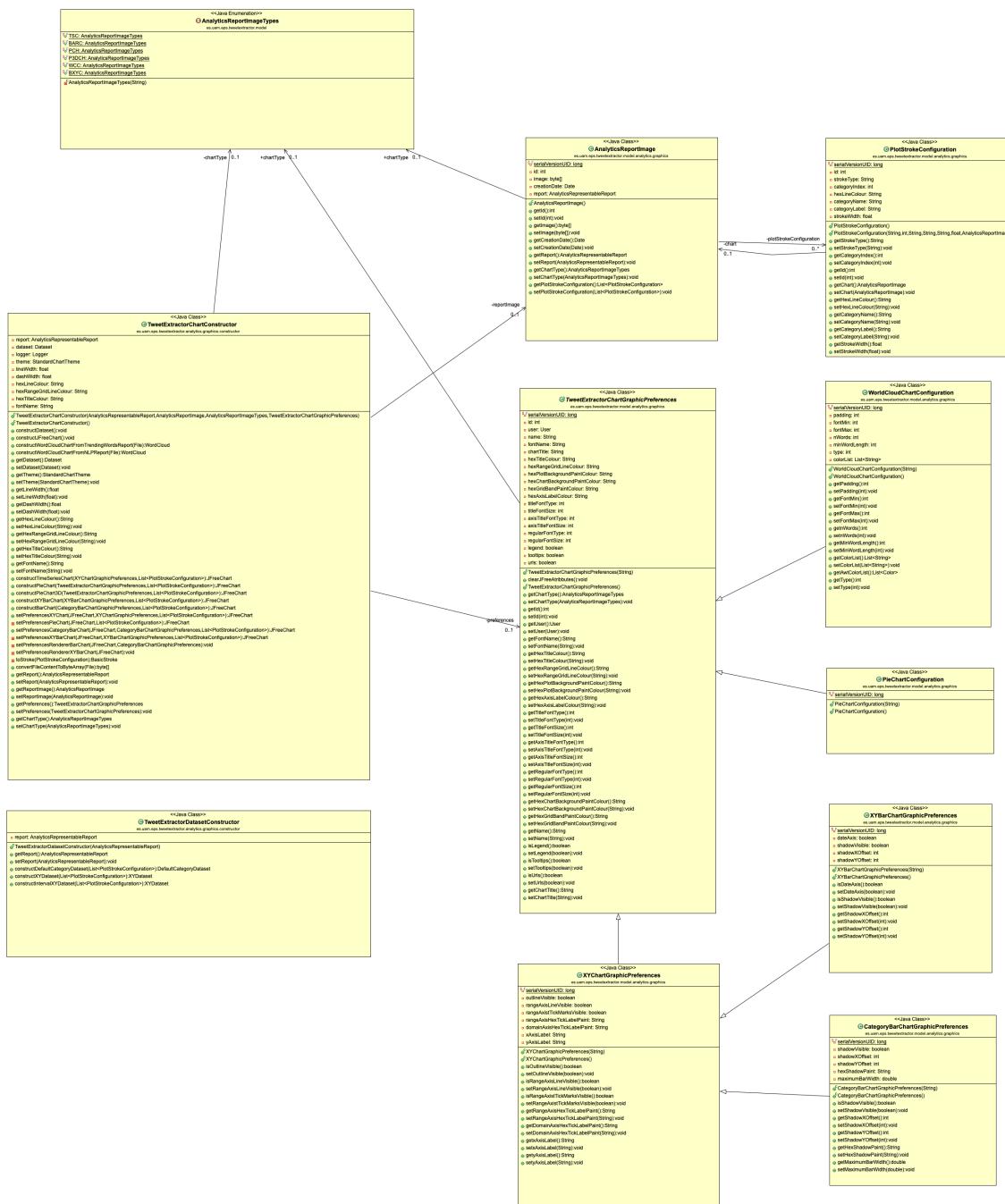
Para la elaboración de gráficos se usará la librería externa JFreeChart. Esta librería nos permitirá, en nuestro entorno Java, configurar y generar gráficos sencillos que mostrarán los resultados de los reportes analíticos que hayamos generado. Soporta distintos tipos de gráficas (barras, línea, circular,etc). Para generar un gráfico, esta librería nos ofrece un constructor que requiere un conjunto de datos o “Dataset” (clases incluidas en la librería que encapsularán los valores de los registros) y una configuración para el gráfico (leyendas, fuentes, tipos de linea, colores, etc.) Este funcionamiento se muestra en este diagrama:



**Figura 4.4:** Diagrama que muestra el proceso de generación de un gráfico con la librería JFreeChart

Para modelar los gráficos que se vayan generando se creará la clase GráficoReporteAnalítico (guardaremos las gráficas en base de datos). También crearemos clases para encapsular las configuraciones posibles de cada tipo de gráfico (estas clases heredarán de la clase TweetExtractorPreferencias-Gráfico).

A continuación aparece el diagrama UML completo que muestra toda esta sección del modelo de datos:



**Figura 4.5:** Diagrama UML que muestra el modelo de datos que representan los gráficos de los reportes y las configuraciones para generarlos.

## Análisis semántico de los Tweets

Por último, y de acuerdo con el último apartado de los requisitos funcionales, se añadirán al modelo los objetos que se usarán para el análisis semántico de los tweets.

Si hablamos de semántica, lo primero que debemos tener en cuenta es el idioma del texto que vamos a analizar.

La API de Twitter ya nos indica de forma nativa en qué idioma está escrito cada tweet. En el anexo puede encontrarse una referencia de los idiomas que la API es capaz de identificar (ver B.1). Para modelar los idiomas se creará un referencial inmutable de idiomas encapsulados en la clase IdiomaTwitterDisponible.

Dado un idioma, se podrán crear distintas listas de palabras ignorables que no se tendrán en cuenta para el cálculo de frecuencias. Dado un idioma y un conjunto de extracciones, un usuario debe poder partir los tweets extraídos en términos y almacenarlos para su posterior clasificación, y para ello construiremos una clase ConjuntoTokensPersonal.

Dado un idioma un usuario también podrá crear diferentes configuraciones para la clasificación de los términos. Estas configuraciones se modelarán como un conjunto de categorías y subcategorías. Los términos obtenidos se podrán clasificar en estas categorías, y esta clasificación se almacenará también en base de datos para tenerla en cuenta en los posteriores análisis.



**Figura 4.6:** Diagrama UML que muestra el modelo de datos utilizado para modelar los objetos relacionados con el análisis semánticos de los datos.

## DESARROLLO Y MANTENIMIENTO

---

### 5.1. Mantenimiento



## RESULTADOS Y EJEMPLOS

---



## CONCLUSIONES Y TRABAJO FUTURO

---



# BIBLIOGRAFÍA

---

[1] YUSUKE YAMAMOTO, *Twitter4J - A Java library for the Twitter API* Visitar.



# DEFINICIONES

---

**acrónimo** Sigla cuya configuración permite su pronunciación como una palabra; por ejemplo, ovni: objeto volador no identificado; TIC, tecnologías de la información y la comunicación.

**definición** Proposición que expone con claridad y exactitud los caracteres genéricos y diferenciales de algo material o inmaterial.

**opción de estilo** Son los valores que modifican el funcionamiento del estilo. Se ponen entre corchetes y separadas por comas en el comando \documentclass y antes de el nombre del estilo que irá entre llaves.



# ACRÓNIMOS

---

**IEEE** Institute of Electrical and Electronics Engineers.

**WYSIWYG** What You See Is What You Get.

**WYTIWYG** What You Think Is What You Get.



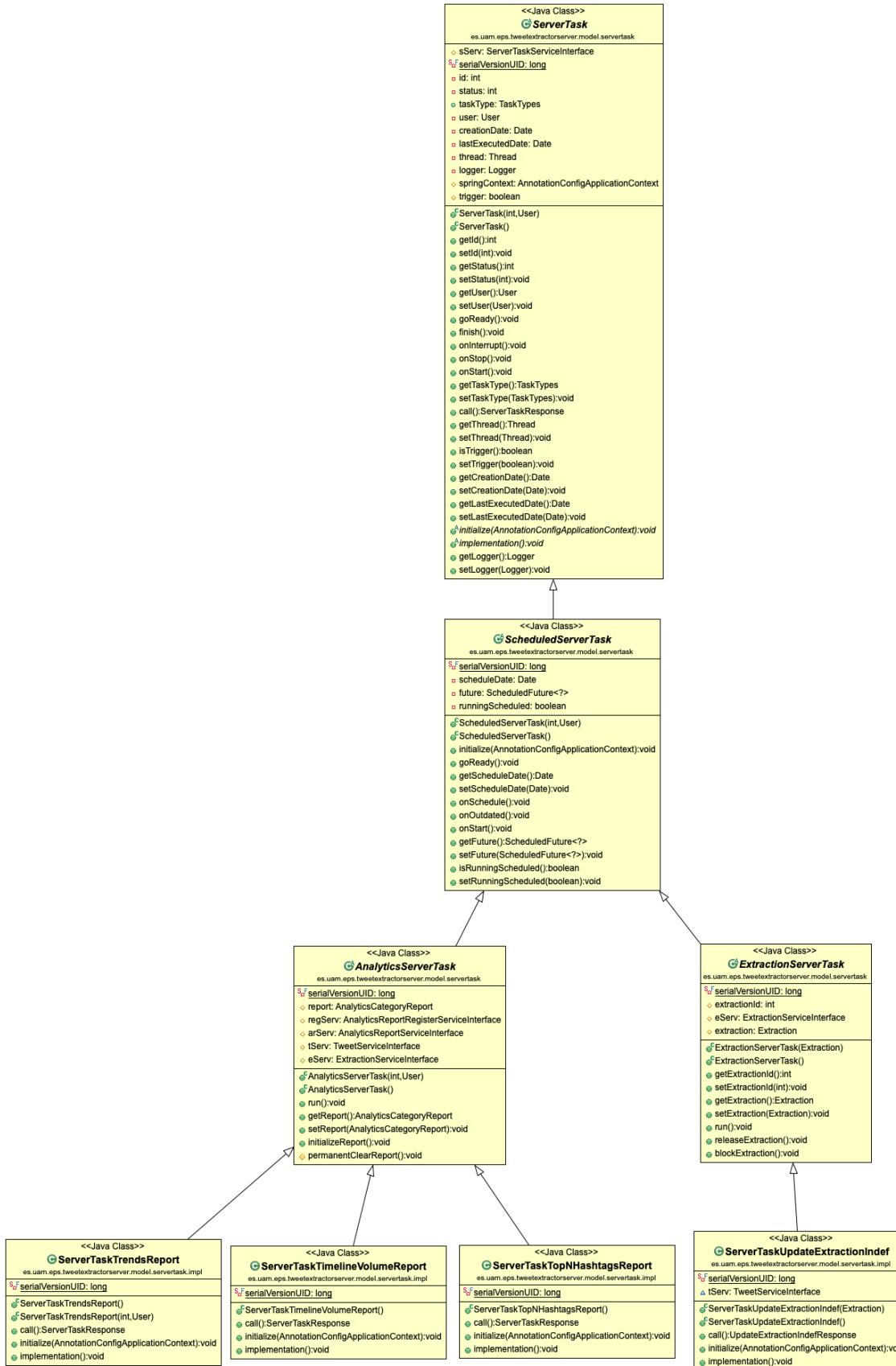
# **APÉNDICES**



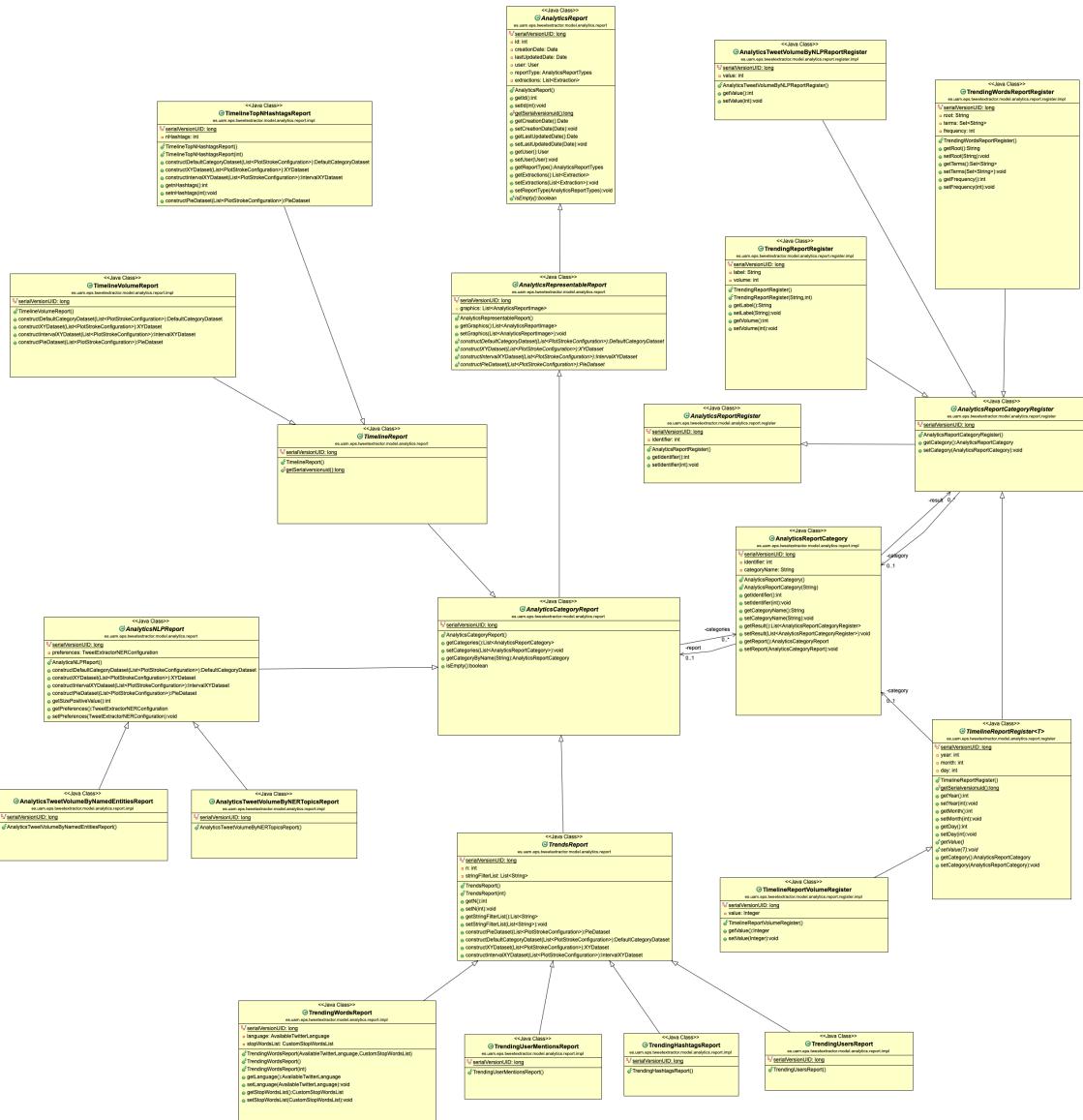
| A

## DIAGRAMAS

---



**Figura A.1:** Diagrama UML que muestra las clases que representan las tareas asíncronas del servidor, con todos sus diferentes tipos.



**Figura A.2:** Diagrama UML que muestra de forma general las clases que representan los reportes de los análisis



# TABLAS

---

Idioma	Código	Idioma	Código
French	fr	Hungarian	hu
English	en	Persian	fa
Arabic	ar	Hebrew	he
Japanese	ja	Urdu	ur
Spanish	es	Thai	th
German	de	Ukrainian	uk
Italian	it	Catalan	ca
Indonesian	id	Irish	ga
Portuguese	pt	Greek	el
Korean	ko	Basque	eu
Turkish	tr	Czech	cs
Russian	ru	Gallegan	gl
Dutch	nl	Romanian	ro
Filipino	fil	Croatian	hr
Msa	msa	En-gb	en-gb
Zh-tw	zh-tw	Vietnamese	vi
Zh-cn	zh-cn	Bengali	bn
Hindi	hi	Bulgarian	bg
Norwegian	no	Serbian	sr
Swedish	sv	Slovak	sk
Finnish	fi	Gujarati	gu
Danish	da	Marathi	mr
Polish	pl	Tamil	ta
		Kannada	kn

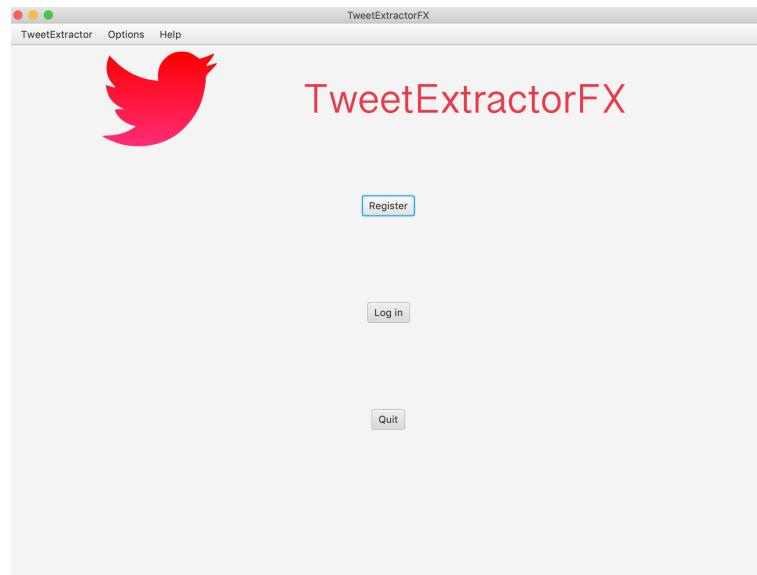
(a) Idiomas (I)

(b) Idiomas (II)

**Tabla B.1:** En esta tabla se muestran los idiomas que la API de Twitter es capaz de identificar. Serán los idiomas para los que funcione nuestro análisis semántico



# IMÁGENES DE LA INTERFAZ GRÁFICA



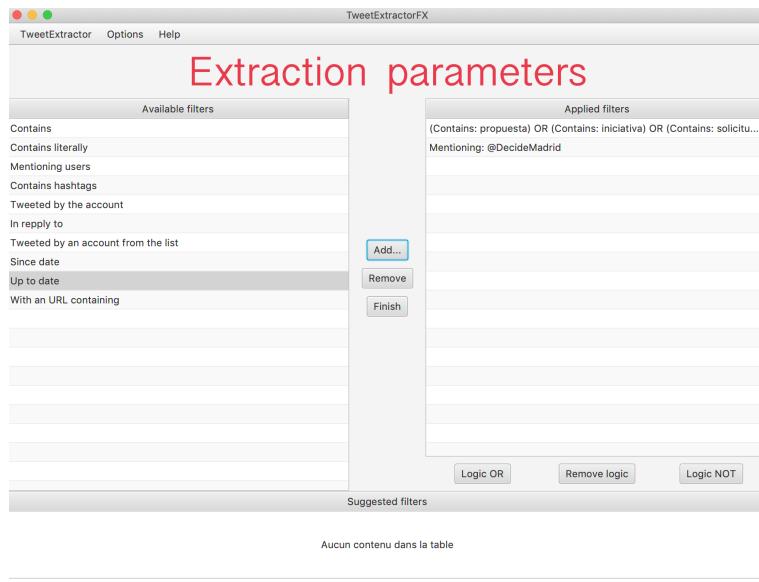
**Figura C.1:** Pantalla principal de bienvenida al abrir la aplicación

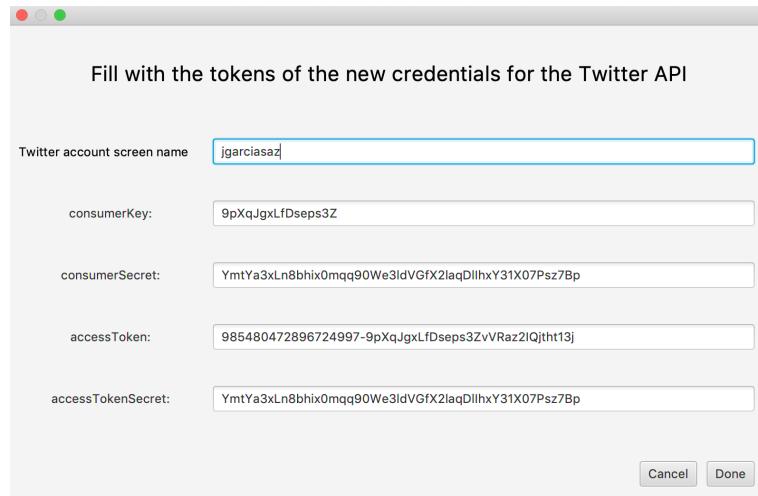
The image contains two side-by-side dialog boxes. The left dialog is titled "Log in" and has fields for "Username" (containing "jose") and "Password" (containing a masked password). It includes "Cancel", "Log in", and "New account" buttons. The right dialog is titled "New account" and has fields for "Username" (containing "newUser"), "Password" (containing a masked password), and "Repeat password" (containing a masked password). It includes "Cancel" and "Create account" buttons.

(a) Inicio de Sesión

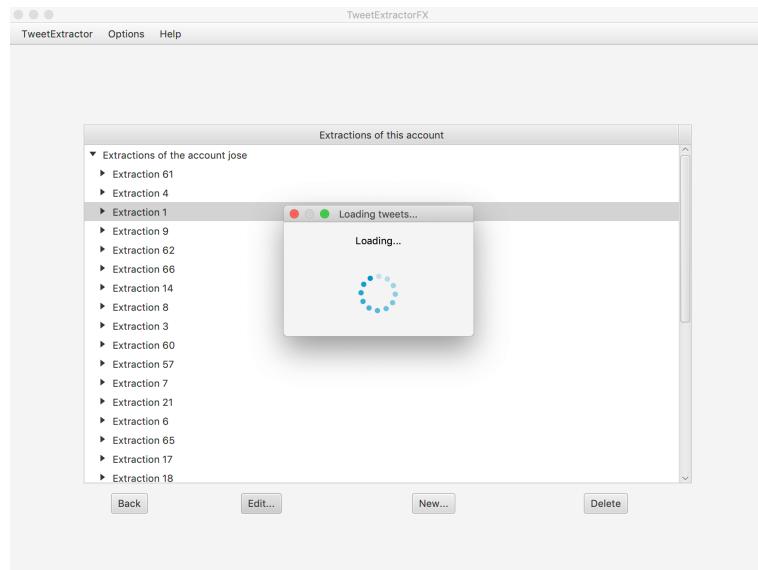
(b) Registro

**Figura C.2:** Diálogos de autentificación en la aplicación.

**Figura C.3:** Pantalla de inicio**Figura C.4:** Pantalla para configurar el perímetro de cada extracción. A la izquierda los filtros disponibles, a la derecha los añadidos.



**Figura C.5:** Configurando los credenciales de la API de Twitter (los credenciales mostrados en la foto no son reales)



**Figura C.6:** Las tareas costosas no bloquean la aplicación. Un nuevo hilo nos muestra un diálogo informativo sobre lo que se está haciendo.



# ÍNDICE TERMINOLÓGICO

---

`budgettitle`, 25

colores, 2

  predefinidos, 2

`eigenvalue`, 40

opciones, 47





