Máster Interuniversitario en Seguridad de las TIC

TFM

DLP-AIL (New local feeds addon)

Autor: Manuel Marrón Cascudo

Consultor: Jordi Guijarro Olivares

Profesor responsable: Victor García Font

2 de junio de 2019

FICHA DEL TRABAJO FINAL

Título del trabajo	DLP-AIL (New local feeds addon)
Nombre del autor	Manuel Marrón Cascudo
Nombre del consultor	Jordi Guijarro Olivares
Nombre del PRA	Victor García Font
Fecha de entrega	2 de junio de 2019
Titulación	Máster Interuniversitario en Seguridad de las TIC
Área del trabajo final	Hacking
Idioma del trabajo	Castellano
Palabras clave	

Resumen del trabajo

El *framework* AIL es una solución DLP desarrollada por el CIRCL para la monitorización de fugas de información en *pastes* publicados en pastebin.com o sitios similares. Al tratarse de un producto generado inicialmente como proyecto interno y publicado posteriormente, está orientada a un entorno limitado y sin posibilidad de utilizar la instancia en modo *multitenant*.
El presente trabajo cubre en primer lugar el funcionamiento del *framework* y sus características desde el proceso de instalación hasta las posibilidades de alimentación de la instancia para dar luego una solución a la limitación anterior consistente en un portal de gestión de términos adaptable a un entorno con múltiples organizaciones o compañías compartiendo la misma instancia. El portal desarrollado exige una autenticación previa de los usuarios que se incluye en la solución propuesta, y hace uso de una base de datos separada para almacenar las credenciales de forma segura y también los roles o permisos de acceso de cada usuario.

Abstract

AIL framework is a data leak prevention solution developed by the CIRCL to analize potential information leaks from pastebin sites like pastebin.com or others. As it was firstly developed as an internal project and later published, it is made for a limited enviroment and it is not intended to be used in a multitenant mode.
This work covers firstly the installation and feeding process of an instance of the framework AIL and then a solution is proposed for the limitation refered before. This solution is based in the deploy of an independent portal to manage the terms tracked by AIL, requiring a previous authentication to the users making use of a database to store the users credentials and the relationships and companies or organizations users belong to.

Índice

Índice de figuras

1. Introducción

1.1. Contexto

La transformación provocada con la irrupción primero de los ordenadores y luego de Internet y las redes de comunicaciones sobre el mundo actual, especialmente en el ámbito empresarial, ha provocado grandes beneficios en la automatización de tareas y el tratamiento de datos logrando una transformación parcial en muchos sectores e incluso total en otros. El impacto ha sido tal que ya se habla de la "Era de la información" o "Era digital". Actualmente la mayor parte de las comunicaciones se realizan utilizando medios informáticos, toda la información se almacena de manera digital y el acceso a la misma es cada vez más rápido, sencillo y directo. Este cambio de paradigma provoca un desafío a nivel de seguridad ya que más que nunca es importante mantener la integridad, la confidencialidad y la disponibilidad de la información que se trata en volúmenes y con un procesamiento ingentes.

Ante este desafío, uno de los problemas a los que se enfrenta cualquier entidad que almacene datos digitalmente es la pérdida de confidencialidad de los mismos, entendida esta como la capacidad de acceso a la información por parte de personas que no cuentan con la debida autorización y que por un motivo u otro no deberían conocerla. El enfoque tradicional ante esta problemática ha sido el de proteger la información de los accesos externos no autorizados, para lo cual se implantan múltiples medidas como la securización física y lógica de los entornos, la clasificación por niveles de criticidad con control de acceso, etc. Este enfoque, aunque necesario no es suficiente dado que no se contemplan las fugas de datos, intencionadas o no, generadas por las personas —o sistemas— con acceso legítimo. Las herramientas de prevención de pérdida de datos (DLP por sus siglas en inglés) se presentan como la solución para cubrir esta brecha y añadir una medida de seguridad adicional que permita aumentar el nivel de seguridad de la información almacenada.

1.2. Estado del arte

Una fecha clave en el ámbito de la seguridad informática es la aparición del gusano Morris el 2 de noviembre de 1988[1], no solo por el impacto no intencionado que este causó, sino porque motivó la creación del primer equipo de respuesta ante emergencias informáticas (CERT por sus siglas en inglés o CSIRT por su equivalente europeo) en Pittsburg por parte de la agencia estadounidense DARPA, el CERT/CC. En la actualidad existen múltiples CSIRTs constituidos por organizaciones supranacionales, estados, compañías del sector, estamentos militares o universidades y centros de investigación. Solo en España, la Agencia de la Unión Europea para la seguridad de la información y las comunicaciones reconoce 23 CSIRTs[2].

Los centros de respuesta ante emergencias informáticas o CERTs o su equivalente europeo CSIRT (Equipo de respuesta ante incidentes de seguridad informática) tienen como principal objetivo el dar una respuesta organizada y estructurada a los diversos incidentes de seguridad informática que puedan ir surgiendo, aunque también realizan tareas de divulgación y prevención. La utilización de herramientas DLP se enmarca dentro de las labores proactivas orientadas a la búsqueda de posibles brechas o fugas de información que realizan estos organismos para poder actuar con la mayor celeridad posible ante ellas.

Como herramienta DLP, el *framework* AIL es un sistema modular para analizar potenciales fugas de información de fuentes de datos no estructuradas. Surge en 2014 como un proyecto interno del Centro de Respuesta ante Incidentes Informáticos de Luxemburgo (CIRCL) para evaluar si era factible automatizar el análisis de información estructurada y no estructurada para detectar fugas de

[1] Información sobre el gusano Morris disponible en Wikipedia [20].
[2] Fuente: Página web de la ENISA [2].

información. La versión inicial fue publicada en mayo de 2018, siendo la versión 1.4 de octubre de 2018 la última versión disponible en el momento de elaboración de este documento. El *framework* está desarrollado en Python, publicado con licencia Open Source y continúa siendo mantenido por el CIRCL.

1.3. Objetivos y metodología

El objetivo que se persigue con la realización del presente trabajo es doble; por un lado realizar una aproximación al *framework* AIL para conocer su funcionamiento y capacidades, y por otro lado cubrir una necesidad existente en el Equipo de Respuesta ante Incidentes de la Anella Científica (CSUC-CSIRT) en el uso de su propia instancia de AIL.

El primer objetivo se centra en que el alumno profundice en el funcionamiento del *framework* AIL mediante el despliegue de una instancia propia haciendo uso del código fuente y el método de instalación automatizado que desde el CIRCL han publicado en la página web de github.com [3].

El segundo objetivo que busca el trabajo es dar solución a una necesidad existente en el CSUC-CSIRT en relación con la instancia AIL que utilizan. Al tratarse de un consorcio que está integrado por múltiples entidades, cada una de estas tiene unos datos propios que desea proteger y que le gustaría incluir en la detección del *software* DLP. El método actual se basa en la notificación por parte de los responsables de cada entidad de las expresiones o cadenas que se pretenden detectar al responsable del DLP dentro del Equipo de Respuesta ante Incidentes, quien introduce los datos manualmente en la herramienta. Se pretende con este trabajo proporcionar un nuevo método de introducción de los conceptos o datos a detectar en la herramienta que permita agilizar y automatizar esta tarea.

El nuevo módulo debe estar integrado en el *framework* de AIL y debe realizarse algún tipo de autenticación previa a la introducción de los datos con el fin de securizarlo y evitar accesos indebidos y ataques potenciales. El nuevo método de introducción de datos debe también permitir la gestión de los mismos de manera que se puedan consultar y modificar los datos que actualmente están siendo monitorizados.

1.4. Tareas a realizar y planificación temporal

El desarrollo del proyecto actual implica la ejecución de una serie de tareas o fases con la idea de lograr cumplir los objetivos propuestos en los plazos adecuados. Para abordar con garantías el proyecto y obtener los resultados esperados se establecen tres fases en base a las cuales se determinará la planificación temporal del proyecto. Mientras que la primera fase está orientada al análisis y profundización en la herramienta por parte del alumno, la segunda y tercera fases se relacionan con el otro objetivo de buscar una solución a la problemática de introducción de datos a monitorizar en la herramienta.

La primera fase consiste en el despliegue y personalización de una instancia del *framework* AIL en base a los criterios que previamente se definirán en función del contexto que el alumno decida establecer. Esta fase permitirá cumplir el primero de los dos objetivos propuestos y detallados en el apartado 1.3, y su duración se estima en cuatro semanas.

La segunda fase consiste en un análisis detallado de los métodos por los que se pueden introducir en la herramienta los términos a monitorizar y el diseño a alto nivel del nuevo módulo describiendo la forma de introducción de los datos y especialmente la integración con el *framework*. Se establece una planificación de dos semanas para la realización de esta fase debido a que está muy influenciada por las fases anterior —en la parte del análisis— y posterior —en la parte del diseño—, pese a lo cual la importancia de esta fase es capital en el desarrollo del proyecto ya que orientará la solución al problema que motiva este TFM y detallado en el segundo objetivo descrito en el apartado 1.3.

La tercera fase consiste en el desarrollo propiamente dicho del nuevo módulo diseñado en la fase anterior y la realización de las pruebas pertinentes para valorar la eficacia de la solución propuesta y el cumplimiento de los requisitos impuestos en el apartado 1.3. Dado que la solución proporciona un nuevo método de introducción de datos a la herramienta se analizarán también las vulnerabilidades que la nueva solución pudiese introducir al *framework* como consecuencia de la ampliación de la superficie de ataque del mismo. Se estima que el desarrollo, realización de pruebas y análisis de vulnerabilidades requerirán entre cuatro y seis semanas de dedicación con el fin de obtener los resultados esperados.

1.5. Sumario de productos obtenidos

El desarrollo realizado para cumplir el segundo de los objetivos propuestos de cubrir la necesidad de gestión de un entorno *multi-tenant* en la instancia de AIL del CSUC-CSIRT genera una serie de archivos. A continuación se listan únicamente los archivos necesarios para realizar la instalación y configuración del *add-on*. Tal y como se describe en el apartado 5, durante el propio proceso de instalación se generan múltiples archivos necesarios para el funcionamiento de la solución propuesta pero no se incluyen al ser producto de los aquí listados.

- ***README.md***. Fichero con la descripción del contenido del repositorio en `github.com`.
- ***identityManagement.py***. Clases de Python para la gestión de la base de datos de usuarios.
- ***initializeDatabase.py***. Script de inicialización de la base de datos de usuarios utilizado durante el proceso de instalación.
- ***managementPortal.PNG***. Imagen del portal de gestión referenciada en el fichero *README.md*.
- ***managementPortal.py***. Script del portal de gestión.
- ***previousConfigurations.sh***. Script de instalación.
- ***requirements.txt***. Fichero que contiene las extensiones de Python a instalar.
- ***login.html***. Fichero HTML de la página de autenticación de usuarios previa al acceso al portal de gestión.
- ***mgmtPortal.html***. Fichero HTML de la página del portal de gestión.
- Contenido de la carpeta ***static***. Incluye todos los ficheros que dan formato a las páginas de inicio y del portal de gestión.
- ***createUsers.py***. Script de creación de usuarios.
- ***createCompanies.py***. Script de creación de compañías.
- ***createRelationships.py***. Script de creación de relaciones entre usuarios y compañías.
- ***HOWTO.md***. Fichero de explicación del uso de los scripts de gestión de la base de datos de usuarios.

1.6. Breve descripción de los otros capítulos de la memoria

En el apartado 2 se realiza una descripción del *framework* AIL, sus características técnicas, el proceso de instalación y las opciones existentes para la alimentación de la instancia, con especial detalle a la posibilidad de uso de la herramienta *pystemon* para alimentar AIL; mientras que el apartado 3 hace un recorrido por todos los menús de AIL describiendo su funcionamiento y detalles.

Por su parte, los apartados 4 y 5 detallan la solución propuesta para resolver la necesidad planteada por el CSUC-CSIRT. El primero describe la necesidad, los requisitos y los detalles técnicos de la solución mientras que en el segundo se detalla el proceso de generación del *add-on* adaptable a cualquier instancia de AIL.

Al final de la memoria se incluye el anexo A en el que se muestran todos los archivos generados.

2. *Framework* AIL. Introducción, características y proceso de instalación

La apuesta que realizó el Centro de Respuesta ante Incidentes Informáticos de Luxemburgo por el desarrollo del *framework* AIL surge en 2014 como un proyecto interno a partir de una necesidad que detectan en su trabajo diario. Ya en ese año como CSIRT tenían que tratar frecuentemente con fugas de información, tanto para la alerta a las compañías u organizaciones que las sufren como para el análisis para los medios de comunicación con fines de prevención y educación o el análisis y detección de *malware*, vulnerabilidades *software* o exfiltración de datos[3].

El proyecto publicado en 2018 es un *framework* para el tratamiento de información de fuentes de datos no estructuradas con el objetivo de detectar posibles fugas. Aunque permite la importación de información por otros medios, está principalmente orientado al tratamiento de *pastes* de sitios como pastebin.com. Este tipo de sitios web denominados *pastebin*[4] están desarrollados para el almacenamiento de texto y se utilizan típicamente para compartir trozos de código fuente o salidas de programas en texto plano entre desarrolladores. Sin embargo en los últimos años la funcionalidad de este tipo de sitios ha provocado que muchos cibercriminales los utilicen para almacenar *exploits*, volcados de base de datos comprometidas u otros listados de información de sitios vulnerables. La cantidad y tamaño de los *pastes* hace necesario un tratamiento automatizado de los mismos a fin de poder abarcar la totalidad de los *pastes* y buscar en ellos patrones complejos.

2.1. Funcionalidades del *framework* AIL

Los múltiples módulos que componen el *framework* AIL permiten tratar los datos en búsqueda de fugas de credenciales, datos de tarjetas de crédito, CVEs, direcciones de correo o teléfonos entre otros. La figura 1 muestra una de las pantallas de AIL (en concreto la pestaña "Browse important pastes") en la que se pueden observar los múltiples resultados que se obtienen a partir del procesamiento de los distintos módulos. Además, la modularidad y sencillez del diseño hace que según sus desarrolladores[5] se pueda añadir un nuevo módulo de análisis en unas 50 líneas de código.

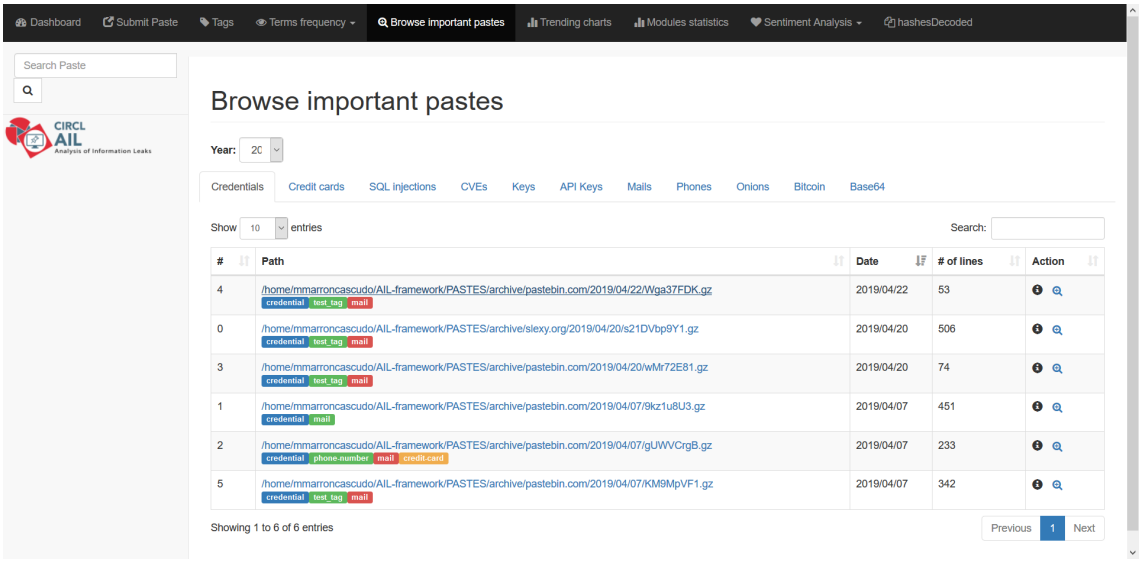


Figura 1: Pestaña "Browse important pastes". Permite consultar los *pastes* que los diversos módulos han etiquetado.

[3] Definición del concepto exfiltración de datos en Wikipedia: [19]
[4] Definición de los sitios web de tipo *pastebin*: [21]
[5] Así se indica en las presentaciones disponibles en en Github de AIL [3].

Entre las fortalezas de AIL destaca también que soporta el uso de múltiples procesadores con la capacidad de arrancar un módulo de análisis múltiples veces para mejorar el rendimiento en momentos de alta demanda; y la recepción de datos de múltiples fuentes de forma concurrente tanto desde la interfaz web (en la pestaña de "Submit paste") como desde consola con el script "import_dir.py" o alimentando la cola ZMQ desde una instancia de Pystemon como se detalla en el apartado 2.4.1.

Como también se puede apreciar en la imagen 1 permite el etiquetado y la customización del mismo, es posible añadir términos, conjuntos e incluso expresiones *regex*[6] como objetivos de las búsquedas y permite una integración con las plataformas de compartición de amenazas y respuesta ante incidentes MISP[7] y TheHive[8].

2.2. Características técnicas del *framework* AIL

El *framework* de AIL está programado íntegramente en la versión 3 de **Python**, apoyándose concretamente en el *microframework* web **Flask**[9] para el despliegue de la interfaz web. En el *back end* utiliza **Ardb**, una base de datos "nosql" totalmente compatible con el protocolo y los comandos de Redis y no limitada al tamaño de la memoria RAM como sí lo está Redis[10]. Por último, además de la implementación de Redis del paradigma de publicación y suscripción de mensajes que se utiliza para la comunicación de los datos entre los módulos, se utiliza también una cola **ZeroMQ**[11] para la alimentación de la interfaz. La vista global de esta arquitectura se presenta en la imagen 2.

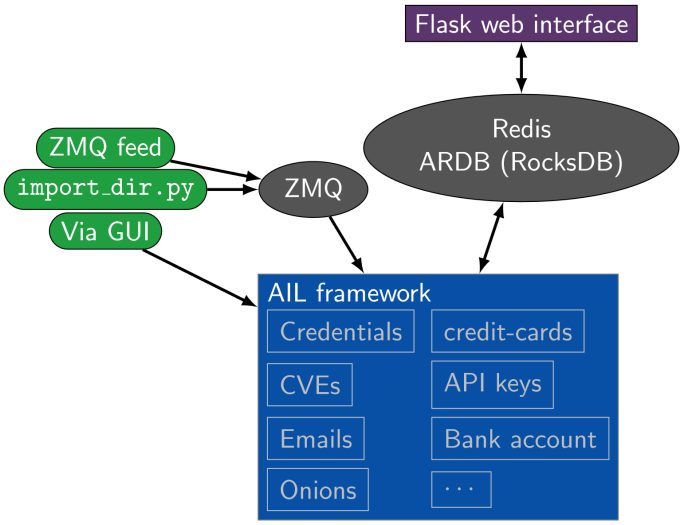


Figura 2: Arquitectura del *framework* AIL.
Fuente: Presentación "ail-training-december-2018.pdf" publicada en el Github de AIL [3], diapositiva 29.

2.3. Proceso de instalación

El proceso de instalación del *framework* AIL, que viene completamente detallado en el fichero README.md disponible en el repositorio Github de AIL [3], consiste en la descarga desde el repositorio del contenido necesario y la ejecución de tres scripts que realizan todas las instalaciones y configuraciones necesarias.

[6] Definición de expresiones *regex*: [17].
[7] Página en Github de MISP: [5]
[8] Página web de TheHive: [12]
[9] Definición de Flask: [15]
[10] Enlace a la página de github de Ardb: [4]
[11] Definición de ZeroMQ: [18]

```
git clone https://github.com/CIRCL/AIL-framework.git
cd AIL-framework
./installing_deps.sh
cd ~/AIL-framework/
. ./AILENV/bin/activate
cd bin/
./LAUNCH.sh
```

En primer lugar se ejecuta el script *"installing_deps.sh"*, en el cual se instalan todas las dependencias de python y otros paquetes y se descargan e instalan si no lo están ya Redis, Faup, tlsh y ardb. A continuación configura la variable de entorno propia y ejecuta el script *"update_thirdparty.sh"* que se encuentra en la ruta *var/www/*. Por último instala otras dependencias como el BGP Ranking y ejecuta varias instalaciones mediante Python.

Una vez completado el primer script se activa en entorno virtual con la llamada al fichero *"activate"* ubicado en *AILENV/bin/*, y con el entorno montado y las variables configuradas se hace la llamada al último script denominado *"LAUNCH.sh"* que permite controlar todos los procesos necesarios para arrancar la instancia de AIL. Este puede ejecutarse con el parámetro *"-l"*(o *"--launchAuto"*) de forma que arranca automáticamente todos los módulos necesarios, o puede ejecutarse sin parámetros mostrando por pantalla el resultado que se aprecia en la imagen 3. En ese caso se deben seleccionar las opciones que se quiere lanzar y pulsar ENTER para comenzar la ejecución, de forma que para concluir el proceso de instalación se deberían marcar las 6 primeras opciones.

2.4. Alimentación de la instancia

El objetivo del *framework* AIL es el tratamiento de datos con el objetivo de detectar fugas de información con el uso de distintos módulos como se presentan en el apartado 2.2; y para ello es necesario alimentar la instancia con los datos que se quieren analizar. El *framework* presenta diversas maneras de alimentar la instancia, aunque su desarrollo está orientado principalmente a la alimentación automática con *pystemon*.

- Mediante un script ubicado en la ruta *bin/import_dir.py* en el directorio de instalación de AIL.
- Gráficamente utilizando la pestaña "Submit Paste" de la web de la instancia, como se puede observar en la imagen 4.
- El CIRCL, como desarrollador del *framework* ofrece a sus colaboradores un feed propio que envían a la IP del solicitante tal y como indican en el documento *"HOWTO.md"* del Github.
- Automáticamente a partir de una instancia de *pystemon* utilizando el script *bin/feeder/pystemon-feeder.py*. Este proceso se detalla en el subapartado 2.4.1.

2.4.1. Pystemon

AIL es una herramienta destinada al análisis de información y por tanto no está diseñada para la recolección. Sin embargo, el potencial de AIL reside en poder analizar la mayor cantidad de fuentes de forma automatizada para lograr así un tratamiento y detección de un volumen mayor de datos. La solución consiste en utilizar otra herramienta como *pystemon* para realizar la recolección de *pastes* y mediante el script de *"feeding"* ubicado en *bin/feeder/pystemon-feeder.py* alimentar automáticamente la instancia de AIL.

Pystemon es una herramienta desarrollada por el CIRCL para la monitorización de sitios de tipo *pastebin* escrita en Python y publicada al igual que AIL en el Github del CIRCL [6]. Está desarrollada

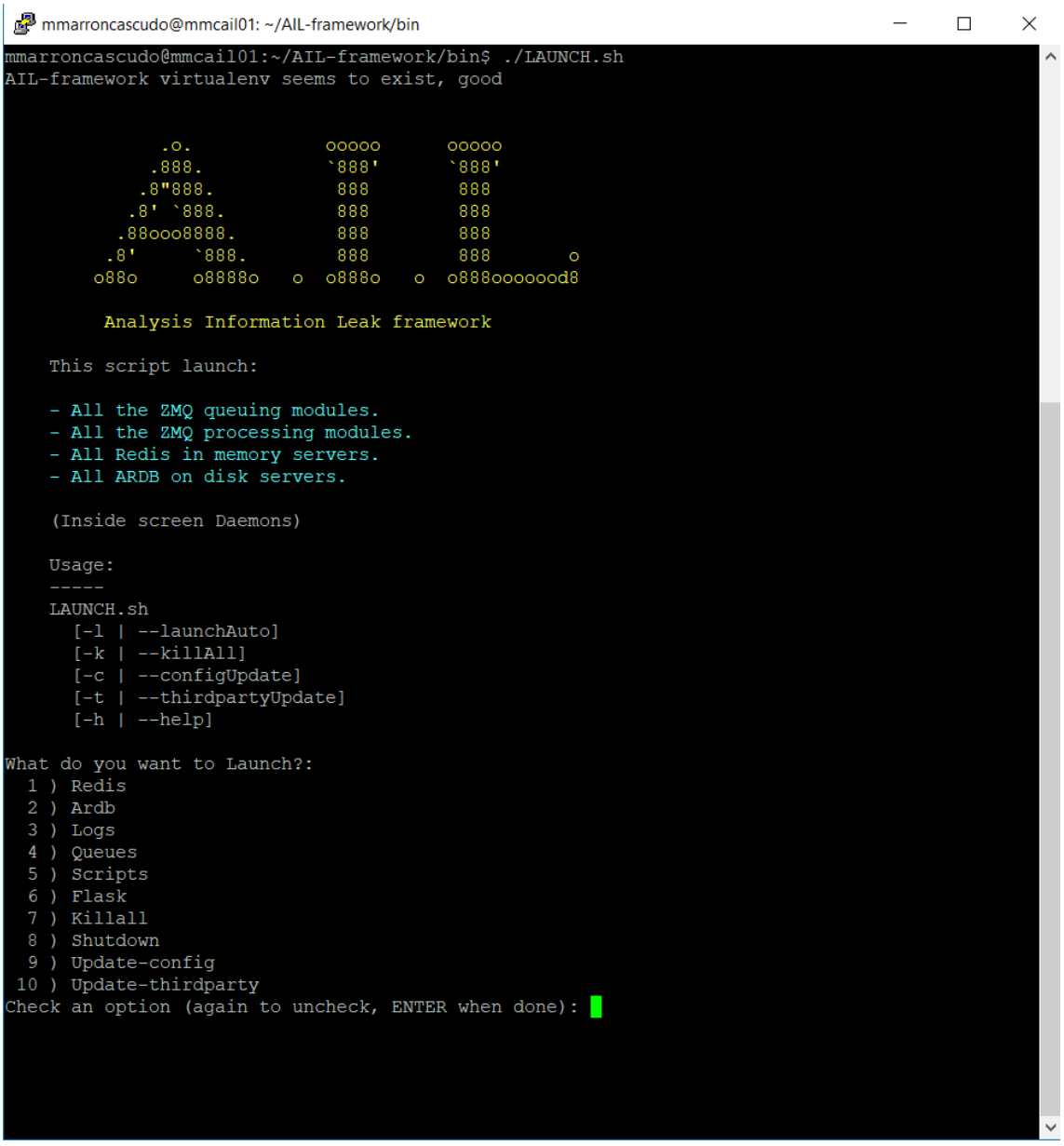


Figura 3: Salida del script *"LAUNCH.sh"* ejecutado sin parámetros.

en Python y no requiere ninguna instalación adicional sino que una vez descargada es suficiente con adaptar la configuración del fichero *pystemon.yaml* y ejecutar el script *pystemon.py*.

Para lograr integrar los datos recogidos por *pystemon* en AIL es necesario modificar la configuración estableciendo los siguientes parámetros del citado fichero YAML:

```
archive:
  save-all: yes # Keep a copy of all pasties

redis:
  queue: yes # Toggle PUSH to redis queue
  queue-all: yes # Keep a copy of all pasties
  server: "localhost"
  port: 6379
  database: 10
```

Y una vez arrancado *pystemon* se debe ejecutar el script *pystemon-feeder.py* habiendo revisado

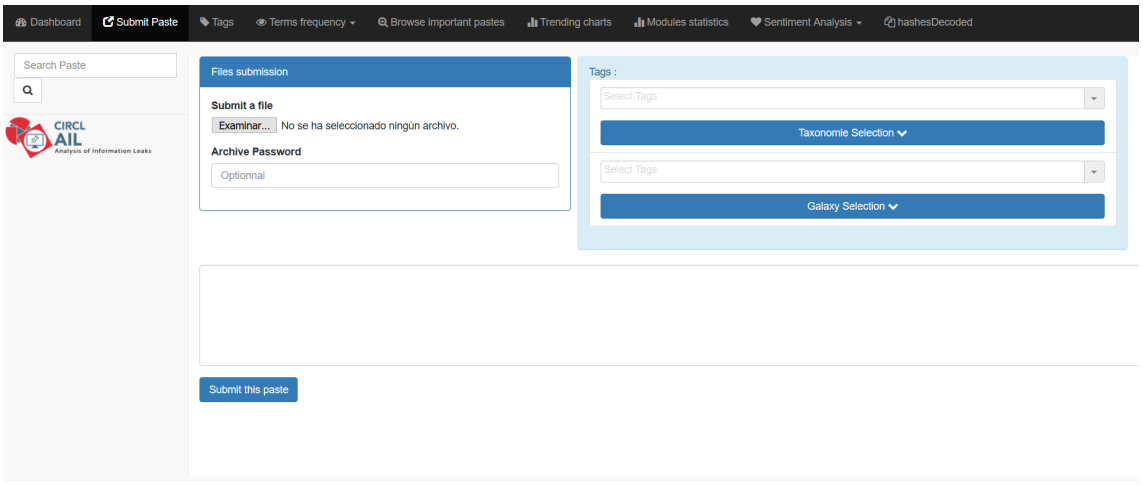


Figura 4: Pestaña "Submit Paste" que permite la alimentación de la instancia manualmente de forma gráfica.

previamente el valor de *"pystemonpath"* definido en el fichero de configuración de AIL ubicado en *bin/packages/config.cfg*. Las figuras 5 y 6 muestran la salida de pystemon y del script que alimenta AIL cuando ambas se encuentran en funcionamiento.

```
[2019-05-13 11:19:24,835] Downloading list of new pastes from codepad.org. Will check again in 18 se
conds
[2019-05-13 11:19:25,248] Found 1 new pasties for site codepad.org. There are now 1 pasties to be do
wnloaded.
[2019-05-13 11:19:31,375] Downloading list of new pastes from slexy.org. Will check again in 18 seco
nds
[2019-05-13 11:19:43,270] Downloading list of new pastes from codepad.org. Will check again in 21 se
conds
[2019-05-13 11:19:50,102] Downloading list of new pastes from slexy.org. Will check again in 27 seco
nds
[2019-05-13 11:20:03,724] Downloading list of new pastes from pastebin.com. Will check again in 50 s
econds
[2019-05-13 11:20:04,125] Found 13 new pasties for site pastebin.com. There are now 13 pasties to be
 downloaded.
[2019-05-13 11:20:04,657] Downloading list of new pastes from codepad.org. Will check again in 25 se
conds
[2019-05-13 11:20:05,096] Found 1 new pasties for site codepad.org. There are now 1 pasties to be do
wnloaded.
[2019-05-13 11:20:17,858] Downloading list of new pastes from slexy.org. Will check again in 19 seco
nds
[2019-05-13 11:20:30,118] Downloading list of new pastes from codepad.org. Will check again in 17 se
conds
[2019-05-13 11:20:30,566] Found 1 new pasties for site codepad.org. There are now 1 pasties to be do
wnloaded.
[2019-05-13 11:20:38,129] Downloading list of new pastes from slexy.org. Will check again in 14 seco
nds
[2019-05-13 11:20:47,598] Downloading list of new pastes from codepad.org. Will check again in 19 se
conds
[2019-05-13 11:20:48,579] Found 2 new pasties for site codepad.org. There are now 2 pasties to be do
wnloaded.
[2019-05-13 11:20:53,065] Downloading list of new pastes from slexy.org. Will check again in 18 seco
nds
[2019-05-13 11:20:54,177] Downloading list of new pastes from pastebin.com. Will check again in 46 s
econds
[2019-05-13 11:20:54,543] Found 13 new pasties for site pastebin.com. There are now 13 pasties to be
 downloaded.
```

Figura 5: Salida correspondiente a un instante de la ejecución de *pystemon*.

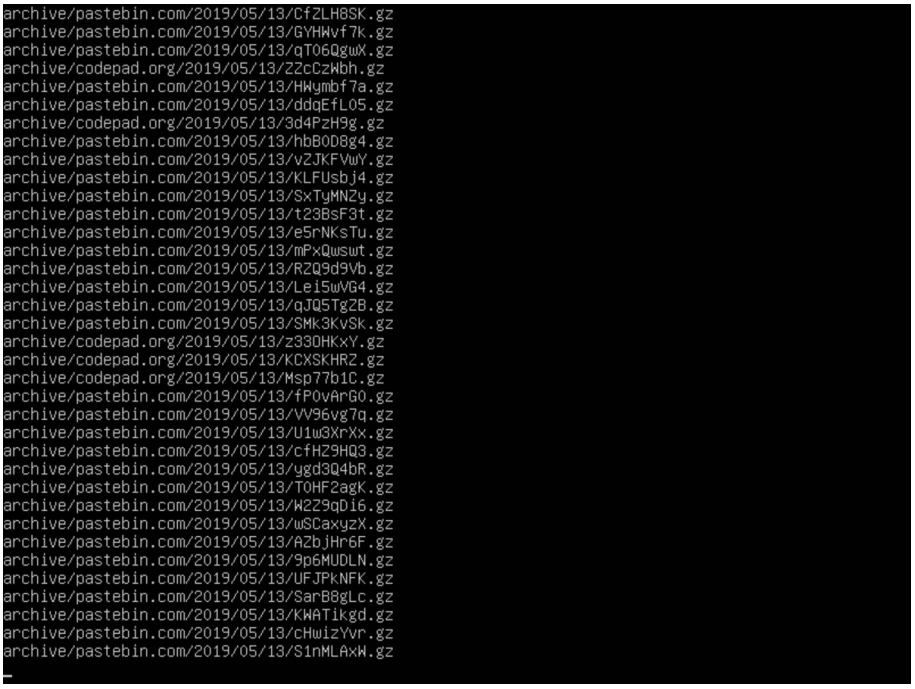


Figura 6: Salida del script “pystemon-feeder.py” alimentando la instancia de AIL con los datos obtenidos de *pystemon*.

3. *Framework* AIL. Menús y funcionalidades

3.1. *Dashboard*

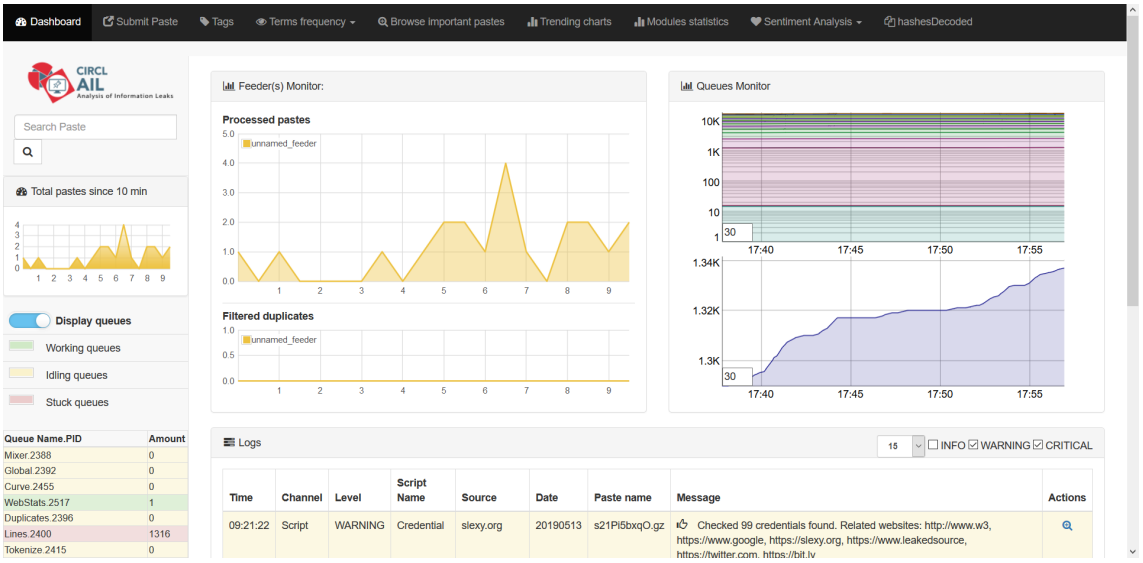


Figura 7: Página inicial de la interfaz web de AIL. Pestaña *"Dashboard"*.

La página principal que se muestra al abrir la interfaz web de AIL corresponde a un *dashboard* en el que se resume en varias gráficas el estado de funcionamiento de la instancia. Como se puede apreciar en la imagen 7 la parte central de la imagen muestra una gráfica con los *pastes* procesados en los últimos 10 minutos y en caso de haberlos, los filtrados por estar duplicados. La parte derecha muestra el estado de las diferentes colas que realizan las tareas de análisis de los datos procesados, mientras que la parte inferior presenta una muestra de los últimos logs de la aplicación permitiendo el filtrado de los más relevantes a criterio del usuario. Por último la columna de la izquierda muestra, además del logo de AIL que es personalizable por la compañía u organización que despliega la instancia, una pequeña gráfica resumen equivalente a la central con los *pastes* procesados en los últimos 10 minutos, y el estado de las diferentes colas de procesamiento.

3.2. *Submit Paste*

El apartado 2.4 detalla los diversos métodos existentes para alimentar la instancia de AIL, entre los cuales se encuentra la pestaña *"Submit Paste"* de la interfaz web. La imagen 4 muestra el contenido de la misma, desde la cual se puede subir un fichero y seleccionar también manualmente las etiquetas que se le asociarán.

3.3. *Tags*

La pestaña de *"Tags"*, que se muestra en la imagen 8, permite realizar búsquedas por etiquetas. Además desde esta pantalla se pueden editar la lista de "taxonomías" y de "galaxias" que vienen prestablecidas en la instancia. Se trata de módulos que vienen precargados y al activarlos cargan una serie de etiquetas de un determinado ámbito. Por último se pueden escoger las distintas etiquetas que se transferirán a los servicios "MISP" y "The Hive" en caso de estar configurados.

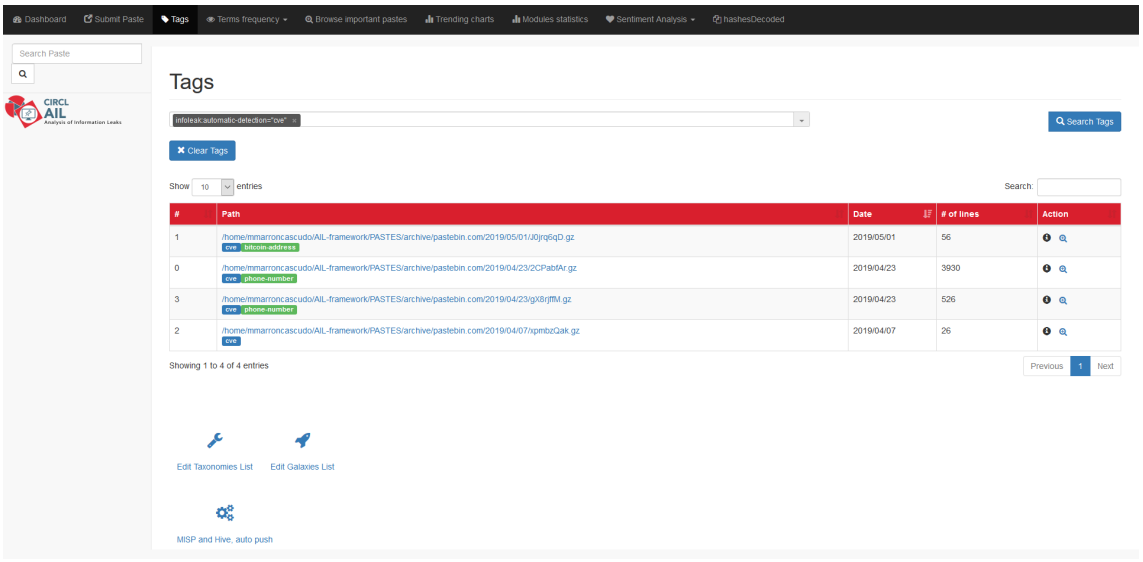


Figura 8: Interfaz web de AIL. Pestaña *"Tags"*.

3.4. *Terms frecuency*

Dentro de la sección de *"Terms frecuency"* se encuadran varias subsecciones. En primer lugar la pestaña *"Terms management"* que se muestra en la imagen 9 y permite gestionar los términos a monitorizar estableciendo etiquetas y correos electrónicos de notificación personalizados asociados a términos individuales, conjuntos de términos y expresiones *regex*; y en ella se puede también gestionar la lista negra de términos.

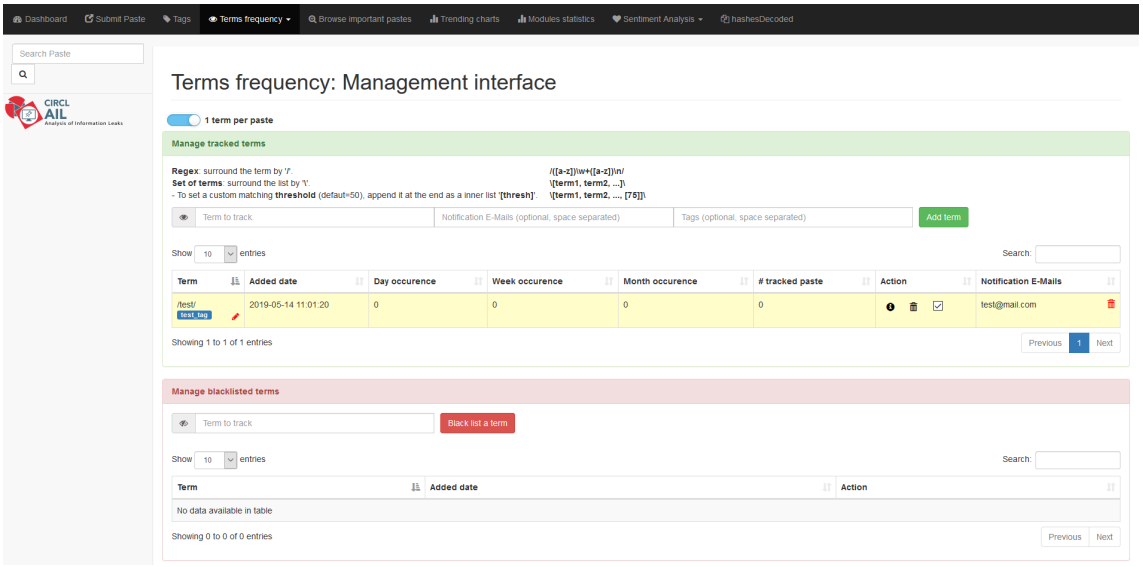


Figura 9: Interfaz web de AIL. Pestaña *"Terms management"* del apartado *"Terms frecuency"*.

La pestaña *"Credential seeker"* permite buscar credenciales que hayan sido publicadas en alguno de los *pastes* analizados. La imagen 10 muestra el resultado para la búsqueda de una típica contraseña[12] y como se puede apreciar se obtienen resultados tanto exactos como similares. Esto no quiere decir que se trate de una contraseña sino que únicamente está mencionado en un *paste*, pero analizando los *pastes* implicados se puede comprobar que varios efectivamente contienen listados de

[12]Contraseñas más utilizadas en 2018: https://cincodias.elpais.com/cincodias/2018/12/13/lifestyle/1544718914_855465.html

correos y contraseñas como la buscada.

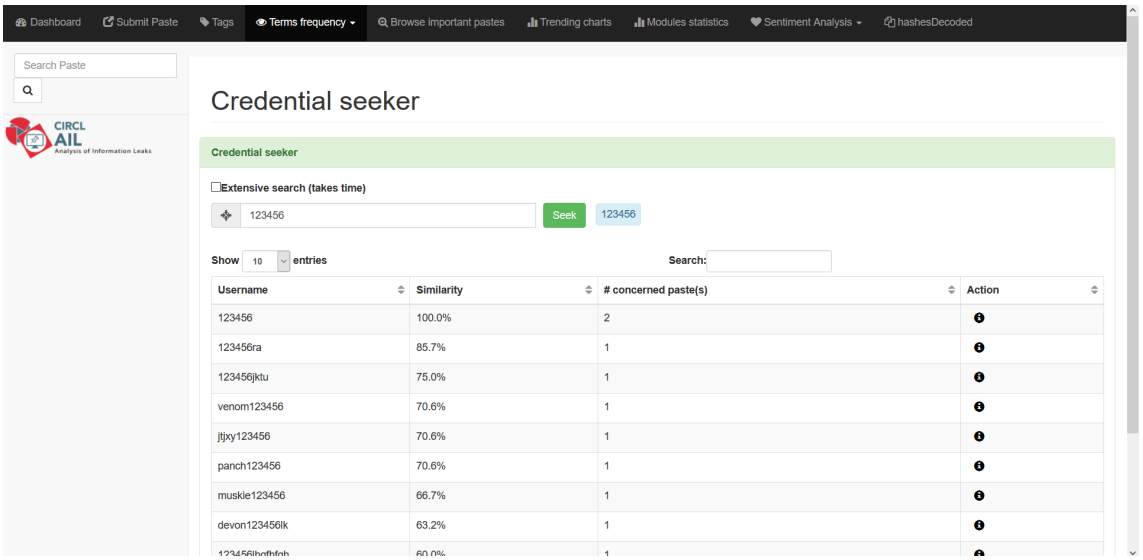


Figura 10: Interfaz web de AIL. Pestaña *"Credential Seeker"* del apartado *"Terms frecuency"*.

Por último las pestañas *"Terms plot top"* y *"Terms plot tool"* generan distintas gráficas sobre los términos más detectados en distintos periodos temporales y ofrecen al usuario la posibilidad de generar gráficas customizadas.

3.5. *Browse important pastes*

La pestaña *"Browse important pastes"* que ya se presenta en la imagen 1 permite al igual que la pestaña *"Tags"* analizar *pastes* asociados a una determinada etiqueta, pero en este caso únicamente con determinados filtros más relevantes como credenciales, tarjetas de crédito o teléfonos.

3.6. *Trending charts*

La pestaña *"Trending charts"* muestra diferentes gráficos en los que se pueden analizar distintas tendencias sobre dominios, protocools o palabras en los últimos días.

3.7. *Modules statistics*

"Modules statistics" presenta gráficos con los dominios más presentes en fugas de credenciales y correos además de unos gráficos sobre los proveedores de *pastes* más relevantes; tal y como se puede observar en las imágenes 11 y 12.

3.8. *Sentimental analysis*

La pestaña *"Sentimental analysis"* muestra y permite generar gráficas sobre el estado de los distintos proveedores de *pastes* que tiene la instancia.

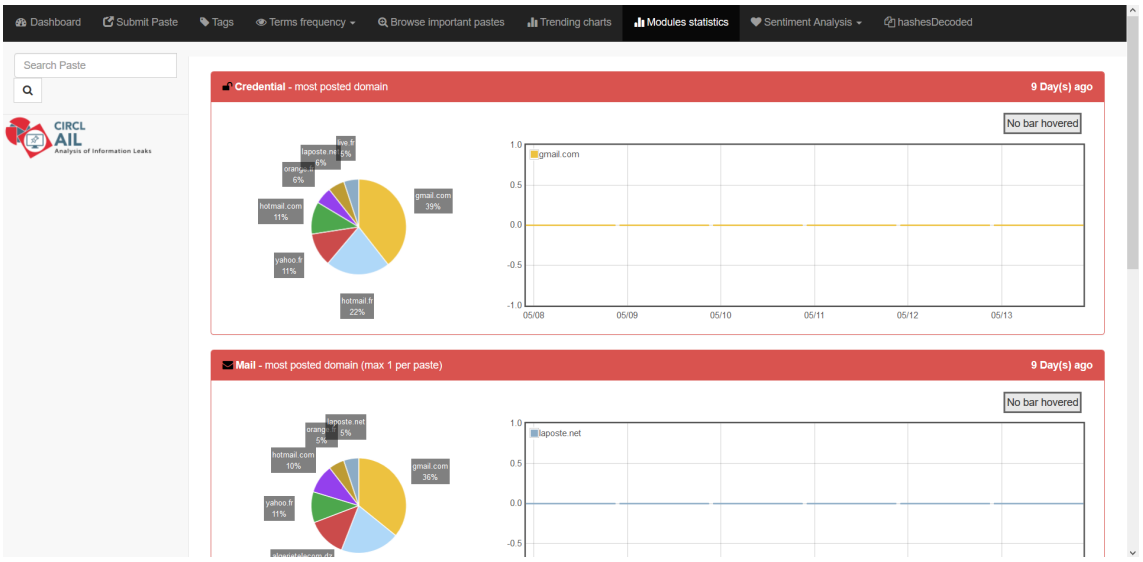


Figura 11: Interfaz web de AIL. Pestaña *"Modules statistics"*.

Figura 12: Interfaz web de AIL. Gráficos sobre los proveedores de *pastes* de la pestaña *"Modules statistics"*.

3.9. *Hashes decoded*

La última pestaña de *"Hashes decoded"* muestra gráficos relativos a *hashes* de ficheros en caso de que se hayan detectado, y detalles como el tipo de *hash*, el tipo de fichero y las ocasiones en que ha sido detectado.

4. Adaptación de AIL a un entorno multitenant

4.1. Necesidad originaria y limitación del *framework* AIL

El *framework* AIL está publicado en github como *software* libre con licencia "GNU Affero General Public License", de manera que cualquier organización puede descargarse e implementar en sus sistemas una instancia de AIL. Sin embargo al ser un producto desarrollado por el CIRCL en base a una necesidad propia es posible que ciertas organizaciones encuentren algunas limitaciones a la hora de adaptarlo a su entorno y jerarquías. Este es el caso del CSUC-CSIRT, el Equipo de Respuesta a Incidentes de la Anella Científica.

El CSUC-CSIRT dispone de una instancia de AIL propia que utilizan para monitorizar y tratar de detectar cualquier fuga de información relativa al consorcio y también a las universidades y entidades que lo componen; y por este motivo la cantidad de términos que requieren monitorizar es muy amplia y diversa. La manera que proporciona AIL para la gestión de los términos a monitorizar es la pestaña *"Terms management"* de la interfaz web que se presenta en el apartado 3.4. Sin embargo desde esa página se pueden consultar todos los términos que se están monitorizando, con lo cual no es posible abrir esta página al acceso por parte de las diferentes entidades ya que supondría que estas tendrían acceso a información del resto a la que no deberían tener acceso.

Por ello se hace necesario adaptar de alguna manera el *framework* a un entorno *multitenant* como este en que las diferentes organizaciones o compañías puedan gestionar únicamente sus términos de forma independiente y sin requerir un administrador global de la instancia que organice y lleve a cabo las modificaciones.

4.2. Requisitos funcionales de la mejora

La mejora a desarrollar debe cumplir una serie de requisitos para ser apta y adaptable a cualquier entorno *multitenant*.

- Se requiere una página que permita a cada organización gestionar los términos que desean monitorizar, de forma que puedan añadir, consultar y borrar los términos, las etiquetas personalizadas que desean introducir y también los correos electrónicos de notificación asociados a cada término.
- La página debe ser independiente de la interfaz web de AIL para que se pueda garantizar el acceso a la misma sin tener que permitir el acceso a toda la instancia en las reglas de seguridad perimetral de la organización.
- Dado que la página mostrará una información u otra en función del usuario, se requiere una autenticación previa que controle el acceso y los datos que se deben mostrar.
- La autenticación debe realizarse de un modo seguro y sin utilizar contraseñas en texto plano.
- Al tratarse de una página de gestión de términos, ciertos aspectos que también cubre la pestaña *"Terms management"* como la posibilidad de consultar los *pastes* asociados no serán incluidos.
- Del mismo modo, la selección del número de términos por *paste* a tener en cuenta o la gestión de la *blacklist*, por ser elementos que afectan a toda la instancia deben ser excluidos de la página de gestión.
- La *blacklist* sí debe ser mostrada dado que contiene información relevante para la gestión de términos, pero únicamente a modo de consulta y sin posibilidad de modificación.

- Los términos monitorizados por varias compañías solo se borrarán cuando los usuarios de todas las compañías los borren.

4.3. Resumen funcional de la solución propuesta

La solución desarrollada para cumplir con los objetivos detallados en el apartado 4.2 se basa en una nueva interfaz web completamente independiente de la interfaz web de AIL a la cual se accede en otro puerto diferente. Por defecto se carga la pantalla de *login* en la que se solicita un usuario y contraseña como se puede apreciar en la imagen 13.

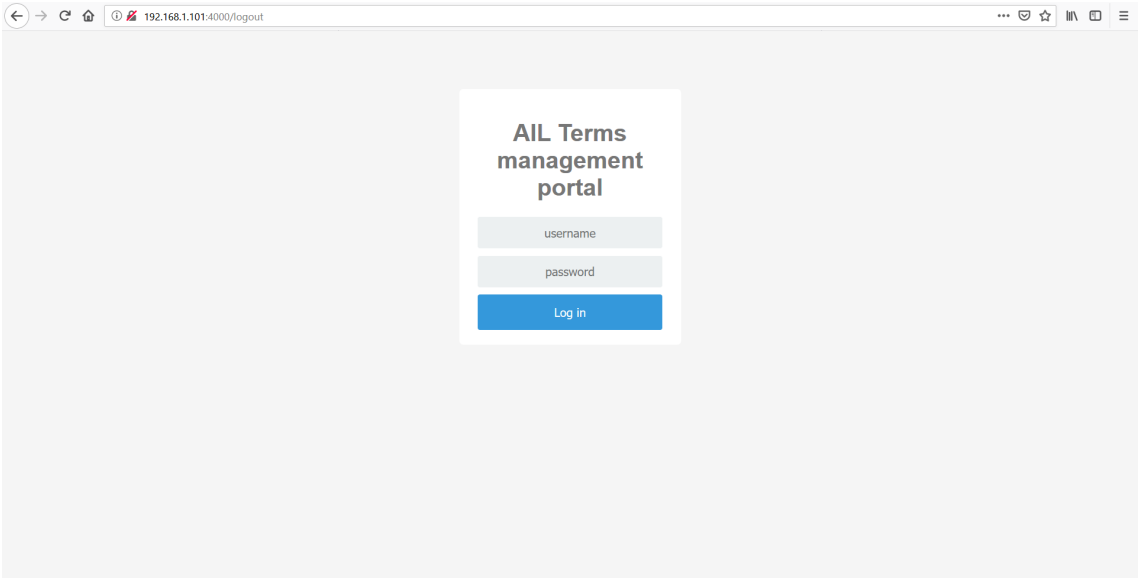


Figura 13: Página de *login* del portal de gestión de términos.

Una vez el usuario se autentica se muestra una pantalla como la que se puede observar en 14. En la parte superior de la misma se puede consultar el usuario logueado, las compañías a las que pertenece y el botón de cierre de sesión o *logout*. A continuación se muestra la sección de gestión de términos a monitorizar, en la que el usuario únicamente puede ver los términos relacionados con compañías a las que pertenece; y por último se muestra la lista de términos en la *blacklist*.

A diferencia de la pestaña *"Terms management"* de AIL, en el portal de gestión no aparece en la columna de *"Actions"* el icono con la *"i"* relativo a *"Show concerned paste(s)"*, dado que como se explica en el apartado 4.2 se ha ideado como un portal exclusivamente para la gestión de los términos. Además, en la sección correspondiente a la *blacklist* tampoco aparecen ni la posibilidad de introducir nuevos términos ni la posibilidad de borrar los existentes.

4.4. Detalles técnicos de la solución propuesta

En los siguientes apartados se presentan con un lenguaje técnico y la debida profundidad los diferentes detalles que componen la solución propuesta.

4.4.1. Configuración de la instancia separada

Uno de los requisitos existentes que incide directamente en la base sobre la que realizar todo el desarrollo posterior exige la separación completa de la interfaz web de AIL de manera que sea

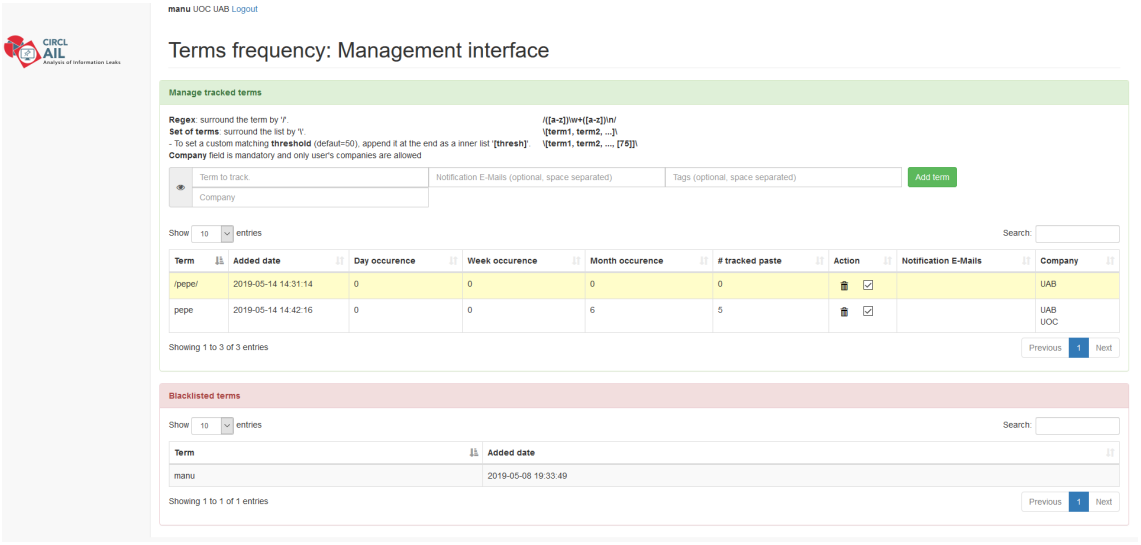


Figura 14: Portal de gestión de términos para un usuario autenticado.

posible proporcionar accesos diferenciados a cada sitio web. Por defecto el servidor web *Flask* de AIL se despliega en el puerto 7000 de la máquina que lo alberga, de modo que se debe utilizar otro puerto diferente para desplegar el segundo servidor web, en lugar de añadir funcionalidades a ese servidor.

Además, como se explica en el proceso de instalación detallado en el apartado 2.3 AIL dispone de un entorno virtual que se tiene que activar para poder arrancar la instancia. Dado que el despliegue del portal de gestión requiere ciertas instalaciones, para tratar de conseguir que la funcionalidad añadida en esta solución sea más independiente se opta por crear un entorno virtual diferenciado denominado "managementPortal", en el cual se instalan las dependencias de Flask necesarias.

Para la creación y activación del entorno virtual simplemente se crea una carpeta a tal efecto y se ejecuta el siguiente comando:

```
# virtualenv -p python3 managementPortal
```

Tras lo cual ejecutando el siguiente comando desde la ruta de la carpeta creada se activa el entorno virtual, y con el entorno virtual activado se instalan *Flask* y *Flask-SqlAlchemy*, una extensión necesaria para el tratamiento de bases de datos desde Python.:

```
# . managementPortal/bin/activate
```

Por último la aplicación python desarrollada debe estar publicada en un puerto diferente al 7000 que utiliza el *Flask server* de AIL. En el fichero *var/www/Flask_server.py* de AIL está establecida en la parte final del mismo la siguiente configuración:

```
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=7000, threaded=True)
```

De manera que en la aplicación que se desarrolla para el portal de gestión se aplica una configuración equivalente, cambiando el puerto 7000 por el 4000.

```
if __name__ == "__main__":
    app.run(debug=True,host='0.0.0.0', port=4000)
```

4.4.2. Conexión a la base de datos de términos

La instancia web del portal de gestión es completamente independiente de la instancia de AIL tal y como se explica en el apartado 4.4.1, pero el portal requiere conectarse a la base de datos ARDB para consultar y gestionar los términos monitorizados.

Para lograr este acceso se define la variable de entorno "AIL_BIN" dentro del script de activación del entorno virtual *managementPortal/bin/activate* apuntando a la ruta de la carpeta "bin" de la instancia de AIL. Utilizando esta variable de entorno, en el fichero de configuración del portal de gestión se realiza la siguiente configuración para la lectura de la configuración de AIL y la conexión a la base de datos:

```
# CONFIG #
configfile = os.path.join(os.environ['AIL_BIN'], 'packages/config.cfg')
if not os.path.exists(configfile):
   raise Exception('Unable to find the configuration file. \
               Did you set environment variables? \
               Or activate the virtualenv.')

cfg = configparser.ConfigParser()
cfg.read(configfile)

# REDIS #
r_serv_term = redis.StrictRedis(
      host=cfg.get("ARDB_TermFreq", "host"),
      port=cfg.getint("ARDB_TermFreq", "port"),
      db=cfg.getint("ARDB_TermFreq", "db"),
      decode_responses=True)

r_serv_db = redis.StrictRedis(
   host=cfg.get("ARDB_DB", "host"),
   port=cfg.getint("ARDB_DB", "port"),
   db=cfg.getint("ARDB_DB", "db"),
   decode_responses=True)
```

Esta solución no solo permite la conexión desde el portal de gestión a la base de datos que contiene los términos a monitorizar, sino que además utiliza el propio fichero de configuración de la instancia AIL, de modo que no es necesario mantener un fichero paralelo con información duplicada.

4.4.3. Configuración de la base de datos de usuarios

Para configurar la autenticación previa al acceso al portal de gestión es necesario implementar un registro de usuarios, contraseñas y compañías asociadas a cada usuario, para lo cual se crea la base de datos *identityManagement.db* ubicada en la carpeta del portal de gestión. El esquema de la base de datos es el siguiente:

```
CREATE TABLE users (
      id INTEGER NOT NULL,
      username VARCHAR,
      password VARCHAR,
      salt VARCHAR,
      PRIMARY KEY (id)
);
CREATE TABLE companies (
      id INTEGER NOT NULL,
      "companyName" VARCHAR,
      termsLimit INTEGER,
      PRIMARY KEY (id)
);
CREATE TABLE IF NOT EXISTS "usersCompany" (
      id INTEGER NOT NULL,
      "userID" INTEGER,
```

```
      "companyID" INTEGER,
      PRIMARY KEY (id),
      FOREIGN KEY("userID") REFERENCES users (id),
      FOREIGN KEY("companyID") REFERENCES companies (id)
);
CREATE TRIGGER userAdminCompanies
      AFTER INSERT ON companies
      BEGIN INSERT INTO usersCompany(userID,companyID)
            VALUES ((SELECT id FROM users WHERE username = 'admin'),new.id);
      END;
```

En la tabla "users" se almacenan los usuarios, el *hash* de la contraseña y el *salt* necesario para calcular dicho *hash* tal y como se detalla en el apartado 4.4.4. La tabla "companies" almacena las distintas compañías existentes y el límite de términos que cada compañía puede monitorizar. Se ha establecido este límite por motivos de control del rendimiento de la instancia ya que demasiados términos siendo monitorizados pueden afectar a la instancia de AIL. Por último la tabla "usersCompany" relaciona las dos tablas anteriores para asignar usuarios a compañías.

Además, con el objetivo de facilitar la gestión se ha incluido en el esquema de la base de datos el trigger "userAdminCompanies", el cual implica que cada vez que se añade una compañía, esta se relaciona con el usuario administrador. Con ello se logra que el administrador —que como se detalla en el apartado 5.1 se genera durante el proceso de instalación— pueda gestionar los términos de todas las compañías y tenga un control completo de forma automatizada.

Para lograr el acceso a la base de datos desde la aplicación del portal de gestión es necesario definir clases que registren las tablas y columnas de la base de datos, lo cual se realiza en el archivo *identityManagement.py* que se presenta en el anexo A.2.

4.4.4. Configuración de la autenticación

El proceso de autenticación se configura en el fichero del portal de gestión que se presenta en el anexo A.1. El portal consta de varias rutas posibles, la ruta por defecto, la ruta de *login*, la de *logout* y la del portal propiamente dicho. El acceso a cualquiera de estas rutas sin una autenticación previa será siempre redirigido a la página de login, en la cual únicamente se solicita usuario y contraseña tal y como se puede observar en la figura 13.

```
def do_admin_login():
   global userName
   Session = sessionmaker(bind=engine)
   s = Session()

   POST_USERNAME = str(request.form['username'])
   POST_PASSWORD = str(request.form['password'])

   query = s.query(User).filter(User.username.in_([POST_USERNAME]))
   result = query.first()
   if result:
      #Get user's salt to validate password
      saltQuery = s.query(User).filter(User.username.in_([POST_USERNAME])).first()
      salt = saltQuery.salt
      hashed_password = hashlib.sha512((POST_PASSWORD + salt).encode('utf-8')).hexdigest()

      if (result.password == hashed_password):
         session['logged_in'] = True
         userName = result.username

         #Get user's companies
         for userCompany in s.query(UsersCompany).filter_by(userID=result.id):
            company = s.query(Company).filter_by(id=userCompany.companyID).first()
            userCompanies.append(company.companyName)
            companyLimit.append(company.termsLimit)
      else: #Wrong password
```

```
        flash('Wrong user or password')#Same message to prevent attacks
    else:
        flash('Wrong user or password')#Same message to prevent attacks
    return home()
```

La función *do_admin_login()* define toda la lógica que acompaña al portal de *login*. El proceso comienza recibiendo los parámetros correspondientes al usuario y la contraseña enviados en la petición de tipo POST. Con el usuario recibido se realiza la primera consulta a la base de datos para confirmar la existencia de dicho usuario, mostrando un mensaje de error en caso contrario.

En caso de que el usuario exista, se obtiene de la base de datos el *salt* y se obtiene el *hash sha512* del conjunto formado por su contraseña y el *salt*, el cual se compara con la contraseña almacenada en la base de datos. Si la autenticación es satisfactoria se obtienen las compañías a las que pertenece el usuario y el límite de términos a monitorizar por cada una.

> *En caso de que la autenticación falle bien porque el usuario no exista en base de datos o bien porque la contraseña no sea correcta se muestra siempre el mismo mensaje. Esta medida se toma como protección ya que si los mensajes son diferentes se estaría proporcionando una información sobre los usuarios existentes a un atacante.*

4.4.5. Filtrado de compañías del usuario

En el portal de gestión de términos, un usuario autenticado únicamente debe poder ver los términos monitorizados por su compañía. En el código del portal de gestión este control se realiza por triplicado para los términos de tipo "regex", los sets y los términos; y hace uso del listado de compañías del usuario "userCompanies" generado en el proceso de autenticación como se detalla en el apartado 4.4.4.

```
#Show only terms of the user's companies
userInCompany = False
for company in userCompanies:
    termCompanies = r_serv_term.smembers(TrackedTermsCompanyPrefix_Name + tracked_term)
    if termCompanies:
        for termCompany in termCompanies:
            if (company == termCompany):
                userInCompany = true
if (not userInCompany):
    continue
```

El proceso obtiene las compañías a las que está asociado el término y las compara con las compañías a las que pertenece el usuario. El código anterior se sitúa dentro de cada uno de los bucles "for" que recorren los tipos de términos almacenados, de manera que si al finalizar el código la variable "userInCompany" continúa siendo falsa se ejecuta la orden "continue" que salta al siguiente término del bucle "for" y por tanto no se ejecutan las ordenes posteriores que corresponden a la obtención de todos los datos del término monitorizado.

4.4.6. Control del límite de términos monitorizados por cada compañía

Para controlar el rendimiento de la instancia y evitar una sobrecarga se establecen límites al número de términos que cada compañía puede monitorizar, que se almacenan en la base de datos como se presenta en el apartado 4.4.3. En el portal se deben contabilizar los términos que están siendo monitorizados por cada compañía para evitar que el usuario añada más términos si ya se ha alcanzado el límite. Se aprovecha para esto el propio proceso de carga de la página ya que en él se obtiene el listado de todos los términos y también las compañías asociadas.

```
#Counting terms tracked by each company
for term in companyTermMapping:
   for idx,userCompany in enumerate(userCompanies):
      for companyOfTerm in companyTermMapping[term]:
         if ( companyOfTerm == userCompany):
            if not userCompany in companiesTrackedTerms:
               companiesTrackedTerms[userCompany] = 1
            else:
               companiesTrackedTerms[userCompany] += 1

#Initialize companiesTrackedTerms for companies without terms
for userCompany in userCompanies:
   if not userCompany in companiesTrackedTerms:
      companiesTrackedTerms[userCompany] = 0
```

El código anterior, que se encuentra al final de la lógica de carga de la página del portal, hace uso de la variable "userCompanies" generada en el apartado 4.4.3 y genera una lista denominada "companiesTrackedTerms" en la que almacena por cada compañía el número de términos que está monitorizando. La parte final del código sirve para añadir a la lista, con valor 0 las compañías a las que pertenezca el usuario y que no estén monitorizando ningún término.

4.4.7. Proceso para añadir términos y controles aplicados

El proceso para añadir términos es similar al que utiliza la página *"Terms management"* de AIL aunque se añaden ciertos controles para asegurar que el término se puede añadir. Este proceso se define en la función *mgmtPortal_action()* correspondiente a la ruta "/mgmtPortal_action/" que se presenta en el fichero del portal de gestión del anexo A.1; y las comprobaciones añadidas son las siguientes:

```
#company field is mandatory
if not company:
   return "None"

#User has to belong to the company to be able to add a term
userInCompany = False
for companyOfUser in userCompanies:
   if (company == companyOfUser):
      userInCompany = true
if (not userInCompany):
   return "None"

#Check for tracked terms company limit
for idx,userCompany in enumerate(userCompanies):
   if (company == userCompany):
      if (companiesTrackedTerms[company] >= companyLimit[idx]):
         flash('Tracked terms limit.')
         return "None"
```

Al igual que el término, el campo "Company" es obligatorio, de modo que la primera comprobación sirve para confirmar que se ha rellenado dicho campo en el formulario; mientras que la segunda confirma que el usuario pertenece a la compañía que ha indicado en el campo Company, de manera que no pueda añadir términos a otras compañías.

La última de las comprobaciones está relacionada con el límite establecido para cada compañía y que se almacena en la variable "companyLimit" en el apartado 4.4.4. El proceso el límite de cada compañía con el valor correspondiente de la lista "companiesTrackedTerms" generada en el apartado 4.4.6 durante la carga de la página; imposibilitando así que se suba un término si se ha alcanzado el límite para la compañía.

4.4.8. Proceso de borrado de términos

Al igual que con el proceso para añadir términos, el proceso de borrado de términos se ha modificado ligeramente ya que un usuario únicamente puede borrar términos de su compañía, pero si un término está siendo monitorizado por varias compañías este no se debe borrar hasta que todas las compañías lo hagan.

```
#delete the company
for company in userCompanies:
   r_serv_term.srem(TrackedTermsCompanyPrefix_Name + term, company)

#delete term from DB only if no company is tracking it
if not r_serv_term.smembers(TrackedTermsCompanyPrefix_Name + term):
   if term.startswith('/') and term.endswith('/'):
      r_serv_term.srem(TrackedRegexSet_Name, term)
      r_serv_term.hdel(TrackedRegexDate_Name, term)
   elif term.startswith('\\') and term.endswith('\\'):
      r_serv_term.srem(TrackedSetSet_Name, term)
      r_serv_term.hdel(TrackedSetDate_Name, term)
   else:
      r_serv_term.srem(TrackedTermsSet_Name, term.lower())
      r_serv_term.hdel(TrackedTermsDate_Name, term.lower())

   # delete the associated notification emails too
   r_serv_term.delete(TrackedTermsNotificationEmailsPrefix_Name + term)
   # delete the associated tags set
   r_serv_term.delete(TrackedTermsNotificationTagsPrefix_Name + term)
```

La modificación con respecto al comportamiento desde la página de *"Terms management"* consiste en que en primer lugar se borran las entradas de base de datos de las compañías del usuario para dicho término, y tras esto, el borrado completo del término, correos y etiquetas de la base de datos solo se realiza si no existe ninguna compañía asociada al término.

4.5. Modificaciones sobre la instancia de AIL

La solución propuesta y detallada a lo largo de los apartados 4.3 y 4.4 es completamente independiente de la instancia de AIL y no interfiere con esta en su funcionamiento. Sin embargo puede resultar interesante modificar la página de *"Terms management"* para incluir la información relativa a las compañías para que el administrador de la instancia AIL pueda gestionar de manera más sencilla y centralizada todos los datos, ya que en caso contrario el control de la información de las compañías que gestionan cada término deberá realizarlo desde el portal de gestión utilizando un usuario "administrador" incluido en todas las compañías".

El fichero Python del módulo con todas las modificaciones se presenta en el anexo A.5, y se listan a continuación las modificaciones realizadas sobre el fichero original.

- En la línea 57 se define el prefijo utilizado en la base de datos de términos para identificar los registros de compañías asociadas a un término.
- En las líneas 168 se define la variable sobre la que se obtiene en la línea 177 los valores de compañía asociados a los términos de tipo *regex*, y lo mismo se realiza en las líneas 205 y 232 para los *sets* y los términos.
- En la línea 266 se devuelve la variable anterior al proceso de renderizado de la página.
- En la línea 336 se obtiene el valor del campo "Company" del formulario al igual que lo detallado en el apartado 4.4.7, mientras que la línea 344 comprueba que este no esté vacío.

- Las líneas 376, 400 y 416 añaden a la base de datos el valor del campo “Company”.
- En la línea 444 se borra de la base de datos la compañía. A diferencia de lo explicado en el apartado 4.4.8, cuando el administrador borra un término se borra de todas las compañías.

También es necesario modificar el fichero HTML que carga la página, y cuyo contenido modificado se presenta en el anexo A.6. Las modificaciones realizadas sobre el archivo original de la instancia de AIL son las siguientes:

- En la línea 110 se añade el campo para introducir “Company” al formulario.
- La línea 125 añade la columna “Company” a la tabla.
- Las líneas 204 a 209, y del mismo modo las líneas 287 a 292 y 370 a 375 sirven para mostrar las compañías de cada término.
- En la línea 572 se incluye la compañía como información a entregar cuando se realizan acciones sobre la página.

El resultado de estas modificaciones es el que se muestra en la imagen 15.

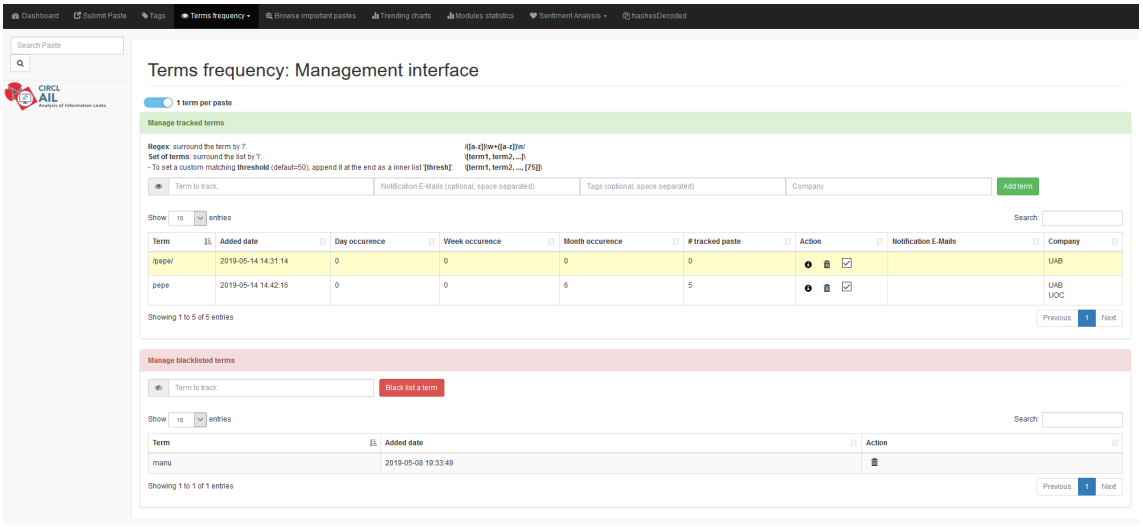


Figura 15: Pestaña *“Terms management”* adaptada con la información de compañía.

5. Generación de la extensión instalable

El objetivo final del trabajo es desarrollar una solución que pueda ser utilizada como *add-on* para complementar a una instancia AIL. Para ello se debe publicar una versión completamente anonimizada con un proceso de instalación tan automático como sea posible y adaptable a cualquier entorno.

De los ficheros generados para completar la solución tal y como se detalla en el apartado 4, los ficheros HTML, los relativos a formato de páginas y la propia aplicación y el script de gestión de la base de datos se pueden publicar directamente, mientras que otros como la base de datos o el entorno virtual, al incluir información relativa a la instancia en que se ejecutan, deben ser generados durante el proceso de instalación del *add-on*.

5.1. Proceso de instalación del *add-on*

El entorno virtual generado tal y como se detalla en el apartado 4.4.1 depende de la ruta en la que se genere, de modo que no se puede publicar directamente toda la estructura de ficheros que lo componen, sino que se debe generar un entorno virtual durante la instalación. Al igual que en apartado 4.4.1, se debe ejecutar la siguiente orden para generarlo:

```
# virtualenv -p python3 managementPortal
```

En el apartado 4.4.2 se describen las configuraciones necesarias para que el portal de gestión sea capaz de acceder a los datos de la base de datos de términos, entre los cuales se incluye la definición de una variable de entorno en el script de activación del entorno virtual haciendo referencia a la ruta de la carpeta *bin* de la instancia de AIL. Al tratarse de una ruta dependiente del entorno en que se ejecuta se solicita al usuario introducir la ruta de instalación de AIL y se compone con ella la ruta completa y la definición de la variable de entorno, que se añade al script de activación con los siguientes comandos:

```
read -p "Introduce AIL installation path: " AILPath
echo "export AIL_BIN="$AILPath"/bin/" >> managementPortal/bin/activate
```

La ejecución del portal de gestión hace uso de algunas extensiones de python específicas como son *flask, sqlalchemy y redis*, de modo que durante el proceso de instalación es necesario instalar estas extensiones en el contexto del entorno virtual creado:

```
source managementPortal/bin/activate
pip3 install -U flask sqlalchemy redis
```

Y por último para completar las tareas necesarias para la correcta configuración del entorno para ejecutar el portal de gestión es necesario generar la base de datos de usuarios detallada en el apartado 4.4.3, lo cual se puede hacer ejecutando el script *identityManagement.py* que se presenta en el anexo A.2.

Adicionalmente es necesario definir el trigger “userAdminCompanies” de forma manual ya que no está incluido en la definición de clases del script en Python, y también inicializar la base de datos con el usuario administrador para que dicho trigger funcione correctamente. Ejecutando el script *initializeDatabase.py* que se presenta en el anexo A.8 se automatizan dichas tareas y también se crea una compañía de ejemplo para que el portal sea operativo para el usuario “admin”. La contraseña predeterminada es “password” y se genera un *salt* aleatorio con el cual se calcula el *hash* de la misma para almacenarla en la base de datos.

5.2. Automatización de las tareas de instalación del *add-on*

A fin de automatizar las tareas detalladas en el apartado 5.1 para lograr una mayor sencillez y comodidad en la instalación se genera el fichero *previousConfigurations.sh* cuyo contenido se presenta en el anexo A.7.

Adicionalmente a los pasos indicados en el apartado 5.1 se incluye en él la línea 23 con la cual se modifican los permisos de la base de datos para establecer únicamente permiso de lectura para el usuario propietario. De este modo la base de datos sigue siendo operativa para el uso que hace de ella el portal de gestión y se securiza evitando que pueda ser modificada. Además, por higiene del código se han extraído a un fichero externo denominado *requirements.txt* las extensiones de Python a instalar en la línea 14.

5.3. Publicación del *add-on* en *github.com*

El *add-on* desarrollado se publica en el repositorio `github.com` para su descarga al igual que está publicado el proyecto del *framework* AIL. La url de acceso al código es la siguiente:

https://github.com/mmc-mistic19/managementPortal

En el repositorio se pueden encontrar los ficheros necesarios para configurar y ejecutar la aplicación y se ha añadido un fichero *README.md* que se puede ver en la imagen 16 y resume el contenido del repositorio y el proceso de instalación y uso.

5.4. Scripts de gestión de la base de datos de usuarios

En el apartado 4.4.3 se describe la solución utilizada para almacenar las credenciales de los usuarios y la relación de usuarios y compañías, que consiste en el uso de una base de datos denominada *identityManagement.db*. La gestión de la misma se podría realizar utilizando *sqlite* u otros gestores de bases de base de datos, pero con el objetivo de facilitar la gestión se han incluido varios scripts en la carpeta *dbManagement* del repositorio.

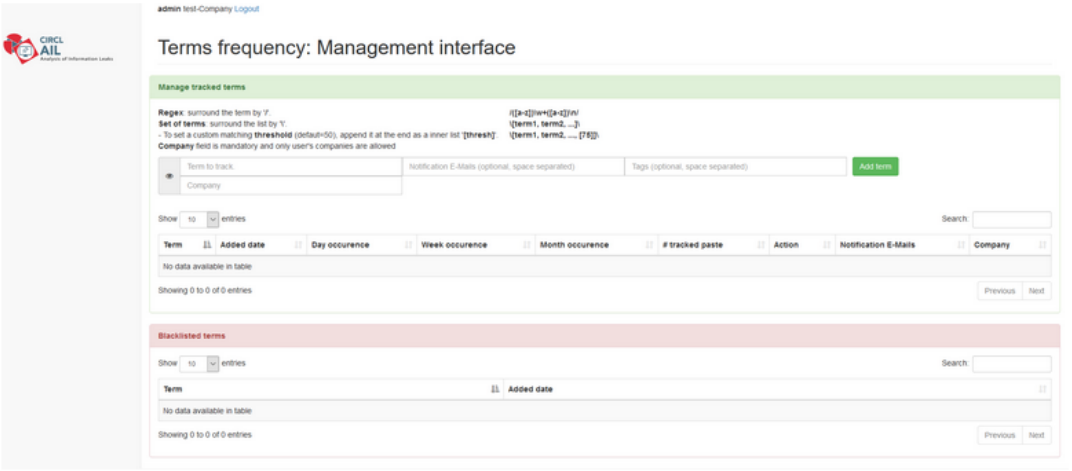
Para la creación de usuarios se proporciona el script *createUsers.py* que se puede consultar en el anexo A.9. En primer lugar el script cambia los permisos del fichero de la base de datos para permitir la escritura, lo cual se deshace posteriormente en la última línea del script para dejar la base de datos con permisos de lectura únicamente como se explica en el apartado 5.2. En el resto del código del script se solicita al usuario que introduzca el nombre y contraseña del usuario, se calcula el *hash* de la misma con un *salt* generado aleatoriamente y se ejecuta el "INSERT" en la base de datos.

De forma equivalente, el script *createCompanies.py* que se puede consultar en el anexo A.10 permite la creación de compañías introduciendo el nombre de la misma y el límite de términos que esta podrá gestionar; y para crear relaciones entre usuarios y compañías se proporciona el script *createRelationships.py* replicado en el anexo A.11.

A modo de tutorial para explicar el uso de los tres scripts se ha publicado en la misma carpeta el archivo *HOWTO.md* cuyo contenido muestra la imagen 17.

Management Portal for AIL multitenant instance

This addon provides a web portal to manage terms in an AIL instance used by multiple companies.



About AIL framework: https://github.com/CIRCL/AIL-framework

Installation

Type these command lines for a fully automated installation (asks for the AIL framework instance's installation path):

```
git clone https://github.com/mmc-mistic19/managementPortal.git
cd managementPortal
chmod +x previousConfigurations.sh
./previousConfigurations.sh
```

Usage

Activate the virtual enviroment and run the flask server. By default server runs in port 4000.

```
source managementPortal/bin/activate
python3 managementPortal.py
```

Default user and password: admin/password

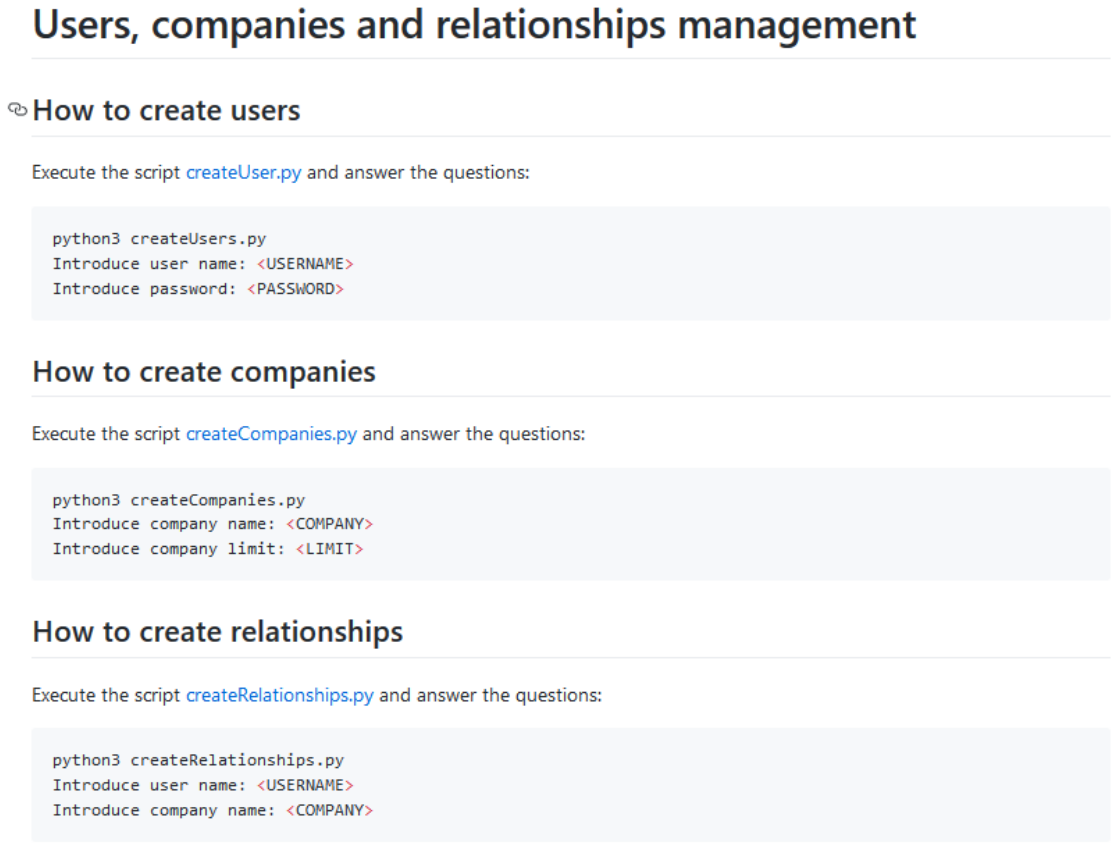
Manage users and companies

Automated scripts to configure users, companies and relationships are provided in the dbManagement folder. More details available in HOWTO.md.

License

MIT

Figura 16: Contenido del fichero *README.md* que describe el repositorio.



Users, companies and relationships management

How to create users

Execute the script createUser.py and answer the questions:

```
python3 createUsers.py
Introduce user name: <USERNAME>
Introduce password: <PASSWORD>
```

How to create companies

Execute the script createCompanies.py and answer the questions:

```
python3 createCompanies.py
Introduce company name: <COMPANY>
Introduce company limit: <LIMIT>
```

How to create relationships

Execute the script createRelationships.py and answer the questions:

```
python3 createRelationships.py
Introduce user name: <USERNAME>
Introduce company name: <COMPANY>
```

Figura 17: Contenido del fichero *HOWTO.md* en el que se explica la utilización de los scripts proporcionados para la gestión de la base de datos.

6. Conclusiones

Las fugas de datos son una de las nuevas amenazas a las que se enfrentan las empresas y organizaciones una vez la informatización y digitalización ha cambiado el paradigma de almacenamiento de la información. Ante ello existen diferentes estrategias y productos para reducir el riesgo y mitigar las consecuencias derivadas de una fuga, desde las técnicas y teorías de protección de datos y control de accesos, hasta los productos DLP como AIL.

A lo largo de la realización del presente trabajo se ha podido comprobar como el *framework* AIL pese a ser un producto enfocado a un tipo de fuga concreto permite obtener un control y conseguir una detección de primera mano de esas fugas cada vez más comunes en sitios de tipo *pastebin*. Como se demuestra a lo largo del apartado 3, las funcionalidades de AIL permiten la detección automática de distintos tipos de información, la creación de términos y expresiones de búsqueda e incluso la integración con herramientas de gestión de incidentes informáticos para cumplir con las buenas prácticas en el ámbito de la gestión de TI y potenciar y mejorar el servicio y la seguridad de los datos.

Por otro lado, la solución propuesta en el apartado 4 ante la problemática de gestión de la instancia AIL en un entorno con diferentes organizaciones utilizando el mismo *tenant* cumple con todos los requisitos funcionales establecidos siendo además completamente independiente de la instancia de AIL tanto en su instalación como en su funcionamiento. La extensión desarrollada permite a usuarios de las diferentes compañías gestionar los términos y expresiones que les afectan sin necesidad de acceder a los del resto, algo a evitar dada la sensibilidad de los términos que cada organización rastrea ya que pueden incluir información de carácter protegido.

Sobre la base de este trabajo se podrían iniciar diversas vías de estudio con el fin de mejorar algunos aspectos de la solución implementada. Una línea de trabajo puede ser mejorar la gestión de los usuarios, términos y relaciones implementando una integración con LDAP u otros sistemas de federación de identidades. Otra posibilidad consiste en impulsar la integración de la solución propuesta en la instancia de AIL mejorando la solución propuesta en el apartado 4.5 de sustitución de ciertos archivos para adaptar la página de gestión de términos de AIL.

Por las características de AIL, su servicio web no debe estar publicado y únicamente los administradores de la instancia deben poder acceder al mismo, pero en el caso del portal de gestión su audiencia potencial es mayor y por ello debe ser publicado al menos a los distintos administradores de las compañías, y dado que accede a los datos de AIL, la implementación de la extensión del portal de gestión aumenta la superficie de exposición de la información que se maneja en la instancia. Una tercera vía podría por tanto orientarse a la realización de un *pentesting* sobre una instancia del portal de gestión para elaborar un proyecto de *hardening* y mejorar las medidas de seguridad implementadas.

7. Glosario

- ***Add-on***: Extensión o expansión. Adición independiente al programa principal que proporciona nuevas funcionalidades.
- **CERT/CSIRT**: Equipo de respuesta ante incidentes informáticos o equipo de respuesta ante incidencias de seguridad informática. Grupo de expertos en la gestión de incidentes relacionados con la seguridad de la información en el ámbito de la informática y las comunicaciones.
- ***Data exfiltration***: Extracción de datos no autorizados por parte de un tercero o un *malware*.
- **DLP**: *Data leak prevention* o prevención de fuga de datos. Práctica para evitar la fuga o el acceso no autorizado a datos sensibles.
- **Entorno virtual**: En Python, entorno aislado generado para el desarrollo y ejecución de un proyecto concreto sin interferir con otros.
- **Flask**: *Framework* escrito en Python que permite el despliegue de aplicaciones web.
- ***framework***: Marco de trabajo, es el esquema o estructura que se establece y que se aprovecha para desarrollar y organizar un software determinado. Es el entorno pensado para hacer más sencilla la programación de cualquier aplicación o herramienta actual.
- ***Hash***: Una función *hash* o función resumen recibe como entrada un conjunto de elementos de longitud variable y lo convierte en un rango de salida finito. Se utiliza para generar una representación compacta de la cadena de entrada.
- **Python**: Lenguaje de programación interpretado y de alto nivel.
- ***Pastebin sites***: Servicio web que permite a los usuarios almacenar texto plano, comúnmente utilizado para almacenar partes de código.
- **Regex**: Secuencia de caracteres que definen un patrón de búsqueda.
- ***Salt***: Conjunto aleatorio de datos que se combina con la contraseña en texto plano a la hora de generar su *hash* para ofrecer mayor seguridad al almacenamiento de contraseñas.
- ***Sqlite***: [En el contexto utilizado] Interfaz por línea de comandos para bases de datos SQLite.
- ***Trigger***: [En bases de datos] Sentencia o conjunto de sentencias que en base a un desencadenante definido se ejecutan de forma autónoma.
- **Variable de entorno**: Variables definidas a nivel del sistema operativo y utilizables por diferentes programas o sentencias.
- **ZeroMQ**: Librería de mensajería asíncrona.

8. Bibliografía

1. Caffey, Brian; **Tutorial sobre el uso de entornos virtuales en Python**. (Mayo 2019).
https://briancaffey.github.io/2017/12/09/setting-up-flask-cli-with-docker.html
2. ENISA; **CSIRTs by Country - Interactive Map**. (Marzo 2019).
https://www.enisa.europa.eu/topics/csirts-in-europe/csirt-inventory/certs-by-country-interactive-map
3. Github; **Repositorio de AIL en github.com**. (Marzo 2019).
https://github.com/CIRCL/AIL-framework
4. Github; **Repositorio de ARDB en github.com**. (Marzo 2019).
https://github.com/yinqiwen/ardb
5. Github; **Repositorio de MISP en github.com**. (Marzo 2019).
https://github.com/MISP/MISP
6. Github; **Repositorio de pystemon en github.com**. (Marzo 2019).
https://github.com/CIRCL/pystemon
7. Howtoforge; **Tutorial sobre uso de git en Ubuntu**. (Mayo 2019).
https://www.howtoforge.com/tutorial/install-git-and-github-on-ubuntu/
8. Neoattack; **Definición de framework**. (Mayo 2019).
https://neoattack.com/neowiki/framework/
9. Pythonspot; **Tutorial sobre autenticación en *Flask***. (Mayo 2019).
https://pythonspot.com/login-authentication-with-flask/
10. Realpython; **Definición de entornos virtuales**. (Mayo 2019).
https://realpython.com/python-virtual-environments-a-primer/
11. Stackoverflow; **Explicación de los tipos de dato en *Redis***. (Abril 2019).
https://stackoverflow.com/questions/37953019/wrongtype-operation-against-a37953349
12. The Hive Project; **Página web del proyecto The Hive**. (Marzo 2019).
https://thehive-project.org/
13. Wikipedia; **Definición de extensión**. (Mayo 2019).
https://es.wikipedia.org/wiki/Expansi%C3%B3n_(videojuegos)
14. Wikipedia; **Definición de función *hash***. (Mayo 2019).
https://es.wikipedia.org/wiki/Funci%C3%B3n_hash
15. Wikipedia; **Definición de *flask***. (Marzo 2019).
https://es.wikipedia.org/wiki/Flask
16. Wikipedia; **Definición de *framework***. (Mayo 2019).
https://en.wikipedia.org/wiki/Software_framework
17. Wikipedia; **Definición de *regex***. (Marzo 2019).
https://es.wikipedia.org/wiki/Expresi%C3%B3n_regular

18. Wikipedia; **Definición de *ZeroMQ***. (Marzo 2019).
`https://en.wikipedia.org/wiki/ZeroMQ`

19. Wikipedia; **"Exfiltración" de datos o *data exfiltration***. (Marzo 2019).
`https://en.wikipedia.org/wiki/Data_exfiltration`

20. Wikipedia; **Gusano Morris**. (Marzo 2019).
`https://en.wikipedia.org/wiki/Morris_worm`

21. Wikipedia; **Sitios web de tipo *pastebin***. (Marzo 2019).
`https://en.wikipedia.org/wiki/Pastebin`

Anexos

A. Archivos generados

A.1. managementPortal.py

```
from flask import Flask
from flask import Flask, flash, redirect, render_template, request, session, abort, url_for, jsonify
import os
from sqlalchemy.orm import sessionmaker
from identityManagement import *
engine = create_engine('sqlite:///identityManagement.db', echo=True)
import redis
import configparser
import datetime
import calendar
import re
import hashlib, uuid

app = Flask(__name__)

# CONFIG #
configfile = os.path.join(os.environ['AIL_BIN'], 'packages/config.cfg')
if not os.path.exists(configfile):
    raise Exception('Unable to find the configuration file. \
                Did you set environment variables? \
                Or activate the virtualenv.')

cfg = configparser.ConfigParser()
cfg.read(configfile)

# REDIS #
r_serv_term = redis.StrictRedis(
      host=cfg.get("ARDB_TermFreq", "host"),
      port=cfg.getint("ARDB_TermFreq", "port"),
      db=cfg.getint("ARDB_TermFreq", "db"),
      decode_responses=True)

r_serv_db = redis.StrictRedis(
    host=cfg.get("ARDB_DB", "host"),
    port=cfg.getint("ARDB_DB", "port"),
    db=cfg.getint("ARDB_DB", "db"),
    decode_responses=True)


bootstrap_label = ['primary', 'success', 'danger', 'warning', 'info']

# VARIABLES #
#tracked
TrackedTermsSet_Name = "TrackedSetTermSet"
TrackedTermsDate_Name = "TrackedTermDate"
#black
BlackListTermsDate_Name = "BlackListTermDate"
BlackListTermsSet_Name = "BlackListSetTermSet"
#regex
TrackedRegexSet_Name = "TrackedRegexSet"
TrackedRegexDate_Name = "TrackedRegexDate"
#set
TrackedSetSet_Name = "TrackedSetSet"
TrackedSetDate_Name = "TrackedSetDate"

# notifications enabled/disabled
# same value as in `bin/NotificationHelper.py`
TrackedTermsNotificationEnabled_Name = "TrackedNotifications"

# associated notification email addresses for a specific term`
```

```
# same value as in `bin/NotificationHelper.py`
# Keys will be e.g. TrackedNotificationEmails_<TERMNAME>
TrackedTermsNotificationEmailsPrefix_Name = "TrackedNotificationEmails_"
TrackedTermsNotificationTagsPrefix_Name = "TrackedNotificationTags_"
TrackedTermsCompanyPrefix_Name = "company_"

userCompanies = []
companyLimit = []
companiesTrackedTerms = {}

# FUNCTIONS #
def Term_getValueOverRange(word, startDate, num_day, per_paste=""):
   passed_days = 0
   oneDay = 60*60*24
   to_return = []
   curr_to_return = 0
   for timestamp in range(startDate, startDate - max(num_day)*oneDay, -oneDay):
      value = r_serv_term.hget(per_paste+str(timestamp), word)
      curr_to_return += int(value) if value is not None else 0
      for i in num_day:
         if passed_days == i-1:
            to_return.append(curr_to_return)
      passed_days += 1
   return to_return

def save_tag_to_auto_push(list_tag):
   for tag in set(list_tag):
      #limit tag length
      if len(tag) > 49:
         tag = tag[0:48]
      r_serv_db.sadd('list_export_tags', tag)


# ROUTES #
@app.route('/')
def home():
   if not session.get('logged_in'):
      return render_template('login.html')
   else:
      return redirect(url_for('mgmtPortal_page'))

@app.route('/login', methods=['POST'])
def do_admin_login():
   global userName
   Session = sessionmaker(bind=engine)
   s = Session()

   POST_USERNAME = str(request.form['username'])
   POST_PASSWORD = str(request.form['password'])

   query = s.query(User).filter(User.username.in_([POST_USERNAME]))
   result = query.first()
   if result:
      #Get user's salt to validate password
      saltQuery = s.query(User).filter(User.username.in_([POST_USERNAME])).first()
      salt = saltQuery.salt
      hashed_password = hashlib.sha512((POST_PASSWORD + salt).encode('utf-8')).hexdigest()

      if (result.password == hashed_password):
         session['logged_in'] = True
         userName = result.username

         #Get user's companies
         for userCompany in s.query(UsersCompany).filter_by(userID=result.id):
            company = s.query(Company).filter_by(id=userCompany.companyID).first()
            userCompanies.append(company.companyName)
            companyLimit.append(company.termsLimit)
      else: #Wrong password
         flash('Wrong user or password')#Same message to prevent attacks
   else:
      flash('Wrong user or password')#Same message to prevent attacks
   return home()
```

```
133
134 @app.route("/logout")
135 def logout():
136     session['logged_in'] = False
137     userCompanies.clear()
138     companyLimit.clear()
139     companiesTrackedTerms.clear()
140     return home()
141
142 @app.route("/mgmtPortal/", methods=['GET'])
143 def mgmtPortal_page():
144     if not session.get('logged_in'):
145         return render_template('login.html')
146     per_paste_text = "per_paste_"
147     per_paste = 1
148
149     today = datetime.datetime.now()
150     today = today.replace(hour=0, minute=0, second=0, microsecond=0)
151     today_timestamp = calendar.timegm(today.timetuple())
152
153     # Map tracking if notifications are enabled for a specific term
154     notificationEnabledDict = {}
155
156     # Maps a specific term to the associated email addresses
157     notificationEMailTermMapping = {}
158     notificationTagsTermMapping = {}
159
160     # Maps a specific term to the associated company
161     companyTermMapping = {}
162
163     #Regex
164     trackReg_list = []
165     trackReg_list_values = []
166     trackReg_list_num_of_paste = []
167     for tracked_regex in r_serv_term.smembers(TrackedRegexSet_Name):
168         #Show only terms of the user's companies
169         userInCompany = False
170         for company in userCompanies:
171             termCompanies = r_serv_term.smembers(TrackedTermsCompanyPrefix_Name + tracked_regex
                ↪ )
172             if termCompanies:
173                 for termCompany in termCompanies:
174                     if (company == termCompany):
175                         userInCompany = true
176         if (not userInCompany):
177             continue
178
179         notificationEMailTermMapping[tracked_regex] = r_serv_term.smembers(
              ↪ TrackedTermsNotificationEmailsPrefix_Name + tracked_regex)
180         notificationTagsTermMapping[tracked_regex] = r_serv_term.smembers(
              ↪ TrackedTermsNotificationTagsPrefix_Name + tracked_regex)
181         companyTermMapping[tracked_regex] = r_serv_term.smembers(
              ↪ TrackedTermsCompanyPrefix_Name + tracked_regex)
182
183         if tracked_regex not in notificationEnabledDict:
184             notificationEnabledDict[tracked_regex] = False
185
186         trackReg_list.append(tracked_regex)
187         value_range = Term_getValueOverRange(tracked_regex, today_timestamp, [1, 7, 31],
              ↪ per_paste=per_paste_text)
188
189         term_date = r_serv_term.hget(TrackedRegexDate_Name, tracked_regex)
190
191         set_paste_name = "regex_" + tracked_regex
192         trackReg_list_num_of_paste.append(r_serv_term.scard(set_paste_name))
193         term_date = datetime.datetime.utcfromtimestamp(int(term_date)) if term_date is not
              ↪ None else "No date recorded"
194         value_range.append(term_date)
195         trackReg_list_values.append(value_range)
196
197         if tracked_regex in r_serv_term.smembers(TrackedTermsNotificationEnabled_Name):
198             notificationEnabledDict[tracked_regex] = True
```

```
199
200     #Set
201     trackSet_list = []
202     trackSet_list_values = []
203     trackSet_list_num_of_paste = []
204     for tracked_set in r_serv_term.smembers(TrackedSetSet_Name):
205        #Show only terms of the user's companies
206        userInCompany = False
207        for company in userCompanies:
208           termCompanies = r_serv_term.smembers(TrackedTermsCompanyPrefix_Name + tracked_set)
209           if termCompanies:
210              for termCompany in termCompanies:
211                 if (company == termCompany):
212                    userInCompany = true
213        if (not userInCompany):
214           continue
215
216        tracked_set = tracked_set
217
218        notificationEMailTermMapping[tracked_set] = r_serv_term.smembers(
               ↪ TrackedTermsNotificationEmailsPrefix_Name + tracked_set)
219        notificationTagsTermMapping[tracked_set] = r_serv_term.smembers(
               ↪ TrackedTermsNotificationTagsPrefix_Name + tracked_set)
220        companyTermMapping[tracked_set] = r_serv_term.smembers(TrackedTermsCompanyPrefix_Name
               ↪ + tracked_set)
221
222        if tracked_set not in notificationEnabledDict:
223           notificationEnabledDict[tracked_set] = False
224
225        trackSet_list.append(tracked_set)
226        value_range = Term_getValueOverRange(tracked_set, today_timestamp, [1, 7, 31],
               ↪ per_paste=per_paste_text)
227
228        term_date = r_serv_term.hget(TrackedSetDate_Name, tracked_set)
229
230        set_paste_name = "set_" + tracked_set
231        trackSet_list_num_of_paste.append(r_serv_term.scard(set_paste_name))
232        term_date = datetime.datetime.utcfromtimestamp(int(term_date)) if term_date is not
               ↪ None else "No date recorded"
233        value_range.append(term_date)
234        trackSet_list_values.append(value_range)
235
236        if tracked_set in r_serv_term.smembers(TrackedTermsNotificationEnabled_Name):
237           notificationEnabledDict[tracked_set] = True
238
239     #Tracked terms
240     track_list = []
241     track_list_values = []
242     track_list_num_of_paste = []
243     for tracked_term in r_serv_term.smembers(TrackedTermsSet_Name):
244        #Show only terms of the user's companies
245        userInCompany = False
246        for company in userCompanies:
247           termCompanies = r_serv_term.smembers(TrackedTermsCompanyPrefix_Name + tracked_term)
248           if termCompanies:
249              for termCompany in termCompanies:
250                 if (company == termCompany):
251                    userInCompany = true
252        if (not userInCompany):
253           continue
254
255        notificationEMailTermMapping[tracked_term] = r_serv_term.smembers(
               ↪ TrackedTermsNotificationEmailsPrefix_Name + tracked_term)
256        notificationTagsTermMapping[tracked_term] = r_serv_term.smembers(
               ↪ TrackedTermsNotificationTagsPrefix_Name + tracked_term)
257        companyTermMapping[tracked_term] = r_serv_term.smembers(TrackedTermsCompanyPrefix_Name
               ↪  + tracked_term)
258
259        if tracked_term not in notificationEnabledDict:
260           notificationEnabledDict[tracked_term] = False
261
262        track_list.append(tracked_term)
```

```
263         value_range = Term_getValueOverRange(tracked_term, today_timestamp, [1, 7, 31],
                ↪ per_paste=per_paste_text)
264
265         term_date = r_serv_term.hget(TrackedTermsDate_Name, tracked_term)
266
267         set_paste_name = "tracked_" + tracked_term
268
269         track_list_num_of_paste.append( r_serv_term.scard(set_paste_name) )
270
271         term_date = datetime.datetime.utcfromtimestamp(int(term_date)) if term_date is not
                ↪ None else "No date recorded"
272         value_range.append(term_date)
273         track_list_values.append(value_range)
274
275         if tracked_term in r_serv_term.smembers(TrackedTermsNotificationEnabled_Name):
276            notificationEnabledDict[tracked_term] = True
277
278     #blacklist terms
279     black_list = []
280     for blacked_term in r_serv_term.smembers(BlackListTermsSet_Name):
281        term_date = r_serv_term.hget(BlackListTermsDate_Name, blacked_term)
282        term_date = datetime.datetime.utcfromtimestamp(int(term_date)) if term_date is not
                ↪ None else "No date recorded"
283        black_list.append([blacked_term, term_date])
284
285     #Counting terms tracked by each company
286     for term in companyTermMapping:
287        for idx,userCompany in enumerate(userCompanies):
288           for companyOfTerm in companyTermMapping[term]:
289              if ( companyOfTerm == userCompany):
290                 if not userCompany in companiesTrackedTerms:
291                    companiesTrackedTerms[userCompany] = 1
292                 else:
293                    companiesTrackedTerms[userCompany] += 1
294
295     #Initialize companiesTrackedTerms for companies without terms
296     for userCompany in userCompanies:
297        if not userCompany in companiesTrackedTerms:
298           companiesTrackedTerms[userCompany] = 0
299
300     return render_template("mgmtPortal.html",
301           black_list=black_list, track_list=track_list, trackReg_list=trackReg_list,
                ↪ trackSet_list=trackSet_list,
302           track_list_values=track_list_values, track_list_num_of_paste=
                ↪ track_list_num_of_paste,
303           trackReg_list_values=trackReg_list_values, trackReg_list_num_of_paste=
                ↪ trackReg_list_num_of_paste,
304           trackSet_list_values=trackSet_list_values, trackSet_list_num_of_paste=
                ↪ trackSet_list_num_of_paste,
305           per_paste=per_paste, notificationEnabledDict=notificationEnabledDict,
                ↪ bootstrap_label=bootstrap_label,
306           notificationEMailTermMapping=notificationEMailTermMapping,
                ↪ notificationTagsTermMapping=notificationTagsTermMapping, companyTermMapping=
                ↪ companyTermMapping, userName=userName, userCompanies=userCompanies)
307
308 @app.route("/mgmtPortal_query_paste/")
309 def mgmtPortal_query_paste():
310     term = request.args.get('term')
311     paste_info = []
312
313     # check if regex or not
314     if term.startswith('/') and term.endswith('/'):
315        set_paste_name = "regex_" + term
316        track_list_path = r_serv_term.smembers(set_paste_name)
317     elif term.startswith('\\') and term.endswith('\\'):
318        set_paste_name = "set_" + term
319        track_list_path = r_serv_term.smembers(set_paste_name)
320     else:
321        set_paste_name = "tracked_" + term
322        track_list_path = r_serv_term.smembers(set_paste_name)
323
324     for path in track_list_path:
```

```
325         paste = Paste.Paste(path)
326         p_date = str(paste._get_p_date())
327         p_date = p_date[0:4]+'/'+p_date[4:6]+'/'+p_date[6:8]
328         p_source = paste.p_source
329         p_encoding = paste._get_p_encoding()
330         p_size = paste.p_size
331         p_mime = paste.p_mime
332         p_lineinfo = paste.get_lines_info()
333         p_content = paste.get_p_content()
334         if p_content != 0:
335            p_content = p_content[0:400]
336         paste_info.append({"path": path, "date": p_date, "source": p_source, "encoding":
              ↪ p_encoding, "size": p_size, "mime": p_mime, "lineinfo": p_lineinfo, "content":
              ↪ p_content})
337
338     return jsonify(paste_info)
339
340
341 @app.route("/mgmtPortal_query/")
342 def mgmtPortal_query():
343     TrackedTermsDate_Name = "TrackedTermDate"
344     term = request.args.get('term')
345     section = request.args.get('section')
346
347     today = datetime.datetime.now()
348     today = today.replace(hour=0, minute=0, second=0, microsecond=0)
349     today_timestamp = calendar.timegm(today.timetuple())
350     value_range = Term_getValueOverRange(term, today_timestamp, [1, 7, 31])
351
352     if section == "followTerm":
353        term_date = r_serv_term.hget(TrackedTermsDate_Name, term)
354
355     term_date = datetime.datetime.utcfromtimestamp(int(term_date)) if term_date is not None
          ↪ else "No date recorded"
356     value_range.append(str(term_date))
357     return jsonify(value_range)
358
359
360 @app.route("/mgmtPortal_action/", methods=['GET'])
361 def mgmtPortal_action():
362     today = datetime.datetime.now()
363     today = today.replace(microsecond=0)
364     today_timestamp = calendar.timegm(today.timetuple())
365
366
367     section = request.args.get('section')
368     action = request.args.get('action')
369     term = request.args.get('term')
370     notificationEmailsParam = request.args.get('emailAddresses')
371     input_tags = request.args.get('tags')
372     company = request.args.get('company')
373
374     if action is None or term is None or notificationEmailsParam is None:
375        return "None"
376     else:
377        if section == "followTerm":
378           if action == "add":
379
380              #company field is mandatory
381              if not company:
382                 return "None"
383
384              #User has to belong to the company to be able to add a term
385              userInCompany = False
386              for companyOfUser in userCompanies:
387                 if (company == companyOfUser):
388                    userInCompany = true
389              if (not userInCompany):
390                 return "None"
391
392              #Check for tracked terms company limit
393              for idx,userCompany in enumerate(userCompanies):
```

```
394                if (company == userCompany):
395                   if (companiesTrackedTerms[company] >= companyLimit[idx]):
396                      flash('Tracked terms limit.')
397                      return "None"
398
399             # Make a list of all passed email addresses
400             notificationEmails = notificationEmailsParam.split()
401
402             validNotificationEmails = []
403             # check for valid email addresses
404             for email in notificationEmails:
405                # Really basic validation:
406                # has exactly one @ sign, and at least one . in the part after the @
407                if re.match(r"[^@]+@[^@]+\.[^@]+", email):
408                   validNotificationEmails.append(email)
409
410             # create tags list
411             list_tags = input_tags.split()
412
413             # check if regex/set or simple term
414             #regex
415             if term.startswith('/') and term.endswith('/'):
416                r_serv_term.sadd(TrackedRegexSet_Name, term)
417                r_serv_term.hset(TrackedRegexDate_Name, term, today_timestamp)
418                # add all valid emails to the set
419                for email in validNotificationEmails:
420                   r_serv_term.sadd(TrackedTermsNotificationEmailsPrefix_Name + term, email)
421                # enable notifications by default
422                r_serv_term.sadd(TrackedTermsNotificationEnabled_Name, term)
423                # add tags list
424                for tag in list_tags:
425                   r_serv_term.sadd(TrackedTermsNotificationTagsPrefix_Name + term, tag)
426                save_tag_to_auto_push(list_tags)
427                # add company
428                r_serv_term.sadd(TrackedTermsCompanyPrefix_Name + term, company)
429
430
431             #set
432             elif term.startswith('\\') and term.endswith('\\'):
433                tab_term = term[1:-1]
434                perc_finder = re.compile("\[[0-9]{1,3}\]").search(tab_term)
435                if perc_finder is not None:
436                   match_percent = perc_finder.group(0)[1:-1]
437                   set_to_add = term
438                else:
439                   match_percent = DEFAULT_MATCH_PERCENT
440                   set_to_add = "\\" + tab_term[:-1] + ", [{}]]\\".format(match_percent)
441                r_serv_term.sadd(TrackedSetSet_Name, set_to_add)
442                r_serv_term.hset(TrackedSetDate_Name, set_to_add, today_timestamp)
443                # add all valid emails to the set
444                for email in validNotificationEmails:
445                   r_serv_term.sadd(TrackedTermsNotificationEmailsPrefix_Name + set_to_add,
                       ↪ email)
446                # enable notifications by default
447                r_serv_term.sadd(TrackedTermsNotificationEnabled_Name, set_to_add)
448                # add tags list
449                for tag in list_tags:
450                   r_serv_term.sadd(TrackedTermsNotificationTagsPrefix_Name + set_to_add, tag
                       ↪ )
451                save_tag_to_auto_push(list_tags)
452                # add company
453                r_serv_term.sadd(TrackedTermsCompanyPrefix_Name + term, company)
454
455             #simple term
456             else:
457                r_serv_term.sadd(TrackedTermsSet_Name, term.lower())
458                r_serv_term.hset(TrackedTermsDate_Name, term.lower(), today_timestamp)
459                # add all valid emails to the set
460                for email in validNotificationEmails:
461                   r_serv_term.sadd(TrackedTermsNotificationEmailsPrefix_Name + term.lower(),
                       ↪  email)
462                # enable notifications by default
```

```
463                 r_serv_term.sadd(TrackedTermsNotificationEnabled_Name, term.lower())
464                 # add tags list
465                 for tag in list_tags:
466                     r_serv_term.sadd(TrackedTermsNotificationTagsPrefix_Name + term.lower(),
                        ↪ tag)
467                 save_tag_to_auto_push(list_tags)
468                 # add company
469                 r_serv_term.sadd(TrackedTermsCompanyPrefix_Name + term, company)
470
471             elif action == "toggleEMailNotification":
472                 # get the current state
473                 if term in r_serv_term.smembers(TrackedTermsNotificationEnabled_Name):
474                     # remove it
475                     r_serv_term.srem(TrackedTermsNotificationEnabled_Name, term.lower())
476                 else:
477                     # add it
478                     r_serv_term.sadd(TrackedTermsNotificationEnabled_Name, term.lower())
479
480             #del action
481             else:
482                 #delete the company
483                 for company in userCompanies:
484                     r_serv_term.srem(TrackedTermsCompanyPrefix_Name + term, company)
485
486                 #delete term from DB only if no company is tracking it
487                 if not r_serv_term.smembers(TrackedTermsCompanyPrefix_Name + term):
488                     if term.startswith('/') and term.endswith('/'):
489                         r_serv_term.srem(TrackedRegexSet_Name, term)
490                         r_serv_term.hdel(TrackedRegexDate_Name, term)
491                     elif term.startswith('\\') and term.endswith('\\'):
492                         r_serv_term.srem(TrackedSetSet_Name, term)
493                         r_serv_term.hdel(TrackedSetDate_Name, term)
494                     else:
495                         r_serv_term.srem(TrackedTermsSet_Name, term.lower())
496                         r_serv_term.hdel(TrackedTermsDate_Name, term.lower())
497
498                     # delete the associated notification emails too
499                     r_serv_term.delete(TrackedTermsNotificationEmailsPrefix_Name + term)
500                     # delete the associated tags set
501                     r_serv_term.delete(TrackedTermsNotificationTagsPrefix_Name + term)
502
503         else:
504             return "None"
505
506         to_return = {}
507         to_return["section"] = section
508         to_return["action"] = action
509         to_return["term"] = term
510         return jsonify(to_return)
511
512 @app.route("/mgmtPortal/delete_terms_tags", methods=['POST'])
513 def delete_terms_tags():
514     term = request.form.get('term')
515     tags_to_delete = request.form.getlist('tags_to_delete')
516     print(term,tags_to_delete)
517
518     if term is not None and tags_to_delete is not None:
519         for tag in tags_to_delete:
520             r_serv_term.srem(TrackedTermsNotificationTagsPrefix_Name + term, tag)
521         return redirect(url_for('mgmtPortal_page'))
522     else:
523         return 'None args', 400
524
525 @app.route("/mgmtPortal/delete_terms_email", methods=['GET'])
526 def delete_terms_email():
527     term = request.args.get('term')
528     email = request.args.get('email')
529
530     if term is not None and email is not None:
531         r_serv_term.srem(TrackedTermsNotificationEmailsPrefix_Name + term, email)
532         return redirect(url_for('mgmtPortal_page'))
533     else:
```

```
534         return 'None args', 400
535
536
537 if __name__ == "__main__":
538     app.secret_key = os.urandom(12)
539 app.run(debug=True,host='0.0.0.0', port=4000)
```

A.2. identityManagement.py

```
from sqlalchemy import *
from sqlalchemy import create_engine, ForeignKey
from sqlalchemy import Column, Date, Integer, String
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import relationship, backref

engine = create_engine('sqlite:///identityManagement.db', echo=True)
Base = declarative_base()

########################################################################
class User(Base):
    """"""
    __tablename__ = "users"

    id = Column(Integer, primary_key=True)
    username = Column(String)
    password = Column(String)
    salt = Column(String)

#----------------------------------------------------------------------
def __init__(self, username, password):
    """"""
    self.username = username
    self.password = password

########################################################################
class Company(Base):
    """"""
    __tablename__ = "companies"

    id = Column(Integer, primary_key=True)
    companyName = Column(String)
    termsLimit = Column(Integer)

########################################################################
class UsersCompany(Base):
    """"""
    __tablename__ = "usersCompany"

    id = Column(Integer, primary_key=True)
    userID = Column(Integer, ForeignKey('users.id'))
    companyID = Column(Integer, ForeignKey('companies.id'))

# create tables
Base.metadata.create_all(engine)
```

A.3. login.html

```
<link rel="stylesheet" href="/static/style.css" type="text/css">

<div class="container" style="min-height:100% width:80%">
      {% with messages = get_flashed_messages() %}
        {% if messages %}
          {% for message in messages %}
              <div class="alert alert-warning alert-dismissible" role="alert">
              <button type="button" class="close" data-dismiss="alert" aria-label="Close"><
                  ↪ span aria-hidden="true">x</span></button>
                  {{message}}
              </div>
            {% endfor %}
        {% endif %}
      {% endwith %}

      {% block body %}{% endblock %}
</div>
<form action="/login" method="POST">
<div class="login">
<div class="login-screen">
<div class="app-title">
<h1>AIL Terms management portal</h1>
</div>
<div class="login-form">
<div class="control-group">
                        <input type="text" class="login-field" value="" placeholder="
                            ↪ username" name="username">
<label class="login-field-icon fui-user" for="login-name"></label></div>
<div class="control-group">
                        <input type="password" class="login-field" value="" placeholder="
                            ↪ password" name="password">
<label class="login-field-icon fui-lock" for="login-pass"></label></div>
<input type="submit" value="Log in" class="btn btn-primary btn-large btn-block">

</div>
</div>
</div>
</form>
```

A.4. mgmtPortal.html

```
<!DOCTYPE html>
<html>

<head>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">

 <title>AIL Management portal</title>
 <link rel="icon" href="{{ url_for('static', filename='image/ail-icon.png') }}">

 <!-- Core CSS -->
 <link href="{{ url_for('static', filename='css/bootstrap.min.css') }}" rel="stylesheet">
 <link href="{{ url_for('static', filename='font-awesome/css/font-awesome.css') }}" rel="
     ↪ stylesheet">
 <link href="{{ url_for('static', filename='css/sb-admin-2.css') }}" rel="stylesheet">
 <link href="{{ url_for('static', filename='css/dataTables.bootstrap.css') }}" rel="
     ↪ stylesheet" type="text/css" />
 <link href="{{ url_for('static', filename='css/switch_checkbox.css') }}" rel="stylesheet"
     ↪ type="text/css" />
 <script language="javascript" src="{{ url_for('static', filename='js/jquery.js')}}"></
     ↪ script>
 <script src="{{ url_for('static', filename='js/bootstrap.min.js') }}"></script>
 <script src="{{ url_for('static', filename='js/jquery.dataTables.min.js') }}"></script>
 <script src="{{ url_for('static', filename='js/dataTables.bootstrap.js') }}"></script>
 <script src="{{ url_for('static', filename='js/jquery.flot.js') }}"></script>
 <script src="{{ url_for('static', filename='js/jquery.flot.time.js') }}"></script>
 <script src="{{ url_for('static', filename='js/jquery.flot.stack.js') }}"></script>

 <style>
   .sparkLineStats ul {
    padding-left:0;
    list-style:none
   }
   .btn-link {
    color: #000000
   }
   .popover-content {
    white-space:pre-wrap;
    word-wrap:break-word;
   }
   .mouse_pointer{
    cursor: pointer;
   }
   .lb-md {
    font-size: 16px;
   }
 </style>
</head>
<body>
  <div class="container" style="min-height:100% width:80%">
    {% with messages = get_flashed_messages() %}
     {% if messages %}
      {% for message in messages %}
        <div class="alert alert-warning alert-dismissible" role="alert">
        <button type="button" class="close" data-dismiss="alert" aria-label="Close"><span
            ↪ aria-hidden="true">x</span></button>
          {{message}}
        </div>
       {% endfor %}
     {% endif %}
    {% endwith %}

    {% block body %}{% endblock %}
  </div>

  <!-- Modal -->
  <div id="mymodal" class="modal fade" role="dialog">
   <div class="modal-dialog modal-lg">
```

```
    <!-- Modal content-->
    <div id="mymodalcontent" class="modal-content">
     <div id="mymodalbody" class="modal-body" max-width="8500px">
      <p>Loading paste information...</p>
      <img id="loading-gif-modal" src="{{url_for('static', filename='image/loading.gif') }}"
          ↪ height="26" width="26" style="margin: 4px;">
     </div>
     <div class="modal-footer">
      <a id="button_show_plot" target="_blank" href=""><button type="button" class="btn btn-
          ↪ info">Plot term</button></a>
      <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
     </div>
    </div>
   </div>
  </div>

    <div class="navbar-default sidebar" role="navigation">
    <a href="{{ url_for('mgmtPortal_page') }}"><img src="{{ url_for('static', filename='image
        ↪ /AIL.png') }}" /></a>
    </div>

<div id="page-wrapper">
 <div class="row">
  <div class="col-lg-12">
 <div class="topnav">
     <b>{{ userName }}</b>
   {% for company in userCompanies %}
     <tr><td>{{ company }}</td></tr>
     {% endfor %}
     <a href="/logout">Logout</a>
    </div>
    <h1 class="page-header" data-page="page-termsfrequency" >Terms frequency: Management
        ↪ interface</h1>
  </div>
  <!-- /.col-lg-12 -->
 </div>
  <!-- /.row -->
  <div class="row">

  <!-- Panel OPTIONS -->
  <div class="row">
   <div class="col-lg-12">
    <div class="row">
     {% set uniq_id = namespace(modal_id=0) %}
      <div class="col-lg-12">
       <div id="panel-today" class="panel panel-success">
        <div class="panel-heading">
         <strong>Manage tracked terms</strong>
        </div>
        <div class="panel-body">

         <div style="margin-bottom: 10px;">
          <table>
           <tr><td><b>Regex</b>: surround the term by '<b>/</b>'. </td> <td><b style="
               ↪ margin-left: 20px;">/([a-z])\w+([a-z])\n/</b></td></tr>
           <tr><td><b>Set of terms</b>: surround the list by '<b>\</b>'. </td> <td><b
               ↪ style="margin-left: 20px;">\[term1, term2, ...]\</b></td></tr>
           <tr><td> - To set a custom matching <b>threshold</b> (defaut=50), append it at
               ↪  the end as a inner list '<b>[thresh]</b>'. </td> <td><b style="margin-
               ↪ left: 20px;">\[term1, term2, ..., [75]]\</b></td></tr>
    <tr><td><b>Company</b> field is mandatory and only user's companies are allowed</td></
        ↪ tr>
          </table>
         </div>
         <div class="form-group input-group" style="margin-bottom: 30px;">
          <span class="input-group-addon"><span class="fa fa-eye"></span></span>
          <input id="followTermInput" class="form-control" placeholder="Term to track."
              ↪ type="text" style="max-width: 400px;">
          <input id="followTermEMailNotificationReceiversInput" class="form-control"
              ↪ placeholder="Notification E-Mails (optional, space separated)" type="text
              ↪ " style="max-width: 400px;">
```

```
  <input id="followTermTag" class="form-control" placeholder="Tags (optional,
      ↪ space separated)" type="text" style="max-width: 400px;">
  <input id="followTermCompanyInput" class="form-control" placeholder="Company"
      ↪ type="text" style="max-width: 400px;">
  <button id="followTermBtn" class="btn btn-success btn-interaction" style="margin
      ↪ -left: 10px;" data-section="followTerm" data-action="add"> Add term</
      ↪ button>
</div>

<table class="table table-striped table-bordered table-hover" id="myTable">
  <thead>
    <tr>
     <th style="max-width: 800px;">Term</th>
     <th>Added date</th>
     <th>Day occurence</th>
     <th>Week occurence</th>
     <th>Month occurence</th>
     <th># tracked paste</th>
     <th>Action</th>
     <th>Notification E-Mails</th>
     <th>Company</th>
    </tr>
  </thead>
  <tbody>
  <!-- SET -->
  {% for set in trackSet_list %}
   <tr style="background-color: #cdffca;">
     <td>
      <span class="term_name">{{ set }}</span>
      <div>
       {% for tag in notificationTagsTermMapping[set] %}
       <span class="label label-{{ bootstrap_label[loop.index0 % 5] }} pull-left
           ↪ ">{{ tag }}</span>
       {% endfor %}
       {% if notificationTagsTermMapping[set] %}
        <div class="btn-link btn-interaction pull-right mouse_pointer" data-toggle
            ↪ ="modal" data-target="#edit_custom_tag_modal_{{ uniq_id.modal_id
            ↪ }}" data-placement="right" title="Edit Tags List"><i class="fa fa-
            ↪ pencil" style="color:Red;"></i></div>

        <div id="edit_custom_tag_modal_{{ uniq_id.modal_id }}" class="modal fade"
            ↪ role="dialog">
         <div class="modal-dialog">

          <!-- Modal content-->
          <div id="mymodalcontent" class="modal-content">
           <div class="modal-header" style="border-bottom: 4px solid #48c9b0;
               ↪ background-color: #48c9b0; color: #ffffff;">
            <h2 class="text-center">Remove Custom Tag</h2>

           </div>

           <div class="modal-body">
            <form action="{{ url_for('delete_terms_tags') }}" id="checkboxForm"
                ↪ method='post'>
             {% for tag in notificationTagsTermMapping[set] %}
              <div class="form-check">
               <input type="hidden" class="form-control" name="term" value="{{ set
                   ↪ }}">
              <input type="checkbox" class="form-check-input" name="tags_to_delete"
                  ↪  value="{{ tag }}">
              <label class="form-check-label">
               <span class="label label-{{ bootstrap_label[loop.index0 % 5] }} lb-
                   ↪ md">{{ tag }}</span>
              </label>
              <br>
             </div>
             {% endfor %}
            </form>
           </div>

           <div class="modal-footer">
```

```
                    <button class="btn btn-danger" type="submit" form="checkboxForm" value
                        ↪ ="Submit">
                      <span class="glyphicon glyphicon-trash"></span>
                      <span class="label-icon">Remove Tags</span>
                    </button>
                    <button type="button" class="btn btn-default" data-dismiss="modal" >
                        ↪ Close</button>

                   </div>
                  </div>
                 </div>
                </div>
                {% set uniq_id.modal_id = uniq_id.modal_id + 1 %}
               {% endif %}
              </div>
             </td>
             <td>{{ trackSet_list_values[loop.index0][3] }}</td>
             <td>{{ trackSet_list_values[loop.index0][0] }}</td>
             <td>{{ trackSet_list_values[loop.index0][1] }}</td>
             <td>{{ trackSet_list_values[loop.index0][2] }}</td>
             <td>{{ trackSet_list_num_of_paste[loop.index0] }}</td>
             <td><p style="margin: 0px; white-space: nowrap;">
              <button class="btn-link btn-interaction" data-toggle="tooltip" data-
                  ↪ placement="left" title="Remove this term" data-content="{{ set }}"
                  ↪ data-section="followTerm" data-action="delete"><span class="glyphicon
                  ↪  glyphicon-trash"></span></button>
                 <input id="checkBoxEMailAlerts" type="checkbox" title="Toggle E
                  ↪ -Mail notifications" class="btn-link btn-interaction" data-content
                  ↪ ="{{ set }}" data-section="followTerm" data-action="
                  ↪ toggleEMailNotification" {% if notificationEnabledDict[set] %}
                  ↪ checked {% endif %}>
             </p></td>
             <td>
              {% for email in notificationEMailTermMapping[set] %}
               <a href="{{ url_for('delete_terms_email') }}?email={{email}}&term={{set}}">
                <div class="btn-link btn-interaction pull-right mouse_pointer" data-toggle
                    ↪ ="tooltip" data-placement="left" data-original-title="Remove this
                    ↪ email">
                 <span class="glyphicon glyphicon-trash" style="color:Red;" ></span>
                </div>
               </a>
               {{ email }}
               <br>
              {% endfor %}
             </td>
             <td>
             {% for companyTermMapping_Company in companyTermMapping[set] %}
             {{ companyTermMapping_Company }}
             <br>
             {% endfor %}
</td>
         </tr>
        {% endfor %}
        <!-- REGEX -->
        {% for regex in trackReg_list %}
         <tr style="background-color: #fffdca;">
          <td>
           <span class="term_name">{{ regex }}</span>
           <div>
{% for tag in notificationTagsTermMapping[regex] %}
            <span class="label label-{{ bootstrap_label[loop.index0 % 5] }} pull-left
                ↪ ">{{ tag }}</span>
            {% endfor %}
            {% if notificationTagsTermMapping[regex] %}
             <div class="btn-link btn-interaction pull-right mouse_pointer" data-toggle
                 ↪ ="modal" data-target="#edit_custom_tag_modal_{{ uniq_id.modal_id
                 ↪ }}" data-placement="right" title="Edit Tags List"><i class="fa fa-
                 ↪ pencil" style="color:Red;"></i></div>

             <div id="edit_custom_tag_modal_{{ uniq_id.modal_id }}" class="modal fade"
                 ↪ role="dialog">
              <div class="modal-dialog">
```

```
                <!-- Modal content-->
                <div id="mymodalcontent" class="modal-content">
                 <div class="modal-header" style="border-bottom: 4px solid #48c9b0;
                     ↪ background-color: #48c9b0; color: #ffffff;">
                  <h2 class="text-center">Remove Custom Tag</h2>

                 </div>

                 <div class="modal-body">
                  <form action="{{ url_for('delete_terms_tags') }}" id="checkboxForm"
                      ↪ method='post'>
                   {% for tag in notificationTagsTermMapping[regex] %}
                    <div class="form-check">
                     <input type="hidden" class="form-control" name="term" value="{{
                         ↪ regex }}">
                     <input type="checkbox" class="form-check-input" name="tags_to_delete"
                         ↪  value="{{ tag }}">
                     <label class="form-check-label">
                      <span class="label label-{{ bootstrap_label[loop.index0 % 5] }} lb-
                          ↪ md">{{ tag }}</span>
                     </label>
                     <br>
                    </div>
                   {% endfor %}
                  </form>
                 </div>

                 <div class="modal-footer">
                  <button class="btn btn-danger" type="submit" form="checkboxForm" value
                      ↪ ="Submit">
                   <span class="glyphicon glyphicon-trash"></span>
                   <span class="label-icon">Remove Tags</span>
                  </button>
                  <button type="button" class="btn btn-default" data-dismiss="modal" >
                      ↪ Close</button>

                 </div>
                </div>
               </div>
              </div>
              {% set uniq_id.modal_id = uniq_id.modal_id + 1 %}
             {% endif %}
            </div>
           </td>
           <td>{{ trackReg_list_values[loop.index0][3] }}</td>
           <td>{{ trackReg_list_values[loop.index0][0] }}</td>
           <td>{{ trackReg_list_values[loop.index0][1] }}</td>
           <td>{{ trackReg_list_values[loop.index0][2] }}</td>
           <td>{{ trackReg_list_num_of_paste[loop.index0] }}</td>
           <td><p style="margin: 0px; white-space: nowrap;">
            <button class="btn-link btn-interaction" data-toggle="tooltip" data-
                ↪ placement="left" title="Remove this term" data-content="{{ regex }}"
                ↪ data-section="followTerm" data-action="delete"><span class="glyphicon
                ↪  glyphicon-trash"></span></button>
               <input id="checkBoxEMailAlerts" type="checkbox" title="Toggle E
                ↪ -Mail notifications" class="btn-link btn-interaction" data-content
                ↪ ="{{ regex }}" data-section="followTerm" data-action="
                ↪ toggleEMailNotification" {% if notificationEnabledDict[regex] %}
                ↪ checked {% endif %}>
           </p></td>
           <td>
            {% for email in notificationEMailTermMapping[regex] %}
             <a href="{{ url_for('delete_terms_email') }}?email={{email}}&term={{regex
                 ↪ }}">
              <div class="btn-link btn-interaction pull-right mouse_pointer" data-toggle
                  ↪ ="tooltip" data-placement="left" data-original-title="Remove this
                  ↪ email">
               <span class="glyphicon glyphicon-trash" style="color:Red;"></span>
              </div>
             </a>
             {{ email }}
```

```
                <br>
               {% endfor %}
              </td>
        <td>
        {% for companyTermMapping_Company in companyTermMapping[regex] %}
                {{ companyTermMapping_Company }}
        <br>
        {% endfor %}
        </td>
              </tr>
             {% endfor %}
             <!-- Normal term -->
             {% for term in track_list %}
              <tr>
                <td>
                 <span class="term_name">{{ term }}</span>
                 <div>
                 {% for tag in notificationTagsTermMapping[term] %}
                  <span class="label label-{{ bootstrap_label[loop.index0 % 5] }} pull-left
                      ↪ ">{{ tag }}</span>
                  {% endfor %}
                  {% if notificationTagsTermMapping[term] %}
                   <div class="btn-link btn-interaction pull-right mouse_pointer" data-toggle
                       ↪ ="modal" data-target="#edit_custom_tag_modal_{{ uniq_id.modal_id
                       ↪ }}" data-placement="right" title="Edit Tags List"><i class="fa fa-
                       ↪ pencil" style="color:Red;"></i></div>

                   <div id="edit_custom_tag_modal_{{ uniq_id.modal_id }}" class="modal fade"
                       ↪ role="dialog">
                    <div class="modal-dialog">

                     <!-- Modal content-->
                     <div id="mymodalcontent" class="modal-content">
                      <div class="modal-header" style="border-bottom: 4px solid #48c9b0;
                          ↪ background-color: #48c9b0; color: #ffffff;">
                       <h2 class="text-center">Remove Custom Tag</h2>

                      </div>

                      <div class="modal-body">
                       <form action="{{ url_for('delete_terms_tags') }}" id="checkboxForm"
                           ↪ method='post'>
                        {% for tag in notificationTagsTermMapping[term] %}
                         <div class="form-check">
                          <input type="hidden" class="form-control" name="term" value="{{ term
                              ↪  }}">
                          <input type="checkbox" class="form-check-input" name="tags_to_delete"
                              ↪  value="{{ tag }}">
                          <label class="form-check-label">
                           <span class="label label-{{ bootstrap_label[loop.index0 % 5] }} lb-
                               ↪ md">{{ tag }}</span>
                          </label>
                          <br>
                         </div>
                        {% endfor %}
                       </form>
                      </div>

                      <div class="modal-footer">
                       <button class="btn btn-danger" type="submit" form="checkboxForm" value
                           ↪ ="Submit">
                        <span class="glyphicon glyphicon-trash"></span>
                        <span class="label-icon">Remove Tags</span>
                       </button>
                       <button type="button" class="btn btn-default" data-dismiss="modal" >
                           ↪ Close</button>

                      </div>
                     </div>
                    </div>
                   </div>
                   {% set uniq_id.modal_id = uniq_id.modal_id + 1 %}
```

```
                {% endif %}
               </div>
              </td>
              <td>{{ track_list_values[loop.index0][3] }}</td>
              <td>{{ track_list_values[loop.index0][0] }}</td>
              <td>{{ track_list_values[loop.index0][1] }}</td>
              <td>{{ track_list_values[loop.index0][2] }}</td>
              <td>{{ track_list_num_of_paste[loop.index0] }}</td>
              <td><p style="margin: 0px; white-space: nowrap;">
               <button class="btn-link btn-interaction" data-toggle="tooltip" data-
                   ↪ placement="left" title="Remove this term" data-content="{{ term }}"
                   ↪ data-section="followTerm" data-action="delete"><span class="glyphicon
                   ↪  glyphicon-trash"></span></button>
                  <input id="checkBoxEMailAlerts" type="checkbox" title="Toggle E
                   ↪ -Mail notifications" class="btn-link btn-interaction" data-content
                   ↪ ="{{ term }}" data-section="followTerm" data-action="
                   ↪ toggleEMailNotification" {% if notificationEnabledDict[term] %}
                   ↪ checked {% endif %}>
              </p></td>
              <td>
               {% for email in notificationEMailTermMapping[term] %}
                <a href="{{ url_for('delete_terms_email') }}?email={{email}}&term={{term}}">
                 <div class="btn-link btn-interaction pull-right mouse_pointer" data-toggle
                     ↪ ="tooltip" data-placement="left" data-original-title="Remove this
                     ↪ email">
                  <span class="glyphicon glyphicon-trash" style="color:Red;"></span>
                 </div>
                </a>
                {{ email }}
                <br>
               {% endfor %}
              </td>
              <td>
              {% for companyTermMapping_Company in companyTermMapping[term] %}
              {{ companyTermMapping_Company }}
              <br>
              {% endfor %}
             </td>
            </tr>
           {% endfor %}
           </tbody>
          </table>
         <!-- /.panel-body -->
        </div>
       </div>
       <!-- /.panel -->
     </div>
     <!-- /.panel -->
   </div>
  </div>

  <!-- Panel OPTIONS -->
   <div class="col-lg-12">
    <div class="row">
     <div class="col-lg-12">
      <div id="panel-today" class="panel panel-danger">
       <div class="panel-heading">
        <strong>Blacklisted terms</strong>
       </div>
       <div class="panel-body">


        <table class="table table-striped table-bordered table-hover" id="myTable2">
         <thead>
          <tr>
           <th style="max-width: 800px;">Term</th>
           <th>Added date</th>
          </tr>
         </thead>
         <tbody>
         {% for term, date in black_list %}
          <tr>
```

```
                <td>{{ term }}</td>
                <td>{{ date }}</td>
              </tr>
             {% endfor %}
             </tbody>
           </table>
         <!-- /.panel-body -->
        </div>
       </div>
       <!-- /.panel -->
     </div>
     <!-- /.panel -->
   </div>
  </div>

  <!-- /.row -->
  </div>
<!-- /#page-wrapper -->
</div>


<!-- import graph function -->
<script>
  function reload_per_paste() {
    window.location.href = {{ url_for('mgmtPortal_page') }};
  }
  var table_track;
  var table_black;
function bindEventsForCurrentPage() {
  // On click, get html content from url and update the corresponding modal
  $("[data-toggle='modal']").unbind().on("click.openmodal", function (event) {
    //console.log(data);
    event.preventDefault();
    var the_modal=$(this);
    var url = "{{ url_for('mgmtPortal_query_paste') }}?term=" + encodeURIComponent($(this).
        ↪ attr('data-term'));
    $.getJSON(url, function (data) {
      if (data.length != 0) {
        var html_to_add = "";
        html_to_add += "<table id=\"modal-table\" class=\"table table-striped\">";
        html_to_add += "<thead>";
        html_to_add += "<tr>";
        html_to_add += "<th>Source</th>";
        html_to_add += "<th>Date</th>";
        html_to_add += "<th>Encoding</th>";
        html_to_add += "<th>Size (Kb)</th>";
        html_to_add += "<th># lines</th>";
        html_to_add += "<th>Max length</th>";
        html_to_add += "<th>Preview</th>";
        html_to_add += "</tr>";
        html_to_add += "</thead>";
        html_to_add += "<tbody>";
        for (i=0; i<data.length; i++) {
          curr_data = data[i];
          html_to_add += "<tr>";
          html_to_add += "<td>"+curr_data.source+"</td>";
          html_to_add += "<td>"+curr_data.date+"</td>";
          html_to_add += "<td>"+curr_data.encoding+"</td>";
          html_to_add += "<td>"+curr_data.size+"</td>";
          html_to_add += "<td>"+curr_data.lineinfo[0]+"</td>";
          html_to_add += "<td>"+curr_data.lineinfo[1]+"</td>";
          html_to_add += "</tr>";
        }
        html_to_add += "</tbody>";
        html_to_add += "</table>";
        $("#mymodalbody").html(html_to_add);
        $("[data-toggle=popover]").popover();
        $('#modal-table').DataTable();
      } else {
        $("#mymodalbody").html("No paste containing this term has been received yet.");
      }
    });
```

```
  });
}
  $(document).ready(function(){
    bindEventsForCurrentPage();
    activePage = $('h1.page-header').attr('data-page');
    $("#"+activePage).addClass("active");
    if({{ per_paste }} == 1) {
      $("#per_paste").attr('checked', true)
    }
    $('[data-toggle="tooltip"]').tooltip();
 table_track = $('#myTable').DataTable();
 table_black = $('#myTable2').DataTable();
    table_track.on( 'draw.dt', function () {
      perform_binding();
    });
    table_black.on( 'draw.dt', function () {
      perform_binding();
    });
    $("#followTermInput").keyup(function(event){
      if(event.keyCode == 13){
        $("#followTermBtn").click();
        $("#followTermInput").val("");
      }
    });
    $("#blacklistTermInput").keyup(function(event){
      if(event.keyCode == 13){
        $("#blacklistTermBtn").click();
        $("#blacklistTermInput").val("");
      }
    });
    perform_binding();
  $("#mymodal").on('hidden.bs.modal', function () {
    $("#mymodalbody").html("<p>Loading paste information...</p>");
    var loading_gif = "<img id='loading-gif-modal' class='img-center' src=\"{{url_for('static
        ↪ ', filename='image/loading.gif') }}\" height='26' width='26' style='margin: 4px
        ↪ ;'>";
    $("#mymodalbody").append(loading_gif); // Show the loading GIF
  });
  });
</script>

<script>
function perform_binding() {
  $(".btn-interaction").unbind("click.interaction");
  $(".btn-interaction").bind("click.interaction", perform_operation);
}
function perform_operation(){
  var curr_section = $(this).attr('data-section');
  var curr_action = $(this).attr('data-action');
  var row_tr = $(this).closest("tr");
  if (curr_action == "add") {
    var curr_term = $('#'+curr_section+'Input').val();
    var email_addresses = $('#followTermEMailNotificationReceiversInput').val();
    var tags = $('#followTermTag').val();
 var company = $('#followTermCompanyInput').val();
  } else {
    var curr_term = $(this).attr('data-content');
    var email_addresses = "";
  }
  var data_to_send = { section: curr_section, action: curr_action, term: curr_term,
      ↪ emailAddresses: email_addresses, tags: tags, company: company};
  if (curr_term != "") {
    console.log(data_to_send);
    $.get("{{ url_for('mgmtPortal_action') }}", data_to_send, function(data, status){
      if(status == "success") {
        var json = data;
        if(json.section == "followTerm") {
          if(json.action == "add") {
            // query data
            $.get("{{ url_for('mgmtPortal_query') }}", { term: json.term, section: json.section
                ↪ }, function(data2, status){
              reload_per_paste();
```

```
          });
        } else if (json.action == "delete") {
         row_tr.remove()
        }
      } else if(json.section == "blacklistTerm"){
        if(json.action == "add") {
          $.get("{{ url_for('mgmtPortal_query') }}", { term: json.term, section: json.section
              ↪ }, function(data2, status){
            console.log(data2);
            var action_button = "<button class=\"btn-link btn-interaction\" data-toggle=\"
                ↪ tooltip\" data-placement=\"right\" title=\"Remove this term\" data-content
                ↪ =\"" + json.term + "\" data-section=\"blacklistTerm\" data-action=\"delete
                ↪ \"><span class=\"glyphicon glyphicon-trash\"></span></button>"
            table_black.row.add( [ json.term, data2[3], action_button ] ).draw( false );
            perform_binding();
          });
        } else if (json.action == "delete") {
          // Find indexes of row which have the term in the first column
          var index = table_black.rows().eq( 0 ).filter( function (rowIdx) {
            return table_black.cell( rowIdx, 0 ).data() === json.term;
          } );
          table_black.rows(index).remove().draw( false );
        }
      }
    }
  });
 }
}
</script>
```

A.5. Flask_terms.py adaptado

```
#!/usr/bin/env python3
# -*-coding:UTF-8 -*

'''
   Flask functions and routes for the trending modules page

   note: The matching of credential against supplied credential is done using Levenshtein
       ↪ distance
'''
import redis
import datetime
import calendar
import flask
from flask import Flask, render_template, jsonify, request, Blueprint, url_for, redirect
import re
import Paste
from pprint import pprint
import Levenshtein

# ============ VARIABLES ============
import Flask_config

app = Flask_config.app
cfg = Flask_config.cfg
baseUrl = Flask_config.baseUrl
r_serv_term = Flask_config.r_serv_term
r_serv_cred = Flask_config.r_serv_cred
r_serv_db = Flask_config.r_serv_db
bootstrap_label = Flask_config.bootstrap_label

terms = Blueprint('terms', __name__, template_folder='templates')

'''TERM'''
DEFAULT_MATCH_PERCENT = 50

#tracked
TrackedTermsSet_Name = "TrackedSetTermSet"
TrackedTermsDate_Name = "TrackedTermDate"
#black
BlackListTermsDate_Name = "BlackListTermDate"
BlackListTermsSet_Name = "BlackListSetTermSet"
#regex
TrackedRegexSet_Name = "TrackedRegexSet"
TrackedRegexDate_Name = "TrackedRegexDate"
#set
TrackedSetSet_Name = "TrackedSetSet"
TrackedSetDate_Name = "TrackedSetDate"

# notifications enabled/disabled
# same value as in `bin/NotificationHelper.py`
TrackedTermsNotificationEnabled_Name = "TrackedNotifications"

# associated notification email addresses for a specific term`
# same value as in `bin/NotificationHelper.py`
# Keys will be e.g. TrackedNotificationEmails_<TERMNAME>
TrackedTermsNotificationEmailsPrefix_Name = "TrackedNotificationEmails_"
TrackedTermsNotificationTagsPrefix_Name = "TrackedNotificationTags_"
TrackedTermsCompanyPrefix_Name = "company_"

'''CRED'''
REGEX_CRED = '[a-z]+|[A-Z]{3,}|[A-Z]{1,2}[a-z]+|[0-9]+'
REDIS_KEY_NUM_USERNAME = 'uniqNumForUsername'
REDIS_KEY_NUM_PATH = 'uniqNumForUsername'
REDIS_KEY_ALL_CRED_SET = 'AllCredentials'
REDIS_KEY_ALL_CRED_SET_REV = 'AllCredentialsRev'
REDIS_KEY_ALL_PATH_SET = 'AllPath'
REDIS_KEY_ALL_PATH_SET_REV = 'AllPathRev'
REDIS_KEY_MAP_CRED_TO_PATH = 'CredToPathMapping'

```

```
69
70
71  # ============ FUNCTIONS ============
72
73  def Term_getValueOverRange(word, startDate, num_day, per_paste=""):
74     passed_days = 0
75     oneDay = 60*60*24
76     to_return = []
77     curr_to_return = 0
78     for timestamp in range(startDate, startDate - max(num_day)*oneDay, -oneDay):
79        value = r_serv_term.hget(per_paste+str(timestamp), word)
80        curr_to_return += int(value) if value is not None else 0
81        for i in num_day:
82           if passed_days == i-1:
83              to_return.append(curr_to_return)
84        passed_days += 1
85     return to_return
86
87  #Mix suplied username, if extensive is set, slice username(s) with different windows
88  def mixUserName(supplied, extensive=False):
89     #e.g.: John Smith
90     terms = supplied.split()[:2]
91     usernames = []
92     if len(terms) == 1:
93        terms.append(' ')
94
95     #john, smith, John, Smith, JOHN, SMITH
96     usernames += [terms[0].lower()]
97     usernames += [terms[1].lower()]
98     usernames += [terms[0][0].upper() + terms[0][1:].lower()]
99     usernames += [terms[1][0].upper() + terms[1][1:].lower()]
100    usernames += [terms[0].upper()]
101    usernames += [terms[1].upper()]
102
103    #johnsmith, smithjohn, JOHNsmith, johnSMITH, SMITHjohn, smithJOHN
104    usernames += [(terms[0].lower() + terms[1].lower()).strip()]
105    usernames += [(terms[1].lower() + terms[0].lower()).strip()]
106    usernames += [(terms[0].upper() + terms[1].lower()).strip()]
107    usernames += [(terms[0].lower() + terms[1].upper()).strip()]
108    usernames += [(terms[1].upper() + terms[0].lower()).strip()]
109    usernames += [(terms[1].lower() + terms[0].upper()).strip()]
110    #Jsmith, JSmith, jsmith, jSmith, johnS, Js, JohnSmith, Johnsmith, johnSmith
111    usernames += [(terms[0][0].upper() + terms[1][0].lower() + terms[1][1:].lower()).strip()]
112    usernames += [(terms[0][0].upper() + terms[1][0].upper() + terms[1][1:].lower()).strip()]
113    usernames += [(terms[0][0].lower() + terms[1][0].lower() + terms[1][1:].lower()).strip()]
114    usernames += [(terms[0][0].lower() + terms[1][0].upper() + terms[1][1:].lower()).strip()]
115    usernames += [(terms[0].lower() + terms[1][0].upper()).strip()]
116    usernames += [(terms[0].upper() + terms[1][0].lower()).strip()]
117    usernames += [(terms[0][0].upper() + terms[0][1:].lower() + terms[1][0].upper() + terms
           ↪ [1][1:].lower()).strip()]
118    usernames += [(terms[0][0].upper() + terms[0][1:].lower() + terms[1][0].lower() + terms
           ↪ [1][1:].lower()).strip()]
119    usernames += [(terms[0][0].lower() + terms[0][1:].lower() + terms[1][0].upper() + terms
           ↪ [1][1:].lower()).strip()]
120
121    if not extensive:
122       return usernames
123
124    #Slice the supplied username(s)
125    mixedSupplied = supplied.replace(' ','')
126    minWindow = 3 if len(mixedSupplied)/2 < 4 else len(mixedSupplied)/2
127    for winSize in range(3,len(mixedSupplied)):
128       for startIndex in range(0, len(mixedSupplied)-winSize):
129          usernames += [mixedSupplied[startIndex:startIndex+winSize]]
130
131    filtered_usernames = []
132    for usr in usernames:
133       if len(usr) > 2:
134          filtered_usernames.append(usr)
135    return filtered_usernames
136
137 def save_tag_to_auto_push(list_tag):
```

```
138     for tag in set(list_tag):
139         #limit tag length
140         if len(tag) > 49:
141             tag = tag[0:48]
142         r_serv_db.sadd('list_export_tags', tag)
143 
144 # ============ ROUTES ============
145 
146 @terms.route("/terms_management/")
147 def terms_management():
148     per_paste = request.args.get('per_paste')
149     if per_paste == "1" or per_paste is None:
150         per_paste_text = "per_paste_"
151         per_paste = 1
152     else:
153         per_paste_text = ""
154         per_paste = 0
155 
156     today = datetime.datetime.now()
157     today = today.replace(hour=0, minute=0, second=0, microsecond=0)
158     today_timestamp = calendar.timegm(today.timetuple())
159 
160     # Map tracking if notifications are enabled for a specific term
161     notificationEnabledDict = {}
162 
163     # Maps a specific term to the associated email addresses
164     notificationEMailTermMapping = {}
165     notificationTagsTermMapping = {}
166 
167     # Maps a specific term to the associated company
168     companyTermMapping = {}
169     #Regex
170     trackReg_list = []
171     trackReg_list_values = []
172     trackReg_list_num_of_paste = []
173     for tracked_regex in r_serv_term.smembers(TrackedRegexSet_Name):
174 
175         notificationEMailTermMapping[tracked_regex] = r_serv_term.smembers(
            ↪ TrackedTermsNotificationEmailsPrefix_Name + tracked_regex)
176         notificationTagsTermMapping[tracked_regex] = r_serv_term.smembers(
            ↪ TrackedTermsNotificationTagsPrefix_Name + tracked_regex)
177         companyTermMapping[tracked_regex] = r_serv_term.smembers(
            ↪ TrackedTermsCompanyPrefix_Name + tracked_regex)
178 
179         if tracked_regex not in notificationEnabledDict:
180             notificationEnabledDict[tracked_regex] = False
181 
182         trackReg_list.append(tracked_regex)
183         value_range = Term_getValueOverRange(tracked_regex, today_timestamp, [1, 7, 31],
            ↪ per_paste=per_paste_text)
184 
185         term_date = r_serv_term.hget(TrackedRegexDate_Name, tracked_regex)
186 
187         set_paste_name = "regex_" + tracked_regex
188         trackReg_list_num_of_paste.append(r_serv_term.scard(set_paste_name))
189         term_date = datetime.datetime.utcfromtimestamp(int(term_date)) if term_date is not
            ↪ None else "No date recorded"
190         value_range.append(term_date)
191         trackReg_list_values.append(value_range)
192 
193         if tracked_regex in r_serv_term.smembers(TrackedTermsNotificationEnabled_Name):
194             notificationEnabledDict[tracked_regex] = True
195 
196     #Set
197     trackSet_list = []
198     trackSet_list_values = []
199     trackSet_list_num_of_paste = []
200     for tracked_set in r_serv_term.smembers(TrackedSetSet_Name):
201         tracked_set = tracked_set
202 
203         notificationEMailTermMapping[tracked_set] = r_serv_term.smembers(
            ↪ TrackedTermsNotificationEmailsPrefix_Name + tracked_set)
```

```
204         notificationTagsTermMapping[tracked_set] = r_serv_term.smembers(
                ↪ TrackedTermsNotificationTagsPrefix_Name + tracked_set)
205         companyTermMapping[tracked_set] = r_serv_term.smembers(TrackedTermsCompanyPrefix_Name
                ↪ + tracked_set)
206
207         if tracked_set not in notificationEnabledDict:
208            notificationEnabledDict[tracked_set] = False
209
210         trackSet_list.append(tracked_set)
211         value_range = Term_getValueOverRange(tracked_set, today_timestamp, [1, 7, 31],
                ↪ per_paste=per_paste_text)
212
213         term_date = r_serv_term.hget(TrackedSetDate_Name, tracked_set)
214
215         set_paste_name = "set_" + tracked_set
216         trackSet_list_num_of_paste.append(r_serv_term.scard(set_paste_name))
217         term_date = datetime.datetime.utcfromtimestamp(int(term_date)) if term_date is not
                ↪ None else "No date recorded"
218         value_range.append(term_date)
219         trackSet_list_values.append(value_range)
220
221         if tracked_set in r_serv_term.smembers(TrackedTermsNotificationEnabled_Name):
222            notificationEnabledDict[tracked_set] = True
223
224     #Tracked terms
225     track_list = []
226     track_list_values = []
227     track_list_num_of_paste = []
228     for tracked_term in r_serv_term.smembers(TrackedTermsSet_Name):
229
230        notificationEMailTermMapping[tracked_term] = r_serv_term.smembers(
               ↪ TrackedTermsNotificationEmailsPrefix_Name + tracked_term)
231        notificationTagsTermMapping[tracked_term] = r_serv_term.smembers(
               ↪ TrackedTermsNotificationTagsPrefix_Name + tracked_term)
232        companyTermMapping[tracked_term] = r_serv_term.smembers(TrackedTermsCompanyPrefix_Name
               ↪  + tracked_term)
233
234        if tracked_term not in notificationEnabledDict:
235           notificationEnabledDict[tracked_term] = False
236
237        track_list.append(tracked_term)
238        value_range = Term_getValueOverRange(tracked_term, today_timestamp, [1, 7, 31],
               ↪ per_paste=per_paste_text)
239
240        term_date = r_serv_term.hget(TrackedTermsDate_Name, tracked_term)
241
242        set_paste_name = "tracked_" + tracked_term
243
244        track_list_num_of_paste.append( r_serv_term.scard(set_paste_name) )
245
246        term_date = datetime.datetime.utcfromtimestamp(int(term_date)) if term_date is not
               ↪ None else "No date recorded"
247        value_range.append(term_date)
248        track_list_values.append(value_range)
249
250        if tracked_term in r_serv_term.smembers(TrackedTermsNotificationEnabled_Name):
251           notificationEnabledDict[tracked_term] = True
252
253     #blacklist terms
254     black_list = []
255     for blacked_term in r_serv_term.smembers(BlackListTermsSet_Name):
256        term_date = r_serv_term.hget(BlackListTermsDate_Name, blacked_term)
257        term_date = datetime.datetime.utcfromtimestamp(int(term_date)) if term_date is not
               ↪ None else "No date recorded"
258        black_list.append([blacked_term, term_date])
259
260     return render_template("terms_management.html",
261           black_list=black_list, track_list=track_list, trackReg_list=trackReg_list,
                  ↪ trackSet_list=trackSet_list,
262           track_list_values=track_list_values, track_list_num_of_paste=
                  ↪ track_list_num_of_paste,
```

```
263           trackReg_list_values=trackReg_list_values, trackReg_list_num_of_paste=
                  ↪ trackReg_list_num_of_paste,
264           trackSet_list_values=trackSet_list_values, trackSet_list_num_of_paste=
                  ↪ trackSet_list_num_of_paste,
265           per_paste=per_paste, notificationEnabledDict=notificationEnabledDict,
                  ↪ bootstrap_label=bootstrap_label,
266           notificationEMailTermMapping=notificationEMailTermMapping,
                  ↪ notificationTagsTermMapping=notificationTagsTermMapping, companyTermMapping=
                  ↪ companyTermMapping)
267
268
269   @terms.route("/terms_management_query_paste/")
270   def terms_management_query_paste():
271       term = request.args.get('term')
272       paste_info = []
273
274       # check if regex or not
275       if term.startswith('/') and term.endswith('/'):
276           set_paste_name = "regex_" + term
277           track_list_path = r_serv_term.smembers(set_paste_name)
278       elif term.startswith('\\') and term.endswith('\\'):
279           set_paste_name = "set_" + term
280           track_list_path = r_serv_term.smembers(set_paste_name)
281       else:
282           set_paste_name = "tracked_" + term
283           track_list_path = r_serv_term.smembers(set_paste_name)
284
285       for path in track_list_path:
286           paste = Paste.Paste(path)
287           p_date = str(paste._get_p_date())
288           p_date = p_date[0:4]+'/'+p_date[4:6]+'/'+p_date[6:8]
289           p_source = paste.p_source
290           p_encoding = paste._get_p_encoding()
291           p_size = paste.p_size
292           p_mime = paste.p_mime
293           p_lineinfo = paste.get_lines_info()
294           p_content = paste.get_p_content()
295           if p_content != 0:
296               p_content = p_content[0:400]
297           paste_info.append({"path": path, "date": p_date, "source": p_source, "encoding":
                  ↪ p_encoding, "size": p_size, "mime": p_mime, "lineinfo": p_lineinfo, "content":
                  ↪ p_content})
298
299       return jsonify(paste_info)
300
301
302   @terms.route("/terms_management_query/")
303   def terms_management_query():
304       TrackedTermsDate_Name = "TrackedTermDate"
305       BlackListTermsDate_Name = "BlackListTermDate"
306       term = request.args.get('term')
307       section = request.args.get('section')
308
309       today = datetime.datetime.now()
310       today = today.replace(hour=0, minute=0, second=0, microsecond=0)
311       today_timestamp = calendar.timegm(today.timetuple())
312       value_range = Term_getValueOverRange(term, today_timestamp, [1, 7, 31])
313
314       if section == "followTerm":
315           term_date = r_serv_term.hget(TrackedTermsDate_Name, term)
316       elif section == "blacklistTerm":
317           term_date = r_serv_term.hget(BlackListTermsDate_Name, term)
318
319       term_date = datetime.datetime.utcfromtimestamp(int(term_date)) if term_date is not None
              ↪ else "No date recorded"
320       value_range.append(str(term_date))
321       return jsonify(value_range)
322
323
324   @terms.route("/terms_management_action/", methods=['GET'])
325   def terms_management_action():
326       today = datetime.datetime.now()
```

```
327     today = today.replace(microsecond=0)
328     today_timestamp = calendar.timegm(today.timetuple())
329 
330 
331     section = request.args.get('section')
332     action = request.args.get('action')
333     term = request.args.get('term')
334     notificationEmailsParam = request.args.get('emailAddresses')
335     input_tags = request.args.get('tags')
336     company = request.args.get('company')
337 
338     if action is None or term is None or notificationEmailsParam is None:
339        return "None"
340     else:
341        if section == "followTerm":
342           if action == "add":
343 
344              #company field is mandatory
345              if not company:
346                 return "None"
347              # Make a list of all passed email addresses
348              notificationEmails = notificationEmailsParam.split()
349 
350              validNotificationEmails = []
351              # check for valid email addresses
352              for email in notificationEmails:
353                 # Really basic validation:
354                 # has exactly one @ sign, and at least one . in the part after the @
355                 if re.match(r"[^@]+@[^@]+\.[^@]+", email):
356                    validNotificationEmails.append(email)
357 
358              # create tags list
359              list_tags = input_tags.split()
360 
361              # check if regex/set or simple term
362              #regex
363              if term.startswith('/') and term.endswith('/'):
364                 r_serv_term.sadd(TrackedRegexSet_Name, term)
365                 r_serv_term.hset(TrackedRegexDate_Name, term, today_timestamp)
366                 # add all valid emails to the set
367                 for email in validNotificationEmails:
368                    r_serv_term.sadd(TrackedTermsNotificationEmailsPrefix_Name + term, email)
369                 # enable notifications by default
370                 r_serv_term.sadd(TrackedTermsNotificationEnabled_Name, term)
371                 # add tags list
372                 for tag in list_tags:
373                    r_serv_term.sadd(TrackedTermsNotificationTagsPrefix_Name + term, tag)
374                 save_tag_to_auto_push(list_tags)
375                 # add company
376                 r_serv_term.sadd(TrackedTermsCompanyPrefix_Name + term, company)
377 
378              #set
379              elif term.startswith('\\') and term.endswith('\\'):
380                 tab_term = term[1:-1]
381                 perc_finder = re.compile("\[[0-9]{1,3}\]").search(tab_term)
382                 if perc_finder is not None:
383                    match_percent = perc_finder.group(0)[1:-1]
384                    set_to_add = term
385                 else:
386                    match_percent = DEFAULT_MATCH_PERCENT
387                    set_to_add = "\\" + tab_term[:-1] + ", [{}]]\\".format(match_percent)
388                 r_serv_term.sadd(TrackedSetSet_Name, set_to_add)
389                 r_serv_term.hset(TrackedSetDate_Name, set_to_add, today_timestamp)
390                 # add all valid emails to the set
391                 for email in validNotificationEmails:
392                    r_serv_term.sadd(TrackedTermsNotificationEmailsPrefix_Name + set_to_add,
                         ↪ email)
393                 # enable notifications by default
394                 r_serv_term.sadd(TrackedTermsNotificationEnabled_Name, set_to_add)
395                 # add tags list
396                 for tag in list_tags:
```

```
397                     r_serv_term.sadd(TrackedTermsNotificationTagsPrefix_Name + set_to_add, tag
                            ↪ )
398                 save_tag_to_auto_push(list_tags)
399                 # add company
400                 r_serv_term.sadd(TrackedTermsCompanyPrefix_Name + term, company)
401
402             #simple term
403             else:
404                 r_serv_term.sadd(TrackedTermsSet_Name, term.lower())
405                 r_serv_term.hset(TrackedTermsDate_Name, term.lower(), today_timestamp)
406                 # add all valid emails to the set
407                 for email in validNotificationEmails:
408                     r_serv_term.sadd(TrackedTermsNotificationEmailsPrefix_Name + term.lower(),
                            ↪  email)
409                 # enable notifications by default
410                 r_serv_term.sadd(TrackedTermsNotificationEnabled_Name, term.lower())
411                 # add tags list
412                 for tag in list_tags:
413                     r_serv_term.sadd(TrackedTermsNotificationTagsPrefix_Name + term.lower(),
                            ↪ tag)
414                 save_tag_to_auto_push(list_tags)
415                 # add company
416                 r_serv_term.sadd(TrackedTermsCompanyPrefix_Name + term, company)
417
418         elif action == "toggleEMailNotification":
419             # get the current state
420             if term in r_serv_term.smembers(TrackedTermsNotificationEnabled_Name):
421                 # remove it
422                 r_serv_term.srem(TrackedTermsNotificationEnabled_Name, term.lower())
423             else:
424                 # add it
425                 r_serv_term.sadd(TrackedTermsNotificationEnabled_Name, term.lower())
426
427         #del action
428         else:
429             if term.startswith('/') and term.endswith('/'):
430                 r_serv_term.srem(TrackedRegexSet_Name, term)
431                 r_serv_term.hdel(TrackedRegexDate_Name, term)
432             elif term.startswith('\\') and term.endswith('\\'):
433                 r_serv_term.srem(TrackedSetSet_Name, term)
434                 r_serv_term.hdel(TrackedSetDate_Name, term)
435             else:
436                 r_serv_term.srem(TrackedTermsSet_Name, term.lower())
437                 r_serv_term.hdel(TrackedTermsDate_Name, term.lower())
438
439             # delete the associated notification emails too
440             r_serv_term.delete(TrackedTermsNotificationEmailsPrefix_Name + term)
441             # delete the associated tags set
442             r_serv_term.delete(TrackedTermsNotificationTagsPrefix_Name + term)
443             # delete the associated companies
444             r_serv_term.delete(TrackedTermsCompanyPrefix_Name + term)
445
446     elif section == "blacklistTerm":
447         if action == "add":
448             r_serv_term.sadd(BlackListTermsSet_Name, term.lower())
449             r_serv_term.hset(BlackListTermsDate_Name, term, today_timestamp)
450         else:
451             r_serv_term.srem(BlackListTermsSet_Name, term.lower())
452     else:
453         return "None"
454
455     to_return = {}
456     to_return["section"] = section
457     to_return["action"] = action
458     to_return["term"] = term
459     return jsonify(to_return)
460
461 @terms.route("/terms_management/delete_terms_tags", methods=['POST'])
462 def delete_terms_tags():
463     term = request.form.get('term')
464     tags_to_delete = request.form.getlist('tags_to_delete')
465
```

```
466     if term is not None and tags_to_delete is not None:
467         for tag in tags_to_delete:
468             r_serv_term.srem(TrackedTermsNotificationTagsPrefix_Name + term, tag)
469         return redirect(url_for('terms.terms_management'))
470     else:
471         return 'None args', 400
472 
473 @terms.route("/terms_management/delete_terms_email", methods=['GET'])
474 def delete_terms_email():
475     term = request.args.get('term')
476     email = request.args.get('email')
477 
478     if term is not None and email is not None:
479         r_serv_term.srem(TrackedTermsNotificationEmailsPrefix_Name + term, email)
480         return redirect(url_for('terms.terms_management'))
481     else:
482         return 'None args', 400
483 
484 
485 @terms.route("/terms_plot_tool/")
486 def terms_plot_tool():
487     term = request.args.get('term')
488     if term is not None:
489         return render_template("terms_plot_tool.html", term=term)
490     else:
491         return render_template("terms_plot_tool.html", term="")
492 
493 
494 @terms.route("/terms_plot_tool_data/")
495 def terms_plot_tool_data():
496     oneDay = 60*60*24
497     range_start = datetime.datetime.utcfromtimestamp(int(float(request.args.get('range_start
        ↪ ')))) if request.args.get('range_start') is not None else 0;
498     range_start = range_start.replace(hour=0, minute=0, second=0, microsecond=0)
499     range_start = calendar.timegm(range_start.timetuple())
500     range_end = datetime.datetime.utcfromtimestamp(int(float(request.args.get('range_end')))
        ↪  if request.args.get('range_end') is not None else 0;
501     range_end = range_end.replace(hour=0, minute=0, second=0, microsecond=0)
502     range_end = calendar.timegm(range_end.timetuple())
503     term = request.args.get('term')
504 
505     per_paste = request.args.get('per_paste')
506     if per_paste == "1" or per_paste is None:
507         per_paste = "per_paste_"
508     else:
509         per_paste = ""
510 
511     if term is None:
512         return "None"
513 
514     else:
515         value_range = []
516         for timestamp in range(range_start, range_end+oneDay, oneDay):
517             value = r_serv_term.hget(per_paste+str(timestamp), term)
518             curr_value_range = int(value) if value is not None else 0
519             value_range.append([timestamp, curr_value_range])
520         value_range.insert(0,term)
521         return jsonify(value_range)
522 
523 
524 @terms.route("/terms_plot_top/")
525 def terms_plot_top():
526     per_paste = request.args.get('per_paste')
527     per_paste = per_paste if per_paste is not None else 1
528     return render_template("terms_plot_top.html", per_paste=per_paste)
529 
530 
531 @terms.route("/terms_plot_top_data/")
532 def terms_plot_top_data():
533     oneDay = 60*60*24
534     today = datetime.datetime.now()
535     today = today.replace(hour=0, minute=0, second=0, microsecond=0)
```

```
536     today_timestamp = calendar.timegm(today.timetuple())
537
538     per_paste = request.args.get('per_paste')
539     if per_paste == "1" or per_paste is None:
540        per_paste = "per_paste_"
541     else:
542        per_paste = ""
543
544     set_day = per_paste + "TopTermFreq_set_day_" + str(today_timestamp)
545     set_week = per_paste + "TopTermFreq_set_week";
546     set_month = per_paste + "TopTermFreq_set_month";
547
548     the_set = per_paste + request.args.get('set')
549     num_day = int(request.args.get('num_day'))
550
551     if the_set is None:
552        return "None"
553     else:
554        to_return = []
555        if "TopTermFreq_set_day" in the_set:
556           the_set += "_" + str(today_timestamp)
557
558        for term, tot_value in r_serv_term.zrevrangebyscore(the_set, '+inf', '-inf',
               ↪ withscores=True, start=0, num=20):
559           position = {}
560           position['day'] = r_serv_term.zrevrank(set_day, term)
561           position['day'] = position['day']+1 if position['day'] is not None else "<20"
562           position['week'] = r_serv_term.zrevrank(set_week, term)
563           position['week'] = position['week']+1 if position['week'] is not None else "<20"
564           position['month'] = r_serv_term.zrevrank(set_month, term)
565           position['month'] = position['month']+1 if position['month'] is not None else "<20"
566           value_range = []
567           for timestamp in range(today_timestamp, today_timestamp - num_day*oneDay, -oneDay):
568              value = r_serv_term.hget(per_paste+str(timestamp), term)
569              curr_value_range = int(value) if value is not None else 0
570              value_range.append([timestamp, curr_value_range])
571
572           to_return.append([term, value_range, tot_value, position])
573
574        return jsonify(to_return)
575
576
577  @terms.route("/credentials_tracker/")
578  def credentials_tracker():
579     return render_template("credentials_tracker.html")
580
581  @terms.route("/credentials_management_query_paste/", methods=['GET', 'POST'])
582  def credentials_management_query_paste():
583     cred = request.args.get('cred')
584     allPath = request.json['allPath']
585
586     paste_info = []
587     for pathNum in allPath:
588        path = r_serv_cred.hget(REDIS_KEY_ALL_PATH_SET_REV, pathNum)
589        paste = Paste.Paste(path)
590        p_date = str(paste._get_p_date())
591        p_date = p_date[0:4]+'/'+p_date[4:6]+'/'+p_date[6:8]
592        p_source = paste.p_source
593        p_encoding = paste._get_p_encoding()
594        p_size = paste.p_size
595        p_mime = paste.p_mime
596        p_lineinfo = paste.get_lines_info()
597        p_content = paste.get_p_content()
598        if p_content != 0:
599           p_content = p_content[0:400]
600        paste_info.append({"path": path, "date": p_date, "source": p_source, "encoding":
               ↪ p_encoding, "size": p_size, "mime": p_mime, "lineinfo": p_lineinfo, "content":
               ↪ p_content})
601
602     return jsonify(paste_info)
603
604  @terms.route("/credentials_management_action/", methods=['GET'])
```

```
605 def cred_management_action():
606
607     supplied = request.args.get('term')
608     action = request.args.get('action')
609     section = request.args.get('section')
610     extensive = request.args.get('extensive')
611     extensive = True if extensive == "true" else False
612
613     if extensive:
614         #collectDico
615         AllUsernameInRedis = r_serv_cred.hgetall(REDIS_KEY_ALL_CRED_SET).keys()
616     uniq_num_set = set()
617     if action == "seek":
618         possibilities = mixUserName(supplied, extensive)
619         for poss in possibilities:
620             num = r_serv_cred.hget(REDIS_KEY_ALL_CRED_SET, poss)
621             if num is not None:
622                 uniq_num_set.add(num)
623             for num in r_serv_cred.smembers(poss):
624                 uniq_num_set.add(num)
625         #Extensive /!\
626         if extensive:
627             iter_num = 0
628             tot_iter = len(AllUsernameInRedis)*len(possibilities)
629             for tempUsername in AllUsernameInRedis:
630                 for poss in possibilities:
631                     #FIXME print progress
632                     if(iter_num % int(tot_iter/20) == 0):
633                         #print("searching: {}% done".format(int(iter_num/tot_iter*100)), sep=' ',
                                ↪ end='\r', flush=True)
634                         print("searching: {}% done".format(float(iter_num)/float(tot_iter)*100))
635                     iter_num += 1
636
637                     if poss in tempUsername:
638                         num = (r_serv_cred.hget(REDIS_KEY_ALL_CRED_SET, tempUsername))
639                         if num is not None:
640                             uniq_num_set.add(num)
641                         for num in r_serv_cred.smembers(tempUsername):
642                             uniq_num_set.add(num)
643
644     data = {'usr': [], 'path': [], 'numPaste': [], 'simil': []}
645     for Unum in uniq_num_set:
646         levenRatio = 2.0
647         username = (r_serv_cred.hget(REDIS_KEY_ALL_CRED_SET_REV, Unum))
648
649         # Calculate Levenshtein distance, ignore negative ratio
650         supp_splitted = supplied.split()
651         supp_mixed = supplied.replace(' ','')
652         supp_splitted.append(supp_mixed)
653         for indiv_supplied in supp_splitted:
654             levenRatio = float(Levenshtein.ratio(indiv_supplied, username))
655             levenRatioStr = "{:.1%}".format(levenRatio)
656
657         data['usr'].append(username)
658
659
660         allPathNum = list(r_serv_cred.smembers(REDIS_KEY_MAP_CRED_TO_PATH+'_'+Unum))
661
662         data['path'].append(allPathNum)
663         data['numPaste'].append(len(allPathNum))
664         data['simil'].append(levenRatioStr)
665
666     to_return = {}
667     to_return["section"] = section
668     to_return["action"] = action
669     to_return["term"] = supplied
670     to_return["data"] = data
671
672     return jsonify(to_return)
673
674
675 # ========= REGISTRATION =========
```

```
676 app.register_blueprint(terms, url_prefix=baseUrl)
```

A.6. terms_management.html adaptado

```
<!DOCTYPE html>
<html>

<head>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">

 <title>Terms Management</title>
 <link rel="icon" href="{{ url_for('static', filename='image/ail-icon.png') }}">

 <!-- Core CSS -->
 <link href="{{ url_for('static', filename='css/bootstrap.min.css') }}" rel="stylesheet">
 <link href="{{ url_for('static', filename='font-awesome/css/font-awesome.css') }}" rel="
     ↪ stylesheet">
 <link href="{{ url_for('static', filename='css/sb-admin-2.css') }}" rel="stylesheet">
 <link href="{{ url_for('static', filename='css/dataTables.bootstrap.css') }}" rel="
     ↪ stylesheet" type="text/css" />
 <link href="{{ url_for('static', filename='css/switch_checkbox.css') }}" rel="stylesheet"
     ↪ type="text/css" />
 <script language="javascript" src="{{ url_for('static', filename='js/jquery.js')}}"></
     ↪ script>
 <script src="{{ url_for('static', filename='js/bootstrap.min.js') }}"></script>
 <script src="{{ url_for('static', filename='js/jquery.dataTables.min.js') }}"></script>
 <script src="{{ url_for('static', filename='js/dataTables.bootstrap.js') }}"></script>
 <script src="{{ url_for('static', filename='js/jquery.flot.js') }}"></script>
 <script src="{{ url_for('static', filename='js/jquery.flot.time.js') }}"></script>
 <script src="{{ url_for('static', filename='js/jquery.flot.stack.js') }}"></script>

 <style>
  .sparkLineStats ul {
   padding-left:0;
   list-style:none
  }

  .btn-link {
   color: #000000
  }

  .popover-content {
   white-space:pre-wrap;
   word-wrap:break-word;
  }
  .mouse_pointer{
   cursor: pointer;
  }
  .lb-md {
   font-size: 16px;
  }
 </style>
</head>
<body>

 <!-- Modal -->
 <div id="mymodal" class="modal fade" role="dialog">
  <div class="modal-dialog modal-lg">

   <!-- Modal content-->
   <div id="mymodalcontent" class="modal-content">
    <div id="mymodalbody" class="modal-body" max-width="8500px">
     <p>Loading paste information...</p>
     <img id="loading-gif-modal" src="{{url_for('static', filename='image/loading.gif') }}"
         ↪ height="26" width="26" style="margin: 4px;">
    </div>
    <div class="modal-footer">
     <a id="button_show_plot" target="_blank" href=""><button type="button" class="btn btn-
         ↪ info">Plot term</button></a>
     <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
    </div>
   </div>
```

```
64    </div>
65   </div>
66
67   {% include 'navbar.html' %}
68
69  <div id="page-wrapper">
70   <div class="row">
71    <div class="col-lg-12">
72     <h1 class="page-header" data-page="page-termsfrequency" >Terms frequency: Management
         ↪ interface</h1>
73    </div>
74    <!-- /.col-lg-12 -->
75   </div>
76   <!-- /.row -->
77   <div class="row">
78
79   <!-- Panel OPTIONS -->
80   <div class="row">
81    <div class="col-lg-12">
82     <div class="row">
83      {% set uniq_id = namespace(modal_id=0) %}
84       <div class="col-lg-12">
85        <label class="switch">
86         <input id="per_paste" class="switch-input" value="per_paste" type="checkbox"
             ↪ onclick="reload_per_paste()">
87         <span class="switch-label" data-on="On" data-off="Off"></span>
88         <span class="switch-handle"></span>
89        </label>
90        <strong style="top: 3px; position: relative;">1 term per paste</strong>
91
92        <div id="panel-today" class="panel panel-success">
93         <div class="panel-heading">
94          <strong>Manage tracked terms</strong>
95         </div>
96         <div class="panel-body">
97
98          <div style="margin-bottom: 10px;">
99           <table>
100           <tr><td><b>Regex</b>: surround the term by '<b>/</b>'. </td> <td><b style="
                ↪ margin-left: 20px;">/([a-z])\w+([a-z])\n/</b></td></tr>
101           <tr><td><b>Set of terms</b>: surround the list by '<b>\</b>'. </td> <td><b
                ↪ style="margin-left: 20px;">\[term1, term2, ...]\</b></td></tr>
102           <tr><td> - To set a custom matching <b>threshold</b> (defaut=50), append it at
                ↪ the end as a inner list '<b>[thresh]</b>'. </td> <td><b style="margin-
                ↪ left: 20px;">\[term1, term2, ..., [75]]\</b></td></tr>
103          </table>
104         </div>
105         <div class="form-group input-group" style="margin-bottom: 30px;">
106          <span class="input-group-addon"><span class="fa fa-eye"></span></span>
107          <input id="followTermInput" class="form-control" placeholder="Term to track."
               ↪ type="text" style="max-width: 400px;">
108          <input id="followTermEMailNotificationReceiversInput" class="form-control"
               ↪ placeholder="Notification E-Mails (optional, space separated)" type="text
               ↪ " style="max-width: 400px;">
109          <input id="followTermTag" class="form-control" placeholder="Tags (optional,
               ↪ space separated)" type="text" style="max-width: 400px;">
110          <input id="followTermCompanyInput" class="form-control" placeholder="Company"
               ↪ type="text" style="max-width: 400px;">
111          <button id="followTermBtn" class="btn btn-success btn-interaction" style="margin
               ↪ -left: 10px;" data-section="followTerm" data-action="add"> Add term</
               ↪ button>
112         </div>
113
114         <table class="table table-striped table-bordered table-hover" id="myTable">
115          <thead>
116           <tr>
117            <th style="max-width: 800px;">Term</th>
118            <th>Added date</th>
119            <th>Day occurence</th>
120            <th>Week occurence</th>
121            <th>Month occurence</th>
122            <th># tracked paste</th>
```

```
123                 <th>Action</th>
124                 <th>Notification E-Mails</th>
125                 <th>Company</th>
126                </tr>
127               </thead>
128               <tbody>
129               <!-- SET -->
130               {% for set in trackSet_list %}
131                <tr style="background-color: #cdffca;">
132                  <td>
133                   <span class="term_name">{{ set }}</span>
134                   <div>
135                    {% for tag in notificationTagsTermMapping[set] %}
136                    <a href="{{ url_for('Tags.get_tagged_paste') }}?ltags={{ tag }}">
137                     <span class="label label-{{ bootstrap_label[loop.index0 % 5] }} pull-left
                             ↪ ">{{ tag }}</span>
138                    </a>
139                    {% endfor %}
140                    {% if notificationTagsTermMapping[set] %}
141                    <div class="btn-link btn-interaction pull-right mouse_pointer" data-toggle
                         ↪ ="modal" data-target="#edit_custom_tag_modal_{{ uniq_id.modal_id }}"
                         ↪  data-placement="right" title="Edit Tags List"><i class="fa fa-
                         ↪ pencil" style="color:Red;"></i></div>
142
143                    <div id="edit_custom_tag_modal_{{ uniq_id.modal_id }}" class="modal fade"
                         ↪ role="dialog">
144                     <div class="modal-dialog">
145
146                      <!-- Modal content-->
147                      <div id="mymodalcontent" class="modal-content">
148                       <div class="modal-header" style="border-bottom: 4px solid #48c9b0;
                             ↪ background-color: #48c9b0; color: #ffffff;">
149                        <h2 class="text-center">Remove Custom Tag</h2>
150                       </div>
151                       <div class="modal-body">
152                        <form action="{{ url_for('terms.delete_terms_tags') }}" id="checkboxForm
                             ↪ " method='post'>
153                         {% for tag in notificationTagsTermMapping[set] %}
154                          <div class="form-check">
155                           <input type="hidden" class="form-control" name="term" value="{{ set
                                 ↪ }}">
156                          <input type="checkbox" class="form-check-input" name="tags_to_delete"
                                 ↪ value="{{ tag }}">
157                          <label class="form-check-label">
158                           <span class="label label-{{ bootstrap_label[loop.index0 % 5] }} lb-md
                                 ↪ ">{{ tag }}</span>
159                          </label>
160                          <br>
161                         </div>
162                        {% endfor %}
163                        </form>
164                       </div>
165                       <div class="modal-footer">
166                        <button class="btn btn-danger" type="submit" form="checkboxForm" value="
                             ↪ Submit">
167                         <span class="glyphicon glyphicon-trash"></span>
168                         <span class="label-icon">Remove Tags</span>
169                        </button>
170                        <button type="button" class="btn btn-default" data-dismiss="modal" >
                             ↪ Close</button>
171                       </div>
172                      </div>
173                     </div>
174                    </div>
175                    {% set uniq_id.modal_id = uniq_id.modal_id + 1 %}
176                    {% endif %}
177                  </div>
178                 </td>
179                 <td>{{ trackSet_list_values[loop.index0][3] }}</td>
180                 <td>{{ trackSet_list_values[loop.index0][0] }}</td>
181                 <td>{{ trackSet_list_values[loop.index0][1] }}</td>
182                 <td>{{ trackSet_list_values[loop.index0][2] }}</td>
```

```
183                 <td>{{ trackSet_list_num_of_paste[loop.index0] }}</td>
184                 <td><p style="margin: 0px; white-space: nowrap;">
185                   <span data-toggle="modal" data-target="#mymodal" data-term="{{ set }}" ><
                          ↪ button class="btn-link" data-toggle="tooltip" data-placement="right"
                          ↪ title="Show concerned paste(s)"><span class="glyphicon glyphicon-info
                          ↪ -sign"></span></button></span>
186                   <button class="btn-link btn-interaction" data-toggle="tooltip" data-
                          ↪ placement="left" title="Remove this term" data-content="{{ set }}"
                          ↪ data-section="followTerm" data-action="delete"><span class="glyphicon
                          ↪  glyphicon-trash"></span></button>
187                      <input id="checkBoxEMailAlerts" type="checkbox" title="Toggle E
                          ↪ -Mail notifications" class="btn-link btn-interaction" data-content
                          ↪ ="{{ set }}" data-section="followTerm" data-action="
                          ↪ toggleEMailNotification" {% if notificationEnabledDict[set] %}
                          ↪ checked {% endif %}>
188                 </p></td>
189                 <td>
190                  {% for email in notificationEMailTermMapping[set] %}
191                   <a href="{{ url_for('terms.delete_terms_email') }}?email={{email}}&term={{
                          ↪ set}}">
192                    <div class="btn-link btn-interaction pull-right mouse_pointer" data-toggle
                          ↪ ="tooltip" data-placement="left" data-original-title="Remove this
                          ↪ email">
193                     <span class="glyphicon glyphicon-trash" style="color:Red;" ></span>
194                    </div>
195                   </a>
196                   {{ email }}
197                   <br>
198                  {% endfor %}
199                 </td>
200                 <td>
201                 {% for companyTermMapping_Company in companyTermMapping[set] %}
202                 {{ companyTermMapping_Company }}
203                 <br>
204                 {% endfor %}
205                 </td>
206               </tr>
207             {% endfor %}
208             <!-- REGEX -->
209             {% for regex in trackReg_list %}
210              <tr style="background-color: #fffdca;">
211                <td>
212                  <span class="term_name">{{ regex }}</span>
213                  <div>
214                   {% for tag in notificationTagsTermMapping[regex] %}
215                   <a href="{{ url_for('Tags.get_tagged_paste') }}?ltags={{ tag }}">
216                    <span class="label label-{{ bootstrap_label[loop.index0 % 5] }} pull-left
                          ↪ ">{{ tag }}</span>
217                   </a>
218                   {% endfor %}
219                   {% if notificationTagsTermMapping[regex] %}
220                   <div class="btn-link btn-interaction pull-right mouse_pointer" data-toggle
                          ↪ ="modal" data-target="#edit_custom_tag_modal_{{ uniq_id.modal_id }}"
                          ↪  data-placement="right" title="Edit Tags List"><i class="fa fa-
                          ↪ pencil" style="color:Red;"></i></div>
221                   <div id="edit_custom_tag_modal_{{ uniq_id.modal_id }}" class="modal fade"
                          ↪ role="dialog">
222                    <div class="modal-dialog">
223                     <!-- Modal content-->
224                     <div id="mymodalcontent" class="modal-content">
225                       <div class="modal-header" style="border-bottom: 4px solid #48c9b0;
                              ↪ background-color: #48c9b0; color: #ffffff;">
226                        <h2 class="text-center">Remove Custom Tag</h2>
227                       </div>
228                      <div class="modal-body">
229                       <form action="{{ url_for('terms.delete_terms_tags') }}" id="checkboxForm
                              ↪ " method='post'>
230                        {% for tag in notificationTagsTermMapping[regex] %}
231                         <div class="form-check">
232                           <input type="hidden" class="form-control" name="term" value="{{ regex
                                  ↪  }}">
```

```
233                     <input type="checkbox" class="form-check-input" name="tags_to_delete"
                            ↪ value="{{ tag }}">
234                     <label class="form-check-label">
235                      <span class="label label-{{ bootstrap_label[loop.index0 % 5] }} lb-md
                             ↪ ">{{ tag }}</span>
236                     </label>
237                     <br>
238                    </div>
239                   {% endfor %}
240                  </form>
241                 </div>
242                 <div class="modal-footer">
243                  <button class="btn btn-danger" type="submit" form="checkboxForm" value="
                         ↪ Submit">
244                    <span class="glyphicon glyphicon-trash"></span>
245                    <span class="label-icon">Remove Tags</span>
246                  </button>
247                  <button type="button" class="btn btn-default" data-dismiss="modal" >
                         ↪ Close</button>
248                 </div>
249                </div>
250               </div>
251              </div>
252              {% set uniq_id.modal_id = uniq_id.modal_id + 1 %}
253              {% endif %}
254             </div>
255            </td>
256            <td>{{ trackReg_list_values[loop.index0][3] }}</td>
257            <td>{{ trackReg_list_values[loop.index0][0] }}</td>
258            <td>{{ trackReg_list_values[loop.index0][1] }}</td>
259            <td>{{ trackReg_list_values[loop.index0][2] }}</td>
260            <td>{{ trackReg_list_num_of_paste[loop.index0] }}</td>
261            <td><p style="margin: 0px; white-space: nowrap;">
262             <span data-toggle="modal" data-target="#mymodal" data-term="{{ regex }}" ><
                    ↪ button class="btn-link" data-toggle="tooltip" data-placement="right"
                    ↪ title="Show concerned paste(s)"><span class="glyphicon glyphicon-info
                    ↪ -sign"></span></button></span>
263             <button class="btn-link btn-interaction" data-toggle="tooltip" data-
                    ↪ placement="left" title="Remove this term" data-content="{{ regex }}"
                    ↪ data-section="followTerm" data-action="delete"><span class="glyphicon
                    ↪  glyphicon-trash"></span></button>
264                <input id="checkBoxEMailAlerts" type="checkbox" title="Toggle E
                    ↪ -Mail notifications" class="btn-link btn-interaction" data-content
                    ↪ ="{{ regex }}" data-section="followTerm" data-action="
                    ↪ toggleEMailNotification" {% if notificationEnabledDict[regex] %}
                    ↪ checked {% endif %}>
265            </p></td>
266            <td>
267             {% for email in notificationEMailTermMapping[regex] %}
268              <a href="{{ url_for('terms.delete_terms_email') }}?email={{email}}&term={{
                     ↪ regex}}">
269               <div class="btn-link btn-interaction pull-right mouse_pointer" data-toggle
                      ↪ ="tooltip" data-placement="left" data-original-title="Remove this
                      ↪ email">
270                <span class="glyphicon glyphicon-trash" style="color:Red;"></span>
271               </div>
272              </a>
273              {{ email }}
274              <br>
275             {% endfor %}
276            </td>
277            <td>
278            {% for companyTermMapping_Company in companyTermMapping[regex] %}
279            {{ companyTermMapping_Company }}
280            <br>
281            {% endfor %}
282            </td>
283           </tr>
284          {% endfor %}
285          <!-- Normal term -->
286          {% for term in track_list %}
287           <tr>
```

```
288                 <td>
289                   <span class="term_name">{{ term }}</span>
290                   <div>
291                    {% for tag in notificationTagsTermMapping[term] %}
292                    <a href="{{ url_for('Tags.get_tagged_paste') }}?ltags={{ tag }}">
293                     <span class="label label-{{ bootstrap_label[loop.index0 % 5] }} pull-left
                           ↪ ">{{ tag }}</span>
294                    </a>
295                    {% endfor %}
296                    {% if notificationTagsTermMapping[term] %}
297                    <div class="btn-link btn-interaction pull-right mouse_pointer" data-toggle
                           ↪ ="modal" data-target="#edit_custom_tag_modal_{{ uniq_id.modal_id }}"
                           ↪  data-placement="right" title="Edit Tags List"><i class="fa fa-
                           ↪ pencil" style="color:Red;"></i></div>
298                    <div id="edit_custom_tag_modal_{{ uniq_id.modal_id }}" class="modal fade"
                           ↪ role="dialog">
299                     <div class="modal-dialog">
300                      <!-- Modal content-->
301                      <div id="mymodalcontent" class="modal-content">
302                       <div class="modal-header" style="border-bottom: 4px solid #48c9b0;
                             ↪ background-color: #48c9b0; color: #ffffff;">
303                        <h2 class="text-center">Remove Custom Tag</h2>
304                       </div>
305                       <div class="modal-body">
306                        <form action="{{ url_for('terms.delete_terms_tags') }}" id="checkboxForm
                             ↪ " method='post'>
307                         {% for tag in notificationTagsTermMapping[term] %}
308                          <div class="form-check">
309                           <input type="hidden" class="form-control" name="term" value="{{ term
                                 ↪ }}">
310                           <input type="checkbox" class="form-check-input" name="tags_to_delete"
                                 ↪ value="{{ tag }}">
311                           <label class="form-check-label">
312                            <span class="label label-{{ bootstrap_label[loop.index0 % 5] }} lb-md
                                 ↪ ">{{ tag }}</span>
313                           </label>
314                           <br>
315                          </div>
316                         {% endfor %}
317                        </form>
318                       </div>
319                       <div class="modal-footer">
320                        <button class="btn btn-danger" type="submit" form="checkboxForm" value="
                             ↪ Submit">
321                         <span class="glyphicon glyphicon-trash"></span>
322                         <span class="label-icon">Remove Tags</span>
323                        </button>
324                        <button type="button" class="btn btn-default" data-dismiss="modal" >
                             ↪ Close</button>
325                       </div>
326                      </div>
327                     </div>
328                    </div>
329                    {% set uniq_id.modal_id = uniq_id.modal_id + 1 %}
330                    {% endif %}
331                   </div>
332                 </td>
333                 <td>{{ track_list_values[loop.index0][3] }}</td>
334                 <td>{{ track_list_values[loop.index0][0] }}</td>
335                 <td>{{ track_list_values[loop.index0][1] }}</td>
336                 <td>{{ track_list_values[loop.index0][2] }}</td>
337                 <td>{{ track_list_num_of_paste[loop.index0] }}</td>
338                 <td><p style="margin: 0px; white-space: nowrap;">
339                   <span data-toggle="modal" data-target="#mymodal" data-term="{{ term }}" ><
                         ↪ button class="btn-link" data-toggle="tooltip" data-placement="right"
                         ↪ title="Show concerned paste(s)"><span class="glyphicon glyphicon-info
                         ↪ -sign"></span></button></span>
340                   <button class="btn-link btn-interaction" data-toggle="tooltip" data-
                         ↪ placement="left" title="Remove this term" data-content="{{ term }}"
                         ↪ data-section="followTerm" data-action="delete"><span class="glyphicon
                         ↪  glyphicon-trash"></span></button>
```

```
341                     <input id="checkBoxEMailAlerts" type="checkbox" title="Toggle E
                         ↪ -Mail notifications" class="btn-link btn-interaction" data-content
                         ↪ ="{{ term }}" data-section="followTerm" data-action="
                         ↪ toggleEMailNotification" {% if notificationEnabledDict[term] %}
                         ↪ checked {% endif %}>
342                 </p></td>
343                 <td>
344                  {% for email in notificationEMailTermMapping[term] %}
345                   <a href="{{ url_for('terms.delete_terms_email') }}?email={{email}}&term={{
                         ↪ term}}">
346                    <div class="btn-link btn-interaction pull-right mouse_pointer" data-toggle
                         ↪ ="tooltip" data-placement="left" data-original-title="Remove this
                         ↪ email">
347                     <span class="glyphicon glyphicon-trash" style="color:Red;"></span>
348                    </div>
349                   </a>
350                   {{ email }}
351                   <br>
352                  {% endfor %}
353                 </td>
354                 <td>
355                 {% for companyTermMapping_Company in companyTermMapping[term] %}
356                 {{ companyTermMapping_Company }}
357                 <br>
358                 {% endfor %}
359                 </td>
360               </tr>
361              {% endfor %}
362              </tbody>
363            </table>
364          <!-- /.panel-body -->
365        </div>
366       </div>
367       <!-- /.panel -->
368     </div>
369     <!-- /.panel -->
370   </div>
371 </div>
372
373  <!-- Panel OPTIONS -->
374   <div class="col-lg-12">
375     <div class="row">
376       <div class="col-lg-12">
377         <div id="panel-today" class="panel panel-danger">
378           <div class="panel-heading">
379             <strong>Manage blacklisted terms</strong>
380           </div>
381           <div class="panel-body">
382
383             <div class="form-group input-group" style="margin-bottom: 30px;">
384               <span class="input-group-addon"><span class="fa fa-eye-slash "></span></span>
385               <input id="blacklistTermInput" class="form-control" placeholder="Term to track"
                         ↪ type="text" style="max-width: 400px;">
386               <button id="blacklistTermBtn" class="btn btn-danger btn-interaction" style="
                         ↪ margin-left: 10px;" data-section="blacklistTerm" data-action="add"> Black
                         ↪  list a term</button>
387             </div>
388
389             <table class="table table-striped table-bordered table-hover" id="myTable2">
390               <thead>
391                 <tr>
392                  <th style="max-width: 800px;">Term</th>
393                  <th>Added date</th>
394                  <th>Action</th>
395                 </tr>
396               </thead>
397               <tbody>
398               {% for term, date in black_list %}
399                <tr>
400                 <td>{{ term }}</td>
401                 <td>{{ date }}</td>
402                 <td><p style="margin: 0px;">
```

```
403                 <button class="btn-link btn-interaction" data-toggle="tooltip" data-
                        ↪ placement="right" title="Remove this term" data-content="{{ term }}"
                        ↪ data-section="blacklistTerm" data-action="delete"><span class="
                        ↪ glyphicon glyphicon-trash"></span></button>
404               </p></td>
405             </tr>
406             {% endfor %}
407             </tbody>
408           </table>
409         <!-- /.panel-body -->
410       </div>
411     </div>
412     <!-- /.panel -->
413   </div>
414   <!-- /.panel -->
415  </div>
416 </div>
417
418 <!-- /.row -->
419 </div>
420 <!-- /#page-wrapper -->
421 </div>
422
423
424 <!-- import graph function -->
425 <script>
426  function reload_per_paste() {
427   var checked = $("#per_paste").prop( "checked" ) ? 1 : 0;
428   window.location.href = {{ url_for('terms.terms_management') }}+"?per_paste="+checked;
429  }
430
431
432  var table_track;
433  var table_black;
434
435 function bindEventsForCurrentPage() {
436  // On click, get html content from url and update the corresponding modal
437  $("[data-toggle='modal']").unbind().on("click.openmodal", function (event) {
438   //console.log(data);
439   event.preventDefault();
440   var the_modal=$(this);
441   var url = "{{ url_for('terms.terms_management_query_paste') }}?term=" +
        ↪ encodeURIComponent($(this).attr('data-term'));
442   $.getJSON(url, function (data) {
443    if (data.length != 0) {
444     var html_to_add = "";
445     html_to_add += "<table id=\"modal-table\" class=\"table table-striped\">";
446     html_to_add += "<thead>";
447     html_to_add += "<tr>";
448     html_to_add += "<th>Source</th>";
449     html_to_add += "<th>Date</th>";
450     html_to_add += "<th>Encoding</th>";
451     html_to_add += "<th>Size (Kb)</th>";
452     html_to_add += "<th># lines</th>";
453     html_to_add += "<th>Max length</th>";
454     html_to_add += "<th>Preview</th>";
455     html_to_add += "</tr>";
456     html_to_add += "</thead>";
457     html_to_add += "<tbody>";
458     for (i=0; i<data.length; i++) {
459      curr_data = data[i];
460      html_to_add += "<tr>";
461      html_to_add += "<td>"+curr_data.source+"</td>";
462      html_to_add += "<td>"+curr_data.date+"</td>";
463      html_to_add += "<td>"+curr_data.encoding+"</td>";
464      html_to_add += "<td>"+curr_data.size+"</td>";
465      html_to_add += "<td>"+curr_data.lineinfo[0]+"</td>";
466      html_to_add += "<td>"+curr_data.lineinfo[1]+"</td>";
467      html_to_add += "<td><div class=\"row\"><button class=\"btn btn-xs btn-default\" data
          ↪ -toggle=\"popover\" data-placement=\"left\" data-content=\""+curr_data.
          ↪ content.replace(/\"/g, "\'")+"\">Preview content</button><a target=\"_blank\"
          ↪  href=\"{{ url_for('showsavedpastes.showsavedpaste') }}?paste="+curr_data.
```

```
            ↪ path+"&num=0\"> <button type=\"button\" class=\"btn btn-xs btn-info\">Show
            ↪ Paste</button></a></div></td>";
468
469         html_to_add += "</tr>";
470       }
471       html_to_add += "</tbody>";
472       html_to_add += "</table>";
473       $("#mymodalbody").html(html_to_add);
474       $("[data-toggle=popover]").popover();
475       $("#button_show_plot").attr("href", "{{ url_for('terms.terms_plot_tool')}}"+"?term="+
              ↪ the_modal.attr('data-term') );
476       $('#modal-table').DataTable();
477     } else {
478       $("#mymodalbody").html("No paste containing this term has been received yet.");
479       $("#button_show_plot").attr("href", "{{ url_for('terms.terms_plot_tool')}}"+"?term="+
              ↪ the_modal.attr('data-term') );
480     }
481   });
482
483 });
484
485 }
486
487
488 $(document).ready(function(){
489   bindEventsForCurrentPage();
490   activePage = $('h1.page-header').attr('data-page');
491   $("#"+activePage).addClass("active");
492   if({{ per_paste }} == 1) {
493     $("#per_paste").attr('checked', true)
494   }
495
496   $('[data-toggle="tooltip"]').tooltip();
497   table_track = $('#myTable').DataTable();
498   table_black = $('#myTable2').DataTable();
499
500   table_track.on( 'draw.dt', function () {
501     perform_binding();
502   });
503   table_black.on( 'draw.dt', function () {
504     perform_binding();
505   });
506
507
508   $("#followTermInput").keyup(function(event){
509     if(event.keyCode == 13){
510       $("#followTermBtn").click();
511       $("#followTermInput").val("");
512     }
513   });
514
515   $("#blacklistTermInput").keyup(function(event){
516     if(event.keyCode == 13){
517       $("#blacklistTermBtn").click();
518       $("#blacklistTermInput").val("");
519     }
520   });
521
522   perform_binding();
523
524 $("#mymodal").on('hidden.bs.modal', function () {
525   $("#mymodalbody").html("<p>Loading paste information...</p>");
526   var loading_gif = "<img id='loading-gif-modal' class='img-center' src=\"{{url_for('static
          ↪ ', filename='image/loading.gif') }}\" height='26' width='26' style='margin: 4px
          ↪ ;'>";
527   $("#mymodalbody").append(loading_gif); // Show the loading GIF
528 });
529
530
531 });
532 </script>
533
```

```
<script>



function perform_binding() {
  $(".btn-interaction").unbind("click.interaction");
  $(".btn-interaction").bind("click.interaction", perform_operation);
}

function perform_operation(){
  var curr_section = $(this).attr('data-section');
  var curr_action = $(this).attr('data-action');
  var row_tr = $(this).closest("tr");
  if (curr_action == "add") {
    var curr_term = $('#'+curr_section+'Input').val();
    var email_addresses = $('#followTermEMailNotificationReceiversInput').val();
    var tags = $('#followTermTag').val();
    var company = $('#followTermCompanyInput').val();
  } else {
    var curr_term = $(this).attr('data-content');
    var email_addresses = "";
  }
  var data_to_send = { section: curr_section, action: curr_action, term: curr_term,
      ↪ emailAddresses: email_addresses, tags: tags, company: company};

  if (curr_term != "") {
    //console.log(data_to_send);
    $.get("{{ url_for('terms.terms_management_action') }}", data_to_send, function(data,
        ↪ status){
      if(status == "success") {
        var json = data;

        if(json.section == "followTerm") {
          if(json.action == "add") {
            // query data
            $.get("{{ url_for('terms.terms_management_query') }}", { term: json.term, section:
                ↪ json.section }, function(data2, status){
              reload_per_paste();
            });
          } else if (json.action == "delete") {
           row_tr.remove()
          }
        } else if(json.section == "blacklistTerm"){
          if(json.action == "add") {
            $.get("{{ url_for('terms.terms_management_query') }}", { term: json.term, section:
                ↪ json.section }, function(data2, status){
              console.log(data2);
              var action_button = "<button class=\"btn-link btn-interaction\" data-toggle=\"
                  ↪ tooltip\" data-placement=\"right\" title=\"Remove this term\" data-content
                  ↪ =\"" + json.term + "\" data-section=\"blacklistTerm\" data-action=\"delete
                  ↪ \"><span class=\"glyphicon glyphicon-trash\"></span></button>"
              table_black.row.add( [ json.term, data2[3], action_button ] ).draw( false );
              perform_binding();
            });
          } else if (json.action == "delete") {
            // Find indexes of row which have the term in the first column
            var index = table_black.rows().eq( 0 ).filter( function (rowIdx) {
              return table_black.cell( rowIdx, 0 ).data() === json.term;
            } );
            table_black.rows(index).remove().draw( false );
          }
        }
      }
    });
  }
}
</script>
```

A.7. previousConfigurations.sh

```
#!/bin/bash

#Create virtual enviroment
virtualenv -p python3 managementPortal

#Configure AIL_BIN path to the AIL instance for redis connections
read -p "Introduce AIL installation path: " AILPath
echo "export AIL_BIN="$AILPath"/bin/" >> managementPortal/bin/activate

#Activate virtual enviroment
source managementPortal/bin/activate

#Install required modules
pip3 install -U -r requirements.txt

#Create database
python3 identityManagement.py

#Initialize database
python3 initializeDatabase.py

#Change database permisions
chmod 400 identityManagement.db
```

A.8. initializeDatabase.py

```
import sqlite3
import hashlib
import secrets

conn = sqlite3.connect('identityManagement.db')
c = conn.cursor()

#Trigger
c.execute("CREATE TRIGGER userAdminCompanies AFTER INSERT ON companies BEGIN INSERT INTO
    ↪ usersCompany(userID,companyID) VALUES ((SELECT id FROM users WHERE username = 'admin
    ↪ '),new.id);END;")

#Admin user: admin/password
salt = secrets.token_hex(8)
hashedPassword = hashlib.sha512(("password"+salt).encode('utf-8')).hexdigest()
c.execute("INSERT INTO users('username','password','salt') VALUES ('admin','"+hashedPassword
    ↪ +"','"+salt+"');")

#test company
c.execute("INSERT INTO companies('companyName','termsLimit') VALUES ('test-Company',15);")

conn.commit()
c.close()
```

A.9. createUsers.py

```
import sqlite3
import hashlib
import secrets
import getpass
import os

#Database permissions to write
os.chmod('../identityManagement.db', 0o600)

#Generate random salt
salt = secrets.token_hex(8)

#Request user name:
user = input ("Introduce user name: ")

#Request user password
password = getpass.getpass("Introduce password: ")

#Calculate password hash
hashedPassword = hashlib.sha512((password+salt).encode('utf-8')).hexdigest()

#Connect to database and insert values
conn = sqlite3.connect('../identityManagement.db')
c = conn.cursor()
c.execute("INSERT INTO users(username,password,salt) VALUES ('"+user+"','"+hashedPassword
    ↪ +"','"+salt+"')")
conn.commit()
c.close()

#restablish database permissions
os.chmod('../identityManagement.db', 0o400)
```

A.10. createCompanies.py

```
1  import sqlite3
2  import os
3
4  #Database permissions to write
5  os.chmod('../identityManagement.db', 0o600)
6
7  #Request company name:
8  companyName = input ("Introduce company name: ")
9
10 #Request company terms limit
11 termsLimit = input ("Introduce company limit: ")
12
13
14 #Connect to database and insert values
15 conn = sqlite3.connect('../identityManagement.db')
16 c = conn.cursor()
17 c.execute("INSERT INTO companies(companyName,termsLimit) VALUES ('"+companyName+"','"+
       ↪ termsLimit+"')")
18 conn.commit()
19 c.close()
20
21 #restablish database permissions
22 os.chmod('../identityManagement.db', 0o400)
```

A.11. createRelationships.py

```
1  import sqlite3
2  import os
3
4  #Database permissions to write
5  os.chmod('../identityManagement.db', 0o600)
6
7  #Request user name:
8  username = input ("Introduce user name: ")
9
10 #Request company name:
11 companyName = input ("Introduce company name: ")
12
13 #Connect to database and insert values
14 conn = sqlite3.connect('../identityManagement.db')
15 c = conn.cursor()
16 c.execute("INSERT INTO usersCompany(userID,companyID) SELECT users.id,companies.id from
       ↪ users,companies where username ='"+username+"' and companyName='"+companyName+"'")
17 conn.commit()
18 c.close()
19
20 #restablish database permissions
21 os.chmod('../identityManagement.db', 0o400)
```