Facultade de Informática

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN

Desarrollo de una herramienta web para la visualización de comunicaciones de red

Estudiante: Adrián Garrido Salgado
Dirección: Francisco Javier Nóvoa Manuel
Dirección: Diego Fernández Iglesias

A Coruña, setembro de 2020.

A mis padres y a mi hermana...

Agradecimientos

Quisiera que estas líneas sirvieran para expresar mi más profundo y sincero agradecimiento a mis padres y a mi hermana, que me estuvieron apoyando a lo largo de toda mi vida académica, sin ellos no habría cumplido este sueño. Espero poder devolverles, en un futuro, una milésima parte de lo que ellos han hecho por mí.

Resumen

A lo largo de los años el número de elementos en una red, los servicios y la cantidad de tráfico generado han ido aumentando exponencialmente. Por esta razón, es cada vez más importante analizar y monitorizar el tráfico en una red para prevenir problemas de seguridad, optimizar la reserva de recursos, etc. Para realizar estas acciones, el análisis basado en paquetes es la opción tradicional. Este proporciona un conocimiento más profundo de la red y las comunicaciones que tienen lugar en ella, pero conlleva un alto consumo de recursos. Con el fin de paliar este alto consumo, surgió el análisis basado en flujos. Este tipo de análisis no sustituye al anterior, sino que lo complementa y constituye un enfoque más escalable, especialmente en redes de alta velocidad. Consiste en agrupar paquetes similares y analizar estas agrupaciones en conjunto, sin tener en cuenta cada paquete de forma individual. La similitud entre los paquetes capturados se basa en varias propiedades como puertos de capa de transporte, direcciones IP, protocolo encapsulado a nivel de red, etc. De hecho, la información de flujo puede ser útil para muchos propósitos: facilitar el diagnóstico y la solución de los problemas de comunicación, detectar anomalías, clasificar el tráfico…

La gran mayoría de las herramientas actuales requieren la instalación de software más o menos complejo por parte del cliente. Además, muchas de estas soluciones conllevan un elevado coste hardware o el pago de licencias. Por esos motivos se propone el diseño de una interfaz gráfica "amigable", que solo exija tener instalado un navegador web y que posibilite la generación de un esquema de red, aparte del propio análisis de flujos de tráfico.

Abstract

Over the years the number of elements in a network, the services and the amount of traffic generated have been increasing exponentially. For this reason, it is increasingly important to analyze and monitor traffic on a network to prevent security problems, optimize resource reservation, etc. To perform these actions, packet-based analysis is the traditional option. This provides a deeper understanding of the network and the communications that take place on it, but involves a high consumption of resources. In order to alleviate this high consumption, flow-based analysis emerged. This type of analysis does not replace the previous one, but rather complements it and constitutes a more scalable approach, especially in high-speed networks. It consists in grouping similar packages and analyzing these groups together, without taking into account each package individually. The similarity between the captured packets is

based on various properties such as transport layer ports, IP addresses, network-level encapsulated protocol, etc. In fact, flow information can be useful for many purposes: to facilitate the diagnosis and solution of communication problems, detect anomalies, classify traffic...

The vast majority of current tools require the installation of more or less complex software by the customer. In addition, many of these solutions carry a high cost of hardware or the payment of licenses. For these reasons, the design of a "friendly" graphical interface is proposed, which only requires having a web browser installed and which allows the generation of a network diagram, in addition to the analysis of traffic flows itself.

Palabras clave:

- Topología
- Análisis
- Informe
- Gráfico
- Flujos
- Spring
- Java
- Maven
- Web
- Thymeleaf

Keywords:

- Topology
- Analysis
- Report
- Graphic
- Flows
- Spring
- Java
- Maven
- Web
- Thymeleaf

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Hoy día la información llega de varias fuentes y en diferentes formas. Esto trae como consecuencia una demanda de mejorar la red o los medios de transmisión final. Las redes de datos han registrado crecimientos muy significativos generados por nuevas formas de hacer llegar la información a los usuarios. En la actualidad las empresas luchan por mejorar sus procesos y su capacidad de poder competir en el mundo de las telecomunicaciones, razón por la cual se deben adoptar soluciones relacionadas con Internet, ya que las redes de cómputo sufren frecuentemente de congestión y colapsos importantes. Estos se producen porque en las redes habitualmente circula tráfico que hasta hace poco no era habitual, como los vídeos, audio, mensajería y multimedia.

En vía a solucionar esta problemática, las empresas deben mejorar sus sistemas de comunicación como estructuras organizadas, ya que nadie escapa de los avances tecnológicos.

Las redes locales fueron diseñadas, fundamentalmente, para soportar aplicaciones de procesamiento de datos que, aunque requieren un servicio de transporte fiable, no son muy exigentes en cuanto a otros parámetros de calidad de servicio como retardo, tasa de pérdidas, entre otros aspectos que se pueden mencionar. Sin embargo, el desarrollo de nuevas tecnologías de información y comunicación, ha permitido la evolución de las redes, dotándolas de la capacidad de satisfacer simultáneamente los requisitos de tráficos de muy diversa naturaleza. Esta situación favorece la aparición de nuevos servicios y aplicaciones para los cuales las características, anteriormente mencionadas, son esenciales.

Actualmente existen redes locales con gran cantidad de dispositivos conectados, lo que puede afectar a la velocidad de la red. Por otro lado, a diario se producen miles de ataques en Internet. La inmensa mayoría son ataques indiscriminados, buscando máquinas desde las que distribuir correo basura. Buscan también utilizar dichas máquinas como plataformas para lanzar nuevos ataques y dificultar cualquier seguimiento.

1.1 ¿Por qué realizar el análisis de nuestra red?

Debido a las problemáticas citadas anteriormente surge la necesidad de buscar una forma, por la cual, podamos obtener información acerca de nuestra red.

Hoy en día existe gran cantidad de software que permite analizar una red de una manera muy sencilla. Mucho software moderno permite enviar alertas a los administradores si se produce algún tipo de anomalía o cambio repentino en la red. Analizar nuestra red nos permite tener información del tráfico, de los dispositivos y de los servidores de nuestra red, ya sea una red corporativa, educacional o de otro tipo. Con ello podemos observar cosas tan importantes como:

- Verificar si tenemos algún componente caído (switches, routers, servidores).
- Comprobar si nuestra red se está ralentizando, dónde y por qué.
- Comprobar si nuestra red sufre de algún cuello de botella.
- Ver estadísticas del tráfico de red.

Como podemos observar hay grandes beneficios en tener un análisis de nuestra red, tanto de buen funcionamiento como de seguridad, por eso creemos que es importante mantener nuestra red analizada mediante un software.

1.2 Tipos de análisis

Existen dos formas de análisis principales [2]:

- **Análisis de paquetes:** Este es un análisis más en profundidad donde se utilizan los paquetes como fuentes de datos, de los que se extraen metadatos como nombres de aplicaciones y sitios web. Por lo tanto, este análisis al ser tan exhaustivo es ideal para analizar aplicaciones en donde podemos ver la información de los paquetes reales involucrados en las conexiones y así identificar la raíz de los problemas. El reconocimiento de aplicaciones mediante análisis de paquetes puede identificar el tráfico mediante la aplicación, incluso cuando se utilizan puertos inusuales o dinámicos.
- **Análisis de flujos:** Un flujo es un conjunto de paquetes que tienen identificadores comunes. Normalmente, un flujo se define por el tráfico que tiene la misma IP de origen y destino, puerto de origen y destino, protocolo, punto de observación y timestamp. Si alguna de estas variables cambia se establecerá un nuevo flujo.

Netflow, sflow e ipfix son diferentes formas de recompilar información sobre el tráfico que atraviesa una red.

El principal precursor de los análisis de flujo fue Cisco, que desarrolló el protocolo Netflow [3], que se ha convertido en un estándar de facto para la definicion de flujos de red. Posteriormente se introdujeron protocolos de distintas empresas:

Protocolo	**Empresa desarrolladora**
Netflow	Cisco
jFlow	Juniper
rFlow	Ericcson
NetStream	Huawei

Tabla 1.1: Protocolos NetFlow y sus fabricantes

1.3 Motivación

Es importante destacar que el precio del software ofrecido por las empresas para el análisis de flujos varían dependiendo de los servicios que se necesiten y elijan para un caso en concreto. Pero en general, resulta muy costoso económicamente, no tanto para grandes corporaciones sino para empresas de tamaño medio o pymes.

Es por eso que en este proyecto intentaremos acercar una solución web gratuita y de interfaz amigable que, aunque no cumpla todas las funcionalidades de un software profesional, sí permita la obtención de un informe y de la topología de una red dada.

1.4 Objetivos

Este proyecto pretende crear un sistema que consiste en una aplicación web, mediante la cual los usuarios, aportando un archivo formato .pcap, puedan obtener información relevante tanto de la topología de su red, como del tráfico que circula en ella, ya sea en forma de informes o en forma de gráficos estadísticos.

A continuación, se exponen las funcionalidades más relevantes:

- **Gestión de usuarios:** Los usuarios podrán registrarse en la aplicación y posteriormente acceder a ella para su uso.

- **Gestión de archivos:** Los usuarios podrán subir los archivos que deseen que sean analizados (o no). Estos se almacenarán en una carpeta con el nombre del usuario que subió

el archivo. Esta carpeta contendrá toda la información de todos los ficheros subidos por este usuario, así como de todos los archivos e informes generados por estos archivos. El usuario podrá ver en la aplicación el listado de los archivos que ha subido, así como visualizarlos o descargarlos.

- **Generador de archivos:** El usuario podrá elegir uno de los archivos subidos y cumplimentar una serie de parámetros para analizar su archivo y recibir la información procesada. Esto generará en su carpeta distintos archivos de información.

  Todos estos archivos serán generados en el propio ordenador del usuario para que tenga acceso a ellos en el caso de querer consultarlos posteriormente sin necesidad de tener acceso a Internet.

1.5 Estructura del proyecto

- Capítulo 2 - Fundamentos tecnológicos:

  Explicación de conceptos básicos de redes.

- Capítulo 3 - Herramientas y tecnologías utilizadas:

  Explicación de los lenguajes, herramientas de desarrollo, frameworks, etc, que se utilizaran a lo largo del proyecto.

- Capítulo 4 - Análisis de viabilidad:

  Explicación en todos los campos, de por que nuestro proyecto es viable.

- Capítulo 5 - Análisis de costes y metodología:

  Análisis de todos los tipos de costes del proyecto y explicación de la metodología e iteraciones utilizadas en el proyecto.

- Capítulo 6 - Introducción al desarrollo realizado:

  Explicación de la arquitectura básica del proyecto y de los patrones de diseño utilizados.

- Capítulo 7 - Iteración 3: Gestión de usuarios:

  Explicación de la tercera iteración, dividida en tres fases: análisis, diseño, implementación y pruebas.

- Capítulo 8 - Iteración 4: Gestión de archivos:

  Explicación de la cuarta iteración, dividida en tres fases: análisis, diseño, implementación y pruebas.

- Capítulo 9 - Iteración 5: Generación de gráficas e informes:

  Explicación de la quinta iteración, dividida en tres fases: análisis, diseño, implementación y pruebas.

- Capítulo 10 - Iteración 6: Generación de la topología:

  Explicación de la sexta iteración, dividida en tres fases: análisis, diseño, implementación y pruebas.

- Capítulo 11 - Datos utilizados:

  Explicación de los ficheros utilizados para asegurar un correcto estudio de la topología.

- Capítulo 12 - Conclusiones y trabajo futuro:

  Explicación de las conclusiones del proyecto y propuesta de posibles lineas de trabajo para el futuro.

Capítulo 2

Fundamentos tecnológicos

Utilizaremos este capítulo para afianzar algunos conceptos básicos de redes, los cuales, nos harán falta para comprender algunos apartados que se irán viendo a lo largo del proyecto.

2.1 Conceptos teóricos básicos sobre redes

Puesto que lo que vamos a tratar en el proyecto esta relacionado con análisis de redes y su topología, lo fundamental, es empezar definiendo que es una red.

Red de Área local (LAN)
Es un sistema de comunicación entre ordenadores que permite compartir información, con la característica de que la distancia entre las computadoras debe ser pequeña. Estas redes son usadas para la interconexión de computadores personales y estaciones de trabajo. Las redes de Área local intentan seguir el estándar modelo *OSI*.

Modelo OSI.

Ha sido desarrollado en 1984 por la organización ISO (*International Organization for Standarization*), su objetivo es conseguir interconectar sistemas de distinta procedencia con la finalidad de que puedan cambiar información sin ningún tipo de impedimento.

El modelo *OSI* está conformado por 7 capas o niveles de abstracción. Cada uno de estos niveles tendrá sus propias funciones, mas esta separación en capas hace posible la intercomunicación entre protocolos distintos al concentrar funciones especificas en cada nivel de operación.

Los niveles de los que se compone el modelo *OSI* son:

- **Física:** Este nivel se encarga directamente de los elementos físicos de la conexión. Gestiona los procedimientos a nivel electrónico para que la cadena de bits de información viaje desde el transmisor al receptor sin alteración alguna.
- **Enlace a datos:** Este nivel se encarga de proporcionar los medios funcionales para establecer la comunicación de los elementos físicos. Se ocupa del direccionamiento físico de los datos, el acceso al medio y especialmente de la detección de errores en la transmisión.
- **Red:** Este nivel se encarga de la identificación del enrutamiento entre dos o mas redes conectadas, haciendo que los datos puedan llegar desde el transmisor al receptor siendo capaz de hacer las conmutaciones y encaminamientos necesarios para que el mensaje llegue. Debido a esto, es necesario que esta red conozca la topología de la red en la que opera.
- **Transporte:** Este nivel se encarga de realizar el transporte de los datos que se encuentran dentro del paquete de transmisión desde el origen hasta el destino. Esto se realiza de forma independiente al tipo de red que haya detectado el nivel inferior.
- **Sesión:** Mediante este nivel se podrá controlar y mantener activo el enlace entre las maquinas que están transmitiendo información. De esta forma se asegurara que una vez establecida la conexión, esta se mantenga hasta que finalice la transmisión.
- **Presentación:** Esta capa se encarga de la presentación de la información transmitida. Asegurará que los datos que nos llegan a los usuarios sean entendibles a pesar de los distintos protocolos utilizados tanto en el recepto como en el emisor.
- **Aplicación:** Este nivel se encarga de permitir a los usuarios ejecutar acciones y comandos en sus propias aplicaciones. Permite también la comunicación entre el resto de capas inferiores.

2.1.1 Elementos de una red local

Una vez visto lo que es una Red de Área local, debemos de conocer algunos de los elementos que la conforman:

- **Servidor:** es el elemento básico de la red que consiste en un software capaz de atender peticiones y devolver una respuesta en concordancia. Los servidores se pueden ejecutar en cualquier tipo de ordenador, aunque se suelen ejecutar en ordenadores dedicados a esto, puesto que aportan más seguridad. Los servidores atienden peticiones de otros ordenadores (clientes) y les permiten la posibilidad de compartir datos, información y recursos de hardware y software.

- **Estaciones de trabajo:** son los equipos que participan en el entorno de red facilitando a los usuarios el acceso a los servidores y periféricos de la red. Suelen ser ordenadores destinados a un trabajo técnico o científico y están conectadas por medio de cables u otros medios no guiados a los servidores. Estos ordenadores han de estar dotados de una tarjeta de red, pues esta es la que maneja los protocolos de comunicación de cada topología específica.

- **Equipos de conectividad:** es un dispositivo electrónico que distribuye banda ancha a determinada cantidad de equipos de una red. Entre ellos destacan:

  - **Hub:** es un dispositivo que distribuye ancho de banda y proporcionan una conexión a nivel físico dedicada para cada dispositivo, lo que ayuda a reducir la posibilidad de que un fallo en un ordenador afecte al resto. El hub ignora que clase de contenido recibe, simplemente reenvía esta información por cada uno de los puertos.

  - **Bridge:** es un dispositivo que interconecta las redes y proporciona un camino de comunicación entre dos o más segmentos de red o subredes a nivel físico y de enlace de datos. Se suele utilizar para ampliar la extensión de la red o el número de nodos que la constituyen, aunque también para unir redes distintas y enviar paquetes entre ellas.

  - **Switch:** es un dispositivo que a nivel de enlace que mejora significativamente el rendimiento de la red en comparación con los hubs, esto se debe, a que ha diferencia de los hubs, proporcionan ancho de banda dedicado a cada dispositivo final, utiliza tablas de direcciones MAC para tomar decisiones de reenvío y utiliza tablas ASIC y CAM para aumentar la velocidad a la que las tramas pueden ser procesadas.

    Los switches utilizan lo mejor de los hubs y de los bridges a la vez que agregan mas capacidades. Utilizan la capacidad multipuerto del hub con el filtrado del puente, lo que permite que solo el destino vea el trafico de difusión.

  - **Router:** es un dispositivo que trabaja a nivel de enlace y es el encargado de interconectar dos o mas redes. Como mínimo se encarga de conectar una red LAN con una red WAN hacia nuestro ISP. Usa cabeceras y tablas de enrutamiento para determinar el mejor camino para que el paquete llegue a su destino.

2.1.2 Topología de Red

Se define como la forma física o lógica mediante la cual los dispositivos de una red se interconectan entre sí sobre un medio de comunicación. Está compuesta por dos partes: la topología física, que es la disposición real de los cables (los medios) y la topología lógica, que define la forma en que las estaciones de trabajo acceden a los medios. Existen diversas topologías físicas. A continuación procedemos a explicar las topologías de red en bus, en estrella, en estrella extendida, en anillo y en anillo doble.

Topología de red en bus

En esta topología, los elementos que constituyen la red se disponen linealmente, es decir, en serie y conectados por medio de un cable: el bus. Las tramas de información emitidas por un nodo (terminal o servidor) se propagan por todo el bus (en ambas direcciones), alcanzando a todos los demás nodos. Cada nodo de la red se debe encargar de reconocer la información que recorre el bus, para así determinar cuál es la que le corresponde, la destinada para él. En la **Figura** 2.1 podemos observar un ejemplo de esta topología.

Es el tipo de instalación más sencilla debido a la facilidad de implementación y crecimiento. Por el contrario, el rendimiento disminuye a medida que la red va creciendo, se producen altas perdidas en al transmisión debido a la colisión entre mensajes, existen limitaciones de longitudes físicas del canal, por lo que existe un limite de equipos dependiendo de la calidad de señal requerida. Por lo que, a pesar de ser la red mas sencilla de implementar, todavía tiene demasiadas desventajas para ser la mas utilizada.

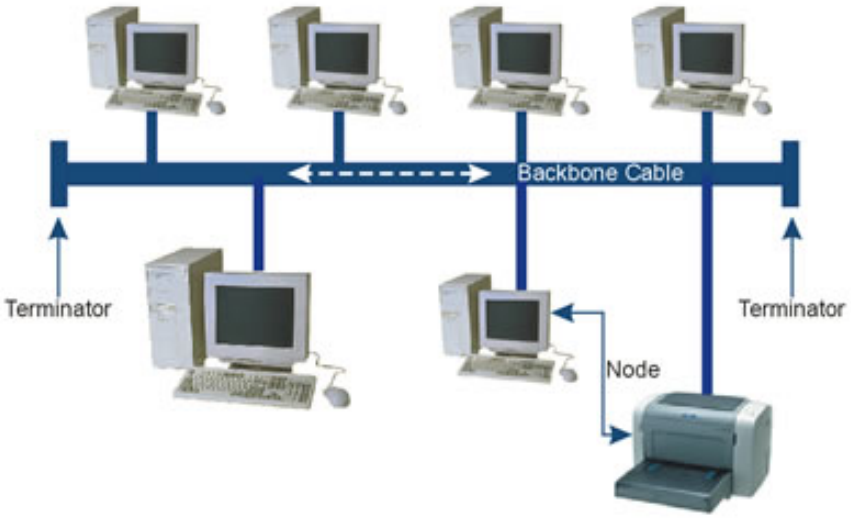


Figura 2.1: Topología de red bus

Topología de red en estrella

La topología de tipo estrella es una de las configuraciones más comunes. En ella cada nodo se conecta a un dispositivo de red central, como un hub o un switch. Esta topología tiene la principal característica de que la comunicación de todos los nodos pasa por un nodo central, es decir, que si se produce un fallo en el nodo central, todos los demás perderán la comunicación entre sí. Por lo tanto, en esta configuración es muy importante asegurar que las capacidades de este nodo serán las suficientes por el alto tráfico que deberá de soportar. En la **Figura** 2.2 [4] podemos ver un ejemplo de esta topología. Existe una variante de esta topología que se denomina estrella extendida.

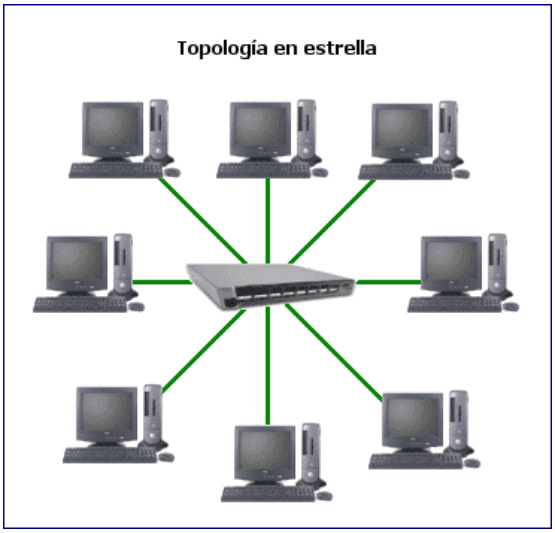


Figura 2.2: Topología de red en estrella (edgaracredes)

Topología de red en estrella extendida

La topología de tipo estrella extendida es igual a la topología en estrella, con la diferencia de que cada nodo que se conecta con el nodo central también es el centro de otra estrella. De tal forma que cada uno de los nodos finales de la estrella central, actúa como nodo central de su propia estrella.

La ventaja de esta topología reside en que el cableado a utilizar es más corto y limita la cantidad de dispositivos que se deben interconectar con cualquier nodo central. Esta es la forma de conexión más utilizada actualmente en los sistemas telefónicos. En la **Figura** 2.3 [5] podemos ver un ejemplo de esta topología.

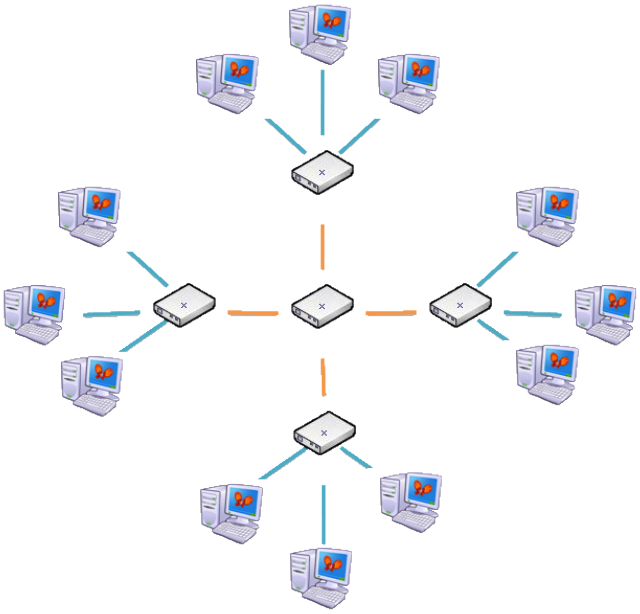


Figura 2.3: Topología de red en estrella extendida

Topología de red en anillo

Una topología en anillo es una configuración de red en la que las conexiones de dispositivos crean una ruta circular. En esta configuración cada nodo es transmisor y receptor al mismo tiempo, pasando las señales de una estación a otra.

Cabe destacar que en esta topología existe el conocido ”paso de token” o testigo, en el que, solo el ordenador que este en posesión del token tiene permiso para transmitir, esto se utiliza debido a que se necesita saber si el tráfico ya ha pasado por un nodo determinado. En la **Figura** 2.4 [6] podemos ver un ejemplo de esta topología. Existe una derivación de esta topología denominada red de anillo doble.

Figura 2.4: Topología de red en anillo

Topología de red en anillo doble

Una topología en anillo doble consta de dos anillos concéntricos, donde cada host de la red está conectado a ambos anillos, aunque los dos anillos no están conectados directamente entre sí. Es análoga a la topología de anillo, con la diferencia de que, para incrementar la confiabilidad y flexibilidad de la red, hay un segundo anillo redundante que conecta los mismos dispositivos. La topología de anillo doble actúa como si fueran dos anillos independientes, de los cuales se usa solamente uno en cada momento. En la **Figura** 2.5 [7] podemos ver un ejemplo de esta topología.

Debido a que el objetivo de este proyecto es el análisis de flujos, explicaremos la arquitectura básica de su monitorizaje, la cual se puede distribuir en varias fases:

1. **Observación de paquetes:** En este proceso se capturan los paquetes en línea haciendo uso de APIs o librerías y se preprocesan para su uso posterior.

2. **Medición y exportación de flujos:** En el proceso de medición los paquetes se agregan en flujos. Un flujo es un conjunto de paquetes IP que pasan por un punto de observación en la red y tienen un conjunto de propiedades en común. Una vez un flujo se considera terminado, se crea un registro de este, el cual es incluido en un datagrama y exportado.

3. **Recopilación de datos:** Las principales tareas de este proceso son la recepción, almacenamiento y preprocesamiento de los datos de flujo generados por la etapa anterior. Las operaciones mas comunes de preprocesamiento incluyen agregación, filtrado, compresión de datos y generación de resúmenes.

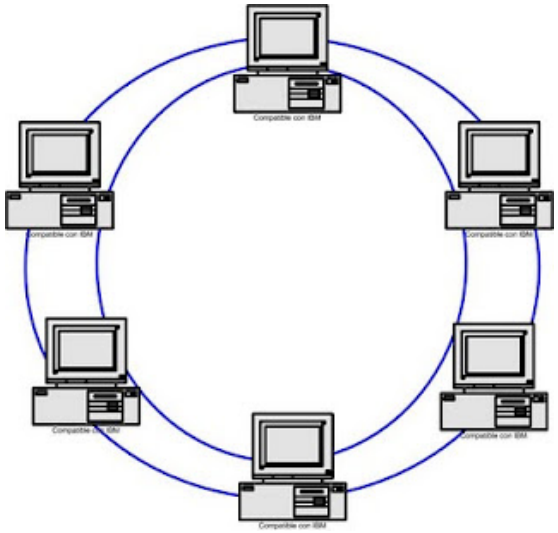


Figura 2.5: Topología de red en anillo doble

4. **Análisis de datos:** Esta etapa final puede ser tanto un análisis manual utilizado con naturaleza exploratoria, como uno automatizado cuya naturaleza es la recopilación de datos. Las funciones de análisis mas comunes incluyen correlación, agregación, perfilado, clasificación, detección de anomalías e intrusiones y búsquedas de archivos para fines de investigación forense o de otro tipo.

En las fases del proceso explicado anteriormente, se utilizan los siguientes componentes:

- **Sondas:** Son dispositivos que se encargan de la fase de observación de paquetes, puede no haber ninguna sonda en un proceso de análisis de flujo.
- **Exportadores:** Son dispositivos como routers y switches, estos comúnmente se encargan de las fases 1 y 2. Los exportadores utilizan una caché para almacenar la información sobre los flujos que van capturando a medida que el tráfico entra y sale a través del switch o enrutador.
- **Colectores:** Son los encargados de recibir los metadatos de los exportadores (fase 3), almacenarlos y preprocesarlos.
- **Analizadores:** Son elementos encargados de permitir el análisis sobre la información contenida en los colectores.

Capítulo 3

Herramientas y tecnologías utilizadas

En este capitulo hablaremos sobre las herramientas y tecnologías usadas a lo largo del proyecto y daremos una breve introducción de las mismas.

3.1 Lenguajes de programación

3.1.1 Java

Java [8] es un lenguaje de programación de propósito general, corriente, basada en clases y orientado a objetos. Es un lenguaje de estructura sencilla, por lo que puede ser utilizado por la mayoría de usuarios de forma fluida. Puede ser ejecutado en diversos sistemas operativos y está pensado principalmente para entornos de producción.

Es un lenguaje de alto nivel. Incluye gestión automática de almacenamiento, con un recolector de basura para evitar, de esta forma, problemas de seguridad de asignación explícita. Java no incluye estructuras inseguras, como puede ser el acceso a *arrays* sin comprobar índices.

Java Platform es una plataforma de desarrollo de aplicaciones Java para el ámbito empresarial. El objetivo de esta herramienta es proveer a los desarrolladores de una potente colección de *APIs* para acortar el tiempo de desarrollo y reducir la complejidad de la aplicación.

3.1.2 HTML

HTML (HyperText Markup Language) [9] es un lenguaje de marcado de hipertexto. Es el estándar en la visualización de páginas web y el que todos los navegadores actuales adop-

taron. Originalmente, *HTML* fue diseñado como un lenguaje para describir semánticamente documentos científicos. Con todo, su diseño general permitió que se adaptara, en los años posteriores, para describir una serie de otros tipos de documentos e incluso aplicaciones.

Es un estándar a cargo del *World Wide Web Consortium (W3C)*. *HTML* se considera el lenguaje web más importante. Es un lenguaje que se estructura en forma de etiquetas.

3.1.3 CSS

CSS (Cascading Style Sheets) [10] es un lenguaje utilizado para describir la presentación de documentos HTML o XML. Esto incluye varios lenguajes basados en XML, como son el XHTML o SVG. Describe cómo debe ser renderizado el elemento estructurado en pantalla, en papel o en otros medios.

CSS pertenece a la base de lenguajes de Open Web y posee una especificación estandarizada por parte del W3C.

3.1.4 LaTex

Latex [11] es un sistema de preparación de documentos para la composición tipográfica de alta calidad. Normalmente se usa para documentos técnicos o científicos de tamaño mediano o grande, pero es posible utilizarlo para cualquier forma de publicación.

Latex está basado en la filosofía de animar a los autores, de forma que estos se centren más en el contenido del documento que en la apariencia y los estilos.

3.2 Frameworks y librerías

3.2.1 Spring

Spring Framework [12] es una solución ligera destinada a la construcción de aplicaciones java centrándose solo en la propia aplicación, delegando en *Spring* el manejo de la infraestructura. Con todo, *Spring* es modular, permite usar solo las partes que se necesitan, sin tener que depender del resto.

Permite emplear la Inversión de Control (IoC) y soporta la gestión de forma declarativa de la transacionalidad, el acceso remoto a la lógica a través de RMI, servicios web y diferentes opciones para preservar los datos. Ofrece un marco MVC (*Model-View-Controler*) completo y permite integrar AOP (Aspect Oriented Programming) de forma transparente en el software.

Spring está compuesto por diferentes características organizadas en 20 módulos. Estos se agrupan en 6 principales: Core Container, Data Access/Integration, Web, AOP, Instrumentation y Test.

En este proyecto se emplearon los siguientes módulos:

- **Core:** Proporciona las partes fundamentales, como son el IoC o la inyección de dependencias
- **Web:** Es un módulo dentro del proyecto que proporciona integración básica de las características relacionadas con la web.
- **Test:** Gracias a este módulo disponemos de un entorno de pruebas, con la posibilidad de datos Mock, sin necesitar servidores de producción.
- **Security:** Es un módulo que nos proporciona métodos de autenticación.

Spring Boot

Spring Framework con el paso de los años comenzó a volverse demasiado complejo. Es necesario pasar por un largo proceso de requisitos para comenzar un nuevo proyecto.

Uno de los principales problemas es el proceso de configuración. Inicialmente se utilizaba XMLs, aunque en posteriores versiones se añadió la configuración basada en java, como una opción mas sencilla a los XMLs. Algunos de estos procesos de configuración son el manejo de transacionalidad, Spring MVC, servlets, etc.

A pesar de las continuas mejoras, estos procesos de configuración eran un proceso necesario para el comienzo del proyecto y durante el mismo. Esto se conoce como ”fricción del desarrollo”, es decir, todo el tiempo que se dedica a la configuración que no pertenece a la lógica de negocio, el desarrollo propiamente dicho de la aplicación. *Spring Boot* cambia este proceso.

Spring Boot [13] facilita la creación de aplicaciones *stand-alone* basadas en *Spring*. Se encarga del proceso de configuración, de modo que el usuario pueda centrarse en la lógica de negocio. A pesar de esto, siempre se necesita un pequeño grado de configuración.

Algunas de las ventajas son:

- Reduce el tiempo de desarrollo, aumentando de esta forma la productividad.
- Evita especificar anotaciones y configuraciones XML.

- Simplifica la integración con ecosistemas Spring, ORMs, Spring Security, etc.
- Proporciona servidores HTTP, como Tomcat o Jetty.
- Proporciona CLI (Command Line Interface) para desarrollar o probar las aplicaciones desde un prompt.
- Incluye *plugins* para trabajar con bases de datos embebidas o en memoria.

3.2.2 Thymeleaf

Thymeleaf [14] es un moderno motor de modelos *Java Server-Side*, tanto para web como para entornos *Standalone*, capaz de procesar *HTML,XML, JavaScript, CSS* o texto plano.

El objetivo principal de *Thymeleaf* es proporcionar una forma elegante y muy mantenible de crear modelos. Para conseguir esto, se basa en el concepto de *Plantillas Naturales* para inyectar su lógica de ficheros de una forma que no afecta al modelo de ser empleado como un prototipo de diseño. Gracias a esto, se mejora la comunicación de diseño y se reduce el espacio entre equipos de desarrollo y de diseño.

3.2.3 JFreeChart

JFreeChart [15] es una librería de java que permite a los desarrolladores generar y mostrar gráficos a partir de datos en sus aplicaciones. Es una librería OpenSource y está distribuida bajo los términos de *GNU Lesser General Public Licence (LGPL)*, lo cual permite su uso en aplicaciones propietarias.

Fue creada por David Gilbert y actualmente es la librería más usada en Java para generar diferentes tipos de gráficos.

3.2.4 vis.js

vis.js [16] es una librería de visualización dinámica basada en *Javascript*. Esta librería permite manejar grandes cantidades de datos dinámicos y permite la manipulación de estos.

Permite crear diferentes tipos de gráficos dinámicos a partir de datasets o datos importados de jsons o lenguaje DOT.

3.3 Herramientas de desarrollo

3.3.1 Eclipse IDE

Eclipse [17] es una plataforma software compuesta por un conjunto de herramientas de desarrollo de código abierto, que se pueden emplear para crear aplicaciones como sitios web, programas JAVA, C++, etc.

Creada inicialmente por IBM, pero actualmente mantenida por la fundación Eclipse, una organización independiente sin ánimo de lucro.

Una de las características principales del IDE Eclipse es la de poder ampliar sus capacidades mediante módulos (*plugins*), en contra de otro tipos de entornos monolíticos.

3.3.2 Maven

Apache Maven [18] es una herramienta para automatizar la construcción, principalmente de proyectos Java. *Maven* aborda dos aspectos del software de construcción. En primer lugar, describe cómo se construye un software y, en segundo lugar, describe sus dependencias. Utiliza convenciones para el procedimiento de compilación.

Un fichero XML (POM) describe el proyecto de software que se esta construyendo, sus dependencias de otros módulos y componentes externos, el orden de compilación, directorios y complementos necesarios. Incluye objetivos predefinidos para realizar ciertas tareas bien definidas, como la compilación de códigos y su embalaje.

Una de las características principales es la descarga de forma dinámica de las bibliotecas Java y de los complementos *Maven* de uno o varios repositorios, como el *Maven Central Repository,* y el guardado localmente.

3.3.3 Git

Git [19] es un sistema de control de versiones distribuido, caracterizado principalmente por su velocidad, rendimiento, flexibilidad y usabilidad. Diseñado por Linus Torvalds, fue creado inicialmente con el propósito de ser el sistema de control de versiones para el kernel de Linux.

3.3.4 Overleaf

Overleaf [20] es una empresa social que construye herramientas modernas de creación colaborativa para científicos. El producto principal es un editor colaborativo en línea a tiem-

po real para trabajos, tesis, informes técnicos y otros documentos escritos en el lenguaje de marcas LaTex.

Overleaf tuvo una rápida adopción en la ciencia y en la investigación, utilizada por instituciones importantes como Stanford y CalTech. Además de la utilización por parte de investigadores, también pasó a formar parte de muchas instituciones académicas.

El objetivo principal es abordar los problemas que surgen a la hora de redactar artículos de colaboración y poder hacer más accesible LaTeX para usuarios de cualquier nivel.

3.4 Sistemas de Gestión de Bases de Datos

3.4.1 MySQL

MySQL [21] ofrece un servidor de base de datos SQL (*Structured Query Language*) muy rápido, multihilo, multiusuario y robusto. Está diseñado para los sistemas de producción de gran carga, así como para incorporar a software desplegado de forma masiva. Es posible utilizar MySQL bajo dos licencias distintas: puede ser utilizado como producto Open Source, bajo las condiciones de la licencia GNU (*General Public License*), o puede ser adquirida la versión comercial estándar de Oracle.

3.5 Otros

3.5.1 HTTP

El protocolo de transferencia de hipertexto (*HTTP*) [22] es un protocolo a nivel de aplicación para sistemas de información hipermedia distribuidos y colaborativos. Es un protocolo genérico, sin estado, que se puede usar para muchas tareas más allá de su uso para hipertexto, como servidores de nombres y sistemas de gestión de objetos distribuidos, a través de la extensión de sus métodos de solicitud, códigos de error y cabeceras.

HTTP se utiliza en la iniciativa de información global World-Wide Web desde 1990. Esta especificación define, entre otros, el protocolo denominado ”HTTP / 1.1”, que es el más usado.

3.5.2 REST

Represental State Transfer (*REST*) [23] es un estilo de arquitectura Web con unas características y restricciones definidas por Roy Fielding en su tesis doctoral.

En la actualidad, es muy sencillo encontrar arquitecturas REST APIs de servicios Web modernos. Una API REST consiste en un conjunto de recursos interconectados, sobre los cuales se pueden realizar diferentes operaciones estándar.

3.5.3 Softflowd

Softflowd [24] es un software de monitorización de tráfico de red basado en flujos. *Softflowd* lee el tráfico de la red y recopila información sobre los flujos de tráfico activos. El uso previsto de este software es el de monitorización de tráfico de flujos del sistema Netflow de Cisco.

Softflowd admite la exportación de datos utilizando las versiones 1, 5 o 9 del protocolo Netflow. Se puede utilizar para mostrar estadísticas, pero estas estadísticas son poco útiles, por lo que su uso realmente es el de depuración.

Softflowd intenta rastrear solo los flujos de tráfico activos. Cuando un flujo está inactivo durante un periodo de tiempo expira automáticamente. Los flujos también pueden expirar anticipadamente si el recuento de su tráfico excede los 2GB o si el número de flujos que se rastrean excede el parámetro *maxflows*, que por defecto es 8192. En este caso, los flujos más antiguos caducan primero.

3.5.4 Nfdump

Nfdump [25] es un conjunto de herramientas para recopilar y procesar datos NetFlow y sFlow, enviados desde dispositivos compatibles con NetFlow/sflow. El conjunto de herramientas es compatible con NetFlow v1, v5, v7, v9, IPFIX y sFflow. Nfdump es compatible tanto con IPv4 como con IPv6.

- **nfcapd** es el daemon colector de flujos netflow de las herramientas nfdump. Lee los datos netflow de la red enviados por los exportadores (Softflowd en nuestro caso) y los almacena en archivos. Se crea un nuevo archivo cada n minutos, por defecto 5. El nuevo archivo se nombra con respecto a las marcas de tiempo del intervalo.

  nfcapd soporta Netflow en sus versiones v1, v2, v7 y v9. También soporta IPFIX.

- **nfdump** es el procesador de los archivos netflow recolectados. Nfdump lee los datos netflow de uno o varios archivos almacenados por *nfcapd*. Su sintaxis de filtro es similar a tcpdump (archivos pcap) pero adaptada para netflow. Nfdump muestra los datos netflow y/o crea las N principales estadísticas de flujos, bytes y paquetes. Nfdump tiene una agregación de flujo potente y flexible que incluye flujos bidireccionales. El formato de salida lo puede seleccionar el usuario e incluye un formato csv simple para el procesamiento posterior.

3.5.5 Tcpreplay

Tcpreplay es un conjunto de utilidades con licencia GPLv3 para sistemas operativos UNIX para editar y reproducir el tráfico de red que previamente fue capturado por herramientas como tcpdump o wireshark. Le permite clasificar el tráfico como cliente o servidor, reescribir paquetes de capa 2 3 y 4 y finalmente reproducir el tráfico en la red a través de otros dispositivos como computadores, enrutadores, firewalls, NIDS e IPS.

Capítulo 4

Análisis de viabilidad

El análisis de viabilidad es un proceso fundamental para cualquier tipo de proyecto, ya que sirve para determinar las probabilidades de finalizarlo con éxito.

Se lleva a cabo antes del comienzo de la realización del proyecto en sí, debido a que dependiendo de dicho análisis, se valora, de forma real, la viabilidad del mismo. Es un proceso crítico.

Dicho análisis está dividido en tres puntos: **económica, técnica** y **de mercado.**

4.1 Viabilidad económica.

En primer lugar, se realiza una viabilidad económica del proyecto. Existen tres costes principales en los que apoyar dicha viabilidad. Estos son:

- Recursos humanos.
- Recursos software.
- Recursos hardware.

Los **recursos humanos**, en el caso de este proyecto, se reducen a una sola persona que se encargará de todas las fases de desarrollo, por lo que el coste será bajo.

En relación a los **recursos software**, como el objetivo de este proyecto es la realización de una aplicación web, existen múltiples opciones de software *open source* adecuado para dicha realización. Utilizando este tipo de software libre, conseguimos que los costes se reduzcan significativamente, de manera muy sencilla.

Por último, en cuanto a los **recursos hardware**, tan solo será necesario un ordenador con conexión a Internet.

Una vez estudiado el impacto de los costes de los recursos citados anteriormente, podemos asumir que el proyecto es económicamente viable, debido a que la inversión necesaria es muy baja.

4.2 Viabilidad técnica

Como vimos en el capítulo anterior, las herramientas y las tecnologías utilizadas en la realización del proyecto son ampliamente conocidas y utilizadas en la comunidad de desarrollo de software. Por lo tanto utilizarlas para el proyecto es sinónimo de bajo riesgo. Otro punto a tener en cuenta es la experiencia previa que tenga el desarrollador en dichas tecnologías, haciendo así posible reducir incluso más el riesgo.

Teniendo en cuenta esto, podemos asumir que el proyecto es viable.

4.3 Viabilidad de mercado

En la actualidad, existen diversas opciones de análisis de flujos en el mercado, pero la mayoría son de pago y de un elevado coste (por ejemplo, ”Netflow Analyzer”).

Por otra parte existen otras tecnologías que, combinadas, pueden acercarnos algunas prestaciones que nos aportan las herramientas de pago. Un ejemplo de esto serían las tecnologías combinadas de ”elasticsearch, kibana y logstash” las cuales hemos probado anteriormente. Mas hemos comprobado que la instalación y puesta en marcha de estas puede llegar a ser complicada y tediosa por la cantidad de configuraciones que hay que llevar a cabo.

Nuestra aplicación pretende llevar a cabo prestaciones similares a la que nos aportan dichas tecnologías combinadas en una aplicación web. De esta forma el usuario solo necesita un navegador y los datos que desea analizar, sin necesidad de instalaciones ni pagos.

Debido a la aportación de prestaciones de forma gratuita y sencilla tras el desarrollo del proyecto, podemos observar que dicho proyecto es viable en el mercado y que está orientado principalmente a la pequeña y mediana empresa, que no puede permitirse el sobrecoste de un analizador de flujos tan potente.

Capítulo 5

Análisis de costes y metodología

En este capítulo hablaremos de los costes económicos que tendrá el proyecto.

La duración prevista del proyecto será aproximadamente de 2 meses, dando comienzo el 10/07/2020 y terminando el 01/09/2020 con un total de 408 horas trabajadas. La jornada laboral se compone de 8 horas diarias de trabajo.

Con respecto a los recursos utilizados para el desarrollo del proyecto, identificaremos dos tipos:

5.1 Recursos humanos

Existen varias tareas a realizar. Dependiendo de qué situación deba afrontar, tenemos los siguientes roles:

- **Analista:** Se encargará de las tareas de análisis y diseño de la aplicación, esta tarea será desempeñada por los tutores del proyecto.
- **Programador:** Su labor será la de implementar las funcionalidades que contempla nuestra aplicación, así como las pruebas. Esta tarea será realizada por el alumno.

En la tabla 5.1, se muestra un coste aproximado de cada uno de ellos por hora, este coste ha sido extraído del convenio colectivo actual [26].

Nombre del recurso	Capacidad máxima	Tasa estándar
Analista	100%	14’9€/hora
Programador	100%	14€/hora

Tabla 5.1: Costes de empleados

5.2 Recursos Materiales

Por otro lado, en el caso de los recursos materiales, necesitaremos:

- **Equipo informático:**
  - **Procesador:** Intel i7-6700K 4.0Ghz = 331€
  - **Placa base:** Asus Z170-A = 150€
  - **Memoria Ram:** Corsair Vengeance LPX DDR4 3000 2x16 = 205€
  - **Refrigeración líquida:** Corsair Cooling Hydro Series H110i = 125€
  - **Disco duro SSD** Samsung 750 Evo SSD Series 250GB SATA3 = 70€
  - **Disco duro HDD** Seagate Barracuda 7200.14 3TB SATA3 64MB = 85€
  - **Caja** Phanteks Eclipse P400 Negra Con Ventana = 65€
  - **Ventilador** Tacens Valeo V 700W 80 Plus Silver Modular = 75€
  - **Tarjeta gráfica** Asus Dual GeForce GTX 1070 OC 8GB GDDR5 = 480€
- **Sistema operativo:** Ubuntu 19.04 LTS 0€
- **IDE:** Eclipse IDE 0€
- **Otros softwares:** 0€

El presupuesto en materiales necesario para la realización del proyecto se resumen en 5.2.

Recurso	Coste
Hardware	1586€
Software	0€
Total	1586€

Tabla 5.2: Costes de materiales

El presupuesto del equipo informático utilizado sobrepasa considerablemente los requisitos necesarios para realizar este proyecto, es decir, podría utilizarse un equipo más asequible. En nuestro caso era el único que se podía utilizar.

5.3 Planificación inicial y coste

Se conoce como planificación a proporcionar un marco de trabajo que nos permita hacer estimaciones razonables de los recursos y de sus costes, lo que significa que supone un paso clave en la elaboración del proyecto.

En 5.3 se indican los diferentes costes de los recursos humanos previstos para el desarrollo de este:

Tarea	**Horas**	**Rol**	**Total**
Análisis	100	Analista	1490€
Diseño	70	Analista	1043€
Implementación	200	Programador	2800€
Pruebas	40	Programador	560€
Total	410	Ambos	5893€

Tabla 5.3: Presupuesto estimado

Una vez finalizado el proyecto los costes cambiaron un poco, pues algunas pruebas tuvieron desviaciones, tanto positivas como negativas. Esto fue debido, sobre todo, a problemas con algunas librerías utilizadas para el proyecto. Tendría el resultado final en la tabla 5.4

Tarea	**Horas**	**Rol**	**Total**
Análisis	80	Analista	1192€
Diseño	90	Analista	1341€
Implementación	230	Programador	3220€
Pruebas	60	Programador	840€
Total	460	Ambos	6593€

Tabla 5.4: Presupuesto final

5.4 Metodología empleada

La metodología utilizada en este proyecto software está basada en un proceso iterativo e incremental basado en Scrum.

5.4.1 Scrum

Scrum [27] es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En *Scrum* se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está parcialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la flexibilidad y la productividad son fundamentales.

Scrum utiliza una aproximación iterativa, es decir, una secuencia de pasos incrementales o iteraciones. Cada iteración incluye alguna, o la mayoría, de las disciplinas de software (requisitos, análisis, diseño, implementación, etc.). Cada iteración tiene bien definidos un conjunto de objetivos y produce una parte parcial del trabajo a implementar en el sistema final. Las siguientes iteraciones se basan en las anteriores y se construyen a base de ellas, para poder así refinar el sistema.

En la metodología *Scrum* existen tres roles importantes:

- **Product Owner:** Es el responsable de maximizar el valor del trabajo del equipo de desarrollo. La maximización del valor del trabajo viene de la mano de una buena gestión del *Product Backlog*, del cual hablaremos mas adelante.

  El *Product Owner* es el unico perfil que habla constantemente con el cliente, lo que le obliga a tener muchos conocimiento sobre negocio.

- **Scrum master:** Es el responsable de que las técnicas Scrum sean comprendidas y aplicadas en la organización. Es el que se encarga de eliminar impedimentos o inconvenientes que tenga el equipo dentro de un sprint aplicando las mejores técnicas.

- **Equipo de desarrollo:** Es un equipo multifuncional y auto-organizado encargado de realizar las tareas priorizadas por el Product Owner sin dejarse influenciar por nadie.

Los equipos de desarrollo no tienen sub-equipos ni especialistas.

El desarrollo iterativo de Scrum se realiza en un sprint, el cual contiene los siguientes eventos:

- **Sprint:** Todo lo que ocurre en una iteración esta dentro de un sprint. La duración máxima es de un mes, el tiempo se determina en base al nivel de comunicación que el cliente quiere tener con el equipo.
- **Sprint planning:** En esta reunión todo el equipo Scrum define qué tareas se van a abordar y cuál serña el objetivo del sprint. En esta reunion el equipo se hace principalmente dos preguntas:
  - **¿Que se va a hacer en el sprint?** En base a ello, se eligen tareas del Product backlog.
  - **¿Como lo vamos a hacer?** El equipo define las tareas necesarias para completar cada item elegido del Product Backlog.

  Esto va a hacer que el equipo tenga un objetivo y se encuentre comprometido con la entrega de valor que se hara al cliente al final del sprint. Esto se llama sprint goal.

  El resultado de esta reunion es el spring goal y un sprint backlog.
- **Daily meeting:** Es una reunión diaria dentro del sprint con una duración máxima de 15 minutos. En ella deben de participar obligatoriamente el equipo de desarrollo y el scrum master.

  En esta reunion se busca responder a las siguientes preguntas:
  - ¿Que hice ayer?
  - ¿Que voy a hacer hoy?
  - ¿Tengo algún impedimento que necesito que me solucionen?
- **Sprint review:** La review del valor que vamos a entregar al cliente se hace en esta reunión, al final de cada sprint. Su duración es de 4 horas para sprints de un mes, y es la única reunión de Scrum a la que puede asistir el cliente. En ella el Product Owner presenta lo desarrollado al cliente y el equipo de desarrollo muestra su funcionamiento. El cliente valida los cambios realizados y además brinda feedback sobre nuevas tareas que el Product Owner tendrá que agregar al Prodcut backlog.

- **Sprint retrospective:** LA retrospectiva es el último evento Scrum, tiene una duración de 3 horas para sprints de un mes, y es la reunión del equipo en la que se hace una evaluación de como se ha implementado la metodología Scrum en el ultimo sprint.

  Es una gran oportunidad para el equipo Scrum de inspeccionarse a sí mismo, proponiendo mejoras par el siguiente sprint.

Las herramientas utilizadas en Scrum están definidas para maximizar la transparencia dentro del equipo, es decir, que todos tengan una misma visión de lo que esta ocurriendo en el proyecto.

Las herramientas principales de Scrum son:

- **Product blacklog:** Básicamente, el product backlog es el listado de tareas que engloba todo un proyecto. Cualquier cosa que debamos hacer debe estar en el product backlog y con un tiempo estimado por el equipo de desarrollo.

  La responsabilidad exclusiva de ordenar el product backlog es del Product Owner, que se encuentra en constante comunicación con el cliente para asegurarse de que las prioridades están bien establecidas.

  La ordenación también es 100% responsabilidad del Product Owner, por lo que las tareas que están más arriba deben de ser las de mayor prioridad.

  El equipo de desarrollo elige tareas del product backlog en el sprint planning para generar tanto el sprint backlog como el sprint goal.

- **Sprint blacklog:** Es el grupo de tareas del product backlog que el equipo de desarrollo elige en el sprint planning junto con el plan para poder desarrollarlas. Debe ser conocido por todo el equipo, para asegurarse de que el foco debe estar en este grupo de tareas.

  El sprint planning no cambia durante el sprint, solo se permite cambiar el plan para poder desarrollarlas.

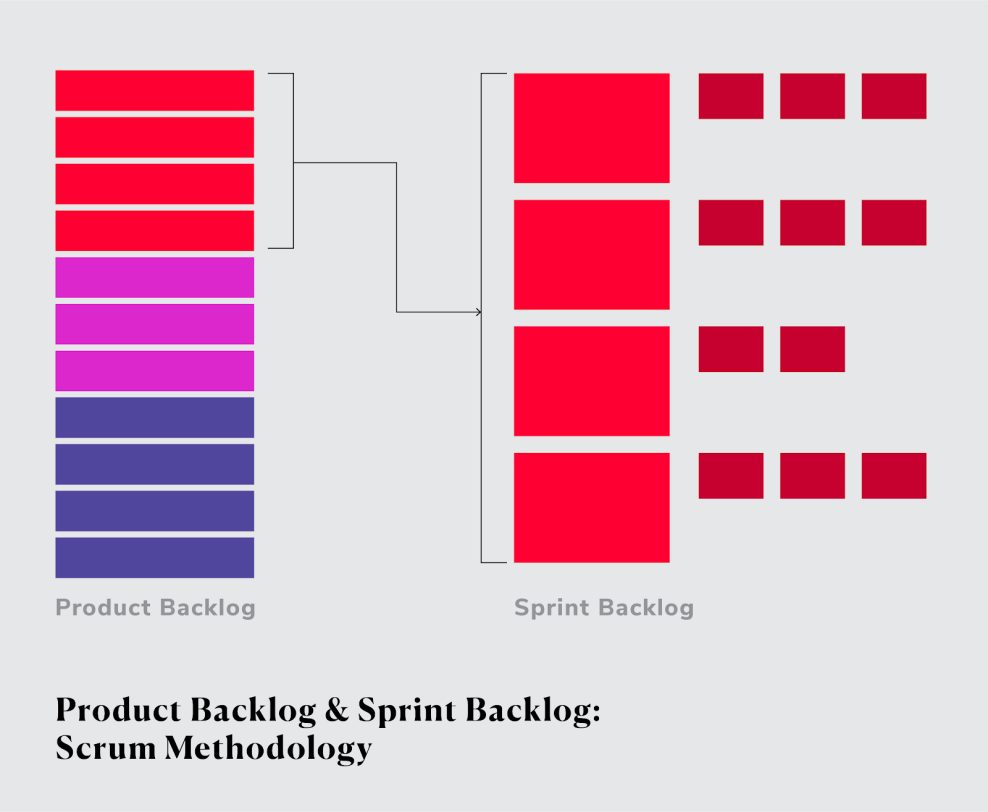


Figura 5.1: Product Backlog y Sprint Backlog [1]

5.5 Iteraciones

Mediante la metodología descrita en la sección anterior, se dividió el proyecto en 6 iteraciones. En cada iteración se incorporan nuevas funcionalidades a la anterior. A continuación se explicarán brevemente las iteraciones. Aunque a partir de la tercera se explicarán en capítulos a parte por motivos de tamaño y estética del documento. Las iteraciones serán las siguientes:

- **Iteración 1 *(60 horas)*: Estudio de las tecnologías a utilizar:** Realizamos una reunión en la que se han establecido diferentes posibilidades de tecnologías y lenguajes que podrían utilizarse en el proyecto. Una vez expuestas todas las posibilidades, se realizó un estudio y una toma de contacto con la mayoría de estas para así reconocer cuáles se utilizarían posteriormente en el proyecto.

- **Iteración 2 *(30 horas)*: Estudio de los casos de uso:** Una vez escogidas las tecnologías que se utilizarían en el proyecto, el siguiente paso fue establecer los casos de uso que debería tener el proyecto, así como la instalación de un entorno funcional con los

arquetipos, dependencias básicas y herramientas necesarias para afrontar el inicio del proyecto.

- **Iteración 3 *(110 horas)*: Gestión de usuarios):** Crearemos ventanas de registro y autenticación con la finalidad de separar los archivos de cada usuario en carpetas diferentes.
- **Iteración 4 *(40 horas)*: Gestión de Archivos:** Implementaremos la subida, descarga y visualización de archivos en formato pcap por parte del usuario.
- **Iteración 5 *(90 horas)*: Generación de gráficas e informes:** Implementaremos un método de procesamiento utilizando tecnologías externas, gracias al cual obtendremos la información que le ofreceremos al usuario.
- **Iteración 6 *(70 horas)*: Generación de la topología:** Implementaremos un método mediante el cuál se generara la topología de la red del usuario mediante los datos obtenidos en la iteración 5.

Capítulo 6

Introducción al desarrollo realizado

6.1 Arquitectura global del sistema

La arquitectura global del sistema se creó empleando el patrón de diseño **Modelo-Vista-Controlador** o **MVC**. Esta arquitectura tiene una amplia aceptación para el desarrollo de software corporativo. Pretende dividir el sistema en tres capas diferentes que se encargan de la lógica de control de la interfaz y el acceso a datos. Esto facilita el mantenimiento y la evolución de los sistemas dependiendo de la dependencia de las clases actuales en cada capa. Se emplea ampliamente en aplicaciones web en las que el usuario interactúa con el sistema y este responde.

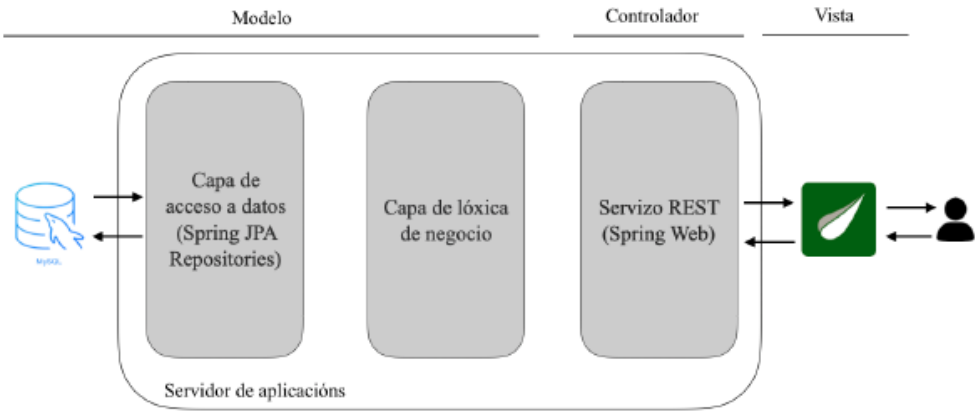


Figura 6.1: Esquema MVC

Como su propio nombre indica, cuenta con tres módulos diferentes. Por una parte, encontramos el modelo **vista**, que se encarga de presentarle al usuario la información del sistema y responder a las solicitudes. Por otras parte, encontramos el **controlador**, cuya principal función es la de procesar las peticiones realizadas por el usuario al interactuar con el sistema. El controlador se encarga de la comunicación entre las capas de la vista y del modelo. Por último, está el **Modelo**, que es la parte del sistema que contiene la lógica de negocio.

En la **Figura** 6.1 podemos observar que, dentro de las partes del patrón MVC, se dividió el sistema en diferentes grupos lógicos. Dentro de estos grupos, se utilizan distintas tecnologías, lenguajes y frameworks. A continuación, se detallará cada uno de los grupos:

- **Capa de acceso a datos:** Es la capa lógica del subsistema que se encarga de acceder a la base de datos, *MySQL* en el caso de este proyecto, y recupera todos los datos necesarios para utilizar en la capa de lógica de negocios. Esta capa está codificada en Java y se utiliza la tecnología Spring JPA (*Java Persistance API*), con la que podemos mapear los objetos Java a las entidades de la base de datos y realizar operaciones sobre ellos a través de los repositorios.
- **Capa de lógica de negocio:** En esta capa se desenvuelve la lógica interna de los casos de uso y las funcionalidades de la aplicación. Está desarrollada en Java, apoyándose en Spring para el manejo de dependencias y transacionalidad. Utiliza los repositorios que se comunican con la capa de acceso a datos.
- **Servicio REST:** Esta capa se encargará de realizar las comunicaciones entre la vista y la capa de lógica de negocio. Recibe peticiones HTTP provenientes de la vista, las procesa ayudándose de la capa de lógica de negocio, que a su vez se ayuda de la capa de acceso a datos, y le devuelve el resultado de nuevo a la vista. Está desarrollada en Java con la ayuda del modulo Spring Web MVC de Spring, encargado de la anotación de los métodos de los controladores que actuarán como manejadores de las operaciones definidas.
- **Vista:** En esta capa se encuentra la lógica de la interfaz de usuario. Esta se encarga de procesar las peticiones que el usuario realiza sobre la interfaz, delegando en el servicio REST para esto, y una vez recibida la respuesta del servicio REST, se la devuelve al usuario. Esta capa está desarrollada utilizando *Java* como lenguaje, *CSS* para los formatos, *JavaScript* para funciones de la librería *vis.js* y con el Framework *Thymeleaf* para el acceso a datos.

6.2 Patrones de diseño

En el siguiente apartado, vamos a explicar brevemente los patrones de diseño utilizados a lo largo del proyecto, también, en cada iteración explicaremos si lo hemos usado y en este caso, pondremos algún ejemplo de estos.

- **Patrón MVC:** El patrón Modelo-Vista-Controlador (MVC) es un patrón de arquitectura software que separa la lógica de aplicación (modelo) de su representación (vista), y para la interacción entre los dos se introduce un elemento llamado controlador, que es el responsable de transferir las solicitudes de la vista al modelo para que este haga lo que corresponda. Acto seguido el modelo devuelve el resultado al controlador y éste le envía el resultado a la vista para que represente los datos.

  Utilizando como base el patrón MVC, para esta proyecto se utilizaron 3 capas base. Una de datos, otra para el servidor de aplicaciones web y otra para los sistemas de visualización. Como sabemos, este patrón de diseño se caracteriza por separar los datos de la lógica de negocio de su representación y del módulo encargado de gestionar los eventos de comunicación.

  A continuación, describiremos estos componentes:

  - **Cliente Web:** A través de un navegador se podrá tanto registrar un nuevo usuario, como autenticarse con uno existente.
  - **Servidor:** Parte del sistema que se compone de varios módulos, los cuales nos permiten construir juntos un sistema que permitirá realizar operaciones sobre la base de datos. Los diferentes módulos son:
    - **Controlador REST:** Mediante el uso de una API REST, los clientes podrán consumir las diferentes funcionalidades que ofrece el sistema. Este también validará las peticiones realizadas. Se encargará también de invocar las operaciones del servicio para obtener los datos que se necesitan.
    - **Servicios:** Componentes donde está encapsulada la lógica de negocio de la aplicación. En el caso de esta iteración: el inicio de sesión y el registro de usuarios.
    - **Repositorios:** Componente de la aplicación donde se realizan las operaciones de acceso a datos dependiendo de las necesidades.
  - **Base de datos:** Aquí se almacenarán todos los datos de usuarios registrados.

- **PatrónDTO:** Un *Data Transfer Objet*, DTO. Este patrón nos permite crear un objeto plano, POJO, con una serie de atributos que puedan ser enviados y recuperados de un

servidor en una sola invocación. Esto nos permite poder devolver un objeto DTO al usuario, ocultando parte de la información de la base de datos que no queramos mostrar al usuario.

- **Patrón Repository:** Para abstraer las entidades, se utiliza el patrón repositorio. Conceptualmente, un Repositorio encapsula el conjunto de objetos persistentes en una base de datos y las operaciones realizadas sobre ellas, proporcionando una concepción más orientada a objetos de capa de persistencia. Además, apoya el objetivo de lograr una separación sobre una dependencia limpia y unidireccional entre el domino y la capa de acceso a datos. El uso del patrón Repositorio reduce también la cantidad de código que es necesario crear para tratar con todas las consultas que se lleven a cabo. Las clases o entidades del dominio del proyecto, gracias al uso del marco Spring Data DPA, harán un uso de este patrón extendiendo una interfaz, para proporcionar operaciones CRUD en un repositorio para cada tipo de entidad específica.

- **Patrón DAO:** Este patrón es usado en conjunto con el patrón DTO. Juntos nos ofrecen la posibilidad de separar por completo la lógica de negocio de la lógica de acceso a datos. De esta forma, el patrón DAO nos proporciona los métodos CRUD: insertar, actualizar, borrar y consultar la información. Por otra parte, la capa de negocio solo se preocupa por la lógica de negocio y utiliza el DAO para interactuar con la fuente de datos. Es decir, el DAO representa la capa de acceso a datos y el DTO es un objeto plano, el cual se encarga de transmitir la información entre el DAO y la lógica de negocio.

Capítulo 7

Iteración 3: Gestión de usuarios

7.1 Análisis

En esta iteración se desarrolló un sistema de gestión los usuarios. Debido a las funcionalidades ligeras de la aplicación y el enfoque de querer hacer una aplicación sencilla para el usuario, nuestra aplicación solo tendrá un tipo de usuario, que será usuario registrado.

Hemos llegado a esta conclusión debido a que solo usaremos la base de datos *MySQL* para el login de usuarios. Los apartados posteriores se guardarán en disco, pues buscamos que el usuario pueda acceder a todos los datos que le ofrecemos en cualquier momento, una vez haya analizado sus ficheros. Esto le permitiría poder acceder a todos los datos incluso sin tener conexión a internet.

No hemos requerido ningún usuario de rol administrador, puesto que en ningún momento el usuario tendrá acceso a datos que no sean sus ficheros y los resultados de su análisis.

La finalidad real de la creación de usuarios residirá únicamente en poder clasificar los ficheros en carpetas, dependiendo del usuario que los suba y/o genere los informes. Dicho esto, desde dentro de la aplicación, un usuario solo podrá acceder y descargar los datos que haya subido con su usuario.

La gestión de usuarios constará de un login y un signup de similares características a las observadas respectivamente en las **Figuras** 7.1 y 7.2:

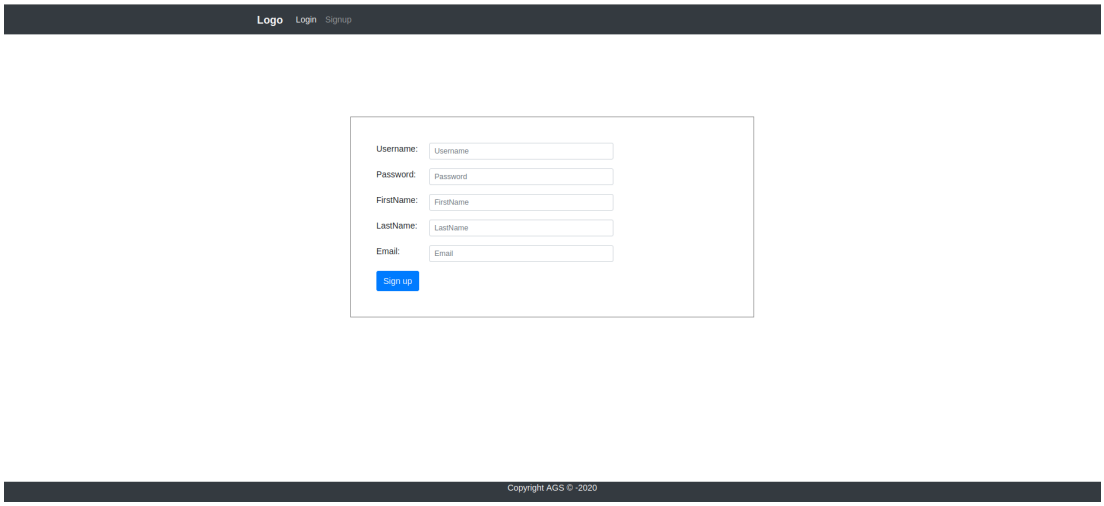


Figura 7.1: Mockup de registro

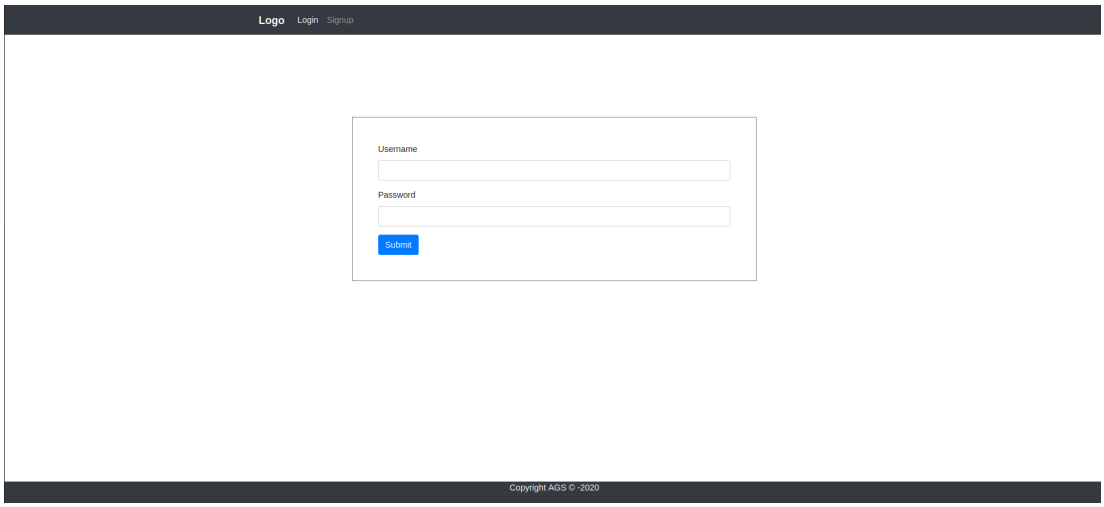


Figura 7.2: Mockup de acceso

7.1.1 Casos de uso

CU-01: Registrar usuario

- **Actores:** Usuario no registrado.
- **Descripción:** Un usuario podrá registrarse en la aplicación cubriendo un formulario con datos que se le ofrecerá en la vista.
- **Flujo básico:**
  1. El usuario selecciona la opción *Sign up*.

2. El sistema muestra un formulario con opciones para completar.
3. El usuario completa todos los campos.
4. El sistema comprueba si los datos introducidos son correctos y muestra un mensaje *Usuario creado*.

- **Flujo alternativo:** Existe algún error, se informa de ello al usuario. Este tendrá que volver a cumplimentar el formulario.
- **Precondiciones:** No puede haber dos usuarios con el mismo *userName* y el *email* tiene que seguir la nomenclatura *****@********.***
- **Postcondiciones:** El usuario es creado.

CU-02: Loguearse un usuario

- **Actores:** Usuario registrado.
- **Descripción:** Un usuario ya registrado podrá loguearse para el posterior uso de la aplicación.
- **Flujo básico:**
  1. El usuario selecciona la opción *Login*.
  2. El sistema muestra un formulario con opciones para completar.
  3. El usuario completa los campos.
  4. El sistema comprueba si los datos introducidos son correctos. De ser así, el usuario es redirigido a la página principal de la aplicación *Usuario creado*.
- **Flujo alternativo:** Existe algún error, se informa de ello al usuario. Este tendrá que volver a cumplimentar el formulario de *login*.
- **Precondiciones:** El usuario ha de estar registrado.
- **Postcondiciones:** El usuario es identificado y puede acceder al uso de la aplicación.

7.2 Diseño

En este apartado mostraremos una explicación mediante la tabla 7.1 de los atributos, de cada entidad, que tendremos en nuestro sistema. Los atributos subrayados se corresponden con la clave primaria de cada entidad.

Nombre del campo	Tipo de dato	Obligatorio	Único
id	Entero	Sí	Sí
email	Carácter variable	Sí	No
firstName	Carácter variable	Sí	No
lastName	Carácter variable	Sí	No
password	Carácter variable	Sí	No
role	Entero	No	No
userName	Carácter variable	Sí	Sí

Tabla 7.1: Modelo lógico de datos

De estos atributos cabe destacar dos cosas:

- Las direcciones de email están obligadas a seguir la nomenclatura ****@******.***
- Para las contraseñas hemos utilizado unas funciones de cifrado Hash para codificar y descodificar las contraseñas por motivos de seguridad. En la **Figura** 7.3 podemos ver alguna de las funciones y en la **Figura** 7.4 podemos observar que en la base de datos se guardan las contraseñas codificadas.

```java
/**
 * Implementation of PasswordEncoder that uses the BCrypt strong hashing function. C
 * can optionally supply a "strength" (a.k.a. log rounds in BCrypt) and a SecureRanc
 * instance. The larger the strength parameter the more work will have to be done
 * (exponentially) to hash the passwords. The default value is 10.
 *
 * @author Dave Syer
 *
 */
public class BCryptPasswordEncoder implements PasswordEncoder {
    private Pattern BCRYPT_PATTERN = Pattern
            .compile("\\A\\$2a?\\$\\d\\d\\$[./0-9A-Za-z]{53}");
    private final Log logger = LogFactory.getLog(getClass());

    private final int strength;

    private final SecureRandom random;

    public BCryptPasswordEncoder() {
        this(-1);
    }

    /**
     * @param strength the log rounds to use, between 4 and 31
     */
    public BCryptPasswordEncoder(int strength) {
        this(strength, null);
    }
```

Figura 7.3: Ejemplo funciones de cifrado hash

```
+--------------------------------------------------------------+
| password                                                     |
+--------------------------------------------------------------+
| $2a$10$NtfsoCXvlHdEczHjsRF1DuZwu/tcYYeNQkm8pcTpsxqyyyL/XUf.m |
| $2a$10$XZNoAC.7D.4fNL.xXzLlp.WB2/iVc7v82rNn2QCqhCZ68zspc9x5y |
| $2a$10$1t6ZIbFG4ouhSk6ZfZOMYe6qr262uLMSP.CKABD9wjE5gf38hT0Ve |
| $2a$10$ugGkYgsASUnrp4hwqTvfiOMdg/dTN6vQ41Bsr7WnTINnokThy2LpK |
| $2a$10$6e6PocFKTYQovP8wMDhXY.3Q2rwIGOYYJxxoVC//Cp8RS46ZZDE1a |
| $2a$10$4Wzx.sJTEkplTTYaC9PU9edwhkb8oz8jU3Ror7vZd4LSbq5am9wMq |
| $2a$10$ZFnkfmdEsvRAm/pNK3UYcebA9sXqlLQwFCFplE0RGmqpJ.9mcMxI. |
| $2a$10$O6kadFGGf0.o9T0nJUOmy.DtMNzuPccb9qF7OOKXLjMVJ1BEpaU0a |
| $2a$10$ZHdBPfxH3uaPRVP9MZEOIeQEJm2C4KCLOrC9xQWaqDEHdxg7Agfe. |
| $2a$10$j1SMnr/2Ml.UTSET6nt.aO8b83bp5pA7e/hoWYvKdY1kcgPxRLcmu |
| $2a$10$8uq1Q7iggXlFOlRnLefgMub2wR6yZIuSHB0x4iOKM2gWmV49K0Rgm |
| $2a$10$uWna6TBxdaDpCTi3HYhBIe3IOaGln/4SBn9O28mteqkXiXl55rkA. |
| $2a$10$BxBqMDm8emQ93Y.ehHkLt.rh9Lg6UNXcddL.ataw6LQqKBo6dtX7u |
| $2a$10$T5dc1wzLyvGw5rIEKpfzlecRTScd.NXOMl5LQzYQHE.38g4dpdJV2 |
+--------------------------------------------------------------+
```

Figura 7.4: Ejemplo contraseña cifrada

7.3 Implementación

En esta iteración se han usado diferentes patrones de diseño. A continuación hablaremos un poco de por que los hemos usado y pondremos algún ejemplo:

- **PatrónDTO:** Un *Data Transfer Objet*, DTO. Hemos utilizado este patrón para crear un objeto plano UserDto, con los atributos que necesitamos recuperar para nuestro login. Esto nos permite la ocultación de algunos casos que no queremos que el usuario vea. Podemos apreciar un ejemplo del uso del patrón DTO en la **Figura** 7.5.

```java
package es.udc.tfg.rest.dtos;

import javax.validation.constraints.Email;

public class UserDto {

    public interface AllValidations {}

    public interface UpdateValidations {}

    private Long id;
    private String userName;
    private String password;
    private String firstName;
    private String lastName;
    private String email;
    private String role;

    public UserDto() {}

    public UserDto(Long id, String userName, String firstName, String lastName, String email, String role) {

        this.id = id;
        this.userName = userName != null ? userName.trim() : null;
        this.firstName = firstName.trim();
        this.lastName = lastName.trim();
        this.email = email.trim();
        this.role = role;

    }

    public Long getId() {
        return id;
    }
```

Figura 7.5: Ejemplo clase DTO

- **Patrón MVC:** Hemos hecho uso del patrón MVC, en nuestro caso en particular el modelo es nuestra clase User, en la que definimos el objeto usuario, el controlador que hace referencia a nuestra clase UserController, la cual se encarga de pedirle los datos a la clase usuario para poder realizar las acciones que se han requerido en la vista, y por ultimo la vista que hace referencia a login.html en la que el usuario intenta loguearse. Podemos ver un ejemplo en nuestro código del modelo en la **Figura** 7.6, del controlador en la **Figura** 7.7 y de la vista en la **Figura** 7.8.

```java
package es.udc.tfg.model.entities;


import javax.persistence.Entity;

@Entity
@BatchSize(size=10)
public class User {

    public enum RoleType {USER};

    private Long id;
    private String userName;
    private String password;
    private String firstName;
    private String lastName;
    private String email;
    private RoleType role;

    public User() {}

    public User(String userName, String password, String firstName, String lastName, String email) {

        this.userName = userName;
        this.password = password;
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
    }

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    public Long getId() {
        return id;
    }
```

Figura 7.6: Ejemplo modelo

```
    @PostMapping("/signUp")
    public ResponseEntity<AuthenticatedUserDto> signUp(
        @Validated({UserDto.AllValidations.class}) @RequestBody UserDto userDto) throws DuplicateInstanceException {

        User user = toUser(userDto);

        userService.signUp(user);

        URI location = ServletUriComponentsBuilder
            .fromCurrentRequest().path("/{id}")
            .buildAndExpand(user.getId()).toUri();

        return ResponseEntity.created(location).body(toAuthenticatedUserDto(generateServiceToken(user), user));

    }

    @PostMapping("/login")
    public AuthenticatedUserDto login(@Validated @RequestBody LoginParamsDto params)
        throws IncorrectLoginException {

        User user = userService.login(params.getUserName(), params.getPassword());

        return toAuthenticatedUserDto(generateServiceToken(user), user);

    }
```

Figura 7.7: Ejemplo controlador

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head th:replace="plantilla/template :: head">
</head>
<body>
    <header th:replace="plantilla/template :: header"></header>

    <div class="container" style="margin-top: 150px;">
        <div class="row align-items-center">
            <div class="col-md-2 my-1"></div>

            <div class="col-md-8 my-1 px-5 py-5" style="border: 0.5px solid grey">
                <form th:action="@{/login}" th:object="${user}" method="post">
                    <div class="form-group">
                        <label for="exampleInputEmail1">Username</label> <input
                            type="text" th:field=*{userName} class="form-control" id="exampleInputUserNaMe1">
                    </div>
                    <div class="form-group">
                        <label for="exampleInputPassword1">Password</label> <input
                            type="password" th:field="*{password}" class="form-control" id="exampleInputPassword1">
                    </div>
                    <button type="submit" class="btn btn-primary">Submit</button>
                </form>
            </div>
            <div class="col-md-2 my-1"></div>
        </div>
    </div>

    <footer th:replace="plantilla/template :: footer"></footer>
</body>
</html>
```

Figura 7.8: Ejemplo vista

- **Patrón DAO:** Hemos utilizado este patrón como medio para ahorrar codigo a la hora de usar algunas de las funciones CRUD. Casi siempre lo hemos usado complementado por el patrón DTO. Podemos ver un ejemplo del patrón en nuestro código en las **Figuras** 7.9 y 7.10.

```java
package es.udc.tfg.model.entities;

import java.util.Optional;

public interface UserDao extends PagingAndSortingRepository<User, Long> {

    boolean existsByUserName(String userName);
    Optional<User> findByUserName(String userName);
```

Figura 7.9: Ejemplo DAO 1

```java
@Autowired
private UserDao userDao;

@Override
public void signUp(User user) throws DuplicateInstanceException {

    if (userDao.existsByUserName(user.getUserName())) {
        throw new DuplicateInstanceException("project.entities.user", user.getUserName());
    }

    user.setPassword(passwordEncoder.encode(user.getPassword()));
    user.setRole(User.RoleType.USER);

    userDao.save(user);

}
```

Figura 7.10: Ejemplo DAO 2

7.4 Pruebas

En esta primera iteración se realizaron pruebas utilizando la biblioteca JUnit. Primeramente, se realizaron pruebas de unidad, pues comprobamos el correcto funcionamiento del código entre sí: DTO y DAOS que acceden a la base de datos.

Una vez que comprobamos que todo funcionaba correctamente, realizamos pruebas de integración, en las que comprobamos que los métodos CRUD (insertar, modificar, borrar) funcionaban. Podemos observar un ejemplo de función test de integridad en la **Figura** 8.1.

```java
@Test
public void testSignUpAndLoginFromId() throws InstanceNotFoundException, DuplicateInstanceException {
    User user = createUser("user5");

    userService.signUp(user);

    User loggedInUser = userService.loginFromId(user.getId());

    assertEquals(user, loggedInUser);
    assertEquals(User.RoleType.USER, user.getRole());

    System.out.println("Antes");
    userService.deleteUser(user);
    System.out.println("Despues");
}
```

Figura 7.11: Ejemplo prueba integridad

Capítulo 8

Iteración 4: Gestión de Archivos

8.1 Análisis

A partir de esta iteración el acceso a la base de datos es innecesario mientras el cliente esté identificado en la aplicación. Esto se debe a que el resto de casos de uso en los que haya que realizar un guardado, este se realizará en disco. Por ello, en este capítulo nos saltaremos alguna sección que sí aparecía en el capítulo anterior o que estaría repetida.

Debido a que no se usará la base de datos en la iteración, el patrón MVC se rompe completamente y, por lo tanto, también los patrones DAO, DTO y repositorio. Por ello se ha decidido que, al ser peticiones que no suponen demasiados cálculos o uso de memoria, lo más sencillo es crear un controlador para las funciones de esta iteración y posteriores, aprovechando así la arquitectura que hemos usado hasta ahora a lo largo del proyecto.

Un controlador se encarga, mediante el uso del framework de spring, de recibir las peticiones HTTP provenientes de la vista, decide qué método Java de la capa de servicios es necesario utilizar o responde con la salida de un método propio. A continuación se detallarán los controladores utilizados en nuestro proyecto:

- **UserController:** El controlador de usuarios agrupa todas las funcionalidades que tienen relación con las actividades de los usuarios. La funcionalidad del controlador engloba las siguientes acciones:
  - Autenticación de usuarios.
  - Registro de un nuevo usuario.
  - Salida de la aplicación.

- **ThymeleafController:** Este controlador se encarga de las peticiones que tienen que ver con un archivo de usuario. La funcionalidad del controlador engloba las siguientes acciones:
  - Subir, visualizar y descargar archivos.
  - Procesar y analizar archivos.
  - Visualizar gráficas.
  - Subir, visualizar y descargar informes.
  - Generar topología.

En esta iteración en concreto se ha llevado a cabo la subida, descarga, guardado y visualización de archivos de formato exclusivamente .pcap.

8.1.1 Casos de uso

CU-03: Subir archivo

- **Actores:** Usuario logueado.
- **Descripción:** Un usuario podrá seleccionar un archivo formato .pcap y subirlo para posterior análisis.
- **Flujo básico:**
  1. El usuario selecciona la opción *Upload*.
  2. El sistema muestra los botones buscar y aceptar.
  3. El usuario busca el archivo que desea subir.
  4. El sistema comprueba si es un archivo formato .pcap y si es inferior a 50 MB. Si es así, sube el archivo.
- **Flujo alternativo:** Si el archivo es de otro formato, o su tamaño es superior a 50 MB, se informa de ello al usuario. Este tendrá que volver al apartado upload e intentar subir el archivo de nuevo.
- **Precondiciones:** El usuario ha de estar logueado.
- **Postcondiciones:** El archivo es subido.

CU-04: Visualizar archivo

- **Actores:** Usuario logueado.
- **Descripción:** Un usuario podrá seleccionar un archivo que haya subido y visualizarlo.
- **Flujo básico:**
  1. El usuario selecciona la opción *Files*.
  2. El sistema muestra la lista de archivos que se encuentran en la carpeta del usuario logueado.
  3. El usuario busca el archivo que desea visualizar.
  4. El sistema mostrará una ventana con la opción visualizar, al hacer clic se abrirá el archivo en caso tener un programa que pueda reproducirlo.
- **Flujo alternativo:** No existen archivos subidos.
- **Precondiciones:** El usuario ha de tener algún archivo subido.
- **Postcondiciones:** El usuario visualiza el archivo.

CU-05: descargar archivo

- **Actores:** Usuario logueado.
- **Descripción:** Un usuario podrá seleccionar un archivo que haya subido y descargarlo.
- **Flujo básico:**
  1. El usuario selecciona la opción *Files*.
  2. El sistema muestra la lista de archivos que se encuentran en la carpeta del usuario logueado.
  3. El usuario busca el archivo que desea descargar.
  4. El sistema mostrará una ventana con la opción descargar, al hacer clic se descargará el archivo.
- **Flujo alternativo:** No existen archivos subidos.
- **Precondiciones:** El usuario ha de tener algún archivo subido.
- **Postcondiciones:** El usuario descarga el archivo.

8.2 Diseño

Para nuestra aplicación queremos que el usuario ya registrado pueda subir un archivo pcap y que se guarde en disco. Esto queremos hacerlo de forma ordenada, por ello creamos un directorio que completaremos en siguientes iteraciones.. Una idea aproximada de lo que tenemos en mente se puede observar en el siguiente mockup:

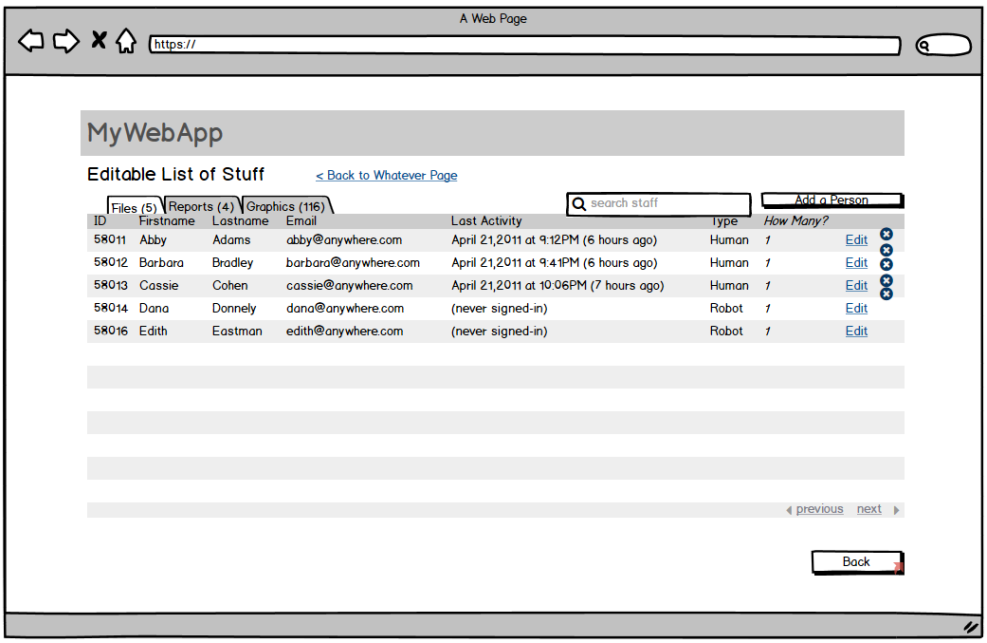


Figura 8.1: Mockup de gestión de ficheros

8.3 Implementación

La implementación de esta iteración ha requerido de gran cantidad de tiempo, pues era difícil conseguir las 3 funciones que queríamos: subir los archivos, ver una lista de archivos y la posibilidad de ver o descargar el archivo. Tal ha sido el tiempo que ha llevado que casi la mitad del retraso de la fase de implementación que calculamos en el capitulo de *Análisis de costes* se ha debido a esta iteración.

Al final, hemos conseguido las 3 funciones mezclando diversas soluciones de otros usuarios. Para cerrar solo hemos hecho uso de funciones de java y thymeleaf. Podemos observar un ejemplo del resultado final en la siguiente figura.

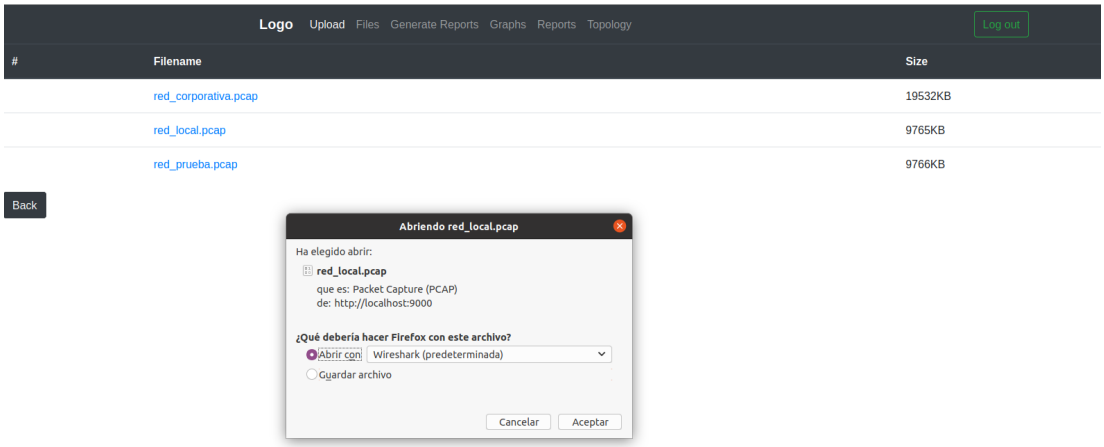


Figura 8.2: Lista de archivos y descarga

8.4 Pruebas

En esta iteración no se realizaron pruebas de integración, pues no hay acceso a la base de datos. Sí se realizaron, sin embargo, pruebas de unidad para comprobar que el código funcionaba correctamente y pruebas de usabilidad en la propia aplicación, para comprobar que la aplicación desempeñaba las funcionalidades requeridas.

Capítulo 9

Iteración 5: Generación de gráficas e informes

9.1 Análisis

En esta iteración, procederemos al análisis de un archivo subido por el usuario después de varios parámetros recibidos en la vista. Procederemos a la generación de gráficos, informes y varios archivos intermedios. También se permitirá la visualización de gráficos en la aplicación, así como la visualización y descarga de los informes.

Esta iteración se ha llevado a cabo siguiendo el mismo modelo que la iteración anterior.

Esta iteración va a tener una larga duración, pues se hará uso de diversas tecnologías externas que han de funcionar en conjunto, lo cual aumenta la dificultad.

9.1.1 Casos de uso

CU-06: Procesar archivo

- **Actores:** Usuario logueado.
- **Descripción:** Un usuario podrá seleccionar un archivo que haya subido para procesarlo.
- **Flujo básico:**
    1. El usuario selecciona la opción *Generate*.
    2. El sistema muestra un formulario con un listado de archivos y varios parámetros opcionales.

3. El usuario busca el archivo que desea procesar y opcionalmente rellenará los parámetros que necesite.
4. El sistema, tras unos segundos de análisis, muestra una ventana conforme al éxito de la operación con varias opciones.

- **Flujo alternativo:** Existe algún error a lo largo del proceso.
- **Precondiciones:** El usuario ha de tener algún archivo subido.
- **Postcondiciones:** El usuario dispondrá de los archivos en el directorio y de diversas opciones en la aplicación.

CU-07: visualizar gráficos

- **Actores:** Usuario logueado.
- **Descripción:** Un usuario podrá seleccionar la opción *Graphs* para visualizar los gráficos generados del ultimo análisis.
- **Flujo básico:**
  1. El usuario selecciona la opción *Graphs*.
  2. El sistema muestra los gráficos del último análisis realizado.
- **Flujo alternativo:** No hay gráficos.
- **Precondiciones:** El usuario ha de tener algún archivo analizado con anterioridad.
- **Postcondiciones:** El usuario visualiza los gráficos.

CU-08: visualizar informe

- **Actores:** Usuario logueado.
- **Descripción:** Un usuario podrá seleccionar un informe que haya sido creado tras un análisis y visualizar.
- **Flujo básico:**
  1. El usuario selecciona la opción *Reports*.

2. El sistema muestra la lista de informes que se encuentran en la carpeta del usuario logueado.
3. El usuario busca el informe que desea visualizar.
4. El sistema mostrará una ventana con la opción visualizar, al hacer clic sobre ella se visualizará el archivo.

- **Flujo alternativo:** No existen informes generados.
- **Precondiciones:** El usuario ha de tener algún archivo analizado con anterioridad.
- **Postcondiciones:** El usuario visualiza el informe.

CU-09: descargar informe

- **Actores:** Usuario logueado.
- **Descripción:** Un usuario podrá seleccionar un informe que haya sido creado tras un análisis y descargarlo.
- **Flujo básico:**
  1. El usuario selecciona la opción *Reports*.
  2. El sistema muestra la lista de informes que se encuentran en la carpeta del usuario logueado.
  3. El usuario busca el informe que desea descargar.
  4. El sistema mostrará una ventana con la opción descargar, al hacer clic sobre ella se descargará el informe.
- **Flujo alternativo:** No existen informes generados.
- **Precondiciones:** El usuario ha de tener algún archivo analizado con anterioridad.
- **Postcondiciones:** El usuario descarga el informe.

9.2 Diseño

A continuación podremos ver unos mockups que mostrarán la idea que teníamos en mente inicialmente:

- En la **Figura** 9.1 podemos observar que la idea principal era tener una ventana exclusivamente para la elección del archivo y los parámetros a utilizar
- En la **Figura** 9.2 vemos la ventana siguiente a analizar el archivo en donde nos da las opciones de volver a atrás a subir otro archivo, ver los gráficos o ver los informes.
- En la **Figura** 9.3 es la ventana de visualización de gráficos.
- En la **Figura** 9.4 es la ventana de visualización de informes.

Cabe destacar que al final del proyecto las ventanas han sufrido modificaciones.

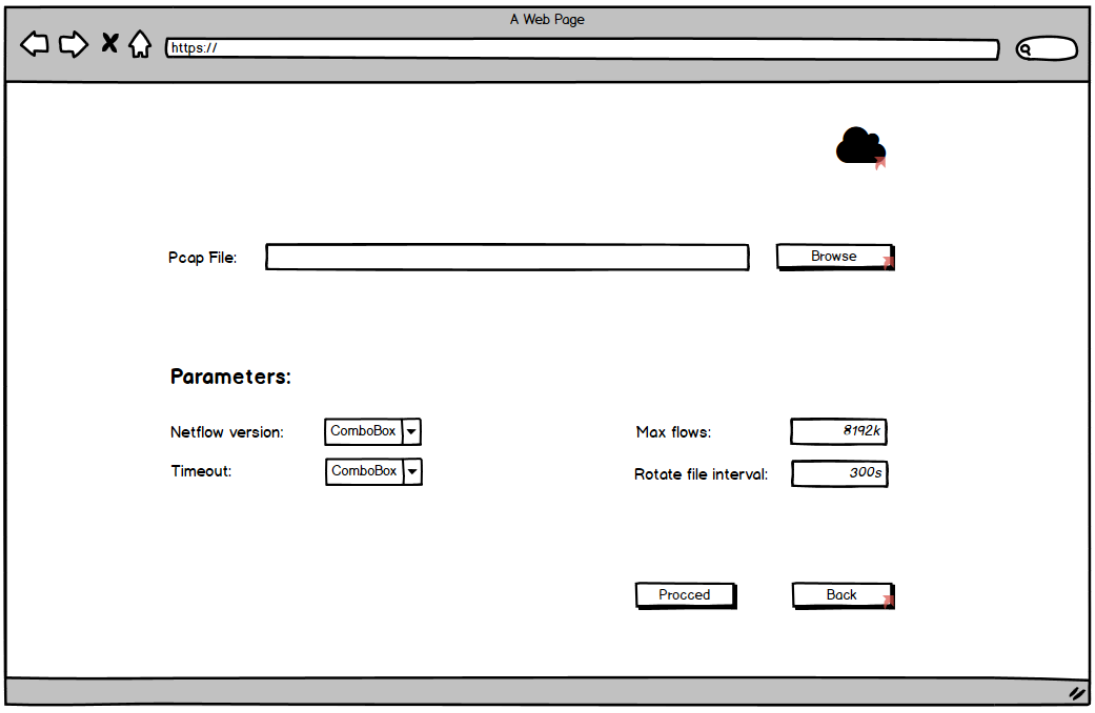


Figura 9.1: Mockup de subida de archivo y parámetros

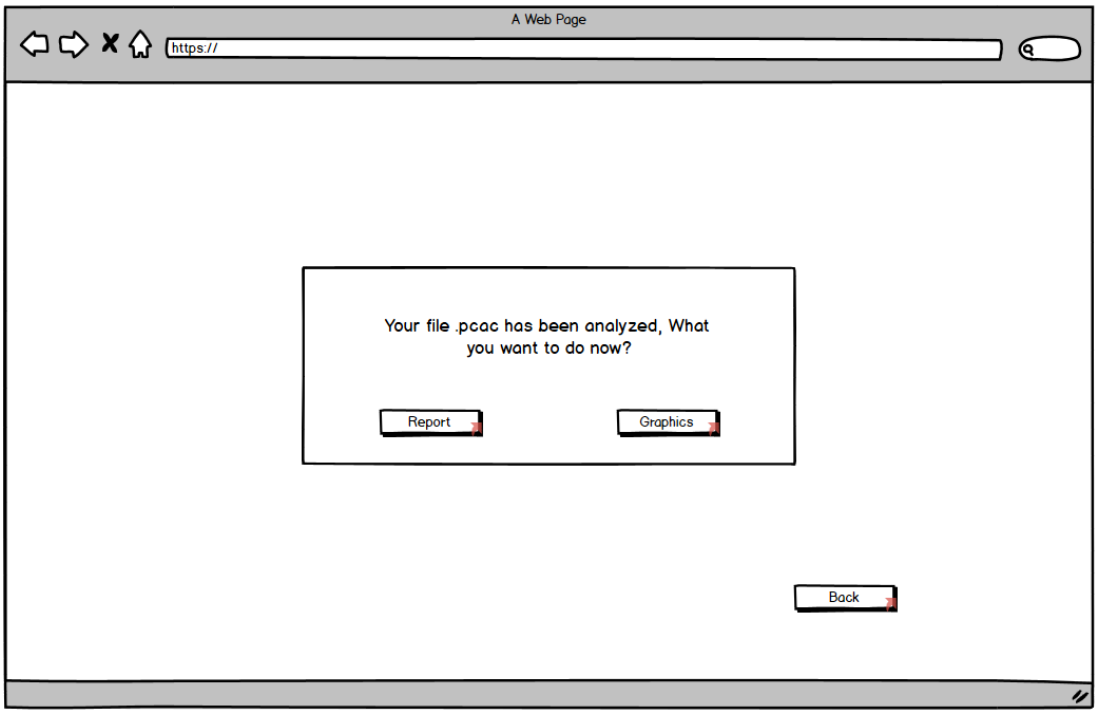


Figura 9.2: Mockup de elecciones

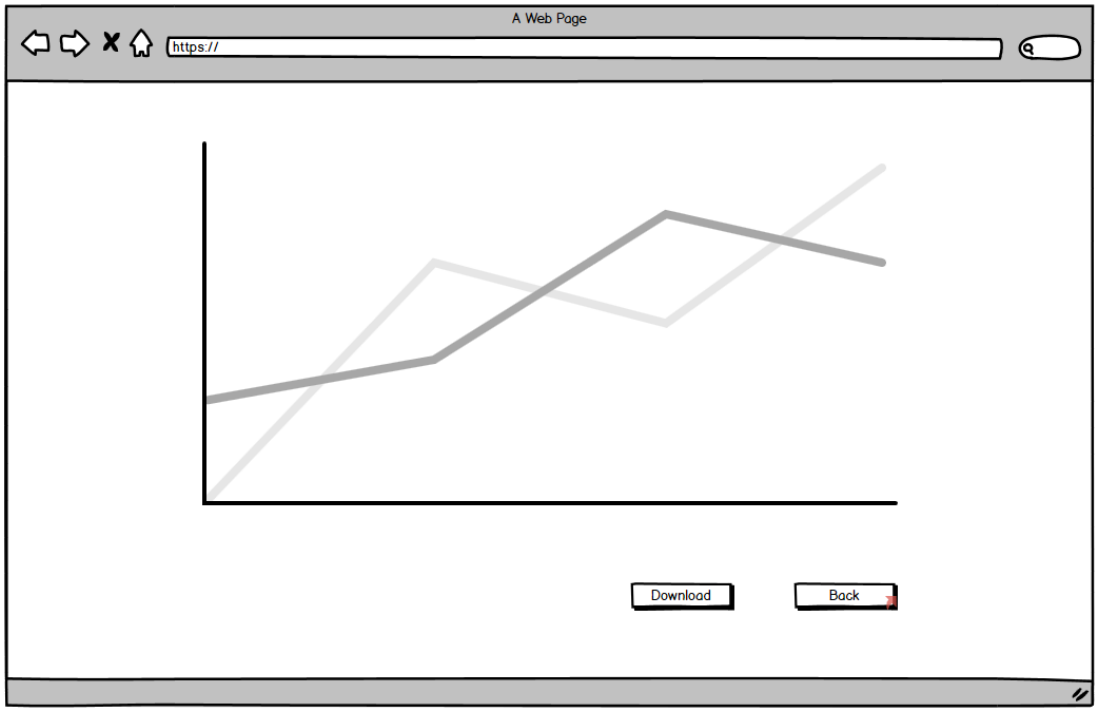


Figura 9.3: Mockup de gráficos de muestra

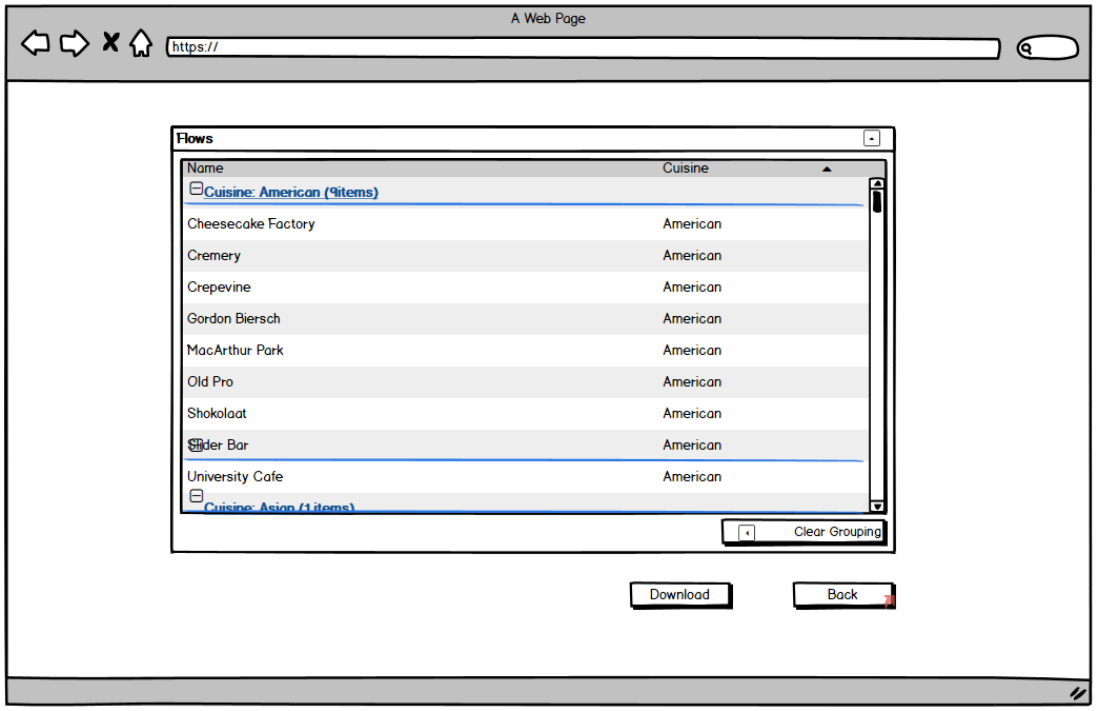


Figura 9.4: Mockup visualización y descarga de informes

9.3 Implementación y tecnologías externas

Para esta iteración hemos usado tecnologías externas para realizar los análisis de flujos del archivo recibido por el usuario. Para ello hemos lanzado procesos que ejecutan comandos de terminal en los que se ejecutarán las diversas tecnologías externas que detallaremos a continuación:

- **Softflowd**(exportador): Lo usaremos para exportar el tráfico del pcap del usuario por un puerto específico.

- **nfcapd**(colector): Este software escuchará el puerto usado por softflowd, y almacenará la información netflow como flujos en diferentes archivos.

- **nfdump**(Analizador): Este software será el encargado de procesar y analizar los datos que se encuentran en los archivos generados por nfcapd, que exportaremos a un csv simple para poder trabajar posteriormente con la información.

- **JFreeChart**(Librería gráfica): Usaremos esta librería gráfica para crear diferentes charts. Para ello necesitaremos pasarle la información del csv por columnas.

- **itext**(librería): Utilizaremos esta librería para generar un pdf (informe), en el cual introduciremos datos después de analizarlos y decidir cuáles son relevantes para el usuario.

En la **Figura** 9.5 podemos observar un diagrama de secuencia en el que se ve de forma un poco mas clara el uso orden de uso de las tecnologías externas.

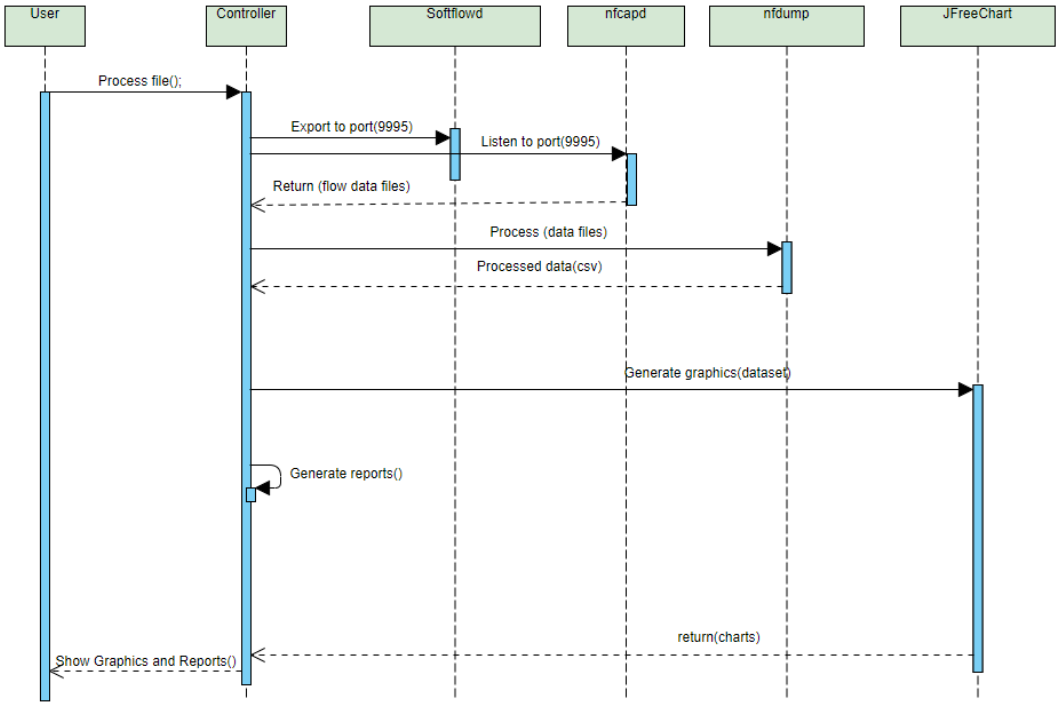


Figura 9.5: Diagrama de secuencia del uso de las tecnologías

Vamos a retomar la idea que nombramos en la iteración anterior para explicar, ahora completo, el directorio creado para guardar los datos y los archivos que cada directorio contiene:

- **Nombre de usuario:** Carpeta raíz.
  - **Flows:** En esta carpeta se creará una carpeta por cada archivo que se haya procesado. En el interior de cada una de estas carpetas encontraremos un subdirectorio recursivo de carpetas que seguirán la siguiente nomenclatura: año->mes->día->hora. Debido a esta nomenclatura, no podremos analizar 2 archivos con el mismo nombre en la misma hora de un día.

    En el interior del subdirectorio anterior encontraremos:
    * Gráficos en formato png generados mediante la librería JFreeChart que se mostrarán en la aplicación.

    - Un archivo binario generado por nfcapd, donde se almacena la información de los flujos.
    - El archivo csv con la información procesada por nfdump.
    - Un fichero JSON que utilizaremos en la siguiente iteración, con información extraída del csv.
  - **Pcaps:** En esta carpeta guardaremos los archivos que suba el usuario.
  - **Reports:** En esta carpeta se guardarán los informes que se generarán con la librería itext. Se creará un informe por cada archivo analizado y su nombre será el del archivo analizado con formato .pdf.

El usuario desconoce todas las tecnologías utilizadas para la generación de sus datos. Para generar los datos solo necesita ir a la pestaña *generar*, elegir el archivo a analizar y rellenar una serie de parámetros que se utilizarán posteriormente en los procesos que lanzamos en el análisis. De no rellenar los campos se utilizarán los valores predeterminados.

Los campos son los siguientes:

- **Netflow version:** El usuario podrá elegir en qué versión del protocolo netflow se exportarán los datos de flujo. Por defecto, versión 5.
- **General Timeout:** Tiempo de inactividad a partir del cual softflowd expirará un flujo. Por defecto 0.
- **max flows:** Permite indicar el número de flujos máximo que se permite procesar concurrentemente. En caso de exceder el número propuesto por el usuario, el flujo más longevo será forzado a expirar. Por defecto 8192 flujos.
- **interval rotate:** El usuario podrá elegir el intervalo de tiempo en el que se generará un archivo nuevo para guardar la información. Por defecto 300 segundos (5 minutos).

El proceso de análisis lleva a cabo los siguientes pasos:

- Se crean los subdirectorios de la fecha actual en la que se guardarán los datos del procesado
- Se lanza un proceso mediante el cual softflowd exporta todos los datos del archivo .pcap a un puerto específico, añadiendo a este proceso, que se lanza en el terminal, los posibles parámetros añadidos en la vista por el usuario.
- Se lanza un segundo proceso mediante el cual nfcapd escuchará en ese puerto toda la información y será guardada en archivos según los parámetros indicados.

- Una vez el sistema comprueba que los procesos anteriores terminaron, se lanza un tercer proceso en el cual nfdump se encarga de procesar todos los datos de los archivos generados por nfcapd y los exporta (en nuestro caso) a un fichero csv simple.
- Se lee el fichero csv del cual se extraen datos, los cuales nosotros hemos considerado relevantes, estos datos serán utilizados o para la generación de gráficos o tras diversas modificaciones, para la generación de informes.
- Se crean diversas gráficas y diversos dataset a los que se les pasará como parámetros a la librería JFreeChart, para que pueda generar diversos gráficos, que se guardarán en disco y se mostrarán en la aplicación. Podemos observar un ejemplo en la **Figura** 9.6.

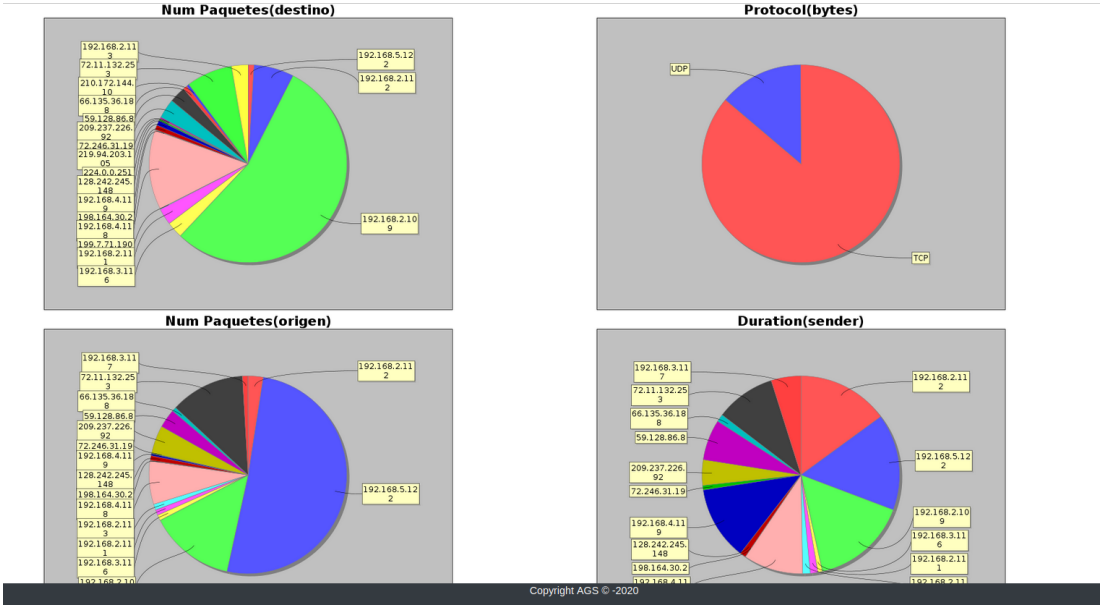


Figura 9.6: Ejemplo de algunas gráficas mostradas en la aplicación

- Posteriormente, se realizarán cálculos con los datos para poder obtener datos relevantes para la generación de informes, que serán subidos a disco posteriormente. Podemos observar un ejemplo en la **Figura** 9.7.

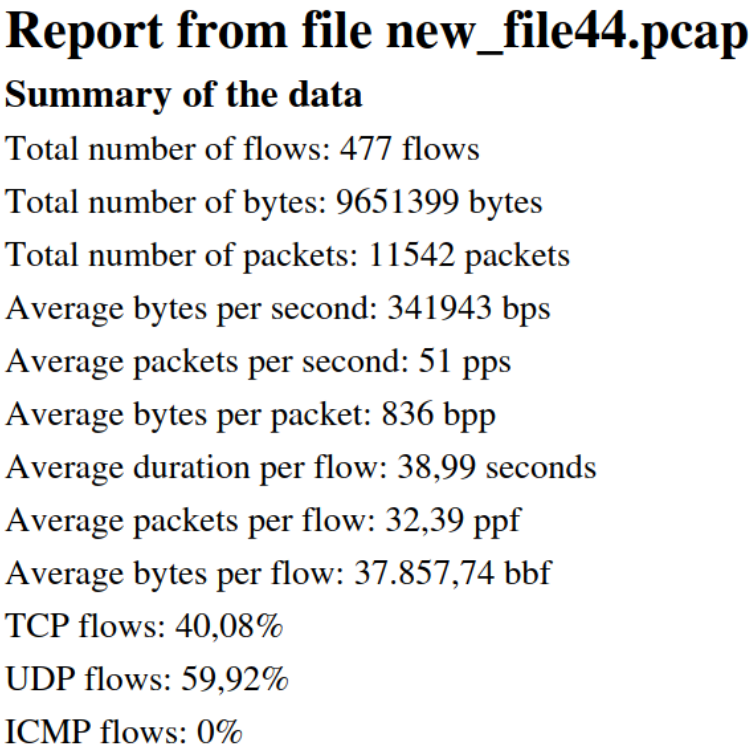


Figura 9.7: Ejemplo del informe de las gráficas anteriores

- Se informa al usuario de la finalización del procedimiento y que ya puede acceder a los datos tanto en disco como en aplicación o, en su defecto, analizar otro archivo.

9.4 Pruebas

En esta iteración no se realizaron pruebas de integración, pues no hay acceso a la base de datos. Sí se realizaron, sin embargo, pruebas de unidad para comprobar que el código funcionaba correctamente y pruebas de usabilidad en la propia aplicación para comprobar que la aplicación desempeñaba las funcionalidades requeridas.

Capítulo 10

Iteración 6: Generación de la topología

10.1 Análisis

Para esta iteración hemos hecho uso de otra tecnología externa, en este caso una librería, para así poder utilizar los datos que procesamos en el apartado anterior para generar un gráfico en el que se pueda observar la topología de la red y las comunicaciones entre los diferentes nodos.

10.1.1 Casos de uso

CU-10: generar topología

- **Actores:** Usuario logueado.
- **Descripción:** Un usuario podrá visualizar la topología de red del último análisis realizado.
- **Flujo básico:**
  1. El usuario selecciona la opción *Topology*.
  2. El sistema después de unos segundos en los que procesa los datos genera la topología de la red.
- **Flujo alternativo:** No existe fichero json.
- **Precondiciones:** El usuario ha de tener algún archivo analizado con anterioridad.
- **Postcondiciones:** El usuario visualiza la topología de la red.

10.2 Diseño

Debido a que no teníamos conocimiento de como se realizaría o vería este apartado, no tenemos un mockup que se parezca al resultado final. De cualquier forma el resultado final ha sido satisfactorio.

10.3 Implementación

Para esta implementación hemos procedido a la extracción de los datos necesarios del fichero csv y posteriormente generamos un fichero JSON en un formato específico para dicha librería. Debido a que el formato que utiliza la librería es un poco extraño fue necesario generar el JSON manualmente mediante bucles.

Una vez se genera el fichero, la librería se encarga de leer todos los nodos y todas las aristas y genera la topología necesaria. Todo este proceso se realiza en la vista, pues la librería utiliza el lenguaje *javascript*. Podemos ver un ejemplo en la **Figura** 10.2

```
31
32 <!-- Add an invisible <div> element to the document, to hold the JSON data: -->
33 <div id="networkJSON-results" class="results" style="display:none"></div>
34
35 <script type="text/javascript">
36
37   // --------------------------------------------------------------------
38   // OPTIONS:
39
40   // http://visjs.org/docs/network/#modules
41   // http://visjs.org/docs/network/edges.html#
42   // http://visjs.org/docs/network/physics.html#
43
44   var options = {
45     edges: {
46       arrows: {
47         to: {enabled: true, scaleFactor:0.75, type:'arrow'},
48         // to: {enabled: false, scaleFactor:0.5, type:'bar'},
49         middle: {enabled: false, scalefactor:1, type:'arrow'},
50         from: {enabled: true, scaleFactor:0.3, type:'arrow'}
51         // from: {enabled: false, scaleFactor:0.5, type:'arrow'}
52       },
53       arrowStrikethrough: true,
54       chosen: true,
55       color: {
56       // color:'#848484',
57       color:'red',
58       highlight:'#848484',
59       hover: '#848484',
60       inherit: 'from',
61       opacity:1.0
62       },
63       dashes: false,
64       font: {
65         color: '#343434',
66         size: 14, // px
67         face: 'arial',
68         background: 'none',
69         strokeWidth: 2, // px
```

Figura 10.1: Librería visjs

En el fichero html hemos introducido el código javascript en el que importamos los datos del json externo y configuramos las opciones de creación de nodos y aristas.

Para la red topológica de este ejemplo, hemos utilizado la resultante de los datos de los ejemplos anteriores. En ella podemos observar que existen algunos nodos que están sueltos y esparcidos por la imagen. Esto se debe a que el dataset del que sacamos las muestras era extremadamente grande, de varios Gygas. Por ello, al trocearlo en datasets más pequeños, en este ejemplo han quedado algunos nodos sueltos. De utilizar un dataset completo no existirían dichos nodos.

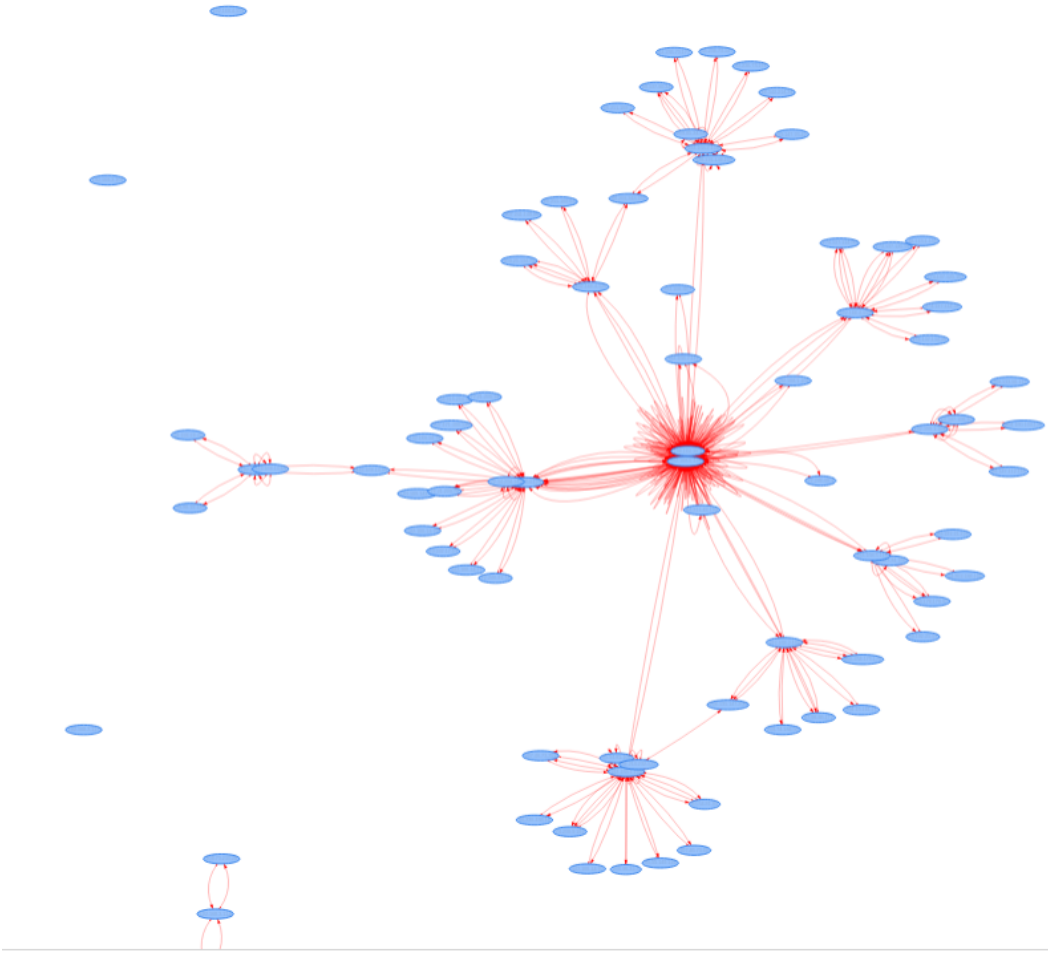


Figura 10.2: Topología

Una gran ventaja de la librería vis.js es que permite acercar o alejar el zoom e incluso se pueden mover los nodos a placer. En la **Figura** 10.3 podemos observar una de las subredes de la topología anterior más de cerca junto a los nodos núcleo de esa red.

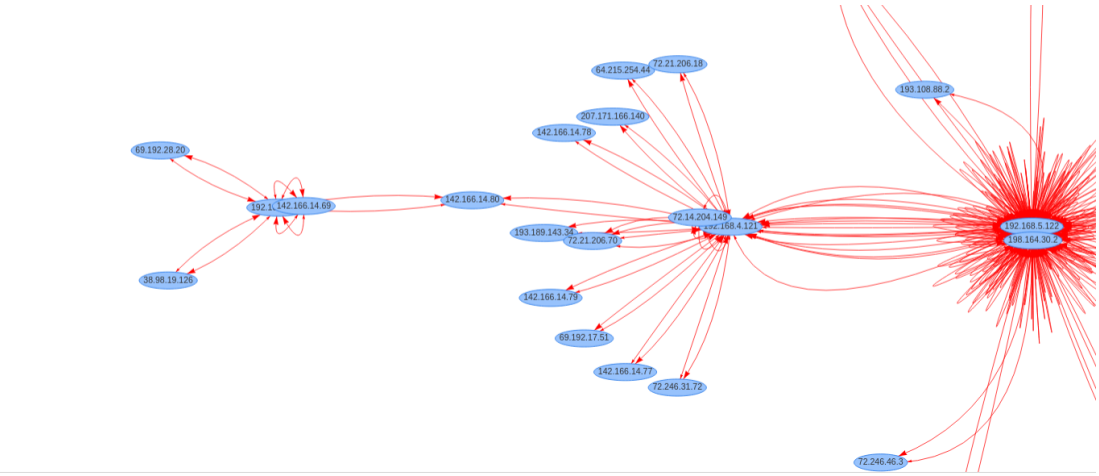


Figura 10.3: Topología acercada

10.4 Pruebas

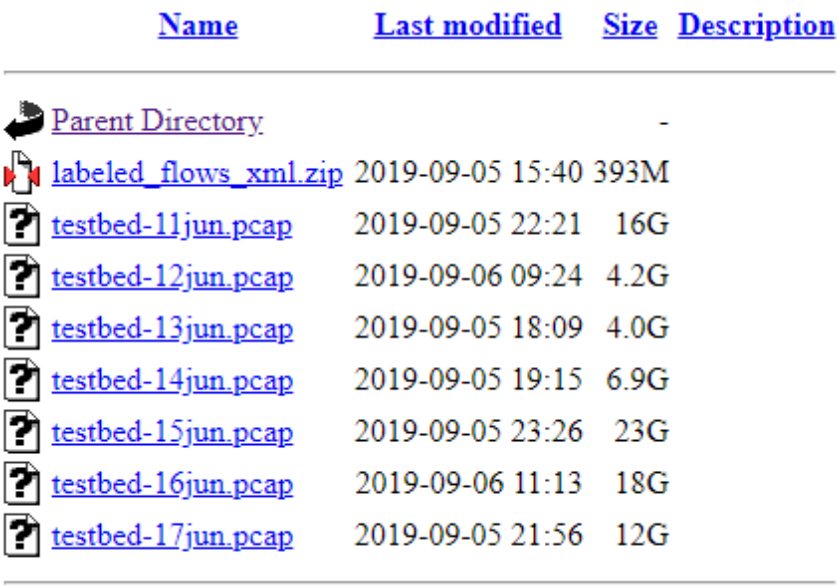
En esta iteración no se realizaron pruebas de integración, pues no hay acceso a la base de datos. Sí se realizaron, sin embargo, pruebas de unidad para comprobar que el código funcionaba correctamente y pruebas de usabilidad en la propia aplicación para comprobar que la aplicación desempeñaba las funcionalidades requeridas.

Capítulo 11

Datos utilizados

Para comprobar el correcto procesamiento de todas las herramientas utilizadas a lo largo del proyecto, hemos decidido trabajar en un entorno controlado y estudiado anteriormente por *University of New Brunswick*, la cual nos proporciona un dataset que han utilizado anteriormente para evaluar la detección de intrusiones en una red local. Este dataset recibe el nombre de *ISCXIDS2012* [28].

Este dataset cuenta con un conjunto de archivos de gran tamaño, pues son los datos del análisis de la red de días enteros. Mostramos los archivos que se pueden utilizar en la siguiente imagen.



Name	Last modified	Size	Description
Parent Directory		-	
labeled_flows_xml.zip	2019-09-05 15:40	393M	
testbed-11jun.pcap	2019-09-05 22:21	16G	
testbed-12jun.pcap	2019-09-06 09:24	4.2G	
testbed-13jun.pcap	2019-09-05 18:09	4.0G	
testbed-14jun.pcap	2019-09-05 19:15	6.9G	
testbed-15jun.pcap	2019-09-05 23:26	23G	
testbed-16jun.pcap	2019-09-06 11:13	18G	
testbed-17jun.pcap	2019-09-05 21:56	12G	

Figura 11.1: Archivos del dataset ISCX-IDS-2012

Puesto que los archivos eran demasiado grandes para realizar las pruebas, nosotros hemos utilizado la herramienta *tcpdump* para dividirlos en varios pcaps del mismo tamaño.

Al principio permitíamos pcaps más grandes pero, una vez se introdujo el caso de uso de la topología, nos vimos obligados a no permitir archivos más grandes de 50 MB, pues el tiempo de espera para generar dicha topología se excedía de un tiempo espera razonable.

Capítulo 12

Conclusiones y trabajo futuro

12.1 Conclusiones

El producto final cumple con los objetivos especificados en el anteproyecto:

- Se dispone de un sistema de control de usuarios para poder distribuir a cada usuario sus archivos
- La aplicación web permite subida, descarga y visualización de ficheros.
- La aplicación utiliza tecnologías externas para conseguir los datos que necesitamos para poder ofrecer al usuario la salida que necesita en diversos ficheros.
- La aplicación web dispone de generación, descarga y visualización de ficheros pdf.
- La aplicación web permite visualizar estadísticas generadas a partir de la información recogida de sus archivos
- La aplicación web permite una función que genera un gráfico de red equivalente a la topología de la red del usuario

Durante el desarrollo de la aplicación surgieron algunas dificultades a la hora de conseguir que todas la tecnologías externas funcionaran en el proceso, como una sola. Un ejemplo de esto fue la comunicación entre exportador, colector y analizador de flujos de datos. A pesar de esto, el desarrollo de proyecto fue abordado y llevado a cabo correctamente.

La interfaz web proporciona la información acordada al usuario. Para su diseño se utilizaron modelos *HTML*, con atributos *Thymeleaf*, dándole estilos con *CSS* y, en algunos casos, dándole funcionalidades con *Javascript*.

A su vez, se diseñó una *API REST* seguido de las guías de calidad del estándar. Esta *API* es consumida por la vista para recuperar los datos persistentes. Se realizó utilizando las tecnologías *Java* y *SpringBoot*, que permiten minimizar, notablemente, el tiempo de desarrollo y puesta en marcha del proyecto. El modelo contra el que parcialmente ataca la *API REST* se diseñó e implementó en una base de datos *MySQL* que encaja con la funcionalidad requerida.

En lo tocante al desarrollo, la utilización de tecnologías modernas y próximas al mundo laboral, posibilitó la adquisición de nuevos conceptos, conocimientos y experiencia.

El objetivo de este proyecto fue desarrollar una aplicación siendo lo mas riguroso y próximo al mundo real posible, además de aprender el proceso llevado a cabo.

12.2 Futuras líneas de trabajo

A continuación, se muestran posibles líneas de trabajo que se pueden implementar en un futuro, mejorando el producto:

- Mejoras visuales de la interfaz.
- Internacionalización de la aplicación (por ejemplo, traducción al inglés o a otros idiomas nacionales).
- Estudio de otra librería de gráficas que permita exportar, también, los cambios realizados pues JFreeChart solo nos deja exportar los valores por defecto.
- Permitir analizar varios archivos a la vez.

Apéndices

Apéndice A

Manual de usuario

A continuación, se muestra un manual de usuario detallando las principales funcionalidades de la aplicación web. Para acceder a la aplicación es necesario acceder desde un navegador web a la *URL* http://localhost:9000/.

Cabe aclarar que, en caso de que ocurra algún error mientras el usuario está haciendo uso de la aplicación, se mostrará un mensaje de error con el motivo del error y posibles soluciones.

A.1 Página de inicio

Como podemos observar en las imágenes, en la primera ventana de la aplicación tenemos un menú con las opciones de loguearse y registrarse. Para ambas hay que cumplimentar todos los datos.

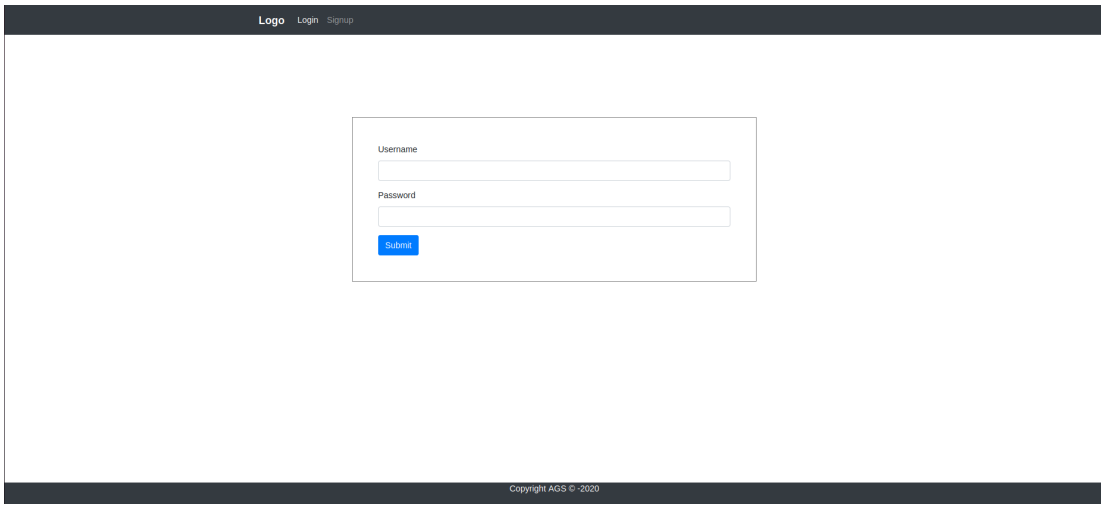


Figura A.1: login

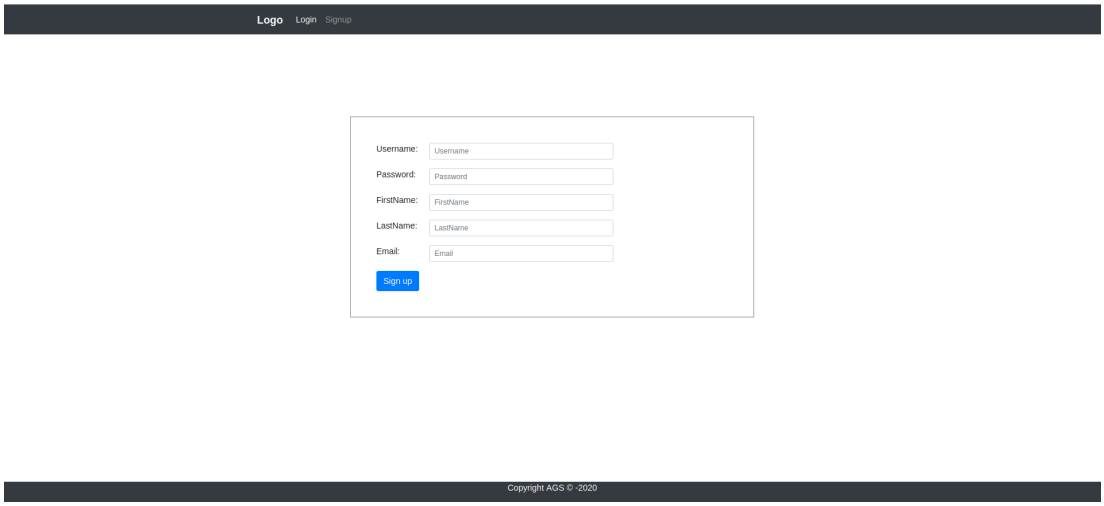


Figura A.2: registro

A.2 Subir archivos

Una vez autenticado en el sistema, nos encontraremos en la ventana que nos permitirá subir un archivo.

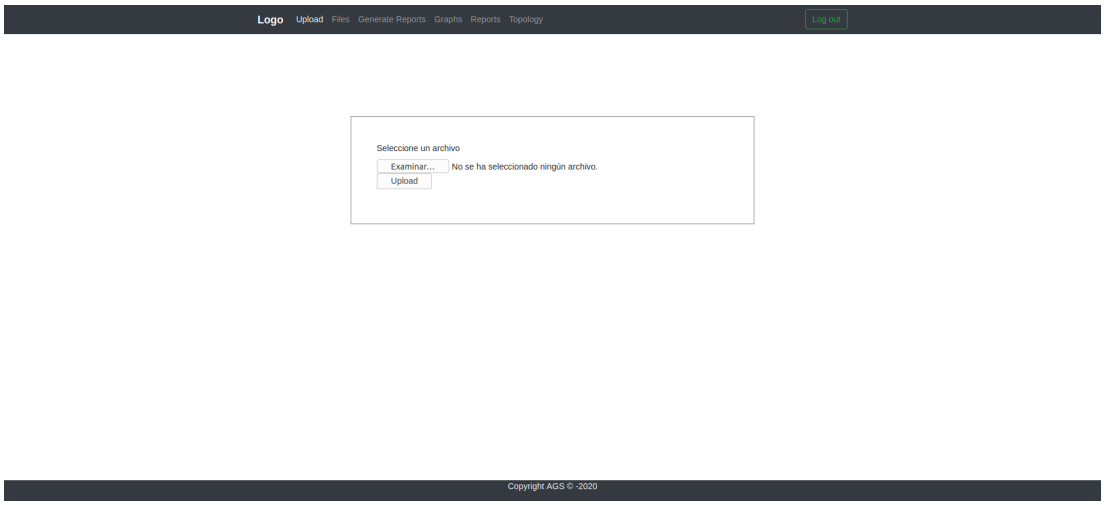


Figura A.3: Ventana de subida de archivo

El usuario pulsará el botón de examinar y escogerá el archivo que desee subir.

A.3 Descarga y visualización de archivos

Posteriormente podremos acceder al menú *files*, donde podremos ver todos los archivos subidos por el usuario y, al hacer clic sobre ellos, el usuario podrá elegir si descargarlos o

visualizarlos.

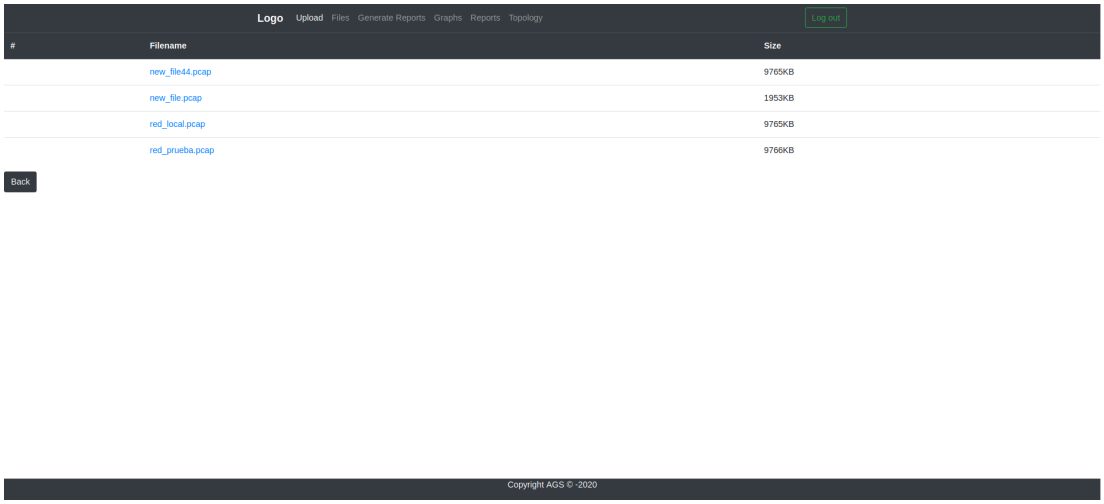


Figura A.4: Lista de archivos

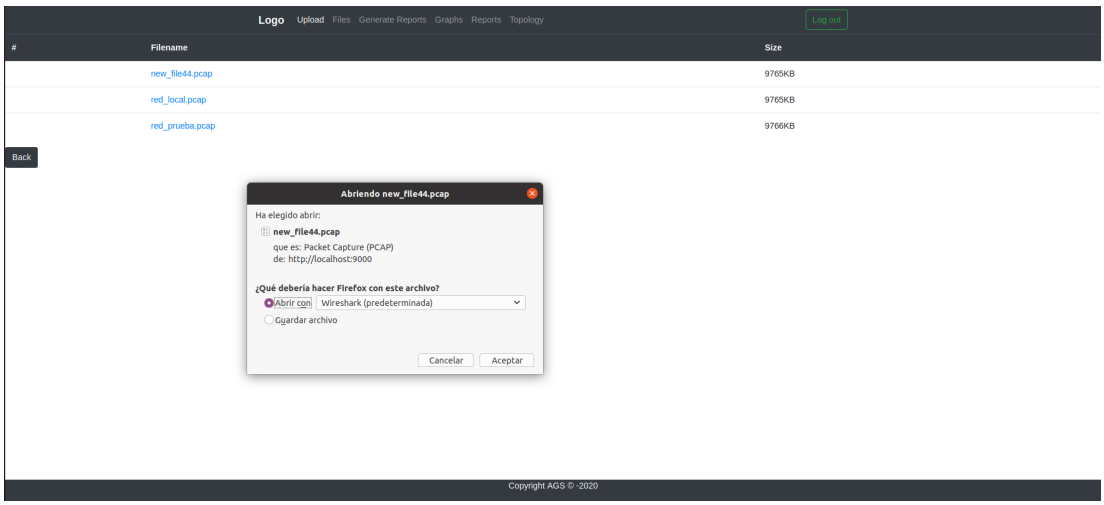


Figura A.5: Descarga y visualización de archivos

A.4 Procesado de archivos

Una vez tenemos archivos subidos, podremos acceder al menú *Generate*, donde aparecerá un formulario en el que podemos escoger el archivo que deseamos analizar y, también, podemos cumplimentar una serie de parámetros opcionales.

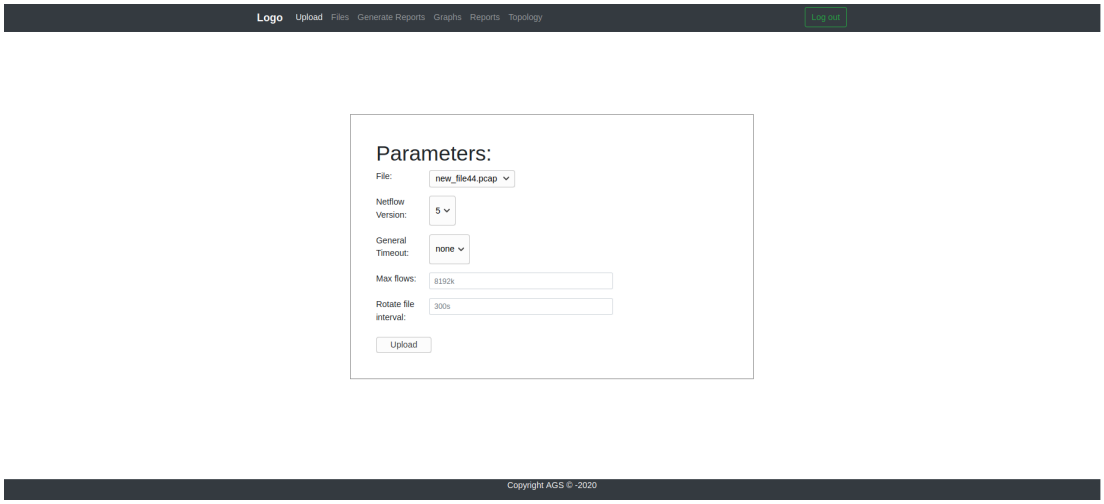


Figura A.6: Selección de archivos en el menú de parámetros

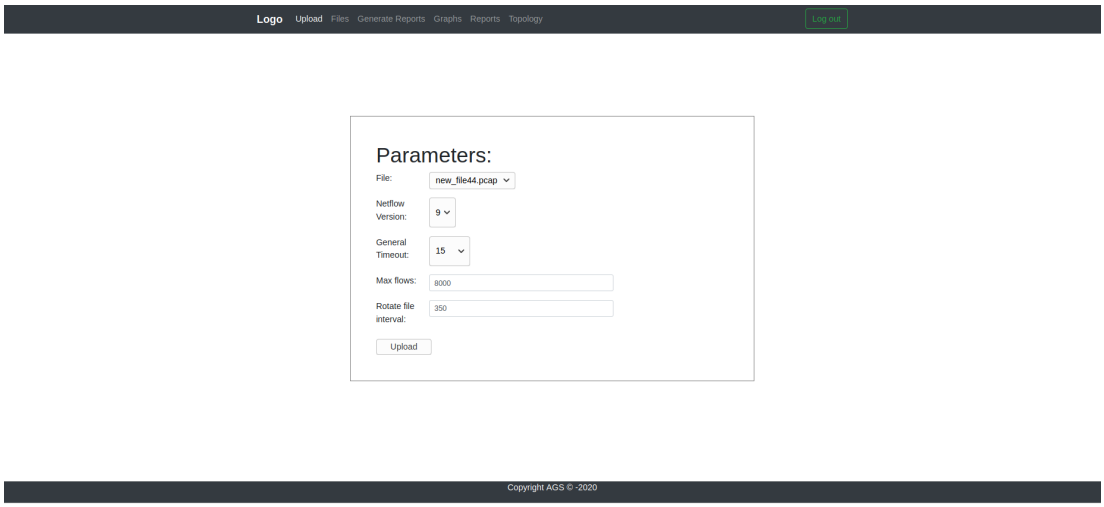


Figura A.7: Cumplimentando de parámetros

A.5 Visualización de gráficas

Una vez procesado el archivo, tras su mensaje de éxito, podemos acceder al menú *Graphs*, el cual nos mostrará las gráficas del último archivo procesado.

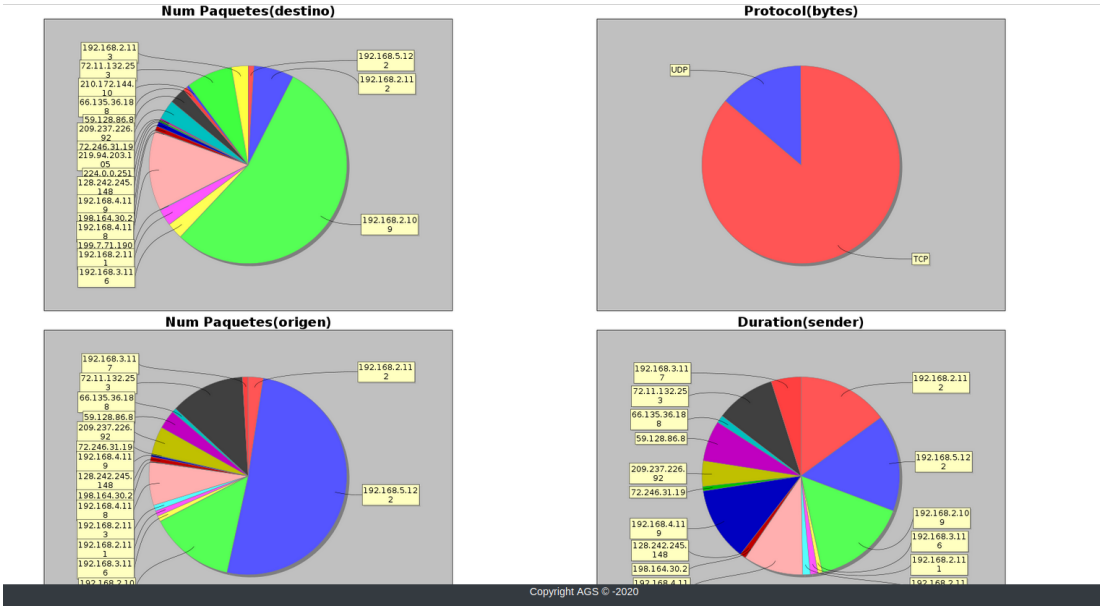


Figura A.8: Visualización de gráficas

A.6 Visualización y descarga de informes

Como en la visualización y descarga de archivos, podremos descargar y visualizar los informes en el menú *Reports*.

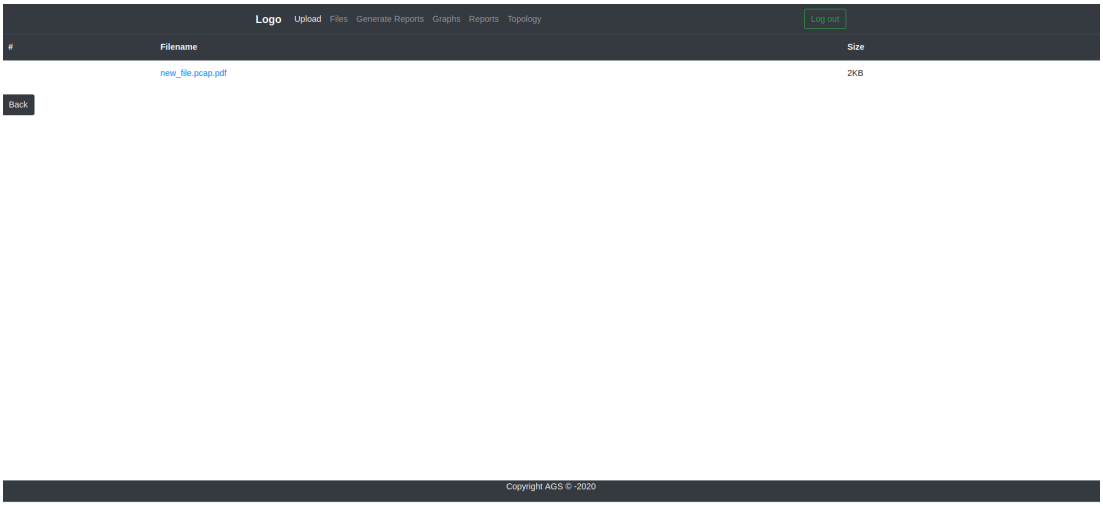


Figura A.9: Listado de informes

Report from file new_file44.pcap

Summary of the data

Total number of flows: 477 flows
Total number of bytes: 9651399 bytes
Total number of packets: 11542 packets
Average bytes per second: 341943 bps
Average packets per second: 51 pps
Average bytes per packet: 836 bpp
Average duration per flow: 38,99 seconds
Average packets per flow: 32,39 ppf
Average bytes per flow: 37.857,74 bbf
TCP flows: 40,08%
UDP flows: 59,92%
ICMP flows: 0%

Figura A.10: Parte de un informe

A.7 Generación de la topología

En la última pestaña del menú *topology*, podremos acceder para generar la topología del último archivo procesado, la cual podremos acercar para ver la dirección IP de cada nodo.

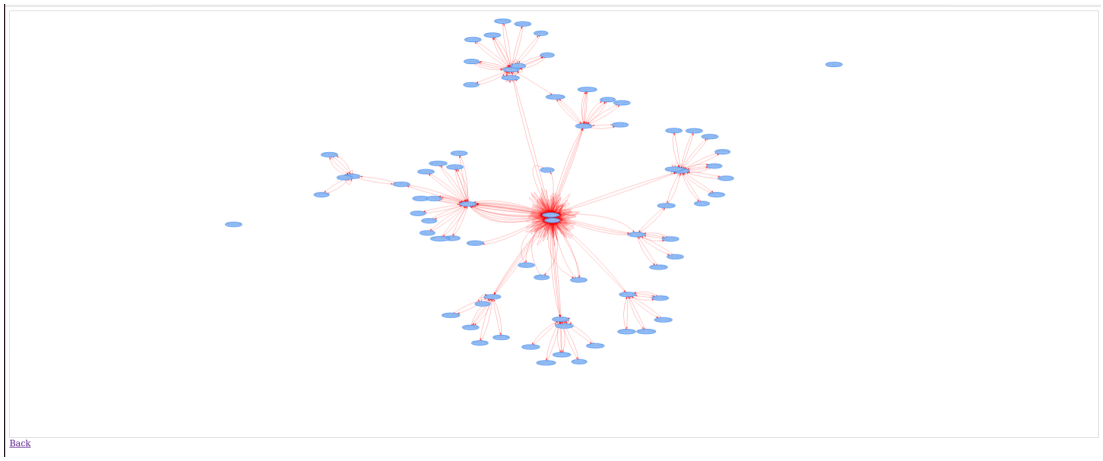


Figura A.11: Topología de una red

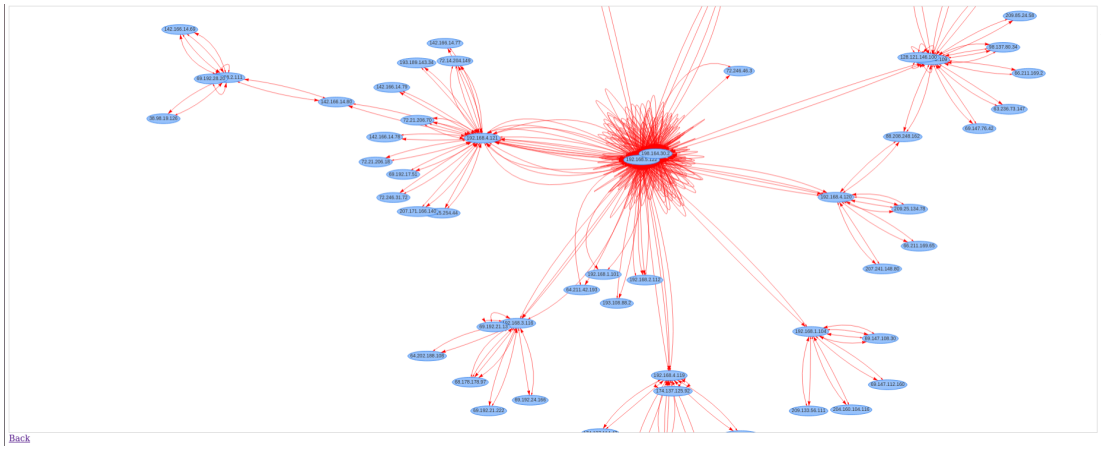


Figura A.12: Topología de una red ampliada

Lista de acrónimos

AOP *Aspect Oriented Programming.*

CLI *Command-Line Interface*

CRUD *Create, Read, Update, Delete*

CSS *Cascading Style Sheets*

HTML *HyperText Markup Language*

HTTP *HyperText Transfer Protocol*

IDE *Integrated Development Enviroment*

IoC *Inversion of Control*

JDK *Java Development Kit*

JPA *Java Persistence API*

JSON *JavaScript Object Notation*

MVC *Model-View-Controller*

PDF *Portable Document Format*

POM *Project Object Module*

REST *REpresentational State Transfer*

SPA *Single-Page Application*

SQL *Structured Query Language*

URL *Uniform Resource Locator*

W3C *World WideWeb Consortium*

XHTML *eXtensible HyperText Markup Language*

XML *Xtensible Markup Language*

Bibliografía

[1] Imgaen product backlog y sprint backlog. [En línea]. Disponible en: https://www.wearemarketing.com/es/blog/metodologia-scrum-que-es-y-como-funciona.html#:~:text=Scrum%20es%20una%20metodolog%C3%ADa%20de,equipos%20que%20manejan%20proyectos%20complejos.

[2] netfort. Flow analysis vs packet analysis. [En línea]. Disponible en: https://netfort-prod.k8s.corpwebsite.gcp.rapid7.com/content/uploads/PDF/WhitePapers/NetFlow-Vs-Packet-Analysis-What-Should-You-Choose.pdf

[3] Cisco. Introduction to cisco ios netflow - a technical overview. [En línea]. Disponible en: https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html

[4] Imagen topología estrella. [En línea]. Disponible en: https://sites.google.com/site/wikitopored/topologias-fisicas/estrella

[5] Imagen topología estrella extendida. [En línea]. Disponible en: https://clasificaciondelasredesblog.wordpress.com/2017/05/09/topologia-estrella-extendida/

[6] Imagen topología anillo. [En línea]. Disponible en: https://sites.google.com/site/wikitopored/topologias-fisicas/anillo

[7] Imagen topología anillo doble. [En línea]. Disponible en: https://redesybasededatoss.wordpress.com/topologia-del-doble-anillo/

[8] J. Gosling, B. Joy, G. Steele, G. Bracha, and A. Buckley. The java language specification java se 8 edition 2015. [En línea]. Disponible en: https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf

[9] S. Faulkner, A. Eicholz, T. Leithead, and A. Danilo. Html 5.1 2nd edition 2017. [En línea]. Disponible en: https://www.w3.org/TR/html51/single-page.html

[10] B. Bos. Cascading style sheets level 2 revision 2 (css 2.2) specification 2016. [En línea]. Disponible en: https://www.w3.org/TR/CSS22/css2.pdf

[11] An introduction to latex. [En línea]. Disponible en: https://www.latex-project.org/about/

[12] R. Johnson. Spring framework reference socumentation 5.0.0 2004. [En línea]. Disponible en: https://docs.spring.io/autorepo/docs/spring-framework/5.0.0.M1/spring-framework-reference/pdf/spring-framework-reference.pdf

[13] P. Webb and D. Syer. Spring boot reference socumentation. [En línea]. Disponible en: https://docs.spring.io/spring-boot/docs/current/reference/pdf/spring-boot-reference.pdf

[14] Tutorial: Using thymeleaf. [En línea]. Disponible en: https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.pdf

[15] Pagina oficial de jfreechart. [En línea]. Disponible en: http://www.jfree.org/jfreechart/

[16] Pagina oficial de vis.js. [En línea]. Disponible en: https://visjs.org/

[17] Eclipse platform technical overview, 2006. [En línea]. Disponible en: https://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.pdf

[18] R. Bharathan, *Apache Maven Cookbook*, 2nd ed. Packt publishing, 2015.

[19] S. Chacon and B. Straub, *Pro Git*, 2nd ed. Apress, 2014.

[20] Pagina oficial de overleaf. [En línea]. Disponible en: https://www.overleaf.com/about

[21] Mysql 8.0 reference manual. [En línea]. Disponible en: https://dev.mysql.com/doc/refman/8.0/en/introduction.html

[22] R. Fielding, U. Irvine, and J.Gettys. Hypertext transfer protocol – http/1.1. [En línea]. Disponible en: https://www.w3.org/Protocols/HTTP/1.1/rfc2616.pdf

[23] M. Masese, *Rest API Design Rulebook: Designing Consistent RESTful Web Services Interfaces.*, 2nd ed. O'Reilly, 2011.

[24] Manual de softflowd en linux. [En línea]. Disponible en: http://manpages.ubuntu.com/manpages/bionic/man8/softflowd.8.html

[25] Pagina oficial de nfdump. [En línea]. Disponible en: https://github.com/phaag/nfdump

[26] Convenio colectivo. resolucion 19 de agosto de 2020. [En línea]. Disponible en: https://www.boe.es/eli/es/res/2020/08/19/(1)

[27] Pagina about us de scrum. [En línea]. Disponible en: https://www.scrum.org/about

[28] Intrusion detection evaluation dataset (iscxids2012). [En línea]. Disponible en: https://www.unb.ca/cic/datasets/ids.html