ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Diseño e implementación de una aplicación para manipulación y análisis de capturas de tráfico

(Design and implementation of an application for handling and analysis of traffic captures)

Para acceder al Título de

Graduado en Ingeniería de Tecnologías de Telecomunicación

Autor: Qunfeng Wang

Septiembre - 2022

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Qunfeng Wang

Director del TFG: Roberto Sanz Gil

Título: "Diseño e implementación de una aplicación para manipulación y análisis de capturas de tráfico "

Title: "Design and implementation of an application for handling and analysis of traffic captures"

Presentado a examen el día:

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:

Presidente (Apellidos, Nombre): Roberto Sanz Gil

Secretario (Apellidos, Nombre): Antonio Quintela Incera

Vocal (Apellidos, Nombre): Marta García Arranz

Este Tribunal ha resuelto otorgar la calificación de:

Fdo.: El Presidente	Fdo.: El Secretario
Fdo.: El Vocal	Fdo.: El Director del TFG (sólo si es distinto del Secretario)
Vº Bº del Subdirector	Trabajo Fin de Grado Nº (a asignar por Secretaría)

INDICE

Lista de acrónimos

API: Application Programming Interface

ASCII: American Standard Code for Information Interchange

CACE: California Association of Compensatory Education

CMake: Connection Manager Administrator Kit

CMD: Command Prompt

DNS: Domain Name System

EBCDIC: Extended Binary Coded Decimal Interchange Code

FTP: File Transfer Protocol

GCC: GNU Compiler Collection

HDLC: High Level Data Link Control

HTTP: Hypertext Transfer Protocol

ICC: Intel C++ Compiler

ICMP: Internet Control Message Protocol

IMAP: Internet Message Access Protocol

IMF: Internet Message Format

IP: Internet Protocol

IPX/SPX: Internetwork Packet Exchange/Sequenced Packet Exchange

ISO: International Standards Organization

ITU-T: International Telecommunications Union – Telecommunication Standardization Sector

JSON: JavaScript Object Notation

LAN: Local Area Network

LAPB: Link Access Procedure D

LLC: Logical Link Control

MAC: Media Access Control

MinGW: Minimalist GNU for Windows

MOC: Meta Object Compiler

MSVS: Microsoft Visual Studio

NFC: Near Field Communication

OS: Operating System

OSI: Open Systems Interconnection

QML: Qt Meta Language

QoS: Quality of Service

SMTP: Simple Main Transfer Protocol

SSH: Secure Shell

TCP: Transmission Control Protocol

TFTP: Trivial File Transfer Protocol

UDP: User Datagram Protocol

UI: User Interface

UIT: University Information Technology

Unicode: Universal Code

USB: Universal Serial Bus

VLAN: Virtual Local Area Network

WLAN: Wireless Local Area Network

XML: Extensible Markup Language

Lista de figuras

Lista de tablas

Resumen

A medida que avanza el tiempo, Internet cada vez está tomando más importancia, no solo en lo que respecta a la tecnología de la comunicación, sino también en otros ámbitos como la educación, medicina, etc. En los que desempeña un papel relevante.

Por esta razón, el objetivo de este proyecto es diseñar una aplicación software para poder gestionar y analizar el tráfico de la red. Para su consecución se va a proceder a leer los archivos de capturas con formato PCAP o PCAPNG que almacenan las tramas y extraer los datos de la captura mostrándolos en una interfaz gráfica que ayuda a comprender la información obtenida.

A diferencia de Wireshark, se ha intentado que este software sea más user-friendly. Toda la información contenida en cada trama capturada se muestra en una ventana en la que podemos visualizar fácilmente el contenido de las capturas. Con ello se puede conocer si la información pertenece a una LAN o a tráfico externo, qué protocolos se están utilizando, cuántos datos se están pasando, en cuánto tiempo, etc.

Para crear esta aplicación se utilizan diferentes tipos de herramientas: algunas para crear la UI, otras para leer la captura o escribir datos a otros archivos, como es el caso de la base de datos. Afortunadamente, Python dispone de unas librerías a través de la cuales se puede hacer uso de dichas herramientas sin necesidad de tener que recurrir a otras aplicaciones.

Abstract

As time goes by, the Internet is becoming more and more important, not only in terms of communication technology, but also in other fields such as education, medical, etc. In which it plays a relevant role.

For this reason, the objective of this project is to design a software application to manage and analyze network traffic. To achieve this, we will proceed to read the PCAP or PCAPNG files, which store the network traffic, and extract the capture data by displaying them in a graphical interface that helps to understand the information circulating in the network.

As opposed to Wireshark, this software is more user-friendly. All network information is displayed in a window where we can easily view the contents of the PCAP files. With this we can find out whether the information belongs to the LAN or WLAN, which protocols are being used, how much data is being passed, etc.

Different types of tools are used to create this application: some to create the UI, others to read the capture or write data to other files or to a database. Fortunately, Python has libraries through which you can be used these tools without the need of other applications.

CAPÍTULO 1. INTRODUCCIÓN Y OBJETIVOS

La aparición de Internet fue un gran avance para la sociedad, permitió la revolución de muchos ámbitos, especialmente en la parte de comunicación. Hoy en día se ha convertido en un medio de comunicación que todos usamos cada día, ya sea para compartir cosas con un amigo, hacer una video llamada con tu familia o comunicarte con tus compañeros de trabajo. Además, Internet nos aporta muchas otras ventajas, como buscar información de cualquier tipo, consultar periódicos de cualquier país, comprar productos, etc. Con ello podemos obtener rápidamente lo que queremos y en cualquier lugar donde haya acceso a Internet.[3]

Sin embargo, si no hay una red que lo sostenga no podemos realizar grandes cosas, por lo tanto, la red es un punto clave de Internet. Esta red está formada por un grupo de computadores conectados que pueden enviarse datos entre sí, como un círculo social, un grupo de personas que se conocen entre sí e intercambiar información entre ellos. Puesto que los computadores se conectan entre sí formando redes, éstas también se conectan entre sí, de esta manera un computador puede comunicarse con otro sin estar en la misma red, todo esto gracias a Internet, que nos permite intercambiar información desde cualquier punto de mundo y en cualquier momento.

Pero para poder comunicar dos computadores se requiere el uso de técnicas de comunicación que puedan entender entre ellos, al igual que las personas, si queremos comunicar con una persona de otro país, uno tiene que hablar el idioma del otro para poder entenderse. Así surgieron los protocolos estandarizados, que permiten que dos o más dispositivos se puedan comunicar entre sí.

Existe diferentes protocolos a la hora de enviar información de un computador a otro. Por ejemplo, si los dispositivos de la misma red quieren intercambiar información entre ellos utiliza el protocolo Ethernet, pero si quieren comunicar con otros dispositivos de otra red utiliza el protocolo IP. Además, para garantizar que la información enviada llegue en el orden correcto se utiliza TCP, también es posible utiliza UDP en vez de TCP, cuando los datos que se van a enviar no importan que lleguen en orden.

Estos protocolos mencionados son los principales, pero existen muchos otros. El escenario que plantea este proyecto es desarrollar una aplicación para representar los

datos que contienen en estos protocolos, para conocer lo que se ha enviado y lo que recibe.

1.1. Objetivos

El objetivo principal de este proyecto es realizar una aplicación que nos permita analizar el tráfico de una red a partir de una captura generada por la aplicación Wireshark, por el comando tcpdump o por cualquiera que genere un formato de fichero compatible tipo pcap o pcapng.

Para poder conseguir este objetivo se crearán varias interfaces gráficas, para representar la información de la captura, el detalle del contenido de los protocolos, como por ejemplo las direcciones IP, direcciones MAC, longitud de la trama, tiempo relativo entre tramas, puertos TCP/UDP, etc. Dicha aplicación también deberá mostrar gráficas y tablas para que el usuario pueda visualizar fácil y rápidamente la información de la captura.

Para lograr todo esto, dividimos el proyecto en las siguientes partes:

- Parte Sesión: Una interfaz gráfica de sesión que se encargará de extraer los datos de un archivo de formato .pcap o .pcapng, pudiendo éste ser generado por la aplicación Wireshark o tcpdump. Además, será posible introducir un filtro a la captura antes de extraer los datos, y posteriormente guardar los datos extraídos en un fichero JSON.

- Parte Main: Una interfaz gráfica que muestra los datos de la captura que se ha guardado en el fichero JSON, generado en la parte sesión. Utilizando los diferentes botones de la ventana podemos visualizar la información de la captura.

- Parte Gráfica: Una interfaz gráfica que representa los datos de JSON, generado en la parte de Sesión en forma de gráficas, con objetivo de visualizar y conocer rápidamente los datos que contiene en la captura.

- Parte Gestión: En esta interfaz gráfica se representa la información de la captura en forma de tabla. Además, se podrá modificar la tabla (Eliminar o Añadir nuevas columnas) y exportar la información de la tabla a un fichero compatible con Excel, para poder manejar los datos de la tabla de forma más cómoda.

Para que todo funcione correctamente, se deben afrontar algunos conceptos teóricos y conocer el funcionamiento de algunas herramientas, como se explica en la estructura de esta memoria en el siguiente apartado.

1.2. Estructura de la memoria

La memoria de este trabajo está dividida en cinco capítulos, algunos dedicados a conceptos teóricos, cuyo conocimiento y compresión es vital para poder entender y conocer la información que contienen en la captura y otros sobre el desarrollo de la parte práctica de diseño e implementación, lo cual se explicará detalladamente mediante los pasos que se han seguido para diseñar las cuatro interfaces gráficas. Los cinco capítulos son los siguientes:

- Capítulo 1: Introducción. Se explica el objetivo del trabajo y cómo se va a realizar.

- Capítulo 2: Herramientas. Se definen las bases teóricas, se profundiza en los conocimientos necesarios y las herramientas en las que se ha apoyado la implementación de este trabajo.

- Capítulo 3: Desarrollo. Se realiza una explicación detallada de la parte visual, las cuatro interfaces gráficas, qué elementos y qué datos muestra cada interfaz gráfica, así como la parte funcional, indicando cómo extraer la información de la captura y cómo pasar estos datos a la interfaz gráfica.

- Capítulo 4: Conclusiones y líneas futuras. En este capítulo se analiza la consecución de los objetivos iniciales del trabajo, cómo fue el resultado del trabajo. Además, se exponen algunas mejoras del trabajo, tanto de la parte visual como de la parte funcional.

CAPÍTULO 2. HERRAMIENTAS

Los códigos fuente de este trabajo están basados en Python, por lo tanto, la mayoría de las herramientas que se han utilizado son las librerías propias de Python, como Pandas, Numpy, Sqlite3, Pyshark y Matplotlib.

Además, para conocer y poder manejar los datos de las capturas generadas por la herramienta Wireshark (ficheros con extensión .pcap), también se realiza un estudio sobre éste, para conocer qué tipo de información puede capturar, especialmente qué protocolos aparecen en la captura. También se hará una breve introducción sobre el modelo OSI, para conocer la funcionalidad de cada capa y los protocolos más relevantes en este trabajo.

Por el ultimo, se mostrarán las herramientas que se han utilizado para guardar la información de la captura, en formato de base de datos, de fichero compatible con Excel o para la creación de la interfaz gráfica.

2.1. Modelo OSI

El modelo OSI es un modelo conceptual, creado por la Organización Internacional de Normalización (ISO) en 1980 [1], y fue publicado en el año 1983 por la Unión Internacional de Telecomunicaciones (UIT). Este modelo conceptual permite la comunicación entre diferentes sistemas utilizando estándares. El modelo OSI dio la base para el modelo TCP/IP.

El modelo OSI se puede entender como un lenguaje universal de comunicación entre diferentes sistemas de redes de datos.[2] Está dividido en siete capas según su funcionalidad, y cada capa se comunica con sus capas adyacentes superior e inferior.

Además, cada capa trabaja de forma independiente de las demás, sin necesidad de saber cómo funciona el resto de las capas, aunque existe una comunicación entre ello. De esta forma cada capa es modificable sin que exista alguna influencia a los demás.

Dentro de cada capa se establecen sus propios protocolos, algunos protocolos se utilizan para comunicar con las capas superiores o inferiores, otros se utilizan para comunicar

con sus homólogos o peer, es decir el mismo protocolo, pero situado en otro extremo de la comunicación.

Las ventajas de dividir en capas facilitan la compresión sobre la transmisión de datos, ya que podemos dividirlo en varias partes más simples, además se puede evitar los problemas de compatibilidad de la red. Detalla las capas permite un mejor aprendizaje y proporciona a los fabricantes un conjunto de estándares que asegura una mayor compatibilidad e interoperabilidad de las diferentes tecnologías utilizadas por diferentes países.

2.1.1. Capa física

La capa física es la primera capa del modelo OSI, aunque está situada en la parte más baja. Es la base de todo el modelo de interconexión de sistemas abiertos. Define las especificaciones eléctricas, mecánicas, de procedimiento y funcional para la activación, el mantenimiento y la desactivación de las conexiones físicas entre los dispositivos de la red. Además, se especifica el medio de transmisión de los datos (guiados y no guiados), y cómo se envían los datos (simplex, half-duplex o full-duplex). [4]

La figura 2.1 representa una forma de enviar los datos, en este caso el medio que se utiliza es un medio guiado, por cable, y los datos se enviar en bits (0 o 1), que es la unidad de dato de esta capa.

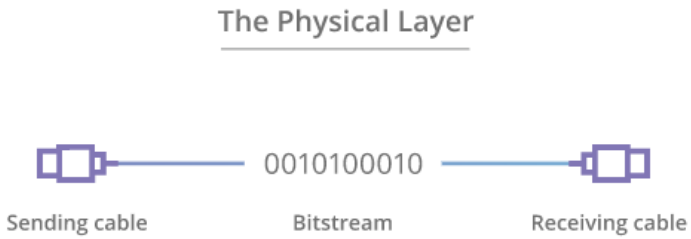


Figura 2.1. Capa física del modelo OSI

La capa física define dos medios para la comunicación de los datos, el medio guiado, que son los cables, los más conocidos son los pares trenzados (RJ-45), el cable coaxial y la fibra óptica. Y el medio no guiado, que es el canal inalámbrico. Para ambos medios se definen las características que deben tener, tanto característica mecánica como eléctrica. Esto incluye el ancho y el largo del cable, la modulación que se utiliza para la transmisión, los niveles de tensión utilizados, la velocidad binaria, el código de línea y la sincronización entre los dispositivos.

2.1.2. Capa de enlace

Esta capa se encarga fundamentalmente de la parte de direccionamiento, cómo acceder al medio (guiado o no guiado) que ha proporcionado la capa física y cómo transmitir los datos al dispositivo final. Además, es capaz de comprobar (a veces también corregir) los datos recibidos con errores, realizar un control de fuljo de los datos y ordenar las tramas en la recepción.[5]

La capa de enlace también se encarga de tomar los paquetes de la capa superior (capa de red) y dividirlos en trozos más pequeños denominados tramas. De esta manera, la transmisión será más rápida y con menos errores, haciendo que al receptor le lleguen las tramas ordenadas, como muestra en la figura 2.2:

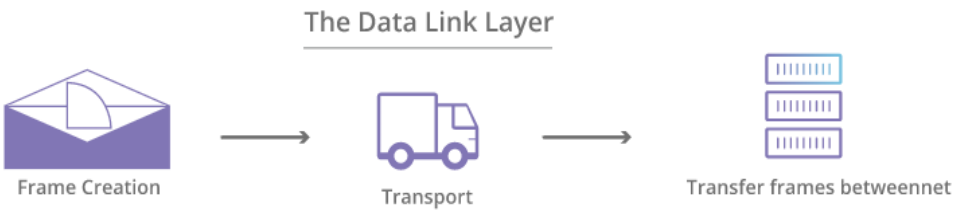


Figura 2.2. Capa de enlace del modelo OSI

La capa de enlace a la vez está dividida en dos subcapas, la subcapa LLC y la subcapa MAC, como se muestra en la figura 2.3:

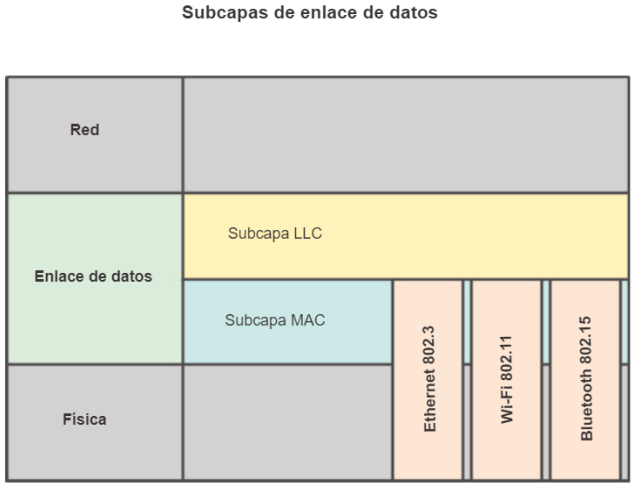


Figura 2.3. Subcapas de enlace del modelo OSI

La subcapa LLC proporciona servicios a la capa de red y la capa MAC se encarga de la parte de direccionamiento y la delimitación de los datos como se ha comentado anteriormente.[5]

Algunos estándares de esta capa:

- ISO 1745: los procedimientos básicos de control para sistemas de comunicación de datos**.**

- ISO 3309: Protocolo de control de enlace de datos de alto nivel (HDLC).

- ISO 7776: Describe los procedimientos de enlace de datos DTE compatibles con X.25 LAPB

- X.25: Es un estándar ITU-T para las redes amplias, su protocolo LAPB está basado en HDLC.

2.1.3. Capa de red

La capa de red es la responsable de la transferencia de los datos entre los diferentes nodos de la red. En el caso de que los nodos estén bajo la misma red, esta capa no actuará. Además, la capa de red se encarga de realizar la fragmentación, dividir los datos que vienen de la capa superior en unidades más pequeñas llamadas paquetes, así como realizar el enrutamiento o búsqueda de rutas para encontrar el camino óptimo entre los nodos.

Cuando un canal físico se establece y es utilizado por un par de usuarios, suele haber mucho tiempo de inactividad entre ellos, por ello, se proporcionan canales lógicos y un circuito virtual para que los usuarios compartan un mismo enlace para enviar y recibir la información. De esta manera reducir el tiempo de inactividad del canal y tener más recursos para otros usuarios.

Figura 2.4. Capa de red del modelo OSI

La figura 2.4 muestra el proceso de envío, antes de ser enviados los datos. Para ello, se crea lo que se denomina paquetes, y el receptor ordena los paquetes recibidos.

Algunos estándares que se incluyen en esta capa son las siguientes:

- IPX/SPX: Es una familia de protocolos de red desarrollado por Novell, se utiliza para la transferencia de los datos en una red Netware. Es un protocolo en modo datagrama no orientado a la conexión.

- ICMP: Es un protocolo de control de los datos de Internet, que se encarga de enviar mensajes de error y notificar de situaciones anómalas en la red.

- IP: Es un protocolo de comunicación de datos, no orientado a la conexión y se encarga de encaminar correctamente los paquetes de datos a través de redes, entre el origen y el destino, pero sin asegurar la recepción de los mismos.[5] Las dos funcionalidades principales de este protocolo son: el direccionamiento IP (un número que identifica una interfaz de un dispositivo) y el enrutamiento (encontrar un camino o una ruta de la red para que los paquetes de datos lleguen a su destino).

2.1.4. Capa de transporte

Esta es la capa 4 del modelo OSI, se ocupa de las conexiones extremo a extremo entre los dispositivos de la red, y tiene una función de amortiguación de la transmisión.

Cuando la calidad del servicio en la capa de red no cumple con los requisitos establecidos, la capa de transporte proporciona una mejora del servicio para cumplir con los requisitos. Además, cuando la calidad del servicio en la capa de red es buena, la capa de transporte también se puede multiplexar, es decir, se pueden crear varias conexiones lógicas en la misma conexión de red, para que pueda enviar los datos por las diferentes conexiones aumentando, de esta manera, la eficiencia de transmisión.

Figura 2.5. Capa de transporte del modelo OSI

Además, la capa de transporte proporciona un control de flujo y un control de errores para los datos enviados. El control de flujo sirve para determinar la velocidad optima de la transmisión de los datos, para que el receptor no se sature debido a una conexión demasiado lenta. El control de errores asegura que el receptor reciba toda la información que ha sido enviada por el emisor sin ninguna perdida. En caso de que haya perdido algún segmento, pedirá el reenvío de la información.

Los protocolos más importantes de esta capa son TCP y UDP. TCP es un protocolo orientado a la conexión que prioriza la integridad de los datos, estableciendo una conexión segura con el nodo destino, asegurando que todos los datos lleguen a su destino en correcto orden, sin ninguna pérdida. El nodo destino confirma los paquetes que le han llegado y, en caso de que haya alguno perdido pide la retransmisión del mismo.

Por el contrario, UDP es un protocolo no orientado a la conexión que sólo asegura la velocidad de entrega, no asegura que las tramas lleguen en orden y no haya pérdida de paquetes.

2.1.5. Capa de sesión

Esta capa es el responsable de establecer, mantener y finalizar las conexiones entre emisor y el receptor. Por lo tanto, la principal función de esta capa es asegurar que, dada una sesión establecida por dos dispositivos, se pueda mantener las operaciones definidas al inicio y al final de la sesión y, en caso de que se produzca algún problema, reanudarla.

Figura 2.6. Capa de sesión del modelo OSI

También permite cifrar los datos y comprimirlos en fragmentos más pequeños, la sincronización y resincronización en el caso de que se haya perdido el sincronismo entre el emisor y el receptor. Para cumplir estas funcionalidades, se necesita un gran número de servicios. A continuación, se describen las principales funciones:

- Establecimiento de conexión entre dos dispositivos: Para establecer una conexión entre dos nodos de red (peer-to-peer), primero el protocolo de capa de red asigna las direcciones IP de los dos nodos. Después se seleccionan los parámetros de calidad de servicio (QoS) necesarios para transmitir la información y, finalmente, se negocian los parámetros de la sesión.

- Fase de transferencia de datos: La transferencia de los datos se envía de forma organizada y sincronizada entre los dos nodos de la red. Por ejemplo, si el nodo emisor enviar 100 MB, la capa de sesión podría fijar un punto control en 5 MB, enviando así en

paquetes de menor tamaño.[2] En caso de desconexión o caída de la red permite sólo reenviar la parte perdida y no enviar el archivo completo.

- Liberación: La desconexión de la conexión se realiza a través de los comandos de liberación.

La capa de sesión junto con las capas de presentación y de aplicación componen las tres capas superiores de OSI, proporcionando el procesamiento de la distribución, la gestión de la conversación y la representación del mensaje.

2.1.6. Capa de presentación

Esta capa se encarga de codificar, cifrar y transformar los datos a caracteres, de manera que el receptor pueda reconocer la información que le ha llegado, ya que los distintos dispositivos pueden tener diferente representación de los caracteres.

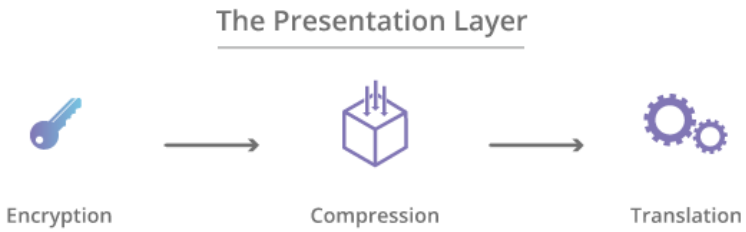


Figura 2.7. Capa de presentación del modelo OSI

Si la conexión entre los dispositivos es una conexión cifrada, la capa de presentación también se encarga de añadir el cifrado en el extremo del emisor, así como decodificar el mensaje en el extremo del receptor, de esta manera asegura que los datos sean legibles.

Si el archivo enviado tiene un tamaño grande (varios megabytes) también encarga de comprimir los datos en unidades más pequeñas, con el fin de mejorar la velocidad de envió y la eficiencia de la comunicación.

Los tres tipos de codificación más usados son los siguientes:

- ASCII: proporciona la mayoría de los caracteres necesarios, utiliza 7 bits para la representación de los caracteres.

- EBDCIC: diseñado por IBM para su uso en mainframes. Esta codificación no es compatible con otros métodos de codificación de caracteres.

- Unicode: puede codificar cualquier carácter conocido, puede usar 8, 16 o 32 bits para la codificación de cada símbolo.

2.1.7. Capa de aplicación

Esta capa es responsable de los servicios de la red para las aplicaciones que usan los usuarios finales y define los protocolos que utilizan las aplicaciones para el intercambio de datos, por ejemplo, correo electrónico (SMTP), gestores de bases de datos o servicios para el envío de ficheros a otros dispositivos, como FTP y TFTP.

El usuario no interactúa directamente en el nivel de aplicación, sino que utiliza algún programa para interactuar con esta capa, lo que oculta la complejidad subyacente.

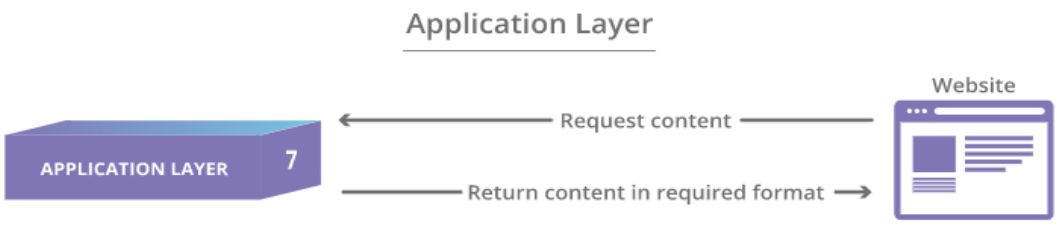


Figura 2.8. Capa de aplicación del modelo OSI

Los protocolos más importantes que incluyen en esta capa son los siguientes: FTP (File Transfer Protocol), SSH (Secure Shell), SMTP (Simple Mail Transfer Protocol), IMAP (Internet Message Access Protocol), DNS (Domain Name Service) y HTTP (Hypertext Transfer Protocol).[6]

2.2. Wireshark

Wireshark fue creado en 1998 por Gerald Combs [7], cuando estaba trabajando para un pequeño proveedor de servicios de Internet. En ese momento, los productos de análisis de protocolos eran bastante caros, propietarias o ambas cosas, por lo que escribió Ethereal, la versión anterior de Wireshark.

En 2006, Combs se fue a CACE Technologies, donde utilizó los repositorios de Subversión de Ethereal como base para crear Wireshark. Con la llegada de Wireshark cambio la situación de los analizadores de protocolos, ya que es gratuita y de código abierto, accesible por lo tanto para todo el que lo desee.

Además, algunas de las razones por las que se utiliza esta herramienta son:

- Permite solucionar problemas de red.
- Para la seguridad de la red es muy importante, ya que podemos examinar los problemas de seguridad de la red.
- Permite controlar la calidad de la red para verificar estado de las aplicaciones de red.

- Se puede utilizar para depurar las implementaciones de los protocolos.
- Aprender los aspectos internos de los protocolos de la red.

Debido a estas razones, Wireshark se convirtió en el mejor analizador de paquetes, también obtuvo varios premios, como eWeek, infoWorld, PC Magazine, etc. Actualmente, Combs continúa trabajando sobre el Wireshark, además, hay más de 600 autores que están participando en la mejora de esta herramienta.

2.2.1. Para qué sirve y sus características principales

Wireshark es un analizador de protocolos utilizados para el estudio de las comunicaciones y para la resolución de los problemas que aparecen en las redes. Un analizador de protocolos se entiende como un dispositivo de medición para examinar lo que ocurre dentro de la red, igual que un electricista que utiliza un voltímetro para examinar lo que ocurre dentro de un circuito eléctrico. [8]

Wireshark es una herramienta didáctica, por lo que cuenta con todas las características estándares y la funcionalidad que proporciona es similar a tcpdump, pero añade una interfaz gráfica, en este caso usa Qt, en el apartado siguiente vamos a hablar sobre ello.

Además, es un software libre, es decir que se puede ejecutar en la mayoría de los sistemas operativos Unix, y es compatible con Linux, Solaris, FreeBSD, NetBSD, OpenBSD, Android, macOS y Microft Windows.

A continuación, se detallará algunos aspectos importantes de Wireshark:

- Es una herramienta muy robusta, tanto en modo promiscuo como en modo no promiscuo.
- Permite almacenar los datos capturados en un archivo, para poder leerlo o realizar otras operaciones con ella.
- La interfaz es muy flexible, representa tanto la información en texto, como gráficas.
- Admite el formato estándar de archivos de tcpdump/WinDump y otros programas de captura de paquetes.
- Inspección profunda: más de 360 protocolos reconocidos, información detallada sobre cada protocolo.
- Captura datos en tiempo real desde una interfaz de red.
- Importar los datos desde archivo de texto que contengan datos en hexadecimal.
- Filtrar y buscar paquetes según varios criterios.
- Crear estadísticas.

Además, Wireshark proporciona un método de seguridad para los usuarios, cuando el usuario desea capturar paquetes directamente de la interfaz de red, necesita tener el permiso de superusuario para poder realizar la captura.

Wireshark es una herramienta compleja y fundamental para la actividad de análisis de tráfico de la red, pero hay que tener en cuenta que no es un sistema de detección de

intrusos. Aunque puede ayudarnos a descubrir lo que está pasando en la red, para ver si alguien está haciendo algo malo en la red a la cual estamos conectados.

Wireshark se considera como un analizador de protocolo pasivo, es decir no actúa directamente sobre la red, sino que únicamente inspecciona y mide los paquetes que circulan por la red, no realiza ninguna transmisión de los datos.

2.2.2. Interfaz y ejemplo de uso

Wireshark permite capturar el tráfico en diferentes medios o interfaces de red, como Ethernet, LAN, inalámbrica, Bluetooth, USB y mucho más, tanto interfaz física como virtual. Algunos interfaces específicos pueden tener alguna limitación por algunos factores de tipo de hardware y el sistema operativo que están usando. [9]

En la figura 2.9 muestra las limitaciones para los diferentes medios según el sistema operativo:

Interface	AIX	FreeBSD	HP-UX	Irix	Linux	macOS	NetBSD	OpenBSD	Solaris	Tru64 UNIX	Windows
ATM	?	?	?	?	✓	✗	?	?	✓	?	?
Bluetooth	✗	✗	✗	✗	✓[1]	✗	✗	✗	✗	✗	✗
CiscoHDLC	?	✓	?	?	✓	?	✓	✓	?	?	?
Ethernet	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FDDI	?	?	?	?	✓	✗	?	?	✓	?	?
FrameRelay	?	?	✗	✗	✓	✗	?	?	✗	✗	✗
IrDA	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
PPP[2]	?	?	?	?	✓	✓	?	?	✗	?	✓
TokenRing	✓	✓	?	✗	✓	✗	✓	✓	✓	?	✓
USB	✗	✗	✗	✗	✓[3]	✗	✗	✗	✗	✗	✗
WLAN[4]	?	✓	?	?	✓	✓	✓	✓	?	?	✓
Loopback (virtual)	?	✓	✗	?	✓	✓	✓	✓	✗	✓	N/A[5]
VLAN Tags (virtual)	✓	✓	✓	?	✓	✓	✓	✓	✓	✓	✓

Figura 2.9. Limitaciones de los sistemas operativos

En la figura podemos ver que los datos de Bluetooth sólo se puede utilizar Linux para realizar la captura. Esto indica que, dependiendo del medio, debemos usar un sistema operativo u otro, pero para algunos interfaces como VLAN, Ethernet, podemos realizar capturas en todos los sistemas operativos.

A continuación, se muestra un ejemplo sobre la realización de una captura de tráfico utilizando la herramienta Wireshark.

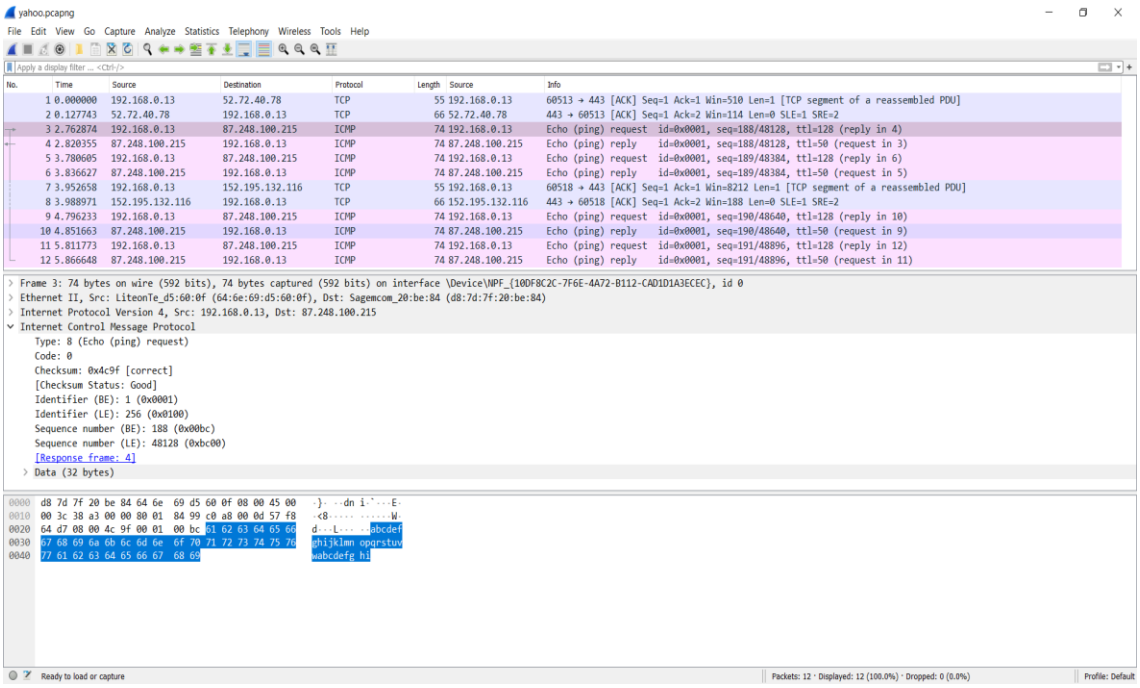


Figura 2.10. Captura Wireshark (Yahoo)

La captura de la figura 2.10 contiene el tráfico de un ping a la dirección IP 87.248.100.215, que corresponde con uno de los servidores de Yahoo.

Como se puede ver, la parte superior presenta la información básica de la captura, como las direcciones IP origen/destino, el tiempo que tarda en enviar cada trama o el protocolo de nivel más alto. En la parte central, se muestran los detalles sobre cada protocolo y la parte inferior muestra la información de la captura en formato hexadecimal.

A partir de esta captura se pueden realizar operaciones, como filtrar, exportar, tanto en formato de texto como formato hexadecimal, y muchas posibilidades más.

2.3. QT

Qt es una herramienta para desarrollar la interfaz de usuario (UI) para múltiples plataformas, como ordenadores o dispositivos móviles. Las plataformas soportadas incluyen Linux, OS X, Windows, Android o IOS. [10]

Qt no es un lenguaje de programación, es un marco de trabajo en C++, donde se utiliza un procesador MOC (Meta-Object Compiler), que sirve para ampliar el lenguaje C++ con características como señales y ranuras (slots). A partir de los códigos fuentes generados por C++, MOC genera un estándar, de esta forma los framework y las aplicaciones/bibliotecas que ha utilizado son compatibles con los compiladores como Clang, GCC, ICC, MinGW y MSVS.

Qt tiene tres componentes principales. La primera sería el Qt Framework, un marco de trabajo donde incluye APIs intuitivas que son utilizados por C++ y JavaScript con Qt Quick para crear la interfaz de usuario.

La segunda es el Qt Creator IDE, que es un entorno para desarrollar la interfaz usuario para multiplataforma, que incluye herramientas de diseño de IU y depuración en el dispositivo.

La última es Tools and Toolchains, que son herramientas que sirven para el soporte de internacionalización, cadenas de herramientas integradas, construcción con Cmake, etc.

2.3.1. PyQt5

Como se ha comentado en el apartado 2.3, además de C++, Qt también ofrece APIs para otros lenguajes de programación, como QML y Python. Permite acceder a las funcionalidades de Qt usando otras herramientas de programación, que incluyen funcionalidades como los servicios de localización y posicionamiento, multimedia, conectividad NFC y Bluetooth. También permite desarrollar la interfaz de usuario a través de las APIs. [11]

PyQt es un conjunto de enlaces/librerías de Python para acceder a las funcionalidades de Qt. Contiene más de 35 módulos y permite utilizar Python para desarrollar la interfaz gráfica de las aplicaciones. También se puede incrustar en aplicaciones basadas en C++ para que los usuarios puedan configurar o mejorar su funcionalidad.

Actualmente, la versión de PyQt es la 6, aunque para este trabajo se ha utilizado PyQt5, ya que aporta elementos suficientes para crear la interfaz gráfica.

Con PyQt5 se puede acceder múltiples APIs de Qt, como Qstring, Qvariant, etc. A continuación, se detallará algunos de los módulos que ofrece PyQt5 y hemos utilizado en el proyecto:

- **QtWidgets:** Este módulo proporciona un conjunto de elementos para la creación de UI, como botón, texto, barra de texto, imágenes, etc.
- **QtGui**: Contiene las clases para la integración de sistemas con la ventana, manejo de eventos, la interfaz gráfica 2D, imágenes básicas, fuentes y formato de texto, etc. También contiene un conjunto de enlaces de OpenGL y OpenGL ES. Los usuarios pueden utilizar estas APIs para extraer los datos de nivel superior (como módulo QtWidgets).
- **QtCore**: Contiene las clases principales de Qt, incluye el bucle de eventos y los mecanismos de señales y ranuras(slots) de Qt. También incluyen animaciones, las máquinas de estado, los hilos, los archivos mapeados, la memoria compartida, la configuración de la ventana del usuario y de la aplicación.

A continuación, se explicarán las clases que se han utilizado en estos módulos, empezado por QtWidgets:

- **QmainWindow:** Es la ventana principal que proporciona un marco para construir la interfaz gráfica de una aplicación.
- **Qwidget:** Es una clase base de todos los objetos de la interfaz gráfica. Pudiendo también recibir eventos del ratón, teclado y otras operaciones del sistema operativo y representarlas en la interfaz gráfica.
- **Qframe:** Es la clase base de widgets, que permite crear marcos simples sin contenido. A diferencia de Qwidgets, éste nos permite editar la forma del marco, la sombra del marco y el grosor del borde.
- **Qlabel**: Proporciona una visualización de texto o imagen.
- **QpushButton**: El botón de un solo pulso, al pulsar este botón realiza una acción específica.
- **QlineEdit:** Es un editor de texto de una sola línea.
- **QcomboBox:** Representa una lista de opciones al usuario de forma que ocupe el mínimo de espacio posible en la pantalla.
- **QscrollArea:** Proporciona una vista de desplazamiento sobre un widget.
- **QstackedWidget:** Proporciona una pila de widgets (paginas), donde el usuario puede editar varios widgets y es visible en el mismo marco de la ventana.

Para QtGui tenemos:

- **Qpixmap:** Permite utilizar imágenes del dispositivo como parte de la ventana.
- **Qicon:** Proporciona iconos escalables en diferentes modos y estados.
- **Qfont:** Especifica la fuente del texto utilizado.
- **Qcursor:** Proporciona un cursor de ratón con una forma arbitraria.

Finalmente, para QtCore tenemos:

- **Qsize:** Define el tamaño (la anchura y la altura) de un objeto.
- **QcoreAplication:** Proporciona un bucle de eventos para aplicaciones de Qt.

2.3.2. Qt Designer

Qt Designer es una herramienta de Qt para diseñar y construir interfaz gráfica de usuario, pero gratuita [12]. Incluye los módulos para diseñar Widgets, diálogos o ventanas principales(QmainWindow) mediante formularios en pantalla, de una manera sencilla. Además, tiene la capacidad de realizar una visualización previa a los diseños, para asegurar de que funcionan como pretendía y permitir crear prototipos, antes de tener escrito algún código.

Qt Designer utiliza archivo XML(.ui) para almacenar los diseños, y no genera ningún código por sí mismo. En Qt incluye el módulo uic para generar el código C++, en nuestro caso, se ha utilizado el comando pyuic5 de PyQt5 para generar los códigos.

El código que se genera por pyuic5 tiene una estructura idéntica a la generada por uic de Qt y se puede utilizar de la misma manera, y está estructurado como una única clase

llamado setpUI(), que se deriva del tipo de objecto de Python. El nombre de la clase y el nombre de los objectos son los mismo que aparece en Qt Designer.

En la clase generada se toma un único argumento, que puede ser Qdialog, Qwidget o QmainWindow, depende de lo que se ha establecido en Qt Designer.

A continuación, muestra dos ejemplos sobre la utilización de estos módulos. El primer ejemplo, en la figura 2.11, utiliza el módulo UIC de PyQt5 para hacer uso del fichero .ui, y a partir de ahí, podemos utilizar todos los componentes que hemos definido en Qt Designer. [28]

```
from PyQt5 import uic, Q

class ventana(QMainWindow):
    def __init__(self):
        super(ventana, self).__init__()
        uic.loadUi("diseño.ui")
```

Figura 2.11. Código Python, uso del módulo UIC

El segundo ejemplo, mostrado en la figura 2.12, genera los códigos de Python utilizando el comando pyuic5.

```
from PyQt5.QtWidgets import QDialog
from ui_imagedialog import Ui_ImageDialog

class ImageDialog(QDialog):
    def __init__(self):
        super(ImageDialog, self).__init__()

        self.ui = Ui_ImageDialog()
        self.ui.setupUi(self)
        self.ui.okButton.clicked.connect(self.accept)
        self.ui.cancelButton.clicked.connect(self.reject)
```

Figura 2.12. Código Python, uso del comando pyuic5

El comando pyuic5 genera todos los objetos que hemos definido en Qt Designer, tales como los botones, el color y el tamaño de la ventana o marco, tipo y tamaño de las letras, etc.

2.4. Pyshark

Pyshark es una librería de Python que utiliza la capacidad de tshark (utilidad de línea de comandos de Wireshark) para analizar el tráfico desde un archivo de captura (FileCapture) o una captura en tiempo real (LiveCapture). Para este proyecto vamos a leer la información del tráfico de un archivo.

Utilizamos el siguiente comando para leer la información de una captura:

cap = pyshark.FileCapture(path_to_file)

En la línea de código hay que especificar la ruta donde se encuentra el archivo de tipo pcap o de pcapng para poder leer la captura, además, es posible filtrar la captura pasándole el filtro que deseamos aplicar. Existe dos tipos de filtros: el filtro de visualización de Pyshark, que es útil para analizar el tráfico centrado en la aplicación, y el filtro BPF, que tiene la misma capacidad que el filtro de visualización, pero no ofrece tanta flexibilidad. [13]

La figura 2.14 muestra un ejemplo de los dos filtros:

```
filtered_cap = pyshark.FileCapture(path_to_file, display_filter='http')
filtered_cap2 = pyshark.LiveCapture('eth0', bpf_filter='tcp port 80')
```

Figura 2.13. Filtro de visualización y filtro BPF

Actualmente, hay problemas con los filtros BPF en FileCapture, por lo que sólo se usa en LiveCapture.

Además de introducir un filtro, también es posible añadir los siguientes elementos a la hora de leer la información de una captura: [15]

- Keep_packet: Se utiliza para conservar la memoria cuando se leen capturas de gran tamaño.
- Only_summary: La información de la captura se devuelve sólo como resumen de los paquetes, de esta forma aumenta la velocidad de leer los datos, pero contiene menos detalle de su contenido.
- Decryption_key: Una clave para cifrar y descifrar el tráfico capturado.
- Encryption_key: Estándar de encriptación utilizando en el tráfico capturado.

```
>>> cap = pyshark.FileCapture('test.pcap', only_summaries=True)
>>>
>>> dir(cap[0])
```

Figura 2.14. Comando Pyshark

Al aplicar el comando de la figura 2.14 devuelve las siguientes informaciones: [14]

- no: Número de índice del paquete en la lista
- time: Tiempo absoluto entre el paquete actual y el primer paquete.
- Source: La dirección IP origen.
- destination: La dirección IP destino.
- protocol: El protocolo de capa más alta reconocido en el paquete.
- length: Longitud del paquete en bytes.
- summary_line: Todos los atributos de resumen en una cadena delimitada por tabulaciones.
- info: Un breve resumen de la capa de aplicación (si la hay).

En cambio, si no aplicamos only_summary, se pueden utilizar los siguientes comandos para obtener la información de la captura:

```
cap = pyshark.FileCapture('./capturas/imap.pcap')
cap.load_packets()
pkt = cap[2]

print(dir(pkt))
print(pkt.tcp.field_names)
```

El comando “dir” indica los elementos que hay debajo y con el comando “field_names” obtenemos todos los campos que contiene la cabecera del protocolo TCP.

2.5. Otras librerías utilizadas

2.5.1. DB Browser SQLite y SQLite3

Una base de datos es una aplicación software que permite almacenar gran número de datos de forma estructurada y accesible para su uso futuro. Las funcionalidades básicas son: consulta, búsqueda e inserción o actualización de nuevos datos.[16]

La base de datos puede almacenar diferentes tipos de datos, como caracteres, números, imágenes, fechas, monedas, texto, bit, decimal y varchar. Dependiendo de la aplicación software que se utilice puede variar la forma de uso. En este trabajo utilizamos DB Browser SQLite que posee una estructura relacional y nos permite guardar datos de tipo INTEGER, TEXT, BOOLEANO, REAL Y NUMERICOS.

Para crear la base de datos hemos utilizado la librería de SQLite3 de Python, que nos permite crear la tabla, leer datos de la tabla e insertar nuevos datos.

2.5.2. Pandas

La librería Pandas es una herramienta muy poderosa y flexible para la manipulación y tratamiento de datos. Se utiliza fundamentalmente para ámbitos como Data Science, Data Analyst y Machine Learning. [17]

Hay dos estructuras de datos dentro del paquete Pandas: Series, que es un array unidimensional capaz de almacenar cualquier tipo de dato; y DataFrame que tiene una estructura bidimensional con varias columnas, pudiéndose entender como la unión de varias estructuras de tipo Series.

Las principales funcionalidades que ofrece esta herramienta se muestran en la figura 2.13:

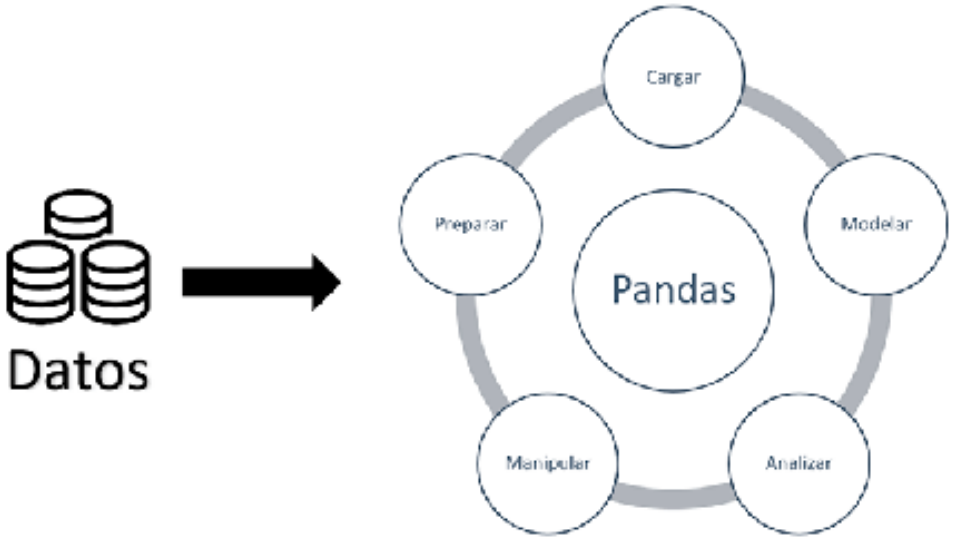


Figura 2.15. Funcionalidades de Pandas

2.5.3. Matplotlib

Matplotlib es una librería de Python que sirve para crear gráficas de calidad, animadas e iterativas a partir listas o arrays, permitiendo además personalizar las gráficas generadas. [18]

Matplotlib permite generar diferentes tipos de graficas según las necesidades. A continuación, se muestran los 4 tipos más conocidos:

- Gráficas básicas: Son gráficas de dos dimensiones con un eje X y un eje Y. Incluye todas las que se muestran en la figura 2.16. Normalmente, se utilizan para representar informaciones sencillas.

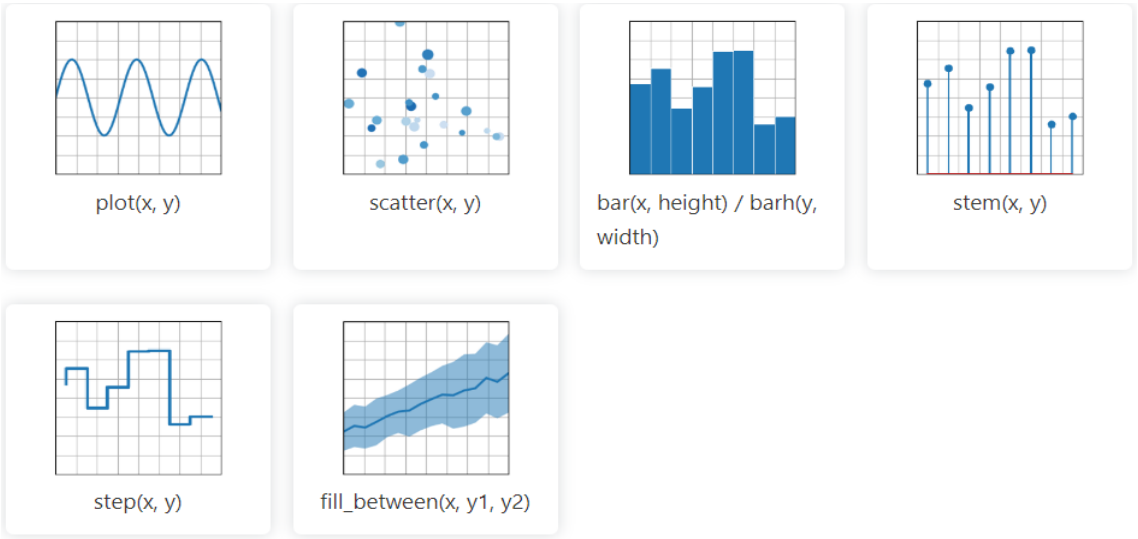


Figura 2.16. Gráficas básicas de Matplotlib

- Gráficas de matrices y campos: Gráficas generadas por matrices de datos o campos, normalmente utilizadas para representación de algún tipo de cálculo matemático como, por ejemplo, cálculo de probabilidades. Las más conocidas son las gráficas que se muestran en la figura 2.17.

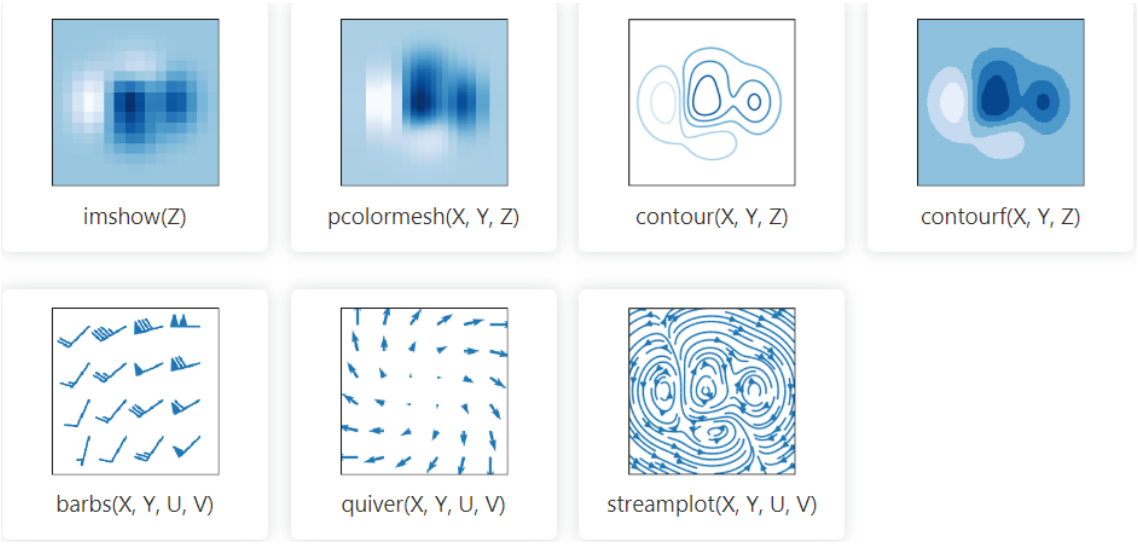


Figura 2.17. Gráficas de matrices y campos de Matplotlib

- Gráficas estáticas: se utilizan para el análisis estadístico y se muestran en la figura 2.18. Entre todas ellas, las más utilizadas son la gráfica de barras (hist(x)), utilizada normalmente para mostrar la evolución o comportamiento de una variable en el tiempo, o la gráfica de tarta o sectores para visualizar las partes de un todo a través de un círculo dividido en sectores.

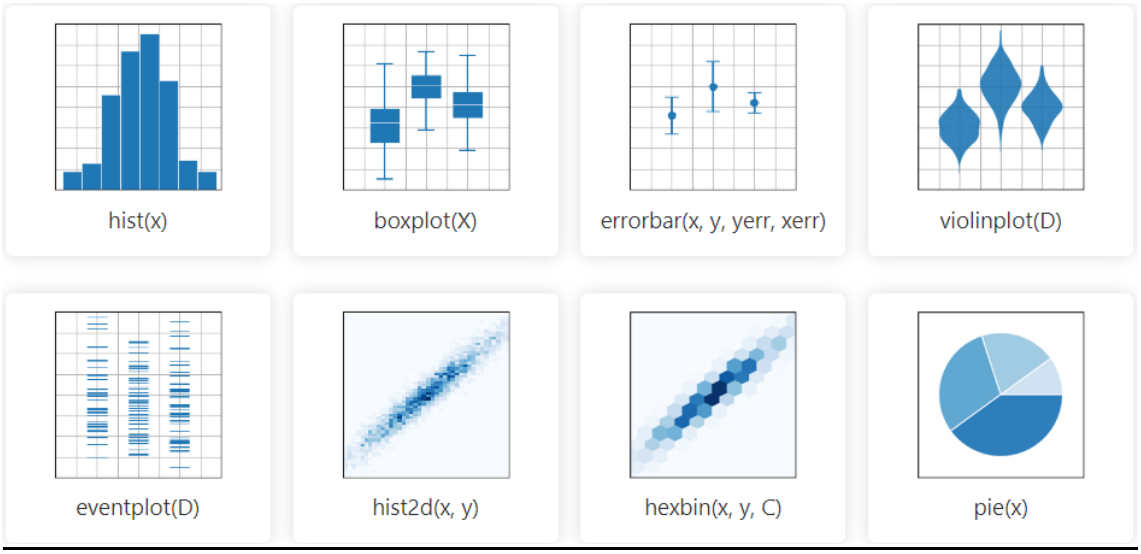


Figura 2.18. Gráficas estadísticas de Matplotlib

- Gráficas con coordenadas no estructuradas: representan datos de coordenadas X, Y, Z que podemos visualizar coordenados como un entorno. Además, este contorno se puede usar como un algoritmo de triangulación para rellenar los triángulos. En la figura 2.19 se muestran algunos ejemplos.

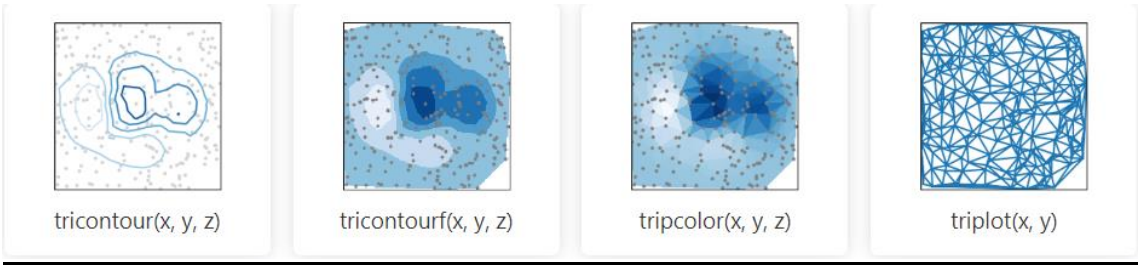


Figura 2.19. Gráficas con coordenadas no estructuradas de Matplotlib

2.4.4. Numpy

NumPy es una librería de Python para crear vectores y matrices multidimensionales. Además, contiene una gran colección de funciones matemáticas de alto nivel para realizar operaciones con los vectores y matrices. [19]

Junto con las operaciones matemáticas, también permite realizar operaciones lógicas, manipulación de formas, ordenación y selección, E/S, transformación discreta de Fourier, algebra lineal, operaciones estadísticas, simulación aleatoria y muchas más.

CAPÍTULO 3. DESARROLLO

En este capítulo se explica lo relativo a la parte de desarrollo del trabajo en Python, utilizando las herramientas mencionadas en el capítulo 2.

Para poder diferenciar bien las diferentes partes funcionales, el trabajo se dividirá en cuatro bloques/ventanas principales, como se ha comentado en el capítulo 1, además tenemos una ventana extra, que representa el mensaje de error. Cada una de estas ventanas tienen sus propios componentes y cada componente tiene una funcionalidad distinta. Todas estas funcionalidades son representadas por un botón, el cual, al pulsarlo, representa una información u otra, con el fin de visualizar la información de un fichero .pcap o .pcapng de Wireshark de forma más fácil.

3.1. Escenario

El objetivo del trabajo es leer los datos de un fichero generado por Wireshark o tcpdump, de tipo .pcap o .pcapng y extraer aquellas informaciones que deseamos obtener, representándolas en las cuatro interfaces gráficas que hemos generado. A continuación, se explicarán las ventanas según la funcionalidad:

- Ventana Sesión: El objetivo de esta ventana es recoger el nombre de la captura y el filtro que vayamos a utilizar, en el caso de que queramos aplicar un filtro a la captura.

Todas las capturas que vayamos a utilizar deben estar en la misma carpeta de capturas; en el caso de que no exista o no se encuentre la captura en la carpeta indicada saltará un aviso de error. Una vez obtenido el nombre de la captura y el filtro, se guarda en un fichero JSON junto con los datos que se han extraído de la captura.

- Ventana Main: El objetivo de esta ventana es representar todos los datos que hemos guardado en el fichero JSON. Por lo tanto, en esta ventana se encuentran las funcionalidades principales del proyecto. Existirán varios botones que nos permitirán representar los diferentes tipos de datos. Además, desde aquí podemos abrir la ventana gráfica y la ventana gestión.

- Ventana Gráfica: El objetivo de esta ventana es mostrar los datos que hemos obtenido de la captura en forma de gráficas. Aquí tenemos cuatro botones que nos permiten

representar cuatro tipos de gráficas diferentes. La primera representa el tiempo de envío delta, que es el retardo entre dos tramas consecutivas; la segunda gráfica representa el tiempo relativo, que toma la referencia de la primera trama como tiempo de inicio. Para estas dos gráficas se puede seleccionar el intervalo de tramas del fichero que se desee generar.

La tercera gráfica tiene un formato sectorial y representa la proporción del tamaño de cada trama, separado en varios rangos de bytes.

La última gráfica es un diagrama de barras, que representa la proporción de los protocolos que aparecen en la captura.

- Ventana Gestión: El objetivo de esta ventana es extraer los datos del fichero JSON o extraer datos de una nueva captura, introduciendo el nombre de la captura, el filtro que deseamos utilizar. Además, es posible leer la captura seleccionada por rango de trama y el rango de tiempo relativo y representar estos datos en una tabla. Posteriormente nos permitirá también exportar estos datos en formato Excel o bien, eliminarlos de la tabla.

Además, las ventanas están interconectadas entre ellas. La figura 3.1 muestra la relación entre estas ventanas y el proceso que se debe realizar para pasar a otra ventana.

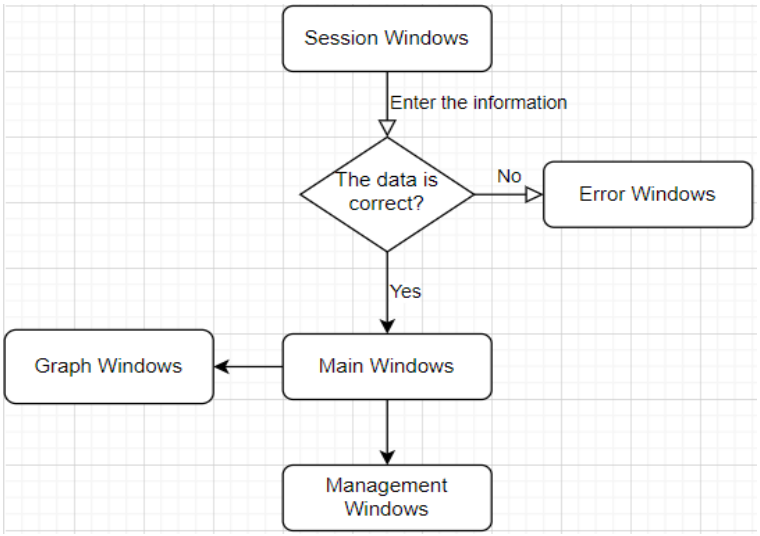


Figura 3.1. Diagrama de proceso

Desde la ventana de sesión recogemos la información de la captura. En el caso de que no exista la captura o se produzca otro tipo de error a la hora de introducir la información de la captura, se abrirá una ventana de aviso de error, indicando el tipo de problema que haya surgido. En el caso contrario, se abrirá la ventana Main, cerrando automáticamente la ventana sesión. Desde esta ventana se puede pasar a la ventana gráfica o a la ventana gestión.

3.2. Diseño de la interfaz gráfica[23] [27]

En esta parte se explica el funcionamiento de las ventanas diseñadas utilizando la herramienta Qt Designer paso a paso. Se muestra el funcionamiento de los objetos y elementos que aparece en cada ventana diseñada y cuáles son sus funcionalidades.

3.2.1. Ventana de sesión

La ventana de sesión está basada en una ventana principal (QmainWindow) con un tamaño de 500x200 pixeles, siempre ubicada en el centro de la ventana del equipo. Dentro de la ventana principal hay un Widget con una ordenación vertical y tiene el mismo tamaño que la ventana principal, con una política de tamaño preferido, por lo que el tamaño de este Widget depende de la ventana principal.

Dentro de este Widget hay varios elementos como muestra la figura 3.2, todos estos están incluidos en dos Qframe que se explican a continuación.

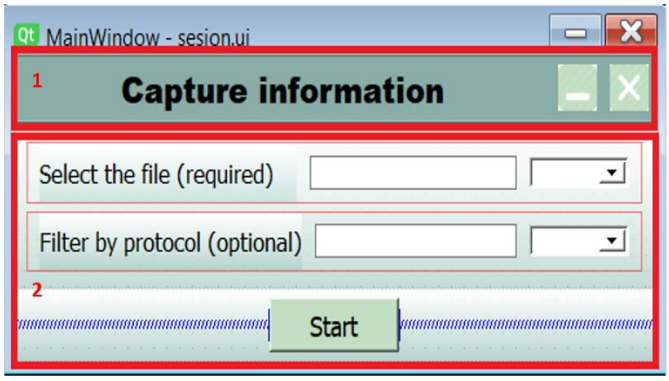


Figura 3.2. Ventana sesión diseñada por la herramienta Qt Designer (1)

3. **Header:** Es un Qframe con una ordenación horizontal, con una altura de 45 pixeles y ancho sin especificar. Dentro del marco hay un Qlabel, donde se muestra el título de la ventana (“Capture Information”). Tiene una política de tamaño preferido, esto indica que puede encogerse o crecer según la ventana principal.

También hay dos botones para controlar la ventana principal (QpushButton), uno que se encarga de cerrar y el otro de minimizar. Tienen un tamaño de 30x30 pixeles e incluyen un icono en el interior del botón.

Todos estos objetos que se encuentran dentro de este marco tienen una separación de 10 pixeles entre ellos y una separación de 5 pixeles con el borde de marco. Además, define una hoja de estilos, donde especifica el color del marco, el color de los botones y la fuente de la letra de Qlabel.

2. Body: Qframe con una ordenación vertical y una política de tamaño preferido. Asigna LayoutStretch de (11,1) para los marcos que se encuentran dentro de este Qframe.

El Qframe superior (Figura 3.3) ocupa 11 del marco "Body" y tiene una política de tamaño preferido. Dentro hay dos QlineEdit donde introducimos el nombre de la captura y el filtro, dos Qlabel que indican qué tipo de formato de fichero debemos introducir en los dos QlineEdit. También hay dos QcomboBox, una indica el formato de la captura y la otra muestra una lista de los protocolos más utilizados. Todos estos objetos tienen una política de tamaño preferido, con una separación de 10 pixeles entre ellos y una separación de 5 pixeles con el borde del marco.

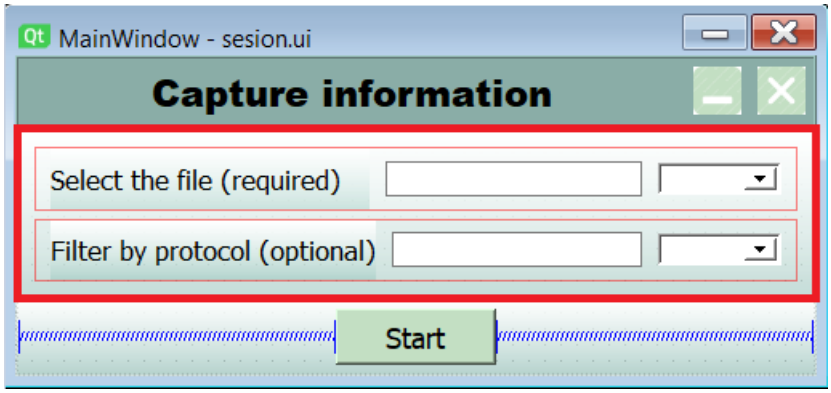


Figura 3.3. Ventana sesión diseñada por la herramienta Qt Designer (2)

El marco de la parte inferior, lo que muestra la figura 3.4, contiene un QpushButton de tipo funcional, con un tamaño de 100x35 pixeles, al pulsar este botón recoge la información de los dos QlineEdit y del QcomboBox. Con estas informaciones es capaz de leer una captura, extraer los datos que necesitamos y guardarlo en un fichero JSON. También se encarga de cerra la ventana de sesión y abrir la ventana Main. En este marco también hay dos espaciadores horizontales, para que el botón se sitúe siempre en el centro del marco.

Además, define una hoja de estilo en este marco "Body", que especifica el color de fondo de los marcos, la fuente de las letras de Qlabel, así como el color de los botones de QlineEdit y de QcomboBox.

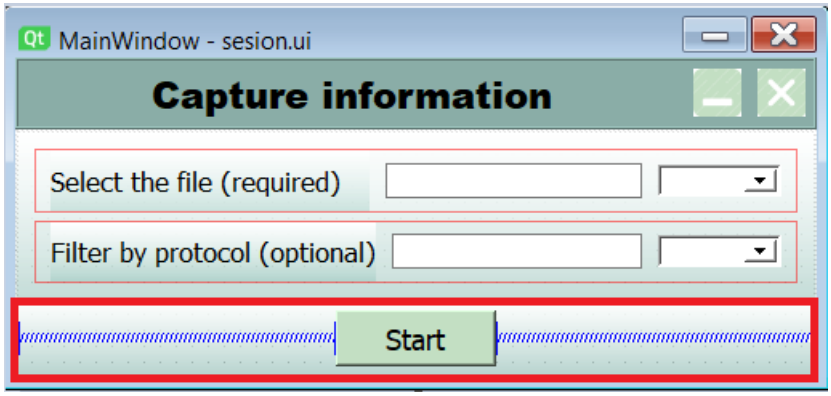


Figura 3.4. Ventana sesión diseñada por la herramienta Qt Designer (3)

3.2.2. Ventana Main

La ventana Main está basada en una ventana principal, con un tamaño inicial de 1151x699 pixeles, una política de tamaño preferido (esto indica que el tamaño de la ventana podría cambiarse) y siempre colocada en el centro de la ventana del equipo.

Dentro de esta ventana principal hay un Widget con una ordenación cuadriculada, tiene el mismo tamaño que la ventana principal y con una política de tamaño preferido, por lo que cambiará su tamaño según la ventana principal.

El Widget a la vez está dividido en dos Qframe, como muestra la figura 3.6.

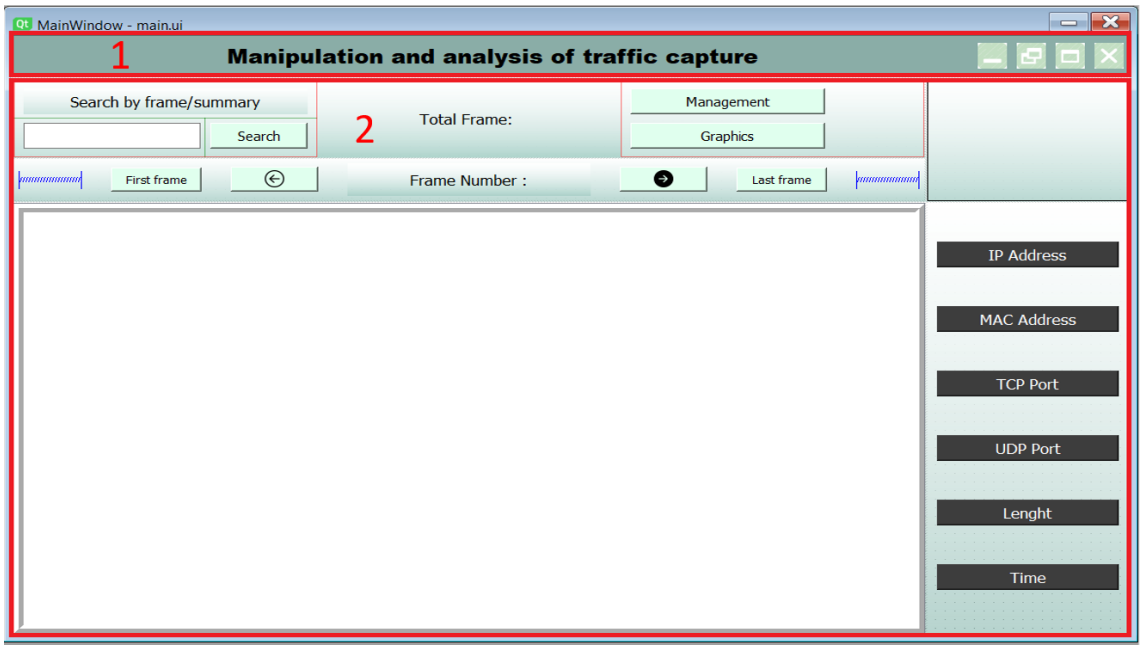


Figura 3.5. Ventana Main diseñada por la herramienta Qt Designer (1)

3. **Header**: Es un Qframe con una ordenación horizontal, con una altura de 50 pixeles y el ancho no está especificado, por tanto, la altura de este marco nunca cambiará, pero el ancho del marco puede cambiarse según el tamaño de la ventana principal.

Dentro de este marco hay un Qlabel, que muestra el título de la ventana("Manipulation and análisis of traffic capture"), con una política de tamaño preferido. Además, hay cuatro QpushButton, estos botones son de tipo control y se encargan de minimizar la ventana, maximizar o minimizar el tamaño de la ventana y cerrar la ventana. Los botones tienen un tamaño de 30x30 pixeles y una imagen en el interior del botón.

Todos los objetos de este marco tienen una separación de 10 pixeles entre ellos y una separación de 5 pixeles con el borde del marco. Además, incluye una hoja de estilo, que especifica el color de fondo del marco, la fuente de texto y el color de los botones.

2. Body: Qframe con una ordenación cuadriculada y con una política de tamaño preferido. También hay una hoja de estilo que especifica el color de marco, la fuente de letra y el color de los botones.

Dentro de este marco hay cuatro Qframe. El primero es el que se muestra en el icono rojo de la figura 3.7. Tiene una anchura de 260 pixeles y no especifica la altura, con una política de tamaño preferido, por lo que la altura cambiará según el tamaño de la ventana principal. Dentro de este marco hay un Qlabel que muestra las direcciones IP, las direcciones MAC, los puertos TCP/UDP, la longitud de las tramas y el tiempo delta.

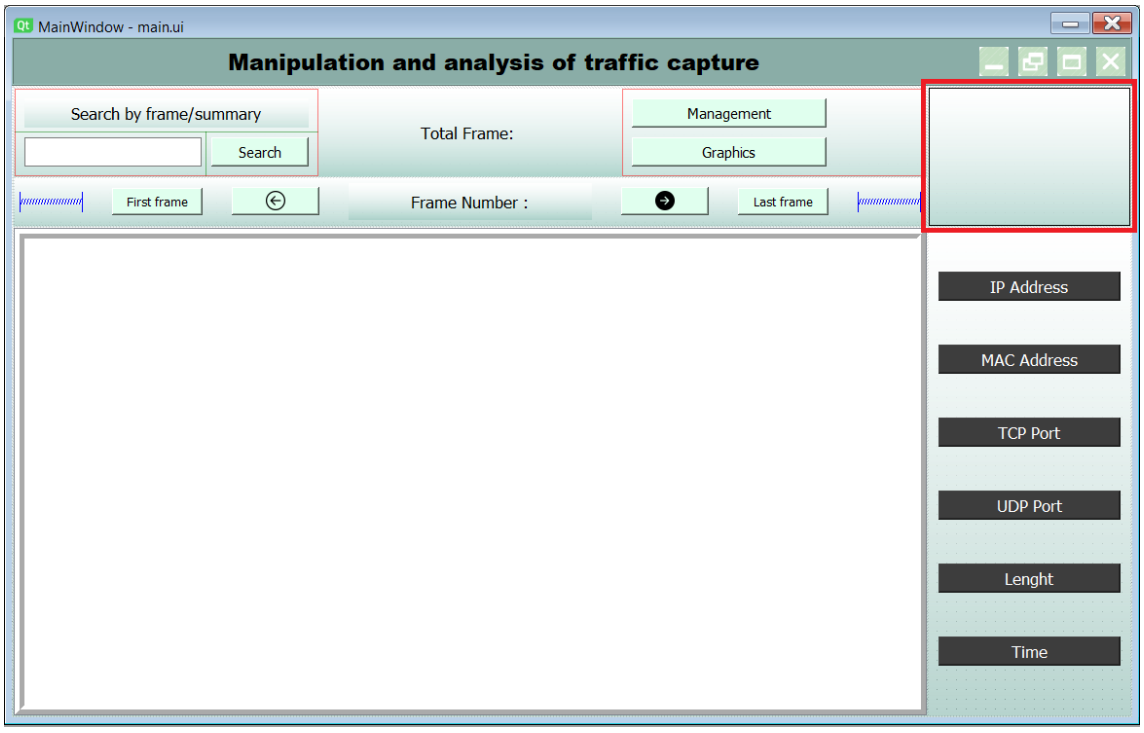


Figura 3.6. Ventana Main diseñada por la herramienta Qt Designer (2)

El segundo marco (Figura 3.7) tiene una ordenación vertical, con una anchura de 260 pixeles y no está especificada la altura del marco, pero tiene una política de tamaño preferido.

Dentro hay seis QpushButton de tipo funcional, cada uno tiene una funcionalidad distinta. Estos botones se muestran en el marco de la figura 3.7. Todos estos botones tienen una política de tamaño preferido, con una separación de 10 pixeles entre ellos y 10 pixeles con el borde del marco.

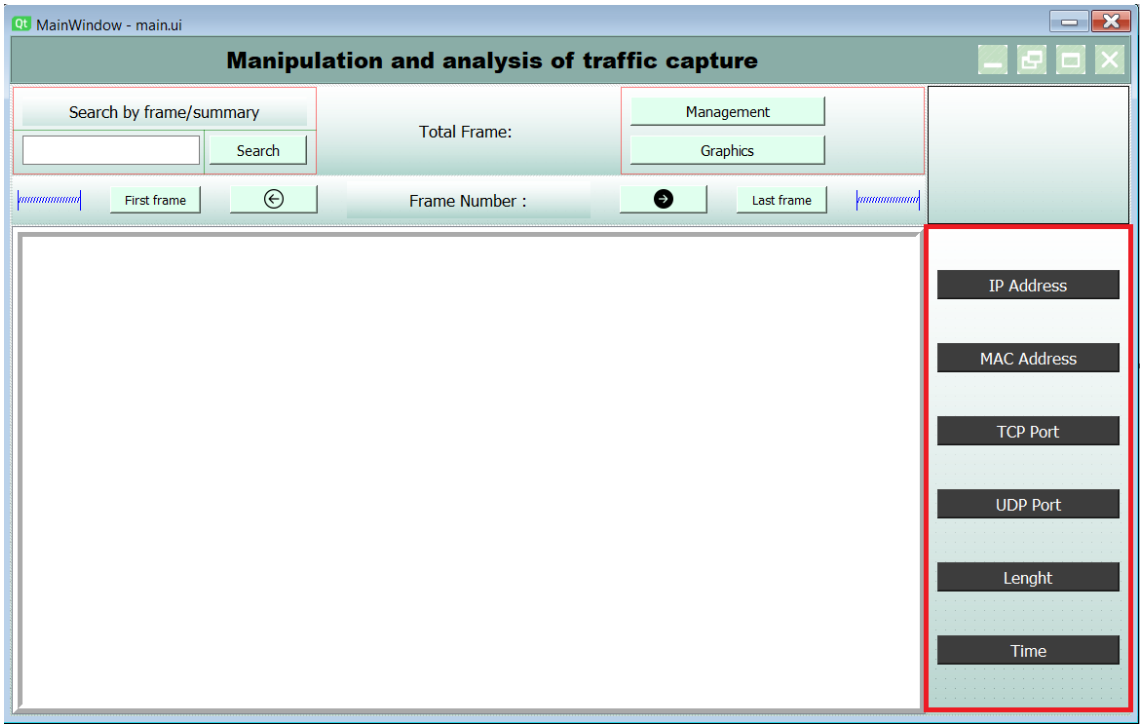


Figura 3.7. Ventana Main diseñada por la herramienta Qt Designer (3)

El tercer Qframe es lo que muestra la figura 3.8, tiene una ordenación vertical, con una política de tamaño preferido, por lo que el tamaño del marco cambiará según el tamaño de la ventana principal. Dentro hay un elemento QscrollArea, con una política de tamaño expansivo. En su interior hay un Qlabel, donde se muestra la información de una trama determinada de la captura o se muestra un resumen de la captura. QscrollArea permite desplazar el contenido que se encuentra en el Qlabel utilizando la barra desplazadora.

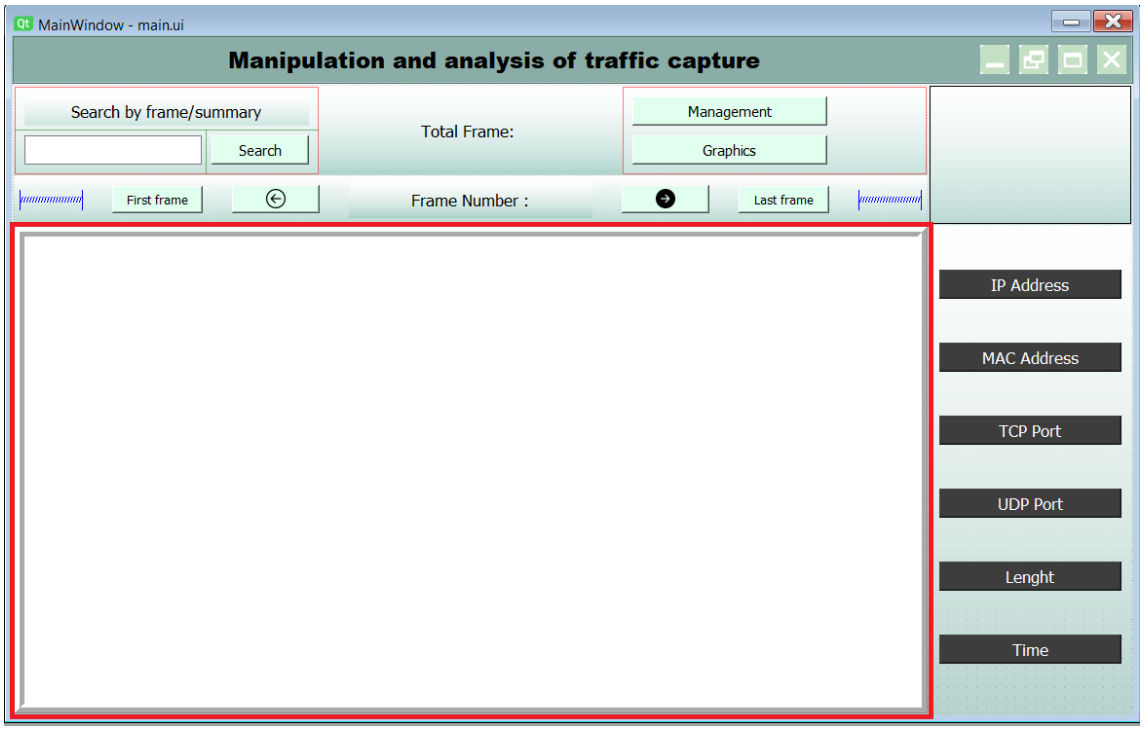


Figura 3.8. Ventana Main diseñada por la herramienta Qt Designer (4)

El ultimo marco (figura 3.9) tiene una ordenación formularia, con una política de tamaño vertical de tipo fixed. Dentro hay otros dos Qframe. El Qframe de la parte inferior contiene cuatro botones de tipo funcional, que se encargan de mover la posición de las tramas de la captura, tienen una anchura de 90 pixeles y una política de tamaño vertical Fixed. En el centro de este marco se encuentra un Qlabel, que indica el número de trama que se está visualizando, con una anchura de 250 pixeles y una política de tamaño vertical expansiva. Además, hay dos espaciadores horizontales para separar los botones con el borde del marco. Todos estos objetos tienen una separación de 30 pixeles entre ellos y una separación de 5 pixeles con el borde.

En el marco superior hay diferentes objetos. Empezando por la parte derecha hay dos botones de control, que nos permiten abrir la ventana de gestión y la ventana gráfica. Estos dos botones tienen un tamaño de 200x30 pixeles. A la izquierda de estos botones hay un Qlabel, con una anchura mínima de 300 pixeles y una política de tamaño vertical preferido. Este label muestra las tramas totales de la captura. Por último, en la parte izquierda del marco hay otro Qlabel que muestra lo que se debe introducir en el QlineEdit, así como un botón de búsqueda, que se encarga de recoger la información de QlineEdit. Con esta información es capaz de leer los datos de la captura y mostrarlo en el marco de la figura 3.9. Todos estos elementos del marco tienen una política de tamaño preferido y con una separación de 10 pixeles entre ellos y 5 pixeles con el borde.

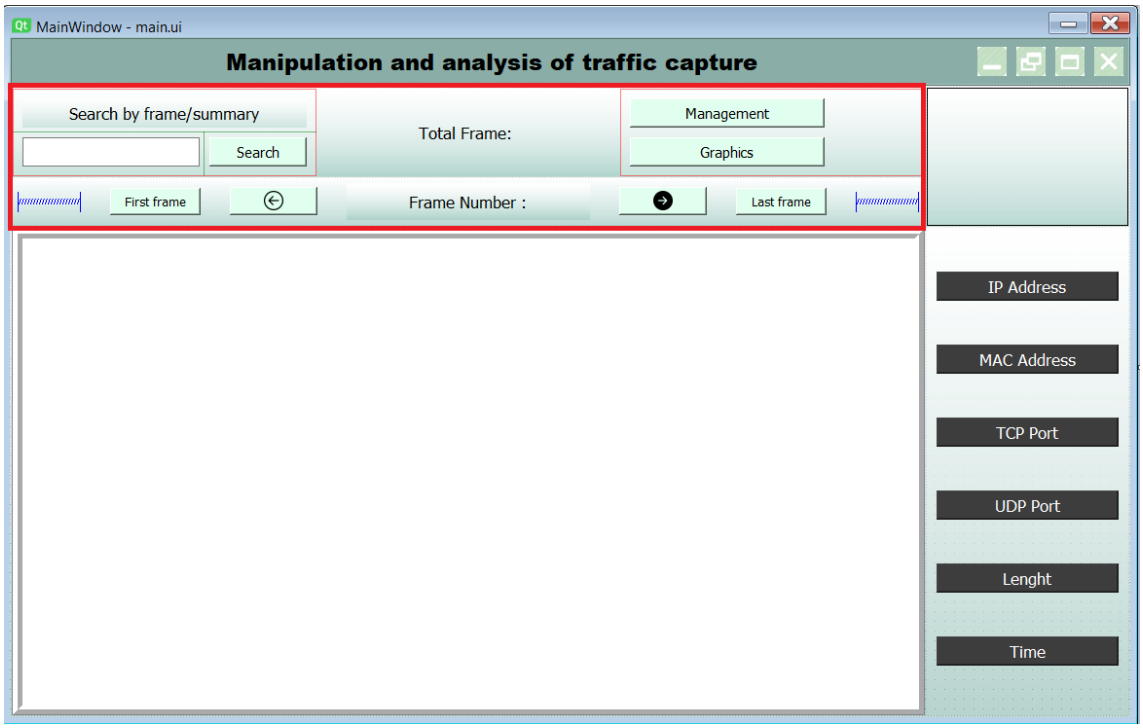


Figura 3.9. Ventana Main diseñada por la herramienta Qt Designer (5)

3.2.3. Ventana de gráficos

La ventana grafica está basada en una ventana principal, con un tamaño inicial de 1000x614 pixeles y una política de tamaño preferido. Dentro hay un Widget con el mismo tamaño que la ventana principal, este widget a la vez está dividido en dos partes/Qframe como se muestra en la figura 3.10.

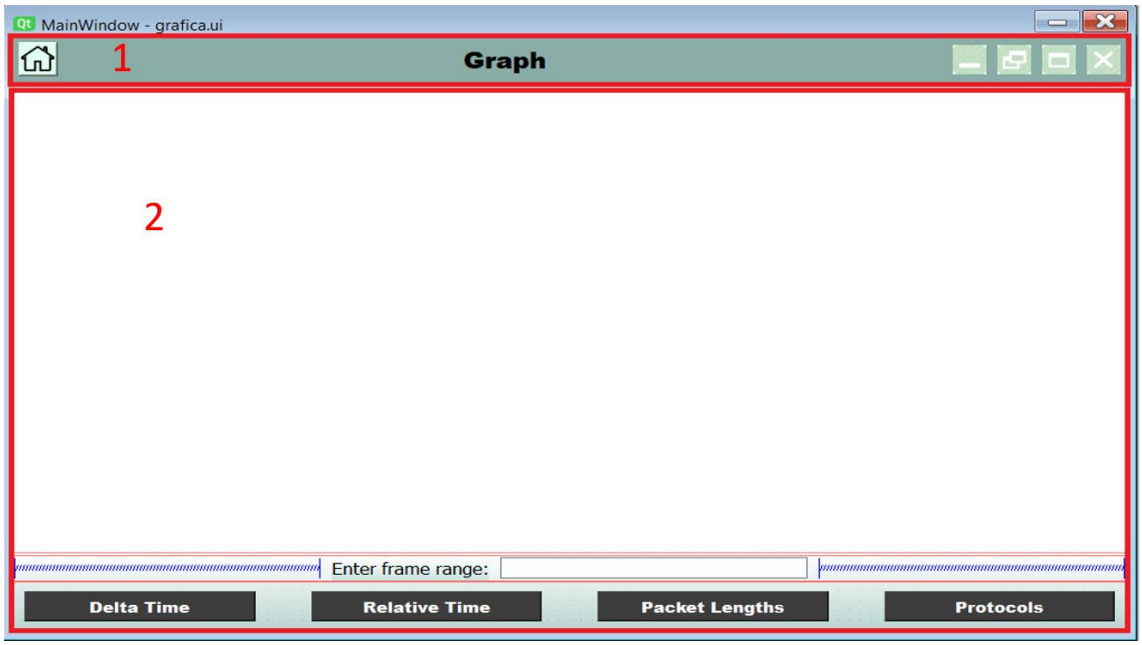


Figura 3.10. Ventana Gráfica diseñada por la herramienta Qt Designer (1)

3. **Header:** Qframe con una ordenación horizontal y con una altura de 50 pixeles, no se especifica la anchura del marco. Dentro hay cuatro botones de control con un tamaño de 30x30 pixeles, que se encargan de minimizar y cerrar la ventana, y de maximizar y minimizar el tamaño de la ventana. Además de los botones de control, hay otro botón con un tamaño de 35x35 pixeles, pulsando este botón retrocedemos a la ventana Main. También hay un Qlabel con una política de tamaño preferido, donde se muestra el título de la ventana ("Graph"). Todos estos objetos tienen una separación de 10 pixeles entre ellos y 5 pixeles con el borde del marco.

En este marco se define una hoja de estilo que especifica el color de los botones, la fuente de letra utilizado y el color de fondo del marco.

2. Body: Qframe con una ordenación vertical y con una política de tamaño preferido. Además, hay una hoja de estilo donde se especifica el color de los botones, el color de marco y la fuente de letra.

Dentro hay otros dos marcos, el marco superior lo que muestra en la figura 3.13, tiene un tamaño inicial de 998x482 pixeles, con una ordenación vertical y con una política de tamaño preferido, por lo que este marco cambiará su tamaño según la ventana principal. Dentro del marco hay un Layout vertical, donde muestra las gráficas que se ha generado por Python.

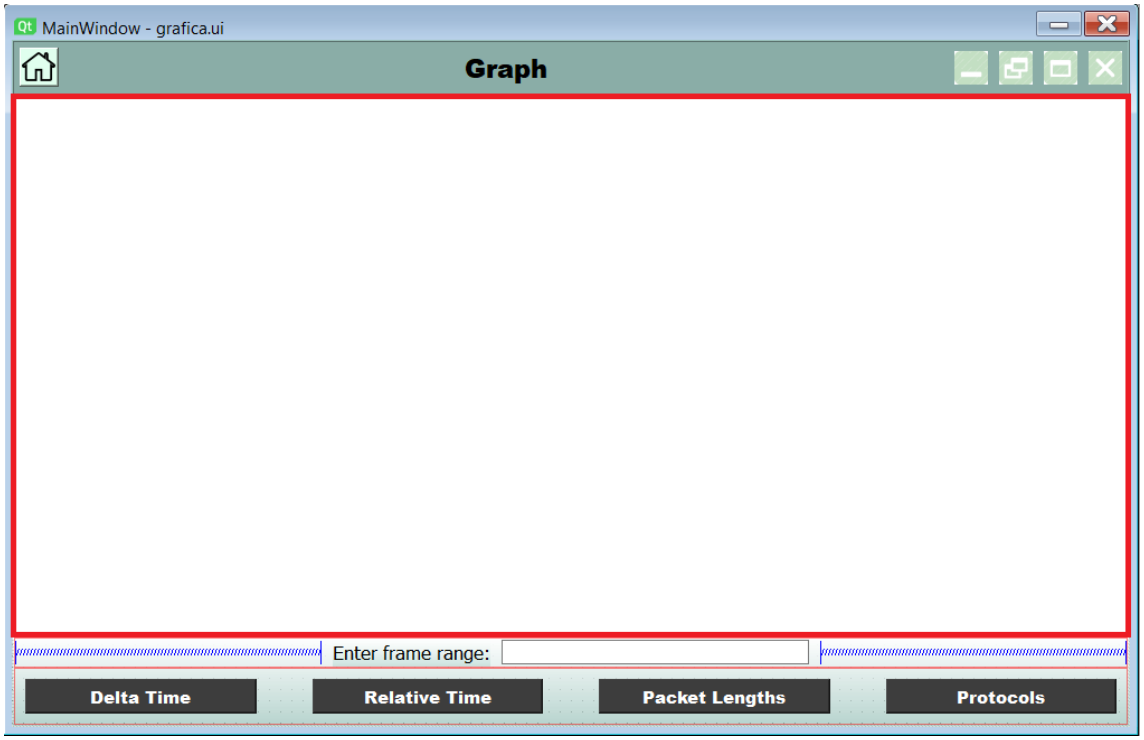


Figura 3.11. Ventana Gráfica diseñada por la herramienta Qt Designer (2)

El marco de la parte inferior (figura 3.12) tiene una ordenación horizontal, con una altura de 80 pixeles. Este marco está separado en dos partes, la parte inferior que es un QHBoxLayout con cuatro botones (QpushButton) que muestra gráficas relacionadas con el tiempo delta, el tiempo relativo, el número de bytes de cada trama y los protocolos que aparecen en la captura. Esta parte tienen una ordenación horizontal, con una política de tamaño preferido y los botones tienen una separación de 50 pixeles entre los botones y 10 pixeles con el borde.

La parte superior también es un QHBoxLayout , tiene una ordenación horizontal y dentro de éste hay un Qlabel, un QlineEdit y dos espaciadores horizontales. El Qlabel tiene un tamaño inicial de 141x25 pixeles, indica lo que se debe introducir en QlineEdit. El QlineEdit tiene un tamaño inicial de 275x22 pixeles con una política de tamaño horizontal expansivo. Los dos espaciadores horizontales sirven como elementos centralizadores, para que los dos objetos se sitúen siempre en el centro.

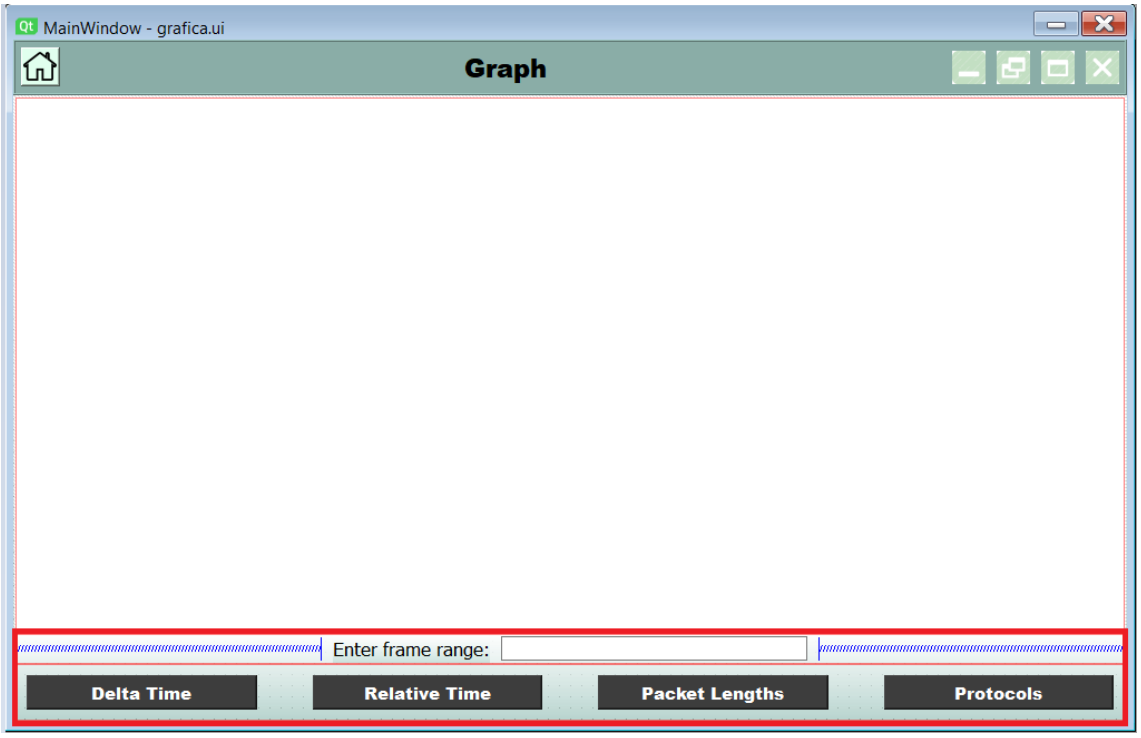


Figura 3.12. Ventana Gráfica diseñada por la herramienta Qt Designer (3)

3.2.4. Ventana gestión

Igual que otras ventanas, la de gestión utiliza una ventana principal como base (QmainWindow), tiene un tamaño inicial de 1169 x 673 pixeles y una política de tamaño preferido. Dentro hay un Widget con una ordenación cuadriculada y tiene el mismo tamaño que la ventana principal. Dentro de este Widget hay dos Qframe, como muestra la figura 3.13.

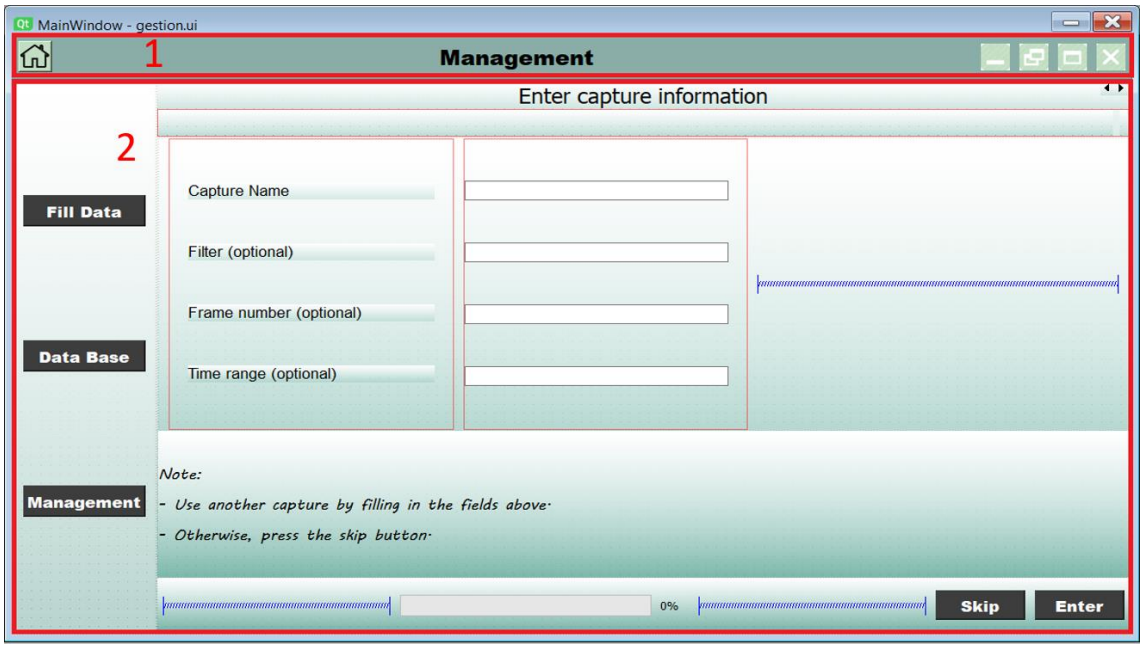


Figura 3.13. Ventana Gestión diseñada por la herramienta Qt Designer (1)

1. Header: Este marco tiene ordenación vertical, una altura de 50 pixeles y no se especifica la anchura del marco, lo que indica que el ancho del marco cambiará según el tamaño de la ventana principal.

Dentro de este marco hay cuatro botones de control (QpushButton) con un tamaño de 30x30 pixeles, que se encargan de minimizar y cerrar la ventana, y de maximizar y minimizar su tamaño. Además de los botones de control, hay otro de 30x30 pixeles. Pulsando este botón se abrirá la ventana Main. También hay un Qlabel dentro del marco que muestra el título de la ventana ("Management"), con una política de tamaño preferido. Todos estos objetos tienen una separación de 10 pixeles entre ellos y 5 pixeles con el borde del marco.

En este marco "Header" define una hoja de estilo, que especifica el color de los botones, la fuente de letra utilizada y el color de fondo del marco.

2. Body: Qframe con una ordenación horizontal y una política de tamaño preferido, en este marco se define una hoja de estilo que especifica el color de fondo del widget y del marco, el color de los botones y la fuente de letra utilizada.

El marco está dividido en dos partes/marcos. La parte izquierda que se muestra en la figura 3.14. Este marco tiene una ordenación vertical, con una política de tamaño preferido. Dentro hay tres botones que se encargan de cambiar la página de StackedWidget, el marco de la parte derecha (figura 3.15). Los tres botones tienen un tamaño mínimo de 120x35 pixeles y una política de tamaño vertical Fixed, con una separación de 10 pixeles entre ellos y con el borde del marco.

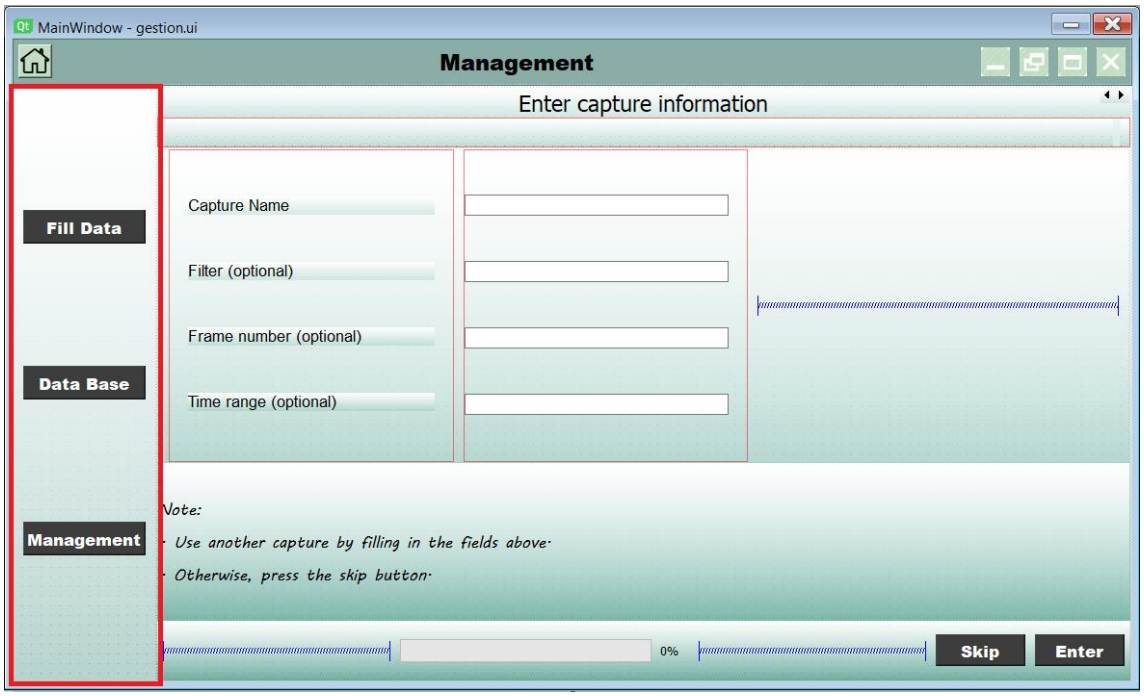


Figura 3.14. Ventana Gestión diseñada por la herramienta Qt Designer (2)

El marco de la parte derecha tiene una ordenación vertical, en el que se ubica un StackedWidget, que consiste en tres Widget. La primera es lo que muestra la figura 3.15. Dentro hay tres marcos: el marco de la parte superior del Widget tiene una ordenación vertical y una política de tamaño preferido. Dentro hay dos Qlabel, uno que muestra el título de este Widget y la otra muestra el estado de captura. Mientras realiza la operación de escribir los datos de la captura a la base de datos muestra un mensaje de "*Processing...*". Tras finalizar la operación, si el resultado de escribir es correcto, se muestra un mensaje de "*Successful*". En caso de que no pueda realizar esa operación de escritura lo notifica mediante un mensaje de error. Ambos Qlabel tienen una política de tamaño preferido.

El marco de centro tiene una ordenación vertical y dentro de éste hay otro marco y un Qlabel. Este marco tiene una ordenación horizontal e incluye cuatro Qlabel y cuatro QlineEdit, todos estos objetos tienen una política de tamaño vertical Fixed y una separación de 20 pixeles entre ellos. En el Qlabel se muestra una nota sobre el funcionamiento de los dos botones que se muestran en el marco inferior, en el que se encuentran dos espaciadoras horizontales de tipo expansivo, para que los dos botones se sitúen en la parte derecha del marco y para que la barra de progreso no se situé en la parte izquierda del marco. La barra de progreso tiene una anchura mínima de 300 pixeles y una política de tamaño Fixed, el valor inicial de la barra de progreso es de 0% y está relacionada con la operación de escribir los datos de la captura a la base de datos. Si llega a 100% significa que la operación de la escritura ha sido completada. Todos los objetos que se encuentran en este marco inferior tienen una separación de 10 pixeles entre ellos y una separación de 5 pixeles con el borde del marco.

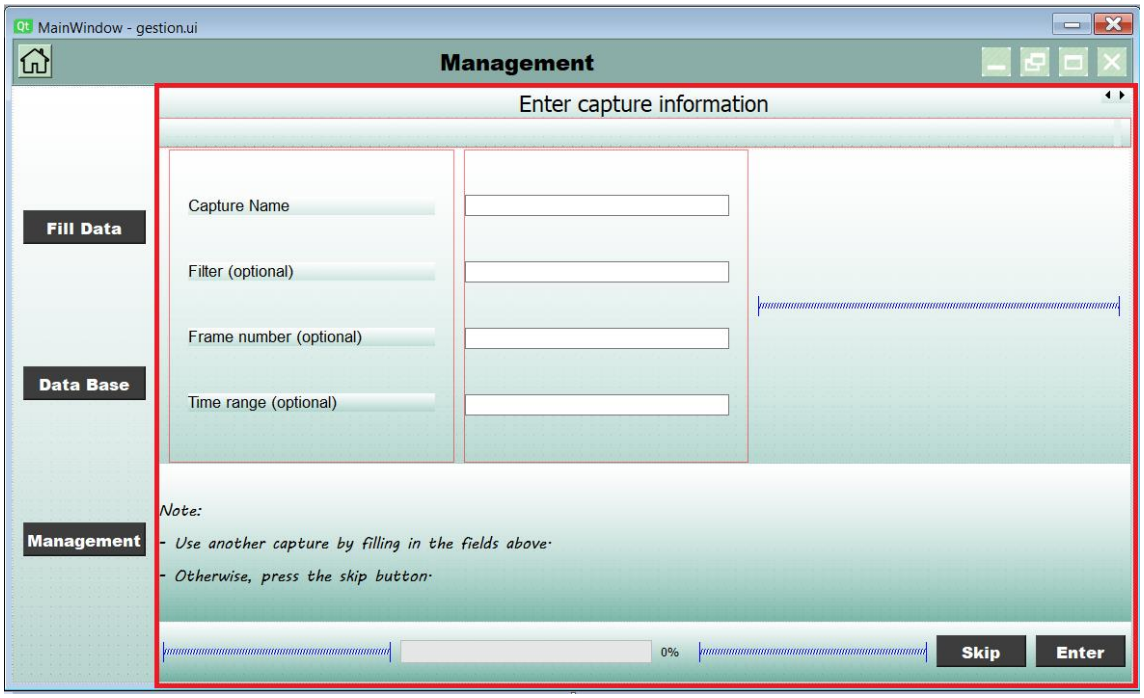


Figura 3.15. Ventana Gestión diseñado por la herramienta Qt Designer (3)

La segunda página (figura 3.16) tiene una ordenación vertical, con una política de tamaño preferido. Incluye un Qlabel donde se muestra el título de este Widget, una tabla que muestra los datos de la captura (QtableWidget) y un botón de tipo funcional. [21] Al pulsar este botón, se lee el contenido de la base de datos y se muestra en la tabla de esta página. El botón y el Qlabel tienen una política de tamaño preferido y la tabla una política de tamaño expansivo.

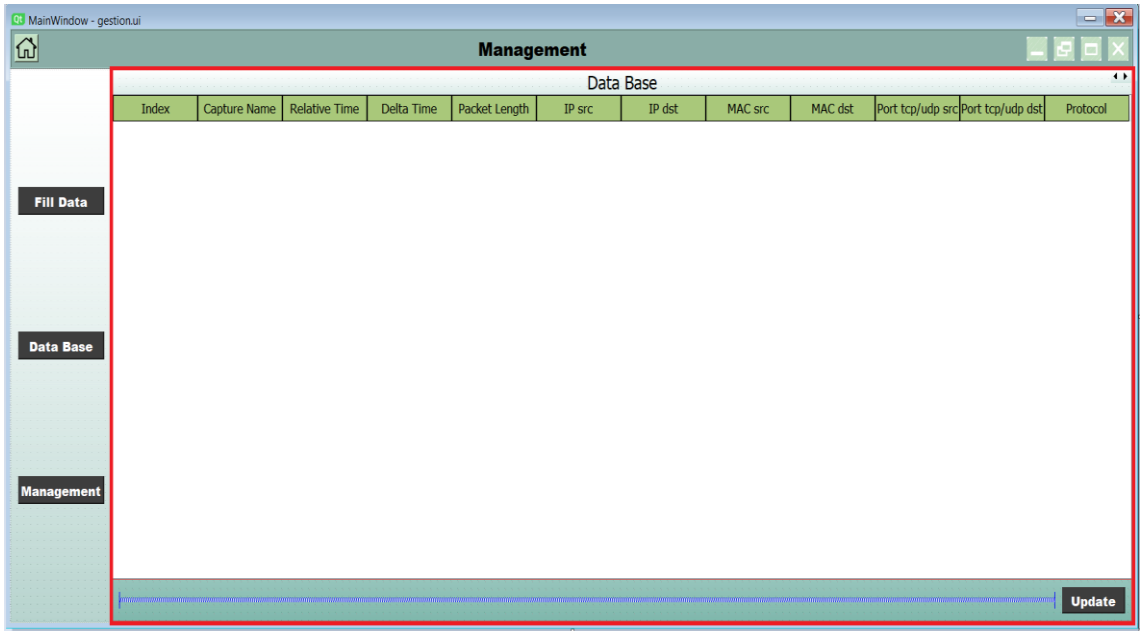


Figura 3.16. Ventana Gestión diseñado por la herramienta Qt Designer (4)

La última página, que se muestra en la figura 3.17, está formada por dos marcos y una tabla, con una ordenación horizontal. El marco de la parte inferior incluye dos botones funcionales y un Qlabel que muestra si el resultado de los botones fue correcto. La funcionalidad del botón “Export” es exportar los datos que se muestran en la tabla de esta página; y el botón “Delete” se encarga de eliminar los datos que se muestran en la tabla desde la base de datos. Ambos tienen una política de tamaño horizontal Fixed.

La tabla que se encuentra en esta página tiene una política de tamaño expansivo. En esta tabla se muestra el resultado de la búsqueda.

El marco superior tiene una ordenación vertical, con una política de tamaño preferido. Dentro de este marco hay dos Qlabel, un QlineEdit, un QcomboBox y un botón de búsqueda. Todos estos objetos tienen una política de tamaño preferido salvo QlineEdit que tiene una política de tamaño horizontal mínima de 200 pixeles y QcomboBox que tiene un tamaño mínimo horizontal de 130 pixeles. Los objetos que se encuentra en este marco tienen una separación de 10 pixeles entre ellos y 5 pixeles con el borde del marco.

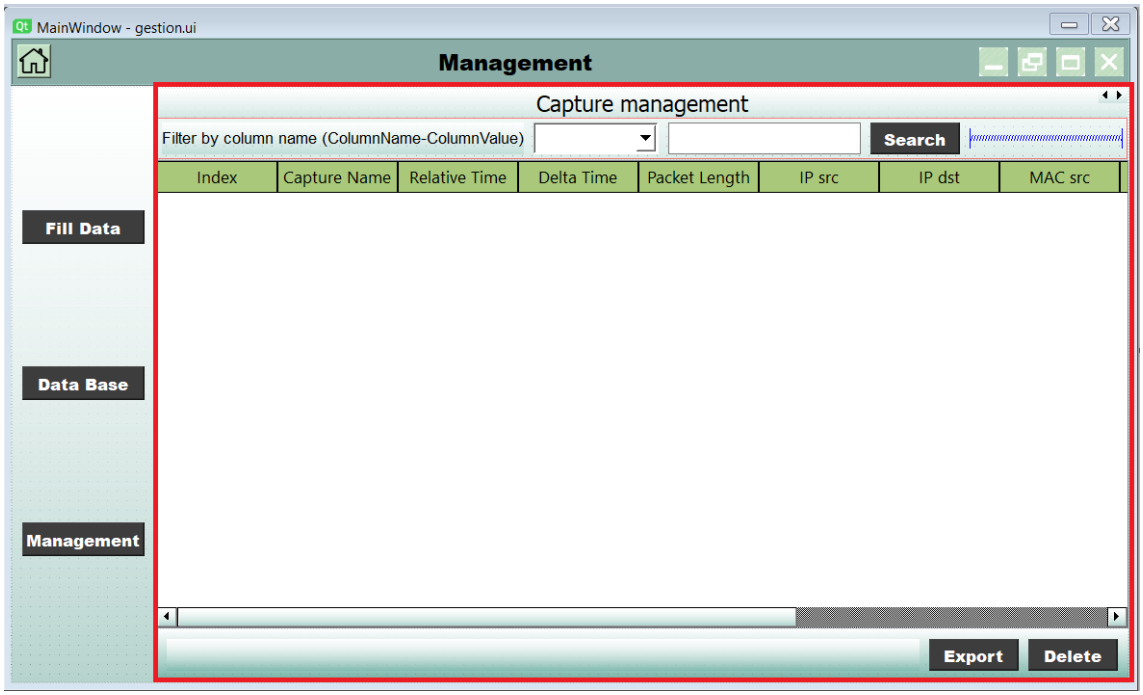


Figura 3.17. Ventana Gestión diseñado por la herramienta Qt Designer (5)

3.2.5. Ventana de error

Además de las cuatro ventanas, también hay una ventana de Error, en la que se muestran mensajes cuando algo no sale bien (figura 3.18). Está basada en una ventana principal y dentro tiene incluido un Widget. Posee un tamaño de ventana fijo de 390x120 pixeles. Dentro de este Widget hay dos marcos: el marco superior contiene un Qlabel, en el que se muestra el título de la ventana, y en el marco inferior hay un Qlabel, donde

se muestra el mensaje de error, una imagen y un botón para cerrar la ventana. Todos estos elementos tienen una política de tamaño preferido.

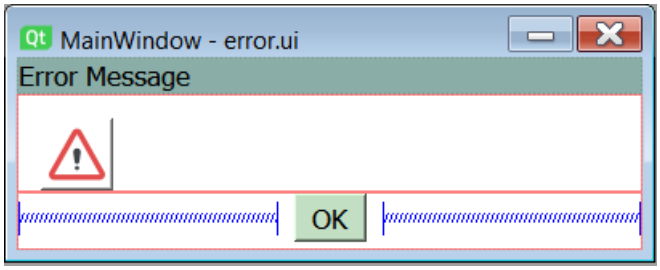


Figura 3.18. Ventana Error diseñado por la herramienta Qt Designer

3.3. Implementación de la parte funcional

En este apartado se tratan los aspectos funcionales del proyecto, al igual que las interfaces gráficas. Se divide en cuatro partes:

- La primera parte corresponde con la ventana Sesión. En el apartado 3.2 hemos visto que la funcionalidad principal de la ventana es leer datos de una captura y guardarlos en un fichero JSON. Por lo tanto, la forma de leer y guardar los datos es algo importante. En nuestro caso, hemos utilizado la librería Pyshark de Python para leer los datos y la librería JSON para guardarlos. En la figura 3.19 se muestra la ventana Sesión generada por Python:

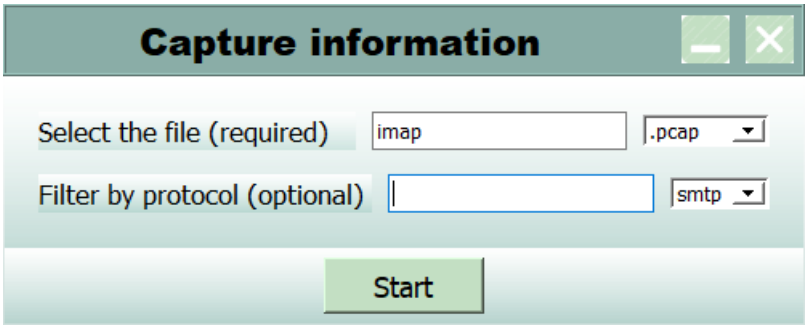


Figura 3.19. Ventana Sesión generada por Python

El primer paso consiste en recoger el contenido de las dos líneas de texto de la ventana Sesión utilizando el comando text() de Qt. El nombre de la captura es un campo obligatorio, si está vacío abrirá una ventana de Error indicando que el campo nombre está vacío y el campo filtro es opcional. Cuando introducimos el nombre de la captura hay que tener en cuenta el formato, si es de tipo .pcap o pcapng y aplicar un filtro válido, para ello debemos conocer qué tipo de tráfico estamos gestionando.

El tipo de filtro lo podemos introducir en el campo de texto del filtro o seleccionando los valores que hay en el cuadro de lista a su derecha (QComboBox), donde se muestran los protocolos más utilizados.

Tras obtener el nombre de la captura y el filtro, se ejecuta el comando que hemos visto en el apartado 2.3.2. Pasándole el nombre de la captura y el filtro podemos leer datos de la misma y guardarlos temporalmente en una variable denominada “cap”, tal y como se muestra en la figura 3.20.

```
def capReturn(self):
    cap = pyshark.FileCapture(input_file=f"../capturas/{self.dict_file['filename']}", display_filter=self.dict_file['filter'])
    cap.load_packets()
    return cap
```

Figura 3.20. Comando Pyshark, Filecapture

La salida de la función “capReturn” devuelve toda la información relativa a la captura y utilizando otros comandos de la librería Pyshark nos permite extraer la información específica que necesitamos. Como resultado de estos comandos tenemos la información relacionada con las direcciones IP origen y destino, direcciones MAC origen y destino, puertos TCP o UDP origen y destino, tiempos delta y relativo, el tamaño de las tramas y el protocolo de la capa más alta. Pero no todas las tramas contienen protocolos de todas las capas superiores. Si, por ejemplo, estamos utilizando una captura que contiene tráfico de Ethernet con el protocolo ARP, en este caso no contiene direcciones IP. En estos casos, se representa con un valor “Non-Exist” indicando que no existe esta información.

Toda esta información se guarda en un fichero JSON. Además de los datos extraídos de la captura, en el fichero JSON también está incluido el nombre de la captura y el filtro que hemos utilizado. Tras generar el fichero JSON se cierra la ventana Sesión y se abre la ventana Main utilizando los comandos de QT.

Todos estos procesos están incluidos dentro de la funcionalidad del botón “Start”, al pulsar este botón, se ejecutarán todos los procesos anteriores.

- La segunda parte se corresponde con la ventana Main. Al abrir esta ventana se muestra un resumen de la captura como se muestra en la figura 3.21.

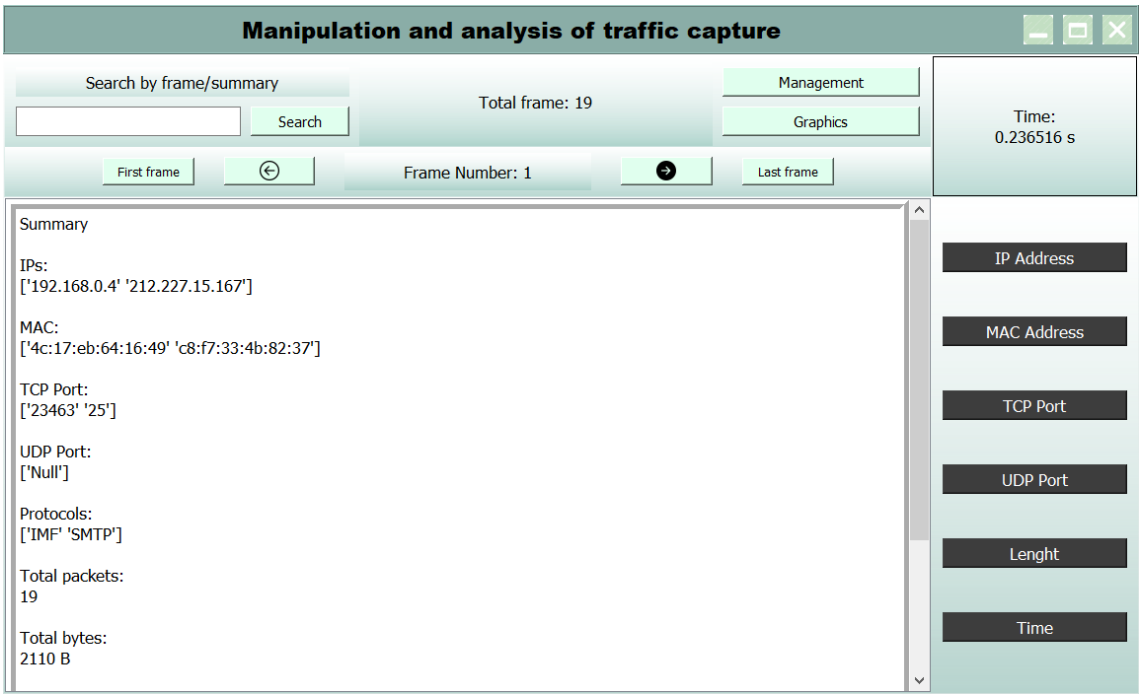


Figura 3.21. Ventana Main generada por Python

En este resumen se muestran las direcciones IP, las direcciones MAC, los puertos TCP o UDP, los protocolos de nivel más alto, el tamaño total de las tramas (en bytes), el número total de tramas, el tiempo total de la captura, la velocidad eficaz (bits/s) y los flags TCP.

Además, en la parte superior derecha de la ventana hay un marco donde se muestra el tiempo delta (figura 3.22) de la trama 1. Normalmente, el tiempo delta de la trama 1 es 0, pero para el caso del ejemplo de la figura, estamos aplicando un filtro de SMTP, por lo que el tiempo delta de la trama 1 no es 0.

Figura 3.22. Marco de la Ventana Main

Además de mostrar el tiempo delta, también se pueden mostrar otros datos de la captura. Para visualizarlo, se utilizan los botones que se encuentran en la parte inferior de este marco, como se puede ver en la figura 3.21. Para ello, existen seis botones, cada uno muestra una información de la captura. Empezando por el de la parte superior, está

el botón “IP Address”, al pulsar este botón se muestran las direcciones IP origen y destino; el botón “MAC Address” muestra las direcciones MAC origen y destino; el botón “TCP Port” muestra los puertos TCP origen y destino; el botón “UDP Port” muestra los puertos UDP origen y destino; el botón “Length” muestra el tamaño de la trama en Bytes. El código que hay detrás de todos estos botones son comandos de Qt, que al pulsar cualquiera de ellos ejecuta un comando y envía la información del fichero JSON a la interfaz gráfica.

Los seis botones sólo permiten mostrar los datos de una trama. Para mostrar la información de otras tramas utilizamos los botones de siguiente y anterior. Y como hemos visto, por defecto, en la ventana se muestra la información de tiempo delta. Si pulsamos los botones anterior y siguiente sólo mostrará el tiempo delta de las tramas. Para mostrar otra información que no sea el tiempo delta es necesario pulsar primero en los seis botones para elegir qué información queremos mostrar en ese marco (figura 3.22). Después, utilizando los botones de anterior y siguiente nos permitirá movernos de una trama a otra. Además, también hay dos botones que nos permiten ir a la primera trama y a la última trama de la captura directamente.

Para conocer en qué estado estamos o, mejor dicho, qué información se está mostrando en ese marco (figura 3.22), hay una lista con seis números que se corresponde con los seis botones. Inicialmente, tenemos todos los números a cero menos el último que es un 1. Este 1 indica que está activado el botón y 0 indica que está desactivado. Por lo tanto, en la pantalla se mostrará el tiempo delta. En la tabla 3.1 se muestra la relación entre la lista y los botones.

Botón	IP Address	MAC Address	TCP Port	UDP Port	Length	Delta Time
Lista	0	0	0	0	0	1

Tabla 3.1. Correspondencia de los botones con la lista

En la parte superior izquierda de la pantalla tenemos un icono de búsqueda que nos permite buscar la trama o buscar por “Summary”. Lo que muestra “Summary” es el resumen de la captura y si buscamos por la trama introduciendo un número de trama, al pulsar el botón “Search” se mostrará toda la información de esa trama, tal y como se muestra en la figura 3.23.

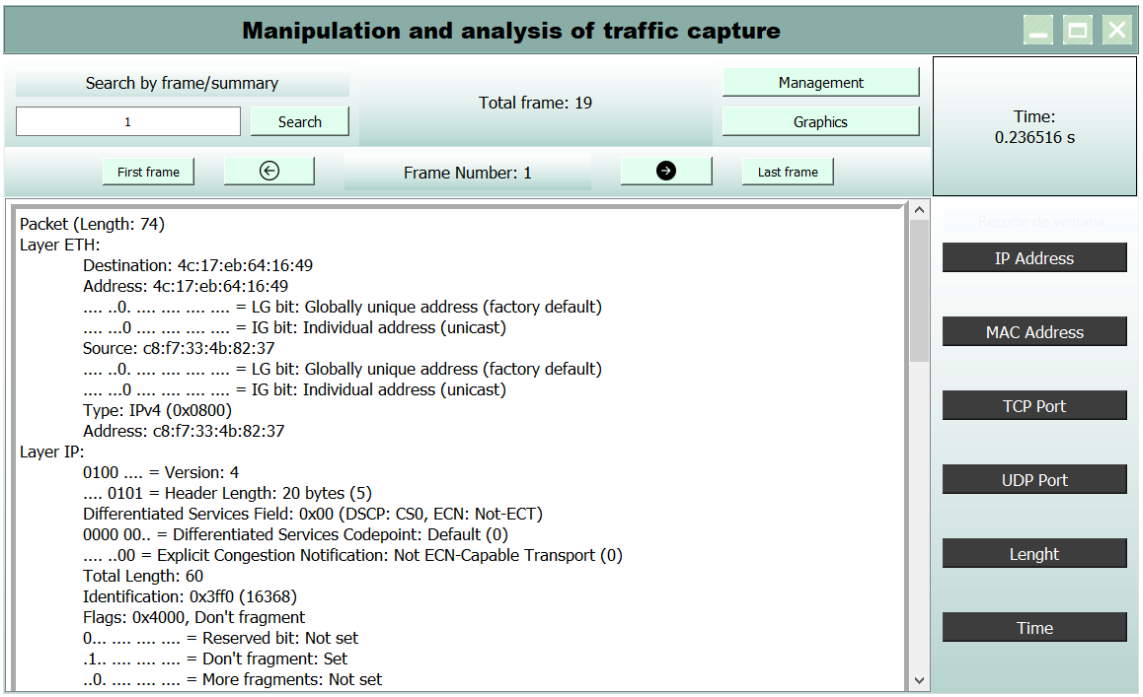


Figura 3.23. Ventana Main generado por Python (2)

En la figura 3.23 se muestra toda la información de la trama 1 de la captura, desde el protocolo de nivel más bajo hasta el nivel más alto, así como los valores de los campos de cada cabecera. Hay una barra de desplazamiento que nos permite visualizar todo el contenido de la trama. Al pulsar el botón “Search” se pasa el número de la trama que queremos leer a una nueva función. Esta función es similar a la de la figura 3.20 y como resultado se obtiene toda la información de la trama. Utilizando el comando de Qt esta información de la trama se mostrará en la interfaz gráfica.

Además, hay dos botones que nos permiten abrir la ventana gráfica y la ventana de gestión. Ambos están conectados con una función que, al hacer click sobre ellos, se llama a las clases que están asociadas con cada ventana. Para llamar a las clases se utiliza el módulo QtWidgets de la librería PyQT5.

- La tercera parte corresponde con la ventana gráfica, en esta parte podemos visualizar 4 gráficas, todas ellas generadas mediante la librería Matplotlib. Al abrir esta ventana lo primero que encontramos son los cuatros botones que se sitúan en la parte inferior, como se muestra en la figura 3.24. Estos botones son los elementos clave de esta ventana.

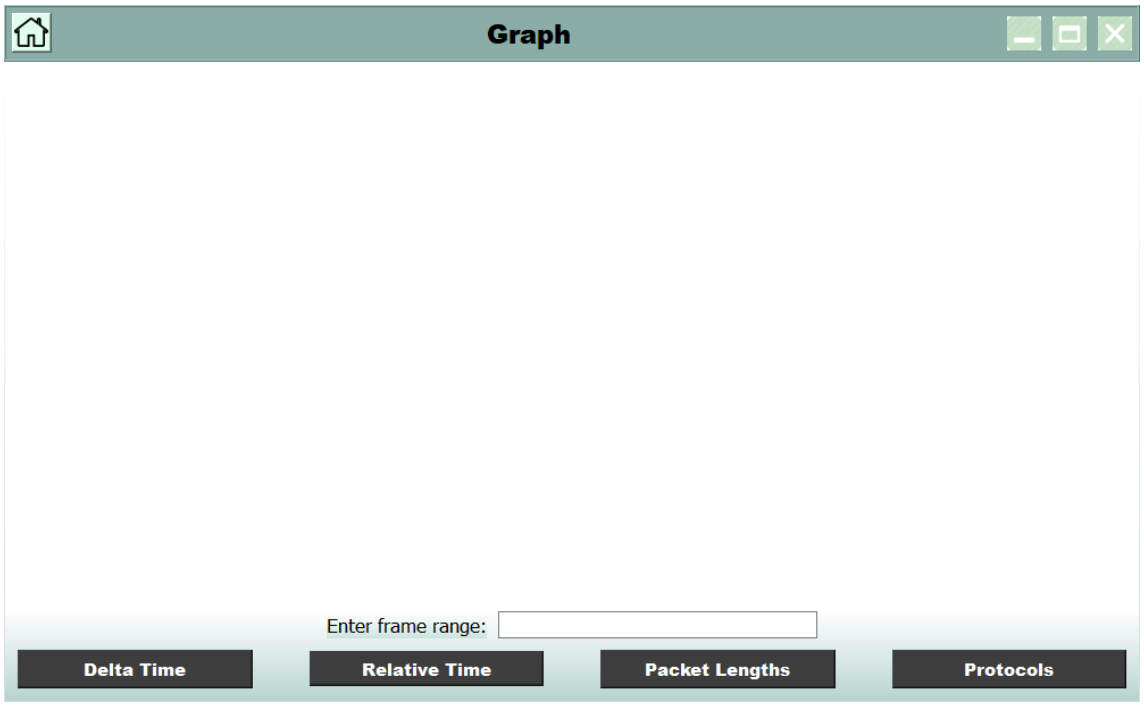


Figura 3.24. Ventana Gráfica generado por Python (1)

Si pulsamos el botón “Delta Time”, se muestra una gráfica relacionada con el tiempo delta y el resultado de este botón es lo que se muestra en la figura 3.25.

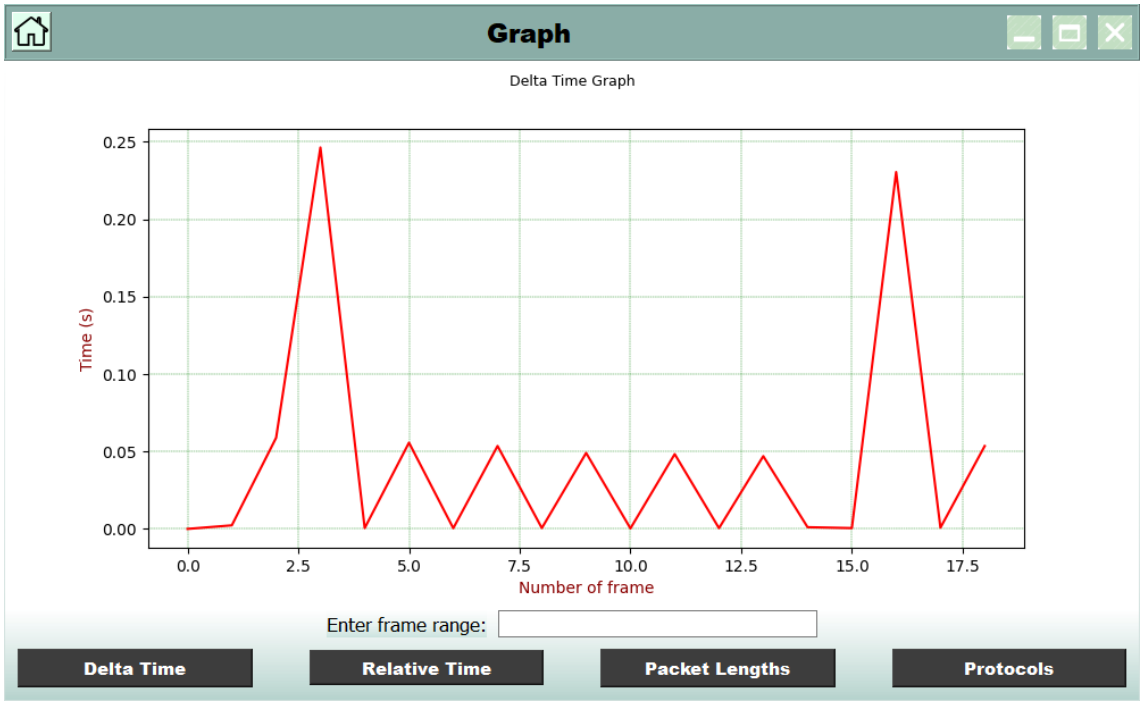


Figura 3.25. Ventana Gráfica generado por Python (2)

Tiempo delta es el tiempo transcurrido entre la trama anterior y la actual, por lo que tiene una forma de diente de sierra que no tiene ninguna dependencia con el tiempo de la trama 1. En cambio, el tiempo relativo, figura 3.25, muestra una gráfica de crecimiento, este tiempo relativo toma como referencia el instante de tiempo inicial el de la primera trama.

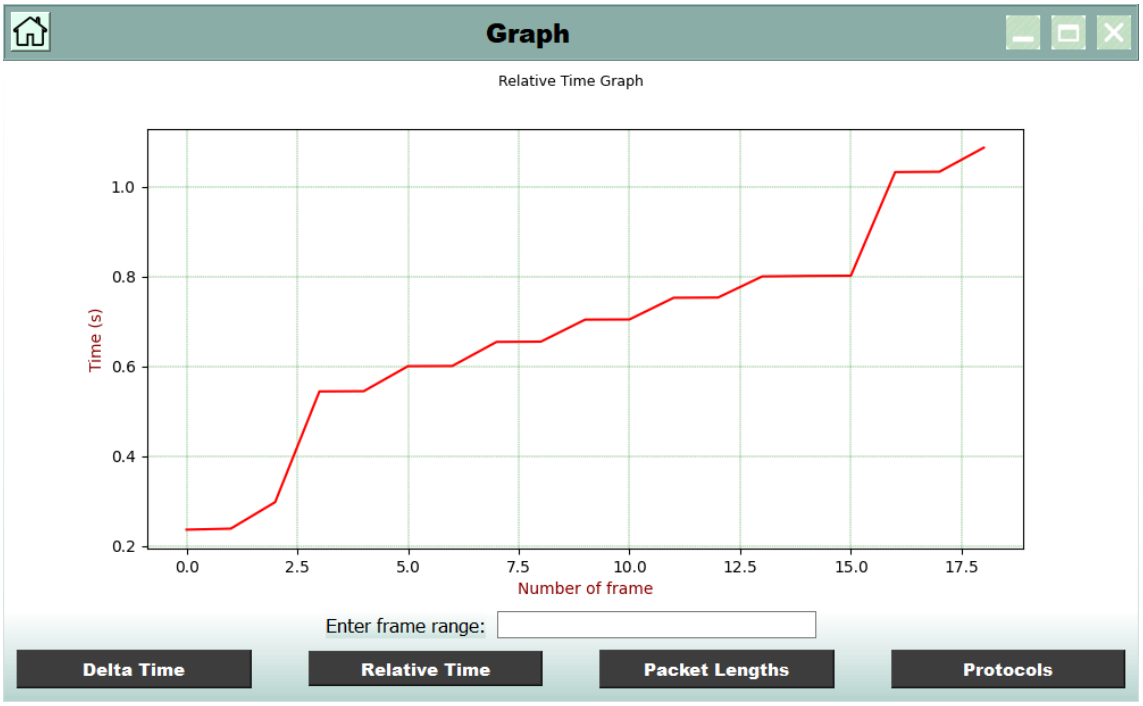


Figura 3.26. Ventana Gráfica generado por Python (3)

Estos dos botones están asociados con una clase denominada “Canvas_graficas*”[26] que tiene como entrada un intervalo de trama. Si simplemente hacemos click sobre el botón “Delta Time” o “Relative Time”, se utilizan todas las tramas de la captura para generar la gráfica. Para mostrar el tiempo relativo o el tiempo delta de unas tramas específicas debemos introducir primero un intervalo de tramas por la línea de texto (QLineEdit), y al pulsar el botón “Delta Time” o “Relative Time”, se recoge el intervalo de tramas que se ha introducido y se generan las gráficas según el intervalo de trama que se le ha pasado.

Hay varias formas de introducir el intervalo de tramas, tal y como se muestra en la tabla 3.2.

Intervalo de tramas introducido	Resultado
12	1-12
5-29	5-29
30	1-29

Tabla 3.2. Intervalo de tramas de la ventana gráfica

Como se muestra en la tabla 3.2, si introducimos un 12 en la línea de texto, al pulsar el botón obtenemos un intervalo de 1-12, indicando que genera la gráfica tomando la información de la trama 1 hasta la 12. Si introducimos un intervalo de 5-29 se mostrará una gráfica con este intervalo de tramas. Pero si introducimos un número de tramas que no existen en la captura, por ejemplo, si tenemos una captura que sólo contiene 29 tramas, al introducir un número superior a 29 se generará una gráfica con todas las tramas que tiene.

Los otros botones no tienen la opción de introducir el intervalo de las tramas. En estos casos, al pulsar sobre ellos se muestra una gráfica relacionada con el tamaño de las tramas (figura 3.27) y una gráfica con la proporción de aparición de cada protocolo en la captura (figura 3.28).

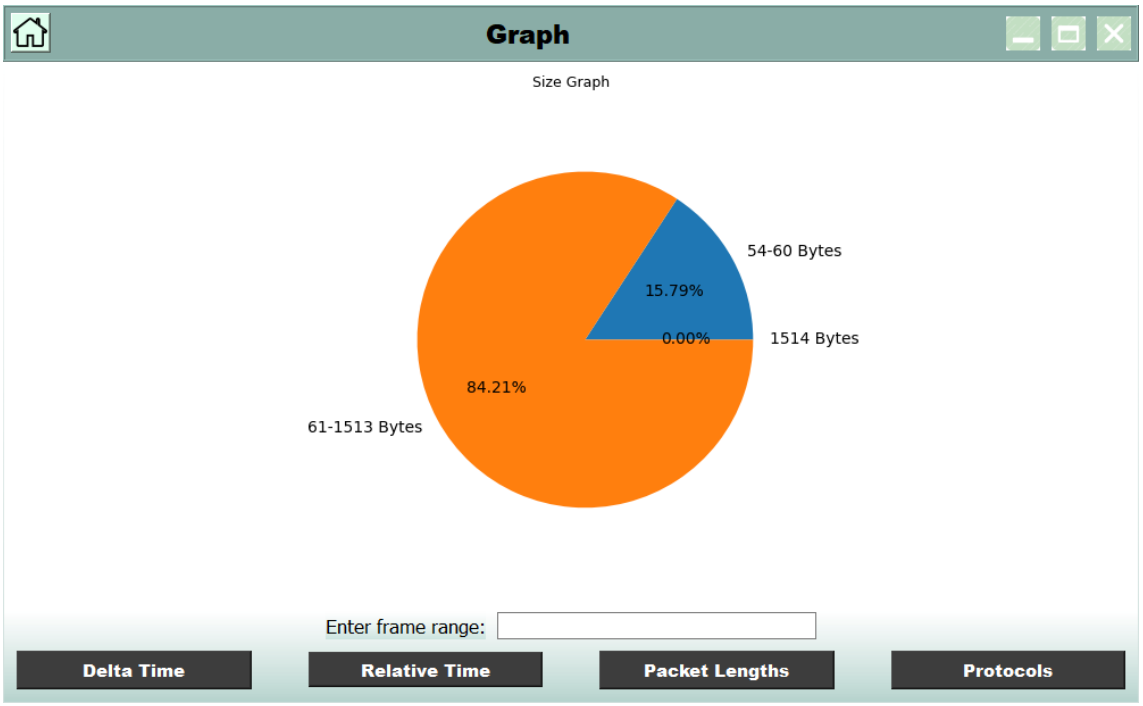


Figura 3.27. Grafica Intervalo de longitud de la captura

En la figura 3.27 se muestra una gráfica sectorial que representa el porcentaje de tramas que tienen un tamaño de bytes entre 54-60 bytes (tamaño mínimo), entre 61-1513 y las tramas que contienen exactamente 1514 bytes (tamaño máximo).

En la figura 3.28 se muestra un diagrama de barras de los protocolos de nivel más alto de cada trama. En este caso tenemos aplicado un filtro SMTP, por lo tanto, sólo muestra el protocolo SMTP y los protocolos de capa superior a él.

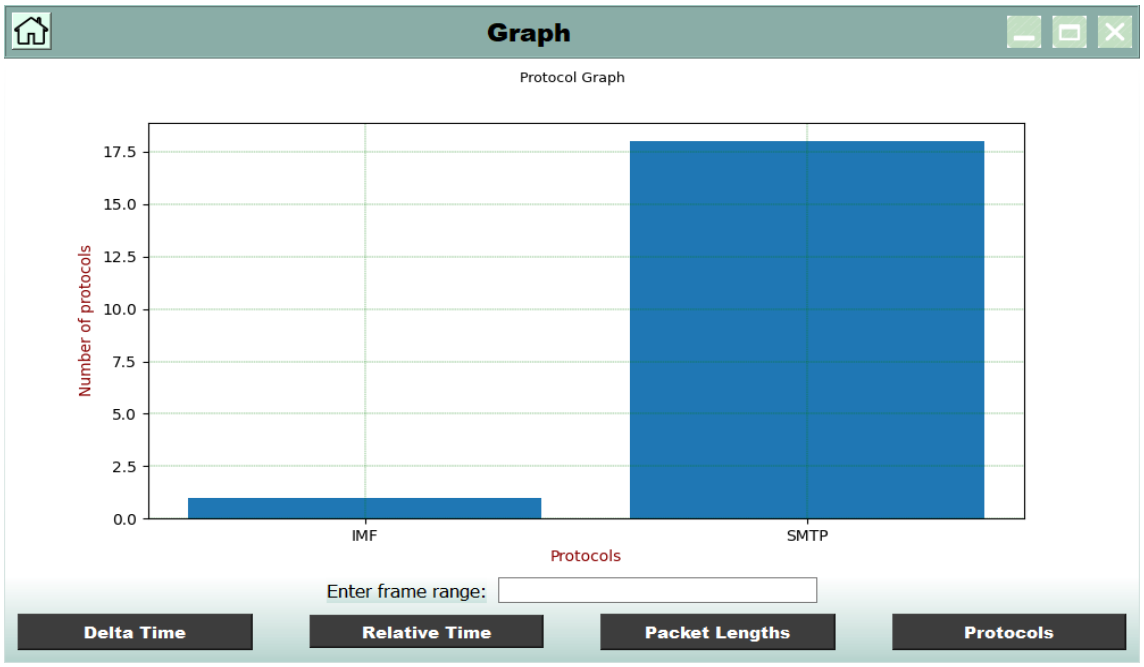


Figura 3.28. Gráfica de protocolos con un filtro SMTP

En la figura 3.29 muestra otra grafica de protocolos sin aplicar el filtro SMTP:

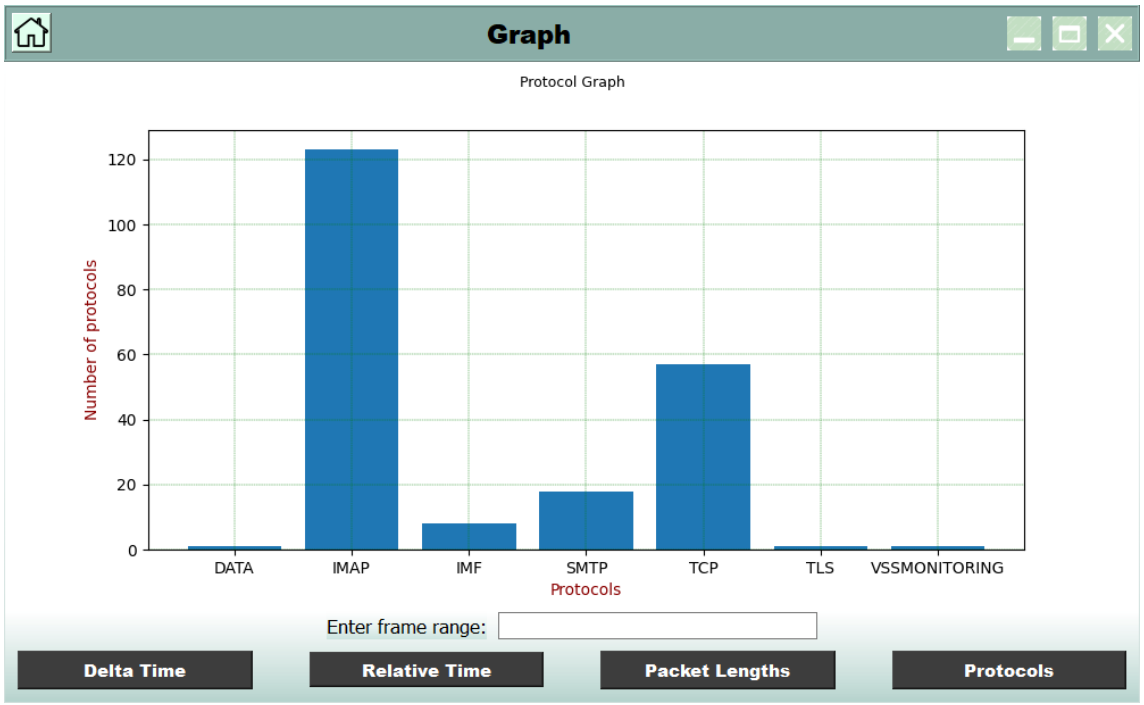


Figura 3.29. Grafica de protocolos sin aplicar filtro SMTP

- La cuarta parte se corresponde con la ventana de gestión, que se encarga de representar la información que se encuentra en la base de datos sobre la interfaz de usuario que hemos generado. Para poder gestionar las diferentes funcionalidades se incluye el uso de las páginas.

En la parte izquierda de la ventana hay tres botones que se usan para moverse de una página (Widget) a otra. Si cliqueamos en el botón Fill Data, nos lleva a una página donde tenemos que elegir cómo vamos a escribir los datos de la captura a la base de datos, si vamos a utilizar la informacion del fichero JSON generado en la ventana sesión o si queremos utilizar una captura nueva. La figura 3.30 muestra los objetos que se encuentra en esta página.

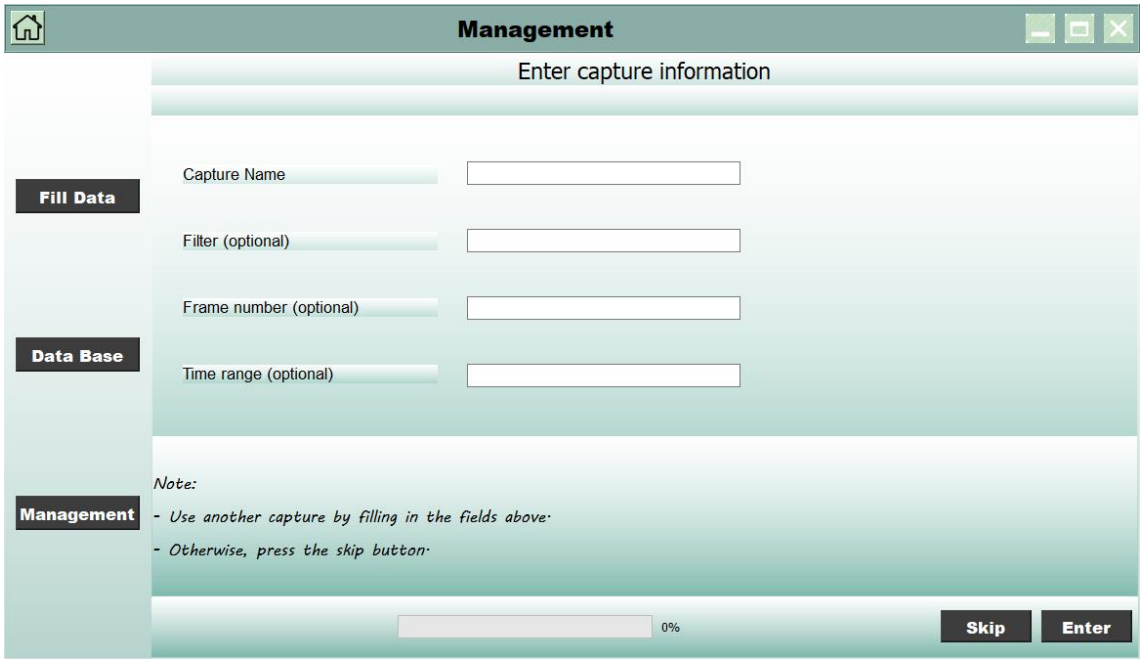


Figura 3.30. Ventana gestión, página Enter Capture Information (1)

Como podemos ver en la figura 3.30, hay una nota que indica la funcionalidad de los dos botones que se sitúan en la parte inferior y cuatro líneas de texto en el centro de la página. Esta nota indica si queremos utilizar una captura diferente que hemos introducido durante la parte 1 (Ventana Sesión), o si hace falta introducir datos en las líneas de texto. El campo "Capture Name" es obligatorio, si está vacío se abrirá una ventana de error indicando que el campo está vacío. El resto de las líneas de texto son opcionales, pero hay que tener en cuenta que solo hace falta rellenar una de las opciones. Si introducimos algún dato en los campos opcionales, aplicaremos un filtro a la captura, pudiendo ser filtrado por un protocolo, por número de tramas o por el tiempo relativo.

Una vez rellenados los datos, al pulsar el botón Enter, en el Qlabel de la parte superior de la página, lo que muestra la figura 3.31, aparece un mensaje de "*Processing...*", y la barra de progreso empieza a incrementar el valor hasta que llega a 100% y se muestra un mensaje de "*Successful*" en el mismo Qlabel (figura 3.32), indicando que los datos de la captura han sido escritos correctamente en la base de datos.

Para leer los datos de la captura se reutilizan las funciones de la parte de ventana sesión y para escribir los datos a la base de datos se utiliza la librería SQLite de Python para insertar los datos de la captura.

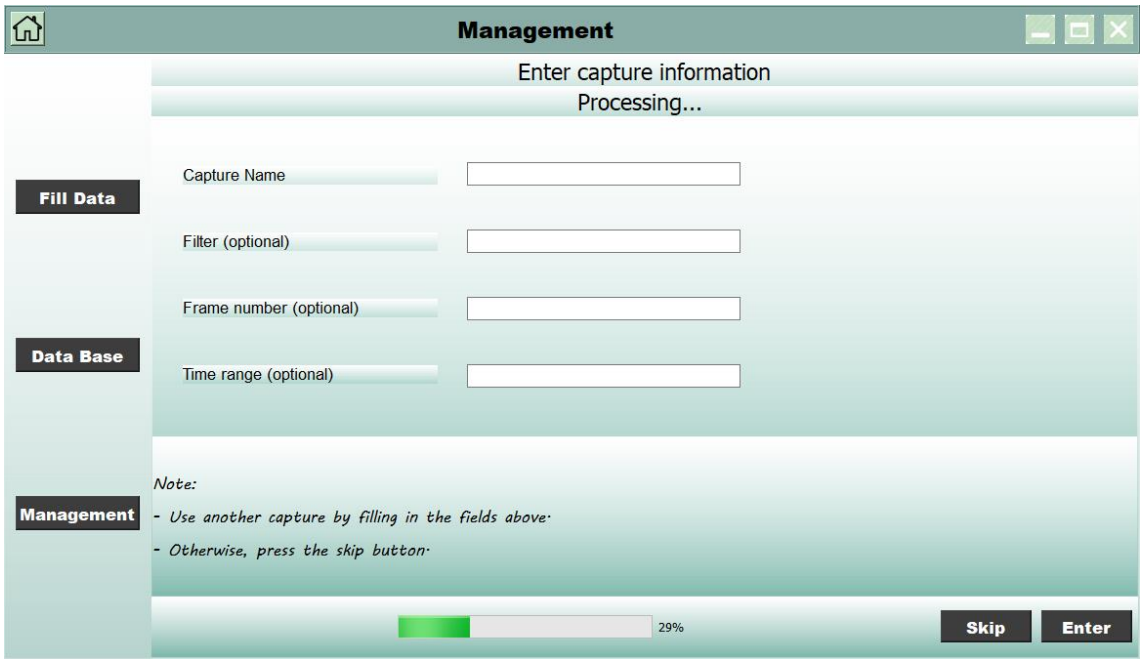


Figura 3.31. Ventana gestión, página Enter Capture Information (2)

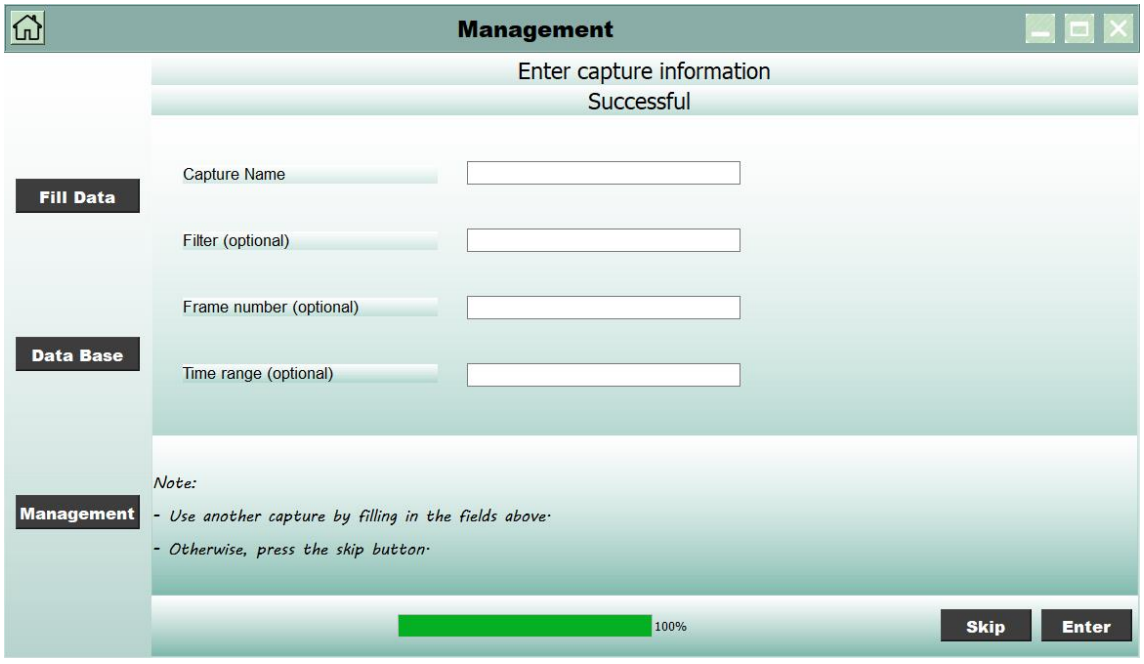
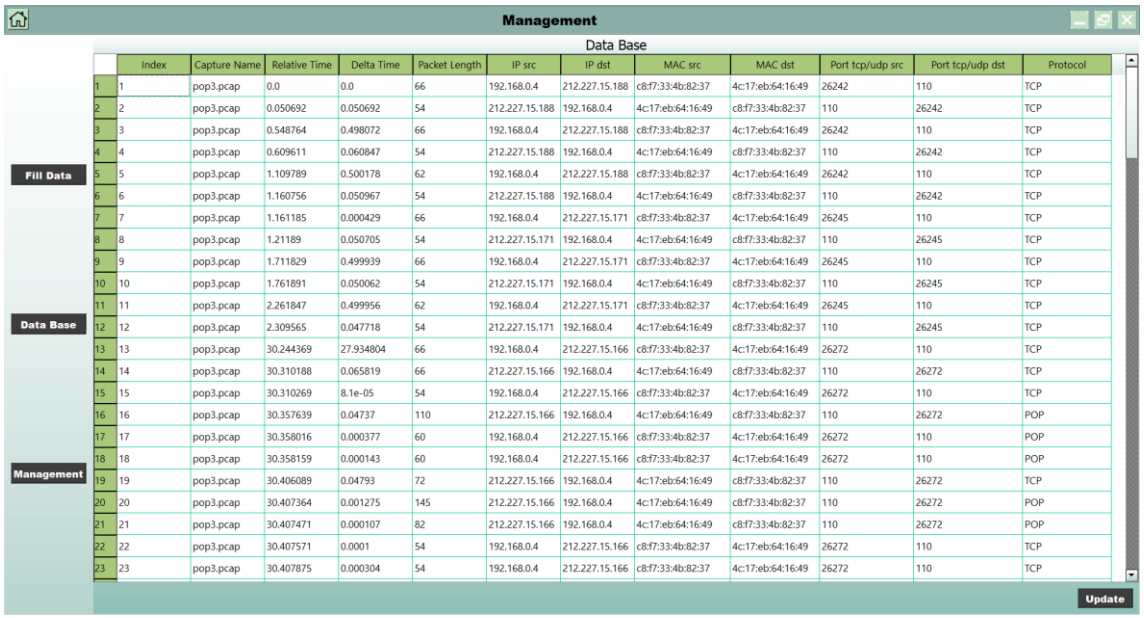


Figura 3.32. Ventana gestión, página Enter Capture Information (3)

Si cliqueamos en el botón Skip, escribimos los datos del fichero JSON a la base de datos, y mientras escribe los datos a la base de datos, la barra de progreso comienza a aumentar el valor hasta que llega a 100% y muestra un mensaje de "*Successful*" en el Qlabel de estado de la captura. [25]

Para visualizar los datos de la base de datos, hace falta ir a la página de Data Base, como se muestra en la figura 3.33.



	Index	Capture Name	Relative Time	Delta Time	Packet Length	IP src	IP dst	MAC src	MAC dst	Port tcp/udp src	Port tcp/udp dst	Protocol
1	1	pop3.pcap	0.0	0.0	66	192.168.0.4	212.227.15.188	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26242	110	TCP
2	2	pop3.pcap	0.050692	0.050692	54	212.227.15.188	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26242	TCP
3	3	pop3.pcap	0.548764	0.498072	66	192.168.0.4	212.227.15.188	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26242	110	TCP
4	4	pop3.pcap	0.609611	0.060847	54	212.227.15.188	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26242	TCP
5	5	pop3.pcap	1.109789	0.500178	62	192.168.0.4	212.227.15.188	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26242	110	TCP
6	6	pop3.pcap	1.160756	0.050967	54	212.227.15.188	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26242	TCP
7	7	pop3.pcap	1.161185	0.000429	66	192.168.0.4	212.227.15.171	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26245	110	TCP
8	8	pop3.pcap	1.21189	0.050705	54	212.227.15.171	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26245	TCP
9	9	pop3.pcap	1.711829	0.499939	66	192.168.0.4	212.227.15.171	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26245	110	TCP
10	10	pop3.pcap	1.761891	0.050062	54	212.227.15.171	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26245	TCP
11	11	pop3.pcap	2.261847	0.499956	62	192.168.0.4	212.227.15.171	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26245	110	TCP
12	12	pop3.pcap	2.309565	0.047718	54	212.227.15.171	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26245	TCP
13	13	pop3.pcap	30.244369	27.934804	66	192.168.0.4	212.227.15.166	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26272	110	TCP
14	14	pop3.pcap	30.310188	0.065819	66	212.227.15.166	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26272	TCP
15	15	pop3.pcap	30.310269	8.1e-05	54	192.168.0.4	212.227.15.166	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26272	110	TCP
16	16	pop3.pcap	30.357639	0.04737	110	212.227.15.166	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26272	POP
17	17	pop3.pcap	30.358016	0.000377	60	192.168.0.4	212.227.15.166	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26272	110	POP
18	18	pop3.pcap	30.358159	0.000143	60	192.168.0.4	212.227.15.166	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26272	110	POP
19	19	pop3.pcap	30.406089	0.04793	72	212.227.15.166	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26272	TCP
20	20	pop3.pcap	30.407364	0.001275	145	212.227.15.166	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26272	POP
21	21	pop3.pcap	30.407471	0.000107	82	212.227.15.166	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26272	POP
22	22	pop3.pcap	30.407571	0.0001	54	192.168.0.4	212.227.15.166	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26272	110	TCP
23	23	pop3.pcap	30.407875	0.000304	54	192.168.0.4	212.227.15.166	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26272	110	TCP

Figura 3.33. Ventana gestión, página Data Base

Para ello tenemos que pulsar el botón Data Base situado en el marco izquierdo de la ventana, tras cliquear este botón muestra una página que contiene una tabla, en la que se mostrarán los datos relacionados con el número de trama, nombre de la captura, tiempo relativo, tiempo delta, longitud de la trama en bytes, dirección IP origen y destino, dirección MAC origen y destino, puerto TCP/UDP origen y destino y, por último, el protocolo de nivel más alto de cada trama.

Inicialmente la tabla está vacía, para mostrar valores en la tabla es necesario cliquear el botón de *update.* La funcionalidad de este botón es extraer los datos de la base de datos y mostrarlos en la tabla. Para extraer los datos se utilizan los comandos de SQL y una vez obtenidos los datos se utilizan los comandos de QT para enviarlos a la tabla.

La tercera página se encarga de filtrar datos de la base de datos y mostrarlos en una tabla, además podemos exportar o eliminar datos de la base de datos. Para ello, hace falta introducir datos en la línea de texto situada en la parte superior de la página y seleccionar el nombre de la columna en el cuadro de listas (QcomboBox) que se sitúa a

la izquierda de la línea del texto. Un ejemplo ilustrativo es lo que se muestra en la figura 3.34.

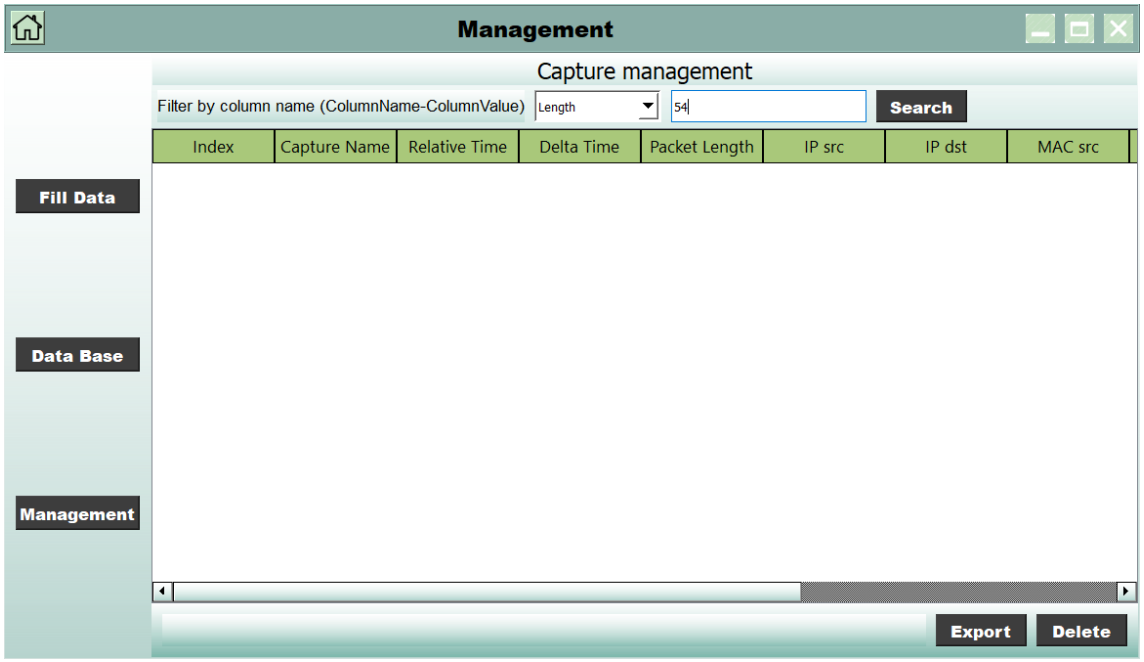


Figura 3.34. Ventana gestión, página Capture Management(1)

Como podemos ver, en el cuadro de la lista hemos seleccionado el nombre de la columna como Length y en la línea de texto hemos introducido un 54. Al cliquear el botón Search, situado en la parte derecha de la línea de texto, en la tabla se muestran todas las tramas que tienen una longitud de 54 Bytes, como muestra la figura 3.35.

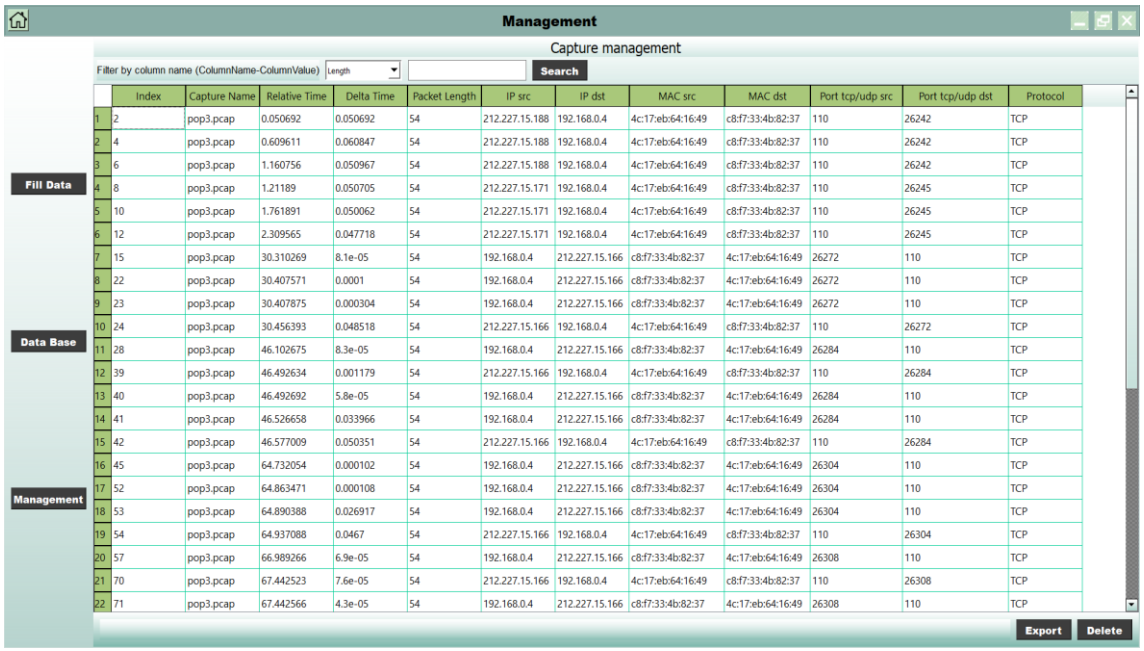
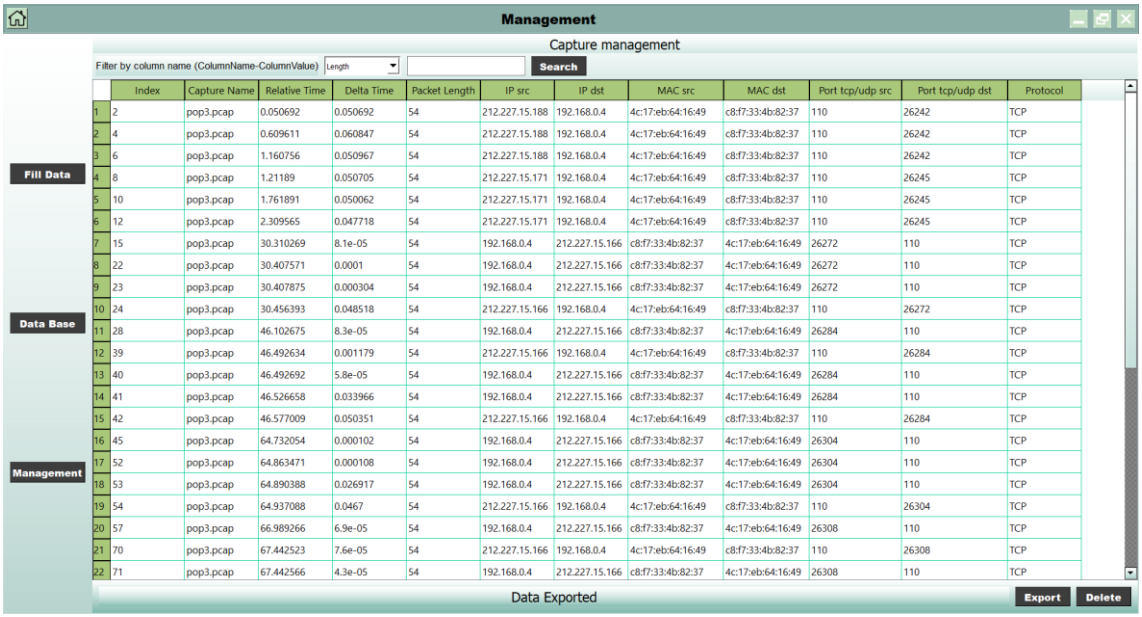


Figura 3.35. Ventana gestión, página Capture Management(2)

La información que se guarda en la base de datos es temporal, es decir, si cerramos esta ventana pulsando el botón de cerrar, desaparecerán todos los datos que se hayan guardado en ella. Por lo tanto, si queremos guardar algún dato de la base de datos hace falta exportarlo. Para esto, hay un botón Export situado en la parte inferior derecha de la ventana. Si la exportación fue exitosa, se muestra un mensaje "*Data Exported*" en Qlabel, como muestra la figura 3.36.



	Index	Capture Name	Relative Time	Delta Time	Packet Length	IP src	IP dst	MAC src	MAC dst	Port tcp/udp src	Port tcp/udp dst	Protocol
1	2	pop3.pcap	0.050692	0.050692	54	212.227.15.188	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26242	TCP
2	4	pop3.pcap	0.609611	0.060847	54	212.227.15.188	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26242	TCP
3	6	pop3.pcap	1.160756	0.050967	54	212.227.15.188	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26242	TCP
4	8	pop3.pcap	1.21189	0.050705	54	212.227.15.171	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26245	TCP
5	10	pop3.pcap	1.761891	0.050062	54	212.227.15.171	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26245	TCP
6	12	pop3.pcap	2.309565	0.047718	54	212.227.15.171	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26245	TCP
7	15	pop3.pcap	30.310269	8.1e-05	54	192.168.0.4	212.227.15.166	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26272	110	TCP
8	22	pop3.pcap	30.407571	0.0001	54	192.168.0.4	212.227.15.166	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26272	110	TCP
9	23	pop3.pcap	30.407875	0.000304	54	192.168.0.4	212.227.15.166	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26272	110	TCP
10	24	pop3.pcap	30.456393	0.048518	54	212.227.15.166	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26272	TCP
11	28	pop3.pcap	46.102675	8.3e-05	54	192.168.0.4	212.227.15.166	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26284	110	TCP
12	39	pop3.pcap	46.492634	0.001179	54	212.227.15.166	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26284	TCP
13	40	pop3.pcap	46.492692	5.8e-05	54	192.168.0.4	212.227.15.166	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26284	110	TCP
14	41	pop3.pcap	46.526658	0.033966	54	192.168.0.4	212.227.15.166	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26284	110	TCP
15	42	pop3.pcap	46.577009	0.050351	54	212.227.15.166	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26284	TCP
16	45	pop3.pcap	64.732054	0.000102	54	192.168.0.4	212.227.15.166	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26304	110	TCP
17	52	pop3.pcap	64.863471	0.000108	54	192.168.0.4	212.227.15.166	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26304	110	TCP
18	53	pop3.pcap	64.890388	0.026917	54	192.168.0.4	212.227.15.166	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26304	110	TCP
19	54	pop3.pcap	64.937088	0.0467	54	212.227.15.166	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26304	TCP
20	57	pop3.pcap	66.989266	6.9e-05	54	192.168.0.4	212.227.15.166	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26308	110	TCP
21	70	pop3.pcap	67.442523	7.6e-05	54	212.227.15.166	192.168.0.4	4c:17:eb:64:16:49	c8:f7:33:4b:82:37	110	26308	TCP
22	71	pop3.pcap	67.442566	4.3e-05	54	192.168.0.4	212.227.15.166	c8:f7:33:4b:82:37	4c:17:eb:64:16:49	26308	110	TCP

Figura 3.36. Ventana gestión, página Capture Management(3)

Todo lo que podemos visualizar en la tabla se guarda en un fichero Excel y todos estos ficheros exportados se almacenan en la carpeta denominada "result". Para eliminar lo que muestra esta tabla (figura 3.36), se utiliza el botón Delete. Al pulsar este botón se eliminan todos los datos relacionados con la longitud de la trama de la base de datos y muestra un mensaje de "*Data Deleted*" en Qlabel cuando la operación de eliminar es correcta, como muestra la figura 3.37.

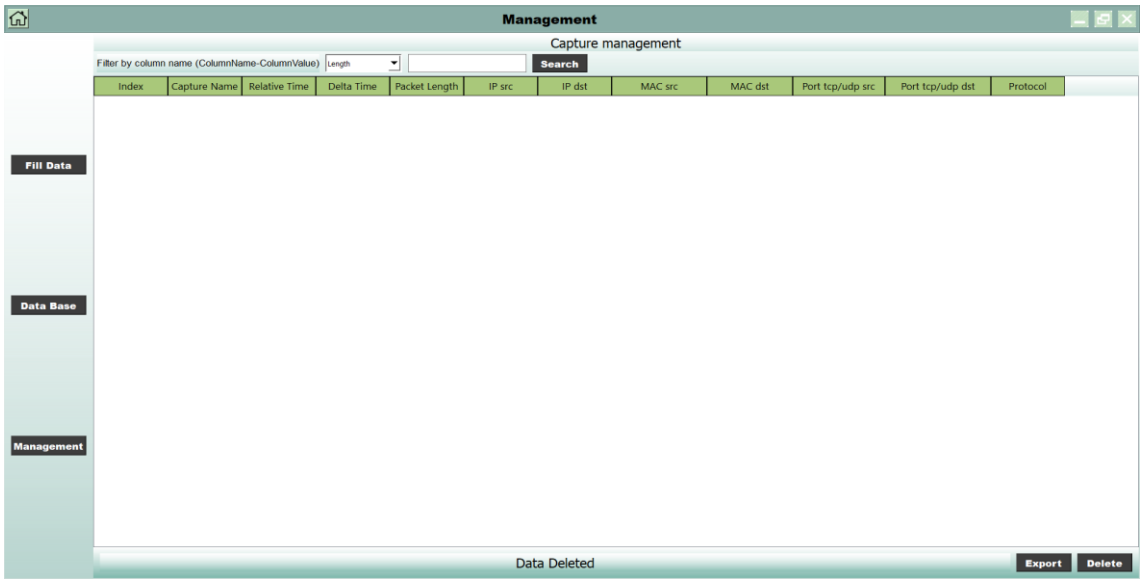


Figura 3.37. Ventana gestión, página Capture Management(4)

3.4. Aplicación final

En esta sección explicaremos cómo convertir los scripts (formato .py) en archivos ejecutables (formato .exe). Para realizar la conversión, utilizamos la librería "auto-py-to-exe" de Python.[22] Cuando se ejecuta el comando "auto-py-to-exe" a través del terminal de Python o CMD, aparecerá la ventana que se muestra en la figura 3.38.

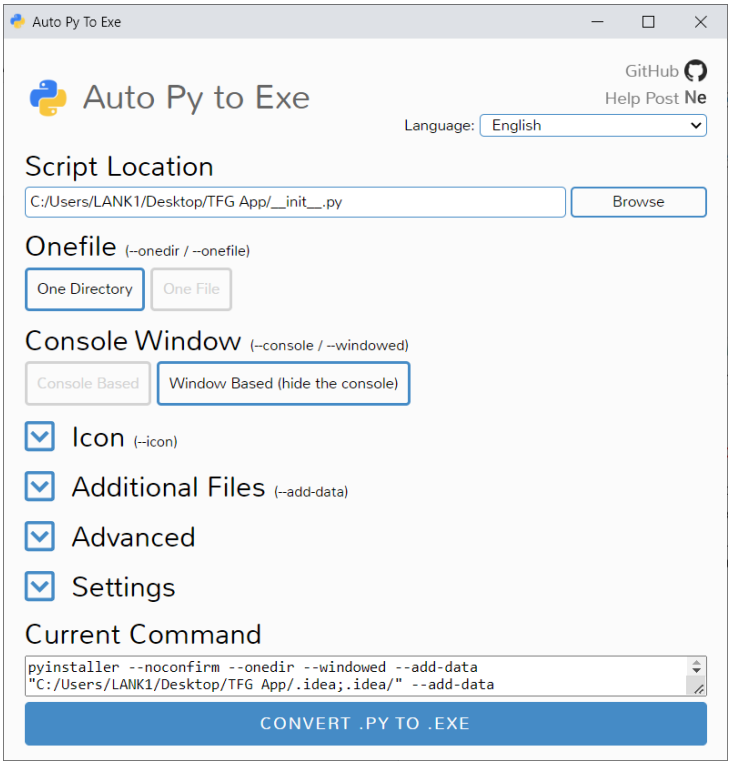


Figura 3.38. Ventana generada por el comando Auto-py-to-exe (1)

En esta ventana hay una serie de parámetros y opciones que tenemos que rellenar. Lo primero, es añadir la ruta donde se encuentra nuestro script. Luego, en el campo “Onefile”, seleccionamos “One Directory”, que significa que el resultado de ejecutar “auto-py-to-exe” será una carpeta en la que estará el ejecutable .exe y otros ficheros necesarios. En este caso, no se ha utilizado la opción “One File” porque al ejecutar el fichero generado no aparecía ninguna ventana, ni siquiera errores, por lo que se ha descartado.

En el campo “Console Windows” seleccionamos la opción de “Window Based (hide the console)” para indicar que se trata de una aplicación que utiliza ventanas y, por lo tanto, las ventanas de terminal o consola deben estar ocultas al usuario.

Además, para que el ejecutable funcione, es necesario añadir las carpetas que están asociadas con el script, como muestra la figura 3.39, en el apartado “Additional Files”.

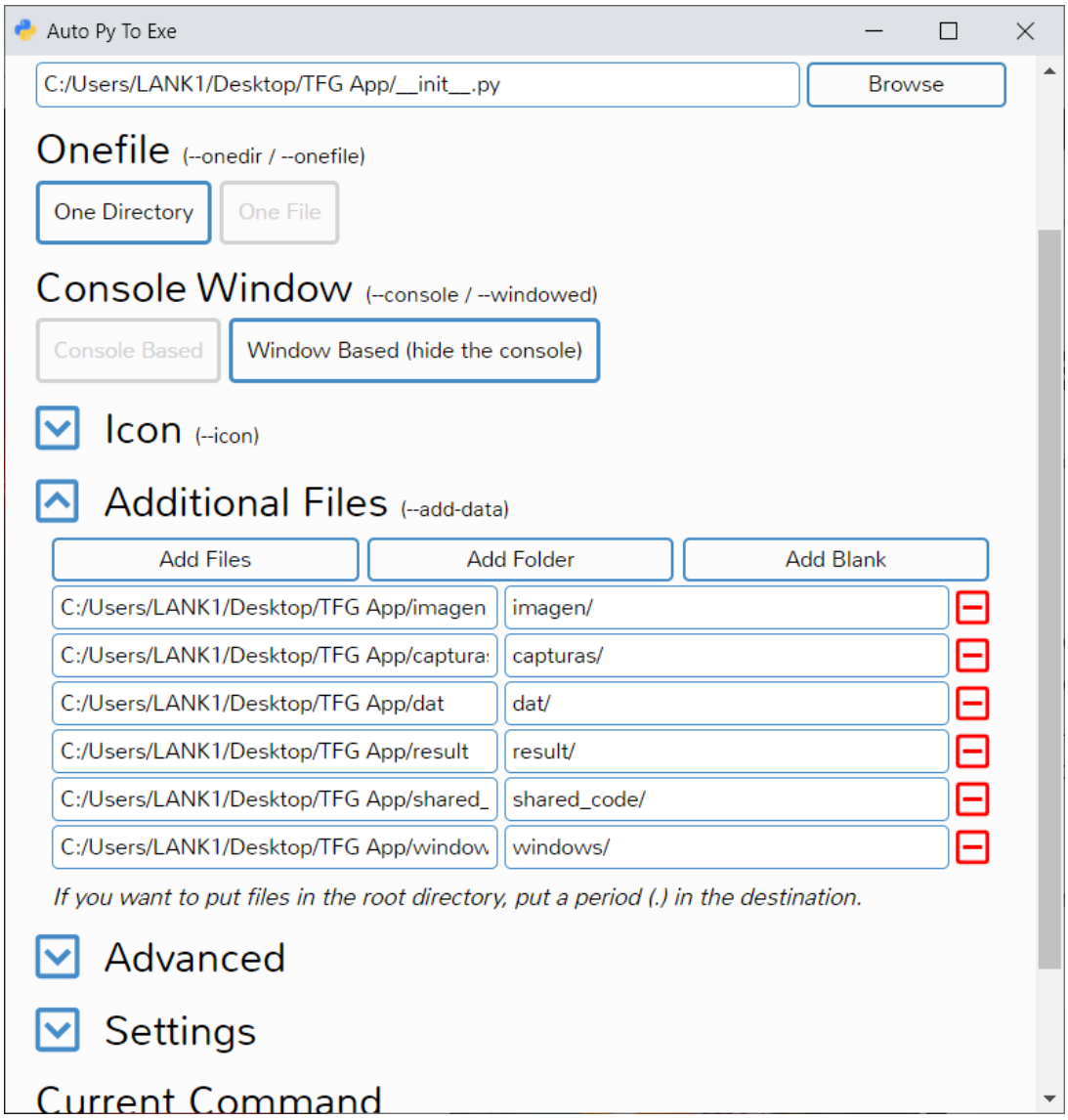


Figura 3.39. Ventana generada por el comando Auto-py-to-exe (2)

Como podemos ver, en este campo se han añadido varias carpetas, como por ejemplo “imagen”, la cual contiene todas las imágenes que se han utilizado en este proyecto. La carpeta “capturas” contiene todos los ficheros de tráfico. En la carpeta “dat” están los ficheros JSON y la base de datos. En la carpeta “result” se guardan todos los ficheros Excel generados por la aplicación. La carpeta “shared_code” contiene todas las funciones utilizadas y en “windows” están todas las ventanas generadas.

Por último, tras cliquear en el botón “CONVERT .PY TO .EXE”, como se muestra en la figura 3.38, se genera una carpeta llamada “output”, en la que se encuentra el ejecutable generado, las librerías de Python que se han utilizado en los scripts y las carpetas que se muestran en la figura 3.39.

3.4.1. Dimensión del trabajo realizado

Con motivo de reflejar la dimensión del trabajo llevado a cabo, se han cuantificado algunas características del mismo. La aplicación diseñada tiene un total de 3363 líneas de código, incluyendo tanto los códigos generados por el comando pyic5 (Apartado 2.4) como los generados por nosotros.

Todo este código fuente está dividido en dos partes: el código dedicado a crear y gestionar las interfaces gráficas que está formado por cinco clases interconectadas una a otra, es decir, cada clase llama a otra clase. Éstas se encuentran en una carpeta llamada “Windows” y ocupa aproximadamente 132 KB. La otra parte de código son los ficheros compartidos, que son pequeñas funciones que son utilizadas por las cinco clases. Éstos se encuentran en la carpeta “shared_code” y tienen un tamaño aproximado de 27 KB.

Además, se ha utilizado una serie de librerías propias de Python, como PyQt5, json, pyshark, matplotlib, sqlite3, numpy, pandas, py, yaml y sys.

CAPÍTULO 4. CONCLUSIONES Y LÍNEAS FUTURAS

Los resultados del proyecto se han ajustado a lo que se había marcado como objetivos inicialmente. Es decir, a partir de una captura de tráfico se ha podido extraer la información que necesitamos, representándola a través de las interfaces gráficas que hemos creado.

Asimismo, se ha conseguido crear una aplicación similar a Wireshark, pero más fácil de manejar y de visualizar los datos de la captura. Se ha logrado también gestionar capturas de tipo pcap o pcapng que contengan diferentes tipos de tráficos y de protocolos, ya sean de capa de enlace (Ethernet, ARP) como de capas superiores (IP, TCP, etc.).

Durante este proyecto, se han utilizado diferentes herramientas, como puede ser la librería Pyshark de Python para extraer los datos de una captura (ya sea con formato pcap o pcapng), la librería PyQt, la aplicación Qt Designer para generar la interfaz gráfica y la librería SQL o JSON para guardar los datos de la captura. Estas herramientas han facilitado la implementación de proyecto, al ser librerías que están asociadas con Python. De esta manera, no hace falta utilizar otros lenguajes de programación para realizar el proyecto y resulta fácil de interactuar con todas ellas.

4.1. Líneas futuras

Aunque se han logrado todos los objetivos iniciales, hay que mencionar que existen algunos problemas cuando queremos utilizar capturas con un tamaño relativamente grande. Por ejemplo, cuando usamos capturas de 200 tramas o más , al pulsar cualquier botón de la ventana gráfica se cierra automáticamente la ventana Main. En este sentido, no ha sido posible detectar el problema que lo causa. Sin embargo, para solucionarlo se ha añadido un botón para, simplemente cliqueando en él, retroceder a la ventana Main.

Además, cuando el programa es ejecutado desde la ventana sesión y usamos ciertas capturas en la ventana gestión, al pulsar el botón “Enter” se cierran todas las ventanas, pero el contenido de la captura se escribe en la base de datos. Esto tampoco se ha

conseguido saber por qué sucedía. En cambio, si todo es ejecutado desde la ventana main no se produce ningún cierre inesperado.

Una mejora que se podía implementar en el futuro es aumentar la velocidad de ejecución a la hora de guardar los datos de la captura. La lectura de la captura se realiza con la librería Pyshark por lo que este tiempo es inevitable, pero la forma en la que se guardan los datos se podría modificar para que no hiciera falta leer de nuevo las capturas. Para ello, una de las opciones que se podían implementar cuando se utilicen capturas que contengan muchas tramas es generar un archivo auxiliar para guardar temporalmente los datos extraídos de la captura o guardar los datos de la captura en la nube. De esta manera no hace falta leer muchas veces la captura aumentando así la velocidad de ejecución de los códigos.

Otra mejora que se podía implementar para aumentar la velocidad de ejecución del código es utilizar otro IDE. La versión que se ha utilizado es la gratuita de Pycharm, por lo que consume más memoria comparado con otras IDE gratuitas, como por ejemplo VS code. Por ello, si queremos seguir utilizando Pycharm se podría utilizar la versión de pago que es más rápida y consume menos memoria. [20]

En cuanto a la UI (User Interface) también se podrían añadir nuevos elementos o funciones, como por ejemplo incluir eventos de teclado, [24] de manera que los botones de la aplicación estén asociados con teclas del teclado del PC.

Además, lo que se podía implementar en el futuro es gestionar los tráficos de la red en tiempo real, es decir, no utilizar una captura ya generada. Para conseguir esto se puede utilizar el comando LiveCapture de la librería Pyshark, con el cual podemos capturas tráfico que está circulando en la red en tiempo real y representar esta información en la interfaz de usuario. Para esto también es necesario modificar las ventanas gráficas generadas, adaptando la representación de los datos en tiempo real.

Referencias

[1]. “Modelo OSI”, Wikipedia. [20 de Agosto de 2022]

https://es.wikipedia.org/wiki/Modelo_OSI

[2]. “Que es modelo OSI?”, Cloudflare. [20 de Agosto de 2022]

https://www.cloudflare.com/es-es/learning/ddos/glossary/open-systems-interconnection-model-osi/

[3]. “El impacto de internet en la vida diaria”, BBVA. [30 de Agosto de 2022]

https://www.bbvaopenmind.com/articulos/el-impacto-de-internet-en-la-vida-diaria/

[4]. “Concepto de capa física”, planificación y Administración de Redes. [20 de Agosto de 2022]

https://planificacionadministracionredes.readthedocs.io/es/latest/Tema03/Teoria.html

[5]. “Comunicación de datos”, Grado en Ingeniería de Tecnologías de Telecomunicación. 2º Curso. Rama Común. Roberto Sanz y Marta García.

[6]. “Aplicaciones y Servicios en Redes”, Grado en Ingeniería de Tecnologías de Telecomunicación. 4º Curso. Mención en Telemática. Alberto Eloy García y

Marta García

[7]. “Wireshark”, Wikipedia. [20 de Agosto de 2022]

https://es.wikipedia.org/wiki/Wireshark

[8]. “What is Wireshark?”, Wiki wireshark. [20 de Agosto de 2022]

https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html

[9]. “NetworkMedia”, Wireshark in GitLab. [20 de Agosto de 2022]

https://gitlab.com/wireshark/wireshark/-/wikis/CaptureSetup/NetworkMedia

[10]. “Qt”, Wiki Qt. [20 de Agosto de 2022]

https://wiki.qt.io/About_Qt

[11]. “The future is written with Qt”, Qt Documentation. [20 de Agosto de 2022]

https://doc.qt.io/

[12]. “Using Qt Designer”, PyQt 5.7 Reference Guide. [20 de Agosto de 2022]

https://doc.bccnsoft.com/docs/PyQt5/designer.html#ref-designer-plugins

[13]. “Pyshark”, Python Packet parser using wireshark´s tshark. [20 de Agosto de 2022]

http://kiminewt.github.io/pyshark/

[14]. “Pyshark”, thePacketGeek. [20 de Agosto de 2022]

https://thepacketgeek.com/pyshark/

[15]. “Pyshark Github”, KimiNewt. [20 de Agosto de 2022]

https://github.com/KimiNewt/pyshark

[16]. "DB Browser for SQLite", The Official home of the DB Browser for SQLite. [20 de Agosto de 2022]

https://sqlitebrowser.org/

[17]. "Pandas documentation”, pandas. [20 de Agosto de 2022]

https://pandas.pydata.org/docs/

[18]. “Matplotlib 3.5.3 documentation”, matplotlib. [20 de Agosto de 2022]

https://matplotlib.org/stable/index.html

[19]. “Numpy Python”, Numpy org. [20 de Agosto de 2022]

https://numpy.org/

[20]. "Pycharm vs. VS Code: Choosing the best Python IDE", LogRocket. [20 de Agosto de 2022]

https://blog.logrocket.com/pycharm-vs-vscode/

[21]. “Tabla con Python y PyQt5”, Andres Niño, YouTube. [20 de Agosto de 2022]

https://www.youtube.com/watch?v=3paHUQiDn8M

[22]. “auto-py-to-exe 2.22.0”, pypi org. [28 de Agosto de 2022]

https://pypi.org/project/auto-py-to-exe/#description

[23]. “PyQt5/PyQt, PyQt Examples”, Github. [20 de Agosto de 2022]

https://github.com/PyQt5/PyQt

[24]. “Enseñanza practica de PyQt-Eventos y señales en PyQt5”, programador clic. [20 de Agosto de 2022]

https://programmerclick.com/article/88261352090/

[25]. “Señal PyQt y mecanismo de procesamiento de eventos de ranura”, programador clic. [20 de Agosto de 2022]

https://programmerclick.com/article/9706524157/

[26]. “Muestra los gráficos dibujados por matplotlib en la interfaz GUI diseñada por PyQt5”, programador clic. [20 de Agosto de 2022]

https://programmerclick.com/article/5902761106/

[27]. “Ejemplos de UI diseñado por Qt Designer”, MagnoEfren, GitHub. [20 de Agosto de 2022]

https://github.com/MagnoEfren

[28]. “Pyshark Tutorial”, KimiNewt, YouTube. [20 de Agosto de 2022]

https://www.youtube.com/watch?v=1COC92XJluA