

**PROGRAMACIÓN ORIENTADA A OBJETOS (BMA15- P)**

**"OptiNet: Solución Inteligente para Redes Eficientes y Seguras”**

**INTEGRANTES:**

* Callupe Menejes Camilo Roger
* Chagua Bustos Lincoln Haziel
* Gamarra Cruz Diego Orlando
* Suica Rojas Victor Manuel

**DOCENTE:**

* Tello Canchapoma Yuri Oscar

**UNIVERSIDAD NACIONAL DE INGENIERÍA 2025-1**

**FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA**

**ÍNDICE**

[Objetivos Principales 2](#_Toc285804374)

[Objetivos Secundarios 2](#_Toc324498256)

[Antecedentes 4](#_Toc547557838)

[Cronograma 7](#_Toc1857696900)

[Diagrama UML 8](#_Toc852949374)

[Clases y métodos 8](#_Toc1882698749)

# **Objetivos Principales**

Desarrollar un software que permita el análisis de la red a la cual un dispositivo se encuentra conectado, emitiendo un diagnóstico certero sobre el estado de la red, lo que permitirá solucionar problemas de rendimiento, detectar anomalías y posibles amenazas de esta manera se podrá mejorar el estado de la red además de mejorar la seguridad de la misma.

# **Objetivos Secundarios**

1.-Análisis de tráfico en tiempo real:

Realizar un análisis del tráfico de red consiste en revisar y examinar los datos que se mueven en una red lo que permite mejorar el rendimiento y detectar posibles amenazas de seguridad.

2.-Detector de dispositivos "fantasma" en la red:

Detectar todos los dispositivos que se encuentran conectados a una red y de esta manera ubicar a los dispositivos no autorizados por el usuario con el fin de expulsarlos de la red.

3.-Analizador de Wi-Fi

Detectar dispositivos no autorizados en la red e interferencias con otras redes locales y de esta manera sugerir el canal más optimo, medir la intensidad de Wifi en cierto punto además de analizar el tipo de cifrado.

4.-Monitoreo de ancho de banda por dispositivo

Medir el consumo de ancho de banda por dispositivo y de esta manera evitar cuellos de botellas y saturación del sistema ajustando el ancho de banda garantizando así un mejor rendimiento de la red además de prevenir ciberataques o un uso indebido de la red.

5.-Sistema de reporte de errores

El software emitirá al usuario un reporte detallado el cual incluirá la información obtenida durante el análisis, este reporte incluirá información sobre la conexión wifi,como su intensidad o tipo de cifrado, tráfico en la red y posibles amenazas en esta.

Para llevar a cabo este proyecto se utilizarán las siguientes librerías :

- Psutil: Obtiene información de los procesos y utilización del sistema como pueden ser la cpu, memoria, discos, redes o sensores y que funciona bajo Windows, MacOS o Linux.

- Scapy: Permite al usuario enviar, rastrear, diseccionar y falsificar paquetes de red.

- Socket: Verificar la conectividad de red

- PyQt: Implementación de GUI

- Matplotlib: Visualización de gráficos

- Wifi: Obtener Información de la red Wifi

- Nmap: Permite descubir redes y auditar la seguridad de estas. Se puede utilizar para monitorizar los dispositivos de red, comprobar su estado,etc.

- Sentry-sdk: Generador de Reportes de errores

# **Antecedentes**

La creación del firewall surgió como resultado de los avances en seguridad de red que muchos expertos han logrado a lo largo del tiempo, como Jeff Mogul, Paul Vixie, Brian Reid, William Cheswick, Steven Bellovin, David Presotto, Marcus Ranum, David Pensak, Nir Zuk, Fred Avolio y Brent Chapman.

El firewall es un sistema que protege las redes informáticas de accesos no autorizados, para cumplir con dicha función controla el tráfico que entra y sale de un dispositivo, verifica la identidad de todo lo que intenta pasar, evita el acceso no autorizado a datos sensibles y frustra posibles infecciones de malware.

Algunos proyectos similares al que se realizará y que cumplen algunas de las funcione de un firewall son lo siguientes:

**Wireshark**

Wireshark es una herramienta de análisis de protocolos de red de código abierto, ampliamente utilizada para capturar y examinar el tráfico que circula por una red en tiempo real.

Entre sus características más resaltantes se encuentran las siguientes:

* **Captura de paquetes**: Puede capturar el tráfico que pasa por una interfaz de red, como Ethernet o Wi-Fi.
* **Análisis detallado**: Puedes inspeccionar cada paquete, ver encabezados, protocolos utilizados (como TCP, UDP, HTTP, etc.).
* **Análisis de problemas de red:** Permite diagnosticar cuellos de botella, pérdida de paquetes, retardos, errores en la transmisión



**Kismet**

Kismet es una herramienta de detección de redes inalámbricas y de análisis de seguridad WiFi. Es utilizada para descubrir, monitorear y auditar redes WiFi, Bluetooth y otras señales inalámbricas. Entre sus principales características se encuentran las siguientes:

* Detección de redes inalámbricas ocultas o no autorizadas
* Monitoreo de tráfico inalámbrico
* Identificación de dispositivos conectados
* Identificación de dispositivos conectados
* Análisis de señal y cobertura

**OpenWRT**

OpenWrt es un sistema operativo basado en Linux diseñado específicamente para routers y dispositivos de red. Sirve para reemplazar el firmware original en routers comerciales, ofreciendo muchas más funciones, flexibilidad y control.Este sistema operativo permite realizar las siguietes acciones:

* Agregar funcionalidades avanzadas a tu router.
* Mayor seguridad y actualizaciones.
* Monitoreo y diagnóstico de red.



**Circle de Disney**

Disney Circle, también conocido como Circle with Disney, es un dispositivo de control parental y gestión del tiempo en línea diseñado para ayudar a las familias a administrar el acceso a Internet en el hogar.

* Control parental
* Gestión del tiempo
* Pausar el Internet
* Supervisión de actividad



**Eero Secure**

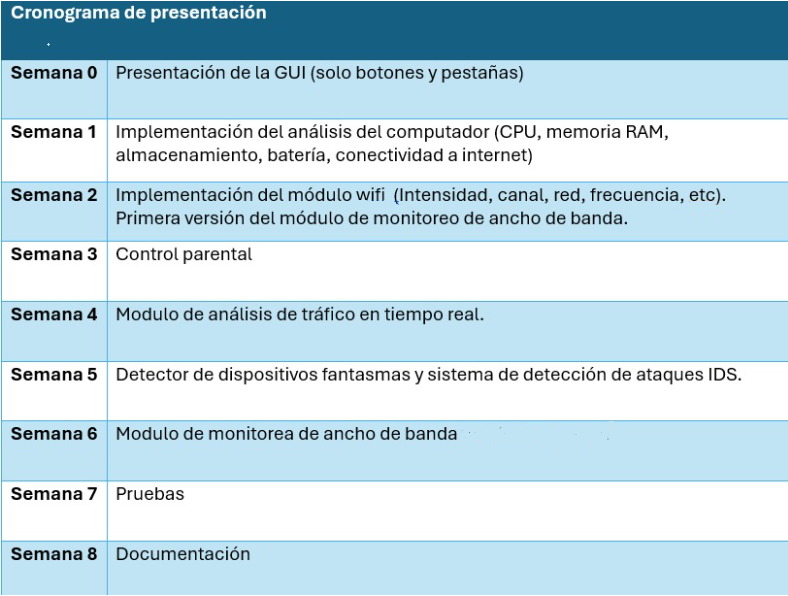
Eero Secure es un servicio de seguridad y control parental que se integra con los routers Eero, una línea de sistemas Wi-Fi de malla. Ofrece protección adicional para todos los dispositivos conectados a la red, sin necesidad de instalar software en cada uno.

* Bloqueo de contenido inapropiado
* Navegación más segura
* Control parental y programación
* Control parental y programación

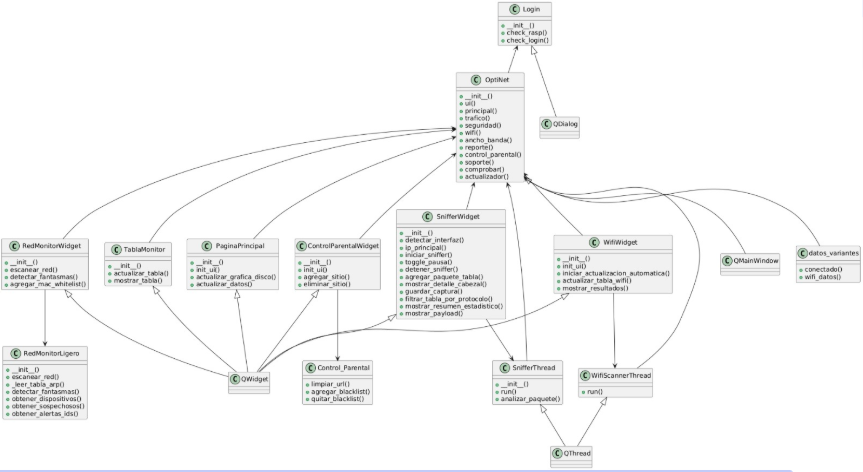
**Pfsense**

Si es así, pfSense es un sistema operativo de código abierto basado en FreeBSD que actúa como un potente firewall y router. Es muy usado en redes empresariales y domésticas avanzadas.

# **Cronograma**



# **Diagrama UML**

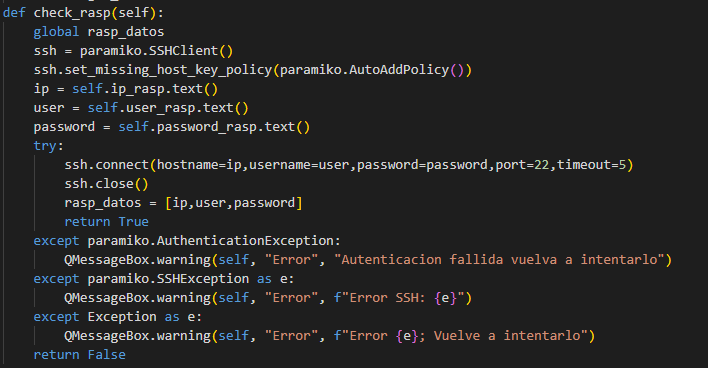


# **Clases y métodos**

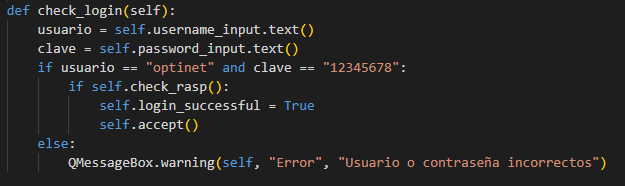
* **Login**

**Métodos:**

* **check\_rasp:**
* Usa paramiko para conectar vía SSH a la Raspberry con los datos ingresados.
* Si la conexión es exitosa, guarda las credenciales en rasp\_datos.



* **check\_login:**
* Verifica que el usuario local y contraseña coincidan
* Si pasa la validación, intenta la conexión SSH mediante check\_rasp.
* Si todo es correcto, activa la bandera login\_successful y cierra el diálogo con .accept().



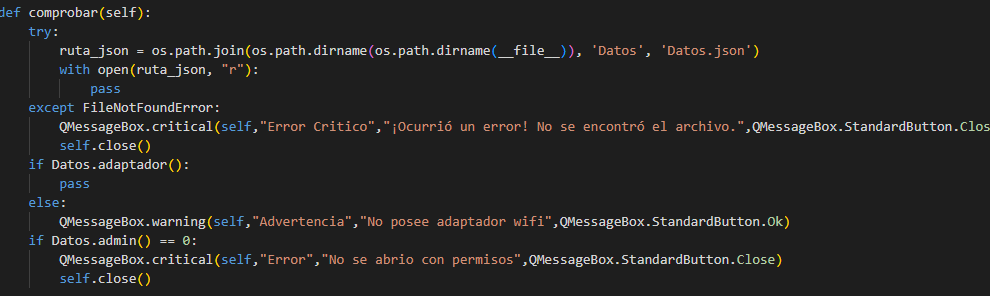
* **Optinet**

**Métodos:**

* + **ui():**
    - Menú superior con acciones (“Configuración” y “Ayuda”).
    - Barra lateral con botones para cambiar de módulo (Wi-Fi, Control Parental, etc.).
    - Panel central (QStackedWidget) que alterna entre widgets como SnifferWidget, WifiWidget, RedMonitorWidget, etc.
  + **principal (), tráfico (), seguridad (), wifi (),ancho\_banda(), reporte (), control\_parental(), soporte ():**
    - Cada uno devuelve el widget asociado a una sección específica de la aplicación



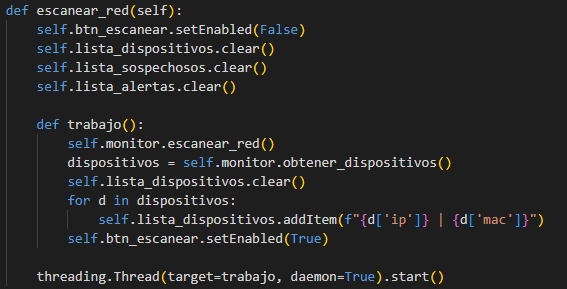
* + **comprobar ():**
    - Revisa la existencia de un archivo JSON, la presencia de un adaptador Wi-Fi y si el programa fue ejecutado con permisos de administrador. Muestra advertencias o errores si falla alguna condición.



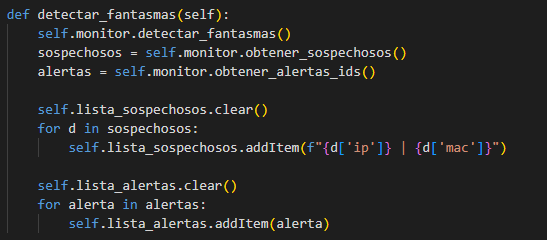
* + **actualizador ():**
    - Espacio reservado para funciones que podrían ejecutarse periódicamente o en segundo plano.
* **RedMonitorWidget**

**Métodos:**

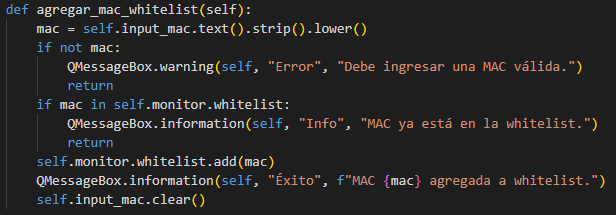
* + **escanear\_red():**
    - Lanza un hilo para ejecuta RedMonitorLigero.escanear\_red() y actualiza la lista de dispositivos detectados.



* + **detectar\_fantasmas():**
    - Llama a RedMonitorLigero.detectar\_fantasmas(), actualiza las listas de sospechosos y muestra alertas correspondientes.



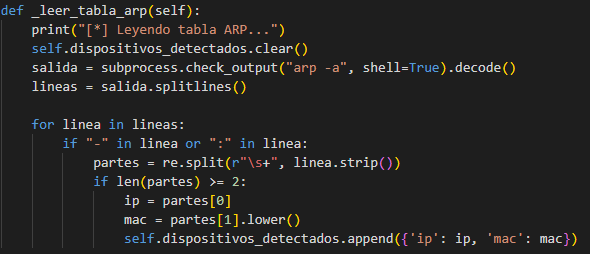
* + **Agregar\_mac\_whitelis():**
    - Añade una dirección MAC escrita manualmente a la whitelist para evitar que sea marcada como sospechosa.



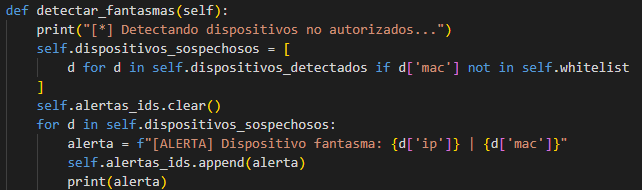
* **RedMonitorLigero**

**Métodos:**

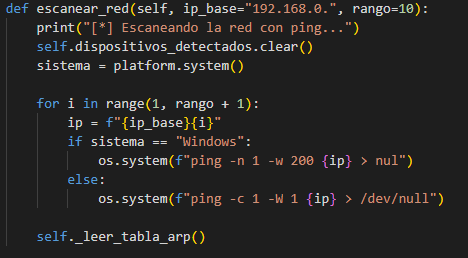
* + **\_leer\_tabla\_arp():**
    - Extrae las entradas actuales de la tabla ARP del sistema, guarda direcciones IP y MAC detectadas.



* + **detectar\_fantasmas():**
    - Compara los dispositivos detectados con la lista blanca y marca como “fantasmas” aquellos desconocidos. Genera alertas en texto.



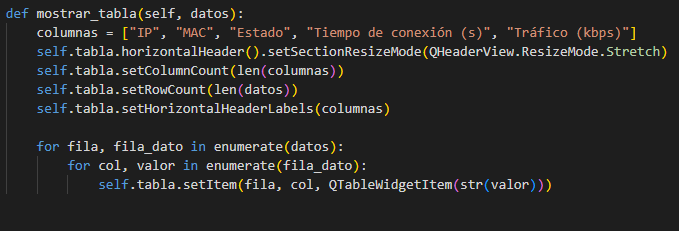
* + **obtener\_dispositivos():**
    - Devuelve todos los dispositivos reconocidos en la red.
  + **obtener\_sospechosos():**
    - Devuelve los dispositivos no autorizados.
  + **obtener\_alertas\_ids():**
    - Devuelve la lista de alertas generadas por dispositivos fantasmas.
  + **escanear\_red():**
    - Realiza pings a un rango de direcciones IP para poblar la tabla ARP del sistema operativo.



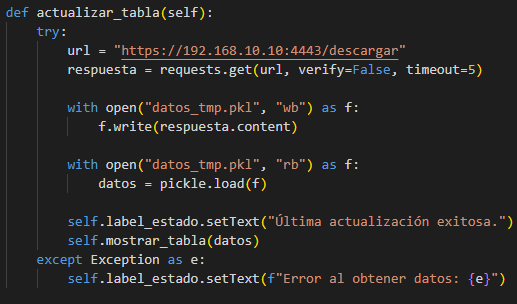
* **TablaMonitor**

**Métodos:**

* + **mostrar\_tabla():**
    - Define las columnas visibles de la tabla.
    - Ajusta el número de filas y columnas.
    - Llena la tabla con los valores extraídos de datos.



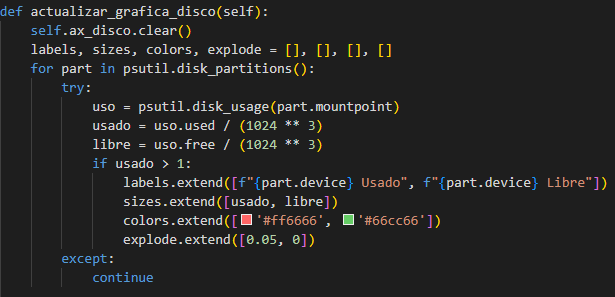
* + **actualizar\_tabla():**
    - Descarga datos desde una URL local (en formato pickle).
    - Guarda el archivo temporalmente y lo deserializa para obtener una estructura de datos.
    - Llama a mostrar\_tabla para actualizar la tabla con esa información.
    - Muestra mensajes de éxito o error en el label\_estado.



* **PaginaPrincipal**

**Métodos:**

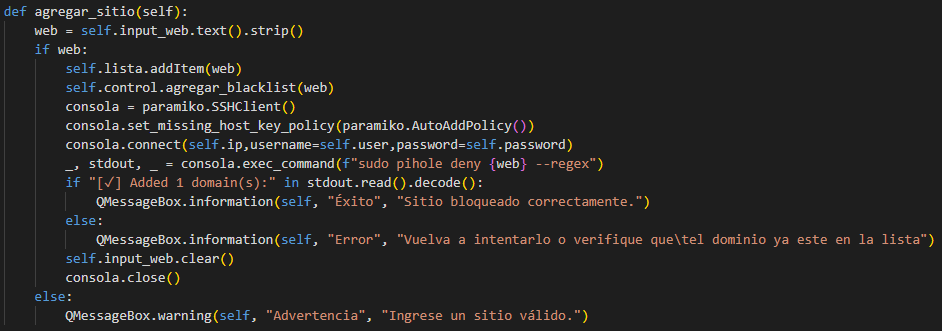
* + **actualizar\_grafica\_disco():**
    - Refresca gráficos y datos del sistema cada segundo



* + **actualizar\_datos():**
    - Muestra distribución de uso por particiones.
* **ControlParentalWidget**

**Métodos:**

* + **agregar\_sitio():**
    - Toma el texto ingresado, lo agrega a la lista.
    - Se conecta vía SSH al servidor y ejecuta el comando pihole deny.
    - Muestra un mensaje si fue exitoso o si ocurrió un error.



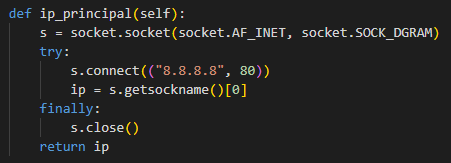
* + **eliminar\_sitio():**
    - Verifica qué sitios están seleccionados en la lista.
    - Ejecuta pihole --regex -d para eliminarlos desde Pi-hole.
    - Si la eliminación es exitosa, actualiza la lista local y muestra un mensaje.



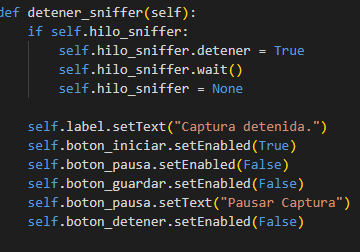
* + **QWidget**
* QWidget es la clase base fundamental para todos los elementos visuales en una interfaz gráfica usando PyQt
* **SnifferWidget**

**Métodos:**

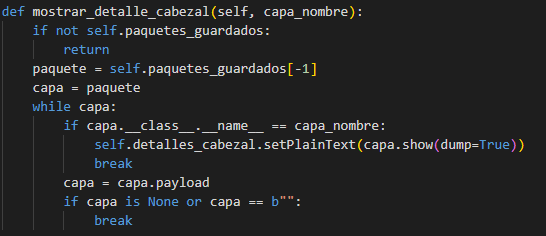
* + **ip\_principal():**
    - Crea un socket UDP (no se envían datos realmente).
    - Intenta conectarse al servidor 8.8.8.8 (Google DNS) en el puerto 80.
    - Extrae la IP asignada al socket, que corresponde a la interfaz de salida predeterminada.
    - Cierra el socket y devuelve esa IP.



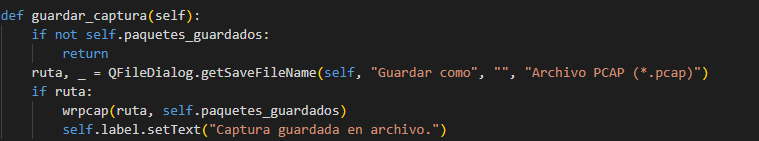
* + **iniciar\_sniffer():**
    - Crea y lanza un SnifferThread con la interfaz y filtro BPF seleccionados. Activa los botones de control y actualiza el estado.
  + **toggle\_pausa():**
    - Alterna el estado de pausa/reanudación del sniffer y actualiza el texto del botón
  + **detener\_sniffer():**
    - Finaliza el hilo de sniffing, desactiva botones y restaura el estado inicial



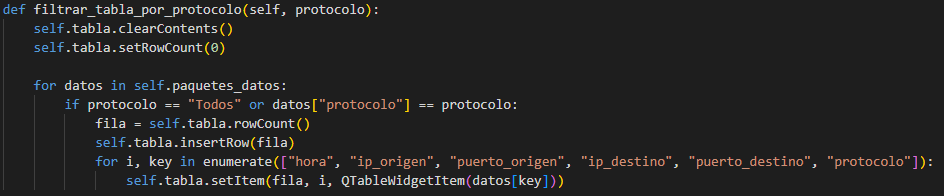
* + **agregar\_paquete\_tabla():**
    - Guarda el paquete en memoria para análisis.
    - Actualiza contadores por protocolo, IP de origen y destino.
    - Llena la tabla si el protocolo coincide con el filtro.
    - Extrae las capas del paquete para mostrarlas en la lista de cabezales.
    - Llama a mostrar\_detalle\_cabezal y mostrar\_payload para mostrar contenido detallado.
  + **mostrar\_detalle\_cabezal():**
    - Busca dentro del paquete la capa seleccionada y muestra su estructura.



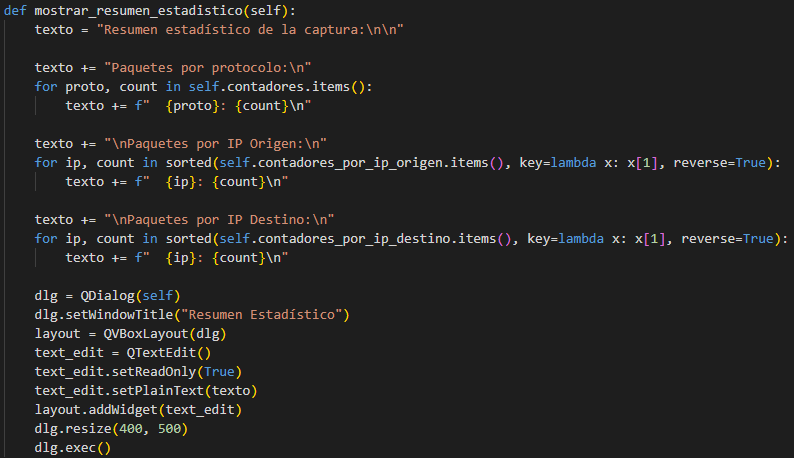
* + **guardar\_captura():**
    - Abre un diálogo para guardar los paquetes capturados como archivo .pcap usando wrpcap.



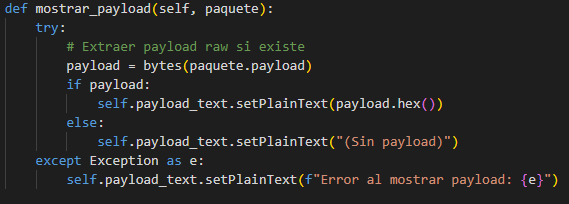
* + **filtrar\_tabla\_por\_protocolo():**
    - Recarga la tabla de visualización para mostrar solo los paquetes del protocolo elegido (TCP, UDP, etc.).



* + **mostrar\_resumen\_estadistico():**
    - Genera un resumen textual con conteos de paquetes por protocolo y por IP (origen y destino). Lo muestra en una ventana tipo QDialog.



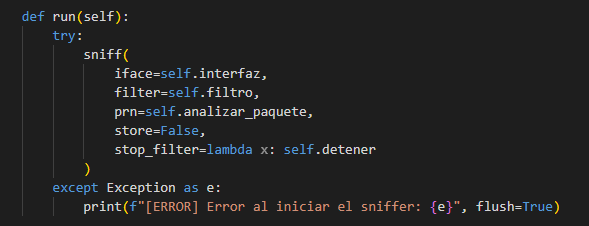
* + **mostrar\_payload():**
    - Extrae el contenido bruto (payload) del paquete y lo muestra en hexadecimal. Indica si no hay contenido o si ocurre algún error.



* **SnifferTread**

**Métodos**

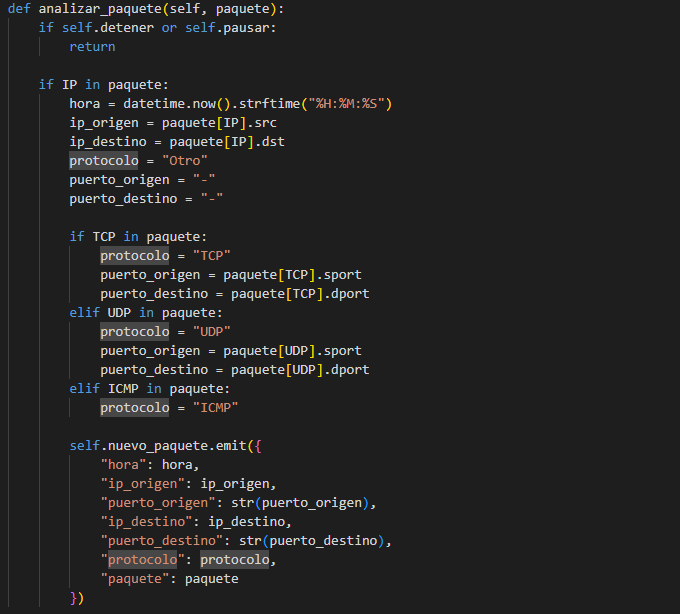
* + **run ():**
* Ejecuta el sniffer de paquetes con scapy.sniff() y detiene la captura cuando self.detener es True.



* + **analizar\_paquete():**

Procesa cada paquete capturado:

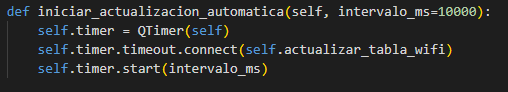
* Si contiene IP, extrae IPs y puertos.
* Detecta si es TCP, UDP, ICMP u otro.
* Emite señal nuevo\_paquete con un diccionario de datos clave del paquete (hora, origen/destino, protocolo, etc).



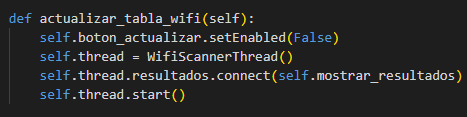
* **WifiWidget**

**Métodos**

* + **iniciar\_actualizacion\_automatica():**
    - Dispara una actualización automática cada cierto intervalo mediante un QTimer.



* + **actualizar\_tabla\_wifi():**
    - Lanza un hilo (WifiScannerThread) para escanear redes Wi-Fi y conecta los resultados al método mostrar\_resultados.



* + **mostrar\_resultados():**
    - Llena la tabla con los datos recibidos, actualiza estadísticas por banda y tipo de red, y actualiza el gráfico con matplotlib.
* **WifiScannerThread**

**Métodos**

* **Run ():**
* Llama a datos\_variantes.wifi\_datos() (que se asume extrae la información de redes disponibles).

