



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería  
Informática**

**Sniffer IV  
Documentación Técnica**



Presentado por Raúl Merinero Sanz  
en Universidad de Burgos — 22 de septiembre  
de 2022

Tutor: José Manuel Sáiz Diez



---

# Índice general

---

Índice general	i
Índice de figuras	iii
Índice de tablas	vi
<b>Apéndice A Plan de Proyecto Software</b>	<b>1</b>
A.1. Introducción . . . . .	1
A.2. Planificación temporal . . . . .	1
A.3. Estudio de viabilidad . . . . .	6
<b>Apéndice B Especificación de Requisitos</b>	<b>11</b>
B.1. Introducción . . . . .	11
B.2. Objetivos generales . . . . .	11
B.3. Catálogo de requisitos . . . . .	12
B.4. Especificación de requisitos . . . . .	17
<b>Apéndice C Especificación de diseño</b>	<b>27</b>
C.1. Introducción . . . . .	27
C.2. Diseño de datos . . . . .	27
C.3. Diseño procedimental . . . . .	28
C.4. Diseño arquitectónico . . . . .	33
<b>Apéndice D Documentación técnica de programación</b>	<b>59</b>
D.1. Introducción . . . . .	59
D.2. Estructura de directorios . . . . .	59
D.3. Manual del programador . . . . .	60

D.4. Compilación, instalación y ejecución del proyecto . . . . .	63
D.5. Comparativa JNI vs jNetPcap . . . . .	65
D.6. Mejora de la biblioteca jNetPcap . . . . .	65
D.7. Pruebas del sistema . . . . .	72
<b>Apéndice E Documentación de usuario</b>	<b>79</b>
E.1. Introducción . . . . .	79
E.2. Requisitos de usuarios . . . . .	79
E.3. Instalación . . . . .	80
E.4. Manual del usuario . . . . .	93
<b>Bibliografía</b>	<b>115</b>

---

# Índice de figuras

---

A.1. Diagrama de Gantt de la planificación final del proyecto. . . . .	5
C.1. Diagrama de secuencia del Menú Archivo . . . . .	28
C.2. Diagrama de secuencia del Menú Captura . . . . .	29
C.3. Diagrama de secuencia del Menú Definición . . . . .	30
C.4. Diagrama de secuencia del Menú Inserción . . . . .	31
C.5. Diagrama de secuencia del Menú Parametrización . . . . .	32
C.6. Diagrama de secuencia del Menú Ayuda . . . . .	33
C.7. Diagrama de paquetes general. . . . .	34
C.8. Paquete dominio . . . . .	35
C.9. Paquete dominio.export . . . . .	36
C.10.Paquete dominio.export.script . . . . .	36
C.11.Paquete dominio.export.xml_Pcaplib . . . . .	37
C.12.Paquete dominio.pcapDumper . . . . .	38
C.13.Paquete dominio.preferences . . . . .	39
C.14.Paquete dominio.preferences.capture . . . . .	40
C.15.Paquete dominio.preferences.definicion . . . . .	41
C.16.Paquete dominio.preferences.identificacion . . . . .	41
C.17.Paquete dominio.properties . . . . .	42
C.18.Paquete presentacion . . . . .	43
C.19.Paquete presentacion.avisos . . . . .	44
C.20.Paquete presentacion.capturando . . . . .	44
C.21.Paquete presentacion.comandos . . . . .	45
C.22.Paquete presentacion.preferencias . . . . .	46
C.23.Paquete presentacion.propiedadesVentana . . . . .	46
C.24.Paquete presentacion.seleccionFicheros . . . . .	47
C.25.Paquete presentacion.ventanaMenuSniffer . . . . .	48
C.26.Paquete presentacion.visualizarCaptura . . . . .	49

C.27.Paquete servicioAccesoDatos . . . . .	50
C.28.Diagrama de clases general. . . . .	51
C.29.Diagrama de clases para la captura. . . . .	52
C.30.Diagrama de clases para la captura desde fichero. . . . .	53
C.31.Diagrama de clases para la exportación. . . . .	54
C.32.Diagrama de clases para la definición. . . . .	55
C.33.Diagrama de clases para la inserción de paquetes capturados. . . . .	56
C.34.Diagrama de clases para la inserción de paquetes definidos. . . . .	57
D.1. Instalación de la aplicación Eclipse. . . . .	64
D.2. Variable de entorno de Ant. . . . .	64
D.3. Primeros campos del protocolo DNS. . . . .	68
E.1. Instalación JDK 1. . . . .	81
E.2. Instalación JDK 2. . . . .	82
E.3. Instalación JDK 3. . . . .	82
E.4. Variables de entorno 1. . . . .	83
E.5. Variables de entorno 2. . . . .	84
E.6. Variables de entorno 3. . . . .	85
E.7. Instalación de Wireshark 1. . . . .	87
E.8. Instalación de Wireshark 2. . . . .	87
E.9. Instalación de Wireshark 3. . . . .	88
E.10.Instalación de Wireshark 4. . . . .	89
E.11.Menú inicial del Sniffer. . . . .	94
E.12.Interfaz gráfica del Sniffer. . . . .	94
E.13.Archivo ->Abrir fichero de capturas. . . . .	95
E.14.Abrir fichero de capturas. . . . .	95
E.15.Fichero de capturas abierto. . . . .	96
E.16.Campos posibles en la tabla. . . . .	96
E.17.Detallado de paquetes. . . . .	97
E.18.Archivo ->Guardar fichero. . . . .	98
E.19.Archivo ->Exportar ->Captura a XML. . . . .	98
E.20.Archivo ->Exportar ->Captura a XML. . . . .	99
E.21.Archivo ->Exportar ->desde fichero a XML. . . . .	99
E.22.Exportar desde fichero a XML. . . . .	100
E.23.Archivo ->Configuración. . . . .	100
E.24.Configuración. . . . .	101
E.25.Captura ->Inicio captura. . . . .	101
E.26.Inicio captura. . . . .	102
E.27.Pantalla de captura. . . . .	103
E.28.Capturar ->Captura desde fichero. . . . .	104

E.29.Captura desde fichero. . . . .	104
E.30.Definición ->Definición de paquetes. . . . .	105
E.31.Definición de paquetes. . . . .	105
E.32.Inserción ->Inserción de paquetes definidos. . . . .	106
E.33.Inserción de paquetes definidos. . . . .	106
E.34.Inserción ->Inserción de paquetes capturados. . . . .	107
E.35.Inserción de paquetes capturados. . . . .	108
E.36.Parametrización ->Captura. . . . .	108
E.37.Preferencias de captura. . . . .	109
E.38.Parametrización ->Exportar. . . . .	110
E.39.Preferencias de exportación. . . . .	110
E.40.Parametrización ->Captura desde fichero. . . . .	111
E.41.Preferencias de captura desde fichero. . . . .	111
E.42.Parametrización ->Detalle paquetes. . . . .	112
E.43.Preferencias del detallado de paquetes. . . . .	112
E.44.Parametrización ->Generar script. . . . .	113
E.45.Preferencias de generar script. . . . .	113
E.46.Ayuda ->Contenidos. . . . .	114

---

# Índice de tablas

---

A.1.	Pruebas multiplataforma realizadas. . . . .	3
A.2.	Costes de personal. . . . .	7
A.3.	Costes de <i>hardware</i> . . . . .	7
A.4.	Otros costes. . . . .	7
A.5.	Otros costes. . . . .	8
A.6.	Simulación de beneficios. . . . .	8
A.7.	Pruebas multiplataforma realizadas. . . . .	9
B.1.	CU-001 Menú Inicial. . . . .	18
B.2.	CU-002 Archivo. . . . .	19
B.3.	CU-003 Captura. . . . .	20
B.4.	CU-004 Definición. . . . .	21
B.5.	CU-005 Inserción. . . . .	22
B.6.	CU-006 Parametrización. (Modificado por el alumno) . . . . .	23
B.7.	CU-007 Ayuda. . . . .	24
B.8.	CU-008 Modo Comando. . . . .	24
B.9.	CU-009 Configurar VMJava. . . . .	25
B.10.	CU-010 Salir. . . . .	25
D.1.	Pruebas del sistema realizadas. . . . .	77
D.2.	Pruebas multiplataforma realizadas. . . . .	78

## *Apéndice A*

---

# **Plan de Proyecto Software**

---

## **A.1. Introducción**

La planificación del proyecto está formada por una planificación temporal y un estudio de viabilidad. El primero describe el tiempo empleado a la consecución de los objetivos, mientras que el segundo sirve para comprobar si la realización del proyecto es factible.

## **A.2. Planificación temporal**

En este apartado, se recoge el orden seguido para la realización de todos los pasos necesarios para alcanzar los objetivos establecidos. Además, se recoge un diagrama con la planificación resultante.

### **1. Planificación inicial**

En esta sección, se mencionan uno a uno los pasos realizados durante el desarrollo del proyecto.

#### **Observación inicial**

En un primer momento, se estudian los proyectos anteriores y su documentación correspondiente, con el fin de comprender la aplicación al completo y las herramientas que se deben utilizar. Uno de los aspectos más importantes es la comprensión de la estructura del proyecto, es decir, la forma en la que están organizados los paquetes y sus clases, en torno a las diferentes funcionalidades de la aplicación.

## Estudio de las bibliotecas utilizadas

Se realiza un estudio de la biblioteca actual, se comprueba si sigue siendo desarrollada y se contemplan otras posibles opciones del mercado.

Tras un largo análisis, entre las diversas opciones tenemos:

- **jNetPcap.** *JnetPcap* es un *Java wrapper*, de código abierto, de las bibliotecas *libPcap* y *winPcap*. Esto quiere decir que es una biblioteca Java que actúa como contenedor para las llamadas nativas de las bibliotecas *libPcap* y *winPcap*, escritas en lenguaje C.

Actualmente, por desgracia, se encuentra abandonada. Este es el principal problema de esta biblioteca, ya que es la que usa el *Sniffer*. Aún así, la posibilidad de mejorar la biblioteca, puede permitir la inclusión de nuevos protocolos que puedan ser útiles para nuestra aplicación.

- **libPcap y winPcap con JNI.** JNI es un framework de programación que permite que programas escritos en lenguajes como C o C++, interactúen con programas escritos en lenguaje Java, y así ser ejecutados por la máquina virtual Java. Es una programación compleja, pero aún así se plantea probar esta opción.

Por otro lado, *libPcap* y *winPcap* son unas bibliotecas escritas en lenguaje C, diseñadas para facilitar el acceso a capas de red de bajo nivel. Estas son las que permiten la captura y el envío de paquetes que realiza el *Sniffer*.

JNI requiere una programación compleja, aunque permitiría estructurar nuestra aplicación como mejor le convenga.

- **nPcap.** *Npcap* es una nueva biblioteca que ha surgido como una versión mejorada de *winPcap*, y que actualmente se encuentra en desarrollo. Por lo tanto, esta opción también necesitaría JNI para la interacción con el *Sniffer*, así como *libPcap* para su funcionamiento en sistemas operativos Linux, dado que *nPcap* solo sirve para Windows, de la misma manera que *winPcap*.

Esta biblioteca tiene la desventaja de ser desconocida, así como la empresa desarrolladora, aunque parece ser comúnmente usada como relevo de *winPcap*.

### Pruebas multiplataforma

Se instalan diversas máquinas virtuales, con el gestor VirtualBox [15], y se configuran al completo para probar la aplicación. Para ello, se debe seguir el manual de instalación en todas ellas.

En la siguiente tabla, recogemos las pruebas realizadas de una forma más concreta:

Sistema Operativo	Tipo de arquitectura	Versión
Windows	x64	Windows 10 Home
Windows	x32	Windows 10 Home
Linux	x64	Ubuntu 22.04 LTS
Linux	x32	Ubuntu 17.04

Tabla A.1: Pruebas multiplataforma realizadas.

Las máquinas virtuales de Windows, sobre todo la de arquitectura x64, necesitan de una mayor memoria. Se han tenido que crear varias hasta dar con un valor de memoria válido.

Para cada arquitectura con el sistema operativo Windows, se necesita una biblioteca DLL concreta. Esto llega a dar bastantes problemas, dado que la máquina virtual de Java debe ser acorde a la arquitectura del sistema.

En cuanto a Ubuntu, son versiones muy diferentes, ya que la 17.04 es la última encontrada para la arquitectura x32.

En este paso, también se actualizan las bibliotecas, con lo que vuelven a realizarse las pruebas anteriores, tanto para Windows como para Linux, con arquitecturas x64 y x32, aunque estas requieren menos tiempo, dado que las máquinas virtuales ya están configuradas.

### Pruebas de funcionalidad

Se realizan las pruebas necesarias para probar todas las funcionalidades de la aplicación.

En un principio, se opta por realizar pruebas de humo, con el fin de asegurar que no hay errores graves que detengan el funcionamiento de la aplicación.

Posteriormente, se realizan varias pruebas de caja negra. Estas buscan comprobar el correcto funcionamiento de las diferentes funcionalidades de la

aplicación. Por ejemplo, se realizan pruebas para comprobar que las capturas se realizan correctamente, y que los ficheros creados pueden ser leídos y mostrar los paquetes capturados en ellos.

### Creación de un entorno de trabajo

Teniendo como base una de las máquinas virtuales creadas para la realización de las pruebas multiplataforma, se instala el entorno de desarrollo para la implementación de código que mejore la aplicación.

El pilar principal de este entorno será la aplicación *Eclipse*. Esta se ha utilizado en proyectos anteriores, contiene un gran número de conjuntos de herramientas útiles para la programación en Java, lenguaje en el que está escrita la aplicación, y es intuitiva y fácil de usar.

### Implementación de mejoras

Se realizan una serie de mejoras, decididas con la ayuda del tutor. Estas, más concretamente, son las siguientes:

- Incluir la posibilidad de ordenar la tabla de paquetes por los diferentes campos.
- Modificar la visualización del detallado de paquetes para poder escoger cuántos paquetes ver a la vez, y en qué distribución.
- Permitir visualizar los primeros bytes, o todos, de cada paquete, tanto en formato haxadecimal como en decimal.
- Parametrizar los cambios realizados relativos al detallado de paquetes.
- Mejorar el protocolo HTTP, ya implementado en la biblioteca *jNetPcap*, pero no incluido en la aplicación.
- Parametrizar el guardado en XML al realizar una captura, reduciendo así los recursos que utiliza la aplicación, ya que previamente había un bucle de código en el que se realizaba la escritura de la captura en ficheros reiteradas veces.
- Calcular y mostrar la longitud de la cabecera de los paquetes, que tiene un campo reservado en la tabla de paquetes.
- Mostrar paquetes en la tabla de paquetes durante la captura.
- Mejorar la biblioteca JnetPcap para la implementación de 3 nuevos protocolos: DNS, ICMPv6 e IGMP.

### Establecer Entorno Virtual

Se crea un entorno virtual para poder probar la aplicación sin necesidad de realizar todo el proceso de instalación. Se usa también el gestor VirtualBox, y se utiliza el sistema operativo Windows 10 Home, con una arquitectura x64.

### Documentación

Se crea la documentación del proyecto, siendo esta en ocasiones una versión actualizada de la documentación de proyectos previos. Este proceso se ha ido haciendo a lo largo de la implementación de mejoras.

## 2. Planificación final

Como resultado del desarrollo del proyecto, hemos obtenido una planificación no muy óptima, pero suficiente para alcanzar los objetivos propuestos.

### Diagrama de Gantt

La planificación resultante se muestra en el siguiente diagrama, realizado en la página web de Online Gantt [6]:



Figura A.1: Diagrama de Gantt de la planificación final del proyecto.

Como se puede ver, se dedicó mucho tiempo a la comprensión de la aplicación y de los proyectos anteriores, así como al estudio de las posibles bibliotecas y a las pruebas, tanto multiplataforma como de funcionalidad.

La creación de un entorno de trabajo fue un proceso relativamente sencillo, pues ya se habían realizado casi todos los pasos necesarios para la realización de las pruebas multiplataforma.

En cuanto a la implementación de mejoras, ha resultado ser un proceso muy largo, en el que cabe destacar que se descansaron un par de semanas en

agosto. Además, este periodo incluye los problemas con el DLL de *jNetPcap*, que finalmente no ha sido necesario.

La creación del entorno virtual ha sido rápida, mientras que la documentación se procuró hacer a la par que la implementación de mejoras, aunque empezando un tiempo más tarde.

### A.3. Estudio de viabilidad

Como se ha mencionado en la introducción, en este apartado comprobaremos si la realización del proyecto es factible. Se tendrá en cuenta el tiempo empleado y los diversos beneficios que supondría, ya sean económicos como el resto que puedan tener repercusión económica en un futuro.

La viabilidad de un proyecto que es aparentemente viable, puede cambiar después de analizar en profundidad los problemas y oportunidades que tiene el proyecto o después de desarrollar el sistema.

#### Viabilidad económica

Comenzamos realizando un estudio de la viabilidad económica del proyecto, analizando los costes y beneficios que habría supuesto el desarrollo del proyecto en un entorno de trabajo real, en España.

##### 1. Análisis de Costes

Primero de todo, cabe destacar que esto solo representa una mejora de la aplicación, por lo que no se contabilizan los costes de las versiones anteriores, que en algunos casos han llegado a ser de más de 15.000€.

Los cálculos se realizarán para los meses dedicados a la realización del proyecto, que han sido 5, aproximadamente. Se descuenta un mes, buscando obtener unos valores más realistas, dado que hubo dos semanas de pausa y encontramos un “callejón sin salida” en la búsqueda de la construcción del DLL de *jNetPcap*.

##### **Costes de Personal.**

El proyecto será realizado por un único desarrollador, que se ocupará de todas las tareas mencionadas anteriormente.

En cuanto a las horas trabajadas diarias, en alguna temporada se emplearon hasta 12 horas diarias, pero no fue de continuo durante todo el proyecto. De esta manera, lo dejaremos en una media de 8 horas diarias.

Así, estableceremos el salario promedio de un programador junior en España en 1082,20 €, según Indeed [10].

Concepto	Desarrollador (€)
Salario total neto	1082,20
Seguridad Social	339,81
<b>Total (5 meses)</b>	<b>7.110,05</b>

Tabla A.2: Costes de personal.

### Costes de Hardware.

Se utiliza un ordenador portátil MSI Modern 14 B11MOU-689XES, con un procesador i5-1155G7, una memoria RAM de 16GB y un disco SSD de 512GB. Se considera una vida media de 3 años.

Concepto	Coste (€)	Coste Amortizado (€)
MSI Modern 14	699,99	19,44
<b>Total (5 meses)</b>	<b>699,99</b>	<b>97,20</b>

Tabla A.3: Costes de hardware.

### Costes de Software.

Todos los programas o aplicaciones utilizados son gratuitos o tienen una licencia de estudiante gratuita, por lo que se omite el cálculo. De la misma manera, también se omite el coste amortizado.

### Otros Costes.

Existen algunos costes más, que mostramos en la siguiente tabla.

Concepto	Coste (€)
Internet	27
Electricidad	15
<b>Total (5 meses)</b>	<b>210</b>

Tabla A.4: Otros costes.

### Costes Totales.

Con todos los costes analizados, solo nos falta recogerlos en una tabla, calculando así los totales.

Concepto	Coste (€)
Coste de personal	7.110,05
Coste de hardware	97,20
Coste de software	0
Otros costes	210
<b>Total (5 meses)</b>	<b>7.417,25</b>

Tabla A.5: Otros costes.

## 2. Análisis de Beneficios

Dado que la aplicación contiene soporte para varios idiomas, está puede ser comercializada a nivel internacional.

Por otra parte, las funcionalidades van más allá de un simple capturador del tráfico de la red. Esto también nos podría ayudar a la hora de sacarla al mercado.

Con esos dos factores a destacar, se realiza una simulación para recuperar la inversión realizada.

Tipo	Número de cuentas	Beneficios (€)
Versión de prueba	0	0
Estudiante <sup>1</sup>	5.000	10.000
Empresa	10.000	30.000
<b>Total</b>		<b>40.000</b>

Tabla A.6: Simulación de beneficios.

Se considera la posibilidad de que empresas puedan solicitar alguna característica concreta, con lo que se establece el precio/hora en 60€. El presupuesto vendrá dado por las horas que sean necesarias para conseguir esa/s característica/s.

<sup>1</sup>Los beneficios que en la tabla vienen como “Estudiante”, se refieren a los contratos realizados con las universidades.

## Viabilidad legal

Se ha utilizado únicamente software de licencia libre, con el fin de reducir al máximo el coste de desarrollo de la aplicación y maximizar los posibles beneficios futuros. La única licencia no libre es la del sistema operativo del portátil empleado por el alumno, que siendo este Windows 10 puede rondar los 145€, pero que venía incluida en la compra del dispositivo.

Para una mayor información, recogemos el software utilizado en la siguiente tabla:

Aplicación	Versión	Licencia
VirtualBox	6.1.34	Privativa / GPLv2
Eclipse	2022-06	EPL
Gedit	(por defecto)	GPL
Ant	(por defecto)	EPL (plug-in)
WinRAR	6.10	(copia de evaluación)
TexMaker	5.1.3	GNU GPL
Visual Studio	17.2	Community

Tabla A.7: Pruebas multiplataforma realizadas.

La licencia de nuestra aplicación, por tanto, será GNU LGPL v3. El objetivo es que los usuarios experimentados puedan modificar las funcionalidades de la aplicación, de forma privada, pero debiendo ser correctamente comprobadas por los autores, para utilizarla de forma comercial.



## *Apéndice B*

---

# **Especificación de Requisitos**

---

## **B.1. Introducción**

Este documento recoge los requisitos mínimos que debe cumplir la aplicación tras la mejora realizada por el alumno. También contiene aquellos necesarios para el funcionamiento de la aplicación propios del sistema.

## **B.2. Objetivos generales**

En esta sección, se explican los diferentes objetivos que se quieren alcanzar con el desarrollo del proyecto.

1. Realizar un estudio de las bibliotecas que pueden ser usadas para la codificación de la aplicación. Estas son las siguientes: *jNetPcap*, *winPcap* y *libPcap* con JNI, y *nPcap* y *libPcap* con JNI.
2. Probar el correcto funcionamiento de la aplicación en Windows y Linux, con ambas arquitecturas, x64 y x32, montando las correspondientes máquinas virtuales sobre las que realizar las pruebas.
3. Actualizar el código obsoleto. Los métodos se marcan como obsoletos cuando se les encuentran fallos de programación graves, que pueden conducir a errores. Por tanto, en la documentación online de las diferentes bibliotecas, se suelen dejar indicados las posibles formas de sustituirlos, pues no es seguro ignorarlos.
4. Realizar mejoras en la aplicación con fines de utilidad. Las mejoras concretas a realizar son:

- Implementar la posibilidad de ordenar la tabla de paquetes por los diferentes campos.
  - Calcular y mostrar la longitud de la cabecera de los paquetes, campo que se encuentra vacío en el panel de la tabla de paquetes.
  - Modificar la visualización del detallado de paquetes para poder escoger cuántos paquetes ver a la vez, y en qué distribución. La modificación de estos valores hace que el panel del detallado de paquetes se actualice automáticamente, y permite almacenar los valores introducidos en el fichero de preferencias indicado.
  - Permitir la visualización de los primeros bytes, o de todos, de los paquetes, tanto en formato hexadecimal, como en formato decimal. Esto se puede parametrizar para que al iniciar la aplicación se muestren los ajustes que prefiera el usuario.
  - Parametrizar el guardado en XML al realizar una captura, ya que la exportación a este formato es un proceso largo, y reducir si es posible la cantidad de recursos que necesita la captura de tráfico.
  - Mostrar los paquetes capturados, mientras están siendo capturados, en el panel de la tabla de paquetes. Sin esta funcionalidad, la realización de capturas pierde parte de su utilidad, ya que para ver la información de los protocolos capturados, se necesita esperar a detener la captura, o a que se detenga automáticamente, en función de los filtros previamente establecidos por el usuario.
5. Implementar el analizador del protocolo HTTP con el fin de conocer esa parte de la aplicación. Esto es porque, si bien en la biblioteca utilizada existe el protocolo, en la aplicación no se había creado un analizador, por lo que esta no lo reconocía.
  6. Modificar la biblioteca usada e incluir los protocolos DNS, ICMPv6 e IGMP. Esto requiere analizar las diferentes implementaciones de los protocolos que ya contiene la biblioteca. De esta forma, la programación se hace más sencilla, pudiendo seguir su mismo estilo.

### **B.3. Catálogo de requisitos**

En este apartado, se explican los diferentes requisitos de la aplicación, tanto funcionales, como no funcionales.

## Requisitos funcionales

Dado que este proyecto es una mejora de otros muchos anteriores, algunos de los requisitos se han tomado de ellos, aspecto que quedará indicado en cada uno de los requisitos que encontramos en la siguiente tabla. Aún así, ante los pocos casos de uso que cubren estos requisitos, se decide ampliar la lista, siendo más específico en algunos casos.

- **RF-1 Menú inicial.** La aplicación debe mostrar un menú al ser iniciada, que permita escoger el modo de funcionamiento y la memoria utilizada, junto con una opción para cerrar la aplicación.
- **RF-2 Interfaz gráfica.** La aplicación debe permitir ser ejecutada con una interfaz gráfica, con el objetivo de facilitar su uso.
  - **RF-2.1 Acceso al Menú Archivo.** Este menú permite acceder a las funciones que nos permiten gestionar ficheros. (Sniffer-II)
    - **RF-2.1.1 Visualización de capturas.** La aplicación debe ser capaz de cargar ficheros de captura (extensión .pcap) y mostrar los paquetes capturados en los diferentes paneles del *layout*.
  - **RF-2.2 Acceso al Menú Captura.** Este menú permite visualizar y posteriormente ejecutar algunas sub-aplicaciones que nos van a dar la funcionalidad de realizar capturas, bien a través del adaptador de red del equipo o bien leyendo ficheros de capturas previas. (Sniffer-II)
    - **RF-2.2.1 Captura de paquetes.** La aplicación debe ser capaz de capturar el tráfico de la red.
      - ◊ **RF-2.2.1.1 Mostrar número de paquetes capturados.** El usuario debe poder ver el número de paquetes capturados, correspondientes a cada protocolo implementado, en la propia ventana de captura.
      - ◊ **RF-2.2.1.2 Mostrar impresión por pantalla de los paquetes capturados.** El usuario debe poder visualizar la impresión por pantalla de los paquetes capturados, en la propia ventana de captura.
      - ◊ **RF-2.2.1.3 Mostrar paquetes capturados en la tabla de paquetes simultáneamente.** El usuario debe poder ver los paquetes que están siendo capturados en la tabla de paquetes de la ventana principal de la aplicación. Así mismo, debe poder desplegar toda su información.

- ◊ **RF-2.2.1.4 Escoger una serie de filtros.** El usuario debe poder dar valores a los diferentes filtros implementados.
- **RF-2.3 Acceso al Menú Defición.** Aquí se pueden definir estructuras de paquetes. (Sniffer-II)
  - **RF-2.3.1 Definición de paquetes.** La aplicación debe ser capaz de definir nuevas estructuras de datos, con distintos campos.
- **RF-2.4 Acceso al Menú Inserción.** Aquí se puede devolverá a su origen paquetes previamente definidos. (Sniffer-II)
  - **RF-2.4.1 Inserción de paquetes.** La aplicación debe ser capaz de insertar paquetes previamente definidos o capturados en la red.
- **RF-2.5 Acceso al Menú Parametrización.** Como una de las funciones principales de la aplicación será la de poder guardar parametrizaciones y ficheros de configuración para posteriormente poder reutilizarlos, deberá existir un submenú para poder acceder a las opciones más comúnmente utilizadas. (Sniffer-II)
- **RF-2.6 Acceso al Menú Ayuda.** A través de esta opción, conseguiremos acceder a la ayuda del programa. (Sniffer-II)
- **RF-2.7 Opción de salir de la aplicación.** Se confirmará la salida del programa y la aplicación terminará. (Sniffer-II)
- **RF-2.8 Panel de tabla de paquetes.** La aplicación deberá tener un panel en el que se muestre una lista de paquetes con una serie de campos relevantes, como las direcciones IP y MAC de origen y destino, y el tamaño del paquete. El usuario debe poder escoger, exactamente, qué campos quiere que se muestren y en qué orden.
- **RF-2.9 Panel de conexiones TCP/IP.** La aplicación deberá tener un panel en el que se muestre una lista de las diferentes conexiones TCP/IP capturadas, indicando el número de paquetes que intervienen en cada una y las dos direcciones IP involucradas.
- **RF-2.10 Panel de detallado de paquetes.** La aplicación deberá tener un panel en el que se muestre la información detallada de los paquetes que se seleccionen en el panel de la tabla de paquetes.
  - **RF-2.10.1 Mostrar información detallada de los protocolos.** El usuario debe poder ver la información detallada

de los protocolos presentes en el paquete, pudiendo desplegar cada uno de ellos con el fin de observar los datos que recoge cada uno, así como mostrar los datos del paquete en representación decimal o hexadecimal.

- **RF-3 Modo comando.** La aplicación debe permitir ser utilizada en modo comando, para aquellos usuarios expertos que prefieran esta opción con intención de ahorrar tiempo.

A continuación, se muestran unos requisitos funcionales mucho más concretos, que surgen de los anteriores, y que son los que han sido modificados en este proyecto, principalmente.

- **RF-2.2.1.5 Filtro de guardado en XML en captura.** Las capturas anteriormente se guardaban tanto en un fichero con formato “.pcap” como en otro de formato XML. La exportación a XML es un proceso lento, por lo que debe haber un filtro donde el usuario elija si quiere que esto se realice, o no.
- **RF-2.2.1.6 Paquetes recién capturados a tabla de paquetes.** Los paquetes capturados deben aparecer en la tabla de paquetes durante la captura, y no cuando esta finaliza.
- **RF-2.5.1 Preferencias de captura.** Los diferentes filtros de las capturas deben poder ser parametrizados, siendo uno de los filtros el guardado en XML.
- **RF-2.5.2 Preferencias del detallado de paquetes.** Los diferentes filtros relativos al panel del detallado de paquetes deben poder ser parametrizados. Estos son el diseño del panel (número de paquetes que se muestran y distribución), la cantidad de bytes que muestran del paquete y el formato en que se muestran esos bytes.
- **RF-2.8.1 Ordenar tabla por diferentes campos.** La tabla de paquetes mostrada en el primer panel deberá permitir la ordenación de los paquetes en función de los diferentes campos. Por ejemplo, los paquetes deberán poder ser ordenados por su identificador, es decir, según el orden de captura.
- **RF-2.8.2 Mostrar el tamaño de la cabecera del paquete.** Uno de los campos en la tabla de paquetes, debe ser el tamaño de la cabecera del paquete.

- **RF-2.10.1 Modificar el panel de detallado de paquetes.** El panel de detallado de paquetes previamente tenía un diseño fijo. Ahora, este debe poder ser definido por el usuario, pudiendo guardar sus ajustes en un fichero de preferencias.
- **RF-2.10.2 Mostrar los primeros bytes de los paquetes.** A la hora de mostrar la información detallada de los paquetes, se debe mostrar los primeros bytes, o todos, de los paquetes seleccionados, tanto en formato hexadecimal como en decimal. Estos ajustes también deben poder guardarse en un fichero de preferencias.

## Requisitos no funcionales

Los requisitos no funcionales son aquellos que ayudan a juzgar el funcionamiento de una aplicación. En concreto, para la nuestra, recopilamos los de proyectos anteriores y añadimos alguno nuevo, obteniendo los siguientes:

- **RNF-1 Usabilidad.** La interfaz de la aplicación debe ser intuitiva y fácil de aprender a utilizar.
- **RNF-2 Rendimiento.** La aplicación debe funcionar correctamente, evitando quedarse parada. Por ejemplo, la captura de tráfico es una de las funcionalidades que más recursos utiliza, por lo que se busca no complicar el proceso, por si acaso la aplicación deja de responder durante su ejecución o al terminarla.
- **RNF-3 Entorno de exploración.** La aplicación debe poderse ejecutar tanto en Windows como en Linux.
- **RNF-4 Entorno de explotación.** Para ejecutar la aplicación será necesario un ordenador que disponga del entorno de ejecución (JRE) Java de versión 1.8.0 o superior y WinPcap instalado en el caso de tratarse de un sistema Windows. (Proyecto anterior)
- **RNF-5 Espacio en disco.** Para la creación de un proyecto se necesitará escribir en disco por lo que será necesario un medio de almacenamiento como un disco duro o memoria extraíble tipo USB. La captura consume espacio en disco por lo que deberán controlarse las mismas borrando las que no nos interesen (borrando la carpeta que contenga dichos ficheros). (Proyecto anterior)
- **RNF-6 Actualización.** Los diferentes elementos software utilizados deben ser actualizados con respecto a proyectos anteriores.

## **B.4. Especificación de requisitos**

Al ser una aplicación tan compleja, se agruparán todas las funcionalidades en casos de uso más generales. La mayoría de los casos de uso se corresponden con los realizados en otros proyectos, pero algunos de ellos han sido modificados por el alumno. Esto se especificará en la descripción de las tablas, encontrada debajo de cada una de ellas.

<b>CU-001</b>	<b>Menú Inicial</b>
<b>Descripción</b>	Accede al Menú Inicial.
<b>Precondición</b>	Haber ejecutado snifferIV.bat o sniffer-IV.sh en función del S.O.
<b>Secuencia normal</b>	<p>1. Se muestra la pantalla inicial.</p> <p>2. Se escoge una de las opciones del menú.</p> <ul style="list-style-type: none"> <li>a) “Modo Gráfico” inicia la aplicación en un entorno gráfico.</li> <li>b) “Modo Comando” inicia la aplicación en modo comando.</li> <li>c) “Configurar MVJava” Permite seleccionar la cantidad de memoria virtual a utilizar por la aplicación.</li> <li>d) “Salir” Detiene la ejecución de la aplicación.</li> </ul>
<b>Postcondición</b>	Introducir correctamente el número correspondiente a cada opción.
<b>Excepciones</b>	No existen excepciones posibles.
<b>Importancia</b>	Alta

Tabla B.1: CU-001 Menú Inicial.

CU-002	Archivo
<b>Descripción</b>	Tratamiento de ficheros.
<b>Precondición</b>	Haber seleccionado “Archivo” en la interfaz gráfica.
<b>Secuencia normal</b>	<p>1. Se muestra un desplegable.</p> <ul style="list-style-type: none"> <li>a) “Abrir fichero de capturas” muestra una pantalla donde seleccionar el fichero que se deseé abrir.</li> <li>b) “Guardar fichero de capturas” muestra una ventana donde se puede guardar el fichero capturado en la ruta que se escoja.</li> <li>c) “Exportar” Se puede escoger entre “Captura a XML...” lo que generará un XML con la captura realizada, o bien, “... desde fichero a XML”, lo que leerá una captura ya realizada y generará un XML.</li> <li>d) “Configuración” Permite configurar las rutas de los directorios para los ficheros de capturas, exportaciones, scripts, parametrización y de sincronismo.</li> <li>e) “Salir” para detener la ejecución de la aplicación.</li> </ul>
<b>Postcondición</b>	Depende de la opción elegida.
<b>Excepciones</b>	No existen excepciones posibles.
<b>Importancia</b>	Alta

Tabla B.2: CU-002 Archivo.

<b>CU-003</b>	<b>Captura</b>
<b>Descripción</b>	Condiciones de la captura.
<b>Precondición</b>	Haber seleccionado “Captura” en la interfaz gráfica.
<b>Secuencia normal</b>	<p>1. Se muestra un desplegable.</p> <p>a) “Iniciar Captura” muestra una pantalla donde se puede seleccionar: adaptador de red, fichero de sincronización, filtros durante la captura, nombre de los ficheros que almacenaran las capturas, si se desean múltiples ficheros y condiciones de parada.</p> <p>b) “Captura desde fichero” muestra una pantalla donde seleccionar el fichero desde el que realizar la captura, filtros durante la captura, nombre de los ficheros que almacenaran las capturas, si se desean múltiples ficheros y condiciones de parada.</p>
<b>Postcondición</b>	Depende de la opción elegida.
<b>Excepciones</b>	<ol style="list-style-type: none"> <li>1. En “Iniciar Captura” se debe seleccionar el adaptador de red.</li> <li>2. En “Captura desde fichero” se debe escoger un fichero desde el que realizar la captura.</li> </ol>
<b>Importancia</b>	Alta

Tabla B.3: CU-003 Captura.

<b>CU-004</b>	<b>Definición</b>
<b>Descripción</b>	Definir nuevos paquetes.
<b>Precondición</b>	Haber seleccionado “Definición” en la interfaz gráfica.
<b>Secuencia normal</b>	<p>1. Se muestra un desplegable con una única opción.</p> <p>2. Se muestra una pantalla donde escoger el fichero .XML que contiene la definición de el/los protocolos.</p> <p>3. Se pueden modificar los campos del protocolo.</p> <p>4. Se puede modificar el fichero .XML con los campos del protocolo que se deseen.</p> <p>5. Si se pulsa el botón “Chequear” la aplicación comprobará si los cambios introducidos en el protocolo son correctos.</p>
<b>Postcondición</b>	Se define y se guarda un nuevo paquete.
<b>Excepciones</b>	<p>1. Es necesario escoger el fichero .XML.</p>
<b>Importancia</b>	Alta

Tabla B.4: CU-004 Definición.

---

<b>CU-005</b>	<b>Inserción</b>
<b>Descripción</b>	Opciones para la inyección de paquetes en la red.
<b>Precondición</b>	Haber seleccionado “Inserción” en la interfaz gráfica.
<b>Secuencia normal</b>	<p>1. Se muestra un desplegable.</p> <p>a) “Inserción Paquetes Definidos” muestra una ventana donde definir puertos, direcciones y protocolos para la inyección de paquetes.</p> <p>b) “Inserción Paquetes Capturados” muestra una ventana donde seleccionar el fichero de capturas a insertar y el número de reenvíos que se deseen hacer.</p>
<b>Postcondición</b>	Depende de la opción escogida.
<b>Excepciones</b>	<ol style="list-style-type: none"> <li>1. Es necesario escoger un adaptador de red.</li> <li>2. Es necesario escoger un fichero de capturas.</li> <li>3. Orden de encapsulación del paquete.</li> </ol>
<b>Importancia</b>	Alta

---

Tabla B.5: CU-005 Inserción.

CU-006	Parametrización
<b>Descripción</b>	Opciones para parametrizar ficheros y generar scripts.
<b>Precondición</b>	Haber seleccionado “Parametrización” en el menú inicial.
<b>Secuencia normal</b>	<p>1. Se muestra un desplegable.</p> <ul style="list-style-type: none"> <li>a) “Captura” permite establecer las preferencias para la realización de capturas.</li> <li>b) “Exportación” permite establecer las preferencias para la exportación a XML.</li> <li>c) “Captura desde fichero” permite establecer las preferencias para la realización de capturas desde fichero.</li> <li>d) “Detalle de paquetes” permite establecer las preferencias para la visualización del detalle de paquetes.</li> <li>e) “Generar script...” permite escoger un fichero XML de preferencias y generar un script con las opciones de tipo de fichero, sistema operativo y cantidad de memoria virtual que escoja el usuario.</li> </ul>
<b>Postcondición</b>	Depende de la opción escogida.
<b>Excepciones</b>	No existen excepciones posibles.
<b>Importancia</b>	Alta

Tabla B.6: CU-006 Parametrización. (Modificado por el alumno)

<b>CU-007</b>	<b>Ayuda</b>
<b>Descripción</b>	Mostrar la ayuda de la aplicación.
<b>Precondición</b>	Haber seleccionado “Ayuda” en el menú inicial.
<b>Secuencia normal</b>	<ol style="list-style-type: none"> <li>1. Se muestra un desplegable.             <ol style="list-style-type: none"> <li>a) “Contenidos” abre un navegador con la ayuda de la aplicación.</li> <li>b) “Acerca de...” muestra los autores y la versión.</li> </ol> </li> </ol>
<b>Postcondición</b>	Se muestra la opción escogida.
<b>Excepciones</b>	No existen excepciones posibles.
<b>Importancia</b>	Alta

Tabla B.7: CU-007 Ayuda.

<b>CU-008</b>	<b>Modo Comando</b>
<b>Descripción</b>	Accede a un menú con las opciones y una pequeña ayuda.
<b>Precondición</b>	Haber seleccionado “Modo Comando” en el menú inicial.
<b>Secuencia normal</b>	<ol style="list-style-type: none"> <li>1. Se muestra la pantalla inicial.</li> <li>2. Se escoge una de las opciones del menú.</li> <li>3. Se siguen los pasos indicados.</li> </ol>
<b>Postcondición</b>	Depende de la opción escogida.
<b>Excepciones</b>	<ol style="list-style-type: none"> <li>1. Generar primero los ficheros de parametrización dentro del Modo Gráfico.</li> </ol>
<b>Importancia</b>	Alta

Tabla B.8: CU-008 Modo Comando.

<b>CU-009</b>	<b>Configurar VMJava</b>
<b>Descripción</b>	Accede a un menú donde seleccionar la memoria virtual máxima.
<b>Precondición</b>	Haber seleccionado “Configurar MVJava” en el menú inicial.
<b>Secuencia normal</b>	<ol style="list-style-type: none"> <li>1. Se muestra la pantalla inicial.</li> <li>2. Se escoge una de las opciones del menú.</li> <li>3. Se siguen los pasos indicados.</li> </ol>
<b>Postcondición</b>	Se establece la memoria máxima escogida.
<b>Excepciones</b>	No existen excepciones posibles.
<b>Importancia</b>	Alta

Tabla B.9: CU-009 Configurar VMJava.

<b>CU-010</b>	<b>Salir</b>
<b>Descripción</b>	Se cierra la aplicación.
<b>Precondición</b>	Haber seleccionado “Salir” en el menú inicial.
<b>Secuencia normal</b>	<ol style="list-style-type: none"> <li>1. Sale de la aplicación.</li> </ol>
<b>Postcondición</b>	Se cierra la aplicación.
<b>Excepciones</b>	No existen excepciones posibles.
<b>Importancia</b>	Media

Tabla B.10: CU-010 Salir.



## *Apéndice C*

---

# Especificación de diseño

---

## C.1. Introducción

En este documento se mostrará el diseño de los diferentes componentes de la aplicación, con el fin de entender sus funcionalidades con mayor profundidad.

Cabe destacar que la mayor parte de la aplicación se corresponde con los proyectos anteriores, ya que solo alguna parte concreta ha sido modificada. Por lo tanto, gran parte de la información será heredada de proyectos anteriores. Aquellos que hayan sido modificados contendrán información adicional de las modificaciones.

## C.2. Diseño de datos

Las diferentes entidades de la aplicación pueden dividirse en:

- **Usuarios.** Son los encargados de hacer uso de la aplicación, pero esta no contiene información relativa a ellos.
- **Capturas.** Son uno de los puntos principales de la aplicación. La captura y la inserción son dos de las acciones relacionadas con ellas. Las capturas son conjuntos de paquetes que son o han sido capturados en un espacio de tiempo concreto.
- **Paquetes.** Son pequeñas piezas de la información transmitida por la red. Están definidos por las cabeceras de protocolos, que declaran el tipo de transmisión utilizado para el envío de esos paquetes.

- **Protocolos.** Estos definen cómo se realizan las transmisiones de información a través de la red.
- **Ficheros.** Las capturas y definiciones de paquetes se guardan en ficheros con extensión Pcap o XML. Con ellos se pueden visualizar capturas realizadas anteriormente, e incluso insertar sus paquetes de nuevo en la red.
- **Ficheros de preferencias.** Estos ficheros son específicos de la parametrización de preferencias de la aplicación. Sirven para guardar los valores escogidos habitualmente por un usuario, con el fin de ahorrarle tiempo.

### C.3. Diseño procedimental

En este apartado se muestran los diagramas de secuencia de las diferentes funcionalidades.

#### Archivo

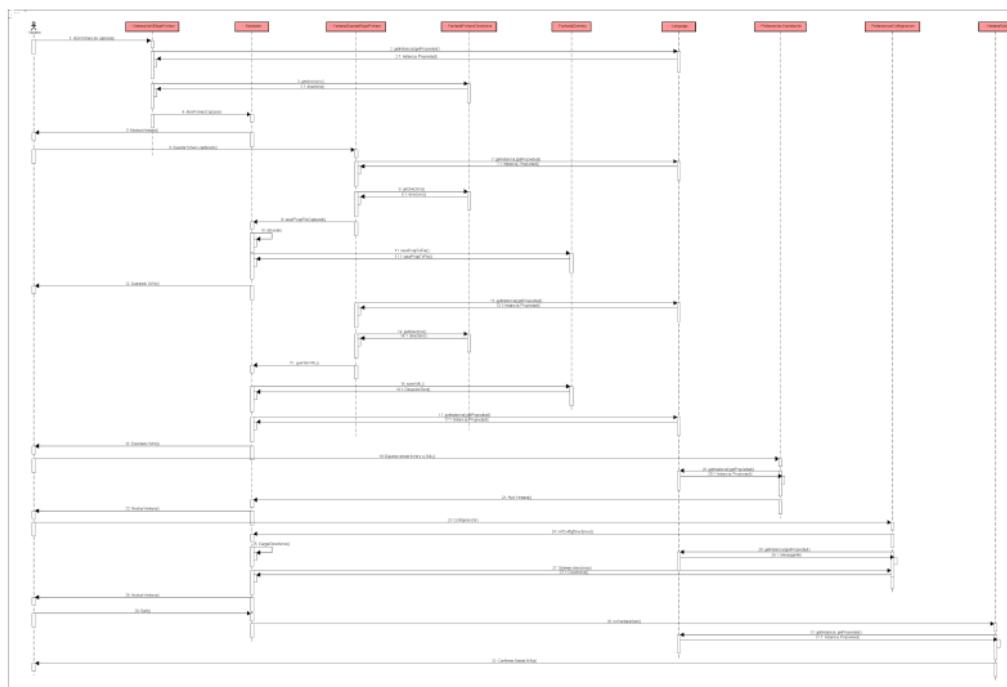


Figura C.1: Diagrama de secuencia del Menú Archivo

## Captura

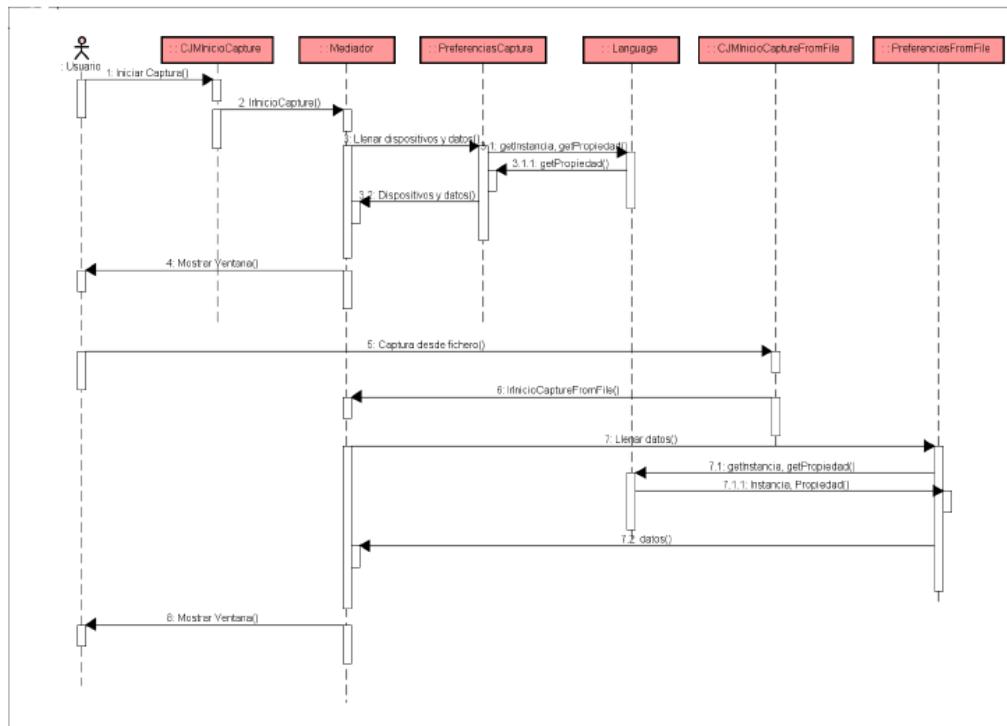


Figura C.2: Diagrama de secuencia del Menú Captura

## Definición

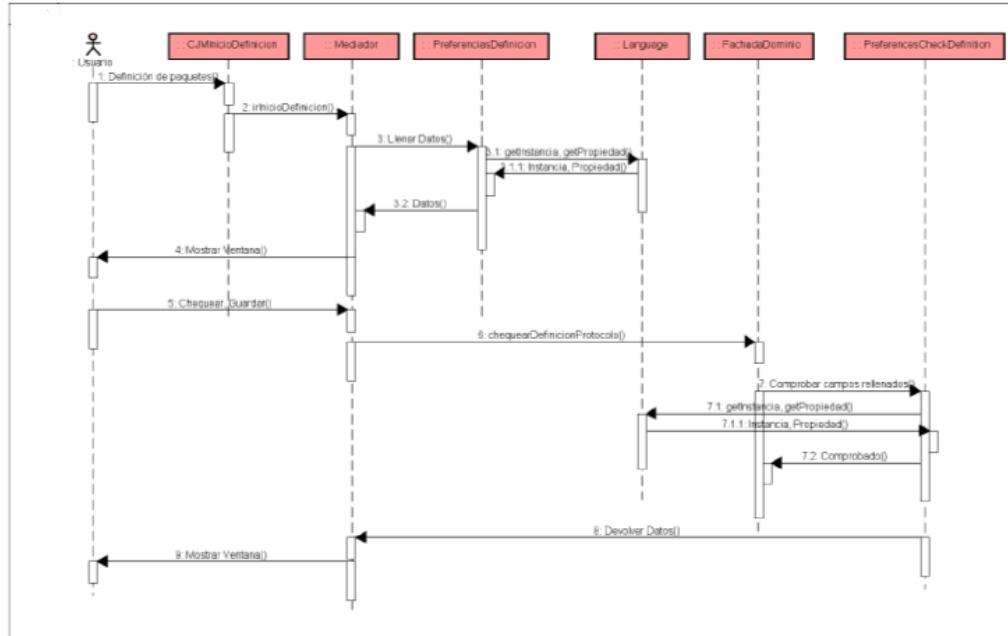


Figura C.3: Diagrama de secuencia del Menú Definición

### Inserción

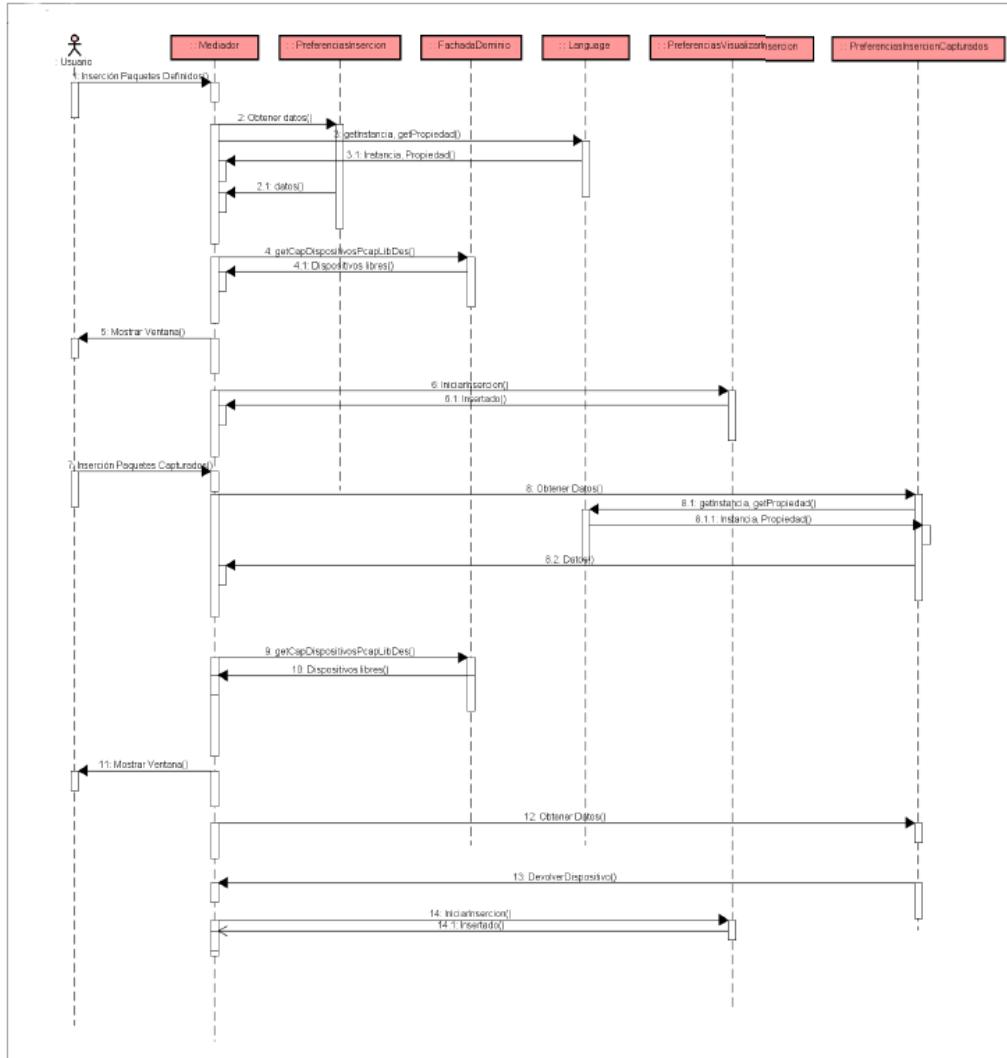


Figura C.4: Diagrama de secuencia del Menú Inserción

## Parametrización

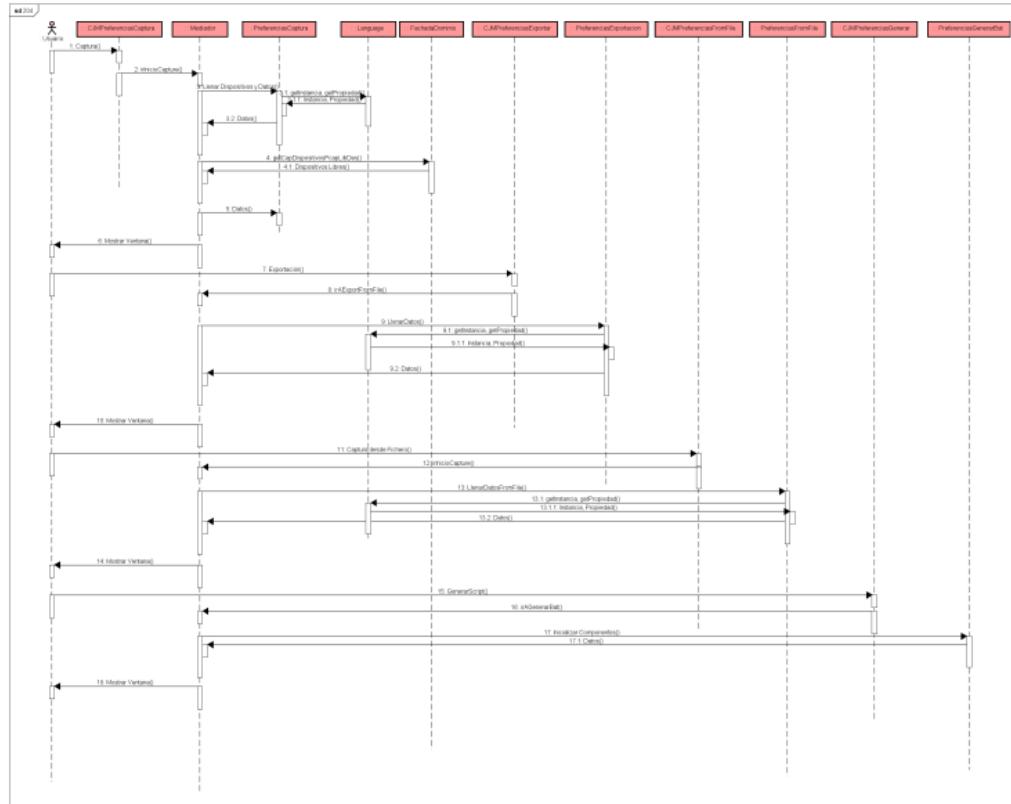


Figura C.5: Diagrama de secuencia del Menú Parametrización

## Ayuda

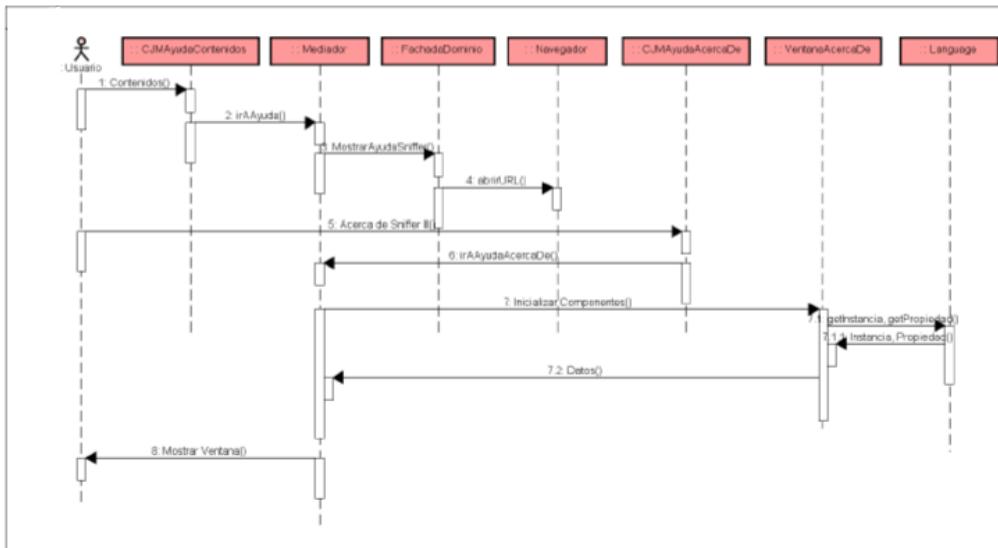


Figura C.6: Diagrama de secuencia del Menú Ayuda

## C.4. Diseño arquitectónico

La arquitectura de la aplicación está basada en el modelo-vista-controlador (MVC).

La estructura de paquetes es la siguiente:

- Dominio. Contiene las clases que contienen la programación backend.
- Presentación. Contiene las clases que influyen en la interfaz gráfica.
- ServicioAccesoDatos. Contiene las clases que crean los ficheros.

## Diagrama de paquetes

En esta sección mostraremos los diferentes diagramas de paquetes de la aplicación, para conocer mejor su diseño.

**Diagrama de paquetes general**

En el siguiente diagrama se muestra una breve representación de la interacción entre paquetes.

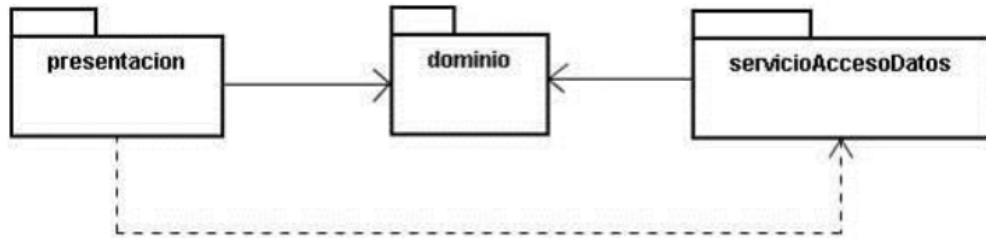


Figura C.7: Diagrama de paquetes general.

## Diagramas específicos de los paquetes

- #### ■ Dominio

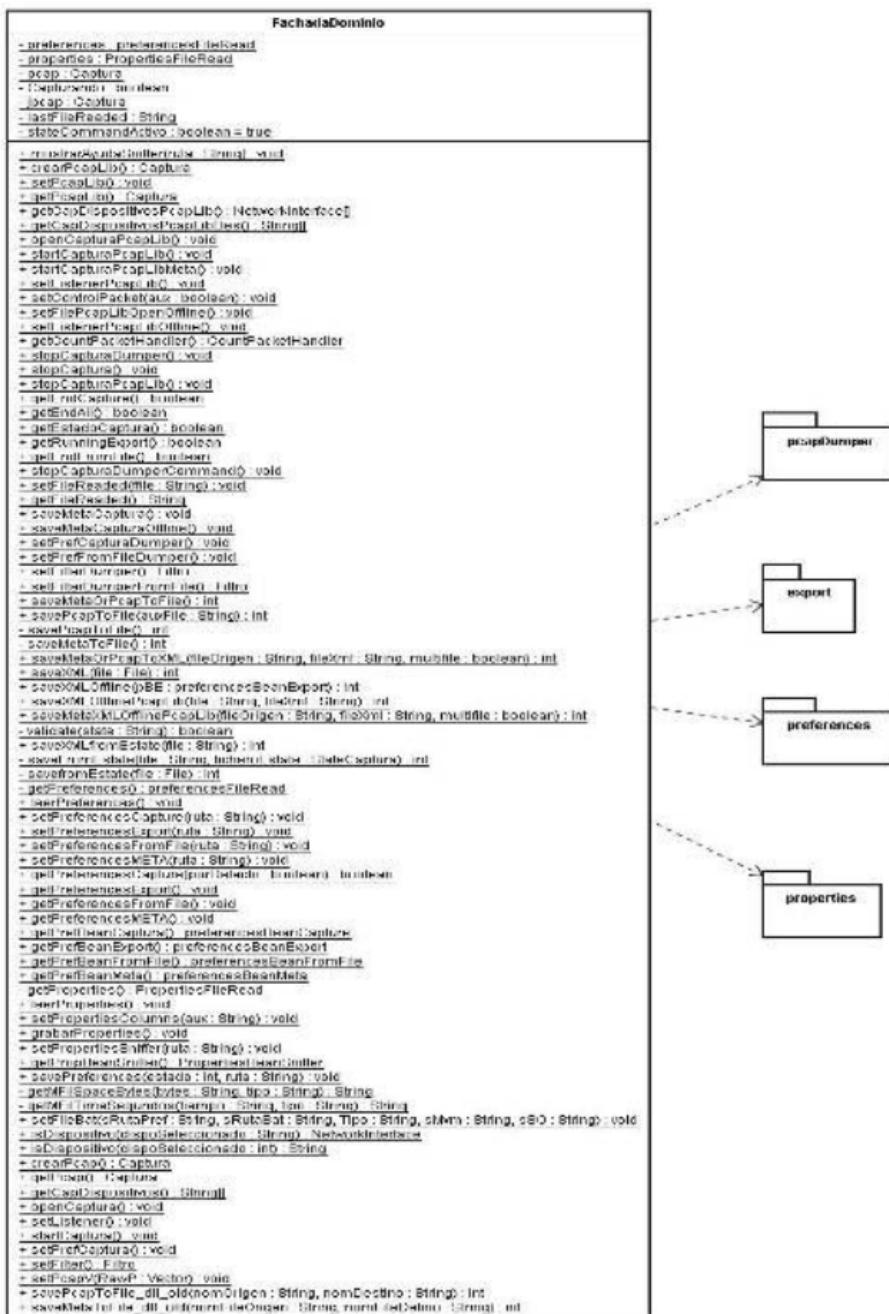


Figura C.8: Paquete dominio

■ Dominio.export

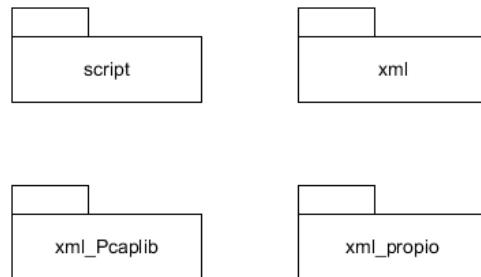


Figura C.9: Paquete dominio.export

■ Dominio.export.script

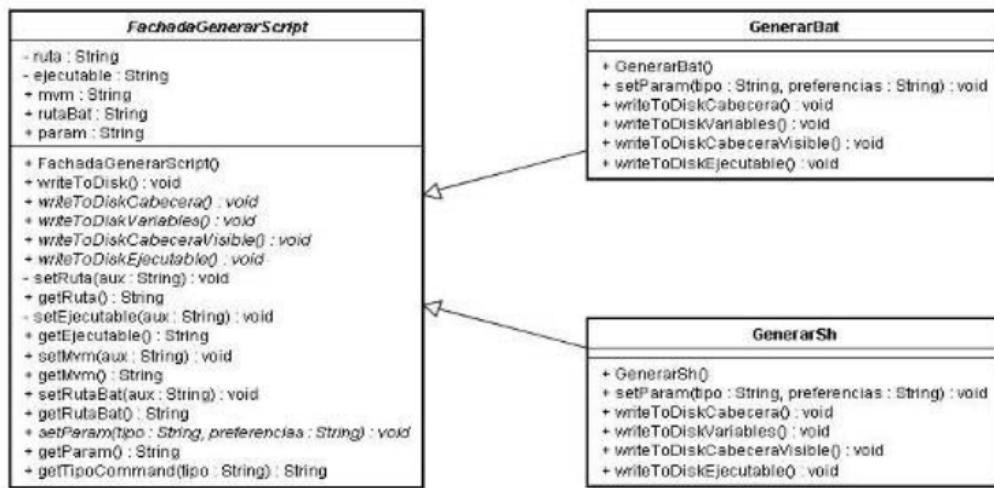


Figura C.10: Paquete dominio.export.script

- Dominio.export.xml\_Pcaplib

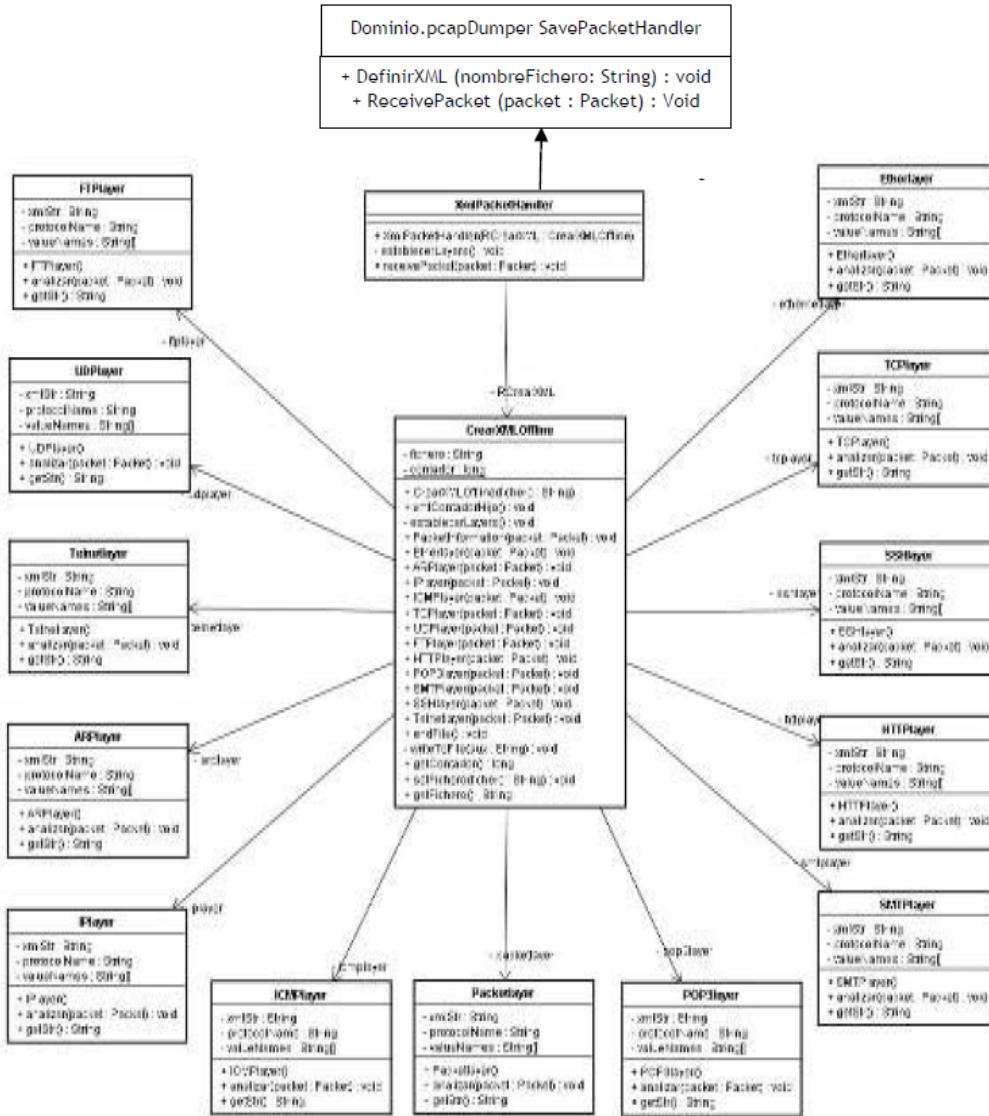


Figura C.11: Paquete dominio.export.xml\_Pcaplib

■ Dominio.pcapDumper

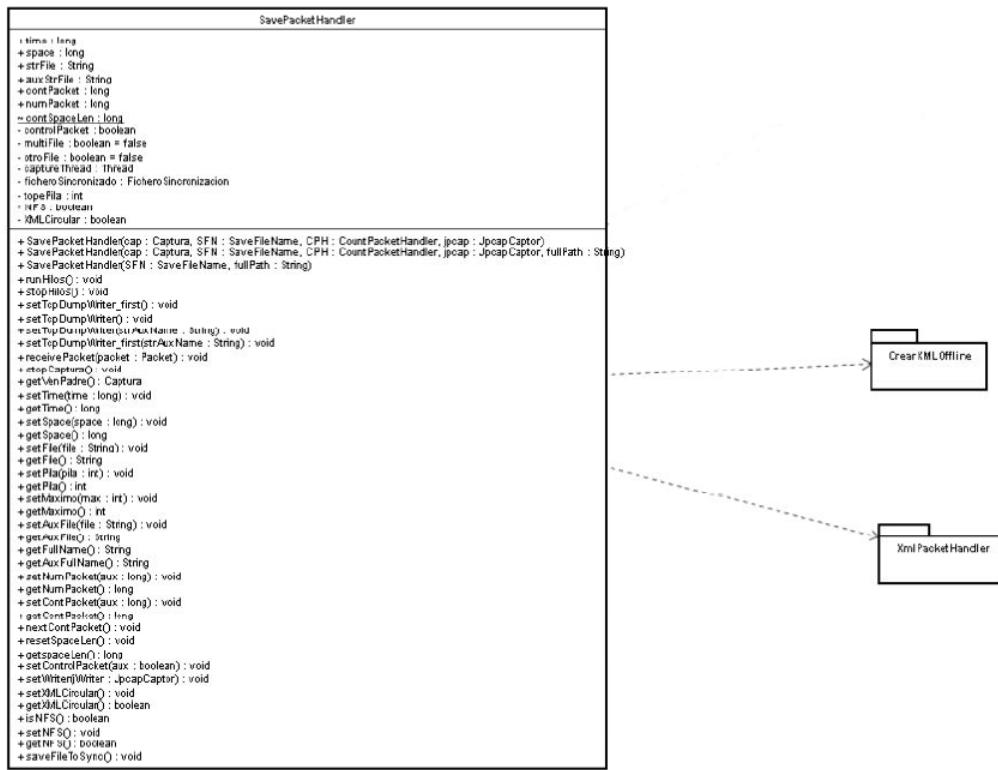


Figura C.12: Paquete dominio.pcapDumper

#### ■ Dominio.preferences

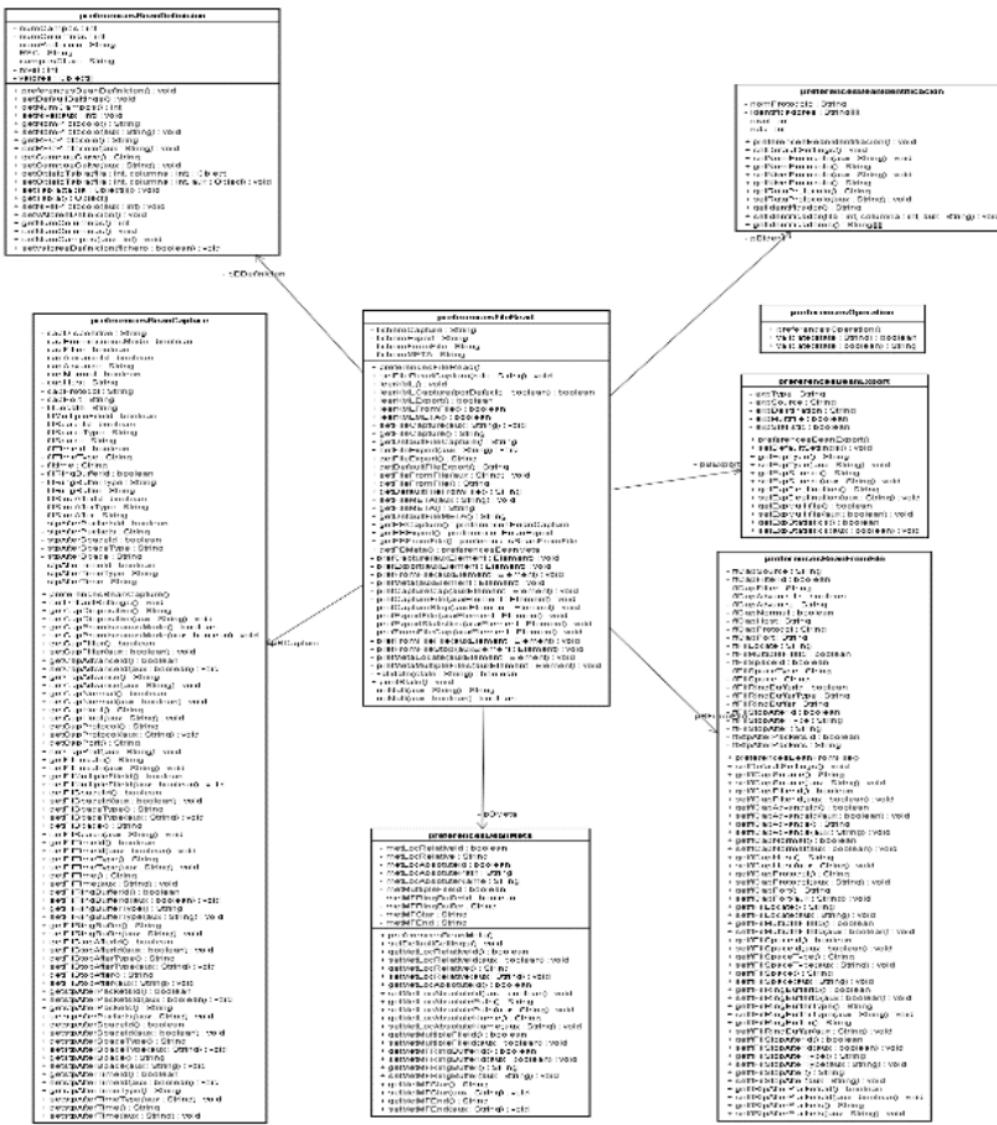


Figura C.13: Paquete dominio.preferences

- Dominio.preferences.capture

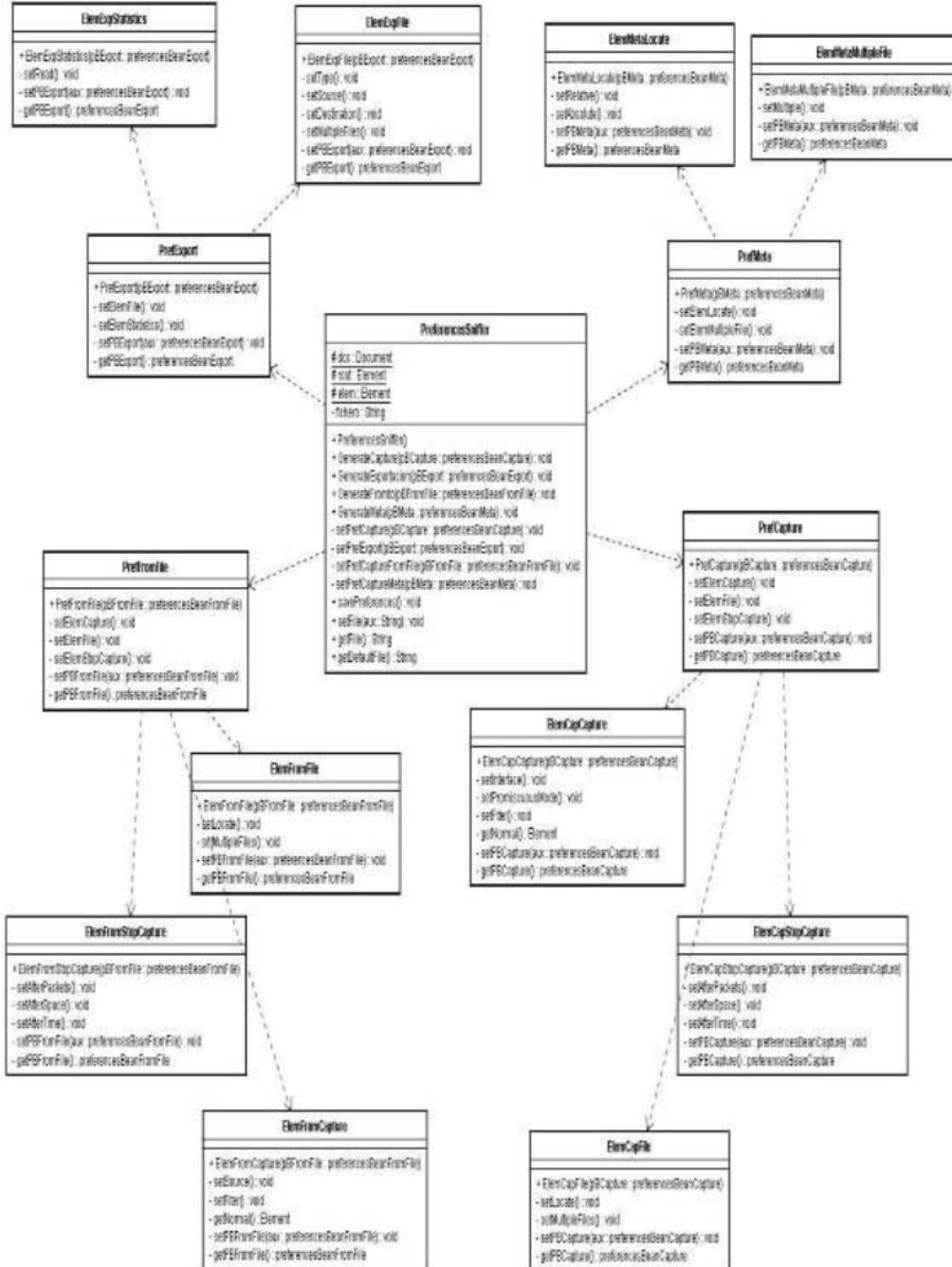


Figura C.14: Paquete dominio.preferences.capture

- Dominio.preferences.definicion

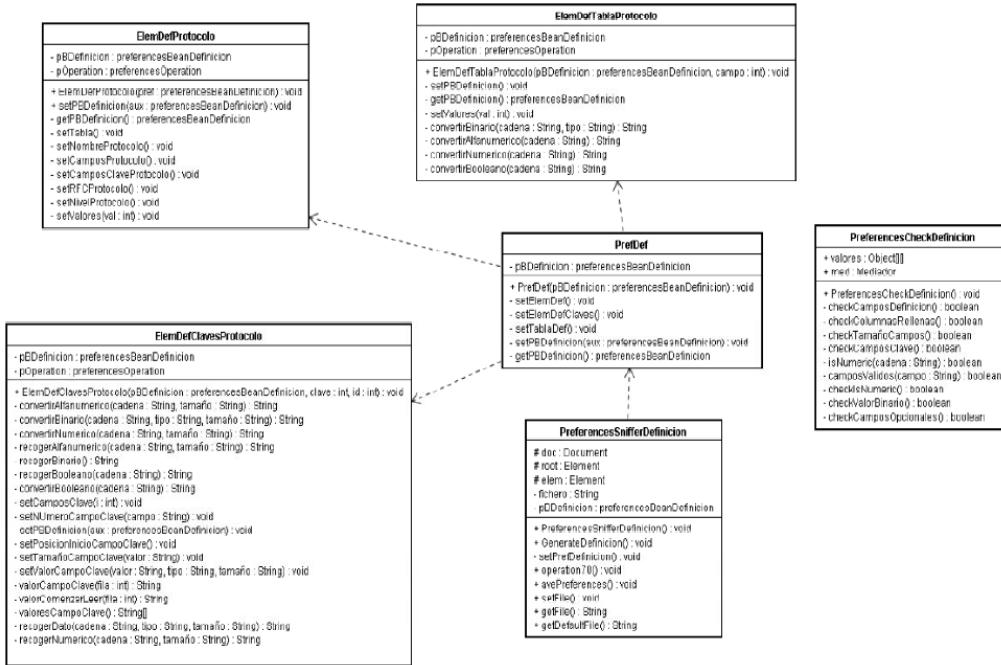


Figura C.15: Paquete dominio.preferences.definicion

- Dominio.preferences.identificacion

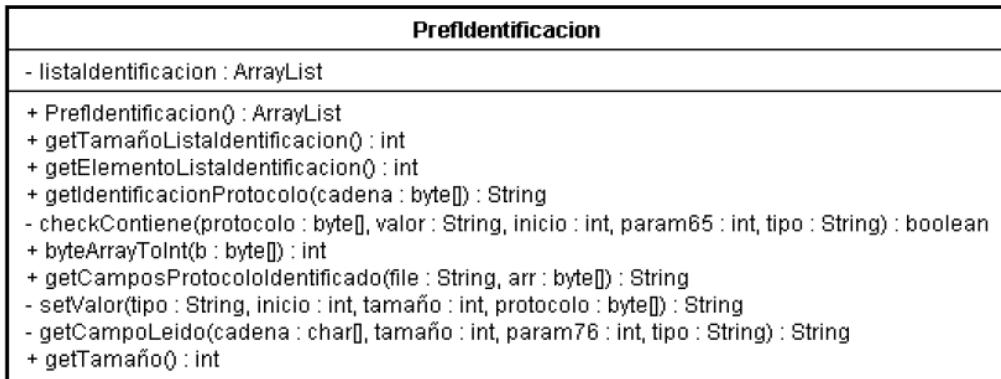


Figura C.16: Paquete dominio.preferences.identificacion

- Dominio.properties

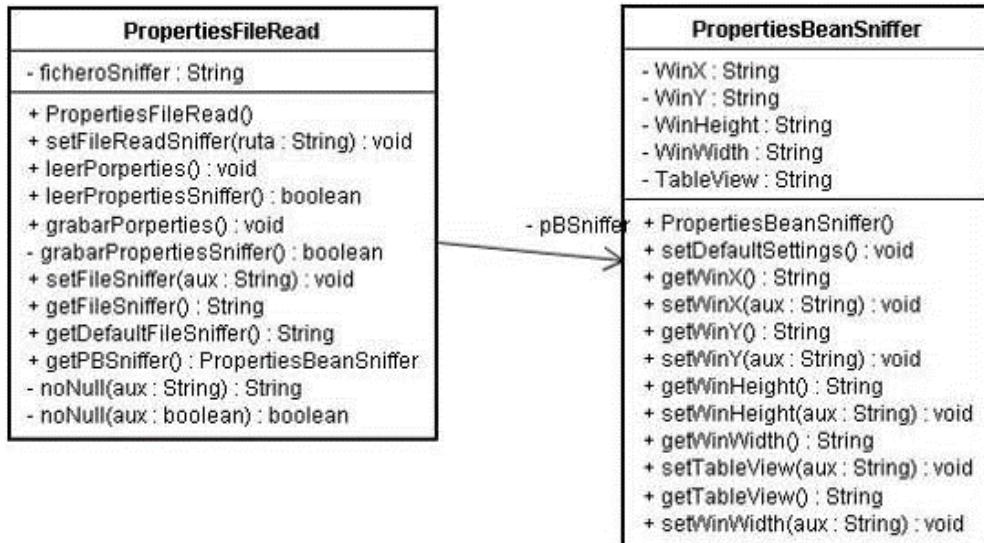


Figura C.17: Paquete dominio.properties

## ■ Presentacion

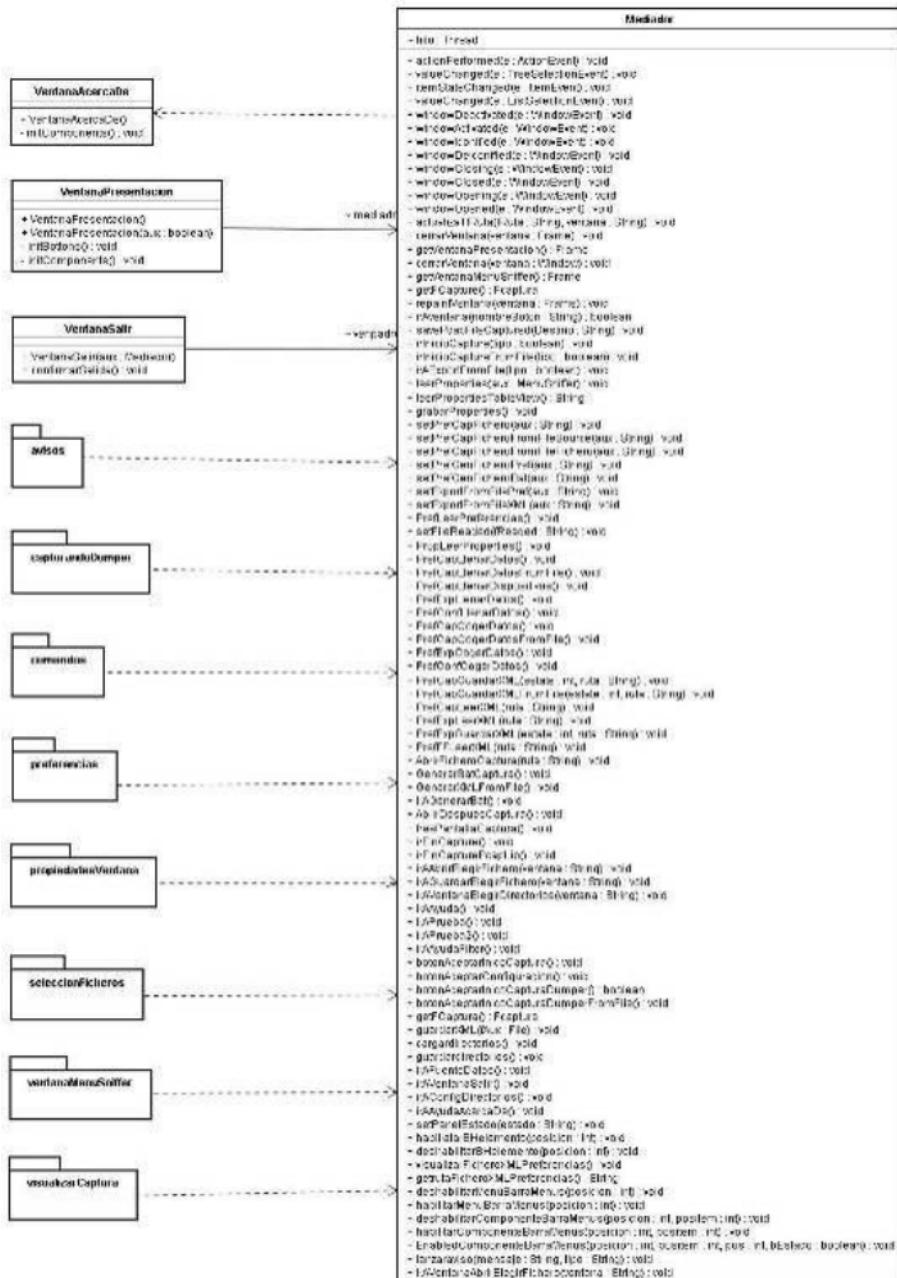


Figura C.18: Paquete presentacion

■ Presentacion.avisos

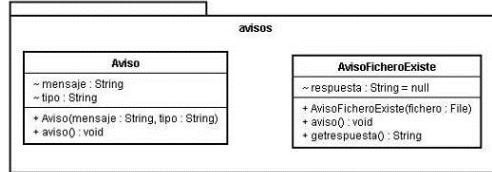


Figura C.19: Paquete presentacion.avisos

■ Presentacion.capturando

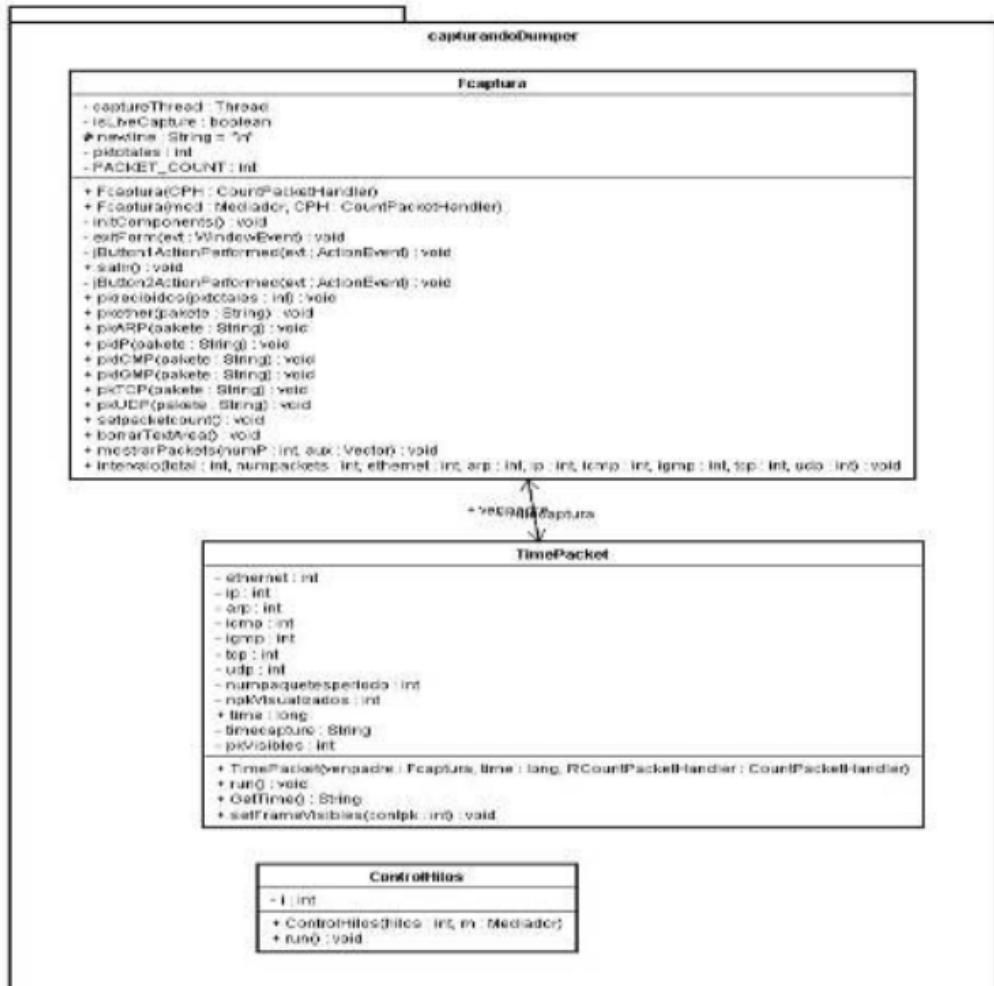


Figura C.20: Paquete presentacion.capturando

#### ■ Presentacion.comandos

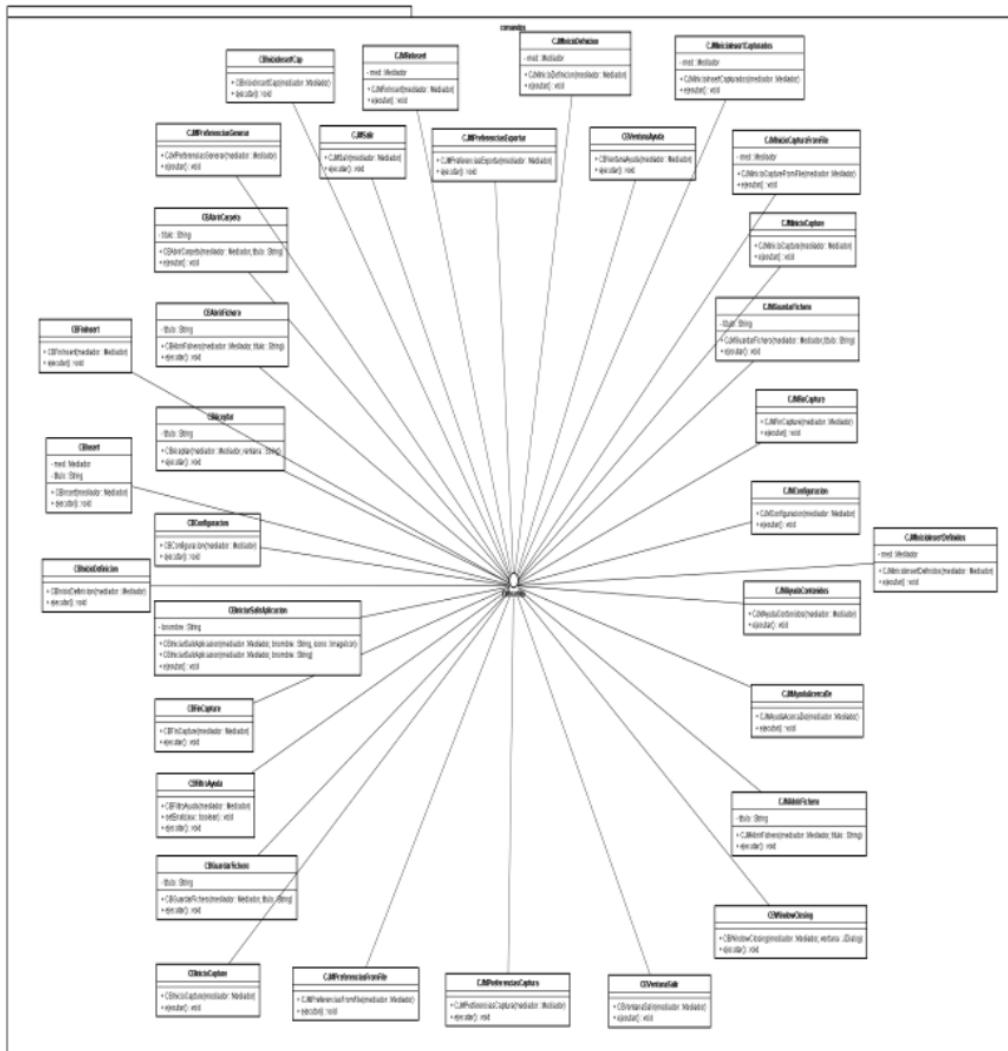


Figura C.21: Paquete presentacion.comandos

- Presentacion.preferencias

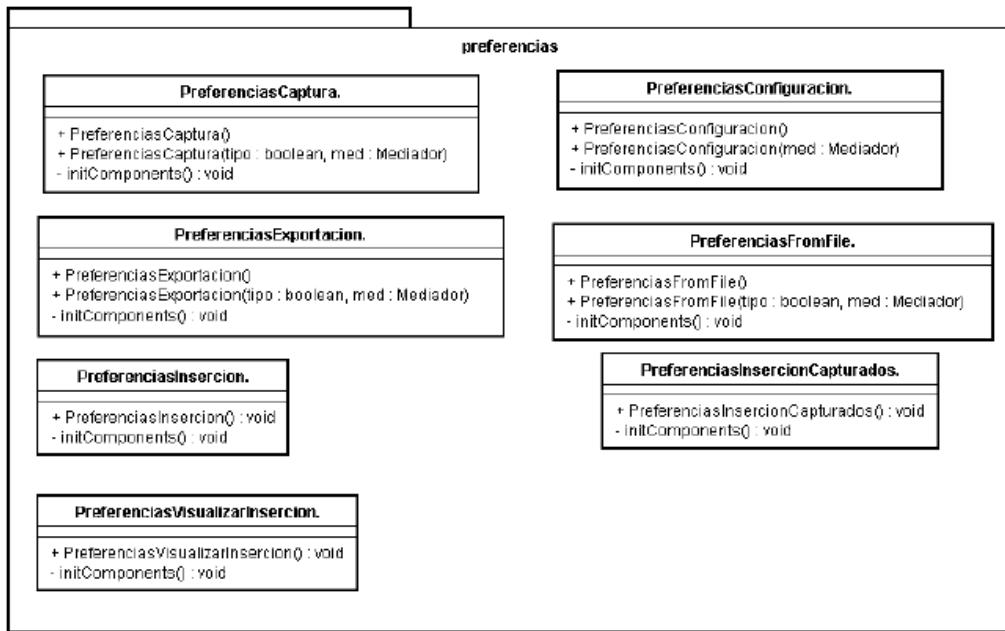


Figura C.22: Paquete presentacion.preferencias

- Presentacion.propiedadesVentana

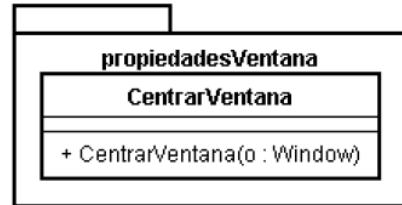


Figura C.23: Paquete presentacion.propiedadesVentana

- Presentacion.seleccionFicheros

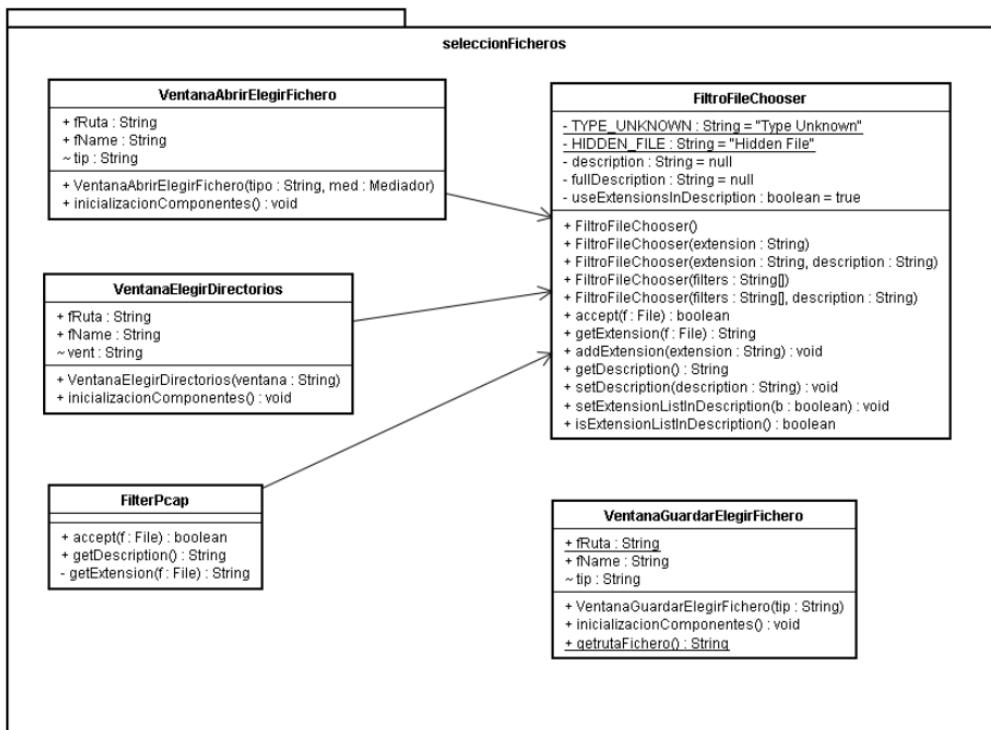


Figura C.24: Paquete `presentacion.seleccionFicheros`

■ Presentacion.ventanaMenuSniffer

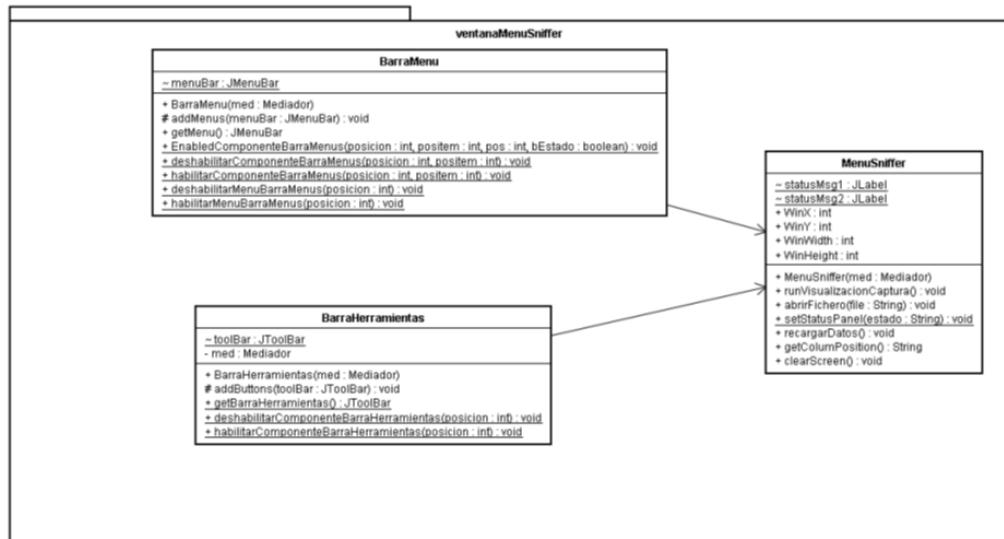


Figura C.25: Paquete `presentacion.ventanaMenuSniffer`

#### ■ Presentacion.visualizarCaptura

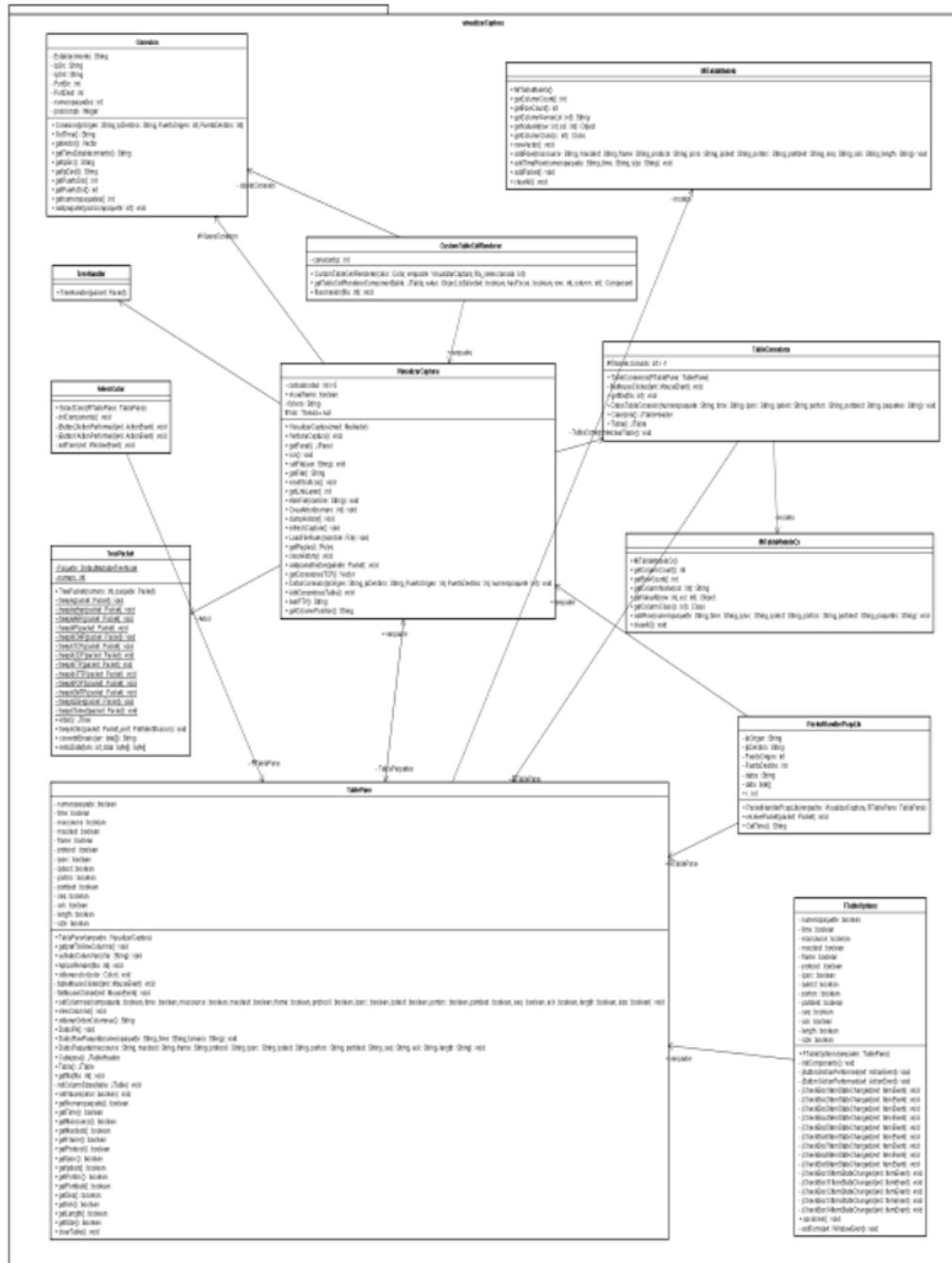


Figura C.26: Paquete presentacion.visualizarCaptura

- ServicioAccesoDatos

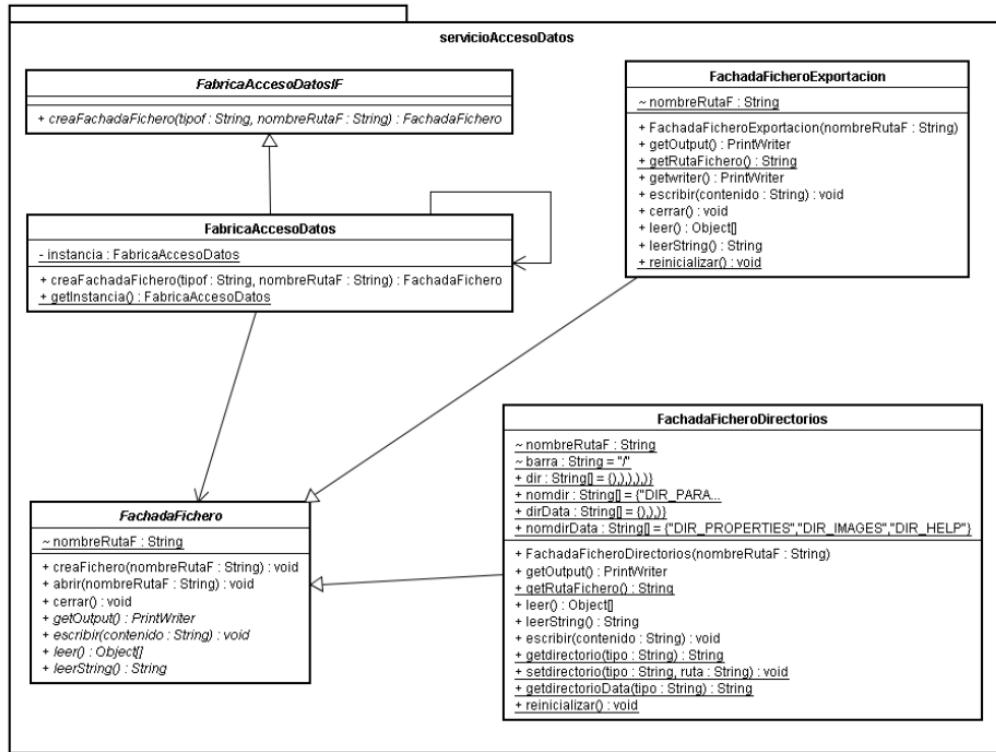


Figura C.27: Paquete servicioAccesoDatos

## Diagramas de clases

A continuación, se muestran los diferentes diagramas de clases de la aplicación, comenzando por el diagrama general. Los más específicos se han realizado sobre las diferentes funcionalidades.

## Diagrama de clases general

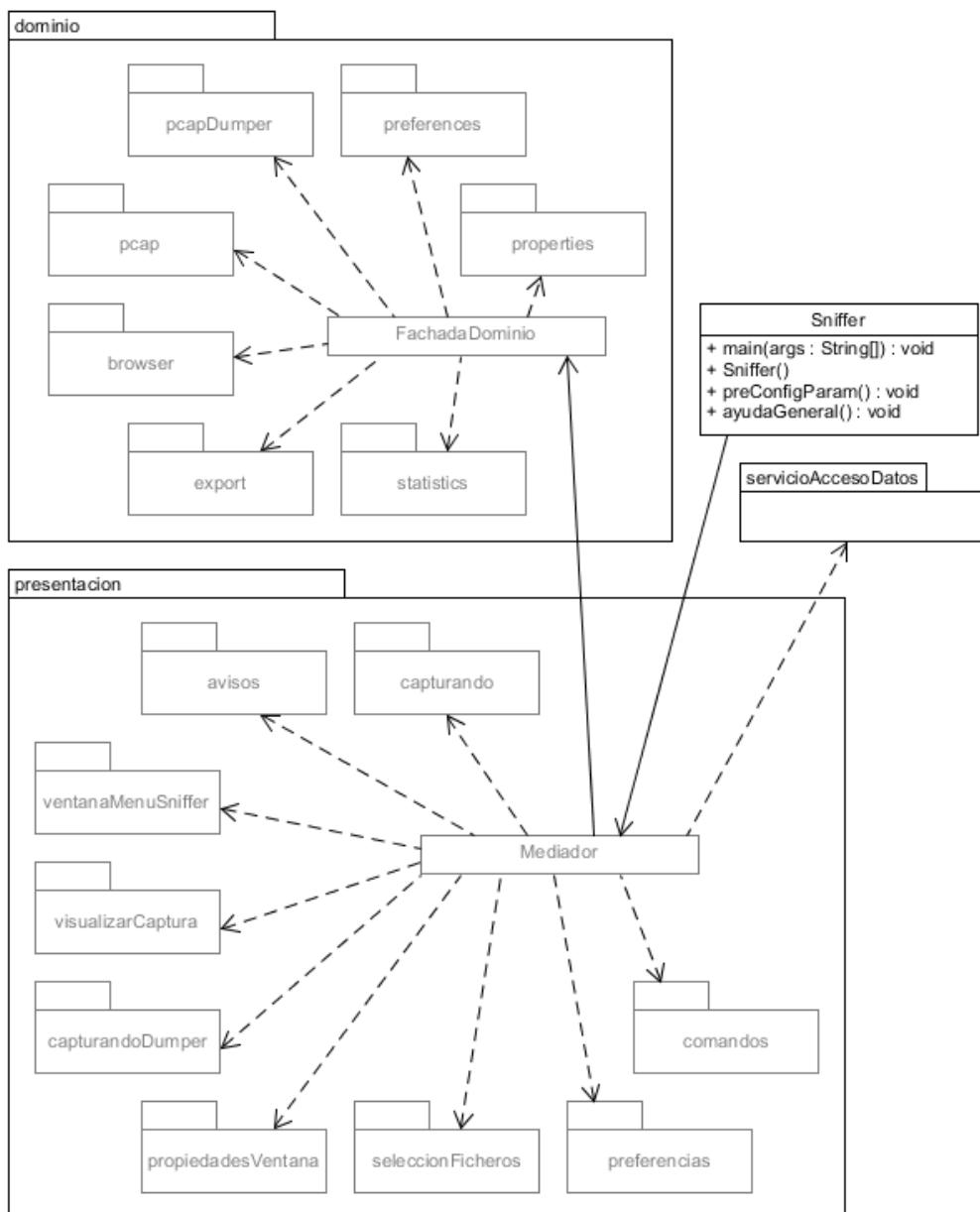


Figura C.28: Diagrama de clases general.

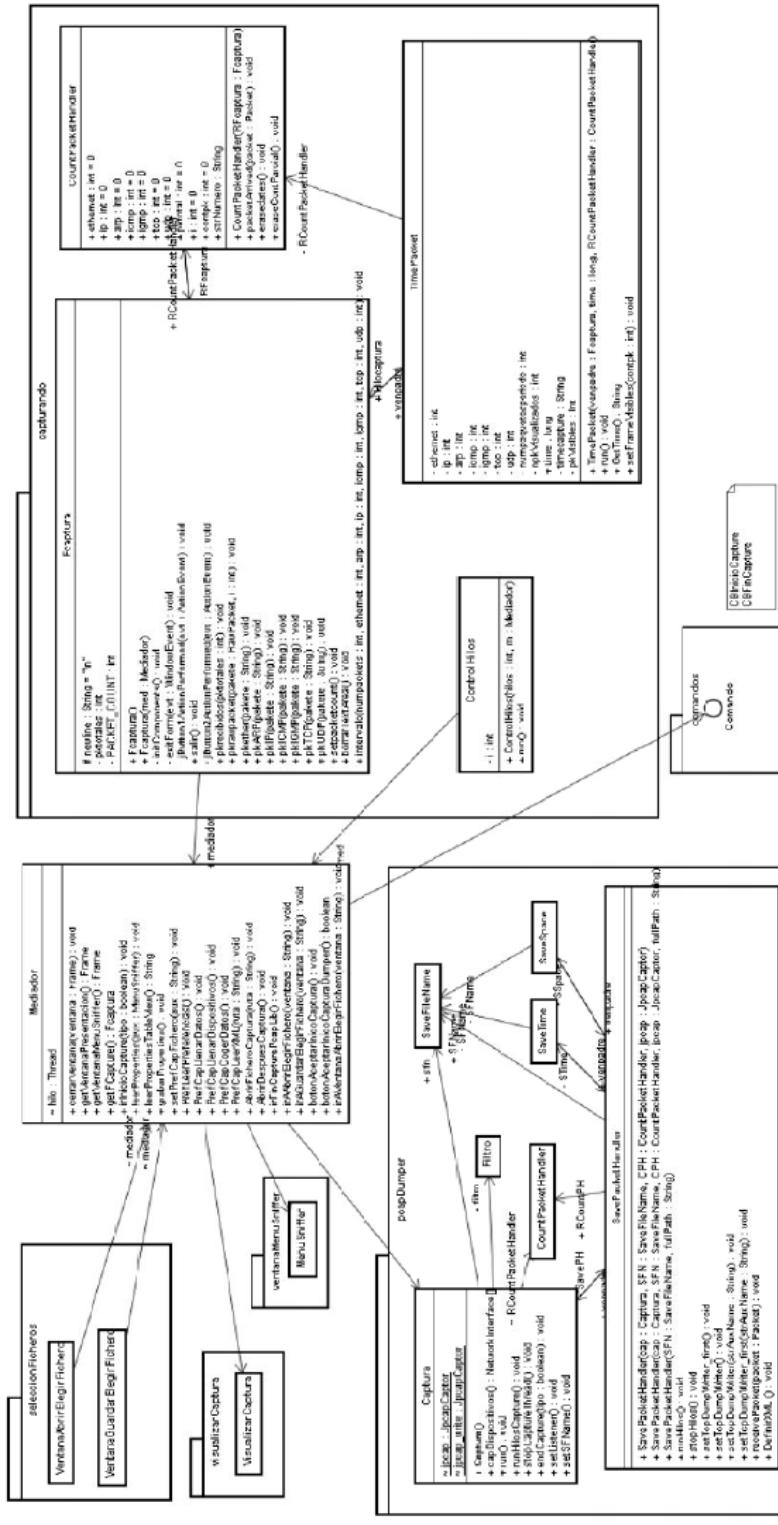


Figura C.29: Diagrama de clases para la captura.

Diagrama de clases – Captura desde fichero

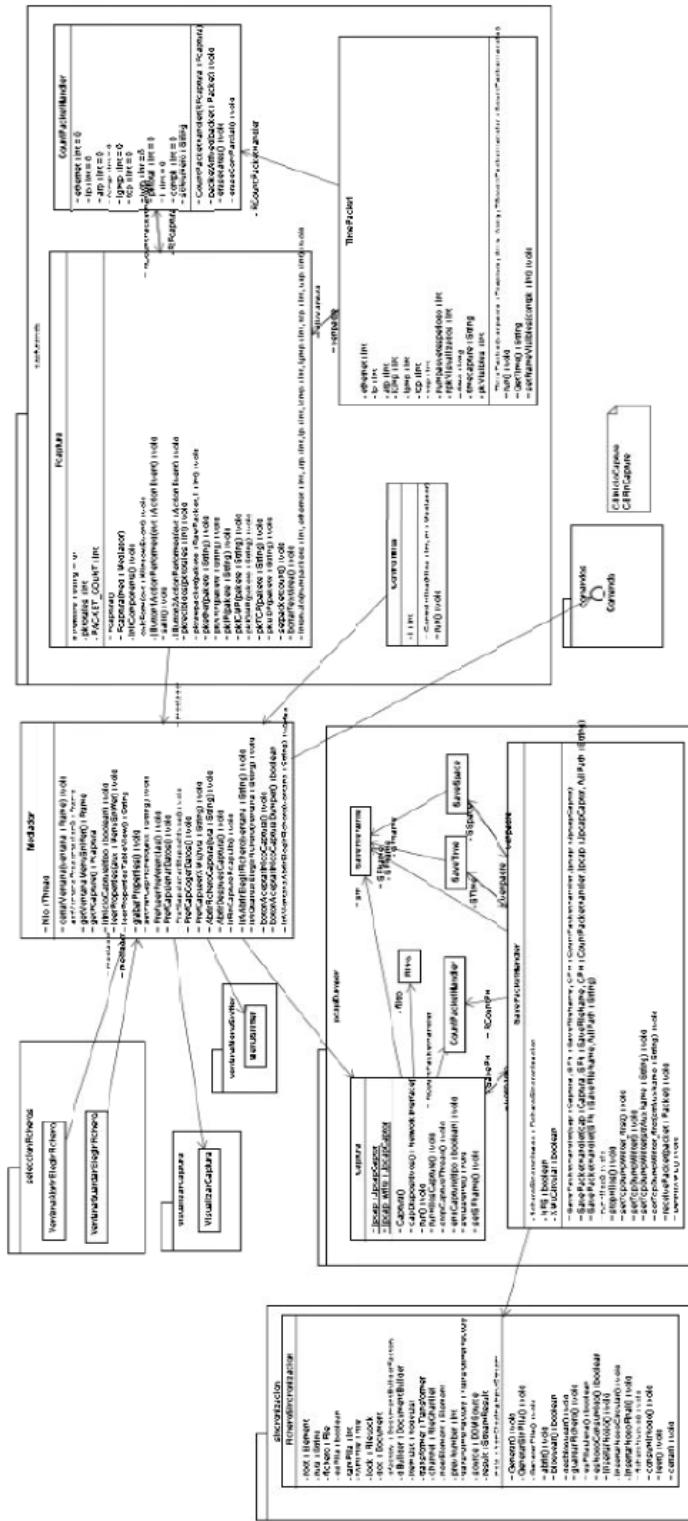


Figura C.30: Diagrama de clases para la captura desde fichero.

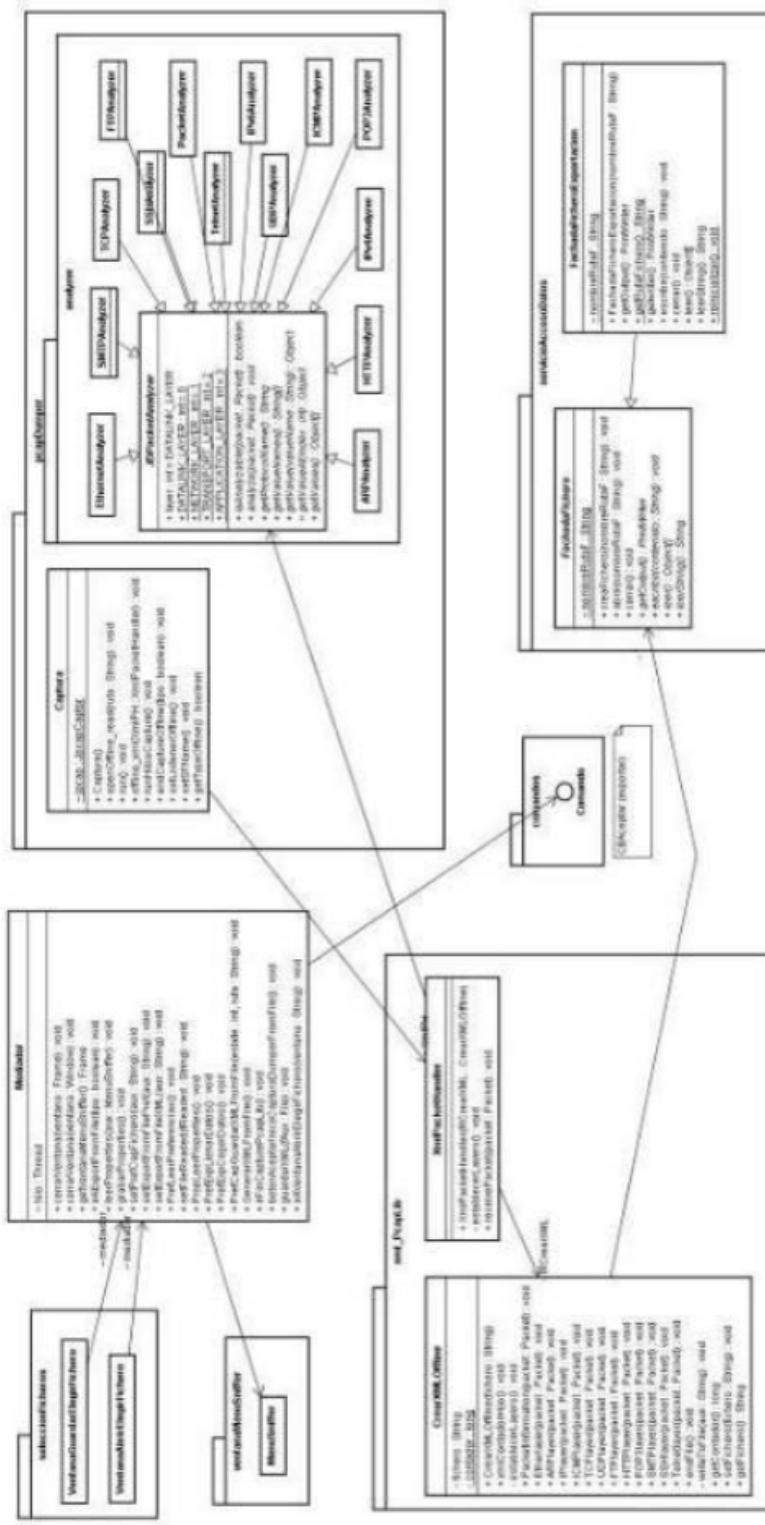


Figura C.31: Diagrama de clases para la exportación.

#### Diagrama de clases – Definición de protocolo

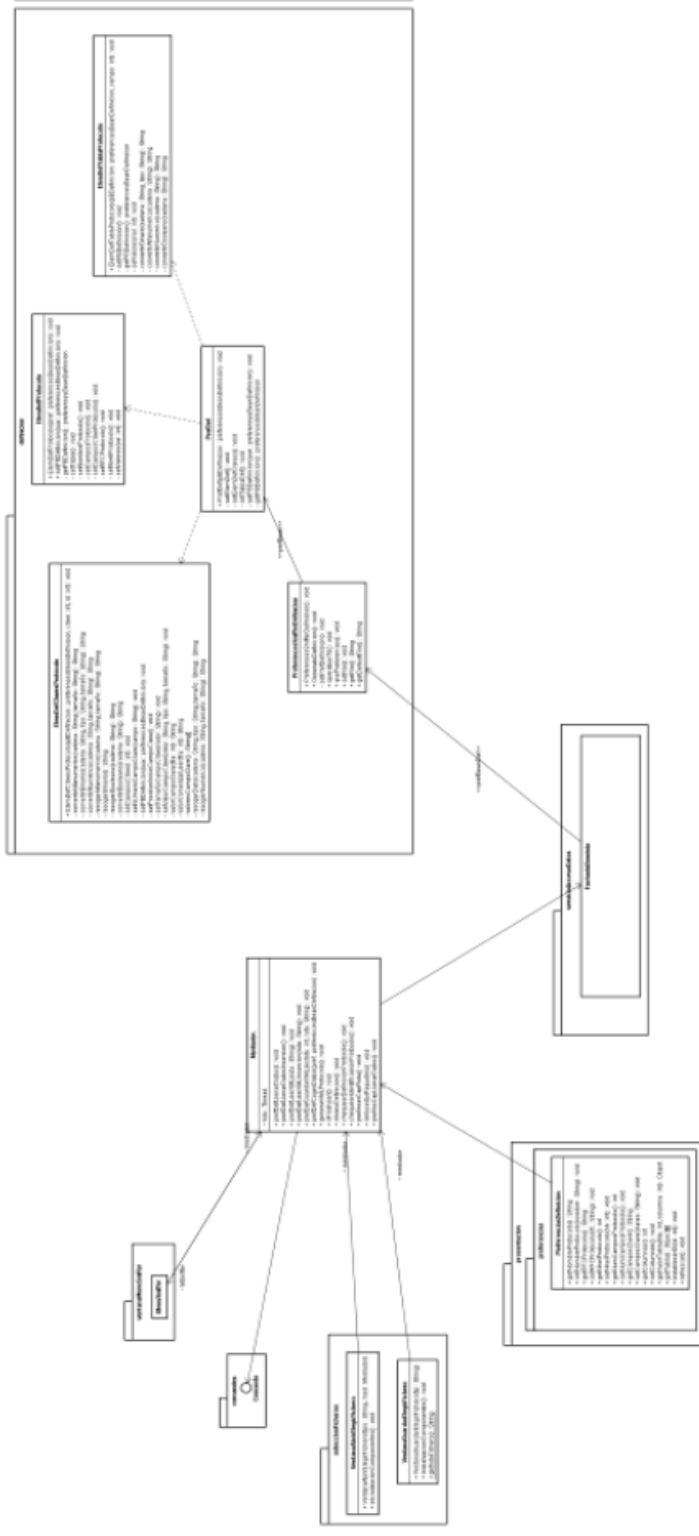


Figura C.32: Diagrama de clases para la definición.

## Diagrama de clases – Inserción de paquetes capturados

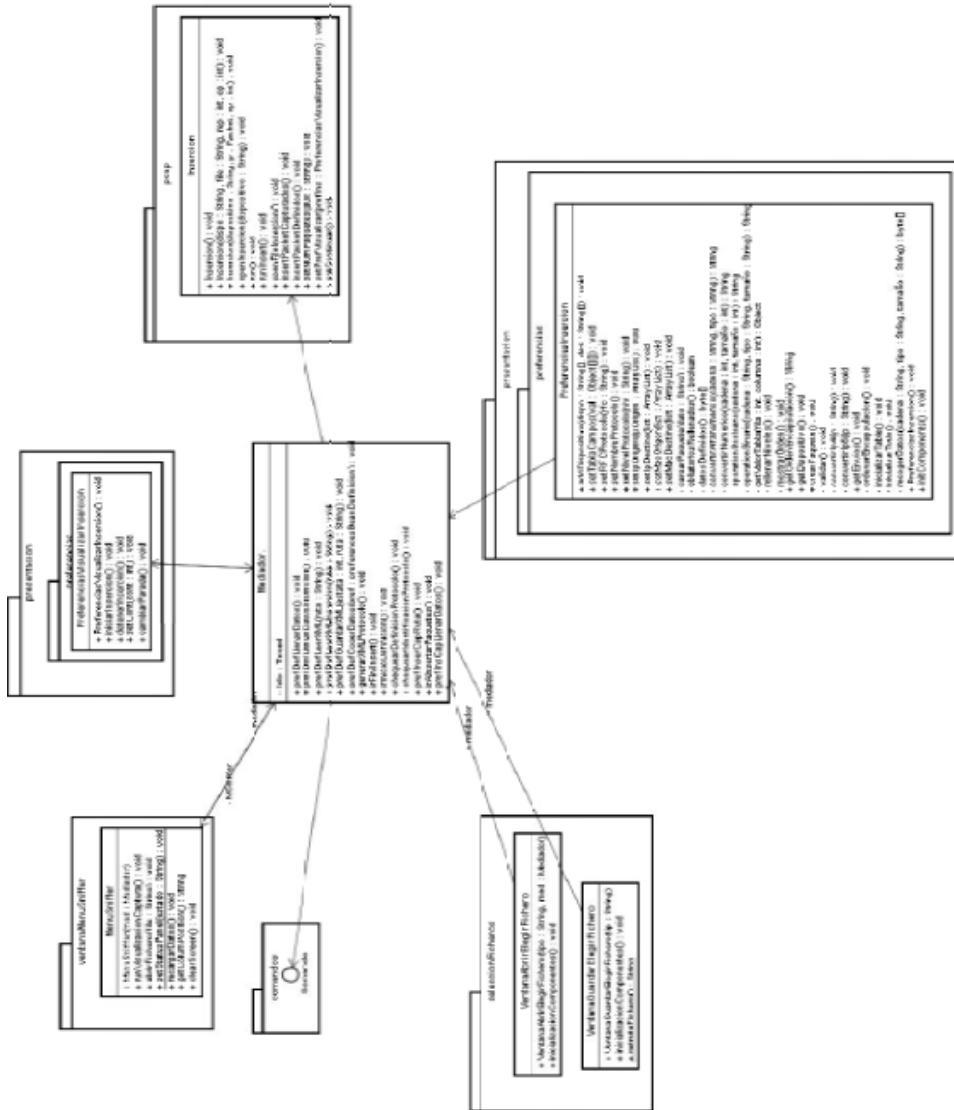


Figura C.33: Diagrama de clases para la inserción de paquetes capturados.

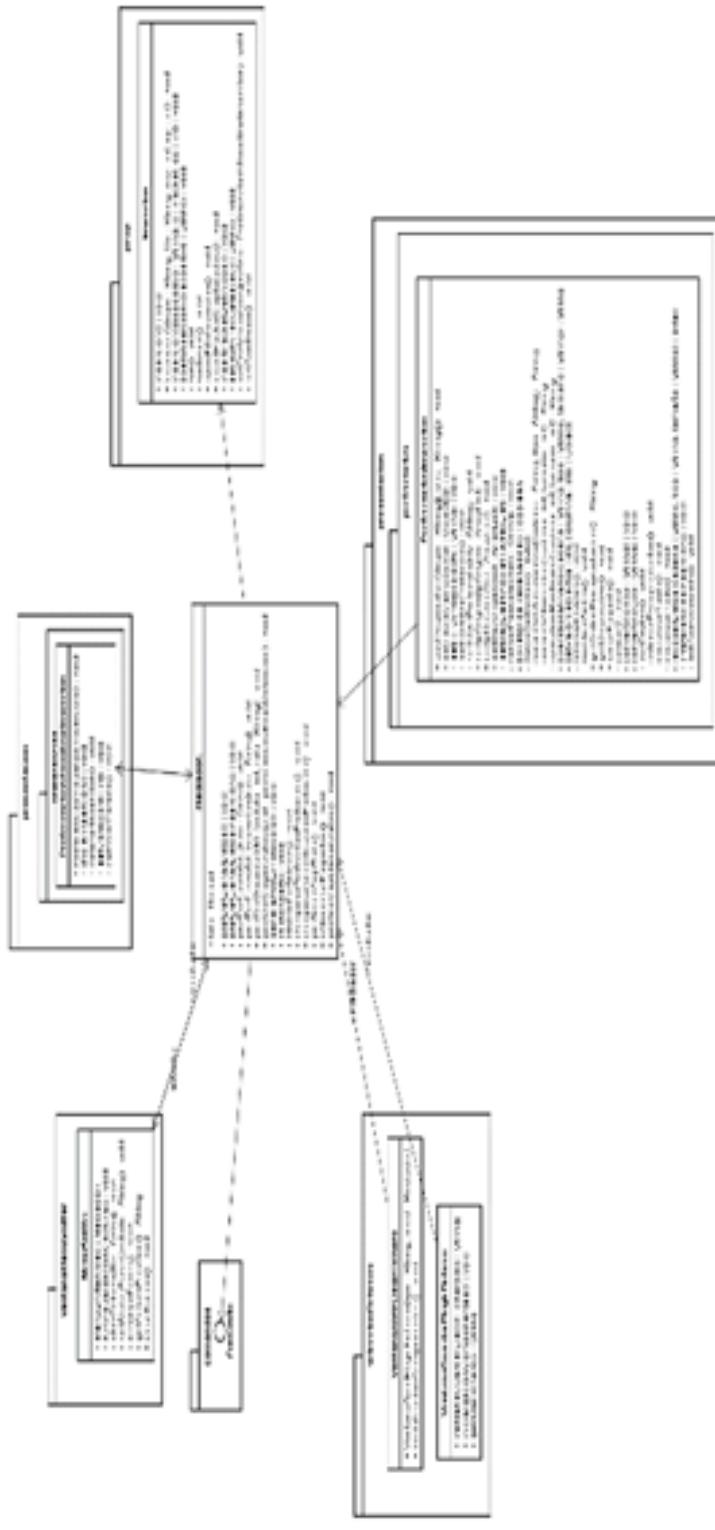


Figura C.34: Diagrama de clases para la inserción de paquetes definidos.



## *Apéndice D*

---

# **Documentación técnica de programación**

---

## **D.1. Introducción**

Este documento recoge los datos necesarios para que un programador comprenda la información relativa a la programación de la aplicación. Se describe la estructura de directorios, así como la de paquetes, las bibliotecas que se han utilizado y los procesos de instalación seguidos, junto a las pruebas del sistema realizadas.

## **D.2. Estructura de directorios**

- / Raíz del proyecto. Se encuentran ahí un fichero README y otros directorios.
- /**Herramientas\_Instalacion\_Sniffer**/ Se encuentran los diferentes archivos cuya instalación es necesaria para el funcionamiento de la aplicación. En el apéndice *E. Documentación de usuario*, se explica cuando se necesita cada uno de ellos.
- /**Herramientas\_Instalacion\_Sniffer/Java**/ JDKs para las diferentes plataformas.
- /**Herramientas\_Instalacion\_Sniffer/jNetPcap**/ Archivos de la biblioteca *jNetPcap* para las diferentes plataformas.

- **/Herramientas\_Instalacion\_Sniffer/libPcap/** Archivos necesarios para la configuración de la biblioteca *libPcap* en Linux.
- **/Herramientas\_Instalacion\_Sniffer/winPcap/** Ejecutable para la instalación de la biblioteca *winPcap* en Windows.
- **/Herramientas\_Instalacion\_Sniffer/Wireshark/** Ejecutables para la instalación de Wireshark en Windows, para las dos arquitecturas.
- **/jnetpcap-original/** En este directorio se encuentra la última versión oficial de la biblioteca *jNetPcap*. Es decir, no se trata de la última versión utilizada en la aplicación. No consiste solo en los paquetes y las clases, sino que contiene mucha información adicional, como los ficheros escritos en lenguaje C que conforman el DLL, o las versiones previas de la biblioteca.
- **/Memoria/** Este directorio almacena los documentos relativos a la memoria, siendo estos los PDFs de los anexos y la propia memoria.
- **/Sniffer/** Directorio del proyecto sobre el que se ha trabajado, usando la aplicación Eclipse. Contiene aquellos directorios que usa la aplicación desarrollada.
- **/jNetPcap/** Directorio del proyecto de la biblioteca que ha sido mejorada, usando la aplicación Eclipse. Contiene aquellos directorios que conforman la biblioteca.

## D.3. Manual del programador

### 1. Introducción

En este apartado, se describen los aspectos más relevantes de la programación de la aplicación.

### 2. Estructura de paquetes

La organización de los paquetes es la siguiente:

- / Raíz del proyecto, contiene la clase Sniffer, que contiene el método *main*.
- **/dominio/** Contiene el *backend* de la aplicación.

- /dominio/browser/
  - /dominio/export/
    - /dominio/export/script/
    - /dominio/export/xml/
    - /dominio/export/xml\_Pcaplib/
    - /dominio/export/xml\_propio/
  - /dominio/pcap/
    - /dominio/pcap/rules/
  - /dominio/pcapDumper/
    - /dominio/pcapDumper/analyzer/
    - /dominio/pcapDumper/rules/
  - /dominio/preferences/
    - /dominio/preferences/capture/
    - /dominio/preferences/definicion/
    - /dominio/preferences/detalle/
    - /dominio/preferences/export/
    - /dominio/preferences/fromfile/
    - /dominio/preferences/identificacion/
    - /dominio/preferences/insercion/
    - /dominio/preferences/meta/
    - /dominio/preferences/sniffer/
  - /dominio/properties/
  - /dominio/statistics/
- /presentacion/ Contiene las clases necesarias para la interfaz gráfica.
- /presentacion/avisos/
  - /presentacion/capturando/
  - /presentacion/capturandoDumper/
  - /presentacion/comandos/
  - /presentacion/preferencias/
  - /presentacion/propiedadesVentana/
  - /presentacion/seleccionFicheros/
  - /presentacion/ventanaMenuSniffer/

- `/presentacion/visualizarCaptura/`
- `/servicioAccesoDatos/` Contiene las clases necesarias para la creación y manipulación de los diferentes tipos de ficheros utilizados en la aplicación.

### 3. Bibliotecas utilizadas

- **Java.swing.\* (Versión anterior)** Proporciona un conjunto de componentes gráficos para la interfaz (eventos, colores, tipos de letra, botones, etc.) que trabajan en cualquier plataforma. A diferencia de AWT, no utiliza código nativo por lo cual ofrece mayor funcionalidad y asegura la portabilidad.
- **Java.awt.\* (Versión anterior)** Abstract Windowing Toolkit proporciona una capa abstracta que permite llevar una aplicación en Java de un sistema de ventanas a otro. Contiene clases para componentes básicos de la interfaz, tales como eventos, colores, tipos de letra, botones, campos de texto, etc.
- **Java.io.\* (Versión anterior)** Biblioteca estándar de entrada y salida. Archivos de stream y acceso aleatorio.
- **Java.util.\* (Versión anterior)** Clases como diccionarios, tabla de hash, pilas, técnicas de codificación y decodificación, hora, fecha, etcétera.
- **Java.lang (Versión anterior)** Colección de tipos básicos siempre importados a cualquier unidad de compilación. Aquí están las declaraciones de objetos, clases, hilos, excepciones, envolturas de los tipos de datos primitivos y otras clases fundamentales.
- **AbsoluteLayout [9]** Se utiliza para colocar elementos en la interfaz gráfica mediante valores explícitos.
- **jfreechart [7]** Biblioteca que facilita la visualización de gráficos de calidad.
- **jcommon [12]** Biblioteca necesaria para *jfreechart*.
- **jdom [11]** Biblioteca para la manipulación de ficheros XML, optimizada para Java.

- **jnetpcap [19]** Biblioteca Java contenedora de las bibliotecas *winPcap* y *libPcap*, escritas en C. Permite la captura y el envío de paquetes por la red, así como la visualización de la información que estos contienen.
- **net.sf.fjep.fatjar [4]** Plug-in de Eclipse utilizado en versiones anteriores que permite crear un JAR con el proyecto Java desplegado.
- **net.sourceforge.jpcap [3]** Biblioteca que permite a los desarrolladores la captura de tráfico en sus propias aplicaciones de captura de paquetes. En algunos aspectos funciona mejor que *jNetPcap*, y por ello sigue utilizándose en algunas clases.
- **swing-layout [16]** Biblioteca que provee diferentes gerentes de diseño para la interfaz gráfica.

## D.4. Compilación, instalación y ejecución del proyecto

### 1. Introducción

En este apartado, se explican de forma detallada los pasos necesarios para modificar y utilizar la aplicación.

### 2. Instalación

Nuestra aplicación no necesita una instalación propia, pero sí de las bibliotecas que utiliza. Por lo tanto, para poder utilizar la aplicación, se deberán seguir los pasos mencionados en el apartado “E.3. Instalación”, dentro del apéndice “E. Documentación de usuario”.

En cuanto a la modificación de la aplicación, se requiere la aplicación Eclipse, disponible en [5]. Una vez descarguemos el instalador y lo ejecutemos, únicamente deberemos elegir el paquete que queremos, en nuestro caso: *Eclipse IDE for Java Developers*.

Con ello, tendremos la aplicación a punto, y podremos abrir el proyecto en el directorio “\Sniffer\”.

Para la modificación de la biblioteca *jNetPcap*, se ha seleccionado el proyecto encontrado en el repositorio “\jnetpcap-original\”, correspondiente a la última versión oficial de la biblioteca. Después, se ha creado otro proyecto con las modificaciones realizadas, que encontramos en “\jNetPcap\”. Este

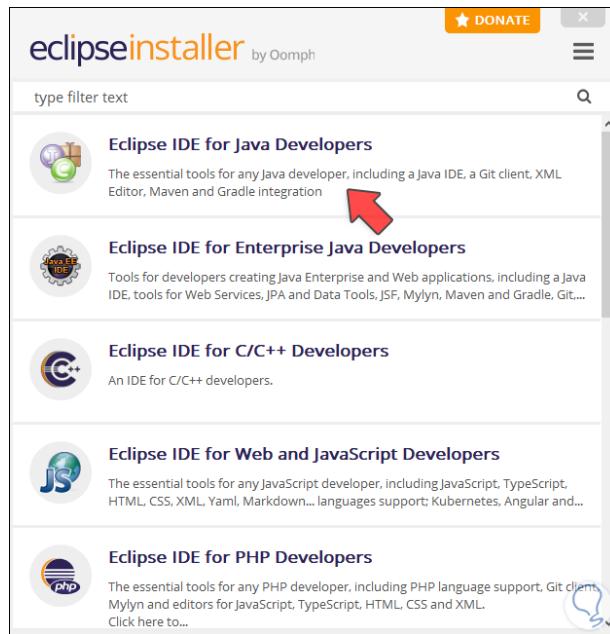


Figura D.1: Instalación de la aplicación Eclipse.

proyecto utiliza Ant, y se ha realizado en Windows. Un prerequisito es tener el JDK instalado, siendo su versión igual o superior a 1.6.0. La instalación, como tal, es la siguiente:

1. Descargar el comprimido en [18].
2. Descomprimir el archivo comprimido en el disco duro.
3. Establecer la variable de entorno *ANT\_HOME* en el directorio donde se haya descomprimido el archivo descargado.

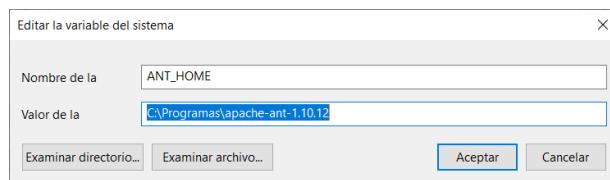


Figura D.2: Variable de entorno de Ant.

4. Añadir la fila *ANT\_HOME\bin* a la variable de entorno *Path*.

### 3. Ejecución

Para ejecutar nuestra aplicación, solo hace falta ejecutar el archivo “Sniffer-IV.bat”, para Windows, o el archivo “Sniffer-IV.sh”, para Linux.

En cuanto a la ejecución en Eclipse, solo debemos ejecutar la clase *Sniffer.java*, encontrada en el paquete por defecto.

Para la construcción de la biblioteca *jNetPcap*, deberemos ejecutar el fichero “build.xml”, seleccionando primero la opción por defecto, y luego la tarea *build-jar*.

## D.5. Comparativa JNI vs jNetPcap

La investigación inicial de las diferentes bibliotecas que podría usar nuestra aplicación, llevó a encontrarnos con la biblioteca actual, *jNetPcap*, y con las bibliotecas que esta misma utiliza, *libPcap* y *winPcap*, aunque esta última debería ser sustituida por *nPcap*.

Por lo tanto, existían dos opciones para mejorar la aplicación: la mejora de *jNetPcap*, una biblioteca Java que parece bastante intuitiva, o el uso de *JNI* en la aplicación, que es una práctica muy compleja para programadores no experimentados, con el fin de utilizar las bibliotecas escritas en lenguaje C.

Aún así, hay un aspecto aún más importante por el cual se decidió la opción contraria a *JNI*. La implementación de esta herramienta en la aplicación llevaba a abrir mucho los objetivos posibles, y no se sabía de forma clara hacia dónde avanzar. Además, las mejoras realizadas en *jNetPcap* podrían servir en un futuro en caso de que se implemente *JNI*, dado que varias de las clases que contiene *jNetPcap* podrían ser incluidas en la aplicación.

## D.6. Mejora de la biblioteca jNetPcap

Uno de los objetivos más destacables de este proyecto es la mejora realizada sobre la biblioteca *jNetPcap*. Por ello, se ha considerado relevante la inclusión de este apartado en la documentación técnica.

Primero de todo, para realizar una mejora en una biblioteca desconocida, se requiere cierto tiempo para “familiarizarse” con ella, lo mismo que para cualquier otro proyecto desconocido. Una vez se conocía de forma básica su funcionamiento, se realiza la búsqueda de información de los protocolos a implementar, con el fin de encontrar la estructura de cada uno de ellos. Los

mejores documentos para comprender este aspecto de los protocolos, son las RFCs [13]. Estas se tratan de unos documentos que definen estándares correspondientes a nuevos protocolos, entre otros aspectos relacionados con las comunicaciones. Para DNS, se encuentra en [17]; para ICMPv6, en [1], y para IGMP, en [2]. Estos protocolos se encuentran explicados detalladamente en los *Conceptos teóricos* del documento de la memoria.

Teniendo ya unas fuentes fiables sobre las que basarnos para construir los protocolos, era necesario observar la programación de los protocolos ya implementados en la biblioteca.

Con toda la información obtenida llegados a este punto, se sitúan las clases nuevas en los paquetes /org/jnetpcap/protocol/application, para DNS, un protocolo de la capa de aplicación, y /org/jnetpcap/protocol/network, para ICMPv6 e IGMP, ambos dos protocolos de la capa de red. Se deben escoger las clases que extienden y/o implementan, basándose en la programación de los demás protocolos. Algunas de estas clases mencionadas son:

- JHeaderMap
- JHeader
- JHeaderChecksum

Comenzando con la programación, se crean los diferentes métodos, inicialmente vacíos, que devolverán los valores correspondientes a los campos definidos en las RFCs.

Posteriormente, utilizamos el paquete /org/jnetpcap/packet/annotate para crear anotaciones del *offset*, o la dirección en la que comienza el campo, y de la longitud del campo, para cada método.

Con estas indicaciones, debemos escoger cuidadosamente los métodos necesarios para la obtención del valor correspondiente al campo buscado, que se encuentran definidos en la clase o en las clases extendidas/implementadas. Entre ellos tenemos:

- `super.getUByte(int offset);`

Este método selecciona la información de los 8 bits posteriores al *offset* pasado como argumento, y lo convierte en un entero de 8 bits sin signo.

- `super.getUShort(int offset);`

Este método selecciona la información de los 16 bits posteriores al *offset* pasado como argumento, y lo convierte en un entero de 16 bits sin signo.

- `super.getInt(int offset);`

Este método selecciona la información de los 32 bits posteriores al *offset* pasado como argumento, y lo convierte en un entero de 32 bits sin signo.

El *offset* pasado como argumento viene dado en bytes y, como se puede ver en la explicación detallada de los protocolos en los *Conceptos teóricos* de la memoria, en algunos de ellos los campos pueden ser banderas o *flags*, por ejemplo, que pueden tener una longitud de unos pocos bits, o que pueden comenzar en una dirección no múltiplo de 8.

Para resolver estos campos, se debe coger el byte o los bytes que contengan ese campo y realizar alguna operación como la del siguiente ejemplo:

```
return (getUShort(2) & 0x7800) >> 11;
```

En esta ocasión, se trata de una bandera de entre un conjunto de estas, que ocupan en total 2 bytes. Esta bandera se encuentra en el bit 17, dado que le precede otra bandera de un solo bit.

Por lo tanto, en la operación anterior, se comienza cogiendo 32 bits y se les aplica una máscara mediante la operación lógica “AND” para descartar los valores de las otras banderas. Dicha máscara es, en binario, la siguiente:

0111 1000 0000 0000 = 0x7800

Así, obtenemos un valor no correspondiente al verdadero. Para ello, se desplazan esos 4 bits a las últimas 4 posiciones, mediante una operación de desplazamiento de 11 bits, en este caso.

Para una mayor comprensión, se trata del campo OPCODE, presente en el protocolo DNS, cuya estructura es la siguiente:

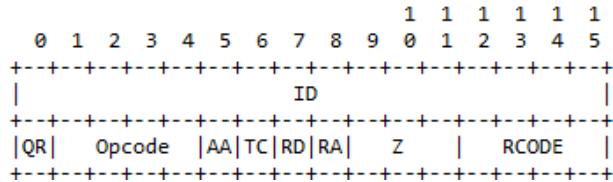


Figura D.3: Primeros campos del protocolo DNS.

En concreto, para los primeros campos del protocolo DNS tenemos el siguiente código:

```
@Protocol(suite = Suite.APPLICATION)
@Header(nicname = "Dns", suite = ProtocolSuite.APPLICATION)
public class DNS extends JHeaderMap<DNS> {

    /** Constant numerical ID assigned to this protocol. */
    public final static int ID = JProtocol.DNS_ID;

    @Field(offset = 0, length = 2 * 8)
    public int identification() {
        return super.getUShort(0);
    }

    public final static int FLAG_QR = 0x8000;
    public final static int FLAG_OPCODE = 0x7800;
    public final static int FLAG_AA = 0x400;
    public final static int FLAG_TC = 0x200;
    public final static int FLAG_RD = 0x100;
    public final static int FLAG_RA = 0x80;
    public final static int FLAG_Z = 0x70;
    public final static int FLAG_RCODE = 0xF;

    @Field(offset = 2 * 8, length = 2 * 8, format = "%x")
    public int flags() {
        return getUShort(2);
    }

    @Field(parent = "flags", offset = 0, length = 1, display = "QR")
    public int flags_QR() {
        return (flags() & FLAG_QR) >> 15;
    }
}
```

```

@Dynamic(Field.Property.DESCRIPTION)
public String flags_QRDescription() {
    return (flags_QR() > 0) ? "1 (response)" : "0 (query)";
}

@Field(parent = "flags", offset = 1, length = 4, display = "Opcode")
public int flags_Opcode() {
    return (flags() & FLAG_OPCODE) >> 11;
}

@Dynamic(Field.Property.DESCRIPTION)
public String flags_OpcodeDescription() {
    String aux = "";
    switch(flags_Opcode()) {
        case 0:
            aux = "0 (standard query (QUERY))";
            break;
        case 1:
            aux = "1 (inverse query (IQUERY))";
            break;
        case 2:
            aux = "2 (server status request (STATUS))";
            break;
        default:
            aux = "3-15 (reserved for future use)";
            break;
    }
    return aux;
}

@Field(parent = "flags", offset = 5, length = 1, display = "AA")
public int flags_AA() {
    return (flags() & FLAG_AA) >> 10;
}

@Dynamic(Field.Property.DESCRIPTION)
public String flags_AADescription() {
    return (flags_AA() > 0) ? "1" : "0";
}

```

```

@Field(parent = "flags", offset = 6, length = 1, display = "TC")
public int flags_TC() {
    return (flags() & FLAG_TC) >> 9;
}

@Dynamic(Field.Property.DESCRIPTION)
public String flags_TCDescription() {
    return (flags_TC() > 0) ? "1" : "0";
}

@Field(parent = "flags", offset = 7, length = 1, display = "RD")
public int flags_RD() {
    return (flags() & FLAG_RD) >> 8;
}

@Dynamic(Field.Property.DESCRIPTION)
public String flags_RDDescription() {
    return (flags_RD() > 0) ? "1" : "0";
}

@Field(parent = "flags", offset = 8, length = 1, display = "RA")
public int flags_RA() {
    return (flags() & FLAG_RA) >> 7;
}

@Dynamic(Field.Property.DESCRIPTION)
public String flags_RADescription() {
    return (flags_RA() > 0) ? "1" : "0";
}

@Field(parent = "flags", offset = 9, length = 3, display = "Z")
public int flags_Z() {
    return (flags() & FLAG_Z) >> 4;
}

@Dynamic(Field.Property.DESCRIPTION)
public String flags_ZDescription() {
    return (flags_Z() > 0) ? "1 (error)" : "0";
}

```

```
@Field(parent = "flags", offset = 12, length = 4, display = "RCODE")
public int flags_RCODE() {
    return flags() & FLAG_RCODE;
}

@Dynamic(Field.Property.DESCRIPTION)
public String flags_RCODEDescription() {
    String aux = "";
    switch(flags_Opcode()) {
        case 0:
            aux = "0 (No error condition)";
            break;
        case 1:
            aux = "1 (Format error)";
            break;
        case 2:
            aux = "2 (Server failure)";
            break;
        case 3:
            aux = "3 (Name Error)";
            break;
        case 4:
            aux = "4 (Not Implemented)";
            break;
        case 5:
            aux = "5 (Refused)";
            break;
        default:
            aux = "6-15 (Reserved for future use)";
            break;
    }
    return aux;
}
```

## D.7. Pruebas del sistema

En este apartado, se describen las pruebas realizadas sobre la aplicación. Estas son extremadamente importantes, dado que una aplicación no puede ser entregada al cliente sin haber sido probada. Esto podría ocasionar que el cliente encontrase errores en la aplicación, como es obvio.

Para un mayor detallado de las pruebas realizadas, estas se recogen en la siguiente tabla:

ID Prueba	Nombre	Resultado esperado	Resultado obtenido
1	Entrar a la aplicación	Se muestra el Menú Inicial.	OK
2	Salir de la aplicación	Sale de la aplicación.	OK
3	Acceder a la interfaz gráfica	Se muestra la interfaz gráfica.	OK
4	Salir de la interfaz gráfica	Sale de la aplicación, previa confirmación.	OK
5	Acceder al modo comando	Se muestra el menú del modo comando.	OK
6	Salir del modo comando	Sale de la aplicación.	OK
7	Acceder a la modificación de la Memoria Virtual JVM	Accede al menú para modificar la Mamoria Virtual de la JVM.	OK
8	Salir de la modificación de la Memoria Virtual JVM	Sale de la aplicación.	OK
9	Selección del menú Archivo	Se despliega el menú y muestra todas las opciones.	OK
10	Selección del menú Captura	Se despliega el menú y muestra todas las opciones.	OK
11	Selección del menú Definición	Se despliega el menú y muestra todas las opciones.	OK
12	Selección del menú Inserción	Se despliega el menú y muestra todas las opciones.	OK

ID Prueba	Nombre	Resultado esperado	Resultado obtenido
13	Selección del menú Parámetrización	Se despliega el menú y muestra todas las opciones.	OK
14	Selección del menú Ayuda	Se despliega el menú y muestra todas las opciones.	OK
15	“Archivo -> Abrir fichero de capturas” y seleccionar un fichero	Se muestra la información capturada del fichero seleccionado, en los diferentes paneles de la interfaz gráfica.	OK
16	“Archivo -> Guardar fichero capturado” y guardar la captura abierta, con un nombre y en una ubicación concretos, comprobando después que el guardado es correcto	Se guarda el fichero tal y como se ha dicho.	OK
17	“Archivo -> Exportar -> Captura a XML” y exportar la captura abierta, con un nombre y en una ubicación concretos, comprobando después que la exportación es correcta	Se exporta la captura tal y como se ha dicho.	OK
18	“Archivo -> Exportar -> Desde fichero a XML”, se selecciona un fichero con formato “.pcap” y se exporta, con un nombre y en una ubicación concretos, comprobando después que la exportación es correcta	Se exporta el fichero tal y como se ha dicho.	OK
19	“Archivo -> Configuración”, se modifica el campo “Capturas” y se comprueba que ahora se guardan en la ubicación nueva	Las nuevas capturas se guardan en la nueva ubicación.	OK

ID Prueba	Nombre	Resultado esperado	Resultado obtenido
20	“Archivo -> Salir”	Sale de la aplicación, previa confirmación.	OK
21	“Captura -> Iniciar captura”, se realizan varias capturas con diferentes filtros y se comprueba que funcionan.	Las capturas se realizan en base a los filtros de forma correcta.	OK
22	“Captura -> Finalizar captura”	Finaliza la captura.	OK
23	“Captura -> Captura desde fichero”, se realizan varias capturas con diferentes filtros y se comprueba que funcionan.	Las capturas se realizan en base a los filtros de forma correcta.	OK
24	“Definición -> Definición de paquetes”, se realizan la definición de un nuevo paquete.	El paquete es generado correctamente.	OK
25	“Inserción -> Inserción paquetes definidos” y se realiza la inserción de un paquete con unos datos básicos.	La inserción se realiza correctamente.	OK
26	“Inserción -> Inserción de paquetes capturados” y se realiza la inserción del paquete que seleccionamos.	La inserción se realiza correctamente.	OK
27	“Parametrización -> Captura” y se crea un fichero nuevo con valores aleatorios.	El fichero se crea de forma correcta.	OK
28	“Parametrización -> Captura” y se abre un fichero previamente creado con valores aleatorios.	Las preferencias se actualizan correctamente.	OK
29	“Parametrización -> Exportación” y se crea un fichero nuevo con valores aleatorios.	El fichero se crea de forma correcta.	OK
30	“Parametrización -> Exportación” y se abre un fichero previamente creado con valores aleatorios.	Las preferencias se actualizan correctamente.	OK

ID Prueba	Nombre	Resultado esperado	Resultado obtenido
31	“Parametrización -> Captura desde fichero” y se crea un fichero nuevo con valores aleatorios.	El fichero se crea de forma correcta.	OK
32	“Parametrización -> Captura desde fichero” y se abre un fichero previamente creado con valores aleatorios.	Las preferencias se actualizan correctamente.	OK
33	“Parametrización -> Detalle paquetes” y se crea un fichero nuevo con valores aleatorios.	El fichero se crea de forma correcta.	OK
34	“Parametrización -> Detalle paquetes” y se abre un fichero previamente creado con valores aleatorios.	Las preferencias se actualizan correctamente.	OK
35	“Parametrización -> Generar script” y se crea un fichero nuevo con valores aleatorios.	El fichero se crea de forma correcta.	OK
36	“Parametrización -> Generar script” y se abre un fichero previamente creado con valores aleatorios.	Las preferencias se actualizan correctamente.	OK
37	“Ayuda -> Contenidos”	La ayuda se abre en el navegador.	OK
38	“Ayuda -> Acerca de Sniffer IV”	Se muestra la ventana “Acerca De”.	OK
39	Comprobar implementación del protocolo <i>Ethernet</i> , mediante la apertura de un fichero de capturas que contenga paquetes de este protocolo, descargado de Internet	El protocolo es reconocido correctamente.	OK

ID Prueba	Nombre	Resultado esperado	Resultado obtenido
40	Comprobar implementación del protocolo <i>IPv4</i> , mediante la apertura de un fichero de capturas que contenga paquetes de este protocolo, descargado de Internet	El protocolo es reconocido correctamente.	OK
41	Comprobar implementación del protocolo <i>ARP</i> , mediante la apertura de un fichero de capturas que contenga paquetes de este protocolo, descargado de Internet	El protocolo es reconocido correctamente.	OK
42	Comprobar implementación del protocolo <i>TCP</i> , mediante la apertura de un fichero de capturas que contenga paquetes de este protocolo, descargado de Internet	El protocolo es reconocido correctamente.	OK
43	Comprobar implementación del protocolo <i>UDP</i> , mediante la apertura de un fichero de capturas que contenga paquetes de este protocolo, descargado de Internet	El protocolo es reconocido correctamente.	OK
44	Comprobar implementación del protocolo <i>Http</i> , mediante la apertura de un fichero de capturas que contenga paquetes de este protocolo, descargado de Internet	El protocolo es reconocido correctamente.	OK
45	Comprobar implementación del protocolo <i>UDP</i> , mediante la apertura de un fichero de capturas que contenga paquetes de este protocolo, descargado de Internet	El protocolo es reconocido correctamente.	OK

ID Prueba	Nombre	Resultado esperado	Resultado obtenido
46	Comprobar implementación del protocolo <i>IPv6</i> , mediante la apertura de un fichero de capturas que contenga paquetes de este protocolo, descargado de Internet	El protocolo es reconocido correctamente.	ERROR
47	Comprobar implementación del protocolo <i>ICMP</i> , mediante la apertura de un fichero de capturas que contenga paquetes de este protocolo, descargado de Internet	El protocolo es reconocido correctamente.	ERROR
48	Comprobar implementación del protocolo <i>DNS</i> , mediante la apertura de un fichero de capturas que contenga paquetes de este protocolo, descargado de Internet	El protocolo es reconocido correctamente.	ERROR
49	Comprobar implementación del protocolo <i>ICMPv6</i> , mediante la apertura de un fichero de capturas que contenga paquetes de este protocolo, descargado de Internet	El protocolo es reconocido correctamente.	ERROR
50	Comprobar implementación del protocolo <i>IGMP</i> , mediante la apertura de un fichero de capturas que contenga paquetes de este protocolo, descargado de Internet	El protocolo es reconocido correctamente.	ERROR

Tabla D.1: Pruebas del sistema realizadas.

Como se puede ver, la aplicación no reconoce los nuevos protocolos implementados (DNS, ICMPv6 e IGMP), pero tampoco algunos de los que están implementados en la última versión oficial de la biblioteca *jNetPcap*, como pasa con IPv6 e ICMP.

Además, se han realizado una serie de pruebas multiplataforma, todas con resultado positivo, que encontramos a continuación:

## Pruebas multiplataforma

Sistema Operativo	Tipo de arquitectura	Versión
Windows	x64	Windows 10 Home
Windows	x32	Windows 10 Home
Linux	x64	Ubuntu 22.04 LTS
Linux	x32	Ubuntu 17.04

Tabla D.2: Pruebas multiplataforma realizadas.

## *Apéndice E*

---

# **Documentación de usuario**

---

## **E.1. Introducción**

En este documento se definen los requisitos mínimos que deberá tener un usuario para la ejecución de la aplicación. Además, se explica detalladamente el proceso de instalación para los diferentes sistemas operativos y los pasos a seguir para utilizar cada una de las funcionalidades de la aplicación.

Cabe destacar que este anexo se basa en las versiones anteriores de otros proyectos, cambiando prácticamente solo las versiones, a parte de algún error encontrado al seguir el manual.

## **E.2. Requisitos de usuarios**

Los usuarios que deseen utilizar la aplicación necesitarán unos requisitos mínimos tanto de hardware como de software.

### **Requisitos hardware**

Se necesita, como mínimo, un equipo con los siguientes componentes:

- Procesador de 32 ó 64 bits.
- Procesador con frecuencia base de 1 Ghz (gigahercios).
- Memoria RAM de 8 GB (gigabytes).
- Disco duro con 100 GB libres.

- Tarjeta de red.

## Requisitos software

En cuanto al software, se necesitan los siguientes mínimos:

### Windows

- Windows 10.
- Java VM 1.8.0.321.
- jNetPcap 1.4.
- WinPcap 4.1.3.

### Linux

- Ubuntu 22.04 (x64) o Ubuntu 17.04 (x32).
- Java VM 1.8.0.321.
- jNetPcap 1.4.
- LibPcap 1.10.1.
- Bison 3.8.2.
- Flex 2.6.4.
- M4 1.4.19.

## E.3. Instalación

Dado que la aplicación soporta los sistemas operativos Windows y Linux, dividiremos la instalación en dos subapartados, uno para cada sistema operativo.

### Para Windows

Los pasos seguidos han sido realizados sobre la versión Windows 10, pero no se debería poder ver un gran cambio sobre otras versiones. Principalmente, podrían variar los nombres que encontraremos en el *Panel de Control*.

### Java - JDK [14]

Para instalar el JDK de Java, simplemente hay que ejecutar el archivo ejecutable, valga la redundancia, que hay en la carpeta “Herramientas\_Instalacion\_Sniffer\Java”, donde se encontrará el archivo concreto para cada arquitectura (“jdk-8u321-windows-x64.exe” para x64 y “jdk-8u321-windows-i586.exe” para x32).

Se prefiere cambiar el directorio de instalación para realizar una instalación más sencilla. Por lo tanto, en la siguiente pantalla, Figura E.1, pulsaremos el botón “Change...”.

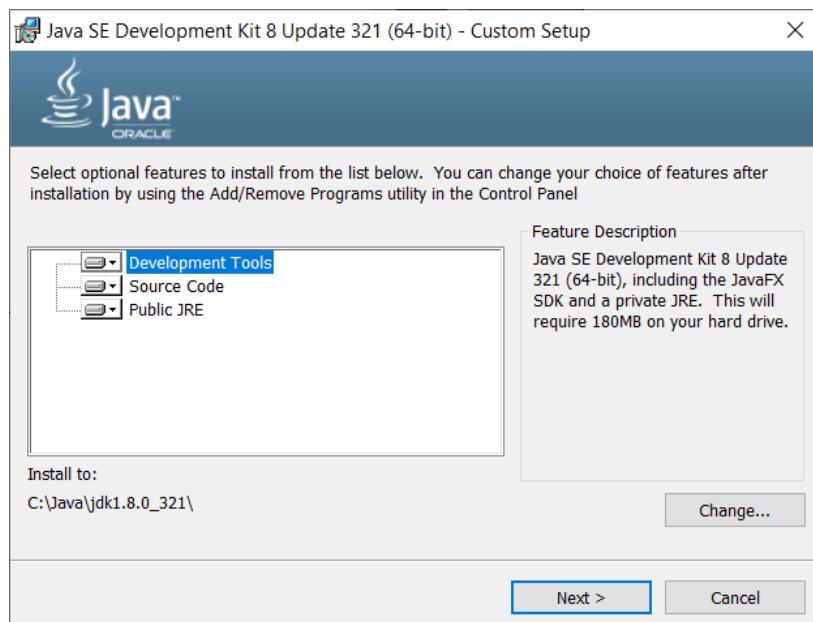


Figura E.1: Instalación JDK 1.

Esto nos llevará a la Figura E.2, donde estableceremos el directorio “C:\Java\jdk-1.8.0\_321\”.

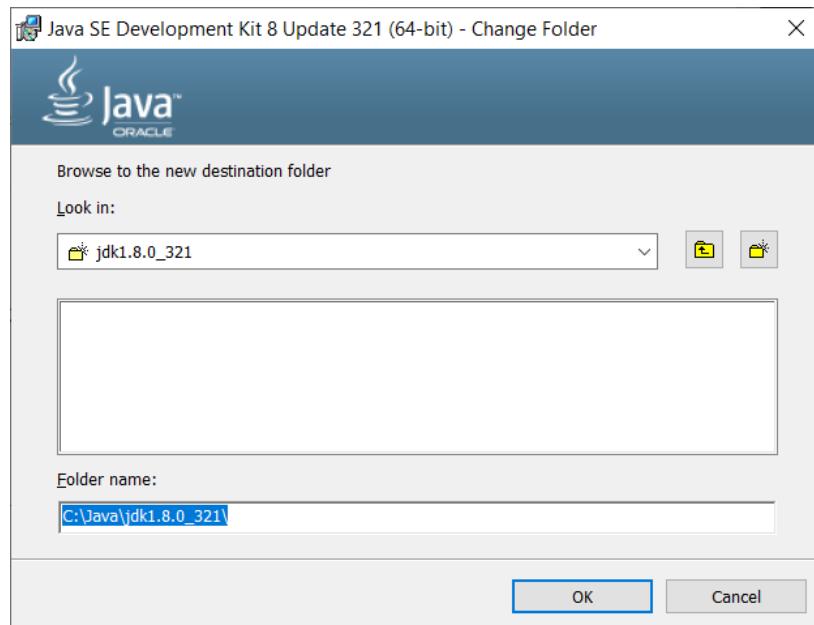


Figura E.2: Instalación JDK 2.

Para el siguiente paso, dejamos la ruta predeterminada y pulsaremos “Siguiente”, hasta que nos aparezca el botón “Close” y se complete la instalación.

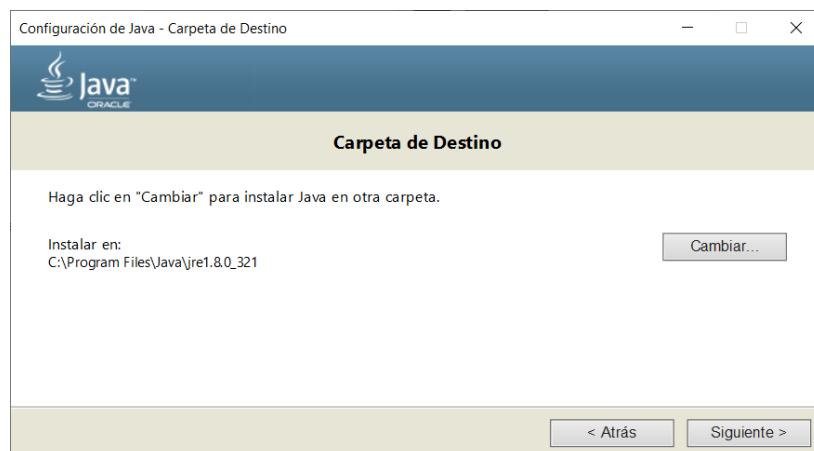


Figura E.3: Instalación JDK 3.

### Variables de entorno

Para añadir la variable de entorno, correspondiente al JDK recién instalado, accederemos al *Panel de Control* del sistema, donde encontraremos el apartado *Sistema y seguridad*. Una vez entremos en ese apartado, haremos *click* sobre la opción *Sistema*. En esta nueva ventana nos aparecerán varios enlaces a la derecha. El que nos interesa es *Configuración avanzada del sistema*, el cual contendrá un botón con el texto *Variables de entorno*, y que nos abrirá la pantalla de la Figura E.4.

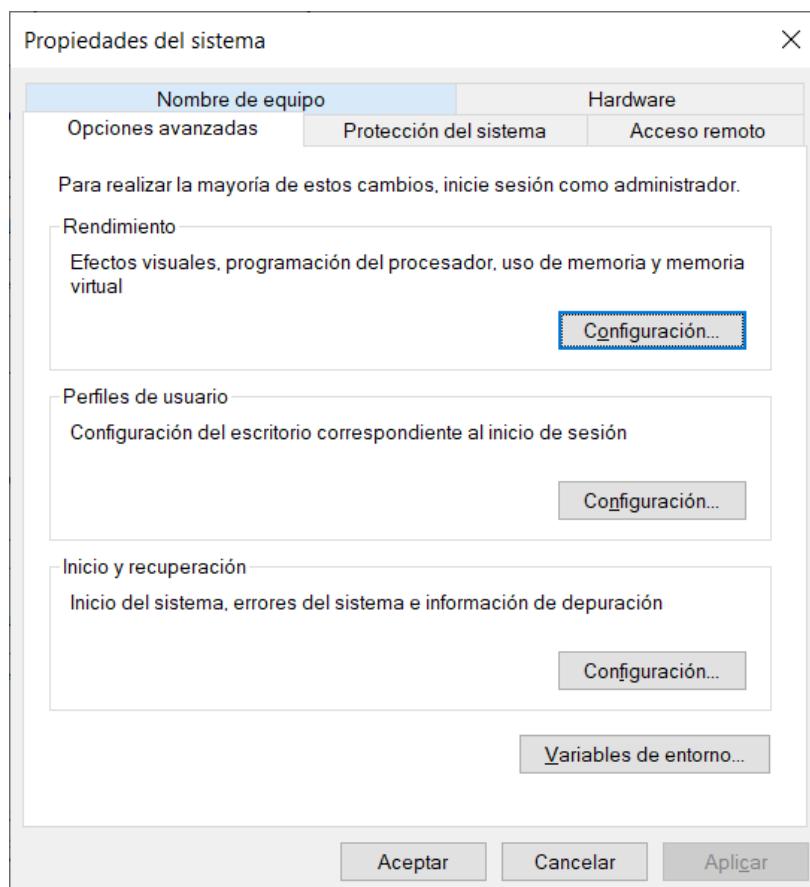


Figura E.4: Variables de entorno 1.

Aquí, podemos ver dos apartados, divididos horizontalmente. El primero se usará cuando queramos modificar las variables de entorno de nuestro usuario en concreto, mientras que el que se encuentra en la parte inferior, contiene las variables del sistema, es decir, aquellas comunes a todos a los usuarios.

Para cualquiera de las dos opciones, hay que pulsar el botón “Nueva...”, que nos permitirá nombrar a nuestra variable, en este caso, *JAVA\_HOME*, y establecer la ruta, que corresponde con la introducida previamente al instalar el JDK.

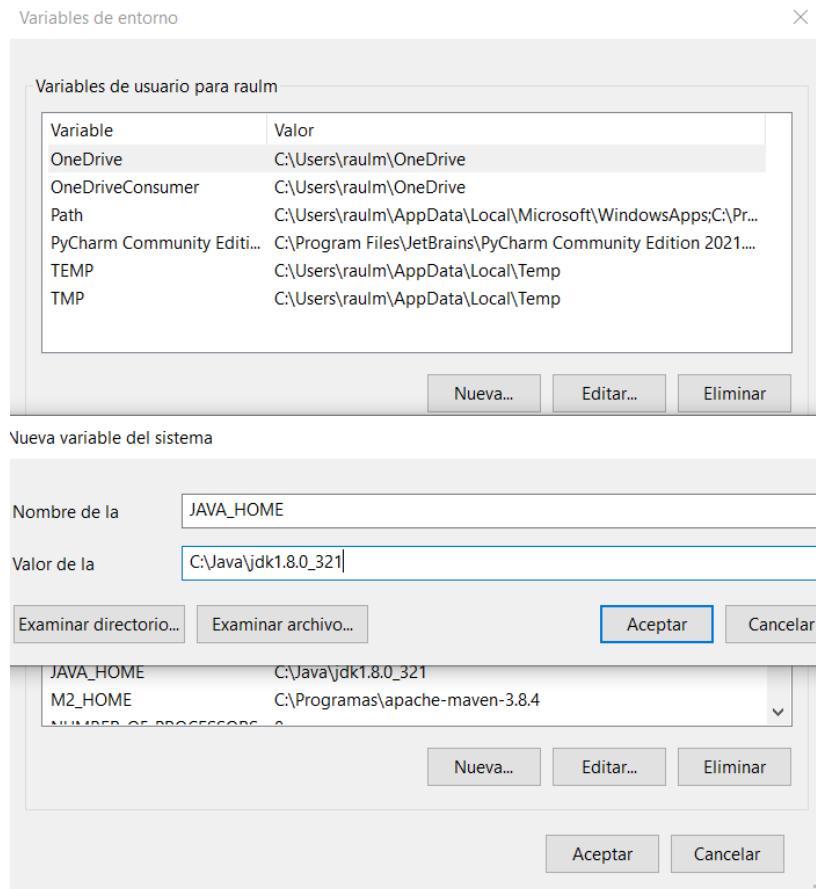


Figura E.5: Variables de entorno 2.

Una vez tengamos el paso anterior, aceptamos para guardar la nueva variable. Después, modificamos la variable *Path* seleccionando esta variable y pulsando el botón “Editar...”. En esta, añadiremos “%JAVA\_HOME%\bin” para hacer la variable anterior efectiva.

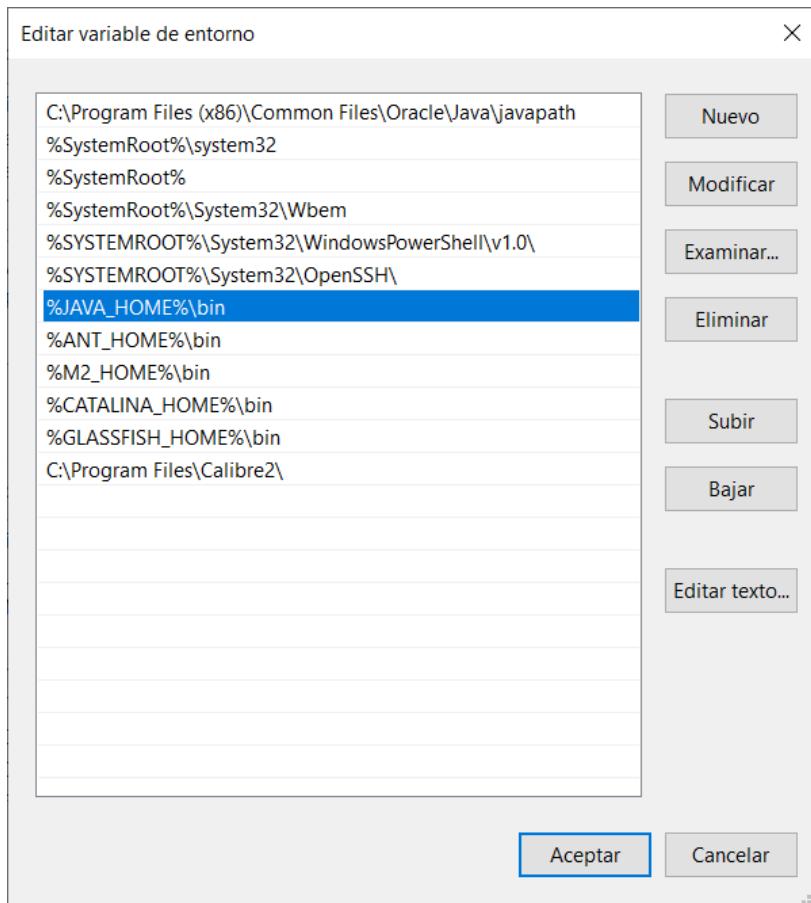


Figura E.6: Variables de entorno 3.

### WinPcap [20]

Accedemos a la carpeta “Herramientas\_Instalacion\_Sniffer” y ejecutamos “WinPcap\_4\_1\_3.exe”.

Aceptar la licencia y realizar la instalación por defecto.

### jNetPcap [19]

Dentro de la carpeta “Herramientas\_Instalacion\_Sniffer\jNetPcap”, se encuentra la biblioteca “jnetpcap.dll”. Solo será necesario copiar este fichero en los directorios del sistema “C:\Windows\System” y “C:\Windows\System32”.

Es posible que funcione sin este paso, ya que puede valer con tener este archivo en la carpeta “dll”, dentro del directorio del proyecto de Eclipse, incluso en el mismo directorio que el ejecutable.

## Tarjeta de red en modo promiscuo [21]

Para capturar todo el tráfico que circula por la red, es necesario establecer el modo promiscuo de nuestra tarjeta de red.

Existen dos posibilidades de realizar este paso, una utilizando Wireshark y otra por comandos.

### Con comandos

Existe otra posibilidad para la configuración de la tarjeta de red.

Se debe ejecutar la consola de comandos (*cmd*) como administrador e introducir el siguiente comando, para averiguar el identificador de la tarjeta:

```
netsh bridge show adapter
```

Cuando conozcamos el identificador, utilizaremos el siguiente comando, sustituyendo *ID* por el número correspondiente:

```
netsh bridge set adapter ID forcecompatmode=enable
```

### Con Wireshark

Se utilizará la aplicación *Wireshark*, ejecutándola como Administrador, para que detecte las interfaces de red.

El instalador de esta aplicación se encuentra en el directorio “Herramientas\_Instalacion\_Sniffer\Wireshark”, con el nombre correspondiente a su arquitectura (“Wireshark-win64-3.6.8.exe” para x64 y “Wireshark-win32-3.6.8.exe” para x32).

Es necesario aceptar la licencia, pero se pueden dejar por defecto el resto de opciones, tanto para la pantalla de la Figura E.7, como para la de la Figura E.8.

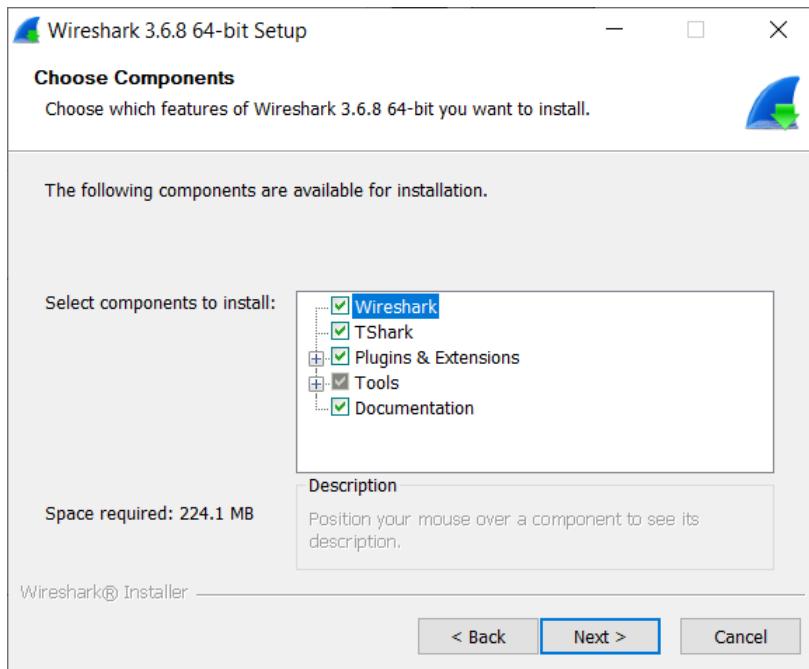


Figura E.7: Instalación de Wireshark 1.

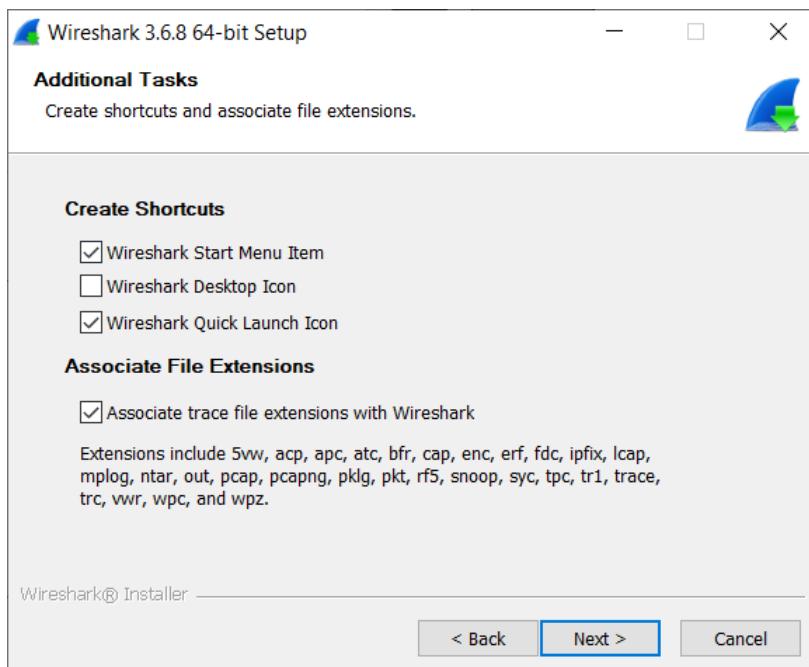


Figura E.8: Instalación de Wireshark 2.

En cierto momento permite instalar también *winPcap* o *nPcap*, cosa que solo haremos si no lo hemos hecho previamente.

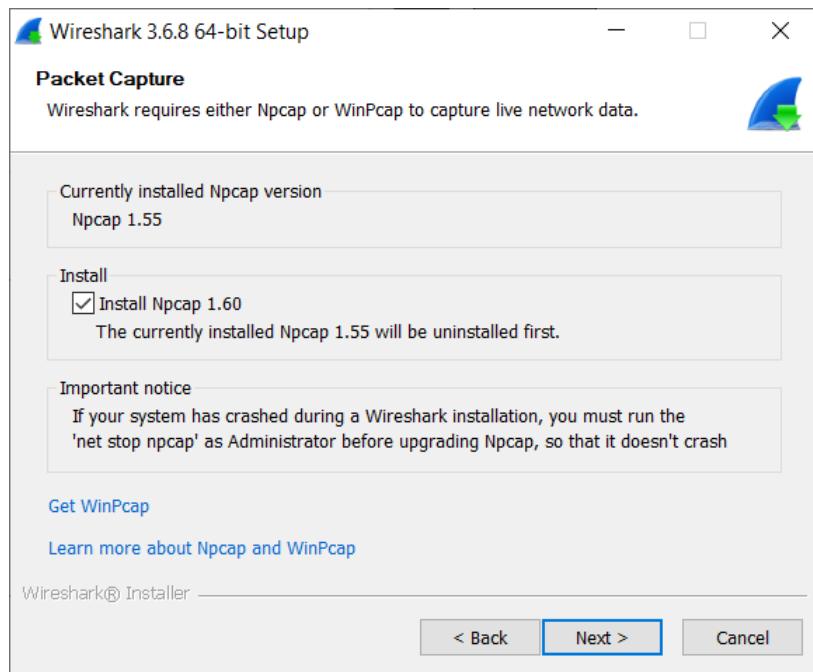


Figura E.9: Instalación de Wireshark 3.

Se deja a su elección la opción de instalar *USBPcap* en la siguiente pantalla, ya que en esta versión no es necesaria.

Una vez la aplicación esté instalada, será necesario realizar una captura, pulsando el botón con el logo de la aplicación, en la esquina superior izquierda.

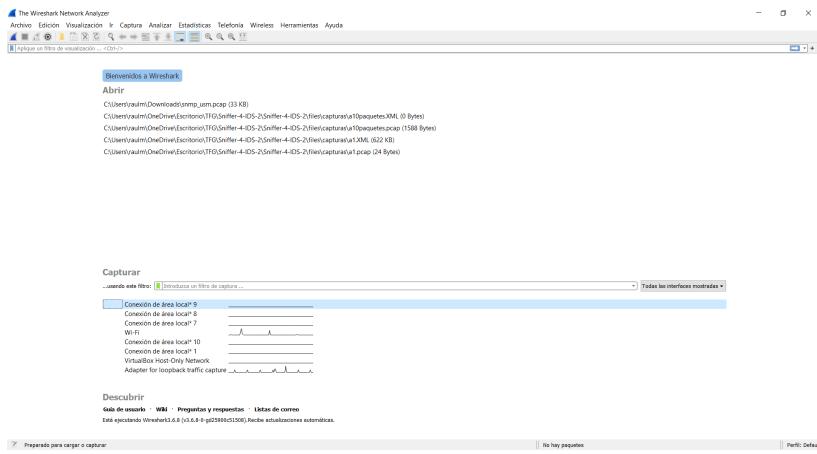


Figura E.10: Instalación de Wireshark 4.

Con esto, la tarjeta de red ya estaría en modo promiscuo.

## Para Linux

Los pasos seguidos han sido realizados sobre la versión Ubuntu 22.04 LTS, pero no se debería poder ver un gran cambio sobre otras versiones.

### Java - JDK [14]

Primero, creamos un nuevo directorio con el nombre *java*, dentro del directorio del sistema “*/usr*”, con el siguiente comando:

```
mkdir -m 777 /usr/java
```

De la carpeta “Herramientas\_Instalacion\_Sniffer\Java”, debemos copiar el archivo correspondiente a la arquitectura a un directorio del sistema, por ejemplo, “Descargas”.

Para desempaquetar e instalar el JDK en el directorio creado, debemos situarnos en el directorio “Descargas” y ejecutar el siguiente comando:

```
tar zxvf jdk-8u321-linux-x64.tar.gz -directory /usr/java
```

En este punto, se puede eliminar el archivo comprimido (.tar.gz), situándonos en “Descargas”, con el comando:

```
rm jdk-8u321-linux-x64.tar.gz
```

Finalmente, creamos los accesos directos:

```
sudo update-alternatives --install /usr/bin/java java
/usr/java/jdk1.8.0_321/bin/java 1
```

```
sudo update-alternatives --install /usr/bin/javac javac
/usr/java/jdk1.8.0_321/bin/javac 1
```

## Variables de entorno

Para configurar la variable de entorno, debemos el archivo “bashrc” con algún editor de texto, en este caso, gEdit:

```
sudo gedit /etc/bash.bashrc
```

Al final del archivo añadir las siguientes líneas:

```
export JAVA_HOME=/usr/java/jdk1.8.0_321
```

```
export PATH=$PATH:$JAVA_HOME/bin
```

Debemos guardar los cambios y reiniciar la máquina para que estos sean efectivos.

## jNetPcap [19]

Dentro de la carpeta “Herramientas\_Instalacion\_Sniffer\jNetPcap” copiar el archivo “jnetpcap-1.4.r1425-1.linux64.x86\_64.tgz” a una carpeta de Linux, por ejemplo en “Descargas”.

Situarse en “Descargas” y desempaquetar en la ruta “/usr/jnetpcap” con el siguiente comando:

**Nota:** Si la ruta no existe, crearla como se ha especificado antes, con el comando “mkdir”.

```
tar zxvf jnetpcap-1.4.r1425-1.linux64.x86_64.tgz --directory /usr/jnetpcap
```

Copiar los archivos correspondientes en las siguientes rutas:

```
sudo cp /usr/jnetpcap/jnetpcap-1.4.r1425/libjnetpcap.so /usr/lib/  
sudo cp -R /usr/jnetpcap/jnetpcap-1.4.r1425/jnetpcap.jar /usr/share/java  
sudo cp -R /usr/jnetpcap/jnetpcap-1.4.r1425 /usr/share/doc
```

### LibPcap (con complementos) [8]

Dentro de la carpeta “Herramientas\_Instalacion\_Sniffer\libPcap” copiar el archivo “libpcap-1.10.1.tar.gz” a un directorio del sistema, por ejemplo, “Descargas”.

Situarse en “Descargas” y desempaquetar en la ruta “/usr/libpcap” con el siguiente comando:

**Nota:** Si la ruta no existe, crearla con el comando “mkdir”.

```
tar zxvf libpcap-1.10.1.tar.gz --directory /usr/libpcap
```

Dentro de la carpeta “Herramientas\_Instalacion\_Sniffer\libPcap” copiar el archivo “bison-3.8.2.tar.gz” a la carpeta “Descargas” y desempaquetar:

```
tar zxvf bison-3.8.2.tar.gz --directory /usr/libpcap
```

Dentro de la carpeta “Herramientas\_Instalacion\_Sniffer\libPcap” copiar el archivo “flex-2.6.4.tar.gz” a la carpeta “Descargas” y desempaquetar:

```
tar zxvf flex-2.6.4.tar.gz --directory /usr/libpcap
```

Dentro de la carpeta “Herramientas\_Instalacion\_Sniffer\libPcap” copiar el archivo “m4-1.4.19.tar.gz” a la carpeta “Descargas” y desempaquetar:

```
tar zxvf m4-1.4.19.tar.gz --directory /usr/libpcap
```

Situarse en el directorio “/usr/libpcap/m4-1.4.19” y ejecutar los tres comandos siguientes y en orden que se muestran:

**Nota:** Es necesario instalar primero el macro preprocesador m4 para instalar los otros después, sino, da error.

```
sudo ./configure  
sudo make  
sudo make install
```

Situarse en el directorio “/usr/libpcap/libpcap-1.10.1” y ejecutar los tres comandos siguientes y en orden que se muestran:

```
sudo ./configure  
sudo make  
sudo make install
```

Situarse en el directorio “/usr/libpcap/bison-3.8.2” y ejecutar los tres comandos siguientes y en orden que se muestran:

```
sudo ./configure  
sudo make  
sudo make install
```

Situarse en el directorio “/usr/libpcap/flex-2.6.4” y ejecutar los tres comandos siguientes y en orden que se muestran:

```
sudo ./configure  
sudo make  
sudo make install
```

Después de hacer todo esto es posible que al lanzar la aplicación, salte un error indicando que no encuentra alguna de las bibliotecas. Si esto ocurre, es necesario crear un enlace del fichero “libpcap.so.1.10.1” en el directorio “/usr/lib” con el nombre “libpcap.so”:

```
ln -s /usr/libpcap/libpcap-1.10.1/libpcap.so.1.10.1  
/usr/lib/libpcap.so
```

**Nota:** si se quita la opción -s, se crearía un acceso directo, que también serviría.

### Tarjeta de red en modo promiscuo

Se debe ejecutar el siguiente comando, para el puerto “eth0”:

```
ifconfig eth0 promisc
```

Si se desea realizar para otro puerto, solo es necesario modificarlo en el comando.

## E.4. Manual del usuario

Antes de iniciar la aplicación se puede cambiar el idioma en el que ésta se mostrará. Para ello, abrir el archivo “sniffer.property” que se encuentra en la ruta “proyecto eclipse\data\properties” y cambiar la línea 7:

Lang=ES

Si se desea que la aplicación se muestre en inglés, cambiar ES por EN:

Lang=EN

Si se desea que la aplicación se muestre en francés, cambiar ES por FR:

Lang=FR

Si se desea que la aplicación se muestre en alemán, cambiar ES por DE:

Lang=DE

Para arrancar la aplicación, si se utiliza Windows se deberá ejecutar el archivo “Sniffer-IV.bat” y si se utiliza Linux se deberá ejecutar el archivo “Sniffer-IV.sh”.

**Nota:** Para que funcione la aplicación sin problemas sobre Linux, se necesita lanzar la aplicación como superusuario “sudo Sniffer-IV.sh”. Si no, no deja hacer capturas en las interfaces de red.

En cualquier caso, se abrirá la siguiente pantalla:

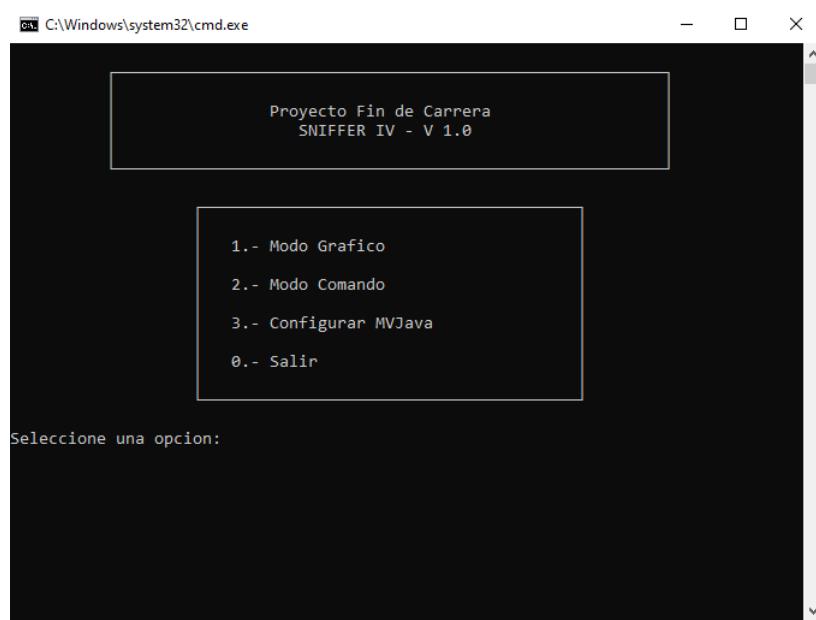


Figura E.11: Menú inicial del Sniffer.

Seleccionando el modo gráfico, escribiendo 1, se mostrará la pantalla principal de la aplicación:

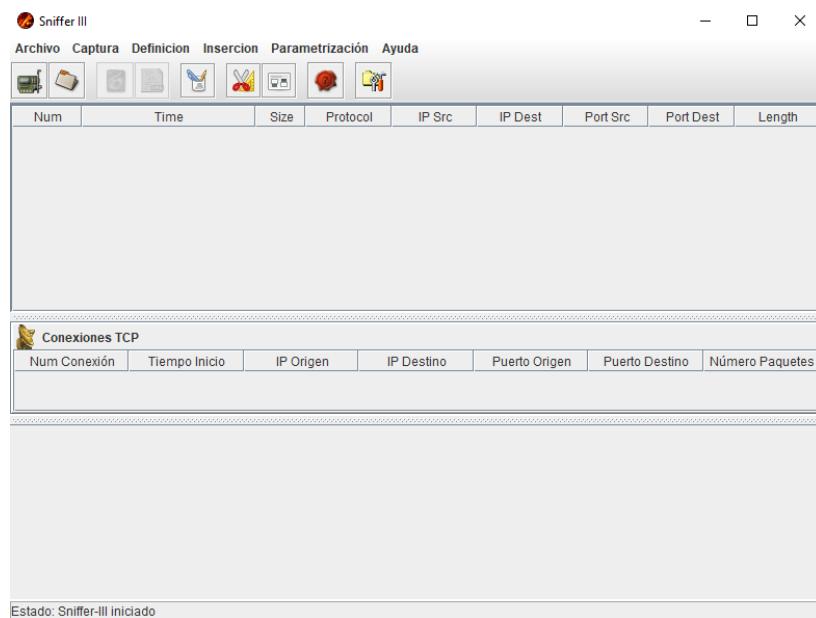


Figura E.12: Interfaz gráfica del Sniffer.

## 1. ARCHIVO -> ABRIR FICHERO DE CAPTURAS...



Figura E.13: Archivo -> Abrir fichero de capturas.

Se abre la siguiente ventana:

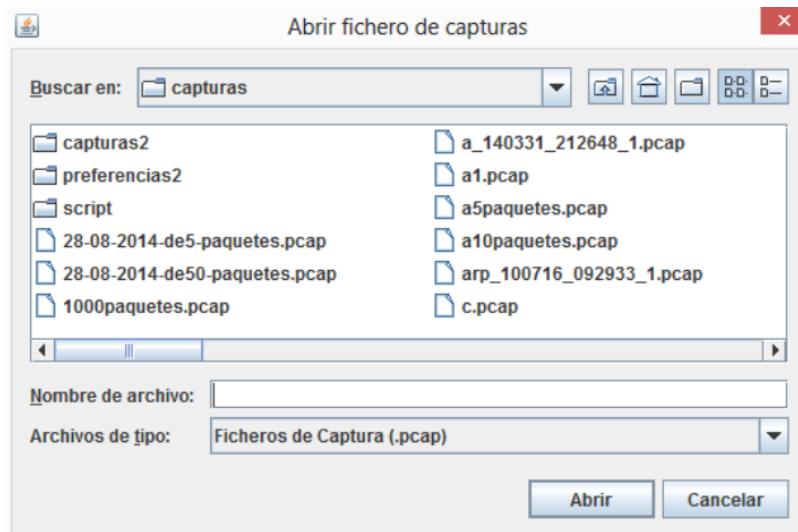


Figura E.14: Abrir fichero de capturas.

Desde esta se puede abrir una captura realizada previamente y se mostrará la información tal y como se ve a continuación:

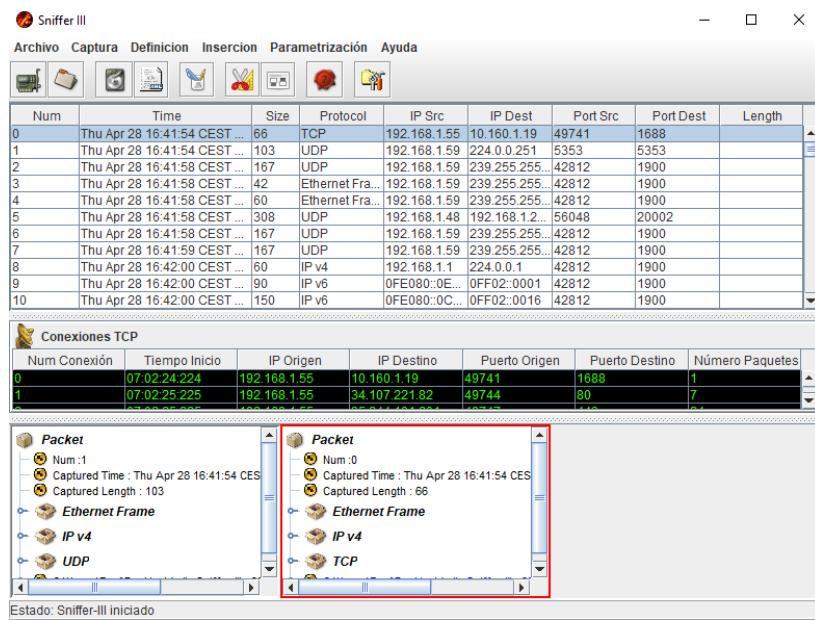


Figura E.15: Fichero de capturas abierto.

En la parte superior de la pantalla se muestra la información genérica de los paquetes capturados.

Si se desean reordenar los campos de la cabecera, bastará con pulsar con el botón izquierdo del ratón sobre el nombre de la columna que se desea mover y, manteniéndolo pulsado, arrastrarla al lugar donde queramos que se coloque.

Además, si se pulsa el botón derecho del ratón sobre la cabecera, se podrán escoger los campos que se desea que se muestren:

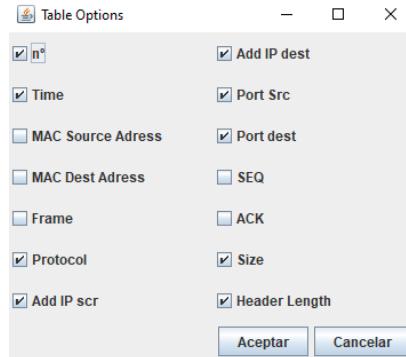


Figura E.16: Campos posibles en la tabla.

En la parte inferior de la pantalla se muestra el detalle de cada paquete. Para verlo, basta con seleccionar el paquete del que queremos ver la información detallada simplemente haciendo un click con el ratón en la fila correspondiente. Podemos desplegar también la información de cada protocolo, como podemos ver en la siguiente imagen:

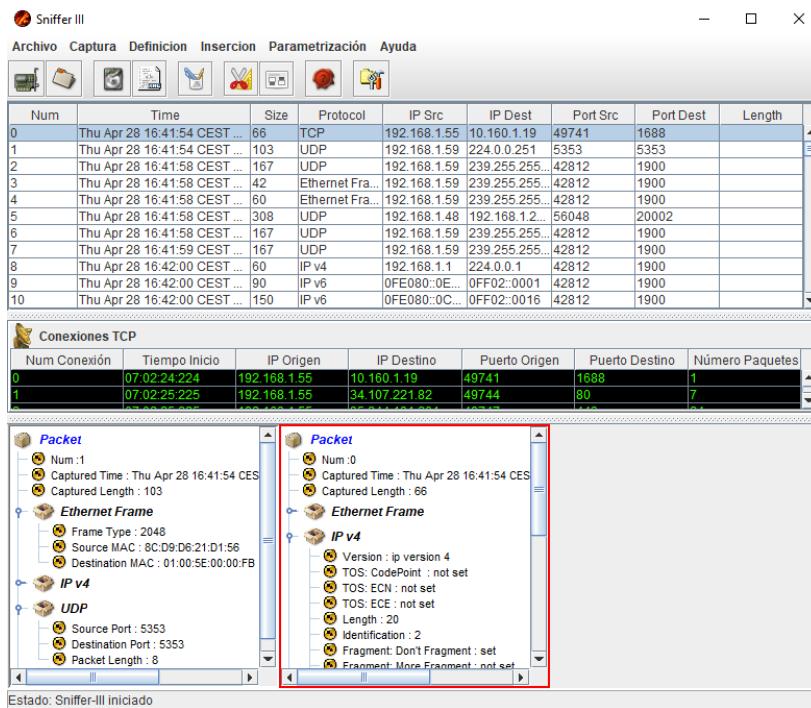


Figura E.17: Detallado de paquetes.

## 2. ARCHIVO -> GUARDAR FICHERO



Figura E.18: Archivo -> Guardar fichero.

Desde esta opción se guardará el fichero que esté abierto en la pantalla anterior.

## 3. ARCHIVO -> EXPORTAR -> CAPTURA A XML

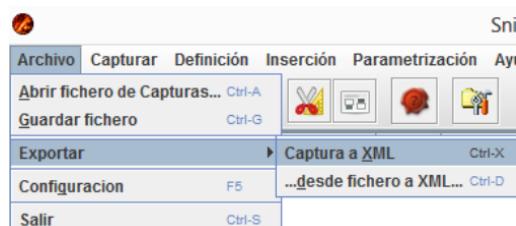


Figura E.19: Archivo -> Exportar -> Captura a XML.

Permite convertir el fichero de captura (extensión .pcap) que se tenga abierto en ese momento a un fichero XML (extensión .xml) y permite guardarlo en la ruta deseada:



Figura E.20: Archivo -> Exportar -> Captura a XML.

En caso de que ya exista, nos pregunta si se desea sobreescribir.

#### 4. ARCHIVO -> EXPORTAR -> DESDE FICHERO A XML

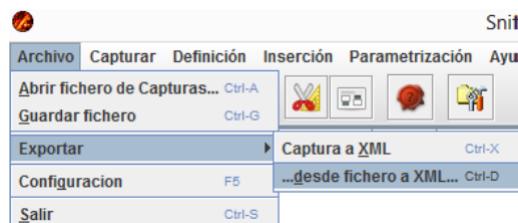


Figura E.21: Archivo -> Exportar -> desde fichero a XML.

En este caso, en vez de convertir el fichero actual, permite escoger cualquier fichero de capturas para exportarlo a un fichero XML.

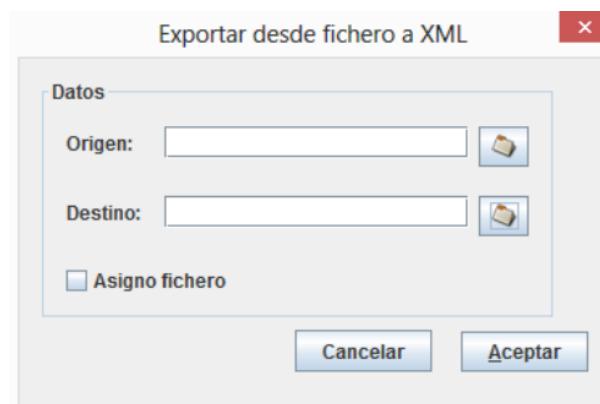


Figura E.22: Exportar desde fichero a XML.

## 5. ARCHIVO -> CONFIGURACIÓN



Figura E.23: Archivo -&gt; Configuración.

Desde aquí se pueden configurar los directorios donde se guardarán por defecto los ficheros de captura, los de exportaciones, los scripts, los ficheros de parametrización y el fichero de sincronismo (necesario para la comunicación con el analizador de XML).

**Nota:** Si la aplicación se va a lanzar junto con el analizador de XML, las rutas deben hacer referencia a la unidad de red compartida.

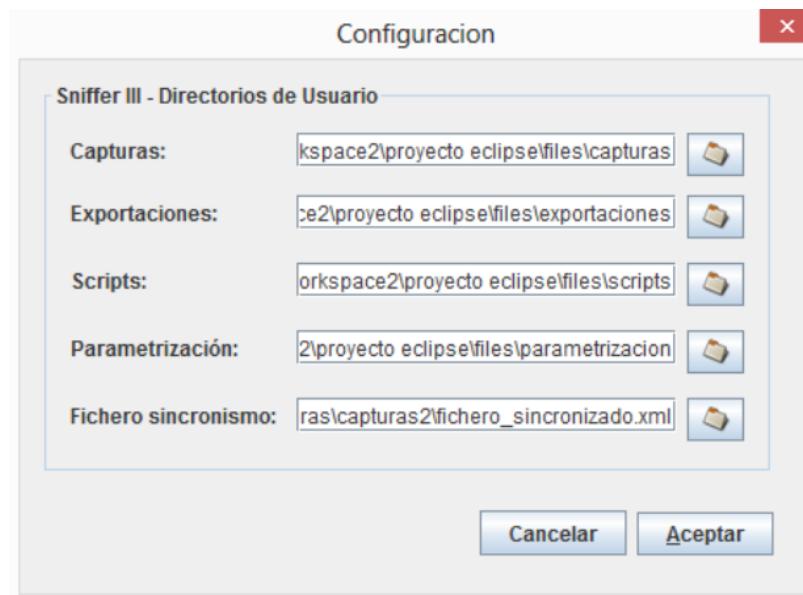


Figura E.24: Configuración.

## 6. CAPTURAR -> INICIO CAPTURA

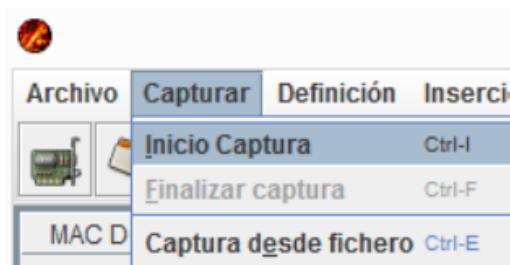


Figura E.25: Captura -&gt; Inicio captura.

Se muestra la siguiente pantalla:

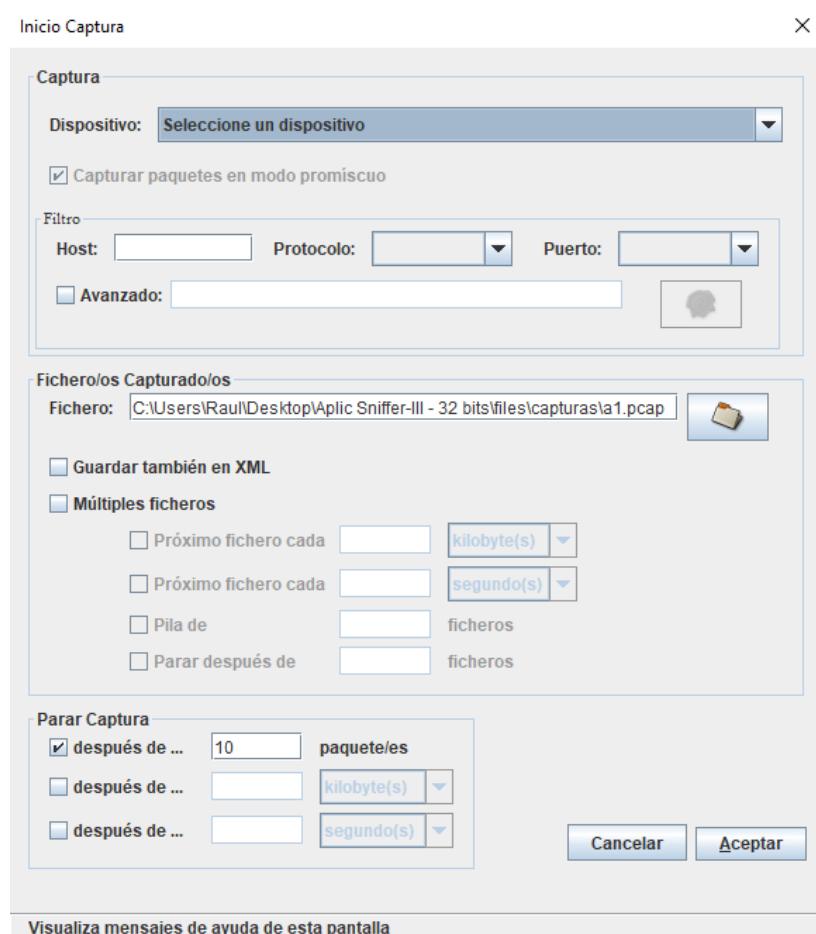


Figura E.26: Inicio captura.

## 6.1 Sección Capturar

La lista desplegable “Dispositivo” muestra los adaptadores de red disponibles y se podrá escoger el adaptador que se quiera utilizar para capturar los paquetes que circulen por la red.

## 6.2 Sección Filtro

Se pueden restringir el tráfico escogiendo las direcciones de origen y/o destino, un puerto determinado o realizar filtros avanzados, para cuyo caso existe una ayuda online.

### 6.3 Sección Ficheros Capturados

En la caja de texto “Fichero” se escogerá la ruta donde se quiere guardar el fichero.

El check “Guardar también en XML” definirá si se escribe un solo fichero con formato “.pcap” o si también se escribirá uno con formato XML.

Si se marca el check “Múltiples ficheros”, se podrá dividir la información en varios ficheros.

Esta opción es necesaria si se desea utilizar la aplicación junto con el Analizador de XML.

Si se marca el check “Pila de”, se activará el check “XML Circular”. Si no se marca este check, se generará un único XML en el cual estarán todos los paquetes capturados.

### 6.4 Sección Parar Captura

Desde esta sección se podrá realizar una parada automática de la captura según las condiciones indicadas.

Pulsando el botón “Aceptar” se abrirá la ventana desde la que se podrá observar la Estadística de paquetes que se están capturando. Se guardan las capturas en formato .pcap y en .xml:

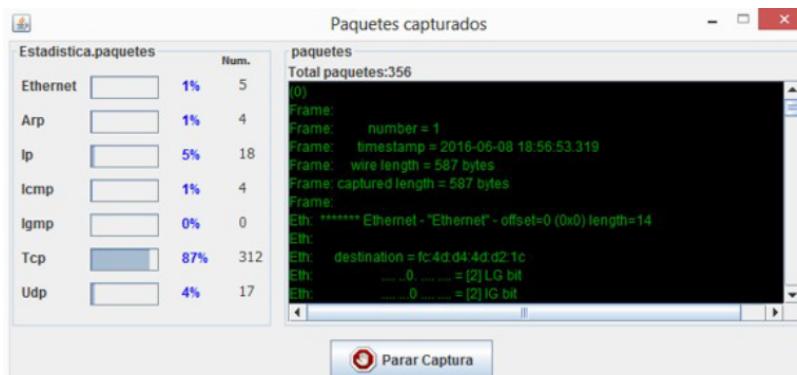


Figura E.27: Pantalla de captura.

## 7. CAPTURAR -> CAPTURA DESDE FICHERO

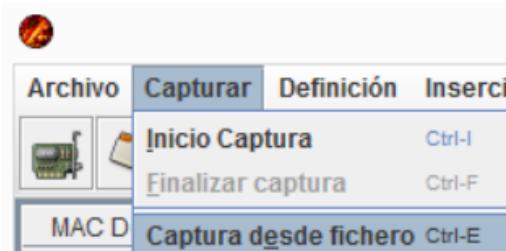


Figura E.28: Capturar -> Captura desde fichero.

Realiza la captura con la información de un fichero de capturas generado anteriormente:

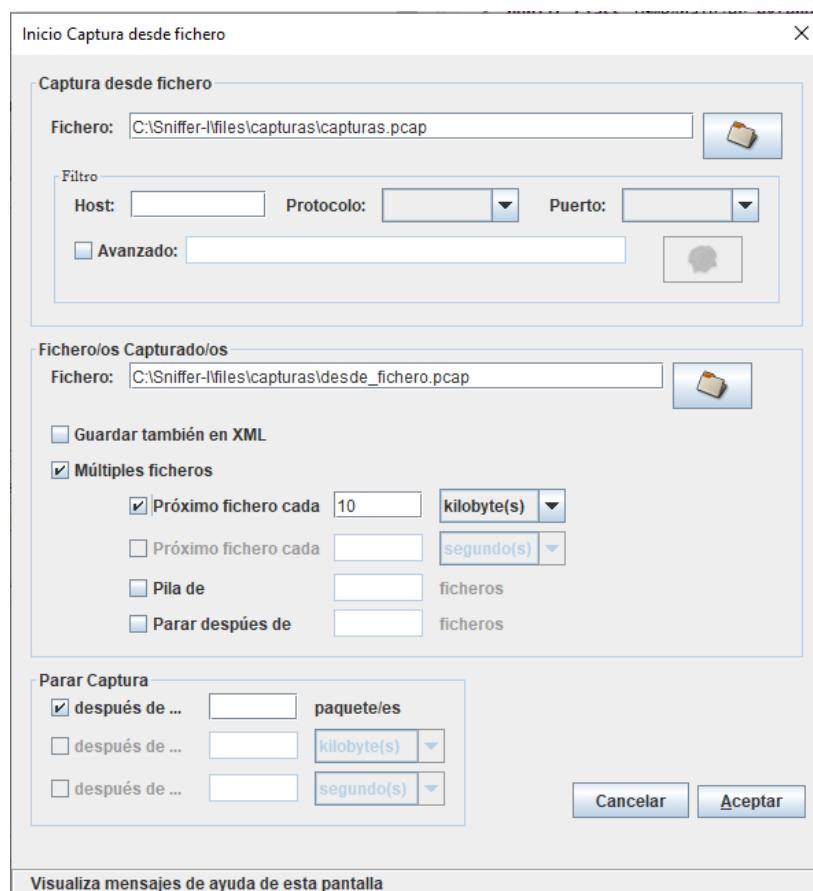


Figura E.29: Captura desde fichero.

## **8. DEFINICIÓN -> DEFINICIÓN DE PAQUETES**

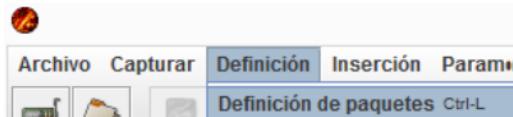


Figura E.30: Definición -> Definición de paquetes.

Se puede definir la estructura de un nuevo protocolo rellenando los campos del formulario siguiente:

Figura E.31: Definición de paquetes.

**Nota:** Si hay varios Campos Clave, éstos se deben separar con un guión.

El botón “Chequear” sirve para comprobar la correcta introducción de los campos.

También se puede abrir un protocolo definido anteriormente para modificarlo.

Para que los protocolos definidos puedan ser usado por la aplicación es necesario guardarlos en el directorio “(DirectorioSniffer)\files\definiciones”.

## 9. INSERCIÓN -> INSERCIÓN DE PAQUETES DEFINIDOS



Figura E.32: Inserción -> Inserción de paquetes definidos.

Esta opción sirve para determinar los niveles de encapsulación del protocolo.

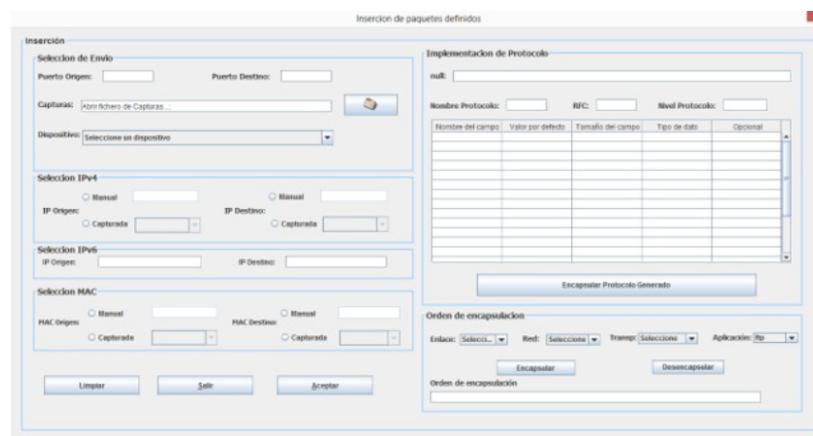


Figura E.33: Inserción de paquetes definidos.

**Nota:** Si se introduce un dato erróneo se producirá un fallo de envío.

### 9.1 Sección Selección de Envío

Se determinarán los puertos de origen y destino.

Permite seleccionar un fichero de capturas previamente guardado. Para poder elegir direcciones previamente capturadas.

También permite seleccionar el dispositivo a través del cual enviar el paquete definido.

### 9.2 Sección Selección IPv4

Permite seleccionar la IP de origen y de destino, introduciéndola manualmente o bien mediante los valores que contenga un fichero de capturas.

### 9.3 Sección Selección IPv6

Permite determinar la IP de origen y la de destino.

### 9.4 Sección Selección MAC

Permite seleccionar la MAC de origen y de destino, introduciéndola manualmente o bien mediante los valores que contenga un fichero de capturas.

### 9.5 Sección Orden de encapsulación

Permite determinar el orden de encapsulación del protocolo.

La lista desplegable “Aplicación” permite seleccionar un protocolo definido anteriormente.

### 9.6 Sección Implementación de Protocolo

Permite añadir los valores del paquete elegido en el nivel de Aplicación.

Encapsula el protocolo.

## 10. INSERCIÓN -> INSERCIÓN DE PAQUETES CAPTURADOS



Figura E.34: Inserción -> Inserción de paquetes capturados.

Muestra la siguiente pantalla, desde la que seleccionar el dispositivo desde el cual enviar los paquetes, seleccionar también un fichero de capturas y el número de reenvíos que soportará dicho fichero:

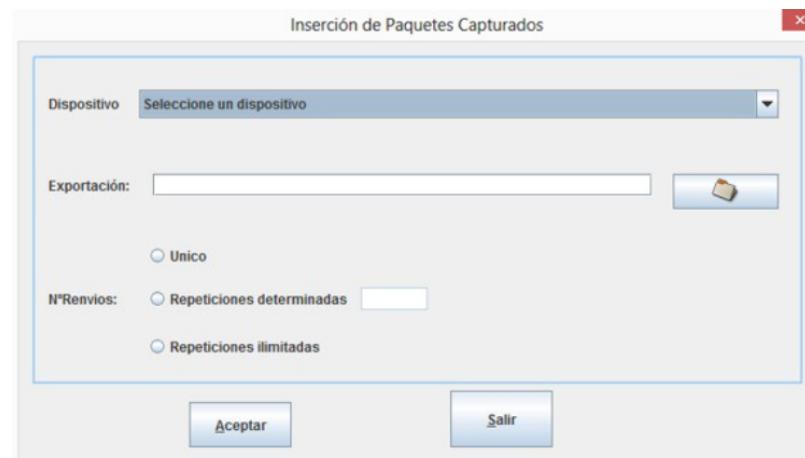


Figura E.35: Inserción de paquetes capturados.

## 11. PARAMETRIZACIÓN -> CAPTURAR

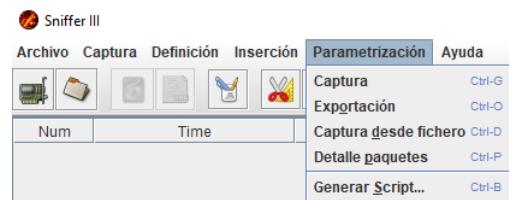


Figura E.36: Parametrización -&gt; Captura.

Se muestra la siguiente ventana, la cual permite generar un fichero para utilizarlo en posteriores capturas:

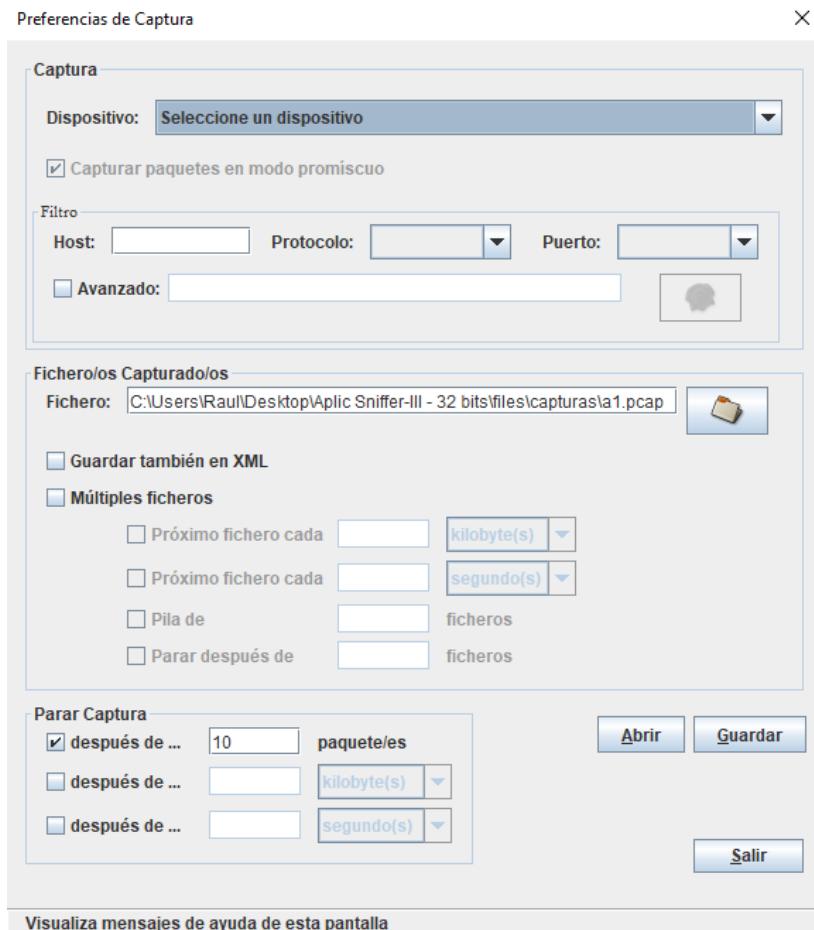


Figura E.37: Preferencias de captura.

Esta opción es más útil en el Modo Comando, aunque también se puede utilizar en el Modo Gráfico.

También permite abrir un fichero de capturas ya existente para modificar algún parámetro.

## 12. PARAMETRIZACIÓN -> EXPORTAR

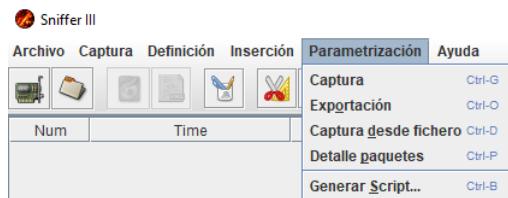


Figura E.38: Parametrización -> Exportar.

Se abre la siguiente ventana:



Figura E.39: Preferencias de exportación.

Con el botón correspondiente a “Origen” se podrá seleccionar un fichero con formato pCap o un fichero XML de Metadatos; éste último contiene las definiciones de varios ficheros de captura concatenados.

Con el botón correspondiente a “Destino” se podrá seleccionar un fichero XML.

Seleccionando el check “Asigno fichero” la exportación se almacenará en un único fichero.

## 13. PARAMETRIZACIÓN -> CAPTURA DESDE FICHERO

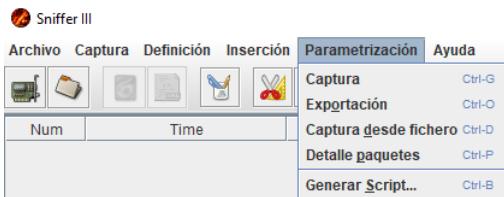


Figura E.40: Parametrización -> Captura desde fichero.

Esta opción es muy similar a la primera de este menú (Parametrización à Capturar); también genera un fichero de configuración para utilizar en posteriores capturas, solo que con esta opción, en vez de seleccionar un dispositivo de red, se utiliza un fichero de capturas que se haya guardado previamente.

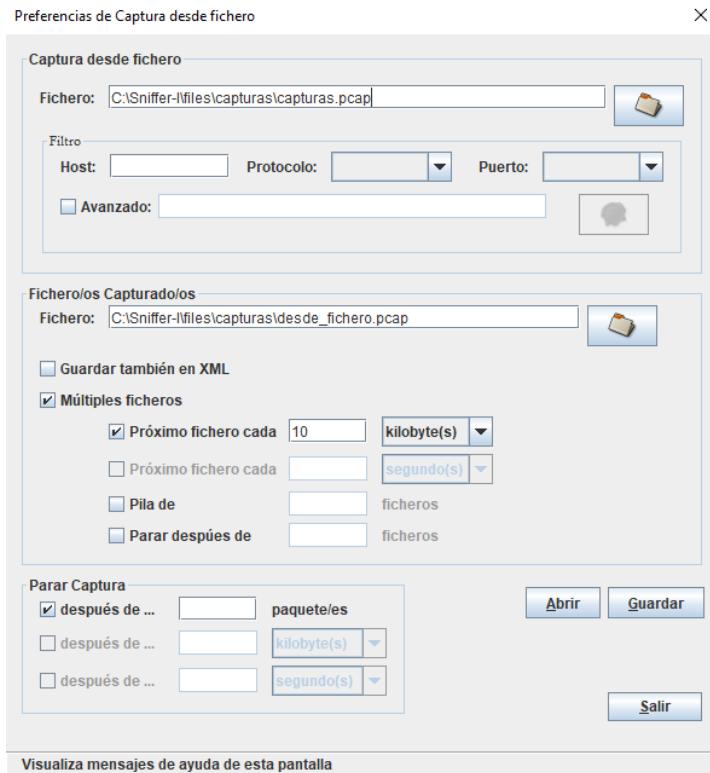


Figura E.41: Preferencias de captura desde fichero.

## 14. PARAMETRIZACIÓN -> DETALLE PAQUETES

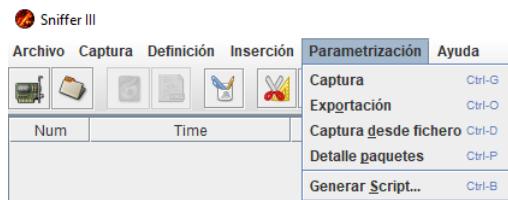


Figura E.42: Parametrización -> Detalle paquetes.

Esta opción permite guardar las preferencias del usuario relativas al panel de detalle de paquetes, como puede ser el diseño del panel (número de paquetes simultáneos), el número de bytes mostrados, o el formato de estos últimos (hexadecimal o binario).

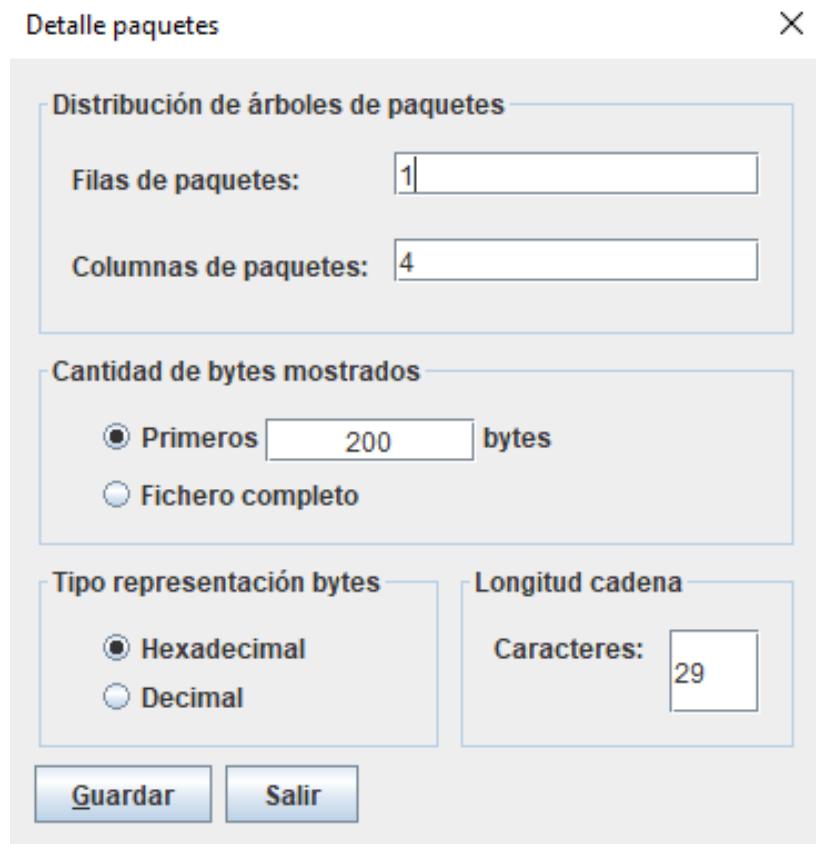


Figura E.43: Preferencias del detallado de paquetes.

## 15. PARAMETRIZACIÓN -> GENERAR SCRIPT

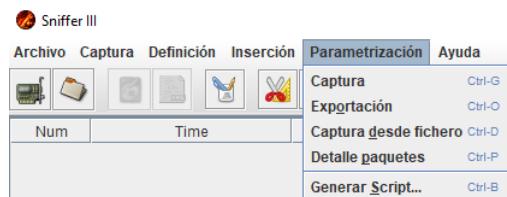


Figura E.44: Parametrización -> Generar script.

Esta opción permite generar scripts para poder ejecutar la aplicación en Modo Comando.

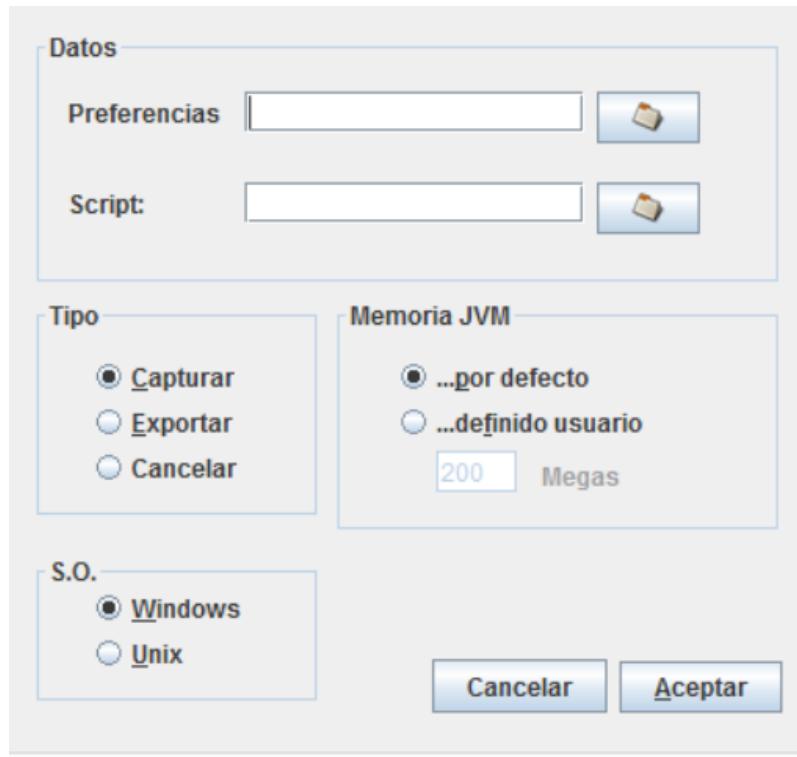


Figura E.45: Preferencias de generar script.

### 14.1 Sección Datos

Para seleccionar el fichero de preferencias de captura y el nombre del script que se desea generar.

## 14.2 Sección Tipo

El tipo del script puede ser de tres tipos.

## 14.3 Sección Memoria JVM

Cantidad de memoria que se quiere asignar a la máquina virtual de Java.

## 14.4 Sección S.O.

Se seleccionará el sistema operativo en el que se ejecutará la aplicación en Modo Comando.

# 16. AYUDA -> CONTENIDOS



Figura E.46: Ayuda -> Contenidos.

Muestra la ayuda de la aplicación en un navegador.

---

## Bibliografía

---

- [1] S. Deering A. Conta, Transwitch. Internet control message protocol (icmpv6) for the internet protocol version 6 (ipv6) specification. <https://datatracker.ietf.org/doc/html/rfc4443>.
- [2] S. Deering B. Cain, Cereva Networks. Internet group management protocol, version 3. <https://datatracker.ietf.org/doc/html/rfc3376>.
- [3] Patrick Charles. jpcap – a network packet capture library. <http://jpcap.sourceforge.net/>.
- [4] Nirav Thaker Ferenc Hechler, Simon Tuffs. Fat jar eclipse plug-in. <http://fjep.sourceforge.net/>.
- [5] Eclipse Foundation. Eclipse ide. <https://www.eclipse.org/>.
- [6] Online Gantt. Online gantt. <https://www.onlinegantt.com>.
- [7] David Gilbert. Jfreechart. <https://www.jfree.org/jfreechart/>.
- [8] The Tcpdump Group. Tcpdump & libpcap. <https://www.tcpdump.org>.
- [9] msaelices ianaya89. Absolutelayout. <https://nativescript-vue.org/es/docs/elements/layouts/absolute-layout/>.
- [10] Indeed. Salario promedio de un programador junior en españa. <https://es.indeed.com/career/programador-junior/salaries>.
- [11] Rolf Lear Jason Hunter. Jdom. <http://www.jdom.org/>.
- [12] Object Refinery Limited. Jcommon. <https://www.jfree.org/jcommon/>.

- [13] NFON. Request for comments (rfc). <https://www.nfon.com/es/get-started/cloud-telephony/lexicon/base-de-conocimiento-destacar/request-for-comments-rfc>.
- [14] Oracle. Java. <https://www.oracle.com/es/java/technologies/javase/javase8u211-later-archive-downloads.html>.
- [15] Oracle. Virtualbox. <https://www.virtualbox.org/>.
- [16] Oracle. A visual guide to layout managers. <https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html>.
- [17] ISI P. Mockapetris. Internet group management protocol, version 3. <https://datatracker.ietf.org/doc/html/rfc1035>.
- [18] The Apache Ant Project. Apache ant. <https://ant.apache.org/>.
- [19] Sly Technologies. jnetpcap - libpcap/winpcap java wrapper. <https://sourceforge.net/projects/jnetpcap/>.
- [20] Riverbed Technology. Winpcap - the industry-standard windows packet capture library. <https://www.winpcap.org/>.
- [21] Wireshark. Wireshark. <https://www.wireshark.org/>.