TRABAJOS FIN DE GRADO

Universidad de Jaén

*Escuela Politécnica Superior de Jaén*

# Sistema de detección de intrusiones basado en aprendizaje automático

Autor: Sergio Villar Serrano

Grado: Ingeniería en Informática

Director: Prof. D. Manuel José Lucena López
Departamento del director: Informática

Fecha: 31/7/2024


UNIVERSIDAD DE JAÉN

D./Dª Prof. D. Manuel José Lucena López, tutor(es) del Trabajo Fin de Grado titulado: **Sistema de detección de intrusiones basado en aprendizaje automático**, que presenta Sergio Villar Serrano, autoriza(n) su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, 31/7/2024

**Agradecimientos**

En esta sección del trabajo, me gustaría expresar mi más sincero agradecimiento a cada una de las personas que me han apoyado y acompañado en la realización de este trabajo del cual sacó una gran experiencia y formación en este ámbito que tanto me apasiona.

En primer lugar, me gustaría mostrar mis muestras de cariño y agradecimiento a mis padres, José y Rosa, sin ellos hubiera sido incapaz de culminar este viaje. A pesar de todas las adversidades, siempre me han estado ayudando y apoyándome en todo lo posible, a veces, centrándose más en el desarrollo de mi trabajo que en sus propios problemas. Muchas gracias de corazón.

Destacar también, a muchos de mis amigos, por sus ánimos infinitos, su gran apoyo e inspiración, han hecho que siga adelante con mis sueños y objetivos haciendo de esta experiencia algo inimaginable.

En segundo lugar, me gustaría acordarme de cada uno de mis seres queridos que no están, sé que desde allí arriba me habías dado muchísimas fuerzas para nunca rendirme y dar todo lo mejor de mí. Allá donde estéis, muchas gracias por todo.

En tercer lugar, no me podía olvidar de mis abuelos, un pilar fundamental en mi vida, gracias a ellos, soy la persona que soy, y también, me gustaría acordarme de mi tito Antonio, un familiar muy especial para mí que desde muy pequeño me ha hecho ser muy feliz.

Por último y no menos importante, darle las gracias a mi Tutor, Manuel José Lucena, por brindarme tantos consejos y ánimos que me han servido para la culminación de un gran trabajo del cual extraigo un gran aprendizaje.

Muchas gracias a todos.

# Tabla de contenidos

# Lista de figuras

# Lista de tablas

# Lista de algoritmos

# Lista de listados de código

# Capítulo 1

# INTRODUCCIÓN

Este capítulo describe cómo emplear la plantilla LaTeX para generar una memoria de TFG ajustada a las normas establecidas por la *Escuela Politécnica Superior de Jaén* (EPSJ). La plantilla está diseñada para su uso con `pdflatex`, el compilador de LaTeX más popular para generar documentos en formato `PDF`.

La estructura de esta plantilla se corresponde con la requerida por la EPSJ para los TFG de tipo teórico o experimental. Estos no son los más habituales en el *Grado en Ingeniería Informática*, cuyo tipo más común es el *Proyecto de Ingeniería*. Puedes usar esta plantilla para generar una memoria de ese tipo modificando los nombres y contenidos de los capítulos y secciones.

Este capítulo de introducción se usa en la plantilla para describir cómo escribir la memoria usando LaTeX , por lo que tendrás que sustituir este contenido por el corresponda a tu memoria. Para *compilar* los fuentes LaTeX y obtener el correspondiente `PDF` necesitarás una distribución local de LaTeX , como puede ser TeX Live (`en.wikipedia.org/wiki/TeX_Live`) o MiKTeX (`en.wikipedia.org/wiki/MiKTeX`), o bien trabajar con un editor LaTeX en la nube como puede ser Overleaf (`https://overleaf.com`).

El presente Proyecto de Fin de Grado aborda una de las problemáticas más acuciantes en el ámbito de la ciberseguridad: la detección proactiva de intrusiones en redes informáticas. En un panorama digital en constante evolución, donde las amenazas son cada vez más sofisticadas y persistentes, la capacidad de identificar y neutralizar actividades maliciosas en tiempo real se ha convertido en un pilar fundamental para la protección de la información y la infraestructura crítica. Tradicionalmente, los Sistemas de Detección de Intrusiones (IDS) se basaban en firmas o reglas predefinidas, un enfoque que, si bien eficaz contra amenazas conocidas, presenta limitaciones intrínsecas ante nuevos ataques o variaciones de los existentes.

Este trabajo propone el desarrollo de un Sistema de Detección de Intrusiones (IDS) innovador, que fusiona la captura y el monitoreo de tráfico de red en tiempo real con la potencia de las técnicas de aprendizaje automático (Machine Learning). El objetivo principal es la creación de un modelo de clasificación robusto y preciso, capaz de discernir de forma autónoma si un determinado flujo de tráfico de red es benigno (legítimo) o malicioso (indicativo de una intrusión o ataque). Esta aproximación busca superar las deficiencias de los sistemas basados en firmas, ofreciendo una mayor adaptabilidad y capacidad para detectar anomalías y comportamientos nunca antes vistos.

Para lograr este cometido, se hará uso de diversas tecnologías y metodologías de vanguardia. La fase de captura y preprocesamiento de datos de red se implementará mediante una aplicación diseñada específicamente para tal fin, garantizando la obtención de información relevante y en el formato adecuado para su análisis. Posteriormente, esta información será alimentada a un modelo de Machine Learning, concretamente un algoritmo de Random Forest (Bosques Aleatorios), seleccionado por su probada eficacia, robustez y capacidad para manejar grandes volúmenes de datos con alta dimensionalidad, características esenciales en el análisis de tráfico de red.

La estructura de este primer capítulo sentará las bases conceptuales y técnicas de todo el proyecto. A través de una descripción detallada de los componentes clave y la visión general del sistema propuesto, se facilitará la comprensión de los antecedentes del trabajo, los cuales serán explorados en profundidad en el segundo capítulo de esta memoria. Este enfoque metodológico permite establecer un marco claro para la posterior justificación, diseño e implementación de la solución planteada, enfatizando la relevancia y el impacto potencial de un IDS basado en aprendizaje automático en el contexto actual de la ciberseguridad.

## 1.1. Fundamentos

En este capítulo se establecerán las bases teóricas y conceptuales fundamentales que sustentan el análisis, diseño e implementación del sistema de detección de intrusiones propuesto. Se abordará en profundidad el paradigma de los Sistemas de Detección de Intrusiones, las bases del aprendizaje automático como disciplina, y se hará un énfasis particular en el comprensión del algoritmo de Random Forest, pieza clave de la solución desarrollada. Finalmente, se detallarán las métricas esenciales para la evaluación rigurosa del rendimiento de modelos de clasificación en el contexto de la ciberseguridad. Se comenzará explicando el concepto de **ciberseguridad y**

**amenazas en redes**.

Una memoria de TFG es similar a un libro y, por extensión, puede llegar a ser incluso más extensa. Por ello su contenido se estructura en varios niveles, facilitando así una numeración jerárquica que puedes ver reflejada en el índice que precede a esta introducción.

Los niveles de uso habitual para generar esa jerarquía son:

- 1. Capítulo
  - 1.1. Sección
    - 1.1.1. Subsección
      - 1.1.1.1. Subsubsección

No se recomienda tener más de estos cuatro niveles, aunque podrían usarse otros para dar mayor profundidad a la jerarquía.

### 1.1.1. Ciberseguridad y amenazas en redes

La seguridad de la información constituye un pilar fundamental en la sociedad digital contemporánea, donde la interconexión global y la dependencia de los sistemas informáticos son crecientes. La protección de los activos digitales se articula en torno a los principios de confidencialidad, integridad y disponibilidad (CID). La confidencialidad garantiza que la información sea accesible únicamente por entidades autorizadas; la integridad asegura que la información no ha sido alterada de forma no autorizada; y la disponibilidad se refiere a la capacidad de los usuarios autorizados para acceder a la información y los sistemas cuando sea necesario. Cualquier evento que comprometa uno o más de estos principios se clasifica como una amenaza, la cual, al explotar una vulnerabilidad existente, puede materializarse en un riesgo para la organización.

El panorama de amenazas en redes informáticas evoluciona constantemente, presentando un desafío dinámico para la ciberseguridad. Si bien históricamente los ataques se centraban en la explotación de vulnerabilidades conocidas o errores de configuración, la sofisticación actual se manifiesta en técnicas de evasión más complejas y la emergencia de amenazas persistentes avanzadas (APT). Entre los tipos de ataques más prevalentes que un sistema de detección de intrusiones debe ser capaz de identificar se encuentran:

- **Ataques de Denegación de Servicio (DoS/DDoS)**: Dirigidos a agotar los recursos de un sistema o red, impidiendo el acceso a servicios legítimos.
- **Escaneo de Puertos y Reconocimiento**: Fases preliminares donde un atacante explora una red en busca de puntos débiles o servicios abiertos.
- **Ataques de Fuerza Bruta**: Intentos sistemáticos y repetitivos para adivinar credenciales de acceso o claves de cifrado.
- **Malware (Software Malicioso)**: Incluye virus, troyanos, ransomware y spyware, diseñados con propósitos destructivos, de espionaje o de control remoto.
- **Explotación de Vulnerabilidades**: Aprovechamiento de fallos de diseño o implementación en software y hardware para obtener acceso no autorizado o ejecutar código malicioso. La capacidad de identificar estos y otros comportamientos anómalos es crucial para mantener la postura de seguridad de una infraestructura de red.

La memoria se generará a partir de un documento principal al que llamaremos habitualmente `main.tex`. Este contendrá los datos necesarios para generar la portada y páginas iniciales de la memoria: nombre del estudiante, nombres de los tutores, título del TFG, fecha en que se deposita, etc.

En el documento principal existirán tantos comandos `\input{archivo.tex}` como capítulos existan, a fin de importar su contenido y generar así el `PDF` completo.

### 1.1.2. Sistema de detección de instrusiones

La memoria se estructura en varios capítulos, en principio esta plantilla tiene tantos como apartados indica la normativa de la EPSJ, si bien existe cierta flexibilidad a la hora de agregar otros que se consideren necesarios.

El fuente LaTeX para cada capítulo se alojará en un archivo independiente, con extensión `.tex`, que será incluido en el documento principal con el comando `\input`. La primera línea del capítulo será:

```
\chapter{TÍTULO DEL CAPÍTULO}
```

Observa que el título se escribe completamente en mayúsculas. La numeración de los capítulos es automática, por lo que no debes preocuparte por este detalle.

En respuesta a la creciente complejidad del panorama de amenazas, los Sistemas de Detección de Intrusiones (IDS) han emergido como componentes esenciales de una estrategia de defensa en profundidad. Un IDS puede definirse como una aplicación de seguridad que monitoriza el tráfico de red o la actividad de un sistema con el fin de identificar patrones o comportamientos indicativos de una intrusión o una violación de políticas de seguridad. Su función principal no es bloquear el ataque (labor de un IPS o firewall), sino generar alertas que permitan a los administradores de seguridad tomar las medidas correctivas oportunas.

La clasificación de los IDS se realiza habitualmente atendiendo a su método de detección y a su ubicación en la infraestructura:

- **IDS Basados en Firmas (Signature-based IDS - SIDS)**: Estos sistemas operan mediante la comparación del tráfico de red o eventos del sistema con una base de datos de firmas predefinidas, las cuales corresponden a patrones conocidos de ataques. Su principal ventaja reside en una alta precisión en la detección de ataques ya identificados y catalogados, con una baja tasa de falsos positivos en esos escenarios. No obstante, su limitación inherente radica en la incapacidad para detectar ataques novedosos o variantes polimórficas para las cuales no existe una firma en su base de datos, lo que exige una constante actualización de las mismas.
- **IDS Basados en Anomalías (Anomaly-based IDS - AVIDS)**: A diferencia de los SIDS, los AVIDS no dependen de firmas de ataques conocidos. En su lugar, construyen un perfil de comportamiento "normal"de la red, los usuarios o las aplicaciones. Cualquier desviación significativa de este perfil es considerada una anomalía y, por ende, una posible intrusión. La principal fortaleza de los AVIDS es su capacidad para detectar ataques "zero-day"(desconocidos previamente) y ataques sutiles que no encajan en patrones preestablecidos. Sin embargo, su principal desafío es la mayor propensión a generar falsos positivos, ya que cualquier cambio en el comportamiento normal (como la introducción de un nuevo servicio o una carga de tráfico inusual pero legítima) puede ser erróneamente clasificado como un ataque, requiriendo un ajuste continuo y afinado.

En cuanto a su ubicación, se distinguen los Network-based IDS (NIDS), que analizan el tráfico de red en puntos estratégicos de la infraestructura sin depender de los hosts individuales, y los Host-based IDS (HIDS), que monitorizan la actividad interna de un sistema operativo o aplicación específica. El sistema desarrollado en este proyecto se enmarca dentro de la categoría de NIDS, enfocándose en el análisis del tráfico de red.

### 1.1.3. Clasificación

Un capítulo se dividirá en secciones, cada una de las cuales se inicia con el siguiente comando:

```
\section{Título de la sección}
```

Es recomendable planificar las secciones en que se dividirá un capítulo antes de comenzar a escribirlo. Debemos pensar qué queremos contar en el capítulo y cómo estructurar ese contenido en partes cohesionadas, cada una de las cuales daría lugar a una sección.

La clasificación es una de las principales tareas dentro de la minería de datos. Como se expresó previamente, el objetivo de esta tarea es predecir una etiqueta categórica para cada ejemplo de un conjunto de datos a partir de sus atributos conocidos [1].

Existen diversos tipos de clasificación. Entre los más destacados, podemos encontrar:

- **Clasificación binaria**: en este problema concreto de clasificación, es necesario determinar a qué clase pertenece cada ejemplo de los datos, pero únicamente hay dos clases a elegir. Se suelen usar para problemas de carácter binario, como si una persona está sana o enferma o, en el contexto de esta memoria, si una conexión determinada es benigna o un ataque.

- **Clasificación multiclase**: al contrario que en el caso previo, se debe determinar la clase de cada ejemplo de entre un conjunto de clases de tamaño superior a dos. En este TFG, también se hará uso de la clasificación multiclase para tratar de determinar el tipo de ataque de una conexión que se sabe maliciosa.

- **Clasificación multietiqueta**: un caso particular dentro de la clasificación multiclase es la clasificación multietiqueta, ya que, en lugar de asignar una sola clase a cada ejemplo del conjunto de datos, se le deben asignar un conjunto de clases en función a las características de los datos, a menudo siendo posible que cada elemento tenga un número de clases asociadas distinto.

### 1.1.4. Descubrimiento de conocimiento en bases de datos

Las secciones de cierta extensión se dividirán en subsecciones. De igual forma, una subsección con mucho contenido se dividiría en subsecciones. Los comandos para generar estos niveles son:

```
\subsection{Título de la subsección}

\subsubsection{Título de la subsubsección}
```

Aunque obviamente es algo subjetivo, en general no es deseable tener varias páginas seguidas de texto sin ninguna división que permita tener un punto de referencia.

El Descubrimiento de Conocimiento en Bases de Datos (KDD) es un proceso sistemático y iterativo, no trivial, diseñado para la extracción de patrones válidos, novedosos, potencialmente útiles y comprensibles a partir de grandes volúmenes de datos. Representa la filosofía subyacente a la transformación de datos crudos en inteligencia accionable, siendo crucial en dominios donde la toma de decisiones basada en evidencia es primordial, como la ciberseguridad. En el contexto de un Sistema de Detección de Intrusiones (IDS) basado en aprendizaje automático, el KDD proporciona el marco metodológico para derivar el çonocimiento"que permite al sistema identificar comportamientos anómalos o maliciosos en el tráfico de red.

El proceso de KDD se articula a través de una secuencia de etapas interdependientes:

**Fases del Proceso KDD**

1. **Selección**: Esta etapa inicial define y adquiere los datos relevantes del dominio de aplicación. Para un IDS, esto implica la selección de datasets de tráfico de red, como el CIC-IDS2018, que contengan una representación adecuada de comportamientos tanto benignos como maliciosos. La exhaustividad y representatividad de esta selección son fundamentales para la generalizabilidad del modelo resultante.

2. **Preprocesamiento**: Considerada a menudo la fase más laboriosa y crítica, el preprocesamiento aborda la preparación de los datos brutos. Los datos de red, por su naturaleza, suelen ser ruidosos, incompletos, o inconsistentes. Esta etapa involucra:

- **Limpieza de Datos**: Eliminación o corrección de datos erróneos, duplicados, inconsistencias y manejo de valores ausentes.
- **Normalización/Estandarización**: Ajuste de las escalas de las características numéricas para evitar que aquellas con rangos de valores más amplios dominen en el análisis.
- **Transformación**: Conversión de datos a formatos adecuados para la minería. En el caso de tráfico de red, esto incluye la agregación de paquetes individuales en flujos de red y la extracción de características de esos flujos (ej., duración, número de paquetes, banderas TCP), elementos esenciales para el aprendizaje automático.

3. **Transformación**: Aunque intrínsecamente ligada al preprocesamiento, esta fase se centra en refinar la representación de los datos para la minería. La ingeniería de características es su componente clave, donde el conocimiento experto del dominio de red se aplica para derivar atributos más complejos y discriminatorios (ej., ratios de bytes por paquete, entropía de los puertos destino). El objetivo es crear un conjunto de características que optimicen la capacidad del algoritmo de Machine Learning para identificar patrones de intrusión.

4. **Minería de Datos (Data Mining)**: Este es el corazón del proceso KDD, donde se aplican algoritmos computacionales para descubrir patrones ocultos, asociaciones, cambios significativos, desviaciones o estructuras significativas en los datos. No es simplemente una técnica, sino un conjunto de enfoques, tareas y técnicas aplicadas a los datos ya preprocesados:

   - **Enfoques de Minería de Datos**:
     - **Aprendizaje Supervisado**: Como se detalla en el punto de Transformación, implica el uso de datos etiquetados para predecir un resultado. La detección de intrusiones es, en esencia, un problema de clasificación supervisada, donde se predice si un flujo es benigno o malicioso.
     - **Aprendizaje No Supervisado**: Se utiliza para encontrar estructuras o patrones en datos sin etiquetas previas (ej., clustering para agrupar tráficos similares, o detección de anomalías para identificar comportamientos que se desvían de lo normal sin una etiqueta de .ªtaque.explícita). Aunque tu proyecto se centra en lo supervisado, estos enfoques complementarios son relevantes en KDD.
     - **Aprendizaje Semi-supervisado**: Combina datos etiquetados y no etiquetados, útil en escenarios donde el etiquetado es costoso.
   - **Tareas Típicas de Minería de Datos**:

- **Clasificación**: Asignar elementos a categorías predefinidas (ej., .ªtaque.º "normal"). Esta es la tarea principal de tu IDS.
- **Regresión**: Predecir un valor numérico continuo.
- **Agrupación (Clustering)**: Dividir un conjunto de datos en grupos (clusters) de elementos similares.
- **Asociación**: Descubrir reglas que describen relaciones entre elementos.
- **Detección de Anomalías/Outliers**: Identificar patrones que no se ajustan a un comportamiento esperado.

- **Técnicas Aplicadas en este Proyecto**: En este trabajo, la fase de minería de datos se concreta en la aplicación del algoritmo Random Forest para la tarea de clasificación. Su capacidad para manejar un gran número de características y su robustez ante el ruido lo hacen idóneo para los complejos datasets de tráfico de red.

5. **Evaluación y Presentación**: La etapa final valida la significancia y utilidad de los patrones descubiertos. Los modelos de clasificación se evalúan utilizando métricas específicas (precisión, sensibilidad, F1-score, curva ROC) en un conjunto de datos de prueba independiente, asegurando que el conocimiento extraído sea generalizable y no un artefacto del entrenamiento. La interpretabilidad del modelo, aunque un desafío, es un objetivo deseable que permite a los analistas de seguridad comprender la razón de una detección. Finalmente, la presentación del conocimiento se realiza a través de informes, visualizaciones (como el dashboard de tu aplicación), que comunican de manera efectiva los insights a los usuarios finales.

### 1.1.5. Machine Learning

Tras los comandos `\chapter`, `\section`, `\subsection` y `\subsection` dispondremos los párrafos de texto con las descripciones y explicaciones que procedan. Es recomendable tener al menos un párrafo de texto entre dos comandos sucesivos de los anteriores, evitando así tener el inicio de una sección justo tras el inicio del capítulo, de una subsección inmediatamente tras la sección, etc.

Los párrafos de texto en LaTeX se separan unos de otros con una línea en blanco. El contenido de un párrafo se escribe todo seguido, sin pulsar nunca **Intro**/**Enter**. Al compilar se ajustará el texto automáticamente, introduciendo guiones donde sea necesario.

El aprendizaje automático (Machine Learning - ML), una rama de la inteligencia artificial, ha transformado diversas disciplinas al dotar a los sistemas de la capacidad de aprender de datos y mejorar su rendimiento con la experiencia, sin ser programados explícitamente para cada tarea. En el contexto de la ciberseguridad, y específicamente en la detección de intrusiones, el ML ofrece un enfoque adaptativo que puede identificar patrones complejos en grandes volúmenes de tráfico de red, superando las limitaciones de los métodos tradicionales basados en reglas fijas.

El ciclo de vida de un modelo de Machine Learning generalmente comprende las siguientes fases interdependientes:

- **Recopilación y Preprocesamiento de Datos**: Etapa crucial que implica la obtención, limpieza, transformación y estandarización de los datos. La calidad de los datos de entrada es determinante para el éxito del modelo.
- **Ingeniería de Características (Feature Engineering)**: Proceso de seleccionar, crear o transformar variables (features) a partir de los datos brutos que mejor representen la información subyacente y sean más relevantes para el problema de predicción.
- **Selección y Entrenamiento del Modelo**: Consiste en elegir el algoritmo de ML más adecuado y ajustar sus parámetros utilizando un subconjunto de datos (conjunto de entrenamiento) para que aprenda los patrones deseados.
- **Evaluación y Optimización del Modelo**: Se mide el rendimiento del modelo con un conjunto de datos no visto (conjunto de validación o prueba) y se realizan ajustes de hiperparámetros para mejorar su eficacia.
- **Despliegue y Monitorización**: Una vez validado, el modelo se integra en un entorno de producción para realizar predicciones en tiempo real, siendo fundamental un monitoreo continuo de su desempeño.

Dentro del ML, el aprendizaje supervisado es el paradigma central para la detección de intrusiones basada en clasificación. En este enfoque, el modelo aprende de un conjunto de datos que incluye tanto las características de entrada (ej., métricas del tráfico de red) como sus correspondientes etiquetas de salida (ej., "tráfico normal.º .ªtaque"). El objetivo es que el modelo infiera una función que mapee las entradas a las salidas, permitiendo clasificar nuevas instancias de datos sin etiquetas. Esta investigación se inscribe en este paradigma, buscando clasificar el tráfico de red como benigno o malicioso.

**Preprocesamiento de datos para modelos de Machine Learning**

La efectividad de cualquier modelo de aprendizaje automático depende críticamente de la calidad y el formato de los datos de entrada. En el contexto de la detección de intrusiones, los datos brutos de tráfico de red, capturados en formato de paquetes, no son directamente utilizables por los algoritmos de Machine Learning. El preprocesamiento de datos es, por tanto, una fase indispensable que transforma estos paquetes en un formato estructurado y significativo para el análisis.

Las etapas fundamentales del preprocesamiento en un IDS basado en ML incluyen:

- **Extracción y Reconstrucción de Flujos de Red**: Los algoritmos de ML operan sobre ïnstancias.º "muestras", que en el contexto de la red, son típicamente flujos. Un flujo se define como una secuencia de paquetes que comparten una tupla común de cinco elementos: dirección IP de origen, puerto de origen, dirección IP de destino, puerto de destino y protocolo de transporte. La reconstrucción de flujos a partir de paquetes individuales permite contextualizar la información y calcular características que abarcan la duración total de la conexión.
- **Generación de Características (Feature Engineering)**: Una vez reconstruidos los flujos, se extraen o derivan diversas características numéricas y categóricas que describen el comportamiento del flujo. Estas características deben ser consistentes con las utilizadas durante la fase de entrenamiento del modelo. Ejemplos de características incluyen:
  - **Métricas de Volumen**: Número total de paquetes, bytes transmitidos (bidireccional), duración del flujo.
  - **Métricas de Tasa**: Paquetes por segundo, bytes por segundo.
  - **Características del Protocolo**: Banderas TCP (SYN, ACK, FIN, RST, PSH, URG), tipo de protocolo (TCP, UDP, ICMP).
  - **Información de Puertos**: Puertos de origen y destino.
  - **Características Temporales**: Variabilidad en el tiempo entre paquetes (jitter).
  - **Entropía de la Carga Útil**: Medida de la aleatoriedad en los datos de la carga útil, que puede indicar cifrado o ciertos tipos de ataques.
- **Manejo de Variables Categóricas**: Las características no numéricas, como los nombres de protocolos (TCP, UDP, ICMP), deben transformarse a un formato numérico. La técnica común de One-Hot Encoding crea nuevas columnas binarias

para cada categoría, donde un '1' indica la presencia de esa categoría y un '0' su ausencia.

- **Escalado de Características Numéricas**: Las características numéricas a menudo presentan rangos de valores muy diferentes (ej., la duración de un flujo puede ser de milisegundos a horas, mientras que el número de paquetes es un entero pequeño). El escalado (normalización o estandarización) es crucial para evitar que las características con rangos más amplios dominen desproporcionadamente en el proceso de aprendizaje del modelo. Técnicas como StandardScaler (resta la media y divide por la desviación estándar) o MinMaxScaler (escala los valores a un rango específico, como [0, 1]) son comúnmente empleadas.
- **Manejo de Valores Ausentes o Ruidosos**: Los datos de red pueden contener valores faltantes o erróneos. Es fundamental aplicar estrategias de imputación (ej., rellenar con la media, mediana o moda) o eliminación de instancias para asegurar la integridad del dataset.
- **Manejo del Desequilibrio de Clases (Class Imbalance)**: En la detección de intrusiones, el tráfico legítimo es abrumadoramente más frecuente que el tráfico malicioso (clases desequilibradas). Si no se aborda, el modelo puede sesgarse hacia la clase mayoritaria y tener un rendimiento deficiente en la detección de la clase minoritaria (ataques). Técnicas como el oversampling (ej., SMOTE para crear instancias sintéticas de la clase minoritaria), undersampling (reducir la cantidad de instancias de la clase mayoritaria), o el uso de pesos de clase durante el entrenamiento del modelo, son esenciales para mitigar este problema.

### Evaluación de modelos en Sistemas de Detección de Intrusiones

La evaluación rigurosa del rendimiento de un modelo de clasificación es un paso crítico en el desarrollo de un IDS basado en aprendizaje automático. Dadas las implicaciones de seguridad, es imperativo no solo medir la precisión global, sino también comprender cómo el modelo maneja los errores, especialmente los falsos negativos, que representan ataques no detectados.

La matriz de confusión es la herramienta fundamental para una evaluación detallada. Esta tabla resume las predicciones del modelo frente a las etiquetas reales del dataset y se compone de cuatro cuadrantes clave en el contexto de clasificación binaria (Normal/Ataque):

- **Verdaderos Positivos (TP)**: Instancias de tráfico malicioso correctamente clasi-

ficadas como ataques.

- **Verdaderos Negativos (TN)**: Instancias de tráfico normal correctamente clasificadas como benignas.
- **Falsos Positivos (FP)**: Instancias de tráfico normal erróneamente clasificadas como ataques (falsa alarma).
- **Falsos Negativos (FN)**: Instancias de tráfico malicioso erróneamente clasificadas como normales (ataque no detectado).

A partir de la matriz de confusión, se derivan métricas esenciales para evaluar el desempeño del IDS:

- **Precisión (Accuracy)**:

```
\begin{equation}
    \left(\frac{TP + TN}{TP + TN + FP + FN}\right )
    \label{Eq.Prob1}
\end{equation}
```

Listado 1.1: Definición de una ecuación para la Precisión

$$\left(\frac{TP+TN}{TP+TN+FP+FN}\right) \tag{1.1}$$

Mide la proporción de predicciones correctas sobre el total de predicciones. Aunque intuitiva, puede ser engañosa en datasets desequilibrados, donde una alta precisión podría deberse simplemente a la correcta clasificación de la clase mayoritaria.

- **Sensibilidad / Exhaustividad (Recall / True Positive Rate - TPR)**:

```
\begin{equation}
    \left(\frac{TP}{TP + TN}\right )
    \label{Eq.Prob2}
\end{equation}
```

Listado 1.2: Definición de una ecuación para la Sensibilidad

$$\left(\frac{TP}{TP+TN}\right) \tag{1.2}$$

Cuantifica la capacidad del modelo para identificar correctamente todas las instancias positivas (ataques reales). En un IDS, maximizar el recall es a menudo

una prioridad, ya que un alto número de falsos negativos implica ataques que pasan desapercibidos.

- **Especificidad (True Negative Rate - TNR)**:

```
\begin{equation}
    \left(\frac{TN}{TN + FP}\right )
    \label{Eq.Prob3}
\end{equation}
```

Listado 1.3: Definición de una ecuación para la Especificidad

$$\left(\frac{TN}{TN + FP}\right) \tag{1.3}$$

Indica la proporción de instancias negativas (tráfico normal) que son correctamente identificadas.

- **Precisión (Precision / Positive Predictive Value - PPV)**:

```
\begin{equation}
    \left(\frac{TP}{TP + FP}\right )
    \label{Eq.Prob4}
\end{equation}
```

Listado 1.4: Definición de una ecuación para otro tipo de precisión

$$\left(\frac{TP}{TP + FP}\right) \tag{1.4}$$

Responde a la pregunta: de todas las instancias que el modelo clasificó como ataques, ¿cuántas fueron realmente ataques? Una alta precisión minimiza los falsos positivos, lo cual es crucial para evitar la "fatiga de alerta.[en] los operadores de seguridad.

- **F1-Score**:

```
\begin{equation}
    \left(2*\frac{Precisión*Recall}{Precisión + Recall}\right )
    \label{Eq.Prob4}
\end{equation}
```

Listado 1.5: Definición de una ecuación para la f1-score

$$\left(2 * \frac{Precisin * Recall}{Precisin + Recall}\right) \tag{1.5}$$

Es la media armónica de la precisión y la sensibilidad. Proporciona una métrica equilibrada que es especialmente útil cuando existe un desequilibrio significativo

entre las clases, ya que penaliza los modelos con altos falsos positivos y falsos negativos.

- **Tasa de Falsos Positivos (False Positive Rate - FPR)**:

```
\begin{equation}
    \left(\frac{FP}{FP + TN}\right )
    \label{Eq.Prob5}
\end{equation}
```

Listado 1.6: Definición de una ecuación para la tasa de falsos positivos

$$\left(\frac{FP}{FP+TN}\right) \tag{1.6}$$

Complementa la especificidad y es una métrica crítica en IDS, ya que una alta tasa de falsas alarmas puede llevar a que los analistas ignoren alertas legítimas.

La Curva ROC (Receiver Operating Characteristic) y el Área bajo la Curva (AUC) son herramientas gráficas y métricas complementarias. La curva ROC representa la relación entre la Tasa de Verdaderos Positivos (TPR) y la Tasa de Falsos Positivos (FPR) a distintos umbrales de clasificación, permitiendo visualizar el compromiso entre ambas. El AUC cuantifica el rendimiento general del clasificador en todos los umbrales posibles; un valor de AUC cercano a 1.0 indica un excelente poder discriminatorio.

Para garantizar la robustez y generalizabilidad del modelo, se emplean técnicas de validación. La Validación Cruzada K-Fold es un método estándar en el que el conjunto de datos se divide en k subconjuntos (folds). El modelo se entrena k veces; en cada iteración, se utiliza un fold diferente como conjunto de prueba y los k1 restantes como conjunto de entrenamiento. Los resultados se promedian para obtener una estimación más fiable del rendimiento del modelo, reduciendo el sesgo y la varianza que podrían surgir de una única división aleatoria. Además, es esencial distinguir entre el conjunto de entrenamiento, el conjunto de validación (utilizado para el ajuste de hiperparámetros) y el conjunto de prueba (utilizado únicamente para la evaluación final imparcial del modelo).

### 1.1.6. Ciencia de datos

La Ciencia de Datos emerge como una disciplina transversal que integra metodologías y principios de campos como la estadística, la informática, las matemáticas, el

conocimiento del dominio y la visualización, con el propósito primordial de extraer conocimiento, patrones e insights accionables a partir de volúmenes de datos complejos y heterogéneos. Su objetivo no es meramente el procesamiento de información, sino la capacidad de transformar datos brutos en inteligencia estratégica, facilitando la toma de decisiones informadas y la predicción de eventos futuros. En la actualidad, su aplicación es ubicua, abarcando desde la optimización de procesos empresariales hasta la investigación científica y, pertinentemente para este proyecto, la ciberseguridad.

Dentro del vasto espectro de la Ciencia de Datos, el aprendizaje automático (Machine Learning - ML) y el aprendizaje profundo (Deep Learning - DL) constituyen dos de sus pilares más potentes y de mayor crecimiento. Mientras que el aprendizaje automático engloba un conjunto de algoritmos que permiten a los sistemas aprender de los datos para realizar predicciones o tomar decisiones sin ser programados explícitamente para cada tarea, el aprendizaje profundo representa una subcategoría del ML que se basa en redes neuronales artificiales con múltiples capas, capaces de aprender representaciones de datos jerárquicas y abstractas.

En el marco de este proyecto de fin de grado, la Ciencia de Datos se erige como el eje metodológico para abordar el problema de la detección de intrusiones en redes. La fase inicial implica la adquisición y preprocesamiento de grandes volúmenes de datos de tráfico de red, los cuales se presentan con una multitud de características y métricas de conexión. Este proceso es crucial para transformar el tráfico en bruto en un formato estructurado y apto para el análisis computacional.

Posteriormente, y de manera exclusiva, se recurrirá a las técnicas de Machine Learning para desarrollar la capacidad predictiva del sistema. Específicamente, se ha optado por el entrenamiento de un modelo basado en el algoritmo Random Forest. Este modelo, conocido por su robustez, eficiencia y capacidad para manejar conjuntos de datos de alta dimensionalidad y características complejas, será el encargado de resolver un problema de clasificación. Su función cardinal será la de discernir, con alta fiabilidad, entre patrones de tráfico de red que corresponden a un comportamiento normal (benigno) y aquellos que denotan una actividad maliciosa o una posible intrusión. La implementación de esta aproximación permitirá que el sistema de detección evolucione y se adapte a nuevas amenazas basándose en el aprendizaje continuo de los patrones inherentes a los datos de la red.

Al escribir un documento extenso, como es el caso de una memoria de TFG, es habitual la necesidad de hacer referencia a partes del mismo (capítulo, sección, subsección) desde ciertos puntos del texto. Por ejemplo, en la Sección 1.1.2 se describe cómo dividir el documento principal en capítulos.

Crear estas referencias en LaTeXesto es muy sencillo y se compone de dos pasos:

1. Colocar una etiqueta tras la llave de cierre correspondiente al comando `\chapter`, `\section`, `\subsection` o `\subsection`, según el caso, usando para ello el comando `\label{etiqueta}`.
2. Desde cada punto del texto en que necesitemos hacer referencia a ese capítulo, sección o subsección usaríamos el comando `\ref{etiqueta}`, que insertará el número que corresponda.

En este propio fuente puedes ver ejemplos de uso de dichos comandos. Además de `\ref`, que inserta el número de capítulo, sección, subsección o subsubsección, también se pueden usar los comandos `\nameref` para obtener el título y `\pageref` para obtener la página. Por ejemplo, el comando[1]:

```
En la Sección~\ref{Sec.Capitulos}~\nameref{Sec.Capitulos} (véase la
pág.~\pageref{Sec.Capitulos}) se explica cómo estructurar los capítulos.
```

Produce el siguiente resultado:

*En la Sección 1.1.2 Sistema de detección de instrusiones (véase la pág. 4) se explica cómo estructurar los capítulos.*

## 1.2. Descripción del proyecto

El propósito principal de este proyecto es la implementación de un capturador de paquetes en tiempo real que sea capaz de capturar dichos paquetes y agruparlos en distintos flujos para que posteriormente, se muestren en una aplicación web basada en un sistema de detección de intrusiones aplicándole mecanismos de machine learning.

El proceso comienza con la captura del tráfico de red mediante una aplicación que tiene como fin la captura de paquetes en claro y a partir de estos, organiza los paquetes en distintos flujos y extrae las características necesarias para que pasen a un formato parecido al que se rige CICFlowMeter, que es una herramienta de generación

[1] El carácter ~ en LaTeX actúa como un espacio irrompible. Permite garantizar que las partes dispuestas a izquierda y derecha no se separarán, saltando juntas a la siguiente línea de texto en caso de que no hubiese espacio suficiente en la actual

y análisis de flujos de tráfico de red que produce flujos bidireccionales a partir de paquetes de red.

Por último, la aplicación mostrará en un dashboard, los flujos de paquetes en un panel, y en otro, aparecerán cuales de estos se consideran una amenaza. La lógica de control para discernir entre un flujo benigno y maligno, se encargá el modelo de IA que hemos elegido, el cual, estará integrado en el backend al igual que el propio capturador en tiempo real que hemos implementado.

El presente Proyecto de Fin de Grado aborda el diseño e implementación de un Sistema de Detección de Intrusiones (IDS) innovador y multifuncional que integra la captura de tráfico de red en tiempo real con técnicas avanzadas de aprendizaje automático (Machine Learning). El objetivo primordial es desarrollar una solución robusta capaz de discernir, con alta fiabilidad, entre tráfico de red legítimo (benigno) y actividad maliciosa o anómala.

El objetivo principal de este trabajo es la creación de una herramienta especializada para la captura y el procesamiento de paquetes de red. Esta aplicación ha sido diseñada con la capacidad de operar en tiempo real, monitorizando continuamente el flujo de datos a través de la red. Un aspecto distintivo de esta herramienta es su versatilidad para generar y persistir flujos de paquetes, transformando la información de bajo nivel de los paquetes individuales en un formato estructurado y significativo para el análisis posterior. La aplicación permite la conversión de estos flujos en formatos de salida estandarizados como .csv o .txt, lo que facilita la creación de nuevas bases de datos de tráfico de red. Esta funcionalidad es crucial, ya que emula y complementa la metodología empleada en datasets de referencia en ciberseguridad, como los de la serie CIC-IDS (ej., CIC-IDS2017, CIC-IDS2018, CIC-IDS2019), proporcionando una base para la investigación y el desarrollo futuros en el campo. Además, la capacidad de invocación por consola dota a la herramienta de flexibilidad para ser integrada en diversos entornos y automatizaciones, permitiendo la generación de datos para el entrenamiento y la evaluación de modelos de Machine Learning.

Para validar y demostrar la eficacia del capturador de paquetes y para completar la arquitectura del IDS, se ha implementado un módulo de detección basado en Machine Learning. Este módulo utiliza un modelo de clasificación Random Forest, seleccionado por su reconocida capacidad para manejar grandes volúmenes de datos con alta dimensionalidad y su robustez ante el ruido. El modelo ha sido entrenado para identificar patrones y características específicas que distinguen el tráfico normal de diversas categorías de ataques, lo que le permite clasificar de forma predictiva cada flujo de red entrante.

Complementando estas funcionalidades, se ha desarrollado una aplicación web, concebida como un dashboard intuitivo, que permite visualizar en tiempo real la actividad del capturador de paquetes y el rendimiento del modelo de detección. Este interfaz gráfico no solo ofrece una representación clara de los eventos de red, sino que también sirve como una herramienta práctica para monitorizar las detecciones de tráfico malicioso, facilitando la interacción del usuario con el IDS.

## 1.3. Objetivos y motivación

La proliferación incesante de las tecnologías digitales y la interdependencia sistémica de las infraestructuras de red han configurado un panorama donde la ciberseguridad ya no es una mera consideración técnica, sino un pilar fundamental para la operatividad y la resiliencia de cualquier entidad, desde organizaciones gubernamentales hasta empresas privadas y usuarios individuales. En este entorno, la capacidad de detectar y responder eficazmente a las intrusiones cibernéticas se ha vuelto crítica. Las amenazas evolucionan con una celeridad asombrosa, adoptando formas cada vez más sofisticadas y evasivas que superan las capacidades de los sistemas de defensa convencionales, como los Sistemas de Detección de Intrusiones (IDS) basados exclusivamente en firmas. Estos últimos, si bien eficientes contra amenazas conocidas y previamente catalogadas, son inherentemente limitados frente a los ataques "zero-day.º las variantes polimórficas que modifican sus patrones para eludir la detección.

Esta brecha en las capacidades de detección tradicional ha impulsado la necesidad de explorar paradigmas más adaptativos y predictivos. El aprendizaje automático (Machine Learning - ML) emerge como una solución prometedora, dada su inherente capacidad para identificar patrones complejos, anomalías y comportamientos desviados en grandes volúmenes de datos, sin requerir una programación explícita para cada posible amenaza [2]. La aplicación de ML en la detección de intrusiones no solo promete una mayor tasa de detección de ataques novedosos, sino que también ofrece la posibilidad de reducir la dependencia de las actualizaciones manuales de firmas y reglas.

La motivación principal de este Proyecto de Fin de Grado radica precisamente en la respuesta a esta necesidad crítica. Se aspira a contribuir de forma tangible al campo de la ciberseguridad mediante el desarrollo de un IDS que fusione la monitorización en tiempo real del tráfico de red con la inteligencia predictiva del Machine Learning. Más allá de la mera detección, se reconoce un desafío recurrente en la investigación de IDS: la escasez de conjuntos de datos de tráfico de red etiquetados que sean re-

presentativos y actuales para el entrenamiento y la validación de modelos de Machine Learning [3]. La creación de tales datasets es un proceso laborioso y costoso, lo que limita el progreso en la evaluación comparativa y el desarrollo de nuevas arquitecturas de detección. Esta dificultad intrínseca proporciona una motivación adicional y un objetivo secundario crucial para este proyecto: desarrollar una herramienta que no solo detecte intrusiones, sino que también facilite la generación de datasets de tráfico de red de alta calidad, emulando las metodologías de benchmarks reconocidos como los de la serie CIC-IDS (ej., CIC-IDS2017, CIC-IDS2018, CIC-IDS2019). Esta capacidad es fundamental para impulsar futuras investigaciones y para el entrenamiento continuo de modelos más robustos y adaptativos.

Con base en esta profunda motivación y el análisis de las limitaciones existentes, los objetivos de este proyecto se estructuran de la siguiente manera:

En primer lugar, el objetivo primordial es el diseño, desarrollo e implementación de un Sistema de Detección de Intrusiones (IDS) holístico y funcional, anclado en los principios del aprendizaje automático. Este sistema debe ser capaz de analizar el tráfico de red en tiempo real, distinguiendo con precisión entre patrones de comportamiento benignos y maliciosos, para así elevar la postura de seguridad de la infraestructura monitoreada.

Para materializar este objetivo general, se han delineado varios objetivos específicos que guiarán las fases de desarrollo:

- **Concebir y Construir una Herramienta Avanzada para la Captura y el Preprocesamiento de Tráfico de Red**: El desarrollo de una aplicación personalizada es fundamental. Esta herramienta no solo deberá ser eficiente en la captura de paquetes en tiempo real, sino que también integrará capacidades robustas para la reconstrucción de flujos de conexión y la extracción detallada de un amplio conjunto de características pertinentes para el análisis de seguridad. Un valor añadido de esta herramienta será su funcionalidad para generar salidas estructuradas en formatos estándar como .csv o .txt, lo que permitirá la creación sistemática de nuevas bases de datos de tráfico de red. Esta capacidad de generación de datasets, invocable incluso desde la consola, es vital para la replicabilidad, la extensibilidad y la contribución a la comunidad de investigación en ciberseguridad.

- **Investigar, Implementar y Optimizar un Modelo de Clasificación de Machine Learning de Alto Rendimiento**: La selección de un algoritmo de aprendizaje automático es crucial. Se ha optado por el algoritmo Random Forest debido a

su probada eficacia en problemas de clasificación multiclase y binaria, su resistencia al sobreajuste, y su capacidad para gestionar conjuntos de datos con un elevado número de características y una significativa dimensionalidad [4]. El modelo será sometido a un riguroso proceso de entrenamiento con datasets de tráfico etiquetado, con el fin de optimizar su rendimiento para la minimización simultánea de falsos positivos (que generan fatiga en los analistas) y, lo que es más crítico, de falsos negativos (que implican ataques no detectados) [3].

- **Asegurar la Integración Fluida entre la Captura de Datos y el Motor de Detección de Machine Learning**: Se establecerá un pipeline de procesamiento de datos que permita la alimentación continua y eficiente de los flujos de red preprocesados desde la herramienta de captura hacia el modelo de Machine Learning. Esta integración garantizará que el IDS pueda clasificar el tráfico de forma dinámica y emitir alertas inmediatas ante la detección de actividades sospechosas, operando en un modo casi en tiempo real.

- **Diseñar y Desarrollar una Interfaz Web Intuitiva para la Monitorización y Visualización**: Para facilitar la interacción del usuario y la supervisión del sistema, se implementará una aplicación web que actúe como un dashboard. Esta interfaz ofrecerá una representación clara y amigable de la actividad de la red, los eventos de detección generados por el modelo de Machine Learning, y las métricas operativas del IDS. La visualización en tiempo real es clave para proporcionar a los administradores de seguridad una panorámica inmediata del estado de la red.

- **Realizar una Evaluación Exhaustiva y Rigurosa del Rendimiento del IDS Propuesto**: La validación empírica del sistema es indispensable. Se llevarán a cabo pruebas exhaustivas utilizando metodologías de evaluación estándar para problemas de clasificación en el ámbito de la ciberseguridad. Esto incluirá el cálculo y análisis de métricas como la precisión (accuracy), sensibilidad (recall), precisión (precision), F1-score, la construcción y análisis de la matriz de confusión, y la curva ROC con su respectivo AUC [3]. El objetivo es demostrar la eficacia del IDS en la identificación de diferentes tipos de ataques y validar su viabilidad como una herramienta de seguridad operativa y de valor añadido.

A través de la consecución de estos objetivos, este proyecto no solo aspira a generar una solución técnica funcional para la detección de intrusiones, sino también a contribuir al cuerpo de conocimiento en la intersección de la ciberseguridad y la inteligencia artificial, ofreciendo herramientas y metodologías que pueden ser de utilidad para la comunidad científica y profesional.

## 1.4. Metodología de desarrollo de software

Una metodología de desarrollo de software es un conjunto de prácticas, técnicas y procedimientos que se utilizan para organizar, planificar y ejecutar proyectos de desarrollo de software. Su objetivo principal es mejorar la eficiencia y la calidad del proceso de desarrollo, asegurando que el software se entregue a tiempo, dentro del presupuesto y cumpla con los requisitos del cliente. Estas metodologías proporcionan un marco estructurado que guía a los equipos de desarrollo a través de las diferentes fases del ciclo de vida del software, desde la concepción y el diseño hasta la implementación, prueba y mantenimiento.

### 1.4.1. SCRUM como modelo de desarrollo

Dada la estructura del proyecto y sus dimensiones, se ha optado por utilizar la metodología Scrum. Scrum es una metodología ágil que se basa en la realización de incrementos pequeños y manejables, permitiendo así una mayor flexibilidad y adaptación a los cambios a lo largo del proceso de desarrollo.

Algunos de los componentes clave de Scrum son:

- **Roles: Scrum define tres roles principales**:
  - **Product Owner**: Responsable de maximizar el valor del producto y gestionar el backlog del producto.
  - **Scrum Master**: Facilita el proceso Scrum, ayuda al equipo a seguir las prácticas de Scrum y elimina impedimentos.
  - **Equipo de Desarrollo**: Grupo multifuncional que trabaja en la entrega de incrementos del producto.
- **Eventos**:
  - **Sprint**: Periodo de tiempo fijo (generalmente de 1 a 4 semanas) durante el cual se realiza un incremento del producto.
  - **Sprint Planning**: Reunión para planificar el trabajo que se realizará durante el sprint.
  - **Daily Scrum**: Reunión diaria de 15 minutos para sincronizar el trabajo del equipo.

- **Sprint Review**: Reunión para revisar el incremento y adaptar el backlog del producto si es necesario.
- **Sprint Retrospective**: Reunión para reflexionar sobre el sprint y buscar mejoras continuas.

- **Artefactos**:
  - **Product Backlog**: Lista priorizada de todo el trabajo que se necesita en el producto.
  - **Sprint Backlog**: Lista de tareas seleccionadas del product backlog para completarse en el sprint actual.
  - **Increment**: El resultado de un sprint, que debe ser un producto utilizable y potencialmente desplegable.

Scrum se caracteriza por su enfoque en la transparencia, inspección y adaptación, permitiendo a los equipos responder rápidamente a los cambios y mejorar continuamente.

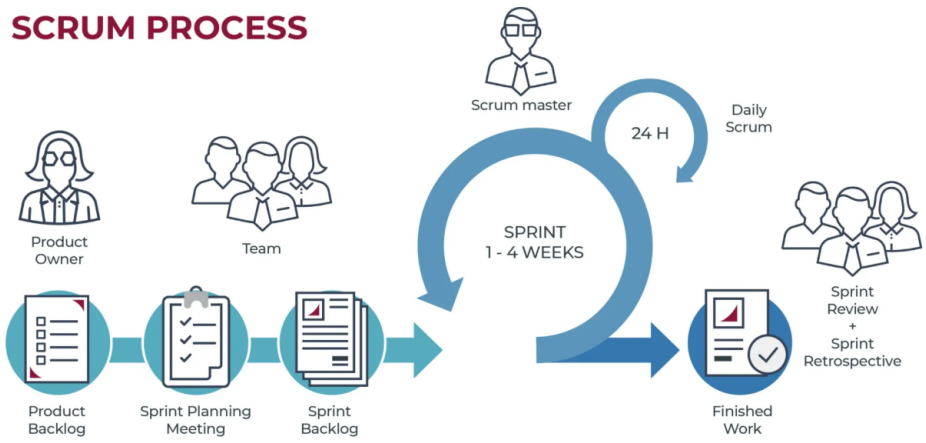


Figura 1.1: Proceso de las fases de desarrollo SCRUM.

### 1.4.2. Adaptación de SCRUM para proyectos individuales

Aunque Scrum está diseñado originalmente para equipos, sus principios y estructura pueden adaptarse para el desarrollo por una sola persona. Basándonos en el artículo “Scrum for One” de Lucidchart [5] vamos a desarrollar el proceso de adaptación de esta metodología para una sola persona.

En un entorno de un solo desarrollador, la misma persona asume los roles de Product Owner, Scrum Master y Equipo de Desarrollo. Esto requiere una clara organización y autogestión. El desarrollador único debe gestionar el backlog del producto, facilitar su propio proceso de desarrollo y eliminar impedimentos de manera autónoma. Este enfoque puede ser beneficioso ya que el desarrollador tiene control total sobre las decisiones y priorizaciones, lo que puede agilizar el proceso.

Se pueden mantener los eventos clave de Scrum, pero de manera simplificada y flexible para una sola persona:

- **Sprint Planning**: Planificar los sprints sigue siendo esencial. El desarrollador único debe dedicar tiempo a establecer objetivos claros para cada sprint, definir las tareas necesarias y priorizarlas en el sprint backlog.

- **Daily Scrum**: Aunque no hay un equipo con el cual sincronizarse, el desarrollador puede realizar una auto-revisión diaria. Esto ayuda a mantener el enfoque y la disciplina, permitiendo reflexionar sobre el progreso y ajustar el plan si es necesario.

- **Sprint Review**: Al final de cada sprint, el desarrollador revisa el trabajo completado. Este evento puede incluir la evaluación de cómo se han alcanzado los objetivos del sprint y la identificación de cualquier ajuste necesario en el backlog del producto.

- **Sprint Retrospective**: Es crucial para la mejora continua. El desarrollador reflexiona sobre lo que funcionó bien y lo que se puede mejorar para los próximos sprints.

Los artefactos de Scrum también siguen siendo importantes ya que una vez vistos las modificaciones en los eventos se puede concluir que los artefactos se usarán de la misma forma que en un equipo con varias personas.

Además, la naturaleza iterativa de Scrum permite una gran flexibilidad y capacidad de adaptación. Esto significa que el desarrollador puede responder rápidamente a los cambios y ajustar su enfoque según sea necesario, lo cual es vital en un entorno de desarrollo dinámico. Las retrospectivas regulares fomentan una cultura de mejora continua, permitiendo identificar áreas de mejora y aplicar cambios en ciclos cortos, lo que facilita la evolución y optimización del proceso de desarrollo.

### 1.4.3. Sprints de desarrollo

El desarrollo de un sistema tan complejo como un Sistema de Detección de Intrusiones (IDS) que integra captura de tráfico en tiempo real, procesamiento de datos, análisis mediante aprendizaje automático y una interfaz web, exige una metodología ágil que permita gestionar la complejidad, adaptarse a los desafíos emergentes y entregar valor de forma incremental. Para este proyecto, se adoptó un enfoque basado en Sprints de Desarrollo, inspirados en la metodología SCRUM. Esta aproximación facilitó la división del proyecto en iteraciones cortas y manejables, cada una enfocada en la consecución de funcionalidades específicas, garantizando así un progreso continuo y la capacidad de integrar retroalimentación a lo largo del ciclo de vida del desarrollo. A continuación, se describen los sprints planificados y ejecutados, delineando los objetivos y entregables clave de cada fase.

**Sprint 1: Fundamentos Teóricos y Análisis de Requisitos**

Este sprint inaugural se centró en la construcción de una base de conocimiento sólida indispensable para abordar un proyecto de la envergadura de un IDS basado en Machine Learning. Las primeras semanas se dedicaron a una inmersión profunda en la literatura científica y técnica. Se revisó una colección de papers y artículos especializados proporcionados por la dirección del proyecto, enfocados en la intersección de la inteligencia artificial y la ciberseguridad, con especial énfasis en los principios y arquitecturas de los Sistemas de Detección de Intrusiones [5], [2]. Paralelamente, se llevó a cabo una investigación exhaustiva en bases de datos académicas y repositorios en línea para complementar el entendimiento de conceptos fundamentales de IDS y los distintos paradigmas de inteligencia artificial, inicialmente explorando tanto el aprendizaje supervisado como el no supervisado. Una vez afianzado el conocimiento sobre las técnicas de aprendizaje automático, la atención se dirigió hacia la identificación y análisis de conjuntos de datos (datasets) adecuados para el entrenamiento y evaluación de modelos de IDS. Se familiarizó con la estructura, características y taxonomía de datasets de referencia en la comunidad, tales como CIC-IDS (2017, 2018, 2019), KDD99, NSL-KDD, y UNSW [3], comprendiendo sus particularidades y la información que estos proporcionan para la detección de anomalías. Este sprint concluyó con una clara comprensión del marco teórico y una dirección definida hacia la aplicación de técnicas de aprendizaje supervisado, sentando las bases conceptuales para las fases de implementación.

**Sprint 2: Diseño del Módulo de Captura y Selección de Tecnologías**

Con la base teórica establecida, el segundo sprint se focalizó en el diseño arqui-

tectónico del módulo de captura de paquetes y la selección de las tecnologías más adecuadas. La fase inicial de diseño incluyó la evaluación de posibles integraciones con herramientas existentes como Wireshark; sin embargo, las complejidades asociadas a su código base y las potenciales dificultades de integración con componentes de IA llevaron a su descarte. Se llevó a cabo una investigación profunda sobre los lenguajes de programación y las librerías disponibles para el desarrollo de capturadores de paquetes eficientes. Esta fase concluyó con la elección de Python, un lenguaje que demostró ser idóneo por su robustez, su amplio ecosistema de librerías para la manipulación de redes [6] y su facilidad de integración con frameworks de Machine Learning [7]. Crucialmente, se identificó una API especializada que proporcionaba un conjunto rico de funcionalidades para la extracción de características detalladas a partir del tráfico de red. Al finalizar este sprint, se dispuso de un diseño conceptual claro del capturador y una pila tecnológica definida para su implementación.

**Sprint 3: Implementación Básica del Capturador de Paquetes y CLI**

Este sprint se dedicó a la implementación inicial del capturador de paquetes en Python. Se desarrolló la funcionalidad core que permite la intercepción y procesamiento de paquetes de red. El foco principal fue la creación de una interfaz de línea de comandos (CLI) que posibilitara la invocación del programa con parámetros específicos, tales como la interfaz de red a monitorear. Se logró que el capturador pudiera iniciar y detener la recolección de flujos de paquetes desde la terminal, sentando las bases operativas de la herramienta. Las pruebas iniciales se centraron en verificar la correcta captura y el parseo básico de los datos de los paquetes, asegurando que el flujo de información se manejara de manera estable.

**Sprint 4: Desarrollo de la Funcionalidad de Exportación de Datos**

Continuando con el módulo de captura, el cuarto sprint se concentró en una funcionalidad crítica para la utilidad del proyecto: la exportación de los flujos de paquetes procesados a formatos estructurados. Reconociendo la dificultad de obtener datasets de IDS actualizados, se implementó la capacidad de generar archivos en formato .csv y .txt a partir de los flujos capturados. Esta funcionalidad es fundamental, ya que permite que el IDS contribuya a la creación de nuevas bases de datos de tráfico de red, emulando la estructura de datasets de referencia como los de la serie CIC-IDS. Esta capacidad es clave para la investigación y el desarrollo de modelos de Machine Learning, al proporcionar acceso a datos más representativos del tráfico moderno. Se realizaron pruebas exhaustivas para validar la integridad y el formato de los archivos generados, asegurando que pudieran ser utilizados directamente para el preprocesamiento y entrenamiento de modelos de IA.

**Sprint 5: Definición de la Arquitectura Web y Prototipado de la Interfaz**

Con el módulo de captura en progreso, este sprint abordó el diseño y la arquitectura de la aplicación web que albergaría el IDS completo. Se llevó a cabo una investigación de frameworks de desarrollo web full-stack, evaluando opciones como Angular, React (para frontend) combinados con backends como Laravel (PHP) o Flask/Django (Python). La decisión final se inclinó por Reflex, un framework de Python que ofrecía una solución unificada para el desarrollo full-stack en un solo lenguaje, lo que simplificaba la integración con los componentes de Python existentes y optimizaba el flujo de trabajo. Una vez seleccionado el framework, se procedió al análisis detallado y el diseño de la interfaz de usuario (UI). Esto incluyó la elaboración de diagramas de flujo de usuario, la creación de mockups visuales y wireframes para definir la estructura y navegación del dashboard. El objetivo fue crear un diseño intuitivo que facilitara la visualización del tráfico de red y las futuras detecciones.

**Sprint 6: Implementación del Frontend y Conexión al Backend del Capturador**

Este sprint se centró en la construcción del frontend de la aplicación web según los diseños establecidos en el sprint anterior. Se implementaron las principales pantallas y componentes de la interfaz de usuario, dando vida al dashboard. Simultáneamente, se inició el desarrollo de la capa de backend de la aplicación web, preparada para recibir y procesar los datos del capturador. Un hito crítico de este sprint fue la integración inicial del módulo de captura de paquetes con el backend de la aplicación web. Para lograr una comunicación en tiempo real y asíncrona entre ambos componentes, se implementó un sistema basado en colas síncronas. Las pruebas de este sprint se dirigieron a verificar que los datos de tráfico capturados por el módulo de Python se transmitieran correctamente al backend y que el panel de visualización del tráfico de red en el frontend se actualizara en tiempo real, confirmando la operatividad del flujo de datos de extremo a extremo.

**Sprint 7: Preprocesamiento y Unificación del Dataset para ML**

Paralelo al avance de la aplicación web, este sprint se dedicó intensivamente a la preparación del conjunto de datos para el entrenamiento del modelo de Machine Learning. Se seleccionó el CIC-IDS2018 como dataset principal debido a su representatividad y variedad de ataques [3]. La tarea crucial fue la unificación de los diversos ficheros que componían este dataset (correspondientes a diferentes días de captura y tipos de ataque) en una única base de datos cohesiva. Esta consolidación fue fundamental para asegurar que el modelo de aprendizaje tuviera acceso a un volumen variado y robusto de flujos de tráfico, crucial para su capacidad de generalización [3]. Tras la unificación, se llevó a cabo un exhaustivo proceso de preprocesamiento, análi-

sis y limpieza de los datos. Esto incluyó la gestión de valores ausentes, la corrección de inconsistencias, la normalización/estandarización de características numéricas y la codificación de variables categóricas, preparando el dataset para la fase de entrenamiento [8].

**Sprint 8: Entrenamiento y Evaluación del Modelo de Machine Learning (Random Forest)**

Una vez que el dataset estuvo preparado y limpio, este sprint se centró por completo en el desarrollo y la evaluación del modelo de Machine Learning. Se procedió al entrenamiento del clasificador Random Forest, un algoritmo elegido por su eficacia en la detección de intrusiones en entornos de alta dimensionalidad y su robustez ante el sobreajuste [4]. Durante esta fase, se ajustaron los hiperparámetros del modelo para optimizar su rendimiento. Tras el entrenamiento, se realizó una evaluación rigurosa del modelo utilizando métricas clave de clasificación como la precisión (accuracy), la sensibilidad (recall), la precisión (precision), el F1-score, y el análisis detallado de la matriz de confusión y la curva ROC con su Área bajo la Curva (AUC) [3]. Este análisis permitió determinar la capacidad predictiva del modelo y su fiabilidad para distinguir entre tráfico benigno y malicioso. El entregable de este sprint fue un modelo de Machine Learning entrenado y validado, listo para ser integrado en el sistema en tiempo real.

**Sprint 9: Integración Final del Modelo ML y Detección en Tiempo Real**

Este sprint fue el punto culminante de la integración de todos los componentes del IDS. El modelo de Machine Learning entrenado fue integrado en el backend de la aplicación web. Esto permitió que los flujos de paquetes, capturados por el módulo Python y transmitidos a través de las colas síncronas al backend, fueran ahora analizados en tiempo real por el modelo Random Forest. La funcionalidad clave de este sprint fue la capacidad del sistema para determinar si un flujo de tráfico era benigno o malicioso de forma automática y mostrar esta clasificación instantáneamente en la interfaz web. Se realizaron pruebas de integración de extremo a extremo para asegurar que el ciclo completo, desde la captura hasta la predicción y la visualización, operara sin fallos y con la menor latencia posible.

**Sprint 10: Pruebas Exhaustivas, Optimización y Validación Final**

El sprint final se dedicó a la consolidación y refinamiento de todo el sistema. Se ejecutaron multitud de pruebas exhaustivas y multifacéticas para garantizar la robustez, la estabilidad y el rendimiento óptimo de la aplicación en su conjunto. Esto incluyó pruebas de estrés para verificar la capacidad del capturador y del modelo bajo cargas

de tráfico elevadas, pruebas de usabilidad de la interfaz web, y pruebas de regresión para asegurar que las nuevas funcionalidades no introdujeran fallos en componentes existentes. Asimismo, se llevó a cabo una fase de optimización general del rendimiento de la aplicación, buscando mejorar los tiempos de respuesta del capturador, la latencia en la predicción del modelo y la fluidez de la interfaz de usuario. Finalmente, se realizó una validación completa del sistema [9] para confirmar que todos los objetivos del proyecto habían sido alcanzados, que el IDS operaba de manera confiable y que estaba listo para su defensa.

## 1.5. Prespuesto del proyecto

La planificación y ejecución de cualquier proyecto, incluyendo un Trabajo Fin de Grado, conlleva una serie de costes asociados que deben ser estimados y desglosados para comprender la inversión económica necesaria. Este apartado detalla el presupuesto aproximado del desarrollo del Sistema de Detección de Intrusiones (IDS) y la herramienta de generación de datasets a lo largo de los ocho meses de duración estimados para el proyecto. Para la elaboración de este presupuesto, se han considerado los recursos humanos implicados, los recursos materiales utilizados y los costes indirectos asociados.

**Recursos Humanos**

Aunque este proyecto ha sido realizado por un único estudiante en el marco de su Trabajo Fin de Grado, para fines de presupuesto y para reflejar un escenario de desarrollo profesional, se estimará el coste equivalente a la contratación de un Programador Junior / Investigador Junior en Ciberseguridad. Este perfil asumiría las responsabilidades de diseño, programación, investigación y gestión del proyecto.

Para la estimación salarial, se puede tomar como referencia el Convenio colectivo estatal de empresas de consultoría y estudios de mercado y de la opinión pública, o bien, promedios de salarios para perfiles junior en el sector tecnológico en España. Considerando un salario anual bruto para un programador/investigador junior en un rango de 18.000€ a 22.000€ (más realista para el perfil y las habilidades requeridas en Machine Learning y ciberseguridad que los 14.800,66€ mencionados en un convenio específico que puede no aplicar directamente al rol técnico avanzado), tomaremos un valor medio de 20.000€ anuales para este cálculo.

- **Salario base anual estimado**: 20.000,00 €

- **Salario mensual:** $\frac{20{,}000{,}00}{12} = 1{,}666{,}67$ €/mes
- **Coste total de recursos humanos (8 meses):** $1{,}666{,}67$ €/mes $\times$ $8$ meses $=$ $13{,}333{,}36$ €

A este coste salarial se le deberían añadir los costes sociales (aportaciones a la Seguridad Social por parte de la empresa), que en España suelen rondar el 30-35 % del salario bruto. Para una estimación, consideraremos un 32 %.

- **Costes Sociales mensuales:** $1{,}666{,}67$ €/mes $\times$ $0{,}32 = 533{,}33$ €/mes
- **Costes Sociales totales (8 meses):** $533{,}33$ €/mes $\times$ $8$ meses $= 4{,}266{,}64$ €
- **Coste total de Recursos Humanos (Salario + Costes Sociales):** $13{,}333{,}36$ € $+$ $4{,}266{,}64$ € $= 17{,}600{,}00$ €

**Recursos Materiales**

El desarrollo del proyecto ha requerido el uso de equipamiento informático para la programación, la ejecución de modelos de Machine Learning y la realización de pruebas de captura de tráfico en tiempo real. Aunque los equipos ya eran propiedad del desarrollador, para el presupuesto se considerará una amortización proporcional al tiempo de uso del proyecto.

Se han utilizado los siguientes equipos principales:

- **Ordenador Portátil (ej. HP Pavilion 16-a0003N o similar)**:
  - **Coste de adquisición**: 899,57 €
  - **Vida útil estimada**: 4 años (48 meses)
  - **Amortización mensual**: $\frac{899{,}57}{48} = 18{,}74$ €/mes
  - **Coste por 8 meses**: $18{,}74$ €/mes $\times$ $8$ meses $= 149{,}92$ €
- **Smartphone (ej. Xiaomi Mi 10 5G o similar, para pruebas de red móvil si aplicase, o como dispositivo de test para la app web)**:
  - **Coste de adquisición**: 300,00 € (basado en la referencia del compañero)
  - **Vida útil estimada**: 3 años (36 meses)
  - **Amortización mensual**: $\frac{300{,}00}{36} = 8{,}33$ €/mes
  - **Coste por 8 meses**: $8{,}33$ €/mes $\times$ $8$ meses $= 66{,}64$ €

- **Software y Herramientas (Licencias/Suscripciones)**: Aunque la mayoría de las herramientas utilizadas (Python, librerías como Scikit-learn, Reflex, Wireshark/Scapy, etc.) son de código abierto o gratuitas, en un entorno profesional se considerarían costes para herramientas de desarrollo (IDEs avanzados), sistemas de control de versiones premium (GitHub Enterprise), o servicios de computación en la nube para el entrenamiento de modelos más grandes (ej. Google Colab Pro, AWS, Azure, etc.). Para este TFG, asumiremos costes mínimos o nulos por licencias de software, pero se podría estimar un coste simbólico por el acceso a ciertos recursos o licencias de herramientas de diseño/gestión si se hubieran utilizado versiones de pago. Aquí asumiremos 0€ dado el contexto de TFG, pero se podría incluir un pequeño monto (ej. 50-100€) para una licencia de IDE o software de diseño.
- **Conexión a Internet y Electricidad**: Estos costes se suelen incluir en los costes operativos indirectos de un proyecto. Considerando una parte proporcional del coste de una conexión doméstica y el consumo eléctrico del equipo durante las horas de desarrollo.
  - **Conexión a Internet**: $\frac{50\ €/\text{mes}}{2}$ (uso profesional) $\times\ 8$ meses $= 200,00\ €$
  - **Electricidad**: $\frac{30\ €/\text{mes}}{2}$ (uso profesional) $\times\ 8$ meses $= 120,00\ €$
- **Total de Recursos Materiales**: $149,92\ € + 66,64\ € + 0,00\ € + 200,00\ € + 120,00\ € = 536,56\ €$

#### Costes Indirectos y Contingencias

Estos costes cubren gastos generales y posibles imprevistos no directamente asignables a las categorías anteriores. Incluyen, por ejemplo, material de oficina, formación específica no prevista, gastos de comunicación, etc. Se suele estimar un porcentaje sobre el total de los costes directos.

- **Costes Directos Totales (Recursos Humanos + Recursos Materiales)**: $17{,}600,00\ € + 536,56\ € = 18{,}136,56\ €$
- **Contingencias (10 % de los costes directos)**: $18{,}136,56\ € \times 0{,}10 = 1{,}813,66\ €$

#### Resumen del presupuesto final del proyecto

En la siguiente tabla 1.1 se puede observar el resumen del desglose de los costes estimados para el desarrollo de este Trabajo Fin de Grado durante un periodo de 8 meses:

| Concepto | Detalle | Coste (euros) |
|---|---|---|
| **Recursos Humanos** | Programador/Investigador Junior (8 meses) | |
| | Salario Bruto | 13.333,36 |
| | Costes Sociales (32 %) | 4.266,64 |
| **Recursos Materiales** | | |
| | Amortización Ordenador Portátil (8 meses) | 149,92 |
| | Amortización Smartphone (8 meses) | 66,64 |
| | Software (Licencias/Suscripciones) | 0,00 |
| | Conexión a Internet (parte proporcional) | 200,00 |
| | Consumo Eléctrico (parte proporcional) | 120,00 |
| **Costes Indirectos** | | |
| | Contingencias (10 % de costes directos) | 1.813,66 |
| **TOTAL GENERAL** | | **19.910,22** |

Tabla. 1.1: Presupuesto del proyecto

Este presupuesto final de aproximadamente 19.910,22 euros proporciona una estimación realista del valor de mercado de un proyecto de esta envergadura y complejidad si se llevara a cabo en un entorno profesional, abarcando la investigación, el desarrollo de un módulo de captura especializado, la implementación de un modelo de Machine Learning y la creación de una aplicación web interactiva.

## 1.6. Planificación de tiempo y costes

La gestión del tiempo es un pilar fundamental en la ejecución exitosa de cualquier proyecto, especialmente en un Trabajo Fin de Grado (TFG) que requiere una secuencia lógica de actividades y una dedicación constante. La planificación temporal de este proyecto se ha basado en la metodología ágil de **Sprints de Desarrollo** citados en el apartado 1.4.3, la cual ha permitido una organización estructurada de las tareas, la adaptación a los desafíos y la entrega incremental de funcionalidades, tal como se detalló en la sección de Metodología.

Para la estimación y seguimiento del tiempo, se ha considerado un período de ocho meses, que abarca desde la fase inicial de los fudamentos teóricos hasta la validación y documentación final. La duración de cada sprint fue estimada en dos semanas apro-

ximadamente, permitiendo flexibilidad para la investigación y el desarrollo intensivo de cada módulo. Este enfoque se alinea con prácticas comunes en la planificación de proyectos de ingeniería de software, buscando optimizar el esfuerzo y los recursos disponibles [9].

Para este proyecto, el cronograma se presenta como una secuencia de los sprints ya definidos, cada uno con sus objetivos específicos y una duración estimada.

### 1.6.1. Cronograma del Proyecto

A continuación, se presenta el cronograma detallado del proyecto, desglosado por los sprints de desarrollo en la figura 1.2. Este cronograma visualiza la secuencia de las actividades principales y la estimación de su duración en meses.

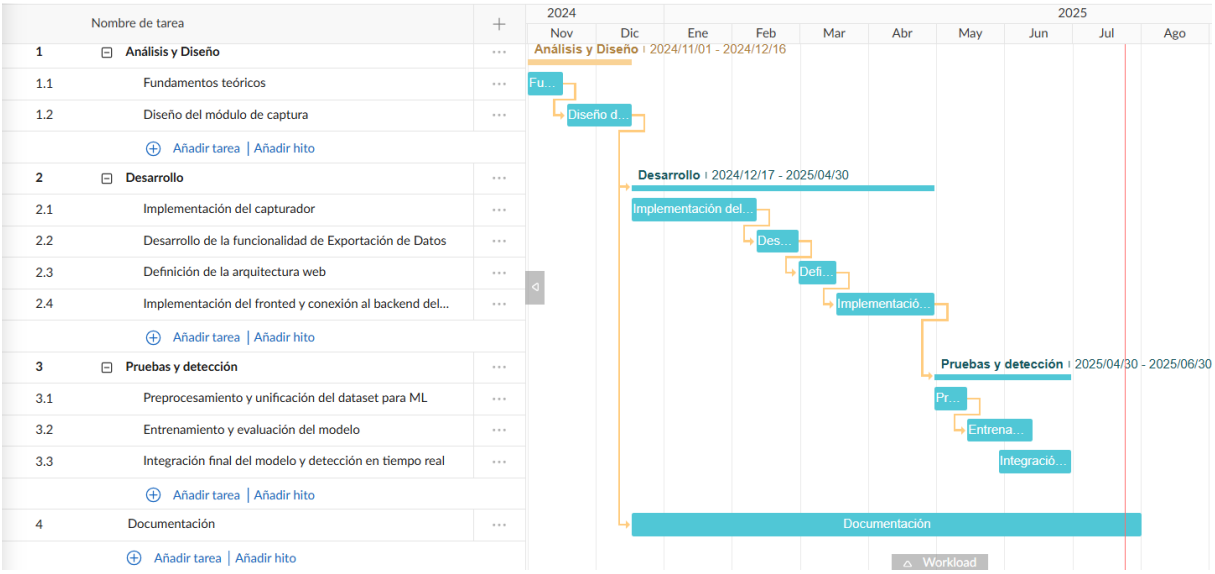


Figura 1.2: Diagrama de Gantt del proyecto. Fuente: Elaboración propia.

Este cronograma proporciona una guía para la ejecución del proyecto, destacando la superposición de algunos sprints (como la investigación teórica con el diseño inicial, o el preprocesamiento de datos con el desarrollo web) para optimizar el tiempo. La naturaleza iterativa de la metodología ágil permite ajustes en la duración de los sprints según las necesidades emergentes y la complejidad de las tareas [7]. La dedicación principal al proyecto se concentró en la ejecución de estos sprints, culminando con una fase intensiva de integración y pruebas antes de la entrega final.

### 1.6.2. Estimación del Esfuerzo

La estimación del esfuerzo para el presente Trabajo de Fin de Grado se ha realizado en conformidad con la carga académica establecida para un TFG de 12 créditos ECTS, lo que se traduce en un total de 300 horas de trabajo dedicado. Este volumen de esfuerzo se ha distribuido estratégicamente a lo largo de la duración del proyecto, que abarca desde Noviembre de 2024 hasta finales de Julio de 2025, como se detalla en el cronograma.

Para un perfil de Programador/Investigador Junior, la dedicación requerida para un proyecto de esta envergadura se traduce en una asignación meticulosa de horas a cada fase, cubriendo desde la investigación y el diseño hasta la implementación, depuración y la elaboración de la documentación. Este enfoque es consistente con las prácticas comunes en la planificación de proyectos de software.

El esfuerzo total de 300 horas se ha distribuido entre las distintas fases y actividades del proyecto, alineándose con la metodología de sprints empleada. A continuación, se detalla la asignación de estas horas en las principales etapas:

- **Análisis y Diseño**: Incluye la investigación teórica, la definición de requisitos funcionales y no funcionales, y la conceptualización de la arquitectura inicial del sistema.

- **Desarrollo e Implementación**: Abarca la codificación y construcción de los diversos módulos, como el capturador de tráfico, la herramienta de generación de datasets, el componente de Machine Learning y la interfaz de usuario web.

- **Pruebas y Validación**: Comprende la ejecución de baterías de pruebas, la validación del rendimiento del modelo de Machine Learning y la verificación integral de la funcionalidad del sistema.

- **Documentación**: Incluye la redacción exhaustiva de esta memoria final de grado, la cual culminará a finales de julio de 2025, y la preparación de otros materiales de apoyo.

  Esta distribución del esfuerzo, similar a la aplicada en otros proyectos académicos que buscan cumplir con un número de créditos específicos, asegura una asignación proporcional de recursos al trabajo técnico, la investigación y la formalización documental. La planificación detallada y el progreso de estas actividades se visualizan en el Diagrama de Gantt del proyecto, presentado en la Figura 1.2.

## 1.7. Estructura de la memoria

La memoria final de este proyecto está organizada en una serie de capítulos que se ajustan a la siguiente estructura:

- Capítulo 2: Capítulo en el que se discute el estado del arte de sistema de ciberseguridad que apican ML, como diversas líneas de investigación y desarrollo al respecto
- Cápitulo 3:
- Cápitulo 4:
- Cápitulo 5:
- Cápitulo 6:
- Cápitulo 7:
- Cápitulo 8:

## 1.8. Texto en negrita, cursiva y monoespacio

Los estilos tipográficos de las distintas partes de la memoria, como son los títulos de capítulos, secciones o los propios párrafos de texto, están predeterminados. En el contenido podemos emplear esencialmente tres variantes:

- **Negrita:** debe usarse de forma puntual para **destacar algo importante**, evitando abusar de ello ya que si hay mucho texto en negrita el efecto tendrá poca utilidad. Se usa el comando `\textbf{texto en negrita}`.
- *Cursiva:* se emplea generalmente para introducir términos en otro idioma, por ejemplo GPU (*Graphics Processing Unit*), y esporádicamente con otros fines. Se usa el comando `\textit{texto en cursiva}`.
- `Monoespacio`: se usa para diferenciar nombres de variables, funciones, archivos y, en general, todo lo relacionado con código, por ejemplo: la función `printf()` de C. Se usa el comando `\texttt{texto en monoespacio}`.

Aunque LaTeX nos permite trabajar con colores (`\textcolor{color}{texto}`), diferentes tamaños de fuente (`\normalsize`, `\small`, `\footnotesize`, etc.), recuadros con fondo de color, etc., en la normativa de TFG de la EPSJ no se especifica que pueda hacerse uso de los mismos, por lo que se desaconseja hacerlo.

## 1.9. Cómo insertar notas

Las notas a pie de página[2] son un recurso útil para aclarar algo o facilitar información complementaria sin romper el hilo de lo que está tratándose en el texto. Al igual que el texto en negrita, es un recurso del que tampoco debe abusarse, empleándose únicamente cuando es estrictamente necesario.

Para incluir una nota al pie nos serviremos del comando `\footnote{Texto}`, colocándolo allí donde queremos que aparezca el número de la nota como superíndice, tal y como se aprecia en el fuente LaTeX de este mismo texto.

En casos excepcionales, cuando ya no queda espacio disponible en la página actual, una nota al pie puede desplazarse a la página siguiente de donde se hace referencia a ella.

## 1.10. Cómo insertar listas numeradas y sin numerar

Un recurso muy habitual al redactar una memoria son las listas, tanto numeradas como sin numerar. Por su disposición en el texto, con el sangrado y las marcas que preceden cada elemento, es recomendable usar listas para cualquier enumeración de elementos o descripción de procedimientos compuestos de múltiples pasos. La lisa numerada se debería usar siempre y cuando el orden sea importante, en caso contrario es preferible una lista sin numerar.

Las listas, al igual que otros elementos en LaTeX, puede ocupar múltiples párrafos de texto, incluso extendiéndose por varias páginas, de ahí la necesidad de marcar su inicio y su fin, definiendo lo que se denomina **un ámbito** o *environment*. Los ámbitos siempre se inician con el comando `\begin{}` y se finalizan con el comando `\end{}`. El tipo de ámbito, y por tanto su contenido, viene determinado por el nombre que dispongamos entre las llaves.

[2] Estas notas se numeran automáticamente dentro de cada capítulo de la memoria.

Para el caso de las listas usaremos dos ámbitos distintos:

- `itemize`: para generar una lista no numerada
- `enumerate`: para producir una lista numerada

En el interior del ámbito usaremos el comando `\item` para ir introduciendo los elementos que forman la lista, por ejemplo:

```
\begin{itemize}
    \item \texttt{itemize}: para generar una lista no numerada
    \item \texttt{enumerate}: para producir una lista numerada
\end{itemize}
```

Listado 1.7: Creación de una lista no numerada con dos elementos

Las listas pueden anidarse, es decir, podemos tener una lista dentro de otra, independientemente de cuál sea el tipo de cada una. El sangrado, tipo de boliche[3] y esquema de numeración se ajustarán de forma automática.

## 1.11. Cómo insertar figuras

A lo largo de una memoria de TFG es preciso incluir distintos tipos de figuras: diagramas que describen cómo funciona un sistema, ilustraciones y gráficas que representan o evalúan unos resultados, capturas de pantalla de un programa, etc.

### 1.11.1. Aspectos importantes al usar figuras

Las normas a tener presentes a la hora de introducir figuras en la memoria son las indicadas a continuación:

- Todas las figuras habrán de estar numeradas, usando una numeración consecutiva para toda la memoria o bien relativa a cada capítulo.
- Todas las figuras han de contar con un pie de figura que describa claramente qué representa. No ocurre nada si el pie de figura es extenso y ocupa varias líneas.

[3] La marca que aparece al inicio de cada elemento de una lista no numerada

- Todas las figuras han de estar referenciadas desde el texto de la memoria, es decir, en algún punto de los párrafos previos o posteriores ha de aparece una referencia del tipo «*como se aprecia en la Figura N*».
- **Importante:** todas las figuras deberán indicar su procedencia, de no indicarse se entenderá que son de elaboración propia.

Trata siempre de que las figuras tengan una calidad suficiente, que en el documento resultante no se vean borrosas o pixeladas, ya que causan un mal efecto.

### 1.11.2. Cómo almacenar las figuras

En ocasiones las figuras a usar serán capturas de pantalla, en otras gráficas producidas con algún programa y otras imágenes procedentes de alguna fuente. Siempre que sea posible, por ejemplo para gráficas propias generadas a partir de datos o diagramas confeccionados con alguna aplicación, se recomienda almacenar la imagen en un formato vectorial, como puede ser `EPS` o `PDF`, ya que al insertarse en el documento generalmente conservan una mejor calidad.

Para imágenes no vectoriales puedes usar los formatos `PNG` y `JPEG`. Dependiendo de la resolución de la pantalla en que se haga la captura, la imagen tendrán mayor o menor calidad. En caso necesario, se puede usar la utilidad `convert` del software ImageMagick (`legacy.imagemagick.org`) para incrementar el número de puntos por pulgada, de manera que al insertar la imagen en el PDF mejore su calidad.

Se recomienda dar una denominación adecuada a los archivos donde se guardan las figuras, de forma que sea fácil de recordar para luego introducirlas en el fuente LaTeX. Asimismo, es recomendable llevar las imágenes a una carpeta aparte, ya sea general para todas las figuras o bien por capítulo.

### 1.11.3. Cómo insertar las figuras en el documento

El comando para incluir un archivo de tipo gráfico, ya sea vectorial o no, en un documento es `\includegraphics[]{}`. Entre los corchetes se usarán habitualmente los parámetros `width` o `height` para establecer el ancho o alto que queremos dar a la figura. Lo más recomendable es usar dimensiones relativas al espacio disponible, por ejemplo: `[width=0.5\textwidth]` para conseguir que la figura ocupe la mitad del an-

cho de página y se mantenga la proporción alto/ancho. Dentro de las llaves se indicará el nombre del archivo que contiene la imagen.

Dado que aparte de la imagen en sí la figura ha de contar también con un pie o título, definido con el comando `\caption`, y además es habitual asociar una etiqueta a la figura para así poder hacer referencia a ella desde el texto, usando el comando `\label` que se explicó anteriormente, es preciso agrupar todos estos elementos dentro de un ámbito de tipo `figure`. Por ejemplo, el código del Listado 1.8 daría lugar a la inserción de la Figura 1.3. El comando `\centering`, introducido dentro del ámbito `figure`, hace que la figura aparezca centrada en lugar de alineada a la izquierda.

```
\begin{figure}[ht!]
  \centering
  \includegraphics[width=0.4\textwidth]{imagenes/uja.jpg}
  \caption{Logo oficial de la Universidad de Jaen.}
  \label{Fig.LogoUJA}
\end{figure}
```

Listado 1.8: Insercción de una figura en el documento


Figura 1.3: Logo oficial de la Universidad de Jaén.

Las referencias desde el texto a la figura se generarían según lo explicado en la subsección *1.1.6 Ciencia de datos*.

## 1.12. Cómo insertar tablas

Las tablas son un elemento también muy habitual en una memoria de TFG, especialmente si de tipo experimental y, en consecuencia, ha de informar sobre resultados obtenidos de los experimentos realizados. Las tablas probablemente sean el tipo de

elemento más complejo de LaTeXpor su flexibilidad, ya que pueden tener distribuciones no regulares y contener cualquier elemento en cada celdilla, incluso otras tablas.

### 1.12.1. Aspectos importantes al usar tablas

Las normas a tener presentes a la hora de introducir tablas en la memoria son las indicadas a continuación:

- Todas las tablas han de estar numeradas, usando una numeración consecutiva para toda la memoria o bien relativa a cada capítulo.
- Todas las tablas han de contar con un pie de tabla que describa claramente qué información facilita. No ocurre nada si el pie de tabla es extenso y ocupa varias líneas. Es especialmente importante describir qué hay en cada columna en caso de que los títulos de la primera fila no sean autoexplicativos[4]
- Todas las tablas han de estar referenciadas desde el texto de la memoria, al igual que las figuras.
- El formato de todas las tablas introducidas en la memoria será homogéneo, constando de:

    1. Una línea horizontal que da inicio a la tabla
    2. Una línea con los encabezados de cada columna en negrita
    3. Una línea horizontal de separación entre encabezados y contenido
    4. Tantas líneas como filas deba contener la tabla
    5. Una línea horizontal de cierre de la tabla
    6. El pie de tabla

Al igual que las figuras o los títulos de sección y subsección, las tablas permiten *romper* una sucesión de múltiples párrafos de texto haciendo más fácil la lectura de la memoria.

[4]En ocasiones conviene poner un título de columna corto a las columnas, para conseguir que el ancho de la tabla no exceda del ancho de la página, usándose el pie para facilitar una descripción más pormenorizada.

### 1.12.2. Cómo almacenar las tablas

Las tablas pueden introducirse directamente en el código fuente LaTeX, pero en ocasiones puede interesar tenerlas almacenadas en archivos independientes, por ejemplo si han sido generadas por otro programa. En ese caso se guardarán en un archivo con extensión `.tex` y se incluirán en el capítulo que corresponda usando el comando `\input{ruta/archivo.tex}`.

### 1.12.3. Cómo insertar las tablas en el documento

El ámbito para incluir una tabla en la memoria es `tabular{}`[5]. Dentro de las llaves se indicará el número de columnas con que contará la tabla y su alineación, usando para ello los siguientes indicadores:

- `l`: columna de longitud variable con alineación a la izquierda
- `r`: columna de longitud variable con alineación a la derecha
- `c`: columna de longitud variable con alineación al centro
- `p{N.Mcm}`: columna de longitud fija `N.M` centímetros

El ancho de la tabla será la suma de los anchos de sus columnas. Debe tenerse en cuenta que ese ancho se extenderá todo lo necesario para acoger su contenido salvo para columnas de tipo `p`.

Dentro del ámbito `tabular` incluiremos los siguientes elementos por este orden:

1. `\toprule`: una línea horizontal que delimita el inicio de la tabla
2. **Encabezado**: una fila de encabezado con los títulos de las columnas en negrita. Cada columna se separará de la siguiente con el carácter `&` y el final de la fila se marcará con `\\`
3. `\midrule`: una línea horizontal que separa el encabezado del contenido de la tabla
4. **Contenido**: tantas filas de contenido como sean precisas, separando el dato de cada columna del de la siguiente con el carácter `&` y marcando el final de cada fila con `\\`

[5] Recuerda que al ser un ámbito deberá ir delimitado con los comandos `\begin` y `\end`.

5. `\bottomrule`: una línea horizontal para marcar el final de la tabla

Dado que la tabla debe contar además con un título y también un etiqueta, como las figuras, será preciso usar los comandos `\caption` y `\label` ya explicados. Todos ellos se unirán en un entorno `table`, tal y como muestra el ejemplo del Listado 1.9 que produce como resultado la Tabla 1.2.

```
\begin{table}[ht!]
  \centering
  \small
  \def\arraystretch{1.5}
  \begin{tabular}{llp{10cm}}
    \toprule
      \textbf{Componente} & \textbf{Requerimiento} & \textbf{Descripcion} \\
    \midrule
      Procesador & Arquitec. x64  & Procesador con varios (\textit{cores}) para poder
          ejecutar los distintos contenedores que forman el sistema \\

      RAM        & 64GB  & Disponer de suficiente memoria RAM es importante para
          garantizar un funcionamiento correcto del sistema \\

      Disco      & 10TB  & Almacenamiento masivo suficiente para contener los datos. Se
          recomienda una velocidad de al menos 7200rpm \\

      GPU        & CCC $\ge$ 3.5 & GPU para juegos con CUDA Compute Capability 3.5 o
          posterior y al menos 8GB de memoria \\

      Red        & GbE & Gigabit Ethernet para el acceso del sistema a web \\
    \bottomrule
  \end{tabular}
  \caption{Requisitos hardware del ordenador }
  \label{Tabla.Requisitos}
\end{table}
```

Listado 1.9: Creación de una tabla

En este ejemplo se han usado, además de los ya descritos, algunos comandos LaTeX adicionales: con `\centering` se consigue que la tabla, si no ocupa todo el ancho de la página, aparezca centrada; con `small` se ajusta el tipo de letra de la tabla para que sea más pequeño que el texto general y, de esta forma, pueda ajustarse al espacio disponible; por último con `\def\arraystrectch{1.5}` se modifica la separación por defecto de las líneas de la tabla, que es `1.0`, incrementándola a `1.5`.

Es habitual que las tablas y figuras, cuando su tamaño es mayor que el espacio que resta en la página, pasen automáticamente al inicio de la página siguiente. LaTeXsiempre tratará de ajustar los contenidos para conseguir el mejor resultado posible, de forma que el texto quede ajustado llenando las páginas. Este párrafo de texto, por ejemplo, está dispuesto tras la Tabla 1.2, pero en el documento aparece dividido rodeando a la tabla, de forma que la página anterior no quede con un espacio vacío

| Componente | Especificación | Descripción |
|---|---|---|
| Procesador | Arquitec. x64 | Procesador de última generación con múltiples núcleos (*cores*) de ejecución para poder ejecutar los distintos contenedores que forman el sistema |
| RAM | 64GB | Disponer de suficiente memoria RAM es importante para garantizar un funcionamiento correcto del sistema |
| Disco | 10TB | Almacenamiento masivo suficiente para contener las imágenes, vídeos, modelos de IA y resto de elementos del sistema. Se recomienda una velocidad de al menos 7200rpm |
| GPU | CCC $\geq$ 3.5 | GPU para la ejecución de los modelos de IA con núcleos CUDA Compute Capability versión 3.5 o posterior y al menos 8GB de memoria |
| Red | GbE | Conexión Gigabit Ethernet para el acceso del sistema a fuentes de datos externas y la compartición de contenidos |

Tabla. 1.2: Requisitos hardware del ordenador en el que se instalará el sistema

en la parte inferior. Puedes probar a mover párrafos de texto, poniéndolos delante o detrás del punto en el que se introduce la tabla/figura, para conseguir la disposición que te interese.

## 1.13. Cómo insertar fórmulas/ecuaciones

LaTeX se caracteriza por su versatilidad a la hora de tratar con símbolos, fórmulas y ecuaciones matemáticas, consiguiendo unos resultados de calidad superior. Esencialmente nos encontraremos con dos escenarios de uso distintos:

- La introducción de expresiones matemáticas en línea con el texto existente en un párrafo. En este caso se usarán el delimitador `$` al inicio y final de la expresión, usando en su interior los comandos específicos de LaTeXpara esta tarea. Por ejemplo, con la expresión `$f(n) = n^5 + 4n^2 + 2$` conseguimos el resultado $f(n) = n^5 + 4n^2 + 2$.
- La introducción de ecuaciones de manera independiente, en cuyo caso deben ir numeradas y se les asignará una etiqueta para poder hacer referencia a ellas desde el texto. Con este fin se usará el ámbito `equation`. La numeración se generará automáticamente y aparecerá en el margen derecho, entre paréntesis.

El código mostrado en el Listado 1.10 genera como resultado la Ecuación 1.7.

Observa el uso del comando `\label` para asignar una etiqueta que se ha usado en este párrafo de texto para hacer referencia a ella con el comando `\ref`.

```
\begin{equation}
    P\left(A=2\middle|\frac{A^2}{B}>4\right)
    \label{Eq.Prob1}
\end{equation}
```

Listado 1.10: Definición de una ecuación

$$P\left(A=2\middle|\frac{A^2}{B}>4\right) \tag{1.7}$$

Sobre el lenguaje que permite generar las expresiones matemáticas se podría escribir un libro completo. En WikiBooks (en.wikibooks.org/wiki/LaTeX/Mathematics) puedes encontrar una introducción que te solventará la mayoría de dudas al respecto.

## 1.14. Cómo insertar algoritmos

En una memoria de TFG es habitual tener que describir cómo funcionan los algoritmos usados o propuestos. Con ese fin lo más recomendable es facilitar un pseudocódigo que describa dichos algoritmos. Es deseable que todos los algoritmos se describan con una sintaxis y formato similares, de forma que sean consistentes a lo largo de toda la memoria.

Por esa razón se recomienda usar el entorno `algorithm` a la hora de describir algoritmos. En este entorno usaremos los comandos `\caption` y `\label`, como haríamos con una figura o una tabla, y aparte un conjunto de comandos específicos para la definición de condicionales, bucles, etc., tal y como se aprecia en el Algoritmo 1. Este ha sido generado con el código LaTeX que aparece en el Listado 1.11.

**Algoritmo 1:** Título del algoritmo descrito

**Result:** Resultado que devuelve algoritmo
inicialización;
**while** *condición* **do**
    instrucciones;
    **if** *condición* **then**
        instrucciones1;
        instrucciones2;
    **else**
        instrucciones3;
    **end**
**end**

Observa que el título del algoritmo aparece en la parte superior, en lugar de en la inferior como ocurre con imágenes, tablas y otros elementos. Al igual que en las tablas, se usan líneas horizontales que delimitan las distintas partes del algoritmo.

```
\begin{algorithm}[H]
\SetAlgoLined
\KwResult{Resultado que devuelve algoritmo}
 inicializacion\;
 \While{condicion}{
  instrucciones\;
  \eIf{condicion}{
   instrucciones1\;
   instrucciones2\;
   }{
   instrucciones3\;
  }
 }
 \caption{Titulo del algoritmo descrito}
\end{algorithm}
```

Listado 1.11: Descripción de un algoritmo

En cuanto a los comandos que pueden emplearse dentro del entorno `algorithm`, encontrarás información de referencia y abundancia de ejemplos en el manual del paquete LaTeX `algorithm2e` (osl.ugr.es/CTAN/macros/latex/contrib/algorithm2e/doc/algorithm2e.pdf), encargado de aportar el citado entorno y sus comandos específicos. En la mayoría de los casos solo precisarás usar un pequeño subconjunto de esos comandos para describir el algoritmo.

## 1.15. Cómo insertar código

Además de algoritmos descritos desde una perspectiva abstracta, mediante pseudo-código, también es probable que en la memoria sea preciso introducir listados de código. En las secciones previas aparecen múltiples listados, todos ellos han sido generados con el entorno `lstlisting`. Este toma, entre paréntesis, tres parámetros:

- `caption`: establece el título que aparecerá bajo el listado
- `language`: indica el lenguaje usado en el listado, de forma que se diferencien con colores y tipos de letra las distintas partes
- `label`: etiqueta a asignar para referenciar el listado

En el Listado 1.12 se muestra el fuente LaTeX usado para producir el Listado 1.13. En el primero faltaría cerrar el entorno con el comando `\end{lstlisting}`.

```
\begin{lstlisting}[caption={Crear una tabla},language=SQL,label=List.CreaTabla]
CREATE TABLE Entradas(
    id INTEGER PRIMARY KEY,
    asiento INTEGER,
    cliente CHAR(50)
);
```

Listado 1.12: Fuente LaTeX que genera el listado en lenguaje SQL

```
CREATE TABLE Entradas(
    id INTEGER PRIMARY KEY,
    asiento INTEGER,
    cliente CHAR(50)
);
```

Listado 1.13: Creación de una tabla con SQL

El paquete que facilita el entorno `lstlisting` contempla el uso de decenas de lenguajes. La lista completa la tenemos en en.wikibooks.org/wiki/LaTeX/Source_Code_Listings, de donde tomaríamos el nombre que es necesario asignar al parámetro `language` del entorno. En esa misma página también se describe cómo definir una configuración a medida para lenguajes que no estén soportados inicialmente. El formato por defecto para esta plantilla está definido en el archivo `Portada.tex`, junto con muchas otras definiciones.

## 1.16. Cómo citar textos y otros materiales

Durante el desarrollo del TFG es habitual recurrir al estudio de diversos materiales, entre los que se incluyen libros, artículos científicos y, por supuesto, páginas web con tutoriales e instrucciones. También durante la redacción de la memoria es posible que se usen definiciones tomadas literalmente o incluso diagramas y figuras.

En todos esos casos es imprescindible citar de manera adecuada la fuente de la que procede la información en que basamos nuestro trabajo o que reproducimos literalmente en la memoria.

### 1.16.1. Citas bibliográficas

Cualquier libro, artículo, contribución a congreso o recurso web que hayamos consultado para poder adquirir los conocimientos necesarios para desarrollar el TFG, ya sean teóricos sobre técnicas o prácticos sobre herramientas, deberíamos citarlos en nuestra memoria. La introducción de estas citas es un aspecto que se descuida en muchas ocasiones y que suele afectar negativamente a la evaluación de una memoria.

La introducción de citas bibliográficas en la memoria se lleva a cabo mediante el comando `\cite{}`, introduciendo en las llaves el identificador que se haya asignado a la referencia bibliográfica en el archivo `bibliografía.bib`. El proceso consta de varios pasos:

1. Obtener los datos bibliográficos en formato BibTeX:
   - En Google Scholar (`scholar.google.es`) hacer clic en el icono en forma de comillas que aparece debajo del título del libro/artículo, a continuación elegir la opción **BibTeX**, seleccionar todo y copiar al portapapeles.
   - La revista/editorial en que se haya publicado el trabajo suele ofrecer la información bibliográfica en formato BibTeX.
   - Crear manualmente la entrada BibTeX con los datos bibliográficos que hayamos obtenido de la fuente de la que procede el material.
2. Introducir los datos bibliográficos en el archivo `bibliografía.bib`, asignando en la primera línea un identificador que nos sea fácil recordar por el tema y/o autor del material.

3. Insertar en la ubicación que proceda de nuestra memoria la referencia bibliográfica, usando para ello el comando `\cite{}` y el identificador que se haya asignado en el paso previo.

Las entradas BibTeX almacenadas en el archivo `bibliografía.bib` tendrán un formato u otro dependiendo de que el material a citar sea un artículo científico, un libro, una contribución a un congreso o un recurso web. Cada entrada BibTeX se compone de múltiples atributos con datos:

- Por regla general, toda entrada deberá contar al menos con los atributos `title`, `author` y `year`, con los que se facilita el título del trabajo, su autor y año de publicación.
- En el caso de los libros también se debe incluir la editorial e ISBN, usando para ello los atributos `publisher` e `isbn`, respectivamente.
- Para los artículos indicar el volumen, número y páginas que ocupa en la revista, así como el nombre de esta, usando los atributos `volume`, `number`, `pages` y `journal`, respectivamente.
- En el caso de los recursos web es indispensable facilitar en el campo `url` la dirección web, así como indicar en el atributo `note` la fecha en que se verificó la disponibilidad del recurso.
- Siempre que sea posible, usar el atributo `doi` para facilitar el DOI (*Digital Object Identifier*) del documento.

Debemos tener presente que algunos servicios, como el mencionado Google Scholar, no facilita información completa en las entradas BibTeX que generan automáticamente. Por ello siempre debemos revisarlas y, en caso necesario, completar los datos que falten, buscándolos en el editor del libro, página web de la revista, etc.

A modo de ejemplo, en el Listado 1.14 se facilitan cuatro referencias bibliográficas, cada una de ellas correspondiente a uno de los tipos de entradas BibTeX mencionados. Estos ejemplos también los encontrarás en el archivo `bibliografía.bib`, por lo que puedes usarlos como plantilla para crear tus propias entradas en caso necesario.

```
1 @article{UnArticulo,
2   title={A Comprehensive and Didactic Review on Multilabel Learning Software Tools},
3   author={Charte, Francisco},
4   journal={IEEE Access},
5   volume={8},
6   pages={50330--50354},
```

```
  year={2020},
  publisher={IEEE},
  doi={10.1109/ACCESS.2020.2979787}
}

@book{UnLibro,
  title={Multilabel Classification: Problem Analysis, Metrics and Techniques},
  author={Herrera, Francisco and Charte, Francisco and Rivera, Antonio J and del Jesus,
      Mar{\'\i}a J},
  year={2016},
  publisher={Springer},
  ISBN={978-3-319-41110-1},
  doi={10.1007/978-3-319-41111-8}
}

@inproceedings{UnCongreso,
  title={A first approach to deal with imbalance in multi-label datasets},
  author={Charte, Francisco and Rivera, Antonio and del Jesus, Mar{\'\i}a Jos{\'e} and
      Herrera, Francisco},
  booktitle={International Conference on Hybrid Artificial Intelligence Systems},
  pages={150--160},
  year={2013},
  organization={Springer},
  doi={10.1007/978-3-642-40846-5\_16}
}

@online{UnaWeb,
  author = {Charte, Francisco},
  title = {Cómo analizar la distribución de los datos con R},
  year = 2019,
  url = {https://fcharte.com/tutoriales/20170114-DistribucionDatosR/},
  note = {comprobado en 2020-09-30}
}
```

Listado 1.14: Entradas BibTex de distintos tipos

Teniendo las entradas BibTeX ya preparadas, solo quedaría introducir las referencias en los puntos adecuados del texto, tal y como se explicó antes. Un párrafo como el mostrado en el Listado 1.15 generaría el resultado que se aprecia justo a continuación. Además, introducirá la información bibliográfica completa en la lista de referencias, al final de la memoria, con un formato adecuado.

```
Como se explica en~\cite{UnArticulo} y se amplía en el libro~\cite{UnLibro}, las técnicas
que se presentaron en~\cite{UnCongreso} pueden ponerse en práctica siguiendo el
tutorial~\cite{UnaWeb}
```

Listado 1.15: Párrafo en el que se citan varios trabajos

Como se explica en [10] y se amplía en el libro [11], las técnicas que se presentaron en [12] pueden ponerse en práctica siguiendo el tutorial [13]

### 1.16.2. Citas textuales

Si además de emplear un material como base para el estudio y desarrollo de nuestro TFG, caso en el que lo citaremos según lo que acaba de explicarse, también hemos tomado alguna parte del mismo para introducirlo literalmente en la memoria, hemos de tener en cuenta las siguientes normas:

- Únicamente pueden citarse textualmente pequeñas porciones de un trabajo, habitualmente no más de un párrafo y en ningún caso páginas completas.

- El texto citado ha de aparecer claramente diferenciado en el texto de la memoria, para lo cual usaremos habitualmente el entorno `\displayquote` y pondremos el texto en cursiva, consiguiendo un resultado como el siguiente:

  > *Learning, like intelligence, covers such a broad range of processes that it is difficult to define precisely. A dictionary definition includes phrases such as "to gain knowledge, or understanding of, or skill in, by study, instruction, or experience," and "modification of a behavioral tendency by experience." Zoologists and psychologists study learning in animals and humans. In this book we focus on learning in machines.*

- La cita textual ha de ir acompañada de la correspondiente cita bibliográfica, de forma que sea posible determinar el origen del material de manera inequívoca.

- En general no es recomendable incluir en la memoria ninguna ilustración, diagrama o figura cuyos derechos de uso no estén claros. Si la licencia de uso es *Creative Commons* se puede incluir la figura indicando tanto la licencia como la procedencia. En cualquier otro caso es necesario obtener permiso del propietario para la reproducción del material. La alternativa es construir nuestra propia ilustración o diagrama indicando que se ha tomado como base una fuente existente y facilitando la correspondiente referencia.

Ten siempre presente que tu memoria de TFG va a publicarse y cualquiera tendrá acceso al trabajo que has hecho durante mucho tiempo. También es habitual que la memoria se sometida a herramientas de detección de plagio. Por todo ello, es importante ser honesto y no usar nunca trabajo de terceros sin citarlo apropiadamente.

## 1.17. Cómo introducir ciertos caracteres en LaTeX

Al trabajar con fuentes LaTeX empleamos un conjunto de caracteres que tienen un significado especial, como es el carácter \ para con el que se inician los comandos, el carácter $ que delimita las expresiones matemáticas, el carácter % para introducir comentarios, los símbolos _ y ^ para escribir subíndices y superíndices en fórmulas matemáticas, etc. Incluso algunos símbolos, como es el caso de las çomillas dobles", pueden tener un resultado no deseado[6] en el documento.

En general, usaremos el símbolo \ como carácter de escape, para que el carácter que dispongamos a continuación aparezca como tal en el texto. No obstante hay excepciones: para incluir una barra invertida no podemos usar \\, porque eso indica un salto de línea.

Si tienes problemas para introducir algún carácter puedes recurrir a en.wikibooks.org/wiki/LaTeX/Special_Characters para saber cómo debes introducirlo en tu memoria.

## 1.18. Cómo obtener esta plantilla

## 1.19. Cómo configurar la plantilla con nuestros datos

A fin de generar nuestra memoria de TFG usando esta plantilla, aprovechando toda la configuración de formatos ya adaptada a la normativa de la EPSJ, tendremos que comenzar abriendo el archivo `main.tex` para configurar nuestros datos personales. Para ello localiza el bloque LaTeX que aparece en el Listado 1.16 y sigue las instrucciones indicadas en los comentarios.

```
% ==== Introducir aquí el nombre del estudiante
\def\Estudiante{Nombre1 Nombre2 Apellido1 Apellido2}

% ==== Introducir aquí el nombre de los tutores. Si solo hay uno dejar las llaves de \TutorB vacías
\def\TutorA{Nombre y apellidos del tutor 1}
\def\TutorB{Nombre y apellidos del tutor 2}

% ==== Introducir aquí el título de completo y abreviado (para las cabeceras) del TFG
\def\TituloTFG{El título del TFG tal y como aparece en la propuesta original}
\def\TituloAbreviado{Título abreviado para el encabezado}
```

[6] En el fuente LaTeX se ha usado el carácter " para iniciar el entrecomillado, pero esto ha tenido como efecto convertir la "c" inicial en una ç.

```
11
12 % ==== Introducir aquí el mes y año de presentación del TFG
13 \def\Fecha{junio de 2021}
```

Listado 1.16: Variables a establecer en `main.tex`

Solo necesitas establecer el contenido de estas variables para que la portada, página de autorización y los encabezados muestren los datos correctos. Habitualmente esto es todo lo que necesitarás cambiar en el archivo `main.tex`. No obstante, si no quieres incluir al inicio de la memoria alguno de los índices que aparecen por defecto, como los de algoritmos, listados, etc., también deberás eliminar o comentar (colocar delante el carácter %) las líneas correspondientes.

## 1.20. En cuanto al contenido de los capítulos

Una vez que hayas establecido la configuración con tus datos, el paso siguiente será escribir el contenido de la memoria, completando los capítulos que se han previsto en la plantilla. Cada uno de ellos está almacenado en un archivo independiente y se inserta desde `main.tex` usando el comando `\input`.

Teniendo presente que la estructura de esta memoria se corresponde con la de un TFG de tipo experimental, los capítulos y su contenido son los siguientes:

1. **Introducción** - `Introduccion.tex`[7]

   La introducción al TFG, como su propio nombre denota, ha de servir como un acercamiento general y desde una perspectiva de alto nivel al trabajo realizado. Debe explicarse la motivación que nos ha llevado a abordar el problema en cuestión, su importancia, cómo se ha tratado hasta le momento, qué creemos que podemos aportar, etc., todo ello sin entrar demasiado en detalle. Es habitual indicar en la parte final de la introducción la estructura de la memoria, enumerando sus capítulos con una breve descripción de lo que se explica en cada uno de ellos. También puedes incluir un índice o tabla, al final de este capítulo, conteniendo todos los acrónimos y términos que uses en la memoria, de forma que sea fácil para quien la lea saber qué significan[8].

[7] En la plantilla este archivo contiene las instrucciones de cómo usar LaTeX y la propia plantilla, por lo que deberás eliminar dicho contenido o bien crear tu propio archivo `Introduccion.tex` desde cero.

[8] La alternativa es definirlos la primera vez que aparezcan en el texto, por ejemplo: «En este TFG (*Trabajo Fin de Grado*) se aborda ...».

2. **Antecedentes** - `Antecedentes.tex`

   Este capítulo tiene como objetivo demostrar que has estudiado el tema que se aborda en el TFG, describiendo primero el problema con todos los detalles necesarios y después las técnicas que se han empleado hasta el momento para tratarlo y las que propones usar tú. Un aspecto fundamental de este capítulo es la bibliografía: has de recopilar los artículos científicos más relevantes sobre el tema en cuestión, consultar libros sobre las técnicas a usar, etc., referenciando todo ello de manera adecuada.

3. **Objetivos** - `Objetivos.tex`

   El título del capítulo lo dice todo: ¿cuáles son los objetivos que se persiguen al desarrollar este TFG? Habitualmente se expondrá un objetivo general, desde una perspectiva de alto nivel, y después se descompondrá en tantos objetivos específicos como sea preciso, detallando cada uno de ellos al máximo. En teoría, solo leyendo este capítulo debería obtenerse una idea muy clara de qué va a hacerse (se ha hecho) en el TFG.

4. **Materiales y métodos** - `MaterialMetodos.tex`

   Para alcanzar los objetivos establecidos en el capítulo previo será preciso usar unos materiales: por ejemplo los datos que servirán para generar los modelos, y también unos métodos: las técnicas y herramientas que servirán para obtener los resultados. Todos esos aspectos han de quedar reflejados en este capítulo hasta el más mínimo detalle, explicando, por ejemplo, cómo se han obtenido los datos, si ha sido necesario prepararlos de alguna manera antes de poder usarlos; qué técnicas/algoritmos van a emplearse y por qué, cómo va a realizarse la ejecución (configuración hardware y software), etc.

5. **Resultados** - `Resultados.tex`

   Como salida de aplicar los métodos a los materiales se obtienen resultados. La finalidad de este capítulo doble: por una parte se deben reflejar todos los resultados obtenidos, usualmente en tablas en gráficas, por otra, ha de efectuarse un análisis pormenorizado de dichos resultados a fin de determinar su valor. Siempre que sea posible, deberían realizarse comparaciones de resultados entre distintos métodos, ya los hayamos aplicado nosotros o hayan sido previamente publicados en la bibliografía citada en los antecedentes.

6. **Conclusiones** - `Conclusiones.tex`

   El último capítulo de la memoria ha de presentar una recopilación de todo el trabajo realizado, los resultados obtenidos y un análisis que determine, y deje claro, si se han alcanzado las metas que se establecieron en la propuesta inicial

del TFG que aprobó la correspondiente comisión. Además, también es habitual incluir una sección describiendo potenciales mejoras que podrían efectuarse o trabajos futuros que podrían derivarse de lo hecho en el TFG.

Además de los archivos anteriores, correspondientes a los capítulos, también debes editar el contenido del archivo `Agradecimientos.tex` para escribir tus propios agradecimientos y dedicatoria.

## 1.21. Consejos sobre redacción

Para terminar estas instrucciones, y aunque no tiene que ver con la plantilla LaTeX ni aspectos relativos a la normativa de realización de TFG de la EPSJ, quiero incidir en un aspecto de vital importancia: la calidad de la redacción.

Tanto para el tribunal que evaluará tu TFG como para las personas que puedan leerlo en el futuro, la corrección del texto será siempre un reflejo de la calidad del trabajo que has llevado a cabo, aunque esta no sea la realidad. Es habitual que un muy buen trabajo vea su evaluación penalizada a causa de una pobre redacción de la memoria. Por el contrario, un trabajo menos brillante pero con una presentación perfecta en la memoria puede ser mejor valorado.

Sin ánimo de ser exhaustivo, porque tampoco es la finalidad de estas instrucciones, he aquí una lista de aspectos a tener presentes en la redacción:

- **Ortografía:** posiblemente no haya nada que cause una peor impresión al leer una memoria que encontrar errores de ortografía. Extrema el cuidado en este sentido: revisa las normas de uso de $b$ y $v$, de $g$ y $j$, de $m$ y $n$, de uso de la $h$, las relativas a las tildes, etc. Tan malo es el defecto como el exceso, por ejemplo poniendo tilde en términos como los pronombres ($este$, $esta$, $estas$) que la normativa indica desde hace años que no deben llevar tilde. Ante la duda usa el Diccionario de la RAE (`buscon.rae.es`) y las consultas de la Fundéu (`www.fundeu.es/consultas`).
- **Gramática:** usar los tiempos verbales correctos, respetar las concordancias de género y número y otros aspectos gramaticales es también vital para facilitar la correcta comprensión de las explicaciones dadas en la memoria. Una de estas normas nos dice que a los acrónimos y abreviaciones no se les añade la $s$ final para formar el plural, sino que el número lo denota el determinante o el contexto,

por ejemplo «En los TFG de este curso ...», en lugar de «En los TFGs de este curso ...». Otra regla es la de no comenzar con mayúsculas tras dos puntos, salvo para nombres propios, tal y como ves en esta misma sección, en la que tras el título en negrita y los dos puntos se continúa usando minúsculas.

- **Vocabulario:** el vocabulario usado al redactar denota diversas capacidades por parte del autor de la memoria. En primer lugar, el uso de términos especializados, específicos y exactos, del campo en el que se está trabajando indica que se conoce la materia, se ha leído sobre el tema. En segundo, el uso de un vocabulario variado, evitando repetir las mismas palabras y términos en un corto espacio, indica riqueza en el uso del lenguaje. Por último, debe tenerse presente que la memoria de un TFG es un documento formal y que, en consecuencia, debe evitarse el uso de términos demasiado coloquiales. Por ejemplo, usar «Según se ha descrito en ...» en lugar de «Como se ha comentado en ...».

- **Estructuración:** la memoria al completo, pero también cada uno de sus capítulos de manera aislada y cada sección de cada capítulo, deben redactarse como una relación de ideas interconectadas entre sí y que tienen sentido. Nada hay más desconcertante que leer una memoria en la que cada párrafo trata sobre ideas inconexas con lo que tiene a su alrededor, en párrafos anteriores y posteriores, provocando una sensación de que va saltándose de un tema a otro sin orden alguno. Por ello es importante llevar a cabo una planificación previa del contenido de cada capítulo y de cada sección, de forma que, al igual que durante el desarrollo de un software, se maximice la cohesión y se minimice el acoplamiento.

Aunque es algo totalmente fuera del ámbito de estas instrucciones, una última recomendación: la mejor forma de aprender a redactar correctamente estriba en leer mucho, especialmente literatura.

# Capítulo 2

# Antecedentes

## 2.1. Introducción

El presente capítulo tiene como objetivo establecer el marco teórico y contextual necesario para comprender el alcance y la relevancia del Trabajo de Fin de Grado. Se abordarán los fundamentos de los modelos de clasificación de Machine Learning que sustentan el enfoque propuesto, se realizará un exhaustivo recorrido por el estado del arte en sistemas de detección de intrusiones (IDS) y se presentarán trabajos relacionados clave que han contribuido al desarrollo de esta disciplina. Finalmente, se analizarán las tecnologías existentes que son relevantes para la implementación de un sistema de detección de anomalías en tráfico de red.

## 2.2. Modelos de Clasificación para Machine Learning

Los modelos de clasificación son un pilar fundamental en el ámbito del Machine Learning, especialmente en aplicaciones donde es necesario categorizar datos en clases predefinidas. En el contexto de la ciberseguridad, estos modelos permiten discernir entre tráfico de red legítimo y malicioso, o incluso identificar el tipo específico de un ataque. A continuación, se describen los modelos de clasificación más relevantes y su aplicabilidad en este dominio.

### 2.2.1. Random Forest

Random Forest [14] es un algoritmo de Machine Learning basado en el concepto de ensemble learning, específicamente mediante el método de bagging (Bootstrap Aggregating). Este modelo construye múltiples árboles de decisión durante la fase de entrenamiento y, para la clasificación, la salida es la clase que más votan los árboles individuales (o el promedio de las predicciones para regresión). Su robustez se debe a la combinación de predicciones de múltiples árboles, lo que reduce la varianza y el sobreajuste, problemas comunes en árboles de decisión únicos.

Lo que diferencia a este modelo de clasificación de otros de esemble es el hecho de que, para cada árbol de forma individual, se eligen una serie de características del conjunto de datos en lguar del conjunto completo de características. A continuación, a partir de los resultados proporcionados por los árboles individuales, se puede determinar que etiqueta o clase es la más apropiada para cada dato.

En el ámbito de los IDS, Random Forest ha demostrado ser particularmente efectivo debido a su capacidad para manejar grandes volúmenes de datos con alta dimensionalidad, su resistencia a características irrelevantes y su habilidad para estimar la importancia de cada característica, lo cual es crucial para la selección de atributos en el tráfico de red. La capacidad de este algoritmo para proporcionar una alta precisión y su relativamente bajo riesgo de sobreajuste lo hacen una opción atractiva para la detección de anomalías y ataques en redes, como se ha evidenciado en la implementación de este proyecto.

Véase en la siguiente ilustración 2.1la arquitectura de clasificación del modelo Random Forest.

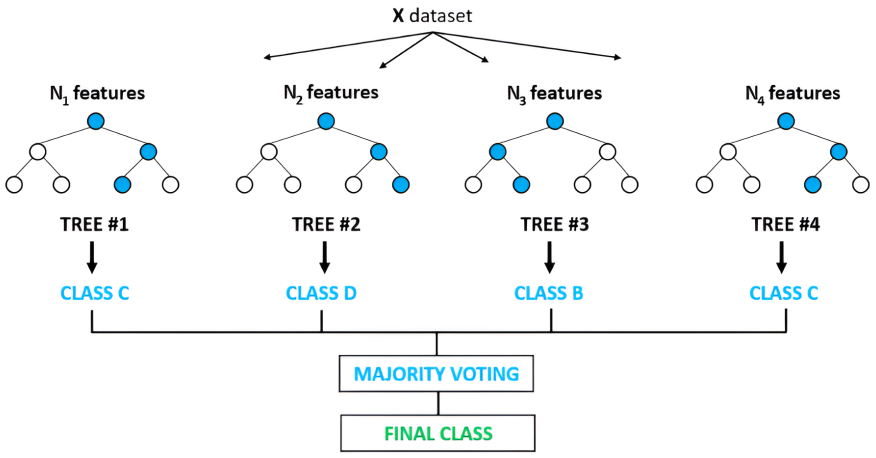


Figura 2.1: Esquema del modelo Random Forest. Fuente: Elaboración propia.

### 2.2.2. Support Vector Machine (SVM)

Las Support Vector Machines (SVM) son modelos de aprendizaje supervisado utilizados tanto para problemas de clasificación como de regresión. El principio fundamental de SVM radica en encontrar el hiperplano óptimo que mejor separe las clases en un espacio de características de alta dimensionalidad. El "óptimo"se refiere al hiperplano que maximiza el margen entre las clases (la distancia entre el hiperplano y los puntos de datos más cercanos de cada clase, conocidos como vectores de soporte).

Las SVM son particularmente potentes en espacios de alta dimensionalidad y son efectivas en casos donde el número de dimensiones es mayor que el número de muestras. Sin embargo, su complejidad computacional puede ser una limitación en datasets muy grandes, como los que se encuentran comúnmente en el análisis de tráfico de red en tiempo real.

### 2.2.3. Naive Bayes

El clasificador Naive Bayes es una familia de algoritmos de clasificación probabilística simple basada en el Teorema de Bayes con una ïngenua"suposición de independencia fuerte entre las características. A pesar de su simplicidad y de la suposición de independencia (que a menudo no se cumple en la realidad), los clasificadores Naive Bayes han demostrado un rendimiento sorprendentemente bueno en muchas aplicaciones prácticas, especialmente en el procesamiento de lenguaje natural y la clasificación de texto.

Para la detección de intrusiones, su simplicidad y velocidad de entrenamiento lo hacen adecuado para conjuntos de datos grandes y para escenarios donde la eficiencia computacional es crítica. No obstante, la suposición de independencia entre las características de red (como puertos, protocolos, tamaños de paquete) puede limitar su precisión en comparación con modelos más complejos que capturan las interacciones entre ellas.

### 2.2.4. K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) es un algoritmo de clasificación no paramétrico y basado en instancias. Funciona clasificando nuevos puntos de datos basándose en la mayoría de votos de sus "K"vecinos más cercanos en el espacio de características. La

distancia entre los puntos de datos (ej. distancia euclidiana) es crucial para determinar la çercanía"de los vecinos.

KNN es simple de entender e implementar y no requiere una fase de entrenamiento explícita, ya que memoriza el conjunto de datos de entrenamiento. Sin embargo, su principal desventaja en el contexto de IDS es la alta complejidad computacional durante la fase de predicción, ya que necesita calcular las distancias a todos los puntos de entrenamiento para cada nueva instancia. Esto lo hace menos adecuado para sistemas en tiempo real que manejan un flujo constante y elevado de tráfico de red.

## 2.3. Estado del Arte en Sistemas de Detección de Intrusiones (IDS)

La ciberseguridad es un campo en constante evolución, y los Sistemas de Detección de Intrusiones (IDS) son componentes críticos para salvaguardar la integridad, confidencialidad y disponibilidad de los sistemas de información. El estado del arte actual se caracteriza por una convergencia creciente entre las metodologías tradicionales de detección y las capacidades emergentes del Machine Learning y la Inteligencia Artificial.

### 2.3.1. Evolución de los IDS: Desde Firmas a Anomalías

Tradicionalmente, los IDS se han clasificado en dos categorías principales: basados en firmas (Signature-based IDS - SIDS) y basados en anomalías (Anomaly-based IDS - AIDS). Los SIDS detectan intrusiones buscando patrones o firmas predefinidos de ataques conocidos en el tráfico de red o en los registros del sistema. Si bien son muy efectivos para identificar ataques ya conocidos, su principal limitación es la incapacidad de detectar nuevas amenazas (ataques de día cero).

En contraste, los AIDS se centran en identificar desviaciones del comportamiento normal o esperado del sistema o la red. Para ello, construyen un perfil de "normalidadz cualquier actividad que se desvíe significativamente de este perfil se marca como una anomalía, y potencialmente como una intrusión. La emergencia del Machine Learning ha potenciado enormemente la capacidad de los AIDS para aprender patrones complejos de tráfico normal y detectar actividades maliciosas novedosas, superando las limitaciones de los SIDS.

### 2.3.2. IDS Basados en Machine Learning: Taxonomía y Enfoques

La aplicación de Machine Learning en IDS ha revolucionado la forma en que se detectan las amenazas. Los enfoques de IDS basados en ML pueden clasificarse en:

- **Supervisados:** Utilizan conjuntos de datos etiquetados (tráfico normal y de ataque) para entrenar modelos que aprenden a clasificar nuevas instancias. Modelos como Random Forest, SVM o Redes Neuronales son comunes aquí.
- **No Supervisados:** Emplean algoritmos para descubrir patrones y estructuras ocultas en datos no etiquetados, identificando como anomalías aquellas instancias que no se ajustan a estos patrones. La detección de outliers es un ejemplo.
- **Semi-supervisados:** Combinan un pequeño conjunto de datos etiquetados con una gran cantidad de datos no etiquetados para el entrenamiento, siendo útil cuando la anotación de datos es costosa.
- **Basados en Redes Profundas (Deep Learning):** Representan una evolución de los modelos de ML, utilizando arquitecturas de redes neuronales con múltiples capas. Modelos como las Redes Neuronales Convolucionales (CNN) son eficientes en la extracción de características jerárquicas de datos secuenciales, y las Redes Neuronales Recurrentes (RNN), especialmente las variantes como LSTM o GRU, son idóneas para el análisis de series temporales de tráfico de red. Recientemente, los modelos basados en Transformers también han mostrado un gran potencial en el análisis de secuencias de paquetes.

Los datasets utilizados para el entrenamiento y evaluación de estos modelos son cruciales, siendo algunos de los más reconocidos CIC-IDS2017, CIC-CSE-IDS2018 y KDD Cup 1999.

### 2.3.3. Detección de Ataques Específicos Mediante IA

La capacidad del Machine Learning permite ir más allá de la simple detección de intrusiones, facilitando la identificación de tipos específicos de ataques con un alto grado de precisión. Por ejemplo, los modelos de clasificación pueden ser entrenados para reconocer patrones asociados con:

- **Ataques de Denegación de Servicio Distribuido (DDoS):** Identificando volúmenes anómalos de tráfico o patrones de conexión coordinados.

- **Escaneos de Puertos (Port Scan):** Detectando intentos de sondear puertos abiertos en un sistema.
- **Inyecciones SQL (SQL Injection):** Analizando el contenido de las peticiones web en busca de secuencias de comandos maliciosas.

Estas detecciones específicas, como las que se presentan en interfaces de dashboard, son vitales para la respuesta a incidentes y la implementación de contramedidas adecuadas.

### 2.3.4. Desafíos y Futuro en la Detección de Intrusiones con IA

A pesar de los avances significativos, la aplicación de IA en IDS enfrenta desafíos importantes. El "desplazamiento de concepto"(concept drift), donde los patrones de ataque evolucionan y los modelos previamente entrenados pueden volverse obsoletos, exige la implementación de mecanismos de re-entrenamiento continuo. El desbalance de clases en los datasets (donde el tráfico normal es significativamente más abundante que el tráfico malicioso) es otro obstáculo que puede llevar a modelos sesgados y con baja capacidad para detectar las clases minoritarias. Además, la interpretabilidad de los modelos complejos de Deep Learning es un área activa de investigación, ya que comprender por qué un modelo clasifica una actividad como maliciosa es crucial para los analistas de seguridad en la toma de decisiones. El futuro de los IDS con IA se dirige hacia el desarrollo de sistemas más adaptativos, proactivos y explicables, capaces de operar eficazmente en entornos de alta velocidad y volumen, integrando aprendizaje continuo y capacidades de respuesta automatizada.

## 2.4. Trabajos Relacionados

La literatura científica y técnica ofrece un vasto cuerpo de trabajos relacionados que han explorado la aplicación de técnicas de Machine Learning para la detección de intrusiones en redes. Muchos estudios se centran en la experimentación con diversos modelos de clasificación sobre datasets de referencia como CIC-IDS2017 o KDD Cup 1999, evaluando su rendimiento en términos de precisión, recall y F1-score para identificar diferentes categorías de ataques.

Algunas investigaciones se han enfocado en la selección de características (feature selection) para optimizar el rendimiento de los modelos, buscando los atributos más

relevantes del tráfico de red que permiten una detección eficaz y reducen la complejidad computacional. Otros trabajos han abordado la problemática del desbalance de clases en los datasets de ciberseguridad, proponiendo técnicas de sobremuestreo o submuestreo para mejorar la capacidad de detección de ataques minoritarios.

La integración de capturadores de tráfico en tiempo real con módulos de Machine Learning es también un área de investigación activa, buscando cerrar la brecha entre la detección offline y la capacidad de respuesta inmediata. Proyectos similares al presente, que buscan desarrollar sistemas capaces de procesar flujos de paquetes en tiempo real y aplicar algoritmos de clasificación para identificar anomalías, han allanado el camino para soluciones más proactivas en ciberseguridad.

## 2.5. Tecnologías Existentes

El desarrollo de un sistema de detección de intrusiones basado en Machine Learning se apoya en un ecosistema de tecnologías y herramientas existentes que facilitan la captura de datos, su procesamiento, el entrenamiento de modelos y la visualización de resultados.

En el ámbito de la captura y análisis de tráfico de red, herramientas como Wireshark o tcpdump son estándares de la industria que permiten la interceptación y el análisis profundo de paquetes. Wireshark, en particular, ofrece capacidades de descifrado y una interfaz gráfica detallada para la inspección manual del tráfico.

Para el desarrollo de los algoritmos de Machine Learning, Python se ha consolidado como el lenguaje de programación por excelencia en ciencia de datos, gracias a su rica colección de librerías. Scikit-learn proporciona una implementación robusta de una amplia gama de algoritmos de clasificación, incluyendo Random Forest, SVM, Naive Bayes y KNN. Para enfoques de Deep Learning, TensorFlow y PyTorch son los frameworks dominantes, ofreciendo flexibilidad y eficiencia computacional para la construcción y entrenamiento de redes neuronales complejas.

El procesamiento y manipulación de grandes volúmenes de datos de red se beneficia enormemente de librerías como Pandas para la gestión de DataFrames y NumPy para operaciones numéricas. La visualización de los datos y los resultados del sistema, como un dashboard de alertas, se puede lograr con frameworks de desarrollo web como Flask o Django en Python, combinados con librerías de visualización front-end.

Finalmente, el uso de entornos de desarrollo integrado (IDE) como VS Code o

PyCharm, junto con sistemas de control de versiones como Git y plataformas como GitHub, son tecnologías esenciales que aseguran un desarrollo colaborativo, organizado y eficiente del proyecto.

# Capítulo 3

# MATERIALES Y MÉTODOS

## 3.1. Materiales empleados

### 3.1.1. Hardware y entorno de desarrollo

elección del modelo de captura de paquetes - evaluar distintos programar y explicar porque he elegido uno - módulo wireshark - módulo tcpdump - módulo nmap - implementación en python

### 3.1.2. Software y herramientas

### 3.1.3. Conjuntos de datos

## 3.2. Métodos de adquisición y preprocesamiento de datos

## 3.3. Métodos para la generación de datasets

## 3.4. Métodos del componente de detección de intrusiones

### 3.4.1. Selección y justificación de modelo de machine learning

### 3.4.2. Proceso de entrenamiento y validación del modelo

### 3.4.3. Métodos de integración del modelo

## 3.5. Métodos de diseño e implementación de la interfaz de usuario

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

# Capítulo 4

# ANÁLISIS DEL SISTEMA

El análisis es una etapa crucial en el ciclo de vida del desarrollo de software, ya que establece una base sólida para el diseño y la implementación del sistema. En este apartado, se detallará la fase de análisis del proyecto, que incluye la identificación y especificación de los requisitos funcionales y no funcionales, la creación de de los correspondientes diagramas de casos de uso, la especificación gramatical de cada uso y por último, la elaboración del diagrama de secuencia correspondiente.

## 4.1. Requisitos del sistema

Los requisitos del sistema describen las funciones, características y limitaciones del sistema de detección de intrusiones basado en Machine Learning. Estos se han dividido en dos categorías principales: requisitos funcionales y requisitos no funcionales.

### 4.1.1. Requisitos funcionales

Los requisitos funcionales definen las funcionalidades que el sistema debe proporcionar para cumplir con los objetivos del proyecto. Se presentan en la siguiente tabla 4.1.

Tabla. 4.1: Requisitos funcionales del sistema

| ID | Requisito Funcional | Descripción |
|---|---|---|
| RF1 | Detección de anomalías en tiempo real | El sistema debe ser capaz de analizar el tráfico de red en tiempo real para identificar patrones anómalos que puedan indicar una intrusión. |
| RF2 | Clasificación de ataques específicos | El sistema debe clasificar los ataques detectados en categorías específicas, como escaneo de puertos (Port Scan), denegación de servicio distribuido (DDoS) e inyección SQL (SQL Injection). |
| RF3 | Visualización de tráfico de red | El sistema debe mostrar una representación visual y detallada del tráfico de red, incluyendo información de los paquetes, protocolos y conexiones para su inspección. |
| RF4 | Generación de alertas de seguridad | El sistema debe generar alertas cuando se detecte una actividad maliciosa, indicando el tipo de ataque, la dirección IP de origen y destino, y su nivel de criticidad (baja, media, crítica). |
| RF5 | Filtrado del tráfico de red | El sistema debe permitir al usuario aplicar filtros de captura de tráfico, como el puerto o protocolo, para monitorear segmentos específicos de la red. |
| RF6 | Persistencia de datos | El sistema debe almacenar un registro histórico del tráfico de red y las alertas generadas para su posterior análisis forense. |
| RF7 | Gestión del modelo de ML | El sistema debe cargar el modelo de Random Forest entrenado para su uso en la detección en tiempo real. |
| RF8 | Interfaz de usuario (Dashboard) | El sistema debe ofrecer una interfaz de usuario visual (dashboard) para mostrar de manera clara y concisa el estado general de la red, los flujos de tráfico y las alertas de seguridad activas. |
| RF9 | Limpieza de los flujos de paquetes | El sistema debe permitir la limpieza del panel del tráfico de red con un botón que tenga esa funcionalidad. |

Continuación en la página siguiente

Tabla. 4.1 – continuación

| ID | Requisito Funcional | Descripción |
|---|---|---|
| RF10 | Inicio de la ejecución del capturador | El sistema debe permitir la incoporación de la funcionalidad del inicio del capturador que se reflejará en los paneles con un botón predeterminado. |
| RF11 | Parada de la ejecución del capturador | El sistema debe permitir la incoporación de la funcionalidad de la parada del capturador que se reflejará en los paneles con un botón predeterminado. |
| RF12 | Soporte para múltiples interfaces de red | El sistema debe permitir la selección de una o más interfaces de red para la captura del tráfico. |
| RF13 | Generación de bases de datos | El sistema debe permitir al usuario generar un conjunto de salida en formato .csv o .txt con la información del monitoreo del tráfico de red en una interfaz dada en cualquier momento. |

### 4.1.2. Requisitos no funcionales

Los requisitos no funcionales se refieren a las características de calidad del sistema, como la usabilidad, la fiabilidad y el rendimiento.En este subapartado, se detallan estos requisitos, proporcionando una visión clara de los estándares de calidad que se espera que la aplicación cumpla. A continuación, se detallan en la siguiente tabla 4.2.

Tabla. 4.2: Requisitos no funcionales del sistema

| ID | Requisito No Funcional | Descripción |
|---|---|---|
| RNF1 | Rendimiento y eficiencia | El sistema debe ser capaz de procesar un alto volumen de tráfico de red con una baja latencia para garantizar la detección en tiempo real. |
| RNF2 | Fiabilidad | El sistema debe operar de manera estable y continua sin fallos inesperados. |

Continuación en la página siguiente

Tabla. 4.2 – continuación

| ID | Requisito No Funcional | Descripción |
|---|---|---|
| RNF3 | Usabilidad | La interfaz de usuario debe ser intuitiva y fácil de usar para los analistas de seguridad, permitiendo una rápida configuración y visualización de la información. |
| RNF4 | Seguridad | El sistema debe garantizar la confidencialidad y la integridad de los datos de tráfico que procesa, evitando fugas de información. |
| RNF5 | Escalabilidad | El sistema debe poder adaptarse para manejar un incremento en el volumen de tráfico y la complejidad de los datos sin una degradación significativa del rendimiento. |
| RNF6 | Mantenibilidad | El código del proyecto debe ser legible, modular y estar bien documentado para facilitar futuras modificaciones y correcciones. |
| RNF7 | Portabilidad | El sistema debe ser compatible con los sistemas operativos más comunes para servidores, como distribuciones de Linux. |
| RNF8 | Disponibilidad | El sistema de detección debe estar disponible y operativo las 24 horas del día, 7 días a la semana, para una monitorización continua de la red. |
| RNF9 | Interoperabilidad | El sistema debe ser capaz de integrar y trabajar con librerías y herramientas externas como Scapy para la captura de paquetes y Scikit-learn para el modelo de Machine Learning. |

### 4.1.3. Diagrama de requisitos y trazabilidad inicial

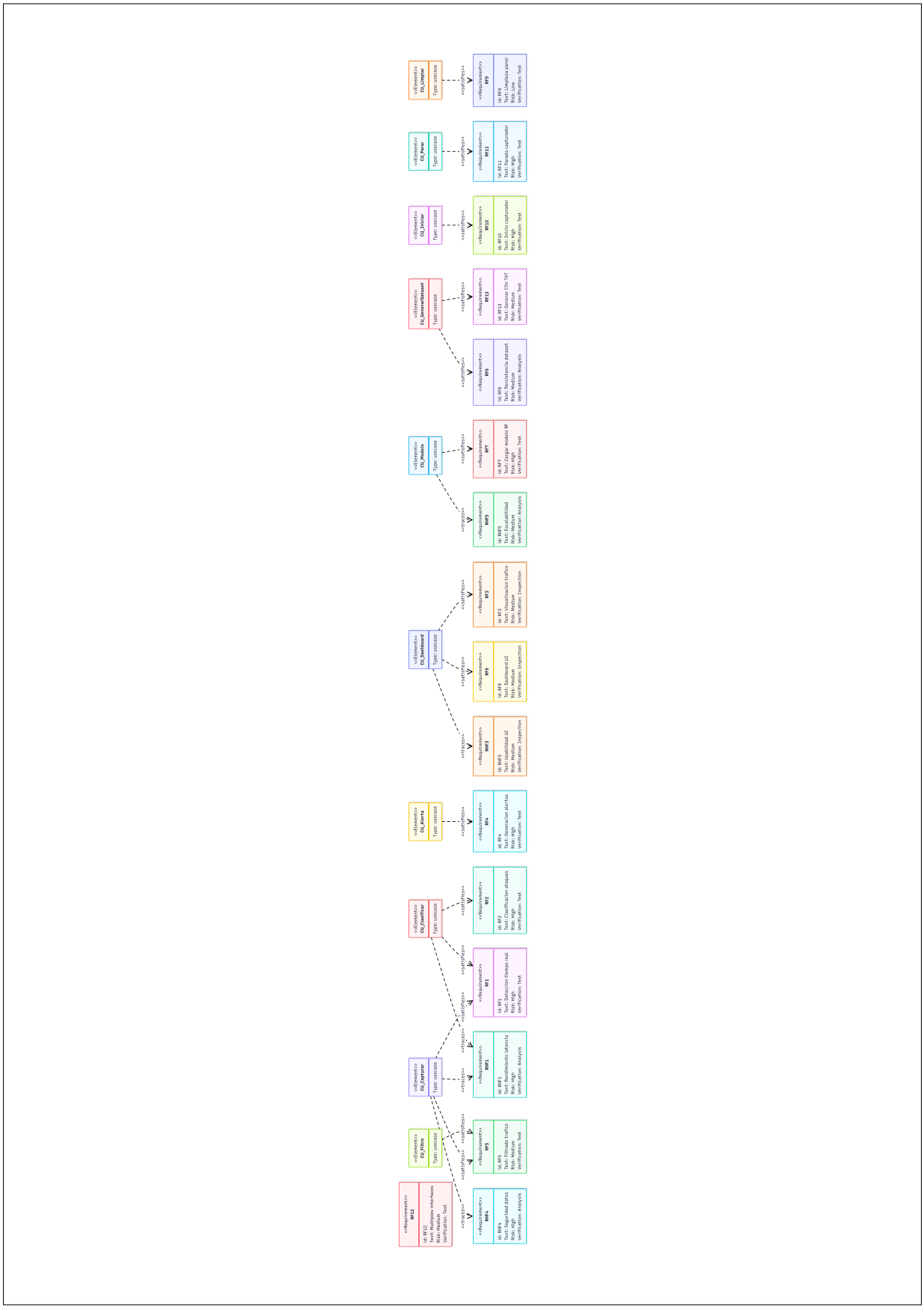

Figura 4.1: Diagrama de secuencia principal.

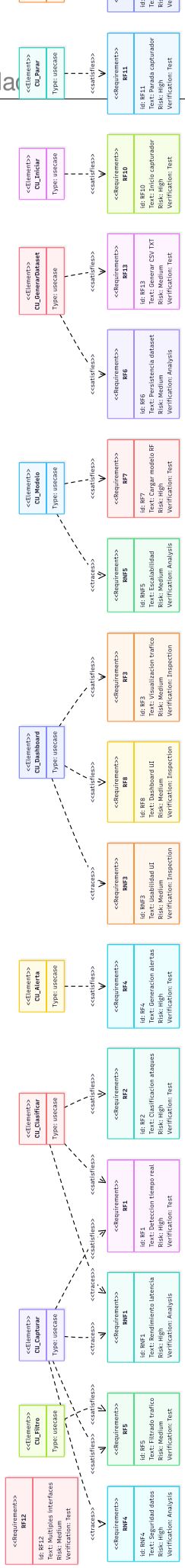


Figura 4.2: Diagrama de requisitos.

<<Element>>
CU_Clasificar
Type:
<<traces>>
<<Element>>
CU_Capturar
Type:
<<traces>>
<<traces>>
<<Element>>
CU_Dashboard
Type:
<<traces>>
<<Element>>
CU_Modelo
Type:
<<traces>>
<<Requirement>>
RNF1
id: RNF1
Text: Rendimiento /
Risk:
<<Requirement>>
RNF4
id: RNF4
Text: Seguridad
Risk:
<<Requirement>>
RNF3
id: RNF3
Text: Usabilidad
Risk:
<<Requirement>>
RNF5
id: RNF5
Text:
Risk:

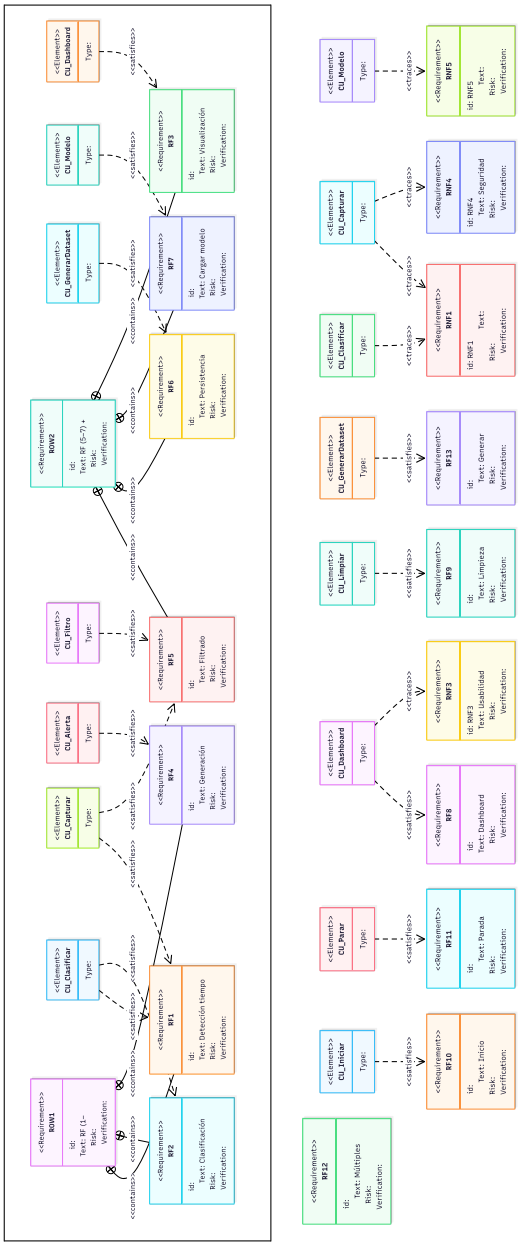

Figura 4.4: Diagrama de requisitos y trazabilidad (Parte 1 y Parte 2).

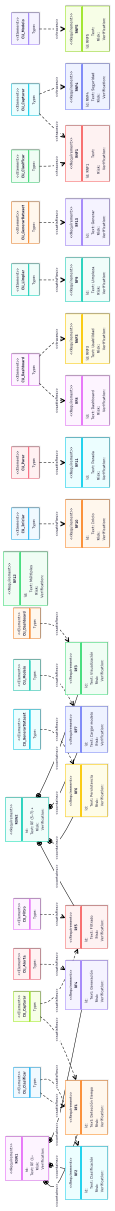

Figura 4.5: Diagrama de requisitos y trazabilidad (Parte 1 y Parte 2).

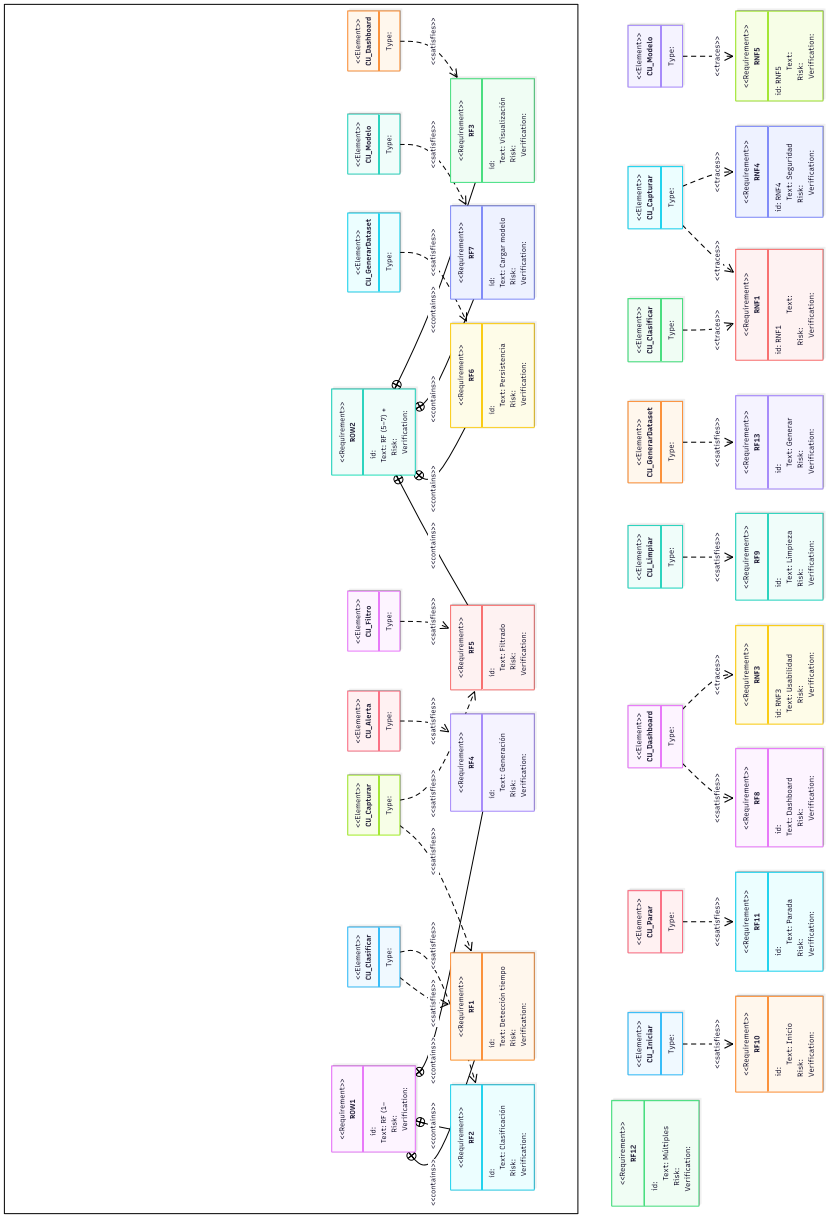

Figura 4.6: Diagrama de requisitos y trazabilidad (Parte 1 y Parte 2).

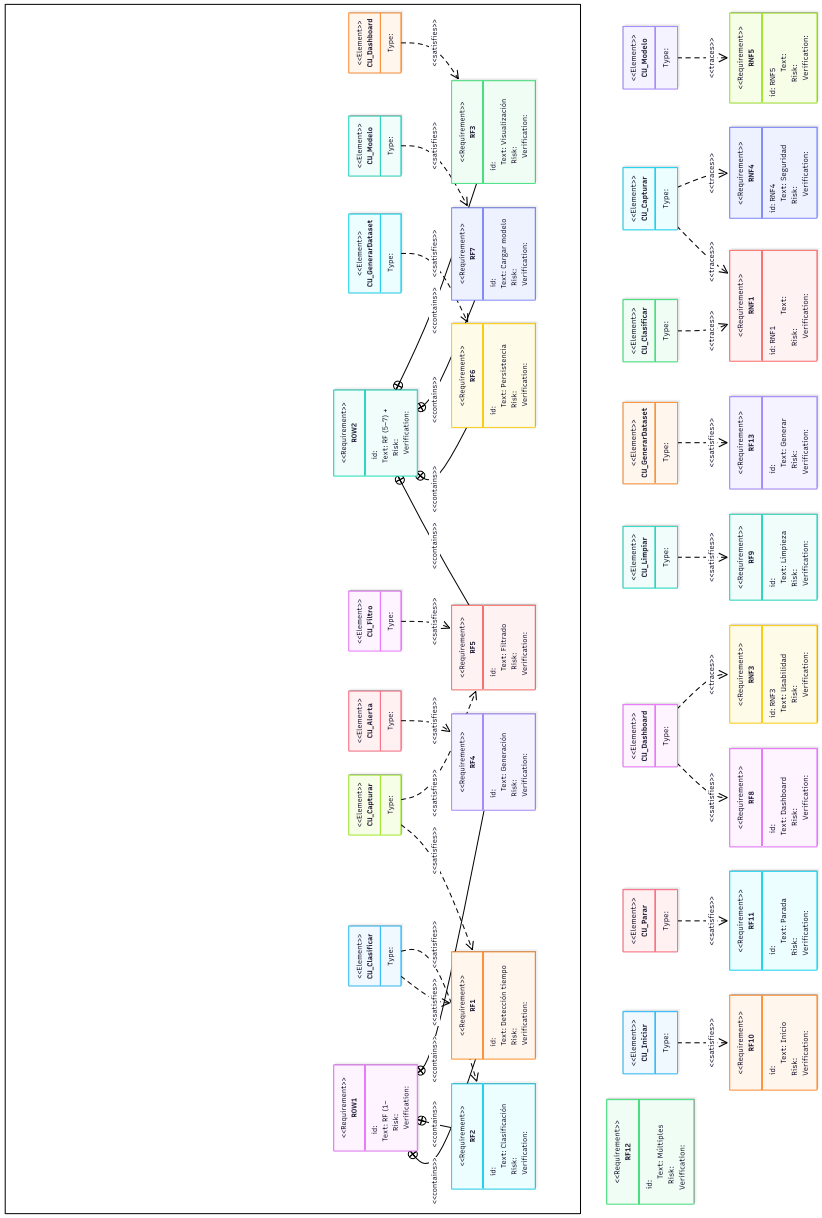

Figura 4.7: Diagrama de requisitos y trazabilidad (Parte 1 y Parte 2).

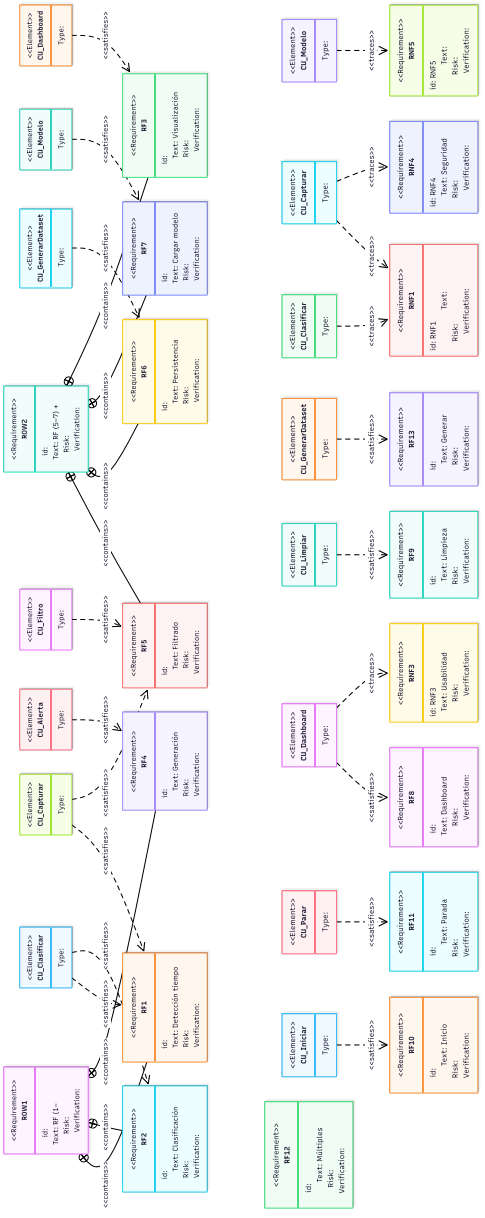

Figura 4.8: Diagrama de requisitos y trazabilidad (Parte 1 y Parte 2).

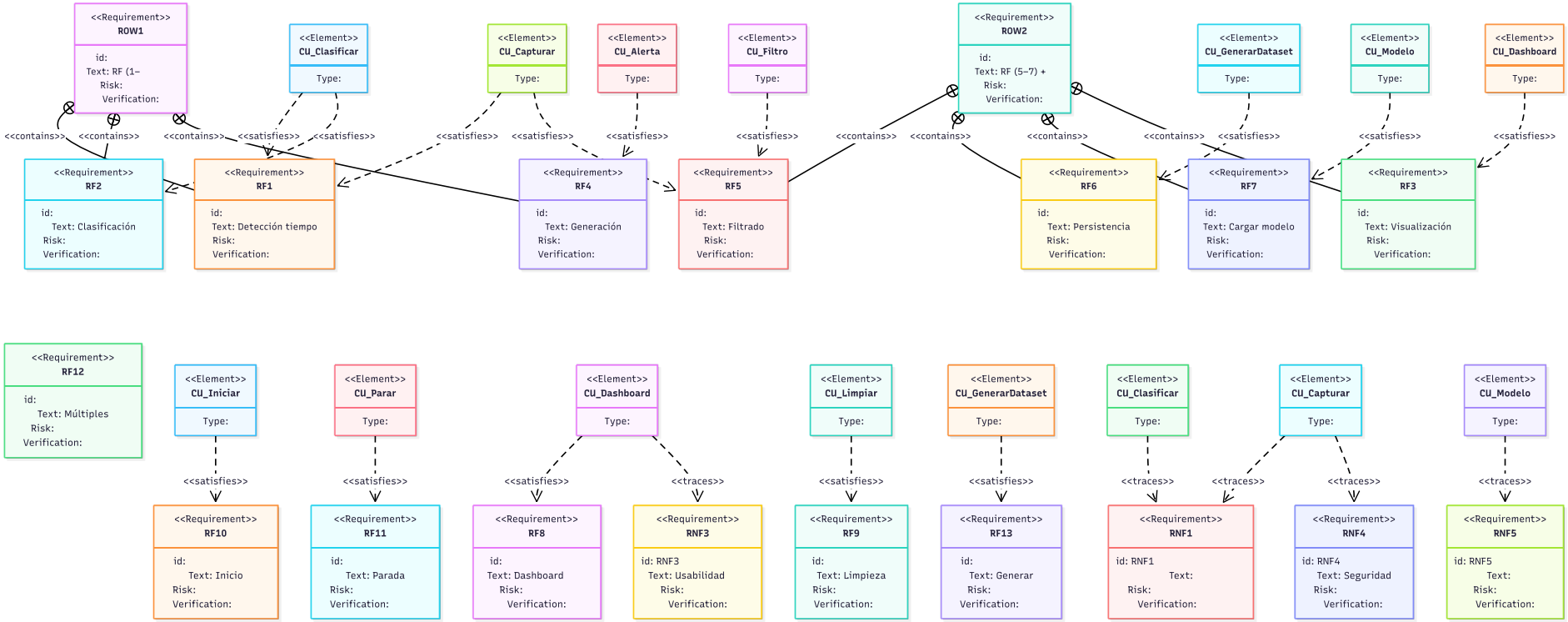


Figura 4.9: Diagrama de requisitos y trazabilidad (Parte 1 y Parte 2).

## Diagrama de requisitos y trazabilidad (resumen de RF y RNF)

La Figura. 4.9 integra en una sola vista los requisitos funcionales (RFx) y no funcionales (RNFx) del sistema y su trazabilidad con los elementos de diseño (casos de uso, `CU_*`).

### Qué muestra el diagrama

- Los nodos con estereotipo `«requirements»` representan requisitos. Cada uno recoge: **Id**, **Text** (resumen), **Risk** (prioridad/impacto) y **Verification** (método de verificación: *Test*, *Inspection* o *Analysis*).
- Los nodos con estereotipo `«element»` representan elementos de diseño; en este proyecto corresponden a casos de uso (*Type: usecase*).

### Convenciones de relaciones

- **contains**: descomposición/agrupación. Un requisito de alto nivel agrupa a otros más concretos.
- **satisfies**: relación de cobertura. Un caso de uso implementa el comportamiento requerido por un RF.
- **traces**: relación de trazado. Un RNF condiciona a un caso de uso (p. ej., restricciones de seguridad, rendimiento o usabilidad) sin implicar implementación funcional directa.

### Estructura de alto nivel

- **ROW1** y **ROW2** actúan como requisitos de organización, desde los que se descomponen y agrupan RF relacionados mediante *contains*.
- Existen dependencias/refinamientos entre RF (flechas entre requisitos) que documentan precondiciones lógicas del flujo (p. ej., para clasificar es necesario antes capturar/filtrar).

**Cobertura funcional (ejemplos destacados)**

- **CU_Iniciar** *satisfies* **RF10** (inicio del sistema).
- **CU_Parar** *satisfies* **RF11** (parada controlada).
- **CU_Dashboard** *satisfies* **RF8** (panel/visualización).
- **CU_Limpiar** *satisfies* **RF9** (limpieza de datos/estado).
- **CU_GenerarDataset** *satisfies* **RF13** (generación de dataset).
- En la parte superior, **CU_Clasificar**, **CU_Capturar**, **CU_Alerta** y **CU_Filtro** cubren el flujo principal de tratamiento (detección, alertado, filtrado, etc.). En el clúster derecho, **CU_Modelo** y **CU_GenerarDataset** completan los RF de modelo y preparación de datos.
- Resultado: cada **RF** queda cubierto al menos por un **CU**; estos enlaces *satisfies* permiten derivar casos de prueba de aceptación.

**Trazabilidad de requisitos no funcionales (RNF)**

- **RNF3** (usabilidad) *traces* a **CU_Dashboard**, donde se materializan decisiones de interfaz/UX.
- RNF transversales (p. ej., *seguridad*, *rendimiento*, *calidad del modelo*) *traces* a **CU_Capturar**, **CU_Clasificar**, **CU_Modelo**, etc., dejando explícito dónde deben comprobarse dichas restricciones.
- La verificación de RNF se documenta en *Verification* (*Analysis*/*Inspection*; *Test* cuando procede, p. ej., tiempos de respuesta o controles de acceso).

**Riesgo y priorización**

- El campo **Risk** clasifica el impacto de no cumplimiento y guía la priorización de implementación y pruebas: los requisitos con riesgo *High* se abordan antes y con mayor cobertura de validación.

**Cómo leer el diagrama de requisitos y trazabilidad**

1. Identificar, de izquierda a derecha, los grupos de RF bajo **ROW1**/**ROW2** para obtener el mapa de alcance.
2. Para cada **RF**, seguir las flechas *satisfies* hacia arriba para ver qué **CU** lo implementa (y dónde se prueba).
3. Para cada **RNF**, seguir las flechas *traces* para localizar los **CU** que deben cumplir la restricción.
4. Consultar **Risk** y **Verification** para conocer la prioridad y el tipo de evidencia planificada en validación.

Por lo tanto, el diagrama o figura proporciona *trazabilidad bidireccional*: desde un requisito es posible localizar los elementos que lo implementan y, desde un caso de uso, los requisitos que satisface o que lo condicionan. Los campos **Risk** y **Verification** completan la planificación al vincular prioridad y evidencias de prueba.

## 4.2. Diagrama de casos de uso

Este apartado describe los casos de uso identificados para el Sistema IDS desarrollado. Cada caso de uso se alinea con uno o varios requisitos funcionales (RF) y no funcionales (RNF). Para la notación se emplea una plantilla homogénea con: Objetivo, Actores, Precondiciones, Postcondiciones, Flujo Principal, Flujos Alternativos / Excepciones, Requisitos Asociados, Datos y Frecuencia.

### 4.2.1. Listado de Casos de Uso

- CU_Configurar Captura
- CU_Iniciar Captura
- CU_Capturar Tráfico
- CU_Filtrar Captura
- CU_Agregar Paquetes a Flujo
- CU_Clasificar Flujo

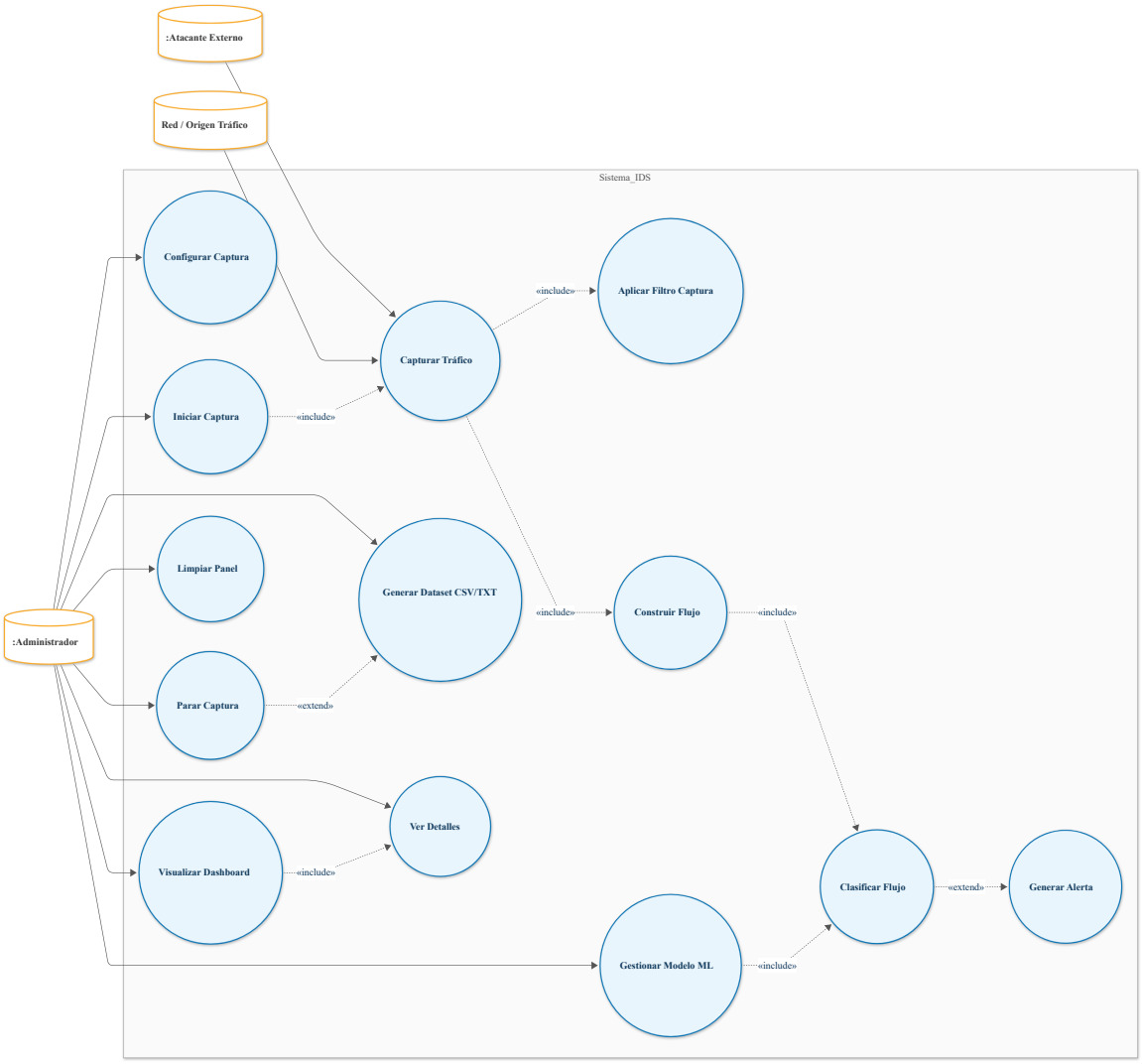


Figura 4.10: Diagrama de casos de uso principal.

- CU_Generar Alerta
- CU_Visualizar Dashboard
- CU_Mostrar Detalles
- CU_Limpiar Panel
- CU_Parar Captura
- CU_Generar Dataset (Exportar CSV/TXT)
- CU_Gestionar Modelo ML

**CU_Configurar Captura**

**Objetivo:** Permitir al Administrador seleccionar interfaz y filtro inicial antes de iniciar la captura.

**Actores:** Administrador (primario).

**Precondiciones:** El sistema está operativo pero la captura no ha comenzado.

**Postcondiciones:** Queda almacenada la configuración (interfaz, filtro BPF u otro) en el estado global.

**Flujo Principal:**
1. El Administrador accede al panel de configuración.
2. Selecciona la interfaz de red disponible.
3. (Opcional) Introduce un filtro (por ejemplo “tcp port 80”).
4. El sistema valida sintaxis mínima (formato no vacío).
5. Se actualiza el estado interno (State.set_interface / set_filter).

**Flujos Alternativos / Excepciones:**
A1 Interfaz inválida: se muestra mensaje y no se guarda.
A2 Filtro vacío: se acepta como “sin filtro”.

**Requisitos Asociados:** RF5, RF12, RF10 (previo al inicio), RNF3.

**Datos:** Nombre de interfaz, cadena de filtro.

**Frecuencia:** Cada vez que se quiera cambiar la configuración antes o durante la operación (si se soporta cambio dinámico).

**CU_Iniciar Captura**

**Objetivo:** Comenzar la captura de paquetes en la interfaz configurada.

**Actores:** Administrador.

**Precondiciones:** Configuración válida establecida; capturador detenido.

**Postcondiciones:** Hilo de captura (sniffer) en ejecución; worker activo.

**Flujo Principal:**
1. El Administrador pulsa “Iniciar”.
2. El sistema crea instancia de FlowSession y reemplaza el writer por ConsoleWriter.

3. Se lanza el hilo de sniffing (scapy AsyncSniffer o bucle sniff).
4. Se lanza el worker (thread) de procesamiento de flujos.
5. Se marca el estado capturing = True.
6. La UI pasa a modo Online.

**Flujos Alternativos:** A1 Error al crear FlowSession: se registra log y no se inicia (capturing permanece False).

A2 Interfaz sin tráfico: el sistema sigue esperando.

**Requisitos Asociados:** RF10, RF1, RNF1, RNF5.

**Datos:** Parámetros de inicialización (interfaz, filtro).

**Frecuencia:** Varias veces a lo largo de la vida del sistema (tras paradas).

**CU_Capturar Tráfico**

**Objetivo:** Interceptar paquetes de red y asociarlos a flujos.

**Actores:** Sniffer (sistema), Red / Atacante (fuentes externas).

**Precondiciones:** Captura iniciada.

**Postcondiciones:** Paquetes tratados y entregados a FlowSession.

**Flujo Principal:** 1. El sniffer recibe un paquete.
2. Aplica filtro (si existe).
3. Invoca on_packet_received(packet).
4. Determina clave de flujo (5-tuple + dirección).

**Flujos Alternativos:** A1 Paquete no TCP ni UDP (si restricción activa) se descarta.

A2 Error parsing -> se ignora.

**Requisitos Asociados:** RF1, RF5, RF12, RNF1.

**Datos:** Paquetes en bruto (headers, tiempos).

**Frecuencia:** Continuamente mientras haya tráfico.

**CU_Filtrar Captura**

**Objetivo:** Limitar el tráfico procesado según criterios (puerto, protocolo).

**Actores:** Administrador.

**Precondiciones:** Configuración aceptada; captura activa o próxima a iniciar.

**Postcondiciones:** Nuevo filtro aplicado (en reinicio o inmediatamente si se soporta).

**Flujo Principal:**
1. Administrador edita campo de filtro.
2. Sistema almacena nuevo valor.
3. En el siguiente reinicio de la captura, se usa el filtro.

**Flujos Alternativos:** A1 Filtro sintácticamente inválido: se ignora y mantiene anterior.

**Requisitos Asociados:** RF5, RNF3.

**Datos:** Expresión BPF o similar.

**Frecuencia:** Esporádica según necesidad.

**CU_Agregar Paquetes a Flujo**

**Objetivo:** Agregar cada paquete al flujo correspondiente y actualizar métricas.

**Actores:** FlowSession, Flow.

**Precondiciones:** Clave de flujo calculada; flujo existente o se creará uno nuevo.

**Postcondiciones:** Estructura Flow actualizada; listas internas (paquetes, IAT, active/idle, bulk).

**Flujo Principal:**
1. FlowSession localiza flujo existente o crea uno nuevo.
2. Invoca flow.add_packet(packet, direction).
3. Actualiza timestamps, tamaños, flags, contadores.
4. Evalúa condiciones de expiración (FIN, duración, inactividad).

**Flujos Alternativos:** A1 Expiración superada -> se fuerza clean_write_flows y se abre un nuevo contador (count++).

**Requisitos Asociados:** RF1, RNF1.

**Datos:** Listas de paquetes y features incrementales.

**Frecuencia:** Por cada paquete.

**CU_Clasificar Flujo**

**Objetivo:** Etiquetar cada flujo como Normal o Malicioso.

**Actores:** Worker (primario), Modelo RF (externo lógico), Administrador (interesado).

**Precondiciones:** Flujo finalizado (expiración, FIN o límite de paquetes); modelo cargado.

**Postcondiciones:** Flujo con etiqueta y probabilidad en el estado global.

**Flujo Principal:**
1. El writer encola datos del flujo.
2. Worker consume de la cola.
3. Mapea/ordena características al formato CIC.
4. Escala / transforma según scaler guardado.
5. Invoca predict().
6. Recibe etiqueta y probabilidad y actualiza contadores.

**Flujos Alternativos:** A1 Modelo no cargado: etiqueta *Unknown*.

A2 Excepción de predicción: etiqueta *Error*, probabilidad 0.

**Requisitos Asociados:** RF1, RF2, RF7, RNF1, RNF5.

**Datos:** Vector de características estandarizadas.

**Frecuencia:** Cada flujo finalizado.

**CU_Generar Alerta**

**Objetivo:** Registrar una alerta cuando un flujo es malicioso.

**Actores:** Worker / Estado, Administrador (visualiza).

**Precondiciones:** Flujo clasificado como Malicious (etiqueta binaria).

**Postcondiciones:** Alerta añadida a la lista de alertas.

**Flujo Principal:**
1. Worker detecta predicción Malicious.
2. Crea estructura de alerta (timestamp, IP origen/destino, severidad).
3. Inserta en lista de alertas (máx. 50 manteniendo las más recientes).
4. UI muestra contador actualizado.

**Flujos Alternativos:** A1 Lista supera capacidad -> se descarta la más antigua.

**Requisitos Asociados:** RF4, RF1, RNF3.

**Datos:** IPs, probabilidad, severidad (actualmente simulada).

**Frecuencia:** Cada flujo malicioso.

**CU_Visualizar Dashboard**

**Objetivo:** Presentar métricas, flujos y alertas en tiempo (casi) real.

**Actores:** Administrador.

**Precondiciones:** Sistema en ejecución (capturando o con datos cargados).

**Postcondiciones:** Información visual actualizada cada intervalo (2 s).

**Flujo Principal:**
1. La UI solicita refresco (timer).
2. Estado devuelve totales y colecciones.
3. Se renderizan tarjetas (flujos, alertas, normales, ataques).

**Flujos Alternativos:** A1 Sin flujos todavía: se muestra mensaje de espera.

**Requisitos Asociados:** RF3, RF8, RNF3.

**Datos:** Listas en memoria (no persistidas).

**Frecuencia:** Cada ciclo de refresco.

**CU_Mostrar Detalles**

**Objetivo:** Inspeccionar características detalladas de un flujo o alerta.

**Actores:** Administrador.

**Precondiciones:** Existen flujos y/o alertas en la UI.

**Postcondiciones:** Modal desplegada con datos completos.

**Flujo Principal:**
1. Administrador pulsa “Detalles”.
2. Sistema localiza el objeto por ID.
3. Copia información en selected_flow / selected_alert.

4. Abre modal con métricas y flags.

**Flujos Alternativos:** A1 ID no encontrado (eliminado por poda) -> se cancela.

**Requisitos Asociados:** RF3, RF8.

**Datos:** Campos completos del flujo (features) y la predicción.

**Frecuencia:** A demanda.

**CU_Limpiar Panel**

**Objetivo:** Vaciar las listas de flujos y alertas para reinicio visual.

**Actores:** Administrador.

**Precondiciones:** Existen flujos/alertas en memoria.

**Postcondiciones:** Listas vacías y contadores a cero.

**Flujo Principal:** 1. Administrador pulsa “Limpiar”.

2. Sistema asigna listas vacías y reinicia contadores.
3. UI refleja estado vacío.

**Flujos Alternativos:** No aplica.

**Requisitos Asociados:** RF9.

**Datos:** N/A (operación destructiva en memoria).

**Frecuencia:** Esporádica.

**CU_Parar Captura**

**Objetivo:** Finalizar la captura y forzar escritura de flujos pendientes.

**Actores:** Administrador.

**Precondiciones:** Captura activa.

**Postcondiciones:** Hilos detenidos; flujos procesados; estado Offline.

**Flujo Principal:** 1. Administrador pulsa “Parar”.

2. Sistema marca capturing = False.

3. Lanza limpieza de flujos (clean_write_flows).
4. Worker vacía cola pendiente.
5. UI muestra estado Offline.

**Flujos Alternativos:** A1 Error en limpieza: se registra log, se continúa parada.

**Requisitos Asociados:** RF11, RF1, RNF1.

**Datos:** Flujos en cola final.

**Frecuencia:** Cada ciclo de captura.

**CU_Generar Dataset**

**Objetivo:** Exportar los flujos capturados a formato CSV o TXT para entrenamiento futuro o análisis.

**Actores:** Administrador.

**Precondiciones:** Existen flujos capturados; el writer soporta el formato.

**Postcondiciones:** Archivo generado en el sistema de ficheros.

**Flujo Principal:**
1. Administrador solicita exportación (o esta ocurre al finalizar un lote).
2. Writer serializa las características en filas.
3. Se cierra (o sincroniza) el descriptor de archivo.
4. Se notifica éxito (log/UI).

**Flujos Alternativos:** A1 Error de escritura: se avisa, pudiendo reintentar.

**Requisitos Asociados:** RF13, RF6 (si se considera histórico), RNF6.

**Datos:** Todas las features por flujo (ver Anexo de Features).

**Frecuencia:** Bajo demanda o al parar.

**CU_Gestionar Modelo ML**

**Objetivo:** Cargar, validar y (potencialmente) actualizar el modelo de Random Forest.

**Actores:** Administrador (dispara entrenamiento externo), Sistema (carga al arrancar).

**Precondiciones:** Ficheros del modelo (.pkl) disponibles.

**Postcondiciones:** Modelo y scaler listos para inferencia (flag is_loaded).

**Flujo Principal:**
1. Al iniciar la aplicación se intenta load_model().
2. Se cargan modelo, scaler, mapping y lista de features.
3. Se valida coherencia (número de features).
4. Se expone is_loaded=True.

**Flujos Alternativos:**
A1 Archivos no encontrados: is_loaded=False, se muestra mensaje.

A2 Error de deserialización: se registra log, mantiene estado no cargado.

**Requisitos Asociados:** RF7, RNF5, RNF6.

**Datos:** Ficheros .pkl (modelo, scaler, mapping, features).

**Frecuencia:** Al arranque; manual tras reentrenamiento.

### 4.2.2. Mapa Caso de Uso Requisitos (Resumen)

| Caso de Uso | Requisitos Principales |
|---|---|
| CU_Configurar | RF5, RF12, RNF3 |
| CU_Iniciar | RF10, RF1, RNF1 |
| CU_Capturar | RF1, RF5, RF12, RNF1 |
| CU_Filtrar | RF5, RNF3 |
| CU_Agregar Paquetes | RF1, RNF1 |
| CU_Clasificar | RF1, RF2, RF7, RNF1, RNF5 |
| CU_Generar Alerta | RF4, RF1 |
| CU_Visualizar Dashboard | RF3, RF8, RNF3 |
| CU_Mostrar Detalles | RF3, RF8 |
| CU_Limpiar Panel | RF9 |
| CU_Parar Captura | RF11, RF1 |
| CU_Generar Dataset | RF13, RF6 |
| CU_Gestionar Modelo ML | RF7, RNF5, RNF6 |

Con esto queda descrita la interacción funcional del sistema. El desarrollo de un sistema de detección de intrusiones basado en Machine Learning se apoya en un ecosistema de tecnologías y herramientas existentes que facilitan la captura de datos, su procesamiento, el entrenamiento de modelos y la visualización de resultados.

## 4.3. Grámatica del caso de uso

El desarrollo de un sistema de detección de intrusiones basado en Machine Learning se apoya en un ecosistema de tecnologías y herramientas existentes que facilitan la captura de datos, su procesamiento, el entrenamiento de modelos y la visualización de resultados.

En el ámbito de la captura y análisis de tráfico de red, herramientas como Wireshark o tcpdump son estándares de la industria que permiten la interceptación y el análisis profundo de paquetes. Wireshark, en particular, ofrece capacidades de descifrado y una interfaz gráfica detallada para la inspección manual del tráfico.

Para el desarrollo de los algoritmos de Machine Learning, Python se ha consolidado como el lenguaje de programación por excelencia en ciencia de datos, gracias a su rica colección de librerías. Scikit-learn proporciona una implementación robusta de una amplia gama de algoritmos de clasificación, incluyendo Random Forest, SVM, Naive Bayes y KNN. Para enfoques de Deep Learning, TensorFlow y PyTorch son los frameworks dominantes, ofreciendo flexibilidad y eficiencia computacional para la construcción y entrenamiento de redes neuronales complejas.

El procesamiento y manipulación de grandes volúmenes de datos de red se beneficia enormemente de librerías como Pandas para la gestión de DataFrames y NumPy para operaciones numéricas. La visualización de los datos y los resultados del sistema, como un dashboard de alertas, se puede lograr con frameworks de desarrollo web como Flask o Django en Python, combinados con librerías de visualización front-end.

Finalmente, el uso de entornos de desarrollo integrado (IDE) como VS Code o PyCharm, junto con sistemas de control de versiones como Git y plataformas como GitHub, son tecnologías esenciales que aseguran un desarrollo colaborativo, organizado y eficiente del proyecto.

El desarrollo de un sistema de detección de intrusiones basado en Machine Learning se apoya en un ecosistema de tecnologías y herramientas existentes que facilitan la captura de datos, su procesamiento, el entrenamiento de modelos y la visualización de resultados.

En el ámbito de la captura y análisis de tráfico de red, herramientas como Wireshark o tcpdump son estándares de la industria que permiten la interceptación y el análisis profundo de paquetes. Wireshark, en particular, ofrece capacidades de descifrado y una interfaz gráfica detallada para la inspección manual del tráfico.

Para el desarrollo de los algoritmos de Machine Learning, Python se ha consolidado como el lenguaje de programación por excelencia en ciencia de datos, gracias a su rica colección de librerías. Scikit-learn proporciona una implementación robusta de una amplia gama de algoritmos de clasificación, incluyendo Random Forest, SVM, Naive Bayes y KNN. Para enfoques de Deep Learning, TensorFlow y PyTorch son los frameworks dominantes, ofreciendo flexibilidad y eficiencia computacional para la construcción y entrenamiento de redes neuronales complejas.

El procesamiento y manipulación de grandes volúmenes de datos de red se beneficia enormemente de librerías como Pandas para la gestión de DataFrames y NumPy para operaciones numéricas. La visualización de los datos y los resultados del sistema, como un dashboard de alertas, se puede lograr con frameworks de desarrollo web como Flask o Django en Python, combinados con librerías de visualización front-end.

Finalmente, el uso de entornos de desarrollo integrado (IDE) como VS Code o PyCharm, junto con sistemas de control de versiones como Git y plataformas como GitHub, son tecnologías esenciales que aseguran un desarrollo colaborativo, organizado y eficiente del proyecto.

## 4.4. Diagrama de actividades

### 4.4.1. Exportación del dataset

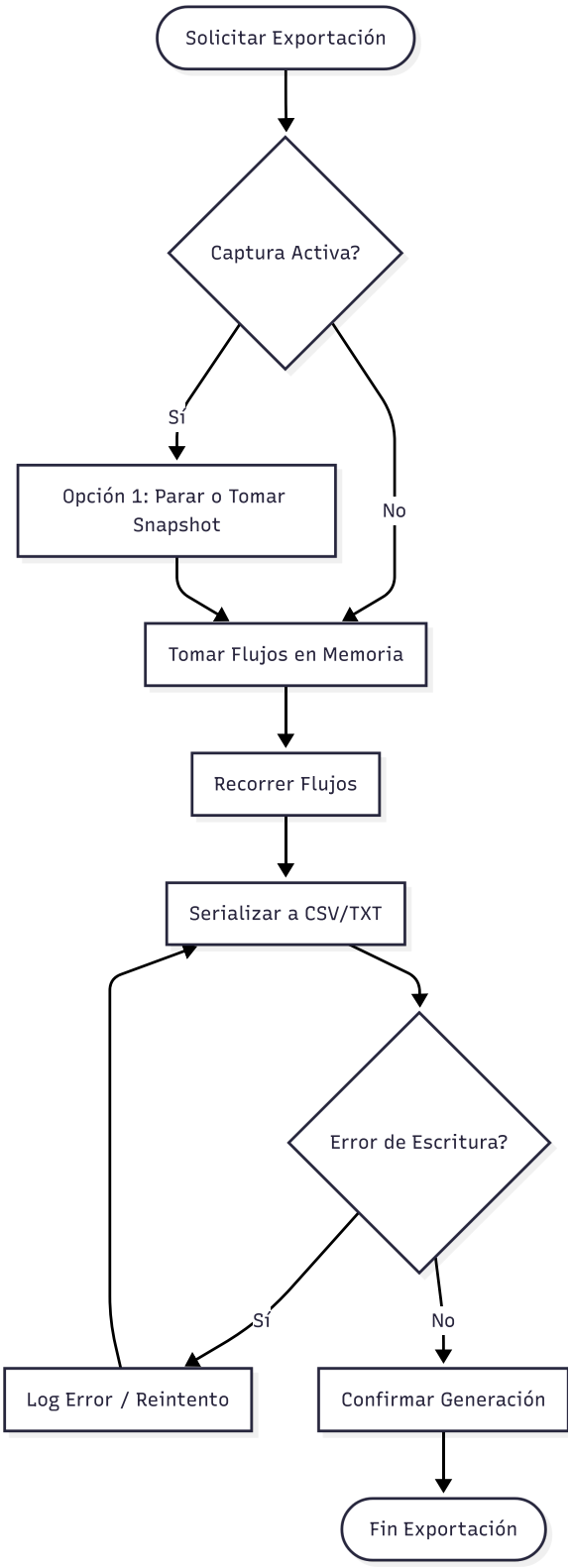


Figura 4.11: Diagrama de actividades de exportación del dataset.

### 4.4.2. Captura y clasificación

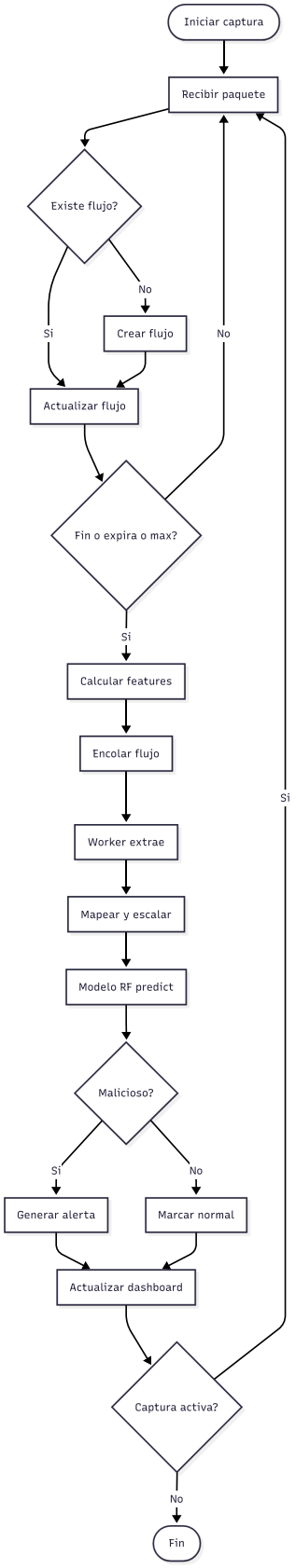


Figura 4.12: Diagrama de actividades de captura y clasificación.

### 4.4.3. Entrenamiento

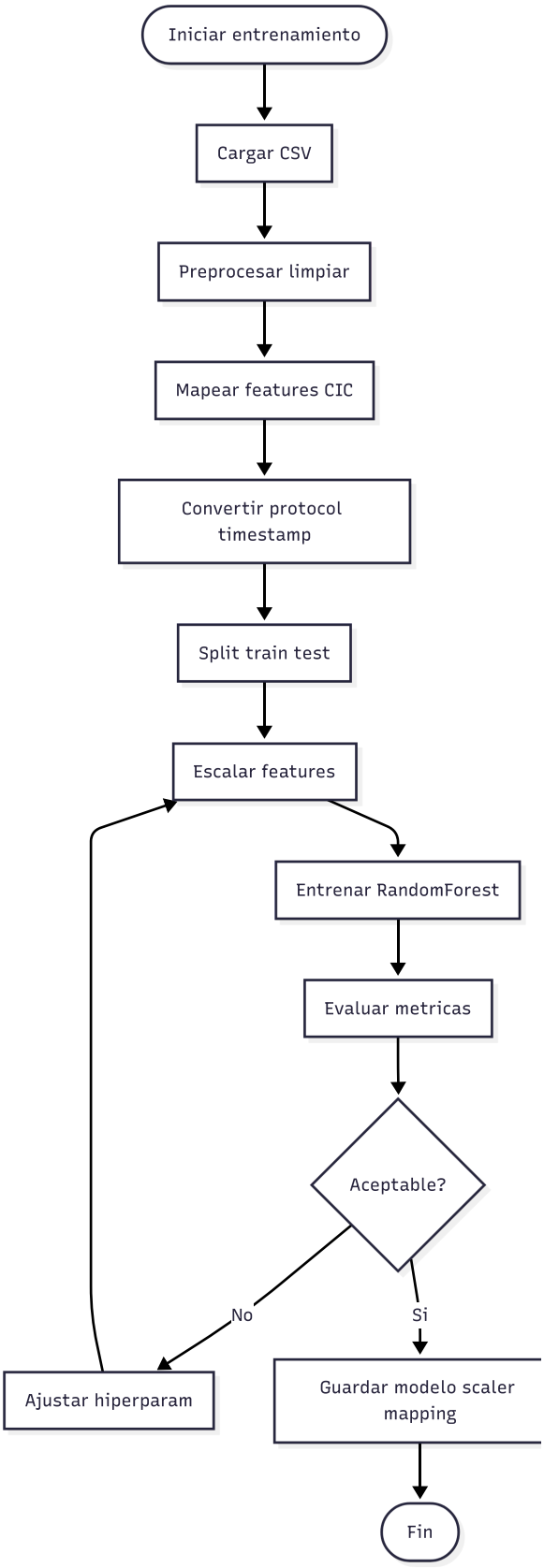


Figura 4.13: Diagrama de actividades de entrenamiento.

## 4.5. Diagrama de secuencia

El desarrollo de un sistema de detección de intrusiones basado en Machine Learning se apoya en un ecosistema de tecnologías y herramientas existentes que facilitan la captura de datos, su procesamiento, el entrenamiento de modelos y la visualización de resultados.

En el ámbito de la captura y análisis de tráfico de red, herramientas como Wireshark o tcpdump son estándares de la industria que permiten la interceptación y el análisis profundo de paquetes. Wireshark, en particular, ofrece capacidades de descifrado y una interfaz gráfica detallada para la inspección manual del tráfico.

Para el desarrollo de los algoritmos de Machine Learning, Python se ha consolidado como el lenguaje de programación por excelencia en ciencia de datos, gracias a su rica colección de librerías. Scikit-learn proporciona una implementación robusta de una amplia gama de algoritmos de clasificación, incluyendo Random Forest, SVM, Naive Bayes y KNN. Para enfoques de Deep Learning, TensorFlow y PyTorch son los frameworks dominantes, ofreciendo flexibilidad y eficiencia computacional para la construcción y entrenamiento de redes neuronales complejas.

El procesamiento y manipulación de grandes volúmenes de datos de red se beneficia enormemente de librerías como Pandas para la gestión de DataFrames y NumPy para operaciones numéricas. La visualización de los datos y los resultados del sistema, como un dashboard de alertas, se puede lograr con frameworks de desarrollo web como Flask o Django en Python, combinados con librerías de visualización front-end.

Finalmente, el uso de entornos de desarrollo integrado (IDE) como VS Code o PyCharm, junto con sistemas de control de versiones como Git y plataformas como GitHub, son tecnologías esenciales que aseguran un desarrollo colaborativo, organizado y eficiente del proyecto.

El desarrollo de un sistema de detección de intrusiones basado en Machine Learning se apoya en un ecosistema de tecnologías y herramientas existentes que facilitan la captura de datos, su procesamiento, el entrenamiento de modelos y la visualización de resultados.

En el ámbito de la captura y análisis de tráfico de red, herramientas como Wireshark o tcpdump son estándares de la industria que permiten la interceptación y el análisis profundo de paquetes. Wireshark, en particular, ofrece capacidades de descifrado y una interfaz gráfica detallada para la inspección manual del tráfico.

Para el desarrollo de los algoritmos de Machine Learning, Python se ha consolidado como el lenguaje de programación por excelencia en ciencia de datos, gracias a su rica colección de librerías. Scikit-learn proporciona una implementación robusta de una amplia gama de algoritmos de clasificación, incluyendo Random Forest, SVM, Naive Bayes y KNN. Para enfoques de Deep Learning, TensorFlow y PyTorch son los frameworks dominantes, ofreciendo flexibilidad y eficiencia computacional para la construcción y entrenamiento de redes neuronales complejas.

El procesamiento y manipulación de grandes volúmenes de datos de red se beneficia enormemente de librerías como Pandas para la gestión de DataFrames y NumPy para operaciones numéricas. La visualización de los datos y los resultados del sistema, como un dashboard de alertas, se puede lograr con frameworks de desarrollo web como Flask o Django en Python, combinados con librerías de visualización front-end.

Finalmente, el uso de entornos de desarrollo integrado (IDE) como VS Code o PyCharm, junto con sistemas de control de versiones como Git y plataformas como GitHub, son tecnologías esenciales que aseguran un desarrollo colaborativo, organizado y eficiente del proyecto.

### 4.5.1. Principal

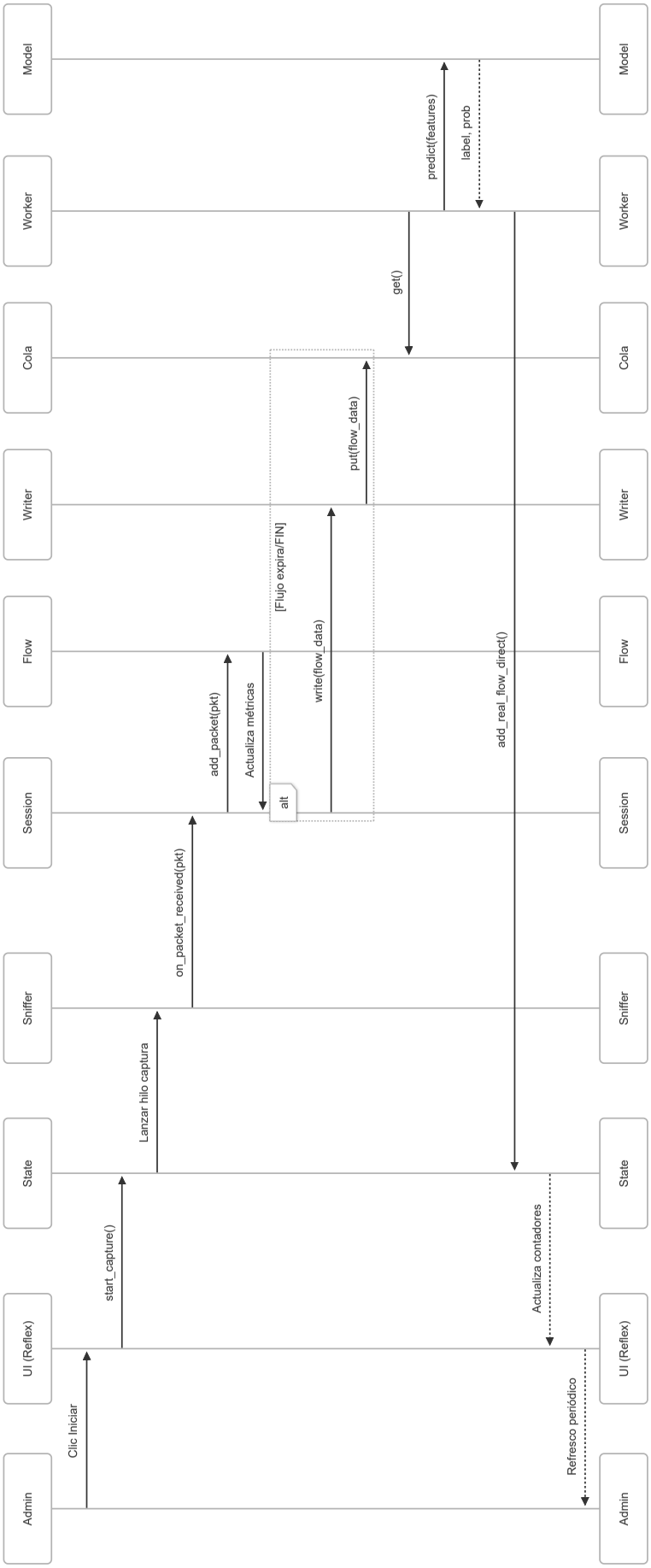


Figura 4.14: Diagrama de secuencia principal.

### 4.5.2. Expiración y cierre de flujos

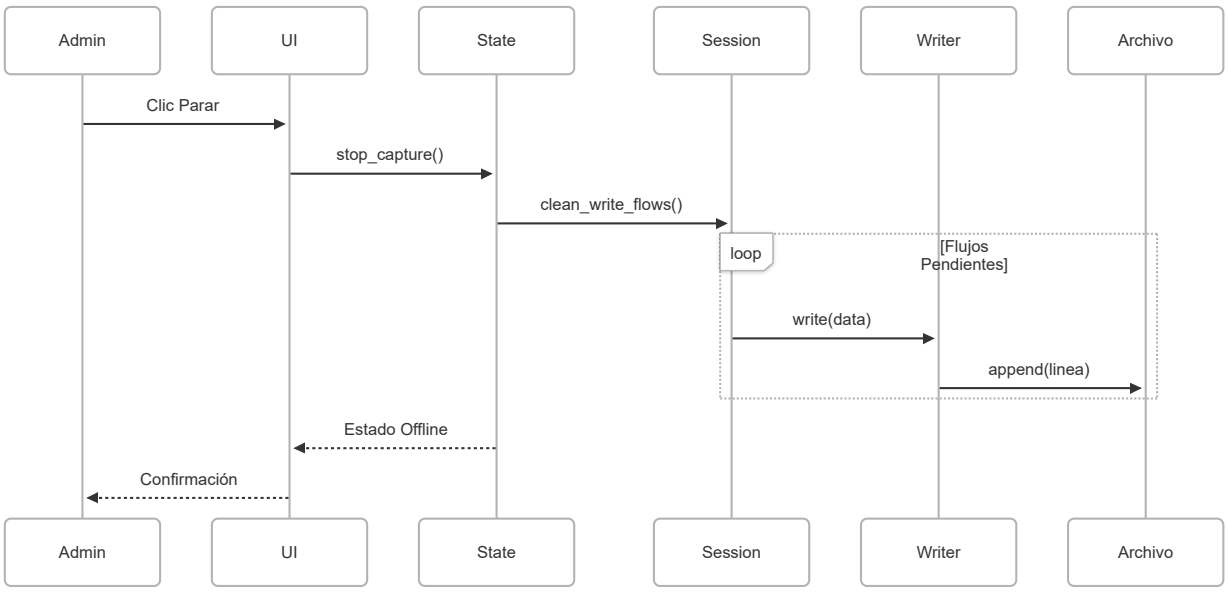


Figura 4.15: Diagrama de actividades de exportación del dataset.

# Capítulo 5

# DISEÑO

## 5.1. Arquitectura del sistema

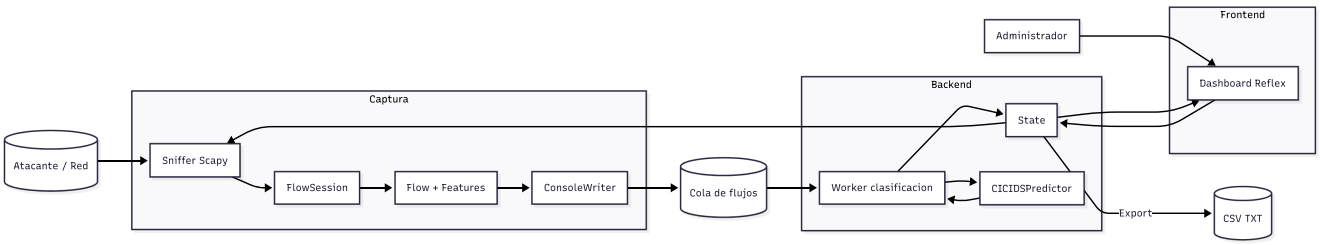


Figura 5.1: Diagrama de arquitectura del sistema.

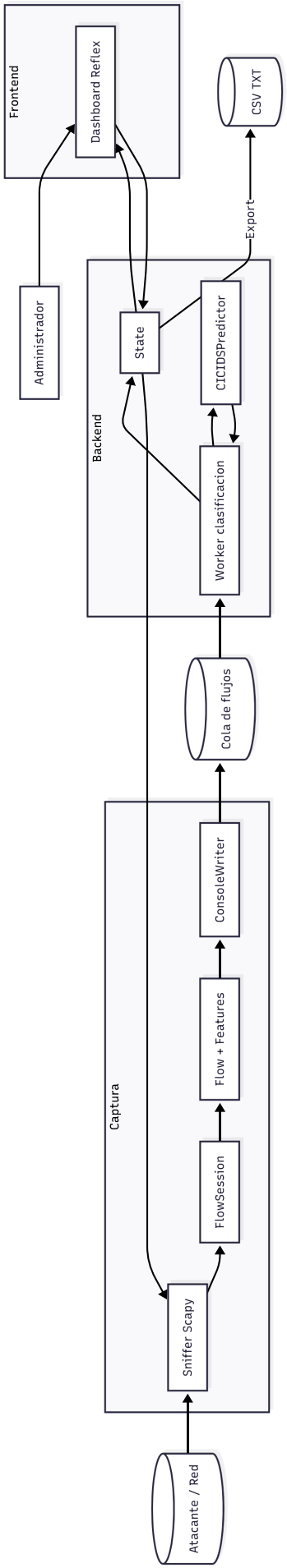


Figura 5.2: Diagrama de arquitectura del sistema.

## 5.2. Wireframe

## 5.3. Vistas necesarias

## 5.4. Mockups

# Capítulo 6

# IMPLEMENTACIÓN

# Capítulo 7

# RESULTADOS

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel

magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

# Capítulo 8

# CONCLUSIONES

## 8.1. Valoración Personal

Esta memoria, culmen de un trayecto académico de cuatro años, representa no solo la materialización de un proyecto final de grado, sino también el reflejo de un profundo aprendizaje, compromiso y constancia. Este Trabajo de Fin de Grado constituye el desafío final para la obtención del título de Ingeniería Informática, encapsulando la evolución formativa de toda la carrera.

Mi interés por el ámbito de la seguridad informática se manifestó incluso antes de iniciar mis estudios universitarios. La fascinación por comprender cómo los sistemas podían ser controlados y manipulados de diversas formas despertó en mí una constante inquietud por identificar y mitigar vulnerabilidades. Esta pasión inicial se consolidó durante el bachillerato, impulsada por el intercambio de conocimientos con un compañero con afinidad por la seguridad, lo que nos llevó a una inmersión más profunda en la investigación de este campo.

Ya en el grado, la asignatura de "Seguridad en Tecnologías de la Información" en segundo curso avivó aún más mi deseo de especializarme en esta disciplina. Desde la complejidad de vulnerabilidades a nivel de hardware hasta la sencillez de otras detectables con una inspección somera en entornos web, cada aspecto me cautivó. Paralelamente, la introducción a la Inteligencia Artificial y las Metaheurísticas en cursos subsiguientes despertó un nuevo interés hacia estos modelos avanzados, capaces de emular el aprendizaje humano para optimizar resultados. Esta confluencia de intereses me llevó a la elección de un proyecto final de grado que integrara ambas disciplinas: ciberseguridad e inteligencia artificial. Adicionalmente, la estrecha relación

entre la seguridad y el mundo de las redes, reforzada por la asignatura de "Redes e Infraestructuras", me decantó finalmente por la implementación de un capturador de tráfico en tiempo real, concebido como la base para un sistema de detección de intrusiones.

Las etapas iniciales del desarrollo del capturador estuvieron marcadas por desafíos significativos, particularmente la dificultad para encontrar APIs y librerías que facilitaran la implementación. La frustración inicial por la falta de información relevante me llevó a explorar incluso el funcionamiento interno de herramientas como Wireshark. Sin embargo, la perseverancia fue clave, y con el tiempo, logré identificar una API que proporcionaba las funcionalidades necesarias para extraer una vasta cantidad de características del tráfico de red. Esta fase de implementación, aunque desafiante, resultó profundamente apasionante, demostrando que la constancia es fundamental para alcanzar todas nuestras metas y objetivos.

Posteriormente, el proyecto evolucionó hacia el diseño de una aplicación web sencilla para visualizar los flujos de paquetes capturados en tiempo real. Esta etapa presentó un nuevo reto al requerir la selección e integración de un modelo de Machine Learning para la detección de intrusiones. Esto implicó una inmersión en los fundamentos del aprendizaje supervisado, decantándome por el modelo Random Forest. La asimilación de conceptos de minería de datos, preprocesamiento, análisis y limpieza de datos supuso un esfuerzo considerable, dada la densidad de la información. No obstante, este proceso culminó con el entrenamiento exitoso del sistema y su posterior integración en la aplicación web.

En lo referente a la documentación, este TFG ha representado una valiosa experiencia de aprendizaje sobre los requerimientos y la extensión que conlleva una memoria técnica completa, un contraste notable con los documentos de menor envergadura redactados en cursos anteriores. Asimismo, ha sido mi primera incursión en el entorno LaTeX , una herramienta que, a pesar de una curva de aprendizaje inicial, ha demostrado ser sorprendentemente intuitiva y sus utilidades para la redacción de memorias técnicas son inestimables.

Para concluir esta valoración personal, deseo expresar mi profunda satisfacción por la formación multidisciplinar adquirida en campos como la ciberseguridad, la inteligencia artificial, las redes y el desarrollo web. Destaco especialmente el dominio del lenguaje de programación Python, esencial en la ciencia de datos y en el mundo de la ciberseguridad, que ha sido fundamental para el éxito y la culminación de este proyecto. Me siento inmensamente orgulloso de este logro, que representa un hito significativo en mi trayectoria académica y profesional.

## 8.2. Posibles mejoras futuras

El presente Sistema de Detección de Intrusiones (IDS) y la herramienta de generación de conjuntos de datos constituyen una base sólida para futuras investigaciones y desarrollos. Las siguientes líneas proponen una serie de mejoras y extensiones que podrían potenciar significativamente las capacidades y la robustez del sistema, abordando tanto aspectos técnicos como funcionales.

1. **Diversificación y Optimización del Componente de Machine Learning**:
   - **Integración de Modelos Avanzados**: Explorar la implementación de otros algoritmos de clasificación supervisada como Support Vector Machines (SVM), Gradient Boosting (XGBoost, LightGBM) o Redes Neuronales Recurrentes (RNN) y Convolucionales (CNN), así como arquitecturas basadas en Deep Learning (como Transformers), las cuales han demostrado gran potencial en el análisis de secuencias de datos de red.
   - **Modelos Ensemble Híbridos**: Investigar la creación de modelos en ensemble que combinen la robustez de algoritmos como Random Forest (especialmente eficaz para la clasificación binaria de tráfico benigno/malicioso) con la capacidad de modelos de Deep Learning para diferenciar entre tipos específicos de ataque, aprovechando las fortalezas de cada técnica.
   - **Funcionalidad de Re-entrenamiento Continuo**: Desarrollar un mecanismo para permitir que los modelos de Machine Learning sean re-entrenados periódicamente con nuevos datos capturados y etiquetados, asegurando que el sistema se adapte a nuevas amenazas y evoluciones en los patrones de tráfico.
2. **Expansión y Refinamiento del Módulo de Adquisición y Preprocesamiento**:
   - **Recolección de Características Adicionales**: Ampliar el conjunto de características extraídas de los paquetes y flujos de red. Esto podría incluir métricas más complejas a nivel de aplicación, características temporales de las conexiones o datos específicos del payload, enriqueciendo la capacidad de discriminación del modelo.
   - **Manejo de Tráfico Cifrado**: Investigar y desarrollar métodos para la inspección y el tratamiento de paquetes cifrados (ej., TLS/SSL). Esto podría implicar el uso de funcionalidades de descifrado (similares a las presentes en herramientas como Wireshark, si se dispone de las claves) o el análisis de metadatos del tráfico cifrado para identificar anomalías sin violar la privacidad.

- **Normalización y Compatibilidad de Datos**: Mejorar los métodos de preprocesamiento para asegurar la compatibilidad y unificación de conjuntos de datos provenientes de diversas fuentes, facilitando la creación de datasets más grandes y robustos para el entrenamiento.

3. **Mejoras en la Interfaz de Usuario y Funcionalidades de Gestión**:

   - **Seguridad y Privacidad de la Aplicación Web**: Implementar mecanismos robustos de seguridad para la aplicación web, incluyendo la encapsulación y cifrado de la información transmitida entre el cliente y el servidor, así como la gestión de autenticación y autorización de usuarios.
   - **Visualización Interactiva y Alertas Avanzadas**: Desarrollar visualizaciones más dinámicas e interactivas del tráfico de red y las detecciones. Esto incluiría dashboards personalizables, la capacidad de filtrar alertas por tipo, gravedad o tiempo, y notificaciones en tiempo real (ej., vía correo electrónico o plataformas de mensajería).
   - **Gestión de Reglas Personalizadas**: Ofrecer la posibilidad a los usuarios de definir y gestionar sus propias reglas de detección, permitiendo una mayor adaptabilidad del sistema a entornos específicos o necesidades particulares.
   - **API RESTful**: Implementar una API RESTful para permitir la integración del IDS con otras herramientas de seguridad, sistemas de gestión de eventos e información de seguridad (SIEM) o plataformas de automatización de respuesta a incidentes.

4. **Escalabilidad, Despliegue y Robustez del Sistema**:

   - **Optimización del Rendimiento**: Mejorar el rendimiento del capturador y del motor de detección para manejar grandes volúmenes de tráfico de red en entornos de producción, optimizando el uso de recursos computacionales.
   - **Contenerización y Orquestación**: Empaquetar los diferentes módulos del sistema en contenedores (ej., Docker) para facilitar el despliegue, la portabilidad y la escalabilidad del IDS en entornos de producción o en la nube, utilizando herramientas de orquestación como Kubernetes.
   - **Pruebas de Resistencia y Ciberseguridad Ofensiva**: Realizar pruebas de estrés exhaustivas y aplicar técnicas de ciberseguridad ofensiva (ej., penetration testing, ataques adversarios a los modelos ML) para evaluar la resiliencia del sistema y su capacidad para detectar amenazas complejas y en evolución.

# Capítulo 9

# APÉNDICES

## 9.1. Instalación y configuración del sistema

## 9.2. Manual de usuario

# Capítulo 10

# DEFINICIONES Y ABREVIATURAS

K-NEAREST-NEIGHBOR

# Bibliografía


[1] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[2] Zhimin Zhang, Huansheng Ning, Feifei Shi, Fadi Farha, Yang Xu, Jiabo Xu, Fan Zhang, and Kim-Kwang Raymond Choo. Artificial intelligence in cyber security: research advances, challenges, and opportunities. *Artificial Intelligence Review*, 55:1029–1053, 2022. doi: 10.1007/s10462-021-09976-0.

[3] Leonel Andrés Polanía Arias. *Evaluación de modelos de Machine Learning para Sistemas de Detección de Intrusos en Redes IoT*. Proyecto de grado, Universidad de los Andes, 2021.

[4] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[5] Fabien Charmet, Harry Chandra Tanuwidjaja, Solayman Ayoubi, Pierre-François Gimenez, Yufei Han, Houda Jmila, Gregory Blanc, Takeshi Takahashi, and Zonghua Zhang. Explainable artificial intelligence for cybersecurity: a literature survey. *Annals of Telecommunications*, 77:789–812, 2022. doi: 10.1007/s12243-022-00926-7.

[6] Randy Shrefler. *Introduction to Networking with Network+*. Pearson, 2017.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R*. Springer, 2013.

[9] National Institute of Standards and Technology (NIST). Technical guide to information security testing and assessment. Technical Report SP 800-115, National Institute of Standards and Technology, Gaithersburg, MD, 2020. URL https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-115.pdf.

[10] Francisco Charte. A comprehensive and didactic review on multilabel learning software tools. *IEEE Access*, 8:50330–50354, 2020. doi: 10.1109/ACCESS.2020.2979787.

[11] Francisco Herrera, Francisco Charte, Antonio J Rivera, and María J del Jesus. *Multilabel Classification: Problem Analysis, Metrics and Techniques*. Springer, 2016. ISBN 978-3-319-41110-1. doi: 10.1007/978-3-319-41111-8.

[12] Francisco Charte, Antonio Rivera, María José del Jesus, and Francisco Herrera. A first approach to deal with imbalance in multi-label datasets. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 150–160. Springer, 2013. doi: 10.1007/978-3-642-40846-5_16.

[13] Francisco Charte. Cómo analizar la distribución de los datos con R, 2019. URL https://fcharte.com/tutoriales/20170114-DistribucionDatosR/. comprobado en 2020-09-30.

[14] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.