



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**Detección de ataques en un
entorno de red de dispositivos
IoT**



Presentado por Alejandro Diez Bermejo
en Universidad de Burgos — 10 de junio
de 2025

Tutor: Daniel Urda Muñoz
Cotutor: Jaime Andrés Rincón Arango

Índice general

Índice general	3
Índice de figuras	5
Índice de tablas	7
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	12
Apéndice B Especificación de Requisitos	19
B.1. Introducción	19
B.2. Objetivos generales	20
B.3. Catálogo de requisitos	20
B.4. Especificación de requisitos	23
Apéndice C Especificación de diseño	53
C.1. Introducción	53
C.2. Diseño de datos	53
C.3. Diseño arquitectónico	57
C.4. Diseño procedimental	62
Apéndice D Documentación técnica de programación	65
D.1. Introducción	65
D.2. Estructura de directorios	65
D.3. Manual del programador	69

D.4. Compilación, instalación y ejecución del proyecto	70
D.5. Pruebas del sistema	73
Apéndice E Documentación de usuario	75
E.1. Introducción	75
E.2. Requisitos de usuarios	75
E.3. Instalación	75
E.4. Manual del usuario	76
Apéndice F Anexo de sostenibilización curricular	93
F.1. Introducción	93
F.2. Reflexión sobre la sostenibilidad en el proyecto	93
F.3. Competencias de sostenibilidad adquiridas	94
F.4. Conclusiones	95
Bibliografía	97

Índice de figuras

A.1. Gráfico temporal del Sprint 1	3
A.2. Gráfico temporal del Sprint 2	4
A.3. Gráfico temporal del Sprint 3	5
A.4. Gráfico temporal del Sprint 4	6
A.5. Gráfico temporal del Sprint 5	7
A.6. Gráfico temporal del Sprint 6	9
A.7. Gráfico temporal del Sprint 7	10
A.8. Gráfico temporal del Sprint 8	12
 B.1. Diagrama de casos de uso	24
 C.1. Diagrama de flujo de datos del simulador	55
C.2. Diagrama de flujo de datos del aprendizaje federado	56
C.3. Arquitectura general del simulador	59
C.4. Diagrama de clases	61
C.5. Diagrama de secuencia de un flujo en el simulador	63
C.6. Diagrama de secuencia del aprendizaje federado	64
 E.1. Logo del Simulador IoT	76
E.2. Vista general del Simulador IoT	77
E.3. Vista de la barra de menús	77
E.4. Vista del lienzo con clic derecho	79
E.5. Vista de la modal para añadir un nodo	80
E.6. Vista del panel para modificar un nodo	81
E.7. Vista de la modal para eliminar un nodo	81
E.8. Vista del panel para conectar un nodo	82
E.9. Vista de la modal para editar una conexión	83
E.10. Vista del panel para enviar un comando	84

E.11.Vista de la modal para visualizar un paquete	85
E.12.Vista del panel para enviar un ataque	86
E.13.Vista de la detección de un ataque DoS	87

Índice de tablas

A.1. Coste de personal	13
A.2. Coste de hardware	14
A.3. Coste adicional	14
A.4. Coste total del proyecto	15
A.5. Licencias de herramientas	17
A.6. Licencias de recursos	18
B.1. CU-1 Diseño de red	25
B.2. CU-2 Añadir nodos a la red	26
B.3. CU-3 Eliminar nodos de la red	27
B.4. CU-4 Modificar propiedades del nodo	28
B.5. CU-5 Conectar los nodos	29
B.6. CU-6 Modificar propiedades de la conexión	30
B.7. CU-7 Importación configuración de red	31
B.8. CU-8 Guardar configuración de red	31
B.9. CU-9 Simulación de flujos	32
B.10.CU-10 Ejecutar un comando	33
B.11.CU-11 Lanzar un ataque	34
B.12.CU-12 Interceptar flujos	35
B.13.CU-13 Registros de flujos	35
B.14.CU-14 Importar biblioteca de comandos, ataques e interceptores	36
B.15.CU-15 Eliminar biblioteca importada	37
B.16.CU-16 Análisis de flujos en la red	38
B.17.CU-17 Importar modelo entrenado	39
B.18.CU-18 Eliminar modelos importados	40
B.19.CU-19 Seleccionar modelo de detección	41
B.20.CU-20 Ver predicciones de ataques	42
B.21.CU-21 Configurar el simulador	43

B.22.CU-22 Ejecutar entrenamiento federado	44
B.23.CU-23 Cargar configuración del modelo base	45
B.24.CU-24 Dividir dataset para entrenamiento federado	45
B.25.CU-25 Cargar dataset	46
B.26.CU-26 Iniciar servidor federado	47
B.27.CU-27 Entrenamiento del modelo global	48
B.28.CU-28 Evaluar modelo global	49
B.29.CU-29 Exportar modelo global	49
B.30.CU-30 Iniciar cliente federado	50
B.31.CU-31 Entrenar modelo	51
B.32.CU-32 Evaluar modelo	51
B.33.CU-33 Visualizar métricas	52
D.1. Argumentos para el script attack_detector	71

Apéndice A

Plan de Proyecto Software

A.1. Introducción

Este apéndice tiene como objetivo detallar el plan de desarrollo del proyecto abordado en el trabajo de fin de grado. Se proporciona una visión general de los aspectos claves relacionados con la planificación, organización, estimación de recursos, cumplimiento de la legalidad, entre otros aspectos esenciales para un correcto desarrollo.

Este apartado se va a dividir en dos secciones fundamentales:

- **Planificación temporal:** se desarrolla la distribución de tareas durante los meses que ha durado el proyecto.
- **Estudio de viabilidad:** se analizan la estimación de los costes y restricciones relacionados con el proyecto.
 - **Viabilidad económica:** se analizan los costes asociados al desarrollo, implementación y posible mantenimiento del software, además de una estimación de los posibles beneficios.
 - **Viabilidad legal:** se evalúan las licencias de los distintos recursos utilizados además de aspectos relativos a la protección de datos.

A.2. Planificación temporal

Tal como se expuso en la memoria, el desarrollo del proyecto se basó en la metodología ágil **SCRUM**. Si bien no se implementó en su totalidad debido

a que es un trabajo educativo, se procuró simular un entorno de desarrollo realista aplicando los principios fundamentales de esta metodología:

- Desarrollo incremental a través de iteraciones (sprints).
- La duración de los sprints¹ fue de dos semanas.
- Tras la finalización del sprint:
 - Se entregaba un evolutivo funcional.
 - Se realizaba una revisión de esta entrega, mediante reuniones o correos electrónicos.
 - Se planificaba la siguiente iteración, desarrollando cada una de las historias de usuario y estimando las tareas a realizar mediante puntos de estimación siguiendo la serie de Fibonacci.
- Todas las tareas se organizaban en un tablero canvas, utilizando la herramienta GitHub Projects.

Sprint 1 (27/01/2025 - 09/02/2025)

Este sprint marcó el inicio oficial del proyecto, organizando toda la planificación previa realizada durante el mes de enero. Se definieron y completaron tareas fundamentales tanto a nivel de investigación como de diseño e implementación inicial, sentando las bases sobre las cuales se desarrollará el resto del sistema.

- **Investigación inicial sobre aprendizaje federado**
 - **Investigación sobre fundamentos:** lectura de recursos sobre aprendizaje federado.
 - **Investigación sobre implementación:** exploración de herramientas, frameworks y técnicas para la implementación de un aprendizaje distribuido.
- **Estructura básica de la interfaz del simulador**
 - **Inicialización del proyecto Angular:** configuración inicial del entorno Angular, instalación de dependencias y organización de la estructura base para el desarrollo del simulador.

¹La octava iteración duró el doble debido a que era documentación y solución de algunos errores.

- **Diseño de la barra de navegación:** diseño del menú principal del simulador y elección de las opciones que se van a dar.
- **Diseño del espacio de simulación:** diseño del espacio de trabajo donde se diseñará la red.
- **Creación de clases para la gestión de red:** desarrollo de la estructura básica de la lógica del simulador.

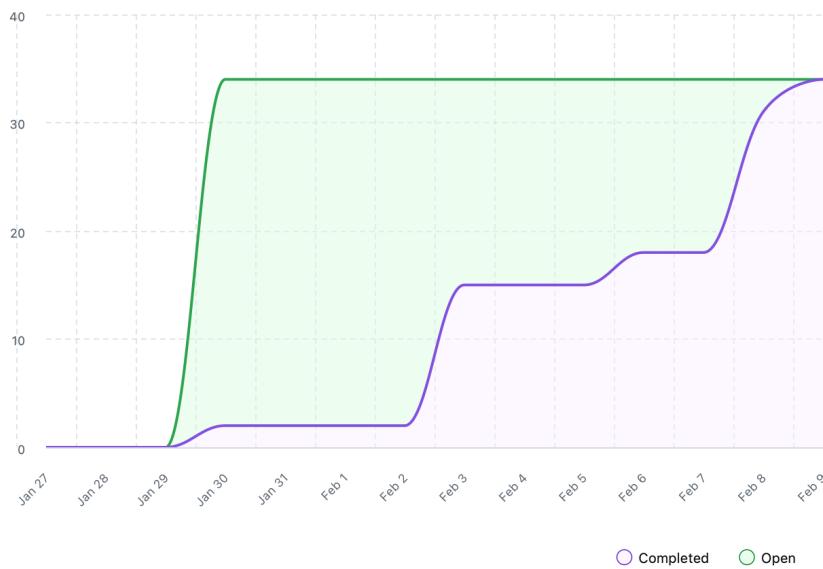


Figura A.1: Gráfico temporal del Sprint 1

Sprint 2 (10/02/2025 - 23/02/2025)

Durante este sprint se avanzó tanto en el desarrollo del entorno de simulación como en la exploración de las posibles técnicas para la implementación del aprendizaje federado.

- **Gestión flexible de nodos en la red**
 - **Configuración de nodo:** se desarrolló la interfaz para la configuración de un nodo al ser seleccionado.
- **Probar y experimentar con las librerías de aprendizaje federado**

- **Realizar una prueba con TensorFlow Federated:** se realizó una prueba básica usando TensorFlow Federated, con el objetivo de aprender de la herramienta y ver la viabilidad en el proyecto.
- **Realizar una prueba con Flower:** se realizó una prueba básica usando Flower, con el objetivo de aprender de la herramienta y ver la viabilidad en el proyecto.

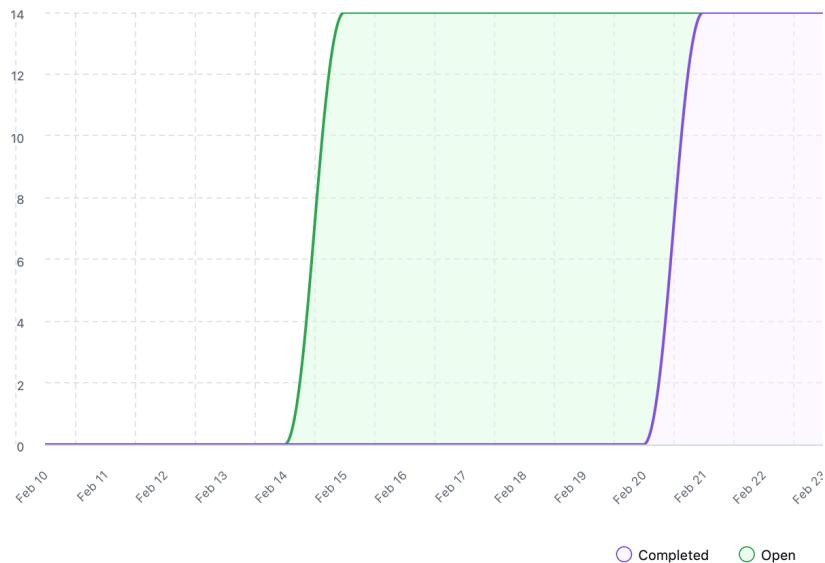


Figura A.2: Gráfico temporal del Sprint 2

Sprint 3 (24/02/2025 - 09/03/2025)

Esta iteración se centró en el simulador de redes IoT. El objetivo principal fue permitir una iteración básica entre los dispositivos, además de añadir utilidades útiles para el desarrollo de los siguientes sprints.

- **Gestión flexible de nodos en la red**
 - **Añadir y eliminar nodos:** implementación de la funcionalidad de añadir y eliminar nodos para la creación de topologías de red.
 - **Exportación e importación de configuración:** desarrollo de la posibilidad de importar y exportar proyectos.
- **Simulador de red**

- **Definición de nodo:** se desarrolló la lógica interna del nodo, añadiendo las propiedades y métodos necesarios.
- **Implementación de dispositivo:** se desarrolló la lógica interna que representaría un dispositivo en la red.
- **Implementación del paquete:** se desarrolló el modelo de lo que sería un paquete de red.
- **Implementación del router:** se desarrolló la lógica interna que representaría un router en la red.
- **Servicio de gestión de red:** se desarrolló un servicio centralizado, que se encargaría de coordinar el estado global de la red.

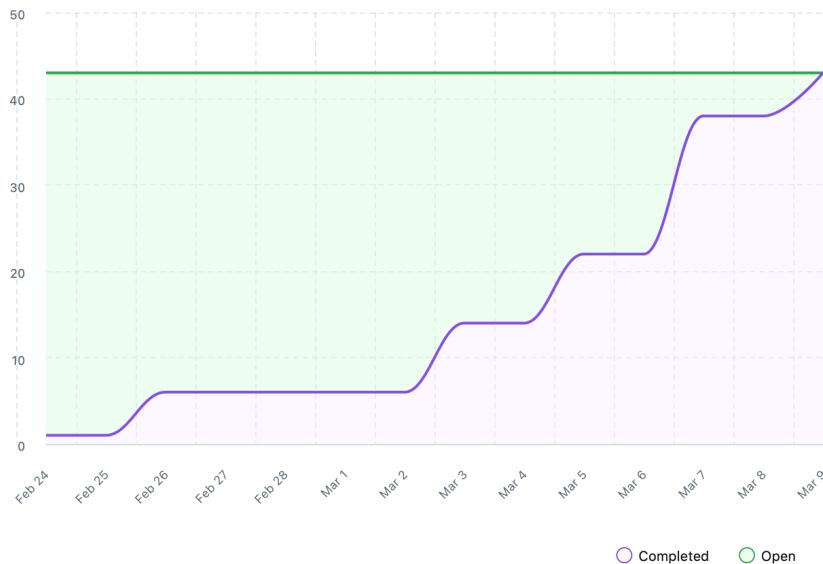


Figura A.3: Gráfico temporal del Sprint 3

Sprint 4 (10/03/2025 - 23/03/2025)

En esta iteración, el foco principal fue el desarrollo del script para el entrenamiento del modelo de inteligencia artificial mediante aprendizaje federado. Además, se realizó una gran reestructuración del código de la parte del simulador.

- **Implementación de aprendizaje federado con Flower**

- **Cargar los datasets desde un archivo:** se añadió la funcionalidad para cargar los datasets desde archivos csv.
- **Cargar modelo de TensorFlow:** se añadió la funcionalidad para cargar la configuración del modelo TensorFlow para que sea entrenado.
- **Configuración de argumentos por consola:** se añadió la funcionalidad de establecer argumentos a la hora de iniciar el script, para seleccionar que se desea hacer.
- **Crear cliente de aprendizaje federado:** se desarrolló el cliente del aprendizaje federado, permitiendo entrenar el modelo que recibe.
- **Crear servidor de aprendizaje federado:** se desarrolló el servidor del aprendizaje federado, permitiendo distribuir el modelo para que lo entrenen los clientes y después realizar una combinación de los modelos.
- **Función para dividir los datasets:** se creó una función que permitiera dividir los datasets entre clientes diferentes.
- **Refactorización de código:** se reestructuro el código del simulador para mejorar su estructura, legibilidad y mantenibilidad.

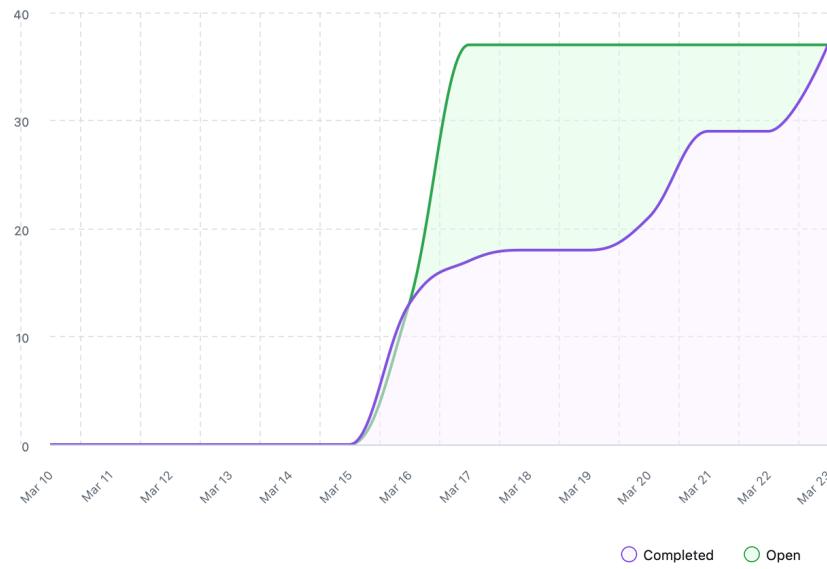


Figura A.4: Gráfico temporal del Sprint 4

Sprint 5 (24/03/2025 - 06/04/2025)

Durante el quinto sprint se consolidaron varias funcionalidades de la simulación del flujo de red y la interacción entre dispositivos. Además, se realizaron pequeñas optimizaciones y correcciones en la interfaz general.

- Mejoras en el simulador IoT

- **Generar un generador de flujo de red:** se implementó la clase que permitiría realizar flujos sobre la red de dispositivos.
- **Permitir la conexión con nodos:** desarrollo de la funcionalidad que permite a los dispositivos establecer enlaces entre ellos.
- **Poder realizar ping entre dispositivos:** implementación del comando ping entre nodos, permitiendo comprobar la conectividad entre los nodos.
- **Refactorizar interfaz de paquete:** modificar el modelo de paquete, para poder distinguir entre varios tipos de paquetes y obtener un código más estructurado.
- **Pequeñas mejoras:** conjunto de cambios menores y solución de errores.

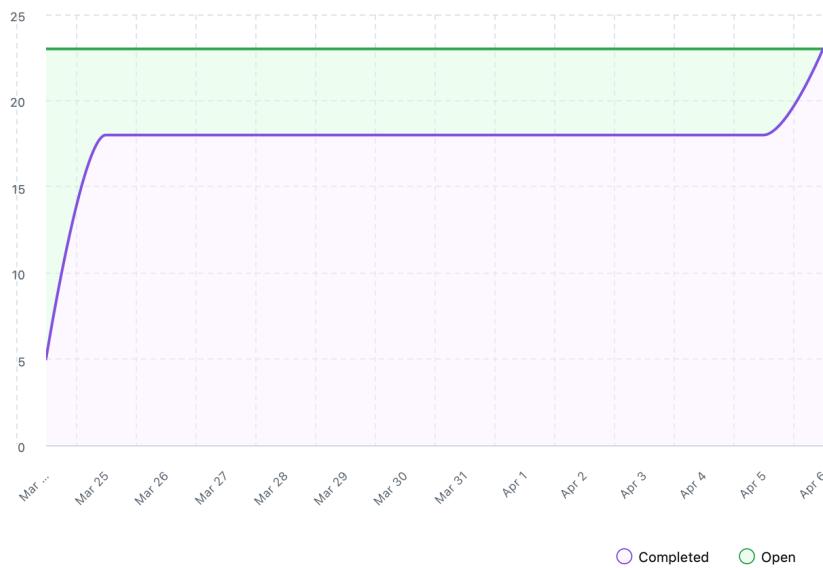


Figura A.5: Gráfico temporal del Sprint 5

Sprint 6 (07/04/2025 - 20/04/2025)

Esta iteración fue una de las más intensivas. Se incluyeron funcionalidades relacionadas con la simulación y la generación de ataques. Además, se realizaron grandes tareas de refactorización de componentes clave del simulador, ya que añadieron nuevas funcionalidades y así se pudo mejorar la comprensión del código.

■ Mejoras en el simulador IoT

- **Guardar estado de la red:** guarda el estado actual de la red, para que una vez se recargue la página web, no se pierda toda la configuración.

■ Mejoras en el aprendizaje federado

- **Exportación de modelo:** se añadió la funcionalidad de exportar el modelo generado durante el entrenamiento.
- **Limpieza de código:** revisión general del código para eliminar redundancias y mejorar la legibilidad.
- **Más métricas:** se ampliaron las métricas disponibles para evaluar el modelo.

■ Implementación Phantom Attacker

- **Crear la clase Phantom Attacker:** implementación de la clase atacante para lanzar ataques.
- **Lanzar ataques:** se desarrolló la lógica de lanzar ataques desde la interfaz.
- **Cargar un archivo JS:** implementación de dar la funcionalidad al usuario de cargar archivos JavaScript que contienen ataques.
- **Obtener lista de ataques del usuario:** análisis del script del usuario para obtener los ataques que ha realizado y mostrarlo en la interfaz gráfica.
- **Generar un archivo con Ataque DoS:** se desarrolló un script que simula un ataque de denegación de servicio (DoS).
- **Refactorización de la parte de generador de flujo e interceptor:** se reorganizó la forma en la que se interceptan paquetes y se genera tráfico, dividiéndolo en dos clases distintas.

■ Mejoras Simulador

- **Mejoras en código y rendimiento:** cambios orientados a optimizar el uso de la aplicación y la legibilidad del código.

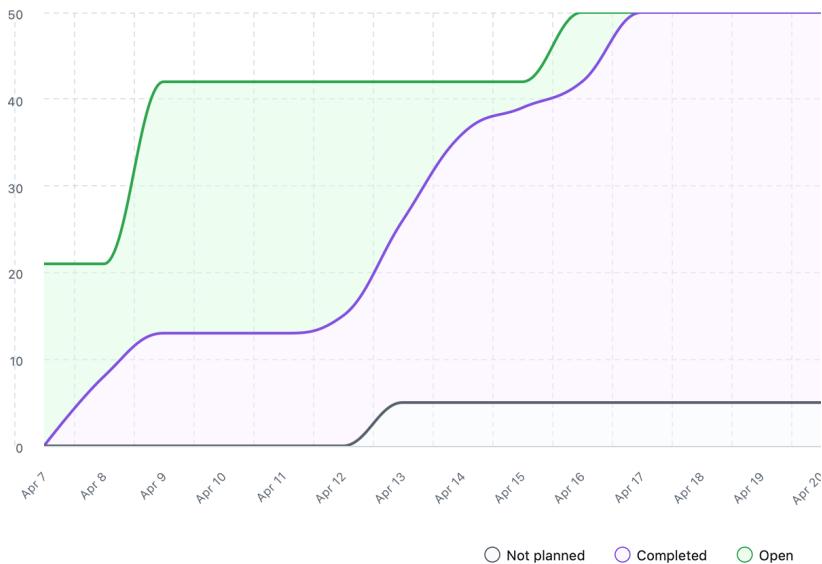


Figura A.6: Gráfico temporal del Sprint 6

Sprint 7 (21/04/2025 - 04/05/2025)

En este sprint, se integraron las funcionalidades finales necesarias para cerrar el desarrollo del simulador. Fue una de las iteraciones con más complejidad, ya que se debía implementar los modelos de inteligencia artificial en el simulador. De hecho, no se logró completar el sprint y se tuvo que dejar alguna tarea para el siguiente sprint.

■ Implementación Cyber Shield

- **Generar clase CyberShield:** creación de la clase principal del sistema de detección de ataques.
- **Cargar modelos:** implementación de un sistema de carga dinámica de modelos TensorFlow.
- **Análisis de trazas de red:** desarrollo la función que permita obtener las trazas de red para analizarla con el modelo.

■ Mejoras simulador

- **Añadir más comandos e interceptores:** incorporación de nuevos comandos e interceptores.
- **Funciones de configuración restantes:** añadir las funciones pendientes relacionadas con la configuración de la interfaz.
- **Mejora en la gestión de modelos de TensorFlow:** refactorización del sistema de gestión de modelos.
- **Nuevas funcionalidades para mayor realismo:** añadir la posibilidad de latencia variable a la hora de enviar un paquete.
- **Permitir editar una conexión:** añadir la funcionalidad de editar los parámetros de las conexiones.
- **Refactorización de servicios:** reorganización de los servicios de la aplicación.
- **Traducción:** implementación de un sistema que permita cambiar el idioma de la aplicación.

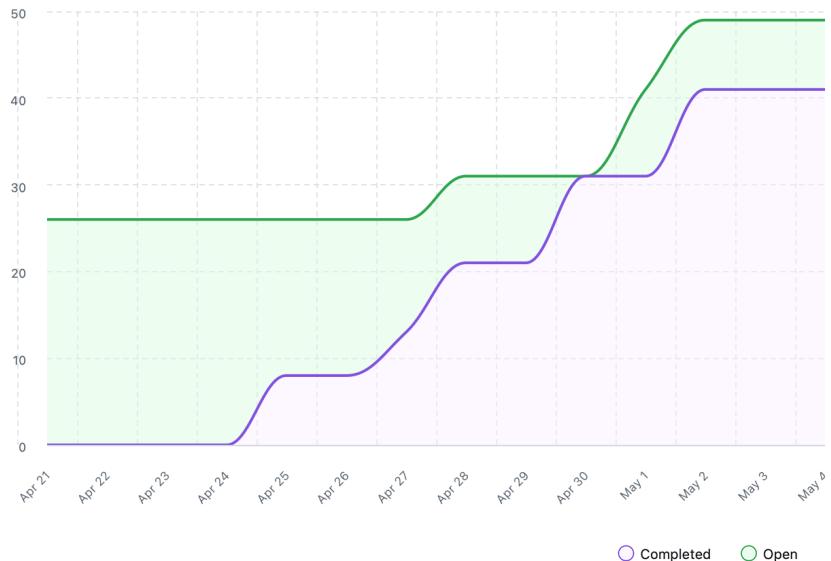


Figura A.7: Gráfico temporal del Sprint 7

Sprint 8 (05/05/2025 - 08/06/2025)

Este fue el último sprint y tuvo como objetivo principal finalizar las tareas de desarrollo y redactar toda la documentación del proyecto. Durante esta

fase, se corrigieron diversos errores detectados durante la fase de pruebas, se añadieron pequeñas mejoras al sistema y se documentó todo el proyecto. La duración de este sprint fue excepcionalmente prolongada, ya que, al tratarse del último, se decidió agrupar en él tanto la resolución de errores pendientes como la elaboración completa de la documentación.

- Mejoras en el aprendizaje federado

- **Generar gráficas para métricas:** desarrollo de visualizaciones para representar el rendimiento del modelo de forma gráfica.

- Mejoras simulador

- **Añadir la posibilidad del ancho de banda de una conexión:** mejora que permite implementar más realismo al simulador.
 - **Permitir ver el paquete completo:** permite ver el paquete con todos los campos que tiene.
 - **Solución de errores en la interfaz:** se solucionan errores detectados en la interfaz.

- Memoria del Trabajo de Fin de Grado

- **Introducción:** redacción de la introducción de la memoria.
 - **Objetivos del proyecto:** redacción de los objetivos del proyecto.
 - **Conceptos teóricos:** redacción de los conceptos teóricos usados durante el proyecto.
 - **Técnicas y Herramientas:** redacción de las técnicas y herramientas usadas durante el proyecto.
 - **Aspectos relevantes del desarrollo del proyecto:** redacción de los aspectos más relevantes durante el desarrollo del proyecto.
 - **Trabajos relacionados:** redacción de los trabajos relacionados durante el desarrollo del proyecto.
 - **Conclusiones y Líneas de trabajo futuras:** redacción de las conclusiones y líneas futuras relacionados con el proyecto.

- Anexo del Trabajo de Fin de Grado

- **Plan de Proyecto Software:** redacción del plan de proyecto.
 - **Especificación de requisitos:** redacción de los requisitos del proyecto.

- **Especificación de diseño:** redacción de la especificación de diseño del proyecto.
- **Documentación técnica de programación:** redacción de la documentación técnica de programación.
- **Documentación de usuario:** redacción de la documentación de usuarios.
- **Anexo de sostenibilización curricular:** redacción del anexo de sostenibilización curricular.

Además, aunque no se refleje explícitamente como tareas, durante este sprint se llevaron a cabo todas las pruebas necesarias para garantizar el correcto funcionamiento del simulador. También se realizó la subida y despliegue de la versión final de producción del simulador.

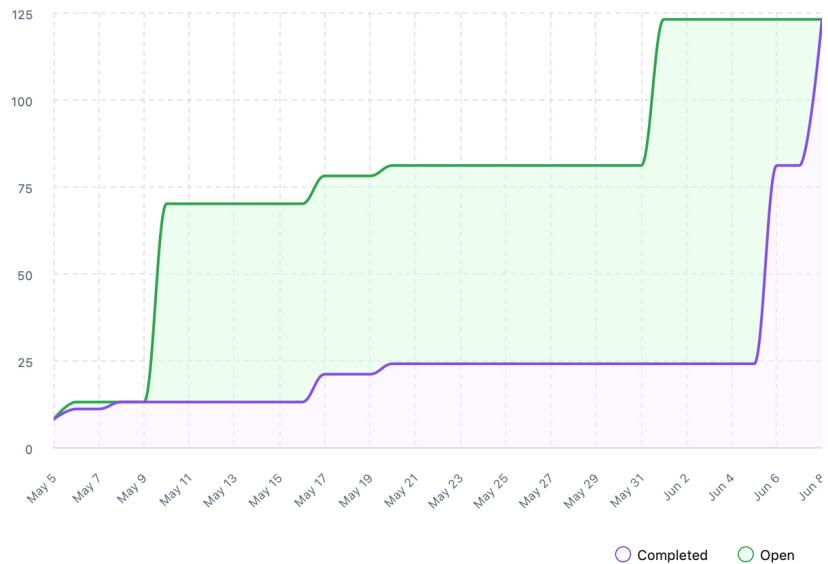


Figura A.8: Gráfico temporal del Sprint 8

A.3. Estudio de viabilidad

La viabilidad de un proyecto de software se refiere a su capacidad para ser completado con éxito y cumplir con los objetivos establecidos.

Viabilidad económica

Para determinar si el proyecto es viable en términos económicos, analizaremos los costes de desarrollo y los beneficios que se obtendrían en caso de comercialización.

Costes

En cuanto a los costes, se pueden desglosar en distintos subapartados:

- **Personal:** este apartado abarca los costes relacionados con la contratación de personal para el desarrollo del proyecto. Se ha estimado que, durante estos 5 meses de trabajo, se necesitará aproximadamente 480 horas de trabajo. Suponiendo que el salario bruto anual en 2025 de un ingeniero junior está en torno a 25.625 €², lo que equivale a aproximadamente 13,14 €/hora, aplicando las cotizaciones dentro del Régimen General de la Seguridad Social [1], el coste total sería:

Concepto	Coste
Salario bruto por hora	13,14 €
Contingencias comunes	3,10 €
Desempleo	0,72 €
Fondo de Garantía Salarial	0,03 €
Formación	0,08 €
Mecanismo de Equidad Intergeneracional	0,09 €
Coste por hora	17,16 €
Total 480 horas	8.235,31 €

Tabla A.1: Coste de personal

- **Hardware:** este apartado abarca los costes relacionados con todo el hardware que se ha necesitado para el desarrollo del proyecto. Su impacto depende especialmente del número de clientes utilizados para el entrenamiento federado. En este caso, se emplearon únicamente dos Raspberry Pi 4, valoradas en 60,00 € cada una. Dado que fueron adquiridas hace seis años, y se considera que su vida útil ya se ha agotado, se entienden como completamente amortizadas, por lo que

²Salario programador junior: <https://es.talent.com/salary?job=ingeniero+junior>

su coste es de 0 €. Por otro lado, el servidor utilizado tanto para el entrenamiento federado como para el desarrollo del simulador ha sido un MacBook Pro valorado en 2.200 €. Este equipo fue adquirido hace tres años, y se considera que tiene una vida útil de seis años, por lo que se aplica una amortización lineal de 30,56 €/mes.

Concepto	Coste	Coste amortizado
Raspberry Pi 4	60,00 €	0,00 €
Raspberry Pi 4	60,00 €	0,00 €
MacBook Pro	2.200,00 €	30,56 €
Coste por mes		30,56 €
Total del proyecto		152,78 €

Tabla A.2: Coste de hardware

- **Software:** como todo el software utilizado es de código abierto y gratuito, no hay costes asociados a licencias de uso.
- **Otros:** este apartado abarca los costes adicionales para el desarrollo del proyecto, como consumo eléctrico, internet, ...

Concepto	Coste
Consumo eléctrico por mes	13,46 €
Internet por mes	16,24 €
Coste por mes	29,70 €
Total del proyecto	148,50 €

Tabla A.3: Coste adicional

Teniendo en cuenta todos los costes calculados, el coste total del proyecto asciende a:

Tipo de coste	Coste
Personal	8.235,31 €
Hardware	152,78 €
Software	0,00 €
Adicional	148,50 €
Total	8.536,59 €

Tabla A.4: Coste total del proyecto

Beneficios

El software creado se distribuye de forma gratuita y sin publicidad, por lo que se necesita una fuente de ingresos alternativa para poder monetizar el proyecto.

- **Donaciones:** se les podría dar la opción a los usuarios de que apoyen el proyecto mediante donaciones.
- **Subvenciones:** se puede establecer acuerdos con instituciones académicas o empresas, para que apoyen el proyecto.
- **Modelo freemium:** consiste en establecer una versión básica gratuita, que ofrezca funciones avanzadas disponibles mediante una suscripción.
- **Licencias comerciales:** aunque el proyecto es de código abierto, se podría contemplar la opción de establecer una licencia para uso comercial.

Viabilidad legal

El proyecto debe cumplir con varias normativas legales, especialmente las relacionadas con las licencias y la protección de datos.

Protección de Datos

Este proyecto no usa directamente datos personales de usuarios, ya que el simulador de red se ejecuta de forma local en los dispositivos de los usuarios y nunca envía información fuera. Sin embargo, es importante recalcar que para el entrenamiento de la red neuronal mediante aprendizaje federado es necesario tener el acceso a flujos de red, los cuales obtienen información sensible y pudiera comprometer la privacidad del usuario.

Aunque estos datos no salen del dispositivo, el usuario que realice el entrenamiento debe plantearse los riesgos asociados al uso de la herramienta y por lo tanto debe garantizar el cumplimiento de la normativa referente a la protección de datos en el país en el que se encuentre.

Además, se recomienda que los datos utilizados sean lo más anónimos posibles y que cumplan normas básicas de seguridad para evitar posibles fugas de información o uso ilícito de estos.

Licencias de Software

Para el desarrollo de este proyecto se han utilizado diversas herramientas y bibliotecas, cada una con su respectiva licencia.

Herramienta	Versión	Licencia
Visual Studio Code	1.100.2	MIT
Chromium	137.0.7127.0	BSD
Google Colab	N/A	Propietaria
Git	2.39.5	GPLv2
Python	3.11.0	PSF
TensorFlow	2.18.0	Apache 2.0
Keras	3.9.2	Apache 2.0
Flower	1.18.0	Apache 2.0
TensorFlow Federated	0.86.0	Apache 2.0
TensorFlowJS	4.22.0	Apache 2.0
Numpy	2.1.3	BSD
Scikit Learn	1.6.1	BSD
Matplotlib	3.10.1	PSF
Seaborn	0.13.2	BSD
Argparse	1.4.0	PSF
Node	22.13.1	MIT
JavaScript	ES2022	ECMAScript
TypeScript	5.8.3	Apache 2.0
Angular	19.2.6	MIT
RxJS	7.8.2	Apache 2.0
TensorFlow.js	4.22.0	Apache License 2.0
js-yaml	4.1.0	MIT
jszip	3.10.1	MIT
Tailwind CSS	3.4.17	MIT
Spartan UI	0.0.1-alpha.439	MIT
Lucide Icons	31.2.0	ISC
GitHub	N/A	Propietaria
LaTeX	2024	LPPL
Overleaf	N/A	Propietaria
Draw.io	27.0.6	Apache 2.0
Zotero	7.0.15	AGPL v3

Tabla A.5: Licencias de herramientas

Como se puede observar, la mayoría de las herramientas utilizadas en el desarrollo de este proyecto usan licencias como MIT, Apache 2.0, BSD e ISC. Estas licencias son ampliamente compatibles entre sí y no imponen restricciones significativas sobre el uso.

Debido a esta compatibilidad de licencias, se ha optado por escoger la licencia **MIT** para este proyecto, permitiendo su libre uso y modificación, con la condición de mantener la atribución correspondiente.

Licencias de Recursos

Para el desarrollo de este proyecto se han utilizado diversos recursos, cada una con su respectiva licencia.

Recursos	Licencia
Dataset NF-ToN-IoT	CC BY-NC-SA 4.0
Iconos de imágenes (Flaticon - Vectorslab)	Propia

Tabla A.6: Licencias de recursos

Como se puede observar, el dataset usa la licencia Creative Commons Atribución/Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional por lo que el uso del modelo entrenado con este dataset, no puede usarse para fines comerciales.

En cuanto a la documentación del proyecto, todas las imágenes que existen son de creación propia, aunque algunas usan iconos que son pertenecientes a Vectorslab, distribuidos en la plataforma Flaticon. Como se le atribuye reconocimiento, se pueden usar con fines comerciales.

Debido a esta compatibilidad de licencias, se ha optado por escoger la licencia **MIT** para la documentación de este proyecto, permitiendo su libre uso y modificación, con la condición de mantener la atribución correspondiente.

Apéndice B

Especificación de Requisitos

B.1. Introducción

Esta sección ofrece una visión general del proyecto, incluyendo su propósito, público objetivo y uso previsto del software desarrollado.

Propósito del producto

El propósito de este software es proporcionar un entorno de pruebas para diferentes modelos de inteligencia artificial orientados a la detección de ataques. La herramienta permite a los usuarios diseñar diferentes tipologías de red, y evaluar los modelos desarrollados para la identificación de ataques.

Además, ofrece la posibilidad de entrenar un modelo desarrollado en TensorFlow mediante el uso de aprendizaje federado.

Valor del producto

Este producto es de gran utilidad para el proceso de desarrollo y validación de modelos de inteligencia artificial en un entorno controlado, específico y realista sin la necesidad de invertir recursos en desarrollar una infraestructura costosa y que implique un alto consumo de tiempo.

Se podría decir que es una solución ideal como primer paso en la evaluación de modelos antes de desplegarlos en entornos reales, pudiendo así identificar errores o aspectos a mejorar. También se podría usar como demostración del funcionamiento de un modelo ante diferentes tipos de flujos sobre la red.

Público objetivo

Este proyecto está diseñado para investigadores en ciencia de datos y seguridad informática, así como para personas apasionadas de la inteligencia artificial que buscan un entorno controlado y accesible para probar y desarrollar modelos, sin la necesidad de una infraestructura compleja.

Uso previsto

La aplicación está diseñada para ser intuitiva y accesible, permitiendo usarse sin necesidad de registro ni de conocimientos avanzados.

B.2. Objetivos generales

Este proyecto se ha desarrollado cumpliendo una serie de objetivos básicos:

- Implementación de una **arquitectura federada** para el entrenamiento de modelos desarrollados con TensorFlow.
- Desarrollo de un software capaz de **simular el flujo de red entre dispositivos conectados**, siendo flexible y adaptable a cada necesidad.
- **Detección de ataques** generados dentro del propio entorno de simulación, mediante el uso de modelos de redes neuronales.
- Posibilidad para los usuarios de **crear sus propios ataques** y definir **flujos** de red personalizados.
- Opción de cargar **modelos propios** para el análisis de trazas de red.

B.3. Catálogo de requisitos

Requisitos funcionales

- **RF-1 Diseño de red:** el usuario debe poder configurar una topología de red de manera sencilla y flexible.
 - **RF-1.1 Añadir nodos:** el usuario debe poder añadir el número indefinido de nodos a la red.

- **RF-1.2 Eliminar nodos:** el usuario debe poder eliminar los nodos de la red.
 - **RF-1.3 Modificar nodos:** el usuario debe poder modificar cada una de las propiedades de los nodos según el diseño lo permita.
 - **RF-1.4 Conectar nodos:** el usuario debe poder conectar los nodos entre sí.
 - **RF-1.5 Modificar conexiones:** el usuario debe poder modificar las propiedades de las conexiones.
 - **RF-1.6 Importar configuración:** el usuario debe ser capaz de importar configuraciones guardadas.
 - **RF-1.7 Guardar configuración:** el usuario debe ser capaz de guardar topologías de red diseñadas.
- **RF-2 Simulación de flujos:** el usuario debe poder simular flujos de red y que estos estén registrados.
 - **RF-2.1 Ejecutar un comando:** el usuario debe poder ejecutar un comando desde un nodo a uno o varios nodos al mismo tiempo.
 - **RF-2.2 Ejecutar un ataque:** el usuario debe poder ejecutar un ataque desde un nodo de tipo ordenador a uno o varios nodos al mismo tiempo.
 - **RF-2.3 Interceptar un flujo:** el nodo debe de ser capaz de interceptar un flujo enviado por otro nodo.
 - **RF-2.4 Registro de flujos:** cada nodo debe tener un registro de los flujos que han pasado sobre este.
 - **RF-2.5 Importar biblioteca:** el usuario debe poder importar una biblioteca de comandos, ataques e interceptores y que aparezcan como disponibles en los nodos.
 - **RF-2.6 Eliminar biblioteca:** el usuario debe poder eliminar la biblioteca que ha sido importada.
 - **RF-3 Análisis de flujos:** el usuario debe poder analizar los flujos de datos en la red.
 - **RF-3.1 Importar modelo entrenado:** el usuario debe ser capaz de importar uno o varios modelos entrenados con sus correspondientes scripts para su uso en el simulador.
 - **RF-3.2 Eliminar modelo entrenado:** el usuario debe ser capaz de eliminar los modelos importados del simulador.

- **RF-3.3 Selección de modelo a usar:** el usuario debe ser capaz de seleccionar en cada conexión que modelo se debe usar para detectar los ataques.
- **RF-3.4 Predicciones de ataques:** el simulador debe ser capaz de mostrar al usuario las predicciones que proporciona el modelo tras pasar las trazas de red.
- **RF-4 Configuración del simulador:** el usuario debe poder configurar todas las opciones de visualización del simulador, así como tener la opción de deshacer y rehacer cambios.
- **RF-5 Entrenamiento federado:** el usuario debe ser capaz de entrenar un modelo de TensorFlow mediante aprendizaje federado.
 - **RF-5.1 Cargar configuración del modelo base:** el usuario debe ser capaz de cargar la configuración del modelo base a cada cliente y servidor antes del entrenamiento federado.
 - **RF-5.2 Dividir dataset:** el usuario debe ser capaz de dividir un dataset en partes iguales para su uso en aprendizaje federado.
 - **RF-5.3 Cargar dataset:** el usuario debe ser capaz de cargar los dataset necesarios a cada cliente y servidor antes del entrenamiento federado.
 - **RF-5.4 Servidor federado:** el usuario debe ser capaz de lanzar un servidor federado con posibles configuraciones para coordinar el entrenamiento distribuido.
 - **RF-5.4.1 Entrenamiento modelo global:** el servidor debe ser capaz de organizar el entrenamiento global del modelo.
 - **RF-5.4.2 Evaluación modelo global:** el servidor debe ser capaz de organizar la evaluación global del modelo.
 - **RF-5.4.3 Exportar modelo global:** el servidor debe ser capaz de exportar el modelo global al finalizar el entrenamiento.
 - **RF-5.5 Cliente federado:** el usuario debe ser capaz de lanzar varios clientes federados con posibles configuraciones desde distintos equipos y conectarlos al servidor.
 - **RF-5.5.1 Entrenamiento modelo:** los clientes deben ser capaces de entrenar el modelo global.
 - **RF-5.5.2 Evaluación modelo:** los clientes deben ser capaces de evaluar el modelo global.

- **RF-5.6 Métricas:** el usuario debe ser capaz de visualizar las métricas del entrenamiento y la evaluación.

Requisitos no funcionales

- **RNF-1 Usabilidad:** la interfaz debe de ser intuitiva y accesible para usuarios no expertos, además de no requerir configuraciones complejas para su uso.
- **RNF-2 Rendimiento y confiabilidad:** las simulaciones deben de ejecutarse en tiempos razonables. También es esperable que se gestionen correctamente los errores y que en caso de actualización de la página no se pierda el estado de la aplicación.
- **RNF-3 Compatibilidad:** debe de ser compatible con la mayoría de los navegadores modernos y no requerir de funciones especiales que sean necesaria la configuración.
- **RNF-4 Seguridad:** debe de ser un software estable ante posibles fallos que cometa el usuario, además de no comprometer al sistema.
- **RNF-5 Mantenibilidad y escalabilidad:** el código debe de estar bien estructurado y la documentación debe ser clara y concisa.

B.4. Especificación de requisitos

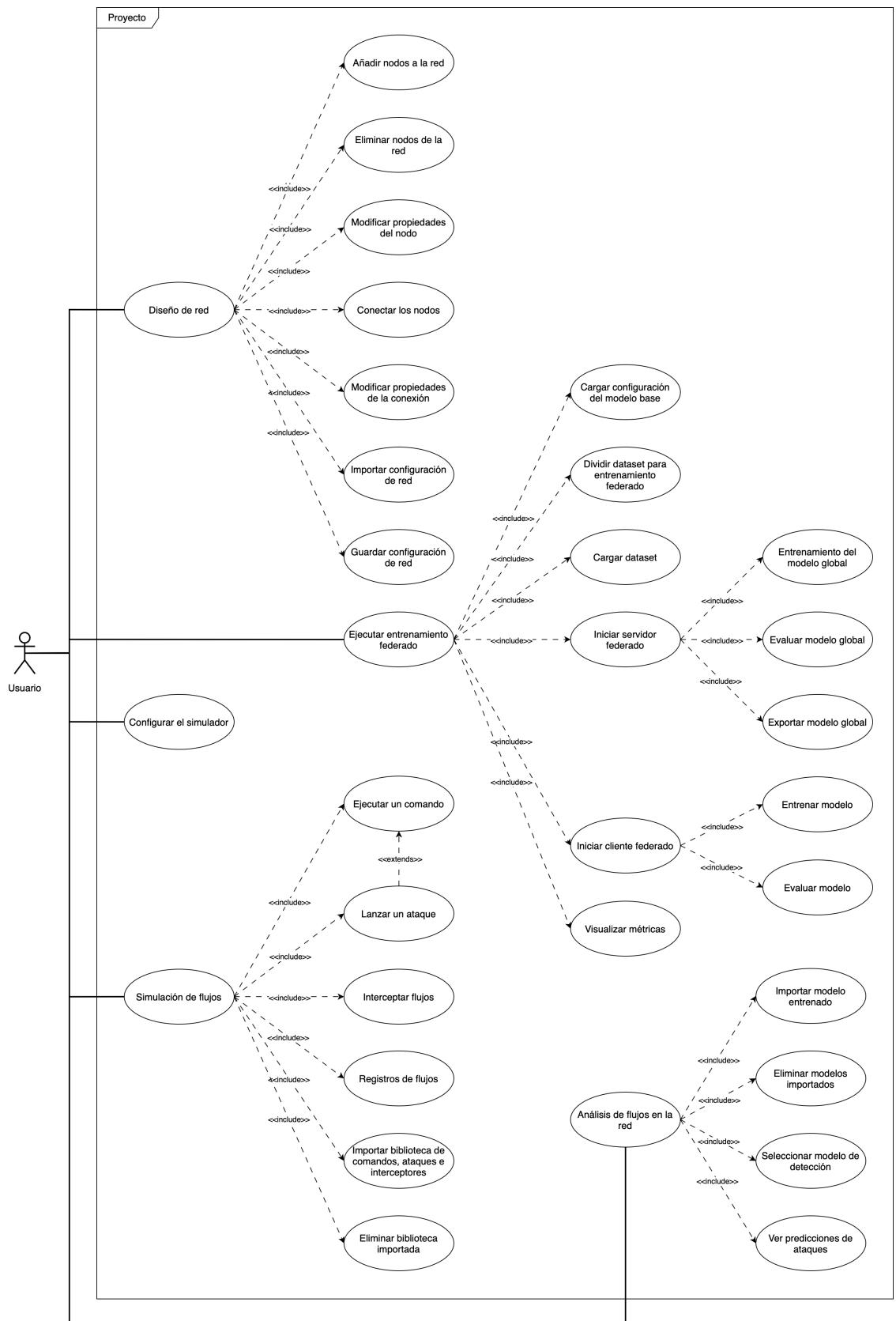


Figura B.1: Diagrama de casos de uso

CU-1	Diseño de red
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-1, RF-1.1, RF-1.2, RF-1.3, RF-1.4, RF-1.5, RF-1.6, RF-1.7
Descripción	El usuario quiere modificar la topología de la red
Precondición	El usuario tiene conexión a internet El simulador está disponible
Acciones	<ol style="list-style-type: none"> 1. El usuario entra al simulador 2. El simulador muestra la interfaz 3. El simulador inicia un proyecto nuevo
Postcondición	Proyecto nuevo
Excepciones	-
Importancia	Alta

Tabla B.1: CU-1 Diseño de red

CU-2	Añadir nodos a la red
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-1.1
Descripción	El usuario añade uno o varios nodos a la red mediante la interfaz gráfica
Precondición	El usuario está dentro del simulador
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción “Añadir router” (en caso de añadir un router) o “Añadir dispositivo” (en caso de añadir un ordenador o dispositivo IoT) 2. El simulador pide al usuario datos del nodo (nombre y tipo) 3. El usuario completa el formulario 4. El simulador añade el nodo a la red
Postcondición	El nodo se ha añadido a la red
Excepciones	Falta algún dato de llenar en el formulario Existe un nodo de tipo router en la red
Importancia	Alta

Tabla B.2: CU-2 Añadir nodos a la red

CU-3	Eliminar nodos de la red
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-1.2
Descripción	El usuario elimina un nodo de la red mediante la interfaz gráfica
Precondición	El usuario está dentro del simulador Existe el nodo a eliminar
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona un nodo a eliminar 2. El simulador solicita la confirmación de eliminación 3. El simulador desconecta el nodo y lo elimina
Postcondición	El nodo y las conexiones de este se eliminan
Excepciones	-
Importancia	Media

Tabla B.3: CU-3 Eliminar nodos de la red

CU-4	Modificar propiedades del nodo
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-1.3
Descripción	El usuario modifica las propiedades de un nodo
Precondición	El usuario está dentro del simulador Existe el nodo a modificar
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona un nodo a modificar 2. El simulador muestra las propiedades del nodo 3. El usuario modifica las propiedades 4. El simulador guarda los cambios
Postcondición	El nodo guarda las modificaciones
Excepciones	Valores no válidos
Importancia	Baja

Tabla B.4: CU-4 Modificar propiedades del nodo

CU-5	Cone&ntilde;r los nodos
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-1.4
Descripción	El usuario conecta dos nodos
Precondición	El usuario está dentro del simulador Existe un dispositivo sin conectar Existe un router al cual conectarse
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona un dispositivo a conectar 2. El simulador muestra la opción de conectarse al router 3. El usuario selecciona la opción de conectarse 4. El simulador conecta los nodos y asigna una IP al dispositivo
Postcondición	El nodo está conectado y tiene una IP asignada
Excepciones	-
Importancia	Alta

Tabla B.5: CU-5 Cone&ntilde;r los nodos

CU-6	Modificar propiedades de la conexión
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-1.5
Descripción	El usuario modifica las propiedades de una conexión
Precondición	El usuario está dentro del simulador Existe una conexión a modificar
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la conexión a modificar 2. El simulador muestra las propiedades de la conexión 3. El usuario modifica las propiedades 4. El simulador guarda los cambios
Postcondición	La conexión guarda las modificaciones
Excepciones	Valores no válidos
Importancia	Media

Tabla B.6: CU-6 Modificar propiedades de la conexión

CU-7	Importación configuración de red
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-1.6
Descripción	El usuario importa una configuración guardada
Precondición	El usuario está dentro del simulador El usuario dispone de un archivo de configuración válido
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de “Abrir” 2. Elige un archivo 3. El simulador carga la configuración
Postcondición	La red se muestra como estaba guardada
Excepciones	Archivo incompatible
Importancia	Baja

Tabla B.7: CU-7 Importación configuración de red

CU-8	Guardar configuración de red
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-1.7
Descripción	El usuario guarda la configuración de red actual
Precondición	El usuario está dentro del simulador
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción de “Guardar” 2. El simulador exporta la configuración de la red actual
Postcondición	El archivo se descarga
Excepciones	-
Importancia	Baja

Tabla B.8: CU-8 Guardar configuración de red

CU-9	Simulación de flujos
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-2, RF-2.1, RF-2.2, RF-2.3, RF-2.4, RF-2.5, RF-2.6
Descripción	El usuario quiere realizar simulaciones de flujos de red
Precondición	El usuario está dentro del simulador Hay una red diseñada Hay nodos conectados
Acciones	<ol style="list-style-type: none"> 1. El usuario entra al simulador 2. El simulador muestra la interfaz 3. El simulador muestra la configuración de red actual
Postcondición	Configuración de red actual
Excepciones	-
Importancia	Alta

Tabla B.9: CU-9 Simulación de flujos

CU-10	Ejecutar un comando
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-2.1
Descripción	El usuario quiere ejecutar un comando desde un nodo hacia otros
Precondición	El usuario está dentro del simulador Hay nodos conectados El simulador dispone de comandos
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona el nodo origen 2. El simulador muestra la vista de tráfico de red 3. El usuario selecciona el comando a enviar 4. El usuario selecciona los objetivos 5. El usuario selecciona la opción de enviar 6. El simulador simula el flujo
Postcondición	Flujo registrado en los nodos
Excepciones	Valores no válidos
Importancia	Alta

Tabla B.10: CU-10 Ejecutar un comando

CU-11	Lanzar un ataque
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-2.2
Descripción	El usuario quiere lanzar un ataque desde un nodo hacia otros
Precondición	El usuario está dentro del simulador Hay nodos conectados Existe un ordenador conectado El ordenador dispone de ataques
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona el ordenador atacante 2. El simulador muestra la vista de tráfico de red 3. El usuario selecciona la sección de ataques 4. El usuario selecciona el ataque a enviar 5. El usuario selecciona los objetivos 6. El usuario selecciona la opción de atacar 7. El simulador simula el ataque
Postcondición	Flujo registrado en los nodos
Excepciones	Valores no válidos
Importancia	Alta

Tabla B.11: CU-11 Lanzar un ataque

CU-12	Interceptar flujos
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-2.3
Descripción	Un nodo puede capturar los flujos que pasan por el
Precondición	Existe un nodo conectado
Acciones	<ul style="list-style-type: none"> 1. El flujo llega al nodo 2. El nodo intercepta el tráfico 3. El nodo realiza una acción
Postcondición	Acción respuesta
Excepciones	-
Importancia	Alta

Tabla B.12: CU-12 Interceptar flujos

CU-13	Registros de flujos
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-2.4
Descripción	Un nodo mantiene un registro de los flujos que pasan por el
Precondición	Se ha iniciado al menos un flujo en la red
Acciones	<ul style="list-style-type: none"> 1. El flujo llega al nodo 2. El nodo registra el flujo en el historial
Postcondición	Historial de flujos actualizado
Excepciones	-
Importancia	Alta

Tabla B.13: CU-13 Registros de flujos

CU-14	Importar biblioteca de comandos, ataques e interceptores
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-2.5
Descripción	El usuario importa una biblioteca externa de scripts
Precondición	El usuario está dentro del simulador Archivo válido
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona “Importar biblioteca externa” 2. El usuario selecciona un archivo a cargar 3. El simulador valida e importa las posibles funciones de la biblioteca
Postcondición	Comandos, ataques e interceptores disponibles
Excepciones	Archivo no válido o con nombres de funciones incorrectas
Importancia	Media

Tabla B.14: CU-14 Importar biblioteca de comandos, ataques e interceptores

CU-15	Eliminar biblioteca importada
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-2.6
Descripción	El usuario elimina una biblioteca que había sido cargada
Precondición	El usuario está dentro del simulador El simulador tiene una biblioteca cargada
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona “Eliminar biblioteca externa” 2. El simulador elimina la biblioteca de todos los nodos
Postcondición	Biblioteca externa eliminada
Excepciones	-
Importancia	Media

Tabla B.15: CU-15 Eliminar biblioteca importada

CU-16	Análisis de flujos en la red
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-3, RF-3.1, RF-3.2, RF-3.3, RF-3.4
Descripción	El usuario quiere analizar los flujos de datos en la red
Precondición	El usuario está dentro del simulador Hay una topología de red establecida
Acciones	<ol style="list-style-type: none"> 1. El usuario entra al simulador 2. El simulador muestra la interfaz 3. El simulador muestra la configuración de red actual
Postcondición	Configuración de red actual
Excepciones	-
Importancia	Alta

Tabla B.16: CU-16 Análisis de flujos en la red

CU-17	Importar modelo entrenado
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-3.1
Descripción	El usuario importa modelos de detección de ataques
Precondición	El usuario está dentro del simulador El usuario dispone de modelos entrenados con su script de análisis correspondiente
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona “Importar modelos” 2. El usuario selecciona los modelos a cargar 3. El simulador valida e importa los modelos
Postcondición	Listas de modelos a utilizar
Excepciones	Modelos no válidos o un script que no sigue la estructura adecuada
Importancia	Alta

Tabla B.17: CU-17 Importar modelo entrenado

CU-18	Eliminar modelos importados
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-3.2
Descripción	El usuario elimina los modelos que habían sido cargados
Precondición	El usuario está dentro del simulador El simulador tiene modelos cargados
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona “Eliminar modelos” 2. El simulador elimina todos los modelos de las conexiones
Postcondición	Modelos eliminados
Excepciones	-
Importancia	Media

Tabla B.18: CU-18 Eliminar modelos importados

CU-19	Seleccionar modelo de detección
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-3.3
Descripción	El usuario selecciona qué modelo usar en cada conexión para la detección de ataques
Precondición	El usuario está dentro del simulador El simulador tiene modelos cargados El simulador tiene nodos conectados
Acciones	<ol style="list-style-type: none"> 1. El usuario accede a la configuración de una conexión 2. El usuario selecciona un modelo disponible 3. El simulador guarda los cambios
Postcondición	Modelos eliminados
Excepciones	-
Importancia	Media

Tabla B.19: CU-19 Seleccionar modelo de detección

CU-20	Ver predicciones de ataques
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-3.4
Descripción	El simulador muestra predicciones sobre los flujos interceptados
Precondición	El usuario está dentro del simulador Existe nodos conectados La conexión tiene un modelo activo
Acciones	<ol style="list-style-type: none"> 1. Un paquete pasa por una conexión 2. El simulador llama a la función “analizar” del script que fue importada con el modelo 3. El simulador cambia de color la conexión según el resultado
Postcondición	Resultado de la predicción
Excepciones	Fallo en el uso del script
Importancia	Alta

Tabla B.20: CU-20 Ver predicciones de ataques

CU-21	Configurar el simulador
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-4
Descripción	El usuario quiere modificar las opciones de visualización y gestionar los estados de esta
Precondición	El usuario está dentro del simulador
Acciones	<ol style="list-style-type: none"> 1. El usuario busca en el menú la configuración a modificar 2. El usuario modifica los cambios que desea 3. El simulador muestra los nuevos cambios
Postcondición	Nuevos cambios aplicados
Excepciones	-
Importancia	Baja

Tabla B.21: CU-21 Configurar el simulador

CU-22	Ejecutar entrenamiento federado
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-5, RF-5.1, RF-5.2, RF-5.3, RF-5.4, RF-5.4.1, RF-5.4.2, RF-5.4.3, RF-5.5, RF-5.5.1, RF-5.5.2, RF-5.6
Descripción	El usuario quiere entrena un modelo de redes neuronales de forma federada
Precondición	Modelo base Datasets Tiene instaladas todas las dependencias
Acciones	<ol style="list-style-type: none"> 1. El usuario accede al módulo de entrenamiento federado 2. El usuario ejecuta el script 3. El script muestra las posibles ejecuciones disponibles
Postcondición	Listado de posibles ejecuciones
Excepciones	-
Importancia	Alta

Tabla B.22: CU-22 Ejecutar entrenamiento federado

CU-23	Cargar configuración del modelo base
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-5.1
Descripción	El usuario quiere cargar la configuración base del modelo
Precondición	Archivo base del modelo
Acciones	<ol style="list-style-type: none"> 1. El usuario ejecuta el script con el argumento <code>--model</code> y la ruta del modelo 2. El script carga el modelo en memoria
Postcondición	Modelo cargado en memoria
Excepciones	No se ha encontrado el modelo Errores en la configuración del modelo
Importancia	Alta

Tabla B.23: CU-23 Cargar configuración del modelo base

CU-24	Dividir dataset para entrenamiento federado
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-5.2
Descripción	El usuario quiere dividir un dataset en partes iguales para los clientes federados
Precondición	Dataset está cargado
Acciones	<ol style="list-style-type: none"> 1. El usuario ejecuta el script con el argumento <code>-d</code> seguido del número de archivos a crear 2. El script genera los archivos
Postcondición	Archivos divididos
Excepciones	-
Importancia	Media

Tabla B.24: CU-24 Dividir dataset para entrenamiento federado

CU-25	Cargar dataset
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-5.3
Descripción	El usuario quiere cargar el dataset en el cliente o servidor
Precondición	Dataset válido
Acciones	<ol style="list-style-type: none"> 1. El usuario ejecuta el script con el argumento --train (entrenamiento) o --test (validación) seguido de la ruta de los archivos de datos y etiquetas 2. El script carga en memoria los datasets
Postcondición	Datasets cargados en memoria
Excepciones	No se han encontrado los datasets Error en los archivos
Importancia	

Tabla B.25: CU-25 Cargar dataset

CU-26	Iniciar servidor federado
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-5.4, RF-5.4.1, RF-5.4.2, RF-5.4.3
Descripción	El usuario quiere iniciar un servidor de aprendizaje federado con parámetros configurables
Precondición	Hay acceso a internet Modelo base está cargado Datasets están cargados
Acciones	<ol style="list-style-type: none"> 1. El usuario ejecuta el script con el argumento <code>-s</code> y la dirección IP del servidor junto a su puerto 2. El usuario indica configuraciones adicionales como <code>--batch-size</code> o <code>--rounds</code> 3. El script inicializa el servidor 4. El servidor está pendiente de posibles clientes
Postcondición	Servidor federado inicializado
Excepciones	IP o puerto ocupado
Importancia	Alta

Tabla B.26: CU-26 Iniciar servidor federado

CU-27	Entrenamiento del modelo global
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-5.4.1
Descripción	El servidor organiza un entrenamiento global del modelo
Precondición	Existen clientes conectados
Acciones	<ol style="list-style-type: none"> 1. El servidor elige diferentes clientes para que entrenen el modelo 2. El servidor recibe los modelos entrenados por los clientes 3. El servidor aplica un algoritmo de agregación 4. Se actualiza el modelo global
Postcondición	Nuevo modelo global
Excepciones	Fallo en el entrenamiento de clientes Modelos de clientes diferentes
Importancia	Alta

Tabla B.27: CU-27 Entrenamiento del modelo global

CU-28	Evaluar modelo global
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-5.4.2
Descripción	El servidor organiza una evaluación global del modelo
Precondición	Dataset de validación Clientes conectados
Acciones	<ol style="list-style-type: none"> 1. El servidor elige diferentes clientes para que evalúen el modelo 2. El servidor evalúa el modelo 3. El servidor recibe las métricas de evaluación de los clientes 4. El servidor guarda las métricas de evaluación
Postcondición	Métricas de evaluación
Excepciones	Fallo en la evaluación de los clientes
Importancia	Alta

Tabla B.28: CU-28 Evaluar modelo global

CU-29	Exportar modelo global
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-5.4.3
Descripción	Una vez terminado el entrenamiento, el servidor exporta el modelo global resultante
Precondición	Se ha terminado el entrenamiento del modelo
Acciones	<ol style="list-style-type: none"> 1. El servidor realiza la exportación del modelo global resultante
Postcondición	Modelo global exportado
Excepciones	El modelo no ha finalizado el entrenamiento
Importancia	Alta

Tabla B.29: CU-29 Exportar modelo global

CU-30	Iniciar cliente federado
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-5.5, RF-5.5.1, RF-5.5.2
Descripción	El usuario quiere iniciar un cliente de aprendizaje federado con parámetros configurables
Precondición	Hay acceso a internet Modelo base está cargado Datasets están cargados
Acciones	<ol style="list-style-type: none"> 1. El usuario ejecuta el script con el argumento <code>-c</code> y la dirección IP del servidor junto a su puerto 2. El usuario indica configuraciones adicionales como <code>--batch-size</code> o <code>--epochs</code> 3. El script inicializa el cliente 4. El cliente se conecta con el servidor
Postcondición	Cliente federado inicializado
Excepciones	No se ha podido establecer conexión con el servidor
Importancia	Alta

Tabla B.30: CU-30 Iniciar cliente federado

CU-31	Entrenar modelo
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-5.5.1
Descripción	El cliente entrena el modelo que ha recibido del servidor
Precondición	Se ha recibido un modelo del servidor Dataset de entrenamiento
Acciones	<ol style="list-style-type: none"> 1. El cliente entrena el modelo 2. El cliente envía el modelo al servidor junto a las métricas de entrenamiento
Postcondición	Modelo local entrenado
Excepciones	Error durante el entrenamiento
Importancia	Alta

Tabla B.31: CU-31 Entrenar modelo

CU-32	Evaluar modelo
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-5.5.2
Descripción	El cliente evalúa el modelo que ha recibido del servidor
Precondición	Se ha recibido un modelo del servidor Dataset de validación
Acciones	<ol style="list-style-type: none"> 1. El cliente evalúa el modelo 2. El cliente envía las métricas de evaluación al servidor
Postcondición	Métricas de evaluación
Excepciones	Error durante la evaluación
Importancia	Alta

Tabla B.32: CU-32 Evaluar modelo

CU-33	Visualizar métricas
Versión	1.0
Autor	Alejandro Diez Bermejo
Requisitos asociados	RF-5.6
Descripción	El usuario quiere visualizar las métricas generadas durante el entrenamiento y la evaluación
Precondición	Existe un archivo de métricas
Acciones	<ol style="list-style-type: none"> 1. El usuario ejecuta el script con el argumento <code>-m</code> seguido de la ruta del archivo de métricas generado por el servidor 2. El script genera diferentes gráficas y archivos a partir del archivo de métricas
Postcondición	Archivos de valores y gráficas
Excepciones	Formato inválido del archivo
Importancia	Alta

Tabla B.33: CU-33 Visualizar métricas

Apéndice C

Especificación de diseño

C.1. Introducción

La especificación de diseño ofrece la perspectiva de cómo debe ser desarrollado un software para que cumpla los requisitos definidos anteriormente. Esta sección es el camino a seguir para poder pasar de un software en fase de análisis a una implementación de software real. Para ello, se van a explicar los distintos aspectos del diseño del software, incluyendo los datos que se van a manejar, la arquitectura de la aplicación y el diseño procedimental.

C.2. Diseño de datos

Entrada y Salida

Las entradas de datos del simulador están formadas principalmente por la topología de red, la cual será diseñada por el usuario. A partir de esta topología, el usuario podrá generar flujos de datos entre los nodos mediante una biblioteca externa. Además, se permite configurar los nodos y las conexiones entre estos. Otra entrada importante corresponde a los modelos de predicción de ataques, que el usuario podrá importar.

Las salidas dependen del modelo utilizado, pero todas ofrecen una salida general como la predicción de los flujos y un historial de los paquetes que han recibido cada nodo. Además, se permite guardar el proyecto en un formato `.yaml`.

En cuanto al script de aprendizaje federado, como entrada recibe el tipo de nodo federado con opciones adicionales, además de la configuración

base del modelo junto con los datasets de entrenamiento y validación. La salida será el modelo entrenado y un archivo de métricas. También existe la posibilidad de obtener gráficas a partir de las métricas.

Flujo de datos

El flujo de datos del simulador sería el siguiente:

1. El usuario importa o genera una nueva topología de red.
2. El usuario puede importar una biblioteca externa de comandos, ataques e interceptores para ampliar las opciones del simulador.
3. El usuario puede importar uno o varios modelos de predicción.
4. El usuario modificará los parámetros de los distintos nodos (nombre, tipo, posición) y conexiones (latencia, variabilidad de latencia, ancho de banda, modelo de predicción).
5. El usuario selecciona un comando o ataque a ejecutar y un objetivo.
6. El simulador transmite los paquetes entre los nodos, mientras estos registran el flujo.
7. Todas las conexiones con un modelo activo de predicción ejecutan su función de análisis y muestran información de la predicción en la pantalla.
8. Finalmente, el usuario puede guardar el proyecto actual.

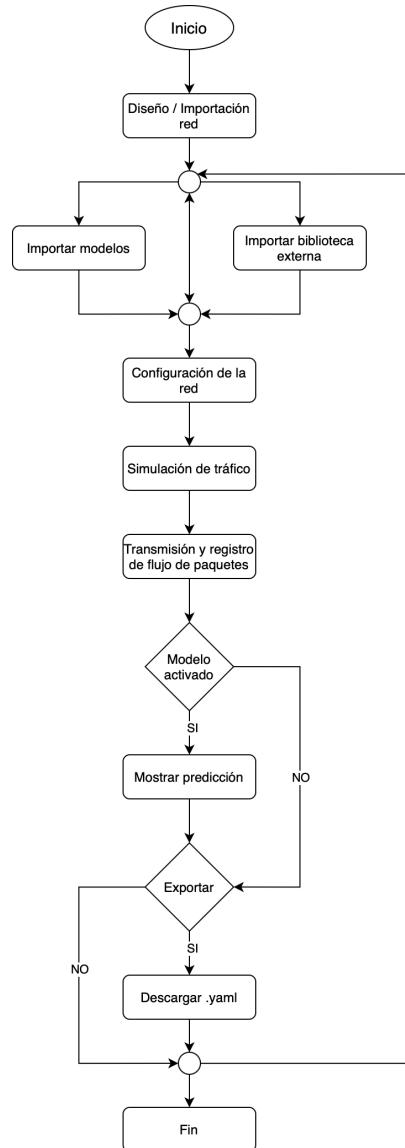


Figura C.1: Diagrama de flujo de datos del simulador

El flujo de datos del aprendizaje federado sería el siguiente:

1. El usuario selecciona el nodo federado que quiere iniciar.
2. El usuario carga la configuración base del modelo.
3. El usuario carga los datasets de entrenamiento y validación.
4. Se distribuye los modelos entre los clientes.

5. Se recopilan métricas.
6. Se genera un modelo global a partir de los locales.
7. Exportación del modelo global.
8. Exportación de las métricas.
9. Posible generación de gráficas.

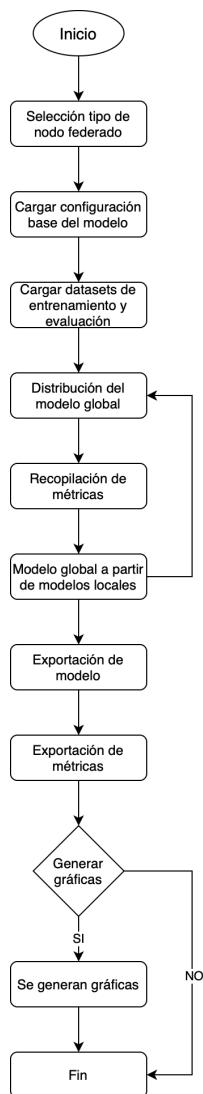


Figura C.2: Diagrama de flujo de datos del aprendizaje federado

C.3. Diseño arquitectónico

Para garantizar que el desarrollo del proyecto sea organizado y escalable, se han seguido patrones específicos, a pesar de las limitaciones impuestas por las tecnologías utilizadas durante la implementación.

Patrón Maestro-Esclavo

El entrenamiento de aprendizaje federado sigue el patrón Maestro-Esclavo. Este patrón se caracteriza por la existencia de un maestro (servidor) que se encargaría de seleccionar los clientes en los que el modelo va a ser entrenado y evaluado, de generar el modelo global y de recopilar las métricas de los clientes. Y de uno o varios esclavos (clientes) que se encargarían de “obedecer” las normas que impone el maestro, en este caso entrenar el modelo y evaluarlo.

Este patrón es muy común en arquitecturas distribuidas y establece una clara jerarquía entre los distintos nodos permitiendo una comunicación eficaz y un control centralizado del flujo de datos.

Patrón Modelo-Vista-Controlador (MVC)

Este patrón de diseño es uno de los más utilizados en el ámbito de diseño de software, aplicándose al desarrollo web, móvil, escritorio... El objetivo principal de este patrón es separar la capa de presentación y la capa de datos. Para ello usa tres componentes:

- **Modelo:** se corresponde con el acceso a los datos. Se encarga de almacenar y proporcionar los datos que maneja la aplicación.
- **Vista:** se corresponde con la visualización de los datos. Comunica todas las acciones que realiza el usuario al controlador.
- **Controlador:** se encarga de conectar el modelo con la vista. Sincroniza todas las acciones que el usuario realiza en la vista con el modelo y ofrece los datos del modelo a la vista.

Como se puede observar este patrón ofrece una gran flexibilidad para realizar cambios en la vista y que no afecte a la representación de los datos.

En este proyecto, se ha elegido realizar una adaptación del patrón MVC con el objetivo de desacoplar la interfaz lo máximo posible de los modelos.

Para ello, se han unido los modelos con los controladores en clases unificadas que gestionan tanto la lógica como el almacenamiento de los datos. Por otro lado, los componentes encargados de la visualización se corresponden con las vistas. Cabe mencionar, que se han implementado controladores generales encargado de gestionar la interacción entre todos los modelos y las vistas, siendo un punto central del flujo del simulador.

Patrón basado en componentes

Además del patrón MVC, este proyecto usa un patrón basado en componentes en la parte de vista. Este patrón lo que intenta es la reutilización de partes de la interfaz para simplificar el desarrollo y posteriormente la mantenibilidad.

Se podría decir que el componente actúa como un pequeño controlador que se encarga de llamar a los métodos del modelo y coordinar todos los eventos que el usuario realice sobre la interfaz.

Patrón Observador

Este patrón se ha utilizado para implementar la comunicación basada en cambios de estados dentro de la aplicación. Consiste en que una clase exponga un *Observable* y las diferentes clases se suscriben a este para poder obtener notificaciones cuando el estado de esa clase cambie y poder realizar tareas asociadas.

Por ejemplo, este patrón se usa en notificar cuando el usuario ha importado una biblioteca externa, o modelos y así los nodos y conexiones puede implementar dichos cambios en sus instancias. También se implementa este patrón en el gestor de estados de la aplicación, así el controlador de Red se puede suscribir y volver o ir a un punto en el que había estado.

Patrón Decorador

Los decoradores se han utilizado en este proyecto como una forma de extender la funcionalidad de las clases o métodos sin modificar directamente su implementación. Esto permite obtener un código más limpio y en caso de una futura modificación, será más fácil y rápida la implementación.

Este patrón se ha usado en los cambios de estado de la aplicación. Solo era necesario añadir el decorador de establecer el estado en las funciones de modificación de propiedades y el decorador obtener el estado en la clase donde era necesario escuchar el estado.

Patrón Singleton

El patrón Singleton ofrece la posibilidad de solo instanciar una única vez un clase y así obtener una instancia global durante la ejecución de la aplicación.

Este patrón fue usado como una solución debido a la imposibilidad de acceder en cualquier instante a un servicio fuera de un contexto determinado. Gracias al uso de este patrón, es posible acceder a estos servicios, que por definición son instancias únicas de una clase, en cualquier instante de ejecución del programa.

Arquitectura general

El resultado de la arquitectura tras aplicar los patrones explicados anteriormente sería:

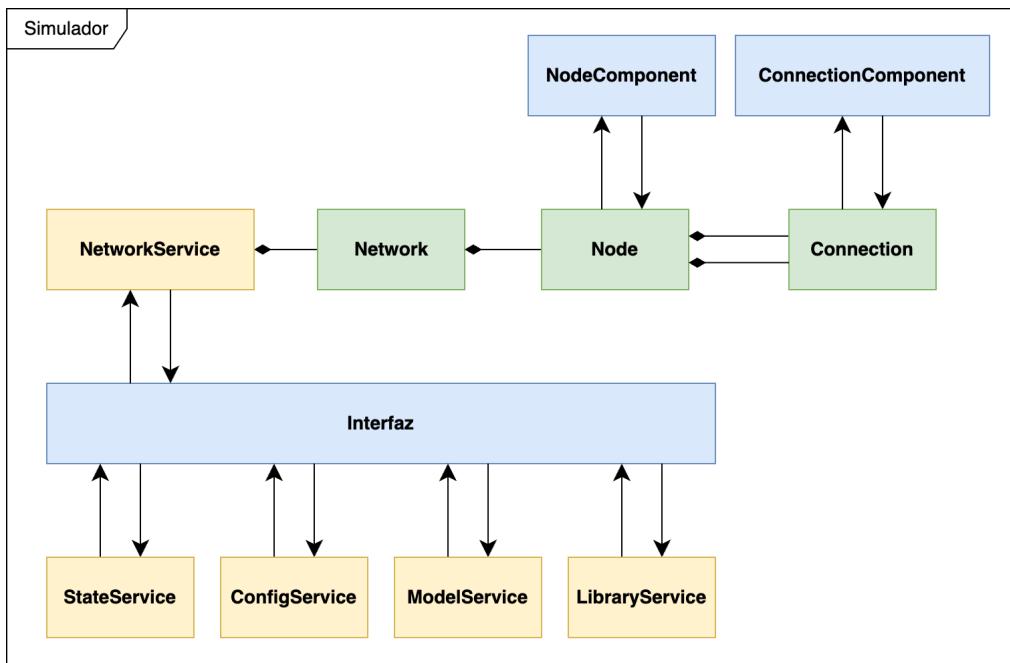


Figura C.3: Arquitectura general del simulador

Los rectángulos verdes representan los modelos, que contienen la lógica principal y la información estructural de la aplicación. Esta capa se detalla más profundamente en el diseño de clases.

Los rectángulos azules representan los componentes de interfaz (vistas). Estos se conectan a los modelos para presentar y modificar la información.

Los rectángulos amarillos representan los servicios, que funcionan como controladores globales. Estos servicios gestionan la comunicación entre los componentes de la interfaz y los modelos, y son accesibles desde cualquier parte del simulador.

Diseño de clases

Una vez descritos los principales patrones de diseño utilizados durante el desarrollo del proyecto, es necesario especificar uno de los aspectos clave que da cohesión y funcionalidad a toda la aplicación: el diseño de clases.

Este diseño establece la base estructural del simulador, permitiendo que sus componentes se comuniquen de formas organizada, mantenible y eficiente. Gracias a esto, el simulador es capaz de cumplir con todos los requisitos funcionales, mediante una arquitectura limpia y escalable.

La siguiente Figura C.4 muestra una simplificación de la estructura básica de clases del simulador.

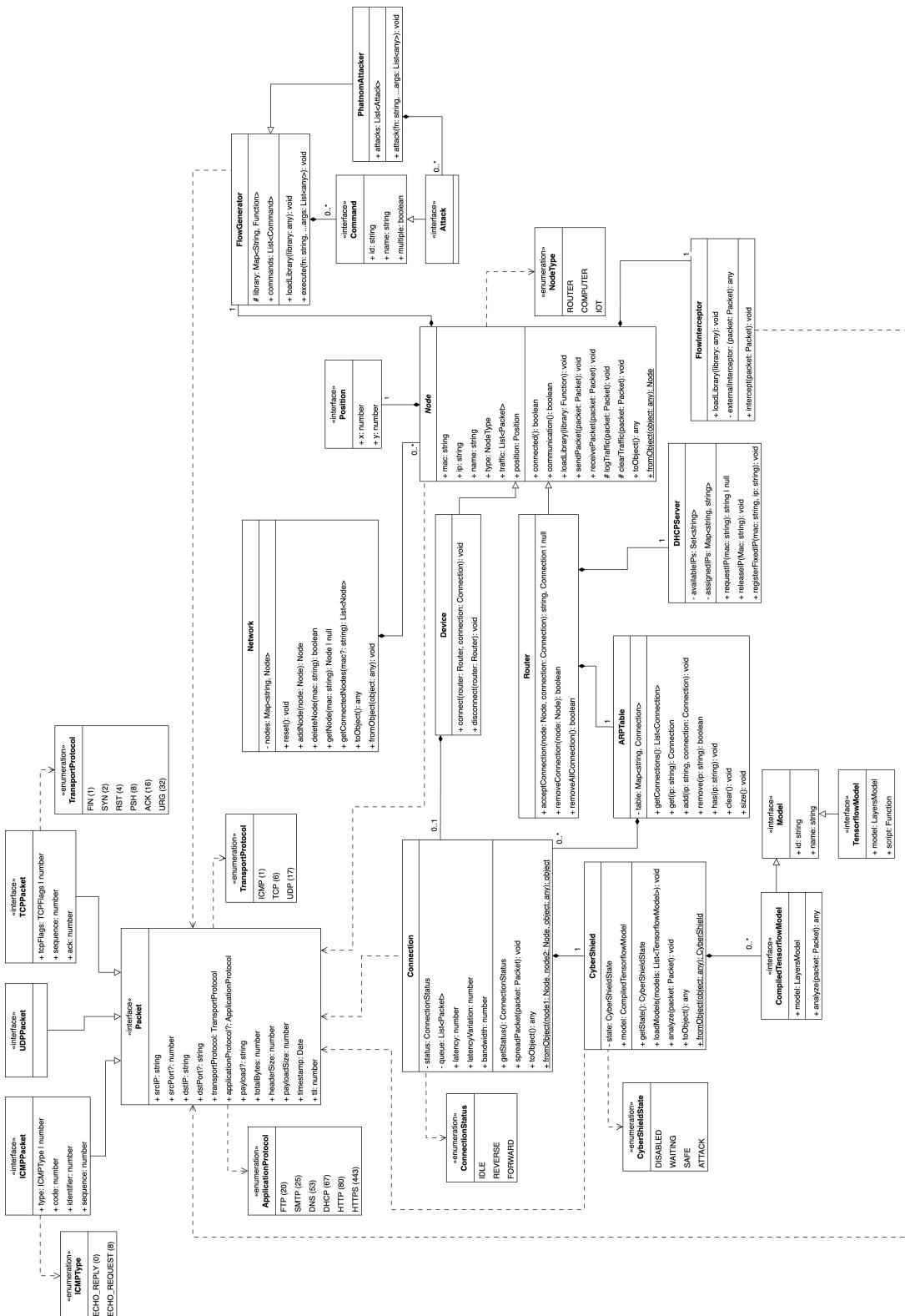


Figura C.4: Diagrama de clases

C.4. Diseño procedimental

En esta sección se describe el flujo de acciones que se realizan tanto en el simulador como durante el aprendizaje federado.

En la Figura C.5 se ha representado las interacciones fundamentales que se genera en el uso del simulador como el inicio de una red, añadir nodos, conexión entre nodos, lanzar y predecir un ataque y la eliminación de un nodo.

En la Figura C.6 se ha representado el flujo de acciones correspondiente al entrenamiento de forma federada.

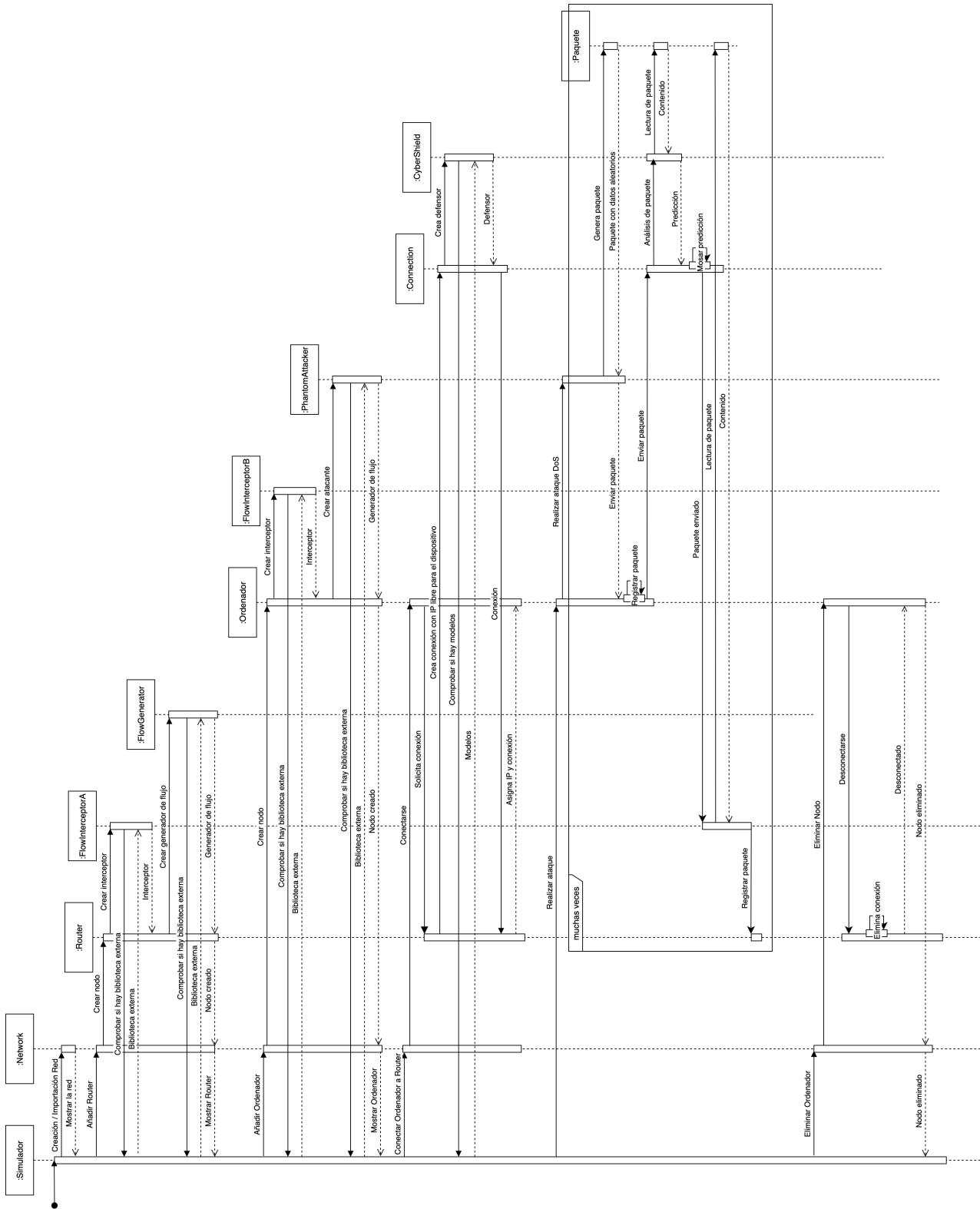


Figura C.5: Diagrama de secuencia de un flujo en el simulador

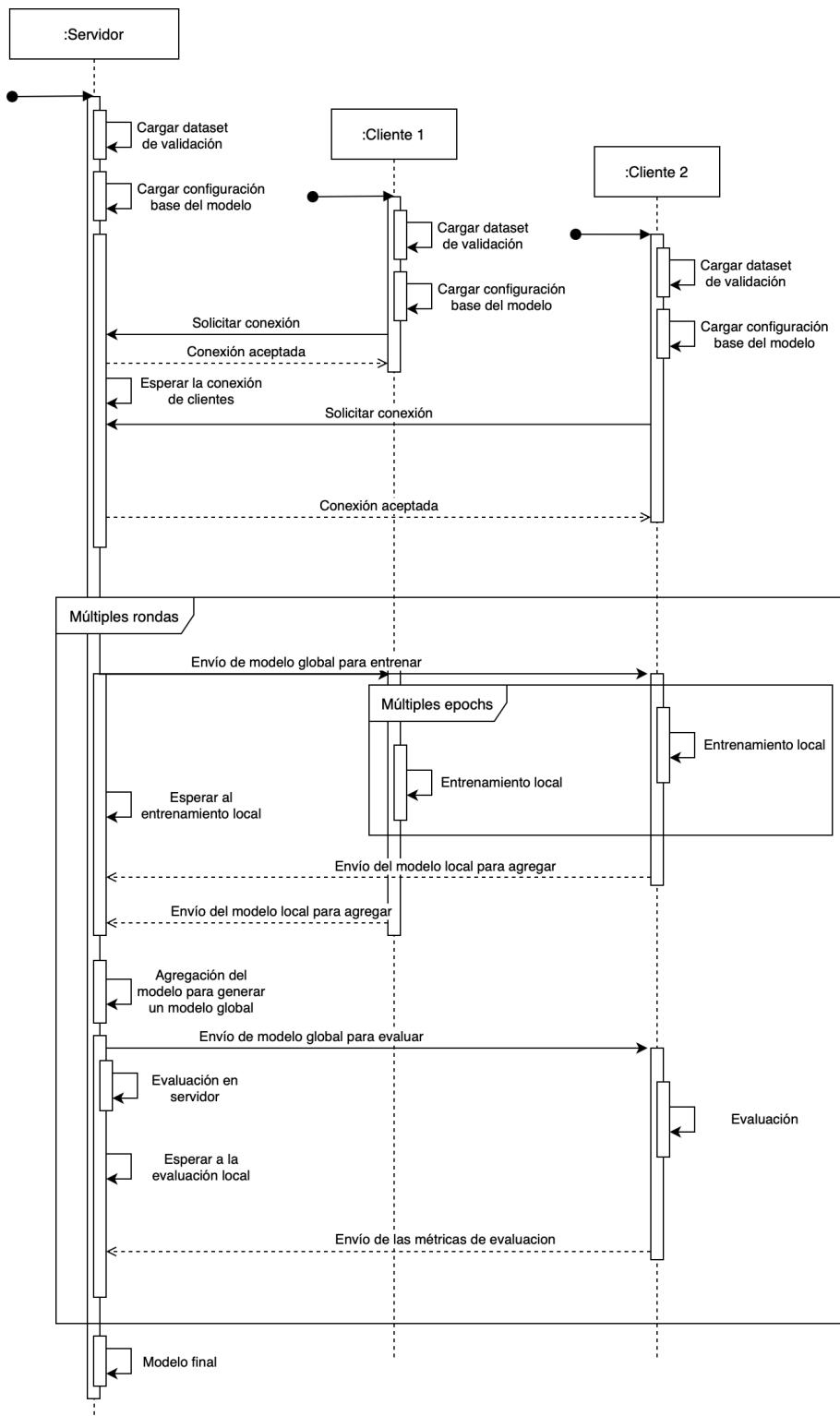


Figura C.6: Diagrama de secuencia del aprendizaje federado

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este apéndice se va a explicar todo lo relacionado con el código desarrollado durante la realización del proyecto. Se va a detallar de una forma clara y estructurada como se organizan los directorios del repositorio, para que en un futuro sirva de ayuda para comprender la estructura del proyecto.

D.2. Estructura de directorios

El código fuente desarrollado durante este proyecto se encuentra alojado en un repositorio público en GitHub¹, bajo la licencia MIT. A continuación, se va a presentar la estructura de directorios que forman este repositorio para poder dar una visión general de la estructura del proyecto.

La organización sigue unos estándares comunes, utilizados en proyectos de mediana y gran escala, lo que facilita su comprensión y mantenimiento por parte de programadores externos al proyecto. A continuación, se va a mostrar la estructura de directorios y archivos del proyecto.

- ***data***: directorio que contiene todos los archivos útiles para el uso del simulador.

¹https://github.com/AlejaDiez/iot_attack_detection

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

- ***layouts***: directorio que contiene archivos de proyectos para importarlos en el simulador.
- ***libs***: directorio que contiene archivos de bibliotecas de comandos, ataques e interceptores.
- ***models***: directorio que contiene modelos ya “compilados” para su uso en el simulador.
- ***docs***: directorio que contiene todos los archivos relacionados con la documentación del proyecto.
 - ***anexo***: directorio que contiene los archivos LaTeX relacionados con los anexos.
 - ***assets***: directorio que contiene recursos comunes de los archivos de documentación.
 - ***memoria***: directorio que contiene los archivos LaTeX relacionados con la memoria.
 - ***videos***: directorio que contiene todos los vídeos explicativos del proyecto.
- ***src***: directorio que contiene el código fuente de los dos proyectos del TFG.
 - ***attack_detector***: directorio principal del proyecto de detección de ataques mediante una red neuronal entrenada por aprendizaje federado.
 - ***iot_simulator***: directorio principal del proyecto del simulador de redes IoT.
- ***tests***: directorio que contiene pruebas realizadas durante las fases de investigación sobre bibliotecas de aprendizaje federado.
 - ***flower***: directorio que contiene pruebas realizadas con la biblioteca Flower.
 - ***tensorflow-federated***: directorio que contiene pruebas realizadas con la biblioteca TensorFlow Federated.
- ***.gitattributes***: archivo de configuración de Git para definir propiedades de archivos específicos.
- ***.gitignore***: archivo de configuración de Git que indica que archivos omitir para no incluirlos en el control de versión.

- **LICENSE**: archivo de licencia, donde se especifica el contenido de la licencia que usa el proyecto.
- **README.md**: archivo de presentación del proyecto, en el que se indica brevemente en que consiste el proyecto.

Dentro del directorio ***src/attack_detector*** nos encontramos con los siguientes directorios y archivos:

- **data**: directorio que contiene los datasets, configuraciones del modelo base para el aprendizaje federado y archivos de salida como métricas y modelos entrenados.
- **src**: directorio que contiene el código fuente del script para aprendizaje federado.
 - **utils**: directorio que contiene utilidades para el desarrollo del script.
 - **client.py**: archivo que contiene toda la lógica del cliente federado.
 - **main.py**: archivo principal del proyecto, que actúa como punto de entrada al sistema federado.
 - **server.py**: archivo que contiene toda la lógica del servidor federado.
- **.gitignore**: archivo de configuración de Git que indica que archivos omitir de este proyecto.
- **requirements.txt**: archivo de bibliotecas con sus versiones usadas en Python para facilitar la instalación.

Dentro del directorio ***src/iot_simulator*** nos encontramos con los siguientes directorios y archivos:

- **public**: directorio que contiene recursos públicos de la web.
- **src**: directorio que contiene todo el proyecto del simulador.
 - **app**: directorio principal de la aplicación, que contiene todas las vistas y lógica de esta.
 - **assets**: directorio de recursos de la aplicación.

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

- **components**: directorio de componentes de la aplicación.
En esta carpeta se encuentra parte de las vistas.
- **guards**: directorio de protectores de rutas de la aplicación.
- **models**: directorio con los modelos de la aplicación. En esta carpeta se encuentra parte de la lógica.
- **resolvers**: directorio de servicios de resolución de datos de la aplicación.
- **routes**: directorio de rutas de la aplicación. En esta carpeta se encuentra parte de las vistas.
- **services**: directorio de servicios de la aplicación. En esta carpeta se encuentran los controladores de la aplicación.
- **utils**: directorio que contiene funciones de utilidad.
- **app.component.html**: archivo html del componente principal de la aplicación.
- **app.component.ts**: archivo ts del componente principal de la aplicación.
- **app.routes.transition.ts**: archivo que contiene las animaciones entre rutas de la aplicación.
- **app.routes.ts**: archivo de rutas de la aplicación. En este archivo se encuentran las rutas definidas de la aplicación.
- **index.html**: archivo html principal que actúa como punto de entrada al simulador.
- **main.ts**: archivo de arranque de la aplicación angular.
- **styles.css**: archivo css principal que contiene los estilos globales de la aplicación.
- **.gitignore**: archivo de configuración de Git que indica que archivos omitir de este proyecto.
- **.prettierrc**: archivo de configuración para Prettier, herramienta de formateo de código
- **angular.json**: archivo de configuración principal del framework Angular.
- **components.json**: archivo de configuración de la biblioteca de componentes.
- **package-lock.json**: archivo generado automáticamente que registra las versiones exactas de las dependencias.

- ***package.json***: archivo principal de configuración del proyecto, contiene tanto el nombre, como versión, dependencias y muchos datos más.
- ***tailwind.config.js***: archivo de configuración de Tailwind CSS.
- ***tsconfig.app.json***: archivo de configuración específico para la compilación de la aplicación Angular.
- ***tsconfig.json***: archivo de configuración de TypeScript.
- ***tsconfig.spec.json***: archivo de configuración para la compilación de pruebas.

D.3. Manual del programador

Gestión del repositorio

Como ya se ha mencionado en el Apéndice A.2, este proyecto se ha desarrollado bajo una metodología SCRUM. Para ello se han planificado sprints en los que se establecieron distintas tareas. La gestión de estas tareas se ha realizado mediante la herramienta **GitHub Projects**², permitiendo realizar un seguimiento organizado del progreso.

Para mejorar la implementación de estas tareas y no interferir en las entregas pasadas, el proyecto se gestionó con un sistema de **ramas**. Cada mejora incremental se implementó en una rama independiente, permitiendo aislar los cambios y trabajar de forma controlada. Una vez finalizada la implementación, la rama se fusionaba con la **rama principal**, obteniendo así una versión actualizada y estable del proyecto.

Para el despliegue, existe una rama especial denominada **release**, que contiene únicamente los archivos generados en el momento de “compilación” del simulador.

Obtención del código fuente

Para obtener el código fuente del proyecto completo, es necesario clonar³ el repositorio o bien descargarlo directamente desde GitHub⁴.

²<https://github.com/users/AlejaDiez/projects/8>

³git clone https://github.com/AlejaDiez/iot_attack_detection.git

⁴https://github.com/AlejaDiez/iot_attack_detection

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

Entorno de desarrollo

Para poder reproducir completamente el proyecto, es necesario contar con la instalación de las siguientes herramientas (se podría usar otras alternativas):

- Python >=3.7 <=3.12
- Node.js >=18.19.1
- Git
- Navegador basado en Chromium
- IDE (Visual Studio Code)
- Acceso a compilador LaTeX (Overleaf)

D.4. Compilación, instalación y ejecución del proyecto

Entrenamiento con aprendizaje federado

Para el entrenamiento federado usando el script, es necesario realizar los siguientes pasos en cada uno de los dispositivos que vaya a participar en el entrenamiento. Hay que tener en cuenta que los datasets y la configuración del modelo deberían estar en cada uno de los dispositivos.

Instalación de dependencias

Para instalar las bibliotecas necesarias, es necesario situarse en la carpeta principal del proyecto **attack_detector** y ejecutar el siguiente comando:

```
$ pip install -r requirements.txt
```

Esto instalará todas las dependencias necesarias con las versiones correspondientes.

D.4. COMPILACIÓN, INSTALACIÓN Y EJECUCIÓN DEL PROYECTO

Ejecución del script

Para ejecutar el script, es necesario situarse en la carpeta principal del proyecto **attack_detector** y ejecutar el siguiente comando:

```
$ python src/main.py ...
```

El script admite los siguientes argumentos para configurar su ejecución:

Argumento	Descripción
-h, --help	Muestra el mensaje de ayuda
-s [HOST:PORT], --server [HOST:PORT]	Ejecuta el servidor de entrenamiento en la dirección y puerto especificados
-c [HOST:PORT], --client [HOST:PORT]	Ejecuta el cliente de entrenamiento con la dirección y puerto del servidor indicados
-d [NUM_FILES], --divide [NUM_FILES]	Divide el conjunto de entrenamiento en el número indicado de partes
-m [METRICS], --metrics [METRICS]	Convierte las métricas en archivos CSV y genera gráficas
--model MODEL	Ruta al archivo que contiene el modelo de la red neuronal
--train X_TRAIN Y_TRAIN	Ruta a los archivos que contienen el conjunto de entrenamiento (entradas y etiquetas)
--test X_TEST Y_TEST	Ruta a los archivos que contienen el conjunto de prueba (entradas y etiquetas)
--output OUTPUT	Ruta al directorio donde se guardarán los archivos de salida
--batch-size BATCH_SIZE	Tamaño del lote de entrenamiento
--rounds ROUNDS	Número de rondas de entrenamiento federado
--epochs EPOCHS	Número de épocas de entrenamiento por ronda

Tabla D.1: Argumentos para el script attack_detector

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

Simulador IoT

Instalación de dependencias

Para instalar Angular de forma global, es necesario ejecutar el siguiente comando:

```
$ npm install -g @angular/cli
```

Para instalar las bibliotecas necesarias, es necesario situarse en la carpeta principal del proyecto **iot_simulator** y ejecutar el siguiente comando:

```
$ npm install
```

Esto instalará todas las dependencias necesarias con las versiones correspondientes.

Ejecución del simulador

Para ejecutar el simulador, es necesario situarse en la carpeta principal del proyecto **iot_simulator** y ejecutar el siguiente comando:

```
$ npm run start
```

Esto ejecuta un comando definido en el archivo *package.json* y lanzará la aplicación en la dirección `http://localhost:4200` que se puede introducir en el navegador del equipo local para visualizar la interfaz.

“Compilación” del simulador

Para construir la versión final de producción del simulador, es necesario situarse en la carpeta principal del proyecto **iot_simulator** y ejecutar el siguiente comando:

```
$ npm run build
```

Esto ejecuta un comando definido en el archivo *package.json* y construirá los archivos finales del simulador, que se guardarán en el directorio *dist* del proyecto. Con estos archivos, es posible desplegar⁵ la aplicación en un servidor web o levantar un servidor local para su ejecución.

⁵Para desplegar la aplicación bajo otro nombre de dominio, hay que modificar el comando cambiando el argumento `--base-href`

D.5. Pruebas del sistema

Una vez terminado el proyecto, se llevó a cabo una fase de pruebas con el objetivo de validar la correcta implementación del sistema.

Pruebas del simulador de red

Se probó el simulador mediante diferentes configuraciones de red, con distinta cantidad de nodos, flujos y parámetros. También se validó que en los formularios de parámetros no era posible introducir valores no deseados y se comprobó que el simulador realizaba las acciones oportunas para solucionar estos datos inválidos.

Además, se comprobó que, al cargar archivos erróneos, ya sea de proyecto, biblioteca o modelo, lanzara una excepción y no se paraliza el programa.

Pruebas de entrenamiento federado

Se realizaron pruebas funcionales para asegurar de que el script respondía correctamente en situaciones habituales. Por ejemplo, se probó qué ocurría al intentar conectar un cliente a una dirección de servidor inexistente, lanzar un cliente antes de que el servidor estuviera en funcionamiento, o desconectar clientes en mitad del entrenamiento. También se hicieron pruebas de estrés, ejecutando varios clientes al mismo tiempo y trabajando con grandes volúmenes de datos y configuraciones más exigentes.

Apéndice E

Documentación de usuario

E.1. Introducción

En este apartado se van a detallar los requisitos que necesita el usuario para poder usar el software desarrollado y una guía detallada de como utilizar el simulador.

E.2. Requisitos de usuarios

Para poder usar el Simulador IoT, es necesario que el usuario cumpla los siguientes requisitos:

- Se requiere una **conexión a internet** para acceder a la aplicación web.
- Si el usuario desea crear su propia biblioteca de comandos, ataques o interceptores, debe contar con un **editor de código**.
- Para utilizar modelos de detección personalizados, el usuario debe tener instalado **Python** y la biblioteca **tensorflowjs**, además de un **editor de código** para implementar la función de análisis.

E.3. Instalación

Para un uso básico del simulador, sin necesidad de implementar funcionalidades adicionales, no se requiere ninguna instalación. Es suficiente con contar con un navegador web compatible.

Sin embargo, si se desea utilizar una biblioteca externa o modelos de predicción personalizados, es necesario contar con un **editor de código**, así como tener instalado **Python** y la biblioteca **tensorflowjs**.

Para instalar tensorflowjs, es necesario ejecutar el siguiente comando en la terminal del sistema:

```
$ pip install tensorflowjs
```

E.4. Manual del usuario

En esta sección se va a realizar un recorrido completo por el simulador, presentando en detalle cada una de sus funciones y utilidades. Además, se explicará de forma detallada el proceso de creación de una biblioteca personalizada de comandos, ataques e interceptores, así como la adaptación de un modelo de inteligencia artificial desarrollado en Python para su integración en el simulador.

Acceso al simulador

Para poder acceder al Simulador IoT, es necesario introducir la siguiente dirección en el navegador web:

https://alejadiez.github.io/iot_attack_detection/



Figura E.1: Logo del Simulador IoT

Vista general

El simulador está compuesto por una barra de menús flotante en la parte superior y un lienzo desplazable en ambas direcciones, donde se diseñará la arquitectura de la red.



Figura E.2: Vista general del Simulador IoT

Para comprender mejor la vista general, hay un video explicativo disponible en el siguiente enlace: https://github.com/AlejaDiez/iot_attack_detection/blob/main/docs/videos/general_view.mov

Barra de menús

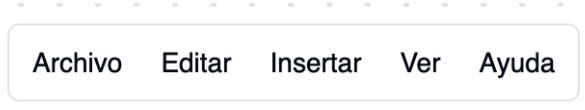


Figura E.3: Vista de la barra de menús

En esta barra de menús se encuentran todas las posibles opciones del simulador, que están divididas por diferentes secciones:

- **Archivo:** en este menú se encuentran todas las opciones relacionadas con la gestión de datos del simulador.

- **Nuevo archivo:** permite crear un nuevo proyecto, eliminando el proyecto actual.
 - **Abrir...:** permite abrir un proyecto guardado en el equipo.
 - **Importar biblioteca externa:** permite importar una biblioteca de comandos, ataques e interceptores. Si ya existe una biblioteca importada, ofrece la opción de eliminar la actual o reemplazarla por una nueva.
 - **Importar modelos:** permite importar modelos para el análisis de las trazas. En caso de haber modelos importados, se puede eliminar o reemplazar por otros.
 - **Guardar:** permite guardar el estado actual del proyecto en un archivo *.yaml*.
- **Editar:** en este menú se encuentran todas las opciones relacionadas con la edición del proyecto actual.
 - **Deshacer:** permite revertir los cambios realizados.
 - **Rehacer:** permite restaurar los cambios deshechos.
 - **Insertar:** en este menú se encuentran todas las opciones relacionadas con la inserción de nodos en la red.
 - **Router:** permite insertar un nodo tipo router en la red.
 - **Dispositivo:** permite insertar un nodo tipo dispositivo en la red.
 - **Ver:** en este menú se encuentran todas las opciones relacionadas con la configuración de visualización del simulador.
 - **Idioma:** permite cambiar el idioma del simulador. Hay disponibles el Alemán, Inglés, Español, Francés, Italiano y Portugués.
 - **Alto contraste:** permite cambiar al modo de alto contraste para personas con dificultad visual.
 - **Mostrar cuadrícula:** permite cambiar el estado de visibilidad de la cuadrícula del lienzo.
 - **Centrar:** permite centrar el lienzo.
 - **Zoom original:** permite restablecer la escala del lienzo.
 - **Acercar:** permite aumentar la escala del lienzo.
 - **Alejar:** permite disminuir la escala del lienzo.

- **Ayuda:** en este menú se encuentran todas las opciones relacionadas con la ayuda para el simulador.
- **Código fuente:** lleva al usuario al repositorio del proyecto, donde puede buscar la documentación o publicar cualquier comentario a mejorar.
- **Versión del simulador:** muestra al usuario la versión actual del simulador.

Lienzo

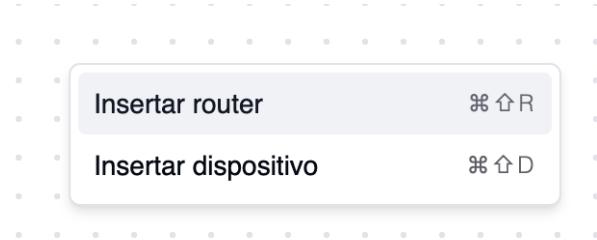


Figura E.4: Vista del lienzo con clic derecho

En el lienzo se va a mostrar toda la arquitectura de la red en el proyecto actual. Al hacer clic derecho sobre el lienzo, se muestra el siguiente menú:

- **Insertar router:** permite insertar un router en la posición actual.
- **Insertar dispositivo:** permite insertar un dispositivo en la posición actual.

Añadir un nodo

Para insertar un nodo, se puede hacer desde la barra de menús o desde el lienzo haciendo clic derecho. Una vez elegido el nodo a insertar router o dispositivo, se nos muestra una modal preguntando el nombre del nodo, y en caso de ser necesario el tipo de este.

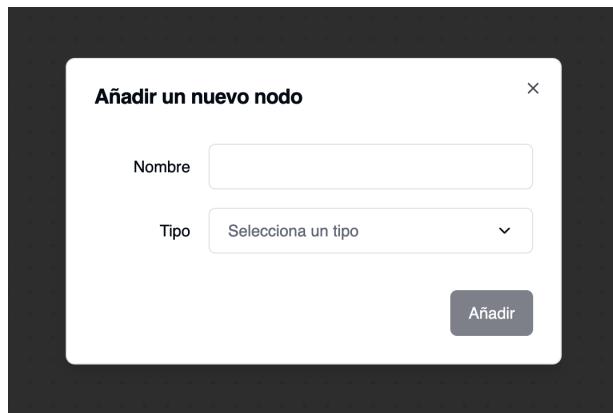


Figura E.5: Vista de la modal para añadir un nodo

Al añadir un dispositivo, es obligatorio seleccionar su tipo. Se puede elegir entre dispositivo **IoT**, que corresponde a un dispositivo normal de la red, o **ordenador**, que es un dispositivo capaz de enviar ataques.

Para comprender mejor como añadir nodos, hay un video explicativo disponible en el siguiente enlace: https://github.com/AlejaDiez/iot_attack_detection/blob/main/docs/videos/add_nodes.mov

Editar un nodo

Para editar un nodo, es necesario seleccionar el nodo e ir al apartado de “Configuración” del panel, o bien hacer clic derecho en el nodo y seleccionar “Configuración”.

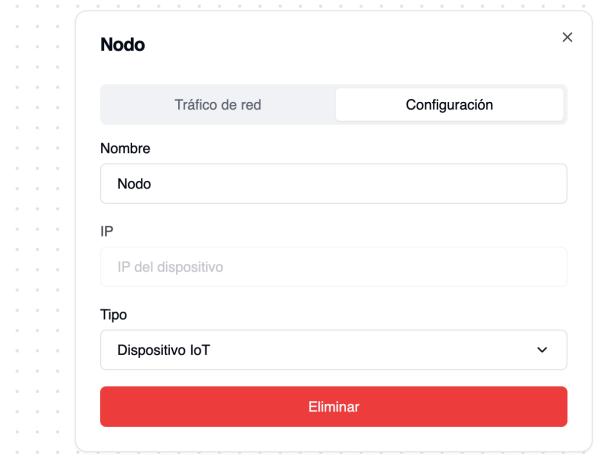


Figura E.6: Vista del panel para modificar un nodo

En esta sección se puede configurar el nombre del nodo, modificar el tipo y eliminar el nodo.

Para comprender mejor como editar un nodo, hay un video explicativo disponible en el siguiente enlace: https://github.com/AlejaDiez/iot_attack_detection/blob/main/docs/videos/edit_node.mov

Eliminar un nodo

Para eliminar un nodo, es necesario seleccionar el nodo e ir al apartado de “Configuración” del panel y seleccionar la opción “Eliminar”, o bien hacer clic derecho en el nodo y seleccionar “Eliminar”.

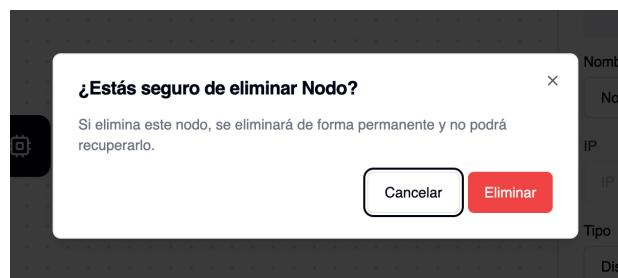


Figura E.7: Vista de la modal para eliminar un nodo

Para comprender mejor como eliminar un nodo, hay un video explicativo disponible en el siguiente enlace: https://github.com/AlejaDiez/iot_attack_detection/blob/main/docs/videos/delete_node.mov

Conectar un nodo

Para conectar un nodo, es necesario disponer de una red con un router y al menos un nodo. Se debe seleccionar el nodo e ir al apartado de “Tráfico de red” del panel y seleccionar la opción “Conejar”.



Figura E.8: Vista del panel para conectar un nodo

Para comprender mejor como conectar un nodo, hay un video explicativo disponible en el siguiente enlace: https://github.com/AlejaDiez/iot_attack_detection/blob/main/docs/videos/connect_node.mov

Editar conexión

Para editar una conexión, se debe seleccionar la conexión y se muestra una modal.

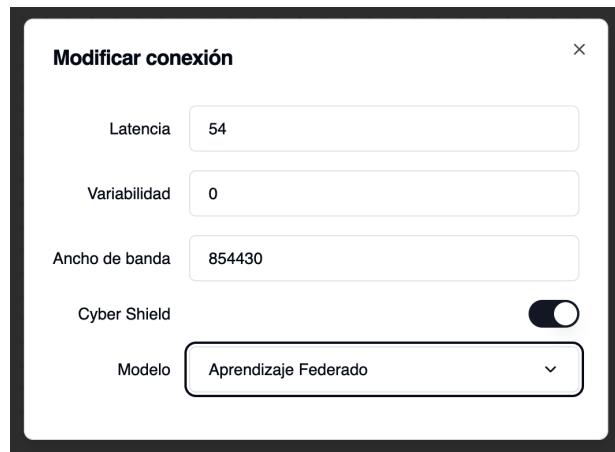


Figura E.9: Vista de la modal para editar una conexión

En esta sección se puede configurar la latencia de la conexión, la variabilidad de la latencia, el ancho de banda (bytes / segundo) y el modelo de detección activado.

Para comprender mejor como editar una conexión, hay un video explicativo disponible en el siguiente enlace: https://github.com/AlejaDiez/iot_attack_detection/blob/main/docs/videos/edit_connection.mov

Ejecutar comando

Para ejecutar un comando, se debe seleccionar el nodo desde donde se va a ejecutar el comando e ir al apartado de “Tráfico de red” del panel y seleccionar el comando a ejecutar además de los objetivos que se desee enviar el comando.

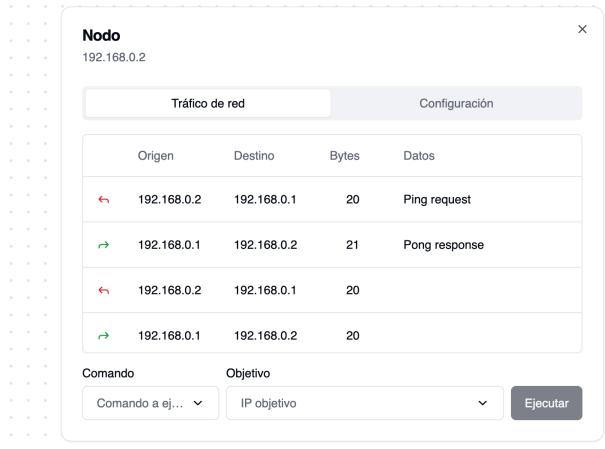


Figura E.10: Vista del panel para enviar un comando

Para comprender mejor como enviar un comando, hay un video explicativo disponible en el siguiente enlace: https://github.com/AlejaDiez/iot_attack_detection/blob/main/docs/videos/send_command.mov

Visualizar paquete

Para visualizar un paquete que se ha enviado, se debe seleccionar el nodo desde donde se va a visualizar el paquete e ir al apartado de “Tráfico de red” del panel y seleccionar el paquete a visualizar.

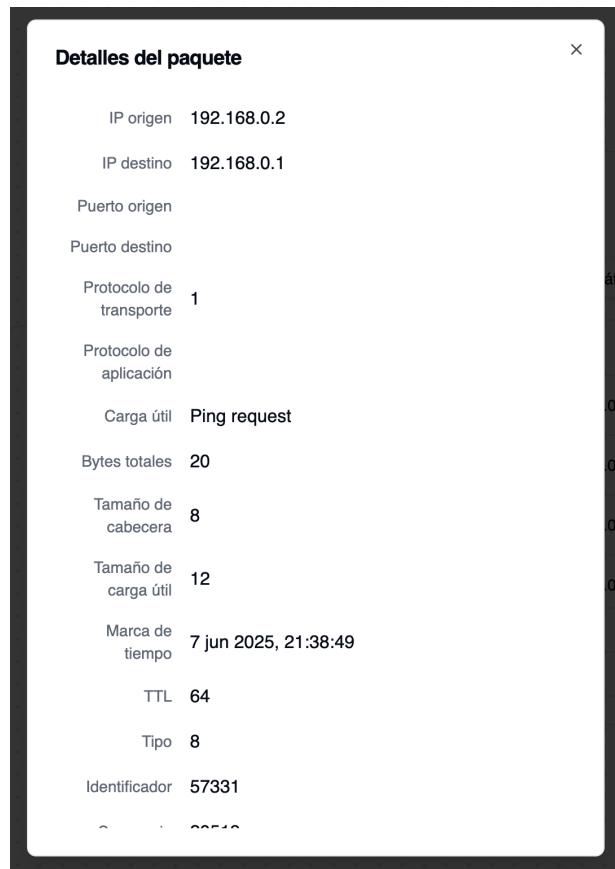


Figura E.11: Vista de la modal para visualizar un paquete

Para comprender mejor como visualizar un paquete, hay un video explicativo disponible en el siguiente enlace: https://github.com/AlejaDiez/iot_attack_detection/blob/main/docs/videos/view_packet.mov

Lanzar ataque

Para lanzar un ataque, se debe seleccionar un nodo de tipo ordenador desde donde se va a lanzar el ataque e ir al apartado de “Phantom Attacker” del panel y seleccionar el ataque a lanzar además de los objetivos que se desee enviar el ataque.

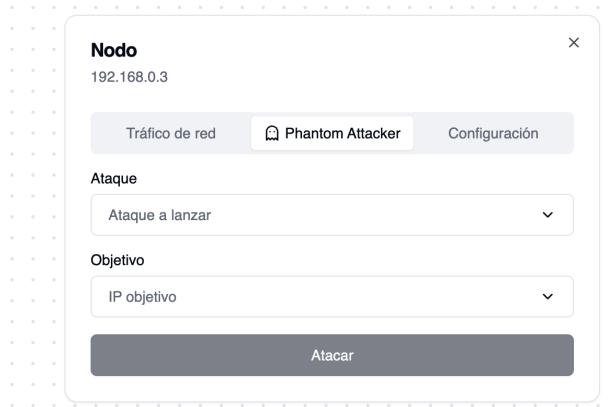


Figura E.12: Vista del panel para enviar un ataque

Para comprender mejor como enviar un ataque, hay un video explicativo disponible en el siguiente enlace: https://github.com/AlejaDiez/iot_attack_detection/blob/main/docs/videos/send_attack.mov

Detectar un ataque

Para detectar los ataques, se debe tener al menos una conexión con un modelo de detección activado, además de generar un flujo maligno desde uno o varios ordenadores.

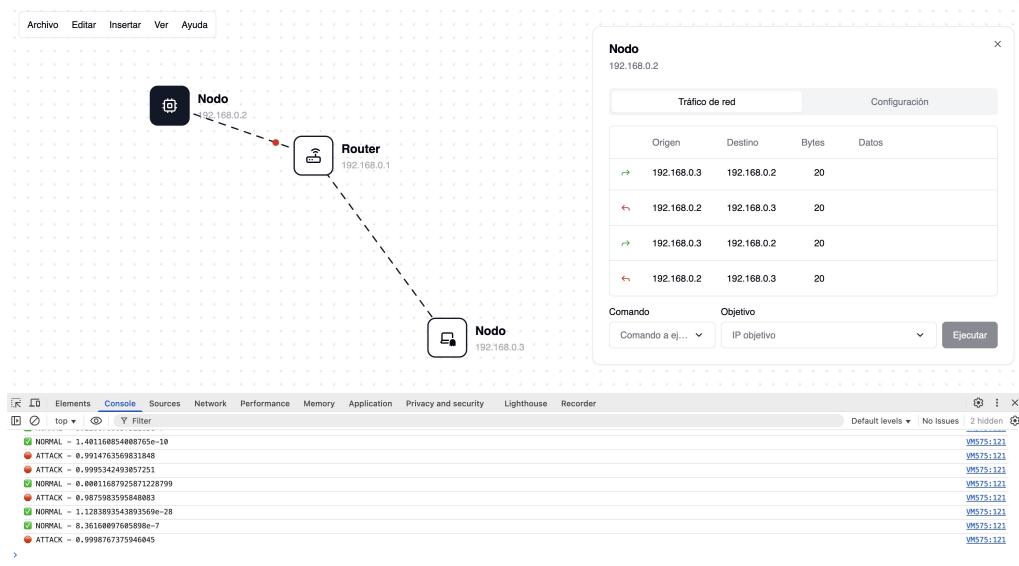


Figura E.13: Vista de la detección de un ataque DoS

Sobre cada conexión hay un indicador que señala si el modelo está activado, además del estado de la predicción. Si el indicador está en **azul**, el modelo aún no dispone de suficientes datos para realizar una predicción. Si está en **verde**, el flujo ha sido clasificado como benigno. En cambio, si está en **rojo**, el flujo ha sido identificado como malicioso.

Para comprender mejor como detectar ataques, hay un video explicativo disponible en el siguiente enlace: https://github.com/AlejaDiez/iot_attack_detection/blob/main/docs/videos/detect_attack.mov

Escribir una biblioteca externa para el simulador

Para crear una biblioteca externa que incluya comandos, ataques e interceptores, es necesario programar un script en JavaScript que siga una estructura específica.

Comandos

Los comandos permiten generar flujos benignos entre nodos. Cada comando debe definirse como una función cuyo nombre comience con el prefijo **cmd_**, seguido del identificador. La función recibe como parámetro el nodo emisor (**self**) y una o varias direcciones IP de destino (**target / ...targets**).

```
/**
 * @name Comando
 * @param {Object} self - El nodo que ejecuta el comando
 * @param {string} self.mac - Dirección MAC
 * @param {string} self.ip - Dirección IP
 * @param {string} self.name - Nombre
 * @param {string} self.type - Tipo
 * @param {function(packet) => void} self.send - Función de envío
 * @param {string} target - Dirección IP objetivo
 * @returns {void}
 */
function cmd_Commando(self, target) {
    ...
}
```

Ataques

Los ataques permiten generar flujos malignos entre un nodo ordenador y los demás nodos. Cada ataque debe definirse como una función cuyo nombre comience con el prefijo **atk_**, seguido del identificador. La función recibe como parámetro el nodo atacante (**self**) y una o varias direcciones IP de destino (**target / ...targets**).

```
/**
 * @name Ataque Multiple
 * @param {Object} self - El nodo que lanza el ataque
 * @param {string} self.mac - Dirección MAC
 * @param {string} self.ip - Dirección IP
 * @param {string} self.name - Nombre
 * @param {string} self.type - Tipo
 * @param {function(packet) => void} self.send - Función de envío
 * @param {...string} targets - Direcciones IP objetivos
 * @returns {void}
 */
function atk_Ataque_Multiple(self, ...targets) {
    ...
}
```

Interceptores

Los interceptores permiten reaccionar a paquetes que llegan a un nodo. Para definir un interceptor debe de ser una función cuyo nombre sea:

- **intcp**: interceptor genérico
- **intcp_router**: interceptor para el nodo de tipo router
- **intcp_iot**: interceptor para el nodo de tipo iot
- **intcp_computer**: interceptor para el nodo de tipo computer

La función recibe como parámetro el nodo interceptor (**self**) y el paquete que se ha interceptado (**packet**). En caso de que se desee descartar la ejecución del interceptor predeterminado, es necesario devolver **null** u otro valor.

```
/**
 * @param {Object} self - El nodo que lanza el ataque
 * @param {string} self.mac - Dirección MAC
 * @param {string} self.ip - Dirección IP
 * @param {string} self.name - Nombre
 * @param {string} self.type - Tipo
 * @param {function(packet) => void} self.send - Función de envío
 * @param {Object} packet - Paquete interceptado
 * @param {string} packet.srcIP - Dirección IP de origen
 * @param {number} [packet.srcPort] - Puerto de origen
 * @param {string} packet.dstIP - Dirección IP de destino
 * @param {number} [packet.dstPort] - Puerto de destino
 * @param {number} packet.transportProtocol - Protocolo de transporte
 * @param {number} [packet.applicationProtocol] - Protocolo de aplicación
 * @param {string} [packet.payload] - Carga útil
 * @param {number} packet.totalBytes - Tamaño total en bytes
 * @param {number} packet.headerSize - Tamaño de la cabecera
 * @param {number} packet.payloadSize - Tamaño de la carga útil
 * @param {Date} packet.timestamp - Marca temporal
 * @param {number} packet.ttl - Tiempo de vida (TTL)
 * @param {number} packet.type - Tipo de mensaje ICMP
 * @param {number} packet.code - Código de mensaje ICMP
 * @param {number} packet.identifier - Identificador del mensaje ICMP
 * @param {number} packet.sequence - Número de secuencia del mensaje ICMP o TCP
```

```

* @param {number} packet.tcpFlags - Banderas TCP
* @param {number} packet.ack - Número de confirmación
* @returns {void} Retornar un valor, cancela el interceptor predeterminado
*/
function intcp(self, packet) {
    ...
}

```

Para comprender mejor como escribir una biblioteca externa, hay un video explicativo disponible en el siguiente enlace: https://github.com/AlejaDiez/iot_attack_detection/blob/main/docs/videos/library.mov

Adaptar un modelo de inteligencia artificial

Para adaptar un modelo de inteligencia artificial desarrollado con **TensorFlow** en Python al simulador, es necesario convertirlo a un formato compatible con **TensorFlow.js** y crear un **script** que utilice el modelo para predecir. Estos archivos se comprimirán en una carpeta **.zip** que será lo que interpretará el simulador.

Convertir modelo

Para adaptar [2] el modelo es necesario tener instalada la biblioteca **tensorflowjs** y tener un modelo¹ en formato **.h5**.

Para realizar la conversión, es necesario ejecutar el siguiente comando en la terminal del sistema:

```
$ tensorflowjs_converter --input_format=keras \
    /ruta/al/modelo_guardado \
    /ruta/de/salida
```

Este proceso realiza una conversión a un modelo compatible con **TensorFlow.js**, generando un archivo **.json** con la configuración del modelo y uno o varios archivos **.bin** que contienen los pesos.

Script de análisis

Para utilizar el modelo, es necesario crear un archivo JavaScript que contenga una función llamada **analyze**. Esta función se encarga de analizar

¹El modelo debe haber sido entrenado con una versión de Keras v2. En caso contrario, es necesario convertirlo para que sea compatible con esta versión.

cada uno de los paquetes que se le pasen como parámetro y devolver un valor según el resultado:

- **true**: se ha detectado un ataque
- **false**: no se ha detectado ataque
- **any**: el modelo no dispone de datos suficientes para realizar un análisis

Cabe destacar que el script dispone de dos variables globales:

- **tf**: biblioteca de TensorFlow.js
- **model**: modelo de predicción a usar

```
/**
 * @param {Object} packet - Paquete interceptado
 * @param {string} packet.srcIP - Dirección IP de origen
 * @param {number} [packet.srcPort] - Puerto de origen
 * @param {string} packet.dstIP - Dirección IP de destino
 * @param {number} [packet.dstPort] - Puerto de destino
 * @param {number} packet.transportProtocol - Protocolo de transporte
 * @param {number} [packet.applicationProtocol] - Protocolo de aplicación
 * @param {string} [packet.payload] - Carga útil
 * @param {number} packet.totalBytes - Tamaño total en bytes
 * @param {number} packet.headerSize - Tamaño de la cabecera
 * @param {number} packet.payloadSize - Tamaño de la carga útil
 * @param {Date} packet.timestamp - Marca temporal
 * @param {number} packet.ttl - Tiempo de vida (TTL)
 * @param {number} packet.type - Tipo de mensaje ICMP
 * @param {number} packet.code - Código de mensaje ICMP
 * @param {number} packet.identifier - Identificador del mensaje ICMP
 * @param {number} packet.sequence - Número de secuencia del mensaje ICMP o TCP
 * @param {number} packet.tcpFlags - Banderas TCP
 * @param {number} packet.ack - Número de confirmación
 * @returns {boolean | void} - true: ataque, false: benigno, void: esperando
 */
function analyze(packet) {
    ...
}
```

Para comprender mejor como adaptar un modelo de inteligencia artificial, hay un video explicativo disponible en el siguiente enlace: https://github.com/AlejaDiez/iot_attack_detection/blob/main/docs/videos/model.mov

Apéndice F

Anexo de sostenibilización curricular

F.1. Introducción

En este anexo se presenta una reflexión personal sobre los aspectos de sostenibilidad considerados durante el desarrollo del proyecto, centrado en la simulación y detección de ataques sobre una red mediante técnicas de inteligencia artificial.

Este proyecto no solo propone solucionar problemas tecnológicos, sino que también tiene como perspectiva una visión de sostenibilidad, tanto en temas sociales, económicos y ambientales.

F.2. Reflexión sobre la sostenibilidad en el proyecto

Sostenibilidad social

El desarrollo de este proyecto contribuye a la sostenibilidad social al centrarse en la mejora de la seguridad en internet, un aspecto de gran importancia para proteger la privacidad y derechos de los usuarios. Esta capacidad de detectar ataques permite reducir el riesgo de colapso en organizaciones de uso común como la sanidad, educación, servicios públicos...

Además, al enfocar el entrenamiento de la red neuronal mediante el aprendizaje federado, se evita la necesidad de centralizar datos utilizados para el entrenamiento del modelo. Esto supone una reducción significativa de

riesgos asociados a la privacidad de los usuarios, cumpliendo con normativas de protección de datos.

Sostenibilidad económica

Desde una visión económica, el proyecto aporta mucho valor a mejorar la eficiencia de detección de ataques, logrando reducir costes asociados a diferentes problemas derivados de estos incidentes, como la interrupción de servicios o la pérdida de datos. La detección temprana de los ataques ayuda a reducir los riesgos económicos de una empresa o institución.

Sostenibilidad ambiental

Gracias a las técnicas de aprendizaje automático, un modelo va a ser mucho más eficiente a la hora de detectar ataques frente a un programa tradicional. Esta mayor eficiencia permite reducir el uso innecesario de recursos, contribuyendo así a un menor consumo energético.

Además, el proyecto ofrece un simulador que permite probar diferentes modelos sin necesidad de desarrollar una infraestructura, así se puede lograr reducir significativamente el impacto ambiental y económico asociado al uso del hardware.

F.3. Competencias de sostenibilidad adquiridas

Durante el desarrollo del proyecto, he podido adquirir una serie de competencias vinculadas a la sostenibilidad, según los principios definidos por la CRUE¹.

SOS 1 - Competencia en la contextualización crítica del conocimiento estableciendo interrelaciones con la problemática social, económica y ambiental, local y/o global

He logrado desarrollar la capacidad de contextualizar críticamente el uso de redes neuronales y otros métodos tecnológicos en relación con la problemática social, económica y ambiental. Puedo comprender las conexiones que

¹Conferencia de Rectores de las Universidades Españolas

hay entre los métodos aplicados en el proyecto y su impacto en el medio ambiente y la sociedad, produciendo así un desarrollo más responsable y alineado con dichos principios.

SOS2 - Competencia en la utilización sostenible de recursos y en la prevención de impactos negativos sobre el medio natural y social

Durante el desarrollo del proyecto, he intentado, en la medida de lo posible, la optimización de recursos, implementando el uso de códigos eficientes y reduciendo el acceso a datos personales para el entrenamiento del modelo, preservando así la privacidad.

SOS4 - Competencia en la aplicación de principios éticos relacionados con los valores de la sostenibilidad en los comportamientos personales y profesionales

He aprendido a aplicar principios éticos relacionados con valores de sostenibilidad. Este aprendizaje ha proporcionado una mayor responsabilidad por el medio ambiente, el fomento de la equidad social y una toma de decisiones que promuevan un desarrollo sostenible.

F.4. Conclusiones

El desarrollo del proyecto sobre aprendizaje federado y simulación de red ha sido una experiencia para aprender conocimientos tanto en temáticas tecnológicas como principios de sostenibilidad aplicados a la informática.

Bibliografía

- [1] Seguridad Social: Cotización / Recaudación de Trabajadores. <https://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537>. [Último acceso 1 de Junio de 2025].
- [2] TensorFlow.js: Convierte un modelo guardado de Python en un formato de TensorFlow.js. <https://codelabs.developers.google.com/codelabs/tensorflowjs-convert-python-savedmodel>. [Último acceso 4 de Mayo de 2025].