UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Transformers para la Clasificación de Tráfico IoT en Ventanas Temporales: Un enfoque basado en NLP

Presentado por Sergio Martín Reizábal
en Universidad de Burgos — 6 de julio de 2025
Tutor: Rubén Ruiz González
Co-tutor: Nuño Basurto Hornillos

Índice general

Índice general i

Índice de figuras iii

Índice de tablas iv

Apéndice A Plan de Proyecto Software 1
A.1. Introducción 1
A.2. Planificación temporal 1
A.3. Estudio de viabilidad 4

Apéndice B Especificación de Requisitos 7
B.1. Introducción 7
B.2. Objetivos generales 7
B.3. Catálogo de requisitos 8
B.4. Casos de uso 8

Apéndice C Especificación de diseño 13
C.1. Introducción 13
C.2. Diseño de datos 13
C.3. Diseño arquitectónico 14
C.4. Diseño procedimental 15

Apéndice D Documentación técnica de programación 19
D.1. Introducción 19
D.2. Estructura de directorios 20
D.3. Manual del programador 21

D.4. Compilación, instalación y ejecución del proyecto 22
D.5. Pruebas del sistema 23

Apéndice E Documentación de usuario **25**
E.1. Introducción 25
E.2. Requisitos de usuarios 25
E.3. Instalación 25
E.4. Manual del usuario 26

Apéndice F Anexo de sostenibilización curricular **31**
F.1. Reflexión sobre sostenibilidad 31

Bibliografía **35**

Índice de figuras

C.1. Diagrama ER simplificado del IDS propuesto 14
C.2. Diagrama de clases del IDS propuesto 16
C.3. Diagrama de secuencia para CSV sin etiquetas 17
C.4. Diagrama de secuencia para CSV con etiquetas 18

E.1. Pantalla inicial. Seleccione un CSV de CICFlowMeter y pulse *Predecir*. 26
E.2. Tabla con las ventanas de 5 s, la etiqueta predicha y la confianza (0–100 %). 26
E.3. Distribución temporal de las etiquetas predichas. Cada barra representa una ventana de 5 s, coloreada según la clase predicha más probable. 27
E.4. Distribución temporal de las etiquetas reales. Cada barra representa una ventana de 5 s, coloreada según la clase real. 28
E.5. Matriz de confusión y métricas de evaluación (solo CSV etiquetado). 28

Índice de tablas

A.1. Dedicación estimada 3
A.2. Resumen de costes del TFG 4

B.1. Requisitos funcionales 8
B.2. Requisitos no funcionales 8
B.3. CU-1 — Clasificación de tráfico IoT sin etiquetas previas. . . . 9
B.4. CU-2 — Evaluación de un CSV etiquetado. 10
B.5. CU-3 — Ventanización y tensorización para inferencia. 11

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este apéndice se describe la planificación y la viabilidad de este Trabajo Fin de Grado "Transformers para la Clasificación de Tráfico IoT en Ventanas Temporales: Un enfoque basado en NLP". Este trabajo conllevó una duración aproximada de 17 semanas comprendidas entre marzo de 2025 hasta el 3 de julio de 2025, en el que se fueron alternando fases de investigación intensa y sprints quincenales de desarrollo ágil.

La documentación de la memoria y los anexos se fue realizando a partir del 5 de mayo de 2025 en paralelo con la fase de desarrollo.

A.2. Planificación temporal

Estrategia ágil adoptada

- **Duración total:** mar 2025 – 03 jul 2025 (17 semanas).
- **Cadencia:** sprints de 2 semanas a partir del primer commit (29 mar 2025).
- **Carga de trabajo:** ∼15 h/*sem* en investigación, ∼20 h/*sem* en desarrollo activo (abr – jul 2025).
- **Herramientas:** Git + GitHub

Línea de tiempo

1. **08 mar - 28 mar 2025 → Investigación**

   Lectura de libros para aprender sobre redes neuronales, pruebas sueltas en PyTorch, lecturas de artículos acerca del estado del arte de sistemas IDS para tráfico de red IoT y búsqueda de datasets.

   Los primeros commits del repositorio implementan un *traductor inglés–español* (Transformer completo) siguiendo la serie de Pepe Cantoral Ph.D.[1]

2. **29 mar - 17 abr 2025 → Implementación del primer modelo**

   Se implementan los módulos `PositionalEncoding`, `MultiHeadAttention`, `PositionFeedForward`, `EncoderSubLayer`, `Encoder` que conforman el `TransformerEncoderClassifierWithCLS`.

   Para verificar que el modelo conserva las dimensiones previstas, se generó un minibatch con datos aleatorios $X \in \mathbb{R}^{[B,128,79]}$, donde B es el *batch size*, 128 es el número máximo de flujos por ventana y 79 el número de atributos por flujo. Cada vector de 79 atributos pasa por una proyección lineal $79 \rightarrow 64$ El encoder mantiene esa misma forma a su salida, luego se extrae la primera fila correspondiente al *token CLS* de cada tensor perteneciente al batch. ($[B, 64]$) y una última capa lineal la convierte en los $[B, C]$ *logits*, donde C es el número de clases de tráfico que el IDS puede predecir. Así se confirma que todo el flujo de datos a través del Transformer respeta las dimensiones previstas.

3. **18 abr - 12 jun 2025 → Preprocesado adaptado al nuevo contexto y comienzo de la fase de documentación**

   Para estas fechas ya se dispone de un dataset IoT (CICIoT2023 [6]) y una herramienta para extraer flujos de red (CICFlowMeter [5]). Se programa el script de ventanizado y se genera el dataset, donde cada instancia es una ventana de 5 s con sus primeros 127 flujos.

   Se divide el dataset en tres particiones, 64 % entrenamiento, 16 % validacion y 20 % test. Tras entrenar y evaluar el modelo, se exportan tanto sus pesos como el escalador ajustado con los datos de entrenamiento.

   A partir del 5 de mayo comienzo a redactar la memoria y los anexos en LaTeX mientras sigo programando.

[1] Cantoral P., *Transformer paso a paso* (playlist), YouTube, 2021. https://www.youtube.com/watch?v=x6E44DDWg5Q&list=PLWzLQn_hxe6Ym2y17FreTcn6robjZ9LMk

4. **13 jun - 23 jun 2025 → Backend y UI implementadas**

   Una vez ya entrenado y evaluado el modelo y viendo los buenos resultados, se considera realizar una interfaz gráfica para poder visualizar los resultados de cualquier traza de red capturada.

   Se implementa una API REST con FastAPI que se encarga de:

   - `POST /predict`: recibe un CSV de CICFlowMeter, ventaniza, tensoriza, ejecuta el Transformer y devuelve un JSON con las predicciones.
   - Si el CSV trae la columna `Label`, añade `accuracy`, `f1_weight` y la matriz de confusión.
   - Despliegue de la web con `uvicorn`.

   Paralelamente se diseña la UI:

   - *Frontend* en Bootstrap 5 + DataTables para la tabla de ventanas y Plotly.js para las gráficas interactivas.
   - Selector de carga de CSV, barra de progreso, tablas para las ventanas, gráficos de barras coloreados para las predicciones y etiquetas reales, matriz de confusión y métricas.

5. **24 jun - 03 jul 2025 → Correción de bugs, obtención de métricas de rendimiento del modelo y cierre de documentación**

   Se dan los ultimos retoques a la UI, se corrigen algunos bugs, se sacan métricas como el tiempo de predicción medio por ventana con el Transformer y se finaliza la documentación de la memoria y los anexos.

Dedicación estimada

La Tabla A.1 detalla la distribución de horas entre las tres grandes actividades del proyecto.

Actividad	**Horas**
Investigación	3 sem × 15 h = 45 h
Desarrollo (código)	14 sem × 20 h = 280 h
Documentación	9 sem × 20 h = 180 h
Total	**505 h**

Tabla A.1: Dedicación estimada

A.3. Estudio de viabilidad

Viabilidad económica

Como el IDS propuesto aun esta en fase de prototipo teórico, los costes se reducen a mano de obra y un único equipo de sobremesa para desarrollo y pruebas. La Tabla A.2 resume el desglose:

Concepto	**Importe (€)**
Mano de obra (505 h × 10 €)	5 050
PC de sobremesa con RTX 4060 Ti 16 GB	1 300
Software libre (PyTorch, FastAPI, Bootstrap 5)	0
Coste directo total	**6 350**

Tabla A.2: Resumen de costes del TFG

Retorno proyectado. El IDS supera el 98 % de precisión con 0.23 ms/ventana en GPU, lo que permite licenciarlo como complemento de ciberseguridad a 25 € por instalación. Para amortizar los 6 350 € iniciales bastarían:

$$\frac{6\,350\ €}{25\ €/\text{licencia}} \approx 254 \text{ licencias.}$$

Viabilidad legal

Desde el inicio se atendieron cuatro aspectos clave:

1. **Licencia de software**. El código se publica bajo MIT; todas las dependencias (PyTorch, FastAPI, Bootstrap 5) son MIT o BSD [7].
2. **Datasets**. El dataset CICIoT2023 se distribuye bajo CC-BY 4.0, apto para uso comercial con cita [6].
3. **Protección de datos**.
   - *Base jurídica*: interés legítimo en la seguridad de la red (art. 6 §1.f RGPD) y art. 20 LOPDGDD 3/2018 [1, 2].
   - *Minimización*: el pipeline trabaja exclusivamente con 79 métricas estadísticas, es decir, jamás inspecciona contenidos de paquetes, por lo que el riesgo de identificar a un usuario es nulo.

- *Ámbito*: el procesamiento se realiza *in-situ*, los CSV no abandonan nunca el entorno local del usuario.

4. **Control de exportaciones**. El modelo carece de criptografía sujeta al Anexo I (Cat. 5) del Reglamento (UE) 2021/821[3], por lo que no necesita licencia de exportación.

Conclusión: con licencias libres, datos abiertos y un tratamiento estrictamente estadístico, el IDS cumple RGPD / LOPDGDD y no está sometido a controles de doble uso por lo que resulta jurídicamente viable.

Apéndice B

Especificación de Requisitos

B.1. Introducción

El sistema de detección de intrusiones (IDS) propuesto transforma capturas de red IoT en información valiosa, permitiendo clasificar el tráfico (incluidas las distintas categorías de ataque) mediante un modelo Transformer ligero. El flujo completo abarca 5 etapas:

- Generación de flujos.
- Agrupación en ventanas.
- Conversión de ventanas a tensores normalizados de tamaño fijo.
- Clasificación mediante un Transformer ligero.
- Presentación de resultados a través de una API REST y de una interfaz web.

B.2. Objetivos generales

1. Alcanzar una exactitud global (*accuracy*) alta, garantizando una alta precision (*precision*) y alta sensibilidad (*recall*) para cada tipo de ataque. La métrica F1 resume ambas mediante su media armónica, útil cuando se requiere equilibrio entre ambas.
2. Garantizar una latencia media de inferencia por ventana de 5 s $\leq$ 1 ms en GPU y $\leq$ 200 ms en CPU.

3. Ofrecer una API y una UI comprensibles sin documentación extensa.

B.3. Catálogo de requisitos

ID	Descripción
RF-01	El backend acepta un archivo CSV exportado por CIC-FlowMeter con 79 columnas numéricas.
RF-02	Agrupa automáticamente los flujos en ventanas consecutivas de 5 s según el tiemestamp de incio de cada flujo y su duración.
RF-03	Con cada ventana genera un tensor 128×79; realiza *padding* con ceros si hay menos de 127 flujos, o truncado si los supera.
RF-04	Clasifica cada tensor y devuelve un JSON con los campos {`interval`, `label`, `confidence`}.
RF-05	Cuando el CSV contiene `Label`, añade en la respuesta la serie temporal de etiquetas reales, la matriz de confusión y las métricas `accuracy` y `f1_weight`.
RF-06	La interfaz web permite cargar el CSV y presenta los resultados en tabla y gráficas interactivas.

Tabla B.1: Requisitos funcionales

ID	Descripción
RNF-01	Exactitud global alta y F1 ponderado alto.
RNF-02	Latencia media $\leq$ 1 ms por ventana en GPU y $\leq$ 200 ms en CPU.
RNF-03	Backend en FastAPI y PyTorch; frontend en Bootstrap 5, DataTables y Plotly.
RNF-04	Código y dependencias bajo licencias libres compatibles MIT.

Tabla B.2: Requisitos no funcionales

B.4. Casos de uso

CU-1	**Clasificar CSV sin etiquetas**
Versión	1.0
Autor	Sergio Martín Reizábal
Requisitos asociados	RF-01, RF-02, RF-03, RF-04, RF-06
Descripción	El usuario web carga un CSV con flujos sin columna `Label`. El sistema genera ventanas, tensoriza y devuelve las etiquetas predichas y su confianza.
Precondición	El servicio `/predict` está operativo y el archivo contiene las 79 columnas numéricas de CICFlowMeter.
Acciones	1. Seleccionar el archivo en la interfaz web y pulsar *Predecir*. 2. El backend agrupa los flujos en ventanas de 5 s. 3. Crea los tensores 128×79 y los normaliza. 4. Ejecuta el modelo Transformer y genera las predicciones. 5. Devuelve el JSON y la UI muestra tabla y gráficas.
Postcondición	Se muestran las etiquetas predichas con su probabilidad para cada ventana.
Excepciones	Archivo sin extensión `.csv` o con columnas incompletas $\rightarrow$ respuesta `HTTP 400`.
Importancia	Alta

Tabla B.3: CU-1 — Clasificación de tráfico IoT sin etiquetas previas.

CU-2	**Clasificar CSV con etiquetas**
Versión	1.0
Autor	Sergio Martín Reizábal
Requisitos asociados	RF-01, RF-02, RF-03, RF-04, RF-05, RF-06
Descripción	El usuario carga un CSV que incluye la columna `Label`. Además de las predicciones, el sistema calcula la matriz de confusión y las métricas de rendimiento.
Precondición	CSV con columna `Label` válida.
Acciones	1. Cargar el CSV y pulsar *Predecir*. 2. El backend ventaniza y tensoriza como en CU-1. 3. Calcula las predicciones. 4. Compara con `Label` real y calcula matriz de confusión y métricas. 5. Envía el JSON; la UI presenta predicciones, matriz y métricas.
Postcondición	Se presentan etiquetas predichas y reales, matriz de confusión y métricas de evaluación.
Excepciones	Desfase ventanas $\leftrightarrow$ etiquetas $\rightarrow$ advertencia y descarta ventana afectada.
Importancia	Alta

Tabla B.4: CU-2 — Evaluación de un CSV etiquetado.

CU-3	**Ventanizar y generar tensores**
Versión	1.0
Autor	Sergio Martín Reizábal
Requisitos asociados	RF-02, RF-03
Descripción	Proceso interno que agrupa los flujos en ventanas y los convierte en tensores normalizados de tamaño fijo para la inferencia.
Precondición	CSV con 79 columnas numéricas cargado en memoria.
Acciones	1. Recorrer los flujos ordenados por marca temporal. 2. Formar ventanas contiguas de 5 s. 3. Para cada ventana, aplicar truncado o *padding* hasta 128 flujos. 4. Normalizar cada fila con el escalador aplicado a la partición de entrenamiento. 5. Devolver el tensor resultante al módulo de inferencia.
Postcondición	Tensores 128×79 listos para el modelo.
Excepciones	Ventanas sin flujos $\rightarrow$ se omiten en la inferencia. Solo se predicen ventanas con flujos activos
Importancia	Media

Tabla B.5: CU-3 — Ventanización y tensorización para inferencia.

Apéndice C

Especificación de diseño

C.1. Introducción

En este apéndice se explican las decisiones de diseño que se han tomado para que el IDS, el sistema que se presenta en este TFG, convierta el tráfico IoT capturado en información de seguridad útil. El documento se divide en tres apartados principales: (1) diseño de datos, (2) diseño arquitectónico y (3) diseño procedimental.

C.2. Diseño de datos

Modelo lógico

Aunque los flujos no se almacenan de forma permanente en una base de datos relacional, sí se mantienen temporalmente en memoria durante el procesamiento; aun así, conviene representar las siguientes dos entidades lógicas en un diagrama ER:

El diagrama ER de la Figura C.1 resume las dos entidades en memoria que intervienen durante la inferencia:

- **ventana**: intervalo fijo de 5 s, identificado por su instante de inicio y con los campos `etiqueta_predicha` y `etiqueta_real`.
- **flujo**: registro generado por CICFlowMeter con 79 atributos numéricos y un enlace (FK) a la ventana a la que pertenece.

La relación entre ambas entidades es esencialmente $1:N$: cada ventana puede contener varios flujos. En sentido inverso, un flujo pertenece a **una única ventana en la mayoría de los casos; excepcionalmente puede dividirse entre dos ventanas consecutivas** cuando su *activity timeout* (5 s) no coincide exactamente con el inicio de la ventana siguiente. Esta situación (poco frecuente) queda igualmente representada, ya que el flujo se replica en la segunda ventana.

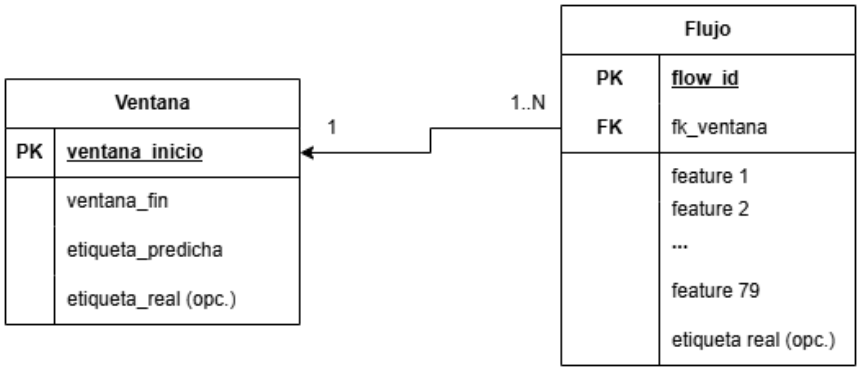

Figura C.1: Diagrama ER simplificado del IDS propuesto

Estructura en memoria

Durante la inferencia los flujos se cargan en un `DataFrame` con 80 columnas (79 numéricas y la etiqueta), se agrupan por `Ventana_inicio` y se convierten en tensores de dimensión $[128 \times 79]$. El dataset PyTorch `WindowDataset` se encarga de aplicar el `StandarScaler` y devuelve los tensores listos para el modelo.

C.3. Diseño arquitectónico

La Figura C.2 muestra el diagrama de clases a nivel de diseño:

- **PredictController** (FastAPI) se encarga de dirigir el pipeline completo.
- **FlowWindowProcessor** agrupa los flujos en ventanas de 5 s.
- **Tensorizer** crea los tensores a partir de las ventanas y `WindowDataset` aplica la normalización a cada ventana de forma indivudual.

- **TransformerEncoderClassifierWithCLS** clasifica cada ventana.
- **ResultsPage** para visualizar las tablas y gráficos interactivos de las predicciones que hace el modelo de las ventanas.
- El bloque gris detalla los componentes internos del Transformer.

C.4. Diseño procedimental

En esta sección se describe el comportamiento dinámico del sistema en los dos escenarios operativos contemplados.

Secuencia: CSV sin etiquetas

La Figura C.3 muestra un diagrama de secuencia para cuando el usuario en un escenario real, carga una traza de tráfico de red sin etiquetar:

1. La **UI Web** envía un `POST /predict` con el CSV adjunto.
2. **PredictController** detecta que el archivo no contiene ventanas y llama a `FlowWindowProcessor` y `Tensorizer`.
3. Los tensores resultantes se pasan al modelo `TransformerEncoderClassifierWithCLS`.
4. Las etiquetas predichas se devuelven en un JSON con los campos `interval`, `label` y `confidence`.
5. La UI actualiza la tabla y la gráfica de etiquetas predichas.

Secuencia: CSV con etiquetas y evaluación

La Figura C.4 muestra un diagrama de secuencia para cuando el usuario en un escenario hipotético en el que quiere evaluar el modelo, carga una traza de tráfico de red etiquetada.

1. Tras la inferencia, **PredictController** compara `y_pred` con `y_true`.
2. El módulo **Métricas** (basado en `scikit-learn`) calcula la matriz de confusión, `accuracy` y `f1_weight`.

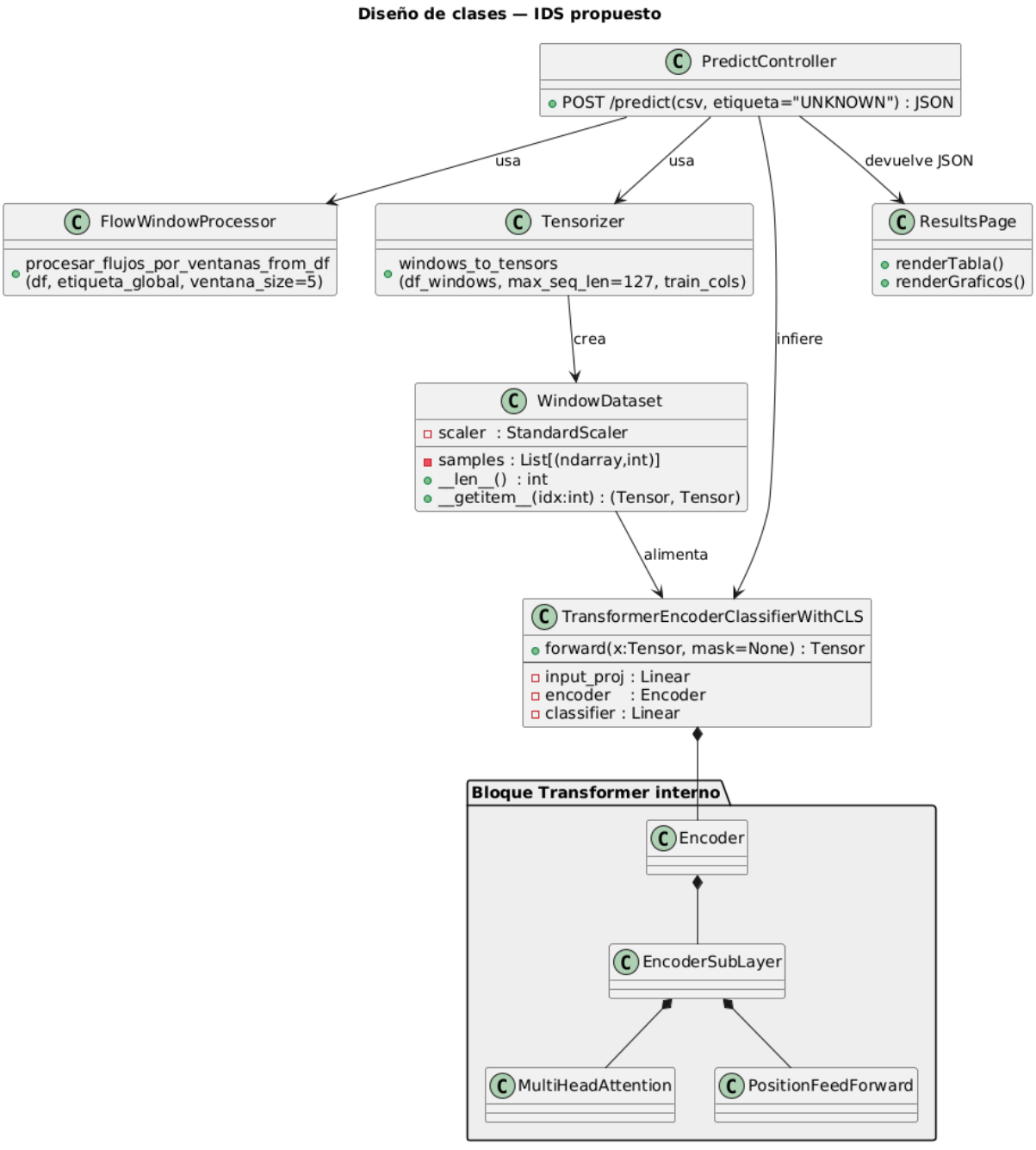

Figura C.2: Diagrama de clases del IDS propuesto

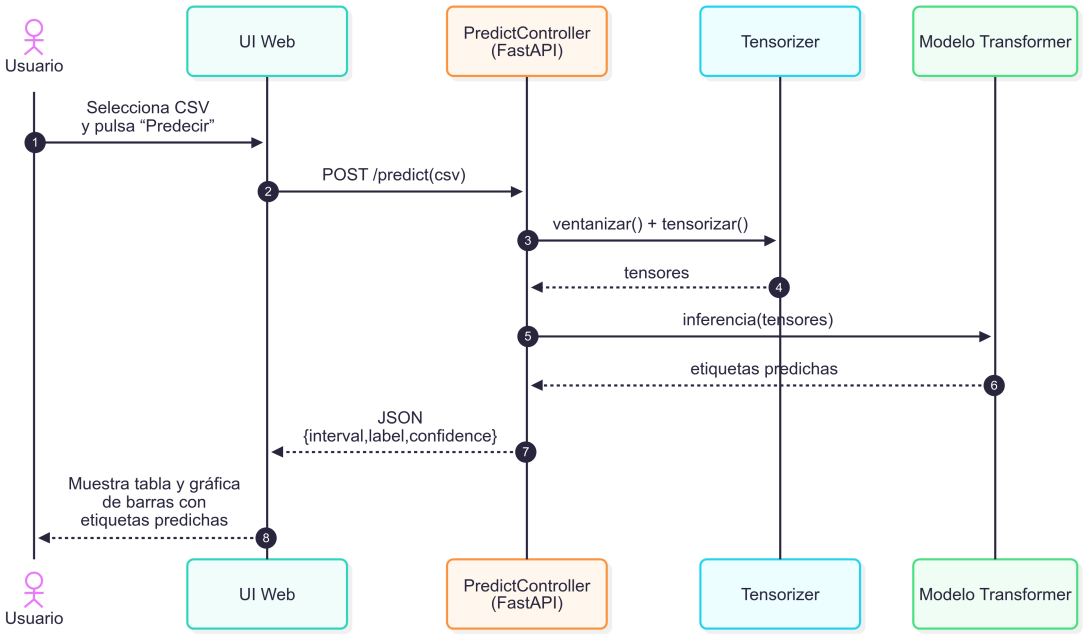

Figura C.3: Diagrama de secuencia para CSV sin etiquetas

3. La respuesta JSON incluye un bloque adicional `evaluation` con dichos resultados.

4. La UI muestra los elementos anteriores más una gráfica con las etiquetas reales, una matriz de confusión y un cuadro con las métricas.

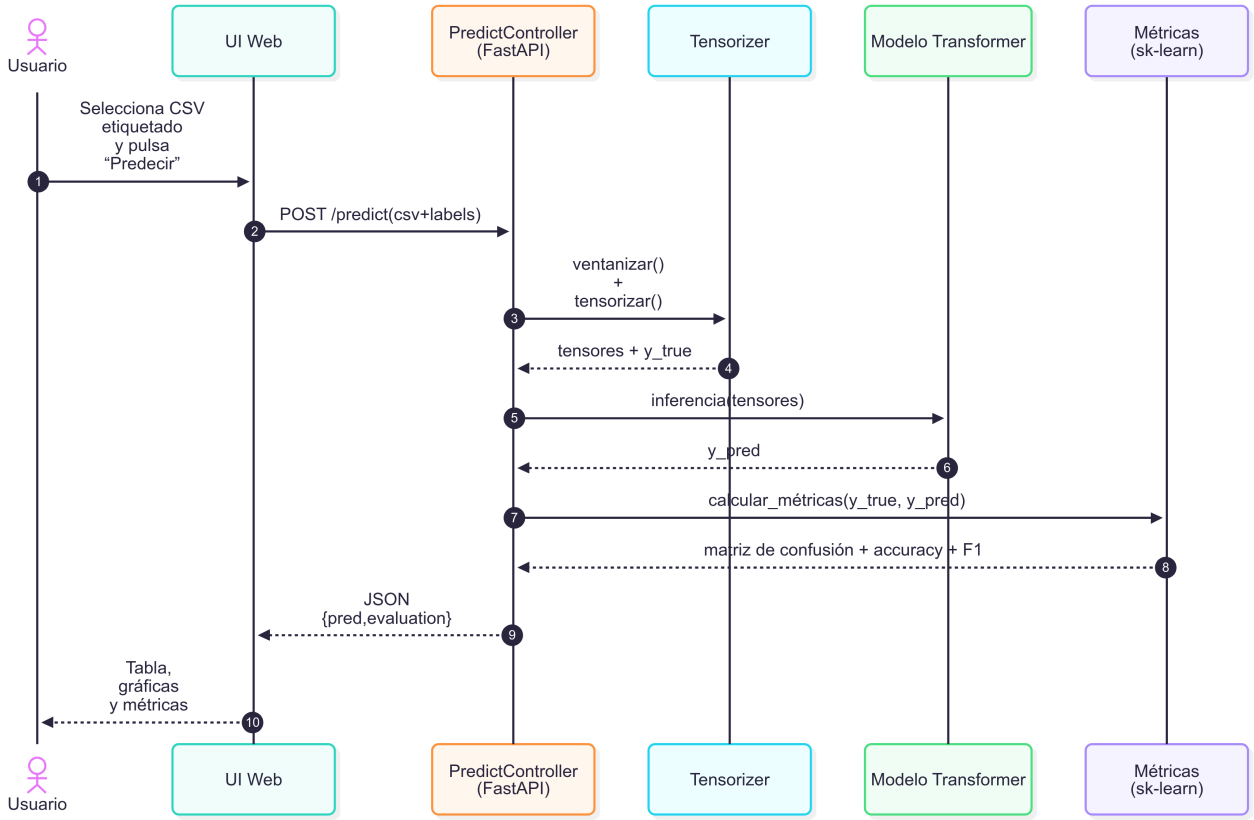

Figura C.4: Diagrama de secuencia para CSV con etiquetas

Apéndice D

Documentación técnica de programación

D.1. Introducción

Este apéndice sirve como guía para cualquier persona que necesite instalar, ejecutar o ampliar el IDS propuesto. Incluye: (1) la estructura de carpetas, (2) los requisitos de software, (3) los pasos de compilación/ejecución y requisitos hardware y (4) las pruebas de calidad con las que el IDS ha sido evaluado para saber si cumple con los requisitos no funcionales exigidos.

D.2. Estructura de directorios

```
sergiomartinreizabal-tfg-gii-transformer-series-temporales-trafico-iot/
|-- GeneradorMixto.py
|-- GenerarVentanas.ipynb
|-- TransformerVentanasFinal.ipynb
|-- requirements.txt
|-- app/
|   |-- main.py
|   `-- modelo/
|       |-- arch.py
|       |-- utils.py
|       `-- export/
|           |-- label_map.json
|           |-- numeric_cols.json
|           |-- model.pth
|           `-- scaler.pkl
|-- frontend/
|   `-- index.html
|-- TrazasPrueba/
|   `-- Benigno_PortScan.csv
`-- documentacion/
    |-- IDS-Transformer.mov
    |-- memoria.pdf
    `-- anexos.pdf
```

`GeneradorMixto.py` Script que crea trazas de tráfico de red mixto (mezclando varios tipos de tráfico) para pruebas rápidas.

`GenerarVentanas.ipynb` Notebook que muestra el algoritmo de ventanización paso a paso y genera CSVs con la columna `Ventana_Inicio`.

`TransformerVentanasFinal.ipynb` Notebook de entrenamiento: carga datos, ajusta el `StandarScaler`, entrena el Transformer, exporta los artefactos al directorio `export/` y evalua el modelo en el conjunto de test.

`requirements.txt` Lista de dependencias Python (*FastAPI*, *PyTorch*, *pandas*, etc.).

`app/` Código fuente del servicio.

arch.py Arquitectura del Transformer (`TransformerEncoderClassifierWithCLS`) y sus submódulos internos.

utils.py Funciones para agrupar flujos en ventanas, convertir a tensores y la clase `WindowDataset`.

export/ Artefactos entrenados.

label_map.json Mapeo etiqueta → índice.

numeric_cols.json Las 79 columnas numéricas del entrenamiento, en el mismo orden.

scaler.pkl `StandardScaler` ya ajustado.

model.pth Los pesos del Transformer ya entrenado. Esto nos permite poner en marcha el modelo en cualquier momento sin tener que volver a entrenarlo.

frontend/ Incluye todo lo relacionado con la UI.

index.html Web dinámica e interactiva completa con tabla, gráficas y métricas. Ej JavaScript, CSS y Bootstrap vienen incluidos en el mismo fichero.

TrazasPrueba/ Incluye trazas de red nunca antes vistas por el modelo durante su entrenamiento, procesadas por CICFlowMeter en formato CSV.

Benigno_PortScan.csv Traza de red con tráfico benigno en la que se realiza un escaneo de puertos periódico cada minuto. Útil para cargar en la web y ver si el modelo es capaz de detectar el ataque de escaneo de puertos.

documentación/ Carpeta reservada para subir la documentación final.

memoria.pdf Documento con la memoria principal.

anexos.pdf Recopilación de todos los anexos.

D.3. Manual del programador

En esta sección se explica cómo preparar el entorno de trabajo y qué herramientas utilizar durante el desarrollo.

Preparar entorno Python

1. Clonar el repositorio:

```
git clone https://github.com/SergioMartinReizabal/TFG-GII-Transformer
    -Series-Temporales-Trafico-IoT.git
cd TFG-GII-Transformer-Series-Temporales-Trafico-IoT
```

2. Crear y activar un *virtualenv*:

```
python -m venv .venv
source .venv/bin/activate        # Windows: .venv\Scripts\activate
```

3. Instalar las dependencias de desarrollo:

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

Herramientas auxiliares necesarias

Como se comenta en la memoria de este Trabajo Fin de Grado, debido a limitaciones de tiempo no se ha podido automatizar la extracción de flujos a partir de un archivo PCAP en el pipeline propuesto. Por ello se hace necesaria la herramienta de extracción de flujos CICFlowMeter [5].

D.4. Compilación, instalación y ejecución del proyecto

Esta sección describe los pasos necesarios para poner en marcha el IDS propuesto en un entorno local.

Arranque rápido en local

1. Asegurarse de tener el entorno virtual activo y tener las dependencias instaladas (Ver Sección D.3).

2. Lanzar el backend con recarga automática:

```
uvicorn app.main:app --host 0.0.0.0 --port 8000 --reload
```

3. Abrir el navegador e ir a la dirección `http://localhost:8000` para cargar la web.

Requisitos de hardware

El IDS propuesto, esta configurado de tal forma que si detecta una GPU NVIDIA con soporte para CUDA, ejecuta el modelo desde la gráfica con tiempos de inferencia mínimos para cada ventana de 5 s (< 1 ms). Sin embargo, esta latencia aumenta considerablemente al ejecutar el modelo desde una CPU.

El Transformer entrenado fue concebido para ser un modelo ligero que pudise ser implementado en sistemas embebidos o externos con recursos limitados, por ello se diseño un Transformer con 72 454 parámetros que ocupa en memoria RAM o VRAM aproximadamente 0.28MB.

D.5. Pruebas del sistema

La calidad del IDS se verifica con un conjunto de 5 228 ventanas de trafico mixto estratificado nunca antes vistas durante el entrenamiento. Según se especifica en la Sección B.3 para que el IDS cumpla con los estándares de calidad propuestos tiene que:

- Tener una exactitud global alta.
- Tener un F1 ponderado alto.
- Que el tiempo de inferencia por ventana sea menor de 1 ms.

El cuaderno `TransformerVentanasFinal.ipynb` realiza la división estratificada, exporta el `test_loader` y ejecuta la evaluación del IDS (Transformer). Los resultados son los siguientes:

- Precisión global de 98,83 %.
- F1 ponderado de 98,85 %.
- Tiempo de inferencia promedio por ventana de 0.23 ms.

Esto confirma que el IDS propuesto cumple con los criterios establecidos de calidad y rendimiento, al alcanzar una exactitud elevada y un F1-score ponderado alto en la detección de todos los tipos de ataque.

Apéndice E

Documentación de usuario

E.1. Introducción

Este apéndice describe, desde el punto de vista del usuario final, cómo utilizar la interfaz web del IDS. La instalación del servicio (clonado del repositorio, creación del entorno virtual y arranque de `uvicorn`) se detalla en el Apéndice D, Sección D.4. Una vez el servidor está en marcha (`http://localhost:8000`), el usuario puede cargar un CSV y visualizar en unos pocos segundos la clasificación del tráfico IoT en ventanas de 5 s.

E.2. Requisitos de usuarios

- Conocimientos básicos de redes (saber qué es un CSV de flujos).
- Un navegador moderno (Chrome, Firefox, Edge) con JavaScript habilitado.
- Opcional: Tener cierta familiaridad con CICFlowMeter para generar los CSV a partir de capturas de red (PCAP).

E.3. Instalación

Los pasos para desplegar el sistema (clonar, instalar dependencias y lanzar `uvicorn`) son idénticos a los del Apéndice D. Cuando el backend esté escuchando, abrimos el navegador en:

`http://localhost:8000`

A partir de aquí se sigue el manual descrito en la sección siguiente.

E.4. Manual del usuario

Carga y predicción

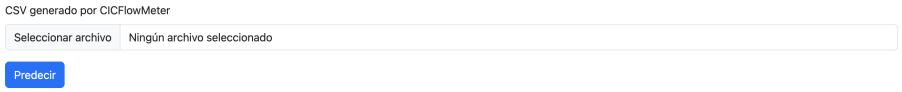

Figura E.1: Pantalla inicial. Seleccione un CSV de CICFlowMeter y pulse *Predecir*.

1. Pulse *Examinar*, elija el CSV y haga clic en *Predecir*.
2. Aparece un mensaje 'Procesando...' mientras el backend ventaniza, tensoriza y clasifica.
3. Tras unos segundos, la tabla y las gráficas se rellenan automáticamente.

Tabla de ventanas

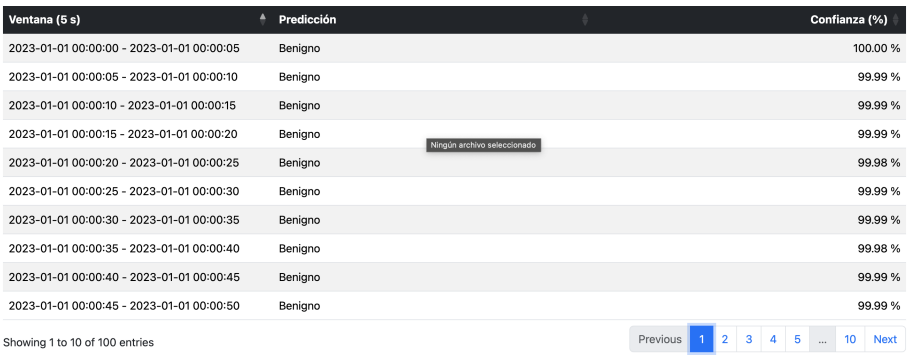

Figura E.2: Tabla con las ventanas de 5 s, la etiqueta predicha y la confianza (0–100 %).

La cabecera permite ordenar por tiempo, etiqueta o confianza; el buscador filtra por intervalo o clase.

Gráfica de etiquetas predichas

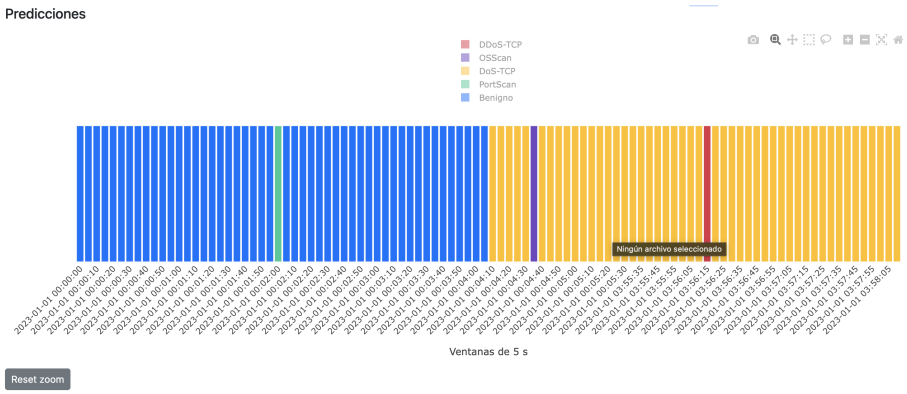

Figura E.3: Distribución temporal de las etiquetas predichas. Cada barra representa una ventana de 5 s, coloreada según la clase predicha más probable.

Se permite hacer zoom con el ratón; el botón 'Reset zoom' devuelve la vista completa.

Etiquetas reales, matriz de confusión y métricas

Si el CSV contiene la columna `Label`, la interfaz muestra:

1. Una gráfica adicional con las etiquetas reales (Fig. E.4).
2. La matriz de confusión (Fig. E.5).
3. Un cuadro con **accuracy** y **F1 ponderado**.

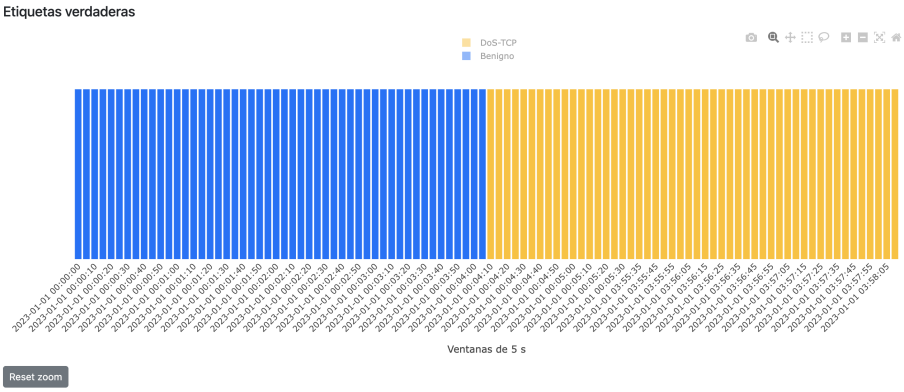

Figura E.4: Distribución temporal de las etiquetas reales. Cada barra representa una ventana de 5 s, coloreada según la clase real.

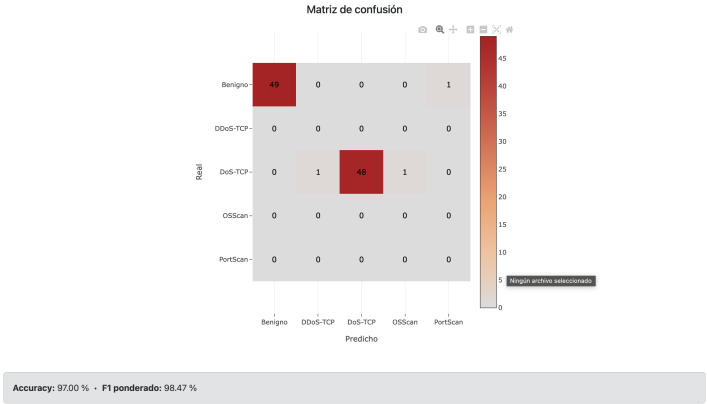

Figura E.5: Matriz de confusión y métricas de evaluación (solo CSV etiquetado).

Solución de problemas

CSV rechazado Si el backend devuelve 'HTTP 400 – Solo se aceptan archivos .csv'. Comprobar que el fichero tenga la extensión `.csv` y las 79 columnas numéricas dextraidas por CICFlowMeter.

Figuras vacías Ocurre cuando ninguna fila contiene datos válidos (valores `NaN` o infinitos). Generar de nuevo el CSV.

Latencia elevada Sin GPU la clasificación es más lenta, pero igualmente válida; usar archivos más pequeños si necesitamos una respuesta más rápida.

Vídeo explicativo

Además de este manual, en la carpeta `Documentación` incluida en los pendrives entregados a cada miembro del tribunal, se encuentra el vídeo `IDS-Transformer.mov`. Este vídeo, elaborado por mí, muestra una demostración interactiva completa del funcionamiento del IDS desde la perspectiva del usuario, explicando detalladamente cómo utilizar la interfaz web, cargar trazas de red, realizar predicciones e interpretar los resultados visuales.

Se recomienda encarecidamente visualizar este recurso como complemento al presente manual.

Apéndice F

Anexo de sostenibilización curricular

F.1. Reflexión sobre sostenibilidad

A lo largo del desarrollo de este TFG he podido aplicar algunas ideas clave relacionadas con la sostenibilidad que se mencionan en las directrices de la CRUE para los estudios universitarios[4]. Aquí dejo una pequeña reflexión sobre ello.

Pensar en el sistema completo

Aunque mi TFG es un sistema de detección de intrusos en red (IDS) basado en Transformers, he intentado pensar más allá de si el modelo funciona o no. Me he dado cuenta de que hay que tener en cuenta todo el sistema: los dispositivos IoT, la red doméstica, el consumo de recursos y también la privacidad de los datos. Todo está conectado y no tiene sentido centrarse solo en optimizar métricas si luego el impacto ambiental o ético es negativo.

Contribución a algunos ODS

- **ODS 9 – Infraestructura e innovación**: el proyecto está hecho con herramientas libres y accesibles, lo que lo hace replicable y fomenta la innovación abierta.

- **ODS 13 – Acción climática**: el modelo tiene pocos parámetros y ejecuta rápido, lo que implica menos consumo de energía y evita depender de centros de datos externos.

- **ODS 16 – Instituciones sólidas**: al proteger mejor la red doméstica y evitar posibles ataques, se mejora la confianza en la tecnología.

Consumo responsable y hardware

No he necesitado comprar ningún equipo nuevo: he trabajado todo el tiempo con mi propio PC de sobremesa. Esto me ha hecho pensar en cómo se pueden desarrollar soluciones útiles sin necesidad de estar cambiando constantemente de hardware, alargando su vida útil y reduciendo residuos.

Privacidad y datos

El modelo trabaja solo con estadísticas de red (79 atributos por flujo), sin leer el contenido de los paquetes, por lo que no maneja datos personales. Creo que eso es importante: la privacidad tiene que estar presente desde el principio, no añadirse después. Además, al compartir el código bajo licencia MIT, cualquier persona puede auditarlo y adaptarlo a su propio entorno.

Reutilización y comunidad

También he intentado aprovechar librerías y datasets abiertos. No solo por ahorrar tiempo, sino porque apoyarse en lo que ya existe ayuda a evitar duplicidades y a que el conocimiento circule mejor. El proyecto se puede ampliar sin tener que empezar de cero, lo que encaja bien con ideas de sostenibilidad como la reutilización o la economía circular.

Lo que me llevo

- Que sostenibilidad no es solo reciclar, también significa diseñar sistemas que duren, sean eficientes y respeten los derechos de las personas.

- Que la privacidad es un derecho básico y no se puede ignorar, ni siquiera en proyectos técnicos.

- Que compartir lo que uno hace ayuda a que otros aprendan y lo mejoren.

Mirando al futuro

Después de este TFG me queda claro que quiero seguir desarrollando soluciones que tengan en cuenta no solo lo técnico, sino también lo social y lo ambiental. Desde elegir mejor las herramientas hasta pensar en cómo reducir el impacto de lo que programamos.

En resumen, este proyecto me ha servido para darme cuenta de que se puede hacer tecnología útil, eficiente y respetuosa con las personas y el entorno. Y eso me parece una buena dirección para mi futuro profesional.

Bibliografía

[1] Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos (RGPD). DOUE L 119, 4 may 2016, pp. 1–88, 2016.

[2] Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales. BOE núm. 294, 6 dic 2018, 2018.

[3] Reglamento (UE) 2021/821 del Parlamento Europeo y del Consejo, de 20 de mayo de 2021, por el que se establece un régimen de la Unión para el control de las exportaciones, el corretaje, la asistencia técnica, el tránsito y la transferencia de productos de doble uso. DOUE L 206, 11 jun 2021, pp. 1–461, 2021.

[4] Conferencia de Rectores de las Universidades Españolas (CRUE). Directrices para la introducción de la sostenibilidad en el curriculum. https://www.crue.org/wp-content/uploads/2020/02/Directrices_Sosteniblidad_Crue2012.pdf, 2012. Documento aprobado por CADEP y revisado en 2012. Consultado el 3 jul 2025.

[5] Arash Habibi Lashkari et al. Cicflowmeter (iscxflowmeter) V4.0: A network traffic flow generator and analyzer. https://github.com/ISCX/CICFlowMeter, 2018. Canadian Institute for Cybersecurity, UNB.

[6] Euclides Carlos Pinto Neto, Sajjad Dadkhah, Raphael Ferreira, Alireza Zohourian, Rongxing Lu, and Ali A. Ghorbani. Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment. *Sensors*, 23(13):5941, 2023.

[7] Open Source Initiative. The MIT License. https://opensource.org/license/mit/, 1988. Licencia de software permisiva OSI-aprobada.