



UNIVERSIDAD DE BURGOS  
ESCUELA POLITÉCNICA SUPERIOR  
Grado en Ingeniería Informática



TFG del Grado en Ingeniería  
Informática

Estrategias de seguridad para  
dispositivos Internet de las  
cosas industriales (IIoT)



Presentado por Víctor De Marco Velasco  
en Universidad de Burgos — 7 de julio de 2025  
Tutor: Carlos Cambra Baseca



---

# Índice general

---

<b>Índice general</b>	<b>i</b>
<b>Índice de figuras</b>	<b>iii</b>
<b>Índice de tablas</b>	<b>iv</b>
<b>Apéndice A Plan de Proyecto Software</b>	<b>1</b>
A.1. Introducción . . . . .	1
A.2. Planificación temporal . . . . .	1
A.3. Estudio de viabilidad . . . . .	6
<b>Apéndice B Especificación de Requisitos</b>	<b>11</b>
B.1. Introducción . . . . .	11
B.2. Objetivos generales . . . . .	11
B.3. Catálogo de requisitos . . . . .	11
B.4. Especificación de requisitos . . . . .	14
<b>Apéndice C Especificación de diseño</b>	<b>23</b>
C.1. Introducción . . . . .	23
C.2. Diseño de datos . . . . .	23
C.3. Diseño arquitectónico . . . . .	24
C.4. Diseño procedimental . . . . .	27
<b>Apéndice D Documentación técnica de programación</b>	<b>31</b>
D.1. Introducción . . . . .	31
D.2. Estructura de directorios . . . . .	31
D.3. Manual del programador . . . . .	33

D.4. Compilación, instalación y ejecución del proyecto . . . . .	34
D.5. Pruebas del sistema . . . . .	36
<b>Apéndice E Documentación de usuario</b>	<b>37</b>
E.1. Introducción . . . . .	37
E.2. Requisitos de usuarios . . . . .	37
E.3. Instalación . . . . .	38
E.4. Manual del usuario . . . . .	38
<b>Apéndice F Anexo de sostenibilización curricular</b>	<b>49</b>
F.1. Introducción . . . . .	49
F.2. Competencias de sostenibilidad adquiridas . . . . .	49
F.3. Conclusiones . . . . .	50
<b>Bibliografía</b>	<b>51</b>

---

# Índice de figuras

---

A.1. Commits realizados en Sprint 3 . . . . .	3
A.2. Commits realizados en el Sprint 4 y 5 . . . . .	5
A.3. Commits realizados en el Sprint 6 . . . . .	6
C.1. Arquitectura del proyecto . . . . .	26
C.2. Diagrama de recepción y almacenamiento de datos. . . . .	28
C.3. Diagrama de carga de archivos csv. . . . .	29
D.1. Imagen de la correcta construcción del contenedor docker . . . .	36
E.1. Imagen de la correcta ejecución de webhook_flask . . . . .	38
E.2. Imagen de la aplicación recibiendo correctamente un paquete . .	39
E.3. Pestaña de login . . . . .	40
E.4. Pestaña de registro . . . . .	40
E.5. Pestaña de recuperar contraseña . . . . .	41
E.6. Pestaña de visualizar las gráficas . . . . .	42
E.7. Pestaña con la lista de ficheros . . . . .	43
E.8. Pestaña para añadir un nuevo fichero . . . . .	44
E.9. Pestaña para analizar un fichero . . . . .	45
E.10. Pestaña para enviar paquetes de datos a la aplicación webhhok .	46
E.11. Imagen de un ejemplo de paquete preparado para ser enviado .	47

---

# Índice de tablas

---

A.1. Resumen de costes estimados durante un año . . . . .	8
A.2. Licencias de herramientas utilizadas en el proyecto . . . . .	9
B.1. CU-1 Recepción y almacenamiento de datos. . . . .	15
B.2. CU-2 Visualización de gráficas. . . . .	16
B.3. CU-3 Gestión de archivos CSV. . . . .	17
B.4. CU-4 Carga de archivos CSV. . . . .	18
B.5. CU-5 Análisis de archivos CSV. . . . .	19
B.6. CU-6 Simulación de envío de datos. . . . .	20
B.7. CU-7 Acceso a la aplicación web. . . . .	21

## *Apéndice A*

---

# **Plan de Proyecto Software**

---

### **A.1. Introducción**

En este apartado se explica la planificación temporal seguida durante el proyecto y el estudio de cuán viable es tanto en el ámbito económico como legal.

### **A.2. Planificación temporal**

La planificación temporal del proyecto se ha llevado a cabo utilizando Sprints haciendo referencia a la metodología Scrum. La duración, el objetivo y cómo cumplir dicho objetivo en cada Sprint deben decidirse entre los integrantes que van a formar parte del proyecto antes de comenzar el Sprint. Cuando llega la fecha límite acordada, se comprueba si ha sido posible realizar todo lo planificado durante la reunión de creación del Sprint. Según lo que se haya logrado realizar, se actualizan los objetivos y tareas del proyecto. Esto continúa hasta que se da por finalizado el proyecto.

### **Sprints**

Este apartado recoge todos los sprints realizados durante el desarrollo del proyecto:

**Sprint 1 (23/03/25 - 14/05/25)**

Este sprint coincide con el comienzo del desarrollo del proyecto. Se centró en recopilar información referente a temas relacionados con el proyecto y la creación del entorno de desarrollo.

- Buscar información sobre redes y dispositivos IoT.
- Buscar información sobre IioT.
- Crear repositorio de github.
- Registro en Overleaf.
- Prototipo de la arquitectura a implementar.

**Sprint 2 (15/05/25 - 15/06/25)**

Este segundo sprint se centró en la configuración y creación de la red LoRaWAN, con el propósito de crear un dataset funcional para luego utilizarlo en las distintas funcionalidades del proyecto.

- Buscar información sobre lo que ofrece The Things Network.
- Registro en The Things Network.
- Crear y configurar una red LoRaWAN.
- Configurar un gateway Dragino y conectarlo a The Things Network.
- Configurar un sensor IoT y conectarlo a The Things Network.
- Crear un payload formatter para decodificar los paquetes enviados por el sensor.
- Crear un webhook para enviar los datos a un sitio donde pudieran ser almacenados.
- Buscar formas de recopilar y almacenar todos los datos enviados por el sensor para crear el dataset.
- Registro en Cloudflare.

Este sería el último sprint no documentado en github debido a que el trabajo realizado durante este tiempo no podía plasmarse en su totalidad en un repositorio github.



**Sprint 3 (16/06/25 - 21/06/25)**

Este sprint se centró en el desarrollo de una aplicación web capaz de recopilar y almacenar los datos enviados por el sensor IoT y crear un prototipo de lo que sería más tarde la aplicación web encargada de interactuar con el usuario. Además, en este sprint se comenzó a desarrollar parte de la documentación del proyecto.

- Creación de la aplicación `webhook_flask`.
- Creación de la aplicación `dashboard_flask`.
- Creación del túnel para conectar el webhook de The Things Network con la aplicación `webhook_flask`.
- Creación del fichero `webhook_dataset`.
- Comenzar a documentar el proyecto.

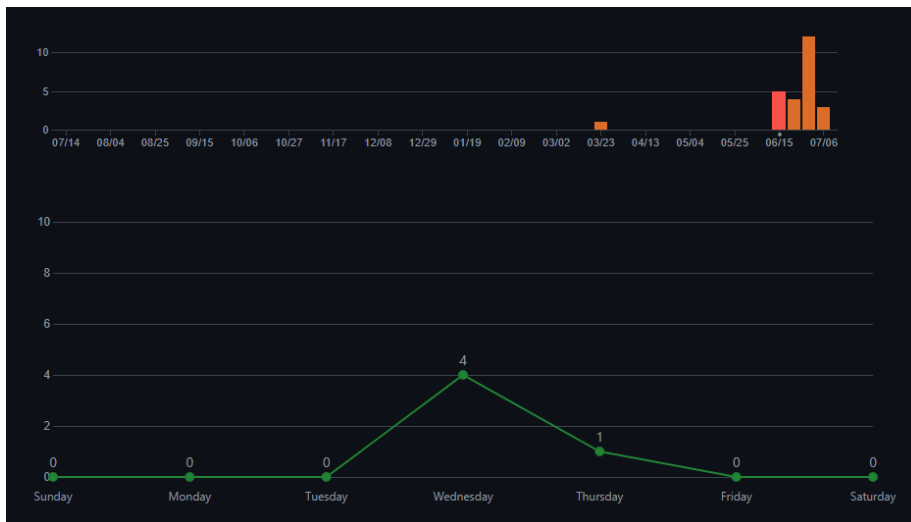


Figura A.1: Commits realizados en Sprint 3

**Sprint 4 (22/06/25 - 25/06/25)**

Este sprint se centró en el desarrollo de las funcionalidades referentes a la aplicación `dashboard_flask`, además de implementar que el proyecto pudiera ejecutarse por contenedor docker.

- Implementar ejecución por contenedor docker.
- Implementar login securizado.
- Implementar elemento común de la estructura html (barra lateral).
- Implementar visualización del listado de archivos csv disponibles.
- Implementar la posibilidad de añadir archivos csv a la aplicación.
- Implementar diferenciación entre los de ficheros disponibles según el usuario que haya iniciado sesión.

### **Sprint 5 (26/06/25 - 29/06/25)**

Este sprint se centró en el desarrollo de la función analizar de la aplicación `webhook_flask`, con el objetivo de que fuera capaz de detectar paquetes recibidos con características sospechosas. Además se añadió a `dashboard_flask` la opción de analizar fichero y de simular el envío de paquetes a `webhook_flask`. Por último, se implementaron algunas mejoras menores a la aplicación `dashboard_flask`.

- Implementar función analizar ficheros recibidos a `webhook_flask`.
- Implementar función infectar y analizar fichero a `dashboard_flask`.
- Mejoras visuales a `dashboard_flask`.
- Implementar pestaña de error a `dashboard_flask`.
- Implementar manejo y notificación de errores al usuario en `dashboard_flask`.
- Mejoras técnicas menores a `dashboard_flask`.

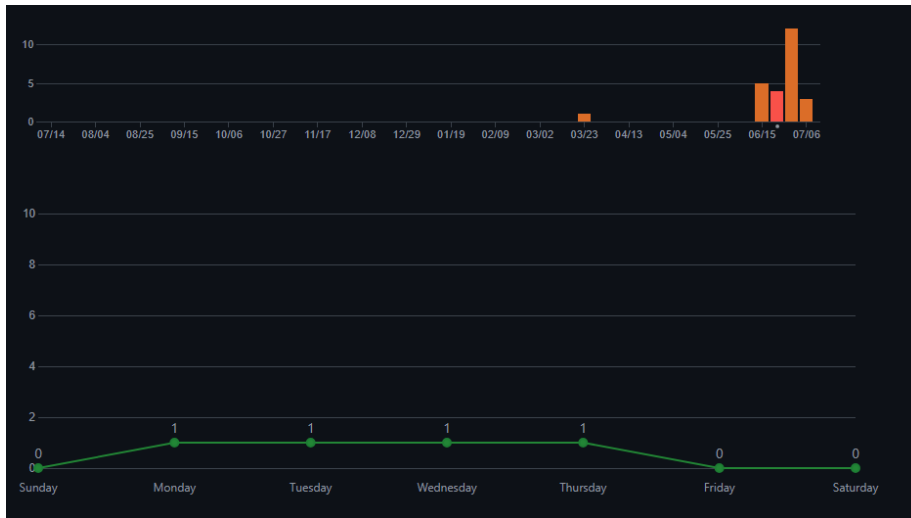
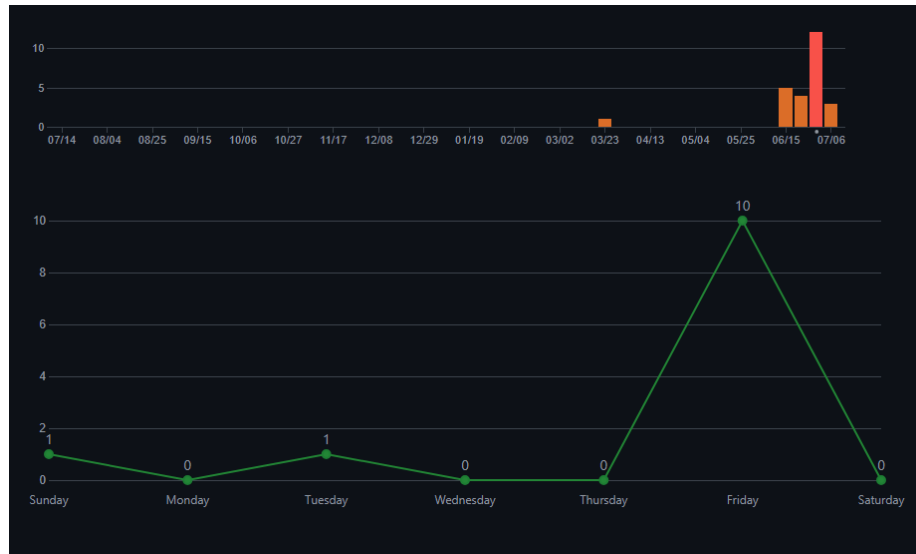


Figura A.2: Commits realizados en el Sprint 4 y 5

### Sprint 6 (30/06/25 - 6/07/25)

Este es el último Sprint del proyecto, duró un día más de lo esperado y se pretendía finalizar el proyecto tanto en apartado técnico como en documentación.

- Implementar pequeño script para obtener las reglas de asociación utilizadas en el proyecto.
- Finalizar la memoria.
- Finalizar los anexos.
- Optimización de alguna parte de código referente a ambas aplicaciones.
- Realización de unos videos explicativos sobre el uso del proyecto.



- Ordenador portátil Asus (4 años): 225€
- Dragino LoRaWAN Gateway (5 años): 33€
- Merry IoT Motion Detection (6 años): 7,5€

Aunque estos dispositivos seguramente podrían seguir usándose pasada la vida útil contemplada, debido a posibles mejoras técnicas y nuevas tecnologías superiores a estos dispositivos, convendría actualizarlos y prescindir de ellos.

### **Costes Software**

Para realizar este proyecto se ha necesitado:

- Sistema operativo Windows 11: 145€
- Dominio web: 4,55€

Aclarar que la compra del dominio web se paga de forma anual y el sistema operativo se amortiza dependiendo del dispositivo en el que esté instalado, por lo tanto dispone de 4 años de vida útil al estar en el ordenador portátil Asus.

### **Coste de Desarrollo**

Considerando que el proyecto lo he desarrollado yo solo, el único coste de desarrollo a tener en cuenta sería mi sueldo.

- Sueldo anual: 6.240€
- Sueldo mensual: 520€

Este sueldo está calculado suponiendo que se trabaja 40 horas mensuales.

### **Costes Adicionales**

De momento no se ha necesitado añadir ningún coste adicional no planteado anteriormente.

### Coste Total

Suponiendo que se llevara a cabo el proyecto durante un año, vamos a calcular el coste del mismo:

Elemento	Coste Anual (€)	Comentario
Hardware: Portatil Asus	225	Vida útil 4 años.
Hardware: Gateway Dragino	33	Vida útil 5 años.
Hardware: Sensor MerryIoT	7,5	Vida útil 6 años.
Software: Dominio web	4,55	Uso exclusivo para el proyecto.
Software: Windows 11	36,25	Licencia del sistema operativo
Horas de desarrollo	6240	Sueldo aproximado
<b>TOTAL</b>	6.546,3	Coste total de un año de desarrollo

Tabla A.1: Resumen de costes estimados durante un año

Una vez observados los cálculos, pretender sacar rédito económico del proyecto parece no ser muy realista. Ya que es un proyecto que individualmente no tiene mucho interés comercial, tendría más interés como parte de un proyecto mucho más grande.

### Viabilidad legal

Para poder determinar si un proyecto es viable legalmente, se deben revisar las licencias utilizadas durante el proyecto y determinar si hay alguna barrera legal que impida el desarrollo y uso del proyecto.

### Protección de Datos

El proyecto desarrollado recoge datos procedentes de un sensor IoT que, si bien no identifican directamente a personas, podrían considerarse datos personales en determinadas circunstancias si permiten inferir comportamientos (por ejemplo, detección de presencia o hábitos). Por eso mismo, en caso de ampliarse el uso de esta solución en entornos reales con usuarios, sería necesario cumplir con los reglamentos y leyes referentes a la protección de datos del lugar donde se fuera a usar:

- Garantizar que la procedencia de los datos sea anónima.
- Limitar el almacenamiento y uso a los fines autorizados.

- Disponer de medidas de seguridad técnicas y organizativas para proteger los datos.

Además, si el usuario utiliza las funciones referentes a la recepción de paquetes enviados desde un servidor web, se expone a estar conectado a la red, pudiendo comprometer su privacidad.

### Licencias y uso de software

El proyecto utiliza las siguientes tecnologías software:

Herramienta	Versión	Licencia
Python	3.11.0	PSF
Flask	2.3	BSD-3-Clause
Pandas	2.3.0	BSD-3-Clause
SQLAlchemy	2.0.41	MIT
Werkzeug	2.3	BSD-3-Clause
Requests	2.32.4	Apache 2.0
Pytz	2025.2	MIT
mlxtend	0.23.4	BSD-3-Clause
IntelliJ IDEA	2024.3.6	Apache 2.0
Docker	28.0.4	Apache 2.0
Docker Desktop	4.40.0	Propietaria
Git	2.45.1	GPLv2
GitHub	N/A	Propietaria
Overleaf	N/A	Propietaria
Draw.io	27.2.0	Apache 2.0
HTML	5	BSD 3-Clause
Bootstrap 5	5.3.0	MIT
Chart.js	4.5.0	MIT

Tabla A.2: Licencias de herramientas utilizadas en el proyecto

Todos los recursos software utilizados permiten su uso académico sin restricción y se respetan las licencias correspondientes.

### Uso de hardware

El hardware utilizado incluye un gateway LoRaWAN Dragino y un sensor MerryIoT de detección de movimiento y condiciones ambientales. Ambos dispositivos se han instalado en mi domicilio, por lo tanto, no se ha

realizado ninguna instalación en un espacio público ni se requiere certificación adicional, ya que ambos operan en bandas ISM de uso libre y conforme a normativa europea.

### **Conclusión**

En conclusión, el proyecto es legalmente viable, no vulnera la normativa de protección de datos ni derechos de terceros, y respeta las condiciones de uso de todo el software y hardware utilizado.



## *Apéndice B*

---

# Especificación de Requisitos

---

### B.1. Introducción

En este apartado se describen los objetivos generales del proyecto, los requisitos tanto funcionales como no funcionales y los distintos casos de uso del software.

### B.2. Objetivos generales

- Diseñar e implementar una solución completa que permite recoger, analizar, almacenar y visualizar los datos generados por un dispositivo IoT conectado a una red LoRaWAN.
- Proporcionar a usuarios no técnicos una interfaz web intuitiva desde la que puedan interactuar con los datos, analizar su contenido y detectar posibles anomalías.

### B.3. Catálogo de requisitos

Una vez claros los objetivos generales del proyecto, podemos definir los requisitos del mismo:

#### Requisitos funcionales

- **RF-1 Recepción de datos:** El sistema debe ser capaz de recoger, analizar y almacenar los datos recibidos:

- **RF-1.1 Recoger datos:** Recibir automáticamente los paquetes enviados por el dispositivo IoT a través de la red LoRaWAN y el webhook configurado.
- **RF-1.2 Analizar datos:** Evaluar los datos recibidos mediante reglas heurísticas y reglas de asociación para determinar si son reales.
- **RF-1.3 Almacenar datos:** Guardar los datos analizados en un archivo CSV que actúa como dataset principal.
- **RF-2 Visualización de gráficas:** El usuario debe poder ver las gráficas correspondientes a los datos almacenados:
  - **RF-2.1 Elegir modo de visualización:** Permitir al usuario seleccionar si que tipo de datos desea visualizar en las gráficas, si solo los datos reales, solo los datos infectados o todos los datos juntos.
  - **RF-2.2 Interacción con las gráficas:** Mostrar gráficas interactivas que permiten al usuario ocultar el contenido de la gráfica o ver el valor correspondiente a un punto de la gráfica.
- **RF-3 Gestión de archivos:** El usuario debe poder interactuar de diferentes formas con los archivos almacenados:
  - **RF-3.1 Ver contenido:** Permitir la visualización del contenido de cada archivo CSV.
  - **RF-3.2 Aplicar archivo:** Activar un archivo específico para que se visualice su contenido en las gráficas.
  - **RF-3.3 Eliminar archivo:** Borrar un archivo CSV del sistema de almacenamiento del usuario.
  - **RF-3.4 Descargar archivo:** Permitir al usuario descargar cualquier archivo CSV que haya subido o generado.
- **RF-4 Cargar archivos:** El usuario debe ser capaz de añadir nuevos ficheros a la aplicación:
  - **RF-4.1 Cargar fichero:** Subir archivos CSV que cumplan con la estructura esperada por la aplicación.
  - **RF-4.2 Analizar fichero:** Procesar automáticamente los datos del archivo cargado para analizar su contenido y otorgar a cada fila de datos el atributo estado.

- **RF-5 Simulación de envío de datos:** El usuario debe tener la opción de enviar datos al dataset:
  - **RF-5.1 Envío de datos manual:** Rellenar un formulario web para generar un paquete con valores personalizados.
  - **RF-5.2 Envío de datos aleatorio:** Generar y enviar automáticamente un paquete simulado con valores aleatorios.
- **RF-6 Gestión de usuarios:** El usuario debe ser capaz de tener su propia cuenta para gestionar sus datos de forma aislada:
  - **RF-6.1 Registrar usuario:** El usuario debe poder registrar una nueva cuenta proporcionando un usuario y contraseña permitidos.
  - **RF-6.2 Inicio de sesión:** El usuario debe poder iniciar sesión en el sistema con sus credenciales.
  - **RF-6.3 Recuperar contraseña:** El usuario debe poder restablecer su contraseña en caso de olvido.
  - **RF-6.4 Cierre de sesión:** El usuario debe poder cerrar su sesión de forma segura.

### Requisitos no funcionales

- **RNF-1 Usabilidad:** La interfaz web debe ser intuitiva y fácil de usar por usuarios sin conocimientos técnicos.
- **RNF-2 Rendimiento:** El tiempo de respuesta del sistema para las acciones comunes (como visualizar gráficas o subir archivos) debe ser razonable.
- **RNF-3 Compatibilidad:** El sistema debe ser compatible con los principales navegadores modernos.
- **RNF-4 Seguridad:** Cada usuario debe tener acceso únicamente a sus propios archivos. La gestión de cuentas debe estar protegida mediante autenticación básica (usuario y contraseña).
- **RNF-5 Portabilidad:** El sistema debe estar preparado para ejecutarse dentro de contenedores Docker para facilitar su despliegue en diferentes entornos.
- **RNF-6 Mantenibilidad:** El código fuente debe estar organizado y documentado para permitir su mantenimiento y evolución. Además, debe estar controlado mediante un sistema de versiones.

- **RNF-7 Escalabilidad básica:** Aunque el sistema está diseñado para un entorno local o pequeño, debe poder ampliarse en el futuro para aceptar nuevos sensores o reglas de análisis.

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

En este apartado se especifican los distintos casos de uso de nuestro proyecto:

<b>CU-1</b>	Recepción y almacenamiento de datos
<b>Versión</b>	1.0
<b>Autor</b>	Víctor De Marco Velasco
<b>Requisitos asociados</b>	RF-1.1, RF-1.2, RF-1.3
<b>Descripción</b>	El sistema recibe automáticamente paquetes enviados por el sensor IoT, los analiza mediante reglas heurísticas y los almacena en un archivo CSV.
<b>Precondición</b>	El sensor debe estar configurado y transmitiendo datos a través de LoRaWAN y TTN. El webhook debe estar activo.
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El sensor IoT envía un paquete a TTN.</li> <li>2. TTN reenvía el paquete al webhook configurado.</li> <li>3. El servidor Flask recibe el paquete.</li> <li>4. El sistema analiza los datos con reglas heurísticas y de asociación.</li> <li>5. El paquete se clasifica como real o infectado.</li> <li>6. El paquete se guarda en un archivo CSV local.</li> </ol>
<b>Postcondición</b>	El archivo CSV es actualizado con una nueva fila correspondiente al paquete recibido y analizado.
<b>Excepciones</b>	Error en la conexión TTN o en el webhook. Paquete malformado. Problemas de almacenamiento en disco.
<b>Importancia</b>	Alta

Tabla B.1: CU-1 Recepción y almacenamiento de datos.

<b>CU-2</b>	Visualización de gráficas
<b>Versión</b>	1.0
<b>Autor</b>	Víctor De Marco Velasco
<b>Requisitos asociados</b>	RF-2.1, RF-2.2, RF-6.2
<b>Descripción</b>	El usuario visualiza datos de temperatura y humedad en gráficas interactivas a partir de los archivos CSV almacenados.
<b>Precondición</b>	El usuario debe estar autenticado.
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario accede a la sección de gráficas.</li> <li>2. Selecciona el tipo de datos (reales, infectados o todos).</li> <li>3. El sistema genera las gráficas correspondientes.</li> <li>4. El usuario puede interactuar con las gráficas: ocultarlas , ver valores exactos, etc.</li> </ol>
<b>Postcondición</b>	Las gráficas se muestran en pantalla correctamente.
<b>Excepciones</b>	Archivo inválido o sin datos. Error en la carga del gráfico.
<b>Importancia</b>	Alta

Tabla B.2: CU-2 Visualización de gráficas.

<b>CU-3</b>	Gestión de archivos CSV
<b>Versión</b>	1.0
<b>Autor</b>	Víctor De Marco Velasco
<b>Requisitos asociados</b>	RF-3.1, RF-3.2, RF-3.3, RF-3.4, RF-6.2
<b>Descripción</b>	El usuario puede ver, activar, eliminar o descargar los archivos CSV que están asociados a su cuenta.
<b>Precondición</b>	El usuario debe haber iniciado sesión correctamente y debe tener archivos disponibles.
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario accede al panel de archivos.</li> <li>2. Visualiza la lista de archivos disponibles.</li> <li>3. Selecciona un archivo para: <ul style="list-style-type: none"> <li>■ Ver su contenido en tabla.</li> <li>■ Aplicarlo para las gráficas.</li> <li>■ Eliminarlo definitivamente.</li> <li>■ Descargarlo localmente.</li> </ul> </li> </ol>
<b>Postcondición</b>	El sistema actualiza el estado del archivo según la acción seleccionada.
<b>Excepciones</b>	Archivo inexistente, error al acceder al almacenamiento o permisos incorrectos.
<b>Importancia</b>	Alta

Tabla B.3: CU-3 Gestión de archivos CSV.

<b>CU-4</b>	Carga de archivos CSV
<b>Versión</b>	1.0
<b>Autor</b>	Víctor De Marco Velasco
<b>Requisitos asociados</b>	RF-4.1, RF-6.2
<b>Descripción</b>	El usuario sube un archivo CSV y el sistema lo almacena en su lista personal".
<b>Precondición</b>	El usuario debe haber iniciado sesión. El archivo debe cumplir con el formato requerido.
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario accede a la sección de carga.</li> <li>2. Selecciona o arrastra un archivo .csv.</li> <li>3. El sistema valida el archivo.</li> <li>4. El archivo queda almacenado en su cuenta.</li> </ol>
<b>Postcondición</b>	El archivo queda almacenado y disponible para ser utilizado por el usuario.
<b>Excepciones</b>	Formato incorrecto o archivo corrupto.
<b>Importancia</b>	Alta

Tabla B.4: CU-4 Carga de archivos CSV.



<b>CU-5</b>	Análisis de archivos CSV
<b>Versión</b>	1.0
<b>Autor</b>	Víctor De Marco Velasco
<b>Requisitos asociados</b>	RF-4.2,RF-6.2
<b>Descripción</b>	El usuario sube un archivo CSV, el sistema lo analiza añadiendo la columna estado y luego lo almacena en su lista personal".
<b>Precondición</b>	El usuario debe haber iniciado sesión. El archivo debe cumplir con el formato requerido.
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario accede a la sección de carga.</li> <li>2. Selecciona o arrastra un archivo .csv.</li> <li>3. El sistema valida el archivo.</li> <li>4. El sistema analiza los datos y genera la columna "estado".</li> <li>5. El archivo queda almacenado en su cuenta.</li> </ol>
<b>Postcondición</b>	El archivo queda analizado y disponible para ser utilizado por el usuario.
<b>Excepciones</b>	Formato incorrecto o archivo corrupto.
<b>Importancia</b>	Alta

Tabla B.5: CU-5 Análisis de archivos CSV.

<b>CU-6</b>	Simulación de envío de datos
<b>Versión</b>	1.0
<b>Autor</b>	Víctor De Marco Velasco
<b>Requisitos asociados</b>	RF-5.1, RF-5.2, RF-6.2
<b>Descripción</b>	El usuario puede simular el envío de paquetes, bien manualmente introduciendo los datos, o automáticamente con valores aleatorios.
<b>Precondición</b>	El usuario debe estar autenticado.
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario accede a la sección de simulación.</li> <li>2. Selecciona si quiere enviar datos manuales o aleatorios.</li> <li>3. Si elige manual, introduce valores personalizados.</li> <li>4. Si elige aleatorio, el sistema genera los datos automáticamente.</li> <li>5. El paquete se analiza y se guarda en el dataset correspondiente.</li> </ol>
<b>Postcondición</b>	El archivo CSV se actualiza con el nuevo paquete generado.
<b>Excepciones</b>	Valores inválidos, error al generar datos, fallo de almacenamiento.
<b>Importancia</b>	Media

Tabla B.6: CU-6 Simulación de envío de datos.

<b>CU-7</b>	Acceso a la aplicación web
<b>Versión</b>	1.0
<b>Autor</b>	Víctor De Marco Velasco
<b>Requisitos asociados</b>	RF-6.1, RF-6.2, RF-6.3
<b>Descripción</b>	El usuario debe poder acceder a la aplicación web con una cuenta personal si así lo desea .
<b>Precondición</b>	
<b>Acciones</b>	<ol style="list-style-type: none"> <li>1. El usuario accede a la dirección de la aplicación web.</li> <li>2. El usuario registra una cuenta si no dispone de una.</li> <li>3. El usuario intenta recuperar la contraseña si no recuerda cual uso para registrarse.</li> <li>4. El usuario procede a iniciar sesión con el usuario y contraseña que uso para registrarse.</li> </ol>
<b>Postcondición</b>	El sistema guarda sus credenciales y le permite acceder a la aplicación.
<b>Excepciones</b>	Credenciales incorrectas o campos vacíos.
<b>Importancia</b>	Alta

Tabla B.7: CU-7 Acceso a la aplicación web.



## *Apéndice C*

---

# Especificación de diseño

---

### C.1. Introducción

En este apartado se documentan los datos , la arquitectura, el procedimiento y la guía de estilo del proyecto.

### C.2. Diseño de datos

El almacenamiento de datos se realiza en un fichero CSV denominado `webhook_dataset.csv`, generado y mantenido por la aplicación Flask. Cada fila del fichero representa un conjunto de datos recibidos de TTN.

Los campos de los datos recibidos son los siguientes:

- **timestamp**: Fecha y hora de recepción del mensaje.
- **occupied (boolean)**: Indica si se detectó movimiento en su campo de visión.
- **button\_pressed (boolean)**: Estado del botón del sensor.
- **tamper\_detected (boolean)**: Detección de presencia.
- **battery\_voltage (float)**: Voltaje actual de la batería (V).
- **temperature\_celsius (int)**: Temperatura medida (°C).
- **humidity\_percent (int)**: Humedad relativa (%).

- **time\_since\_last\_event\_min (int)**: Minutos desde el último evento.
- **event\_count (int)**: Contador de eventos.

Por último, la aplicación añade un campo más (estado) y le otorga el valor correspondiente al resultado obtenido en el análisis del paquete de datos recibido.

Todos los datos son almacenados en texto plano, separados por comas y con codificación UTF-8.

Además, en este proyecto se ha utilizado una base de datos SQL encargada de almacenar y gestionar los credenciales de inicio de sesión de los diferentes usuarios que quieran usar la aplicación.

Por último, cada usuario puede almacenar diversos ficheros csv en su lista de ficheros csv personal. Todo fichero que se encuentre en esa lista podrá ser utilizado por el usuario. para aplicar las diversas funcionalidades de la aplicación. Los ficheros cargados en la aplicación para ser analizados también son guardados en esta lista tras el análisis.

### C.3. Diseño arquitectónico

En este apartado se van a mostrar las distintas estructuras utilizadas durante el desarrollo del proyecto.

#### Modelo-Vista-Controlador (MVC)

El patrón MVC es un patrón de diseño muy utilizado en aplicaciones web. Divide una aplicación en tres componentes principales:

- **Modelo**: Gestiona los datos (modificar, almacenar...).
- **Vista**: Se encarga de mostrar los datos al usuario. Es la parte visual de la aplicación (interfaz).
- **Controlador**: Intermediario entre la vista y el modelo. Recibe las peticiones del usuario, llama al modelo y selecciona la vista adecuada para la respuesta.

## Modelo

En el proyecto, el Modelo estaría representado por las funciones que leen, procesan y almacenan datos en esos archivos o los propios archivos csv donde se almacenan los datos.

## Vista

En el proyecto, la Vista estaría representada en las gráficas que la aplicación muestra al usuario tras aplicar un fichero csv o la visualización del contenido de un paquete antes de que el usuario decida enviarlo al dataset.

## Controlador

En el proyecto, el Controlador estaría representado por los formularios que permiten al usuario introducir valores que serán utilizados por el programa para desempeñar distintas funciones o los botones con los que el usuario puede interactuar para llevar a cabo una acción.

## Frontend y Backend

En el proyecto desarrollado se nota una clara diferenciación entre el frontend y el backend. La primera aplicación ,webhook\_flask, está centrada en el backend, mientras que la segunda aplicación ,dashboard\_flask, se centra en el frontend.

### Frontend

dashboard\_flask es la aplicación encargada de todas las funcionalidades relacionadas con interactuar con el usuario y permitir al mismo disfrutar de la lógica detrás de las funciones del backend sin necesidad de comprender su funcionamiento.

### Backend

webhook\_flask en cambio es la aplicación más centrada en procesamiento de datos y comprobación de que el flujo de datos con la red LoRaWAN esté funcionando correctamente, para permitir así al usuario disfrutar plenamente de las funciones que ofrece el proyecto.

Esta clara separación entre las dos aplicaciones facilita la organización y mantenimiento del proyecto, además de ayudar al desarrollo de nuevas funcionalidades en el futuro.

## Arquitectura del proyecto

La arquitectura final del proyecto desarrollado es:

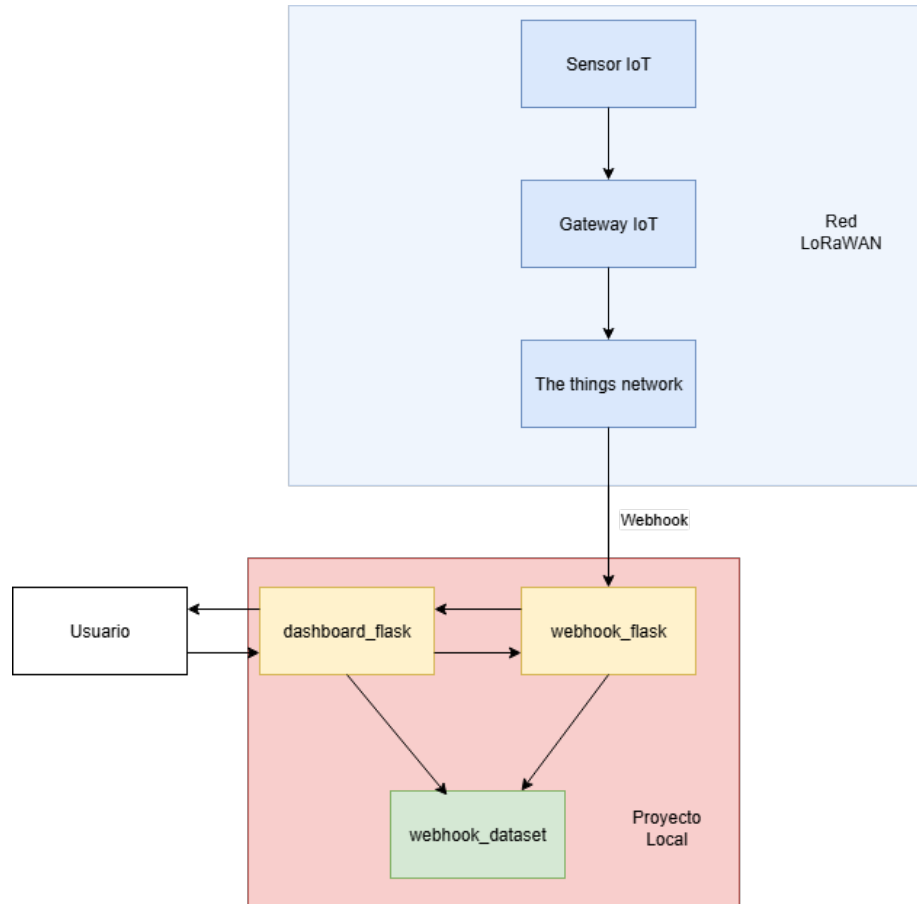


Figura C.1: Arquitectura del proyecto

Los rectángulos azules hacen referencia a la parte del proyecto más relacionada con una red LoRaWAN, es la parte que no he tenido que desarrollar, sino configurar para asegurar su correcto funcionamiento durante el desarrollo del proyecto.

Los rectángulos amarillos hacen referencia a las dos aplicaciones web que he desarrollado para lograr los objetivos del proyecto.

El rectángulo verde representa el archivo csv denominado `webhook_dataset`, el archivo principal del proyecto y con el que ambas aplicaciones interactúan.

Por último, se representa con un rectángulo blanco al usuario, el cual interactúa con el proyecto a través de la primera aplicación web.



## C.4. Diseño procedimental

A continuación se muestran distintos diagramas referentes a distintas acciones que se llevan a cabo durante el proyecto.

La **Figura C.2** representa el caso de recepción y almacenamiento de datos.

La **Figura C.3** representa el caso de carga de archivos csv.

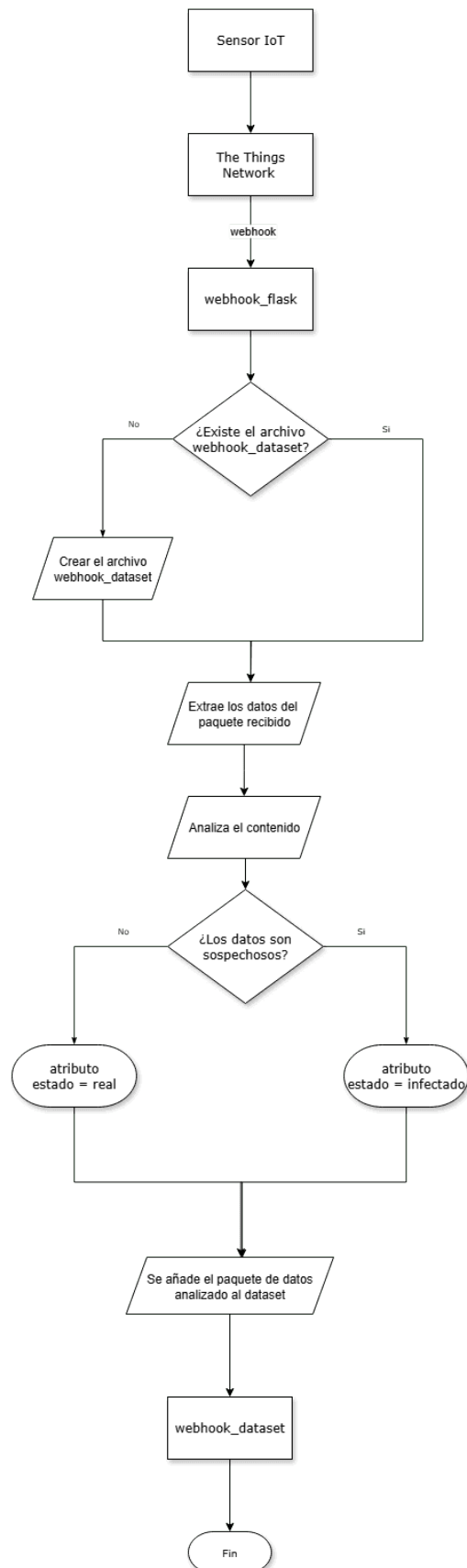


Figura C.2: Diagrama de recepción y almacenamiento de datos.

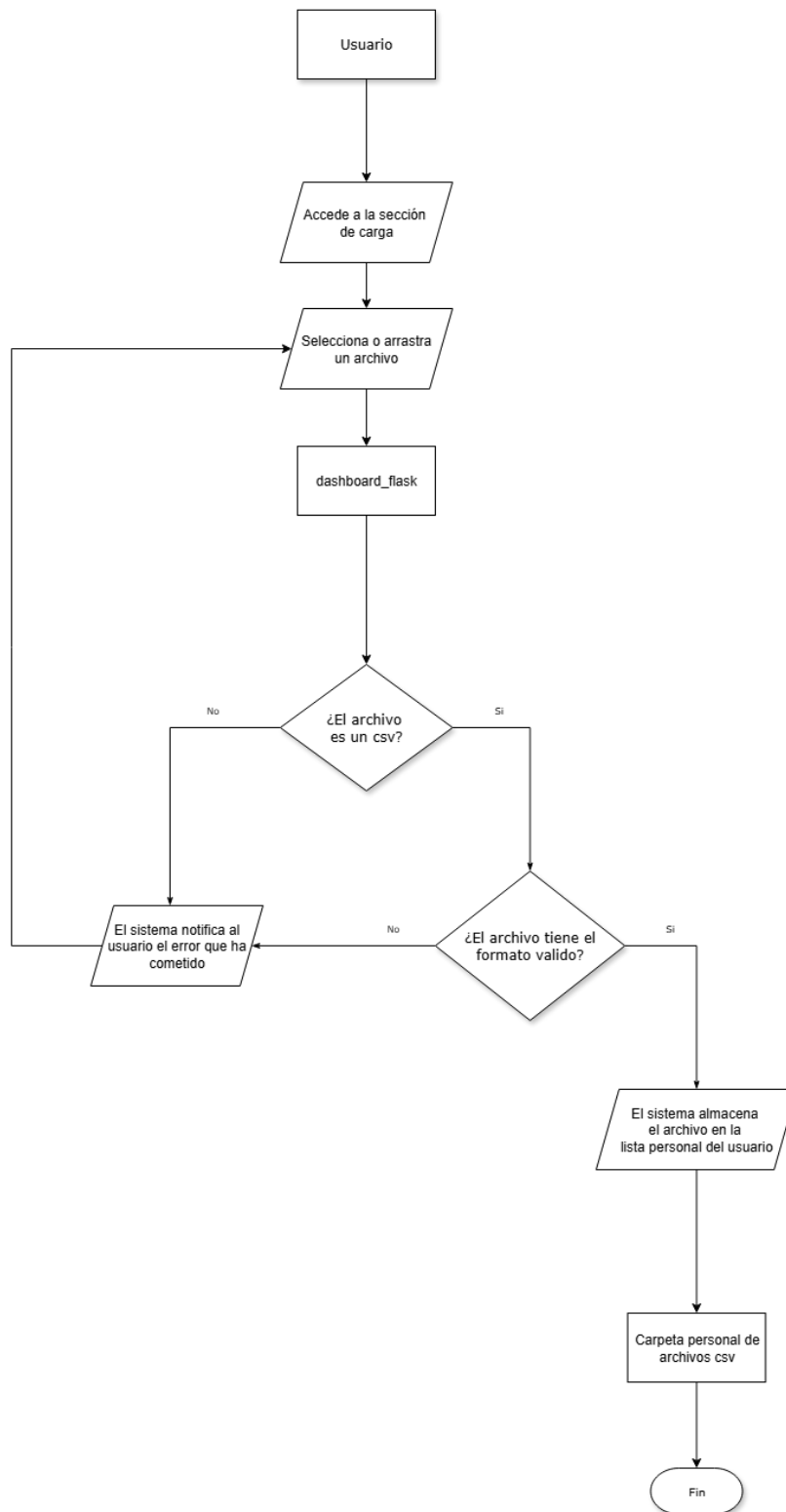


Figura C.3: Diagrama de carga de archivos csv.



## *Apéndice D*

---

# Documentación técnica de programación

---

## D.1. Introducción

En este apartado se va a explicar todo lo relacionado con el código desarrollado durante el proyecto. Desde la estructura de directorios y archivos que componen el proyecto, hasta los pasos necesarios para compilar, instalar y ejecutar el proyecto de diferentes formas.

## D.2. Estructura de directorios

Todo el código relacionado con el proyecto se encuentra en un repositorio público en GitHub<sup>1</sup>.

A continuación, se va a mostrar la estructura de directorios y archivos que se pueden encontrar en el repositorio del proyecto:

- **examples:** directorio que contiene archivos csv de ejemplo para probar las distintas funcionalidades del proyecto.
- **docs:** directorio que contiene los archivos referentes a la documentación del proyecto:
  - **anexo:** directorio que contiene los archivos referentes a los anexos del proyecto.

---

<sup>1</sup>[http://github.com/VictorDeMarco/IoT\\_data\\_gestion](http://github.com/VictorDeMarco/IoT_data_gestion)

- **memoria:** Directorio que contiene los archivos referentes a la memoria del proyecto.
- **videos:** Directorio que contiene Readme.md con los enlaces a los videos explicativos.
- **src:** directorio que contiene los archivos referentes al código fuente del proyecto:
  - **csv:** directorio donde se almacenan todos los ficheros csv utilizados por el proyecto, en este directorio también se crean las carpetas personales de cada usuario que utiliza la aplicación dashboard.
  - **dashboard\_visualizer:** directorio principal de la aplicación de visualización y gestión de archivos csv.
  - **webhook\_receptor:** directorio principal de la aplicación de recepción, análisis y almacenamiento de datos enviados por sensores IoT.
- **docker:** directorio que contiene los archivos necesarios para ejecutar el proyecto en un contenedor docker.
- **.gitignore:** archivo de configuración de Git que contiene los elementos a omitir en el control de versiones.
- **README.md:** archivo de presentación del proyecto donde se resume en qué consiste el proyecto y diversos puntos importantes del mismo.

Dentro del directorio **docker** se encuentran los siguientes archivos:

- **docker-compose.yml:** archivo de configuración que permite configurar y ejecutar múltiples contenedores Docker como un solo proyecto.
- **Dockerfile:** archivo necesario para desplegar el proyecto en un contenedor Docker.
- **requirements.txt:** archivo que contiene las bibliotecas necesarias para ejecutar correctamente el proyecto.

Dentro del directorio **src/dashboard\_visualizer** se encuentran los siguientes archivos y directorios:

- **instance:** directorio donde se guarda la instancia de la base de datos encargada de gestionar los usuarios de la aplicación.

- **routes**: directorio que contiene los archivos de código con las funcionalidades de la aplicación.
- **templates**: directorio que contiene todas las plantillas de las distintas vistas de la aplicación.
- **utils**: directorio que contiene utilidades secundarias para el funcionamiento de la aplicación.
- **dashboard\_flask.py**: archivo principal de la aplicación encargado de iniciar la misma.

Dentro del directorio **src/webhook\_receptor** se encuentran los siguientes archivos y directorios:

- **rules**: directorio donde se guarda un pequeño script encargado de la generación de reglas de asociación.
- **webhook\_flask.py**: archivo principal de la aplicación encargado de iniciar la misma y de todas sus funcionalidades, como recibir y analizar datos.

Dentro del directorio **src/webhook\_receptor/rules** se encuentran los siguientes archivos y directorios:

- **rules.py**: archivo principal con el script encargado de generar las reglas.
- **csv\_rules**: directorio donde se guarda el archivo csv con solo datos reales del sensor IoT, utilizado para generar las reglas de asociación.

## D.3. Manual del programador

En este apartado se explica como crear el entorno de desarrollo necesario para desplegar el proyecto.

Lo primero seria obtener el código completo del proyecto , ya sea descargándolo directamente de GitHub<sup>2</sup> o clonando el repositorio<sup>3</sup> .

Una vez se dispone del código completo del proyecto es necesario instalar las siguientes herramientas:

---

<sup>2</sup>[http://github.com/VictorDeMarco/IoT\\_data\\_gestion](http://github.com/VictorDeMarco/IoT_data_gestion)

<sup>3</sup>git clone [https://github.com/VictorDeMarco/IoT\\_data\\_gestion.git](https://github.com/VictorDeMarco/IoT_data_gestion.git)

- Python  $\geq 3.10.2$  o  $\leq 3.11$
- Docker desktop
- Git
- Navegador web
- IDE (Intellij)
- Acceso a compilador LaTeX

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

El proyecto se puede ejecutar de dos formas, las cuales se van a explicar a continuación.

### Ejecución en local

Primero se va a explicar el método de ejecución en local:

Suponiendo que se han cumplido todos los pasos mencionados anteriormente lo primero que debemos hacer es instalar las dependencias necesarias para el proyecto, para ello ejecutaremos el siguiente comando desde la carpeta principal del proyecto **IoT\_data\_gestion**:

```
pip install -r docker\requirements.txt
```

El siguiente paso es ejecutar el script principal para iniciar cualquiera de las dos aplicaciones web que componen el proyecto.

Para la aplicación webhook:

```
python -m src.webhook_receptor.webhook_flask
```

Para la aplicación dashboard:

```
python -m src.dashboard_visualizer.dashboard_flask
```



Para ejecutar el script que genera las reglas de asociación utilizadas durante el análisis de los paquetes de datos recibidos los pasos son:

Comenzando en la carpeta principal del proyecto nos movemos hasta donde se encuentra el script:

```
cd .\src\webhook_receptor\rules\
```

Y ejecutamos el siguiente comando una vez estamos en la carpeta correspondiente:

```
python rules.py
```

Este script solo puede ser ejecutado de manera local y mostrará en la terminal el resultado de su ejecución.

## Ejecución en Docker

Por último se va a explicar el método de ejecución en Docker:

Lo primero es instalar docker desktop en tu dispositivo, puedes hacerlo desde este enlace:

<https://www.docker.com/products/docker-desktop/>

Asegúrate de descargar el instalador correspondiente a tu sistema operativo.

Ejecutar el instalador y seguir los pasos guiados y una vez finalizada la instalación, reiniciar el sistema si es necesario.

Una vez instalado docker desktop, ejecuta la aplicación.

Luego desde la carpeta principal del proyecto **IoT\_data\_gestion** ejecuta el siguiente comando para componer el contenedor docker con todos los servicios y dependencias necesarias:

```
docker-compose -f docker\docker-compose.yml up
```

Una vez ejecutado el comando en tu docker desktop debería verse algo similar a esto:

Si todo ha funcionado ambas aplicaciones se deben haber ejecutado correctamente y puedes acceder a la aplicación con interfaz diseñada para el usuario desde un navegador usando el siguiente enlace:

<http://localhost:50001/>

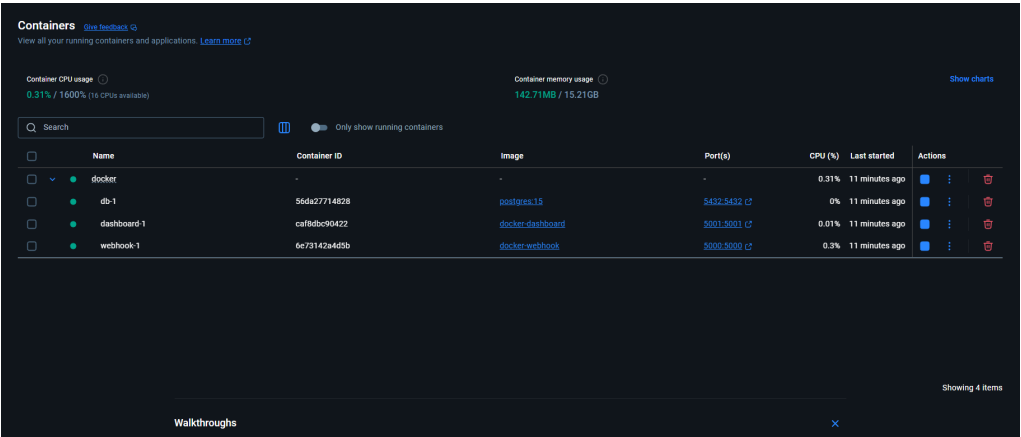


Figura D.1: Imagen de la correcta construcción del contenedor docker

D.5. Pruebas del sistema

Una vez finalizado el proyecto, se realizaron distintas pruebas para asegurar que el sistema funcione correctamente.

En la aplicación **webhook\_flask** se probó que el sistema fuera capaz de soportar recibir peticiones erróneas o mal formuladas, notificándolo por la terminal.

En la aplicación **dashboard\_flask** se realizaron diversas pruebas para asegurar la integridad del proyecto, desde evitar que el usuario intentara dejar en blanco algún campo de los diversos formularios de la aplicación hasta controlar que el usuario no pueda borrar el archivo base del dataset, evitando así problemas severos en la estructura del proyecto.

También se probó que el usuario no pudiera cargar archivos no soportados por la aplicación evitando así un desajuste en el funcionamiento de la aplicación.

## *Apéndice E*

---

# Documentación de usuario

---

## E.1. Introducción

En este apartado se van a explicar los requisitos que necesita el usuario para poder utilizar el software realizado durante el proyecto, además de una guía detallada de las funciones que puede realizar.

## E.2. Requisitos de usuarios

El usuario va a necesitar diferentes requisitos para poder utilizar las aplicaciones del proyecto:

- El usuario necesita un navegador moderno capaz de soportar elementos como HTML5 utilizados en el proyecto.
- Si se pretende ejecutar en un contenedor docker es necesario tener instalado docker desktop.
- Se necesita conexión a internet para poder disfrutar correctamente de la aplicación web, además de para poder ejecutar el proyecto en un contenedor docker.
- Si el usuario pretende crear sus propios archivos csv de forma manual para cargarlos en la aplicación puede utilizar un editor de código.

## E.3. Instalación

### Ejecución en local

Para poder ejecutar correctamente las aplicaciones web en modo local se recomienda al usuario seguir los pasos especificados en el apartado: **Ejecución en local** referente a la documentación del manual del programador.

### Ejecución en Docker

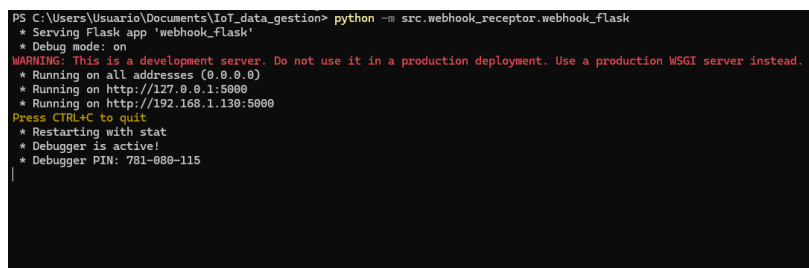
Para poder ejecutar correctamente el proyecto en un contenedor docker se recomienda al usuario seguir los pasos especificados en el apartado: **Ejecución en Docker** referente a la documentación del manual del programador.

## E.4. Manual del usuario

En este apartado se van a explicar en detalle todas las funcionalidades de las que puede disponer el usuario al ejecutar el proyecto. Como el proyecto se compone de dos aplicaciones web se explicaran por separado.

### webhook\_flask

Suponiendo que se hayan seguido de forma correcta todos los pasos indicados anteriormente sobre la instalación y ejecución del proyecto, si iniciamos la aplicación `webhook_flask` la terminal debería verse así:



```
PS C:\Users\Usuario\Documents\IoT_data_gestion> python -m src.webhook_receptor.webhook_flask
* Serving Flask app 'webhook_flask'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://192.168.1.130:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 781-888-115
```

Figura E.1: Imagen de la correcta ejecución de `webhook_flask`

Como se muestra en la imagen, la aplicación actualmente se encuentra ejecutándose en el puerto local 5000. En esta terminal se irán mostrando mensajes referentes a cómo va interactuando la aplicación, ya sea recibiendo, analizando o almacenando paquetes:

```

* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://192.168.1.130:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 781-080-115
❖ Paquete recibido: {'end_device_ids': {'application_ids': {'application_id': 'simulated-app'}, 'device_id': 'simulated-device'}, 'received_at': '2025-07-06T08:46:44.611587Z', 'uplink_message': {'decoded_payload': {'battery_voltage': 3.2, 'button_pressed': True, 'event_count': 4900, 'humidity_percent': 26, 'occupied': True, 'tamper_detected': False, 'temperature_celsius': 47, 'time_since_last_event_min': 33}}}
✅ Paquete almacenado como: infectado
Motivos: Voltaje fuera del parametro establecido
127.0.0.1 - - [06/Jul/2025 10:46:48] "POST /ttn HTTP/1.1" 200 -

```

Figura E.2: Imagen de la aplicación recibiendo correctamente un paquete

En esta imagen se observa que el paquete ha sido recibido correctamente, se muestra el contenido del mismo, además de notificar qué estado se ha otorgado a ese paquete y el motivo por el cual se le ha otorgado dicho estado.

## dashboard\_flask

Suponiendo que se hayan seguido de forma correcta todos los pasos indicados anteriormente sobre la instalación y ejecución del proyecto, si iniciamos la aplicación `dashboard_flask` deberíamos poder acceder a ella a través de esta dirección:

<http://localhost:50001/>

### Inicio de sesión

Lo primero que el usuario visualiza en la aplicación es una pestaña de login securizada que no le permite acceder a ninguna funcionalidad de la aplicación hasta que se registre e inicie sesión. Si el usuario no desea crearse una cuenta personal, puede iniciar sesión con:

nombre de usuario: admin

contraseña: admin

La siguiente imagen muestra la pestaña referente al login donde el usuario debe introducir un usuario y contraseña registrados en la base de datos para poder acceder a la aplicación.

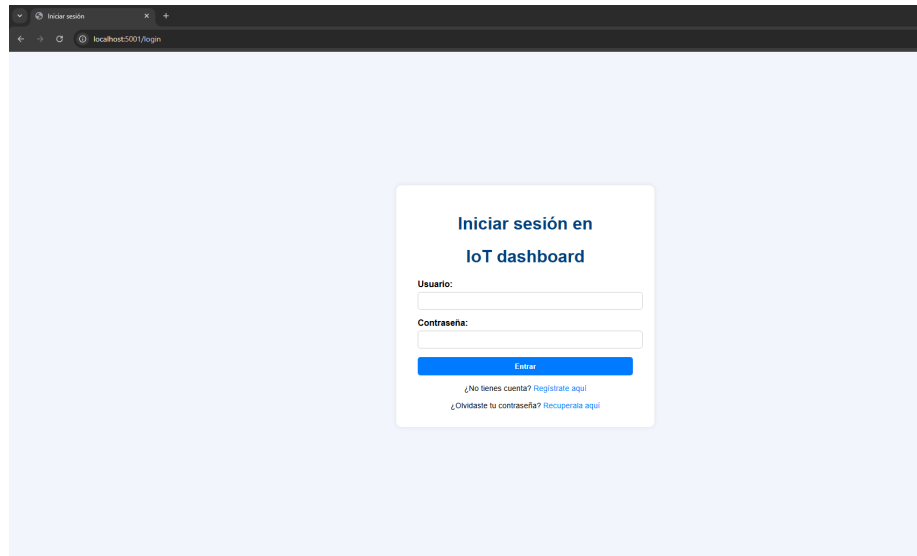


Figura E.3: Pestaña de login

La siguiente imagen muestra la pestaña referente al registro donde el usuario debe introducir un usuario no registrado y una nueva contraseña para poder registrar las credenciales en la base de datos y poder iniciar sesión en el futuro.

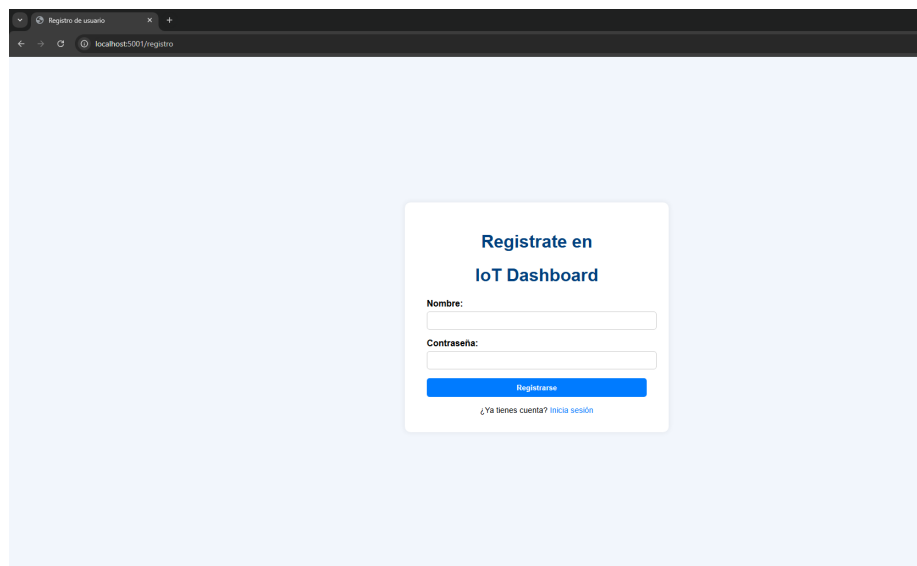


Figura E.4: Pestaña de registro

La siguiente imagen muestra la pestaña referente a recuperar contraseña donde el usuario debe introducir el nombre de usuario del que desea restablecer la contraseña, un archivo csv personal que se encuentre en su carpeta de almacenamiento, para asegurar que realmente la contraseña que desea restablecer pertenece a su cuenta y una nueva contraseña para poder iniciar sesión en la cuenta con la contraseña olvidada.

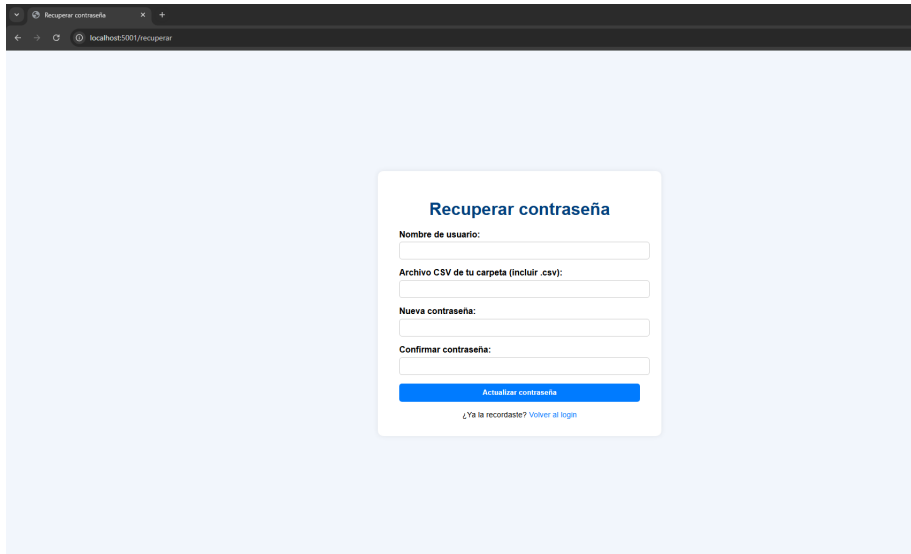


Figura E.5: Pestaña de recuperar contraseña

Para visualizar mejor el funcionamiento de estas pestañas, se recomienda al usuario consultar el Vídeo Login:

[https://github.com/VictorDeMarco/IoT\\_data\\_gestion/tree/main/docs/videos](https://github.com/VictorDeMarco/IoT_data_gestion/tree/main/docs/videos).

### Visualizar gráficas

Una vez iniciada sesión lo primero que ve el usuario son las gráficas correspondientes al fichero base (webhook\_dataset):

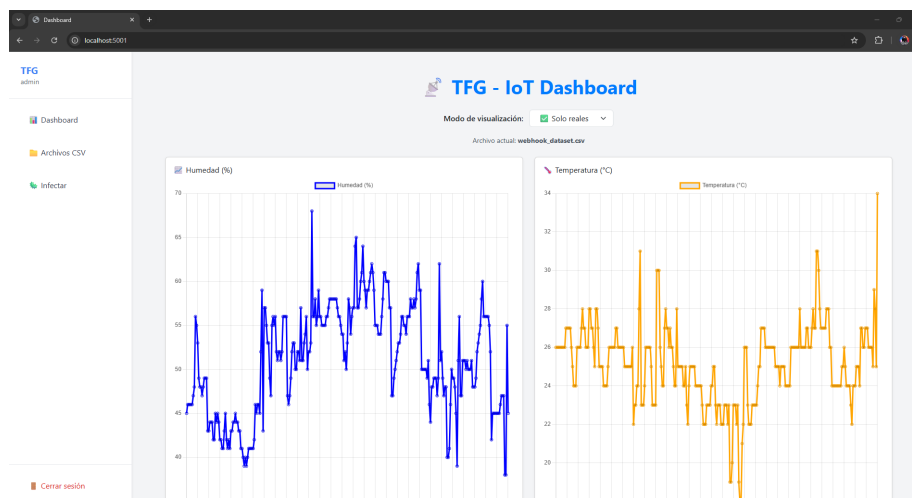


Figura E.6: Pestaña de visualizar las gráficas

En esta imagen ya se empieza a observar la estructura de la aplicación web, con una barra lateral común a todas las pestañas que permite navegar por la aplicación.

En esta pestaña el usuario puede interactuar con las gráficas eligiendo qué modo desea ver de las mismas (solo reales, solo infectados y todos los datos), pulsando en el desplegable **Modo de visualización**.

Además, al pasar con el ratón por encima de la gráfica, puede observar el valor de cada punto y el paquete al que pertenece.

A continuación, se explicará el funcionamiento de la barra lateral. Al ser común a todas las pestañas de la aplicación, lo descrito a continuación se aplica a toda la interfaz.

- Se puede observar debajo de TFG el nombre de usuario que esta utilizando la aplicación actualmente.
- Si se pulsa el botón **Dashboard** te redirige a la pestaña de visualización de gráficas.
- Si se pulsa el botón **Ficheros CSV** te redirige a la pestaña de listado de ficheros.
- Si se pulsa el botón **Infectar** te redirige a la pestaña de simulación de envío de paquetes a la aplicación webhook .



- Si se pulsa el botón **Cerrar sesión** te redirige a la pestaña de inicio de sesión, obligándote a volver a iniciar sesión si deseas continuar usando la aplicación.

Para visualizar mejor el funcionamiento de la visualización de gráficas, se recomienda al usuario consultar el Vídeo Gráficas:

[https://github.com/VictorDeMarco/IoT\\_data\\_gestion/tree/main/docs/videos](https://github.com/VictorDeMarco/IoT_data_gestion/tree/main/docs/videos).

## Ficheros csv

Como se ha mencionado anteriormente, si pulsas el botón **Ficheros CSV** en la barra lateral, llegas a esta pestaña, donde se muestra el listado de ficheros csv almacenados en el proyecto, con los cuales puedes:

- Visualizar su contenido.
- Aplicar su contenido para visualizarlo en las gráficas de la pestaña Dashboard.
- Eliminar el fichero.
- Descargar el fichero.

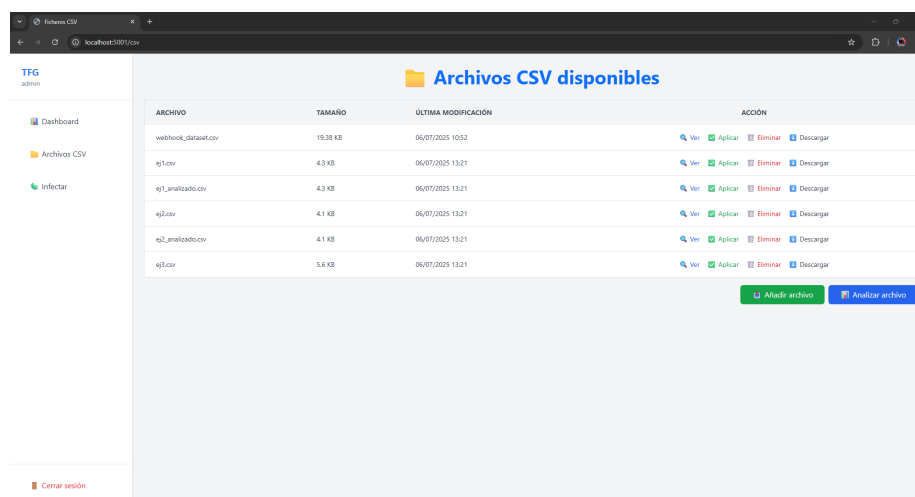


Figura E.7: Pestaña con la lista de ficheros

En esta imagen se muestra la pestaña con la lista de ficheros disponibles en el proyecto, en este ejemplo solo se dispone del archivo base. Además de

las funciones mencionadas en la gestión de archivos, en esta pestaña hay dos funciones más relacionadas con el subir nuevos ficheros csv al proyecto.

La primera de ellas es Añadir fichero:

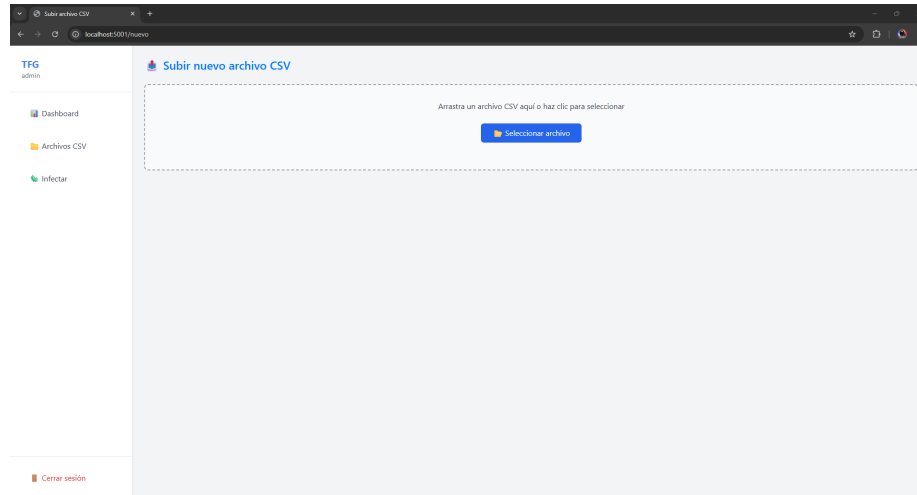


Figura E.8: Pestaña para añadir un nuevo fichero

Después de que en la pestaña con la lista de ficheros el usuario pulse el botón añadir fichero, se le redirige a esta pestaña, donde puede elegir entre arrastrar un archivo desde su dispositivo a la web o pulsar el botón seleccionar archivo, el cual le mostrará los archivos csv que se encuentran en su dispositivo, para que seleccione uno que quiera añadir a la lista.

Después de que el programa analice si el contenido del archivo es compatible con la aplicación, lo añade a la lista personal del usuario o notifica al usuario el error por el cual no se ha podido añadir.

La segunda de ellas es Analizar fichero:

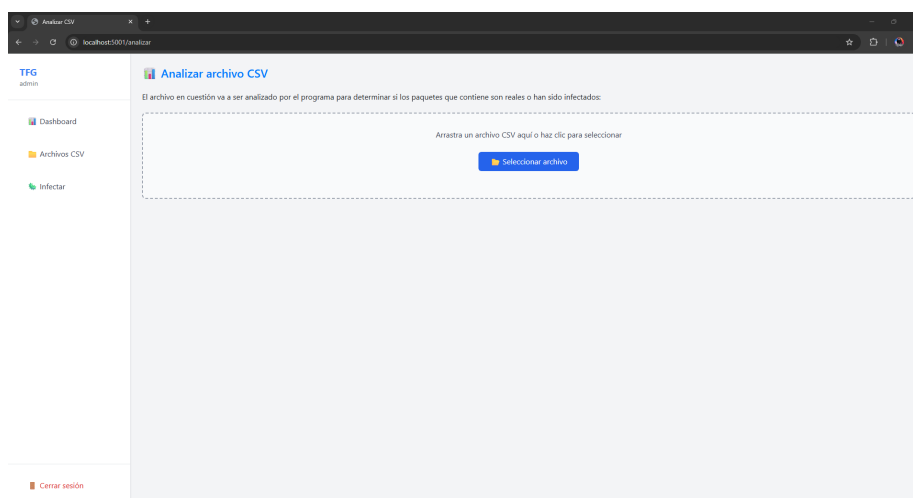


Figura E.9: Pestaña para analizar un fichero

Después de que en la pestaña con la lista de ficheros el usuario pulse el botón analizar fichero, se le redirige a esta pestaña, donde puede elegir entre arrastrar un archivo desde su dispositivo a la web o pulsar el botón seleccionar archivo, el cual le mostrará los archivos csv que se encuentran en su dispositivo, para que seleccione uno que quiera añadir a la lista.

El programa comprueba que el contenido del archivo csv sea válido y después hace uso de la función analizar de la aplicación webhook para determinar el valor del atributo estado de cada fila del fichero csv a analizar.

Una vez analizado correctamente el archivo, lo añade a la lista personal del usuario con el sufijo `_analizado`.

Para visualizar mejor el funcionamiento de la gestión de archivos csv se recomienda al usuario consultar el Vídeo CSV:

[https://github.com/VictorDeMarco/IoT\\_data\\_gestion/tree/main/docs/videos](https://github.com/VictorDeMarco/IoT_data_gestion/tree/main/docs/videos).

## Infectar

Para acceder a esta función, el usuario debe pulsar el botón infectar, que se encuentra en la barra lateral. Este botón te redirige a la pestaña en la cual se puede simular un envío de paquetes de datos a la aplicación webhook.

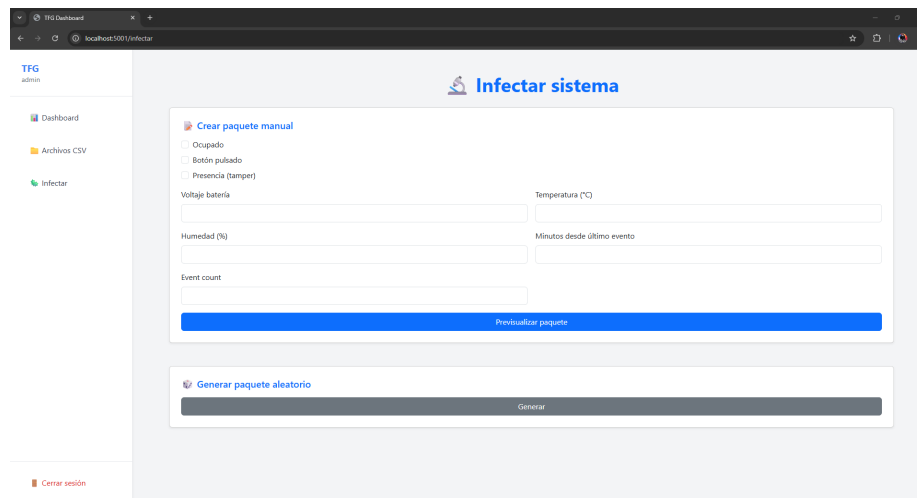


Figura E.10: Pestaña para enviar paquetes de datos a la aplicación webhhok

En esta imagen se muestra que en la pestaña infectar el usuario puede elegir entre dos modos para simular el envío de un paquete de datos, modo manual donde es el usuario el que el contenido del paquete a enviar y generar paquete aleatorio para generar un paquete con datos aleatorios.

En el caso del modo manual no se permite dejar al usuario campos vacíos ni introducir letras en los campos, evitando así que el usuario trate de enviar un paquete de datos con datos que la lógica del sistema no es capaz de procesar, pudiendo terminar en la interrupción del proyecto.

Una vez elegido el modo de generación del paquete, antes de ser enviado se le permitirá al usuario la previsualización del contenido para que pueda confirmar si desea enviar ese paquete a la aplicación webhook.

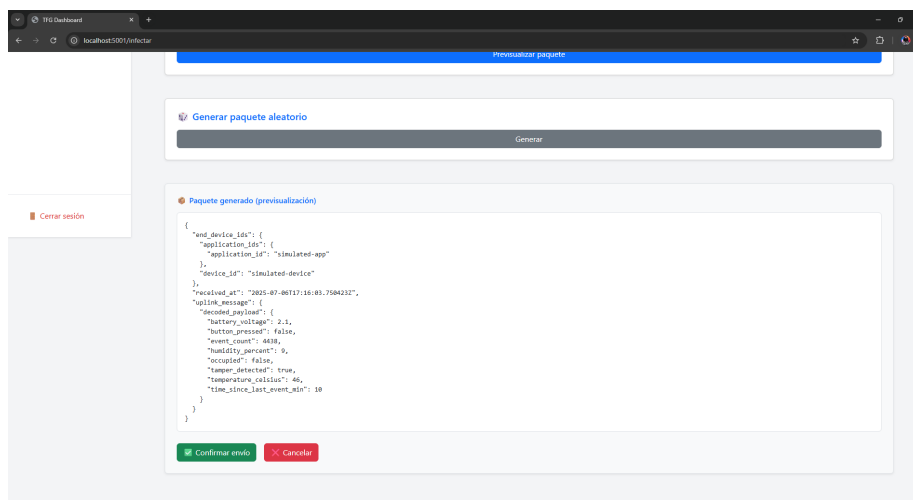


Figura E.11: Imagen de un ejemplo de paquete preparado para ser enviado

Para visualizar mejor el funcionamiento de la simulación de envío de paquetes se recomienda al usuario consultar el Vídeo Infectar:

[https://github.com/VictorDeMarco/IoT\\_data\\_gestion/tree/main/docs/videos](https://github.com/VictorDeMarco/IoT_data_gestion/tree/main/docs/videos).



---

## Anexo de sostenibilización curricular

---

### F.1. Introducción

Este anexo incluirá una reflexión personal del alumnado sobre los aspectos de la sostenibilidad que se abordan en el trabajo.

La reflexión se ha realizado según las directrices estipuladas en el documento sobre la introducción de la sostenibilidad de la CRUE [1].

### F.2. Competencias de sostenibilidad adquiridas

- **Contextualización crítica del conocimiento (SOS1):** En el trabajo he procurado analizar no solo el problema técnico planteado, sino también su relación con el entorno social y su posible repercusión ambiental. Esta visión me ha ayudado a comprender cómo una solución aparentemente eficiente puede no ser sostenible si no se considera su contexto.
- **Uso sostenible de recursos (SOS2):** A lo largo del trabajo he sido consciente de la necesidad de aplicar criterios de eficiencia y reducción del consumo computacional en la fase de pruebas o despliegue, cuando ha sido posible.

- **Participación comunitaria (SOS3):** Durante el desarrollo del proyecto he intentado tener en cuenta las necesidades de posibles usuarios y comunidades relacionadas con el tema tratado en el proyecto. La sostenibilidad también implica diseñar soluciones accesibles, inclusivas y que respondan a demandas reales.
- **Principios éticos (SOS4):** He reflexionado sobre el impacto ético de las tecnologías tratadas, evitando decisiones que pudieran contribuir a la exclusión o la injusticia social. La sostenibilidad y equidad, ha sido un criterio clave en la toma de decisiones.

### **F.3. Conclusiones**

EL desarrollo de este proyecto no solo ha aumentado mis conocimientos técnicos sobre diversos campos de la informática, sino que además he aprendido la importancia de tener en cuenta los principios de sostenibilidad aplicados a la informática a la hora de desarrollar un proyecto.



---

## Bibliografía

---

- [1] CRUE. Directrices para la introducción de la sostenibilidad en el curriculum. [https://www.crue.org/wp-content/uploads/2020/02/Directrices\\_Sostenibilidad\\_Crue2012.pdf](https://www.crue.org/wp-content/uploads/2020/02/Directrices_Sostenibilidad_Crue2012.pdf).