



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

Monitorización sensores IOT



Presentado por Daniel Mellado Hurtado
en Universidad de Burgos — 19 de septiembre
de 2021

Tutor: Bruno Baruque Zanon

Índice general

Índice general	I
Índice de figuras	III
Índice de tablas	v
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	5
Apéndice B Especificación de Requisitos	9
B.1. Introducción	9
B.2. Objetivos generales	9
B.3. Catálogo de requisitos	10
B.4. Especificación de requisitos	12
Apéndice C Especificación de diseño	31
C.1. Introducción	31
C.2. Diseño de datos	31
C.3. Diseño procedimental	35
C.4. Diseño arquitectónico	41
Apéndice D Documentación técnica de programación	45
D.1. Introducción	45
D.2. Estructura de directorios	45
D.3. Manual del programador	47

D.4. Compilación, instalación y ejecución del proyecto	50
D.5. Pruebas del sistema	53
Apéndice E Documentación de usuario	61
E.1. Introducción	61
E.2. Requisitos de usuarios	61
E.3. Instalación	61
E.4. Manual del usuario	64
Bibliografía	81

Índice de figuras

A.1. Gráfico de commits realizados en el tiempo	4
B.1. Diagrama de casos de uso	12
C.1. Sensor del consumo de la bomba de agua	31
C.2. Sensor de la humedad en el ambiente	32
C.3. Sensor de la humedad en la parcela	32
C.4. Sensor de la presión de la tubería de agua	32
C.5. Diagrama de clases paquete Predicción	34
C.6. Diagrama de clases paquete Utilidades	35
C.7. Diagrama de secuencias descarga de datos	36
C.8. Diagrama de secuencias entrenamiento	36
C.9. Diagrama de secuencias predicción	37
C.10. Diagrama de paquetes	42
C.11. Diagrama de despliegue	43
D.1. Directorios del proyecto	46
D.2. Dataset sensor de Presión Tubería de Agua	54
D.3. Dataset sensor Consumo de bomba de agua	54
D.4. Consumo Bomba Agua en las fechas: 28/08/2021 - 29/08/2021 .	56
D.5. Consumo Bomba Agua en las fechas: 25/08/2021 - 26/08/2021 .	56
D.6. Consumo Bomba Agua en las fechas: 05/09/2021 - 06/09/2021 .	56
D.7. Presión tubería de agua en las fechas: 28/08/2021 - 29/08/2021	57
D.8. Presión tubería de agua en las fechas: 25/07/2021 - 26/07/2021	57
D.9. Presión tubería de agua en las fechas: 31/07/2021 - 01/08/2021	57
D.10. dataset 1	59
D.11. dataset 3	59
D.12. dataset 4	59

E.1. inicio PRTG	62
E.2. dispositivos PRTG	63
E.3. ejemplo sensor PRTG	63
E.4. Crear un índice	67
E.5. Index Managment	68
E.6. Index pattern menú inicial si no se encuentran datos en Elasticsearch	70
E.7. Index pattern menú inicial	71
E.8. Creación de un index pattern	71
E.9. Configuración de un index pattern	72
E.10.predicción sensor de Presión Tubería de Agua	77
E.11.pagina inicio kibana	77
E.12.kibana discover	78
E.13.kibana dashboard	78
E.14.kibana filtrados	79
E.15.kibana ayuda de filtrado	79

Índice de tablas

A.1. Costes de personal	5
A.2. Coste hardware	5
A.3. Coste Software	6
A.4. Coste Software	6
A.5. Licencias	7
B.1. Caso de uso 1: Activar sistema.	13
B.2. Caso de uso 2: Desactivar sistema.	14
B.3. Caso de uso 3: Almacenamiento de datos.	15
B.4. Caso de uso 4: Entrenamiento.	16
B.5. Caso de uso 5: Predicción.	17
B.6. Caso de uso 6: Gestión del sistema.	18
B.7. Caso de uso 7: Añadir sensor.	19
B.8. Caso de uso 8: Borrar sensor	20
B.9. Caso de uso 9: Configurar modelo.	21
B.10.Caso de uso 10: Monitorización de datos.	22
B.11.Caso de uso 11: Ver datos reales.	23
B.12.Caso de uso 12: Ver datos entrenamiento.	24
B.13.Caso de uso 12: Ver datos predicción.	25
B.14.Caso de uso 13: Búsquedas filtradas.	26
B.15.Caso de uso 14: Representación gráfica.	27
B.16.Caso de uso 15: Creación de gráficas.	28
B.17.Caso de uso 16: Borrado de gráficas.	29
B.18.Caso de uso 16: Visualización de gráficas.	30
D.1. métricas Consumo Bomba Agua	57
D.2. métricas Presión tubería de agua	58
D.3. métricas dataset	60

Apéndice A

Plan de Proyecto Software

A.1. Introducción

La planificación y el seguimiento de metodologías de desarrollo es esencial para el éxito de cualquier proyecto. En esta sección se describe la planificación temporal que se ha seguido en el proyecto, así como el estudio de viabilidad de este.

A.2. Planificación temporal

Para la realización del proyecto se ha optado por la utilización del método **Scrum**, una metodología ágil de gestión de proyectos en la que se ha dividido el desarrollo en **sprints** de aproximadamente 2 semanas de duración en la que se tenía una reunión con el tutor donde se planteaban las próximas tareas que se iban a realizar.

Para la organización de tareas se utilizó una tabla con 6 columnas:

- **Product Backlog:** En esta columna se colocan las tareas que se han de realizar a lo largo del proyecto.
- **Sprint Backlog:** Al inicio de cada sprint se mueven a esta columna las tareas que se planea realizar durante el transcurso de dicho sprint.
- **In Progress:** En esta columna se encuentran todas aquellas tareas que se están realizando en ese momento.

- **Review/QA:** Cuando en alguna tarea sea necesario realizar una revisión por parte del propio desarrollador o por parte del tutor, se mueve a esta columna.
- **Done:** Cuando las tareas ya han sido finalizadas se mueven a esta columna
- **IceBox:** En esta última columna se colocan las tareas que han sido abandonadas o que tienen muy baja prioridad.

La planificación de las tareas se puede consultar en GitHub en el repositorio: <https://github.com/dmh1001/TFG-Monitorizacion-IOT>

Debido a la situación actual del COVID-19 las reuniones se llevaron a cabo mediante videollamadas por **Microsoft Teams**

Sprint 0 - Introducción (30/11/2020 - 14/12/2020)

Durante este primer **Sprint** lo principal fue investigar sobre las posibles herramientas que se podrían utilizar en este proyecto así cómo documentar dichas herramientas en la memoria.

Durante esta parte del proyecto se creó el repositorio de Github y se instaló **ZenHub** para gestionar las tareas del proyecto.

Sprint 1 (14/12/2020 - 04/01/2021)

Durante este **Sprint** se especificaron cuáles serían los objetivos del proyecto, se decidieron las herramientas que se iban a utilizar y se comenzó con su documentación en la memoria.

Sprint 2 (04/01/2021 - 06/02/2021)

Durante este **Sprint** se instaló el servidor PRTG en una máquina virtual Windows 10.

Sprint 3 (06/02/2021 - 25/02/2021)

Durante este **Sprint** se instaló Elasticsearch, programa el cual serviría de BBDD y monitorización de nuestros sensores, en un equipo con sistema Window y se procedió a conectarlo con PRTG.

Sprint 4 (25/02/2021 - 12/03/2021)

Durante este **Sprint** se migró Elasticsearch a una máquina Virtual Ubuntu server para poder simular un servidor real.

Sprint 5 (12/03/2021 - 26/03/2021)

Durante este **Sprint** se fue estructurando y creando el manual de instalación con lo desarrollado hasta la fecha.

Sprint 6 - Conectividad PRTG-ELK (28/03/2021 - 12/04/2021)

En este **Sprint** se finaliza con la conexión entre PRTG y Elasticsearch, se idea una forma de transformar los datos procedentes de PRTG mediante un sencillo programa en Flex para que los datos sean más fáciles de leer por Logstash.

Con los datos obtenidos y almacenados el Elasticsearch se pudo empezar a monitorizar los datos gracias a Kibana

Sprint 7 - acercamiento machine learning (12/04/2021 - 26/04/2021)

En este **Sprint** se investigó las diferentes bibliotecas y técnicas de machine learning y predicción de datos.

Sprint 8 (1/05/2021 - 10/05/2021)

En este **Sprint** se configuró los sensores de la bodega y se desarrolló una función que permitía descargar los datos de Elasticsearch al servidor.

Además, se realizaron diversas pruebas y tutoriales con algoritmos de machine learning para aprender y decidir cuáles serían los más apropiados en el proyecto.

Sprint 9 (15/05/2021 - 10/06/2021)

En este **Sprint** se creó el modelo y se desarrolló una función capaz de devolver los datos predichos por el modelo a Elasticsearch para poder ser visualizados.

Sprint 10 (10/06/2021 - 24/06/2021)

En este **Sprint** se realizaron una serie de funciones que permitían guardar y cargar el modelo al programa.

También se refactorizó el código para aumentar su legibilidad y mantener un formato y una coherencia.

Sprint 11 (31/08/2021 - 14/09/2021)

En este **Sprint** se implementaron nuevos modelos y se facilitó la creación de otros en el futuro.

Se detecto un error a la hora de descargar los datos de PRTG y transformar los datos y se decidió migrar la funcionalidad de transformación de datos de Flex a Python.

Se fue redactando la memoria para dejarla finalizada para la entrega.

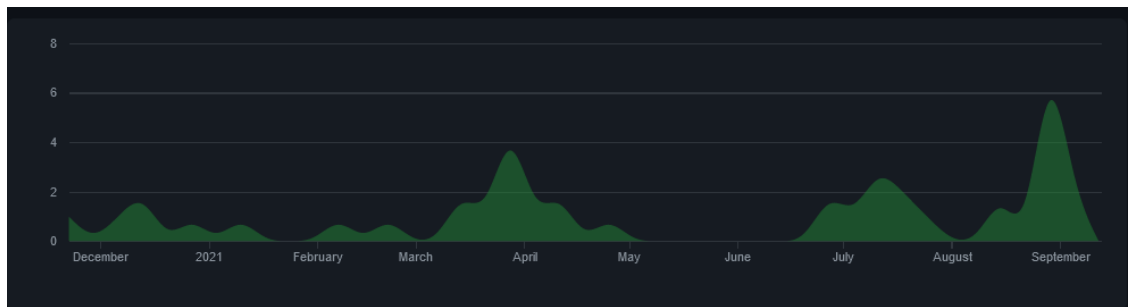


Figura A.1: Gráfico de commits realizados en el tiempo

A.3. Estudio de viabilidad

Antes de adentrarse en la realización de cualquier proyecto se ha de estudiar su viabilidad en el mercado, estimar que costes se van a tener y que beneficios se van a obtener. Durante este apartado se verá la viabilidad tanto económica como legal del proyecto.

Viabilidad económica

A continuación, se hará un estudio sobre los costes y beneficios del proyecto si este se hubiera realizado en un ambiente empresarial real.

Costes

Costes personales

El desarrollo del proyecto ha sido realizado por un desarrollador junior, suponiendo que el salario bruto mensual es de 2000€.

Concepto	Coste
Salario mensual neto	1217.25
Retención IRPF (15 %)	216.75
Seguridad social (28.3 %)	566
Salario mensual bruto	2000
Coste total (6 meses)	12000

Tabla A.1: Costes de personal

Costes hardware

En este apartado se mostrará el coste de los dispositivos hardware empleados para el desarrollo del proyecto. Dado que este se ha realizado durante 8 meses y suponiendo que la amortización es de 4 años.

Concepto	Coste	Amortización
Ordenador	1200€	50€

Tabla A.2: Coste hardware

Costes software

En este apartado se revisarán los costes asociados a licencias de software no gratuitos. Consideraremos que la amortización del software es de 2 años.

Concepto	Coste	Amortización
Windows 10 pro	259€	5.4€

Tabla A.3: Coste Software

Costes totales

Concepto	Coste	Amortización
Ordenador	1200€	50€
Windows 10 pro	259€	5.4€
Total	1459€	55.4€

Tabla A.4: Coste Software

Beneficios

Al tratarse de un proyecto estudiantil es muy complicado estimar un beneficio.

Viabilidad legal

Para poder distribuir un producto software es necesario cumplir ciertas obligaciones legales como son el caso de las licencias.

A continuación, se mostrarán las licencias del software empleado.

Herramienta	Licencia
Elasticsearch	Apache License 2.0
PRTG	PRTG Freeware Edition
virtualBox	Privativa / GPLv2
Ubuntu server	GPL, y otras licencias libres
Windows 10	Microsoft CLUF (EULA) OEM
Git	GNU GPL v2
GitHub	Propietaria/Privativa

Tabla A.5: Licencias

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este anexo se recogen los objetivos del proyecto y los requisitos que definen el comportamiento del programa.

B.2. Objetivos generales

Con este proyecto se persigue la realización de los siguientes objetivos:

- La instalación y configuración de un sistema capaz de recoger, almacenar y gestionar datos de sensores.
- La implementación de un motor de búsqueda para facilitar encontrar los datos deseados con la mayor precisión posible.
- Que la monitorización de los datos se muestre lo más clara y accesible posible para el usuario.
- Realizar un algoritmo de aprendizaje automático que pueda predecir comportamientos y patrones en los datos de los sensores.

B.3. Catálogo de requisitos

A continuación, se procederá a enumerar los distintos requisitos funcionales:

- **RF-1 Activar sistema:** El usuario ha de poder activar el sistema cuando desee.
- **RF-2 Desactivar sistema:** El usuario ha de poder desactivar el sistema cuando desee.
- **RF-3 Captación de sensores:** El sistema ha de almacenar los datos procedentes de los sensores.
- **RF-4 Entrenamiento:** El sistema ha de entrenar un modelo con los datos de los sensores.
- **RF-5 Predicción:** El sistema ha de realizar una predicción sobre los datos de los sensores.
- **RF-6 Gestión del sistema:** El usuario ha de ser capaz de gestionar el sistema.
 - **RF-6.1 Añadir sensor:** El usuario ha de poder añadir sensores.
 - **RF-6.2 Borrar sensor:** El usuario ha de poder eliminar sensores.
 - **RF-6.3 Configurar modelo:** El usuario ha de poder configurar sus propios modelos.
- **RF-7 Monitorización de datos:** El usuario ha de poder ver los datos de los sensores.
 - **RF-7.1 ver datos reales:** Se ha de monitorizar los datos reales procedentes del sensor.
 - **RF-7.2 ver datos entrenamiento:** Se ha de monitorizar los datos de entrenamiento del modelo.
 - **RF-7.3 ver datos predicción:** Se ha de monitorizar los datos de predicción.
- **RF-8 Búsquedas filtradas:** El usuario ha de ser capaz de hacer búsquedas y filtrados de los datos.

- **RF-9 Representación gráfica:** El usuario ha de poder representar de forma gráfica los datos.
 - **RF-9.1 Creación de gráficas:** El usuario ha de poder crear gráficas.
 - **RF-9.2 Borrado de gráficas:** El usuario ha de poder borrar gráficas.
 - **RF-9.3 Visualización de gráficas:** El usuario ha de poder visualizar y consultar las gráficas creadas.

B.4. Especificación de requisitos

En esta sección se mostrarán los casos de uso y su diagrama.

Diagrama de casos de uso

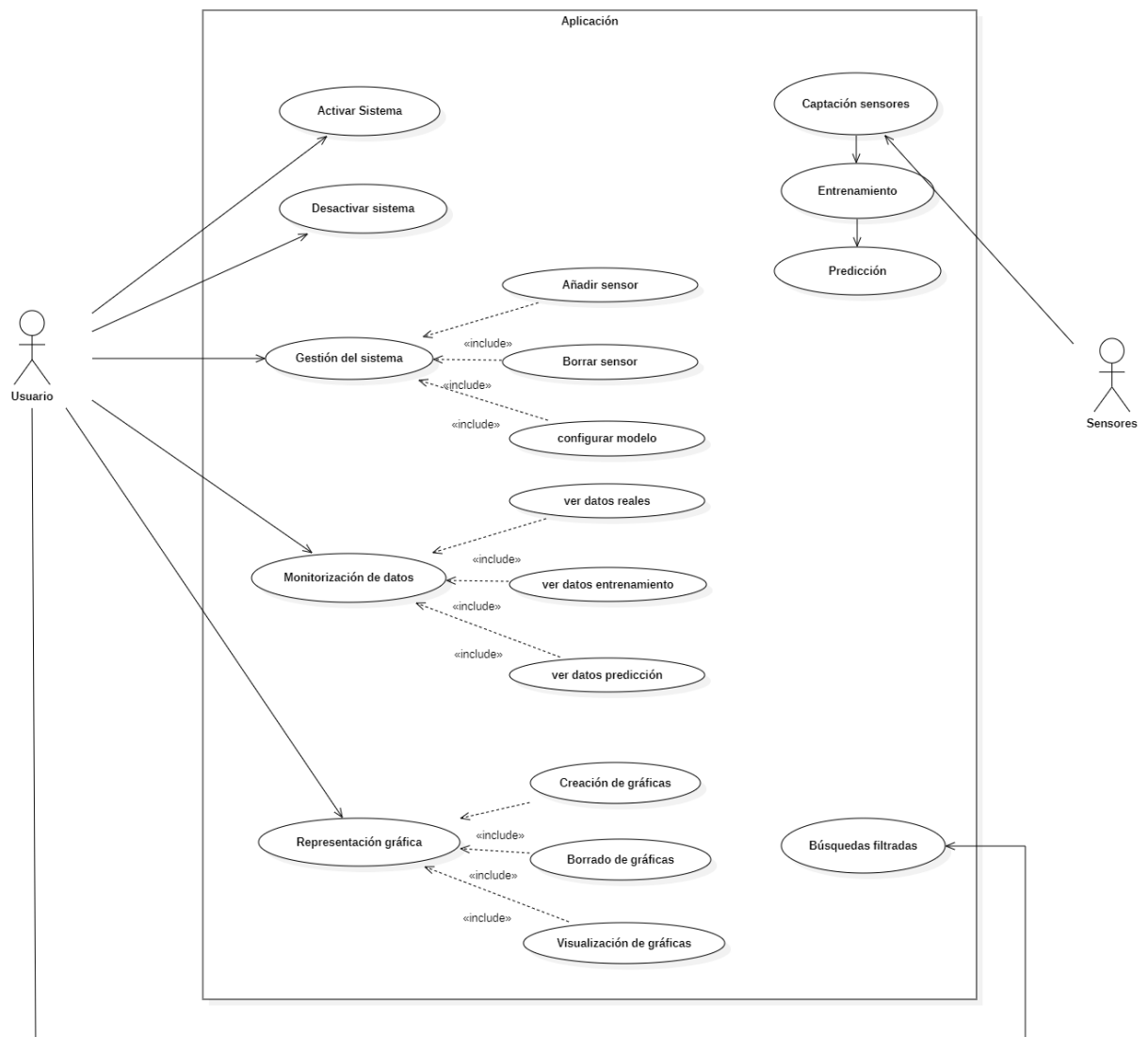


Figura B.1: Diagrama de casos de uso

Actores

En esta aplicación interactúan dos actores, los sensores de los que se captan la información y el usuario que gestiona la aplicación y monitoriza el sistema.

Casos de uso

Caso de uso 1: Activar sistema.		
Versión	1.0	
Autor	Daniel Mellado Hurtado	
Descripción	El usuario ha de poder activar el sistema cuando desee.	
Requisitos	RF-1	
Precondiciones	El programa ha de estar correctamente instalado.	
Secuencia normal	Paso	Acción
	1	El usuario introduce el comando de inicialización del programa en la consola
Postcondiciones	El sistema se encuentra activo y funcionando	
Excepciones	No se encuentra el servicio correspondiente al programa.	
Importancia	Alta	
Urgencia	Alta	

Tabla B.1: Caso de uso 1: Activar sistema.

Caso de uso 2: Desactivar sistema.		
Versión	1.0	
Autor	Daniel Mellado Hurtado	
Descripción	El usuario ha de poder desactivar el sistema cuando desee.	
Requisitos	RF-2	
Precondiciones	El sistema se encuentra activo	
Secuencia normal	Paso	Acción
	1	El usuario introduce el comando de parada del programa en la consola
Postcondiciones	El sistema se encuentra desactivado	
Excepciones		
Importancia	Alta	
Urgencia	Alta	

Tabla B.2: Caso de uso 2: Desactivar sistema.

Caso de uso 3: Almacenamiento de datos.		
Versión	1.0	
Autor	Daniel Mellado Hurtado	
Descripción	El sistema ha de almacenar los datos procedentes de los sensores	
Requisitos	RF-3	
Precondiciones	Los sensores se encuentran activos y PRTG recibe sus datos	
Secuencia normal	Paso	Acción
	1	El sistema descarga los datos procedentes de PRTG
	2	Se transforman los datos descargados
	3	El sistema envía los datos a Elasticsearch
Postcondiciones	Los datos de los sensores son recogidos y almacenados correctamente	
Excepciones	No se encuentra el servidor de PRTG. No hay datos disponibles	
Importancia	Alta	
Urgencia	Alta	

Tabla B.3: Caso de uso 3: Almacenamiento de datos.

Caso de uso 4: Entrenamiento.	
Versión	1.0
Autor	Daniel Mellado Hurtado
Descripción	El sistema ha de entrenar un modelo con los datos de los sensores
Requisitos	RF-4
Precondiciones	Existen datos de los sensores en Elastitcsearch. Exista un modelo para cada sensor que se vaya a entrenar
Secuencia normal	Paso Acción
	1 El sistema descarga los datos que se desea entrenar de Elasticsearch
	2 Se carga el modelo asociado a dicho sensor
	3 Se realiza el entrenamiento.
	4 Se suben los datos del entrenamiento
	5 Se guarda el modelo entrenado.
Postcondiciones	El modelo se encuentra entrenado.
Excepciones	No existen datos en Elasticsearch para dicho sensor. No se encuentra un modelo para dicho sensor
Importancia	Media
Urgencia	Media

Tabla B.4: Caso de uso 4: Entrenamiento.

Caso de uso 5: Predicción.		
Versión	1.0	
Autor	Daniel Mellado Hurtado	
Descripción	El sistema ha de realizar una predicción sobre los datos de los sensores.	
Requisitos	RF-5	
Precondiciones	Exista un modelo para cada sensor que se vaya a predecir	
Secuencia normal	Paso	Acción
	1	El sistema carga el modelo del sensor que se vaya a predecir
	2	Se realiza la predicción
	3	Se suben los datos de la predicción a Elasticsearch .
Postcondiciones	Se ha realizado la predicción de un sensor	
Excepciones	No se encuentra un modelo para dicho sensor	
Importancia	Media	
Urgencia	Media	

Tabla B.5: Caso de uso 5: Predicción.

Caso de uso 6: Gestión del sistema.	
Versión	1.0
Autor	Daniel Mellado Hurtado
Descripción	El usuario ha de ser capaz de gestionar el sistema.
Requisitos	RF-6 RF-6.1, RF-6.2, RF-6.3
Precondiciones	El sistema se encuentra funcionando. El usuario posee permisos de administrador.
Secuencia normal	Paso Acción
	1 El usuario entra al fichero de configuración.
	2 Se muestran las variables que se pueden configurar.
	3 El usuario guarda el fichero.
	4 El usuario reinicia el programa.
Postcondiciones	El sistema se encuentra configurado.
Excepciones	Algún dato introducido es inválido.
Importancia	Alta
Urgencia	Alta

Tabla B.6: Caso de uso 6: Gestión del sistema.

Caso de uso 7: Añadir sensor.	
Versión	1.0
Autor	Daniel Mellado Hurtado
Descripción	El usuario ha de poder añadir sensores.
Requisitos	RF-6.1
Precondiciones	El sistema se encuentra funcionando.
Secuencia normal	Paso Acción
	1 El usuario entra al fichero de configuración.
	2 Se muestra la lista de ids de sensores.
	3 El usuario añade uno o varios sensores.
	3 El usuario guarda el fichero.
	4 El usuario reinicia el programa. .
Postcondiciones	El programa realizara la obtención, entrenamiento y predicción de los nuevos sensores introducidos.
Excepciones	No se encuentran los sensores introducidos
Importancia	Alta
Urgencia	Alta

Tabla B.7: Caso de uso 7: Añadir sensor.

Caso de uso 8: Borrar sensor.		
Versión	1.0	
Autor	Daniel Mellado Hurtado	
Descripción	El usuario ha de poder eliminar sensores.	
Requisitos	RF-6.2	
Precondiciones	El sistema se encuentra funcionando.	
Secuencia normal	Paso	Acción
	1	El usuario entra al fichero de configuración.
	2	Se muestra la lista de ids de sensores.
	3	El usuario elimina uno o varios sensores.
	3	El usuario guarda el fichero.
	4	El usuario reinicia el programa. .
Postcondiciones	El programa ya no realizara la obtención, entrenamiento y predicción de los nuevos sensores eliminados.	
Excepciones		
Importancia	Alta	
Urgencia	Alta	

Tabla B.8: Caso de uso 8: Borrar sensor

Caso de uso 9: Configurar modelo.		
Versión	1.0	
Autor	Daniel Mellado Hurtado	
Descripción	El usuario ha de poder configurar sus propios modelos.	
Requisitos	RF-6.3	
Precondiciones	El sistema se encuentra funcionando.	
Secuencia normal	Paso	Acción
	1	El usuario entra al fichero de Crear modelo.
	2	El usuario especifica el id del sensor con el cual se va a entrenar el modelo.
	3	El usuario elige el tipo de modelo de desea utilizar
	3	El usuario introduce los parámetros de dicho modelo
	4	El usuario ejecuta el fichero de crear modelo
	5	El sistema guarda el nuevo modelo en disco. .
Postcondiciones	El modelo se encuentra guardado y podrá ser utilizado para el entrenamiento y la predicción	
Excepciones	El modelo que se desea implementar no existe.	
Importancia	Alta	
Urgencia	Alta	

Tabla B.9: Caso de uso 9: Configurar modelo.

Caso de uso 10: Monitorización de datos.	
Versión	1.0
Autor	Daniel Mellado Hurtado
Descripción	El usuario ha de poder ver los datos de los sensores.
Requisitos	RF-7 RF-7.1, RF-7.2, RF-7.3
Precondiciones	El sistema se encuentra funcionando.
Secuencia normal	Paso Acción
	1 El usuario accede a la url de Kibana.
	2 El usuario accede al apartado <i>Discover</i> .
	3 El usuario elige un rango de fechas
	4 El usuario introduce filtra los datos por los campos que vea apropiados.
Postcondiciones	Se muestran los datos de los sensores
Excepciones	No existen datos.
Importancia	Alta
Urgencia	Alta

Tabla B.10: Caso de uso 10: Monitorización de datos.

Caso de uso 11: Ver datos reales.		
Versión	1.0	
Autor	Daniel Mellado Hurtado	
Descripción	Se ha de monitorizar los datos reales procedentes del sensor.	
Requisitos	RF-7.1	
Precondiciones	El sistema se encuentra funcionando.	
Secuencia normal	Paso	Acción
	1	El usuario accede a la url de Kibana.
	2	El usuario accede al apartado <i>Discover</i> .
	3	El usuario elige un rango de fechas
	4	El usuario introduce filtra por el campo <i>valor</i>
Postcondiciones	Se muestran los valores reales de los sensores	
Excepciones	No existen datos.	
Importancia	Alta	
Urgencia	Alta	

Tabla B.11: Caso de uso 11: Ver datos reales.

Caso de uso 12: Ver datos entrenamiento.	
Versión	1.0
Autor	Daniel Mellado Hurtado
Descripción	Se ha de monitorizar los datos de entrenamiento procedentes del sensor.
Requisitos	RF-7.2
Precondiciones	El sistema se encuentra funcionando.
Secuencia normal	Paso Acción
	1 El usuario accede a la url de Kibana.
	2 El usuario accede al apartado <i>Discover</i> .
	3 El usuario elige un rango de fechas
	4 El usuario introduce filtra por el campo <i>entrenamiento</i>
Postcondiciones	Se muestran los valores de entrenamiento de los sensores
Excepciones	No existen datos.
Importancia	Alta
Urgencia	Alta

Tabla B.12: Caso de uso 12: Ver datos entrenamiento.

Caso de uso 12: Ver datos predicción.		
Versión	1.0	
Autor	Daniel Mellado Hurtado	
Descripción	Se ha de monitorizar los datos de predicción del sensor.	
Requisitos	RF-7.3	
Precondiciones	El sistema se encuentra funcionando.	
Secuencia normal	Paso	Acción
	1	El usuario accede a la url de Kibana.
	2	El usuario accede al apartado <i>Discover</i> .
	3	El usuario elige un rango de fechas
	4	El usuario introduce filtra por el campo <i>prediccion</i>
Postcondiciones	Se muestran los valores de predicción de los sensores	
Excepciones	Elasticsearch no está activo, Kibana no esta activo.	
Importancia	Alta	
Urgencia	Alta	

Tabla B.13: Caso de uso 12: Ver datos predicción.

Caso de uso 13: Búsquedas filtradas.		
Versión	1.0	
Autor	Daniel Mellado Hurtado	
Descripción	El usuario ha de ser capaz de hacer búsquedas y filtrados de los datos.	
Requisitos	RF-8	
Precondiciones	Elasticsearch y Kibana se encuentran funcionando	
Secuencia normal	Paso	Acción
	1	El usuario accede a la url de Kibana.
	2	El usuario accede al apartado <i>Discover</i> .
	3	El usuario elige un rango de fechas
	4	El usuario introduce las restricciones que se desea en el buscador y en los filtros.
Postcondiciones	Se muestran los valores filtrados	
Excepciones	No existen datos.	
Importancia	Alta	
Urgencia	Alta	

Tabla B.14: Caso de uso 13: Búsquedas filtradas.

Caso de uso 14: Representación gráficas.		
Versión	1.0	
Autor	Daniel Mellado Hurtado	
Descripción	El usuario ha de poder representar de forma gráfica los datos.	
Requisitos	RF-9	
	RF-9.1, RF-9.2, RF-9.3	
Precondiciones	Elasticsearch y Kibana se encuentran funcionando	
Secuencia normal	Paso	Acción
	1	El usuario accede a la url de Kibana.
	2	El usuario accede al apartado <i>Dashboards</i> .
Postcondiciones		
Excepciones	No existen datos.	
Importancia	Alta	
Urgencia	Alta	

Tabla B.15: Caso de uso 14: Representación gráfica.

Caso de uso 15: Creación de gráficas.	
Versión	1.0
Autor	Daniel Mellado Hurtado
Descripción	El usuario ha de poder crear gráficas.
Requisitos	RF-9.1
Precondiciones	Elasticsearch y Kibana se encuentran funcionando
Secuencia normal	Paso Acción
	1 El usuario accede a la url de Kibana.
	2 El usuario accede al apartado <i>Dashboards</i> .
	3 El usuario da al botón <i>Create Dashboard</i>
	4 El usuario da al botón <i>Create Visualization</i>
	5 El usuario elige el tipo de gráfico
	6 El usuario introduce las variables que se van a representar.
	7 El usuari guarda el grafico, dando al botón <i>save and return</i> .
Postcondiciones	Se ha creado un gráfico representando las variables elegidas.
Excepciones	No existen datos.
Importancia	Alta
Urgencia	Alta

Tabla B.16: Caso de uso 15: Creación de gráficas.

Caso de uso 16: Borrado de gráficas.	
Versión	1.0
Autor	Daniel Mellado Hurtado
Descripción	El usuario ha de poder borrar gráficas.
Requisitos	RF-9.2
Precondiciones	Elasticsearch y Kibana se encuentran funcionando
Secuencia normal	Paso Acción
	1 El usuario accede a la url de Kibana.
	2 El usuario accede al apartado <i>Dashboards</i> .
	3 El usuario accede al la ruleta de opciones del gráfico que se desea eliminar
	4 El usuario selecciona la opción <i>more</i>
	5 El usuario selecciona la opción <i>Delete from Dashboards</i>
.	
Postcondiciones	La gráfica se borra
Excepciones	
Importancia	Alta
Urgencia	Alta

Tabla B.17: Caso de uso 16: Borrado de gráficas.

Caso de uso 16: Visualización de gráficas.		
Versión	1.0	
Autor	Daniel Mellado Hurtado	
Descripción	El usuario ha de poder visualizar y consultar las gráficas creadas	
Requisitos	RF-9.3	
Precondiciones	Elasticsearch y Kibana se encuentran funcionando	
Secuencia normal	Paso	Acción
	1	El usuario accede a la url de Kibana.
	2	El usuario accede al apartado <i>Dashboards</i> .
	2	El usuario elige el rango de fechas para ver los datos representados en las gráficas
Postcondiciones	Se muestran todas las gráficas creadas	
Excepciones		
Importancia	Alta	
Urgencia	Alta	

Tabla B.18: Caso de uso 16: Visualización de gráficas.

Apéndice C

Especificación de diseño

C.1. Introducción

En esta sección se describen algunas características relacionadas con el diseño del proyecto.

C.2. Diseño de datos

Sensores

Para este proyecto era necesario la obtención dinámica de datos de sensores en tiempo real, para ello el cliente facilitó una serie de sensores que disponía en una bodega y almacenaba los datos en PRTG, los sensores en cuestión miden el consumo de la bomba de agua [C.1](#), la humedad en el ambiente [C.2](#), la humedad en la parcela [C.3](#) y la presión de la tubería de agua [C.4](#).

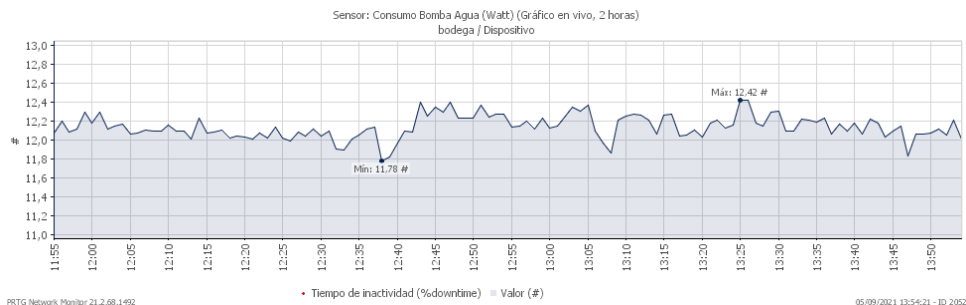


Figura C.1: Sensor del consumo de la bomba de agua

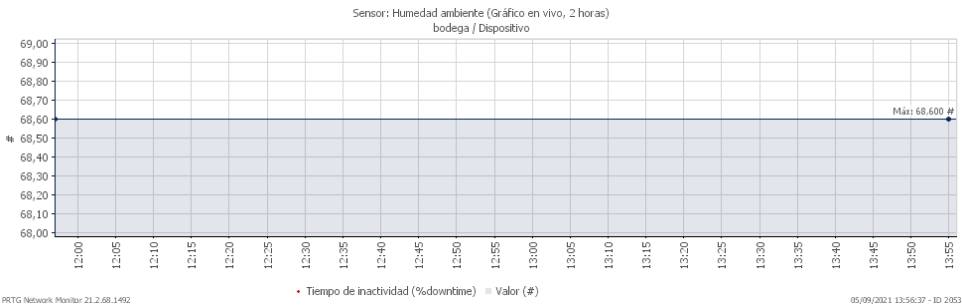


Figura C.2: Sensor de la humedad en el ambiente

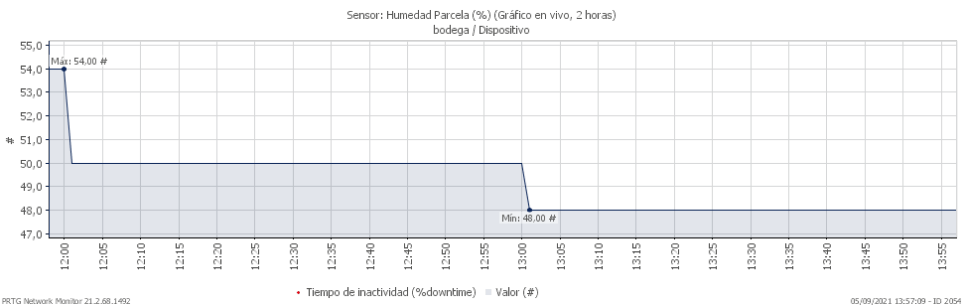


Figura C.3: Sensor de la humedad en la parcela

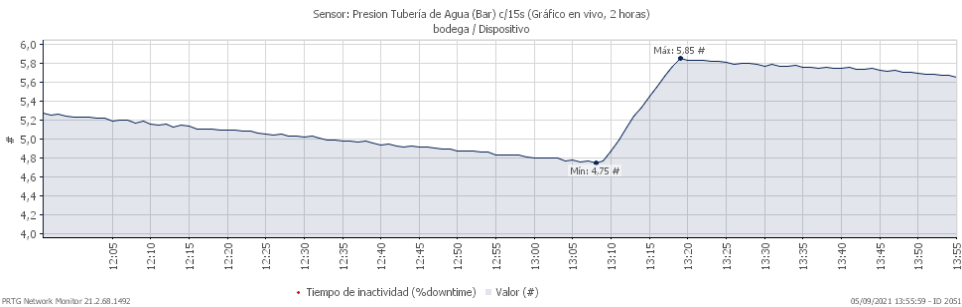


Figura C.4: Sensor de la presión de la tubería de agua


```
1  {
2    "sensorId":"2051",
3    "datetime":"27/07/2021 16:13:06",
4    "reading":
5    {
6      "Valor":4.6700,
7      "Tiempo de ejecución":799.0000
8    }
9  }
```

Listing 1: ejemplo línea de un fichero JSON

Base de datos

Los datos de los sensores, una vez extraídos de PRTG, se almacenan en Elasticsearch, una base de datos NoSQL orientada a documentos. A diferencia de las bases de datos relacionales, esta no está estructurada en filas y columnas, Elasticsearch almacena cada registro junto a sus datos asociados en un único documento JSON [1](#) el cual es asociado a un index. Para ver más información sobre la indexación y la estructura de los ficheros consultar el manual de usuario en el anexo E.

Diagrama de clases

A continuación, se mostrarán los diagramas de clase de la aplicación para facilitar el entendimiento de la estructura del sistema. En la figura C.5 se muestra las clases del paquete */Prediccion* en la cual se encuentra el modelo de aprendizaje y en la figura C.6 se representan las clases del paquete */Utilidades* y */Gestion_datos*.

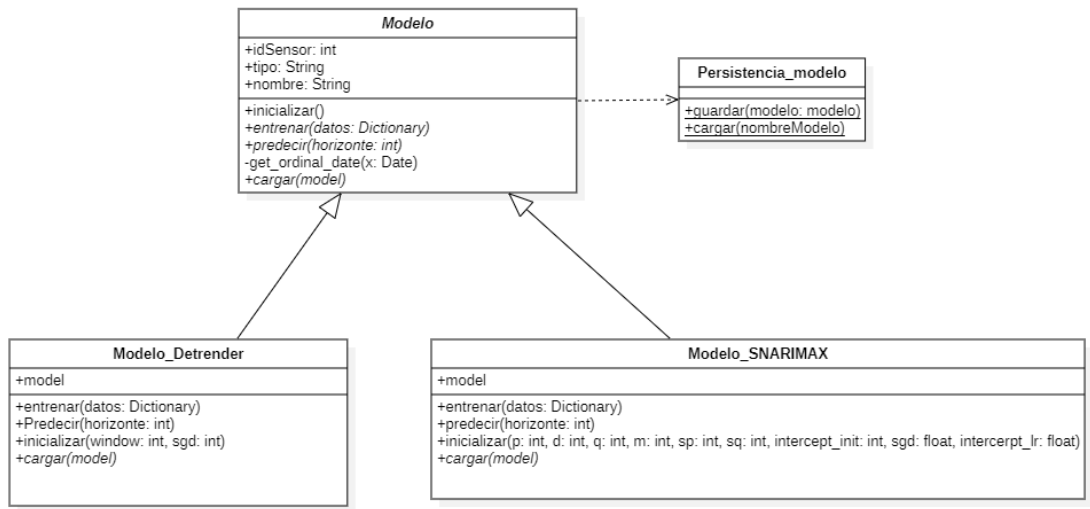


Figura C.5: Diagrama de clases paquete Predicción

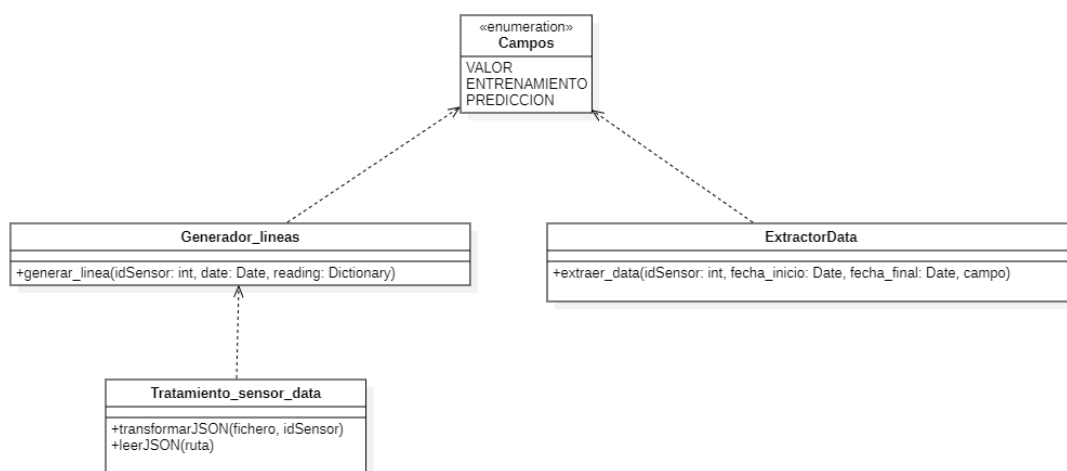


Figura C.6: Diagrama de clases paquete Utilidades

C.3. Diseño procedimental

En este apartado se mostrará el funcionamiento interno del proyecto.

Diagrama de secuencias

Para facilitar el seguimiento de el diagrama de secuencias se ha dividido en tres, el primero muestra el proceso por el cual se obtienen los datos de PRTG [C.7](#), después se realiza el entrenamiento de los datos [C.8](#) y por último se hace la predicción [C.9](#).

Los elementos coloreados hacen referencia a sistemas externos, cómo son PRTG y Elasticsearch.

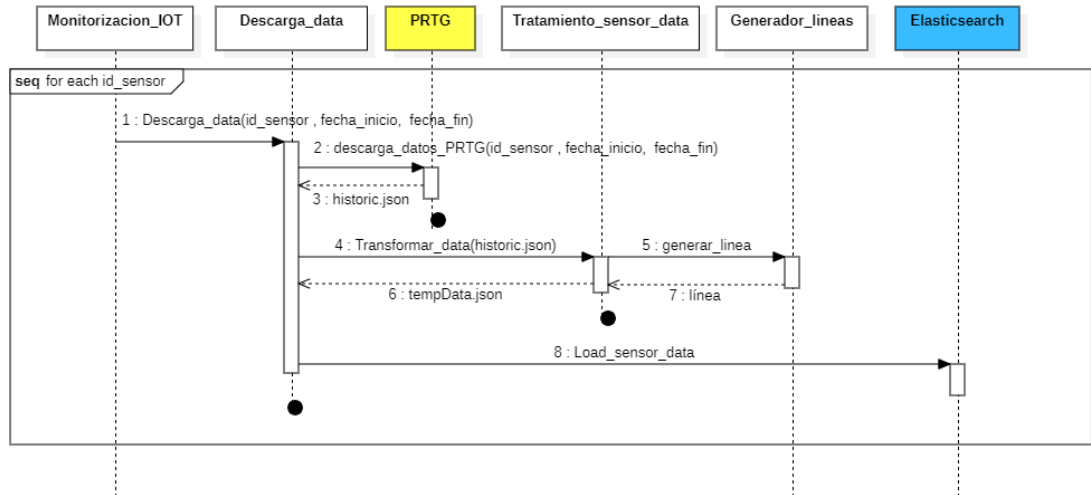


Figura C.7: Diagrama de secuencias descarga de datos

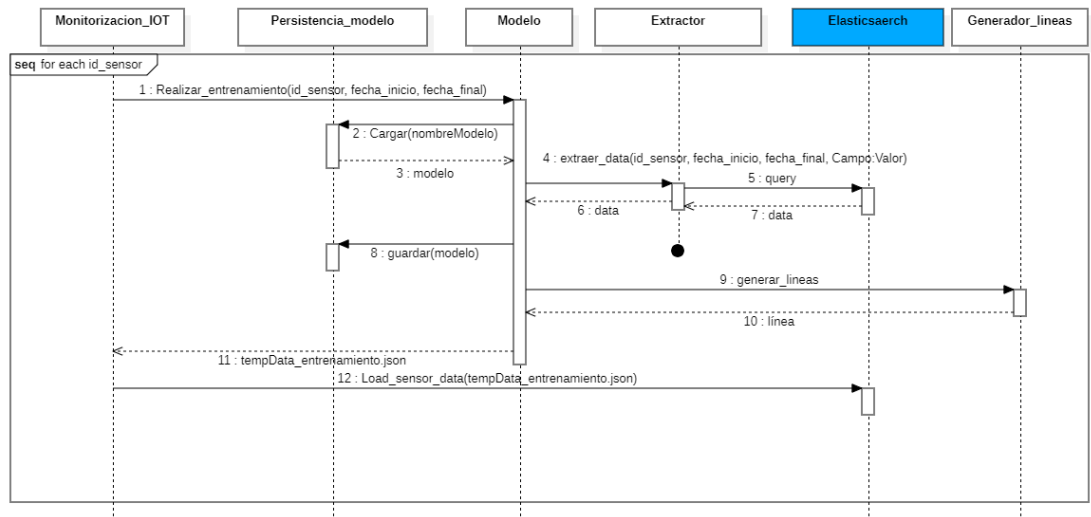


Figura C.8: Diagrama de secuencias entrenamiento

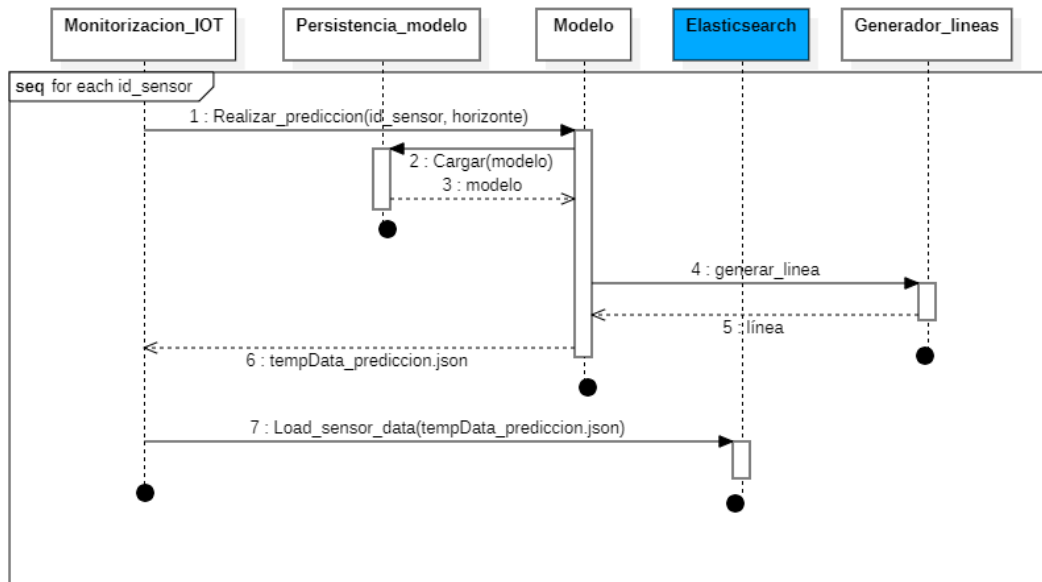


Figura C.9: Diagrama de secuencias predicción

Funcionamiento de Monitorizacion_IOT

El núcleo del programa es el script *Monitorizacion_IOT.sh* el cual es llamado por un servicio que se repite cada X segundos. A continuación, se mostrará el pseudocódigo del script:

```

1: while True do
2:   fecha_inicio  $\leftarrow$  Fecha_actual - 30minutos
3:   fecha_fin  $\leftarrow$  Fecha_actual
4:   for id_sensor in lista_sensores do
5:     Descargar_data(id_sensor, fecha_inicio, fecha_fin)
6:     Realizar_entrenamiento(id_sensor, fecha_inicio, fecha_fin)
7:     Load_sensor_datos ("tempData_entrenamiento + id_sensor +
      .json")
8:     Borrar_datos_predicciones(id_sensor, horizonte_prediccion)
9:     Realizar_predicciones(id_sensor, fecha_fin)
10:    Load_sensor_datos("tempData_prediccion + id_sensor +
      .json")
11:   end for
12: end while

```

Pseudocódigo del script *Descargar_data.sh*, el cual es referenciado en el primer script:

```

1: if fichero tempData + id_sensor + .json exist then
2:   descarga_datos_PRTG
3:   transformar_datos
4:   comparar_ficheros
5:   Load_sensor_datos
6: else
7:   descarga_datos_PRTG
8:   transformar_datos
9:   Load_sensor_datos
10: end if

```

Recolección de los datos de sensores PRTG

Para recuperar datos históricos, PRTG permite descargarlos en diversos formatos cómo pueden ser XML, CSV o JSON, en el caso de esta aplicación, debido a la facilidad que tiene Elasticsearch con formatos JSON se decidió obtener los datos en este formato. [1]

Los parámetros que se introducen a la llamada al comando Curl son:

- **id**: el id del sensor del que se quiere descargar los históricos.
- **avg(average)**: se pone esta variable con valor a 0 para descargar los datos en bruto.
- **usecaption**: se pone esta variable con valor a 1 para obtener más información del sensor y no solo la tabla de datos.
- **sdate**: fecha de inicio del rango
- **edate**: fecha de fin del rango
- **username**: nombre de usuario de PRTG
- **passhash**: passhash del usuario de PRTG

Los datos obtenidos son volcados a un fichero JSON, el cual responde al nombre `historic+id_sensor+.json`

```
# Función que descarga datos de PRTG

# $1 : idSensor
# $2 : fecha inicio
# $3 : fecha fin
# $4 : volcado datos
function descarga_datos_PRTG(){

    curl -o "$4" $IP_PRTG'/api/historicdata.json?id='$1'
        &avg=0
        &usecaption=1
        &sdate='$2'
        &edate='$3'
        &username='$usuario_PRTG'
        &passhash='$passhash_PRTG'
}
```

función del fichero /usr/bin/Monitorizacion_IOT/Gestion_datos/Descargar_data.sh

Subida de datos a Elasticsearch

Para poder realizar la subida de datos a Elasticsearch de forma continua hay que configurar logstash, para ello se usa un *plugin* llamado Http input plugin.

Con este *plugin* una aplicación puede enviar una petición http y Logstash lo procesará y enviará a Elasticsearch.^[3]

```
input{
  http{
    id=> "sensor_data_http_input"
  }
}

filter{

  ruby {
    code => '
      event.get("reading").each { |k, v|
        event.set(k,v)
      }
      event.remove("reading")
    '
  }
}

output{

  elasticsearch{
    hosts => ["localhost:9200", "IP_Ubuntu_server:9200"]
    index => "sensor_data-%{+YYYY.MM.dd}"
  }
}
```

fichero /etc/logstash/conf.d/logstashSensor.conf

Para enviar las líneas JSON con los datos a logstash se hace una petición http por línea.


```
#!/bin/bash

IFS=$'\n'
for LINE in $(cat $1)
do
    echo $LINE

    curl -XPOST -u sensor_data:sensor_data --header
    "Content-Type: application/json"
    "http://localhost:8080/" -d '$LINE'
done
```

fichero /usr/bin/Monitorizacion_IOT/Gestion_datos/Load_sensor_data.sh

C.4. Diseño arquitectónico

El proyecto consta de tres paquetes.

- **Gestion_data:** Este paquete se encarga de todo lo relacionado con la gestión de los datos de los sensores:
 - Descargar los datos de PRTG.
 - Subir datos a Elasticsearch.
 - Descargar datos de Elasticsearch.
 - Eliminar datos de Elasticsearch.
- **Utilidades:** Este paquete se encarga de realizar tareas variadas como:
 - Transformar los ficheros JSON.
- **Prediccion:** Este paquete se encarga de todo lo relacionado con el entrenamiento y la predicción de los sensores:
 - Entrenar el modelo.
 - Realizar la predicción.
 - Crear modelos.

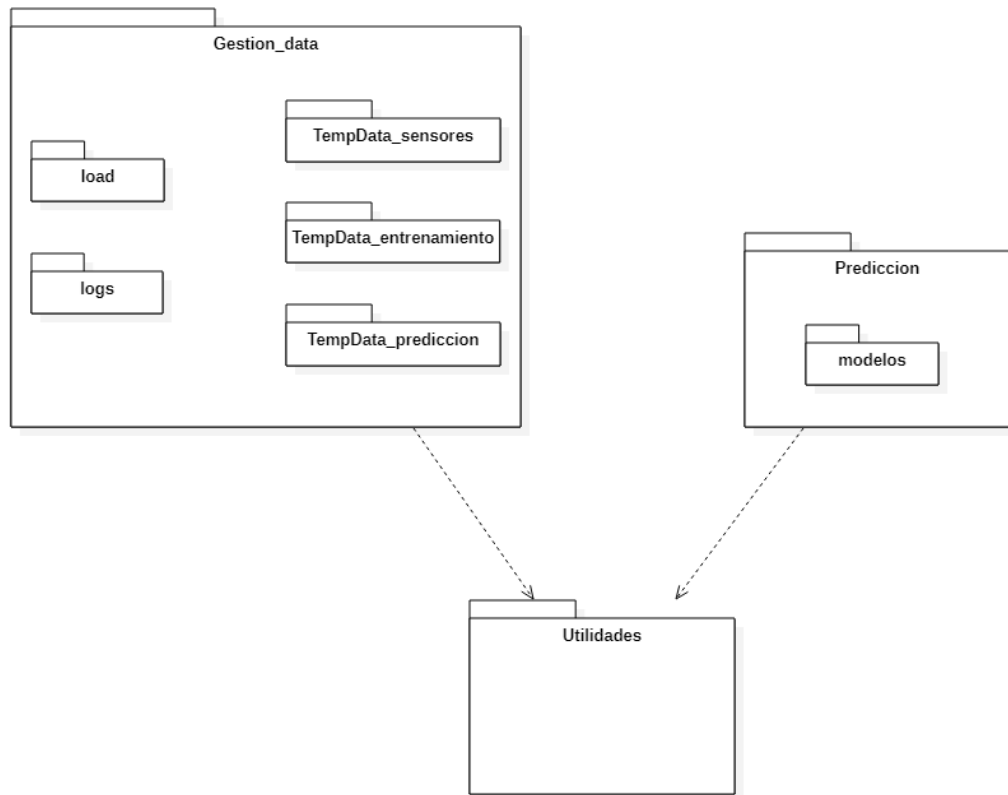


Figura C.10: Diagrama de paquetes

Para facilitar el entendimiento de la topología tanto del hardware cómo del software se presenta el siguiente diagrama de despliegue C.11. En él se pueden observar los dos sistemas: ubuntu server, albergando Elasticsearch y Monitorizacion_IOT y Windows 10 cuya única función es albergar PRTG.

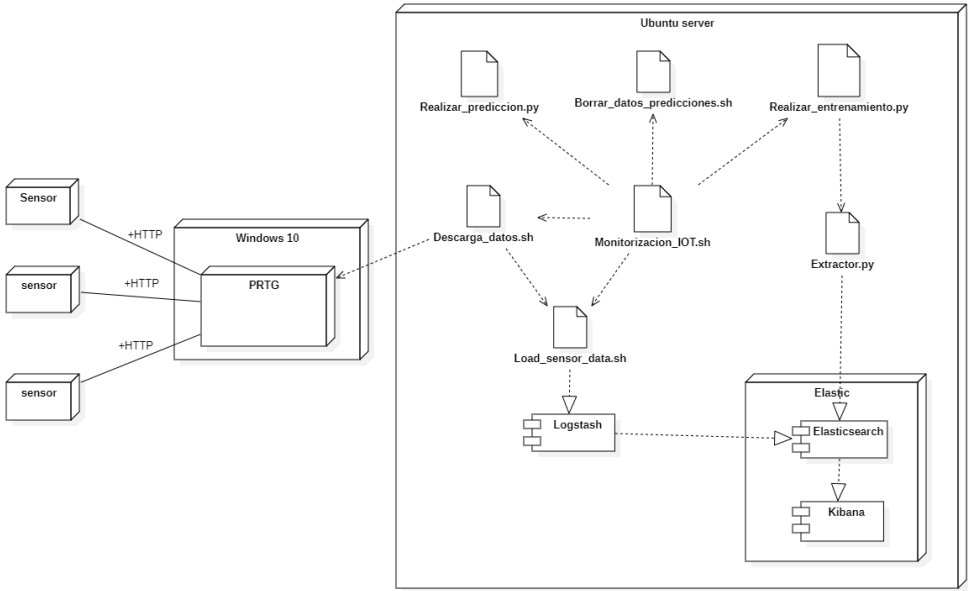


Figura C.11: Diagrama de despliegue

Apéndice *D*

Documentación técnica de programación

D.1. Introducción

En el siguiente anexo se describe la documentación técnica del programa, se detalla cómo está estructurado, cómo realizar la correcta instalación y configuración del mismo, así como la batería de pruebas realizadas.

D.2. Estructura de directorios

Monitorizacion_IOT se instala en la carpeta */usr/bin/* dentro un servidor Ubuntu. El programa consta de un script principal que mediante un daemon se ejecuta constantemente en segundo plano. Por cada ciclo de ejecución (por defecto es de un minuto) se descargan los datos en bruto desde PRTG y se almacenan en la carpeta */usr/bin/Monitorizacion_IOT/Gestion_datos/logs/*. De ahí son procesados y guardados temporalmente en la carpeta

/usr/bin/Monitorizacion_IOT/Gestion_datos/TempData_sensores/

Tras realizar las comparaciones necesarias y asegurarse que no se van a subir datos repetidos a Elasticsearch se almacenan los datos finales en la carpeta */usr/bin/Monitorizacion_IOT/Gestion_datos/load* desde la cual se suben a la base de datos.

Al realizar el entrenamiento o la predicción de los sensores se cargan los modelos correspondientes a dichos sensores, estos se encuentran almacenados en la carpeta */usr/bin/Monitorizacion_IOT/Prediccion/modelos/*.

Cuando se realiza el entrenamiento de un modelo este devuelve los datos generados y los almacena en

`/usr/bin/Monitorizacion_IOT/Gestion_datos/TempData_entrenamiento/`.

A la hora de predecir también se devuelven los datos generados, los cuales se almacenan en

`/usr/bin/Monitorizacion_IOT/Gestion_datos/TempData_prediccion/`

por último, en la carpeta `/usr/bin/Monitorizacion_IOT/Utilidades/` se encuentran diversas funciones de utilidad.

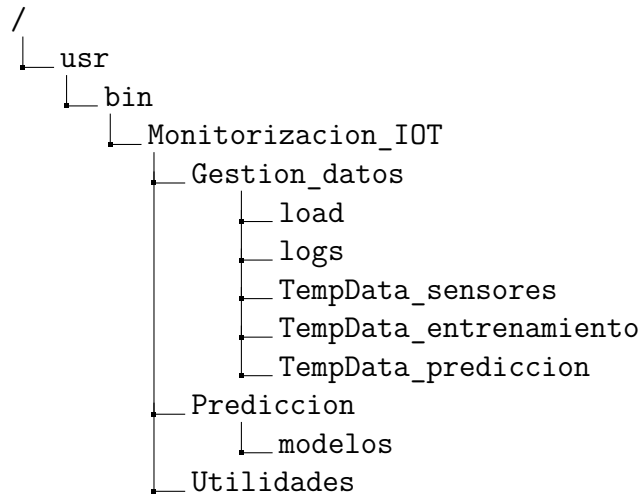


Figura D.1: Directorios del proyecto

D.3. Manual del programador

El manual del programador tiene como objetivo ayudar a todas aquellas personas que vayan a trabajar con este proyecto.

Entorno de desarrollo

Para la realización de este proyecto se necesita tener instalados los siguientes programas y dependencias:

- Python 3: versión 3.8.10
- Elasticsearch: versión 7.4
- Kibana: versión 7.14.1
- Logstash: versión 7.14.1
- Openjdk-11-jre-headless: versión 11.0.11

bibliotecas python:

- River versión 0.71
- Pandas: versión 1.3.2
- Elasticsearch: versión 7.14.0
- Numpy: versión 1.21.2

Archivos

A continuación, se mostrarán todos los ficheros que conforman el programa así como una breve descripción:

- **Monitorizacion_IOT.sh**: Este es el script principal de la aplicación el cual está sujeto a un daemon, de tal forma que se repite en bucle cada minuto descargando los datos de PRTG, subiéndolos a Elasticsearch, realizando el entrenamiento y la predicción de cada sensor.
- **/Gestion_datos/Descargar_data.sh**: Es el script que se encarga de descargar los datos de PRTG, transformar dichos datos y subirlos a Elasticsearch.

- **/Gestion_datos/Borrar_datos_predicciones.sh**: Debido a que para cada sensor se repite la predicción cada minuto y se suben los resultados a Elasticsearch es necesario borrar la predicción anterior para que siempre se pueda visualizar la predicción actualizada, este script hace exactamente eso.
- **/Gestion_datos/Load_sensor_data.sh**: Este script se encarga de enviar línea por línea de un fichero a Logstash para poder ser enviado a Elasticsearch.
- **/Gestion_datos/Extractor.py**: En este fichero se encuentra la clase encargada de la descarga de datos de Elasticsearch.
- **/Predicciones/Crear_modelo.py**: En este fichero el usuario puede configurar y crear un modelo para un sensor.
- **/Predicciones/Modelo.py**: En este fichero se encuentran las clases que corresponden a todos los modelos.
- **/Predicciones/Persistencia_modelo.py**: En este fichero se encuentra la clase que permite guardar en disco y cargar cada modelo.
- **/Predicciones/Realizar_entrenamiento.py**: Es el fichero encargado de la realización del entrenamiento del modelo de un sensor, se carga el modelo correspondiente al sensor, se descargan los últimos datos subidos a Elasticsearch y se realiza el entrenamiento del modelo y finalmente se guarda el modelo en disco.
- **/Predicciones/Realizar_prediccion.py**: Es el fichero encargado de realizar la predicción de un sensor, se carga el modelo correspondiente al sensor que se desea predecir y se realiza la predicción a partir de la fecha actual y tantos instantes de tiempo como este especificado en el fichero principal, Monitorizacion_IOT.sh.
- **/Utilidades/campos.py**: En este fichero se encuentra una clase de enumeración donde se encuentran los tres tipos de campos que se suben a Elasticsearch.
 - **VALOR**: Corresponde al valor real recogido por el sensor.
 - **PREDICCION**: Corresponde al valor de la predicción
 - **ENTRENAMIENTO**: Corresponde al valor del entrenamiento.

- **/Utilidades/Generador_lineas.py**: En este fichero se encuentra la clase encargada de generar las líneas de texto en un formato específico para poder ser procesadas por Logstash.
- **/Utilidades/Tratamiento_sensor_data.py**: En este fichero se encuentra la clase encargada de tratar con los datos de los sensores, permite leer ficheros JSON y transformarlos.
- **/Utilidades/Transformar_data.py**: Es el fichero encargado de transformar los ficheros JSON con los datos procedentes de PRTG en un fichero apto para ser procesado por Logstash.

Modelos

La biblioteca de Python, River, facilita dos tipos de modelos para series temporales permitiendo realizar el entrenamiento de manera incremental con un data stream procedente de los sensores así cómo realizar una predicción de su comportamiento.

Pese a que solo estén implementados en la aplicación dos tipos de modelos está preparada para que se pueda implementar nuevos tipos de modelos de diversas bibliotecas.

Implementar nuevos modelos

Monitorizacion_IOT es flexible a la implementación de nuevos modelos de incremental learning, para crear un nuevo modelo se ha de acceder a la carpeta donde se ha instalado el programa y editar el fichero */Predicciones/Modelo.py*, en este fichero se puede ver la clase abstracta *Modelo* la cual implementa cuatro métodos abstractos:

- **inicializar(self)**: El método encargado de inicializar el modelo, implementando el algoritmo deseado.
- **cargar(self, modelo)**: El método encargado de cargar el modelo.
- **entrenar(self, datos)**: El método encargado del entrenamiento del modelo a partir de una serie de datos
- **predecir(self, horizonte)**: El método encargado de realizar la predicción a futuro de un sensor a partir de un modelo ya entrenado.

Para crear un nuevo modelo se a de programar una nueva clase que herede de *Modelo* e implemente estas cuatro funciones.

D.4. Compilación, instalación y ejecución del proyecto

En este apartado se mostrará cómo instalar Monitorizacion_IOT . Para facilitar la utilización del programa se adjunta una máquina virtual

Configuración Ubuntu Server

Para albergar el programa se utilizará una máquina virtual Ubuntu Server.

Antes de comenzar la instalación del programa se tiene que asegurar que estén instaladas las herramientas de red de Linux, las cuales ayudarán a conocer la IP del servidor, y con ella poder acceder al servidor de forma remota.

```
apt install net-tools
```

También es necesario comprobar la zona horaria del servidor y si es necesario cambiarla, se puede utilizar el siguiente comando:

```
dpkg-reconfigure tzdata
```

Instalación Monitorizacion_IOT

Una vez instalado y configurado Ubuntu Server en una máquina virtual se puede proceder a la instalación de Monitorizacion_IOT . Para ello es necesario tener descargado del repositorio <https://github.com/dmh1001/TFG-Monitorizacion-IOT> tanto el script *install.sh* como el paquete Debian *MonitorizacionIOT_1.0_all.deb*. Importante, a la hora de instalar el programa ambos ficheros se han de encontrar en un mismo directorio.

Una vez se tenga estos ficheros en la máquina virtual se podrá comenzar la instalación, para mayor comodidad, asegurarse que se accede con el usuario *root*, para ello se puede utilizar el siguiente comando:

```
sudo su
```

Para instalar el programa y sus dependencias sólo es necesario ejecutar el script *install.sh* que instalará una serie de paquetes, como son Elasticsearch, Logstash, Kibana, bibliotecas de Python y el propio programa.

```
bash install.sh
```

a continuación, se mostrará el código que realiza el instalador:

```
wget -qO - https://artifacts.elastic.co
/GPG-KEY-elasticsearch | sudo apt-key add -
echo "deb https://artifacts.elastic.co/packages/7.x
/apt stable main"
| sudo tee -a /etc/apt/sources.list.d
/elastic-7.x.list
apt update
sudo apt-get install apt-transport-https
sudo apt install elasticsearch
sudo apt install kibana
sudo apt install logstash
sudo apt install openjdk-11-jre-headless
apt update && apt upgrade
sudo apt install python3-pip
pip3 install river
pip3 install pandas
pip3 install elasticsearch
pip3 install numpy
sudo dpkg -i
--force-overwrite MonitorizacionIOT_1.0_all.deb
```

Este proceso se puede demorar unos minutos y es posible que pida la confirmación, por parte del usuario de la instalación de ciertos paquetes.

El instalador se encarga de instalar el paquete Debian que contiene los archivos del programa, el cual se instalará en la ruta `/usr/bin/Monitorizacion-IOT/`.

Configuraciones

El script también se encarga de configurar Elasticsearch, Kibana y Logstash así como iniciar los servicios. Tras instalarse el paquete se realizan una serie de configuraciones sobre los servicios de Elasticsearch y Kibana, éstas se hacen automáticamente si necesidad de la supervisión del usuario.

Elasticsearch

Se configura el archivo que se encuentra en la ruta:

```
/etc/elasticsearch/elasticsearch.yml
```

Kibana

El archivo de configuración se encuentra en la ruta:

```
/etc/kibana/kibana.yml
```

Se crea una carpeta y se le añaden permisos para guardar los logs que Kibana genere.

```
mkdir /var/log/kibana
chown -R Kibana:kibana /var/log/kibana/
```

Inicio de procesos

Tras instalar todos los paquetes y ficheros necesarios se inicializan los procesos:

Elasticsearch

```
systemctl daemon-reload &&
systemctl enable elasticsearch.service
systemctl start elasticsearch
```

Kibana

```
systemctl daemon-reload && systemctl enable Kibana
systemctl start kibana
```

Logstash

```
systemctl daemon-reload && systemctl enable Logstash
systemctl start logstash
```

Este es el último paso que realiza el instalador, a partir de aquí el resto de la configuración corre a cargo del usuario.

Comprobación de procesos

Es importante comprobar que los procesos funciones con normalidad y que estos estén activos

```
systemctl status Monitorizacion_IOT

systemctl status elasticsearch
systemctl status kibana
systemctl status logstash
```

Si resulta que alguno de los componentes no se encuentra activos podemos reiniciar el servicio.

```
systemctl restart Monitorizacion_IOT

systemctl restart elasticsearch
systemctl restart Kibana
systemctl restart Logstash
```

D.5. Pruebas del sistema

Pruebas de regresión

A la hora realizar predicciones es extremadamente importante saber cómo de fiable es el modelo que se está utilizando por lo que realizar pruebas es completamente necesario.

Datasets

Debido a que no se posee históricos de los datos de los sensores y ya que PRTG sólo recopila datos cuando está conectado en un equipo en funcionamiento, los *datasets* empleados para realizar las pruebas no son lo suficientemente extensos cómo para entrenar el modelo y que saque predicciones fidedignas.

Por ello, se realizarán las pruebas con datos recogidos en intervalos de varias de horas y se comprobará cómo se comporta el algoritmo a corto plazo.

Para estas pruebas se usarán dos de los sensores, el que mide la presión de la tubería de agua y el que mide el consumo de la bomba de agua.

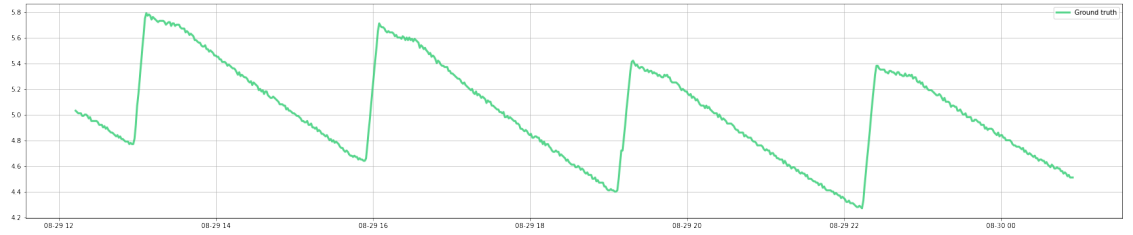


Figura D.2: Dataset sensor de Presión Tubería de Agua

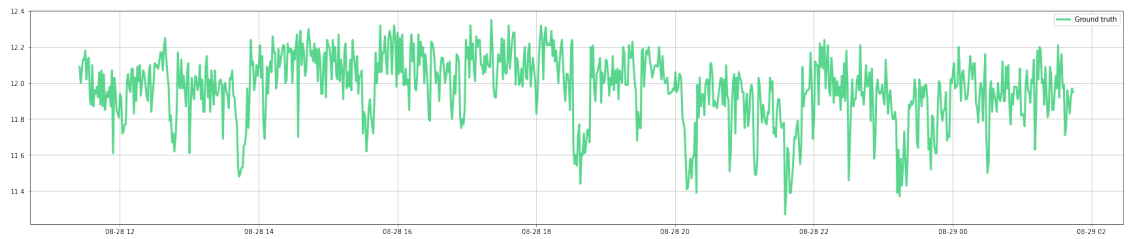


Figura D.3: Dataset sensor Consumo de bomba de agua

Métricas de regresión

Las métricas de regresión dan un promedio de cuanto de acertadas están las predicciones obtenidas sobre el valor real. Para ello se utilizarán las siguientes métricas:

MSE

Mean Squared Error o MSE es una de las métricas más comunes en la evaluación de regresión. Mide el error cuadrado promedio de las predicciones.

Se calcula como el promedio de las diferencias al cuadrado entre los valores reales y los predichos.

$$MSE = \frac{1}{N} \cdot \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

Al elevarse al cuadrado tiende a inflar errores grandes, por lo que cuanto mayor sea la diferencia entre la predicción y los valores reales mayor será el error.

RMSE

Root Mean Squared Error o RMSE es una extensión de MSE. Esta métrica calcula la raíz cuadrada de MSE, haciendo que las unidades de las métricas sean las mismas que las unidades de los valores originales.

$$RMSE = \sqrt{\frac{1}{N} \cdot \sum_{i=1}^N (y_i - \hat{y}_i)^2}$$

Al elevar el MSE al cuadrado se deja de penalizar los errores grandes,

MAE

Mean Absolute Error o MAE al igual que RMSE las unidades de error coinciden con las unidades de los valores reales y predichos.

MAE no da más o menos peso a los diferentes tipos de error y las puntuaciones aumentan linealmente con el aumento del error.

Se calcula como el promedio del valor absoluto de la diferencia entre los valores reales y predichos.

$$MAE = \frac{1}{N} \cdot \sum_{i=1}^N |y_i - \hat{y}_i|$$

R2

R2 es una métrica que calcula la bondad del modelo.

El mejor valor posible es 1, indicando una predicción perfecta y puede tomar valores negativos si la predicción es muy mala.

$$R^2 = 1 - \frac{\sum (Y_i - \hat{y}_i)^2}{\sum (Y_i - \bar{y})^2}$$

Para los siguientes test se utilizará el modelo SNARIMAX. Se realizan las pruebas con datos recopilados de siete días distintos,

Pruebas Sensor Consumo Bomba Agua

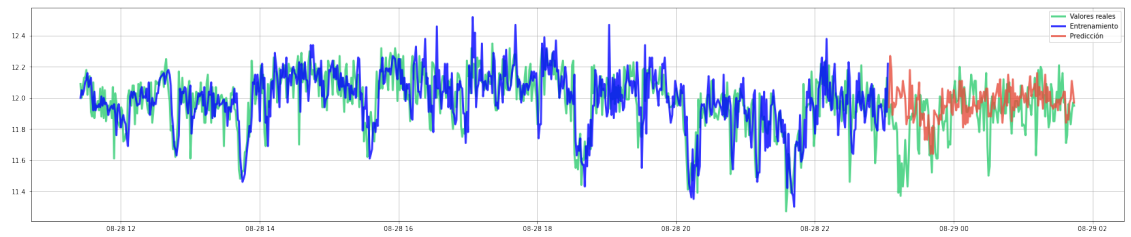


Figura D.4: Consumo Bomba Agua en las fechas: 28/08/2021 - 29/08/2021

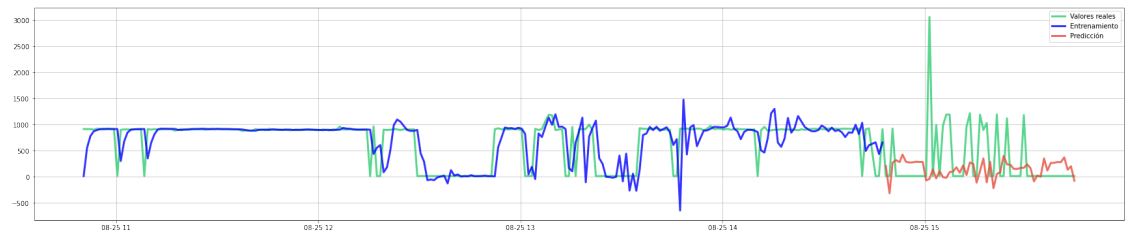


Figura D.5: Consumo Bomba Agua en las fechas: 25/08/2021 - 26/08/2021

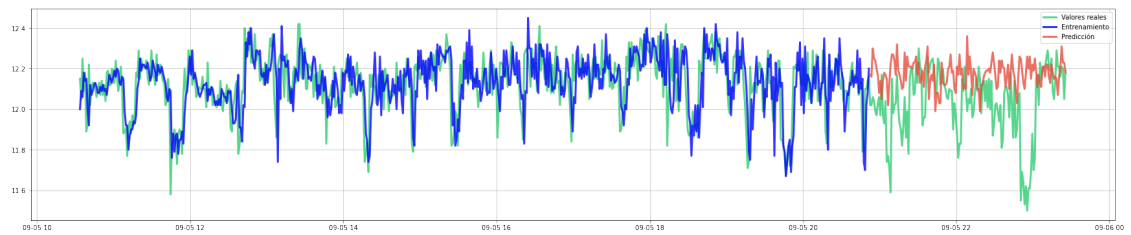


Figura D.6: Consumo Bomba Agua en las fechas: 05/09/2021 - 06/09/2021

Pruebas Sensor Presión tubería de agua

Rango de fechas	MSE	RMSE	MAE	R2
05/09/2021 - 06/09/2021	0.056	0.237	0.176	-0.891
28/08/2021 - 29/08/2021	0.04	0.2	0.153	-0.417
27/08/2021 - 28/08/2021	0.068	0.261	0.186	-0.474
25/08/2021 - 26/08/2021	431303.37	656.737	428.61	-0.272
31/07/2021 - 01/08/2021	0.035	0.188	0.142	-1.125
24/07/2021 - 24/07/2021	0.019	0.137	0.111	-0.439
Media	61614.798	93.965	61.436	-0.516

Tabla D.1: métricas Consumo Bomba Agua

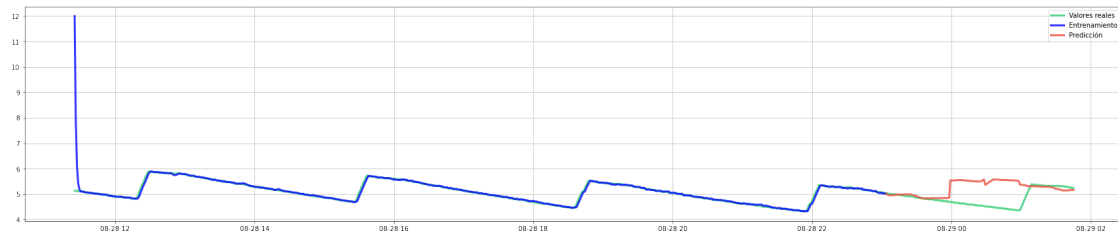


Figura D.7: Presión tubería de agua en las fechas: 28/08/2021 - 29/08/2021

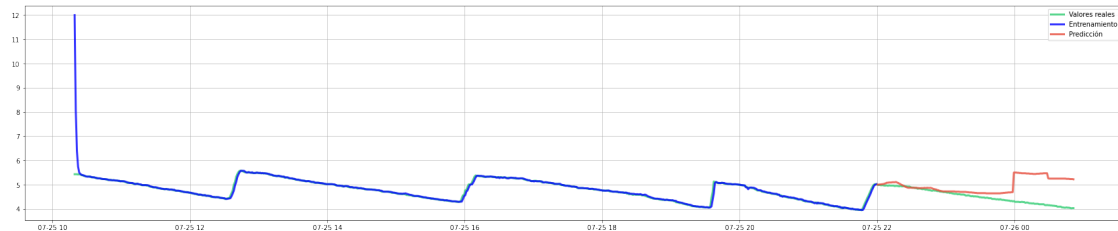


Figura D.8: Presión tubería de agua en las fechas: 25/07/2021 - 26/07/2021

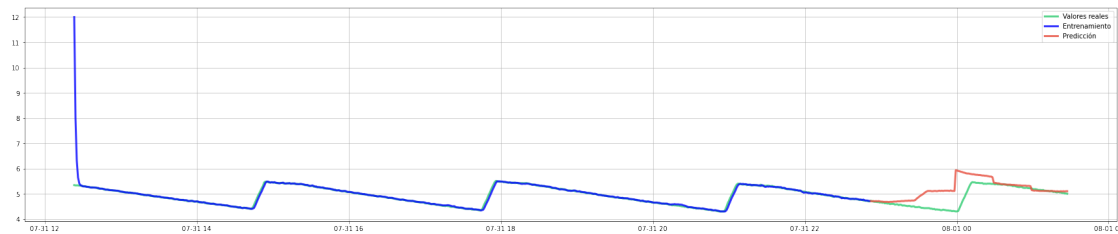


Figura D.9: Presión tubería de agua en las fechas: 31/07/2021 - 01/08/2021

Rango de fechas	MSE	RMSE	MAE	R2
08/09/2021 - 09/09/2021	0.125	0.354	0.322	-0.217
29/08/2021 - 30/08/2021	0.193	0.44	0.358	-1.658
28/08/2021 - 29/08/2021	0.404	0.635	0.441	-2.936
25/08/2021 - 26/08/2021	0.159	0.399	0.358	-0.187
31/07/2021 - 01/08/2021	0.218	0.467	0.298	-0.381
25/07/2021 - 26/07/2021	0.461	0.679	0.453	-4.156
Media	0.222	0.424	0.318	-1.362

Tabla D.2: métricas Presión tubería de agua

Pruebas con conjuntos de datos más grandes

Para probar cómo funcionaría el modelo con un conjunto de datos más amplio, de cinco meses. Se ha descargado de la página web kaggle un *dataset* que recoge datos procedentes de varios sensores de una bomba de agua.

<https://www.kaggle.com/nphantawee/pump-sensor-data>

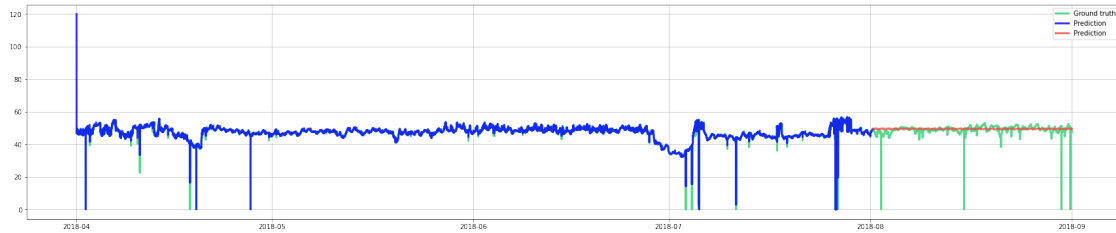


Figura D.10: dataset 1

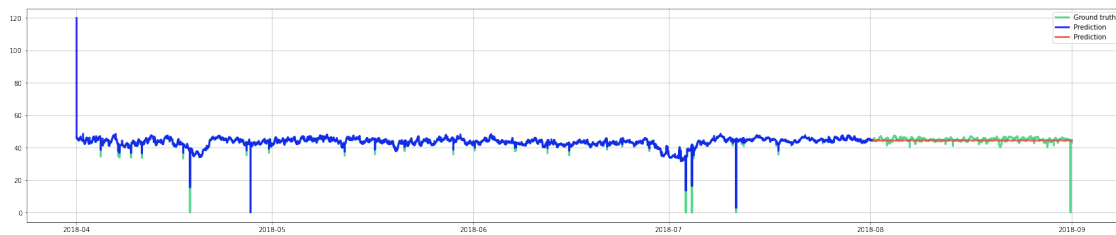


Figura D.11: dataset 3

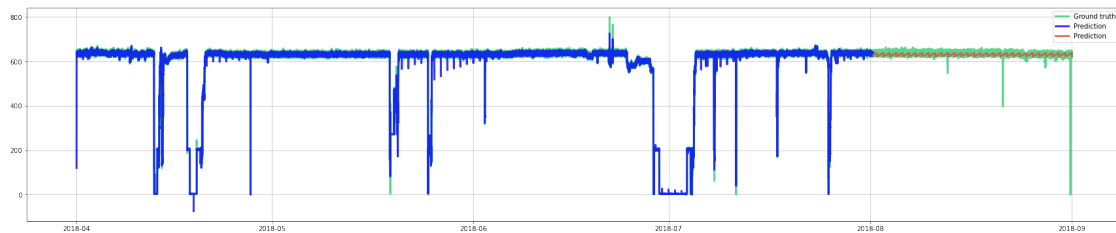


Figura D.12: dataset 4

Cómo se puede observar, las pruebas no han sido satisfactorias, al entrenar un *dataset* con un conjunto de datos más amplio los resultados obtenidos no muestran que se haya aplicado un aprendizaje en el momento en el que el modelo deja de entrenar para únicamente predecir.

	MSE	RMSE	MAE	R2
dataset1	2.883	1.698	1.213	-0.055
dataset2	1.719	1.311	1.005	-0.115
dataset3	1.613	1.270	0.982	-0.287
dataset4	130.277	11.414	7.697	-0.428
Media	34.123	3.923	2.724	-0.221

Tabla D.3: métricas dataset

Apéndice *E*

Documentación de usuario

E.1. Introducción

En este apartado se mostrarán los requisitos necesarios para instalar la aplicación así como un detallado manual en el que se explica cómo funciona la aplicación.

E.2. Requisitos de usuarios

El usuario, para poder ejecutar la aplicación ha de poseer un equipo con sistema operativo Windows 10 y una máquina virtual con un sistema operativo Ubuntu-server.

E.3. Instalación

Monitorizacion_IOT

Para instalar Monitorizacion_IOT es necesario tener descargados y almacenados en un mismo directorio del servidor ubuntu los ficheros *install.sh* y el paquete *Monitorizacion_IOT_1.0_all.deb*

Para instalarlo tan solo es necesario ejecutar el siguiente comando:

```
sudo bash install.sh
```

PRTG

Para la instalación de PRTG en un sistema operativo Windows 10 se ha de descargar el programa de la página principal <https://www.paessler.com/es/prtg>. La instalación es sencilla y tras acceder al programa se pedirá introducir el usuario y la contraseña, por defecto es *prtgadmin* para ambas, tanto la contraseña cómo el usuario se podrá cambiar.

PRTG por defecto viene ya con una serie de sensores configurados E.2, los cuales miden parámetros dentro del sistema.



Figura E.1: inicio PRTG

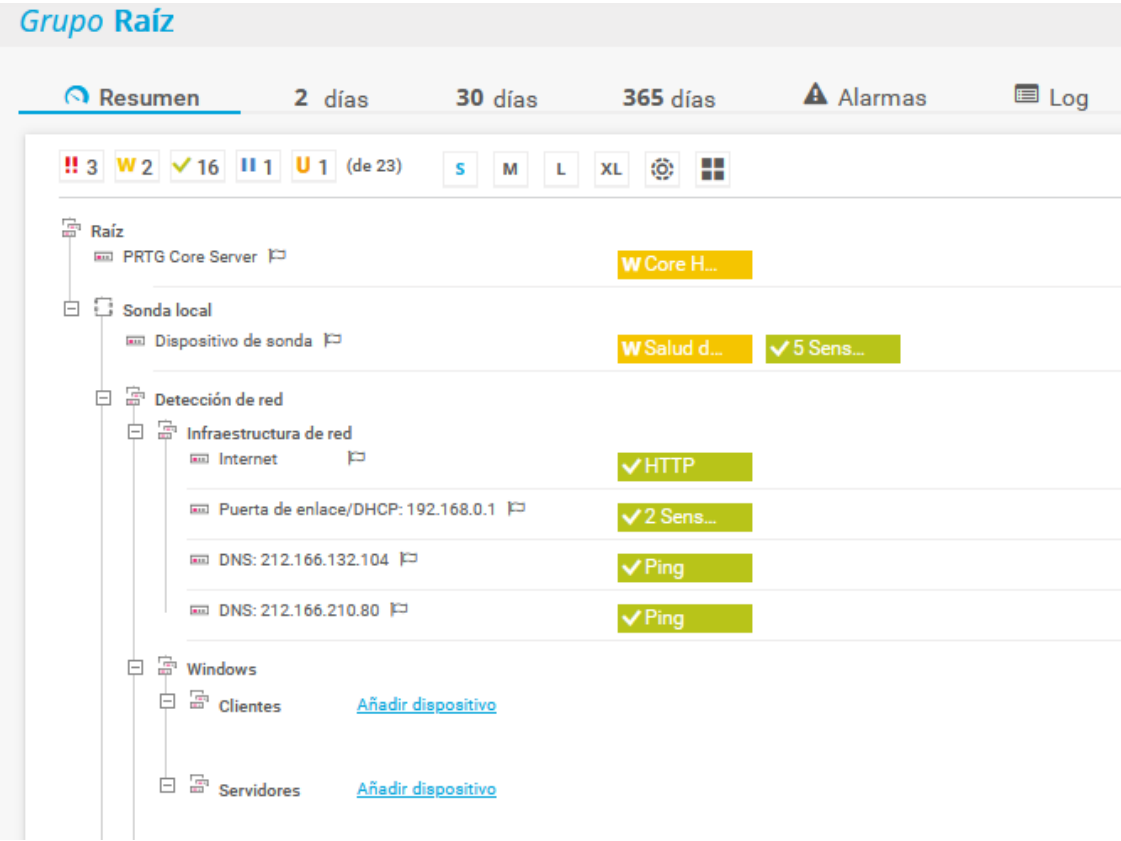


Figura E.2: dispositivos PRTG

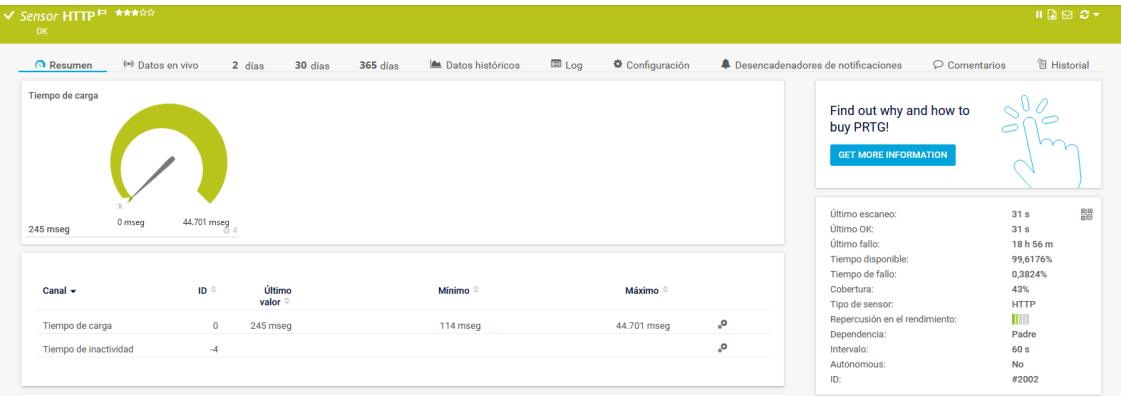


Figura E.3: ejemplo sensor PRTG

E.4. Manual del usuario

Este apartado pretende hacer de guía al usuario a través de la aplicación, mostrando cómo funciona el sistema y cómo es posible configurarlo.

Configuración

Una vez se haya instalado Monitorizacion_IOT hay una serie de modificaciones que hay que realizar manualmente antes de poder ser ejecutado.

Logstash

El primero sería en el fichero del logstash, el cual se encuentra en la ruta: */etc/logstash/conf.d/logstashSensor.conf*

En él se encuentra el mecanismo que permite a logstash procesar los archivos de logs con los datos de los sensores y enviárselos a la BBDD de Elasticsearch.

```
input{
  http{
    id=> "sensor_data_http_input"
  }
}

filter{
  ruby {
    code => '
      event.get("reading").each { |k, v|
        event.set(k,v)
      }
      event.remove("reading")
    '
  }
}

output{
  elasticsearch{
    hosts => ["localhost:9200", "IP_Ubuntu_server:9200"]
  }
}
```



```
        index => "sensor_data-%{+YYYY.MM.dd}"
    }
}
```

Para que logstash mande la información a Elasticsearch hay que sustituir donde pone “IP_Ubuntu_server” por la IP de Ubuntu Server, en el cual se ha instalado Elasticsearch.

Índice

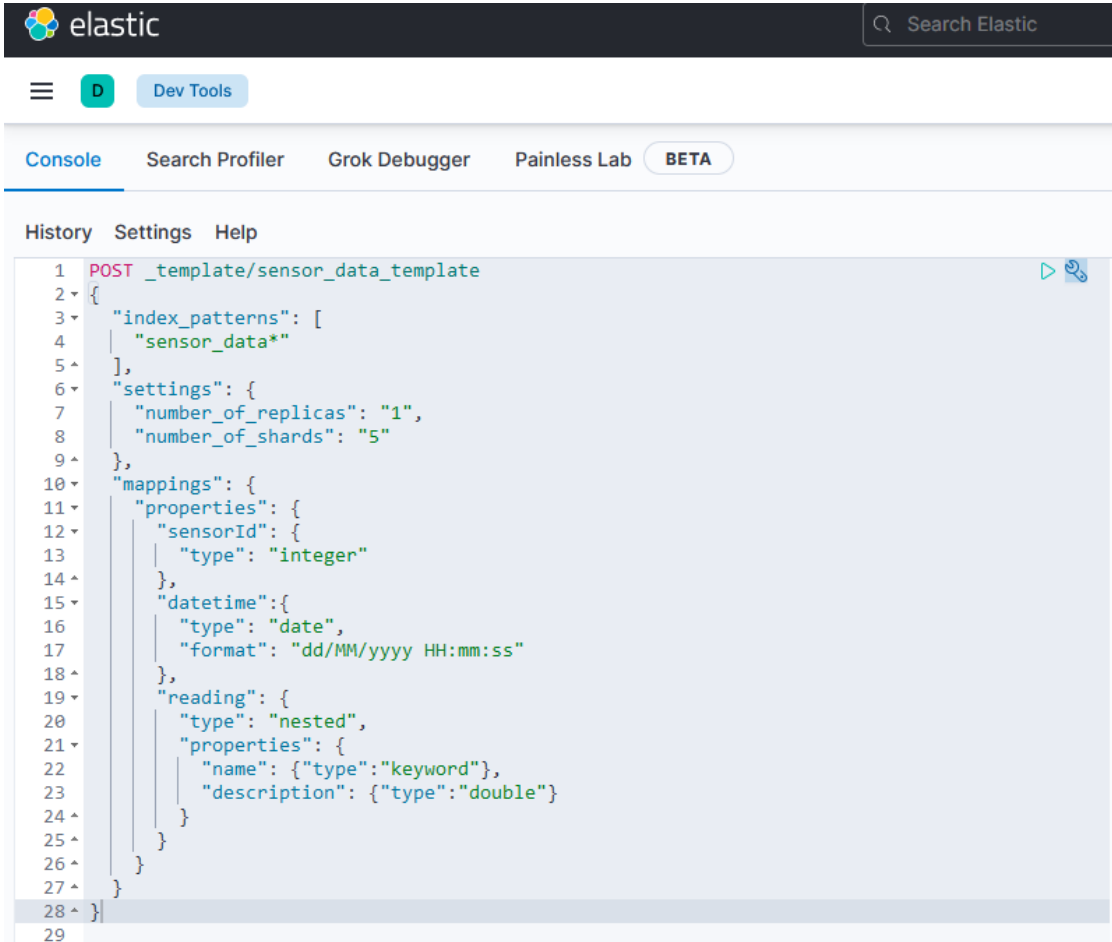
Logstash ahora ya sabe a dónde enviar los datos, pero se ha de preparar Elasticsearch para su llegada, se ha de crear un índice que diga a Elasticsearch que datos esperar.

Para ello se ha de acceder a la página principal de Elastic, introduciendo en un navegador web del host la url: *IP_Ubuntu_server:5601*, donde *IP_Ubuntu_server* es la ip del servidor.

Una vez se haya accedido a Elastic, para crear un índice se ha de acceder a *Dev Tools*, accediendo desde el menú desplegable izquierdo, en el apartado de *Management* se ha de introducir el POST con el índice **2** y enviar la *request* cómo se puede observar en la figura **E.4**

```
1  POST _template/sensor_data_template
2  {
3    "index_patterns": [
4      "sensor_data*"
5    ],
6    "settings": {
7      "number_of_replicas": "1",
8      "number_of_shards": "5"
9    },
10   "mappings": {
11     "properties": {
12       "sensorId": {
13         "type": "integer"
14       },
15       "datetime": {
16         "type": "date",
17         "format": "dd/MM/yyyy HH:mm:ss"
18       },
19       "reading": {
20         "type": "nested",
21         "properties": {
22           "name": {"type": "keyword"},
23           "description": {"type": "double"}
24         }
25       }
26     }
27   }
28 }
```

Listing 2: Índice



The screenshot shows the Elastic Dev Tools interface. At the top, there's a search bar labeled "Search Elastic". Below it, a navigation bar includes "Console", "Search Profiler", "Grok Debugger", "Painless Lab", and a "BETA" badge. The "Console" tab is active, displaying a "History" section with a single entry. The entry is a POST request to the endpoint `_template/sensor_data_template`. The request body is a JSON object defining an index template. The JSON structure includes `index_patterns` (an array with `"sensor_data*"`), `settings` (with `number_of_replicas` set to "1" and `number_of_shards` set to "5"), and `mappings` (with `properties` containing `sensorId` (integer), `datetime` (date with format `dd/MM/yyyy HH:mm:ss`), and `reading` (nested object with `name` as a keyword and `description` as a double)).

```
1 POST _template/sensor_data_template
2 {
3   "index_patterns": [
4     "sensor_data*"
5   ],
6   "settings": {
7     "number_of_replicas": "1",
8     "number_of_shards": "5"
9   },
10  "mappings": {
11    "properties": {
12      "sensorId": {
13        "type": "integer"
14      },
15      "datetime": {
16        "type": "date",
17        "format": "dd/MM/yyyy HH:mm:ss"
18      },
19      "reading": {
20        "type": "nested",
21        "properties": {
22          "name": {"type": "keyword"},
23          "description": {"type": "double"}
24        }
25      }
26    }
27  }
28 }
29
```

Figura E.4: Crear un índice

Una vez creado el índice todos los datos procedentes de Logstash se almacenarán con ese índice. Dentro del menú *Stack Management* en el apartado *index Managment* se puede consultar los datos que va recibiendo E.5.

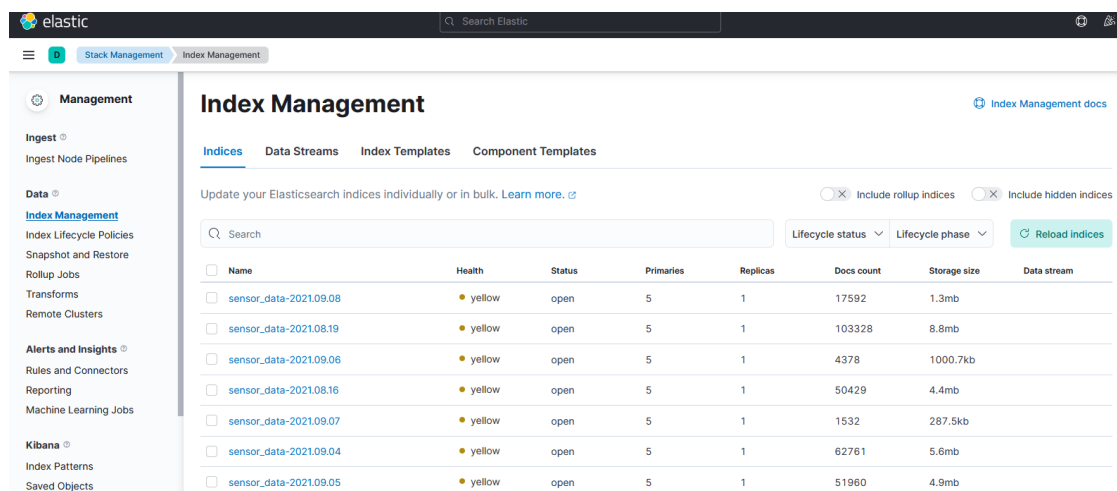


Figura E.5: Index Managment

configuración de parámetros

El usuario puede elegir que sensores desea almacenar y entrenar así cómo cada cuanto tiempo se desea ejecutar el programa y seleccionar cuantos instantes de tiempo en el futuro se desea predecir. Para ello se ha de acceder al fichero `/usr/bin/Monitorizacion_IOT/Monitorizacion_IOT.sh` en él, se enciñtarán una serie de variables que el usuario ha de modificar.

- **usuario_PRTG**: se ha de introducir el nombre de usuario de PRTG
- **passhash_PRTG**: se ha de introducir el passhash de PRTG
- **lista_id_sensor**: en esta lista se ha de introducir el id de los sensores, separados por un espacio.
- **horizonte_predicción**: se ha de introducir cuantos minutos a futuro se desea que llegue la predicción.
- **repeticion_ciclo**: Se ha de introducir el tiempo en segundos que ha de esperar el sistema entre ciclo y ciclo.

Para saber la dirección IP de PRTG hay que acceder, dentro de PRTG a configuración >interfaz de usuario >combinaciones de puertos / servidor web direcciones IP activas y para saber el passhash se ha de acceder a configuración >mi cuenta >Configuración de cuenta de usuario y mostrar passhash.

Crear un modelo para los sensores

Para la realización del entrenamiento sobre los datos procedentes de un sensor, es esencial crear y configurar cada modelo a mano, respondiendo a las necesidades de estos.

Para crear un modelo hay que modificar de forma manual el fichero `/usr/bin/Monitorizacion-IOT/Prediccion/Crear_modelo.py`, una vez en el fichero, se ha de generar una instancia del modelo que se dese crear pasando cómo parámetro el id del sensor. A continuación, llamaremos a la función inicializar del modelo, pasándole los parámetros necesarios. Por último se llama al método guardar de la clase `Persistencia_modelo`. Para guardar el modelo simplemente compilamos el fichero, este se guardará en la carpeta `/usr/bin/Monitorizacion-IOT/Prediccion/modelos/` y se podrá usar para el entrenamiento y la predicción.

```
if __name__ == "__main__":

    idSensor = 4051

    modelo = Modelo_SNARIMAX(idSensor)
    modelo.inicializar(q=2,
                      m=30,
                      sp=6,
                      sq=10,
                      intercept_init=12,
                      sgd=0.01,
                      intercerpt_lr=0.3)

    Persistencia_modelo.guardar(modelo)
```

Inicialización y parada del sistema

Una vez configurado el programa se puede inicializar y poner en marcha todo el sistema, para ello se ha de introducir el siguiente comando:

```
systemctl start Monitorizacion_IOT
```

para realizar la parada completa del sistema y que este deje de captar datos de PRTG se ha de introducir el siguiente comando:

```
systemctl stop Monitorizacion_IOT
```

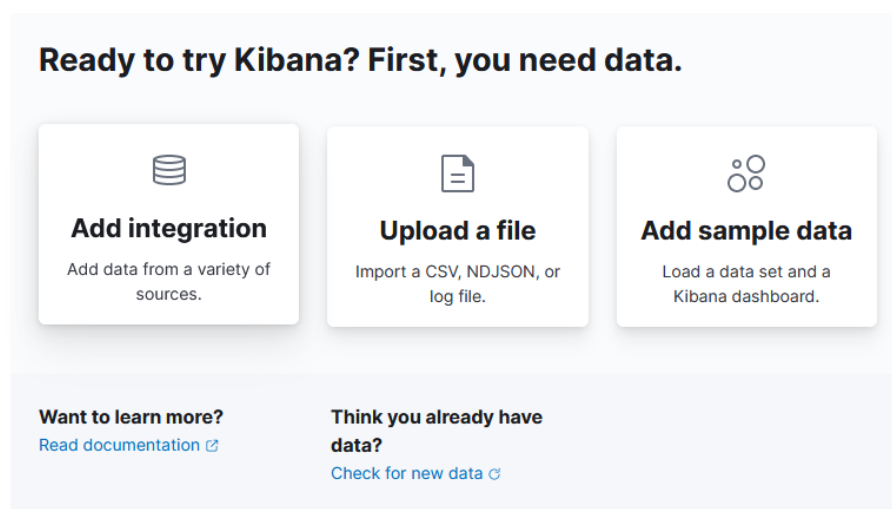
Index pattern

Por último, para poder explorar y visualizar los datos en Kibana es necesarios que exista un *index pattern* que diga a Elasticsearch que índices contienen los datos, así como especificar de que tipo son. [2]

Para poder crear un *index pattern*, primero Elasticsearch ha de tener datos E.6, es por ello que este es el último paso a realizar.

Una vez se disponga de datos se podrá crear un *index pattern* E.7

En este caso se ha de crear un patrón para que albergue todos los datos procedentes de sensores “sensor_data-*” E.8



Some indices may be hidden. Try to [create an index pattern](#) anyway.

Figura E.6: Index pattern menú inicial si no se encuentran datos en Elasticsearch

Una vez creado el patrón se pedirá que se configure, se puede especificar cuál va a ser el índice de los datos, si el tiempo en el que es indexado a en Elasticsearch “@timestamp” u otro campo fecha que se prefiera. E.9

También se puede cambiar el tipo de cualquier otro campo.

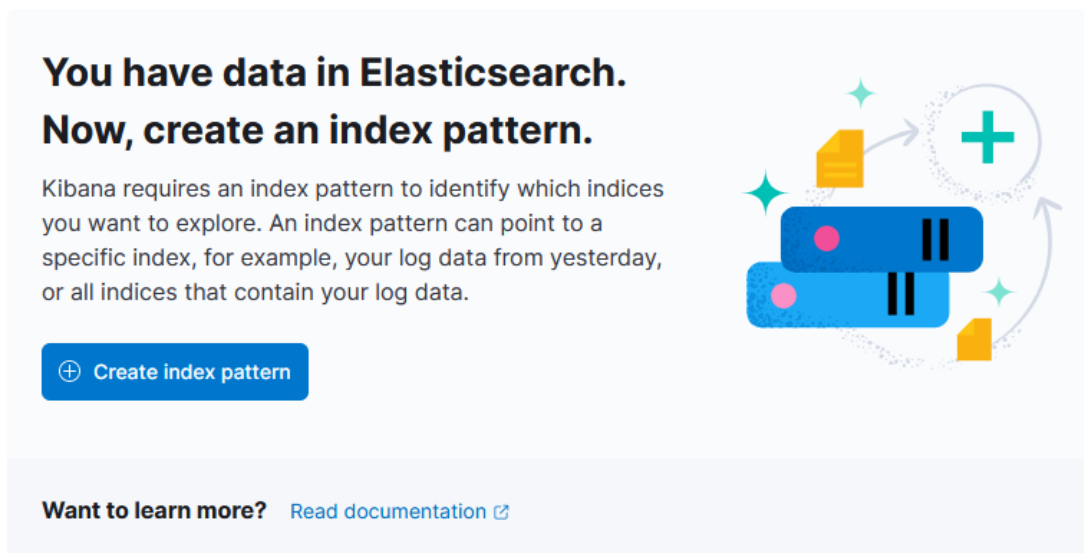


Figura E.7: Index pattern menú inicial

Create index pattern

An index pattern can match a single source, for example, `filebeat-4-3-22`, or **multiple** data sources, `filebeat-*`.
[Read documentation](#)

Step 1 of 2: Define an index pattern

Index pattern name

Use an asterisk (*) to match multiple indices. Spaces and the characters \, /, ?, *, <, >, | are not allowed.

☒ Include system and hidden indices

Next step >

Figura E.8: Creación de un index pattern

Fields (40)Scripted fields (0)Field filters (0)

Q Search

All field types

Add field

Name ↑	Type	Format	Searchable	Aggregatable	Excluded	
@timestamp	date		•	•		
@version	text		•			
@version.keyword	keyword		•	•		
Entrenamiento	float, long		•	•		
Prediccion	float		•	•		
Tiempo de ejecución	float		•	•		
Valor	float		•	•		
._id	._id		•	•		
._index	._index		•	•		
._score						

Figura E.9: Configuración de un index pattern

¿Cómo funciona Monitorizacion_IOT ?

Una vez configurado e inicializado, el programa captará los datos de los sensores, los entrenará y realizará las predicciones. A continuación, se explicarán que procesos sigue.

Monitorizacion_IOT es un programa que se repite en bucle, cada ciclo se repite cada x tiempo, dependiendo del valor de la variable *repeticion_ciclo*. En cada ciclo se descargan, se entrenan y se predicen los datos de cada sensor.

Obtención y almacenamiento de datos

Para poder trabajar con los datos de los sensores, provenientes de PRTG es necesario descargarlos y almacenarlos para poder realizar el estudio y entrenamiento de dichos datos.

Monitorizacion_IOT , mediante un script, descarga los datos de los sensores que se encuentran en la lista **lista_id_sensor**, estos se guardan en un fichero JSON, el cual se ha de transformar para adecuar el formato y que Elasticsearch pueda procesarlo. Para almacenar los datos ya obtenidos y transformados a la base de datos de Elasticsearch, estos ficheros son preprocesados por logstash que se encarga de enviar las líneas de datos al servidor de Elastic e indexarlos correctamente.

Los datos son subidos con el siguiente nombre: “sensor_data-yyyy.mm.dd” de tal forma que todos los datos que se suben en un día se almacenan bajo un mismo índice. Para poder explorar y visualizar los datos en Kibana es necesario que exista un *index pattern* que diga a Elasticsearch que índices contienen los datos, así como especificar de que tipo son, en este caso se ha creado uno para que albergue todos los datos procedentes de sensores “sensor_data-*”.

Una vez los datos estén correctamente almacenados e indexados en Elasticsearch, se puede explorar y visualizar los datos con Kibana.

```
1  {
2    "prtg-version":"21.2.68.1492",
3    "treesize":30,
4    "histdata":
5    [
6      {
7        "datetime":"27/07/2021 16:11:06",
8        "Valor":4.6800,
9        "Tiempo de ejecución":782.0000,
10       "coverage":"100 %"
11     },
12     {
13       "datetime":"27/07/2021 16:12:06",
14       "Valor":4.6800,
15       "Tiempo de ejecución":797.0000,
16       "coverage":"100 %"
17     },
18     {
19       "datetime":"27/07/2021 16:13:06",
20       "Valor":4.6700,
21       "Tiempo de ejecución":799.0000,
22       "coverage":"100 %"
23     }
24   ]
25 }
```

Listing 3: JSON descargado de PRTG

Transformación de datos

Durante la transformación de los ficheros JSON se eliminan los datos irrelevantes que se obtenían de PRTG, se añade el id del sensor y se cambia el formato de la fecha de dd/MM/yyyy H:mm:ss a dd/MM/yyyy HH:mm:ss.

Logstash esta preparado para trabajar con logs, es por eso que además también se separa cada entrada de datos por líneas, cómo se puede observar el los Listings 3 y 4

```
1  {
2      "sensorId":"2051",
3      "datetime":"27/07/2021 16:11:06",
4      "reading":
5      {
6          "Valor":4.6800,
7          "Tiempo de ejecución":782.0000
8      }
9  }
10 {
11     "sensorId":"2051",
12     "datetime":"27/07/2021 16:12:06",
13     "reading":
14     {
15         "Valor":4.6800,
16         "Tiempo de ejecución":797.0000
17     }
18 }
19 {
20     "sensorId":"2051",
21     "datetime":"27/07/2021 16:13:06",
22     "reading":
23     {
24         "Valor":4.6700,
25         "Tiempo de ejecución":799.0000
26     }
27 }
```

Listing 4: JSON transformado

Machine learning

Incremental/Online Learning

Cómo se ha especificado en el apartado de conceptos teóricos el incremental learning va entrenando el modelo con datos en un flujo continuo. Esto es muy beneficioso ya que en este proyecto en concreto no se dispone de muestras lo suficientemente grandes como para entrenar un modelo mediante la forma tradicional de *batch learning* de forma eficaz.

Entrenamiento

Una vez los datos de los sensores están correctamente indexados y almacenados en Elasticsearch se puede proceder al entrenamiento.

Por cada sensor se cargará el modelo que se haya creado para ese modelo en concreto y se descargarán los datos en un rango de fecha determinado desde Elasticsearch. Los valores de entrenamiento serán volcados a un fichero y devueltos a Elasticsearch, para poder monitorizar el entrenamiento.

Una vez se ha finalizado se guarda el modelo en disco para que la siguiente vez que se requiera entrenar modelo o hacer predicciones se utilice siempre el modelo más actualizado.

Predicción

Cada ciclo del programa se hace una predicción de los siguientes minutos (según se indique en la variable *horizonte_prediccion*), para no acumular datos repetidos en la BBDD, se eliminan los datos que se han predicho en el ciclo anterior. Así cada predicción que hace el programa esta actualizada.

Esta predicción se sube a Elasticsearch para poder ser visualizada.

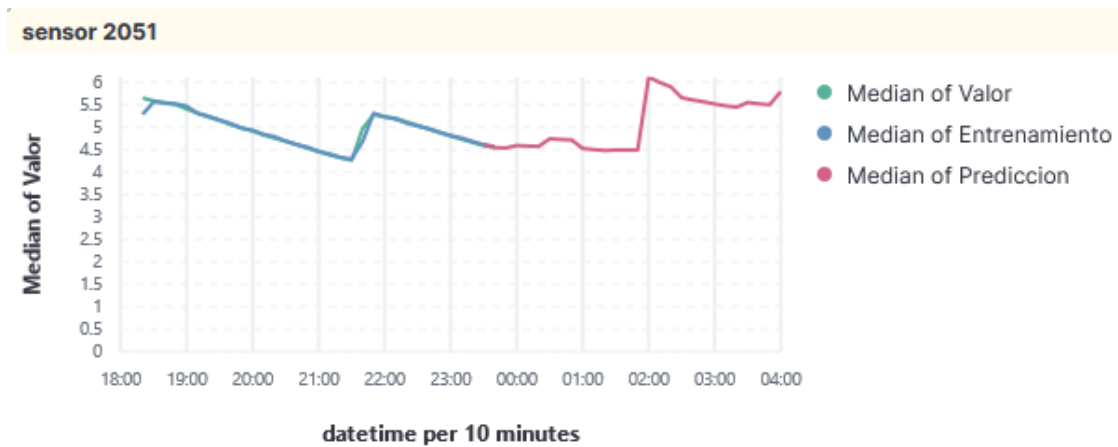


Figura E.10: predicción sensor de Presión Tubería de Agua

Visualización y monitorización

Mediante un navegador introducimos la siguiente url: IP_ubuntu_server:5601 (el puerto 5601 corresponde al de kibana).

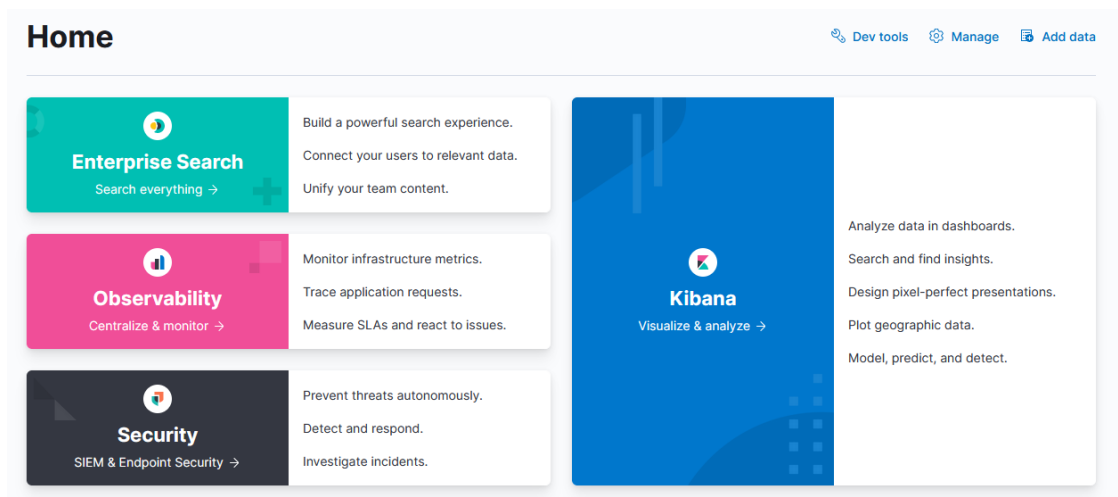


Figura E.11: pagina inicio kibana

Una vez en el apartado home podemos acceder al apartado *kibana Visualize & analyze* para poder echar un vistazo a nuestros datos. Para ver nuestros datos, así como realizar filtrados y búsquedas accederemos a *Discover*

Se pueden visualizar los datos que estén vinculados a un patrón de índices.

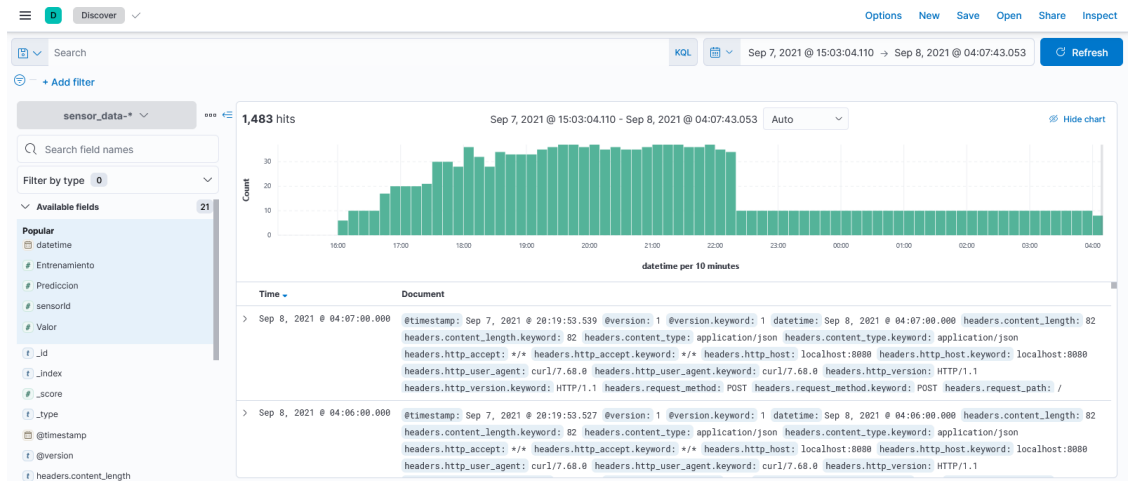


Figura E.12: kibana discover

Para visualizar de forma gráfica los datos se ha de acceder al apartado *Dashboard* en él se puede crear cualquier tipo de gráfico.

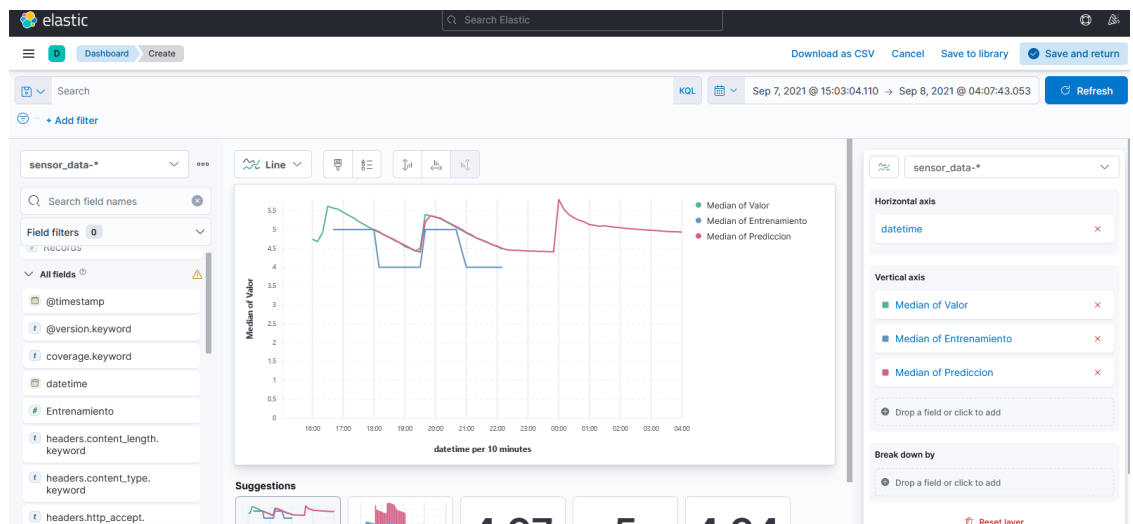


Figura E.13: kibana dashboard

Búsquedas y filtrados de datos

Kibana provee varias formas de construir *queries* con las cuales poder hacer filtrados y búsquedas en los datos almacenados en Elasticsearch.

En el apartado *Discover* se muestran los datos de un índice, se pueden realizar una serie de filtrados y búsquedas, el filtrado de tiempo limita los datos mostrados en un rango de fechas, este filtrado, normalmente, se aplica al campo de tiempo del *index pattern*

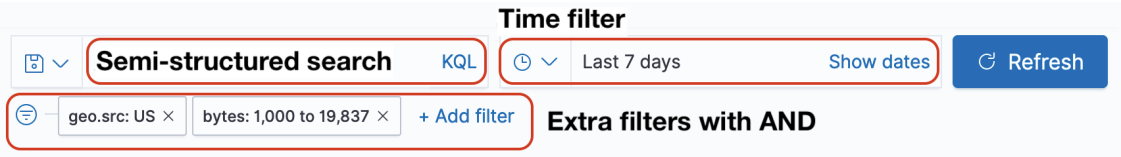


Figura E.14: kibana filtrados

La sintaxis que se utilizada es KQL (Kibana Query Language) un lenguaje creado específicamente para facilitar las búsquedas y filtrados en Elasticsearch. [4]

Valor		
	Valor	Filter results that contain Valor
<=>	:	equals some value
<=>	<=	is less than or equal to some value
<=>	>=	is greater than or equal to some value
<=>	<	is less than some value
<=>	>	is greater than some value
<=>	: *	exists in any form
&	and	Requires both arguments to be true
	or	Requires one or more arguments to be true
Q	Valor : 5 or 4	

Figura E.15: kibana ayuda de filtrado

Bibliografía

- [1] Brandy Greger and Maike Guba [PEASSLER support]. How can I export historic data from the PRTG API? <https://kb.paessler.com/en/topic/76768-how-can-i-export-historic-data-from-the-prtg-api>, 2019.
- [2] elastic. Kibana Guide [7.9] Create an index pattern. <https://www.elastic.co/guide/en/kibana/7.9/index-patterns.html>, 2021.
- [3] Elastic Docs. Http input plugin. <https://www.elastic.co/guide/en/logstash/current/plugins-inputs-http.html>, 2021.
- [4] Elastic Docs. Kibana concepts. <https://www.elastic.co/guide/en/kibana/7.14/kibana-concepts-analysts.html#autocomplete-suggestions>, 2021.