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 — 15 de septiembre de 2021
Tutor: Bruno Baruque Zanon

Índice general

Índice de figuras

Índice de tablas

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 describirá la planificación temporal que se ha seguido del proyecto, así como el estudio de viabilidad del mismo.

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 había una reunión con el tutor y se planteaban las próximas tareas que se iban a realizar.

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

- **Procut 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 cómo 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 **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 issues 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 así cómo Elasticsearch.

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 migro 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 librerías 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 tutórales 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 seré 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.

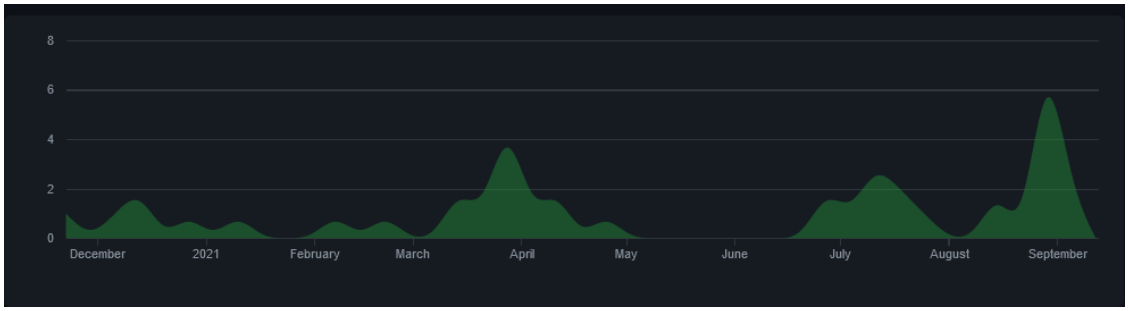

Figura A.1: Gráfico de commits realizados en el tiempo

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 legar 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

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	
PRTG	
virtualBox	
Ubuntu	
Windows 10	
Git	GPL2
GitHub	Propietaria/Privativa

Tabla A.4: 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:

- Realizar un algoritmo de aprendizaje automático que pueda predecir comportamientos y patrones en los datos de los sensores.
- La implementación de motores de búsqueda para felicitar 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.

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 programa ha de entrenar un modelo con los datos de los sensores.
- **RF-5 Predicción**: El programa 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 puede añadir sensores.
  - **RF-6.2 Borrar sensor**: El usuario puede eliminar sensores.
  - **RF-6.3 Configurar modelo**: El usuario puede 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-7.2 ver datos predicción**: Se ha de monitorizar los datos de predicción.

B.4. Especificación de requisitos

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

Diagrama de casos de usos

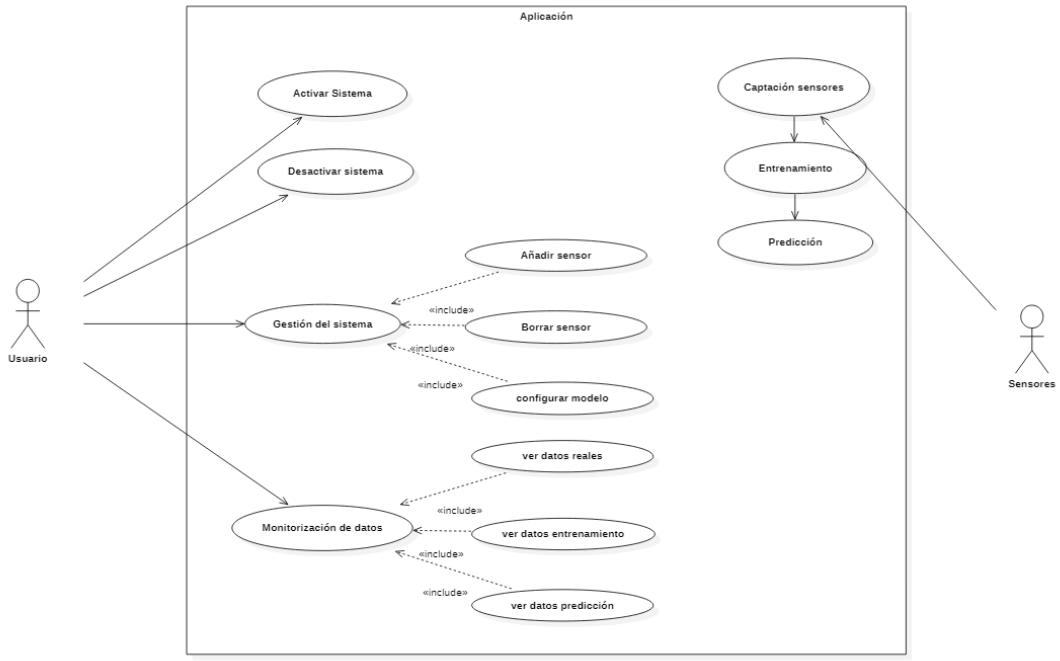

Figura B.1: Diagrama de casos de usos

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 los datos de los sensores.

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 programa 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 Elasicsearch
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 programa ha de entrenar un modelo con los datos de los sensores	
Requisitos	RF-4	
Precondiciones	Existen datos de los sensores en Elasitcsearch. 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 programa ha de entrenar un modelo con 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 puede 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 puede 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 puede 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 *Discover*.
	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	Elasticsearch no está activo, kibana no esta activo.	
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 *Discover*.
	3	El usuario elige un rango de fechas
	4	El usuario introduce filtra por el campo *valor*
Postcondiciones	Se muestran los valores reales de los sensores	
Excepciones	Elasticsearch no está activo, kibana no esta activo.	
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 *Discover*.
	3	El usuario elige un rango de fechas
	4	El usuario introduce filtra por el campo *entrenamiento*
Postcondiciones	Se muestran los valores de entrenamiento de los sensores	
Excepciones	Elasticsearch no está activo, kibana no esta activo.	
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 *Discover*.
	3	El usuario elige un rango de fechas
	4	El usuario introduce filtra por el campo *prediccion*
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.

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 E.1, la humedad en el ambiente C.4, la humedad en la parcela C.4 y la presión de la tubería de agua.

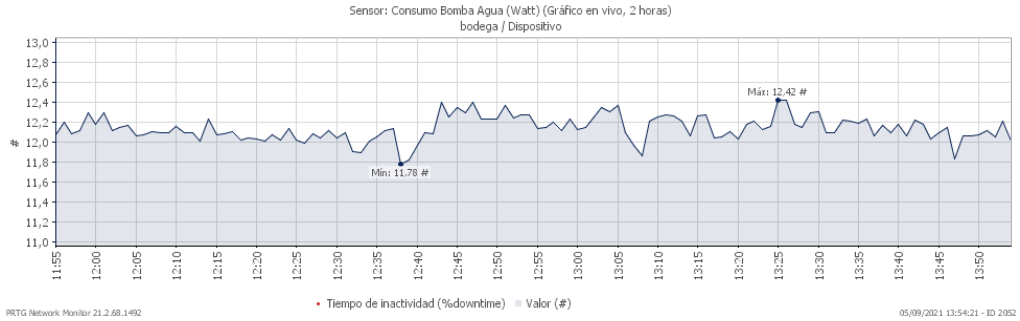

Figura C.1: Sensor del consumo de la bomba de agua

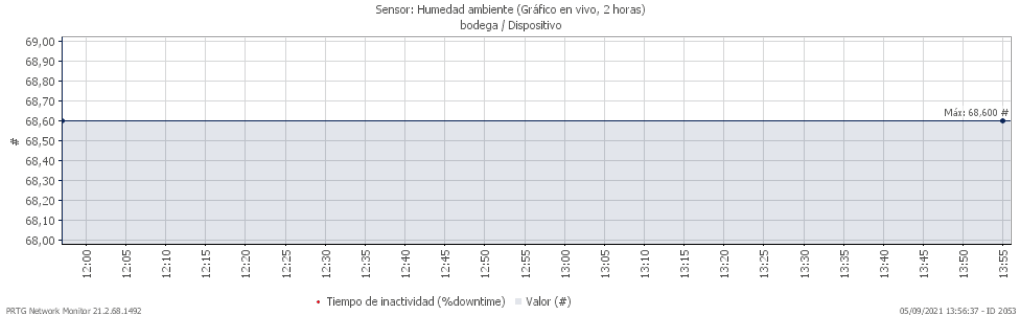

Figura C.2: Sensor de la humedad en el ambiente

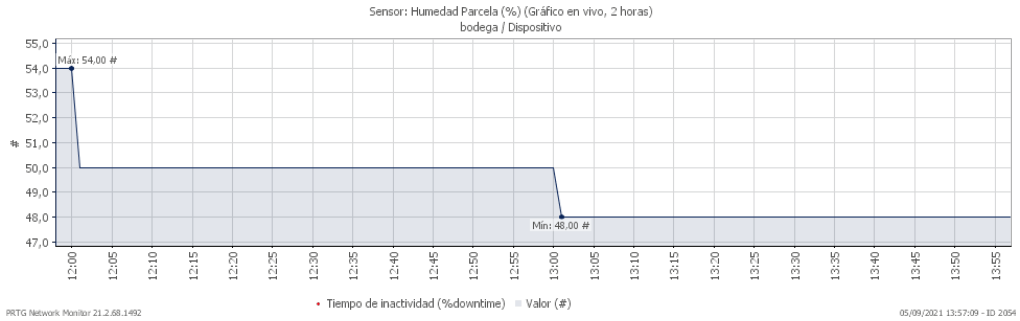

Figura C.3: Sensor de la humedad en la parcela

Base de datos

Los datos de los sensores 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

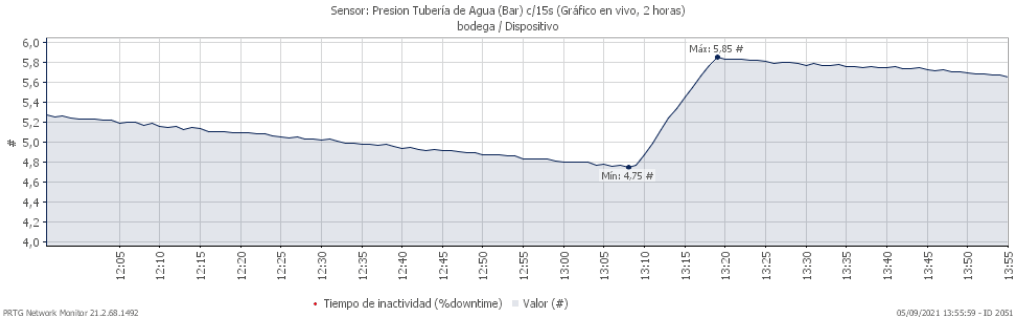

Figura C.4: Sensor de la presión de la tubería de agua

JSON 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 **??**

Diagrama de clases

A continuación, se mostrarán los diagramas de clase de la aplicación para facilitar el entendimiento de la estructura del sistema.

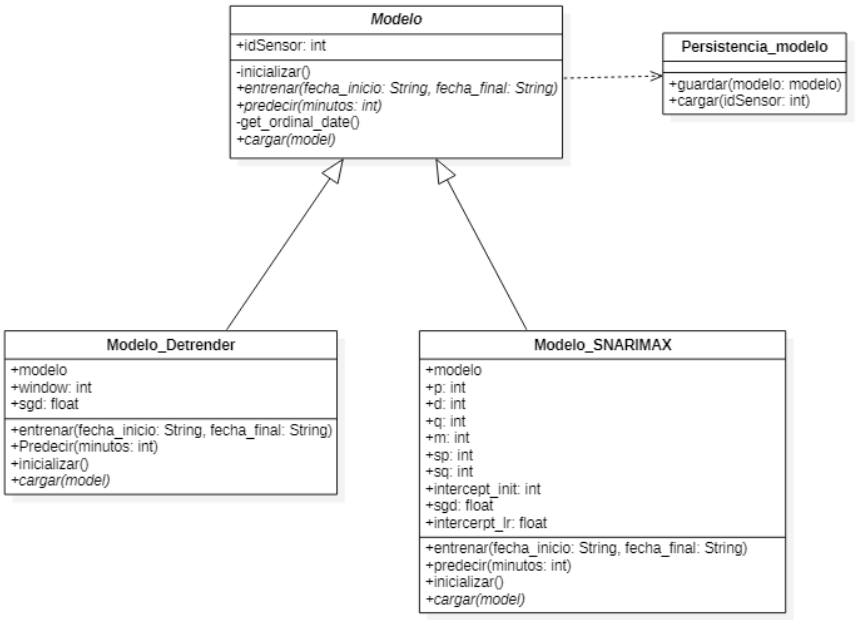

Figura C.5: Diagrama de clases paquete Predicción

C.3. Diseño procedimental

En este apartado se muestran el diagrama de secuencia

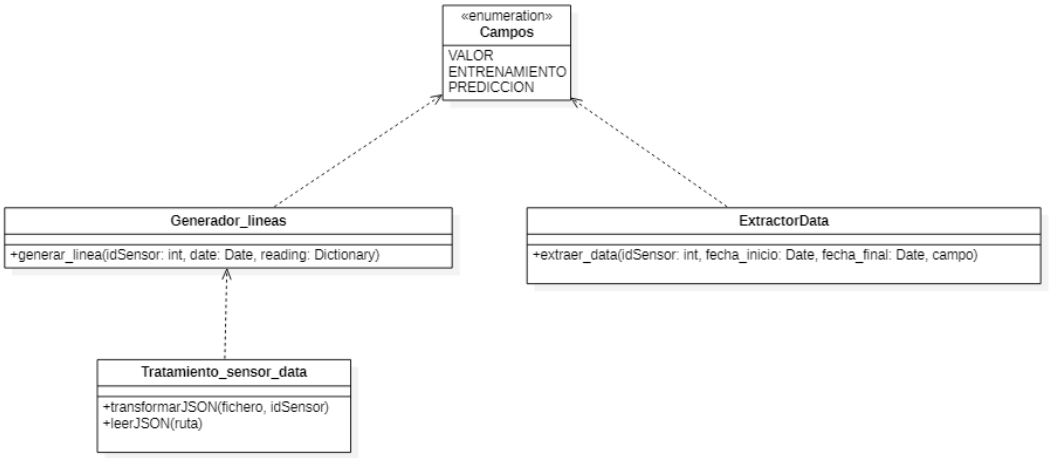

Figura C.6: Diagrama de clases paquete Utilidades

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.

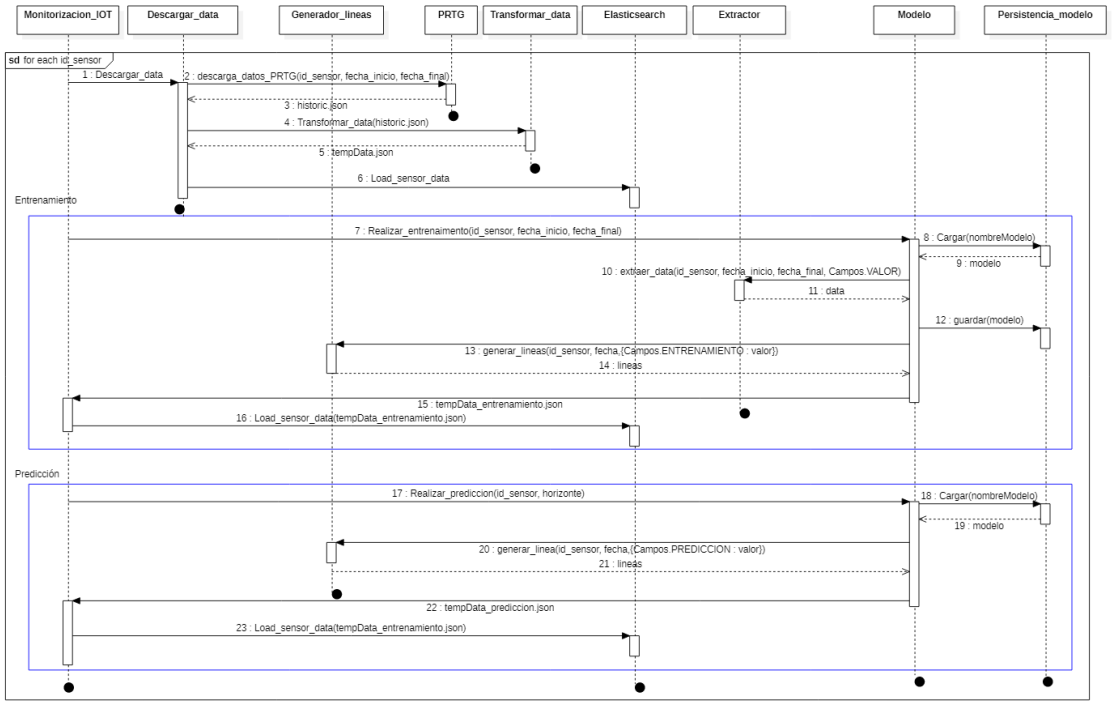

Figura C.7: Diagrama de secuencias

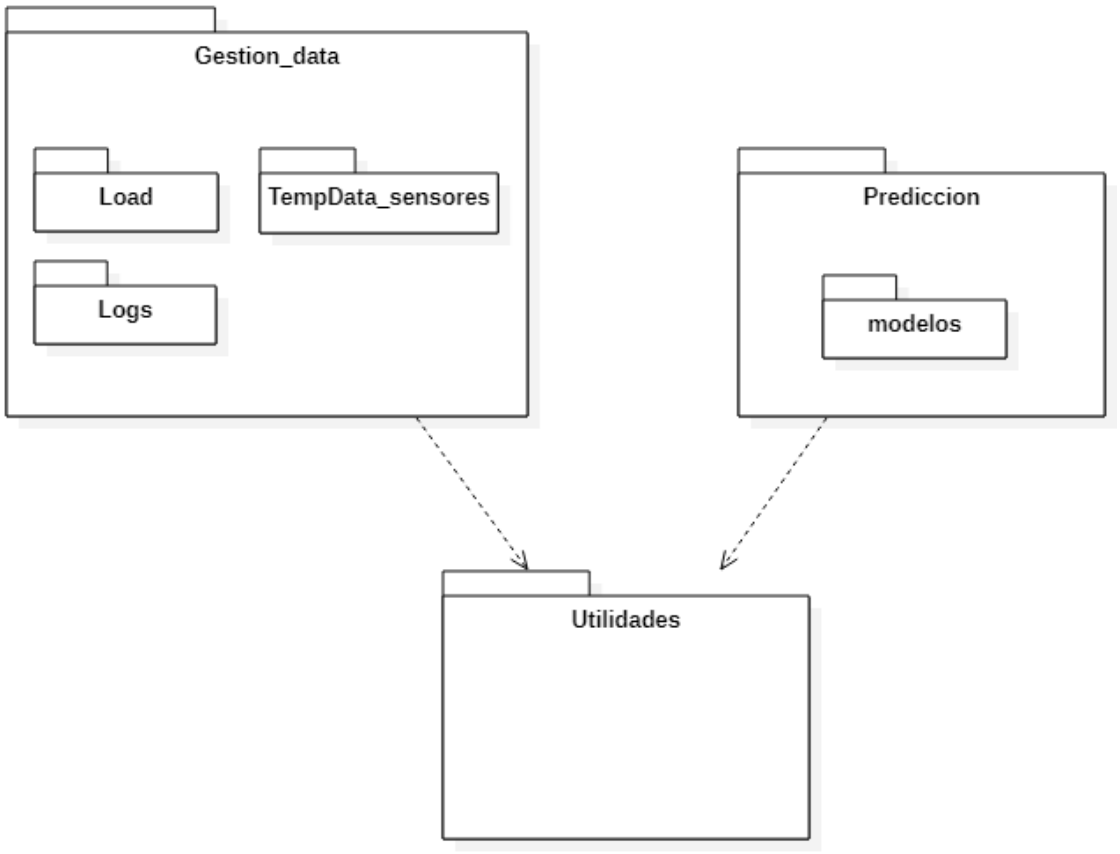

Figura C.8: Diagrama de paquetes

Apéndice D

Documentación técnica de programación

D.1. Introducción

En el siguiente anexo se describe la documentación técnica del programa, se detalla como 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 que se almacenan en la carpeta */usr/bin/Monitorizacion_IOT/Gestion_Data/Logs/*. De ahí son procesados y guardados temporalmente en la carpeta

/usr/bin/Monitorizacion_IOT/Gestion_Data/TempData_sensores/

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

En la carpeta */usr/bin/Monitorizacion_IOT/Utilidades/* se encuentran diversas funciones de utilidad.

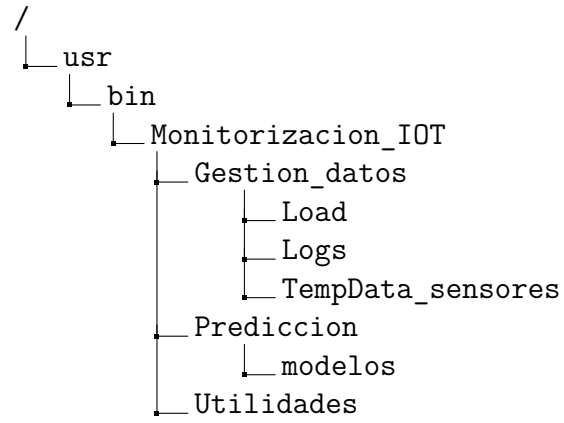

Figura D.1: Directorios del proyecto

D.3. Manual del programador

El manual del programador tiene como objetivo ayudar a todas aquellas personas que vayan a trabajar en 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

Librerías python:

- river versión 0.71
- pandas: versión 1.3.2
- Elasticsearch: versión 7.14.0
- numpy: versión 1.21.2

Archivos

- **Monitorizacion_IOT.sh**: Este es el script principal de la aplicación el cual esta hilado a un daemon, de tal forma que se repite en bucle cada minuto en el cual se descargan los datos de PRTG, se suben a Elasticsearch, se realiza 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
- **/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 en entrenamiento del modelo, se suben las predicciones realizadas durante el entrenamiento por el modelo a Elasticsearch y finalmente se guarda el modelo en disco.
- **/Predicciones/Realizar_prediccion.py**: Es el fichero encargado de realizar la perdición de un sensor, se carga el modelo correspondiente al sensor que se desea predecir y se realza la perdición a partir de la fecha actual y tantos instantes de tiempo cómo 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/Extractor.py**: En este fichero se encuentra la clase encargada de la descarga de datos de Elasticsearch.
- **/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/Tatamiento_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.

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, model)**: 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

Configuración Ubuntu Server

Para albergar nuestro programa utilizaremos una máquina virtual Ubuntu Server.

Antes de comenzar la instalación tendremos que asegurarnos que estén instaladas las herramientas de red de Linux, las cuales ayudarán a saber la IP, y con ella poder acceder al servidor de forma remota.

```
apt install net-tools
```

También es necesario comprobar la zona horaria del server 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 podemos proceder a la instalación de Monitorizacion_IOT . Para ello es necesario descargarse del repositorio tanto el script de *install.sh* cómo el paquete debian *MonitorizacionIOT_1.0_all.deb*

Una vez tengamos estos ficheros en nuestra máquina virtual comenzaremos la instalación, para mayor comodidad, asegurarse que accedemos con el usuario root, para ello se puede utilizar el siguiente comando:

```
sudo su
```

ejecutaremos el script *install.sh* que instalará una serie de paquetes necesarios, cómo son Elasticsearch, logstash, kibana y librerías de python.

```
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
```

Una vez instaladas las dependencias procederemos a instalar el paquete Debian.

```
sudo dpkg -i  MonitorzacionIOT_1.0_all.deb
```

El paquete se encarga de instalar los archivos del programa, el cual se encuentra en la ruta */usr/bin/Monitorizacion-IOT/*

El paquete también se encarga de configurar Elasticsearch, kibana y logstash así cómo iniciar los servicios.

Configuración

Tras instalarse el paquete se realizan una serie de configuraciones sobre los servicios de Elasticsearch y kibana.

Elasticsearch

El archivo de configuración se encuentra en la ruta:

/etc/elasticsearch/elasticsearch.yml

nota: *Importante, en el archivo de configuración no puede haber ningún espacio al inicio de las líneas.*

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
```

Configurar Monitorizacion_IOT

Una vez se haya instalado el paquete Debian hay una serie de cambios que hay que realizar manualmente antes de poder ser ejecutado.

El primero sería en el fichero del logstash:

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

En el se encuentra el mecanismo que permite a logstash procesar los archivos de logs con los datos de nuestros 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 que logstash mande la información a nuestro Elasticsearch tendremos que sustituir donde pone "IP_Ubuntu_server" por la IP de nuestro server, en el cual se ha instalado Elasticsearch.

Index pattern

A continuación accederemos a Kibana, para ello introduciremos en un navegador web de nuestro host la url: IP_Ubuntu_server:5601

accederemos a Dev Tools y configuraremos el índice que utilizará logstash para poder subir y clasificar los datos.

```
POST _template/sensor_data_template
{
  "index_patterns": [
    "sensor_data*"
  ],
  "settings": {
    "number_of_replicas": "1",
    "number_of_shards": "5"
  },
  "mappings": {
    "properties": {
      "sensorId": {
        "type": "integer"
      },
      "datetime":{
        "type": "date",
        "format": "dd/MM/yyyy HH:mm:ss"
      },
      "reading": {
        "type": "nested",
        "properties": {
          "name": {"type":"keyword"},
          "description": {"type":"double"}
        }
      }
    }
  }
}
```

Listing 1: Índice

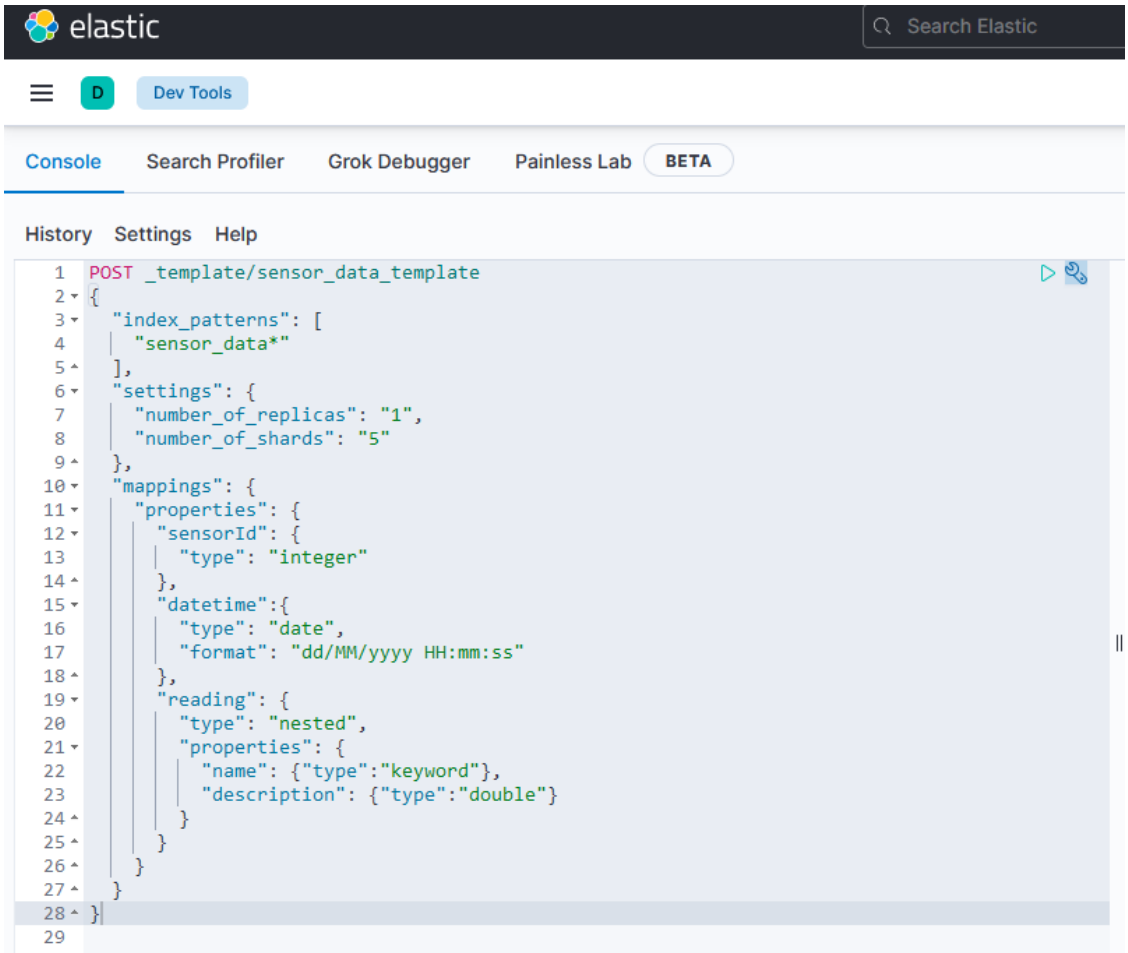

Figura D.2: Crear un índice

Una vez creado el índice todos los datos procedentes de losgstash se almacenarán con ese índice permitiendo, dentro del menú Stack Management en el apartado index Managment consultar los datos que va recibiendo.

Para finalizar y 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, en este caso se ha de crear uno para que albergue todo los datos procedentes de sensores "sensor_data-*"[1]

Una vez creado el patrón nos pedirá que lo configuremos a nuestro gusto, podremos especificar cuál va a ser el índice de nuestros datos, si el tiempo en el que es indexado a en Elasticsearch "@timestamp" u otro campo fecha que nosotros prefiramos.

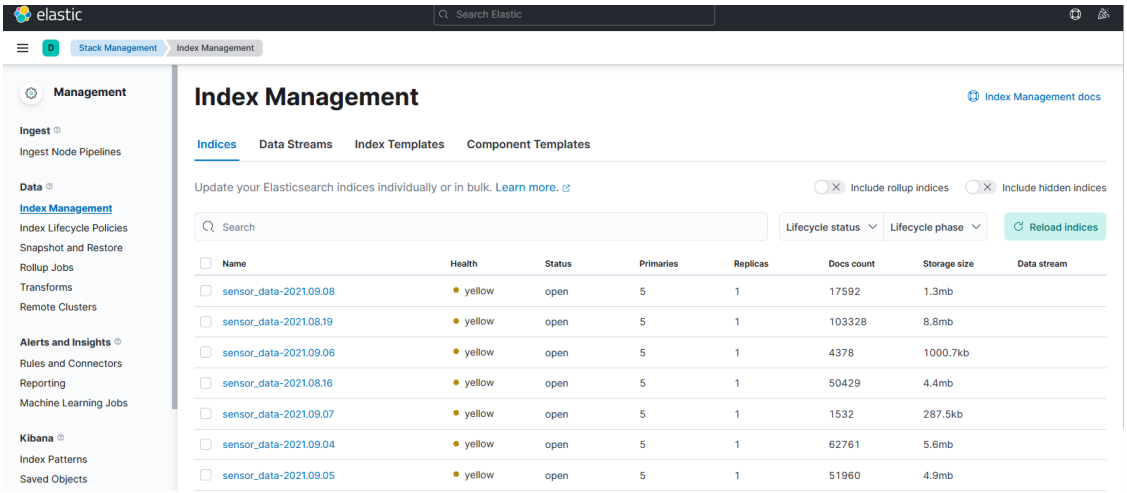

Figura D.3: Index Managment

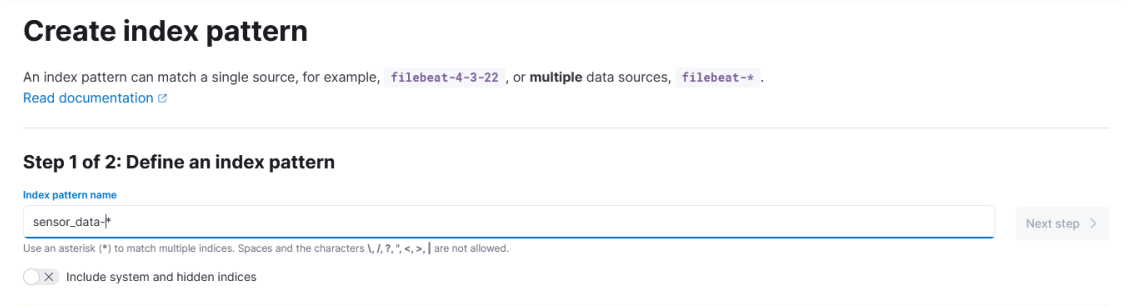

Figura D.4: Creación de un index pattern

También se puede cambiar el tipo de cualquier otro campo.

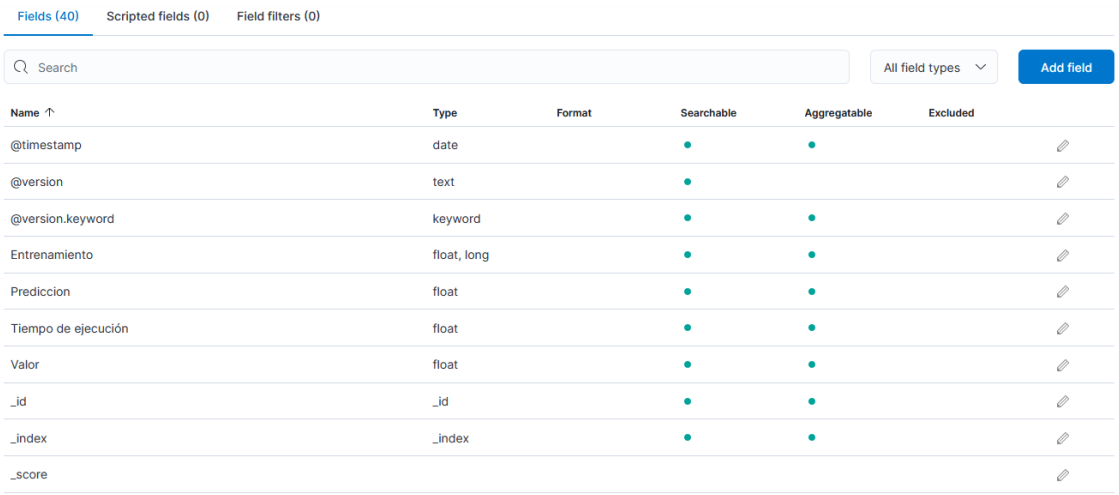

Figura D.5: Configuración de un index pattern

Inicializar Monitorizacion_IOT

Una vez instalado y configurado ya podremos iniciar nuestro programa introduciendo el siguiente comando como usuario administrador.

```
systemctl start Monitorizacion_IOT
```

También puede ser de utilidad comprobar tanto el estado de nuestro programa como el de Elasticsearch, kibana o logstash en caso de que se haya producido algún fallo.

```
systemctl status Monitorizacion_IOT

systemctl status elasticsearh
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 elasticsearh
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.

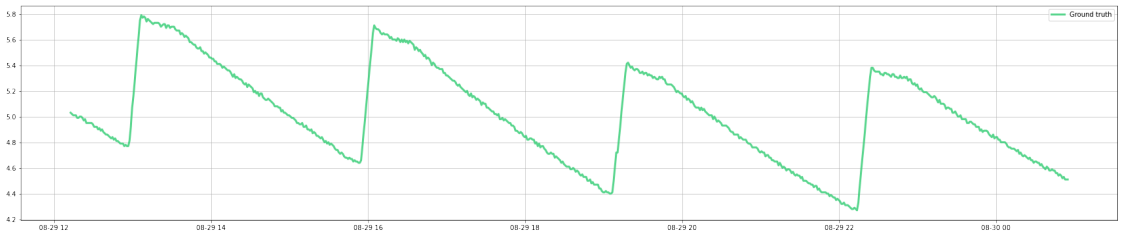

Figura D.6: Dataset sensor de Presión Tubería de Agua

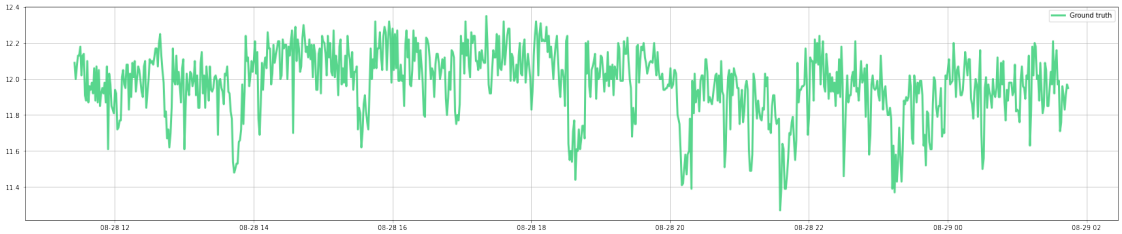

Figura D.7: Dataset sensor Consumo de bomba de agua

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.

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

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

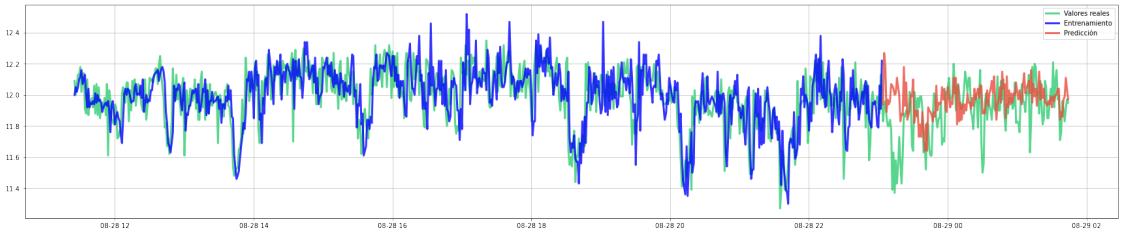

Figura D.8: Consumo Bomba Agua en las fechas: 28/08/2021 - 29/08/2021

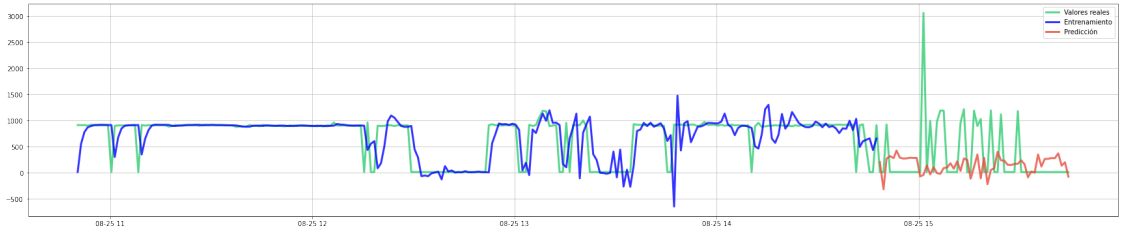

Figura D.9: Consumo Bomba Agua en las fechas: 25/08/2021 - 26/08/2021

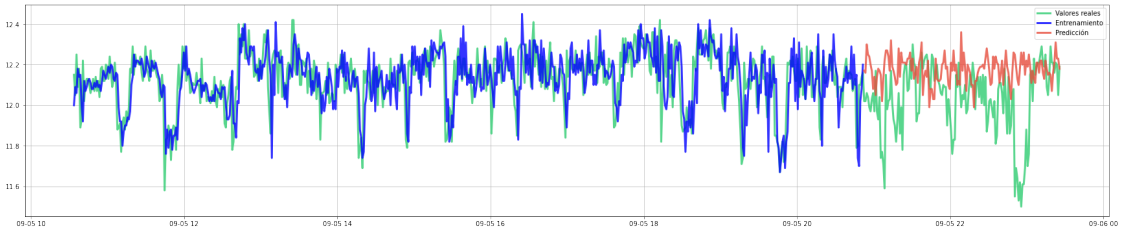

Figura D.10: Consumo Bomba Agua en las fechas: 05/09/2021 - 06/09/2021

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

Pruebas Sensor Presión tubería de agua

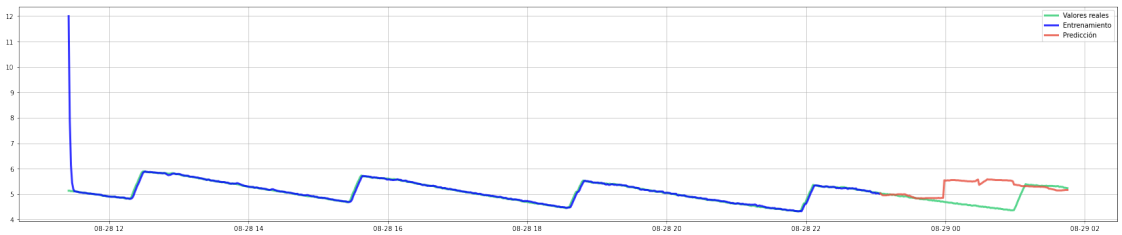

Figura D.11: Presión tubería de agua en las fechas: 28/08/2021 - 29/08/2021

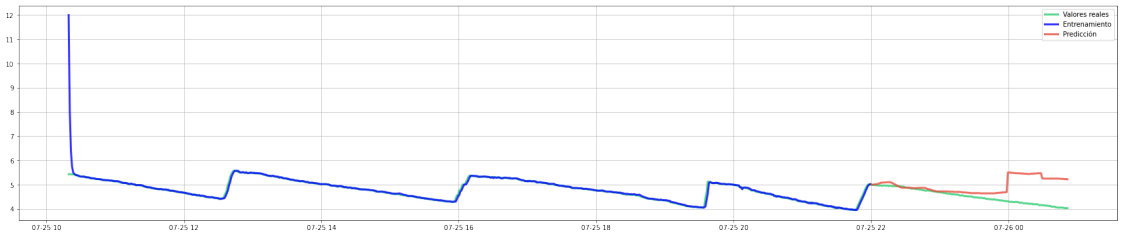

Figura D.12: Presión tubería de agua en las fechas: 25/07/2021 - 26/07/2021

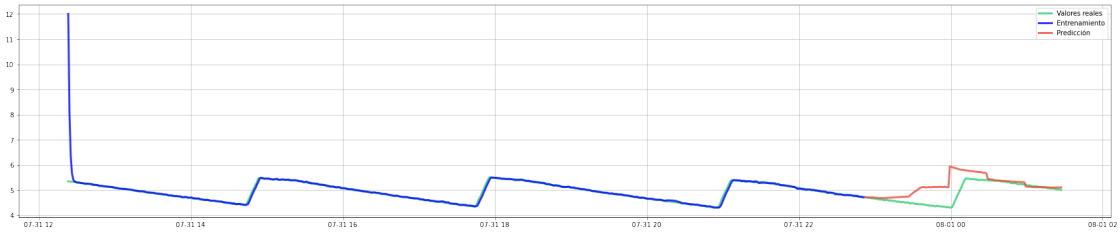

Figura D.13: Presión tubería de agua en las fechas: 31/07/2021 - 01/08/2021

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

Apéndice E

Documentación de usuario

E.1. Introducción

En este apartado se mostrarán los requisitos necesarios para instalar la aplicación y 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

Los pasos para instalar la aplicación se encuentran detalladas en el anexo dedicado a la instalación y compilación dentro del manual del programador.

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 instalado el programa el usuario podrá elegir que sensores desea almacenar y entrenar así cómo elegir 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 encintarán 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.
- **tiempo_espera**: Se ha de introducir el tiempo en segundos que ha de esperar el sistema entre ciclo y ciclo.

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
```

Obtención y almacenamiento de datos

Para poder trabajar con los datos de los sensores inicialmente almacenados en PRTG era necesario descargarlos y almacenarlos por nuestra cuenta, 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 hay que transformar para adecuar el formato del

fichero 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 almacenan en un día se almacenan bajo un mismo índice. 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, en este caso se ha creado uno para que albergue todo los datos procedentes de sensores "sensor_data-*".

Una vez los datos estén correctamente almacenados e indexados en Elasticsearch, podemos proceder a explorar y visualizar los datos con Kibana.

Elasticsearch provee un motor de búsqueda utilizando Query DSL (lenguaje de dominio específico) lo que nos permite filtrar fácilmente entre todos los datos que se van almacenando.

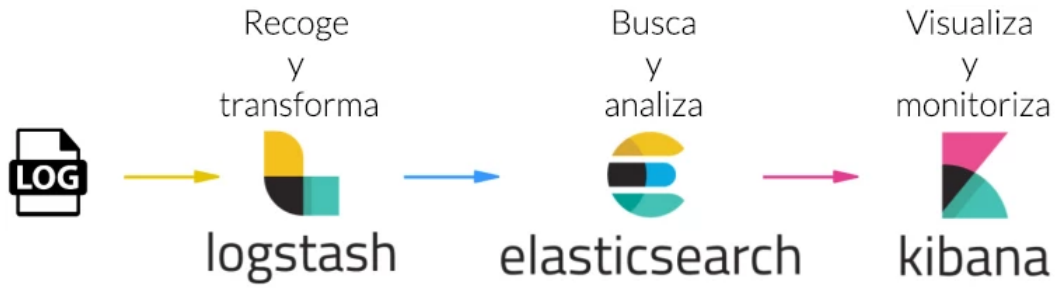

Figura E.1: ELK stack

Transformación de datos

Para la transformación del fichero JSON, se empiezo utilizando un sencillo programa en Flex, un procesador de lenguaje, que eliminaba los datos irrelevantes que se obtenían de PRTG y añadía el id del sensor, así como separaba las líneas dentro del fichero para poder mandárselo a Elasticsearch a través de logstash cómo campo el ID del sensor, cómo se puede observar el los Listings 2 y 3

Tras un tiempo y debido a que era necesario modificar el formato de las fechas se decidió pasar esta funcionalidad a Python ya que facilita el tratamiento con fechas.

```
{
    "prtg-version":"21.2.68.1492",
    "treesize":30,
    "histdata":
    [
        {
            "datetime":"27/07/2021 16:11:06",
            "Valor":4.6800,
            "Tiempo de ejecución":782.0000,
            "coverage":"100 %"
        },
        {
            "datetime":"27/07/2021 16:12:06",
            "Valor":4.6800,
            "Tiempo de ejecución":797.0000,
            "coverage":"100 %"
        },
        {
            "datetime":"27/07/2021 16:13:06",
            "Valor":4.6700,
            "Tiempo de ejecución":799.0000,
            "coverage":"100 %"
        }
    ]
}
```


Listing 2: JSON descargado de PRTG

E.5. Machine learning

Incremental/Online Learning

Cómo se ha especificado en apartado de conceptos teóricos el incremental learning va entrenando el modelo con datos en un flujo continuo.

En nuestro caso esto es muy beneficioso ya que no disponemos de muestras lo suficientemente grandes como para entrenar un modelo de forma eficaz.

por lo que el modelo va aprendiendo a medida que se van recogiendo los datos, esto hace a nuestro algoritmo sea escalable.

```
{
    "sensorId":"2051",
    "datetime":"27/07/2021 16:11:06",
    "reading":
    {
        "Valor":4.6800,
        "Tiempo de ejecución":782.0000
    }
}
{
    "sensorId":"2051",
    "datetime":"27/07/2021 16:12:06",
    "reading":
    {
        "Valor":4.6800,
        "Tiempo de ejecución":797.0000
    }
}
{
    "sensorId":"2051",
    "datetime":"27/07/2021 16:13:06",
    "reading":
    {
        "Valor":4.6700,
        "Tiempo de ejecución":799.0000
    }
}
```

Listing 3: JSON transformado

Modelos

La librería de Python, River, nos facilita dos tipos de modelos para series temporales que nos permitirán realizar el entrenamiento de manera incremental con un data stream procedente de nuestros sensores así cómo realizar una predicción del comportamiento de nuestros sensores.

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

Antes de nada, se ha de crear un modelo para un id en concreto, se puede decidir qué tipo de modelo así cómo los parámetros necesarios.

Crear un modelo para un sensor

para crear un modelo para un sensor 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 de nuestro modelo, pasándole los parámetros necesarios. 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)
```

Entrenamiento

Una vez los datos procedentes de los sensores han sido correctamente se puede realizar el entrenamiento de los modelos.

Por cada sensor que se dese entrenar 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.

Una vez acabado el entrenamiento 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, 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 y utiliza el modelo más entrenado disponible.

Para ello cada vez que se hace la perdición se carga el modelo que se ha entrenado para un sensor específico y se le pide que saque una predicción de los instantes de tiempo que se especifiquen

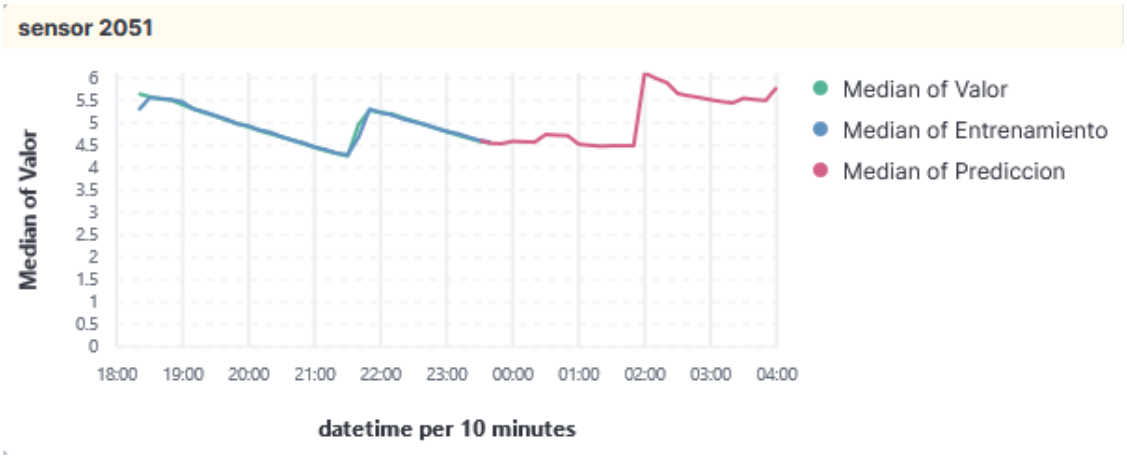

Figura E.2: predicción sensor de Presión Tubería de Agua

E.6. Visualización y monitorización

Para poder visualizar y monitorizar nuestros datos debe de estar nuestro servidor funcionando.

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

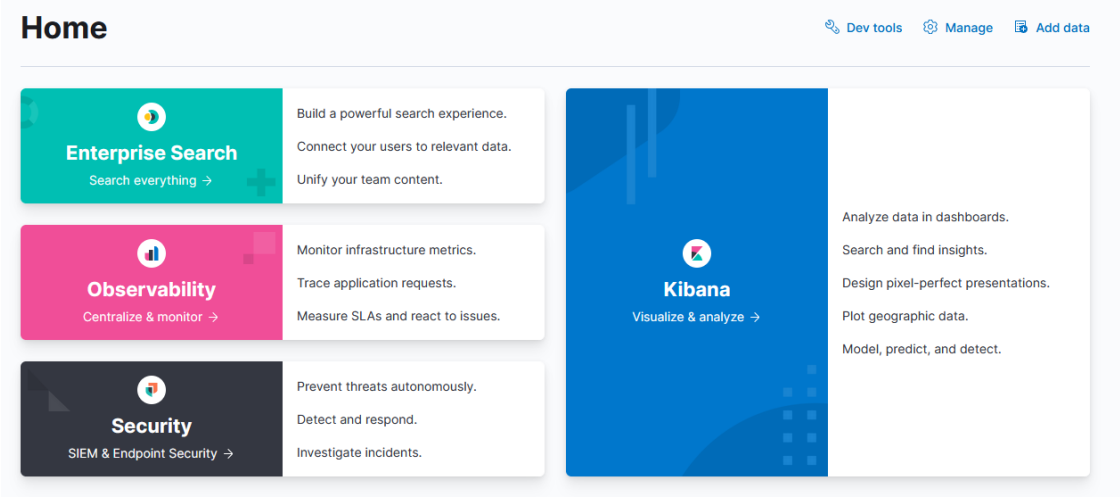

Figura E.3: pagina inicio kibana

Una vez en el apartado home podemos acceder al apartado “kibana Visualize analyze” para poder echar un vistazo a nuestros datos. Para ver 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.

Para visualizar de forma gráfica nuestros datos accederemos al apartado “Dashboard” en el podemos crear cualquier tipo de gráfico.

Para más información acceder a la documentación de kibana en el que se detalla todo lo que se puede realizar con esta herramienta de visualización.

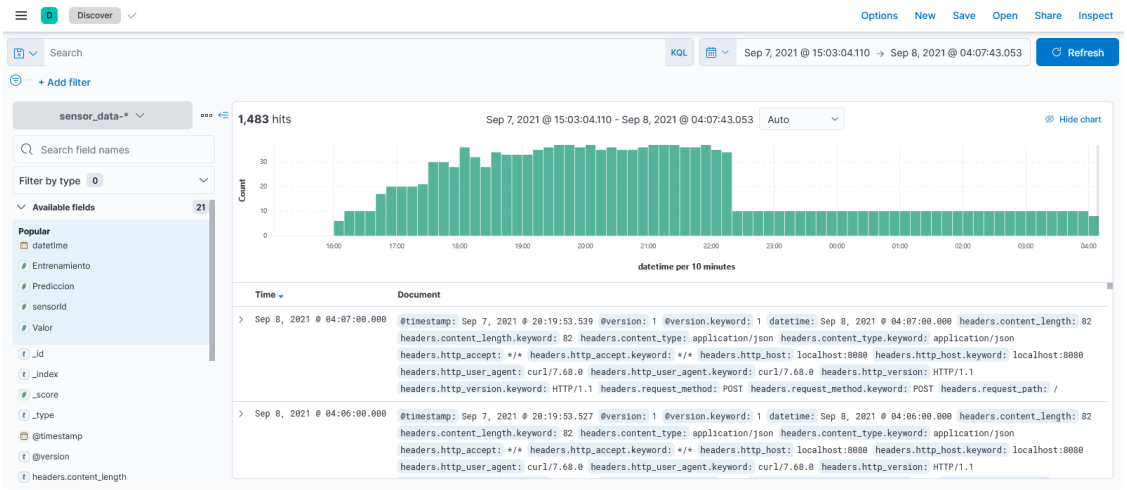

Figura E.4: kibana discover

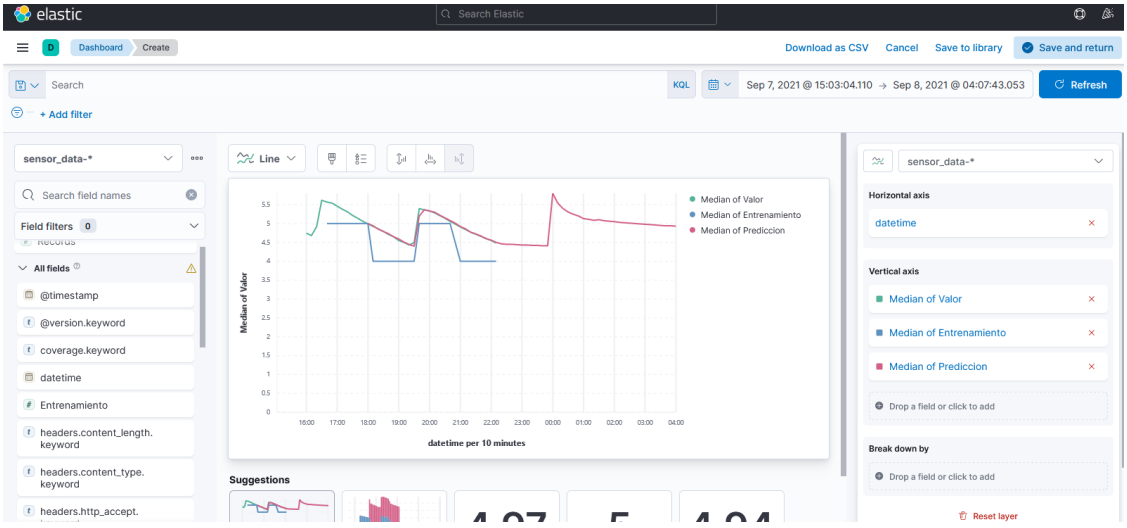

Figura E.5: kibana dashboard

Bibliografía

[1] elastic. Kibana guide [7.9] create an index pattern, 2020.