Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Electrónica de Comunicaciones

Trabajo Fin de Grado

Aplicación de visión artificial basada en Siemens Industrial Edge.

Autor: Diego Muñoz López

Tutor: Alfredo Gardel Vicente

2022

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

Grado en Ingeniería Electrónica de Comunicaciones

Trabajo Fin de Grado

Aplicación de visión artificial basada en Siemens Industrial Edge

Autor: Diego Muñoz López

Tutor: Alfredo Gardel Vicente

TRIBUNAL:

Presidente: Marta MARRÓN ROMERA

Vocal 1º: Cristina LOSADA GUTIÉRREZ

Vocal 2º: Alfredo GARDEL VICENTE

FECHA: Junio/2022

A mis hijos Laura y Diego.
A mis compañeros Raul y Joaquín por compartir sus amplios conocimientos de puesta en marcha virtual conmigo, a Alberto De La Merced por la guía y consejo que me ha dado durante muchos años, gracias a todos.

Y especialmente gracias a ti Silvia, ya que sin tu comprensión y apoyo esto no hubiera sido posible.

Non quia difficilia sunt non audemus, sed quia non audemus difficilia sunt.
[No nos atrevemos a muchas cosas porque son difíciles, pero son difíciles porque no nos atrevemos a hacerlas.]

Seneca, Lucio Anneo (4 a.C.-65 d.C.)
Carta a Lucilio: 104, 26)

Índice

Índice de figuras III
Lista de Acrónimos y Abreviaturas IX
Resumen 1
Abstract 1
1. Introducción 2
1.1 Descripción del TFG 2
1.2 Industrial Edge Computing 4
1.3 Docker 6
1.4 Visión Artificial 11
1.5. Técnicas de Machine Learning para Visión Artificial 14
1.6 Objetivos 31
2. Alternativas comerciales en computación perimetral industrial 32
2.1 Amazon Web Services (AWS) 32
2.2 Microsoft Azure IoT Edge 34
2.3 Siemens Industrial Edge 40
2.4 Comparativa de alternativas Edge Computing 42
3. Arquitectura Industrial Edge 45
3.1 Visión general 45
3.2 Conceptos de redes con Siemens Industrial Edge 48
4. Gestión de Apps: Industrial Edge Management 54
4.1 Visión general 54
4.2 Inicializar el IEM 56
4.3 Servidor web local del IEM 59
4.4 Gestión centralizada IEM 64
4.5 Instalación de una app 65
5. Entorno de desarrollo de apps para Siemens IE 68
5.1 Visión general 68
5.2 Configuración del PC de desarrollo 69
5.3. Interfaz web de usuario (Web UI). Flask 75
5.4. Docker Engine 82
5.5 Industrial Edge App Publisher 87
6. Desarrollo Apps para Siemens IE 89
6.1 Visión general 89
6.2. Desarrollo con contenedores 90
6.3. Desarrollo con apps para IE 101
6.4. Depuración y diagnostico 117
7. Visión Artificial con Azure Cognitive Services 126
7.1 Azure Cognitive Services 126
7.2 Analizar y clasificar imágenes con el servicio Computer Vision 127
7.3 Detección de objetos con Azure Custom Vision 132
8. Desarrollo de una App para visión artificial. 145

8.1 Backend de la aplicación 145
8.2 Frontend de la aplicación 148
8.3 Dockerización de la aplicación 151
8.4 Publicación con IEAP 152
8.5 Instalación App con IEM 161
8.6 Proyecto HMI/PLC en TIA Portal 166
9. Puesta en marcha virtual 169
9.1 Visión general 169
9.2 Gemelo Digital 172
9.3 Arrancar la simulación 175
10. Conclusiones y trabajo futuro 178
10.1 Conclusiones 178
10.2 Trabajo futuro 179
Presupuesto 181
Bibliografía y referencias 184

Índice de figuras

Figura 1-1 Esquema del caso de uso Industrial Edge 2
Figura 1-2 Infraestructura computación perimetral. Fuente: Wikipedia 4
Figura 1-3 Casos de uso en función de los requisitos de latencia. Fuente: GSMA Intelligence 5
Figura 1-4 Entorno de desarrollo y motor Docker. Fuente: Siemens Industry Image Database 4.11 7
Figura 1-5 Esquema de la arquitectura Docker. Fuente: Docker Inc. 8
Figura 1-6 Concepto Docker – Dockerfile, Imágenes y Contenedores. Fuente: Platformer 10
Figura 1-7 Arquitectura Docker izquierda y de virtualización a la derecha. Fuente: Docker Inc. 10
Figura 1-8. Docker Hub 11
Figura 1-9 Etapas de un sistema de Visión Artificial. Fuente: Industrial image processing on SIMATIC IPCs 14
Figura 1-10 Tipos de inteligencia artificial. Fuente: UAH. 15
Figura 1-11 Diseño de un sistema de reconocimiento de objetos 15
Figura 1-12 Ejemplo clustering utilizando k-means. Fuente: UAH 19
Figura 1-13 Algoritmos de Machine Learning. Fuente: UAH. 20
Figura 1-14 Redes neuronales multicapa. Fuente: UAH. 23
Figura 1-15 Resultados de la búsqueda de imágenes de cajas de medicamentos en Google.Fuente: Google 25
Figura 1-16 Métricas del modelo obtenidas con Custom Visión Portal de Azure. 26
Figura 1-17 Problemas típicos a la hora de entrenar una red neuronal convolucional. Fuente: UAH. 27
Figura 1-18 "Piedra de Rosetta" de marcos de aprendizaje profundo. Fuente: Github 28
Figura 1-19 Training Time(s): CNN (VGG-style, 32bit) on CIFAR-10 - Image Recognition. Fuente: Github 29
Figura 1-20 Deep Learning Framework Power Scores 2018 by Jeff Hale. Fuente: www.towardsdatascience.com 30
Figura 1-21 Fases para el desarrollo de aplicaciones Industrial Edge. Fuente: Siemens Industry Image Database 4.11 31
Figura 2-1 El siguiente ejemplo muestra cómo AWS IoT Greengrass interactúa con la nube de AWS. Fuente AWS 33
Figura 2-2 Microsoft Azure Marketplace con los “módulos” disponibles. Fuente: Azure 35
Figura 2-3 Entorno de ejecución de Azure IoT Edge. Fuente: Azure 36
Figura 2-4 Tabla de precios Azure IoT Hub. Fuente: Azure 36
Figura 2-5 Tabla de precios de Módulos Azure IoT Edge. Fuente: Azure 37
Figura 2-6 Configuración de dispositivos. Fuente: Azure 37
Figura 2-7 Sincronización con IoT Hub. Fuente: Azure 38
Figura 2-8 Desconexión. Fuente: Azure 38
Figura 2-9 Nueva conexión y sincronización con IoT Hub. Fuente: Azure 39
Figura 2-10 Arquitectura del ecosistema Siemens Industrial Edge. Fuente: Siemens Industry Image Database 4.11 41
Figura 2-11 Tabla de precios orientativos Siemens Industrial Edge. Fuente: Siemens Industry Image Database 4.11 41
Figura 2-12 Negocio mundial Cloud. Fuente: Synergy Research Group 42

Figura 3-1 Componentes Siemens Industrial Edge. Fuente: Siemens Industry Image Database 4.11....45
Figura 3-2 Consola principal del Industrial Hub.. Fuente: Siemens Industry Image Database 4.11....46
Figura 3-3 Industrial Edge Management. Fuente: Siemens Industry Image Database 4.11....46
Figura 3-4. Estadísticas Industrial Edge Management 47
Figura 3-5 Resumen de las apps de Siemens Industrial Edge en función de la funcionalidad. Fuente: Siemens Industry Image Database 4.11 48
Figura 3-6. Opciones de arquitecturas de red IE. Fuente: Siemens Industry Image Database 4.11....48
Figura 3-7 Arquitectura típica de red para el despliegue de Industrial Edge en planta. Fuente: Siemens Industry Image Database 4.11 49
Figura 3-8. Arquitectura de red interna IE. Fuente: Siemens Industry Image Database 4.11 .50
Figura 3-9 Diseño de red dentro de un dispositivo Siemens Industrial Edge. Fuente: Industrial Edge - App Developer Guide V1.2.1 51
Figura 3-10. Bridge entre aplicaciones IE. Fuente: Industrial Edge - App Developer Guide V1.2.1....52
Figura 3-11Una imagen de ejemplo del funcionamiento de la exposición de puertos. Fuente: Colaboratorio.net 53
Figura 4-1Gestión local de los IED. Fuente: Siemens Industry Image Database 4.11 54
Figura 4-2 Los dispositivos Edge se pueden administrar de forma escalable. Fuente: Siemens Industry Image Database 4.11....55
Figura 4-3 Recursos HW destinados para IE en un Unified Comfort Panel. Fuente: Siemens Industry Image Database 4.11....56
Figura 4-4. Panel de control de un Unified Comfort Panel. 57
Figura 4-5. Captura de imagen del IEM 58
Figura 4-6 Captura de imagen del IEM 58
Figura 4-7 Captura de imagen del IEM 59
Figura 4-8 Captura de imagen del IEM 59
Figura 4-9 Captura de imagen del IEM 60
Figura 4-10 Captura de imagen del IEM 60
Figura 4-11 Captura de imagen del IEM 61
Figura 4-12 Captura de imagen del IEM 61
Figura 4-13 Captura de imagen del IEM 62
Figura 4-14 Captura de imagen del IEM 62
Figura 4-15 Captura de imagen del IEM 63
Figura 4-16 Captura de imagen del IEM 63
Figura 4-17 Página de inicio de Industrial Edge Hub después de iniciar sesión 64
Figura 4-18 Página de descarga de Industrial Edge App Publisher en IE Hub 65
Figura 4-19 Captura del dialogo instalación apps IEM 66
Figura 4-20 Captura de mensaje del proceso de instalación del IEM....66
Figura 4-21 Captura IEM....67
Figura 5-1 Arquitectura de la plataforma Siemens Industrial Edge. Fuente: Siemens Industry Image Database 4.11....68
Figura 5-2 Orden de instalación en el PC de desarrollo de las herramientas SW necesarias. Fuente: Siemens Industry Image Database 4.11 69
Figura 5-3 Descarga de la ISO de instalación de Ubuntu 20.04.4 LTS. Fuente: www.ubuntu.com....70
Figura 5-4 Crear una máquina virtual de desarrollo de Ubuntu - Captura 1 71
Figura 5-5 Crear una máquina virtual de desarrollo de Ubuntu - Captura 2 71

Figura 5-6 Características de la MV creada para desarrollo. 72
Figura 5-7 Pantalla de inicio de sesión de Ubuntu después de una instalación exitosa 72
Figura 5-8 Marketplace aplicaciones Ubuntu 73
Figura 5-9 Extensión Python en Visual Studio Code 74
Figura 5-10 Extensión Jupyter en Visual Studio Code 74
Figura 5-11 Extensión Docker en Visual Studio Code 74
Figura 5-12 Interfaz web de aplicaciones IE. Fuente: Siemens Industry Image Database 4.11 76
Figura 5-13 Extensión Flask para Visual Studio Code 77
Figura 5-14 Extensión Jinja para Visual Studio Code 78
Figura 5-15 Estructura de una App Industrial Edge. Fuente: Siemens Industry Image Database 4.11 81
Figura 5-16 Ejemplo de aplicación en Flask 82
Figura 5-17 Volcado de tolas versiones de Docker disponibles 84
Figura 5-18 Prueba de Hello World después de la instalación de Docker 85
Figura 5-19 Descripción general del software gráfico en Ubuntu con IEAP 87
Figura 6-1 Flujo de trabajo de desarrollo de aplicaciones para Unified Comfort Panel. Fuente: Siemens Industry Image Database 4.11 89
Figura 6-2 Arquitectura de la plataforma Siemens Industrial Edge. Fuente: Siemens Industry Image Database 4.11 90
Figura 6-3 Estructura de un programa listo para “Dockerización” 91
Figura 6-4 Uso de imágenes base descargadas desde Docker Hub. Fuente: Siemens Industry Image Database 4.11 92
Figura 6-5 Ejemplo típico de fichero Dockerfile 93
Figura 6-6 Imagenes para Node. Fuente: www.hub.docker.com 93
Figura 6-7 Instrucciones para el Dockerfile. Fuente: Siemens Industry Image Database 4.11 94
Figura 6-8 Página de inicio de aplicación ejemplo IE 98
Figura 6-9 Resumen de los comandos básicos de Docker. Fuente: Siemens Industry Image Database 4.11 99
Figura 6-10 Recursos disponibles en una UCP. Fuente: Siemens Industry Image Database 4.11 99
Figura 6-11 Ejemplo: reducir el tamaño total de las imágenes usando una imagen base optimizada. Fuente: Siemens Industry Image Database 4.11 100
Figura 6-12. Openpipe Unified Comfort Panels. Fuente: Siemens Industry Image Database 4.11 102
Figura 6-13 Figura 6 12. Openpipe Unified Comfort Panels. Fuente: 102
Figura 6-14 Sintaxis OpenPipe. Fuente SIMATIC HMI WinCC Unified Open Pipe 103
Figura 6-15 Kit Data Service para desarrolladores Industrial Edge. Fuente: Kit SDK Data Service 108
Figura 6-16 Publicación de imagenes con IEAP. Fuente: Siemens Industry Image Database 4.11 109
Figura 6-17 Captura IEAP 109
Figura 6-18 Captura IEAP 110
Figura 6-19 Captura IEAP 110
Figura 6-20 Captura IEAP 111
Figura 6-21 Captura IEAP 111
Figura 6-22 Captura IEAP 112
Figura 6-23 Captura IEAP 113
Figura 6-24 Captura IEAP: Parámetros de registro de logs 113

Figura 6-25 Captura IEAP: Parámetros de almacenamiento 114
Figura 6-26 Captura IEAP: Parámetros de red 114
Figura 6-27 Captura IEAP: Parámetros de sistema 115
Figura 6-28 Captura IEAP 115
Figura 6-29 Captura IEAP 116
Figura 6-30 Captura IEAP 116
Figura 6-31 Captura IEM: Estadisticas 117
Figura 6-32 Captura IEM: Vista desarrollador 118
Figura 6-33 Captura IEM: Vista desarrollador detallada 118
Figura 6-34 Captura IEM: Ajustes 119
Figura 6-35 Captura IEM: Ajustes de sistema 119
Figura 6-36 Captura IEM: Habilitación logs de sistema 120
Figura 6-37 Contenido Log file (1/2) Fuente: Siemens Industry Image Database 4.11 120
Figura 6-38 Contenido Log File (2/2). Fuente: Siemens Industry Image Database 4.11 120
Figura 6-39 Captura IEM: Aplicaciones 121
Figura 6-40 Logs de aplicaciones 122
Figura 6-41 Ejemplo de archivo Docker Compose con controlador de registro. Fuente: Siemens Industry Image Database 4.11 123
Figura 6-42 Asistente del IE App Publisher (IEAP) 124
Figura 6-43 Formatos de registros soportados por Docker. Fuente: Docker Inc. 124
Figura 6-44 Ejemplo se archivo Docker Compose 125
Figura 6-45 Mensaje JSON 125
Figura 7-1 MS Azure Cognitive services. Fuente: www.enmilocalfunciona.io 126
Figura 7-2 Esquema del protocolo entre nuestras aplicaciones y la API de Cognitive Services. Fuente: Azure 127
Figura 7-3 Captura Azure Cognitive Services 128
Figura 7-4 Captura Azure Cognitive Services 128
Figura 7-5 Captura Azure Cognitive Services 129
Figura 7-6 Captura Azure Cognitive Services 129
Figura 7-7 Descripción de la imagen analizada. Fuente: Azure 131
Figura 7-8 Resultado del análisis de características de una imagen de prueba 132
Figura 7-9 Tabla de límites de Custom Vision en función del modelo se suscripción. Fuente: Azure 133
Figura 7-10 Captura Azure Custom Vision 133
Figura 7-11Captura Azure Custom Vision 134
Figura 7-12 Captura Azure Custom Vision 135
Figura 7-13 Captura Azure Custom Vision 135
Figura 7-14 Captura Azure Custom Vision 136
Figura 7-15 Captura Azure Custom Vision 136
Figura 7-16 Captura Azure Custom Vision 137
Figura 7-17 Captura Azure Custom Vision 137
Figura 7-18 Captura Azure Custom Vision 138
Figura 7-19 Captura Azure Custom Vision 138
Figura 7-20 Captura Azure Custom Vision 138
Figura 7-21 Captura Azure Custom Vision: Rendimiento predicción 139
Figura 7-22 Captura Azure Custom Vision: métricas 139
Figura 7-23 Captura Azure Custom Vision. Rendimiento segunda iteración 140
Figura 7-24 Captura Azure Custom Vision; Métricas 140
Figura 7-25 Captura Azure Custom Vision 141
Figura 7-26 Captura Azure Custom Vision 142

Figura 7-27 Captura Azure Custom Vision 142
Figura 8-1 Credenciales de conexión Azure 145
Figura 8-2 Captura frame de video 146
Figura 8-3 Carga de la imagen 146
Figura 8-4 Conexión a Azure Custom Vision 146
Figura 8-5 Obtenemos los resultados de la predicción 146
Figura 8-6 Diccionario para guardar el resultado de la predicción 147
Figura 8-7 Asignación del resultado de la predicción al diccionario 147
Figura 8-8 Conectamos el Pipe del HMI 147
Figura 8-9 Escritura del disccionario en las variables del HMI vía OpenPipe 147
Figura 8-10 Página de inicio aplicación 149
Figura 8-11 Página principal aplicaión 150
Figura 8-12 Página informativa 150
Figura 8-13 Mensaje de error 150
Figura 8-14 Listado de librerías a instalar 151
Figura 8-15 Error al instalar opencv con Alpine 152
Figura 8-16 Tamaño de la imagen de Docker de nuestra aplicación 152
Figura 8-17 Docker file aplicación visión 152
Figura 8-18 Publicación de la app con IEAP. Paso 1 153
Figura 8-19 Publicación de la app con IEAP. Paso 2 153
Figura 8-20 Publicación de la app con IEAP. Paso 3 153
Figura 8-21 Publicación de la app con IEAP. Paso 4 154
Figura 8-22 Publicación de la app con IEAP. Paso 5 154
Figura 8-23 Publicación de la app con IEAP. Paso 6 155
Figura 8-24 Publicación de la app con IEAP. Paso 7 155
Figura 8-25 Publicación de la app con IEAP. Paso 8 156
Figura 8-26 Publicación de la app con IEAP. Paso 9 156
Figura 8-27 Publicación de la app con IEAP. Paso 10 156
Figura 8-28 Publicación de la app con IEAP. Paso 11 157
Figura 8-29 Publicación de la app con IEAP. Paso 12 157
Figura 8-30 Publicación de la app con IEAP. Paso 13 158
Figura 8-31 Publicación de la app con IEAP. Paso 14 158
Figura 8-32 Publicación de la app con IEAP. Paso 15 158
Figura 8-33 Publicación de la app con IEAP. Paso 16 159
Figura 8-34 Publicación de la app con IEAP. Paso 17 159
Figura 8-35 Publicación de la app con IEAP. Paso 18 160
Figura 8-36 Publicación de la app con IEAP. Paso 19 160
Figura 8-37 Publicación de la app con IEAP. Paso 20 161
Figura 8-38 Panel de control de un UCP MTP700 161
Figura 8-39 Gestión de Apps 162
Figura 8-40 Habilitamos trabajar con Edge en la UCP MTP700 162
Figura 8-41 Nos registramos con nuestro usuario y contraseña 163
Figura 8-42 Instalamos la App 163
Figura 8-43 Dialogo de instalación de App en IEM 164
Figura 8-44 Progreso instalación de App 164
Figura 8-45 Instalación de App finalizada 165
Figura 8-46 Resultado final donde se puede ver nuestra App lista para funcionar. 165
Figura 8-47 Pantalla principal de la aplicación 166
Figura 8-48 Pantalla modo automático de la cinta 167
Figura 8-49 Control manual 167

Figura 8-50 Pantalla volcado resultado detección de objetos.....168
Figura 9-1 Figura 0 1 Ilustración 1 Ecosistema SIMATIC para puesta en marcha virtual. Fuente: SIMATIC Machine Simulator Virtual commissioning of machines Getting Started 169
Figura 9-2 Coste errores en cada fase del proyecto. Fuente Siemens Industry Image Database 4.11.....170
Figura 9-3Entorno de desarrollo Siemens para puesta en marcha virtual. Fuente: Siemens Industry Image Database 4.11.....170
Figura 9-4 Mundo real. Fuente: SIMATIC Machine Simulator Virtual commissioning of machines Getting Started.....171
Figura 9-5 Mundo virtua. Fuente: SIMATIC Machine Simulator Virtual commissioning of machines Getting Started.....172
Figura 9-6 Mapeado de entradas externas a NX MCD.....173
Figura 9-7 Mapeado de salidas externas a NX MCD 174
Figura 9-8 Figura 1 21 2.- Gemelo digital de la célula de producción.....175
Figura 9-9 PLCSim Advanced 4.0.....175
Figura 9-10 Mapeo de señales del exterior con NX MCD 176
Figura 9-11 Play de la simulación NX MCD.....176
Figura 9-12 Simulación en marcha.....177
Figura 10-1 Crear un clasificador de imágenes con Custom Vision. Fuente Azure. 180
Figura 10-2 Comparación de YOLO con otros detectores. Fuente YOLO Darknet 180

Lista de Acrónimos y Abreviaturas

AGV ..... Automatic Guided Vehicle
API REST ..... Application Programming Interface. Representational State Transfer
App ..... Aplicación
AWS ..... Amazon Web Services.
CLI ..... Command Line Interface.
CNN ..... Convolutional Neural Networks
CPU ..... Central Processing Unit
DL ..... Deep Learning.
DNS ..... Domain Name System
GB ..... Gigabyte
GPU ..... Unidad de procesamiento gráfico
HMI ..... Interfaz Hombre-Máquina
HW ..... Hardware
IA ..... Inteligencia Artificial.
IE ..... Industrial Edge
IEAP ..... Industrial Edge App Publisher
IEM ..... Industrial Edge Manager
IoT ..... Internet Of Things
IP ..... Internet Protocol
ISO ..... International Organization for Standardization
JPEG ..... Joint Photographic Experts Group
JSON ..... JavaScript Object Notation
MAE ..... Mean Absolute Error
ML ..... Machine Learning.
NAT ..... Network Address Translation
NTP ..... Network Time Protocol
PIP ..... Paquetes de Instalación PIP
PLC ..... Controlador lógico programable
PNG ..... Portable Network Graphics
RAE ..... Relative Absolute Error
RSE ..... Relative Squared Error
RMSE ..... Root Mean Squared Error
S3 ..... Simple Storage Service
SDK ..... Software Development Kits
SO ..... Sistema Operativo
SSH ..... Secure Shell
TCP ..... Transmission Control Protocol
TFG ..... Trabajo Final de Grado.
UAH ..... Universidad de Alcalá.
UCP ..... Unified Comfort Panel
URL ..... Uniform Resource Locator
USB ..... Universal Serial Bus
VM ..... Virtual Machine

Resumen

El principal objetivo de este TFG ha sido el estudio de las aplicaciones que la computación perimetral (Edge Computing) tiene en el ámbito industrial. En el desarrollo se ha implementado una aplicación de visión artificial basada en los servicios ofrecidos por Azure Custom Vision y el motor de ejecución de aplicaciones industriales perimetrales (Edge) de Siemens, Industrial Edge.

El hecho de elegir para nuestra aplicación de Industrial Edge un ejemplo con Visión Artificial viene motivado por el hecho de que las aplicaciones de visión artificial son fundamentales en multitud de procesos industriales hoy en día, desde la detección de anomalías, el guiado de AGVs o el empaquetado con robots. En el presente trabajo se han aplicado los últimos avances en visión artificial en combinación con las tecnologías de contenedores y computación perimetral (Edge Computing) de Siemens Industrial Edge. De esta manera se ha intentado resolver, de un modo lo más eficiente posible, tareas de reconocimiento y posicionado de materiales para robots con visión artificial en entornos industriales.

Abstract

The main objective of this TFG has been the study of the applications that perimeter computing (Edge Computing) has in the industrial field, for which an artificial vision application has been implemented based on the services offered by Azure Custom Vision and the running industrial edge applications from Siemens, Industrial Edge.

The fact of choosing an example with Machine Vision for our Industrial Edge application is motivated by the fact that machine vision applications are essential in many industrial processes today, like anomaly detection, AGV guidance or the picking-pack with robots. In the present work, the latest advances in artificial vision have been applied in combination with container technologies and edge computing (Edge Computing) from Siemens Industrial Edge. In this way it has been possible to solve, in the most efficient way possible, tasks of recognition and positioning of materials for robots with artificial vision in industrial environments.

1. Introducción

1.1 Descripción del TFG

En este primer capítulo vamos a tratar dos tecnologías que van a ser la base de este TFG. Por un lado, Docker, tecnología que nos permite empaquetar aplicaciones en contenedores aislados y que es la base de la computación perimetral con Siemens Industrial Edge como veremos más adelante y por otro las técnicas de visión artificial basadas en aprendizaje profundo o "Deep Learning" que es la tecnología que vamos a utilizar para nuestro caso de uso a través de la herramienta Custom Vision de Azure.

A continuación, y después de un breve repaso por las opciones de mercado actuales en computación perimetral industrial más importantes (Azure IoT Hub, AWS Greengrass…) vamos a centrarnos en la solución de Siemens, Industrial Edge para repasar en profundidad sus características y posibilidades.

Por último, y una vez estudiadas las posibilidades de las tecnologías en las que se basa nuestra aplicación y en su implementación práctica, se ha implementado un caso de uso basado en una aplicación de Siemens Industrial Edge que utiliza de una red neuronal entrenada a través de los servicios ofrecidos en la nube por Custom Vision de Azure.
Una vez entrenado el modelo de DL y alojado en la nube este es capaz de realizar una tarea de visión artificial industrial identificando diferentes productos y sus características.
Recopilando todos los elementos de SW necesarios para el reconocimiento automático se ha creado una imagen de Docker para posteriormente generar una App que se ejecute a través de Industrial Edge y comunique mediante API con el servicio de Custom Visión ya entrenado.

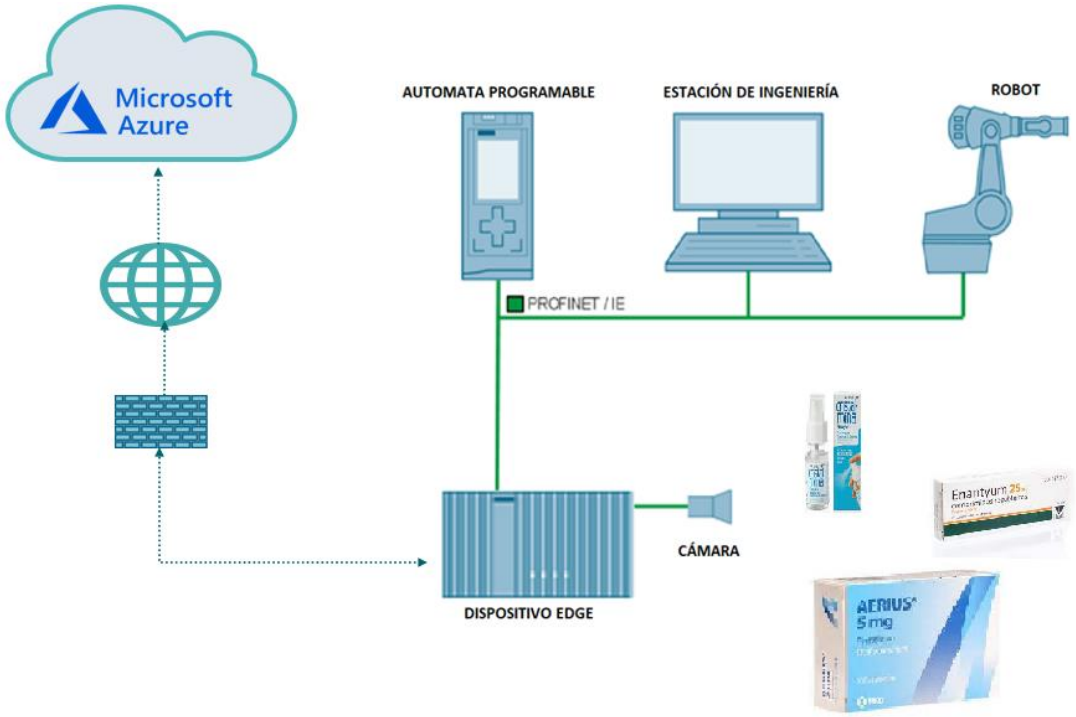


Figura 1-1 Esquema del caso de uso Industrial Edge

En la figura 1-1 se puede ver la arquitectura del sistema implementado para probar la viabilidad de la aplicación.
Por una parte, una estación de ingeniería desde donde desarrollaremos, simularemos y cargaremos la aplicación final al dispositivo "Edge".
Este dispositivo Edge está conectada a una cámara que capta las imágenes de los productos para su procesado por el modelo de Custom Vision de Azure que nos devuelve la información de los objetos de la imagen la cual que transfiere al autómata programable marca Siemens S7-1500 para que este pueda comandar un brazo robótico. (SCARA)
Para simulación y prueba de la aplicación se ha implementado el "Gemelo Digital" de una línea de producción. Utilizando PLCSim Advanced (Simulación de autómatas programables marca Siemens S7-1500) y Siemens NX MCD (Mechatronics Concept Designer)

Para tener una visión más clara de cómo se estructura este TFG podemos dividirlo tres grandes bloques temáticos:

- **Tecnología base:** En el capítulo 1 y 2 se hace un repaso por las tecnologías más importantes que dan soporte a este trabajo como son la tecnología de microservicios basada en Docker ejecutada en dispositivos perimetrales (Edge) la Visión Artificial basada en Machine y Deep Learning y el estado del arte actual de la tecnología de computación perimetral aplicadas a entornos industriales con los principales proveedores de este tipo de tecnologías: Azure, AWS y Siemens.
  En este bloque los conceptos de contenedores y microservicios serán fundamentales manejarlos ya que son la base de las aplicaciones para Industrial Edge que se verán más adelante.

- **Siemens Industrial Edge:** En este bloque que abarca los capítulos 3, 4, 5 y 6 se hace un estudio en profundidad del ecosistema de Siemens Industrial Edge para computación perimetral tratado desde la arquitectura del sistema, la forma de implementar el entorno de desarrollo para aplicaciones y la gestión de estas en dispositivos Industrial Edge.

- **Caso de uso:** Para ejemplificar la potencialidad de la computación perimetral en entornos industriales en los capítulos 7, 8 y 9 se implementa un modelo de reconocimiento de cajas de medicamentos mediante las herramientas de aprendizaje profundo que nos aporta Azure Custom Vision. Este modelo que esté alojado en la nube se conecta vía API con una aplicación de Siemens Industrial Edge que se encarga de seleccionar las imágenes que capta una cámara sobre una cinta transportadora y enviarlas a la nube para su procesamiento el cual nos devuelve los objetos detectados en la imagen y la posición. Nuestra aplicación se encarga de hacer de interfaz entre la cámara y el modelo de reconocimiento de objetos de la nube y entre esta y el controlador de la línea un controlador industrial (PLC) Siemens modelo S7-1500 que es el encargado de comandar un brazo robótico para la selección y recogida de las cajas.

1.2 Industrial Edge Computing

Si consideramos una planta de fabricación moderna. En ella existen numerosos dispositivos y equipos que generan datos susceptibles de ser monitorizados (por ejemplo, estados del PLC, temperatura de motores, velocidad de las cintas transportadoras, ángulo de los brazos del robot, etc.)
Los datos que se generaban en planta se usaban para garantizar el funcionamiento estándar, pero en los últimos años se están empezando a utilizar para aplicaciones más avanzadas de análisis. (por ejemplo, aprendizaje automático para predecir cuándo fallará el equipo, visión artificial para determinar si los trabajadores permanecen en áreas seguras…).
Estas aplicaciones más avanzadas desarrolladas en base a los datos generados en planta se venían desarrollando con aplicaciones más novedosas en la nube, pero la tendencia reciente ha sido implementar estas aplicaciones en el perímetro local.
Esto viene determinado por la demanda de los clientes industriales para implementar aplicaciones en las propias instalaciones, compartiendo la misma infraestructura por razones de escalabilidad y ciberseguridad. De esta manera puede llegar haber una gran cantidad de dispositivos diferentes que generen datos, gran número de aplicaciones que operen con esos datos y un gran número de usuarios finales que accedan a las aplicaciones y dispositivos (con diferentes roles y permisos)

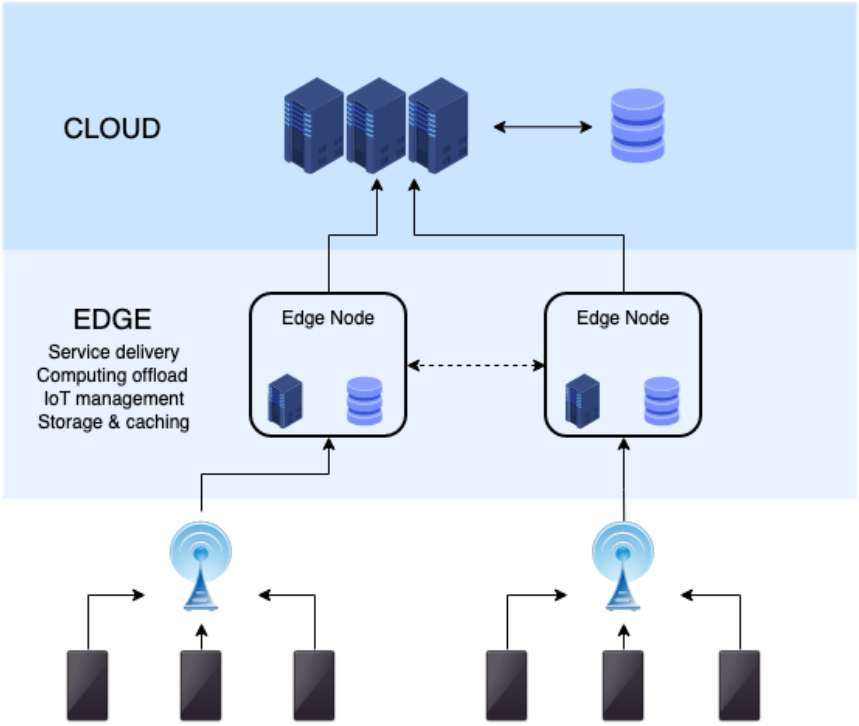


Figura 1-2 Infraestructura computación perimetral. Fuente: Wikipedia[1]

Esto se puede complicar rápidamente por lo que es necesario implementar un sistema para que sea manejable. Un sistema para abordar este desafío se considerará una solución de

[1] https://es.wikipedia.org/wiki/Edge_computing

computación perimetral o “Edge Computing", que es especialmente necesaria a medida que este escenario se amplía a múltiples instalaciones.
Se podría definir la tecnología de computación perimetral, Edge Computing o en adelante Edge como un paradigma de la computación distribuida que acerca el almacenamiento de los datos y su procesamiento a la ubicación en la que se necesita para mejorar, entre otros, la latencia y el ancho de banda que se obtendría con arquitecturas basadas exclusivamente en computación en la nube.[2]

La computación perimetral o Edge Computing viene a dar respuesta a esa necesidad de la mano de las últimas novedades en tecnologías de virtualización y contenedores para permitir un despliegue de soluciones escalable, seguro y eficiente.
Las tecnologías Edge se están extendiendo en muchos ámbitos, y son ya imprescindibles actualmente en ámbitos como la conducción autónoma, la realidad aumentada o la realidad virtual.

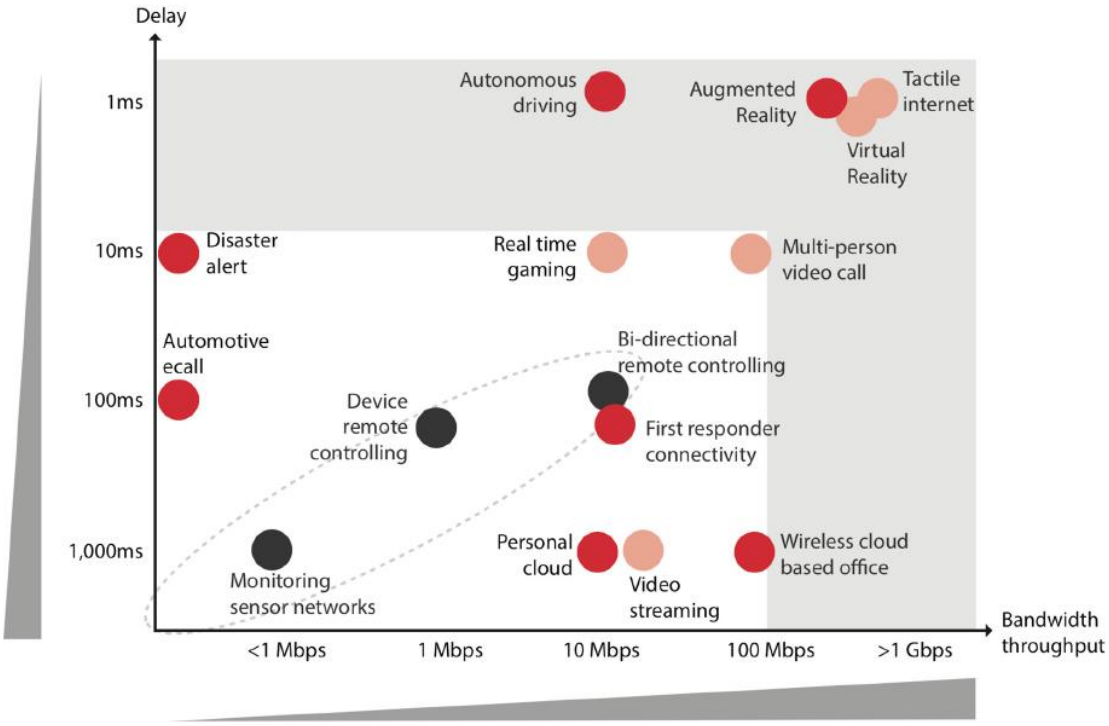


Figura 1-3 Casos de uso en función de los requisitos de latencia. Fuente: GSMA Intelligence[3]

Edge Computing, requisitos básicos:
A continuación, vamos a detallar las características que debería tener cualquier solución del mercado para ser un proveedor de soluciones de Edge Computing industriales. Mas adelante en el capítulo 2 haremos un repaso del estado del arte en Industrial Edge detallado los principales proveedores actuales de soluciones.

[2] https://es.wikipedia.org/wiki/Computaci%C3%B3n_distribuida
[3] GSMA Intelligence: Understanding 5G: Perspectives on future technological advancements in mobile (2014)

- Motor de orquestación de aplicaciones (por ejemplo, Docker Compose ó Kubernetes), un sistema para alojar, instalar, configurar, ejecutar y detener aplicaciones. Por lo general, no hay una especificación definida sobre cómo se debe empaquetar y configurar una aplicación, pero debemos tender a la mayor estandarización posible.

- Software de tiempo de ejecución (por ejemplo, Azure Edge IoT Core, AWS Greengrass) que administra el sistema en el que se ejecuta. Por lo general, configura la identidad del sistema, controla el motor de orquestación y se comunica con el sistema de gestión central.

- Sistema de administración centralizado (p. ej.: panel de AWS Greengrass, Azure IoT Hub) una ubicación única para que los usuarios finales vean todos los sistemas que tienen el motor de tiempo de ejecución instalado (runtime) y los administren a escala. Puede haber uno o más sistemas de gestión central implementados y cada uno tendrá la propiedad o el control de varios dispositivos con tiempos de ejecución instalados que se le asignen.

1.3 Docker

Docker fue liberado como código abierto en marzo de 2013. El 13 de abril de 2015, el proyecto tenía más de 20 700 estrellas de GitHub (haciéndolo uno de los proyectos con más estrellas de GitHub, en 20ª posición), más de 4 700 bifurcaciones (forks), y casi 900 colaboradores. [4]

1.3.1. Historia:

Docker es una plataforma abierta para desarrollar, compartir y ejecutar aplicaciones. Docker permite independizar las aplicaciones de la infraestructura hardware de manera que se pueda desarrollar software rápidamente. Con Docker, es posible administrar una infraestructura de la misma manera que se administran aplicaciones. Al aprovechar las metodologías de Docker para compartir, probar e implementar el código rápidamente, podemos reducir significativamente la demora entre escribir el código y ejecutarlo en producción.
Entre los partners de Docker se cuentas empresas como Microsoft Azure, aws o RedHat[5]

1.3.2. Plataforma Docker:

Docker brinda la capacidad de empaquetar y ejecutar una aplicación en un entorno aislado llamado contenedor. El aislamiento y la seguridad permiten ejecutar muchos contenedores simultáneamente en un host determinado. Los contenedores son ligeros y contienen todo lo necesario para ejecutar la aplicación, por lo que no es necesario depender de lo que está instalado actualmente en el host. Se puede compartir contenedores fácilmente mientras se trabaja y asegurarse de que todas las personas con las que comparten obtengan el mismo contenedor que funciona de la misma manera independientemente del hardware del Host o del resto de software instalado en ese Host.

[4] https://es.wikipedia.org/wiki/Docker_(software)

[5] https://www.docker.com/partners

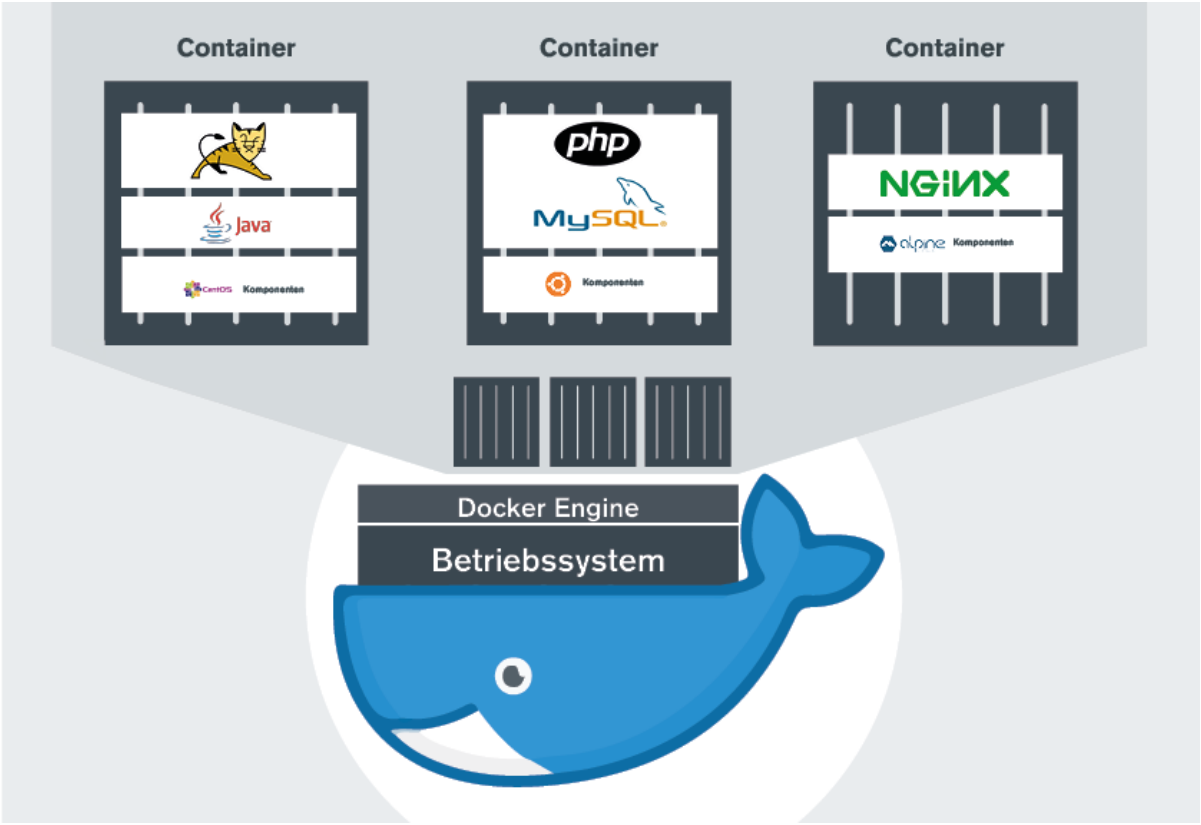


Figura 1-4 Entorno de desarrollo y motor Docker. Fuente: Siemens Industry Image Database 4.11[6]

- ✓ Características:

El contenedor es la unidad base para probar y distribuir aplicaciones
Una vez probado, el despliegue y orquestación de los contenedores en el entorno de producción se puede hacer tanto si este es dispositivo edge local, un proveedor de nube o un híbrido de los dos. Esto es debido a que la plataforma basada en contenedores de Docker permite cargas de trabajo altamente portátiles. Los contenedores Docker pueden ejecutarse en el ordenador portátil del desarrollador, en máquinas físicas o virtuales en un centro de datos, en proveedores de la nube o en una combinación de entornos.
La portabilidad y la naturaleza ligera de Docker también facilitan la administración dinámica de cargas de trabajo, ampliando o eliminando aplicaciones y servicios según lo dicten las necesidades comerciales, casi en tiempo real.
La ligereza y la rapidez de Docker proporciona una alternativa viable y rentable a las máquinas virtuales basadas en hipervisor, por lo que se pueden utilizar la mayor parte de las capacidades de cómputo para producción. Docker es perfecto para entornos de alta densidad y para implementaciones pequeñas y medianas en las que los recursos hardware son escasos.

- ✓ Tecnología base de Docker

Docker está escrito en el lenguaje de programación Go y aprovecha varias características del kernel de Linux para ofrecer su funcionalidad. Docker usa una tecnología llamada espacios de nombres para proporcionar el espacio de trabajo aislado llamado contenedor. Cuando ejecuta un contenedor, Docker crea un conjunto de espacios de nombres para ese contenedor.

[6] https://www.automation.siemens.com/bilddb/index.aspx?lang=en

Estos espacios de nombres proporcionan una capa de aislamiento. Cada aspecto de un contenedor se ejecuta en un espacio de nombres independiente y su acceso está limitado a ese espacio de nombres.

1.3.3. Arquitectura Docker:

Docker utiliza una arquitectura cliente-servidor. El cliente de Docker se comunica con el demonio de Docker, que hace el trabajo pesado de crear, ejecutar y distribuir los contenedores de Docker. El cliente y el demonio de Docker pueden ejecutarse en el mismo sistema, o puede conectar un cliente de Docker a un demonio de Docker remoto. El cliente Docker y el "Daemon" se comunican mediante una API REST, a través de sockets UNIX o una interfaz de red. Otro cliente de Docker es Docker Compose, que le permite trabajar con aplicaciones que consisten en un conjunto de contenedores.

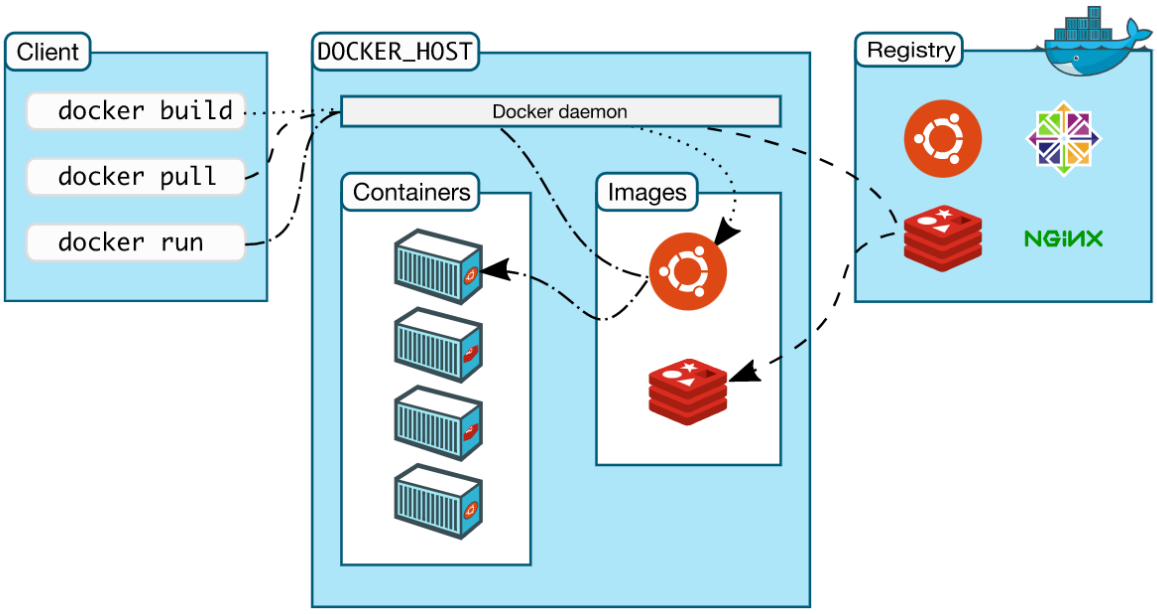


Figura 1-5 Esquema de la arquitectura Docker. Fuente: Docker Inc[7].

- ✓ El demonio Docker

El demonio de Docker (dockerd) escucha las solicitudes de la API de Docker y administra los objetos de Docker, como imágenes, contenedores, redes y volúmenes. Un demonio también puede comunicarse con otros demonios para administrar los servicios de Docker.

- ✓ El cliente Docker

El cliente de Docker (docker) es la forma principal en que los usuarios de Docker interactúan con Docker. Cuando se usa comandos como docker run, el cliente envía estos comandos a dockerd, que los lleva a cabo. El comando docker utiliza la API de Docker. El cliente de Docker puede comunicarse con más de un daemon.

- ✓ Escritorio

[7] https://docs.docker.com/get-started/overview/

Docker Desktop es una aplicación fácil de instalar para entorno Mac o Windows que permite crear y compartir microservicios y aplicaciones en contenedores. Docker Desktop incluye el demonio de Docker (dockerd), el cliente de Docker (docker), Docker Compose, Docker Content Trust, Kubernetes y Credential Helper.

- ✓ Registros

Un registro de Docker almacena imágenes de Docker. Docker Hub es un registro público que cualquiera puede usar. Docker está configurado para buscar imágenes en Docker Hub de manera predeterminada. Incluso puede ejecutar su propio registro privado.
Cuando se utilizan los comandos docker pull o docker run, las imágenes requeridas se extraen de su registro configurado. Cuando usa el comando docker push, la imagen se envía al registro configurado.

Objetos:
Cuando se usa Docker, se están creando y usando imágenes, contenedores, redes, volúmenes, complementos y otros objetos. Vamos a ver un resumen de estos objetos:

- ✓ Imágenes

Una imagen es una plantilla de solo lectura con instrucciones para crear un contenedor Docker. A menudo, una imagen se basa en otra imagen, con alguna personalización adicional. Por ejemplo, puede crear una imagen que se base en la imagen de ubuntu, pero instala el servidor web Apache y su aplicación, así como los detalles de configuración necesarios para ejecutar su aplicación.
Puede crear imágenes propias o se pueden usar solo las creadas por otros y publicadas en un registro. Para construir una imagen propia, creamos un Dockerfile con una sintaxis simple para definir los pasos necesarios para crear la imagen y ejecutarla. Cada instrucción en un Dockerfile crea una capa en la imagen. Cuando cambia el Dockerfile y se reconstruye la imagen, solo se reconstruyen las capas que han cambiado. Esto es parte de lo que hace que las imágenes sean tan livianas, pequeñas y rápidas, en comparación con otras tecnologías de virtualización.

- ✓ Contenedores

Un contenedor es una instancia ejecutable de una imagen. Se pueden crear, iniciar, detener, mover o eliminar un contenedor mediante la API o la CLI de Docker. Se puede conectar un contenedor a una o más redes, adjuntarle almacenamiento o incluso crear una nueva imagen basada en su estado actual.
De forma predeterminada, un contenedor está relativamente bien aislado de otros contenedores y de la máquina host. Dicho aislamiento es totalmente controlable.
Un contenedor se define por su imagen, así como por cualquier opción de configuración que se le proporcione al crearlo o iniciarlo. Cuando se elimina un contenedor, desaparece cualquier cambio en su estado que no esté almacenado en el almacenamiento persistente.

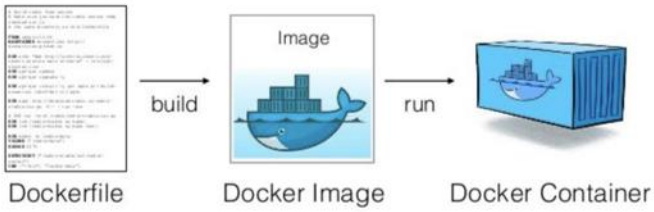


Figura 1-6 Concepto Docker – Dockerfile, Imágenes y Contenedores. Fuente: Platformer[8]

1.3.4. Comparativa entre contenedores y máquinas virtuales:

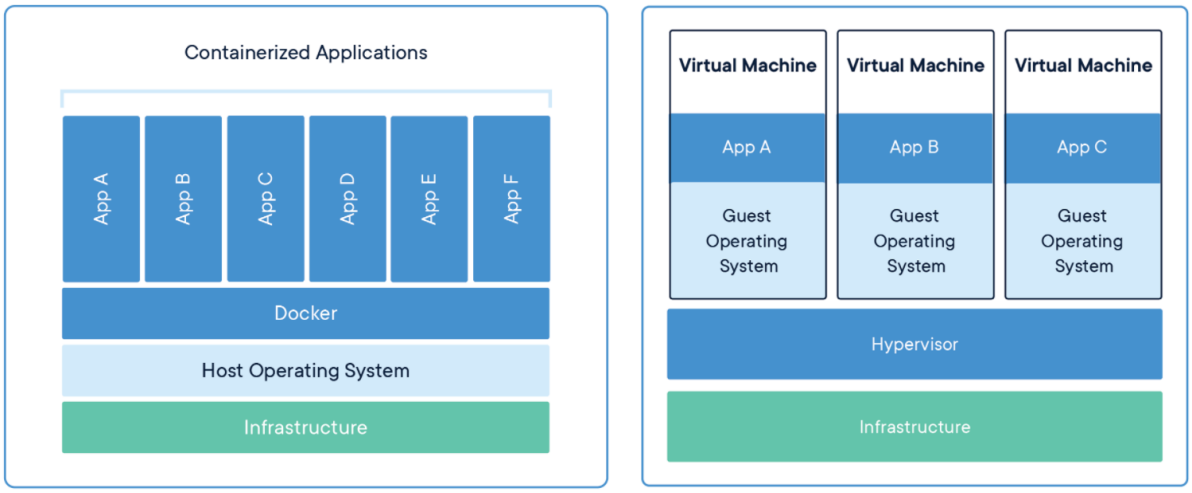


Figura 1-7 Arquitectura Docker izquierda y de virtualización a la derecha. Fuente: Docker Inc.[9]

- ✓ Contenedores

Los contenedores son una abstracción de la capa de la aplicación que empaqueta el código y las dependencias juntos. Varios contenedores pueden ejecutarse en la misma máquina y compartir el kernel del sistema operativo con otros contenedores, cada uno ejecutándose como procesos aislados en el espacio del usuario. Los contenedores ocupan menos espacio que las máquinas virtuales (las imágenes de contenedores suelen tener un tamaño de decenas de MB), pudiendo manejar más aplicaciones y requiriendo menos máquinas virtuales y sistemas operativos.

- ✓ Máquinas Virtuales

Las máquinas virtuales (VM) son una abstracción del hardware físico que convierte un servidor en muchos servidores. El hipervisor permite que varias máquinas virtuales se ejecuten en una sola máquina. Cada máquina virtual incluye una copia completa de un sistema operativo, la aplicación, los archivos binarios y las bibliotecas necesarios, lo que ocupa decenas de GB.

1.3.5. Docker Hub:

Se trata de un repositorio proporcionado por Docker Inc. para compartir y guardar imágenes de contenedores de forma pública y/o privada. Nos permite extraer y enviar imágenes de

[8] https://medium.com/platformer-blog/practical-guide-on-writing-a-dockerfile-for-your-application-89376f88b3b5

[9] https://www.docker.com/resources/what-container/

Docker hacia y desde Docker Hub. Podemos tratar esto como un GitHub, donde buscamos y enviamos nuestro código fuente, pero en el caso de Docker Hub, descargamos o publicamos las imágenes de nuestro contenedor. Es un repositorio en línea basado en la nube que almacena ambos tipos de repositorios, es decir, el repositorio público y el repositorio privado. Los repositorios públicos son accesibles para todos.
En cuanto a las limitaciones hay que comentar que con la cuenta gratuita solo es posible utilizar este repositorio de forma privada.
En los repositorios públicos se encuentran distintas versiones de la misma imagen. Por ejemplo, imágenes con distintas versiones de Ubuntu para diferentes arquitecturas hardware: arm, 386, amd…
.

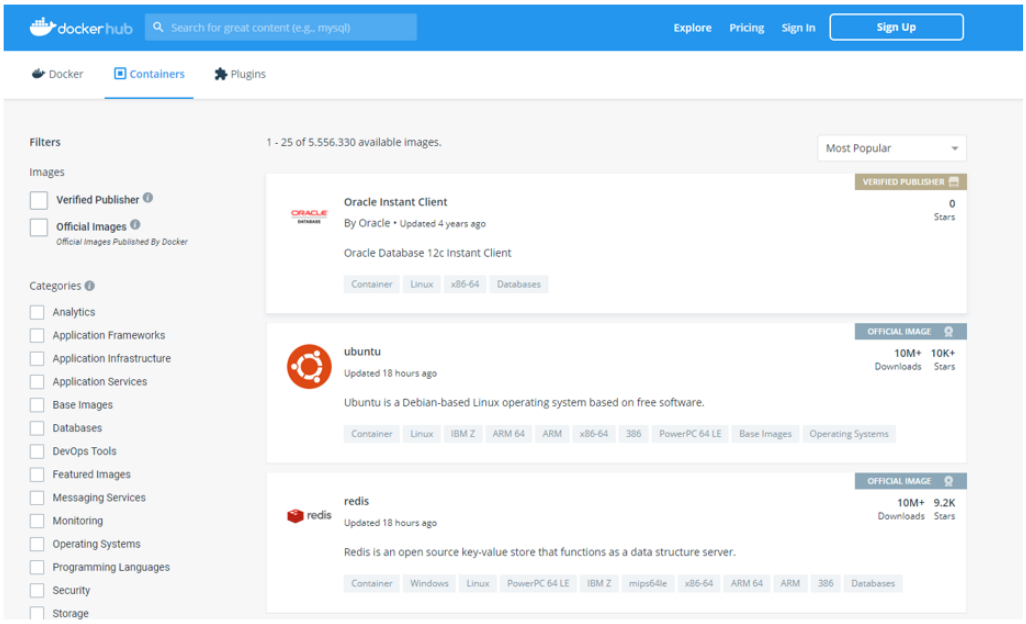


Figura 1-8. Docker Hub

1.4 Visión Artificial

1.4.1. Introducción:

La visión artificial es la ciencia que estudia los procesos de adquisición, procesado, análisis e interpretación de la información procedentes de imágenes 2D de un mundo 3D.
El tratamiento de imágenes en la industria mediante aplicaciones de visión artificial es cada vez más importante en los últimos años, principalmente debido a las crecientes demandas sobre la calidad del producto y el seguimiento de la calidad, pero no solo ahí, otras aplicaciones como el guiado de robot lo ha convertido en una tecnología clave en los procesos de fabricación.
El procesamiento de imágenes funciona con métodos de medición óptica. Estos tienen la ventaja de que trabajan sin contacto, en toda el área, de forma no destructiva y más confiablemente que los humanos.
Los sectores de aplicación actuales incluyen ingeniería mecánica, construcción naval, aeronáutica construcción, automoción, microelectrónica, farmacéutica, logística, así como tecnología de tráfico y seguridad.
Para el procesamiento de las imágenes se utilizan diferentes técnicas de procesado, en el presente TFG nos centraremos en las técnicas de inteligencia artificial aplicadas al

procesamiento de imágenes, más en concreto a las técnicas de “Machine Learning” y “Deep Learning.”

Un ejemplo de utilización exhaustiva de las técnicas de visión artificial al proceso industrial podría ser el de una línea de embotellado, siendo en ese caso las aplicaciones posibles las siguientes:

1. Testeo de materiales
Las botellas entrantes se revisan en busca de grietas e irregularidades. Las botellas sin daños quedan entonces disponibles para su posterior procesamiento. Las botellas dañadas se retiran.
2. Comprobación de integridad
A continuación, se comprueba que las botellas llenas estén completas. Las no completamente llenas se retiran.
3. Control de calidad
Todas las botellas que no cumplan con el estándar (cantidad de llenado, impresión errónea de la etiqueta, sin corcho.) se retiran.
4. Guía del robot/identificación de la posición
Para que el robot pueda agarrar la botella adecuada, se necesita una detección de posición, para determinar las coordenadas x, y, z. A continuación, las botellas se colocan en la caja.
5. Reconocimiento de fuentes/caracteres
Antes de que el palé con cajas salga de la producción, el código de barras o código QR se Comprueba para, entre otros, el seguimiento del envío o el alta en el inventario de almacen.

1.4.2. Etapas de un sistema de Visión Artificial:

Adquisición de imágenes:
Para la adquisición de las imágenes a utilizar en procesos industriales hay tres aspectos clave a considerar a la hora de plantear la implementación de un sistema de Visión Artificial.

- Características de los objetos:

Cada objeto cuenta con características particulares de geometría, superficie, posición, opacidad, reflectividad y otras que hay que considerar a la hora de implementar un sistema de visión artificial. En la mayoría de los casos un requisito previo será el estudio del tipo de objetos a examinar para buscar la solución de Visión Artificial más adecuada para cada caso.

- Iluminación:

La correcta selección de la iluminación nos va a permitir resaltar las características que más definitorias sean de los objetos a examinar y de esa manera facilitar la labor de procesamiento posterior de las imágenes captadas.
Las técnicas de iluminación más frecuentes son[10]:

- Directa: La cámara recibe la luz reflejada del objeto
- Lateral: La cámara recibe la luz reflejada del objet, el foco emisor de lus se coloca lateralmente al mismo y sirve para resaltardetalles.
- Estroboscopica: Luz pulsada para objetos a gran velocidad
- Por campo oscuro: Utilizada para resaltar defectos superficiales.

[10] https://infaimon.com/enciclopedia-de-la-vision/

- Por contraste: A contraluz para mediciones precisas y detección de grietas, rayas…
- Domo: No produce sombras y está indicada para iluminar superficies especulares.

- Lentes/cámara:

Además de elegir el sistema de iluminación y la lente correctos, es de vital importancia importante determinar el sensor de imagen correcto.
Los sensores de imagen utilizan componentes para convertir la luz, en señales eléctricas. Estos son luego discretizados en el tiempo, cuantificados y finalmente almacenados. La carga eléctrica de los sensores de imagen depende de la intensidad de la luz incidente y del tiempo de exposición.
Los sensores de imagen más utilizados son los sensores CMOS y CCD. Por un lado, un componente acoplado por carga (sensor CCD), por el otro lado, un sensor con tecnología de semiconductores de óxido de metal (sensor CMOS). Ambos sensores son sensores de área. El tamaño de la imagen está determinado por el número de filas y columnas. El sensor de imagen tiene así la función de convertir la luz que se produce en una imagen digital óptima.

Preprocesamiento de imágenes

El preprocesamiento de imágenes sirve para preparar las imágenes y resaltar las características más definitorias de las mismas o que más nos interesen de cara a facilitar la extracción de características en la fase de procesado.
Se utilizan en esta etapa técnicas para aumentar el contraste de las imágenes, eliminar errores, suprimir perturbaciones (componentes de ruido) o enfatizar ciertas estructuras

Segmentación de imagen

A través de la segmentación de imágenes es posible aislar objetos de una imagen.
Para la segmentación se utilizan métodos que encuentran objetos en la imagen de forma estática (áreas de trabajo) o de forma dinámica.
La región de interés (ROI) es una subárea de la imagen que se está examinando.
Gracias a la delimitación de áreas ROI el tiempo de computación se reduce, ya que se deben examinar menos píxeles.
Para estas tareas se pueden suelen utilizan algunos de los siguientes algoritmos:

- Segmentación binaria:

La imagen se divide por un valor de umbral en una imagen binaria [0,1]. El objeto se forma sobre la base de los dos estados. Los histogramas son los mas adecuados para definir valores de umbral en este procedimiento.

- Segmentación basada en modelos:

El sistema busca directamente un objeto predefinido. Una medida de similitud encuentra el objeto de la imagen.

- Segmentación basada en bordes:

Esto calcula los bordes o contornos de un objeto. Esto hace posible identificar objetos de forma única en función de su representación de contorno.

Extracción de características

La extracción de características determina las propiedades individuales de los objetos. Las características pueden ser, por ejemplo, circunferencia, área, posición, orientación, dimensiones lineales, formas geométricas, distancias o similitudes. A partir de esta información, un clasificador puede luego determinar mejor qué clase es.

Clasificación
En la clasificación, un objeto se asigna únicamente a una clase sobre la base de sus características. En el caso de un conjunto de caracteres, una letra como "A", por ejemplo, se puede describir sin ambigüedades.
En los últimos años, el aprendizaje automático o machine learning ha permitido programar mejores y más eficientes clasificadores que con los métodos anteriores.
Ya no es tan necesario preparar las imágenes con complejas etapas de preprocesado y procesado siendo posible "entrenar" el algoritmo en base al tipo de imágenes que nos vayamos a encontrar.
Mediante el aprendizaje de imágenes, por ejemplo, es posible con un algoritmo de aprendizaje profundo (Deep Learning) definir clases a partir de un conjunto de imágenes existente.

Actuación
En función de la decisión sobre la corrección de una clase existente, se pueden tomar decisiones en el proceso de fabricación. Por ejemplo, las piezas podrían clasificarse en piezas "buenas" y "malas" y así mejorar la calidad del producto fabricado.

Figura 1-9 Etapas de un sistema de Visión Artificial. Fuente: Industrial image processing on SIMATIC IPCs[11]

1.5. Técnicas de Machine Learning para Visión Artificial

1.5.1. Machine Learning

"El Machine Learning es una rama de la Inteligencia Artificial que utilizan métodos computacionales para "aprender" información directamente de los datos sin depender de una ecuación predeterminada como modelo."[12]

[11] https://support.industry.siemens.com/cs/es/es/view/109766012

[12] Apuntes Visión Artificial, Grado Ingeniería en Electrónica de Comunicaciones UAH. Tema 5, pág. 3.

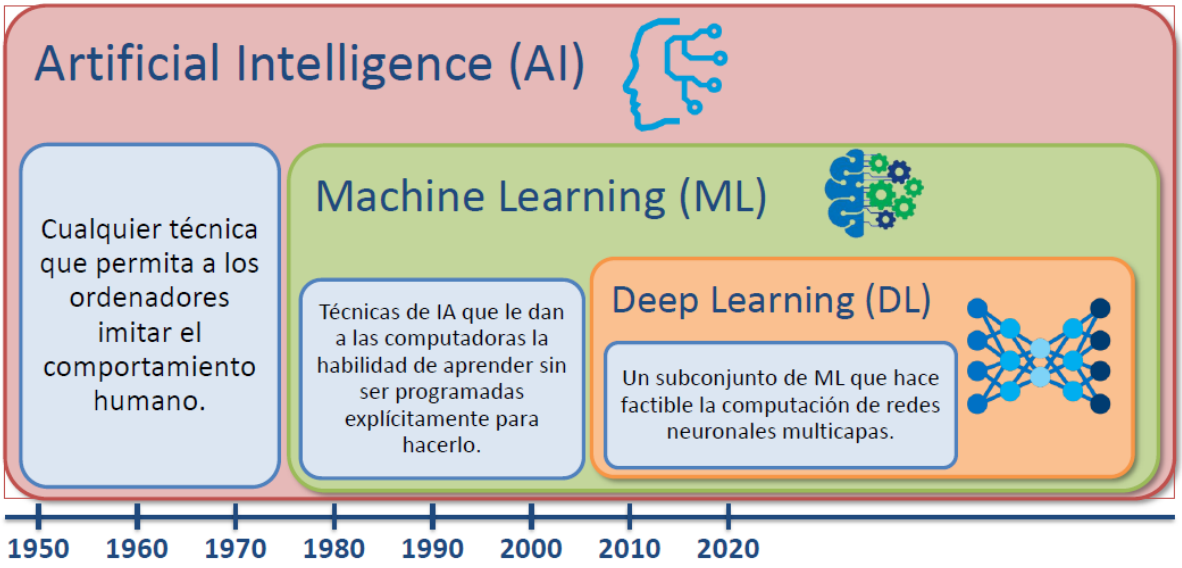


Figura 1-10 Tipos de inteligencia artificial. Fuente: UAH.

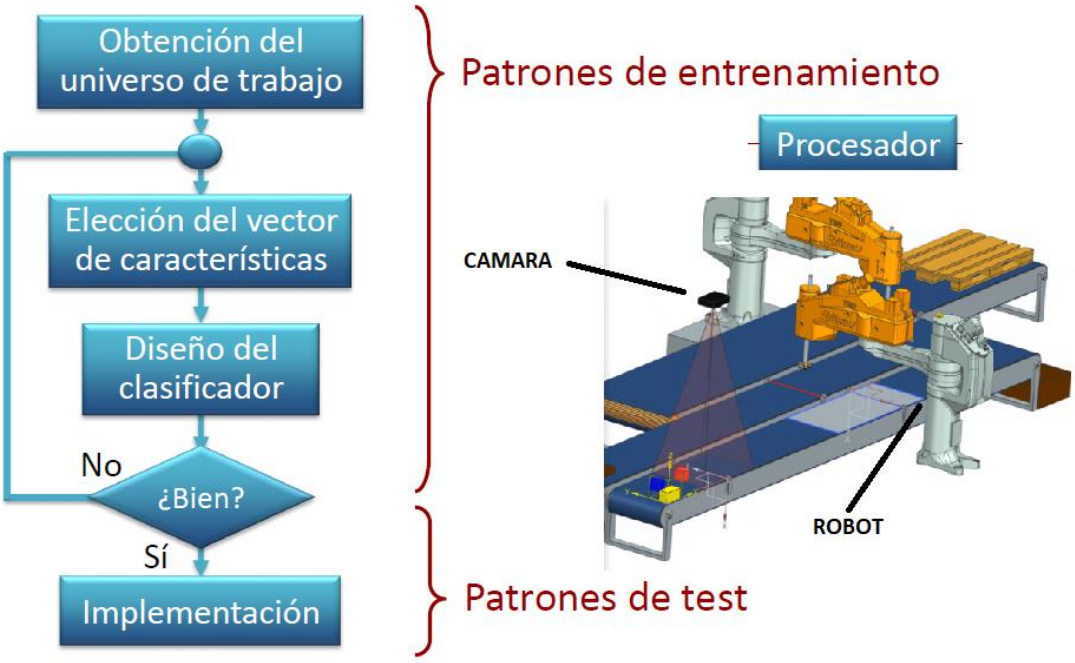


Figura 1-11 Diseño de un sistema de reconocimiento de objetos

En muchas ocasiones, los problemas a resolver por un sistema de Visión Artificial pueden ser difíciles de modelar por un sistema basado en algoritmos clásicos de Visión y se hace necesario recurrir a las técnicas de Machine Learning que pueden mejorar automáticamente su rendimiento en base al entrenamiento previo y nos permiten abordar tareas como serían la regresión, clasificación y clustering,

La primera tarea, por parte del programador será preparar los datos de entrada al sistema para alimentarlo con datos que de verdad aporten información útil y sean característicos del sistema. Las computadoras aprenden a modelar la relación entre estos conjuntos de datos de características de entrada y el resultado que intentan predecir;

El tipo más común de ML es la máquina supervisada, en el entrenamiento se utilizan datos etiquetados, con información sobre las salidas correctas que se corresponden con los datos de entrada, y el algoritmo encuentra la relación entre ellos.
Por otra parte, el entrenamiento no supervisado se va a utilizar cuando lo que necesitemos obtener es una clasificación de los datos de entrada en base a las características de los mismos.

Para utilizar técnicas de Machine Learning lo primero que deberemos hacer será utilizar un método de aprendizaje determinado en función del tipo del problema al que nos enfrentemos y dentro de este método un algoritmo que se ajuste a nuestras necesidades. Esto no es sencillo y acerca a las técnicas de Machine Learning más al "arte" que a las ciencias exactas.
A continuación, vamos a comentar los tres métodos más usualmente utilizados, cada uno de los cuales dispone de diferentes posibilidades de algoritmos para su implementación práctica[13].

Aprendizaje / Tipo de datos	Aprendizaje no supervisado	Aprendizaje supervisado
Discreto	Agrupamiento (Clustering)	Clasificación o categorización
Continuo	Reducción de la dimensionalidad	Regresión

- **Regresión:**

El aprendizaje automático usa algoritmos para identificar patrones dentro de los datos, y esos patrones luego se usan para crear un modelo de datos que puede hacer predicciones. La regresión es una forma de aprendizaje automático que se utiliza para predecir una etiqueta numérica en función de las características de un elemento. Un ejemplo sería la predicción de la probabilidad de sufrir una afección cardiaca en función del historial clínico del paciente, su peso, sexo, si es fumador…. En este caso, las características son los datos del paciente y la probabilidad de sufrir cáncer es la etiqueta. La regresión es un ejemplo de técnica de aprendizaje automático supervisado en la que entrena un modelo utilizando datos. Los datos incluyen tanto las características como valores conocidos para la etiqueta, de modo que el modelo aprende a ajustar las combinaciones de características a la etiqueta. Luego, una vez completado el entrenamiento, puede usar el modelo entrenado para predecir etiquetas para nuevos elementos de los que se desconoce el valor de la etiqueta.

Las métricas para medir el desempeño de nuestro clasificador de regresión serían las siguientes:
• Mean Absolute Error (MAE): La diferencia promedio entre los valores pronosticados y los valores verdaderos. Este valor se basa en las mismas unidades que la etiqueta. Cuanto más bajo es este valor, mejor predice el modelo.
• Root Mean Squared Error (RMSE): la raíz cuadrada de la diferencia cuadrática media entre los valores predichos y verdaderos. El resultado es una métrica basada en la misma unidad que

[13] https://docs.microsoft.com/en-us/azure/machine-learning/algorithm-cheat-sheet

la etiqueta. Cuando se compara con el MAE (arriba), una mayor diferencia indica una mayor variación en los errores individuales (por ejemplo, algunos errores son muy pequeños, mientras que otros son grandes).
• Relative Squared Error (RSE): una métrica relativa entre 0 y 1 basada en el cuadrado de las diferencias entre los valores predichos y verdaderos. Cuanto más cerca de 0 esté esta métrica, mejor será el rendimiento del modelo. Debido a que esta métrica es relativa, se puede usar para comparar modelos donde las etiquetas están en diferentes unidades.
• Relative Absolute Error (RAE): una métrica relativa entre 0 y 1 basada en las diferencias absolutas entre los valores predichos y reales. Cuanto más cerca de 0 esté esta métrica, mejor será el rendimiento del modelo. Al igual que RSE, esta métrica se puede usar para comparar modelos donde las etiquetas están en diferentes unidades.
• Coefficient of Determination (R2): esta métrica se conoce más comúnmente como R-Squared y resume qué parte de la varianza entre los valores predichos y verdaderos se explica por el modelo. Cuanto más cerca de 1 sea este valor, mejor será el rendimiento del modelo[14]

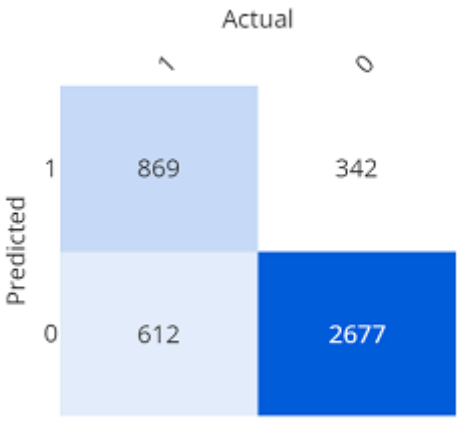
- **Clasificación:**

La clasificación es un ejemplo de una técnica de aprendizaje automático supervisado en la que entrena un modelo utilizando datos que incluyen tanto las características como los valores conocidos de la etiqueta, se utiliza para predecir a qué categoría o clase pertenece un elemento. Un ejemplo sería clasificar préstamos hipotecarios de riesgo o sin riesgo en función del historial bancario del sujeto. En este caso, los datos del prestatario son las características y la etiqueta es una clasificación de cero o uno que representa riesgo o no riesgo. Con la clasificación, el modelo aprende a ajustar las combinaciones en función a la etiqueta. En el ejemplo se recopilan datos bancarios como la nómina, otros prestamos, sector de actividad, etc., para cada sujeto. Estas son las características representadas por x en nuestra ecuación. También sabemos si el préstamo fue impagado o no. Cuando se completa el entrenamiento, se puede usar el modelo entrenado para predecir etiquetas para elementos nuevos para los que se desconoce la etiqueta. Cuando se toman datos de un préstamo nuevo, el modelo entrenado usa esas características para predecir la probabilidad de que el préstamo sea impagado o no.

Las métricas para medir el desempeño de un clasificador serían las siguientes:

• Matriz de confusión del modelo: es una tabulación de los recuentos de valores previstos y reales para cada clase posible. Para un modelo de clasificación binaria como este, en el que se predice uno de dos valores posibles, la matriz de confusión es una cuadrícula de 2x2 que muestra los recuentos de valores predichos y reales para las clases 0 y 1, similar a esto:

[14] Evaluate Model component. Azure Machine Learning Documentation. https://docs.microsoft.com/en-us/azure/machine-learning/component-reference/evaluate-model



La matriz de confusión muestra los casos en los que tanto los valores predichos como los reales fueron 1 (conocidos como verdaderos positivos) en la parte superior izquierda, y los casos en los que tanto los valores predichos como los reales fueron 0 (verdaderos negativos) en la parte inferior derecha. Las otras celdas muestran casos en los que los valores predichos y reales difieren (falsos positivos y falsos negativos).
Para un modelo de clasificación de clases múltiples (donde hay más de dos clases posibles), se usa el mismo enfoque para tabular cada combinación posible de recuentos de valores reales y predichos, por lo que un modelo con tres clases posibles daría como resultado una matriz de 3x3 con una línea diagonal de celdas donde coinciden las etiquetas predichas y reales.

• Accuracy: la proporción de predicciones correctas (verdaderos positivos + verdaderos negativos) con respecto al número total de predicciones. Diciéndolo de otra manera, ¿proporción de predicciones de impagados acertó el modelo?
• Precision: La fracción de casos positivos identificados correctamente (el número de verdaderos positivos dividido por el número de verdaderos positivos más los falsos positivos). En otras palabras, de todos los préstamos que el modelo predijo que serían impagados, ¿cuántos son impagados realmente?
• Recall: La fracción de los casos clasificados como positivos que son realmente positivos (el número de verdaderos positivos dividido por el número de verdaderos positivos más los falsos negativos). En otras palabras, de todos los préstamos que realmente fueron impagados, ¿cuántos identificó el modelo?
• Puntuación F1: una métrica general que combina esencialmente precisión y recuperación.

De estas métricas, accuracy es la más intuitiva. Sin embargo, hay que tener cuidado al usarla como una medida de lo bien funciona un modelo. Supongamos que sólo el 5% de los prestamos son impagados. Se podría crear un modelo que siempre prediga 0 y tendría una precisión del 95 % Por esta razón, la mayoría de los científicos de datos utilizan otras métricas como la "Precision" y el "Recall" para evaluar el rendimiento del modelo de clasificación.

- **Clustering:**

El clustering es una forma de aprendizaje automático que se utiliza para agrupar elementos similares en clústeres según sus características. Un ejemplo sería la segmentación de clientes en función de sus gustos musicales. Con esta técnica, se entrena un modelo con datos que incluyen tanto las características como los valores conocidos de la etiqueta. La agrupación en clústeres es un ejemplo de aprendizaje automático no supervisado en el que entrena un modelo para separar elementos en clústeres basándose únicamente en sus características. No existe un valor de clúster o una etiqueta previamente conocidos a partir de los cuales entrenar el modelo.

Las métricas para medir el desempeño de un modelo de "clustering" serían las siguientes:

• Distancia promedio a otro centro: Esto indica como de cerca, en promedio, está cada punto en el grupo de los centroides de todos los demás grupos.
• Distancia promedio al centro del cluster/grupo: Esto indica como de cerca, en promedio, cada punto del cluster está del centroide del cluster
• Número de puntos: el número de puntos asignados al clúster.
• Distancia máxima al centro del cluster/grupo: el máximo de las distancias entre cada punto y el centroide del grupo de ese punto. Si este número es alto, el grupo puede estar muy disperso.

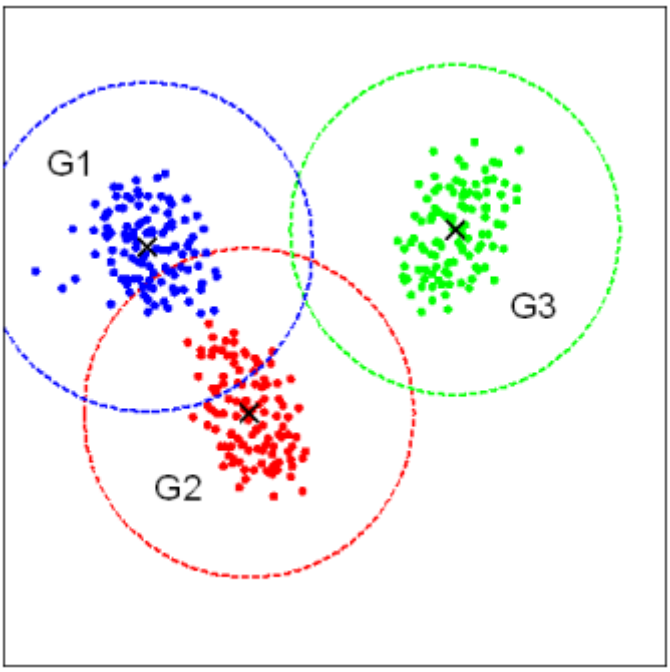


Figura 1-12 Ejemplo clustering utilizando k-means. Fuente: UAH[15]

Diseño de clasificadores:
El diseño de clasificadores de ML consta de tres etapas en las que se necesita utilizar un conjunto de datos etiquetados.

- Entrenamiento: Se utiliza aproximadamente un 80% de los datos etiquetados disponibles (datos de entrenamiento)
- Validación: Si las métricas obtenidas no alcanzan un valor umbral se reentrena aplicando diferentes métodos, pero siempre con el conjunto de datos de entrenamiento.
- Test: Para el testeo de nuestro modelo se utiliza el conjunto de datos que habíamos reservado, usualmente el 20% de los datos etiquetados disponibles.

[15] Apuntes Visión Artificial, Grado Ingeniería en Electrónica de Comunicaciones UAH.

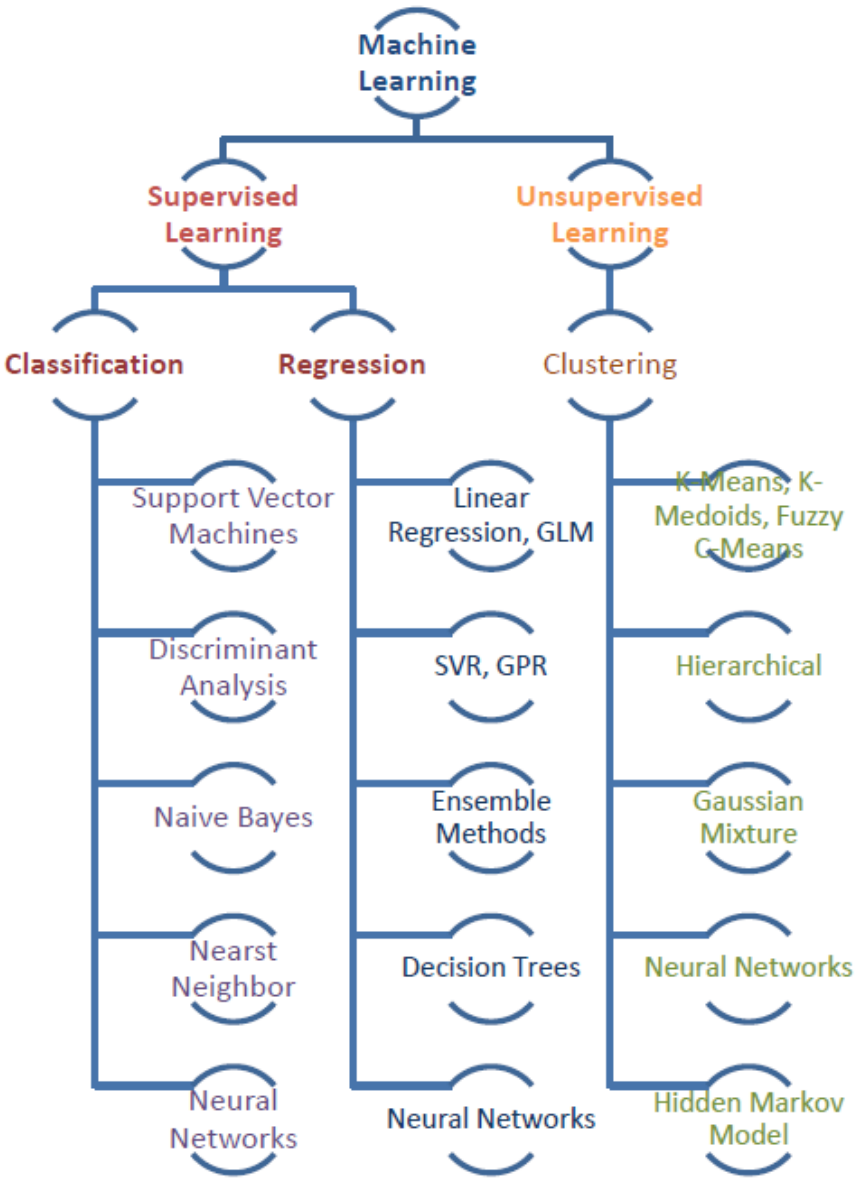


Figura 1-13 Algoritmos de Machine Learning. Fuente: UAH.[16]

1.5.2. Deep Learning

El Deep Learning o DL se podría definir como un subconjunto del aprendizaje automático que se basa en redes neuronales artificiales. El proceso de aprendizaje es profundo porque la estructura de las redes neuronales artificiales consta de múltiples capas de entrada, salida y ocultas. Cada capa contiene unidades que transforman los datos de entrada en información que la siguiente capa puede usar para una determinada tarea predictiva. Gracias a esta estructura, una máquina puede aprender a través de su propio procesamiento de datos.[17]

[16] Apuntes Visión Artificial, Grado Ingeniería en Electrónica de Comunicaciones UAH. Tema 5, pág. 3.
[17] https://docs.microsoft.com/en-us/azure/machine-learning/concept-deep-learning-vs-machine-learning

1.5.3. Técnicas de Deep learning vs. Machine learning

En el aprendizaje automático mediante ML requiere de un exhaustivo conocimiento de los datos a tratar y sus características para seleccionar el modelo y el consiguiente algoritmo que mejor se adapte para la resolución del problema propuesto. Pudiendo ser necesario combinar diferentes algoritmos y fases de procesado de los datos para llegar a obtener un modelo de predicción con la precisión necesaria.
Por contra, con Deep Learning vamos a podernos abstraer de como el algoritmo ejecuta el proceso de predicción simplificando enormemente la tarea de programación de nuestro modelo, pero penalizándonos en cuanto a la cantidad de datos de entrenamiento necesarios y el hardware a utilizar. Por ejemplo, GPU y FPGA que son más eficientes que las CPU tradicionales para los cálculos en una red neuronal para DL.

La mejora continua del rendimiento de las redes neuronales utilizadas para DL (por ejemplo, el entrenamiento más rápido basado en la nube) y el aumento exponencial de la capacidad de cómputo de los procesadores utilizados en la industria están haciendo que poco a poco las técnicas de DL vayan desplazando al ML en aplicaciones de Visión Artificial industrial y sea esta la técnica que más se está desarrollando actualmente por todos los actores implicados en este entorno.
De ahí que este TFG se enfoque en el DL y las herramientas que pone a disposición MS Azure para llevar a cabo la tarea de modelar una aplicación industrial basada en Visión Artificial.

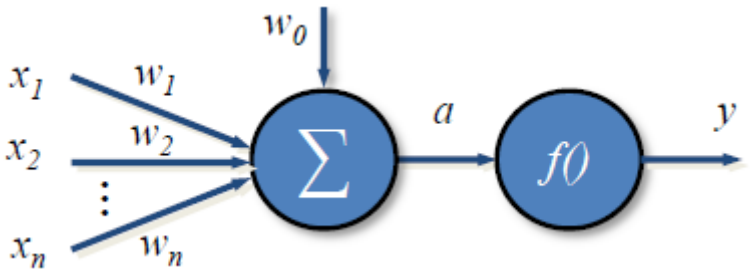
	Machine learning	**Deep learning**
Datos de entrenamiento	Puede usar pequeñas cantidades de datos para hacer predicciones.	Necesita usar grandes cantidades de datos de entrenamiento para hacer predicciones.
Dependencias hardware	Puede trabajar en máquinas de gama baja. No necesita una gran cantidad de potencia computacional.	Depende de máquinas de gama alta. Intrínsecamente hace una gran cantidad de operaciones de multiplicación de matrices. Una GPU puede optimizar de manera eficiente estas operaciones.
Etiquetado	Requiere que los usuarios identifiquen y creen con precisión las características.	Aprende funciones de alto nivel a partir de datos y crea nuevas funciones por sí mismo.
Tipo de aprendizaje	Divide el proceso de aprendizaje en pasos más pequeños. Luego combina los resultados de cada paso en una sola salida.	Se mueve a través del proceso de aprendizaje resolviendo el problema de principio a fin.

Tiempo de ejecución	Toma comparativamente poco tiempo para entrenar, desde unos pocos segundos hasta unas pocas horas.	Por lo general, lleva mucho tiempo entrenarlo porque un algoritmo de aprendizaje profundo involucra muchas capas.
Salida	El resultado suele ser un valor numérico, como una puntuación o una clasificación.	La salida puede tener múltiples formatos, como un texto, o un sonido.

Comparación de técnicas de IA[18]

1.5.4. Redes Neuronales Convolucionales:

Las redes neuronales RNs son una forma práctica de hacer un clasificador por regiones. están formadas por un conjunto de neuronas básicas, que son elementos de cálculo no lineales con varias entradas y una salida, cuya estructura más simple se muestra a continuación:



Las entradas de la red neuronal serán las características de los objetos, x1, ..., xn (con n=número de características). Cada una de las entradas tiene un peso asociado (w1, ..., wn), que multiplica su valor, más un offset o bias w0. la entrada de la neurona (también llamada estado de activación de la neurona) es: a = x1·w1 + x2·w2 + x3·w3 + . . .+ w0
El conocimiento de las RNs se encuentra en los pesos de la red, que son constantes, pueden tomar cualquier valor (positivo o negativo) y se ajustan durante el período de entrenamiento, donde aprenden a clasificar los patrones de entrenamiento.
Las entradas se ponderan y se suman al llegar a la neurona, y ésta, según su función de transferencia (a veces llamada función de activación), dará a la salida un valor determinado.

Las redes neuronales se pueden clasificar atendiendo a:

- **Tipo de entradas:** entradas continuas y entradas discretas.
- **Arquitectura:** Las neuronas normalmente se conectan entre sí formando unidades jerárquicas llamadas capas. Según el número de éstas, podemos clasificarlas en redes de una sola capa (perceptrón, Kohonen), o redes multicapa (backpropagation).
- **Sentido de avance de la información:** redes feed-forward donde la información siempre se transmite desde las entradas hacia las salidas, sin que exista ningún bucle de

[18] https://docs.microsoft.com/en-us/azure/machine-learning/concept-deep-learning-vs-machine-learning

realimentación, y redes feed-back donde existen bucles de realimentación y la salida depende de las entradas actuales y de los estados anteriores de la red.

Redes neuronales multicapa

Para implementar funciones de decisión que reconozcan patrones multiclase, con independencia de que las clases sean o no linealmente separables, se utilizan arquitecturas con numerosas neuronas.

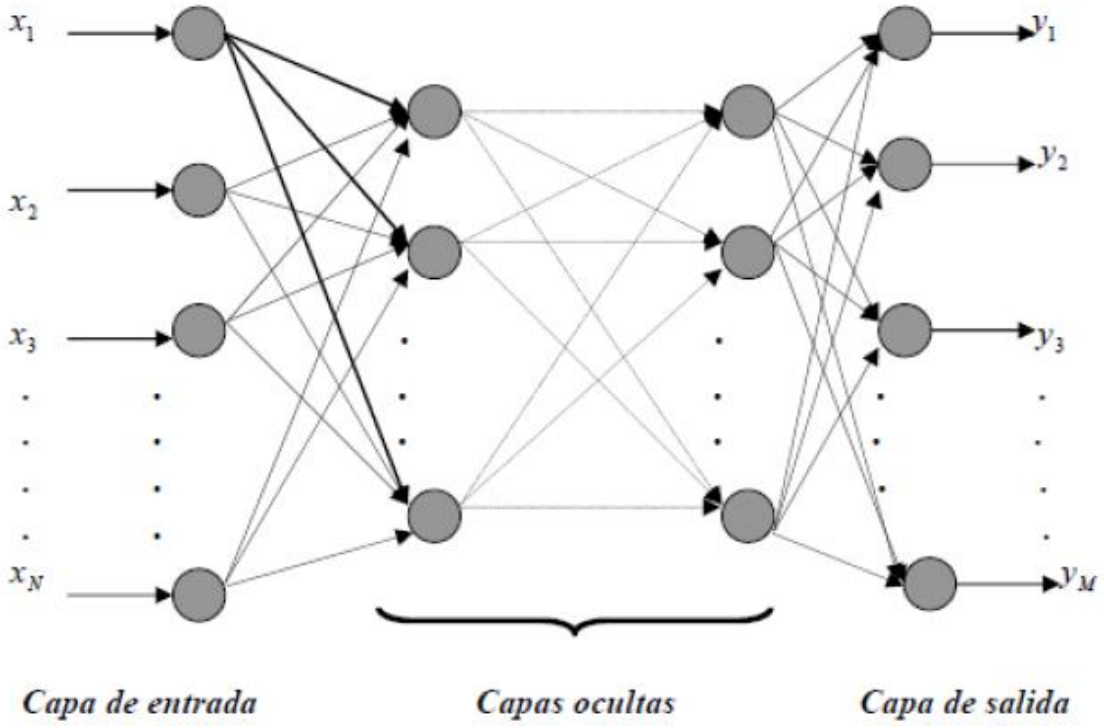


Figura 1-14 Redes neuronales multicapa. Fuente: UAH.[19]

Convolutional Neural Networks CNN

Las CNN son simplemente redes neuronales que utilizan el operador de convolución en al menos una de sus capas. Las CNN son modelos de redes neuronales de avance ó "feedforward" estas son claves para la resolución de problemas de visión artificial. "Feedforward" implica que la información siempre se introduce en una dirección en la red y no hay bucles en la estructura. Las CNN también se han utilizado en otras áreas, como el reconocimiento de voz y el procesamiento del lenguaje natural para ciertas tareas.

Las CNN trabajan bajo la premisa de la invariancia de la translación, para las imágenes, esto se basa en la idea de que un objeto dentro de la imagen es el mismo objeto incluso si se mueve, Esto es importante, ya que la red no tiene que volver a aprender qué es cada objeto en cada posición de la imagen. Esto requiere significativamente menos datos para entrenar y puede

[19] Apuntes Visión Artificial, Grado Ingeniería en Electrónica de Comunicaciones UAH. Tema 5, pág. 3.

generalizarse mejor para aprender a procesar imágenes que si tuviéramos que aprender por separado a reconocer objetos en cada ubicación, como sería necesario en un perceptrón multicapa.

Por ejemplo, si queremos que el modelo sea capaz de aprender a identificar qué es un gato, sin importar dónde se encuentre el gato en la imagen, comparte las mismas características de las que debe aprender el modelo: cómo identificar el pelaje, las orejas de gato, cola, y así sucesivamente.

En las CNN, la imagen de entrada se alimenta a través de lo que a menudo se denomina filtro o kernel, que actúa como un detector de características en la red. Estos detectores de características tratan de aprender aspectos como bordes, formas o patrones dentro de la imagen y los resultados de esta operación de convolución se guardan en lo que se denomina una imagen convolucionada o un mapa de características. La profundidad de una capa convolucional en una red neuronal corresponde al número de filtros utilizados en esa capa.
Se utiliza una forma de muestreo descendente mediante el uso de capas de agrupación para reducir el tamaño de los datos que pasan y eliminar los aspectos potencialmente redundantes a los que la red en esa etapa ha aprendido a reaccionar.

La parte del extractor automático de funciones de CNN permite que la red aprenda aspectos como los bordes y las formas de la imagen sin tener que programar explícitamente la red para calcular estas funciones, como se hizo con el uso de algoritmos de ML.

Es importante destacar que las CNN aprenden automáticamente los valores de los filtros ("detectores de características") mediante el entrenamiento de la red con grandes cantidades de datos etiquetados utilizando un concepto llamado retro propagación, y continúan mejorando los pesos dentro de la red hasta que se minimiza el error de clasificación. En las primeras capas de las redes, la red suele crear filtros que parecen reconocer aspectos de las imágenes, como bordes, formas básicas y colores. Las capas posteriores aprenden patrones cada vez más complejos hasta que todos estos patrones juntos pueden ayudar a la red a aprender la clasificación de la entrada.
Hay muchas maneras de combinar los componentes básicos de las capas convolucionales, las capas de agrupación y las capas conectadas. También hay muchas formas de entrenar y formular la red, y gran parte de la investigación actual se centra en cómo diseñar las capas, las conexiones y en aspectos como la profundidad.

1.5.5. Flujo de trabajo en Deep Learning:

Una vez que se identifica la tarea a resolver mediante DL, un flujo de trabajo típico de aprendizaje profundo incluirá lo siguiente:

1. Identificar conjuntos de datos relevantes.
Una de las mayores dificultades a la hora de implementar un modelo predictivo con DL es encontrar conjuntos de datos relevantes que se puedan usar para entrenar los modelos de aprendizaje profundo para un escenario específico.
Además, el conjunto de datos debe etiquetarse. Por ejemplo, para entrenar una CNN para identificar el tipo de caja de medicamentos, un conjunto de datos que consta de imágenes de cajas de medicamentos con etiquetas que indican si la imagen es de una marca u otra. Estas

imágenes pueden provenir de catálogos de productos existentes, conjuntos de datos de imágenes públicas (por ejemplo, diversos conjuntos de imágenes de ImageNet, CIFAR-10, Deep Fashion) y extraídas de varios sitios web.
Para aumentar el conjunto de datos de entrenamiento y validación inicial, los científicos de datos a menudo usan un motor de búsqueda para realizar una búsqueda de imágenes de una clase específica (por ejemplo, medicamentos), donde el propietario de la imagen ha etiquetado la imagen como de uso gratuito para uso comercial.

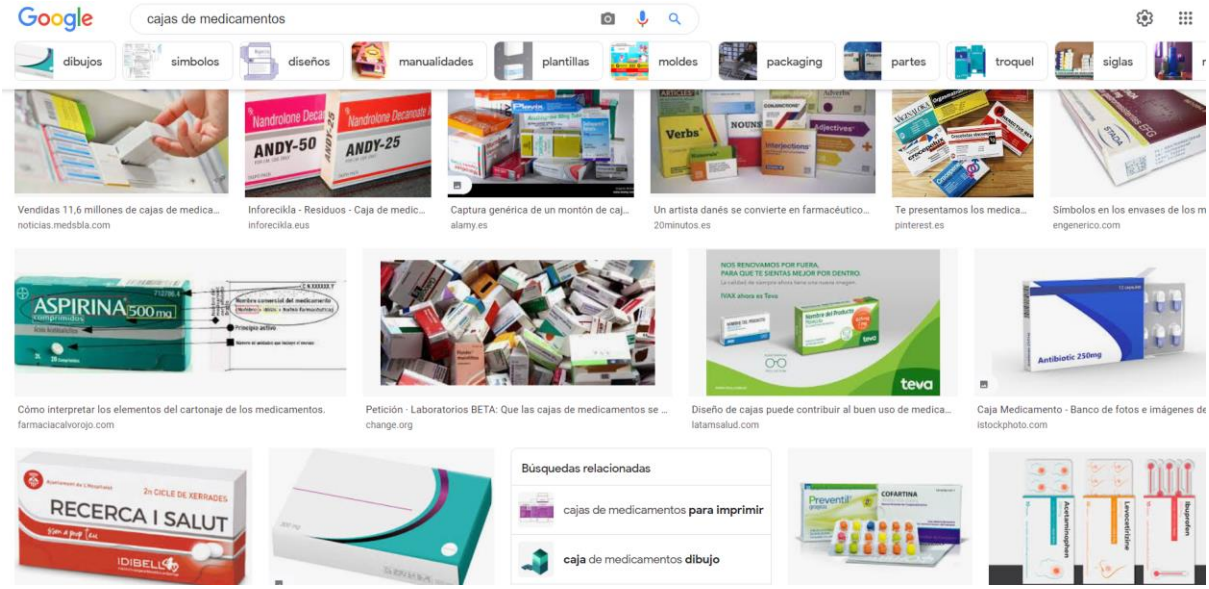


Figura 1-15 Resultados de la búsqueda de imágenes de cajas de medicamentos en Google. Fuente: Google

2. Preprocesar el conjunto de datos.
Una vez que se ha adquirido los conjuntos de datos de imagen relevantes, deberán ser preparados para la etapa de entrenamiento. A menudo, muchos conjuntos de datos de imágenes del mundo real están desequilibrados (comúnmente conocido como el problema de la clase minoritaria).
Esto significa que puede haber más imágenes para una clase específica (Aspirina) y menos imágenes para otra clase (Cristalmina). Para resolver el problema del conjunto de datos desequilibrados, un científico de datos aplica varios trucos para aumentar la cantidad de imágenes en la clase minoritaria o reducir la muestra de las clases más frecuentes hasta lograr la paridad.
Otra técnica de preprocesamiento de uso común es el aumento de datos para ayudar al modelo a generalizar sobre múltiples condiciones, para mejorar su invariancia en aspectos como la rotación, la translación y la escala. Esto incluye aplicar varias transformaciones a la imagen, como escalar, rotar, recortar aleatoriamente la imagen, voltear la imagen, ajustar el brillo y el contraste, y más.

3. Entrenar al modelo.
Una vez que el conjunto de datos ha sido preprocesado y preparado, estamos listos para comenzar a diseñar la arquitectura del modelo de aprendizaje profundo y entrenar el modelo. Los puntos clave que permiten el modelado y el entrenamiento efectivos de los modelos de aprendizaje profundo son, primero, elegir un conjunto de herramientas de aprendizaje profundo y segundo, entrenar usando hardware potente con GPU.
Según el tamaño del conjunto de datos, el modelo se puede entrenar en una máquina local (por ejemplo, un ordenador portátil) o usar la infraestructura disponible en la nube pública, como

Microsoft Azure. Azure proporciona máquinas virtuales (VM) de la serie NC con GPU Nvidia, así como un servicio administrado, llamado Azure Batch AI, que permite aumentar y reducir fácilmente las GPU que se necesita para trabajos de aprendizaje profundo.

4. Comprobar el rendimiento del modelo.

Durante el entrenamiento de la red neuronal profunda, existen varias métricas clave que proporcionarán información sobre la eficiencia del aprendizaje y la calidad de los modelos en cada iteración.

Por ejemplo. Custom Vision Service de Azure utiliza las imágenes que ha enviado para entrenamiento para calcular la precisión y la coincidencia, mediante un proceso denominado validación cruzada de k iteraciones. La precisión y la coincidencia constituyen dos medidas diferentes de la eficacia de un clasificador:

La precisión indica la fracción de las clasificaciones identificadas que fueron correctas. Por ejemplo, si el modelo identificó 100 imágenes como perros y 99 de ellas eran realmente de perros, la precisión sería del 99 %.

La coincidencia indica la fracción de las clasificaciones reales que se identificaron correctamente. Por ejemplo, si había realmente 100 imágenes de manzanas y el modelo identificó 80 como manzanas, la coincidencia sería del 80 %.

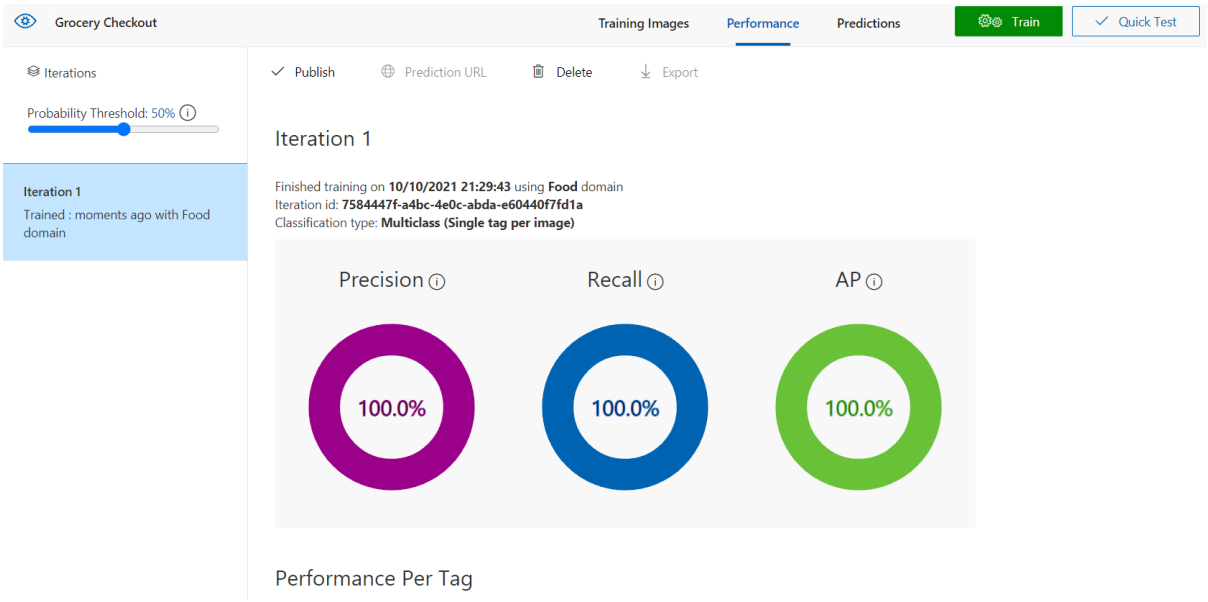


Figura 1-16 Métricas del modelo obtenidas con Custom Visión Portal de Azure.

5. Ajustar el modelo.

Al evaluar el rendimiento en cada iteración, se puede evaluar la calidad del modelo al final de cada una. Hay muchos hiperparámetros que se establecen incluso antes de que comience el proceso de aprendizaje: la tasa de aprendizaje es un hiperparámetro importante que puede tener un impacto significativo en los resultados del modelo. Al graficar la pérdida (eje y) y las iteraciones (eje x), se puede entender si la tasa de aprendizaje se ha establecido correctamente: una buena tasa de aprendizaje conduce a una pérdida menor en un período de tiempo más corto. A menudo, la tasa de aprendizaje se rastrea tanto para el conjunto de datos de entrenamiento como para el de validación. Sin embargo, también es importante asegurarse de que el modelo no se haya sobre ajustado a los datos de entrenamiento. En la práctica, las curvas de tasa de aprendizaje no son uniformes y es posible modificar la tasa de aprendizaje durante el proceso de entrenamiento según sea necesario.

La segunda métrica comúnmente rastreada es la precisión del entrenamiento y la coincidencia. Al graficar la precisión (eje y) y la iteración (eje x), se puede entender si el modelo se ha sobre ajustado al conjunto de datos de entrenamiento. Si las curvas de precisión de entrenamiento y validación están cerca una de la otra, entonces se ha producido poco sobreajuste. Si las curvas de entrenamiento y validación están muy separadas, se ha producido un sobre entrenamiento y es importante revisar el modelo, ya que no se generaliza a nuevos datos como se esperaría.

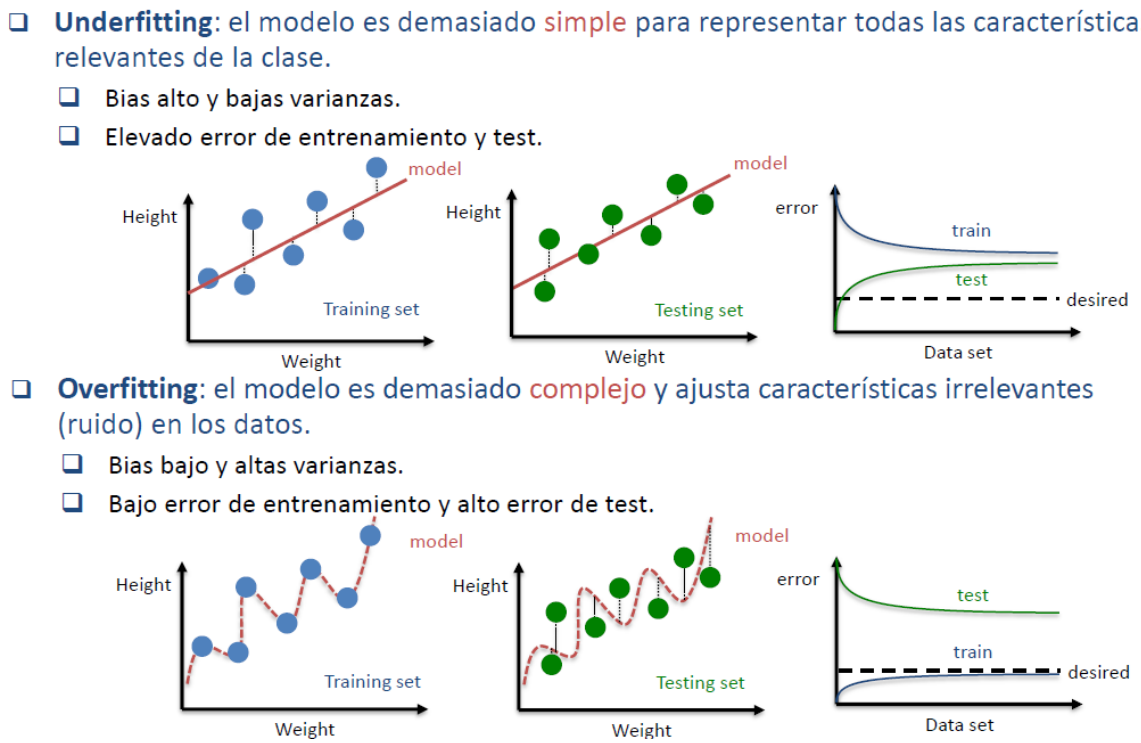


Figura 1-17 Problemas típicos a la hora de entrenar una red neuronal convolucional. Fuente: UAH.[20]

6. Implementar el modelo.

Una vez que la calidad del modelo es lo suficientemente buena para los requisitos de la solución, el siguiente paso es implementarlo. Hoy en día, los modelos de aprendizaje profundo se pueden implementar en la nube como un servicio API REST, implementarse en dispositivos móviles (ej., iPhone, teléfonos Android, iPad y más) o dispositivos perimetrales (ej., IoTs o Edge Devices). Esto dependerá de cómo se piense utilizar el modelo de aprendizaje profundo entrenado. Por ejemplo, si está desarrollando una aplicación web, tiene sentido poner en funcionamiento los modelos de aprendizaje profundo como API REST, que la aplicación web puede consumir más fácilmente. Si está desarrollando una aplicación perimetral (Edge), habrá que tener en cuenta los escenarios de conexión y desconexión, así como los requisitos de latencia, con modelos que se ejecuten sin conexión en el dispositivo edge o un modelo híbrido en el que tengamos combinaciones de modelos que se ejecutan en el dispositivo y API REST que brindan una funcionalidad más potente en la nube.

Para implementar los modelos de aprendizaje profundo como API REST, existen varias opciones. Podemos aprovechar los servicios de Azure Machine Learning para alojar el modelo en un contenedor accesible en la nube o en el perímetro (Edge) y exponer uno o más REST, o podemos crear nuestro propio alojamiento (por ejemplo, usando Flask, respaldado por un servidor web ligero como Guniorn).

[20] Visión Artificial, Grado Ingeniería en Electrónica de Comunicaciones UAH. Tema 5, pág. 15.

El tipo de hardware, como las GPU, también es un factor relevante para considerar.

1.5.6. Deep Learning Frameworks

Como se mencionó anteriormente, para realizar un entrenamiento de aprendizaje profundo es necesaria la utilización de un "framework" de aprendizaje profundo y realizar el entrenamiento con una GPU. La computación en GPU, especialmente mediante el uso eficiente de la multiplicación de matrices, se ha acelerado a través de frameworks como CUDA y OpenCL. Estos han habilitado bibliotecas de nivel superior como cuDNN sobre CUDA para construir redes neuronales profundas.
Hoy en día existen numerosos frameworks de aprendizaje profundo populares, como Tensorflow, PyTorch, CNTK, MXNet y Caffe2, así como API populares de nivel superior, como Keras y Gluon. La elección de un conjunto de herramientas de aprendizaje profundo depende de muchos factores, como son desde la disponibilidad de tutoriales, de arquitecturas modelo y modelos preentrenados, pasando por la experiencia previa del programador, o de la disponibilidad de funcionalidades auxiliares integradas y de la capacidad de aprovechar de manera eficaz tanto las CPU como las GPU, o la capacidad de realizar entrenamiento distribuido.

Para permitirnos comparar entre todos ellos una buena opción es el repositorio de comparación de aprendizaje profundo disponible en http://bit.ly/DLComparisons

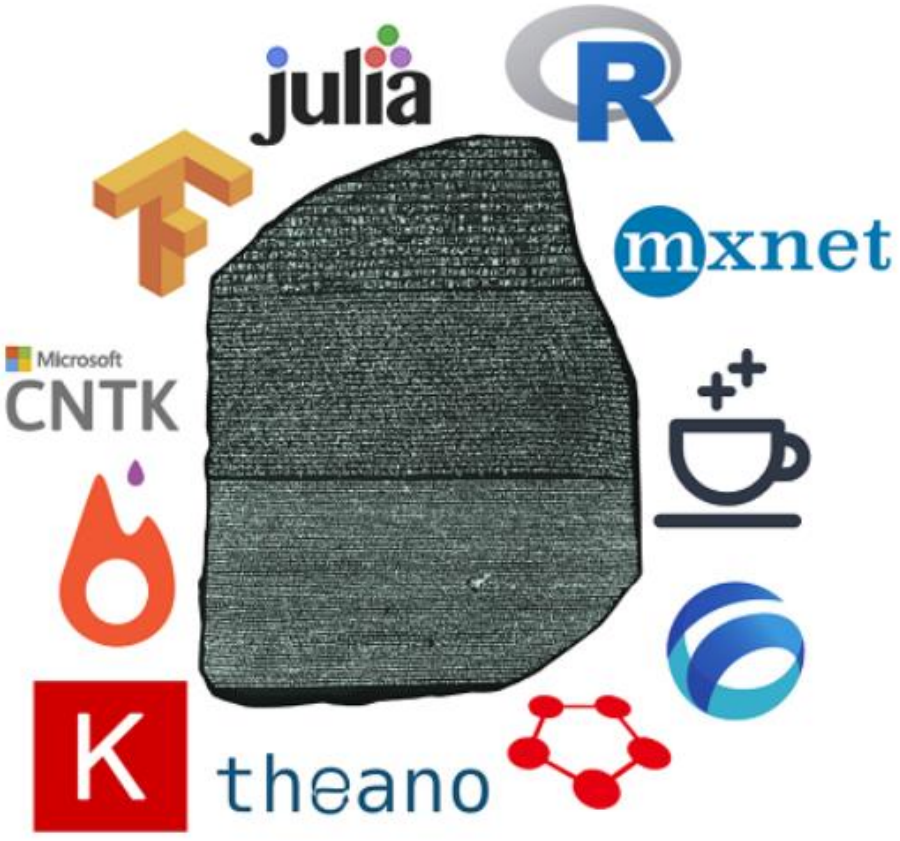


Figura 1-18 "Piedra de Rosetta" de marcos de aprendizaje profundo. Fuente: Github[21]

[21] http://bit.ly/DLComparisons

Este repositorio tiene varios objetivos declarados:

1. Una "piedra de Rosetta" de marcos de aprendizaje profundo para permitir que los científicos de datos aprovechen fácilmente su experiencia de un marco a otro.
2. Código de GPU optimizado utilizando las API de más alto nivel más actualizadas.
3. Una configuración común para comparaciones entre GPU.
4. Una configuración común para las comparaciones entre idiomas (Python, Julia, R).
5. La posibilidad de verificar el rendimiento esperado de la propia instalación.
6. Colaboración entre diferentes comunidades de código abierto.

Estas comparaciones sirven como una buena manera de comenzar y comparar muchos frameworks populares para escenarios comunes.

DL Library	K80/CUDA 8/CuDNN 6	P100/CUDA 8/CuDNN 6
Caffe2	148	54
Chainer	162	69
CNTK	163	53
MXNet(Gluon)	152	57
Keras(CNTK)	194	76
Keras(TF)	241	76
Keras(Theano)	269	93
Tensorflow	173	57
Lasagne(Theano)	253	65
MXNet(Module API)	145	52
PyTorch	169	51
Julia - Knet	159	??
R - Keras(TF)	205	72

Figura 1-19 Training Time(s): CNN (VGG-style, 32bit) on CIFAR-10 - Image Recognition. Fuente: Github[22]

Empresas y desarrolladores de frameworks populares se han unido a este esfuerzo de estandarizar la interoperabilidad de código abierto para transferir modelos de aprendizaje profundo entre frameworks. También hay paquetes que permiten convertir directamente de un

[22] https://github.com/ilkarman/DeepLearningFrameworks

framework a otro, como MMdnn, que ayuda a los usuarios a convertir directamente entre diferentes marcos, así como a visualizar la arquitectura del modelo.
Muchas de las bibliotecas de aprendizaje profundo también incluyen varios algoritmos de ML. La mayoría de estas bibliotecas de aprendizaje profundo admiten el entrenamiento distribuido, y esto ayuda mucho a realizar un aprendizaje profundo a escala. La mayoría de las bibliotecas de aprendizaje profundo tienen contenedores de Python.

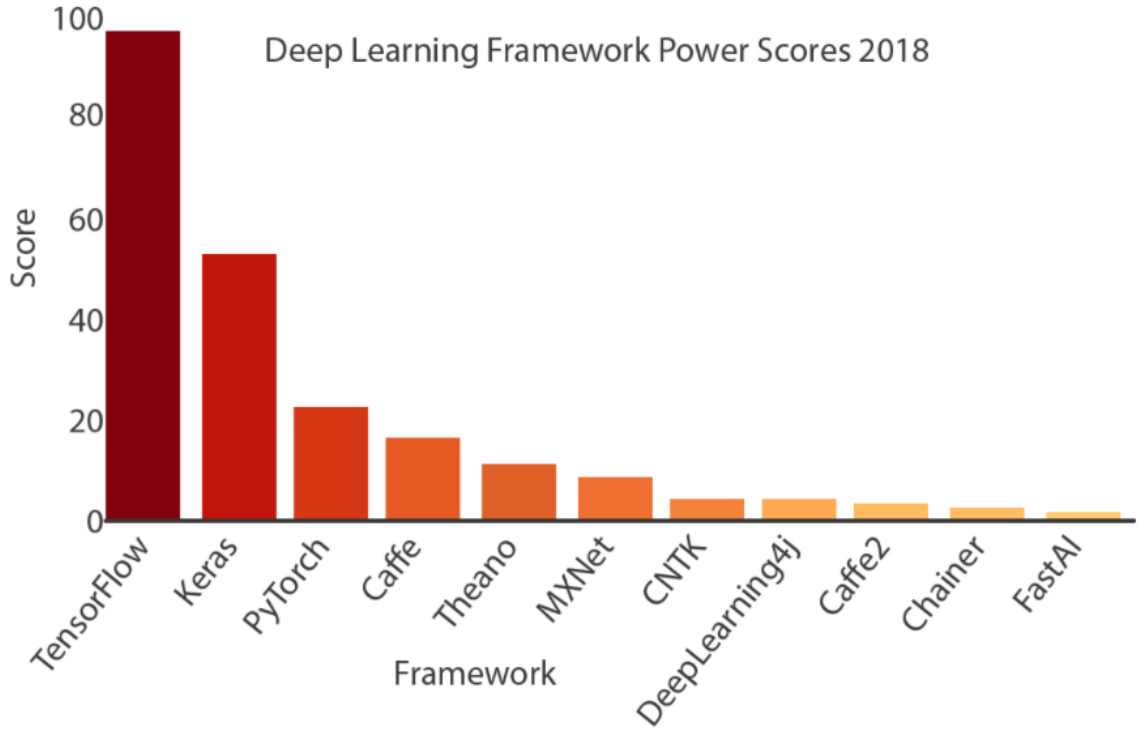


Figura 1-20 Deep Learning Framework Power Scores 2018 by Jeff Hale. Fuente: www.towardsdatascience.com[23]

Las GPU hacen posible el entrenamiento de modelos de aprendizaje profundo en un espacio de tiempo razonable. En los últimos años, las innovaciones tanto en los algoritmos como en la disponibilidad de GPU más rápidas han permitido completar rápidamente el entrenamiento de los modelos de aprendizaje profundo. Por ejemplo, la capacitación de CNN como ResNet-50 utilizando el conjunto de datos de ImageNet disponible públicamente solía tardar14 días o más antes de 2017. En 2017, el tiempo necesario para capacitar a ResNet-50 disminuyó significativamente, de una hora a aproximadamente 15 minutos. Preferred Network pudo entrenar el modelo ResNet-50 CNN con ChainerMN con 1024 GPU P100 en 15 minutos en noviembre de 2017, por ejemplo.

Biblioteca de modelos
Muchas redes neuronales profundas preentrenadas están disponibles para cada una de las bibliotecas de aprendizaje profundo. Por ejemplo, Microsoft CNTK y TensorFlow proporcionan modelos previamente entrenados para varias CNN de última generación

[23] https://towardsdatascience.com/top-5-deep-learning-frameworks-to-watch-in-2021-and-why-tensorflow-98d8d6667351

(AlexNet, GoogLeNet, ResNet y VGG). Caffe's Model Zoo proporciona un amplio conjunto de más de 40 modelos preentrenados para CNN de última generación (ResNet, Inception, VGG, etc.) y admite varios escenarios (por ejemplo, identificación de modelos de automóviles, reconocimiento de diferentes puntos de referencia, lugares…

1.6 Objetivos

El objetivo principal de este TFG ha sido realizar un estudio en profundidad de la tecnología de computación perimetral o edge computing y de cómo aplicarla en entornos industriales reales para lo cual se ha hecho uso del ecosistema Siemens Industrial Edge desarrollando un aplicativo de visión artificial que nos ha permitido repasar las principales ventajas de trabajar con este tipo de tecnología.

Entre los posibles campos de aplicación de la computación perimetral en entornos industriales estarían los siguientes:

- Sustitución de aplicaciones basadas en PC de difícil mantenimiento y despliegue
- Adquisición de datos de producción para su procesado y análisis local o en la nube
- Conectividad de aplicaciones haciendo de pasarela entre las redes de operación OT (Operational Technology) y las redes IT (Information Technology) para dar cobertura a las necesidades crecientes de comunicar un volumen cada vez mayor de datos.
- Control de procesos en lazo cerrado basados en algoritmos implementados en lenguajes de alto nivel y/o con inteligencia artificial.

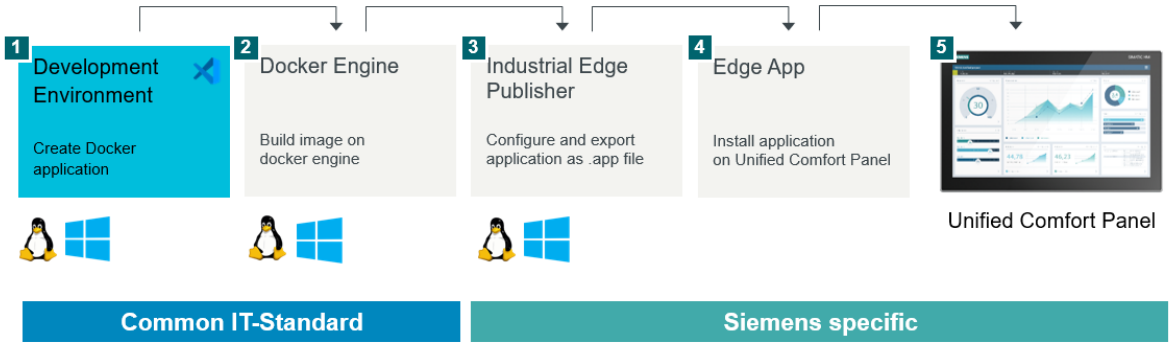


Figura 1-21 Fases para el desarrollo de aplicaciones Industrial Edge. Fuente: Siemens Industry Image Database 4.11

El otro gran objetivo perseguido por este TFG es que el mismo sirva de guía para cualquiera que quiera empezar en el desarrollo de aplicaciones (apps) para dispositivos Industrial Edge de Siemens, por lo que se han repasado todas fases implicadas en el desarrollo de aplicaciones y dando las bases necesarias para cualquiera que se quiera iniciar en Industrial Edge.

2. Alternativas comerciales en computación perimetral industrial

En esta sección se va a revisar las distintas alternativas comerciales existentes para desplegar soluciones de Edge computing en el ámbito industrial. Se evalúan las principales características de AWS, Azure y Siemens Industrial Edge.

2.1 Amazon Web Services (AWS)

Fundada en 2006 por Amazon, los ingresos de AWS en 2020 fueron de 45.000 millones de dolares y tiene 25.000 empleados
La solución de AWS para Edge computing industrial se denomina AWS Greengrass.
AWS Greengrass extiende algunas funciones clonadas de los servicios en la nube de AWS a dispositivos locales con su motor de tiempo de ejecución AWS instalado.

2.1.1. Funciones de AWS Greengrass

La idea de su oferta es fundamentalmente un paquete de software que proporciona los siguientes servicios de AWS (con acceso a la API del desarrollador) cuando se instala en un dispositivo:

- Ejecución de funciones de AWS Lambda:
  AWS IoT Greengrass incluye soporte para AWS Lambda. Con AWS IoT Greengrass, puede ejecutar funciones de AWS Lambda en el dispositivo para responder rápidamente a eventos locales, interactuar con recursos locales y procesar datos para minimizar el costo de transmitir datos a la nube.
- Administración de contenedores:
  Puede desplegar, ejecutar y gestionar contenedores de Docker en dispositivos de AWS IoT Greengrass. Sus imágenes de Docker pueden almacenarse en registros de contenedores de Docker como Amazon Elastic Container Registry (Amazon ECR), Docker Hub, o Docker Trusted Registries (DTRs) privados.
- Asistencia local para sombras de dispositivos con AWS IoT:
  AWS IoT Greengrass también incluye la funcionalidad de las sombras de dispositivos con AWS IoT. La sombra del dispositivo almacena en caché el estado de un dispositivo, como una versión virtual, o una “sombra”, de cada dispositivo que monitorea el estado actual del dispositivo en comparación con el estado que se desea alcanzar, y sincroniza dicho estado con la nube cuando hay conectividad disponible.
- Mensajería:
  AWS IoT Greengrass permite la mensajería entre AWS IoT Greengrass Core y los dispositivos mediante SDK de AWS IoT Device en una red local para facilitar la comunicación incluso cuando no hay conexión con AWS. Con AWS IoT Greengrass, los dispositivos pueden procesar mensajes y entregarlos a otro dispositivo o a la nube a partir de reglas empresariales que defina.

- Acceso directo a HW:
  Las funciones de AWS Lambda implementadas en AWS IoT Greengrass Core tienen acceso a los recursos locales adjuntos al dispositivo. Esto permite usar puertos serie, periféricos como dispositivos de seguridad agregados, sensores y accionadores, GPU incorporados o el sistema de archivos local para obtener acceso rápidamente a los datos locales y poder procesarlos.
- Depuración y otras herramientas de desarrollo local:
  AWS IoT Greengrass le permite desarrollar rápidamente un código de depuración en un dispositivo de prueba antes de utilizar la nube para implementar en los dispositivos de producción. Puede utilizar la interfaz de línea de comandos (CLI) de AWS IoT Greengrass para desarrollar y depurar localmente las aplicaciones de su dispositivo, y la consola de depuración local para ayudarle a depurar visualmente las aplicaciones.
- AWS Sagemaker ML Inference:
  La inferencia de aprendizaje automático de AWS IoT es una característica de AWS IoT Greengrass que facilita la inferencia mediante el aprendizaje automático de forma local en los dispositivos AWS IoT Greengrass que utilicen modelos creados y programados en la nube. Eso significa que no incurrirá en costos de transferencia de datos ni se provocará mayor latencia en las aplicaciones que usen inferencia de aprendizaje automático. Para obtener más información sobre la característica de inferencia de aprendizaje automático.
- AWS Kinesis Stream Manager
  Puede usar AWS IoT Greengrass para recopilar, procesar y exportar secuencias de datos desde dispositivos IoT y administrar el ciclo de vida de esos datos en el dispositivo para minimizar el tiempo de desarrollo. AWS IoT Greengrass proporciona un mecanismo estándar para procesar secuencias de datos, gestionar políticas de retención de datos y transmitir los datos del dispositivo a servicios en la nube de AWS, como el servicio de almacenamiento simple de Amazon (Amazon S3), Amazon Kinesis, AWS IoT Core y AWS IoT Analytics.[24]

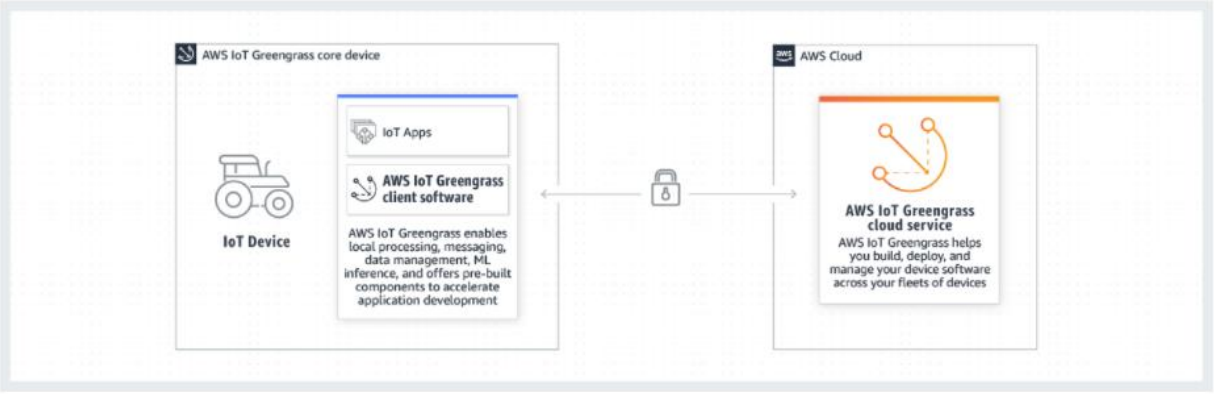


Figura 2-1 El siguiente ejemplo muestra cómo AWS IoT Greengrass interactúa con la nube de AWS. Fuente AWS

2.1.2. Precios

Con AWS IoT Greengrass, se paga solamente por lo que se utiliza. Se cobra en función del número de dispositivos AWS IoT Greengrass Core que se conectan al servicio de nube de AWS

[24] https://aws.amazon.com/es/greengrass/features/?pg=ln&sec=hs

IoT Greengrass en un mes determinado. No se cobra si un dispositivo AWS IoT Greengrass Core no se conecta al servicio de la nube.

Se considera que un núcleo de AWS IoT Greengrass ha estado activo durante el mes cuando se autentica contra AWS. AWS IoT Greengrass Core únicamente se identifica mediante AWS IoT que representa el dispositivo conectado y utiliza un certificado de dispositivo para autenticarlo con AWS IoT. Puede conectar dispositivos con AWS IoT Greengrass Core de manera local sin cargo adicional.[25]

Número de dispositivos	Precio mensual por dispositivo
1 a 10 000 dispositivos	0,18 USD por mes
Más de 10 000 dispositivos	Contáctenos

También se puede incurrir en cargos adicionales con AWS IoT Greengrass si las aplicaciones utilizan otros servicios de AWS o se transfieren datos

Nivel gratuito de AWS
El nivel gratuito de AWS incluye los tres primeros dispositivos AWS IoT Greengrass Core que se conectan cada mes, durante un año.
El uso del nivel gratuito se calcula mensualmente en todas las regiones de AWS, excepto en la región de AWS GovCloud, y se aplica a su factura de manera automática; no se acumulará el uso mensual no consumido.

2.2 Microsoft Azure IoT Edge

Microsoft se fundó en 1975 como una start-up. Hoy tiene 163.000 empleados, y unos ingresos 143 mil millones de dólares.
Por su parte MS Azure está en el segmento de “Nube Inteligente”; Los ingresos estimados de MS Azure fueron de 25 a 30 mil millones en 2020.
Dentro de los servicios y productos de MS Azure, Azure IoT Edge es una solución edge para ejecutar contenedores (llamados módulos en la terminología de Azure) y administrar dispositivos on- premise. Toda la gestión se realiza desde la nube, utilizando su servicio denominado “Azure IoT Hub”. También es una oferta muy centrada en el desarrollador como otras plataformas edge “Hyperscaler” (pero un poco menos que AWS Greengrass). Se comercializa junto con su nube, y está destinado a ser utilizado como un componente de un producto completo como cualquiera de sus otros servicios en la nube.

2.2.1. Características de Azure IoT Edge

Azure IoT Edge tiene menos funciones listas para usar que AWS Greengrass, y se centra en lo siguiente:

[25] https://aws.amazon.com/es/greengrass/pricing/

- Servicios de comunicación: permitir la comunicación entre dispositivos OT de planta aguas abajo (por ejemplo, PLC de planta), otros dispositivos Azure IoT Edge y la nube de servicios de Azure.
- Administración de módulos: "módulo" es solo la terminología de Azure para un contenedor, la oferta de Azure IoT Edge proporciona varias funciones para administrarlos (por ejemplo, instalar un contenedor, reiniciar un contenedor, etc.)

Todas las demás características proporcionadas en una implementación de Azure IoT Edge se encapsulan en un "módulo".
El mercado de Azure IoT Edge tiene bastantes más módulos que otras ofertas, incluidas aplicaciones propias y de terceros que tienen los siguientes tipos de funcionalidad:

- Analitica de datos y monitorización
- Conectividad IoT
- IoT Core Services
- Almacenamiento y bases de datos
- Otros

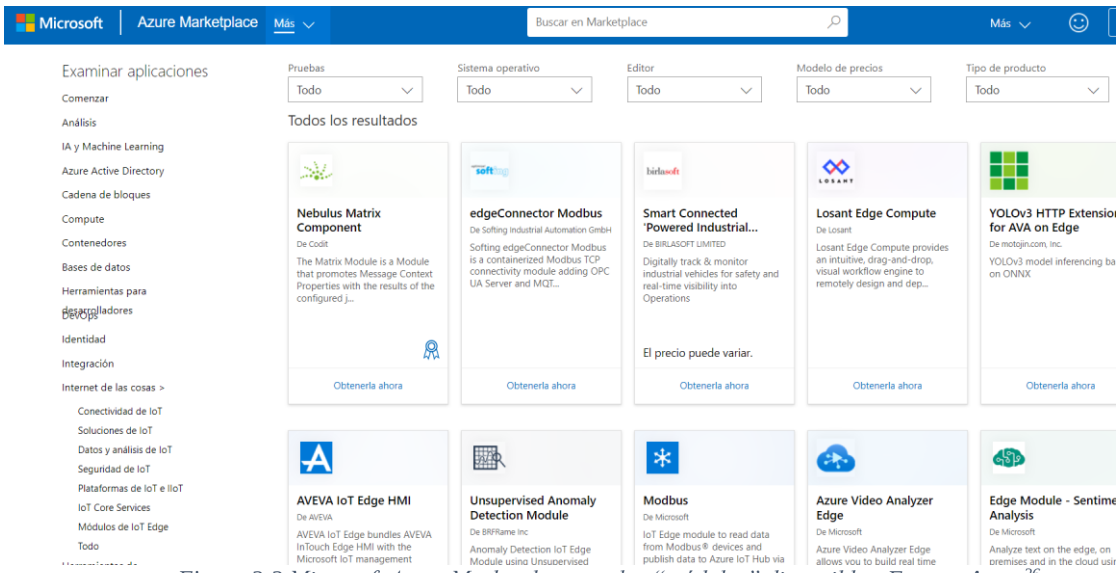


Figura 2-2 Microsoft Azure Marketplace con los "módulos" disponibles. Fuente: Azure[26]

2.2.2. Arquitectura de Azure IoT Edge

- **Los módulos de IoT Edge:** son contenedores que ejecutan servicios de Azure, de terceros o código personalizado. Se implementan en dispositivos habilitados para IoT Edge y se ejecutan en ellos.[27]
- **El entorno de ejecución de IoT Edge** se ejecuta en todos los dispositivos habilitados para IoT Edge y administra los módulos que se implementan en cada dispositivo. Azure IoT Edge está diseñado para ejecutarse en Windows y Linux, y en el hardware que elijamos. (Siempre que elijamos HW certificado)

[26] https://azure.microsoft.com/es-es/marketplace/

[27] https://devicecatalog.azure.com/

- **La interfaz basada en la nube** que supervisa y administra de forma remota los dispositivos habilitados para IoT Edge.

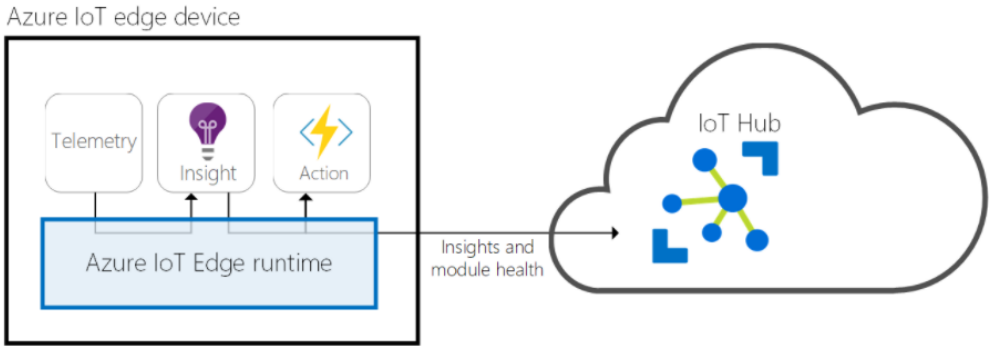


Figura 2-3 Entorno de ejecución de Azure IoT Edge. Fuente: Azure[28]

2.2.3. Precios

El servicio Azure IoT Edge en los dispositivos locales es gratuito, pero es necesario Azure IoT Hub para la administración de dichos dispositivos. Por otro lado, si se van a utilizar servicios o aplicaciones de Azure (módulos) con IoT Edge, se factura ese módulo concreto según su modelo de facturación para usarlo en el dispositivo edge.
Por ejemplo, si elige ejecutar Azure Stream Analytics en IoT Edge, se cobra el uso de IoT Hub y el precio de Azure Stream Analytics en dispositivos edge.

- Runtime Azure IoT Hub:
  Es gratuito y de código abierto bajo licencia MIT[29]

- Azure IoT Hub:
  Es necesario para la administración de los dispositivos Edge de Azure de forma segura

Nivel Standard

Tipo de edición	Precio por unidad (al mes)	Número total de mensajes al día por unidad	Tamaño del medidor de mensajes
Gratis	Gratis	8.000	0,5 KB
S1	€22,495	400.000	4 KB
S2	€224,942	6.000.000	4 KB
S3	€2249,416	300.000.000	4 KB

Figura 2-4 Tabla de precios Azure IoT Hub. Fuente: Azure[30]

- Módulos Azure IoT Edge

[28] https://docs.microsoft.com/es-es/azure/iot-edge/iot-edge-runtime?view=iotedge-2020-11

[29] https://github.com/Azure/iotedge

[30] https://azure.microsoft.com/es-es/pricing/details/iot-edge/

El precio dependerá del módulo y si es de Azure o de terceros.

Módulo Edge	Precio (al mes)
Azure Stream Analytics en IoT Edge	€0,900/dispositivo/mes
Custom Vision (versión preliminar)	Gratis
Azure Functions	Gratis
Azure SQL Database	Traiga su propia licencia

Figura 2-5 Tabla de precios de Módulos Azure IoT Edge. Fuente: Azure[31]

Nivel gratuito de Azure IoT Hub

El entorno de tiempo de ejecución o runtime de IoT Edge es gratuito y de código abierto, pero con limitaciones de transferencia de datos, también es posible ejecutar muchos de los módulos edge de forma gratuita.

2.2.4 Operación sin conexión a Azure IoT Hub

Azure IoT Edge en sus últimas versiones permite trabajar sin conexión de los dispositivos Edge al IoT Hub. Será necesario una única conexión de sincronización y después de está se podrá trabajar de forma indefinida en el dispositivo Edge local.

El procedimiento lo podemos seguir en las siguientes figuras.

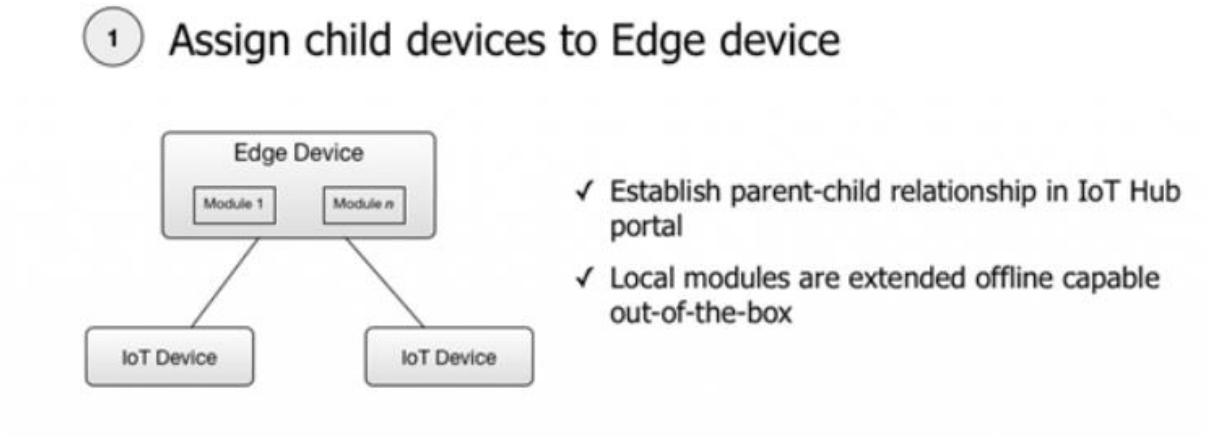


Figura 2-6 Configuración de dispositivos. Fuente: Azure

[31] https://azure.microsoft.com/es-es/pricing/#product-pricing

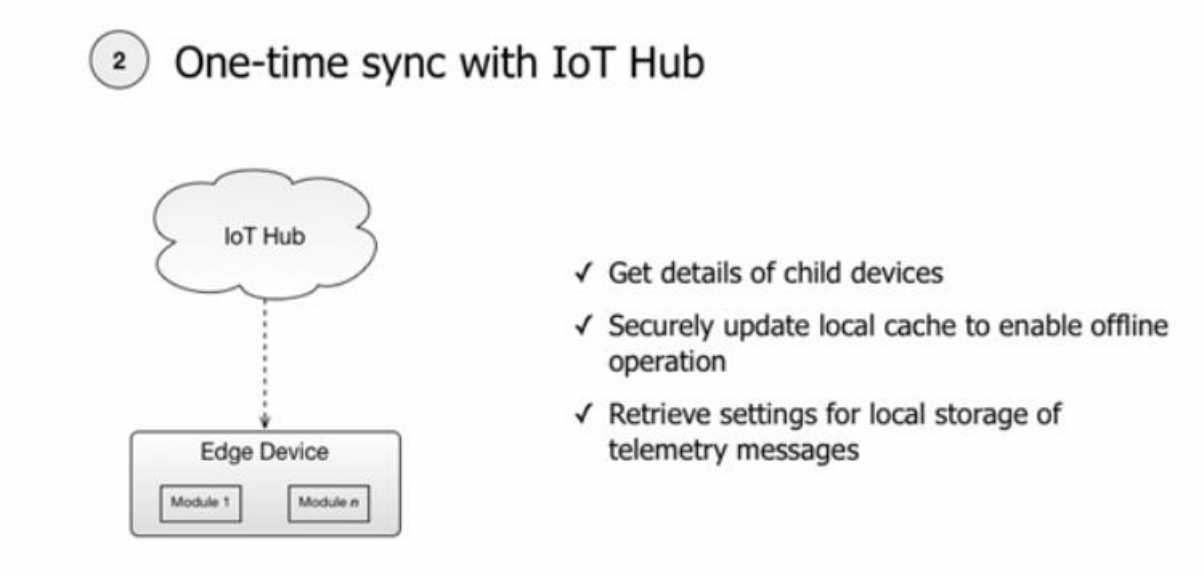


Figura 2-7 Sincronización con IoT Hub. Fuente: Azure

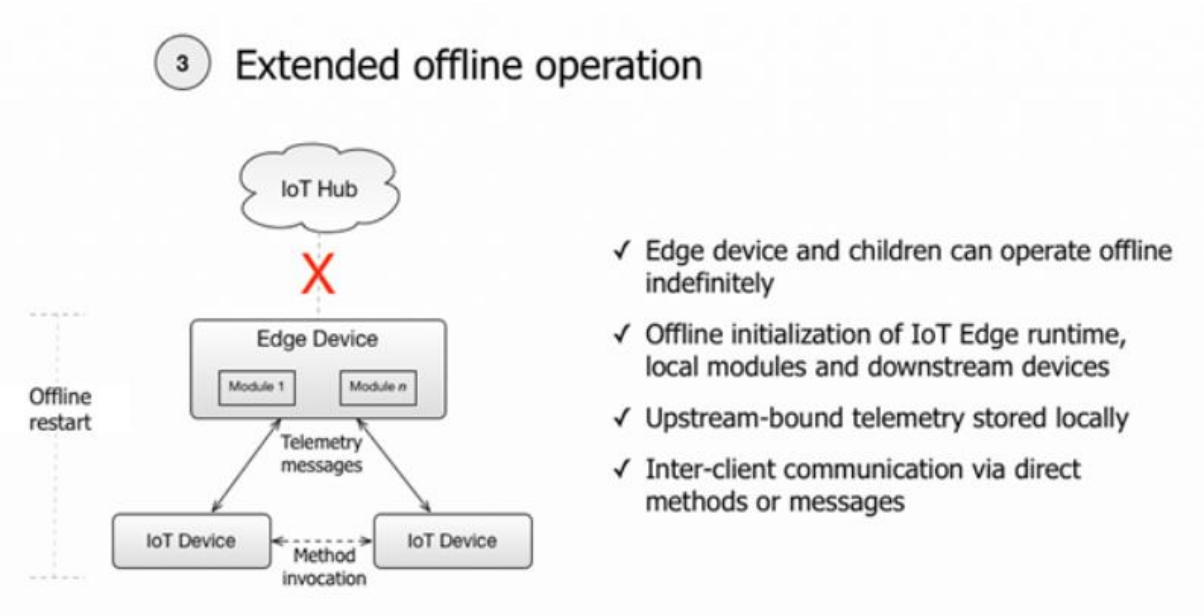


Figura 2-8 Desconexión. Fuente: Azure

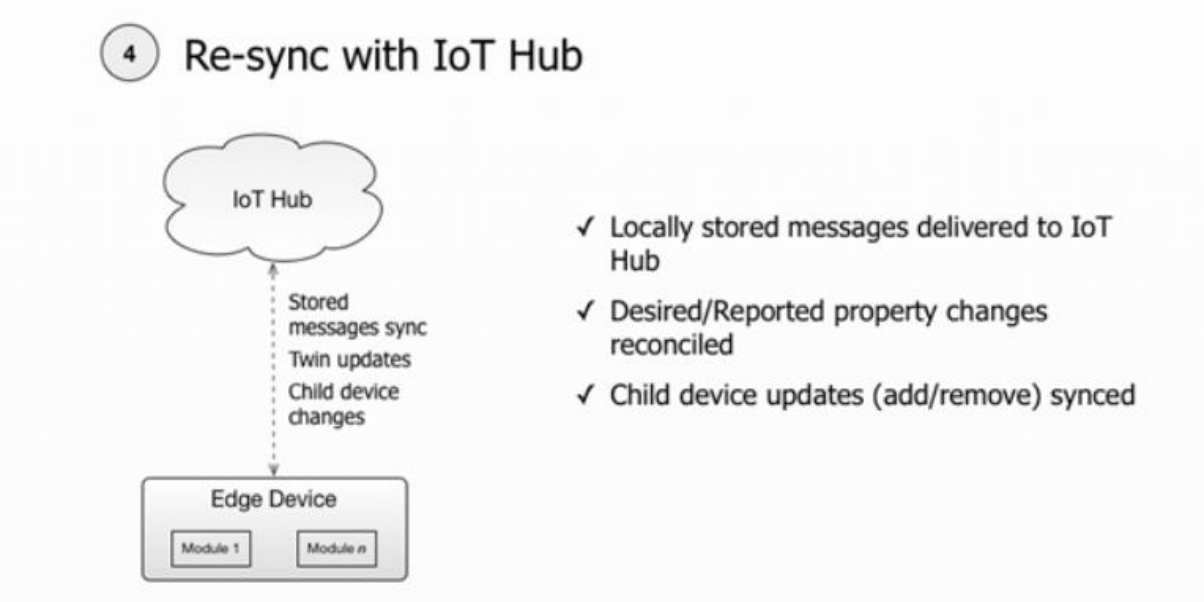


Figura 2-9 Nueva conexión y sincronización con IoT Hub. Fuente: Azure[32]

- Configurar dispositivos
  De forma automática, los dispositivos de IoT Edge tienen habilitadas funcionalidades sin conexión. Para hacer extensible esa capacidad a otros dispositivos, hay que configurar los dispositivos secundarios para que confíen en su dispositivo principal asignado y enrutar las comunicaciones de dispositivo a nube a través del principal como puerta de enlace.
- Sincronización con IoT Hub
  Es necesario que, al menos una vez después de instalar el entorno de ejecución de IoT Edge, el dispositivo de IoT Edge tenga conexión para sincronizarse con IoT Hub. En esta sincronización, el dispositivo IoT Edge obtiene información detallada sobre los dispositivos secundarios asignados a él. El dispositivo IoT Edge también actualiza de forma segura su caché local para permitir las operaciones sin conexión y recupera la configuración del almacenamiento local de mensajes de telemetría.
- Desconexión
  Mientras el dispositivo IoT Edge está desconectado de IoT Hub, sus módulos implementados y los dispositivos secundarios pueden operar indefinidamente. Mientras están sin conexión, los módulos y los dispositivos secundarios pueden iniciarse y reiniciarse autenticándose con el centro de IoT Edge. Los datos de telemetría que deben enviarse a IoT Hub se almacenan localmente. La comunicación entre módulos o entre dispositivos secundarios se realiza mediante mensajes o métodos directos.
- Nueva conexión y nueva sincronización con IoT Hub
  Una vez que se restaura la conexión con IoT Hub, el dispositivo de IoT Edge vuelve a sincronizarse. Los mensajes almacenados de manera local se entregan a la instancia de IoT Hub directamente, pero dependen de la velocidad de la conexión, la latencia de IoT Hub y otros factores relacionados. Los mensajes se entregan en el mismo orden en el que se almacenaron.[33]

[32] https://azure.microsoft.com/es-es/blog/extended-offline-operation-with-azure-iot-edge/

[33] https://docs.microsoft.com/es-es/azure/iot-edge/offline-capabilities?view=iotedge-2020-11

Las diferencias entre las propiedades deseadas y notificadas de los módulos y los dispositivos se concilian. El dispositivo IoT Edge actualiza los cambios en su conjunto de dispositivos secundarios asignados.

2.3 Siemens Industrial Edge

2.3.1 Características Siemens Industrial Edge

El ecosistema Industrial Edge de Siemens permite la ejecución de aplicaciones implementadas en lenguajes de alto nivel basadas en tecnología de contenedores Docker.
El contenedor nos permite un modo de virtualización de aplicaciones ligero para poder correr en hardware de bajas prestaciones (comparable a aplicaciones de teléfonos inteligentes)

Esto permite el procesado y análisis de datos en tiempo real a nivel de campo por lo que resulta de especial utilidad en el área de la visión artificial para aplicaciones industriales.
Con Industrial Edge podemos integrar en un mismo dispositivo Edge aplicaciones de visión artificial con, por ejemplo, sistemas HMI/SCADA de amplio uso en entornos industriales.
Las aplicaciones desarrolladas para Industrial Edge se pueden desplegar desde un sistema de gestión centralizada basado en la nube o bien en un servidor dedicado en la propia instalación.
De esta manera el despliegue y mantenimiento de las aplicaciones se simplifica y estandariza permitiendo el escalado de las aplicaciones.

2.3.2 Arquitectura Siemens Industrial Edge

Siemens Industrial Edge se compone de tres partes fundamentales, el motor de tiempo de ejecución “Runtime” que ejecuta las aplicaciones, el gestor de estas para su despliegue y mantenimiento y las aplicaciones propiamente dichas.
Estas aplicaciones pueden ser tanto desarrolladas por Siemens como por terceros y subidas al “store” de Siemens Industrial Edge.
La arquitectura del sistema Siemens Industrial Edge está basado en tres pilares:

- Industrial Edge Runtime: ejecución segura y escalable de aplicaciones Edge para procesamiento de datos y conectividad en automatización a pie de máquina.
- Industrial Edge Apps: para procesamiento de datos, conectividad y tareas analíticas basadas en lenguajes de programación de alto nivel y Docker.
- Industrial Edge Management: administración centralizada de dispositivos distribuidos para actualizaciones masivas, de seguridad y distribución de aplicaciones

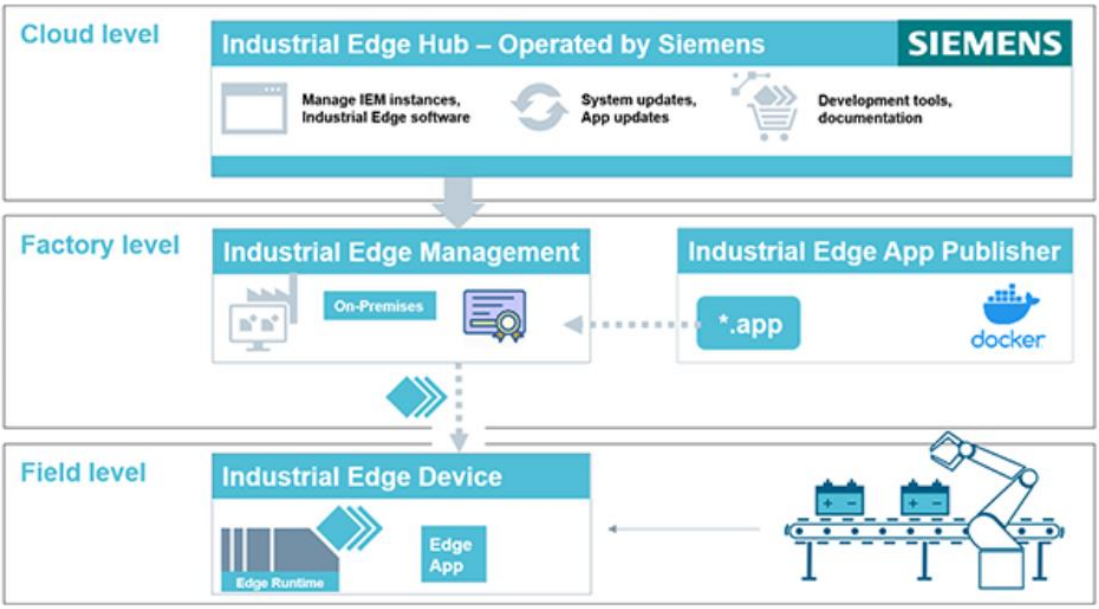


Figura 2-10 Arquitectura del ecosistema Siemens Industrial Edge. Fuente: Siemens Industry Image Database 4.11

Posibles campos de aplicación

- Sustitución de aplicaciones basadas en PC de difícil mantenimiento y despliegue
- Adquisición de datos de producción para su procesado y análisis local o en la nube
- Conectividad de aplicaciones haciendo de pasarela entre las redes de operación OT (Operational Technology) y las redes IT (Information Technology) para dar cobertura a las necesidades crecientes de comunicar un volumen cada vez mayor de datos.
- Control de procesos en lazo cerrado basados en algoritmos implementados en lenguajes de alto nivel y/o con inteligencia artificial.

2.3.3. Precios

En la figura siguiente se muestra una tabla con los precios orientativos de los distintos sistemas basados en Industrial Edge.

1	**Industrial Edge Access** Get Initial Access to IE Hub	6ES7823-0EE00-4AX0	1 €
2	**Industrial Edge Devices**	Device specific	Device specific
3	**Industrial Edge Device Licences** Per device per year	6ES7823-0EE00-4AY0	120 € p.a.
4	**Industrial Edge Apps**	App specific	App specific p.a.

Figura 2-11 Tabla de precios orientativos Siemens Industrial Edge. Fuente: Siemens Industry Image Database 4.11

2.4 Comparativa de alternativas Edge Computing

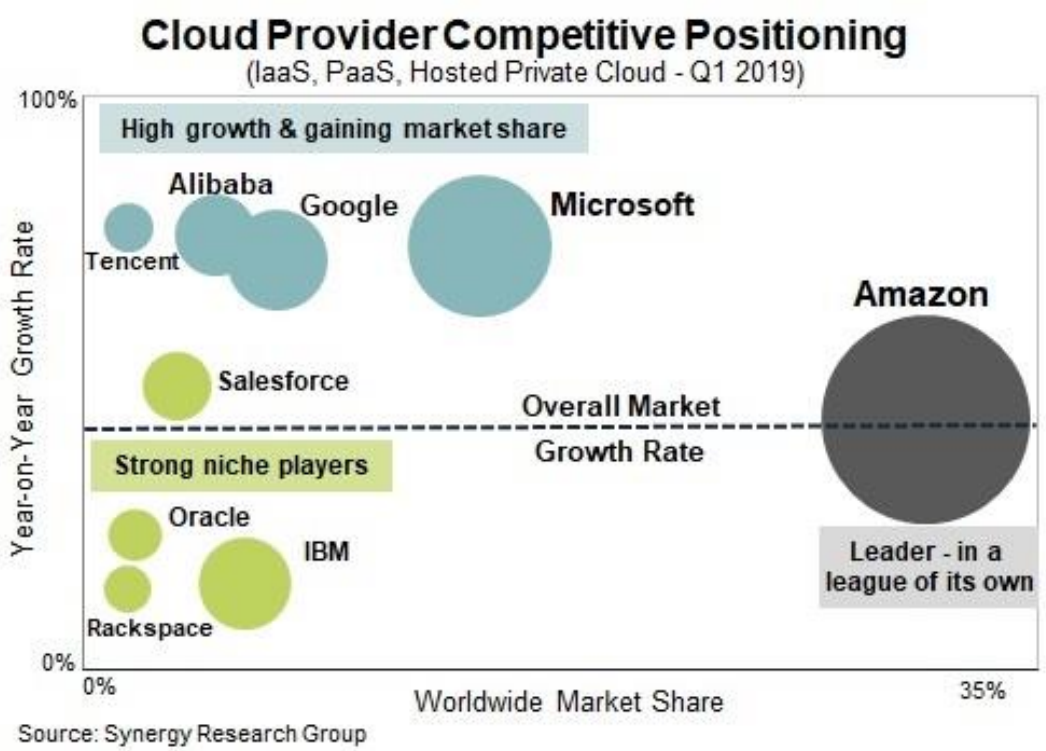


Figura 2-12 Negocio mundial Cloud. Fuente: Synergy Research Group

2.4.1 AWS Greengrass vs Siemens Industrial Edge

A continuación, vamos a repasar los puntos principales de diferenciación entre AWS Greengrass y Siemens Industrial Edge:

- Conexión a internet: todos los cambios o actualizaciones de los dispositivos Greengrass deben realizarse mediante la consola basada en la nube en un centro de datos remoto de AWS, por lo que se requiere una conexión constante a Internet para garantizar que siempre exista la opción de resolver problemas o realizar actualizaciones en las operaciones de OT de la planta. Por contra Siemens Industrial Edge puede alojar todo el sistema de gestión y la runtime de tiempo de ejecución en las instalaciones.

- Ciclo de vida: cada versión de Greengrass solo tiene soporte durante tres años (con parches de seguridad, correcciones de errores críticos, etc.). En el sector industrial, donde los productos se implementan y operan a menudo durante más de 10 años sin tocarlos, esta es un punto débil para AWS. Además, la carga del trabajo recae en los clientes para realizar la actualización de manera oportuna o correr el riesgo de tener problemas de seguridad.

- Usabilidad: diseñado para desarrolladores: la única forma de usar Greengrass es con API de desarrollador y acceso a la línea de comandos/SDK. Es difícil implementarlo y requiere un largo flujo de trabajo para configurar "registros" y "kits de conectores". No existe una interfaz de usuario simple para operarlo. Por contra Siemens Industrial Edge cuenta con una interfaz de usuario limpia y "modo TI vs OT" para operaciones fáciles con un solo clic.

- ▪ Sin recuperación ante desastres y fallos críticos, los mecanismos de backup y recuperación son responsabilidad exclusivamente del usuario.

2.4.2 Comparativa AWS Vs Siemens Industrial Edge

- Conexión a Internet: para la mayoría de las funciones: todos los cambios o actualizaciones de los dispositivos Greengrass deben realizarse mediante la consola basada en la nube en un centro de datos remoto de AWS, por lo que se requiere una conexión constante a Internet para garantizar que siempre exista la opción de abordar problemas o realizar actualizaciones en las operaciones de TI de la planta. Siemens Industrial Edge puede alojar todo el sistema de gestión y el tiempo de ejecución en las instalaciones. AWS requiere con su solución un 1Gbit permanente de conexión a Internet.

- Ciclo de vida de soporte a muy corto plazo: cada versión de Greengrass solo tiene soporte durante tres años (con parches de seguridad, correcciones de errores críticos, etc.). En el entorno industrial, donde los productos se implementan y operan a menudo durante más de 10 años sin tocarlos, esto es una desventaja de AWS.

- Usabilidad: diseñado para desarrolladores: la única forma de usar Greengrass es con API de desarrollador y acceso a la línea de comandos/SDK. Es difícil implementarlo y requiere un largo flujo de trabajo para configurar "registros" y "kits de conectores". No existe una interfaz de usuario simple para operarlo.

- Sin funcionalidad de recuperación ante desastres: dado que Greengrass es solo una runtime de tiempo de ejecución de software, la instalación de la misma y mantenimiento en el PC recae en el cliente que deberá determinar su propio plan de recuperación ante desastres e implementar los mecanismos relacionados (p. ej., particiones azul/verde, reinicio automático, etc.).

2.4.3 Comparativa Azure IoT Edge Vs Siemens Industrial Edge.

- Conexión a internet: de forma predeterminada, todos los cambios o actualizaciones se realizan mediante la consola basada en la nube en un centro de datos remoto de Azure. Hay algunas características adicionales de las que AWS, por ejemplo, no dispone que permiten algunas funciones básicas en un escenario sin conexión, pero solo después de que los dispositivos Azure IoT Edge se comuniquen primero con la nube para configurarlos y sincronizarlos.

- Usabilidad: diseñado para desarrolladores: hay alguna interfaz de usuario en la nube para administrar dispositivos, pero aparte de eso, el uso de Azure IoT Edge requiere conocimiento de las API para desarrolladores y acceso a la línea de comandos/SDK.

- Sin funcionalidad de recuperación ante desastres: dado que Azure IoT Edge es solo una runtime de tiempo de ejecución de software, la carga recae en el cliente para determinar su propio plan de recuperación ante desastres e implementar los mecanismos relacionados de backup y restore.

2.4.4 Ventajas Siemens Industrial Edge.

- Hardware dedicado: El hardware para correr la runtime de tiempo de ejecución de Siemens Industrial Edge incorpora preinstalado y configurada la misma runtime para un despliegue rápido de soluciones Edge en planta. Con mecanismos de backup y restore bien establecidos. Por otra parte, si se requiere correr la rumtime en un PC no siemens está previsto el lanzamiento en unos meses de la “Runtime Virtual” para habilitar esa posibilidad.

- Conexión a Internet: Siemens Industrial Edge puede trabajar de forma completa sin necesidad de conexión a internet de los dispositivos Edge pudiendo ser estos configurados y mantenidos desde un Industrial Edge Management (IEM) instalado en un PC de la red de planta.

- Usabilidad: Interfaz de gestión sencilla tanto en los dispositivos Edge locales como en el gestor que permite a usuarios sin conocimiento avanzados IT desplegar soluciones basadas en Industrial Edge.

3. Arquitectura Industrial Edge

3.1 Visión general

Como hemos indicado anteriormente la plataforma Siemens Industrial Edge está basada en cuatro componentes principales que son: los dispositivos Industrial Edge con la runtime preinstalada, el Industrial Edge Management que puede estar centralizado en un PC de planta o la nube localizado en la nube o distribuido aprovechando el gestor incluido en el propio dispositivo Edge, el Industrial Edge Hub, el cual es un repositorio de App de Siemens en la nube y las propias aplicaciones (apps)

Figura 3-1 Componentes Siemens Industrial Edge. Fuente: Siemens Industry Image Database 4.11

Industrial Edge Hub (IEH)

Se trata de un repositorio global de Industrial Edge, para monitorizar todos los sistemas de administración implementados y seleccionar aplicaciones para instalar desde Edge App Store.

Las principales funcionalidades serían:

- Administración de muchas implementaciones de Industrial Edge alrededor del mundo
- Gestionar un repositorio centralizado de aplicaciones
- Control de licenciamiento.

Industrial Edge Hub contiene:

- Imagen del software del sistema de gestión de Edge industrial
- Actualizaciones de firmware para dispositivos Industrial Edge, incluido Industrial Edge Runtime
- Actualizaciones para aplicaciones Industrial Edge
- Documentación del usuario
- Una licencia de administración de edge industrial para incorporar y administrar un dispositivo de edge industrial

- Las siguientes aplicaciones del sistema Industrial Edge están disponibles en Industrial Edge Hub de forma predeterminada y gratuita:
    - Conector SIMATIC S7
    - Conector OPCUA
    - Conector de nube (MQTT)
    - Aplicación Flow Creator Edge

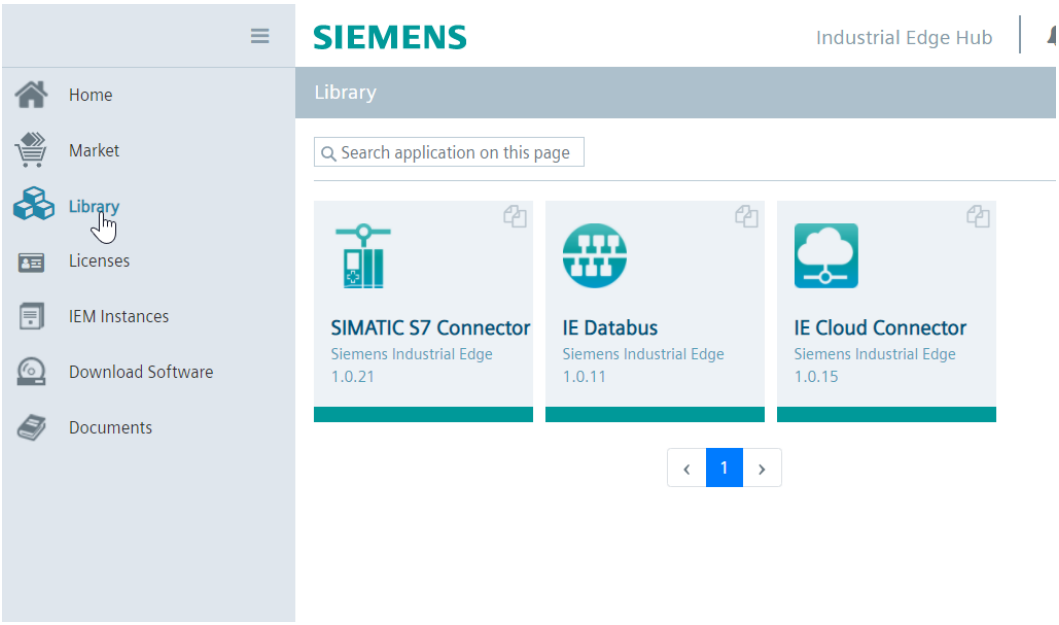


Figura 3-2 Consola principal del Industrial Hub.. Fuente: Siemens Industry Image Database 4.11

Industrial Edge Management (IEM)

Se trata de una herramienta de control centralizado para gestionar todos los dispositivos, aplicaciones y usuarios de un despliegue Edge en planta.

Las funcionalidades principales son:

- Almacenamiento de imágenes locales de aplicaciones para los dispositivos con Industrial Edge
- Gestión de usuarios
- Gestión de actualizaciones de seguridad y aplicaciones

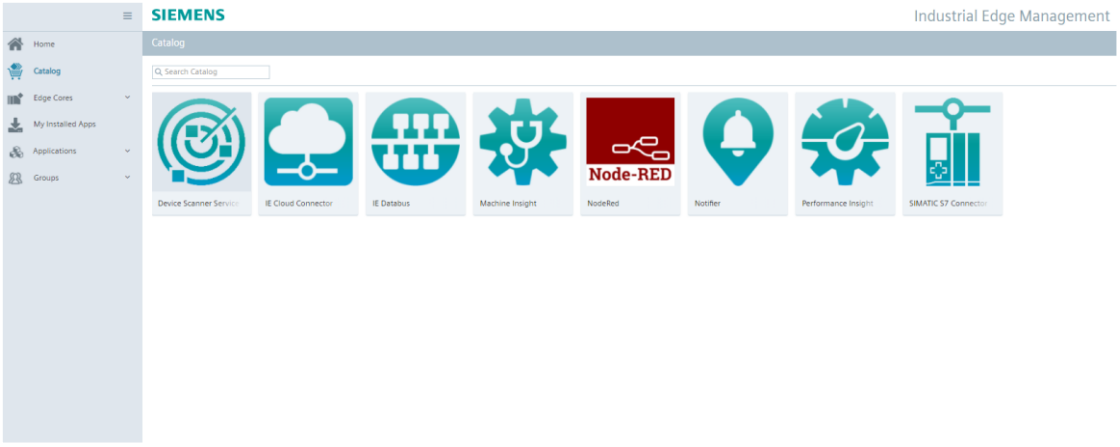


Figura 3-3 Industrial Edge Management. Fuente: Siemens Industry Image Database 4.11

Industrial Edge Runtime
Se trata de una capa de software que permite ejecutar aplicaciones de contenedores que se descargan desde Industrial Edge Management. También proporciona servicios de usuario de autenticación/IAM y comunicación con dispositivos OT aguas abajo.

- Diseñado para adaptarse a entornos industriales al garantizar la seguridad y la confiabilidad
- Conectividad de dispositivos integrados a sistemas de automatización y en la nube
- Funciones para cumplir con las políticas de la empresa, como administración de usuarios y capacidades de proxy inverso (número de puertos utilizados)

Dispositivo industrial Edge (IED)
El IED con la runtime se coloca en el nivel de campo, donde tiene lugar la generación de datos de los sistemas de automatización y la adquisición de los mismos. Estos dispositivos pueden almacenar datos de automatización localmente y recuperarlos según sea necesario. Además, los IED pueden cargar estos datos en infraestructuras en la nube como MindSphere, AWS o Azure y recuperarlos en cualquier momento. El IED se activa mediante un archivo de configuración, que se crea en el IEM.

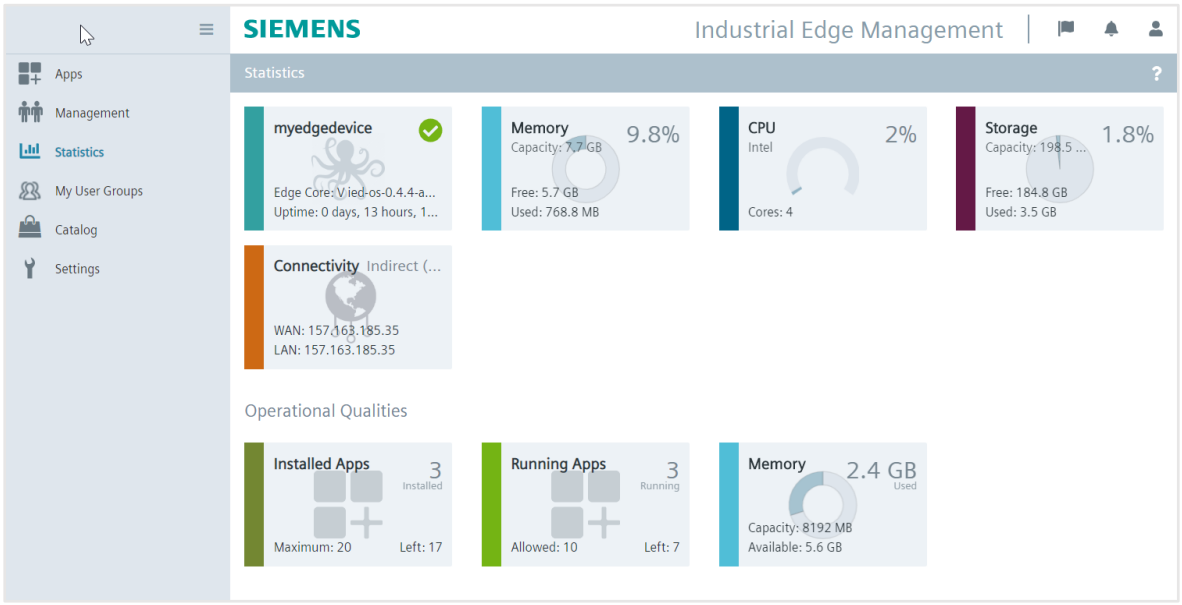


Figura 3-4. Estadísticas Industrial Edge Management

Aplicación Industrial Edge:
Las aplicaciones Industrial Edge se utilizan para el procesamiento inteligente de datos de automatización. Estas aplicaciones están disponibles directamente de Siemens, de terceros o desarrolladas por nosotros mismos.

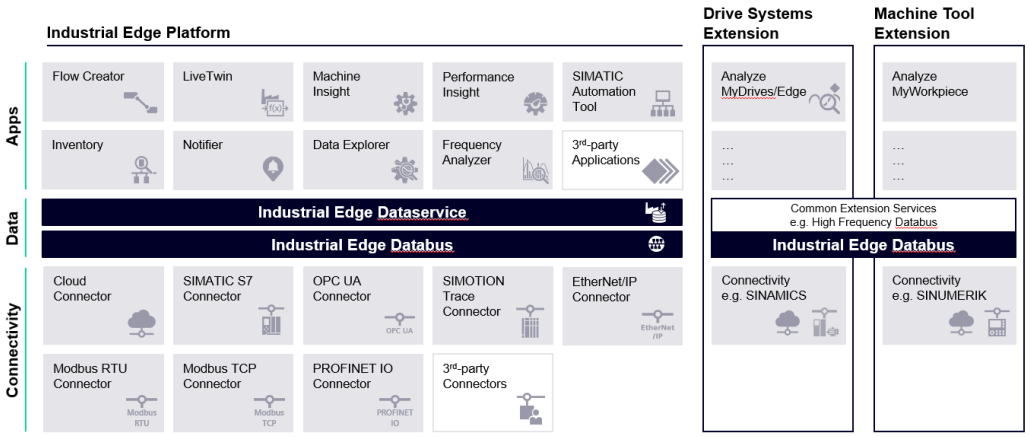


Figura 3-5 Resumen de las apps de Siemens Industrial Edge en función de la funcionalidad. Fuente: Siemens Industry Image Database 4.11

3.2 Conceptos de redes con Siemens Industrial Edge

Redes de comunicación industriales en entornos de utilización de Industrial Edge

Un proxy (servidor) es una interfaz de comunicación en una red, que maneja la comunicación entre dos sistemas informáticos que se encuentran en distintas subredes. La tarea principal de un proxy es la recepción de una solicitud de cliente para el servidor. El proxy enrutará la solicitud al servidor. De esta manera no se establece una conexión directa entre el cliente y el servidor, pero controlado para solo permitir el acceso de los clientes a una "lista blanca" de servidores" en caso de que el cliente se encuentre dentro de la planta y para mejorar la experiencia de usuario de los clientes y mejorar la seguridad de las comunicaciones en caso de que sea el servidor el que se encuentre en planta.

De esta manera el servicio proxy puede trabajar en dos direcciones:

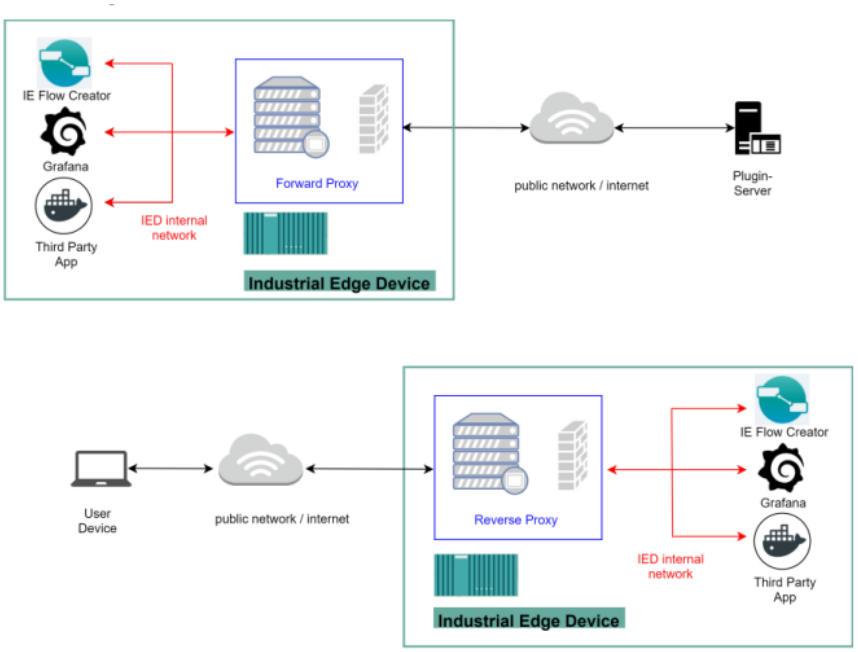


Figura 3-6. Opciones de arquitecturas de red IE. Fuente: Siemens Industry Image Database 4.11

• Proxy (“Foward Proxy”): El servidor proxy se coloca entre una red privada y una red/internet pública. Esta configuración ayuda a evitar que los clientes privados se vean totalmente expuestos a las redes públicas. Las solicitudes son tomadas por el proxy y redirigidas a las direcciones IP habilitadas del proxy para ser transferidas al destino. El proxy toma la respuesta a esta solicitud y la redirige al cliente. En Industrial Edge, el proxy de reenvío enruta toda la comunicación saliente de aplicaciones como cargar complementos o solicitar datos de la nube.

• Proxy inverso: para proteger el servidor web del acceso directo desde la red pública/internet, se puede conectar un servidor proxy inverso. Los clientes públicos no obtendrán acceso directo al servidor de destino. Las solicitudes se toman de los clientes y se pueden verificar mediante reglas de seguridad. La solicitud se puede enviar a una instancia de servidor después de las comprobaciones. En Industrial Edge, el proxy inverso maneja las solicitudes entrantes, como abrir la página de IE Flow Creator o un tablero de Grafana para la visualización de datos del dispositivo local desde una ubicación fuera de planta.

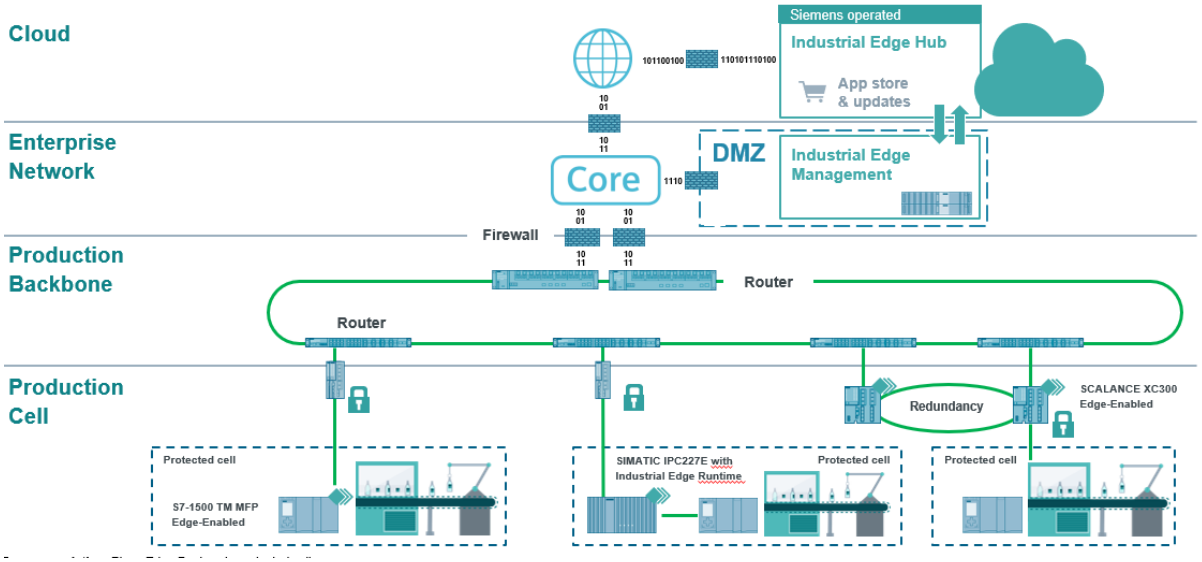


Figura 3-7 Arquitectura típica de red para el despliegue de Industrial Edge en planta. Fuente: Siemens Industry Image Database 4.11

Ya dentro de la propia planta de proceso real, un escenario de red típico es que los componentes de Industrial Edge se coloquen en redes separadas. La Figura anterior muestra un escenario típico de segmentación de red de planta a nivel OT. En la red de planta se configuran los Industrial Edge Devices. El sistema Industrial Edge Management se coloca en la DMZ estando de esta manera accesible a los dispositivos Edge locales de las diferentes subredes, pero controlando los accesos desde el exterior para que solo el IEM pueda acceder a la red pública directamente.
La comunicación normalmente se realiza a través de una pasarela **NAT** (router). Es importante indicar que el IEM y el IED deben estar sincronizados horariamente por lo que el protocolo **NTP** debe estár habilitado en los routers para permitir la sincronización horaria entre los equipos Edge.

Redes de datos y de almacenamiento internas de dispositivos Industrial Edge
En un dispositivo industrial Edge existe una configuración de red interna ya preconfigurada. Están disponibles un servicio, Industrial Edge Databus, para proporcionar una interfaz de comunicación entre aplicaciones y el Industrial Edge Data Service para operar como base de datos interna. Ambos se muestran en la figura siguiente.

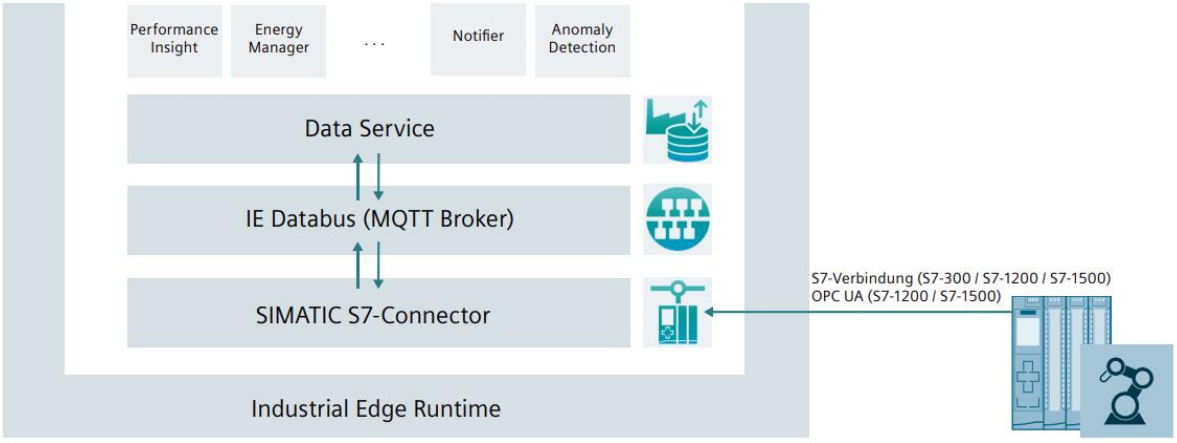


Figura 3-8. Arquitectura de red interna IE. Fuente: Siemens Industry Image Database 4.11

- Industrial Edge Data Service: Con la aplicación Data Service, se conectan otras aplicaciones, como puede ser la app de Performance Insight, al IE Databus vía MQTT Broker o si es Unified Comfort Panel vía Open Pipe. En el Servicio de datos, se podrán agrupar los datos y guardarlos durante un tiempo determinado, limitado en el caso del PC por la licencia de uso y en el caso del Panel por la memoria disponible (4Gb).
  El IE Data Service se comporta como una app y dispone de una interfaz gráfica para gestionar las variables que se almacenan y dejarlas disponibles para el resto de las aplicaciones.[34]

- Industrial Edge Databus Industrial Edge Databus proporciona la interfaz entre las aplicaciones del sistema y sus propias aplicaciones. Industrial Edge Databus proporciona un patrón de publicación-suscripción basado en el protocolo MQTT. Se pueden configurar canales en los que desea publicar datos en otras aplicaciones. Otras aplicaciones se suscriben a estos canales definidos para consumir los datos.
  Dentro de esta red el "proxy-redirect" Se comporta como una red virtual interna predefinida en Docker para el IED. Con esta red predefinida, es posible comunicarse con las aplicaciones del sistema uniéndose a esta red. Hay que tener en cuenta que la aplicación no creará esta red, ya que es una red virtual preexistente en el IED. Por lo tanto, debemos marcar esta red como externa en el archivo docker-compose de la aplicación si necesitamos comunicarnos con las aplicaciones del sistema. Esta red solo debe usarse para la comunicación con las aplicaciones del sistema a través de Industrial Edge Databus.

[34] https://support.industry.siemens.com/cs/es/en/view/109781417/149110787083

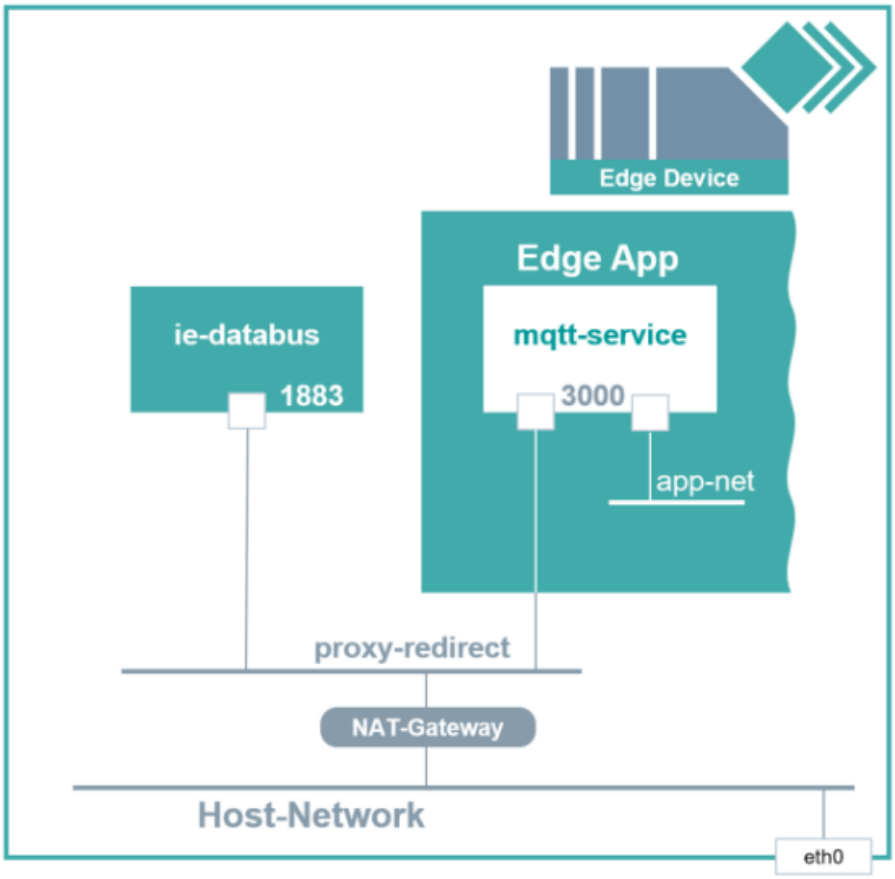


Figura 3-9 Diseño de red dentro de un dispositivo Siemens Industrial Edge. Fuente: Industrial Edge - App Developer Guide V1.2.1 [35]

Redes en el nivel de las aplicaciones

De forma predeterminada, el contenedor Docker se ejecuta en una red privada y Docker se ocupa de su gestión. Estos contenedores pueden comunicarse entre sí o conectarse a otros servicios que no sean de Docker. El subsistema de red de Docker utiliza controladores. A continuación, vamos a repasar algunos controladores existentes disponibles de forma predeterminada que se pueden usar con contenedores:

- Bridge: Es el controlador de red predeterminado. Este tipo se creará automáticamente si no se especifica ningún controlador. Las redes bridge generalmente se usan cuando las aplicaciones se ejecutan en contenedores independientes que necesitan comunicarse entre sí. La dirección IP predeterminada está en el rango 172.17.0.0/16.

[35] https://support.industry.siemens.com/cs/es/en/view/109795865

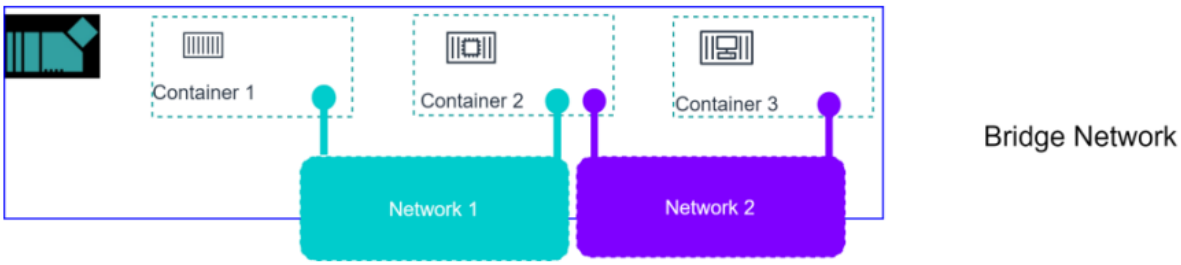


Figura 3-10. Bridge entre aplicaciones IE. Fuente: Industrial Edge - App Developer Guide V1.2.1

- Host: este controlador elimina el aislamiento de red entre el contenedor y el host de Docker. La red del anfitrión (dispositivo Edge) se utiliza directamente. No hay asignación de puertos y los del host se utilizan directamente. Normalmente, no es necesario utilizar la red host en absoluto en la aplicación Industrial Edge además de que existen algunas excepciones.

- Ninguno: En este caso, todas las redes están deshabilitadas. Dicho contenedor ser un contenedor cerrado, porque funciona aislado en el host.

Para administrar la funcionalidad de la red Docker, se utiliza el comando 'docker network'. Estos comandos afectarán al sistema Docker y no a los contenedores individuales. En general, la configuración de la configuración de red para el contenedor se puede realizar de forma manual o automática mediante docker-compose. Otro aspecto importante de Docker cuando se usan redes bridge es la publicación de los puertos del contenedor al exponer los puertos. Esto permitirá que los puertos de Docker sean accesibles por servicios desde fuera del sistema host. La exposición de puertos a otros contenedores dentro de la misma red bridge se realiza mediante exposición. La exposición es una instrucción para el archivo docker-compose. Para utilizarla se debe llamar a la instrucción EXPOSE dentro del fichero de creación de imágenes de Docker, Dockerfile. Un ejemplo de la regla EXPOSE en un Dockerfile tendría este aspecto:

Expose 8080
Expose 80

En este ejemplo, se exponen dos puertos (80 y 8080). Deben evitarse los puertos expuestos no asignados en las aplicaciones Industrial Edge. Si se desea especificar el protocolo del puerto, se agrega como se hace para el puerto 8080 en el ejemplo de ña figura inferior.
EXPOSE es un mecanismo de documentación que proporciona información sobre qué puertos entrantes iniciales proporcionarán servicios. Por defecto, la instrucción EXPOSE no expone los puertos del contenedor para que sean accesibles desde el host. Solo hace que los puertos indicados estén disponibles para la interacción entre contenedores. Para publicar los puertos expuestos, hay dos formas de usar el indicador -P y -p en tiempo de ejecución. El indicador -P publica todos los puertos expuestos en puertos aleatorios en las interfaces del host y es la forma abreviada de -publish-all. El mecanismo de mapeo automático evita posibles conflictos de mapeo de puertos. El indicador -p publica los puertos específicos de un contenedor en el host de Docker. Permite mapear el puerto de un contenedor o un rango de puertos al host de forma explícita. El formato para el indicador -p es “ip:hostPort:containerPort”. Las referencias de ip y hostPort se pueden omitir. Aquí hay algunos ejemplos para usar el indicador -p:

Flag Value.	Description
`-p 8080:80`	Bind container´s TCP port 80 to host´s port 8080
`-p 192.0.2.1:8080:80`	Bind container´s TCP port 80 to host´s port 8080 for connections to host IP 192.0.2.1. By default, Docker binds published containers to 0.0.0.0 IP address, which matches any IP address on the system
`-p 2346-2348:2346-2348`	Specify hostPort and containerPort as a range of ports. Note: Number of container ports specified in the range should be equivalent to the number of host ports specified

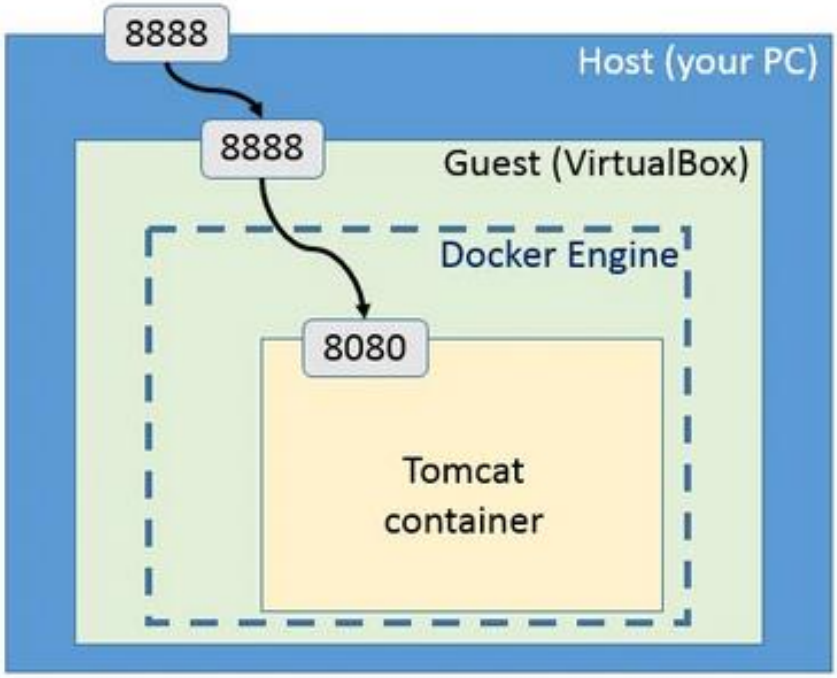


Figura 3-11Una imagen de ejemplo del funcionamiento de la exposición de puertos. Fuente: Colaboratorio.net[36]

[36] https://colaboratorio.net/davidochobits/sysadmin/2017/redes-en-docker-exposicion-de-puertos/

4. Gestión de Apps: Industrial Edge Management

4.1 Visión general

La gestión de las aplicaciones y los dispositivos Industrial Edge se puede hacer tanto de forma centralizada como local al propio dispositivo, vamos a ver las características principales de cada opción:

Características de la gestión local:

- Instalación y desinstalación de aplicaciones Edge en el dispositivo
- Actualización de aplicaciones Edge
- Inicio y detención de aplicaciones
- Información de diagnóstico de todas las aplicaciones Edge actualmente en ejecución
- Acceso remoto o local a través del servidor web

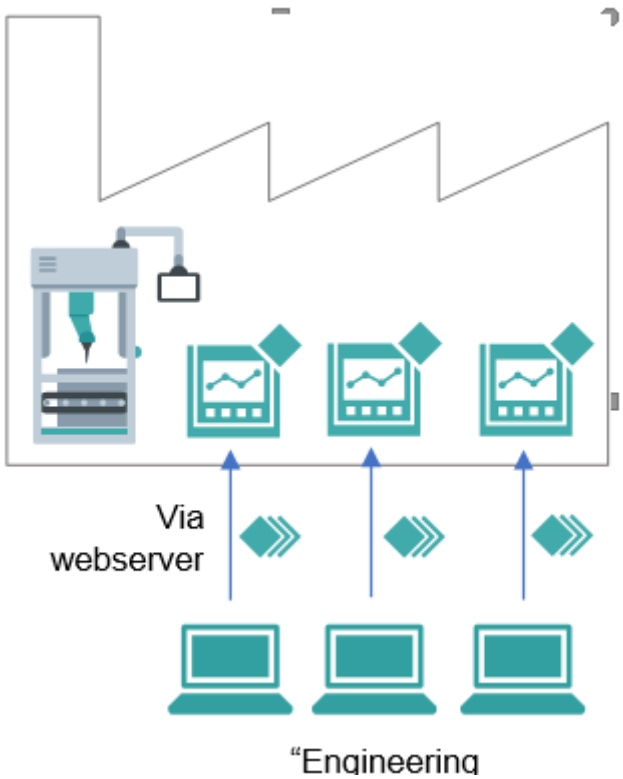


Figura 4-1Gestión local de los IED. Fuente: Siemens Industry Image Database 4.11

Características de la gestión centralizada:

- Gestión masiva.
- Información de diagnóstico
- Gestión de versiones de aplicaciones
- Instalación y desinstalación de aplicaciones Edge (remota)
- Iniciar y detener aplicaciones Edge (remoto)
- Diagnóstico de aplicaciones (remoto)
- Implementación de parches de seguridad

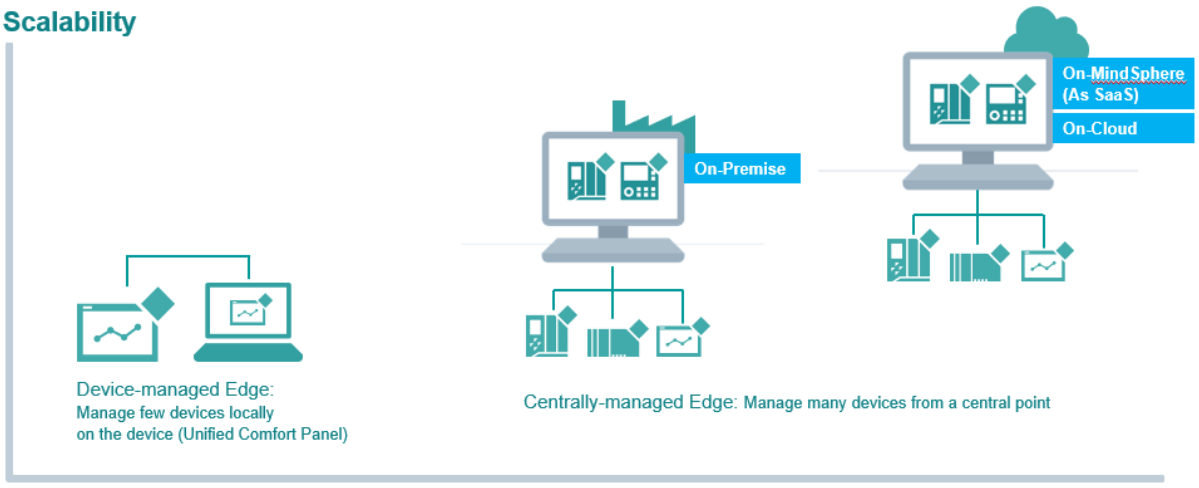


Figura 4-2 Los dispositivos Edge se pueden administrar de forma escalable. Fuente: Siemens Industry Image Database 4.11

Portfolio Siemens de dispositivos IE

Dispositivos Edge dedicados: Los dispositivos Edge dedicados están diseñados únicamente para la computación Edge.

- Hardware IPC con Industrial Edge Runtime preinstalado
- Plataformas potentes y escalables con el único propósito de ejecutar Edge Apps
- Conectividad integrada a los sistemas de automatización, así como conectividad al Edge Management System centralizado
- Edge Runtime se podrá ejecutar como una MV junto a otro software basado en PC

Dispositivos habilitados para Edge: Se trata de dispositivos hardware de automatización industrial con una función principal que no es Edge Computing (por ejemplo, HMI, PLC), pero con preparación para correr la runtime de Edge.
Dicha funcionalidad se puede activar o desactivar por el usuario a conveniencia. El primer dispositivo habilitado para Edge que está disponible comercialmente son los Unified Comfort Panels o en adelante UCP

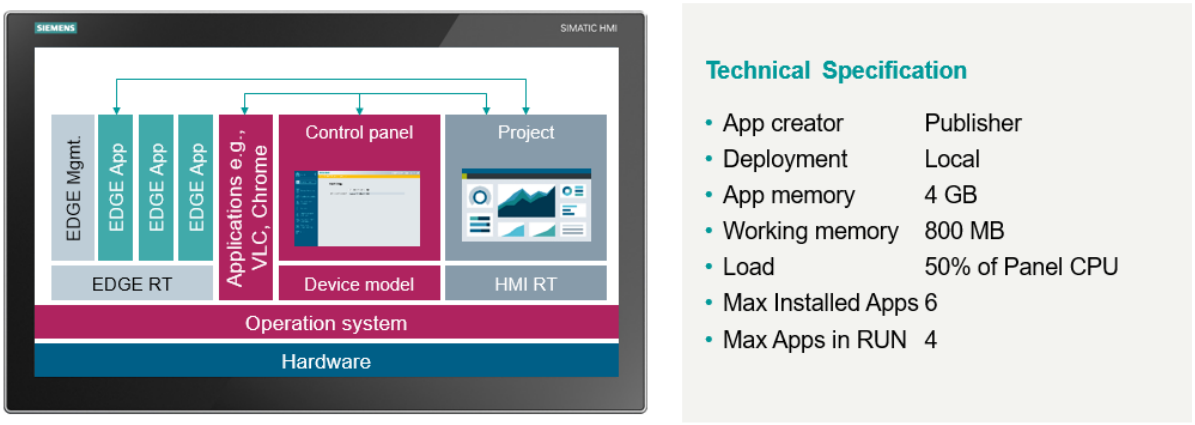


Figura 4-3 Recursos HW destinados para IE en un Unified Comfort Panel. Fuente: Siemens Industry Image Database 4.11

4.2 Inicializar el IEM

En el presente TFG vamos a utilizar la funcionalidad de actuar como dispositivo Industrial Edge de los Unified Comfort Panel, en adelante UCP. Para poder utilizarla dicha funcionalidad en un entorno de ensayo no es necesaria licencia, y eso es lo que nos ha decantado por esta opción dado que nos va a permitir gestionar las apps en local y probarlas sin necesidad de mas licencias o hardware adicional que el propio panel.

Requisitos previos:
Para poder trabajar con la funcionalidad IE del UCP es necesario haber cargado un proyecto al panel desde TIA Portal que tenga configurado al menos un usuario con permisos de administrador.

Procedimiento de activación:

1. Encendemos la fuente de alimentación de 24 V. del Unified Comfort Panel.
2. Esperamos a que arranque y vamos al Panel de control.

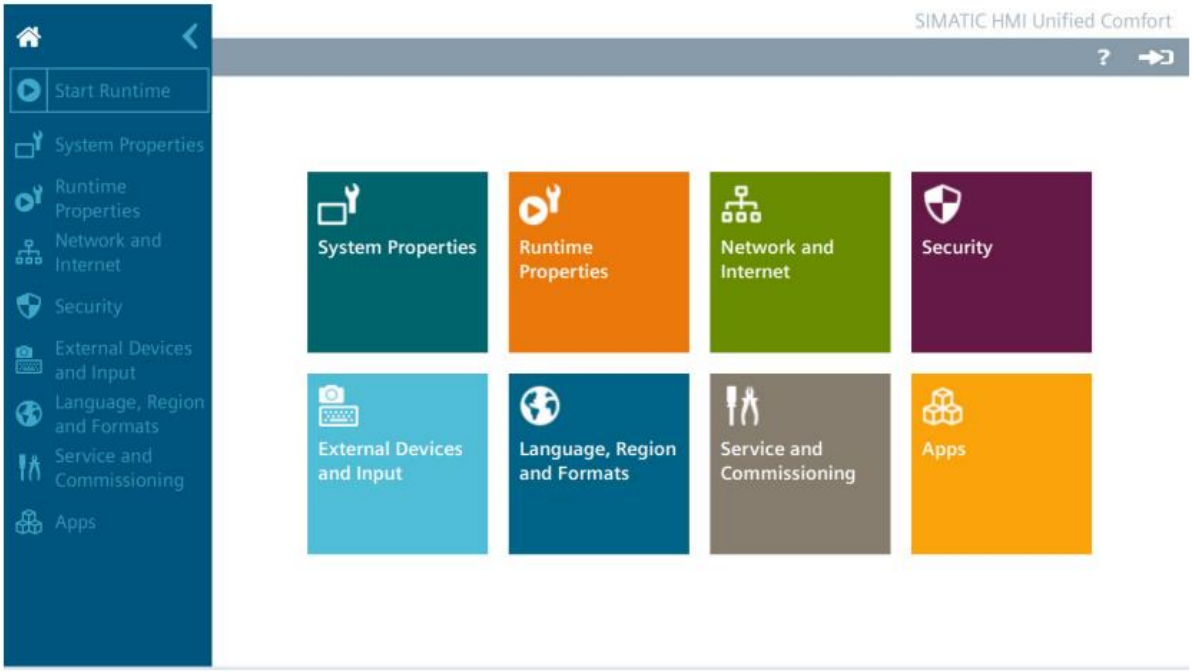


Figura 4-4. Panel de control de un Unified Comfort Panel.

3. Seleccionamos "Aplicaciones" > "SIMATIC Edge".

4. Activamos la opción "Habilitar SIMATIC Edge" y esperamos hasta que se active el botón “Open Edge management".

5. Hacemos clic en el botón " Open Edge management ". La primera vez y si no hemos gestionado los certificados del servidor web aparecerá una advertencia de riesgo de seguridad, podemos obviarla haciendo clic en "Avanzado" > "Aceptar el riesgo y continuar".
Entonces si es la primera vez que nos conectamos se mostrará el cuadro de diálogo "Activate Edge Device".

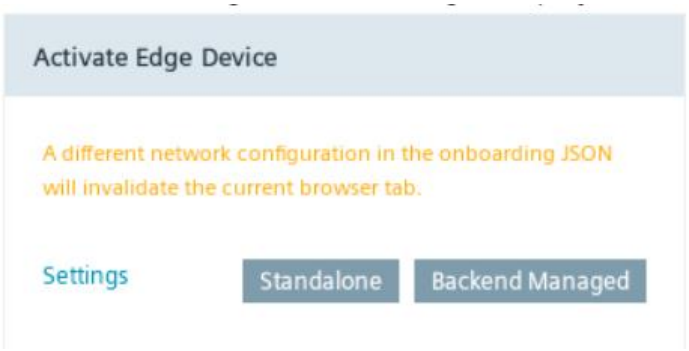


Figura 4-5. Captura de imagen del IEM

Aquí seleccionaremos si queremos gestionar el dispositivo Edge de forma local ("Standalone") o centralizada (Backen Managed). Ojo está selección queda registrada en el dispositivo y ya no es posible deshacerla a menos que le hagamos al equipo un reinicio a ajustes de fábrica o le actualicemos la imagen de sistema operativo.

6. A continuación se abrirá el gestor del IEM y deberemos iniciar sesión para comenzar a trabajar. También nos permitirá gestionar los certificados para sucesivas conexiones desde el mismo navegador.
Para esto utilizaremos un usuario con permisos: "HMI Adminsitrator"
Nota: Con la V17 upd 2 el nombre de usuario no debe tener caracteres en letra mayúscula.

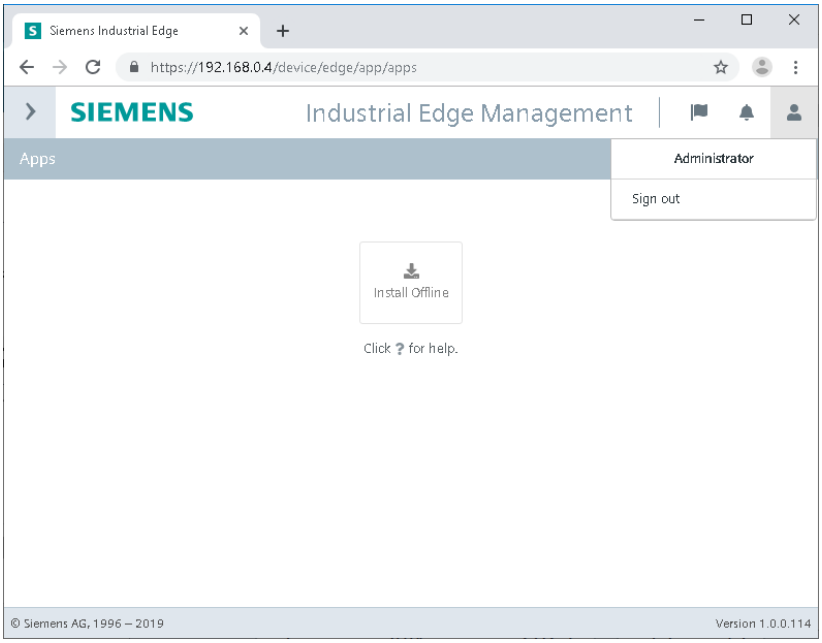


Figura 4-6 Captura de imagen del IEM

Este gestor local estará ya accesible desde el interfaz táctil del mismo equipo así como remotamente vía navegador web accediendo a la IP de la UCP.

4.3 Servidor web local del IEM

Vamos a repasar a continuación los principales elementos de configuración y manejo que nos vamos a encontrar en el servidor web local del IEM.

Página de inicio:

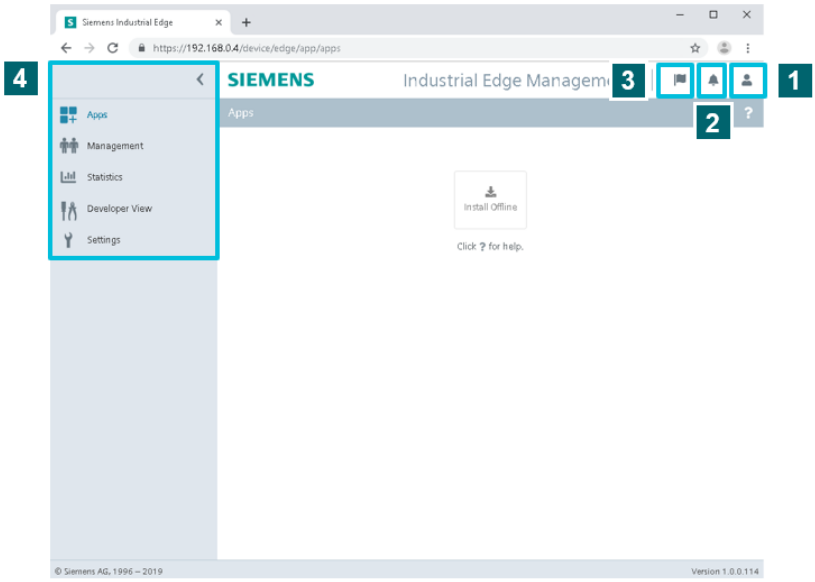


Figura 4-7 Captura de imagen del IEM

1. Ver usuario actual o cerrar sesión
2. Mostrar/Ocultar vista de eventos recientes
3. Verificar las tareas en ejecución y finalizadas
4. Árbol de navegación

Árbol de navegación:

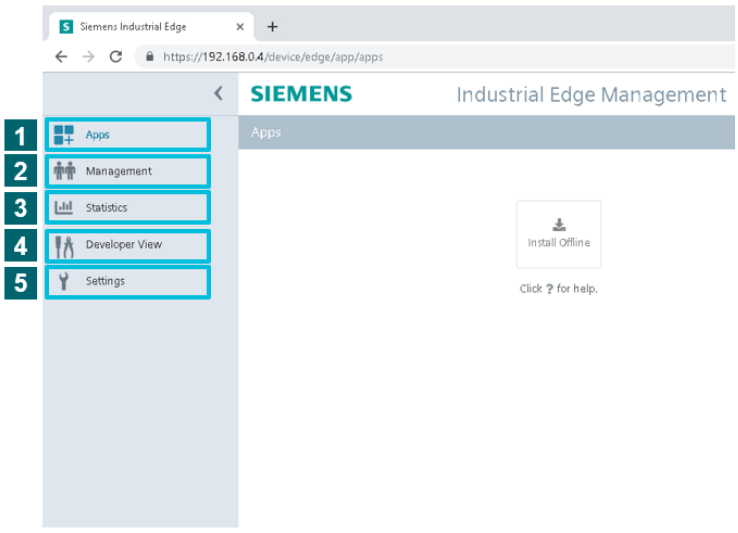


Figura 4-8 Captura de imagen del IEM

1. Descripción general de las aplicaciones instaladas y realizar acciones con ellas (instalar/desinstalar).
2. Configuración y rendimiento de las aplicaciones.
3. Configuración y rendimiento del dispositivo edge.
4. Vista desarrolaldor
5. Cambiar la configuración del sistema, p. Modo Proxy o Desarrollador

Instalación de Apps:

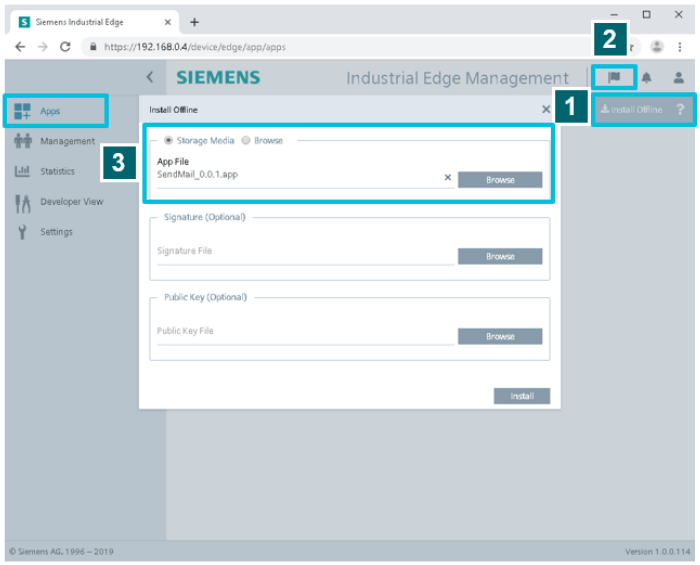


Figura 4-9 Captura de imagen del IEM

1. Abrir diálogo de instalación
2. Instalar la aplicación desde un medio de almacenamiento local (USB o tarjeta SD) o explorar el sistema de archivos de la conexión remota y seleccione el archivo .app
3. Comprobación del estado actual de la instalación en la vista de tareas

Apps:

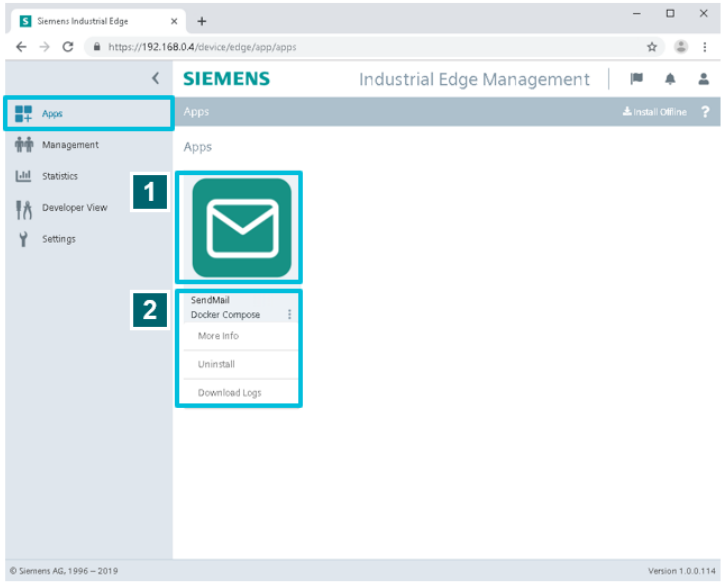


Figura 4-10 Captura de imagen del IEM

1. Abrir la interfaz web de la aplicación en una nueva pestaña del navegador presionando el icono. La interfaz se direcciona a través de https://IP-Adress:CustomPort. Siempre que la app tenga internamente un servidor web y una aplicación de escritorio accesible.
2. Seleccionar acciones adicionales como "Más información", "Desinstalar" o "Descargar registros"

Management:

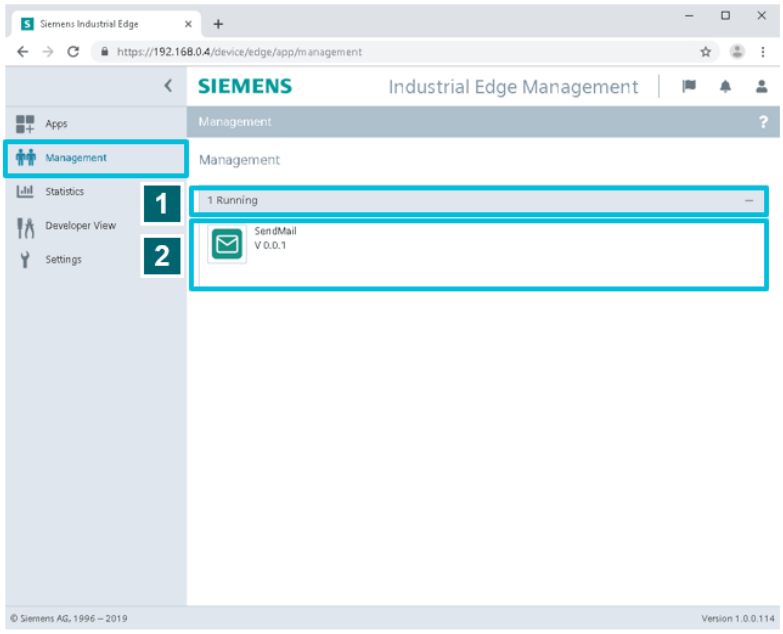


Figura 4-11 Captura de imagen del IEM

1. Supervisar las aplicaciones con su modo de funcionamiento actual, p. “En ejecución” o “Detenida”.
2. Abrir vista detallada pulsando sobre la aplicación

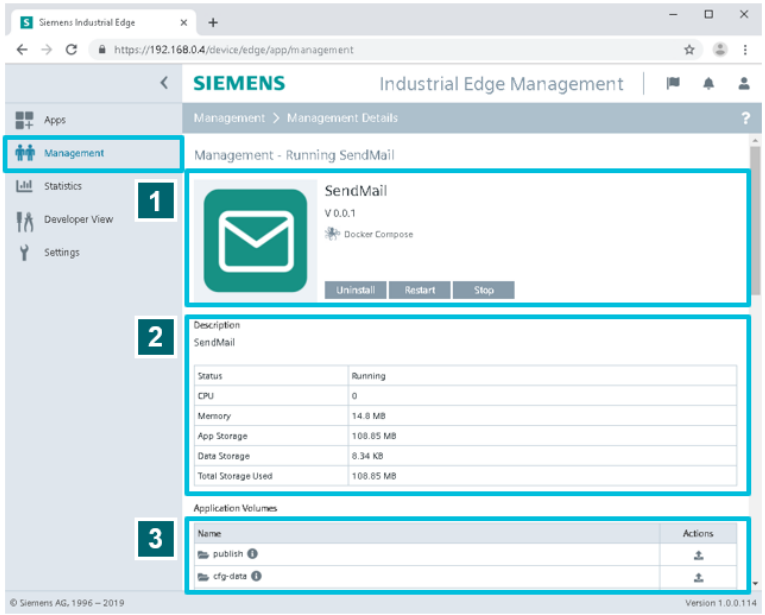


Figura 4-12 Captura de imagen del IEM

1. “Desinstalar”, “Iniciar/Reiniciar” o “Detener” la aplicación seleccionada.
2. Compruebar la configuración de rendimiento, por ejemplo, memoria usada o almacenamiento de datos.
3. Ver los volúmenes de la aplicación y cargar, descargar o eliminar archivos si es necesario.

Statistics:

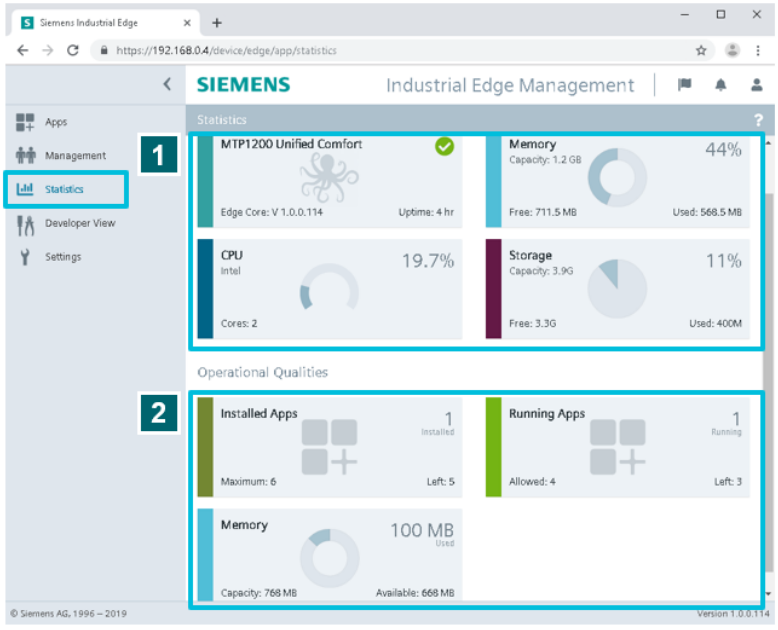


Figura 4-13 Captura de imagen del IEM

1. Comprobar el rendimiento general del dispositivo, por ejemplo el consumo de memoria y CPU o almacenamiento disponible
2. Verificar las estadísticas de la aplicación, por ejemplo aplicaciones instaladas y en ejecución o memoria usada en total

Developer View:

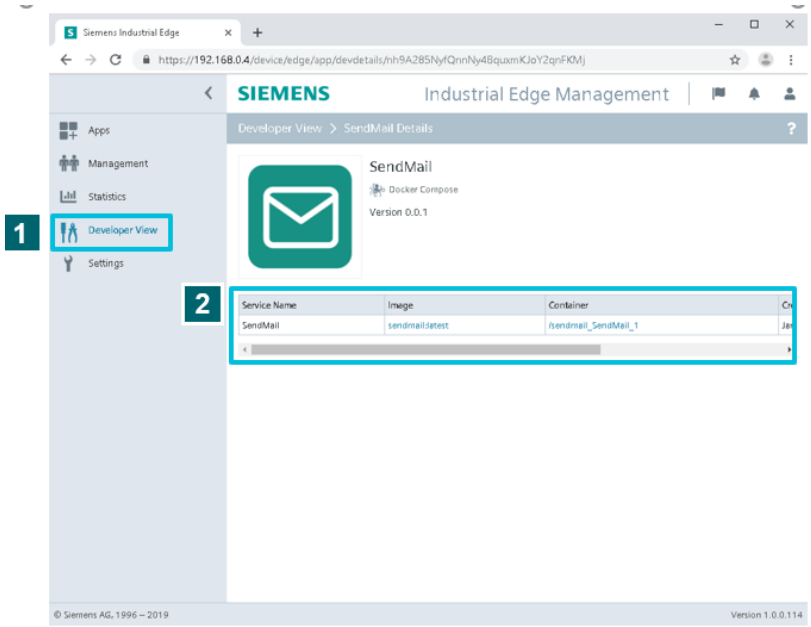


Figura 4-14 Captura de imagen del IEM

1. Activar "Vista de desarrollador" en Configuración
2. Ver información detallada de la imagen de la aplicación y el contenedor para verificar, por ejemplo, puertos expuestos y sistemas de archivos

Settings:

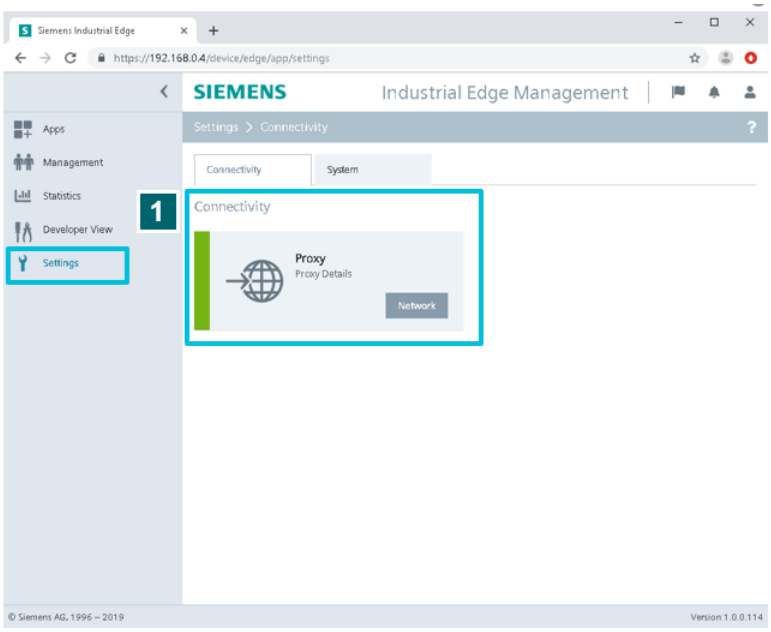


Figura 4-15 Captura de imagen del IEM

1. “Connectivity”: Activar y ajustar la configuración del "Servidor proxy" para Edge Management

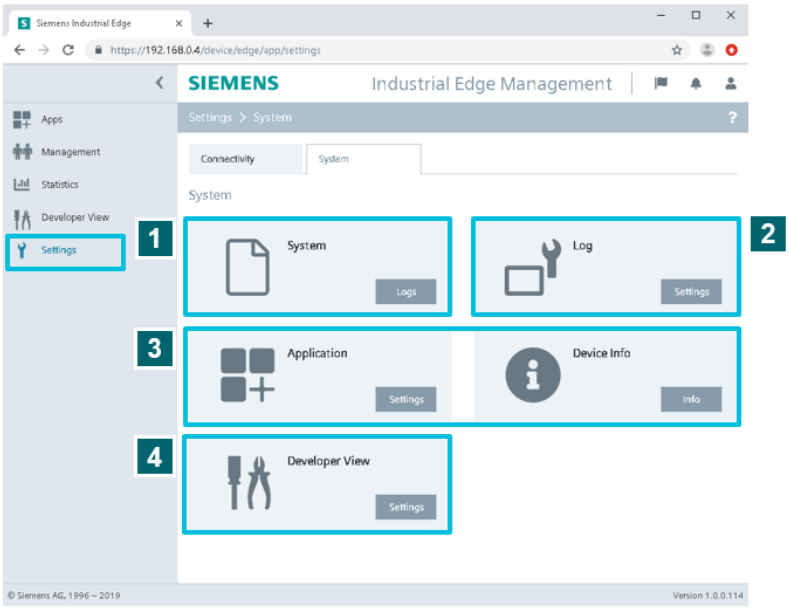


Figura 4-16 Captura de imagen del IEM

System:

1. Descargar registros del sistema Edge
2. Permitir que la aplicación Edge inicie sesión en la memoria interna
3. Verifique la configuración de la aplicación y la información del dispositivo
4. Activar vista de desarrollador

4.4 Gestión centralizada IEM

Acceder a Industrial Edge Hub

Si se va a requerir una gestión centralizada, porque, por ejemplo, vamos a querer gestionar multitud de dispositivos o vamos a trabajar con Apps del "Market Place" de Siemens entonces es necesario obtener el acceso al IE-Hub mediante el pago de una licencia de uso a través de Siemens Industry Mall. No ha sido nuestro caso como comentamos anteriormente, ya que para entornos de prueba y prototipos la gestión local de apps desarrolladas por el usuario es gratuito y está habiltado en las UCPs.

Volviendo a la gestión centralizada, para adquirir la licencia es necesaria una cuenta para el Industry Mall.

Una vez adquirida la licencia vía registro al Industrial Edge Hub (IE-Hub), podemos ir al IE-Hub abriendo este enlace en su navegador: https://iehub.eu1-alpha.edge.siemens.cloud/

Iniciaríamos sesión con las credenciales proporcionadas. Después de iniciar sesión correctamente, debería verse la página de descripción general de IE-Hub.

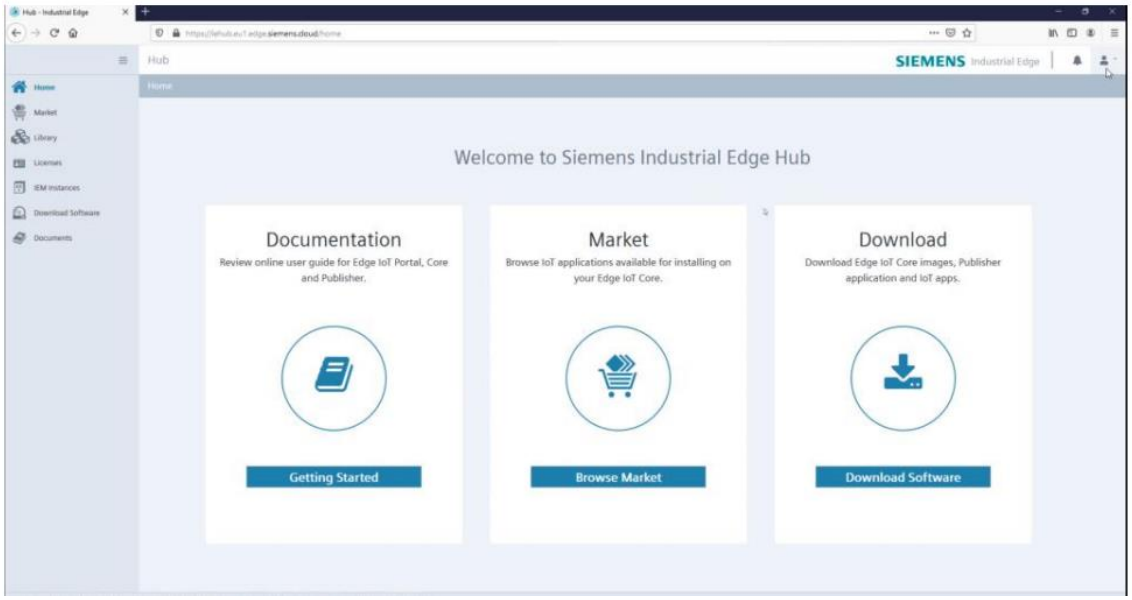


Figura 4-17 Página de inicio de Industrial Edge Hub después de iniciar sesión

Descargar paquetes requeridos

En IE Hub podemos obtener el archivo ISO para configurar el IEM. Esto lo vamos a necesitar cuando querramos gestionar múltiples dispositivos IE de planta desde un único PC y no de forma local uno a uno. Lo mismo se aplica al software Industrial Edge App Publisher. Como se muestra en la Figura, el IEAP está disponible para Windows y Linux (Ubuntu), así como la interfaz de línea de comandos (CLI) para ambos sistemas operativos.

Todos estos productos de software se pueden encontrar en "Descargar software" en las entradas del menú.
En la Figura 3-3 puede ver toda la documentación relativa a las aplicaciones del ecosistema

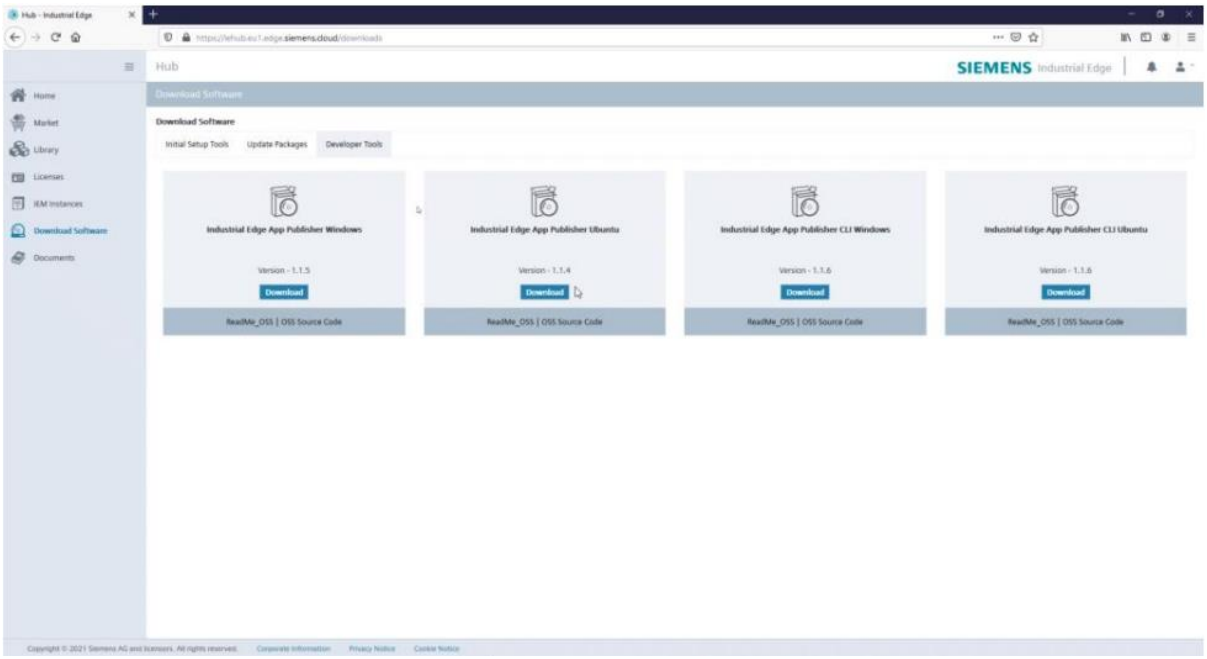
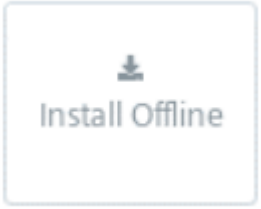


Figura 4-18 Página de descarga de Industrial Edge App Publisher en IE Hub

4.5 Instalación de una app

El siguiente procedimiento es de aplicación solo a IEM local y describe cómo instalaríamos una aplicación que se creó con Industrial Edge Publisher y se almacena en un medio de almacenamiento externo o en una unidad de red a la que pueda acceder el dispositivo Edge.

1. Si aún no hay una aplicación instalada, haremos clic en "Instalar sin conexión" en la ventana principal de "Aplicaciones".



Si ya tenemos aplicaciones instaladas, deberemos asegurarnos de que la aplicación que deseamos instalar no supera el número máximo de aplicaciones y el valor máximo de asignación de memoria permitido, esto lo podemos consultar en la ventana de estadísticas. Después de eso, haremos clic en "Instalar sin conexión" en la barra de título.

Se mostrará el cuadro de diálogo "Instalar sin conexión".

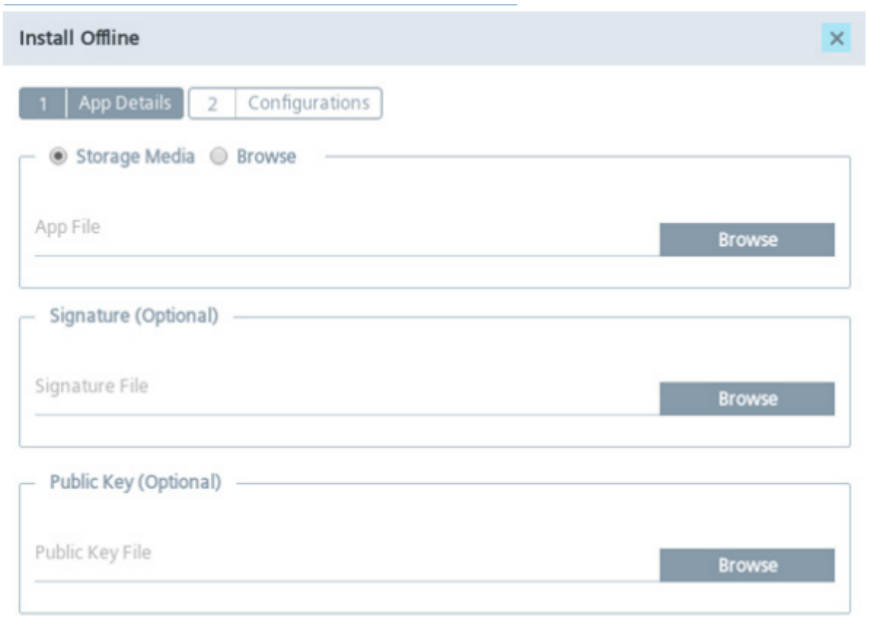


Figura 4-19 Captura del dialogo instalación apps IEM

2. Seleccionamos la opción "Medios de almacenamiento" para instalar la aplicación desde un medio de almacenamiento o la opción "Examinar" para instalar la aplicación desde una unidad de red. Hacemos clic en el botón "Examinar" para seleccionar el archivo .app desde la ubicación especificada.
3. Si es necesario, exploramos la firma de la aplicación que se generó durante la exportación del archivo de la aplicación en la sección "Firma".
4. Si fuera necesario, podemos introducir la clave pública en el campo de entrada "Clave pública. Tanto el archivo de firma como la clave pública verifican la integridad del archivo de la aplicación.
5. Por último hacemos clic en "Instalar" para iniciar el proceso de instalación.
6. Si vamos al icono "Tareas" en el área del encabezado podremos mostrar el progreso de la instalación.

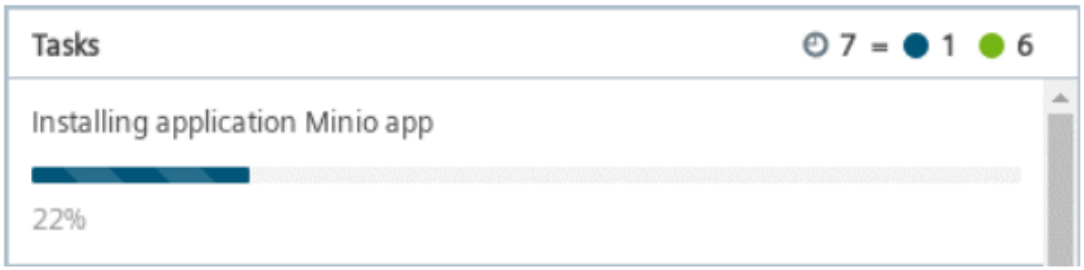


Figura 4-20 Captura de mensaje del proceso de instalación del IEM

Después de finalizada la instalación sin contratiempos, la aplicación se mostrará en la ventana principal de "Aplicaciones".

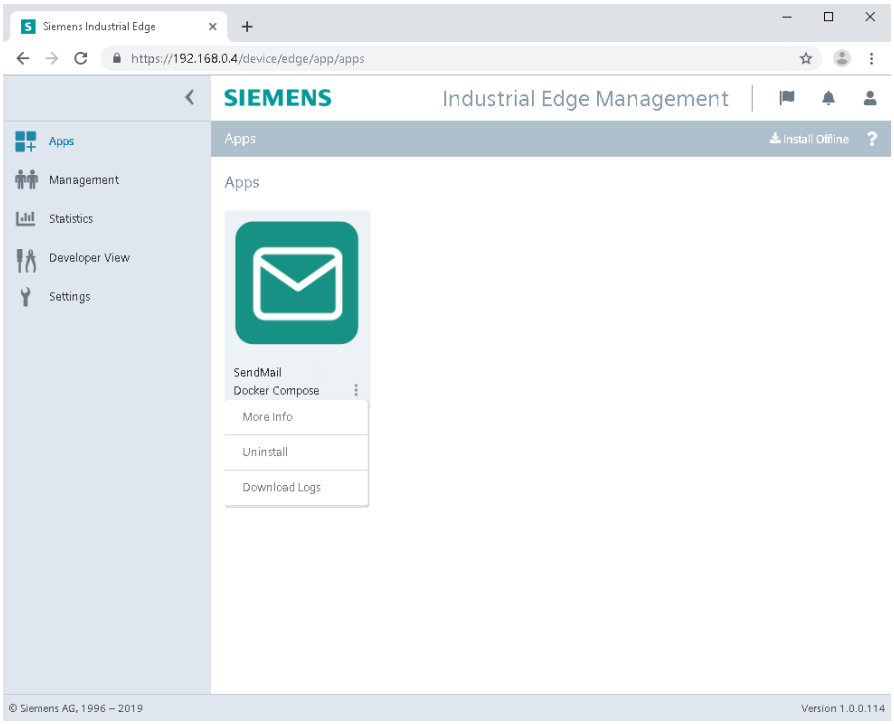


Figura 4-21 Captura IEM

5. Entorno de desarrollo de apps para Siemens IE

5.1 Visión general

El desarrollo de apps para correr en dispositivos IE consiste, básicamente, en desarrollar aplicaciones valiéndose de herramientas estándares IT, pero siempre teniendo en cuenta las peculiaridades y limitaciones del hardware sobre el que va a correr la aplicación y una vez desarrollada empaquetaremos la misma en un contenedor estandarizado Docker para a continuación, en base a la imagen de Docker generada crear la aplicación .app que pueda ejecutarse en un dispositivo IE.

Vamos a tratar a continuación acerca de las principales herramientas que vamos a necesitar para cada etapa del desarrollo, que resumiendo serían Docker, un IDE como Visual Estudio Code, un framework de desarrollo web UI como Flask y Siemens Industrial Edge Publisher para publicar las aplicaciones y que sean ejecutables por el dispositivo IE. así de como instalarlas y configurarlas para empezar a desarrollar.

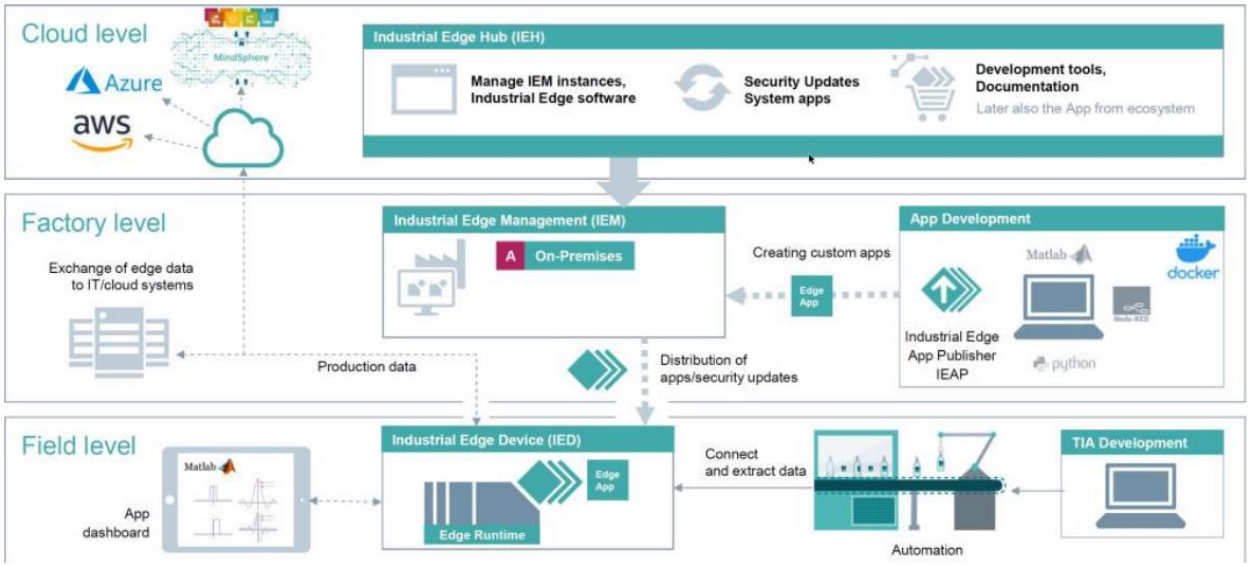


Figura 5-1 Arquitectura de la plataforma Siemens Industrial Edge. Fuente: Siemens Industry Image Database 4.11

5.2 Configuración del PC de desarrollo

La mejor forma de implementar el entorno de desarrollo para Siemens IE será mediante una máquina virtual que podrá estar basada en un sistema operativo de MS Windows o en una distribución de Linux como Ubuntu. En nuestro caso optaremos por Ubuntu versión 20.04.4 LTS.
Una vez creada la MV con Ubuntu esta se debe completar con un editor de código o IDE como puede ser Visual Studio Code y los paquetes de extensión necesarios para el desarrollo de aplicaciones (Python, Docker, Flask…). En el último paso, se realizará la instalación y configuración de Industrial Edge App Publisher (IEAP) y lo conectaremos a Docker Engine a través de la API de este último, para poder dar por finalizada la configuración del entorno de desarrollo.
En nuestro caso, como decíamos, para la creación del entorno de desarrollo se ha utilizado una máquina virtual Ubuntu en un reproductor VMware Workstation.
En la imagen inferior se puede apreciar un esquema del flujo de instalación de herramientas en el PC de desarrollo. La instalación de Visual Studio Code y Docker Engine se puede perfectamente invertir en orden como de hecho se ha hecho para este TFG.

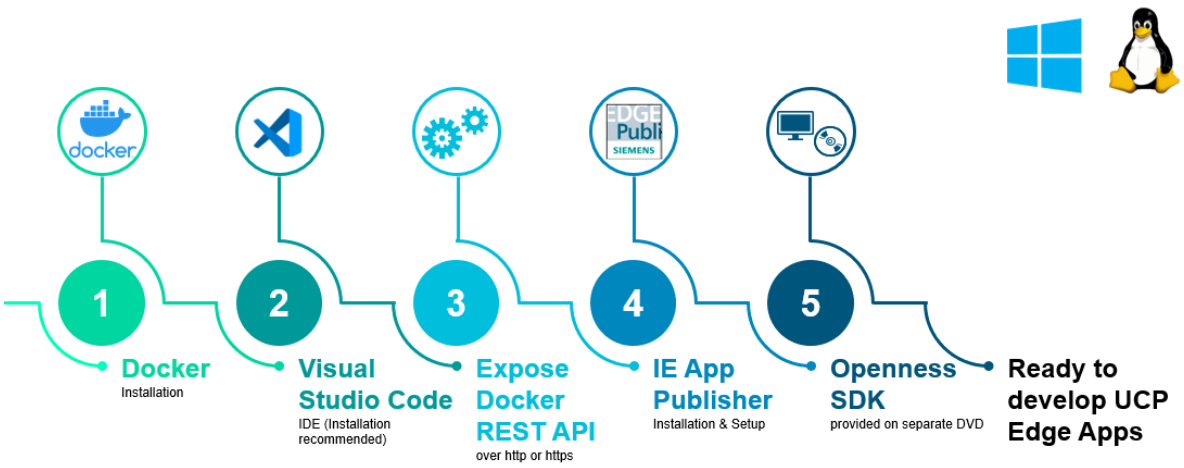


Figura 5-2 Orden de instalación en el PC de desarrollo de las herramientas SW necesarias. Fuente: Siemens Industry Image Database 4.11

Requisitos previos
Las especificaciones mínimas de hardware para una máquina de desarrollo con Ubuntu serían las siguientes:

- 2 núcleos de CPU
- 4 GB de RAM
- Al menos 25 GB de almacenamiento

Una de las razones de optar por Ubuntu fue debida a que la instalación del motor Docker Engine en MV con S.O. Windows va a ser problemática por las dependencias con Hyper-V de MS Windows.

Por otra parte, para el desarrollo de aplicaciones con Docker, es recomendable utilizar un repositorio privado de GitHub, pero en esta ocasión no lo hemos utilizado, descargando las imágenes necesarias para este TFG del repositorio público.

Crear el entorno de Desarrollo:
Con VMware Workstation va a ser muy sencillo preparar la MV con la distribución de Linux Ubuntu. Simplemente nos descargaremos la ISO de instalación desde la página oficial de Ubuntu: https://ubuntu.com/download/desktop
Una vez descargada la ISO de Ubuntu iniciamos la aplicación VMware Workstation en el PC de desarrollo y hacemos clic en 'Archivo > “Nueva máquina virtual' en la barra de menú de la estación de trabajo.
Entonces se abre un cuadro de diálogo de configuración. Los pasos más importantes se muestran en las dos figuras de abajo para lo que no se ve en las figuras dejamos los ajustes por defecto.

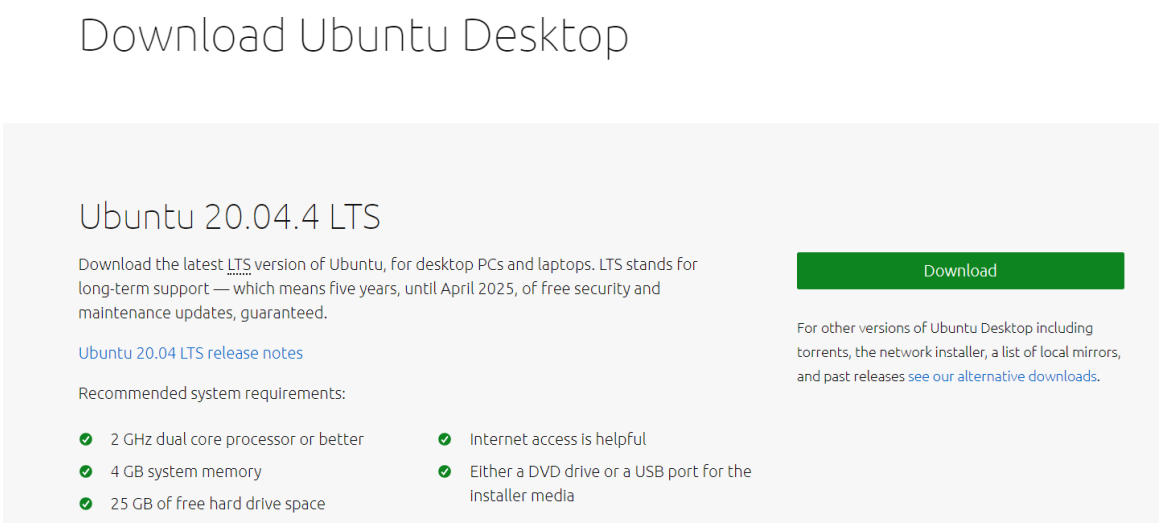


Figura 5-3 Descarga de la ISO de instalación de Ubuntu 20.04.4 LTS. Fuente: www.ubuntu.com

1. Seleccionamos “Custom” para obtener posibilidades de configuración avanzadas
2. Next
3. Seleccionamos el archivo ISO descargado de Ubuntu
4. Next
5. Indicar usuario y contraseña-
6. Next
7. Seleccionamos el número de procesadores
8. Next
9. Configuramos la RAM de la máquina virtual en 4 GB
10. Next
11. Configuramos el espacio en disco en 25 GB
12. Next

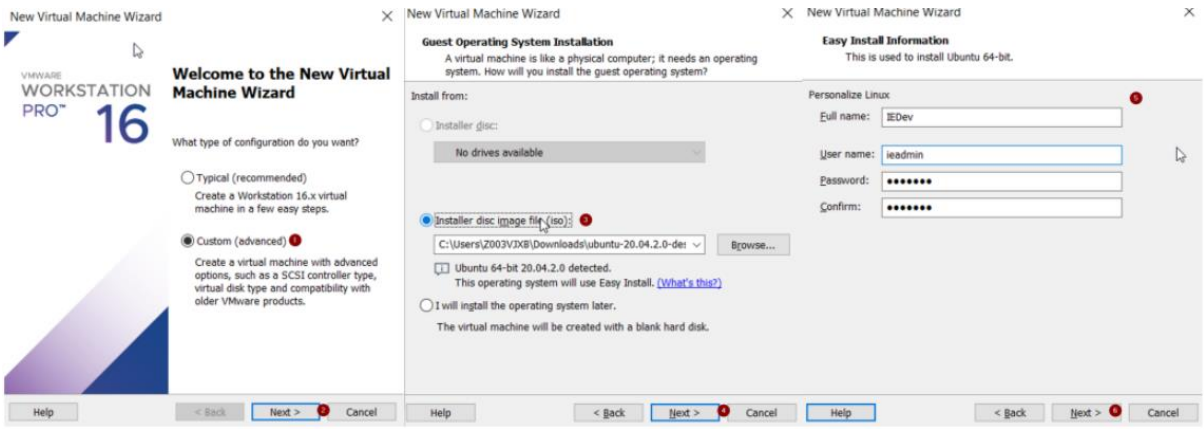


Figura 5-4 Crear una máquina virtual de desarrollo de Ubuntu - Captura 1

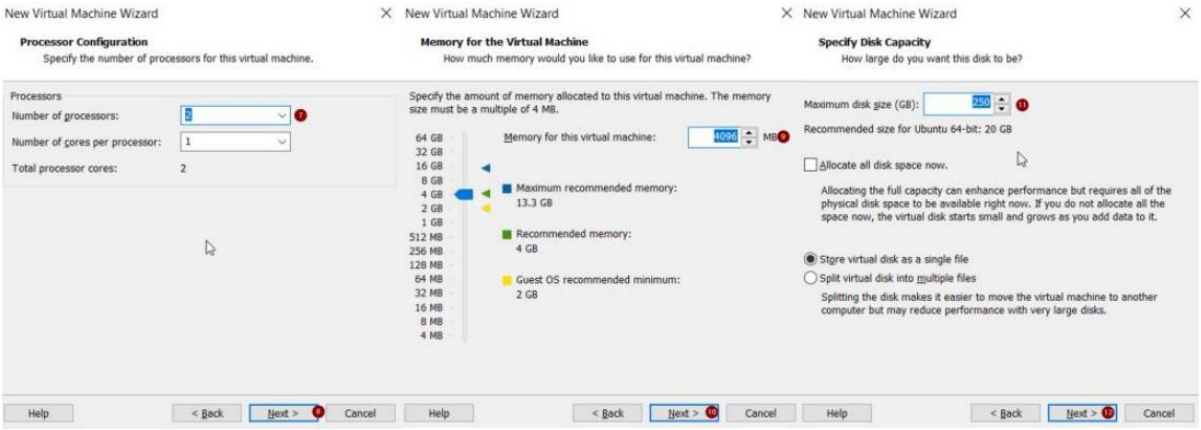


Figura 5-5 Crear una máquina virtual de desarrollo de Ubuntu - Captura 2

Después de completar el cuadro de diálogo de creación de la nueva máquina virtual, la misma arrancará y se completará la instalación de Ubuntu. La instalación finalizará automáticamente y se podrá iniciar sesión para realizar la última configuración del sistema operativo.
A continuación, personalizaremos el sistema operativo según nuestras necesidades, como los idiomas, Libre Office, Visual Studio Code…

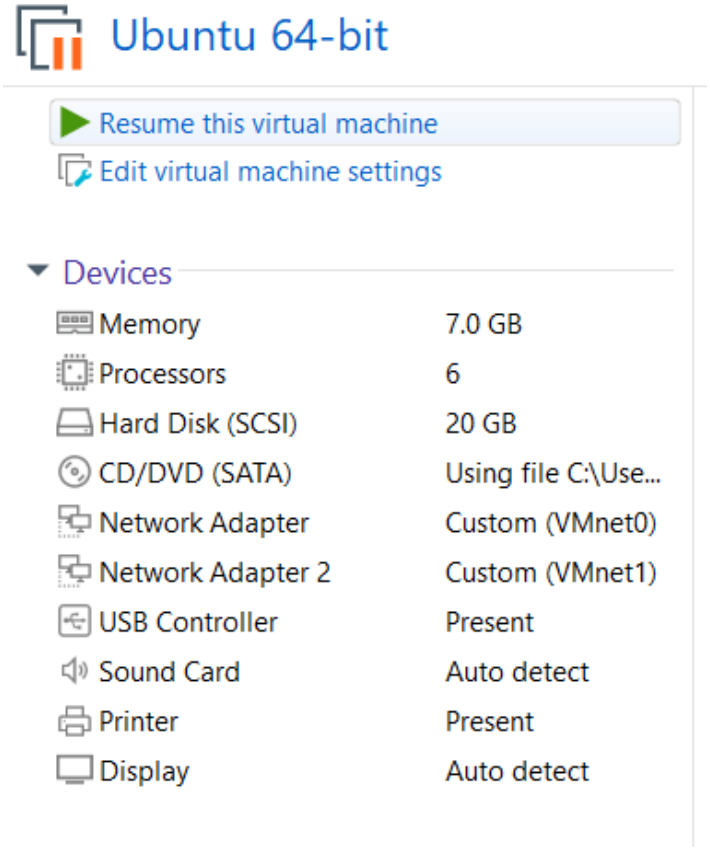


Figura 5-6 Características de la MV creada para desarrollo.

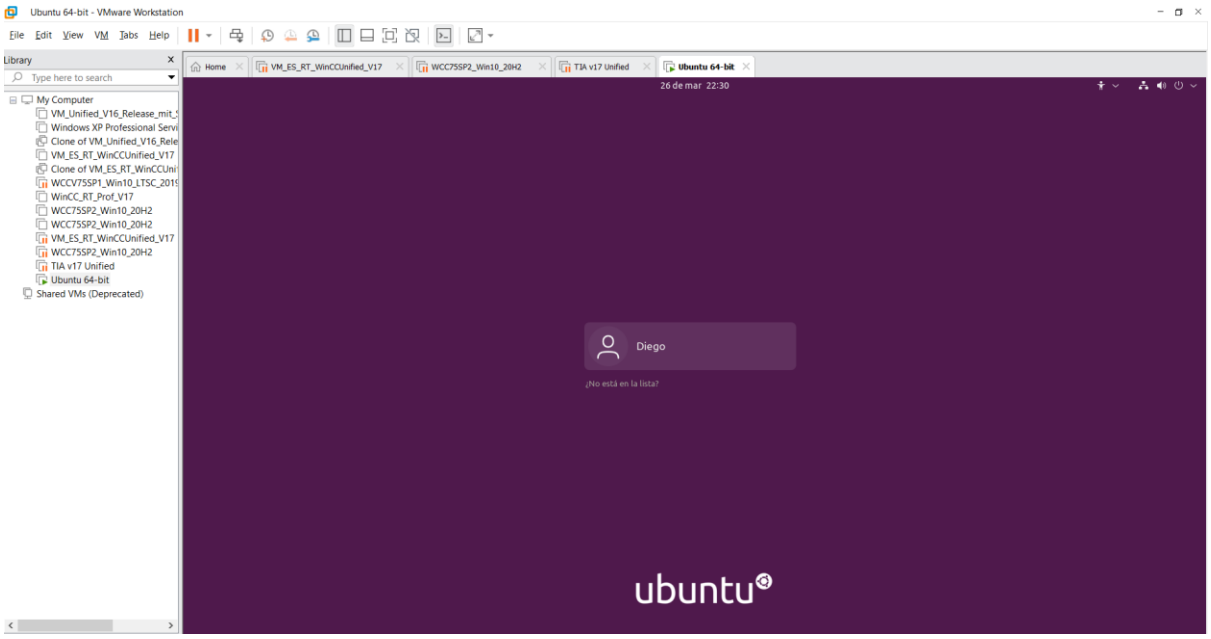


Figura 5-7 Pantalla de inicio de sesión de Ubuntu después de una instalación exitosa

Instalar los paquetes necesarios en el entorno de desarrollo

Después del paso anterior ya tenemos nuestra MV preparada para empezar a instalar la "caja de herramientas" (Framework) necesaria para empezar a desarrollar apps de Siemens Industrial Edge.

Como desarrollador, necesitaremos, en primer lugar, un entorno de desarrollo o IDE (Integrated Development Environment) que más nos convenga como puede ser Visual Studio Code. En nuestro caso hemos seleccionado VSC debido a que es gratuito, muy potente y flexible con multitud de extensiones para diferentes lenguajes y con gran cantidad de ejemplos y tuoriales en línea para permitirnos profundizar en su manejo.
Visual Studio Code lo descargaremos e instalaremos desde la propia "tienda de aplicaciones software" de Ubuntu (Ubuntu Software)

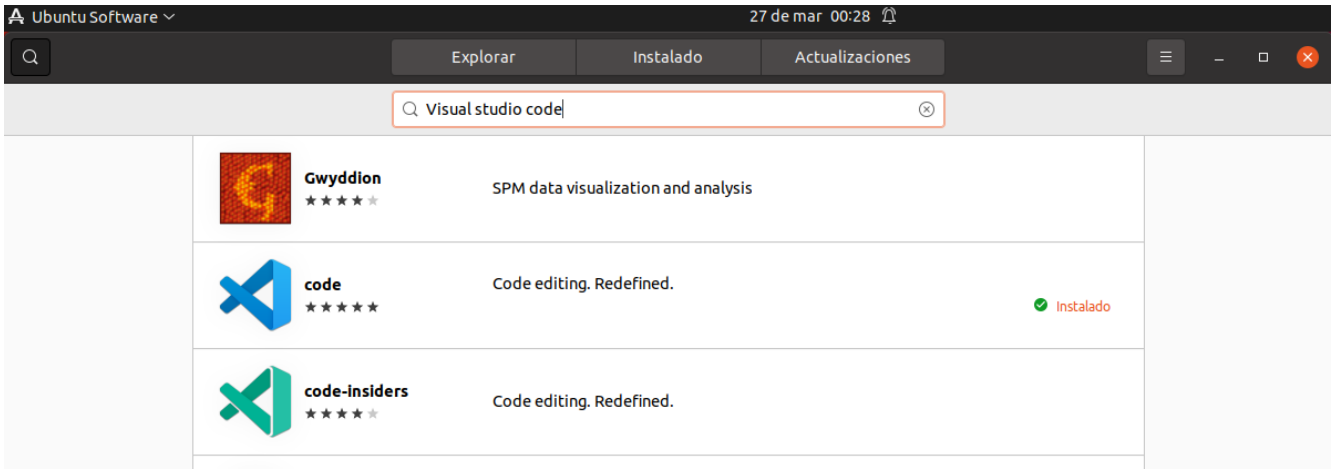


Figura 5-8 Marketplace aplicaciones Ubuntu

A continuación, instalaremos tres extensiones imprescindibles para casi cualquier app para IE que tengamos pensada desarrollar hoy en día:

1. **Python:** *Python es un lenguaje de programación de computadoras o, en otras palabras, un vocabulario y un conjunto de reglas gramaticales para instruir a una computadora para que realice tareas. Guido van Rossum lo nombró así por el programa de televisión de la BBC 'Monty Python's Flying Circus'.*[37]
2. **Jupyter Notebook:** *JupyterLab es el último entorno de desarrollo interactivo basado en la web para cuadernos, código y datos. Su interfaz flexible permite a los usuarios configurar y organizar flujos de trabajo en ciencia de datos, computación científica, periodismo computacional y aprendizaje automático. Un diseño modular invita a las extensiones para ampliar y enriquecer la funcionalidad.*[38]
3. **Docker:** *Docker ayuda a los desarrolladores a dar vida a sus ideas al conquistar la complejidad del desarrollo de aplicaciones. Simplificamos y aceleramos los flujos de trabajo de desarrollo con una canalización de desarrollo integrada y mediante la consolidación de los componentes de la aplicación. Usados activamente por millones de desarrolladores en todo el mundo, Docker Desktop y Docker Hub brindan simplicidad, agilidad y opciones inigualables.*[39]

[37] https://python.land/python-tutorial/what-is-python

[38] https://jupyter.org/

[39] https://www.docker.com/company/

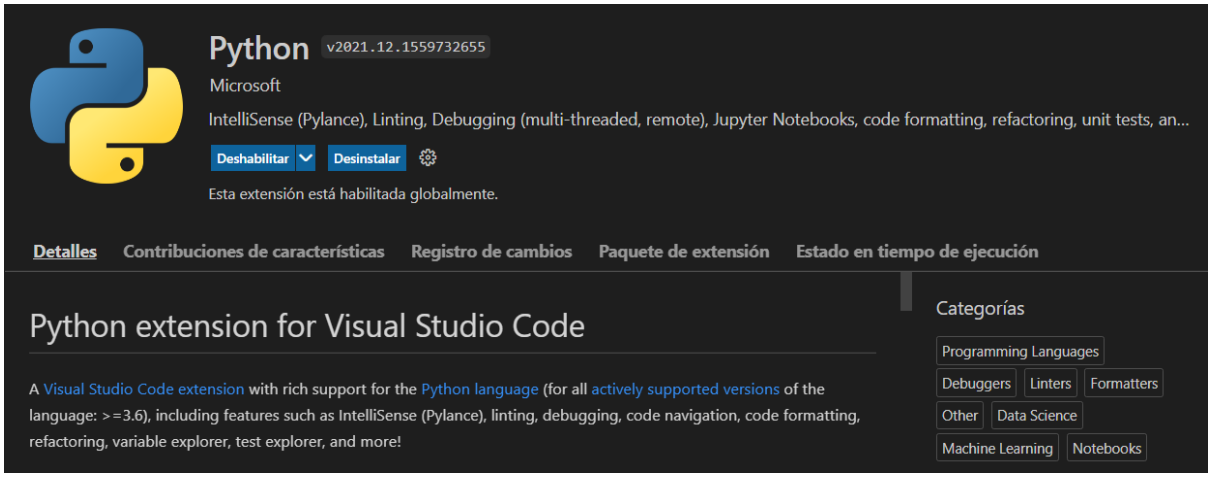


Figura 5-9 Extensión Python en Visual Studio Code

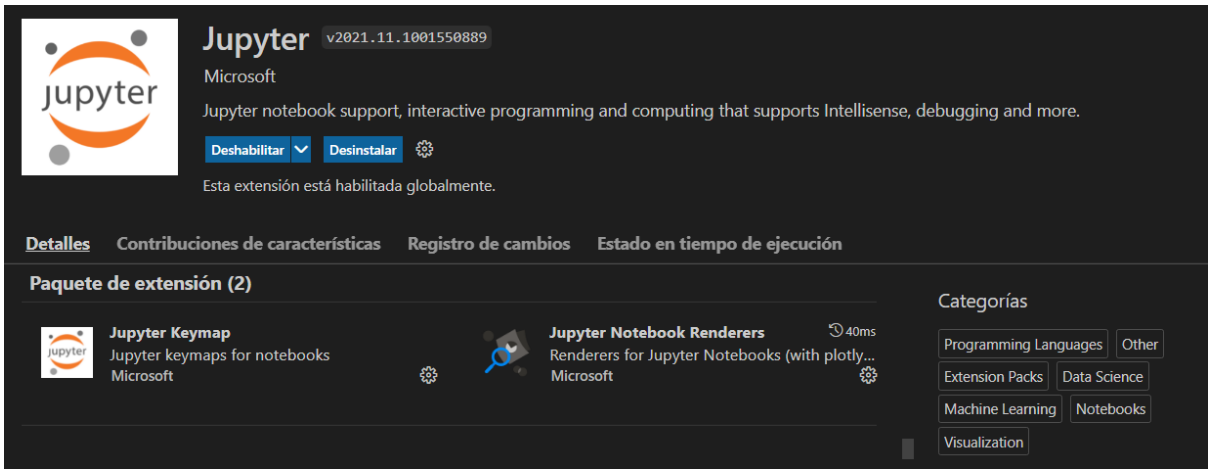


Figura 5-10 Extensión Jupyter en Visual Studio Code

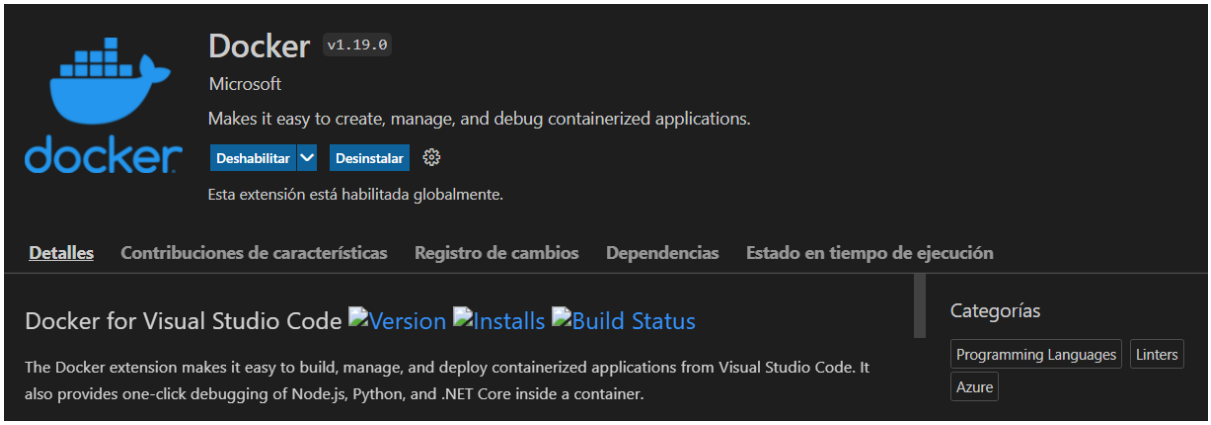


Figura 5-11 Extensión Docker en Visual Studio Code

5.3. Interfaz web de usuario (Web UI). Flask

Introducción
Para el desarrollo de la interfaz web de usuario (Web UI) existen multitud de opciones de frameworks disponibles vamos a repasar algunos de ellos:

1. **Framework para node.js:**
    - Express: Muy popular con disponibilidad de gran cantidad de información en internet. *Express es un marco de aplicación web de Node.js mínimo y flexible que proporciona un conjunto sólido de funciones para aplicaciones web y móviles.*[40]
    - Feathers: interfaz común para enrutamiento, solicitudes, ... *Feathers es un marco web ligero para crear aplicaciones en tiempo real y API REST utilizando JavaScript o TypeScript. Feathers puede interactuar con cualquier tecnología de backend, admite más de una docena de bases de datos y funciona con cualquier tecnología de frontend como React, VueJS, Angular, React Native, Android o iOS.*[41]
    - Fastify: (framework sencillo, analiza json automáticamente) *Fastify es un marco web altamente enfocado en brindar la mejor experiencia de desarrollador con la menor sobrecarga y una poderosa arquitectura de complementos, inspirada en Hapi y Express. Hasta donde sabemos, es uno de los marcos web más rápidos.*[42]
    - Koa: *Es un nuevo marco web diseñado por el equipo detrás de Express, que pretende ser una base más pequeña, más expresiva y más sólida para aplicaciones web y API. Al aprovechar las funciones asíncronas, Koa le permite deshacerse de las devoluciones de llamadas y aumentar considerablemente el manejo de errores. Koa no incluye ningún middleware en su núcleo y proporciona un elegante conjunto de métodos que hacen que escribir servidores sea rápido y agradable*.[43]

2. **Frameworks para otros lenguajes de programación**
    - Django (Python) *Django es un marco web Python de alto nivel que fomenta un desarrollo rápido y un diseño limpio y pragmático. Creado por desarrolladores experimentados, se encarga de gran parte de las molestias del desarrollo web, por lo que podemos concentrarnos en escribir la aplicación sin necesidad de reinventar la rueda. Es gratis y de código abierto.*[44]
    - Angular (Javascript): Angular es una plataforma de desarrollo, construida en TypeScript. *Como plataforma, Angular incluye: Un marco basado en componentes para crear aplicaciones web escalables. Una colección de bibliotecas bien integradas que cubren una amplia variedad de funciones, incluido el enrutamiento, la gestión de formularios, la comunicación cliente-*

[40] https://expressjs.com/
[41] https://docs.feathersjs.com/
[42] https://www.fastify.io/
[43] https://koajs.com/#introduction
[44] https://www.djangoproject.com/

servidor y más. Y por último un conjunto de herramientas de desarrollo para ayudarlo a desarrollar, compilar, probar y actualizar el código.[45]

- Flask: Se trata de un marco de desarrollo web muy simple y ligero basado en Python y es el elegido para trabajar en este TFG. A continuación profundizaremos más en él.

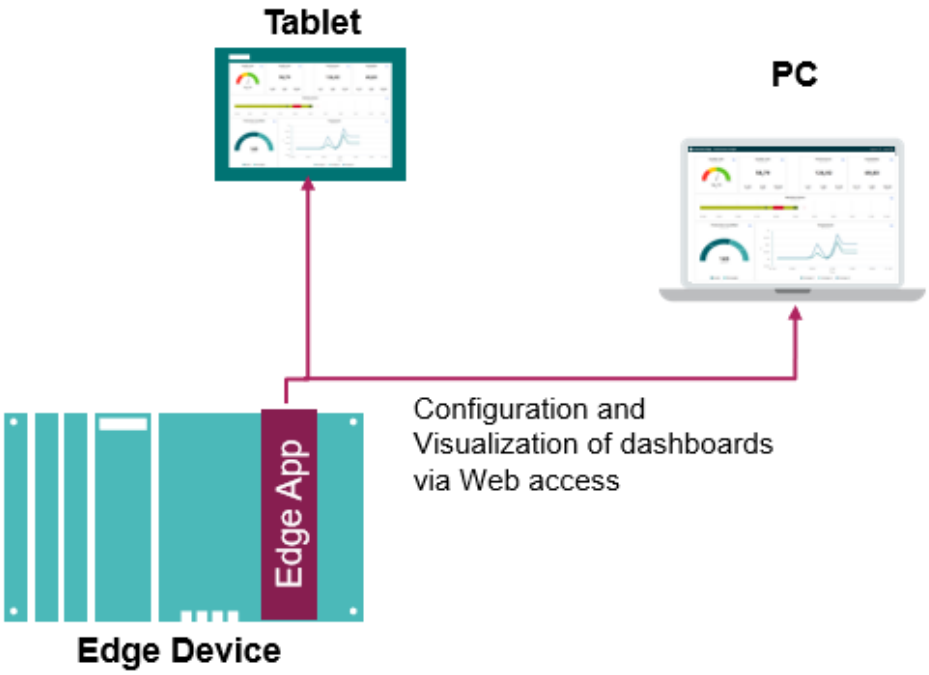


Figura 5-12 Interfaz web de aplicaciones IE. Fuente: Siemens Industry Image Database 4.11

Flask:
Flask o Matraz como también se le conoce es un Microframework para desarrollo web WSGI creado por Armin Ronacher[46] y desarrollado y soportado por The Pallets organization que se sostiene por las donaciones recibidas[47]
Está diseñado para que empezar sea rápido y fácil, se puede empezar creando una aplicación con un único archivo de Python, pero manteniendo la posibilidad de escalar a aplicaciones complejas. En un primer momento comenzó como un simple marco de Werkzeug y Jinja y se ha convertido en uno de los marcos de aplicaciones web de Python más populares debido a su sencillez y ligereza.

Flask ofrece sugerencias, pero no impone dependencias ni diseño de proyecto. Depende del desarrollador elegir las herramientas y bibliotecas que desea utilizar. Hay muchas extensiones proporcionadas por la comunidad que facilitan la adición de nuevas funciones.

Flask da soporte y permite trabajar con:

[45] https://angular.io/guide/what-is-angular
[46] https://es.wikipedia.org/wiki/Armin_Ronacher
[47] https://psfmember.org/

- **Werkzeug:** enrutamiento, depuración y la puerta de enlace del servidor web (WSGI). Se trata de una completa biblioteca de aplicaciones web WSGI. Comenzó como una simple colección de varias utilidades para aplicaciones WSGI y se ha convertido en una de las bibliotecas de utilidades WSGI más avanzadas.[48]
- **Jinja2:** es uno de los motores de plantillas más utilizados para Python. Está inspirado en el sistema de plantillas de Django pero lo amplía con un lenguaje expresivo que proporciona a los autores de plantillas un conjunto de herramientas más potente. Además de eso, agrega ejecución en espacio aislado y escape automático opcional para aplicaciones donde la seguridad es importante.[49]
  Se basa internamente en Unicode y se ejecuta en una amplia gama de versiones de Python desde 2.5 hasta las versiones actuales, incluida Python 3.
- **Click:** es un paquete de Python para crear atractivas interfaces de línea de comandos con tan poco código como sea necesario. Es altamente configurable aunque viene con valores predeterminados listos para usar.[50]

Flask no tiene soporte nativo para acceder a bases de datos, validar formularios web, autenticación de usuarios u otras tareas de alto nivel típicas de la mayoría de los sitios web.
Las funcionalidades de ese tipo que se necesiten están disponibles a través de extensiones que se integran con el paquete principal.
Como desarrollador, tenemos la potestad de elegir las extensiones que mejor se adapten a nuestro proyecto, o incluso escribir uno extensión propia. Esto contrasta con otros marcos más amplios, donde la mayoría de esas funcionalidades ya están incorporadas y son difíciles o a veces imposibles de cambiar.

Instalación
En nuestro caso, lo primero va a ser instalar las extensiones para Flask y Jinja directamente desde Visual Studio Code
Esto nos va a permitir además de escribir la aplicación probarla dentro del entorno de VSC.

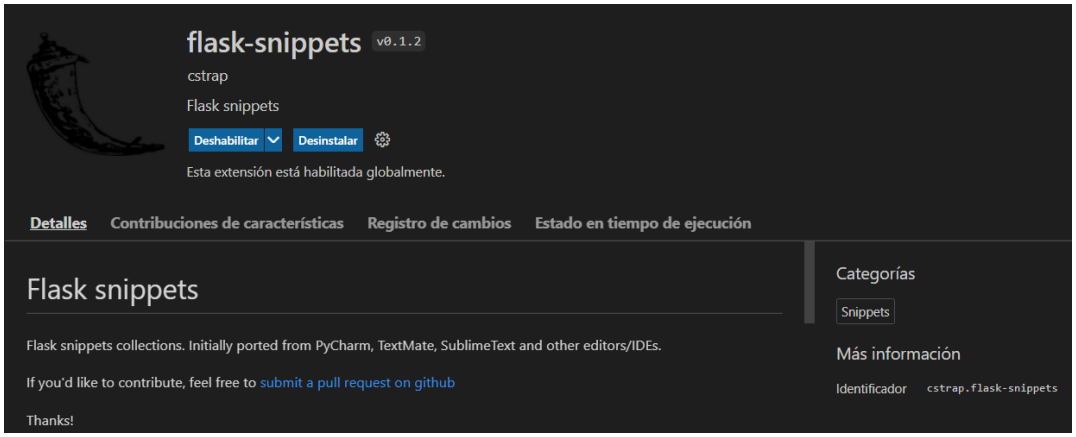


Figura 5-13 Extensión Flask para Visual Studio Code

[48] https://werkzeug.palletsprojects.com/en/2.0.x/

[49] https://palletsprojects.com/p/jinja/

[50] https://click.palletsprojects.com/en/8.0.x/

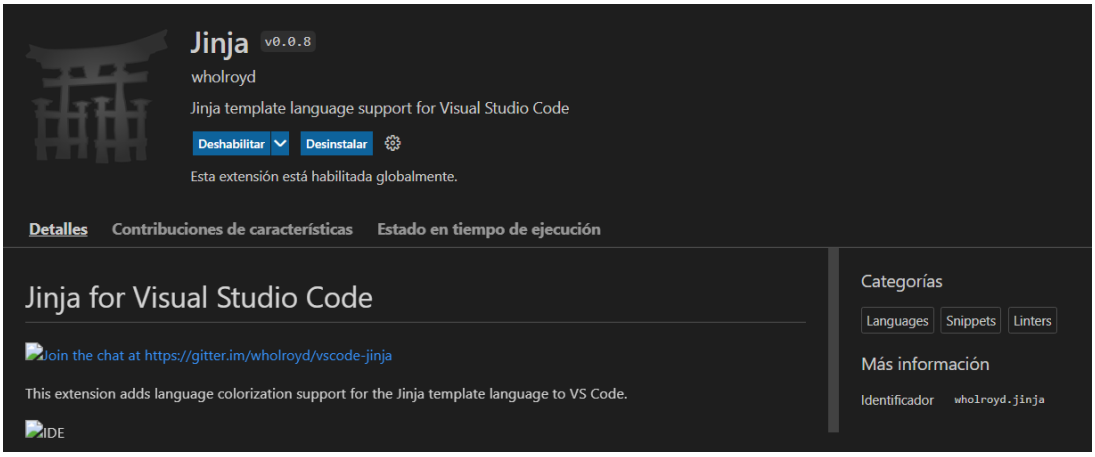
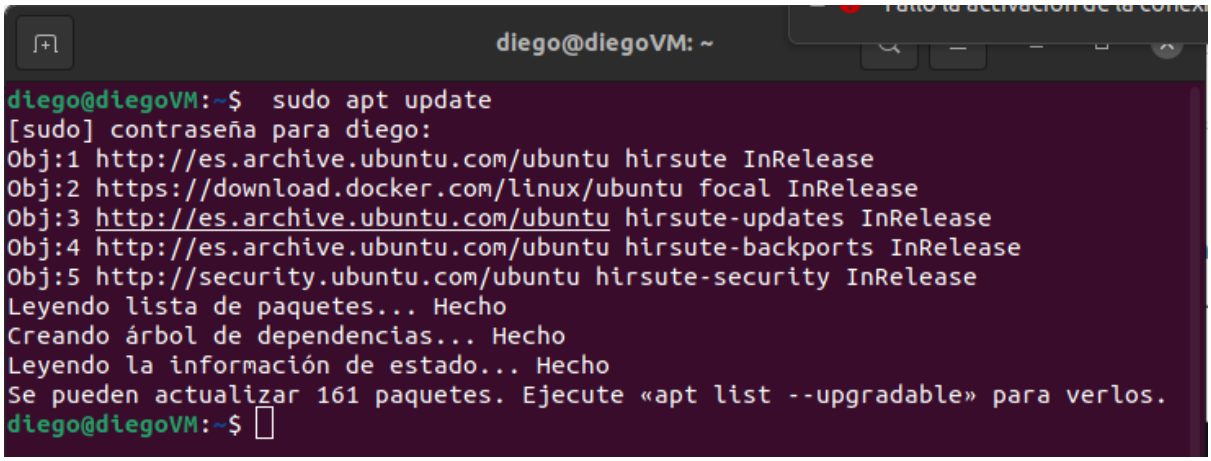


Figura 5-14 Extensión Jinja para Visual Studio Code

Si queremos probar la aplicación antes de empaquetarla en un contenedor fuera del entorno de VSC entonces es necesario instalar Flask también en nuestro PC con Ubuntu. Para esto se recomienda crear un entorno virtual venv para trabajar dentro de él.

1. Primero actualizamos el repositorio de aplicaciones disponibles.

```
$ sudo apt update
```



2. Comprobamos que tenemos instalada una versión de Python compatible con Flask.

```
$ python3 –version
```

3. Ejecutamos el siguiente comando para crear un entorno virtual de python a través del paquete python3-venv:

```
$ sudo apt install python3-venv
```

```
diego@diegoVM:~$  sudo apt install python3-venv
[sudo] contraseña para diego:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  python3.9-venv
Se instalarán los siguientes paquetes NUEVOS:
  python3-venv python3.9-venv
0 actualizados, 2 nuevos se instalarán, 0 para eliminar y 161 no actualizados.
Se necesita descargar 6.692 B de archivos.
Se utilizarán 33,8 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s
Des:1 http://es.archive.ubuntu.com/ubuntu hirsute-updates/universe amd64 python3
.9-venv amd64 3.9.5-3ubuntu0~21.04.1 [5.452 B]
Des:2 http://es.archive.ubuntu.com/ubuntu hirsute/universe amd64 python3-venv am
d64 3.9.4-1 [1.240 B]
Descargados 6.692 B en 0s (28,5 kB/s)
Seleccionando el paquete python3.9-venv previamente no seleccionado.
(Leyendo la base de datos ... 201663 ficheros o directorios instalados actualmen
te.)
Preparando para desempaquetar .../python3.9-venv_3.9.5-3ubuntu0~21.04.1_amd64.de
b ...
Desempaquetando python3.9-venv (3.9.5-3ubuntu0~21.04.1) ...
Seleccionando el paquete python3-venv previamente no seleccionado.
Preparando para desempaquetar .../python3-venv_3.9.4-1_amd64.deb ...
Desempaquetando python3-venv (3.9.4-1) ...
Configurando python3.9-venv (3.9.5-3ubuntu0~21.04.1) ...
Configurando python3-venv (3.9.4-1) ...
```

4. Creamos un nuevo directorio para la aplicación Flask y usamos el comando mencionado a continuación para navegar hasta allí:

```
$ mkdir flask_dir && cd flask_dir
```

```
diego@diegoVM:~$  mkdir flask_dir && cd flask_dir
diego@diegoVM:~/flask_dir$
```

5. Una vez que hemos creado el directorio, ejecutamos el siguiente comando para crear un entorno virtual dentro del directorio "flask_dir":

```
$ python3 -m venv venv
```

```
diego@diegoVM:~/flask_dir$  python3 -m venv venv
diego@diegoVM:~/flask_dir$
```

6. El “venv” es el nombre del directorio del entorno virtual. Para instalar Flask, necesitamos activar este directorio:

```
$ source venv/bin/actívate
```

```
diego@diegoVM:~/flask_dir$ source venv/bin/activate
(venv) diego@diegoVM:~/flask_dir$
```

7. Después de la configuración completa de la instalación de python y la creación de directorios, ahora podemos instalar el marco Flask con todos sus componentes a través del administrador de paquetes de Python "pip":

```
$ pip install flask
```

```
(venv) diego@diegoVM:~/flask_dir$  pip install flask
Collecting flask
  Downloading Flask-2.0.3-py3-none-any.whl (95 kB)
     |████████████████████████████████| 95 kB 1.7 MB/s
Collecting itsdangerous>=2.0
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting Werkzeug>=2.0
  Downloading Werkzeug-2.0.3-py3-none-any.whl (289 kB)
     |████████████████████████████████| 289 kB 7.3 MB/s
Collecting Jinja2>=3.0
  Downloading Jinja2-3.1.1-py3-none-any.whl (132 kB)
     |████████████████████████████████| 132 kB 30.2 MB/s
Collecting click>=7.1.2
  Downloading click-8.0.4-py3-none-any.whl (97 kB)
     |████████████████████████████████| 97 kB 12.1 MB/s
Collecting MarkupSafe>=2.0
  Downloading MarkupSafe-2.1.1-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86
_64.whl (25 kB)
Installing collected packages: MarkupSafe, Werkzeug, Jinja2, itsdangerous, click
, flask
Successfully installed Jinja2-3.1.1 MarkupSafe-2.1.1 Werkzeug-2.0.3 click-8.0.4
flask-2.0.3 itsdangerous-2.1.2
(venv) diego@diegoVM:~/flask_dir$
```

8. Verificamos la instalación.

```
$ python -m flask –version
```

```
(venv) diego@diegoVM:~/flask_dir$ python -m flask --version
Python 3.9.5
Flask 2.0.3
Werkzeug 2.0.3
```

Hola mundo con Flask

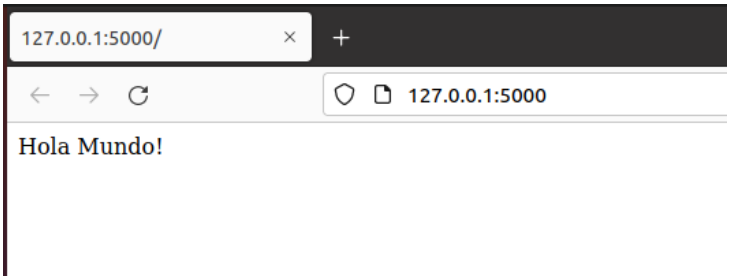
Vamos a escribir una aplicación muy sencilla para probar que todo funcione.

Una vez creada nuestra aplicación prueba.py la guardamos en el directorio Flask_dir que creamos anteriormente y la ejecutamos con los siguientes comandos:

```
export Flask_APP=prueba.py
flask run
```

```
(venv) diego@diegoVM:~/flask_dir$ export FLASK_APP=prueba.py
(venv) diego@diegoVM:~/flask_dir$ flask run
 * Serving Flask app 'prueba.py' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [27/Mar/2022 23:44:27] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/Mar/2022 23:44:27] "GET /favicon.ico HTTP/1.1" 404 -
```

Esto nos arranca un servidor web en la dirección del localhost de nuestro PC al que podemos acceder haciendo clic directamente sobre la IP.
Aquí es importante destacar que Flask utiliza un servidor web sencillo para presentar nuestra aplicación en un entorno de desarrollo, con el depurador de Flask ejecutándose en paralelo para hacer que sea más fácil depurar nuestra aplicación. Este servidor de desarrollo no debería usarse en una implementación de producción. En la página Opciones de implementación en la documentación de Flask podemos informarnos de las opciones disponibles de servidores web para producción.
Flask Deployment Options: https://flask.palletsprojects.com/en/2.0.x/deploying/



Estructura de un proyecto Flask

La estructura de una aplicación desarrollada en python mediante Flask deberá tener una estructura similar a la de la siguiente figura.

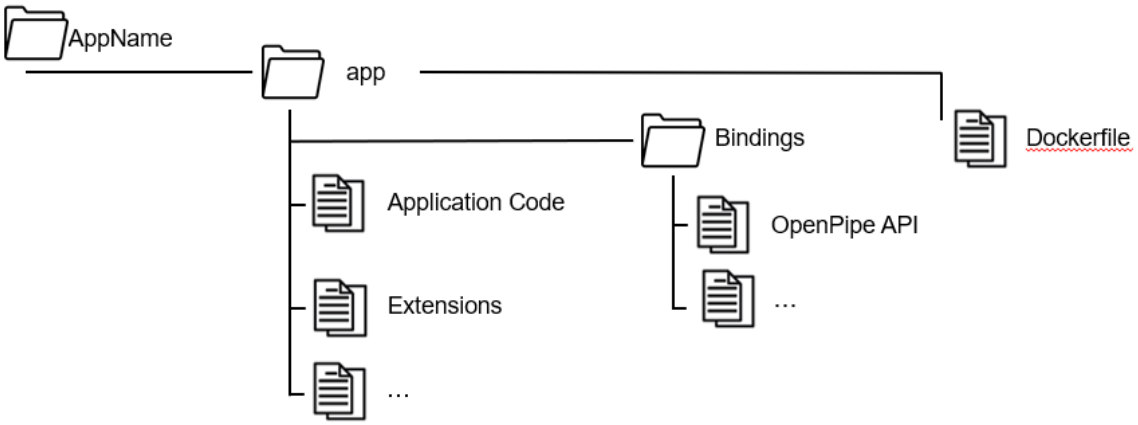


Figura 5-15 Estructura de una App Industrial Edge. Fuente: Siemens Industry Image Database 4.11

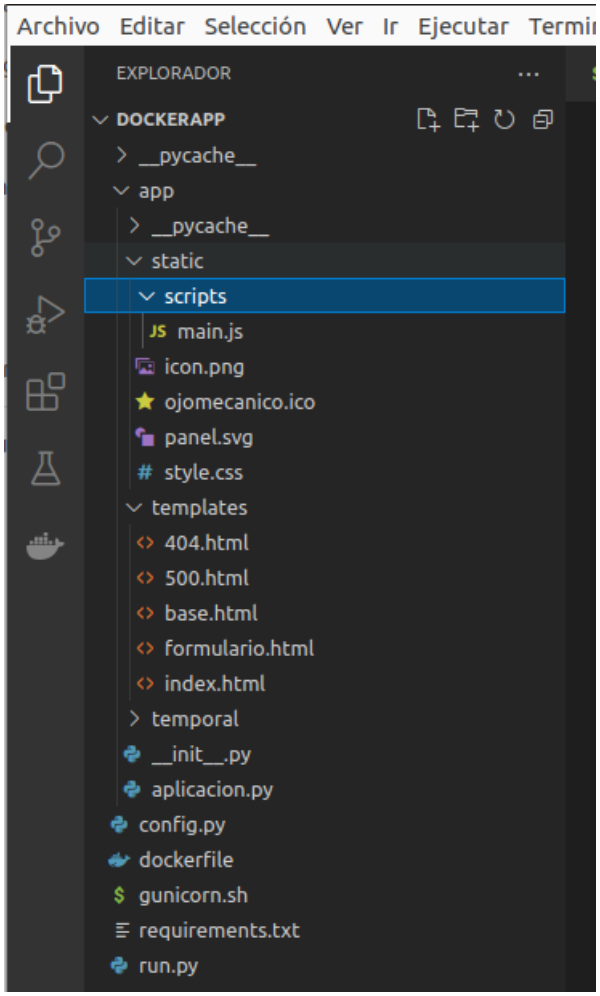


Figura 5-16 Ejemplo de aplicación en Flask

- **App:** Contiene el código y los ficheros de inicialización de nuestra aplicación en ficheros con extensión .py
- **Templates:** Contiene las páginas html desarrolladas para nuestra interfaz de usuario web
- **Static:** Esta carpeta contiene las imágenes e iconos que vamos a utilizar en nuestra aplicación, también la utilizamos para guardar los ficheros con extensión .css que sirven para dar un estilo y formato a nuestras páginas web y en la carpeta script los ficheros con extensión .js para dotar de interactividad a la web.

5.4. Docker Engine

Instalación:
Después de la instalación y configuración del entorno de desarrollo, con las extensiones de paquetes necesarias como Python o Flask lo siguiente es instalar Docker Engine, este debe instalarse con una versión superior a la V18.09.
En el presente TFG se ha utilizado la 20.10.12 de Docker Engine.

En los siguientes pasos vamos a mostrar la instalación en Ubuntu. Para trabajar con MS Windows o para obtener más información, deberemos consultar la documentación oficial de Docker.[51]
Es importante no confundir la instalación de la extensión de Docker para Visual Studio Code que nos va a permitir, una vez finalizada nuestra aplicación en Python, Flask, etc crear la correspondiente imagen de Docker con Docker Engine, el motor de Docker que va a permitirnos crear un repositorio de imágenes de Docker en nuestro PC, probarlas y compartirlas a través de la API abierta con Industrial Edge Publisher para la generación de la app propiamente dicha con extensión .app.

Nota: si se tiene una versión anterior de Docker y se desea actualizarla, primero es necesario desinstalar antes de instalar una nueva versión.

Vamos a ver los pasos a seguir.

1. Desinstalamos versiones antiguas si las hubiera.

```
$ sudo apt-get remove docker docker-engine docker.io containerd runc
```

2. Configuramos el repositorio actualizando el índice de paquetes apt y habilitamos que apt use un repositorio a través de HTTPS:

```
$ sudo apt-get update


$ sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

3. Agregamos la clave GPG oficial de Docker

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor
-o /usr/share/keyrings/docker-archive-keyring.gpg
```

4. Utilizamos el siguiente comando para configurar el repositorio estable.

```
$ echo \
  "deb [arch=$(dpkg --print-architecture) signed-
by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu \
```

[51] https://docs.docker.com/engine/install/ubuntu/

```
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list
> /dev/null
```

5. Actualizamos el índice del paquete apt e instalamos la última versión de Docker Engine y containerd:

```
$ sudo apt-get update

$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

6. Otra opción es enumerar todas las versiones de Docker disponibles e instalar la deseada con:

```
$ sudo apt-cache madison docker-ce
```

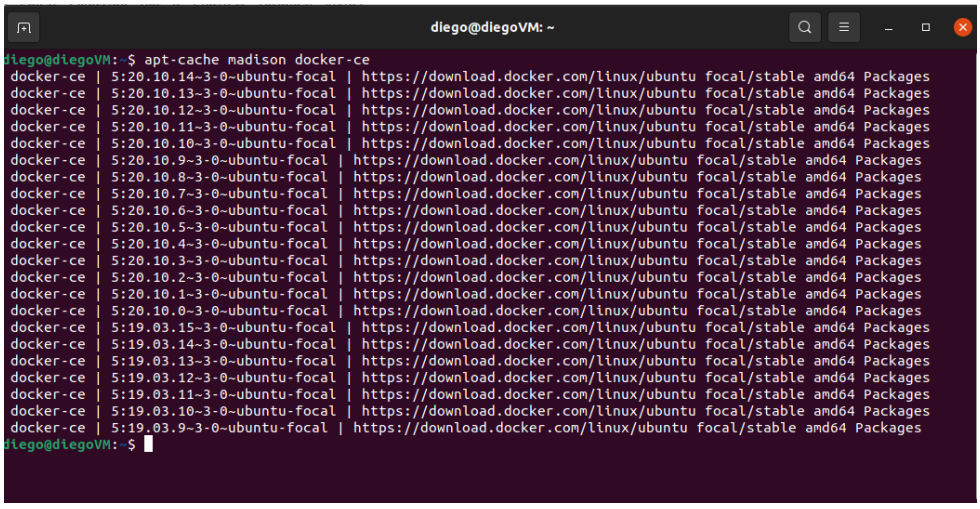


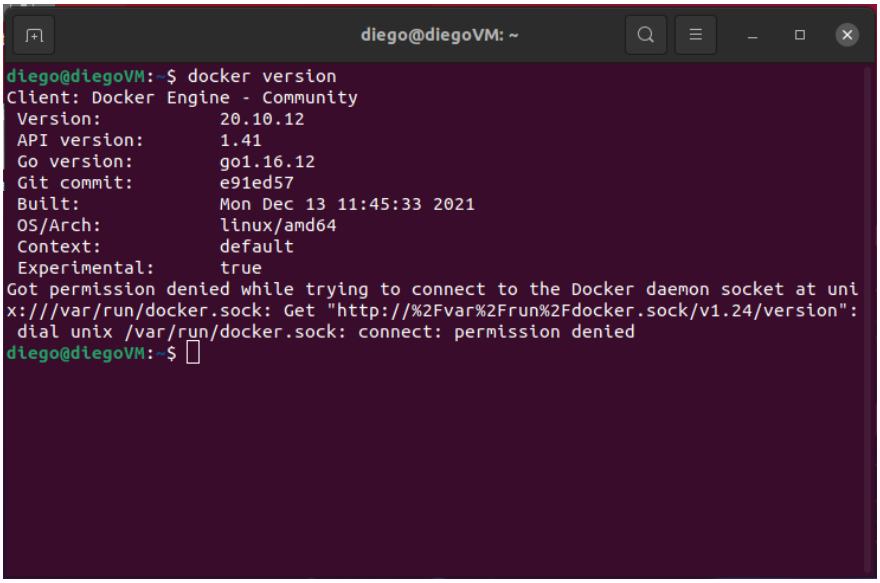
Figura 5-17 Volcado de tolas versiones de Docker disponibles

En nuestro caso instalamos Docker con la versión 20.10.12 reemplazando el número de versión <VERSION_STRING> en el comando:

```
  $ sudo apt-get install docker-ce=<VERSION_STRING> docker-ce-
cli=<VERSION_STRING> containerd.io
```

7. Una vez instalado será aconsejable configure Docker para que se pueda usar como un usuario no root y se inicie automáticamente al iniciar el sistema operativo. De otra manera deberemos iniciar todos los comandos Docker con la instrucción sudo y registrarnos con la contraseña de administrador.
   Vamos a comprobar la versión instalada:

```
$ sudo docker version
```



8. Por último, para verificar la correcta instalación de Docker Engine utilizaremos el comando:

```
$ sudo docker run hello-world
```

Mediante dicho comando podemos ver que en efecto todo está correctamente instalado y funcionando.

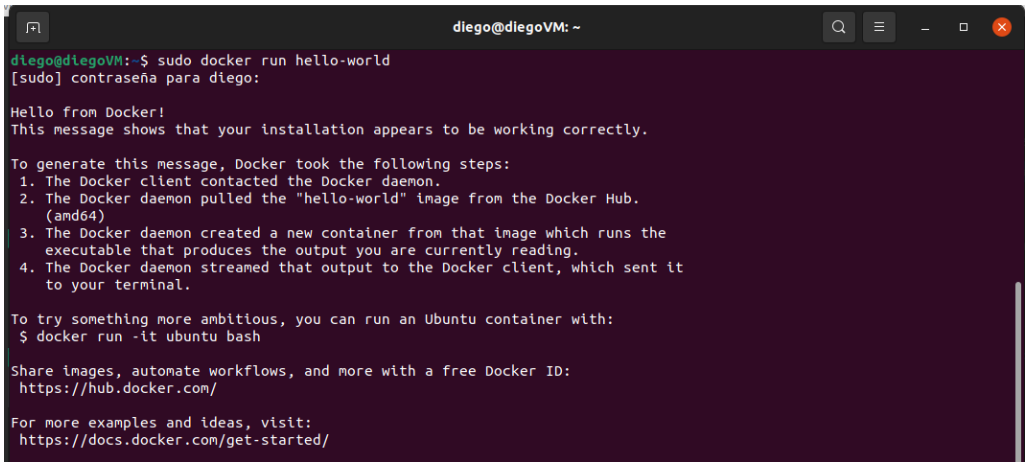


Figura 5-18 Prueba de Hello World después de la instalación de Docker

9. Por último, deberemos **exponer la API de Docker** para que nuestro Industrial Edge App Publisher se pueda conectar a fin de seleccionar la imagen de Docker que queremos publicar.

   Para habilitar la exposición de la API, debemos editar el siguiente archivo[52]:

[52] https://cduser.com/pique-19-como-exponer-la-api-de-docker/

```
$ vim /lib/systemd/system/docker.service
```

Buscamos lo siguiente:

ExecStart

La línea que veremos, por defecto, se verá así:

```
ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

Y nosotros le agregaremos lo siguiente, para que quede así:

```
ExecStart=/usr/bin/dockerd -H fd:// -H=tcp://0.0.0.0:2375 --containerd=/run/containerd/containerd.sock
```

Guardamos, cerramos y ejecutamos lo siguiente:

```
$ sudo systemctl daemon-reload
```

Por último:

```
$ sudo systemctl restart docker
```

Para comprobar que esta todo bien, podemos consultar las imágenes de Docker que tenemos descargadas. Lo hacemos, de la siguiente manera:

```
$ curl http://localhost:2375/images/json
```

Nos debería devolver algo como esto:

Ya lo tendríamos, ahora podemos usar la API para conectar nuestro IEAP o monitorizar remotamente el host.

5.5 Industrial Edge App Publisher

Industrial Edge Publisher para Unified Comfort Panels nos permite convertir las imágenes Docker en apps Edge y firmarlas para posteriormente poderlas instalar en dispositivos IE

Instalación
La instalación de esta aplicación va aser muy sencilla pudiéndose descargar directamente desde la página de soporte de Siemens (SIOS) tanto para Linux como para MS Windows.
Link de descarga:
https://support.industry.siemens.com/cs/es/en/view/109778875

Instalación en Ubuntu:

```
sudo -s

apt-get update

apt-get install -f industrial_edge_app_publisher_xxx.deb
```

Después de ejecutar los comandos, IEAP se instala correctamente como se muestra en la salida de la consola. En Ubuntu también puede verificarlo gráficamente como se ve en la siguiente figura

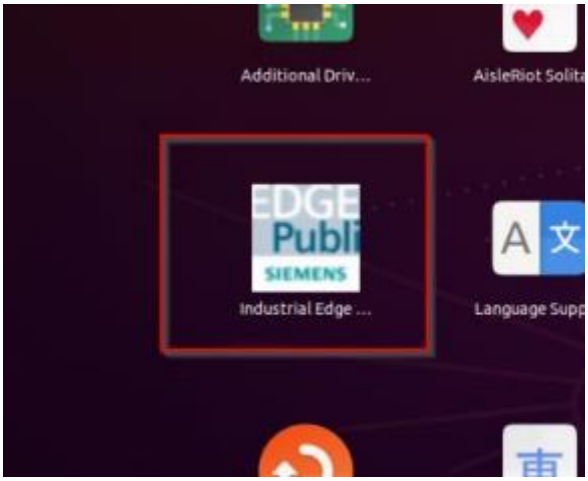
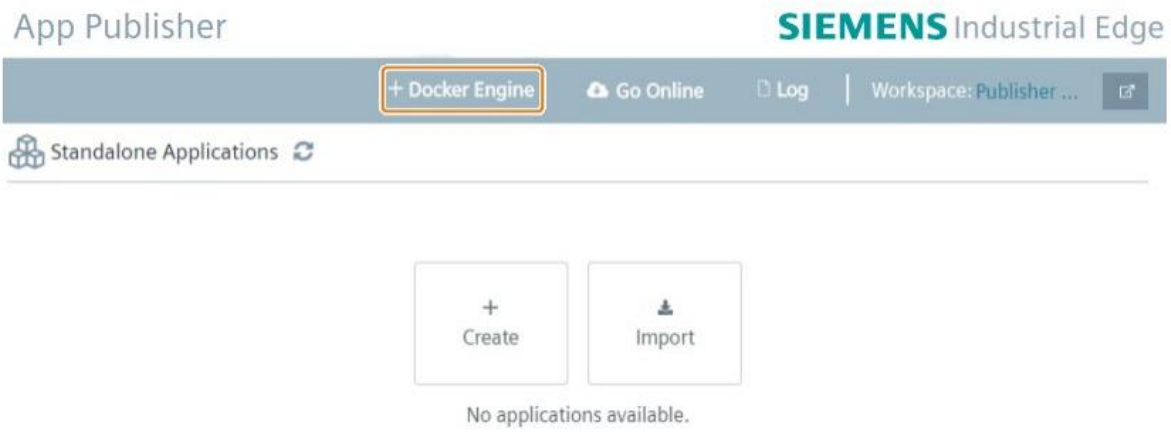


Figura 5-19 Descripción general del software gráfico en Ubuntu con IEAP

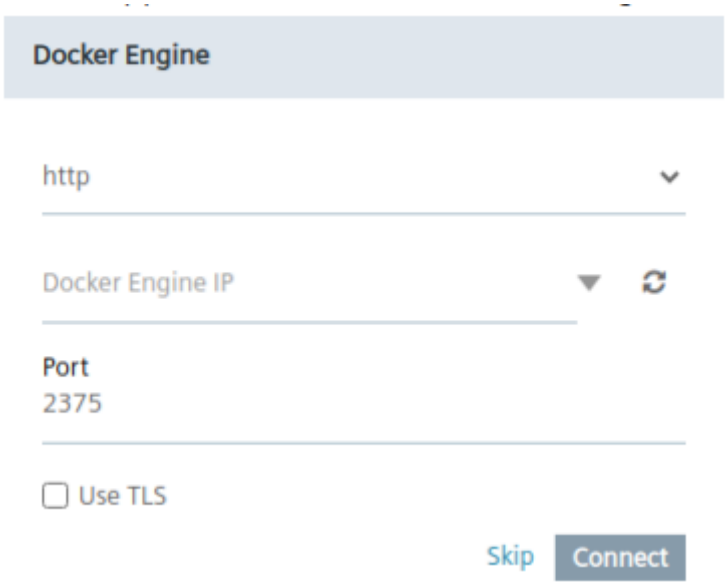
Conexión de IEAP a Docker Engine

En este apartado vamos a explicar cómo conectarse a un Docker Engine para poder integrar imágenes directamente desde allí.

1. Hacemos clic en el botón "+Docker Engine"



2. IE App Publisher muestra la configuración de Docker Engine.



3. Elegimos el protocolo de aplicación que utiliza Docker Engine en el menú desplegable: http

4. Indicamos la IP y el puerto de Docker Engine en los campos de entrada respectivos ya que Docker Engine y IEAP pueden estar en el mismo PC o en PC diferentes.
   De manera predeterminada, está configurado el puerto 2375, pero se puede cambiar.
   Para verificar una conexión con Docker Engine, presionamos el símbolo de actualizar.

5. Para una comunicación segura debemos activar el campo de opción TLS (Transport Layer Security).

6. Hacemos clic en el botón "Conectar" y se establece la conexión con Docker Engine.

También es posible publicar aplicaciones sin conexión con Docker Engine si se dispone del fichero yaml de la imagen de Docker de nuestra aplicación y está en un repositorio accesible.

6. Desarrollo Apps para Siemens IE

6.1 Visión general

El desarrollo de apps para correr en dispositivos IE consiste, básicamente, en desarrollar aplicaciones valiéndose de herramientas estándares IT, pero siempre teniendo en cuenta las peculiaridades y limitaciones del hardware sobre el que va a correr la aplicación.
Una vez desarrollada empaquetaremos la misma en un contenedor estandarizado Docker que podrá ejecutarse en un dispositivo de campo industrial mediante la runtime del IE.

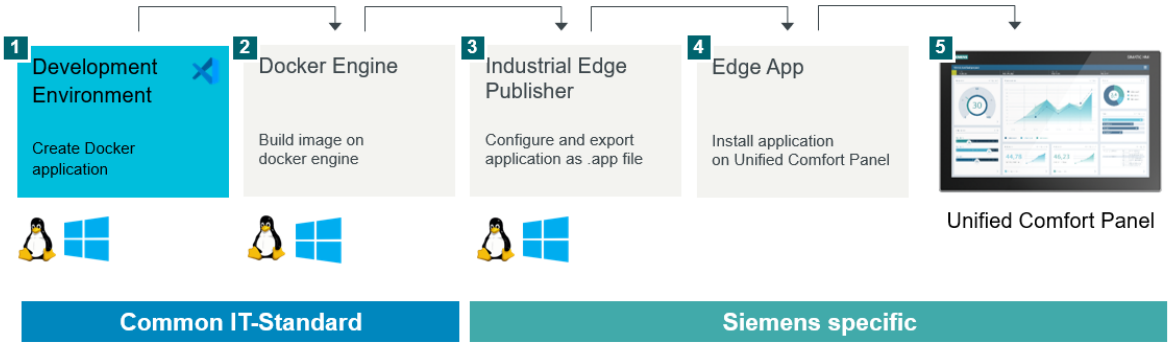


Figura 6-1 Flujo de trabajo de desarrollo de aplicaciones para Unified Comfort Panel. Fuente: Siemens Industry Image Database 4.11

El desarrollo y servicio de las apps de IE va a constar de las siguientes etapas:

1. Desarrollo de la aplicación:
   En esta etapa vamos a desarrollar nuestra aplicación con el framework de programación que mejor se adapte a nuestras necesidades. En el presente TFG se ha utilizado Python con las librerías OpenCV para la parte de backend (gestión de imágenes, conexión a la API de Azure Cognitive Services, comunicación con el HMI…) y Flask/Jinja para la parte del Frontend (La interfaz web de usuario).
   La elección de esté framework y no otro ha venido motivado por la ligereza de las aplicaciones desarrolladas en Flask/Python, cuestión de vital importancia teniendo en cuanta que nuestra app no va a correr en un PC con IE si no en una UCP limitada a 800 Mb de memoría para apps.
2. Dockerización de la aplicación. Aquí sí que estamos “obligados” a utilizar Docker Engine para empaquetar y probar la aplicación. En nuestro caso en un entorno Linux (Ubuntu) debido al mejor desempeño para desarrolladores que entornos basados en MS Windows.
3. Publicación de la aplicación como una app de IE. Para esta fase recurriremos al Industrial Edge Plublisher de Siemens, software gratuito y con distribuciones tanto para Linux como para MS Windows. Este software nos va a permitir conectar con el repositorio local de Docker a través de su API, seleccionar una imagen y convertirla en una aplicación con extensión .app lista para instalar en nuestro dispositivo IE.

4. Instalación de la app en el dispositivo IE. Aquí vamos a tener dos alternativas, podremos ir a una gestión centralizada con un IEM que podrá estar en planta o en la nube, o utilizar el IEM que la UCP incorpora para la gestión directa sobre el propio dispositivo.
   En nuestro caso se ha optado por la primera opción ya que la segunda todavía no está liberada al público.
5. Ejecución de la aplicación. Una vez instalada podremos arrancar la aplicación en nuestro dispositivo IE y comprobar el correcto funcionamiento.

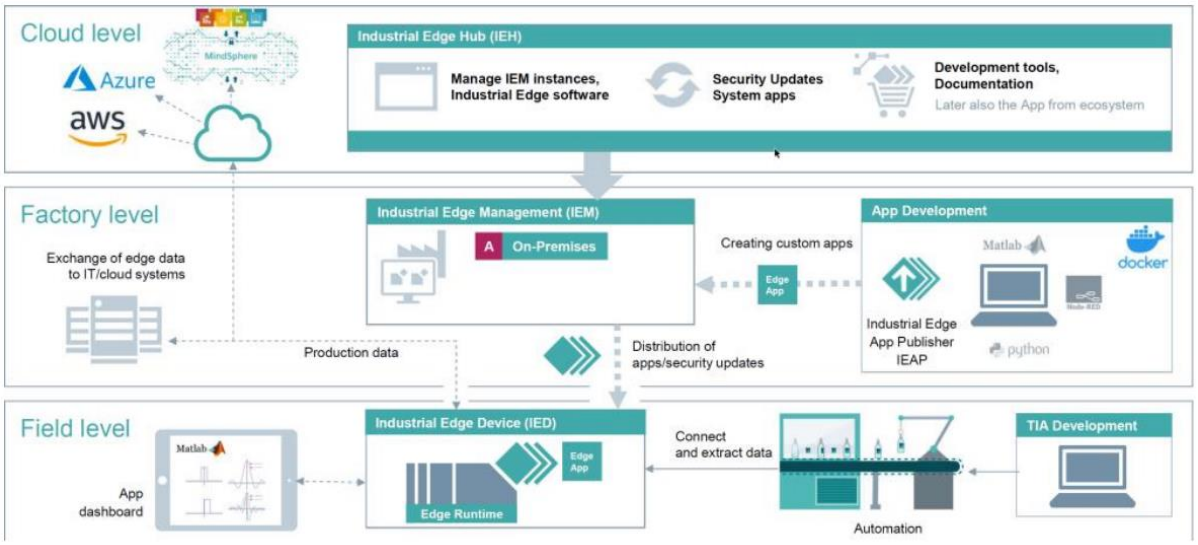


Figura 6-2 Arquitectura de la plataforma Siemens Industrial Edge. Fuente: Siemens Industry Image Database 4.11

6.2. Desarrollo con contenedores

6.2.1 Docker: Imagenes & Dockerfile

Vamos a ver el desarrollo con Docker un poco más detalladamente: la idea es empaquetar la funcionalidad en un contenedor estandarizado que pueda ejecutarse en un dispositivo de automatización.

Cualquier aplicación escrita en cualquier lenguaje se pueden colocar en contenedores y luego ejecutar en cualquier hardware, en nuestro caso en un dispositivo Edge y la gran ventaja es que todo se puede administrar de forma centralizada, además un dispositivo Edge puede contener varias aplicaciones/contenedores a la vez. No se afectan entre sí si no se requiere y están aislados. Esta característica nos permite ampliar la funcionalidad de nuestro dispositivo Edge fácilmente con aplicaciones adicionales o nuevas versiones sin afectar todo el sistema.
Básicamente el desarrollo consistirá en escribir nuestra aplicación en el lenguaje preferido y utilizando un marco web que se ajuste a ese lenguaje si deseamos que nuestra app tenga interfaz web de usuario.

Dicha aplicación deberá tener un fichero Dockerfile descriptivo para que a partir de el se pueda generar la imagen de Docker que nos va a permitir por un lado publicarla vía IEAP y por otro probarla con Docker Engine generando el contenedor correspondiente.
Es importante recordar que para que el fichero Dockerfile sea reconocido y se pueda llamar a la instrucción de creación de la imagen de Docker es necesario que nuestro IDE (en nuestro caso Visual Studio Code) cuente con la extensión de Docker correspondiente como vimos en capítulos anteriores.

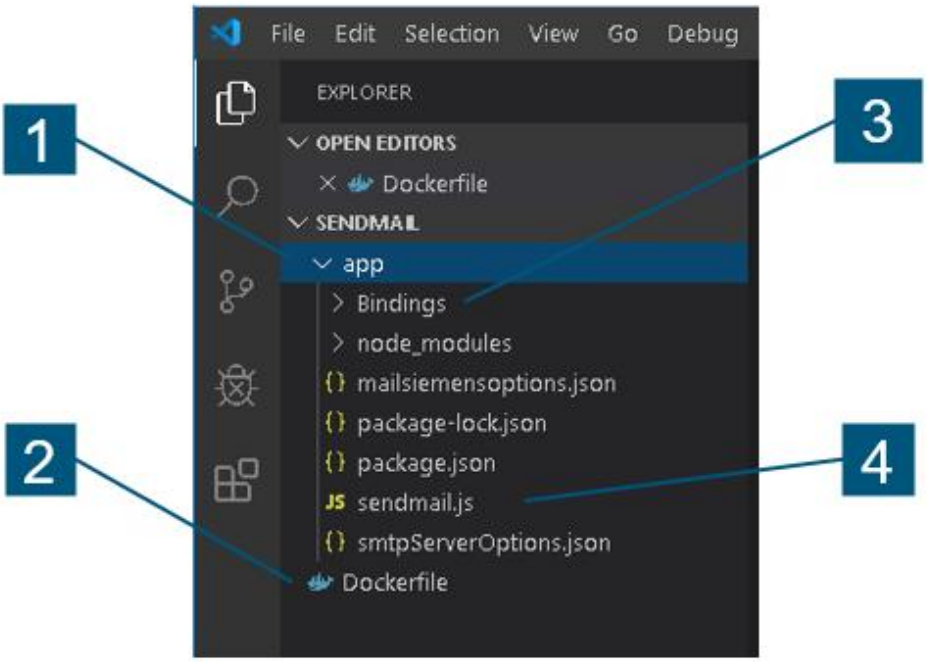


Figura 6-3 Estructura de un programa listo para "Dockerización"

1. App: Creamos una nueva aplicación en cualquier lenguaje de programación
2. Dockerfile: Archivo de instrucciones para construir la imagen del contenedor.
3. Connectividad: API para la conectividad vía OpenPipe a las las variables de automatización, en este caso las variables de la runtime del HMI
4. Sendmail.js: Aplicación de JavaScript que se ejecuta en el contenedor.

Estas aplicaciones las podremos haber escrito nosotros, pero también pueden ser de terceros. Desde Docker Hub tendremos acceso a la descarga de imágenes de Docker tanto básicas, de aplicaciones populares (Envío de Mails, Node-Red, Brokers MQTT…) como sistemas operativos "lite" para ejecutar nuestro código en ellos.

Descarga de imágenes desde Docker Hub:

1. Podemos usar filtros para buscar solo editores verificados o imágenes oficiales
2. Las reseñas y la cantidad de descargas también pueden darnos una indicación de si la imagen es ampliamente utilizada y confiable.

3. Es conveniente leer la descripción de la imagen para asegurarse de que haya suficiente documentación y que estamos seguros de la funcionalidad.
4. Podemos buscar las etiquetas si busca una versión específica, p. una versión LTS
5. Para descargar una imagen usamos el comando de extracción para la versión que desea descargar.

En nuestra máquina virtual de desarrollo, cuando estamos creando nuestra propia aplicación, comenzamos especificando un Dockerfile aunque esto no es obligatorio, también se puede realizar una vez hemos terminado y probado la app.
La primera línea en este Dockerfile especifica la imagen base que está utilizando, por lo que, para un ejemplo de ubuntu, el PC de desarrollo extraerá la imagen de Docker hub y la almacenará en local.
Una vez que hemos terminado de escribir nuestro Dockerfile, agregamos algunas funciones a esta imagen de Ubuntu, instalaremos librerías, indicaremos directorios internos de la aplicación o ejecutaremos la linea de comandos. Una vez editado nuestro Dockerfile podemos ejecutar el comando de compilación de Docker y crearemos una nueva imagen.
Luego tenemos dos opciones, publicar la imagen directamente con IEAP de Siemens o optar por ejecutar esta imagen en un contenedor en nuestra máquina de desarrollo para probarla.

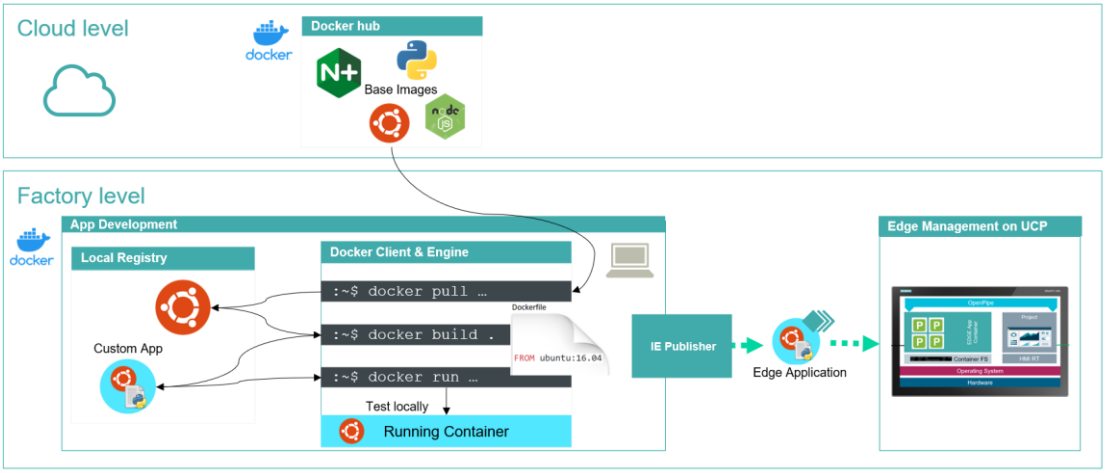


Figura 6-4 Uso de imágenes base descargadas desde Docker Hub. Fuente: Siemens Industry Image Database 4.11

Dockerfile
Docker puede crear imágenes automáticamente leyendo las instrucciones de un Dockerfile. Un Dockerfile es un documento de texto que contiene todos los comandos que un usuario podría llamar en la línea de comandos para ensamblar una imagen. Al usar docker build los

usuarios pueden crear una compilación automatizada que ejecuta varias instrucciones de la línea de comandos.[53]

```
FROM ubuntu:16.04
RUN apt-get update && apt-get install -y python3.5
COPY app.py .
CMD ["python3", "app.py"]
```

Figura 6-5 Ejemplo típico de fichero Dockerfile

Un fichero Dockerfile constará típicamente de los siguientes cinco apartados:

- **FROM:** Seleccionar una imagen base, será un mini S.O. con lo justo para ejecutar nuestro código, una buena opción aquí por su ligereza sería utilizar la imagen de Docker Alpine como veremos posteriormente.
- **WORKDIR:** Crear directorio de trabajo
- **COPY:** Copiar archivos o directorios en la imagen (Mínimo el que contenga nuestra aplicación).
- **RUN:** Instalar los paquetes y/o librerías necesarias como por ejemplo Python.
- **CMD:** Especificar qué comandos/software contiene la imagen para ejecutar. (Básicamente lanzar nuestra aplicación con las dependencias necesarias).

Elegir la imagen (FROM) es un paso esencial y mas cuando el HW para que estemos desarrollando nuestra app esté muy limitado, en la siguiente figura podemos observar la diferencia entre diferentes imágenes Docker disponibles para ejecutar aplicaciones de NodeJS

NodeJS Version	Image	Size
8.10.0	base	256 MB
8.10.0	alpine	23 MB
8.10.0	onbuild	266 MB
8.10.0	slim	92 MB

Figura 6-6 Imagenes para Node. Fuente: www.hub.docker.com[54]

[53] https://docs.docker.com/engine/reference/builder/
[54] https://hub.docker.com/_/node/

Instruction	Description
FROM	Select the base image
LABEL	Label image for better organization
RUN	Execute the commands and create a new layer
CMD	Specify commands to run within the container
EXPOSE	Indicates a port which is listen for a connection
ENV	Update environmental Variables
ADD /COPY	Add/copy files to image
WORKDIR	Create working directory
VOLUME	Expose database storage area
USER	Change application to an non-root-user

Figura 6-7 Instrucciones para el Dockerfile. Fuente: Siemens Industry Image Database 4.11

Construir el contenedor de Docker y ejecutarlo:
Una vez tenemos listo el fichero Dockerfile debemos ejecutar desde el terminal de VSC el siguiente comando para construir nuestra imagen y alojarla en el repositorio de Docker local de nuestro PC.

Escribimos: `docker build -t aplicacionejemplo .`

Lo que hace este comando es usar el comando de compilación, con el parámetro "-t" que especifica que queremos que Docker cree un nombre de etiqueta de imagen "aplicaciónejemplo" y el punto "." es para especificar que vamos a utilizar la ruta donde se encuentra nuestro Dockerfile, en nuestro caso la misma carpeta.

Si hacemos esto obtendremos algo parecido a la imagen de abajo:

```
Build an image from a Dockerfile
diego@diegoVM:~/DockerApp$ sudo docker build -t app1 .
Sending build context to Docker daemon  45.57kB
Step 1/7 : FROM python:3.9.5-alpine
 ---> 25d7d3156527
Step 2/7 : COPY requirements.txt /
 ---> Using cache
 ---> ef7975f62eeb
Step 3/7 : RUN pip3 install -r /requirements.txt
 ---> Using cache
 ---> c267661070d3
Step 4/7 : COPY . /DOCKERAPP
 ---> Using cache
 ---> 34846d920f2c
Step 5/7 : WORKDIR /DOCKERAPP
 ---> Using cache
 ---> ecc4d4d26840
Step 6/7 : EXPOSE 8080
 ---> Using cache
 ---> 0a915bd23e81
Step 7/7 : CMD ["gunicorn", "-b", "0.0.0.0:8080", "--workers", "2", "app.aplicacion:app"]
 ---> Using cache
 ---> b7e05aeea31f
Successfully built b7e05aeea31f
Successfully tagged app1:latest
diego@diegoVM:~/DockerApp$
```

A la hora de construir la imagen desde VSC podemos encontrarnos con el siguiente error o similar:
"Could not find any downloads that satisfy the requirement blinker==1.3 (from -r requirements.txt (line 1))
No distributions at all found for blinker==1.3 (from -r requirements.txt (line 1))"

```
Step 3/7 : RUN pip3 install -r /requirements.txt
 ---> Running in f4df97af6f88
WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<pip._vendor.urllib3
.connection.HTTPSConnection object at 0x7f4aaab518b0>: Failed to establish a new connection: [Errno -3] Try again')': /simple/azure-cognitiveservices-vision
-customvision/
WARNING: Retrying (Retry(total=3, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<pip._vendor.urllib3
.connection.HTTPSConnection object at 0x7f4aaaeb5fd0>: Failed to establish a new connection: [Errno -3] Try again')': /simple/azure-cognitiveservices-vision
-customvision/
WARNING: Retrying (Retry(total=2, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<pip._vendor.urllib3
.connection.HTTPSConnection object at 0x7f4aaaeb5e20>: Failed to establish a new connection: [Errno -3] Try again')': /simple/azure-cognitiveservices-vision
-customvision/
WARNING: Retrying (Retry(total=1, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<pip._vendor.urllib3
.connection.HTTPSConnection object at 0x7f4aaaeb5580>: Failed to establish a new connection: [Errno -3] Try again')': /simple/azure-cognitiveservices-vision
-customvision/
WARNING: Retrying (Retry(total=0, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<pip._vendor.urllib3
.connection.HTTPSConnection object at 0x7f4aaaeb5d60>: Failed to establish a new connection: [Errno -3] Try again')': /simple/azure-cognitiveservices-vision
-customvision/
ERROR: Could not find a version that satisfies the requirement azure-cognitiveservices-vision-customvision==3.0.0 (from versions: none)
ERROR: No matching distribution found for azure-cognitiveservices-vision-customvision==3.0.0
```

Dicho problema proviene del hecho de que Docker no esté utilizando el servidor DNS apropiado. Se puede Puede solucionarlo de tres formas diferentes[55]:

- Agregar DNS de Google a su configuración local
  Modificando /etc/resolv.conf y agregando las siguientes líneas al final
  # Google IPv4 nameservers nameserver 8.8.8.8 nameserver 8.8.4.4

- Modificación de la configuración de Docker como servidor de nombres predeterminado en el archivo /etc/resolv.conf.

  Para especificar un servidor DNS para que lo use Docker:

[55] https://qastack.mx/programming/28668180/cant-install-pip-packages-inside-a-docker-container-with-ubuntu

1. Log into Ubuntu as a user with sudo privileges.
2. Open the /etc/default/docker file for editing :

   $ sudo nano /etc/default/docker

3. Add the following setting for Docker.

   DOCKER_OPTS="--dns 8.8.8.8"

4. Save and close the file.
5. Restart the Docker daemon :

   $ sudo systemctl restart docker

- Usar un parámetro al ejecutar Docker
  Cuando ejecute Docker, simplemente agregue el siguiente parámetro: --dns 8.8.8.8

Para comprobar que efectivamente hemos creado la imagen y está alojada en nuestro repositorio interno desde la consola de Ubuntu ejecutamos:

```
$ sudo Docker images
```

```
(venv) diego@diegoVM:~/flask_dir$ sudo docker images
[sudo] contraseña para diego:
REPOSITORY            TAG             IMAGE ID       CREATED         SIZE
app1                  latest          b7e05aeea31f   2 months ago    57.4MB
cesar                 latest          b7e05aeea31f   2 months ago    57.4MB
openpipe5             latest          4a6303d7a497   2 months ago    57.4MB
openpipe3             latest          502104e0599e   2 months ago    57.4MB
openpipe4             latest          502104e0599e   2 months ago    57.4MB
<none>                <none>          7046e9f6d167   2 months ago    57.4MB
openpipe2             latest          f47206354011   2 months ago    57.4MB
openpipe1             latest          b3d458d062b7   2 months ago    57.4MB
<none>                <none>          ea6375570cd5   2 months ago    57.4MB
<none>                <none>          10a1fa02f3fa   2 months ago    57.4MB
<none>                <none>          bcd60d81cd15   2 months ago    57.4MB
<none>                <none>          1b0f700f5b86   2 months ago    57.4MB
<none>                <none>          c4668c244c20   2 months ago    57.4MB
<none>                <none>          638e2e15234b   2 months ago    57.4MB
<none>                <none>          f1e2b1ad1b95   2 months ago    57.4MB
<none>                <none>          5789a454148e   2 months ago    57.4MB
<none>                <none>          4f97fe8640b7   2 months ago    57.4MB
<none>                <none>          9cc49899617e   2 months ago    57.4MB
<none>                <none>          9785914fa210   2 months ago    57.4MB
pflask                latest          d536008369b1   2 months ago    57.4MB
flask/flask_docker2   latest          d536008369b1   2 months ago    57.4MB
flask/flask_docker1   latest          685a2609c0e8   2 months ago    57.4MB
<none>                <none>          8713ad12f174   2 months ago    57.4MB
<none>                <none>          55e8073451cd   2 months ago    57.4MB
flask/flask_docker    latest          1386a1cfd771   2 months ago    57.4MB
<none>                <none>          66f0fc180b2f   2 months ago    57.4MB
<none>                <none>          44851f0523f8   2 months ago    57.4MB
<none>                <none>          735935823037   2 months ago    45MB
<none>                <none>          e68cc466de02   2 months ago    45MB
python                latest          cecf555903c6   2 months ago    917MB
ubuntu                latest          d13c942271d6   2 months ago    72.8MB
hello-world           latest          feb5d9fea6a5   6 months ago    13.3kB
python                3.9.5-alpine    25d7d3156527   9 months ago    45MB
(venv) diego@diegoVM:~/flask_dir$
```

En la imagen anterior vemos un ejemplo de los que nos devolvería la consulta de las imágenes creadas y alojadas en nuestro repositorio local de Docker.

A continuación, vamos a ejecutar el contenedor basado en la imagen que hemos creado ejecutando el comando:

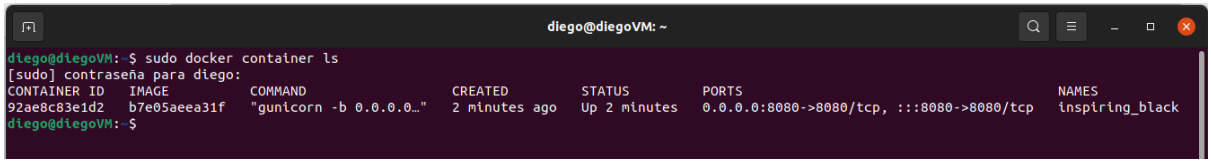
```
$ sudo Docker run -d -p 8080:8080  <ID imagen>
```

```
diego@diegoVM:~$ sudo docker run -d -p 8080:8080 b7e05aeea31f
e9f2d4c5c06e62ef84fe20b9cbd0f25c7c790bdcf4977e35f9c07b94df3b1620
diego@diegoVM:~$
```

El parámetro “-d” indica que se ejecute en segundo plano, para no bloquear la terminal.
Después de esto, debemos especificar en qué puerto se ejecuta nuestra aplicación Flask dentro de nuestro contenedor, hemos especificado en el Dockerfile 8080 y estará expuesto a nuestro puerto 8080 del localhost del PC

A continuación, podemos comprobar que contenedores se están ejecutando con la siguiente instrucción:

```
$ sudo Docker container ls
```



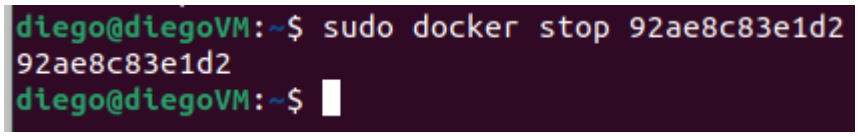
La aplicación ya se estaría ejecutando dentro de su contenedor, como hemos expuesto el puerto 8080 podemos acceder a la misma indicando en el navegador:

```
http://0.0.0.0:8080
```

Figura 6-8 Página de inicio de aplicación ejemplo IE

Una vez terminada la comprobación de nuestra aplicación podemos parar el contenedor ejecutando la siguiente instrucción:

$ sudo docker stop <ID del contenedor>



Command	Description
docker build	Build an image from a Dockerfile
docker save	Save built docker images
docker config	Manage Docker configs
docker container	Manage containers
docker image	Manage images
docker images	List images
docker ps	List containers
docker rm	Remove one or more containers
docker rmi	Remove one or more images
docker run	Run a command in a new container

Figura 6-9 Resumen de los comandos básicos de Docker. Fuente: Siemens Industry Image Database 4.11

6.2.2 Técnicas de programación y buenas prácticas

A la hora de desarrollar aplicaciones de IE lo primero es tener claro el dispositivo en el que van a ejecutrse y sus limitaciones.
En el caso de los Unified Comfort Panels (UCPs) el límite de memoría de trabajo disponible son 800 MB.

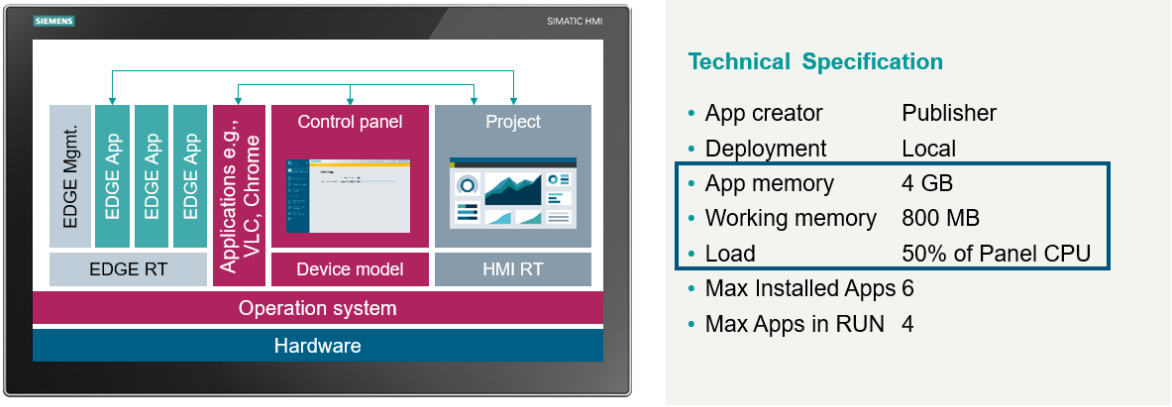


Figura 6-10 Recursos disponibles en una UCP. Fuente: Siemens Industry Image Database 4.11

Figura 6-11 Ejemplo: reducir el tamaño total de las imágenes usando una imagen base optimizada. Fuente: Siemens Industry Image Database 4.11

Comprobar el tamaño de la imagen

Docker Hub muestra el tamaño comprimido de cada imagen:

Una vez que haya extraído la imagen, podemos verificar su tamaño real usando el comando de imágenes de Docker:

```
docker images
```

Después buscamos la entrada de la imagen que acabamos de descargar

```
REPOSITORY      TAG                 IMAGE ID            CREATED             SIZE
 ubuntu         20.10               ccc6e87d482b        4 days ago          64.2MB
```

Comprobar requisitos de RAM de la imagen:

Para hacer una estimación de la RAM requerida de su aplicación, primero ejecutamos los contenedores localmente y usamos el comando docker stats.

```
docker stats
```

Este comando nos devuelve algo similar a esto:

```
CONTAINER ID        NAME                CPU %               MEM USAGE / LIMIT     MEM %
b95a83497c91        container1          0.28%               5.629MiB / 1.952GiB   0.28%
```

MEM USAGE= la memoria total que usa el contenedor
LIMIT = la cantidad total de memoria que se permite usar
CPU % / MEM % = el porcentaje de la CPU del host y la memoria que utiliza el contenedor

Buenas prácticas:

- Utilizar siempre imágenes oficiales/certificadas de Docker para uso en productivo.
- Cada imagen siempre debe tener una versión, se deben utilizar etiquetas claras y definidas.
- Mantener las imágenes contenidas en tamaño (por ejemplo, usar alpine), evite instalar paquetes innecesarios. Es preferible dos imágenes que una muy grande.
- Todo el archivo Docker debe comentarse tanto como sea posible.
- Usar compilaciones de varias etapas para reducir el tamaño del contenedor[56].
- Solo exponer los puertos necesarios.
- Dividir las aplicaciones en Microservicios. Si es posible, solo una tarea por contenedor.[57]
- Evitaremos usar un usuario raíz dentro de un contenedor para evitar riesgos de seguridad. No ejecutaremos contenedores en modo privilegiado a menos que no tengamos otra opción.

Seguridad y actualizaciones

Los contenedores Docker aportan un entorno más seguro para las cargas de trabajo que los modelos tradicionales de servidor y máquina virtual (VM), porque divide las aplicaciones en componentes mucho más pequeños y poco acoplados, cada uno aislado entre sí.
Entre las ventajas relativas a la seguridad cabría destacar:

- Superficie de ataque significativamente reducida
- Hace que sea más difícil que una brecha se propague en caso de un ataque

Por otro lado deberemos utilizar siempre que sea posible la última versión de una imagen disponible en un registro, si estubiera disponible, elijiremos versiones LTS (soporte a largo plazo) ya que se mantiene durante un período de tiempo más largo

6.3. Desarrollo con apps para IE

6.3.1 Comunicación vía WinCC Unified OpenPipe

Ya hemos comentado anteriormente que en este TFG estamos utilizando como dispositivo edge un Unified Comfort Panel, el cual se trata de un dispostivo de doble proposito, runtime de IE y runtime del HMI (WinCC Unified) por lo que la forma de conectar con las variables de campo difiere de la utilizada con un dispositivo edge dedicado donde se va a utilizar app específicas de conectividad para los diferentes protocolos (OPC UA, Profinet, Ethernet IP, Modbus...) y la app de Data Service y Data Bus.
En el caso de las apps desarrolladas para UCP la comunicación con los protocolos OT va a estar a cargo de la propia runtime del HMI que se comunicará con nuestras app a traves de un

[56] https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#use-multi-stage-builds
[57] https://microservices.io/patterns/microservices.html

"Pipe" o tuberia denominado WinCC Unified OpenPipe (OpenPipe en lo sucesivo).

Este método de comunicación tiene como gran ventaja que nos permite implementar el código de nuestra aplicación en cualquier lenguaje permitiéndonos la librería de comandos disponibles acceder tanto a las variables como a las alarmas de nuestro HMI.

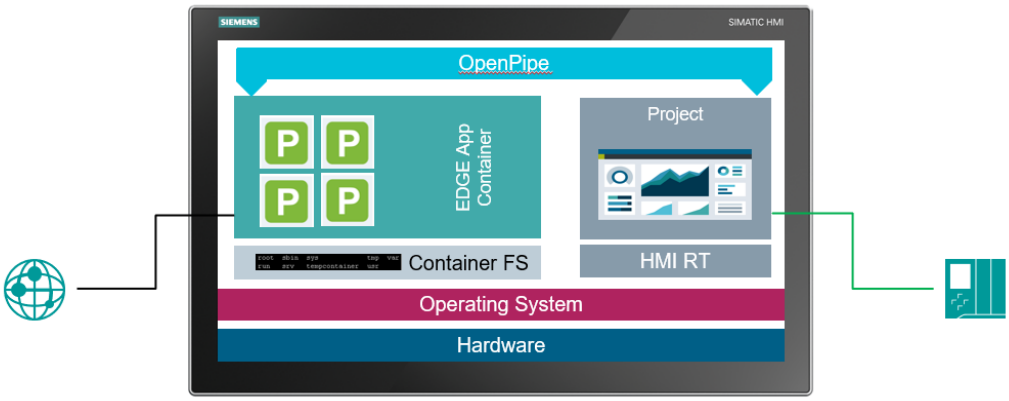


Figura 6-12. Openpipe Unified Comfort Panels. Fuente: Siemens Industry Image Database 4.11

Tecnología Pipe

Un Pipe es un flujo de datos con memoria de respaldo entre dos procesos que funciona según el principio FIFO (First In First Out). WinCC Unified RT (OpennessManager) genera un proceso. Crea el Pipe y procesa las peticiones (Requests) de la aplicación del cliente. El segundo proceso es la aplicación del cliente. Se conecta al Pipe a través del nombre, envía peticiones (Requests) y recibe respuestas (Responses).

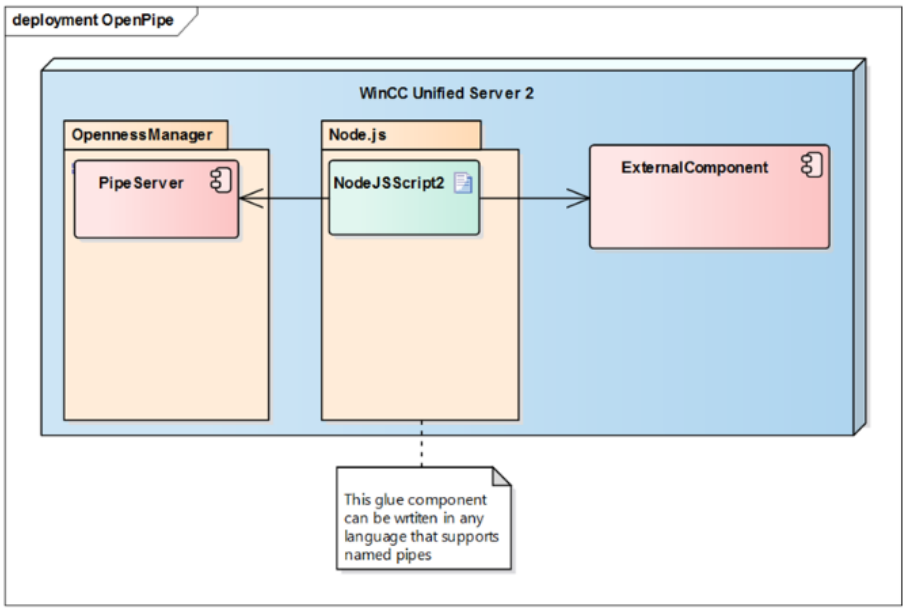


Figura 6-13 Figura 6 12. Openpipe Unified Comfort Panels. Fuente:WinCC Unified Open Pipe[58]

Nombre del Pipe:

[58] https://support.industry.siemens.com/cs/es/es/view/109778823

Para conectar con el Pipe desde nuestro código de aplicación, lo primero será indicar el nombre del Pipe que difiere de si estamos en un entorno Windows o en el caso de la UCP un entorno Linux.

- En Windows: "\\.\pipe\HmiRuntime"
- En Linux: "/tmp/HmiRuntime"

Cuando el Pipe está abierto se pueden enviar comandos de una línea que deben terminar con un salto de línea ("\n" o "\r\n"). Las Responses se devuelven a través de la misma instancia de Pipe.

Sintaxis: simple y avanzada

Para WinCC Unified Open Pipe están disponibles dos sintaxis:

- Sintaxis simple

  La utilizamos para trabajar con archivos batch simples (p. ej., en un CMD.exe o batch).

- Sintaxis avanzada

  La utilizamos para trabajar con scripts o lenguajes de programación que cuenten con un parser JSON, como, p. ej., Python, Node.js o Powershell.

Ejemplos:

Podemos encontrar ejemplos de código en el propio DVD de instalación de WinCC Unified en la siguiente ruta:

"Support\Openness\Siemens.Unified.Openness_SDK_<número de versión>.zip"

Si descomprimimos el archivo localmente en la subcarpeta "OpenPipe\Samples" encontraremos ejemplos de uso de WinCC Unified Open Pipe.

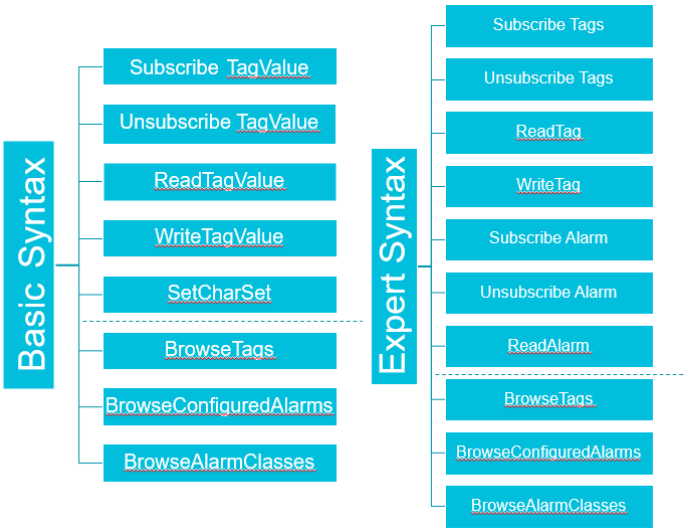


Figura 6-14 Sintaxis OpenPipe. Fuente SIMATIC HMI WinCC Unified Open Pipe

Ejemplo de sintaxis simple para comunicar con dos variables tag1 y tag2 de la UCP

```python
# standard libraries
import socket
import sys

pipeName = "/tmp/HmiRuntime"

try:
    pipe = socket.socket(socket.AF_UNIX, socket.SOCK_STREAM)
    pipe.connect(pipeName)

    ################### READ TAG ##############################################
    # send request to openpipe
    pipe.sendall(bytes('ReadTagValue tag1\n', 'utf-8'))

    resp = pipe.recv(10*1024)
    print(resp)
    print("Respuesta del Runtime sobre la tag leida: ", resp[1].decode("utf-8"))

    ########################## WRITE TAG ######################################
    write = "WriteTagValue tag2 192\n"
    pipe.sendall(write.encode("utf-8"))
    ###########################################################################


except:
    e = sys.exc_info()
    print( "Error: %s" % e[0])
    print(" %s" % e[1])
    pass
```

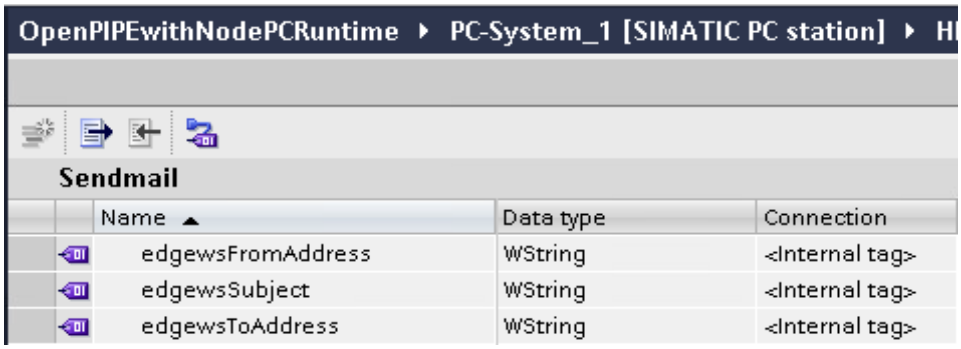
Ejemplo para comunicar con con 4 variables de la RT de WinCC Unified: RINT, RSTRING, WINT y WSTRING.

```python
# standard libraries
import time
import os, platform
import sys
import pprint

import win32pipe, win32file, win32api


def isDataInPipe(pipeHandler):
        try:
            buffer, bytesToRead, result = win32pipe.PeekNamedPipe(pipeHandler, 1)
        except win32api.error:
            print("Error in PeekNamedPipe")
        return len(buffer)

def getPipe():
    pipe = win32file.CreateFile(pipeName, win32file.GENERIC_READ | win32file.GENERIC_WRITE, 0, None, win32file.OPEN_EXISTING, 0, None, )
    err = win32api.GetLastError()
    print("Número de error al conectar: ", err)
    return pipe



pipeName = r'\\.\pipe\HmiRuntime'
```

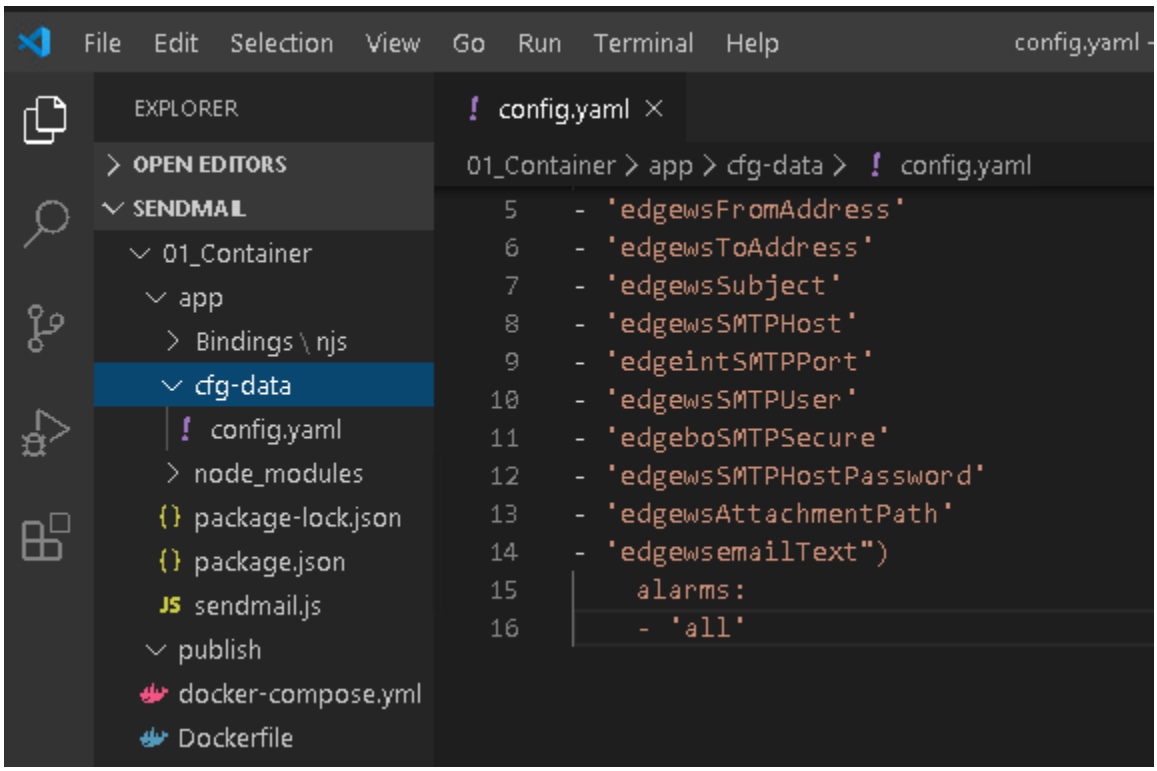
```
try:
    pipe = getPipe()
    pipeInfo = win32pipe.GetNamedPipeInfo(pipe)

    inbuffersize = pipeInfo[2]
    ################### READ TAG ##############################################
    res = win32file.WriteFile(pipe, bytes('ReadTagValue RINT\n', 'utf-8'))

    resp = win32file.ReadFile(pipe, inbuffersize)
    print(resp)
    print("Respuesta del Runtime sobre la tag leida: ", resp[1].decode("utf-8"))

    res = win32file.WriteFile(pipe, bytes('ReadTagValue RSTRING\n', 'utf-8'))

    resp = win32file.ReadFile(pipe, inbuffersize)
    print(resp)
    print("Respuesta del Runtime sobre la tag leida: ", resp[1].decode("utf-8"))


    ###################################################################
    #WRITE TAG
    write = "WriteTagValue WINT 666\n"
    res = win32file.WriteFile(pipe, write.encode("utf-8"))
    ###################################################################

    write = "WriteTagValue WSTRING qUE PACHA\n"
    res = win32file.WriteFile(pipe, write.encode("utf-8"))

    win32file.CloseHandle(pipe)

except win32pipe.error as e:
    if e.args[0] == 2:
        print("no pipe, trying again in one second")
        time.sleep(1)
    elif e.args[0] == 109:
        print("Pipe not working")
    else:
        print("Something goes wrong: " + e.args[2])
```

Es importante recalcar que los nombres de las variables en el proyecto del HMI en TIA Portal deben ser identicos a los que utilicemos en nuestro código para que pueda haber link entre ambos.
En las siguientes figuras se puede apreciar lo comentado:

```
var tagsToRead = [];   // Array to store to be read tags
tagsToRead.push("edgewsFromAddress");
tagsToRead.push("edgewsToAddress");
tagsToRead.push("edgewsSubject");
```

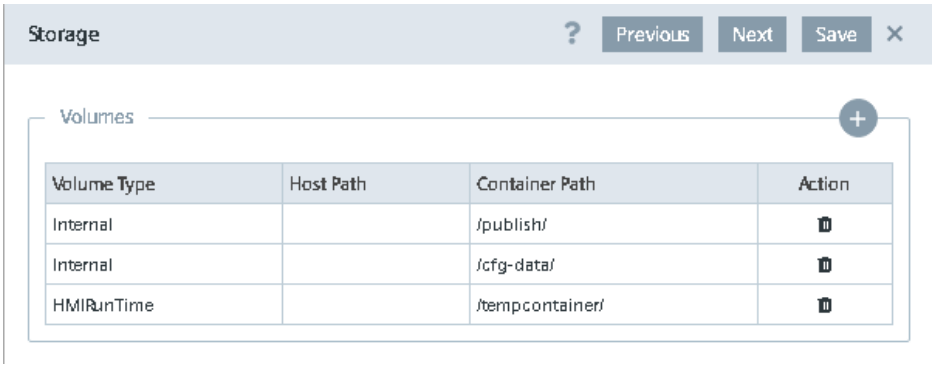


Para estandarizar nuestra aplicación y que cambios en los nombres en las variables del proyecto del HMI en TIA Portal no nos obliguen a modificar nuestro código es conveniente disponer de un fichero yaml de configuración donde indicaremos el nombre de todas las variables a las que accederemos vía OpenPipe.



Archivo de configuración que se leerá al inicio con el nombre de las variables/tags

A la hora de publicar la app en el IEAP deberemos indicar que vamos a hacer uso de la comunicación vía OpenPipe habilitando el siguiente directorio:



Simulación de la comunicación OpenPipe:
Es recomendable probar el código de la aplicación y su interconexión con la runtime de WinCC Unified o de la UCP simulada previo a su paquetización en un contenedor.

Pasos para la simulación:
1. Iniciar un PC-Runtime de WinCC Unified en TIA Portal
2. Ejecutamos el código con derechos de administrador
3. Comprobamos si la conexión está establecida y todo funciona como se desea

Si hemos preparado nuestra aplicación para ejecutarse en una UCP con S.O. Linux el código de la conexión OpenPipe difiere de la de Windows que estamos ejecutando en el entorno simulado por lo que es conveniente tener esto en cuenta en nuestro código previamente para discriminar por código si el S.O. es Linux o Windows.

```
39  function get_pipe_name() {
40      if (os.platform() === 'win32') {
41          return '\\\\.\\pipe\\HmiRuntime';
42      }
43      else {
44          return os.tmpdir() + '/HmiRuntime';
45      }
46  }
```

6.3.2 Comunicación vía Data Service OpenAPI

Este tipo de comunicación lo vamos a utilizar para comunciar nuestras apps cuando estas corran en dispositivos edge dedicados o en una UCP a partir de la versión de imagen de firmware V17 update 2.
Con la aplicación Data Service, se conectan otras aplicaciones, como Performance Insight, a IE Databus (MQTT Broker) o a Unified Comfort Panel (Open Pipe). En el Servicio de datos, podemos agrupar los datos y guardarlos durante un tiempo determinado. El IE Databus recibe los datos directamente de la planta con la ayuda de apps de conectividad como S7 Connector.

Descripción
La especificación OpenAPI del servicio de datos es un estándar para describir las interfaces de programación compatibles con REST (API). Con OpenAPI, podemosconectar la aplicación desarrollada por el usuario al Servicio de datos y acceder a las interfaces del mismo.

Requisito
La OpenAPI del servicio de datos está disponible en la red Docker de Industrial Edge Device-wide "proxy-redirect".

Para comunicarse con OpenAPI desde el servicio de datos, una aplicación debe definir esta red "externa" con el controlador "puente":

```
networks:
  proxy-redirect:
    external:
      name: proxy-redirect
    driver: bridge
```

Según el entorno, el servicio de datos está disponible bajo esta URL:

http://edgeappdataservice:4203

Kit de desarrollo Data Service para Industrial Edge
Se trata de una imagen de Docker que lanza la API de Data Service en el ordenador del desarrollador y permite simular el dispositivo Edge y hacer llamadas al mismo a traver del Data Service OpenAPI. Con esta herramienta, podemos desarrollar las apps completamente sin acceso a un Industrial Edge real todo integrado en el PC de desarrollo. No es necesario disponer de un dispositivo real que proporcione datos. Con el servicio de simulación, los datos de la planta se pueden simular.

Este kit lo tenemos disponible en el siguiente enlace:
https://support.industry.siemens.com/cs/es/es/view/109792717

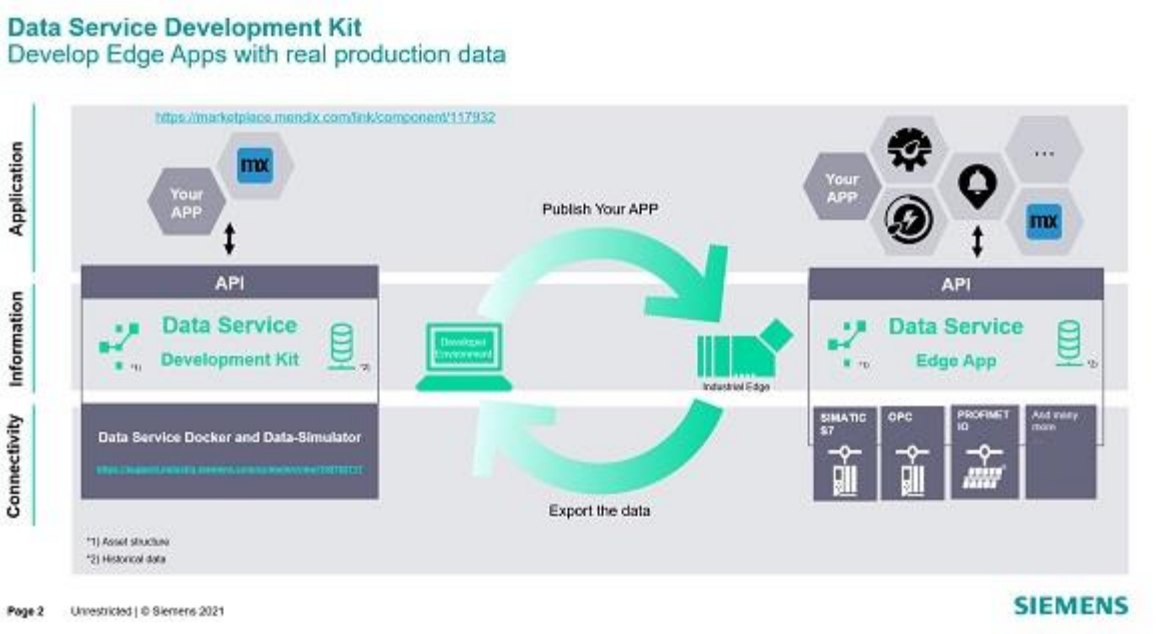


Figura 6-15 Kit Data Service para desarrolladores Industrial Edge. Fuente: Kit SDK Data Service[59]

[59] https://support.industry.siemens.com/cs/es/es/view/109792717

6.3.3 Industrial Edge Publisher

Siemens Industrial Edge Publisher es la herramienta que instalamos el PC de desarrollo para convertir imágenes de Docker en aplicaciones Industrial Edge que se pueden usar en dispositivos Siemens Industrial Edge.
Industrial Edge App Publisher está disponible para los sistemas operativos Windows y Linux.
La siguiente figura ilustra el proceso de conversión básico.

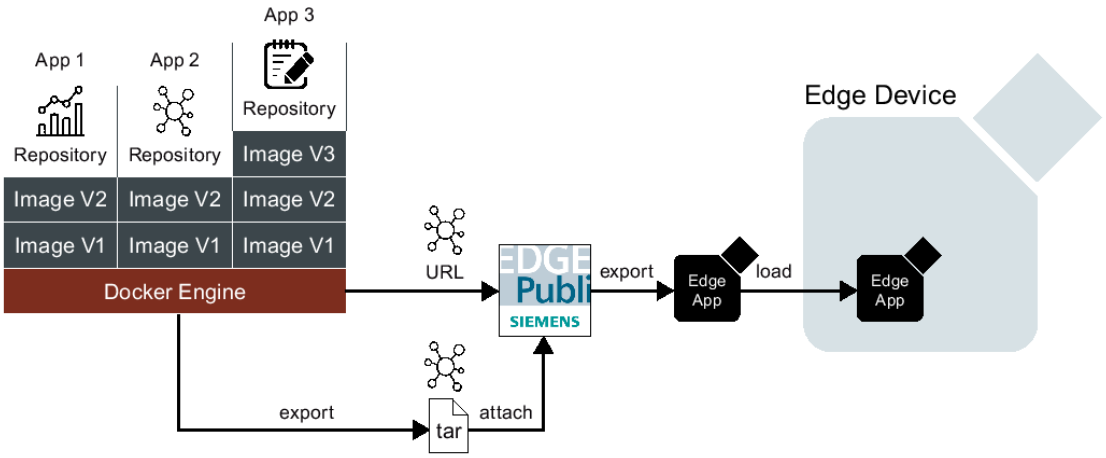


Figura 6-16 Publicación de imagenes con IEAP. Fuente: Siemens Industry Image Database 4.11

Industrial Edge Publisher convierte versiones seleccionadas de imágenes de Docker desde un archivo tar o a través de una URL de Docker Engine (previa exposición de la API de Docker como vimos anteriormente) a Industrial Edge Apps.
Luego, las aplicaciones Industrial Edge se pueden cargar en un dispositivo Edge.

Página de inicio:

1. Configuramos el motor de docker remoto con el puerto predeterminado 2375
2. Creamos una nueva aplicación o importamos aplicaciones ya existentes

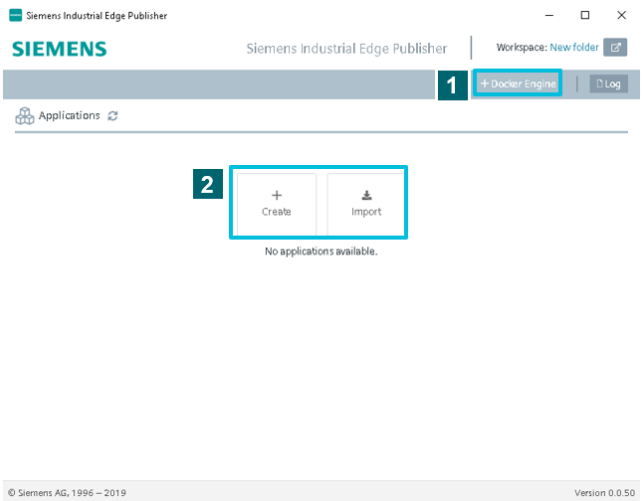


Figura 6-17 Captura IEAP

Crear una nueva aplicación:

1. Definimos un nombre de aplicación único, un nombre de repositorio y una descripción de la aplicación
2. Seleccionamos un icono individual que represente nuestra aplicación. También se pueden utilizar iconos personalizados.
3.

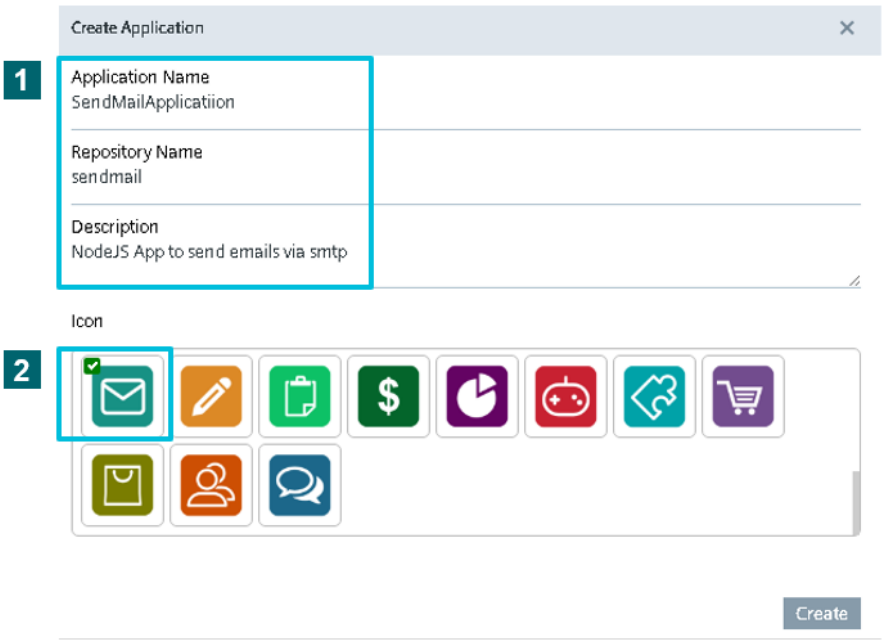


Figura 6-18 Captura IEAP

Definición de la aplicación:
Definimos la aplicación, por ejemplo, la versión o imagen de docker usada.

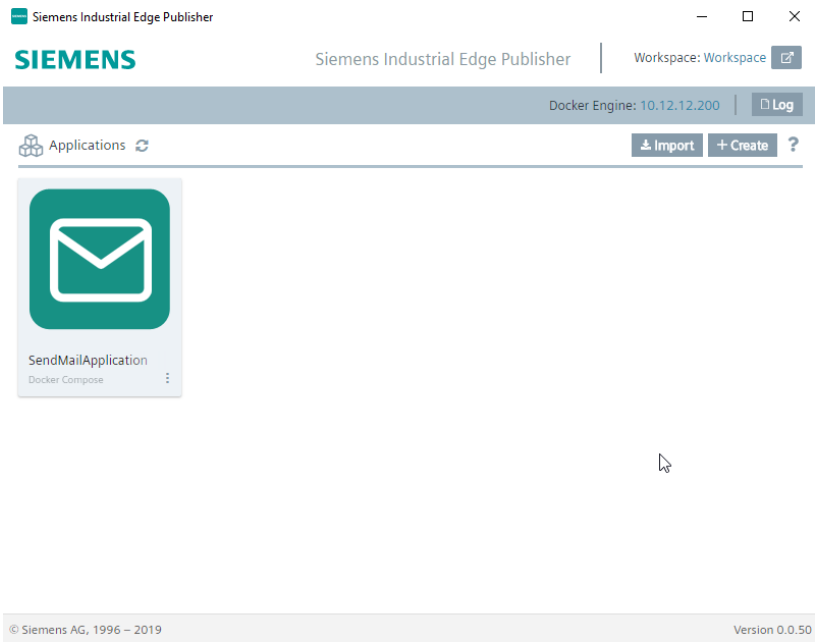


Figura 6-19 Captura IEAP

Versionado de la aplicación y Docker Compose

1. Agregamos una nueva versión de la aplicación
2. Seleccionamos la versión adecuada de Docker Compose.
   Ver: https://docs.docker.com/compose/
3. Después del reconocimiento de la versión, se configura el servicio de esta versión.
4.

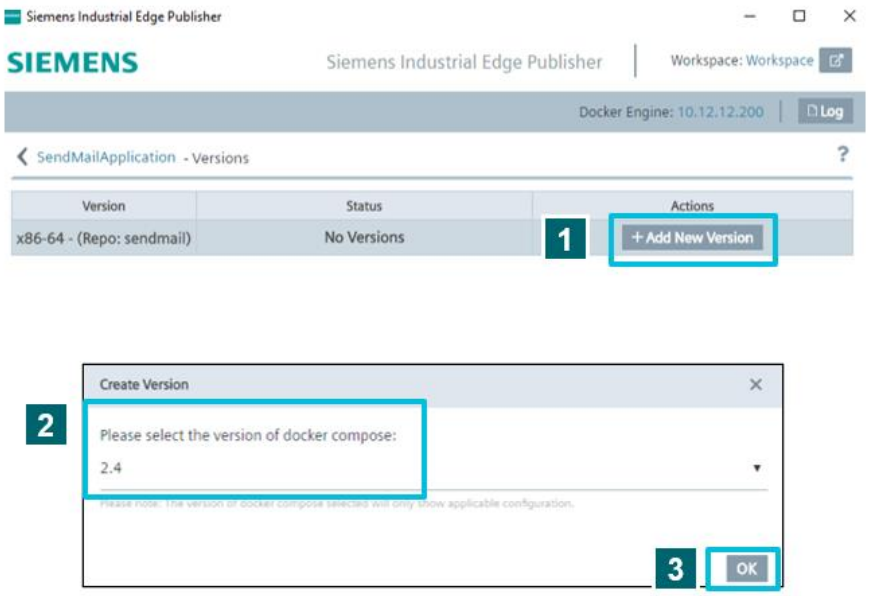


Figura 6-20 Captura IEAP

Definimos servicios de la aplicación:

1. Importamos la configuración ya existente desde el archivo YAML.
   https://docs.docker.com/compose/
2. Agregamos un nuevo servicio con todos los parámetros requeridos para la aplicación Edge y creamos el archivo YAML desde la configuración

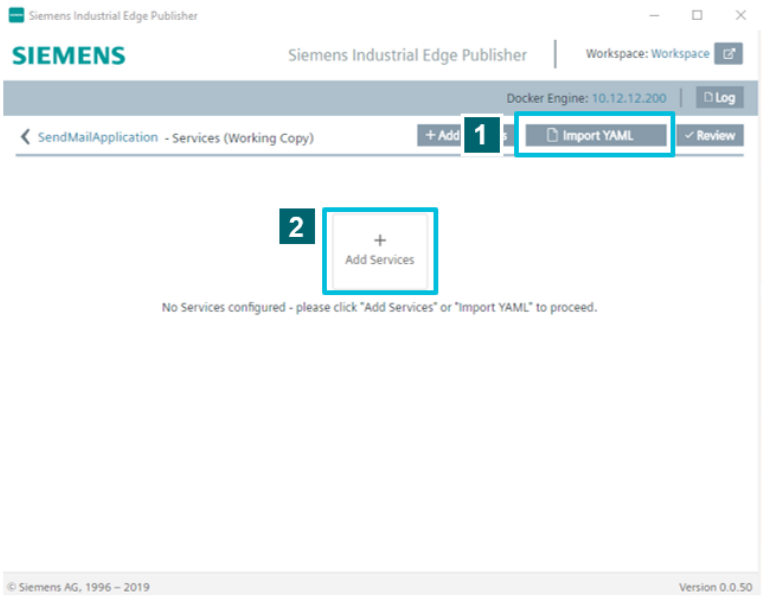


Figura 6-21 Captura IEAP

Configuramos servicios de la aplicación:

1. Seleccionamos la configuración general, por ejemplo, el nombre de la imagen.
2. Configuramos los ajustes de registro del contenedor
3. Agregamos el acceso de almacenamiento, por ejemplo, USB o OpenPipe
4. Introducimos la configuración de red, por ejemplo, puertos de comunicaciones
5. Ajustamos la configuración del sistema, por ejemplo, el límite de memoria

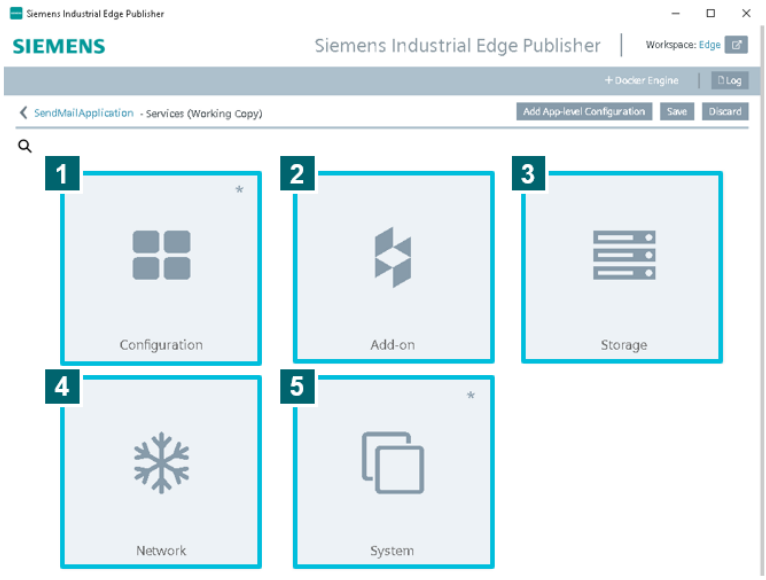


Figura 6-22 Captura IEAP

Parametrización:

1. Definimos el nombre del servicio dentro de la aplicación
2. Seleccionamos el archivo .tar manualmente o la imagen si Docker está conectado
3. Definimos el comportamiento de reinicio de la aplicación en caso de reinicio del Panel fallo
4. Guardamos y/o continuamos con el siguiente parámetro.

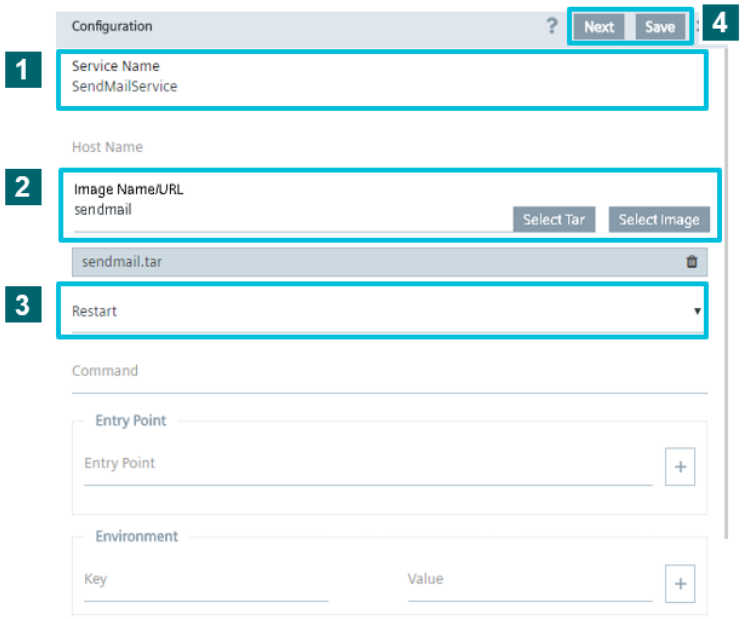


Figura 6-23 Captura IEAP

1. Configuración de registro para el servicio
2. Especificamos las opciones de registro para el controlador de registro con claves de opción
3. Redes a las que unirse, entradas de referencia y la clave de redes a nivel de aplicación
4. Guardar o continuar con el siguiente juego de parámetros.

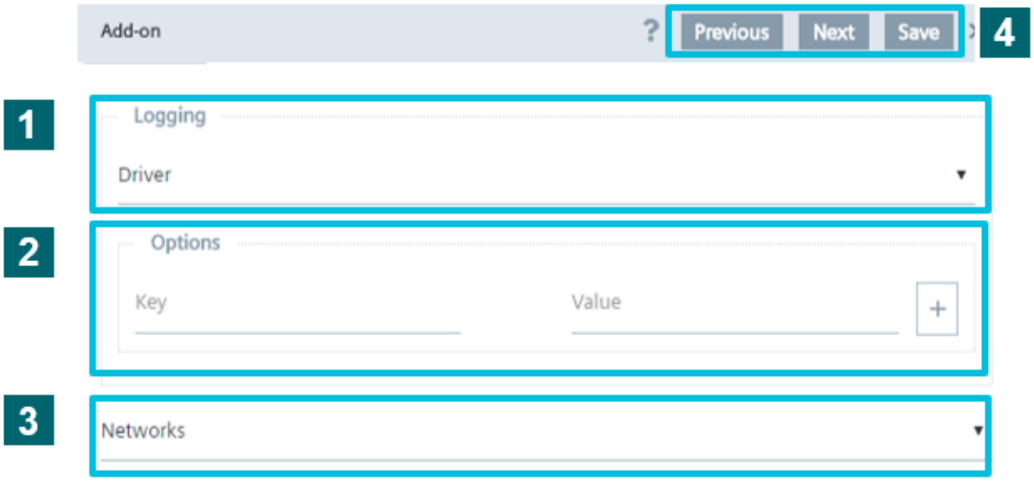


Figura 6-24 Captura IEAP: Parámetros de registro de logs

1. Agregamos diferentes volúmenes a la aplicación.
2. Agregamos el volumen OpenPipe Socket para establecer la comunicación entre la aplicación y el tiempo de ejecución (runtime) o el almacenamiento de datos para acceder al medio de almacenamiento USB y SD
3. Guardar y/o continúar con el siguiente juego de parámetros.

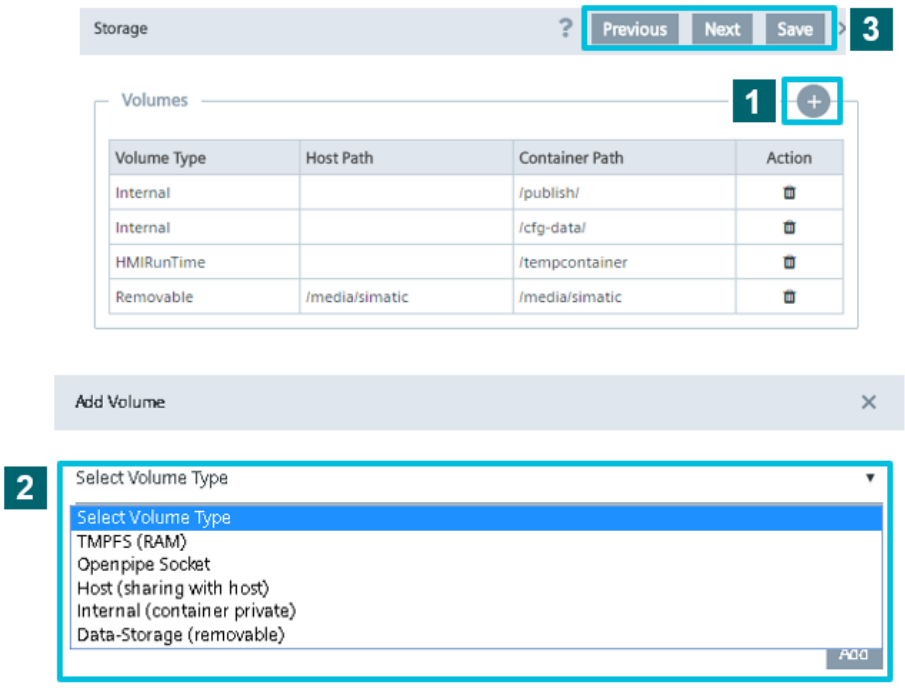


Figura 6-25 Captura IEAP: Parámetros de almacenamiento

1. Ajustamos la configuración general de la red y el DNS
2. Introducimos los puertos necesarios para la comunicación de las aplicaciones por ejemplo 25, 465 y 587 para la funcionalidad SMTP
3. Los puertos personalizados que comienzan con 30000 están disponibles para, por ejemplo, la interfaz web de la aplicación
4. Guardar o continúar con el siguiente juego de parámetros.

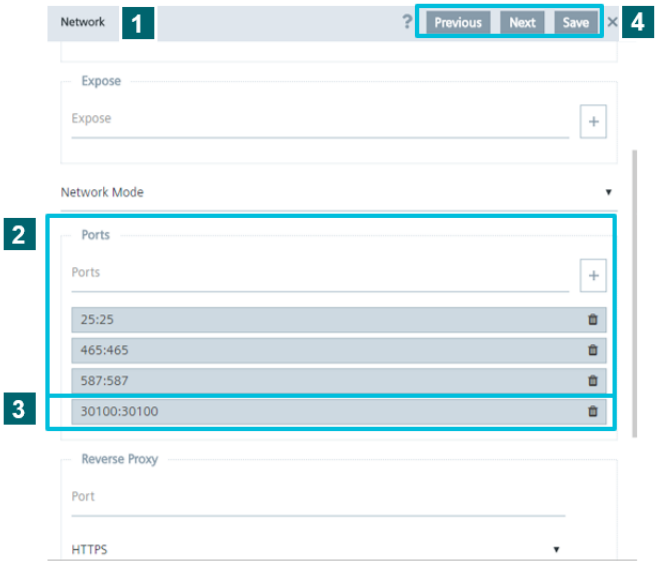


Figura 6-26 Captura IEAP: Parámetros de red

1. Ajustamos el límite de memoria de la aplicación. Es necesario tener en cuenta la limitación de la memoria total
2. Guardamos todos los ajustes de configuración e iniciamos la revisión.

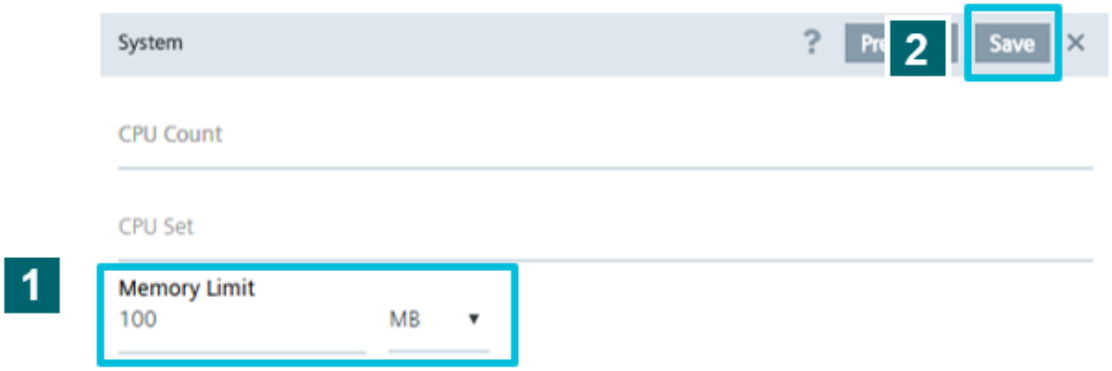


Figura 6-27 Captura IEAP: Parámetros de sistema

Revisamos la configuración de la parametrización:

1. Verificamos dos veces todos los ajustes de configuración
2. Verificamos las advertencias sobre los puertos comunes configurados.
3. Revisión completa.

Figura 6-28 Captura IEAP

Creamos la app desde YAML y una imagen de Docker:

1. Creamos la aplicación final a partir de la configuración YAML y la imagen Docker
2. Verificamos la URL para acceder a la interfaz web personalizada (Web UI)
3. Seleccionamos "Permitir el acceso a la aplicación sin iniciar sesión" si fuera necesario

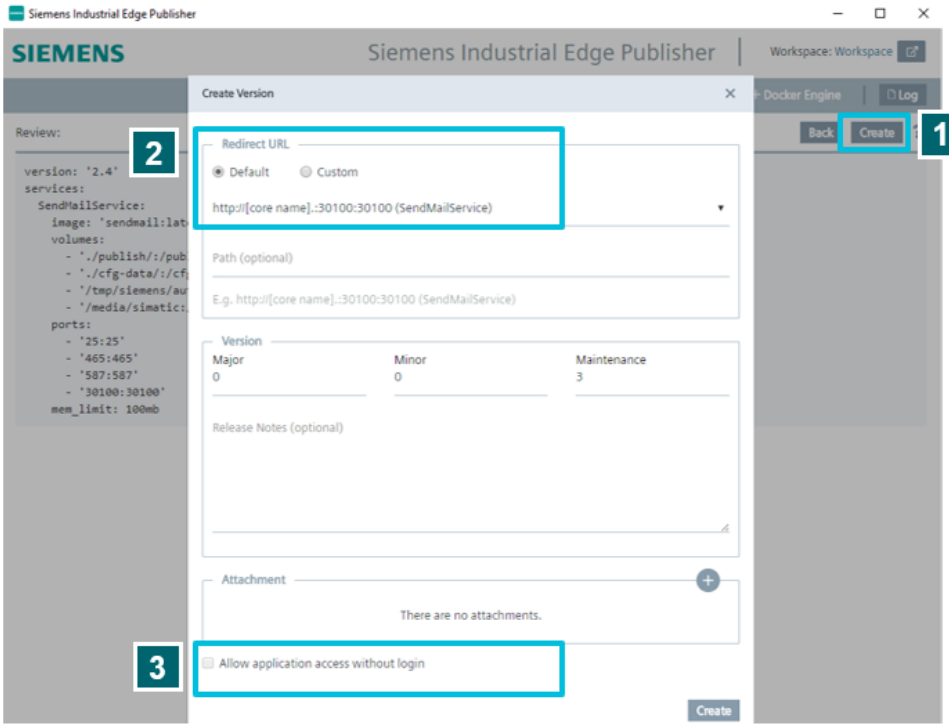


Figura 6-29 Captura IEAP

Exportamos el fichero .app:

1. Exportamos la aplicación en formato .app para instalar en Unified Comfort Panel
2. El manejo de versiones se realiza automáticamente.
3. Seleccionamos la ruta de exportación

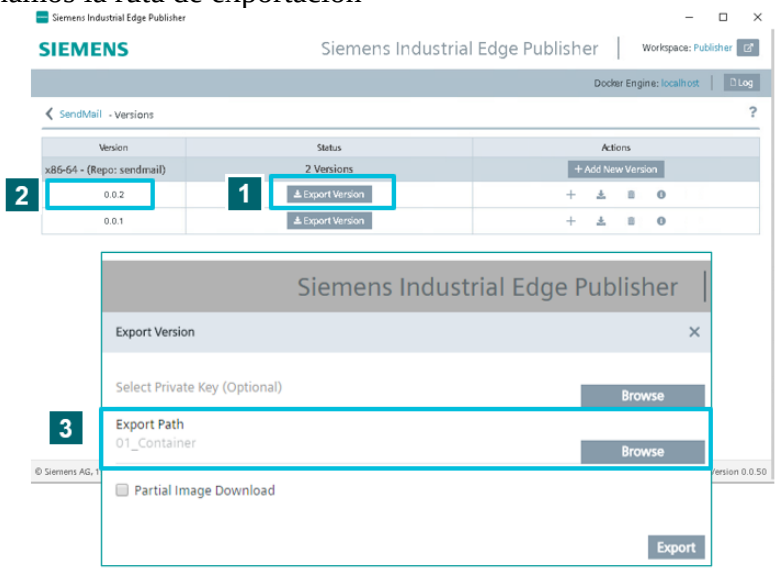


Figura 6-30 Captura IEAP

6.4. Depuración y diagnostico

6.4.1 Diagnóstico y depuración con Industrial Edge Management

Industrial Edge Management ofrece múltiples opciones para la depuración y diagnostico de aplicaciones, vamos a repasar como se accede a ellas y se configuran:

Device & App Statistics

1. Rendimiento general del dispositivo, por ejemplo, consumo de memoria y CPU o almacenamiento disponible

2. Estadísticas de la aplicación, por ejemplo, aplicaciones instaladas y en ejecución o memoria usada en total

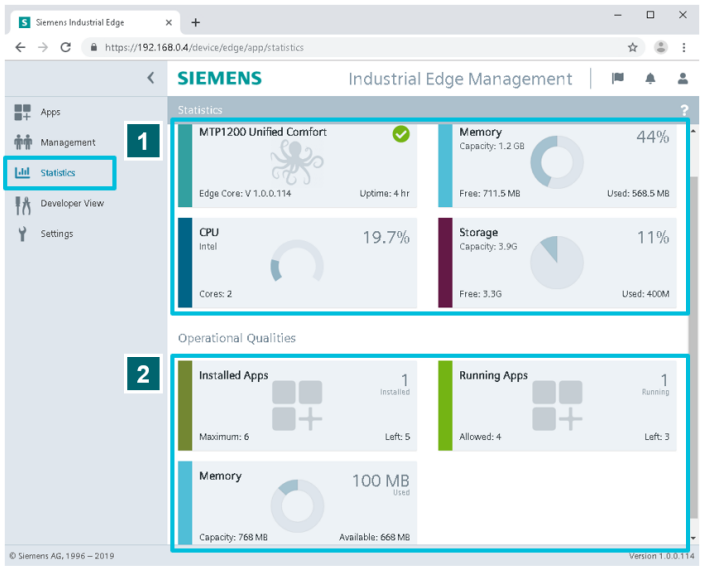


Figura 6-31 Captura IEM: Estadisticas

Developer View for installed App

1. Activación de “Developer View”.

2. Si hacemos clic en la imagen de la aplicación o en los nombres de las apss podemos ver información detallada como, puertos, sistemas de archivos…
   Información de la imagen: comando de entrada del contenedor e información de la capa de la imagen, etc.
   Información del contenedor: puertos expuestos del contenedor, uso de CPU, uso de memoria, etc.

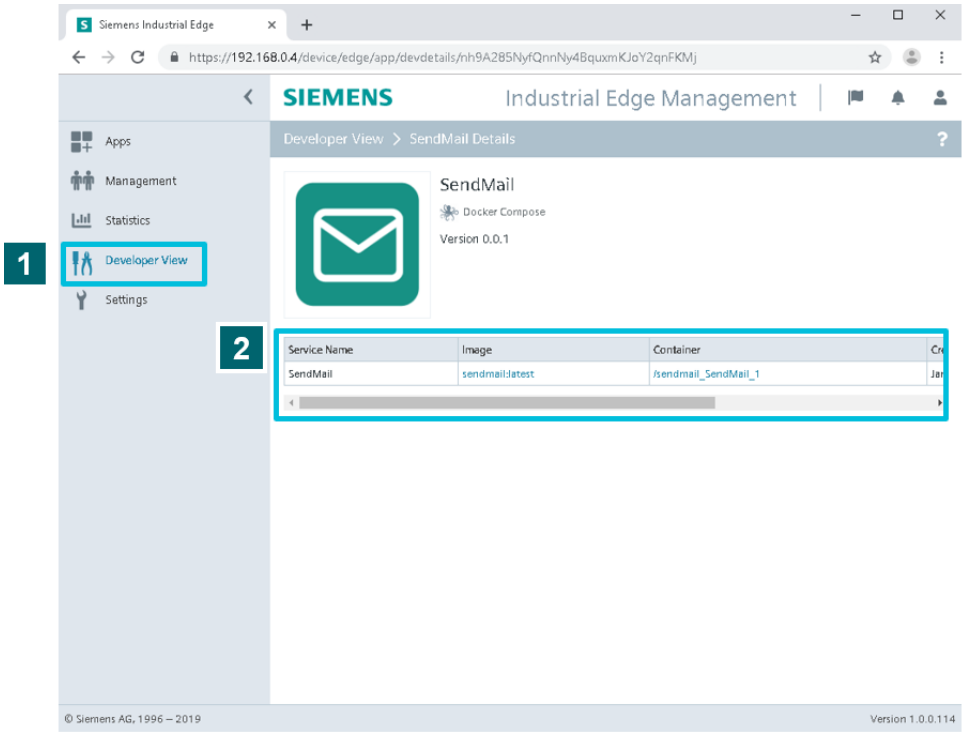


Figura 6-32 Captura IEM: Vista desarrollador

La vista de desarrollador permite verificar el estado, las estadísticas, los puertos, los sistemas de archivos para cada contenedor de nuestra aplicación.

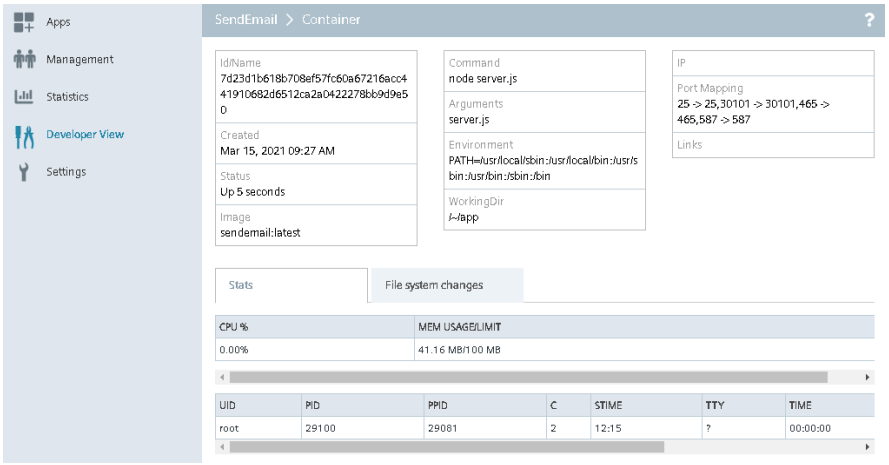


Figura 6-33 Captura IEM: Vista desarrollador detallada

Pero no se active por defecto, es necesario seguir los siguientes pasos para activarla.

1. Ir a Setting
2. Seleccionar la pestaña "System" y seleccionar Configuración 2Developer View Setting”
3. Marcar la casilla para "Enable Developer View"
4. Confirmamos con "Update"
5. Entonces aparece una nueva página "Developer View" en la navegación de la izquierda

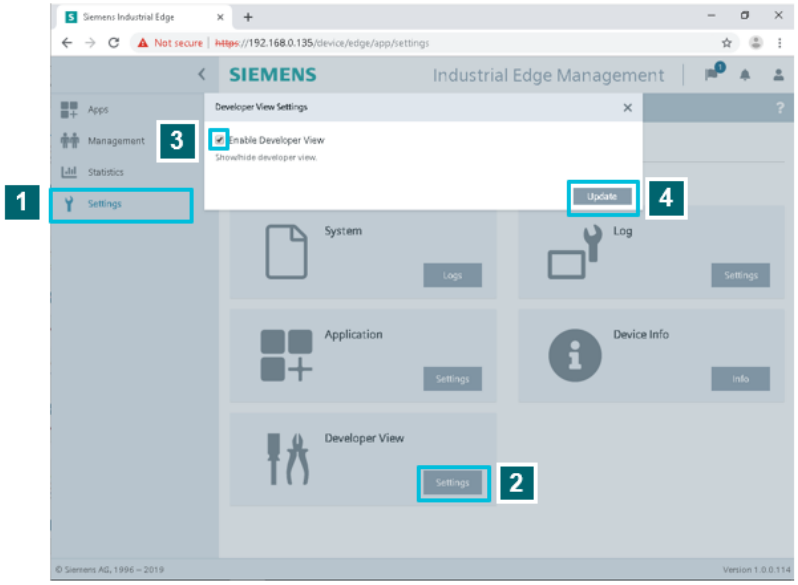


Figura 6-34 Captura IEM: Ajustes

Download System Logs

1. Vamos a "Setting" en Edge Management
2. Vamos a la pestaña "System"
3. Clic en "Log"
4. Se descargará un archivo TAR GZ, se descomprimirá y se abrirán los archivos de registro.

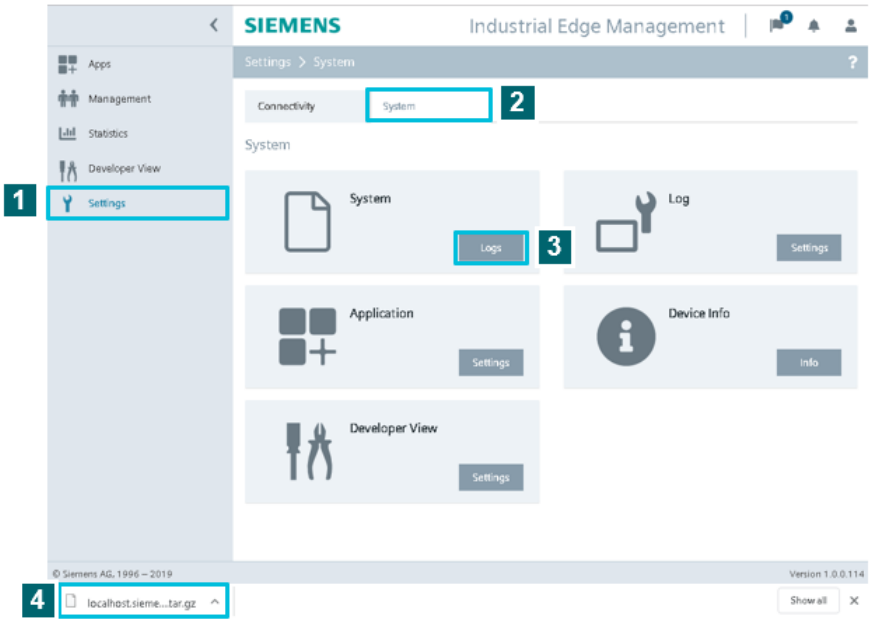


Figura 6-35 Captura IEM: Ajustes de sistema

Pero antes deberemos haberlos habilitados, por defecto no lo están y no es recomendable hacerlo una vez el equipo vaya a estar en productivo ya que acorta el tiempo de vida del dispositivo. Esto es debido a que la memoria flash interna tiene un número limitado de ciclos

de lectura y escritura y con los logs habilitados estamos aumentando el número de ciclo de acceso a memoria del dispositivo.

1. Vamos a "Setting" en Edge Management
2. Pestaña "System"
3. Clic en "Setting" en la sección de registro
4. Seleccionamos "Enable Logs" y actualizar la configuración de registro, esperamos hasta que los servicios edge se hayan reiniciado

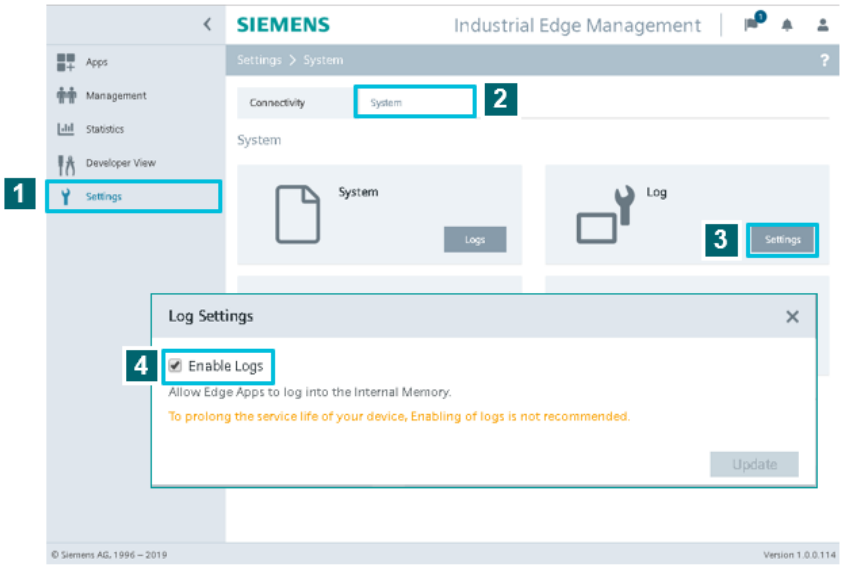


Figura 6-36 Captura IEM: Habilitación logs de sistema

Log file (in folder */var/log/*)	**Explanation**
nginx • nginx/access.log • nginx/error.log	These logs can be checked to find out reverse proxy related errors. • This contains logs of all NGINX access requests. • This contains logs of NGINX failure events.
journal **chrony**	(folders will be empty)
auth.log	Contains all authentication related events.
cron.log	Contains log from cronjobs.
debug	Contains information about debugging of system event.
fail2ban.log	Contains information about failure access attempts.

Figura 6-37 Contenido Log file (1/2) Fuente: Siemens Industry Image Database 4.11

Log file (in folder */var/log/*)	**Explanation**
kern.log	Contains information logged by the kernel.
messages syslog	Contain generic system activity events, logs. These logs can be checked to find out general system issue.
user.log	Contains information about all user level logs.

Figura 6-38 Contenido Log File (2/2). Fuente: Siemens Industry Image Database 4.11

Download App Logs

Con esta funcionalidad del IEM podremos descargar los registros de una aplicación directamente desde Edge Management e inspeccionar los archivos de registro de los contenedores.

Lo registros de log de una app están disponibles directamente sobre el ícono de la aplicación respectiva, estos son probablemente los registros más interesantes cuando intentamos depurar nuestras aplicaciones.

Podemos descargar tanto registros de aplicaciones oficiales/Siemens, así como de las aplicaciones personalizadas que desarrollemos por nuestra cuenta si se configura en el archivo docker-compose

Es importante que en el archivo docker-compose especifiquemos el nombre del contenedor y la opción de registro donde definiremos que registros estarán en un formato específico (como json, por ejemplo) y algunas otras opciones como la cantidad máxima de archivos de registro y la cantidad máxima de memoria que estos registros pueden consumir y así sucesivamente...

Todo esto lo estudiaremos en el siguiente apartado.

1. Vamos a "Apps" en Edge Management
2. Clic en los tres puntos en la parte inferior derecha de la aplicación
3. Seleccionamos " Download Logs"
4. Entonces se descarga un archivo TAR que contiene los archivos de registro.

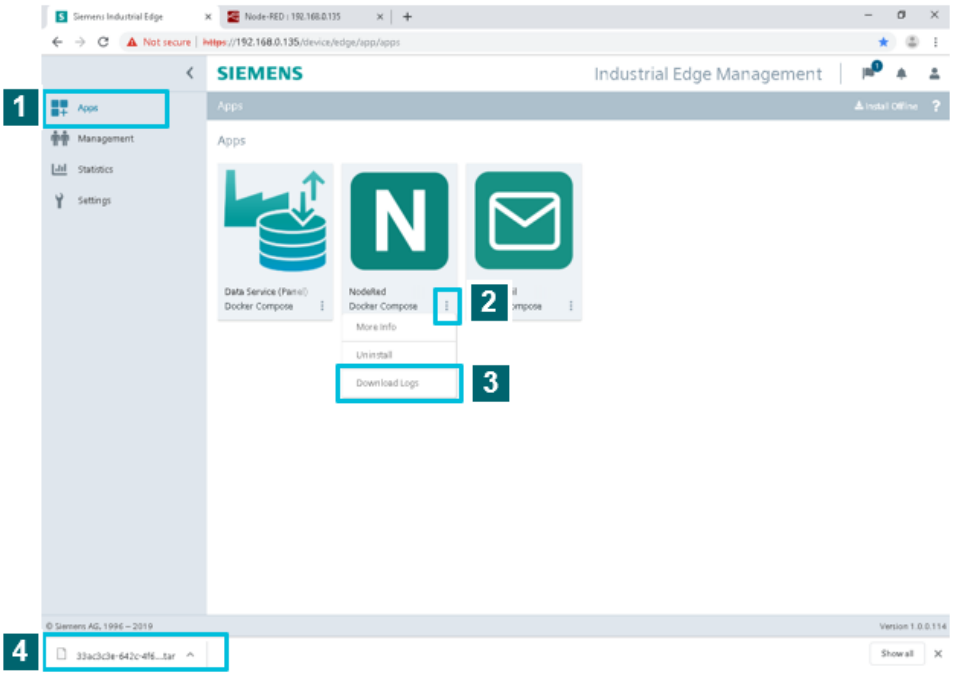


Figura 6-39 Captura IEM: Aplicaciones

1. Descomprimimos el archivo TAR descargado (la estructura del archivo y los nombres de las carpetas corresponden a los nombres/ID de contenedor de la aplicación)
2. Abrimos los archivos de registro para inspeccionarlos.

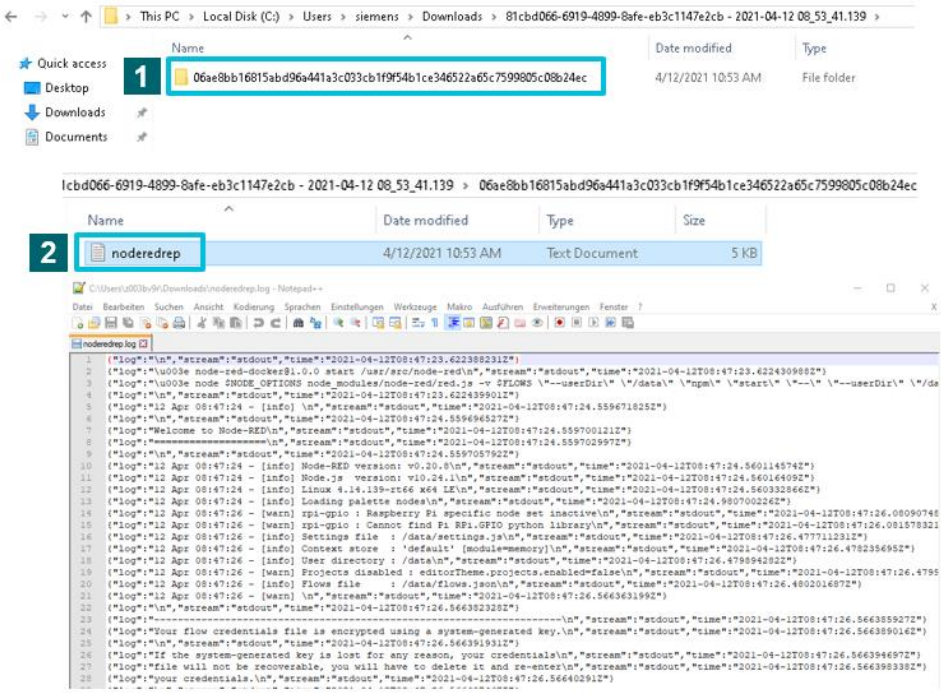


Figura 6-40 Logs de aplicaciones

Los registros de Edge Apps contendrán de forma predeterminada los registros de cada contenedor en ejecución, mostrando el stderr y stdout del contenedor, similar a ejecutar el comando en una terminal.
El controlador de registro de la aplicación accederá a las transmisiones stderr y stdout y las escribirá en los archivos de registro
De forma predeterminada, Docker captura la salida estándar (y el error estándar) de todos sus contenedores y los escribe en archivos usando el formato JSON. El formato JSON anota cada línea con su origen (stdout o stderr) y su marca de tiempo. Cada archivo de registro contendrá información relativa a un solo contenedor.

6.4.2 Configurar el log interno de las apps

Configurar nuestras apps para que generen un log interno de eventos es una buena practica que nos va a aportar entre otras las siguientes ventajas:

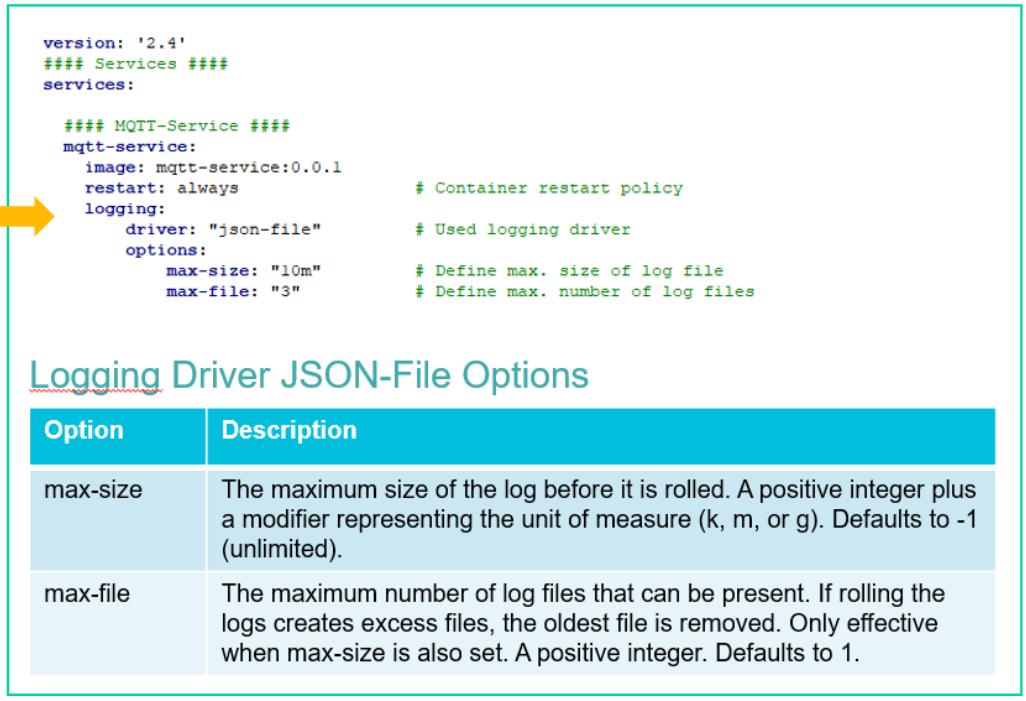
- Diagnostico avanzado: Encontrar la causa del problema cuando algo no va bien en nuestra aplicación
- Transparencia: podremos ver el estado actual de nuestro Equipo
- Seguridad: el registro puede ayudarnos a monitorizar el acceso no autorizado en nuestra aplicación. Lo más probable es, por ejemplo, intentos de inicio de sesión root cuyo acceso ha sido denegado, dichos intentos se registrarán con un sello de tiempo.

Si se trata de una app oficial de Siemens, el registro de log será importante para que cuando tengamos que contactar con el servicio técnico estos puedan tener la información suficiente para averiguar de donde proviene el problema.

Los archivos de registro pueden ocupar mucho espacio y va a ser necesario limitarlos, entre las opciones que ofrece el controlador de archivos json, podemos especificar la cantidad máxima de archivos de registro y la cantidad máxima de memoria que estos registros pueden consumir

1. Configuramos el controlador de registro de aplicaciones predeterminado de Industrial Edge para limitar el uso de almacenamiento en Edge Management y Unified Comfort Panel

2. Este se descarga directamente desde la interfaz de usuario de Industrial Edge Management en Unified Comfort Panel



```
version: '2.4'
#### Services ####
services:

  #### MQTT-Service ####
  mqtt-service:
    image: mqtt-service:0.0.1
    restart: always                 # Container restart policy
    logging:
        driver: "json-file"         # Used logging driver
        options:
            max-size: "10m"         # Define max. size of log file
            max-file: "3"           # Define max. number of log files
```

Logging Driver JSON-File Options

Option	Description
max-size	The maximum size of the log before it is rolled. A positive integer plus a modifier representing the unit of measure (k, m, or g). Defaults to -1 (unlimited).
max-file	The maximum number of log files that can be present. If rolling the logs creates excess files, the oldest file is removed. Only effective when max-size is also set. A positive integer. Defaults to 1.

Figura 6-41 Ejemplo de archivo Docker Compose con controlador de registro. Fuente: Siemens Industry Image Database 4.11

También podemos especificar la opción de registro para el servicio mediante el asistente de publicación de IEAP.
En la sección de registro en IEAP podemos especificar estos valores directamente aquí y una vez que seleccionamos revisarlos, esta configuración también se guarda en el archivo docker-compose.
Luego, estos registros se pueden descargar desde el dispositivo Edge en el ícono de la aplicación.

1. Especificamos el controlador de registro para el servicio.

2. Especificamos las opciones de registro para el controlador de registro con claves de opción (no debe olvidarsenos agregarlas haciendo clic en "+")

3. Después de completar la configuración a través del asistente, el editor genera automáticamente un archivo docker-compose que incluye nuestra configuración para las opciones de registro.

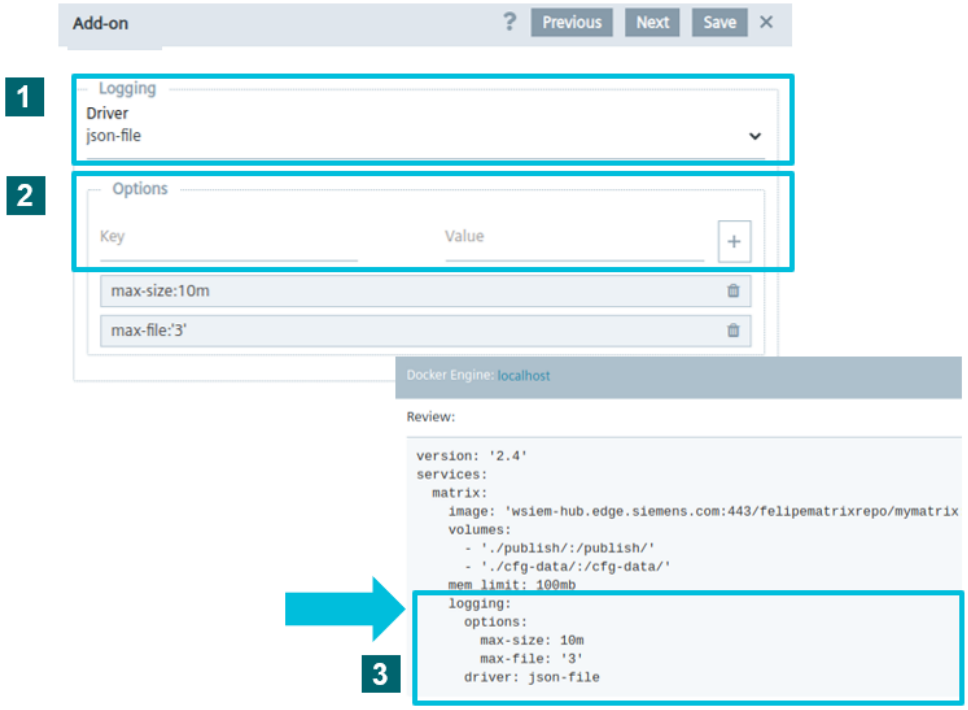


Figura 6-42 Asistente del IE App Publisher (IEAP)

Docker soporta diferentes formatos para los registros[60] de logs que podemos configurar como podemos observar en la siguiente figura, pero de momento Siemens IE solo soporta formato json. Es importante tener esto en cuenta ya que, aunque podamos configurar otros solo funcionará con json.

.

Driver	Description
json-file	The logs are formatted as JSON. The **default** logging driver for Docker.
syslog	Writes logging messages to the **syslog** facility. The syslog daemon must be running on the host machine.
journald	Writes log messages to **journald**. The journald daemon must be running on the host machine.
gelf	Writes log messages to a Graylog Extended Log Format (GELF) endpoint such as **Graylog or Logstash**.
fluentd	Writes log messages to **fluentd** (forward input). The fluentd daemon must be running on the host machine
awslogs	Writes log messages to **Amazon CloudWatch** Logs.
spunk	Writes log messages to **splunk** using the HTTP Event Collector.
etwlogs	Writes log messages **as Event Tracing for Windows** (ETW) events. Only available on Windows platforms.
gcplogs	Writes log messages to **Google Cloud Platform** (GCP) Logging.
logentries	Writes log messages to **Rapid7 Logentries**.

Figura 6-43 Formatos de registros soportados por Docker. Fuente: Docker Inc.

[60] https://docs.docker.com/config/containers/logging/configure/#supported-logging-drivers

De forma predeterminada, Docker captura la salida estándar (y el error estándar) de todos sus contenedores y los escribe en archivos usando el formato JSON. El formato JSON anota cada línea con su origen (stdout o stderr) y su marca de tiempo. Cada archivo de registro contiene información sobre un solo contenedor.

```
logging:
  driver: "json-file"
  options:
    max-size: "10m"
    max-file: "3"
```

Figura 6-44 Ejemplo se archivo Docker Compose

```
{
  "log":"Log line is here\n",
  "stream":"stdout",
  "time":"2019-01-01T11:11:11.111111111Z"
}
```

Figura 6-45 Mensaje JSON

7. Visión Artificial con Azure Cognitive Services

7.1 Azure Cognitive Services

La Visión Artificial es una rama de la inteligencia artificial (IA) que explora el desarrollo de sistemas de IA que pueden "ver" el mundo, ya sea en tiempo real a través de una cámara o analizando imágenes y videos. Esto es posible gracias al hecho de que las imágenes digitales son esencialmente matrices numéricas de valores de píxeles, y podemos usar esos valores de píxeles como características para entrenar modelos de aprendizaje automático que pueden clasificar imágenes, detectar objetos discretos en una imagen o identificar texto.

Microsoft Azure incluye una serie de servicios cognitivos (Cognitive Services) que encapsulan funciones comunes de IA, incluidas algunas que pueden ayudar a crear soluciones de visión artificial.
El servicio cognitivo Computer Vision proporciona el punto de partida para nuestra aplicación de visión artificial en Azure. Este utiliza modelos de aprendizaje automático previamente entrenados para analizar imágenes y extraer información sobre ellas.

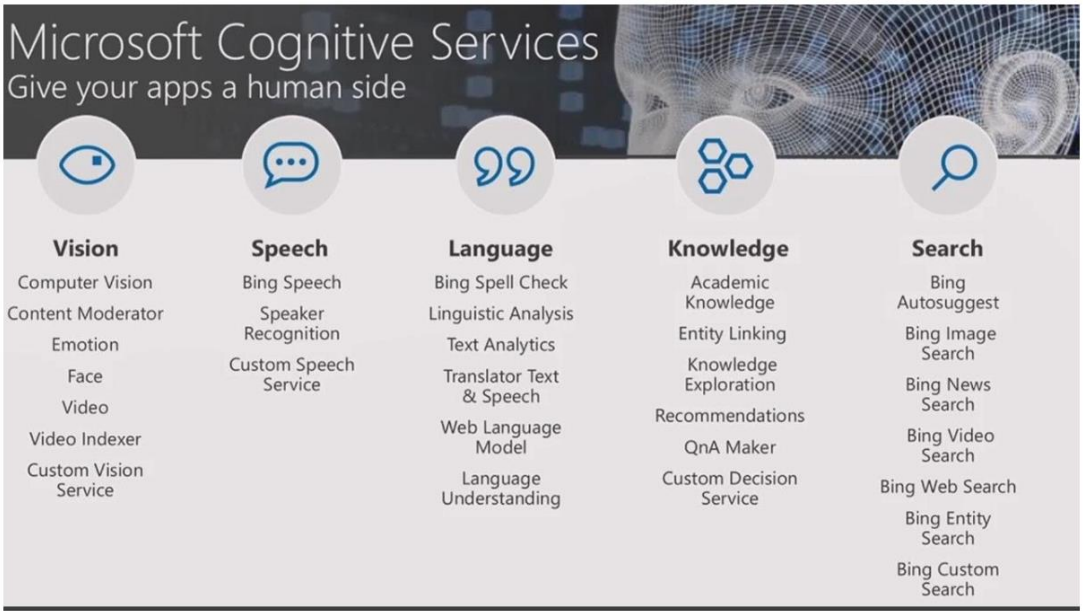


Figura 7-1 MS Azure Cognitive services. Fuente: www.enmilocalfunciona.io[61]

Cognitive Services permite a los desarrolladores comenzar rápidamente aprovechando los modelos de IA prediseñados. Para desarrollar una aplicación de IA que utilice uno o más de los Servicios Cognitivos, los desarrolladores aprovechan las API proporcionadas por cada uno de los Servicios Cognitivos. Esto permite a los desarrolladores desarrollar aplicaciones

[61] https://enmilocalfunciona.io/aplicaciones-inteligentes-con-microsoft-cognitive-services/

inteligentes usando varios lenguajes de programación (por ejemplo, C#, Java, JavaScript, PHP, Python, Ruby, etc.).
La siguiente figura muestra cómo interactúa una aplicación con Cognitive Services.
La aplicación emite una solicitud a una URL de Cognitive Services. Por ejemplo, una URL de solicitud para usar Cognitive Services para etiquetar una imagen (identificar cuáles son las etiquetas para los objetos que se encuentran en una imagen).

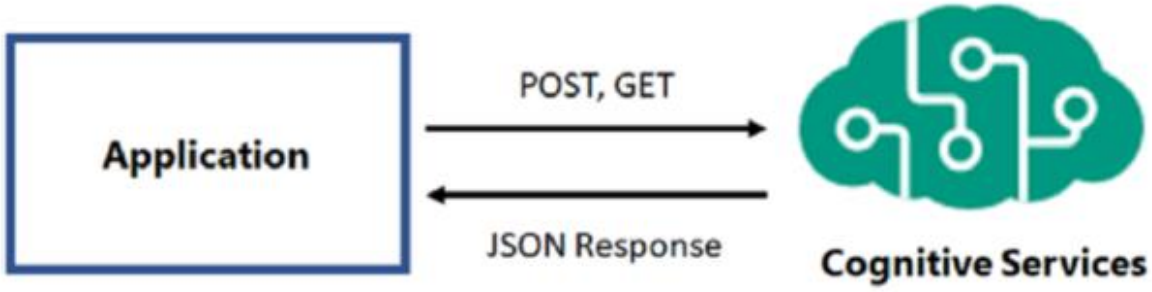


Figura 7-2 Esquema del protocolo entre nuestras aplicaciones y la API de Cognitive Services. Fuente: Azure

7.2 Analizar y clasificar imágenes con el servicio Computer Vision

Vamos a ver ahora como haciendo uso del Cognitive Service de Computer Vision podemos analizar una imagen o detectar objetos en la misma.

Creación de un recurso Cognitive Services
Partiendo de una suscripción de Azure vamos a crear un recurso de Cognitive Services, el procedimiento sería el siguiente:

1. Abrimos Azure Portal en https://portal.azure.com e iniciamos sesión con la cuenta de Microsoft.
2. Hacemos clic en el botón ＋Crear un recurso, buscamos Cognitive Services y creamos un recurso de Cognitive Services con la siguiente configuración:
   - Suscripción: Su suscripción de Azure.
   - Grupo de recursos: seleccione o cree un grupo de recursos con un nombre único.
   - Región: Elija cualquier región disponible:
   - Nombre: Ingrese un nombre único.
   - Nivel de precios: S0
   - Confirmo haber leído y entendido los avisos: Seleccionado.

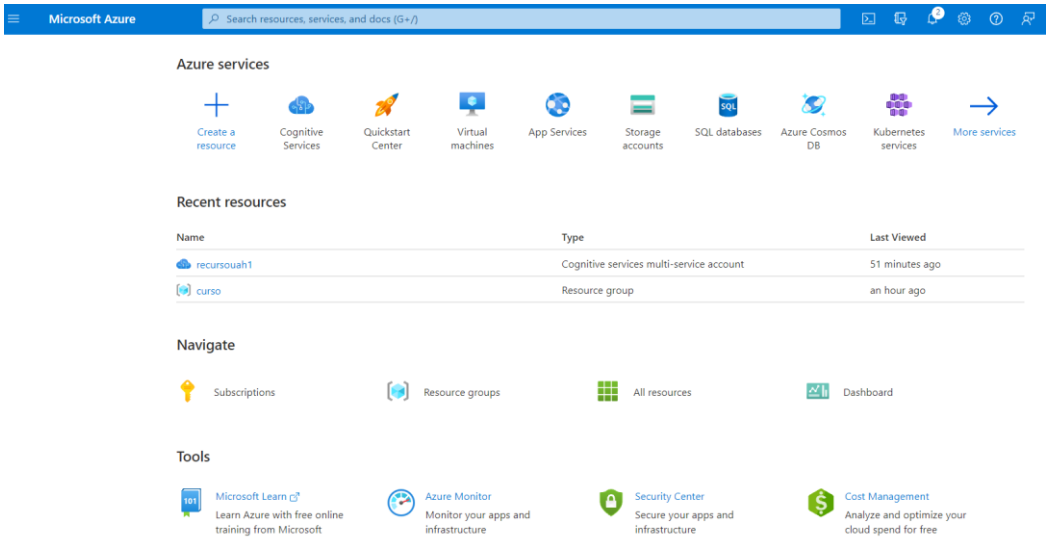


Figura 7-3 Captura Azure Cognitive Services

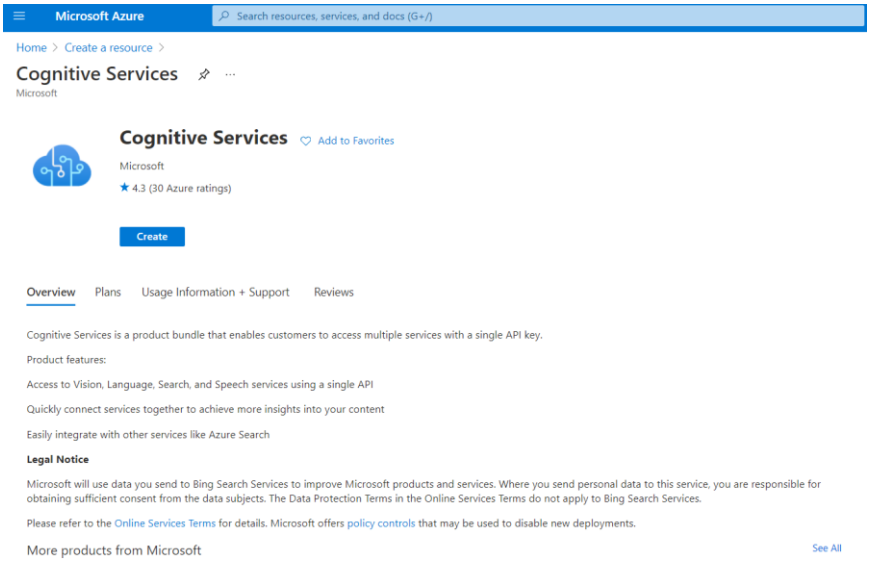


Figura 7-4 Captura Azure Cognitive Services

1. Esperamos a que se complete la implementación. Luego, vamos al recurso de servicios cognitivos y, en la página Descripción general, hacemos clic en el enlace para administrar las claves del servicio. Copiamos el “endpoint” y las claves para conectarse al recurso de servicios cognitivos desde las aplicaciones cliente.

Obtenemos la clave y el punto final para el recurso de Cognitive Services
Para usar el recurso de servicios cognitivos, las aplicaciones cliente necesitan su punto final y clave de autenticación, los copiamos.

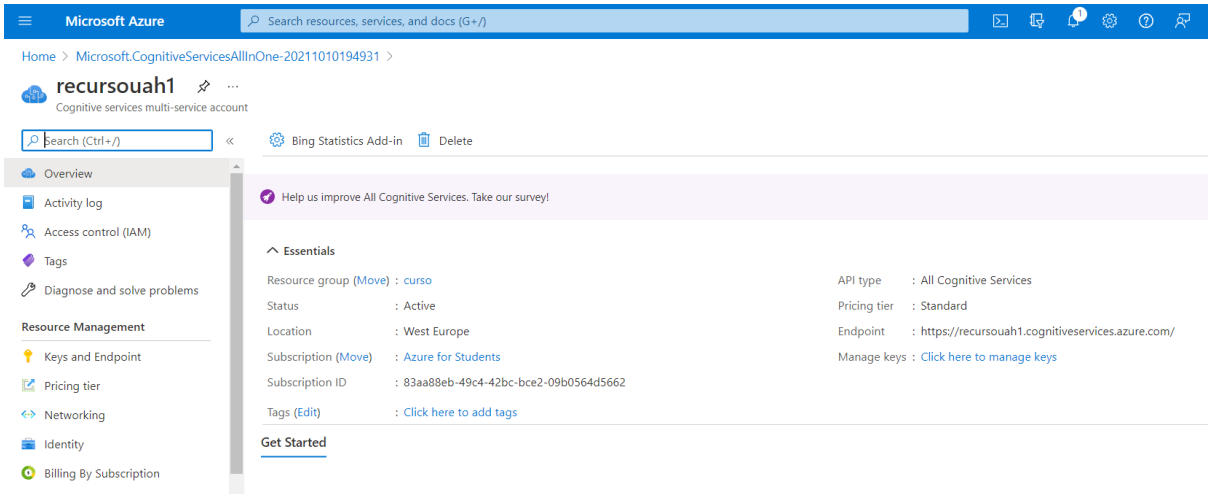


Figura 7-5 Captura Azure Cognitive Services

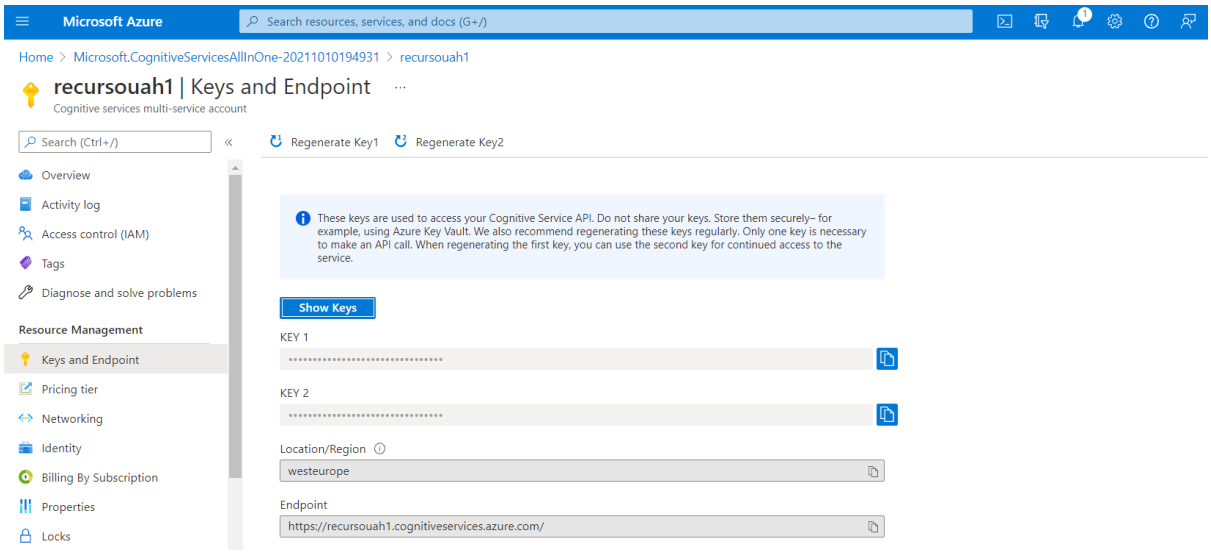
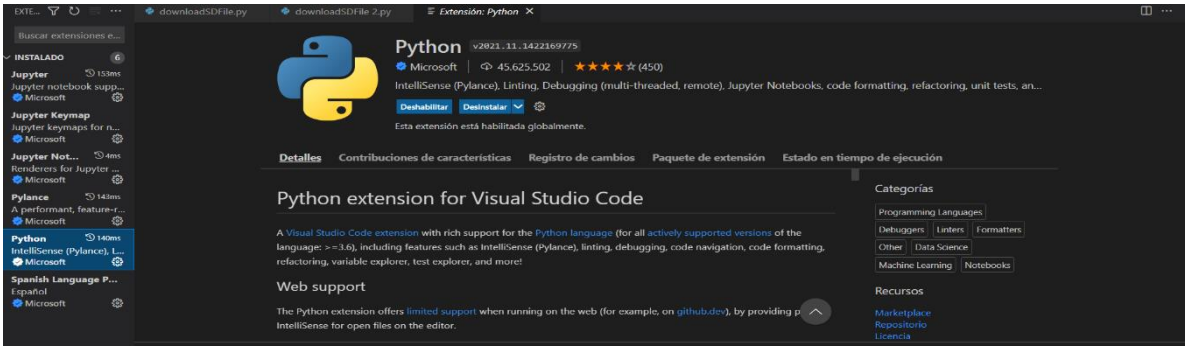


Figura 7-6 Captura Azure Cognitive Services

Ejemplo en Python

Para probar este servicio de Azure para analizar imágenes vamos a utilizar un pequeño ejemplo en Python que incluye Microsoft en su curso de formación en Azure AI Fundamentals.
Para poder ejecutar este código en nuestra máquina virtual con Visual Studio Code deberemos primeramente instalar los paquetes necesarios además de Python:

1. Download and install the Visual C++ Redistributable (x64)
2. Install Python 3.6.1 como mínimo



3. Después de la instalación, abrimos el símbolo del sistema e instalamos los paquetes necesarios:

pip install ipython jupyter matplotlib pillow requests azure-cognitiveservices-vision-computervision azure-cognitiveservices-vision-customvision azure-cognitiveservices-vision-face azure-cognitiveservices-language-textanalytics azure.cognitiveservices.speech azure_ai_formrecognizer

Primeramente, indicaremos en nuestro programa la clave y el código del endpoint que hemos generado previamente.

```
cog_key = 'YOUR_COG_KEY'
cog_endpoint = 'YOUR_COG_ENDPOINT'

print('Ready to use cognitive services at {} using key {}'.format(cog_endpoint, cog_key))
```

El siguiente fragmento de código permite obtener una descripción de una imagen ubicada en /data/vision/store_cam1.jpg file.

```
from azure.cognitiveservices.vision.computervision import ComputerVisionClient
from msrest.authentication import CognitiveServicesCredentials
from python_code import vision
import os
%matplotlib inline

#  Pasamos la ruta de la imagen a analizar
image_path = os.path.join('data', 'vision', 'store_cam1.jpg')

# Conectamos con el servicio como cliente
computervision_client = ComputerVisionClient(cog_endpoint, CognitiveServicesCredentials(cog_key))

# Obtenemos una descripción
image_stream = open(image_path, "rb")
description = computervision_client.describe_image_in_stream(image_stream)

# Mostramos la imagen y su descripción.
vision.show_image_caption(image_path, description)
```

Figura 7-7 Descripción de la imagen analizada. Fuente: Azure

Analizar características de imágenes

Anteriormente hemos utilizado el servicio Computer Vision para generar una leyenda descriptiva para una imagen, pero el servicio Computer Vision proporciona capacidades de análisis que pueden extraer información detallada como:

- Las ubicaciones de objetos comunes detectados en la imagen.
- Ubicación y edad aproximada de los rostros humanos en la imagen.
- Si la imagen contiene contenido "para adultos", "subido de tono" o "sangriento".
- Etiquetas relevantes que podrían asociarse con la imagen en una base de datos para que sea fácil de encontrar.

```
# Indicamos la ruta de la imagen
image_path = os.path.join('data', 'vision', 'store_cam1.jpg')

# Especificamos las caracteristicas que queremos analizar
features = ['Description', 'Tags', 'Adult', 'Objects', 'Faces']

# Obtenemos el analisis desde computer vision service
image_stream = open(image_path, "rb")
analysis = computervision_client.analyze_image_in_stream(image_stream, visual_features=features)

# Mostramos los resultados del analisis
vision.show_image_analysis(image_path, analysis)
```

Figura 7-8 Resultado del análisis de características de una imagen de prueba

7.3 Detección de objetos con Azure Custom Vision

Introducción a Custom Vision:
Custon Vision es un servicio de Azure Cognitive Service que nos va a permitir, de forma gráfica y guiada entrenar un algoritmo basado en Deep Learning proporcionado por Azure mediante un aplicativo web que nos va guiando en las diferentes fases del proceso de subir nuestro set de imágenes de entrenamiento y validación del modelo de identificación de imágenes.
Una vez entrenado el algoritmo este estará disponible vía API para podernos conectar con él a fin de enviarle una imagen y que nos devuelva la información relativa a la clasificación de esta o los objetos detectados en ellas.
Este algoritmo entrenado y validado por nosotros estará alojado por defecto en la nube de Azure aunque también es posible descargarlo como una imagen de Docker preparada para ejecutarse en un dispositivo hardware compatible y con la runtime de IoT Hub de Azure instalada.

La funcionalidad de Custom Vision se puede dividir en dos funciones. La clasificación de imágenes aplica una o más etiquetas a una imagen completa. La detección de objetos es similar, pero devuelve las coordenadas en la imagen donde se pueden encontrar las etiquetas aplicadas.

Límites y precios:
Existen dos niveles de claves para el servicio Custom Vision. Podemos registrarnos para obtener una suscripción F0 (gratuita) o S0 (estándar) a través de Azure Portal.

Factor	F0 (free)	S0 (standard)
Projects	2	100
Training images per project	5,000	100,000
Predictions / month	10,000	Unlimited
Tags / project	50	500
Iterations	20	20
Min labeled images per Tag, Classification (50+ recommended)	5	5
Min labeled images per Tag, Object Detection (50+ recommended)	15	15
How long prediction images stored	30 days	30 days
Prediction operations with storage (Transactions Per Second)	2	10
Prediction operations without storage (Transactions Per Second)	2	20
TrainProject (API calls Per Second)	2	10
Other API calls (Transactions Per Second)	10	10
Accepted image types	jpg, png, bmp, gif	jpg, png, bmp, gif
Min image height/width in pixels	256 (see note)	256 (see note)
Max image height/width in pixels	10,240	10,240
Max image size (training image upload)	6 MB	6 MB
Max image size (prediction)	4 MB	4 MB
Max number of regions per image (object detection)	300	300
Max number of tags per image (classification)	100	100

Figura 7-9 Tabla de límites de Custom Vision en función del modelo se suscripción. Fuente: Azure

Creación de un proyecto de Custom Vision:

Vamos a crear un proyecto de Custom Vision y entrenaremos un modelo de detección de objetos, en este caso cajas de medicamentos que será el que utilizaremos en el último capitulo para el proyecto final de prueba de nuestro aplicativo. Para ello, utilizamos el portal de Custom Vision

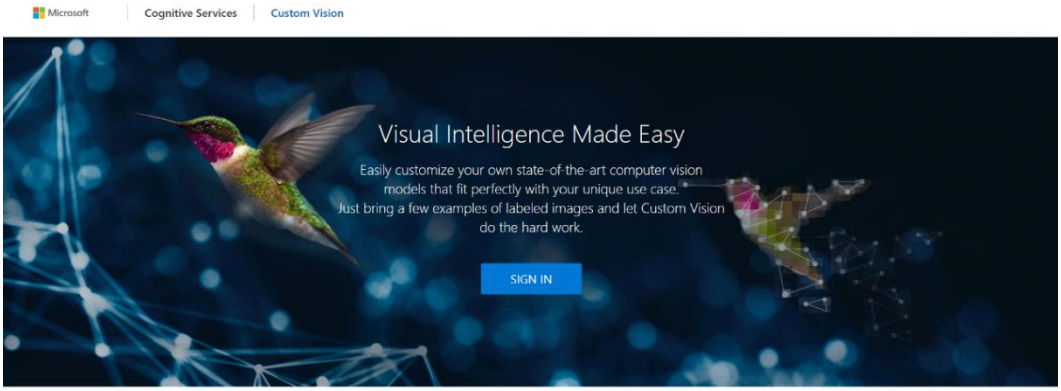


Figura 7-10 Captura Azure Custom Vision

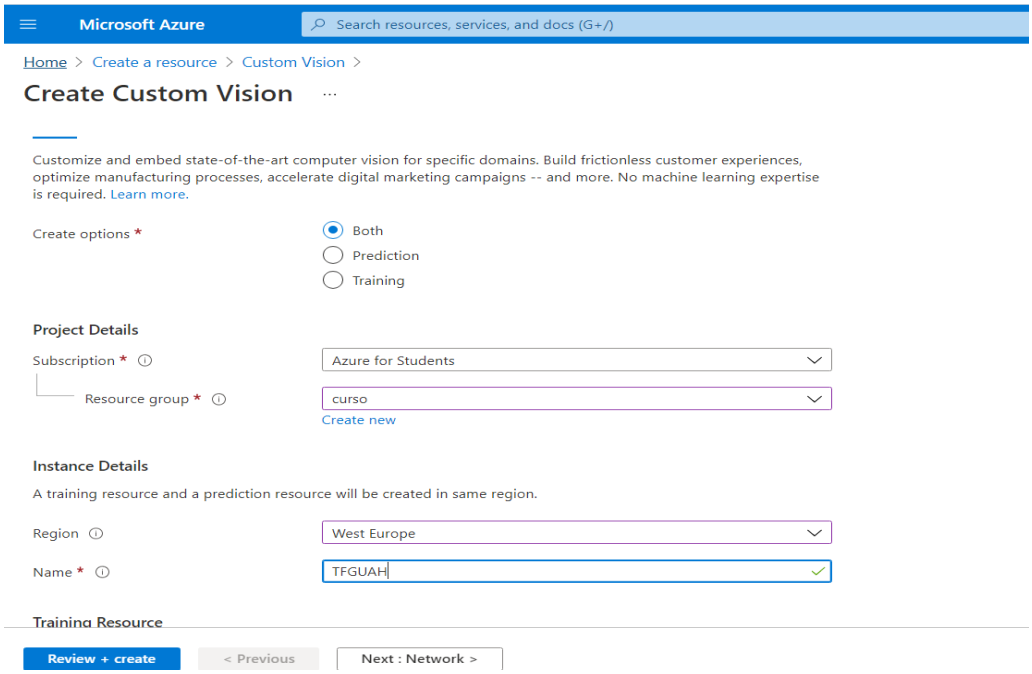


Figura 7-11Captura Azure Custom Vision

1. Para entrenar la red neuronal hemos realizado un set de fotos desde diferentes ángulos a las cajas de medicamentos que utilizaremos para el entrenamiento.

2. Abrimos el portal de Custom Vision en https://customvision.ai e iniciamos sesión con la cuenta de Microsoft asociada con la suscripción de Azure y aceptamos los términos del servicio.

3. En el portal de Custom Vision, creamos un nuevo proyecto con la siguiente configuración:

- Nombre: Medicinas
- Descripción: Cajas de medicinas
- Recurso: recursouah2 (el recurso de Custom Vision que creó anteriormente)
- Tipos de Proyectos: Object Detection
- Tipos de clasificación: Multiclase (etiqueta única por imagen)
- Dominio: General

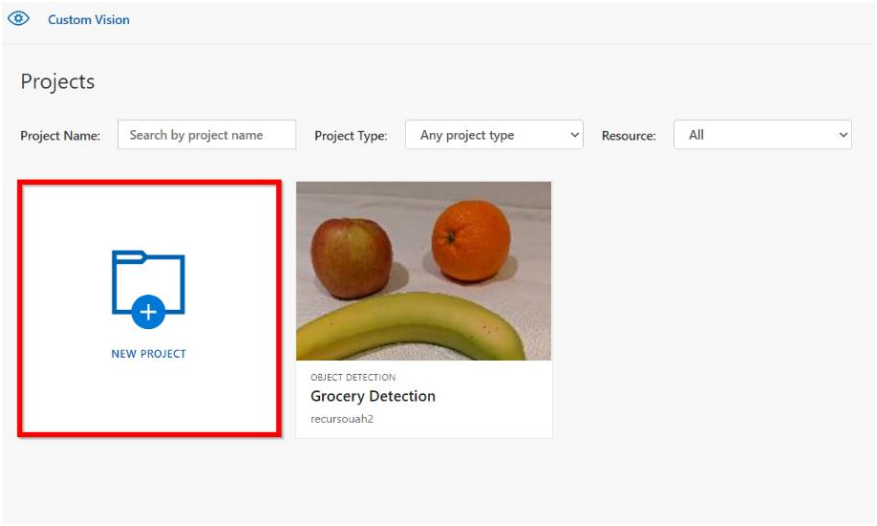


Figura 7-12 Captura Azure Custom Vision

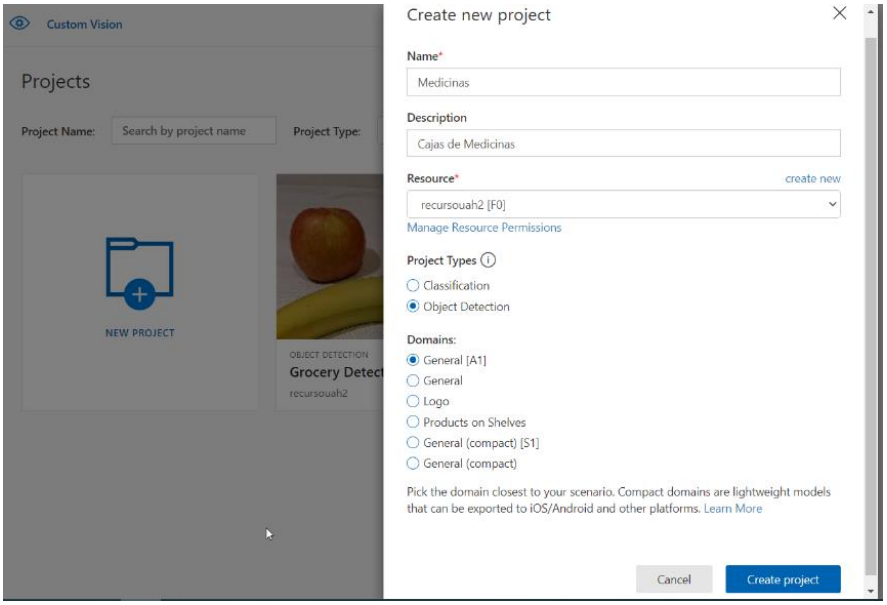


Figura 7-13 Captura Azure Custom Vision

4. Hacemos clic en [+] Agregar imágenes y seleccionamos todos los archivos en la carpeta donde tenemos nuestro juego de imágenes de las cajas de medicamentos. Luego cargamos los archivos de imagen, especificando la etiqueta correspondiente.

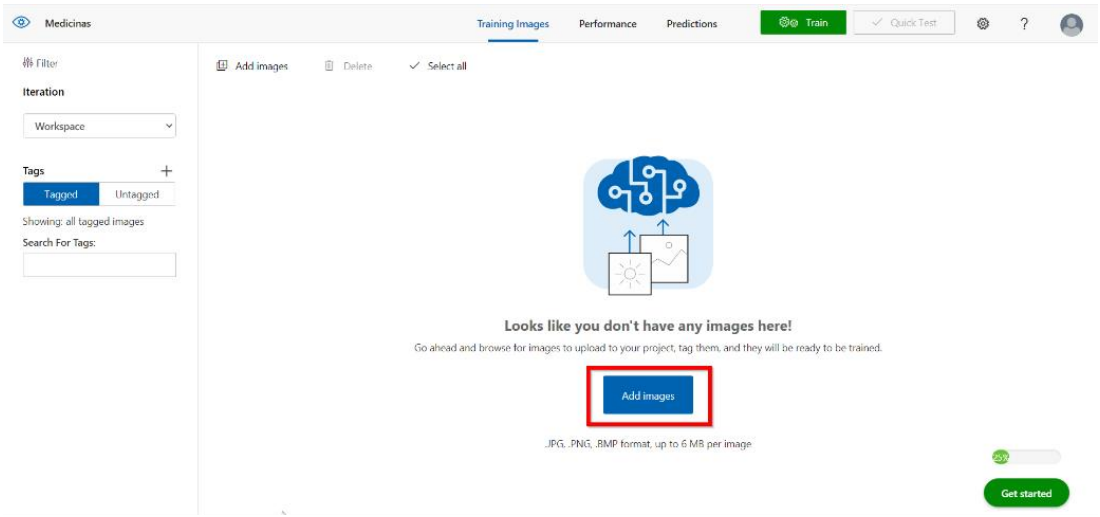


Figura 7-14 Captura Azure Custom Vision

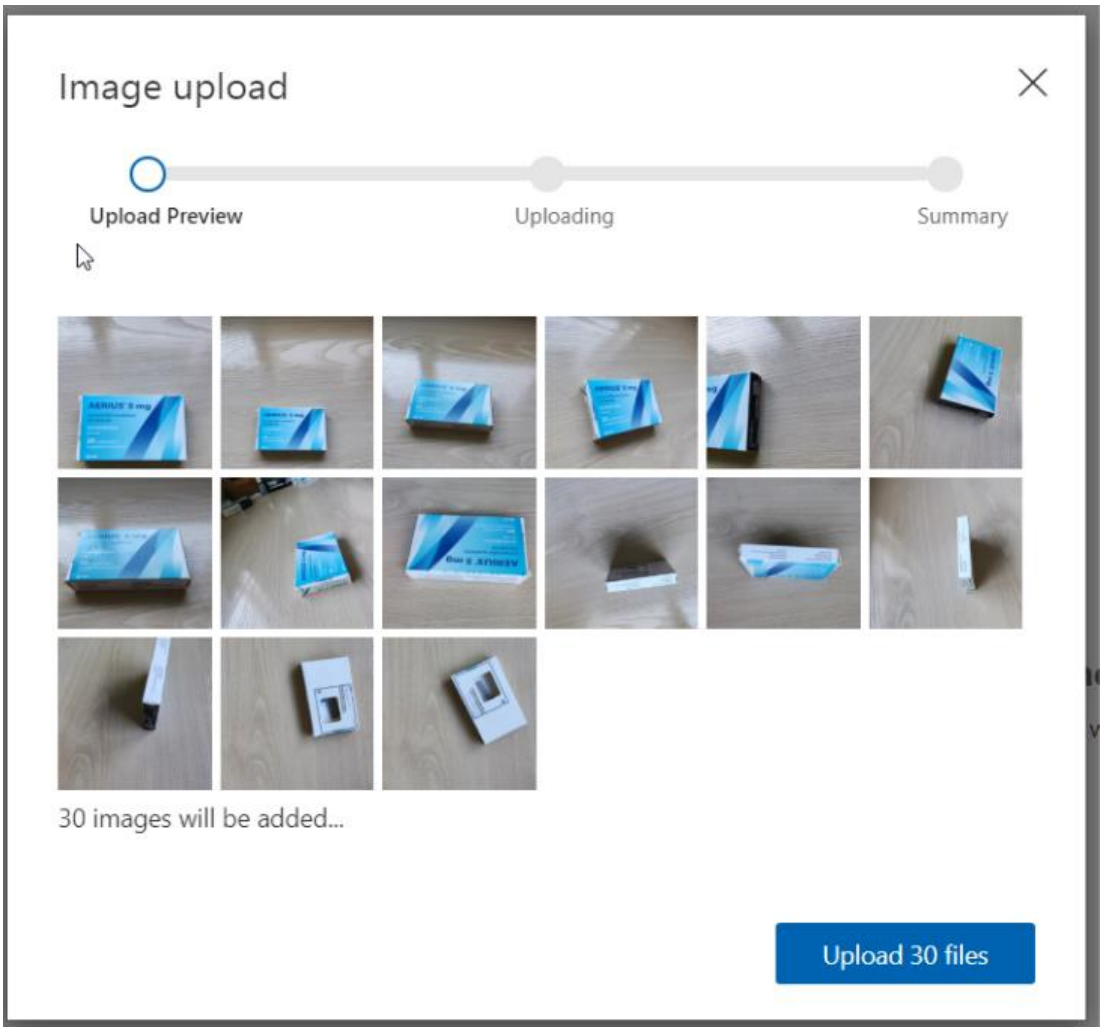


Figura 7-15 Captura Azure Custom Vision

5. Repetimos el paso anterior para cargar las imágenes en la carpeta Aerius con la etiqueta Aerius, las imágenes en la carpeta omeprazol con la etiqueta omeprazol y así sucesivamente

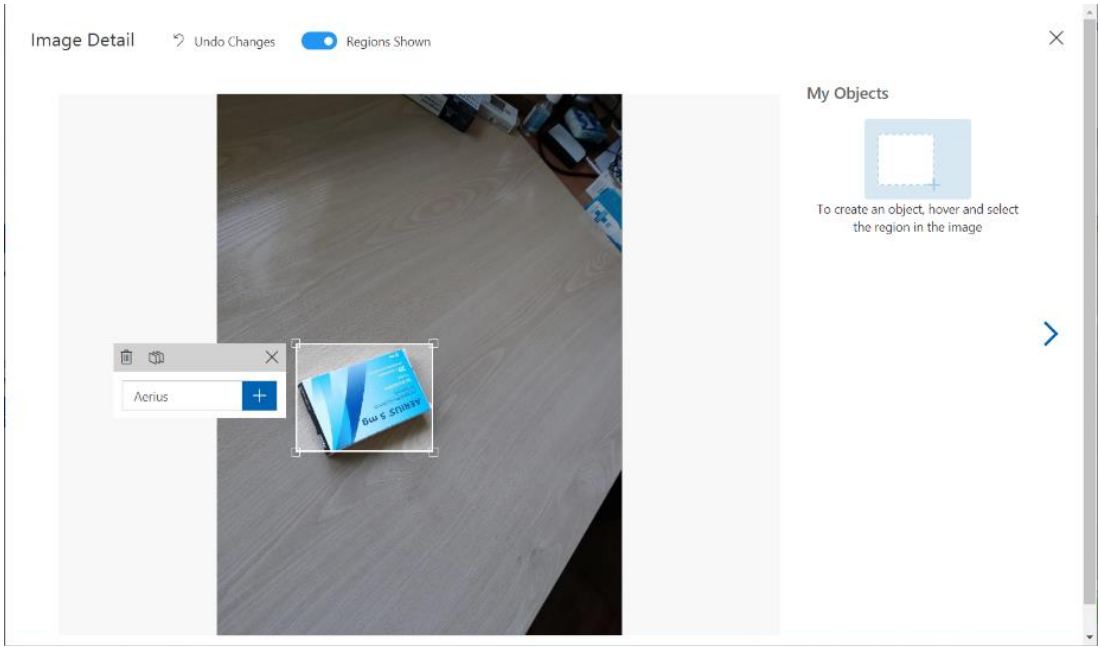


Figura 7-16 Captura Azure Custom Vision

6. Exploramos las imágenes que cargamos en el proyecto Custom Vision; en una primera instancia utilizamos un mínimo de 20 imágenes de cada clase, como esta:

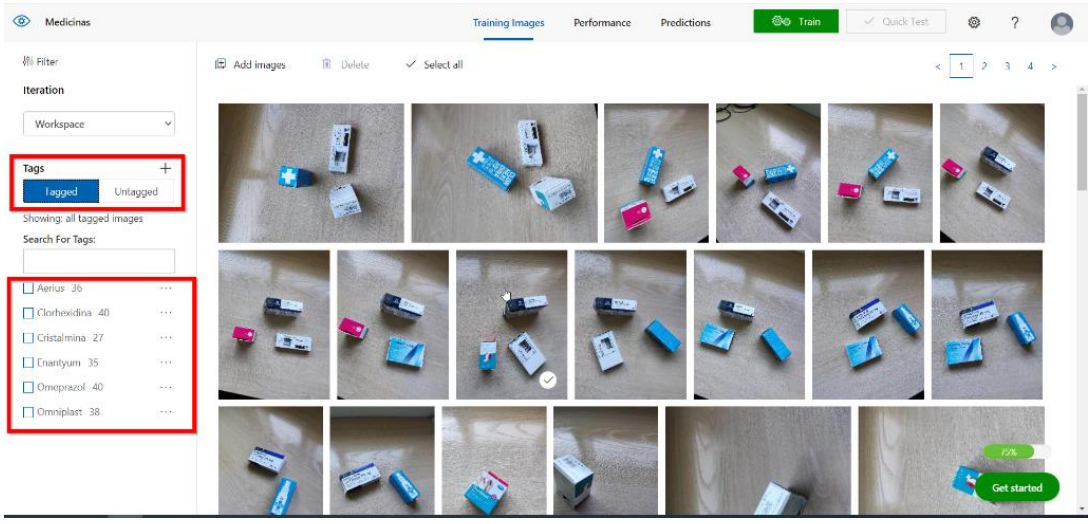


Figura 7-17 Captura Azure Custom Vision

7. En el proyecto de Custom Vision, arriba de las imágenes, hacemos clic en “Train” para entrenar un modelo de clasificación usando las imágenes etiquetadas. Seleccionamos la opción Entrenamiento rápido y luego esperamos a que se complete la iteración de entrenamiento,esto puede demorar unos minutos en función del número de imágenes, en nuestro caso 110 minutos.

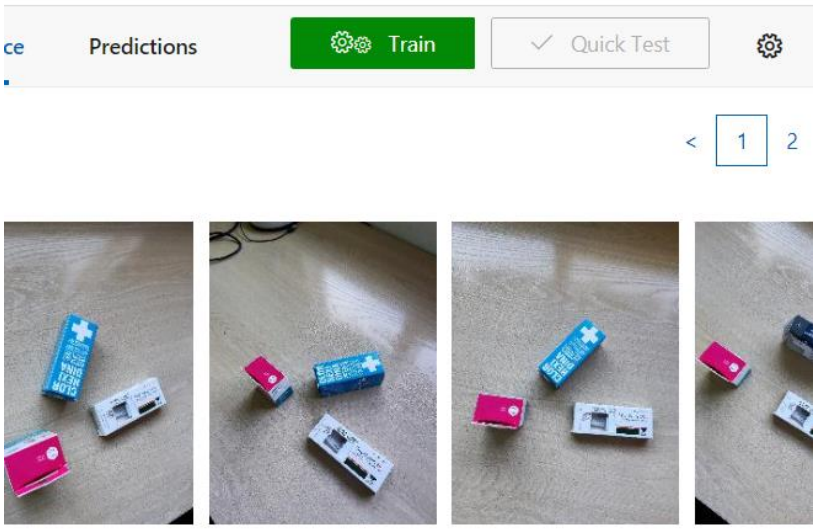


Figura 7-18 Captura Azure Custom Vision

Figura 7-19 Captura Azure Custom Vision

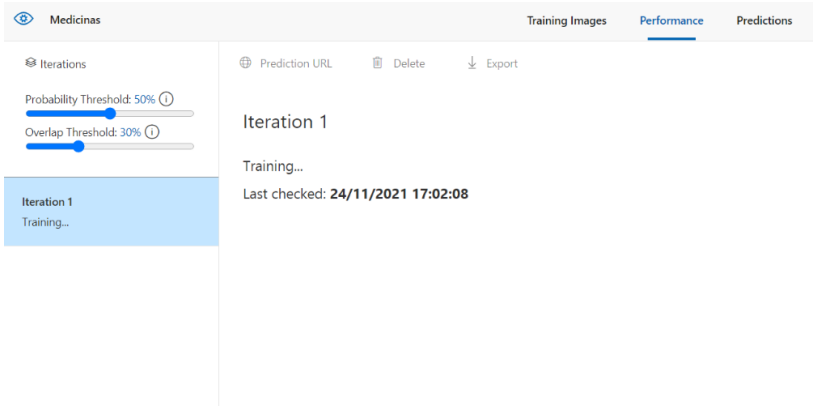


Figura 7-20 Captura Azure Custom Vision

8. Cuando se ha terminado el entrenado de la iteración del modelo, revisamos las métricas de rendimiento de precisión, recuperación y AP; estas miden la precisión de predicción del modelo de clasificación, todos en una primera interacción con un numero de imágenes por objeto entre

20 y 40 no obtenemos la precisión deseada, estando está por debajo del 80% global y en algunos casos como en con las cajas de omeprazol por debajo del 50%

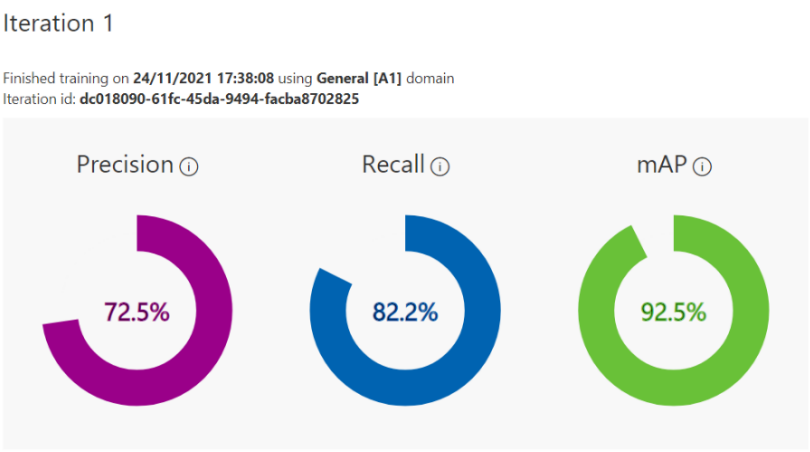


Figura 7-21 Captura Azure Custom Vision: Rendimiento predicción

Performance Per Tag

Tag	Precision	Recall	A.P.	Image count
Omniplast	100.0%	37.5%	69.8%	38
Enantyum	100.0%	100.0%	100.0%	35
Clorhexidina	88.9%	88.9%	98.9%	40
Cristalmina	83.3%	83.3%	94.8%	27
Aerius	66.7%	85.7%	91.6%	36
Omeprazol	47.1%	100.0%	100.0%	40

Figura 7-22 Captura Azure Custom Vision: métricas

9. Realizamos un segundo entrenamiento, en este caso aumentando las imágenes por objeto a un mínimo de 50. Se puede observar que las métricas mejoran ostensiblemente con un 80% de precisión en el peor caso (omeprazol).

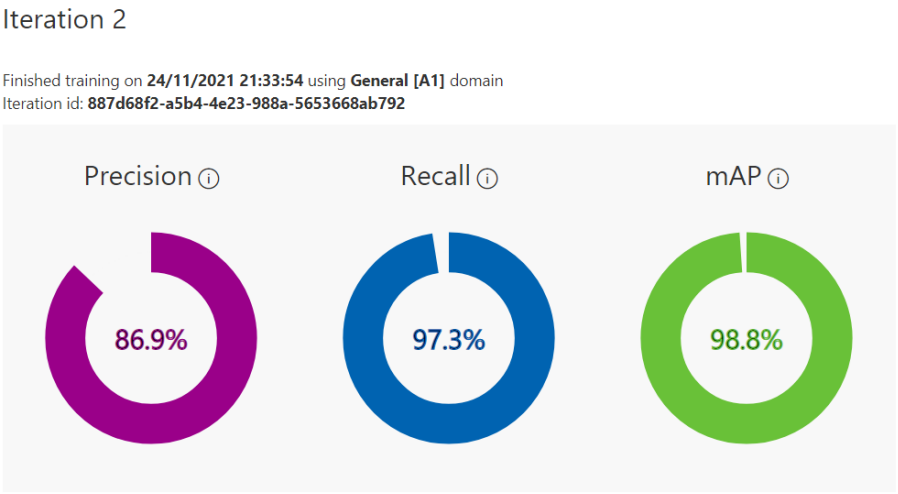


Figura 7-23 Captura Azure Custom Vision. Rendimiento segunda iteración

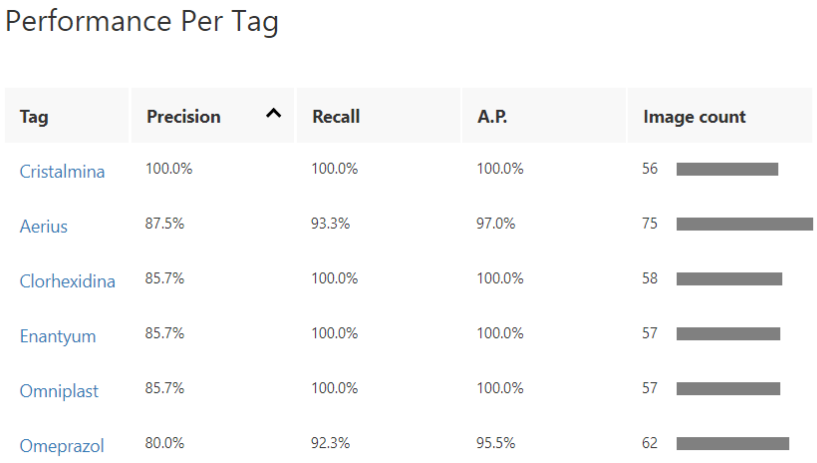


Figura 7-24 Captura Azure Custom Vision; Métricas

10. Probamos el modelo con imágenes diferentes a las utilizadas para el entrenamiento con la utilizad que incorpora Azure Custon Vision (Quick Test).

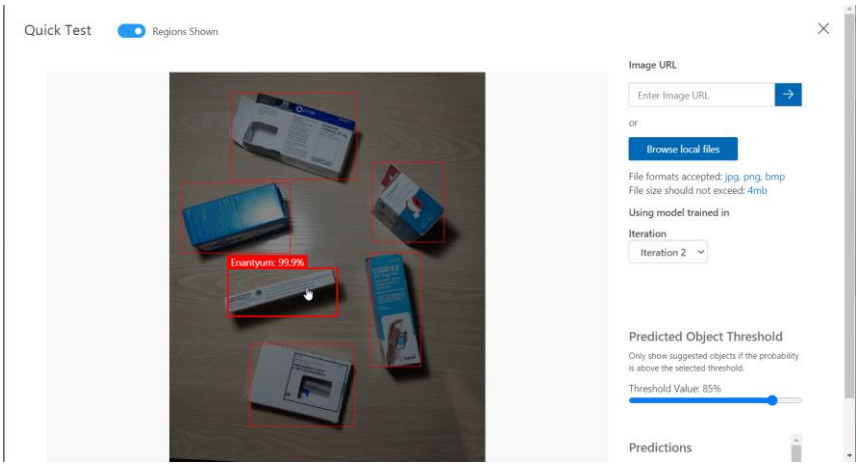


Figura 7-25 Captura Azure Custom Vision

Publicar el modelo

9. Hacemos clic en publicar para publicar el modelo entrenado con la siguiente configuración:

- Nombre del modelo: Medicinas
- Recurso de predicción: el recurso de predicción que creamos anteriormente.

10. Después de publicar, hacemos clic en el icono de configuración (⚙) en la parte superior derecha de la página para ver la configuración del proyecto y copiar el ID del proyecto. Esta ID la utilizaremos en nuestro código para habilitar la conexión.

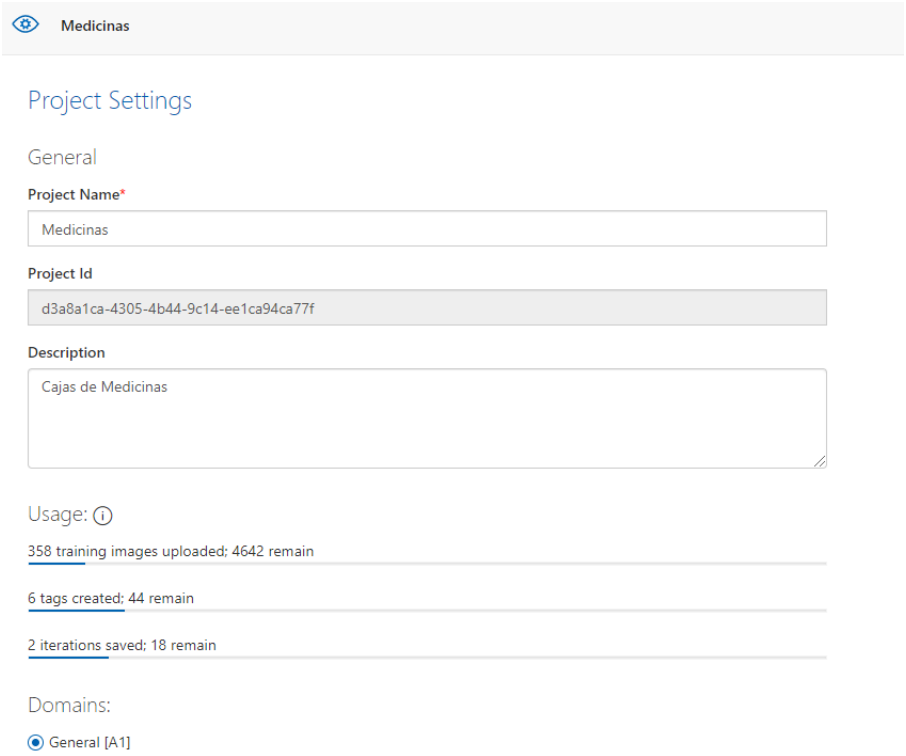


Figura 7-26 Captura Azure Custom Vision

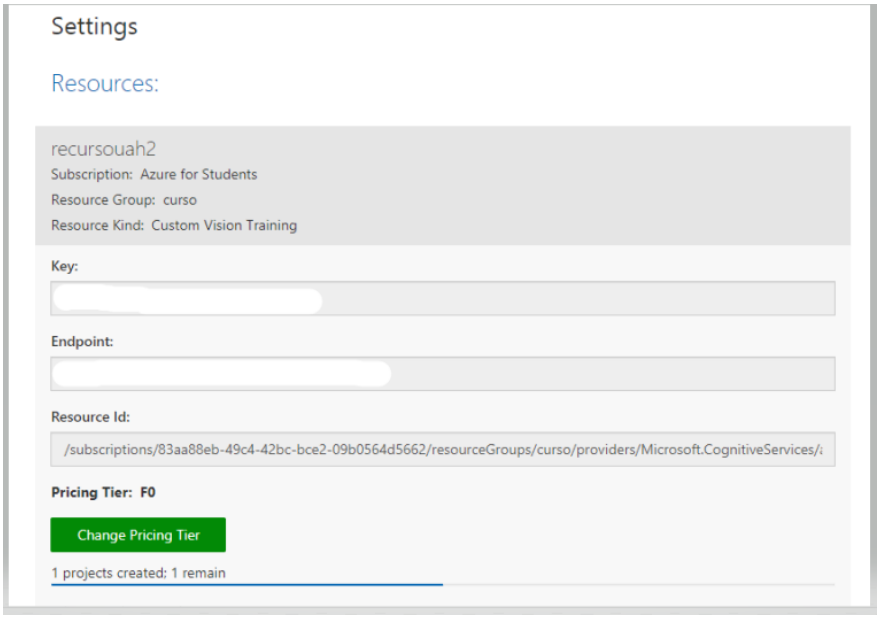


Figura 7-27 Captura Azure Custom Vision

11. En la parte superior izquierda de la página en "Configuración del proyecto", hacemos clic en el icono Galería de proyectos ( ) para volver a la página de inicio del portal de Custom Vision, donde ahora aparece el proyecto.

12. En la página de inicio del portal Custom Vision, en la parte superior derecha, hacemos clic en el icono de configuración ( ) para ver la configuración del servicio Custom Vision. Después, en recursos, expandimos el recurso de predicción (no el recurso de entrenamiento) y copiamos sus valores KEY y ENDPOINT

Estos dos valores junto con el PROJECT_ID los necesitaremos para que nuestro código cliente se pueda conectar a la API del modelo de predicción que hemos entrenado y está alojado en la nube de Azure.

Ejemplo de código:

```
project_id = '71993c26-ff8d-4c62-9a4f-f5d35d81abae'
cv_key = 'dd64681e082c4f5687d5f93e016bf756'
cv_endpoint = 'https://recursotrainuah2.cognitiveservices.azure.com/'

model_name = 'groceries' # this must match the model name you set when publish
ing your model iteration (it's case-sensitive)!
print('Ready to predict using model {} in project {}'.format(model_name, proje
ct_id))


from azure.cognitiveservices.vision.customvision.prediction import CustomVisio
nPredictionClient
from msrest.authentication import ApiKeyCredentials
import matplotlib.pyplot as plt
from PIL import Image
import os
%matplotlib inline
# Get the test images from the data/vision/test folder
test_folder = os.path.join('data', 'image-classification', 'test-fruit')
test_images = os.listdir(test_folder)
# Create an instance of the prediction service
credentials = ApiKeyCredentials(in_headers={"Prediction-key": cv_key})
custom_vision_client = CustomVisionPredictionClient(endpoint=cv_endpoint, cred
entials=credentials)

# Create a figure to display the results
fig = plt.figure(figsize=(16, 8))

# Get the images and show the predicted classes for each one
print('Classifying images in {} ...'.format(test_folder))
for i in range(len(test_images)):
        # Open the image, and use the custom vision model to classify it
            image_contents = open(os.path.join(test_folder,
       test_images[i]), "rb")
            classification = custom_vision_client.classify_image(project_id, mo
del_name, image_contents.read())
            # The results include a prediction for each tag, in descending orde
r of probability - get the first one
            prediction = classification.predictions[0].tag_name
            # Display the image with its predicted class
            img = Image.open(os.path.join(test_folder, test_images[i]))
            a=fig.add_subplot(len(test_images)/3, 3,i+1)
            a.axis('off')
```

```
        imgplot = plt.imshow(img)
        a.set_title(prediction)
    plt.show()
```

8. Desarrollo de una App para visión artificial.

8.1 Backend de la aplicación

El backend de la aplicación se va a encargar por un lado de conectar con la cámara para capturar las imágenes y enviarlas al modelo de detección de objetos entrenado en Azure Custom Vision y por otro de recibir la predicción desde Azure y transmitirla a las correspondientes variables del PLC para que esté pueda gobernar el robot en base a la información recibida. Vamos a desgranar a continuación esas funcionalidades.

Conexión API de Azure Custom Vision:
Para esta parte hemos creado un programa de Python (azure.py) que es invocado desde nuestro programa principal.
Simplemente vamos a indicar la información necesaria para que nuestra aplicación se pueda conectar con el modelo entrenado de Azure Custom Vision como vimos en el apartado 7.3 de este TFG.

app > azure.py > ...

```python
from flask import *
import os
from azure.cognitiveservices.vision.customvision.prediction import CustomVisionPredictionClient
from msrest.authentication import ApiKeyCredentials
from app import app



# Indicamos la ID de nuestro proyecto de Azure Custom Vision
project_id = 'd3a8a1ca-4305-4b44-9c14-ee1ca94ca77f'
# Indicamos la clave de la interacción del modelo a la que nos queremos conectar
cv_key = '12d4b9303c4b4718b9c5bd2e3d9ea269'
# Indicamos el endpoint de la conexión
cv_endpoint = 'https://recursouah2-prediction.cognitiveservices.azure.com/'
# El model_name debe coincidir exactamente con el nombre de nuestro modelo de Azure
model_name = 'Iteration2'
print('Listo para detección con el modelo {} en proyecto {}'.format(model_name, project_id))
```

Figura 8-1 Credenciales de conexión Azure

Captura del frame de la imagen:
En la aplicación real, cuando la fotocélula detectara una caja de medicamentos pararía la cinta transportadora y mandaría la señal para capturar un frame del video en streaming que esta emitiendo la cámara para su envío a la nube.
En nuestra aplicación hemos simulado este comportamiento con el botón “Detectar”.
Cuando pulsamos dicho botón emitimos una petición POST desde la interfaz web que desencadena, por un lado la captura de un frame del video en streaming y por otro su envío a la nube para obtener la correspondiente detección y clasificación de los objetos contenidos en dicho frame de video.

```
if request.method == 'POST':
    success, image = cv2.VideoCapture('rtsp://Siemens:Siemens!00@192.168.100.152:554/stream2').read()
    cv2.imwrite("app/static/tapo.jpg", image)
```

Figura 8-2 Captura frame de video

Indicar que como se ve en la captura antes del envío del frame a la nube debemos convertirlo a un formato permitido por Azure Custom Vision, en nuestro caso lo convertimos a .jpg mediante las funcionalidades que nos aporta la librería OPENCV

Detección de objetos:
Una vez tenemos establecido la conexión con la nube de Azure y tenemos la imagen en el directorio adecuado y en formato .jpg podemos obtener una predicción de esta.

Cargamos la imagen a enviar

```
# Cargamos nuestra imagen capturada y obtenemos sus dimensiones
test_img_file = os.path.join(filename)
test_img = Image.open(test_img_file)
test_img_h, test_img_w, test_img_ch = np.array(test_img).shape
```

Figura 8-3 Carga de la imagen

Nos conectamos a Azure mediante las credenciales indicadas anteriormente.

```
# Obtenemos un cliente de predicción para el modelo de detección de objetos
credentials = ApiKeyCredentials(in_headers={"Prediction-key": cv_key})
predictor = CustomVisionPredictionClient(endpoint=cv_endpoint, credentials=credentials)

print('Detectando objetos en {} usando modelo {} en proyecto {}...'.format(test_img_file, model_name, project_id))
```

Figura 8-4 Conexión a Azure Custom Vision

Obtenemos el resultado de la predicción

```
# Detectar objetos en nuestra imagen
with open(test_img_file, mode="rb") as test_data:
    results = predictor.detect_image(project_id, model_name, test_data)
```

Figura 8-5 Obtenemos los resultados de la predicción

Comunicación con el HMI/Controlador vía OpenPipe.

Para esta parte vamos primeramente a crear un diccionario en Python donde mediante un bucle iremos escribiendo el resultado de la predicción de la imagen para cada objeto detectado.
Lo siguiente será conectar con la runtime del HMI vía OpenPipe para lo que indicaremos el canal del pipe e iremos escribiendo en las variables correspondientes de nuestro proyecto de HMI.

```
objetos = {
    "tag_id": "db6c7a97-213e-4b18-bd62-1cfcf0b59e77",
    "tag_name": "Aerius",
    "tag_probability": 0.893478,
    "bounding_box_left": 0.21961562,
    "bounding_box_top": 0.21961562,
    "bounding_box_height": 0.21961562,
    "bounding_box_width": 0.21961562
    }
```

Figura 8-6 Diccionario para guardar el resultado de la predicción

```
################### OpenPipe ##########################
objetos['tag_id'] = prediction.tag_id
objetos['tag_name'] = prediction.tag_name
objetos['tag_probability'] = prediction.probability
objetos['bounding_box_left'] = prediction.bounding_box.left
objetos['bounding_box_top'] = prediction.bounding_box.top
objetos['bounding_box_height'] = prediction.bounding_box.height
objetos['bounding_box_width'] = prediction.bounding_box.width
```

Figura 8-7 Asignación del resultado de la predicción al diccionario

```
pipeName = r'\\.\pipe\HmiRuntime'
var = {"var1":0,"var2":0,"var3":0}
pipe = getPipe()

objn = str(count)
```

Figura 8-8 Conectamos el Pipe del HMI

```
try:

    pipeInfo = win32pipe.GetNamedPipeInfo(pipe)
    inbuffersize = pipeInfo[2]

###################### WRITE TAG ##########################################
###################### WRITE TAG ##########################################

    # Creamos diccionario
    dict_subscribe_tags = {"Message":"WriteTag","Params":{"Tags":[{"Name":"obj" + objn + ".tag_id","Value":prediction.tag_id},
    {"Name":"obj" + objn + ".tag_name","Value":prediction.tag_name}, {"Name":"obj" + objn + ".tag_probability","Value":prediction.probability},
    {"Name":"obj" + objn + ".bounding_box_left","Value":prediction.bounding_box.left}, {"Name":"obj" + objn + ".bounding_box_top","Value":predic
    {"Name":"obj" + objn + ".bounding_box_height","Value":prediction.bounding_box.height}, {"Name":"obj" + objn + ".bounding_box_width","Value":
    "ClientCookie":"mySubscription1"}


    # necesitamos serializar el objeto json (diccionario de python) en una cadena para enviarlo a pipe abierto
    subscribe_tags_string = json.dumps(dict_subscribe_tags) + '\n'


    win32file.WriteFile(pipe, subscribe_tags_string.encode('utf-8'))

    resp = win32file.ReadFile(pipe, inbuffersize)
    print("Response in the pipe: ", resp)

    win32file.CloseHandle(pipe)
```

Figura 8-9 Escritura del disccionario en las variables del HMI vía OpenPipe

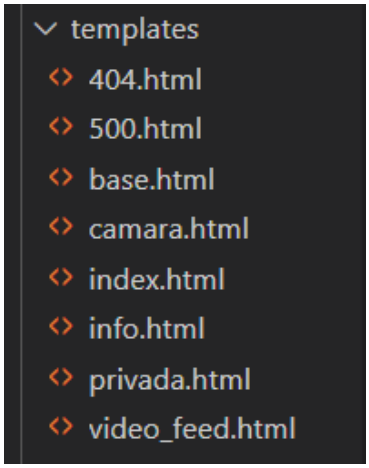
8.2 Frontend de la aplicación

Aquí es donde nos vamos a encargar de la interfaz de usuario web que nos va a permitir comprobar el funcionamiento de nuestro sistema de cámara y predicción, así como su activación y desactivación por parte del usuario.
Para esta aprte hemos utilizado el framwork de Flask ya que nos permitica combinar la parte de código del backend escrito en Python con la parte de interfaz web escrito en HTML.
Como en los capítulos anteriores ya hemos comentado en detalle en que consiste aquí simplemente vamos a repasar como es la estructura de nuestra aplicación en flask.

Programa principal:
Está va a ser la aplicación propiamente dicha. En ella vamos a controlar que página html se carga en función de la navegación del usuario y que código se está ejecutando por detrás en la misma.

```
@app.route('/')
def home():
    return render_template('index.html')

def gen(camera):
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')


@app.route('/video_feed')
def video_feed():
    return Response(gen(VideoCamera()), mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/info')
def info():
    return render_template('info.html')

@app.route('/privada')
def privada():
    return render_template('privada.html')

@app.route('/camara', methods=["GET", "POST"])

def camara(): ···
```

Nuestra web tendrá la siguiente estructura:



http://127.0.0.1:5000 – página de inicio
http://127.0.0.1:5000/camara - aplicación propiamente dicha
http://127.0.0.1:5000/info - Info de nuestra aplicación
http://127.0.0.1:5000/video_feed - streaming de video
http://127.0.0.1:5000/privada - para test
http://127.0.0.1:5000/404 - info error 404
http://127.0.0.1:5000/505 - info error 500

En la página de inicio haremos clic en el incono central (Figura 8-10) y navegaremos a la página principal de la aplicación

Figura 8-10 Página de inicio aplicación

Una vez en la página principal nos encontraremos lo siguiente:

1. Emisión de video en vivo de nuestra cámara.
2. Última predicción realizada sobre imagen capturada.
3. Volcado de la predicción en un string de json.
4. Botón para realizar captura de imagen y predicción sobre la misma.

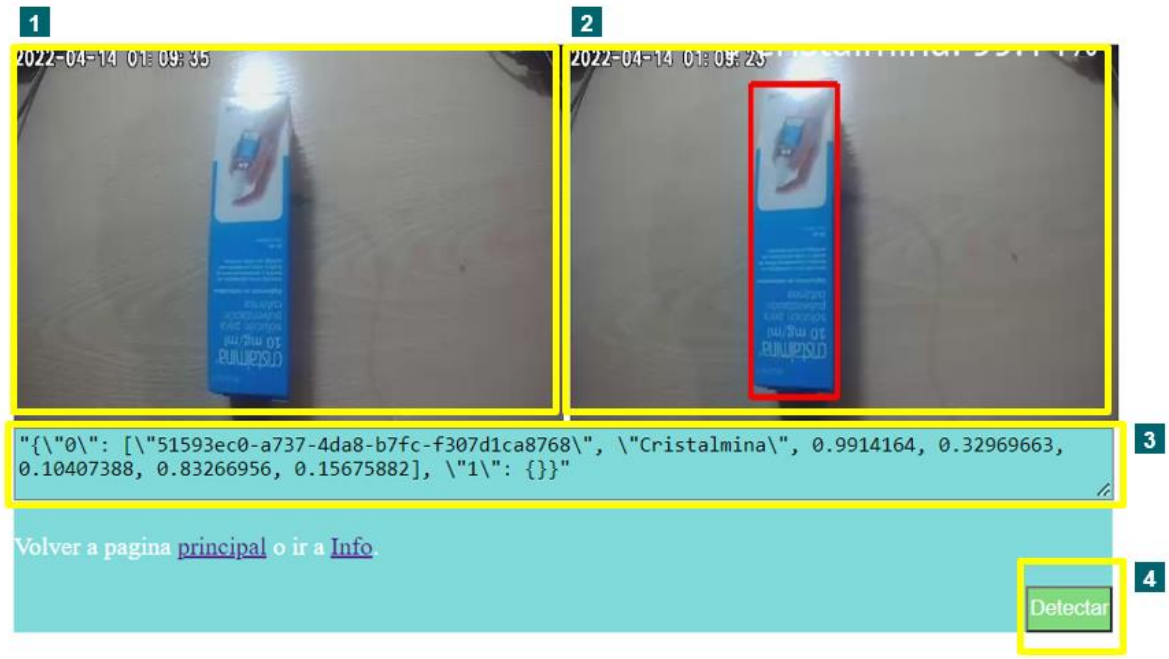


Figura 8-11 Página principal aplicaión

Otras páginas serán la de información de la aplicación o la de error como podemos ver en las figuras 8-12 y 8-13

Figura 8-12 Página informativa

Figura 8-13 Mensaje de error

8.3 Dockerización de la aplicación

Una vez tenemos la aplicación terminada y probada vamos a empaquetarla en un contenedor de Docker para lo cual necesitaremos crear un par de nuevos ficheros:

Requirementes.txt:
Este archivo nos sirve para indicar el listado de librerías que deberemos instalar en nuestra imagen con su correspondiente versión: OpenCV, Flask, Numpy…etc

requirements.txt

```
azure-common==1.1.25
camera==1.3.0
certifi==2020.6.20
chardet==3.0.4
click==7.1.2
Flask==2.0.2
future==0.18.2
idna==2.10
isodate==0.6.0
itsdangerous==2.0
Jinja2==3.0
MarkupSafe==2.0.0rc2
msrest==0.6.17
numpy==1.19.0
oauthlib==3.1.0
pyngrok==4.1.3
PyYAML==5.3.1
requests==2.24.0
requests-oauthlib==1.3.0
six==1.15.0
urllib3==1.25.9
opencv-python==4.2.0.34
gunicorn==20.0.4
```

Figura 8-14 Listado de librerías a instalar

Dockerfile:
Ya vimos en el capítulo 1.2 (introductorio) y 6.2 el significado que para Docker tiene el archivo Dockerfile, básicamente se trata de la descripción de que imagen de Docker vamos a utilizar, que vamos a instalar dentro (Python, OpenCV…), los volúmenes internos que va a contener, los puertos que se van a exponer y como se va a ejecutar la aplicación que cargaremos dentro.
En esta ocasión nos ha sido imposible utilizar la imagen de Docker Alpine, que hubiera sido lo óptimo por su reducido tamaño. Estó ha sido debido a la imposibilidad de instalar OpenCV en dicha imagen. (Ver figura 8-15)
En su lugar hemos utilizado **python:3.7-slim,** la cual nos permite instalar todas las librerias necesarias sin ningún problema, pero a cambio de un tamaño mayor (325MB) como podemos ver en la figura 8-16

```
ERROR: Could not find a version that satisfies the requirement opencv-python==4.2.0.34 (from versions: 3.4.0.14, 3.4.10.37, 3.4.11.39, 3.4.11.41, 3.4.11.43,
 3.4.11.45, 3.4.13.47, 3.4.15.55, 3.4.16.57, 3.4.16.59, 3.4.17.61, 3.4.17.63, 4.3.0.38, 4.4.0.40, 4.4.0.42, 4.4.0.44, 4.4.0.46, 4.5.1.48, 4.5.3.56, 4.5.4.58
, 4.5.4.60, 4.5.5.62, 4.5.5.64)
ERROR: No matching distribution found for opencv-python==4.2.0.34
The command '/bin/sh -c pip3 install -r /requirements.txt' returned a non-zero code: 1
diego@diegoVM:~/Unified_App$ ^C
```

Figura 8-15 Error al instalar opencv con Alpine

```
diego@diegoVM:~$ sudo docker images
[sudo] contraseña para diego:
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
visionedge          latest              fddc6767f88a        40 seconds ago      325MB
```

Figura 8-16 Tamaño de la imagen de Docker de nuestra aplicación

Finalmente, el archivo dockerfile necesario para la correcta implementación de nuestra aplicación se puede ver en la figura 8-17

```
Dockerfile > ...
1   FROM python:3.7-slim
2
3
4   RUN pip install --upgrade pip
5
6
7   COPY requirements.txt /
8   RUN pip3 install -r /requirements.txt
9
10  COPY . /UNIFIED_APP
11  WORKDIR /UNIFIED_APP
12
13  EXPOSE 8080
14
15  #ENTRYPOINT ["./gunicorn.sh"]
16
17
18  CMD ["gunicorn", "-b", "0.0.0.0:8080", "--workers", "2", "app.aplicacion:app"]
19
20
```

Figura 8-17 Docker file aplicación visión

Como siempre para construir la imagen e Docker hemos utilizado el comando built con la siguiente sintaxis:

```
diego@diegoVM:~/Unified_App$ sudo docker build -t visionedge .
```

8.4 Publicación con IEAP

Una vez tenemos creada la imagen de Docker podemos publicarla con IEAP para la generación del fichero .app que se podrá instalar en el dispositivo Edge.

Lo primero será crear una nueva versión.

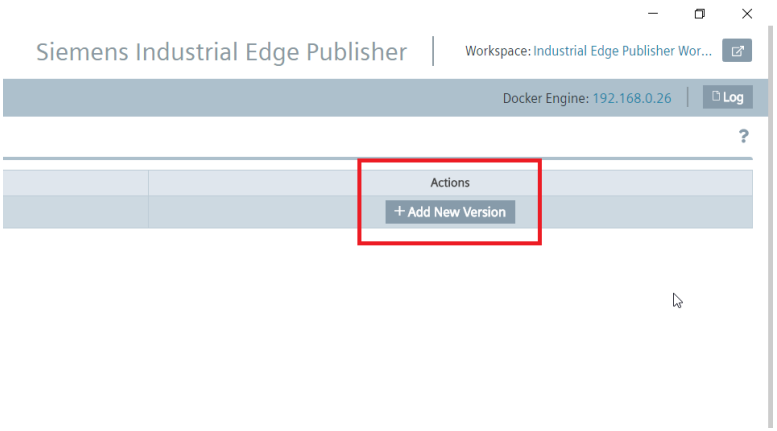


Figura 8-18 Publicación de la app con IEAP. Paso 1

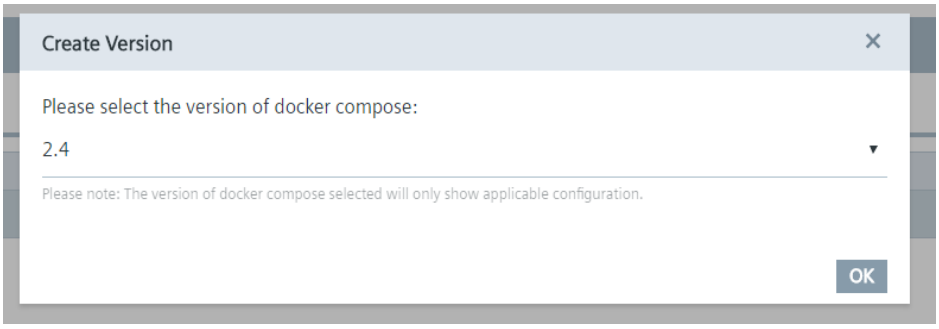


Figura 8-19 Publicación de la app con IEAP. Paso 2

Una vez creada la nueva versión de aplicación le añadiremos los servicios necesarios.

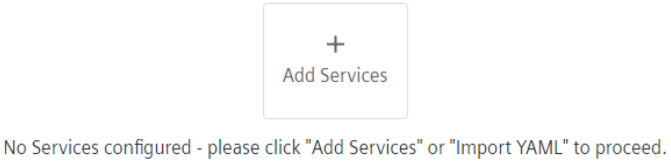


Figura 8-20 Publicación de la app con IEAP. Paso 3

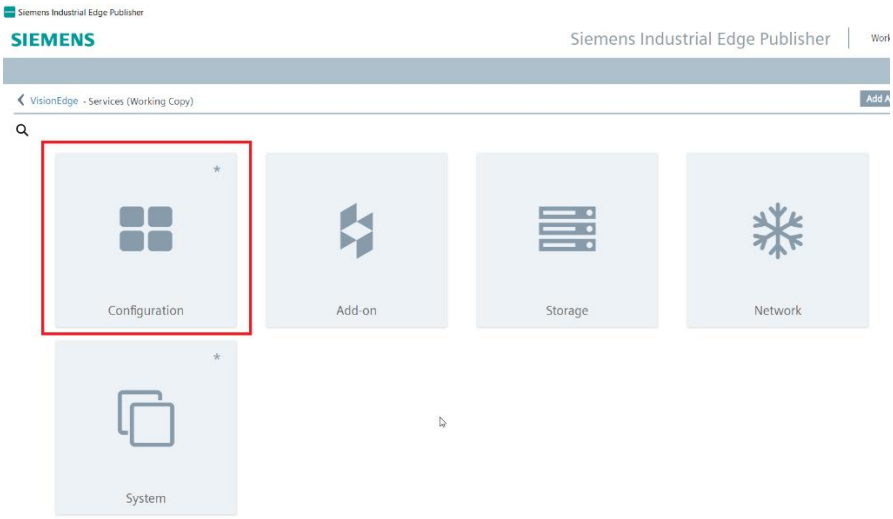


Figura 8-21 Publicación de la app con IEAP. Paso 4

En la ventana de configuración indicaremos el nombre de los servicios de la app, el comportamiento en caso de fallo (Restart) y seleccionaremos a imagen de Docker a publicar a través de la conexión vía API a Docker Engine.

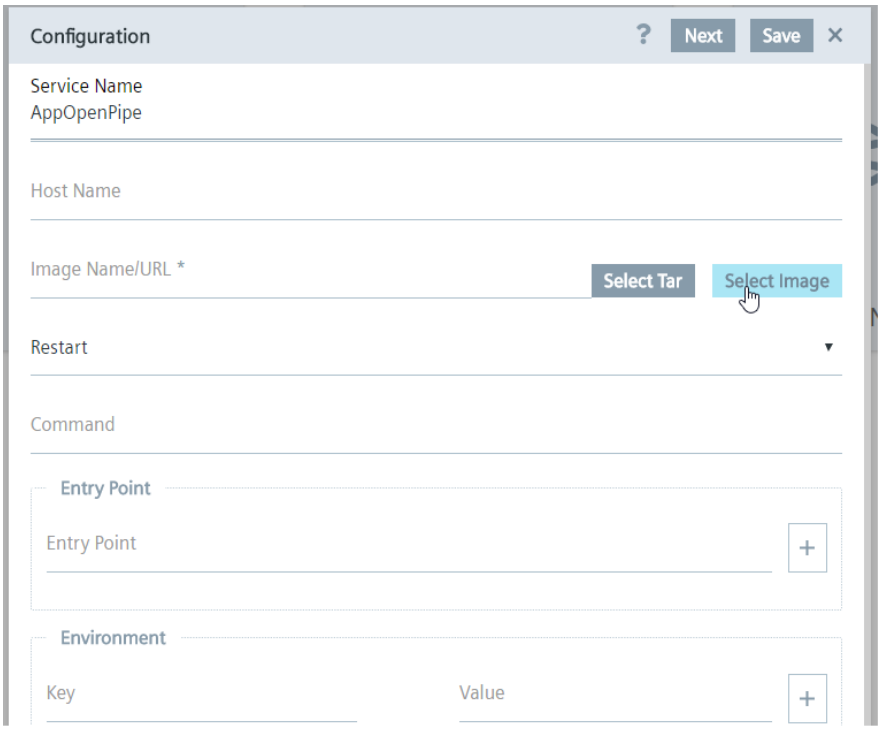


Figura 8-22 Publicación de la app con IEAP. Paso 5

Del listado seleccionaremos la imagen que nos interese de nuestro repositorio de imágenes Docker local.

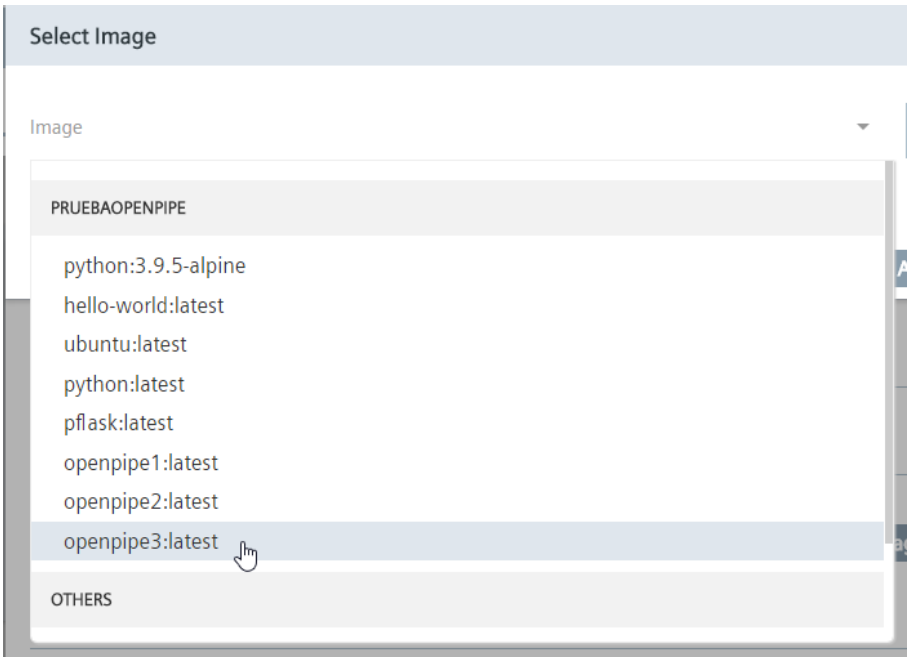


Figura 8-23 Publicación de la app con IEAP. Paso 6

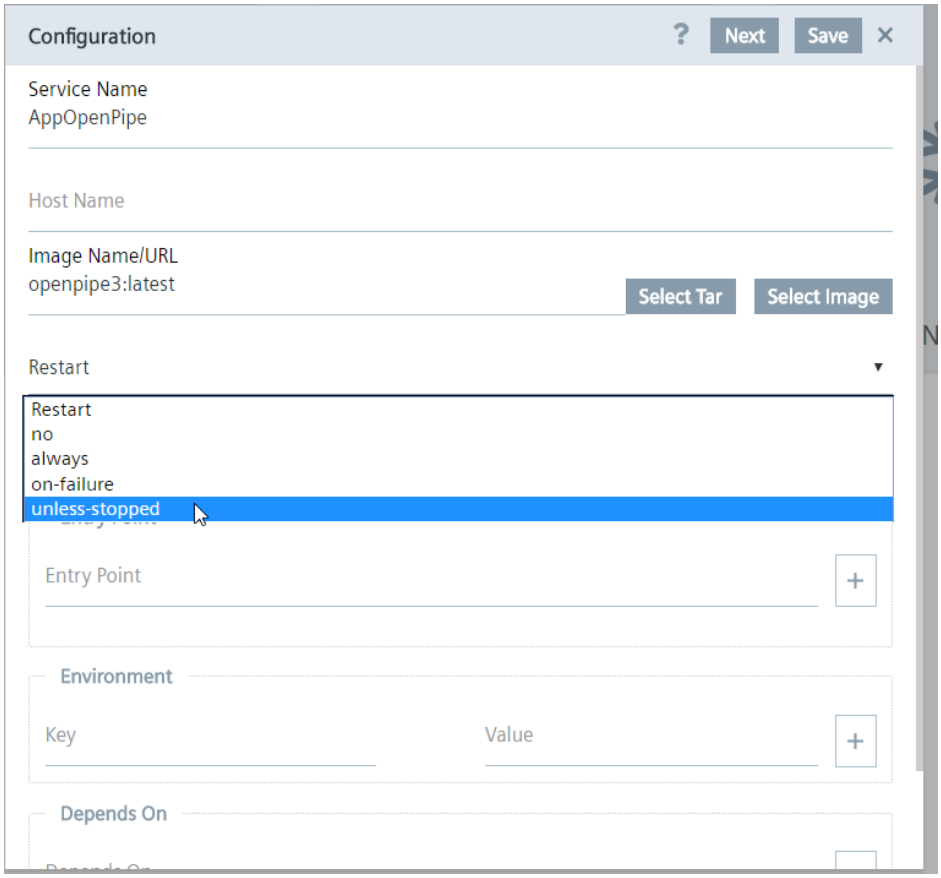


Figura 8-24 Publicación de la app con IEAP. Paso 7

Para la comunicación de nuestra app con las variables del HMI/PLC debemos añadir un volumen interno para habilitar la interfaz OpenPipe.

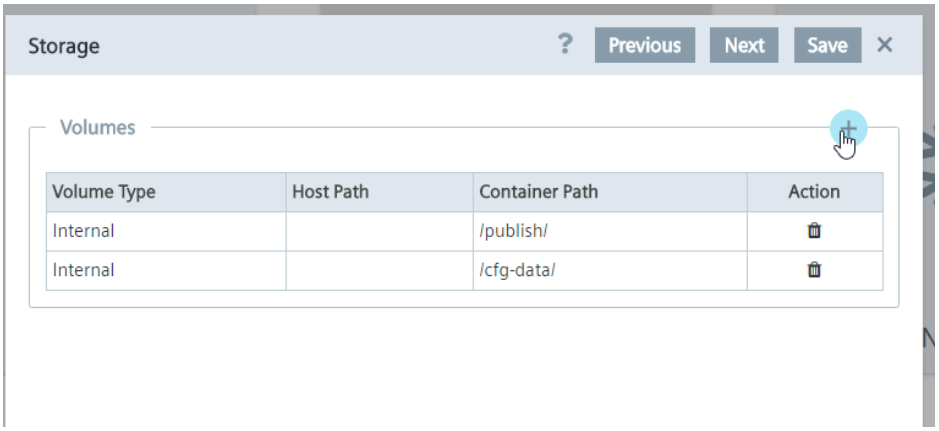


Figura 8-25 Publicación de la app con IEAP. Paso 8

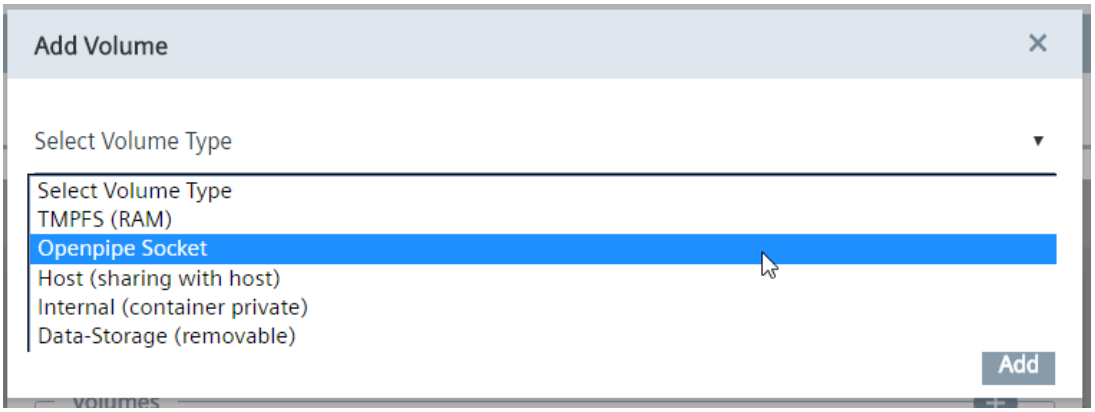


Figura 8-26 Publicación de la app con IEAP. Paso 9

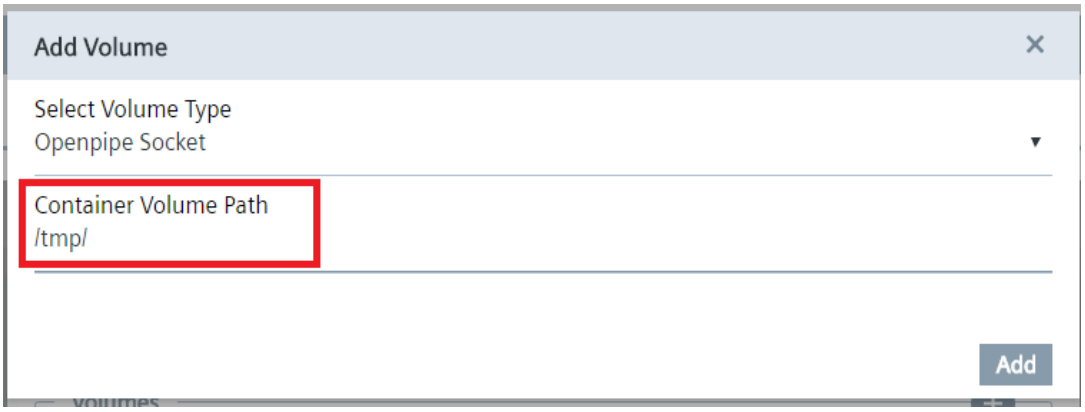


Figura 8-27 Publicación de la app con IEAP. Paso 10

Así se debería ver para asegurarnos de que hemos habilitado el canal OpenPipe.

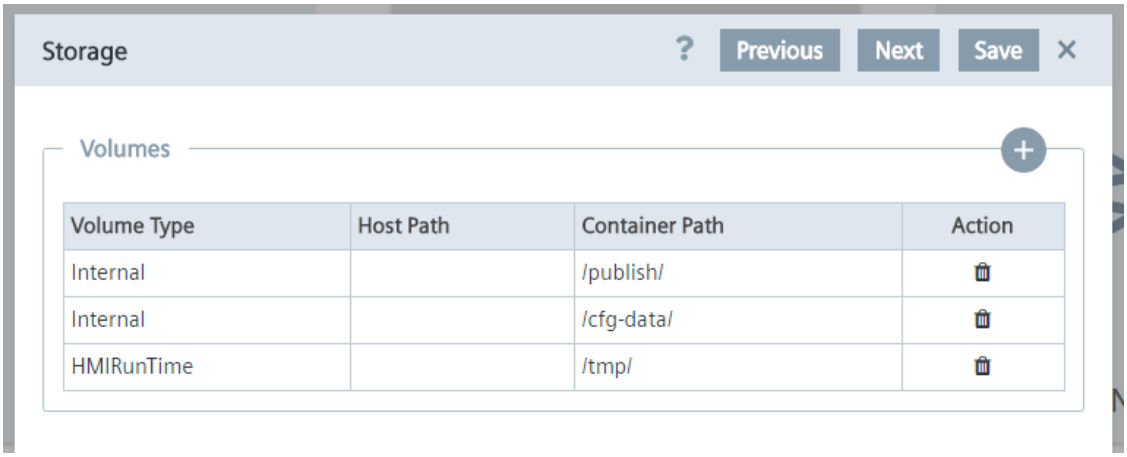


Figura 8-28 Publicación de la app con IEAP. Paso 11

Nuestra aplicación dispone de una interfaz web a al que accedemos a través del puerto 8080 que redireccionamos al puerto 30081 de la UCP para poder acceder desde el exterior.

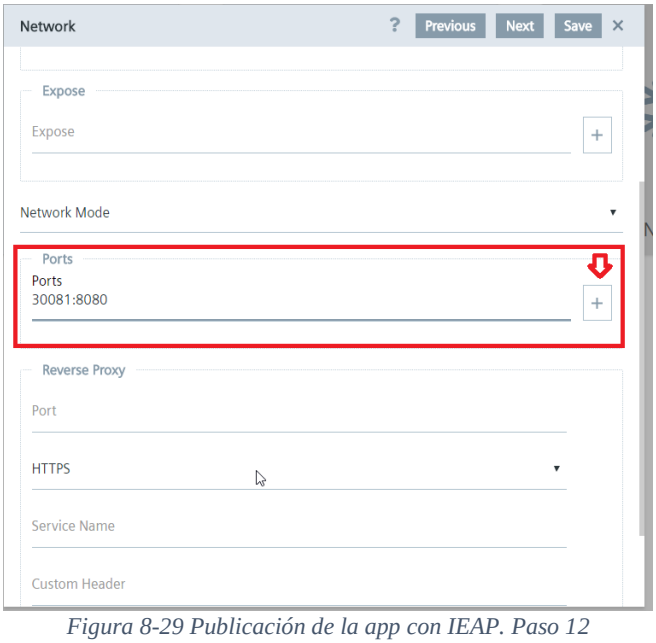


Figura 8-29 Publicación de la app con IEAP. Paso 12

Limitamos la app a un tamaño de 100 MB. Esto es importante ya que el máximo de memoría disponible para apps en una UCP son 800 MB.

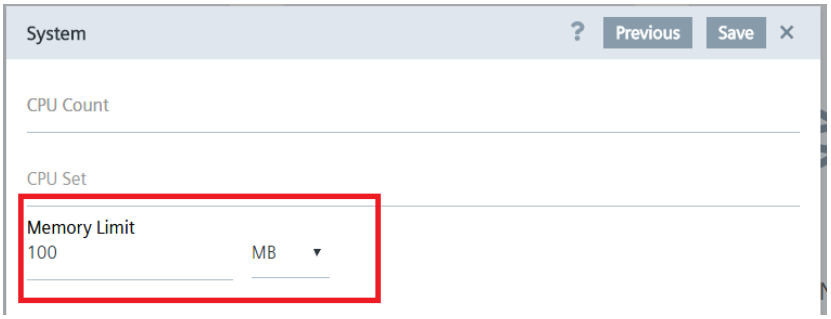


Figura 8-30 Publicación de la app con IEAP. Paso 13

Vemos un resumen de la configuración realizada.

Figura 8-31 Publicación de la app con IEAP. Paso 14

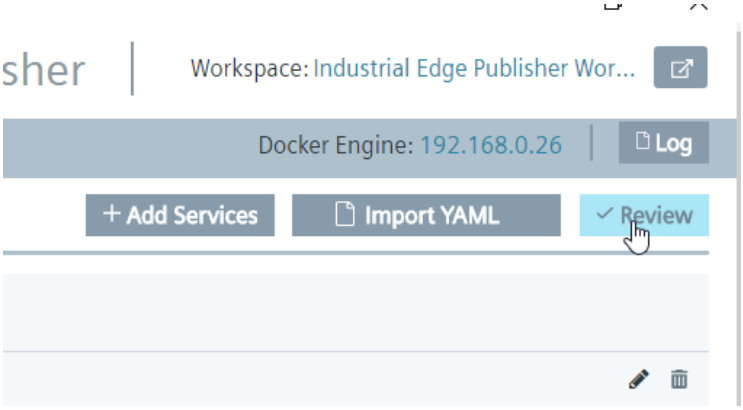


Figura 8-32 Publicación de la app con IEAP. Paso 15

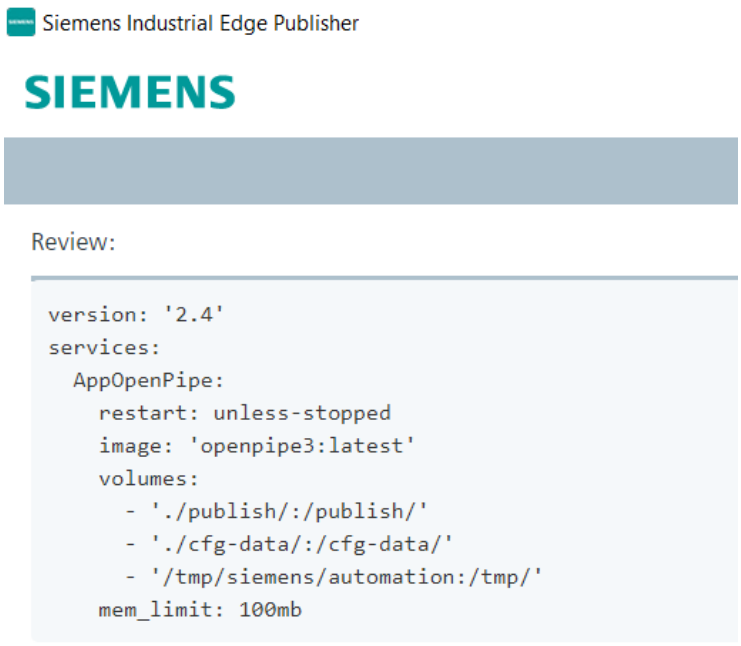


Figura 8-33 Publicación de la app con IEAP. Paso 16

Si todo está OK creamos la App y la exportamos a un directorio de nuestro PC.

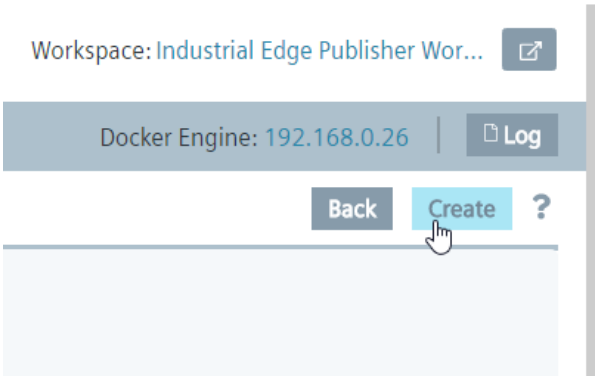


Figura 8-34 Publicación de la app con IEAP. Paso 17

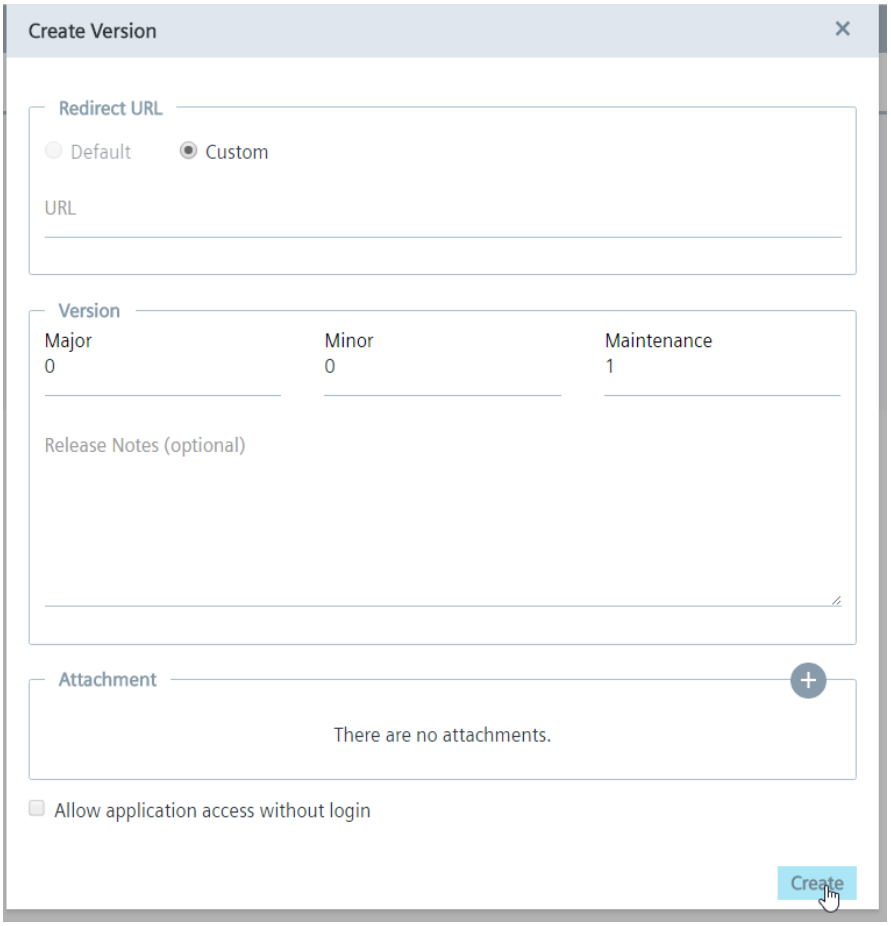


Figura 8-35 Publicación de la app con IEAP. Paso 18

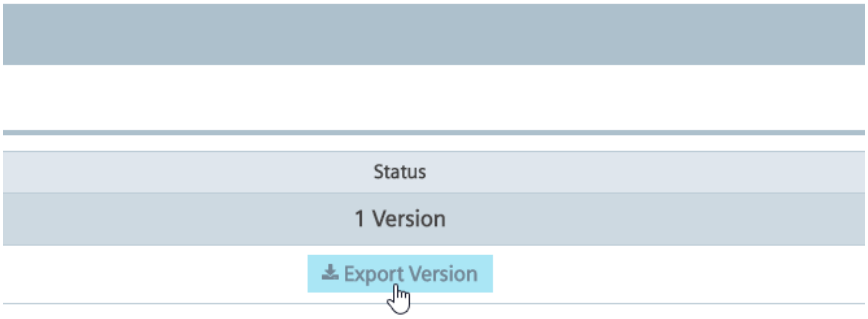


Figura 8-36 Publicación de la app con IEAP. Paso 19

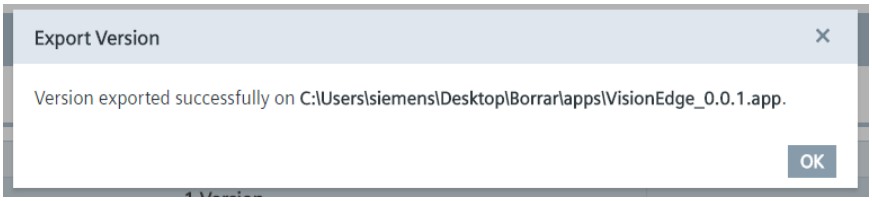


Figura 8-37 Publicación de la app con IEAP. Paso 20

8.5 Instalación App con IEM

Una vez hemos publicado nuestra app con IEAP y generamos el fichero con extensión .app podemos instalar dicha app en cualquier dispositivo UCP con Industrial Edge habilitado.

Vamos a ver el procedimiento para, desde el paneld e control local de la UCP instalar la app.

Accedemos al panel de control de la UCP y accedemos a las aplicaciones:

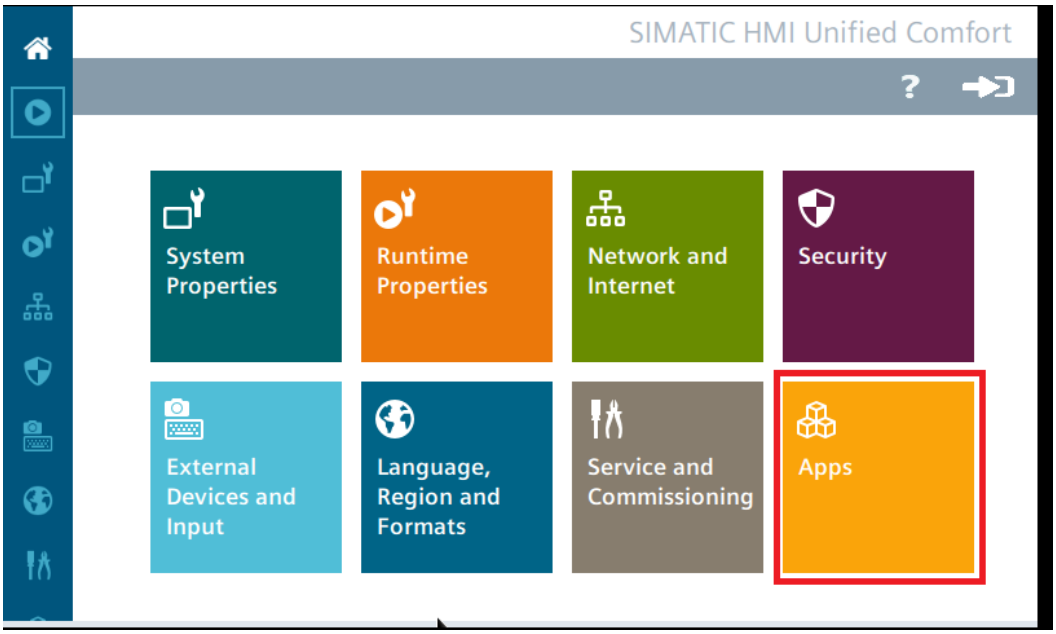


Figura 8-38 Panel de control de un UCP MTP700

Abrimos la configuración de Industrial Edge:

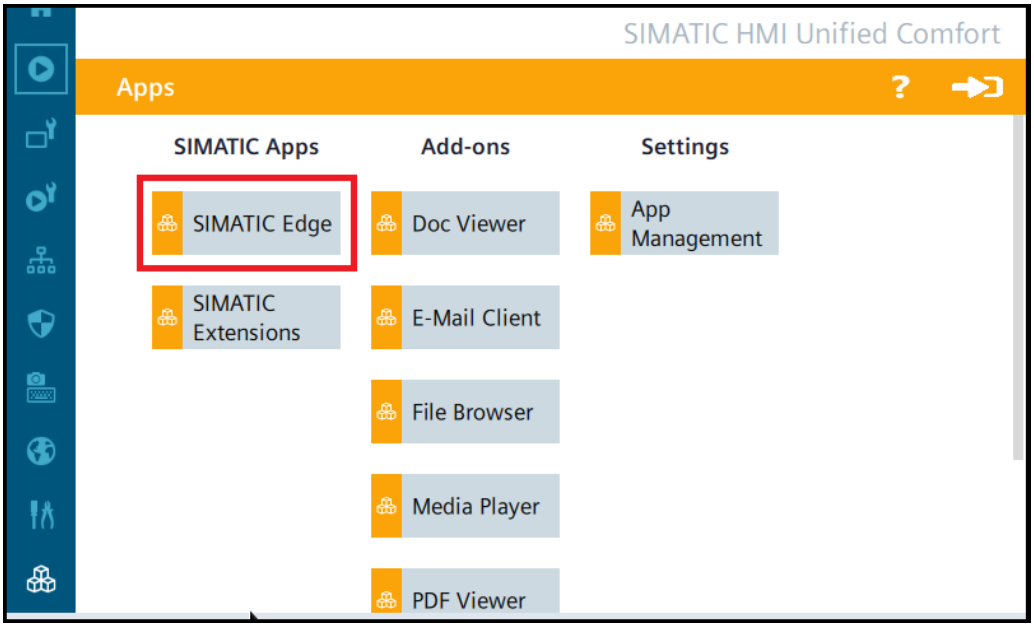


Figura 8-39 Gestión de Apps

Habilitamos trabajar con Industrial Edge:

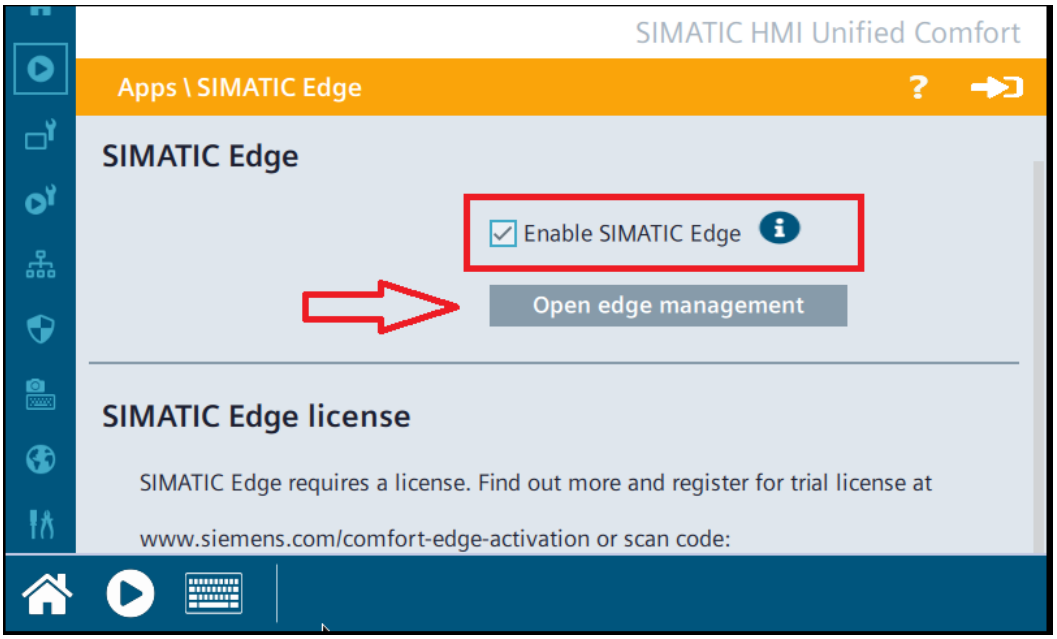


Figura 8-40 Habilitamos trabajar con Edge en la UCP MTP700

Nos registramos con el usuario con permisos de administrador que hemos creado en nuestro proyectod e TIA Portal.

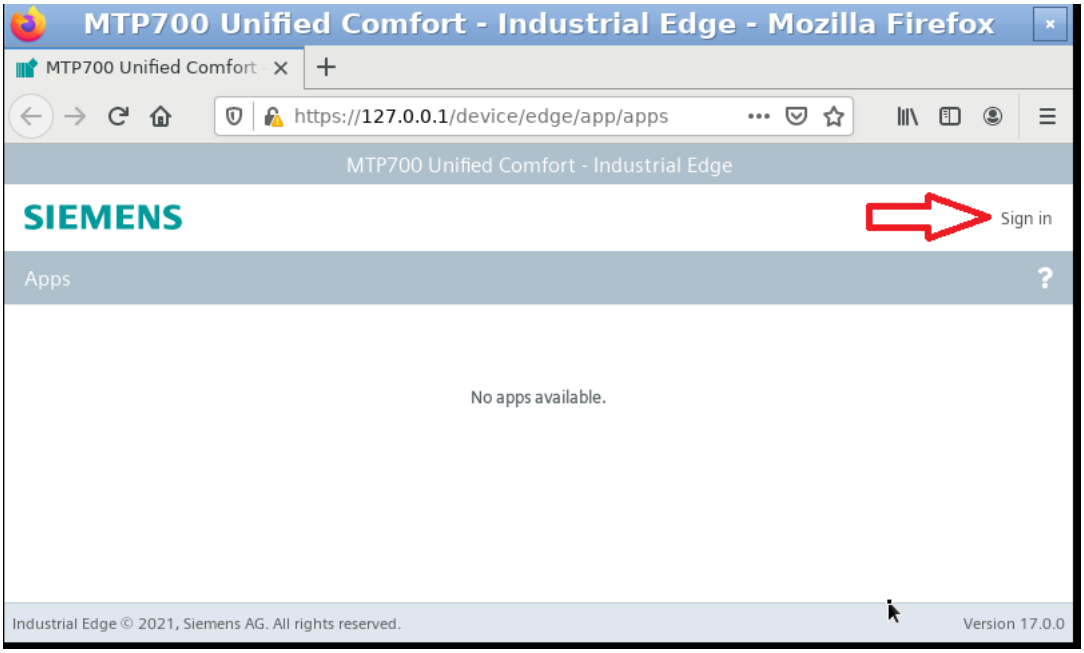


Figura 8-41 Nos registramos con nuestro usuario y contraseña

Hacemos clic en instalar offline.

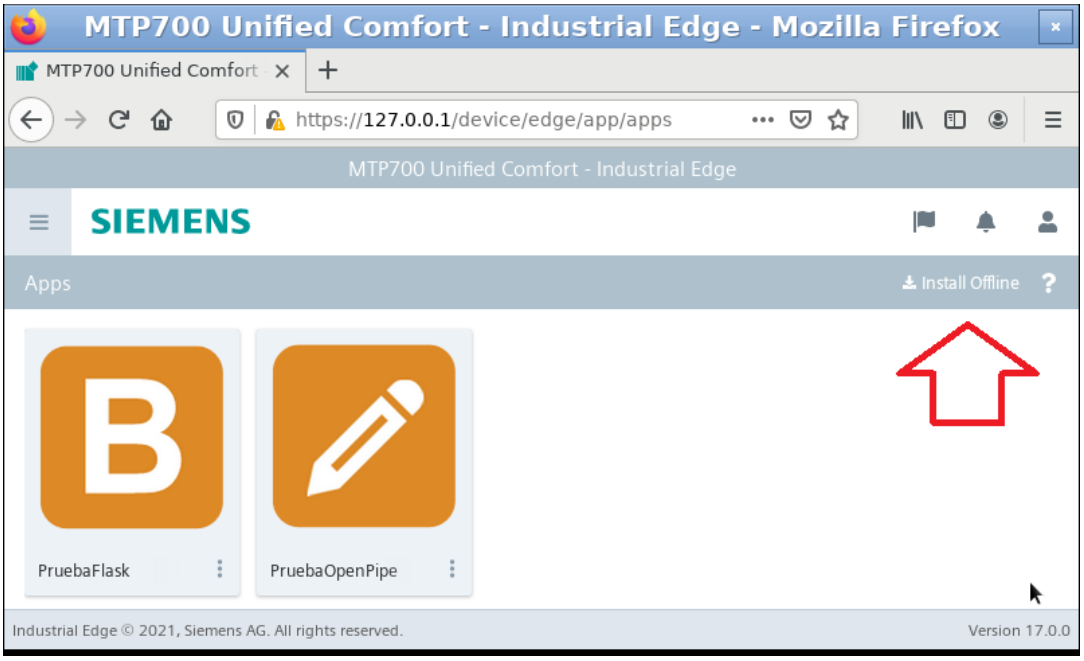


Figura 8-42 Instalamos la App

Nos aparecerá una ventana emergente como la de la figura 8-23

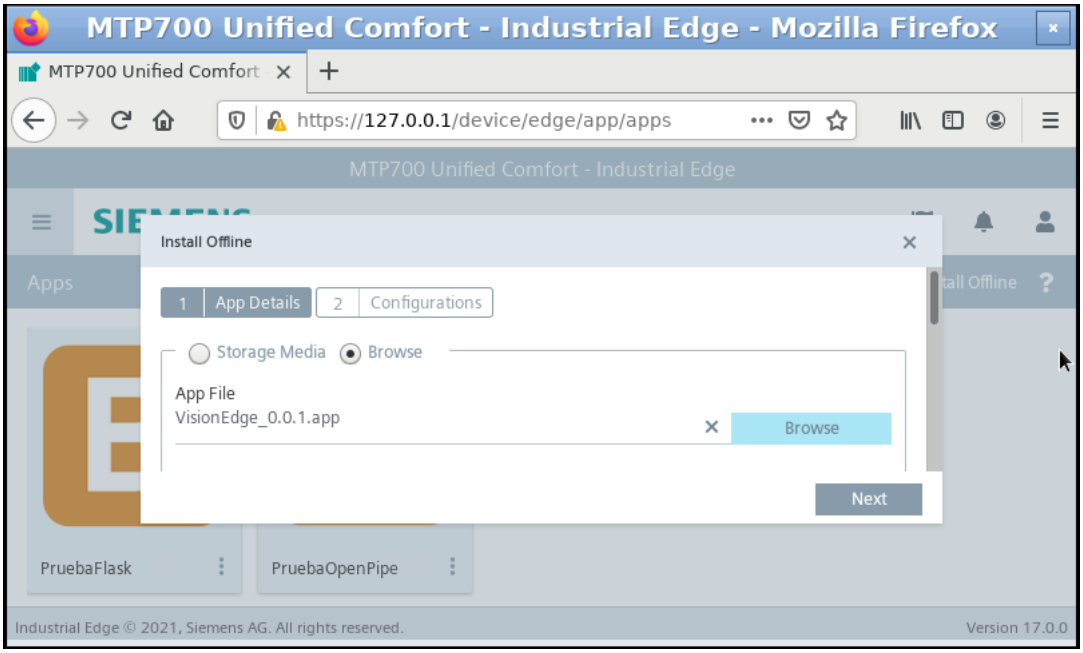


Figura 8-43 Dialogo de instalación de App en IEM

Vemos el progreso de instalación.

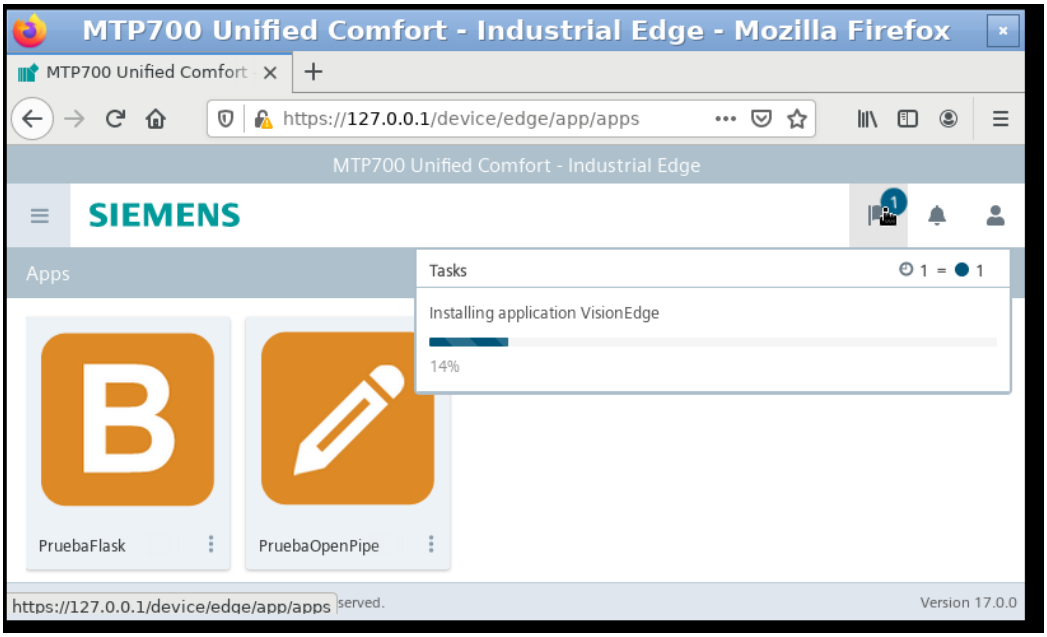


Figura 8-44 Progreso instalación de App

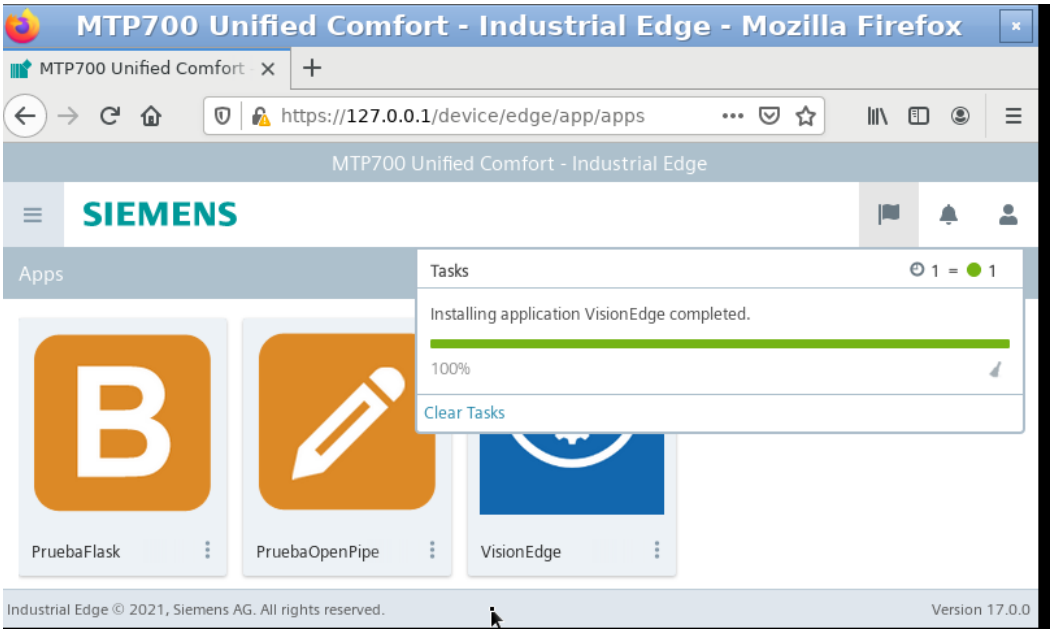


Figura 8-45 Instalación de App finalizada

Una vez finalizado el proceso de instalación ya tenemos lista nuestra App para funcionar.

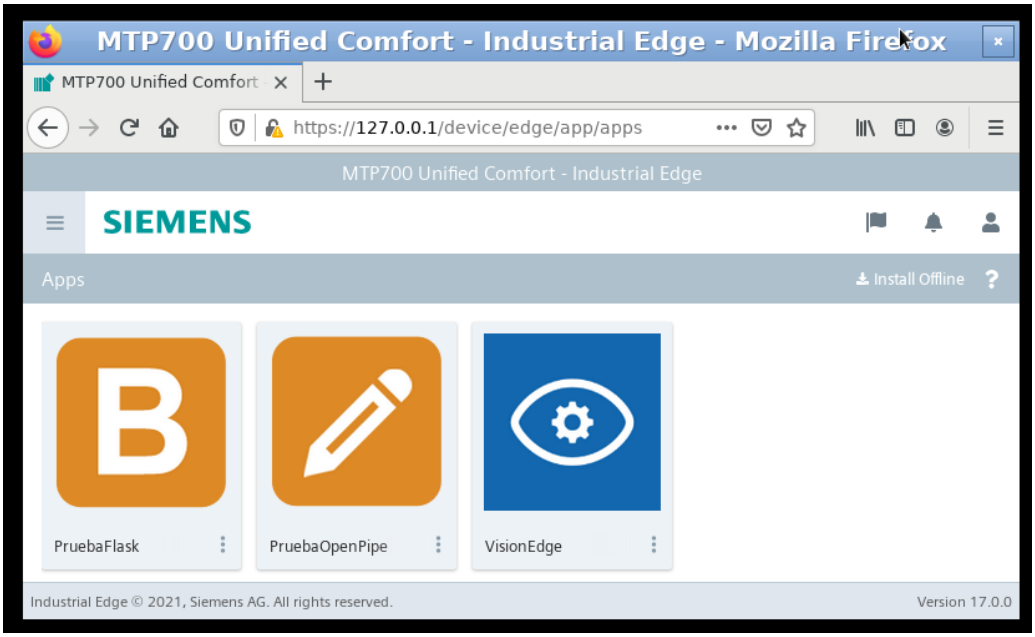


Figura 8-46 Resultado final donde se puede ver nuestra App lista para funcionar.

8.6 Proyecto HMI/PLC en TIA Portal

Se ha desarrollado un proyecto en TIA Portal V17 tanto para la visualización de la línea de clasificación de cajas mediante una MTP700 como para su control mediante un S7-1516F-3PN/DP.
El funcionamiento es el siguiente, cuando la fotocélula detecta una caja se toma una imagen con la cámara situada en la parte superior de la cinta que se manda vía API a Azure Custom Vision para obtener una clasificación de los objetos detectados.
De esta parte se encarga la aplicación desarrollada en Industrial Edge.
Una vez tenemos el resultado de la clasificación desde Azure la aplicación escribe en el HMI la información de los objetos (tipo) y su posición. A continuación se mandan via HMI al PLC y este se encarga de comandar al brazo robótico que es el encargado de recogerlos y depositarlos en una caja que se encuentra en una cinta paralela para su paletizado.
Para controlar este proceso disponemos de las siguientes pantallas en nuestro HMI:

Pantalla principal:
En esta pantalla se va a seleccionar el modo de trabajo de la cinta (manual o automático) y disponemos de tres botones para ir al modo manual, automático o a la pantalla de volcado de la detección (Openpipe)

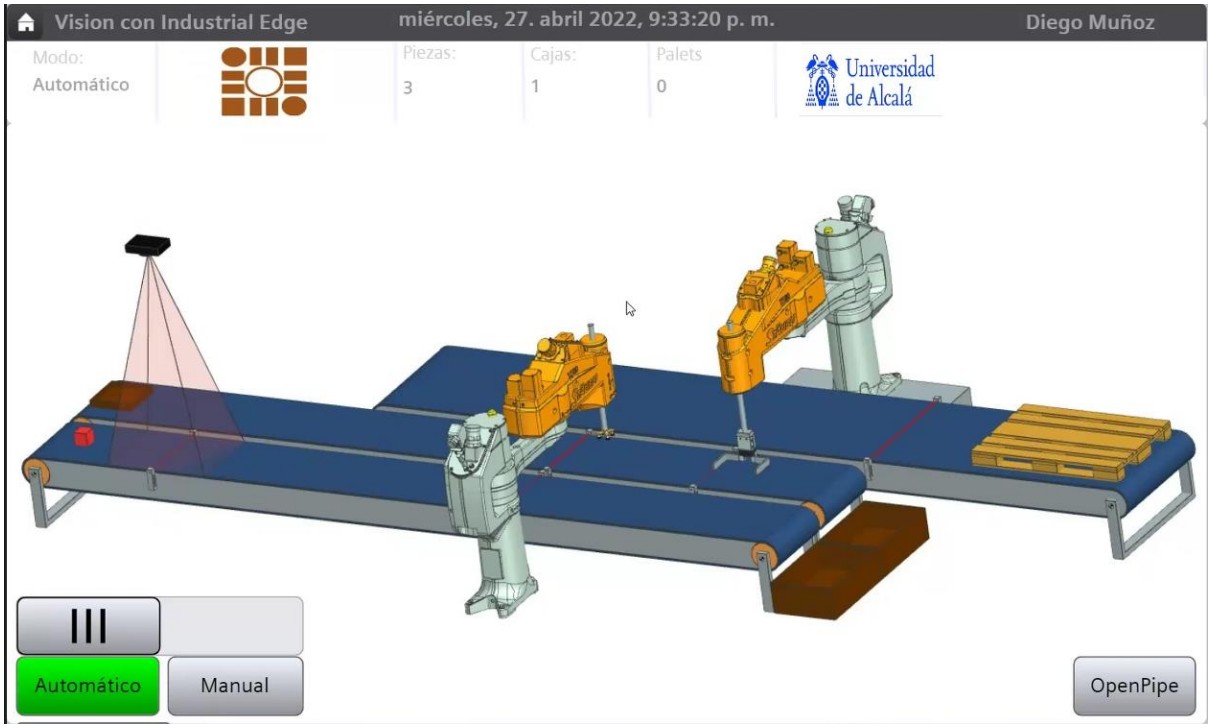


Figura 8-47 Pantalla principal de la aplicación

Pantalla modo automático:
En esta pantalla nos va a permitir seleccionar mediante un desplegable que tipo de caja debe recoger el brazo robótico y lanzar el funcionamiento de la cinta.

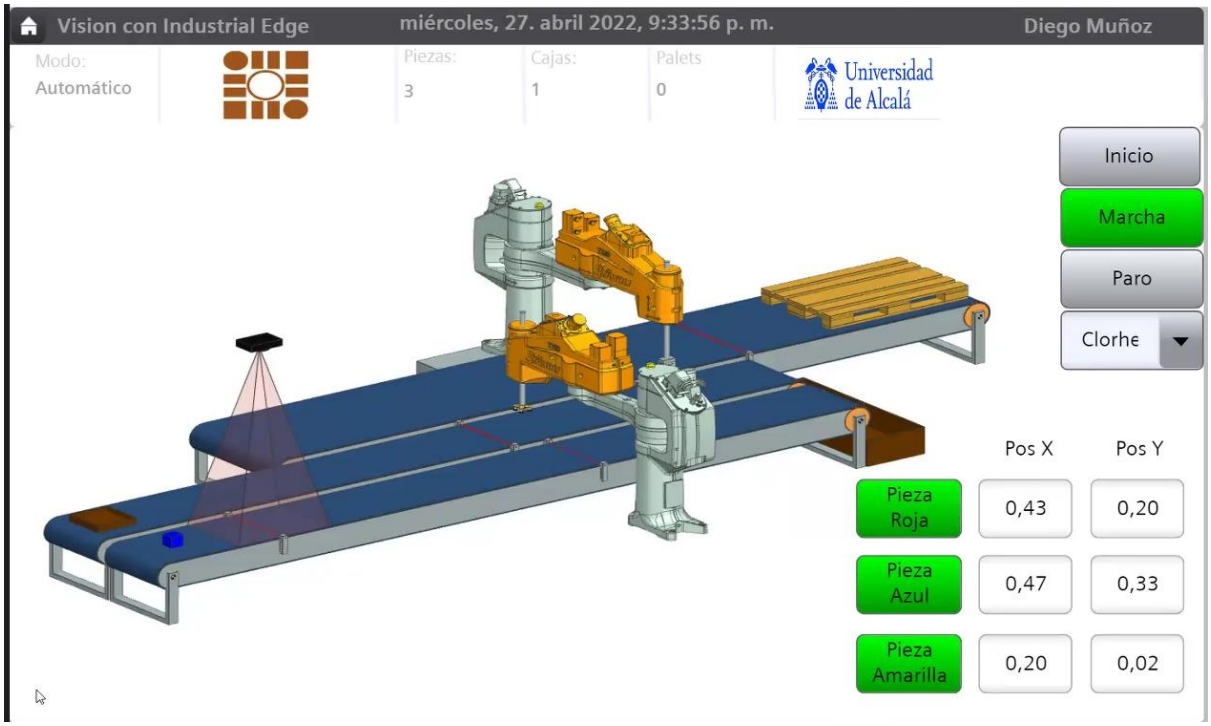


Figura 8-48 Pantalla modo automático de la cinta

Pantalla modo manual:

En esta pantalla el funcionamiento de las cintas se hará de modo manual sin intervención de la cámara.

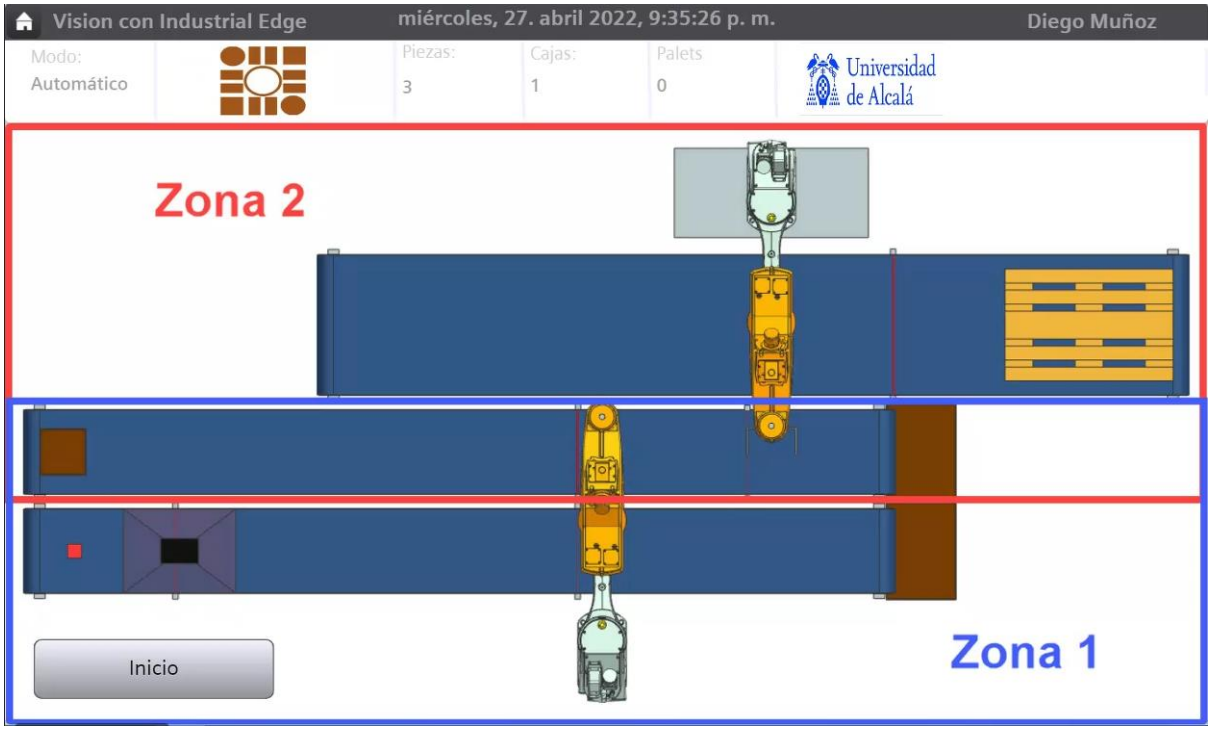


Figura 8-49 Control manual

Pantalla volcado detección:
En esta pantalla podemos visualizar el resultado de la última predicción obtenida de una imagen tomada por la cámara.

Figura 8-50 Pantalla volcado resultado detección de objetos

9. Puesta en marcha virtual

9.1 Visión general

Una vez tenemos nuestro aplicativo terminado vamos a probarlo en un entorno virtual que simula un proceso real de fabricación con las herramientas que provee Siemens para "virtual commissioning".
Para realizar esta "puesta en marcha virtual", se requiere una imagen de la máquina real. Esta imagen se conoce como el gemelo digital de la máquina.
Por medio de un gemelo digital, la interacción de los componentes individuales de una máquina se puede simular y optimizar en el mundo virtual, sin necesidad de un prototipo real. La puesta en marcha virtual de una máquina nos permite reducir el riesgo de errores y la carga de trabajo necesaria para la puesta en marcha real. Lo que impacta en los tiempos de lanzamiento al mercado de un nuevo prototipo, así como una mayor flexibilidad, eficiencia y calidad.

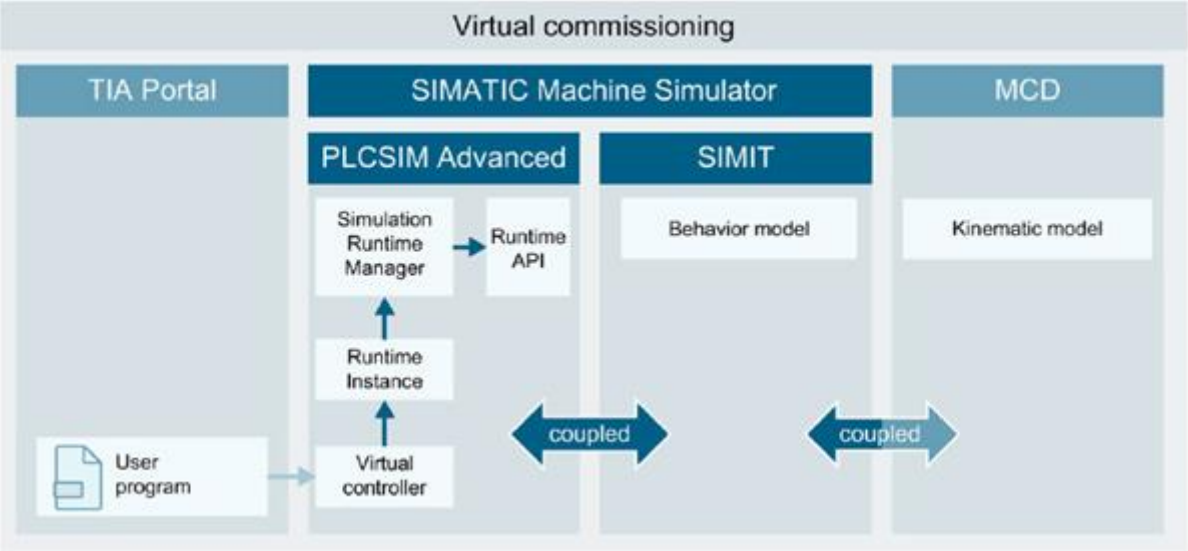


Figura 9-1 Figura 0 1 Ilustración 1 Ecosistema SIMATIC para puesta en marcha virtual. Fuente: SIMATIC Machine Simulator Virtual commissioning of machines Getting Started[62]

Ventajas de la puesta en marcha virtual

- Calidad: Nos permite optimizar el proyecto del HMI, controlador y la funcionalidad de la máquina en un entorno virtual
- Rapidez: Menos tiempo para la puesta en marcha en la planta del cliente final. Paralelización de ingeniería mecánica y automatización. No tenemos que esperar a que la máquina esté montada para empezar a probar la parte de control y programación.
- Costes: Un error que corregir nos pueda costar un euro en la fase de planificación serían 1000 euros en la parte de operación según estudios de Siemens Industry Image Database 4.11 Reducimos también los costes de puesta en marcha (Por ejemplo, tiempo del personal desplazado).

[62] https://support.industry.siemens.com/cs/es/en/view/109758943

- Riesgo: Pruebas seguras y eficientes utilizando el modelo. Reducimos el riesgo de la puesta en marcha real y con menos fallos de operación.
- Flexibilidad: Disponemos de un "laboratorio" para la creación de conceptos de control alternativos sobre el mismo modelo mecánico de la máquina que nos permite la evaluación de las modificaciones de la máquina durante el funcionamiento.

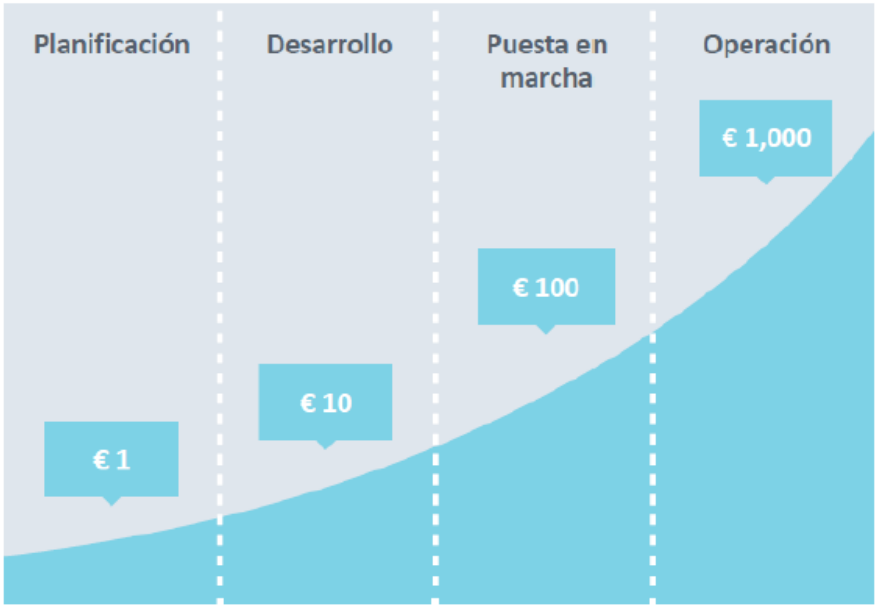


Figura 9-2 Coste errores en cada fase del proyecto. Fuente Siemens Industry Image Database 4.11

Para nuestro proyecto vamos a crear el "Gemelo Digital" de una línea para clasificado de cajas de medicamentos compuesta por una cámara que envía la imagen al Unified Comfort Panels (dispositivo Edge y HMI) para su procesado, este dispositivo será simulado mediante las herramientas al efecto de las que disponemos en TIA Portal, una vez procesada la imagen el resultado de la clasificación de los objetos se manda a un controlador Siemens S7-1500, el cual será simulado mediante PLCSim Advanced. Este controlador es quien gobierna el brazo robótico (SCARA) que es el encargado de recoger las cajas que nos interesen para su paso a otra línea de procesado. La parte del modelo físico de la línea (Camara, cinta, brazo robótico, cajas…) corre a cargo de la herramienta NX Mechatronics Concept Designer.

Figura 9-3Entorno de desarrollo Siemens para puesta en marcha virtual. Fuente: Siemens Industry Image Database 4.11

Vamos a ver a continuación como se puede implementa este modelo digital de la instalación con las herramientas antes mencionadas.

Herramientas necesarias para la puesta en marcha virtual:

- **TIA Portal, STEP 7 y WinCC Engineering:** En el TIA Portal, creamos nuestro programa de usuario con STEP 7 y las pantallas para operar y controlar máquinas con WinCC en un dispositivo HMI, en nuestro caso un Unified Comfort Panels que es a su ves un dispositivo IE.

- **PLCSIM Advanced:** Con PLCSIM Advanced, simulamos el programa del controlador sin el hardware real. El programa de STEP 7 se carga en un controlador S7-1500 virtual a través de PLCSIM Advanced.

- **MCD**: Con MCD, simulamos y probamps los componentes mecánicos de la máquina en un entorno virtual. MCD reúne datos CAD con un motor físico: los datos CAD se hacen cinemáticos. La especificación de los grados de libertad de los diferentes componentes mecánicos nos permite definir la cinemática del modelo.

- **SIMIT:** El software de simulación SIMIT mapea el comportamiento de los componentes activos (ej, accionamientos o válvulas). En SIMIT, podemos simular escenarios de error para analizar el comportamiento de la máquina en un entorno virtual. SIMIT intercambia consignas/valores reales de la simulación a través de un acoplamiento con MCD. En nuestro caso no ha sido necesario utilizar SIMIT ya que al tratarse de un movimiento del brazo robótico muy sencillo este se ha implementado directamente en el código del PLC.

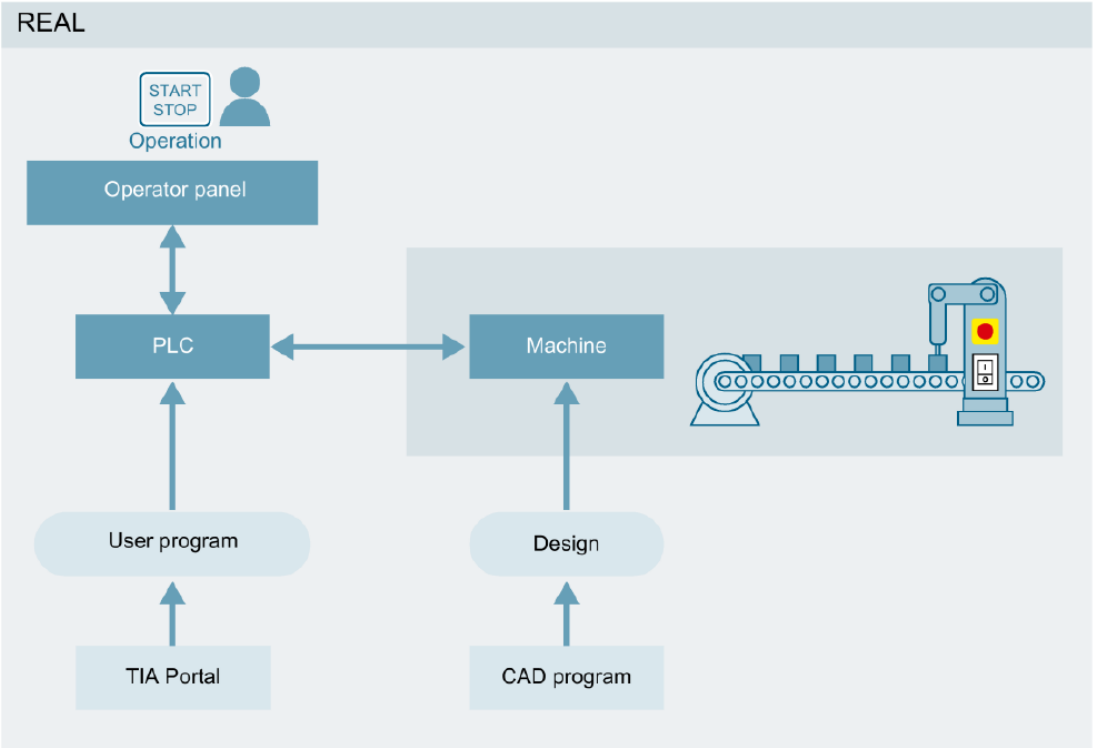


Figura 9-4 Mundo real. Fuente: SIMATIC Machine Simulator Virtual commissioning of machines Getting Started

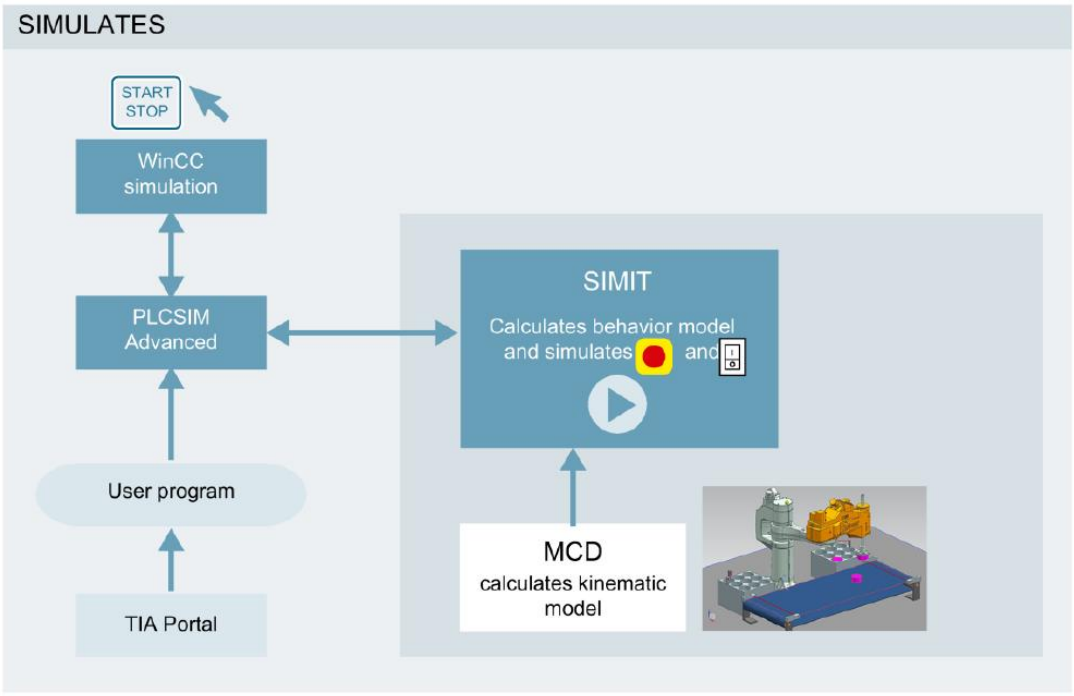


Figura 9-5 Mundo virtua. Fuente: SIMATIC Machine Simulator Virtual commissioning of machines Getting Started

9.2 Gemelo Digital

Con PLCSIM Advanced podemos simular y probar el programa de usuario creado en un controlador virtual. Por otro lado PLCSim Adv dispone de una API que permite que las interfaces de usuario estén disponibles para acceder en tiempo de ejecución de la simulación a las variables del controlador por parte del modelo de comportamiento.
En este ejemplo de aplicación, la comunicación local a través de PLCSIM-Softbus se utiliza para el intercambio de datos con el TIA Portal. Con PLCSIM Advanced, también es posible la comunicación distribuida a través de un adaptador Ethernet virtual.
La interfaz con MCD se realiza a través del acoplamiento PLCSIM Advanced, que permite una importación sencilla de las entradas y salidas del hardware a simular.
.

El modelo virtual del ejemplo de aplicación consta de:
• 1 SCARA con ejes 1 a 3
• 1 cinta transportadora
• 1 fotocelula
• Piezas de trabajo (cajas de medicamentos)
• 2 cajas para almacenar las piezas de trabajo

Nuestro ejemplo de aplicación procede de la siguiente manera:

- La cinta transportadora se activa.

- Cuando la fotocélula de la cinta transportadora detecta una caja la cinta se para y la cámara toma una foto que se envía a el dispositivo Edge (UCP) para clasificar los objetos detectados.
- La información relativa a los objetos detectados se envía al PLC y arrancamos la cinta de nuevo.
- El PLC le indica al robot la posición de la caja para que este proceda a su recogida si es una caja de las seleccionadas para recoger y su depósito en un palet de otra cinta contigua.
- Una caja se llena al máximo con 2 cajas de medicamentos entonces el robot cambia a una caja vacía.

MCD simula los componentes mecánicos en función de los grados de libertad y puntos de ajuste definidos y representa visualmente la máquina simulada.

Variables y tipos de datos:
Las siguientes variables de entrada y salida se definen para la comunicación del controlador con el sistema de transporte:

MCD_inputs

	Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint
1	Static							
2	Posicion_caja	Real	0.0					
3	Pos_roja_X	Real	0.0					
4	Pos_roja_Y	Real	0.0					
5	Pos_azul_X	Real	0.0					
6	Pos_azul_Y	Real	0.0					
7	Pos_amarillo_X	Real	0.0					
8	Pos_amarillo_Y	Real	0.0					
9	Sensor_1_C_1	Bool	false					
10	Sensor_2_C_1	Bool	false					
11	Sensor_1_C_2	Bool	false					
12	Sensor_2_C_2	Bool	false					
13	Sensor_1_C_3	Bool	false					
14	Scara_1_pos_act_X	Real	0.0					
15	Scara_1_pos_act_Y	Real	0.0					
16	Scara_1_pos_act_Z	Real	0.0					
17	Scara_1_Coger_FC	Bool	false					
18	Scara_2_pos_act_ang.	Real	0.0					
19	Scara_2_pos_act_ang.	Real	0.0					
20	Scara_2_pos_act_Z	Real	0.0					
21	Scara_2_Coger_FC	Bool	false					
22	Bit_vida_MCD	Bool	false					

Figura 9-6 Mapeado de entradas externas a NX MCD

MCD_outputs

	Name	Data type	Start value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint
1	Static							
2	Vel_cinta_1	Real	0.0					
3	Vel_cinta_2	Real	0.0					
4	Vel_cinta_3	Real	0.0					
5	Leer_posicion	Bool	false					
6	Producto_rojo	Bool	false					
7	Producto_azul	Bool	false					
8	Producto_amarillo	Bool	false					
9	Caja_grande	Bool	false					
10	Palet	Bool	false					
11	Scara_1_X	Real	2010.023					
12	Scara_1_Y	Real	629.98					
13	Scara_1_Z	Real	100.0					
14	Scara_1_Coger	Bool	false					
15	Scara_1_Dejar	Bool	false					
16	Scara_2_angulo_base.	Real	0.0					
17	Scara_2_angulo_eje1..	Real	360.0					
18	Scara_2_Z	Real	150.0					
19	Scara_2_Coger	Bool	false					
20	Scara_2_Dejar	Bool	false					
21	Bit_vida_PLC	Bool	false					

Figura 9-7 Mapeado de salidas externas a NX MCD

Componentes hardware de la simulación:

Referencia	Descripción
6ES7 516-3FN02-0AB0	CPU 1516F-3PN/DP Fail-safe CPU with display; work memory 1.5 MB code and 5 MB data; can be used for safety applications; supports consistent safety upload; supports PROFIsafe V2; 10 ns bit operation time; 5-stage protection concept, technology functions: motion control, closed-loop control, counting and measuring; tracing; Runtime options; isochronous mode (central); for all PROFINET interfaces: transport protocol TCP/IP.
6AV2 128-3GB06-0AX1	MTP700 Unified Comfort Panel 7.0" TFT display, 800 x 480 pixels, 16M colors; Multi touch; 1 x 422/485, 1 x PROFINET/Industrial Ethernet interface with MRP (2 Ports); 1 x Ethernet (Gigabit); 2 x SD card slot; 4 x USB
TS2-80	SCARA robot

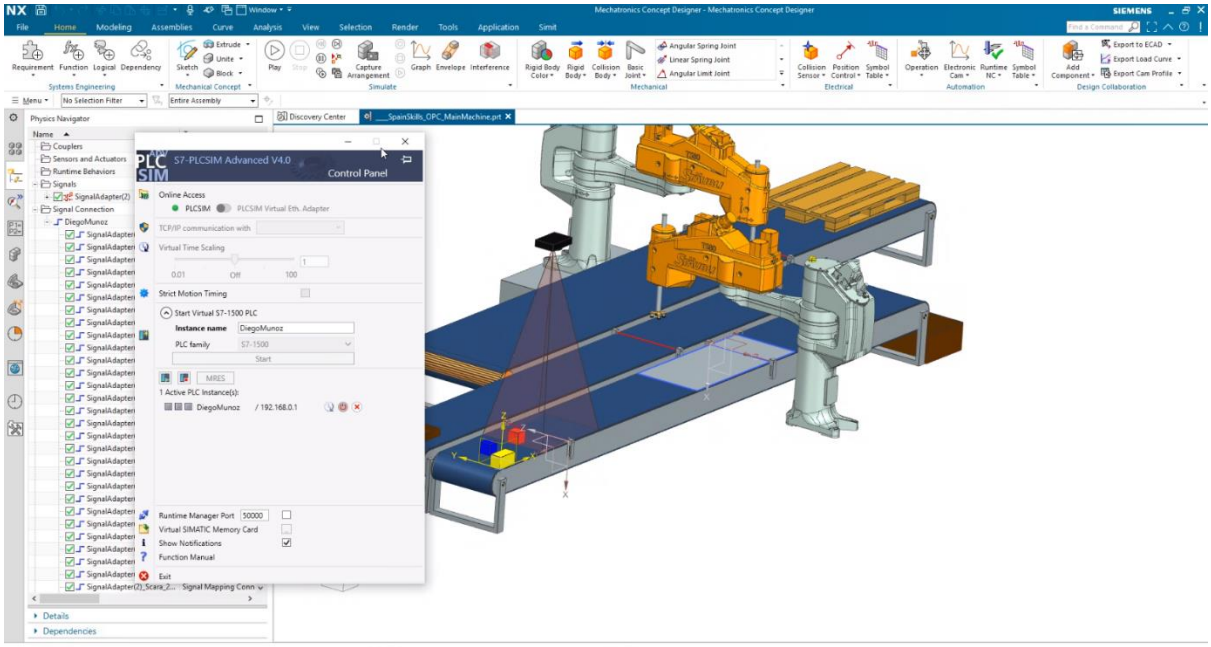


Figura 9-8 Figura 1 21 2.- Gemelo digital de la célula de producción

9.3 Arrancar la simulación

Lo primero que haremos será arrancar PLCSim Advanced y cargarle nuestro proyecto de TIA Portal.

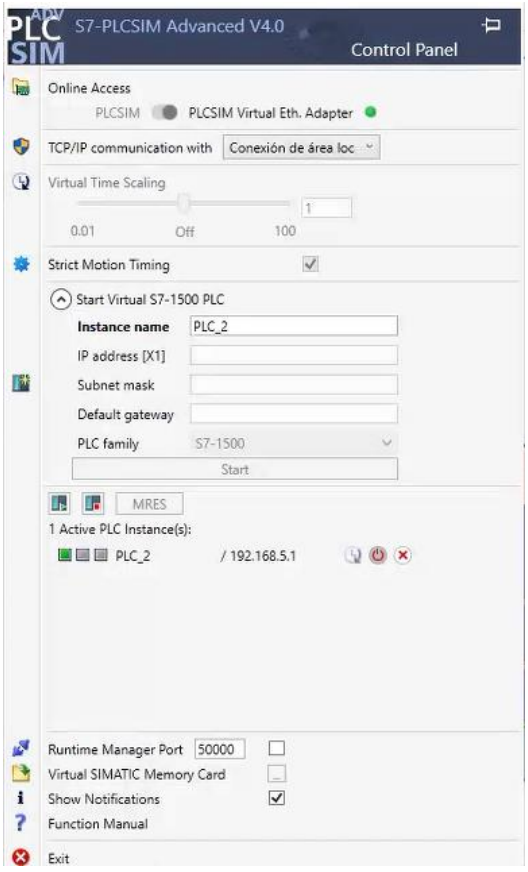


Figura 9-9 PLCSim Advanced 4.0

A continuación, arrancamos MCD NX y conectamos nuestro modelo CAD con PLCSim Adv. vía API y hacemos un match entre las variables del proyecto del PLC con las del modelo mecatrónico.
Para que el match sea automático hemos creado en ambos softwares las variables con el mismo nombre.

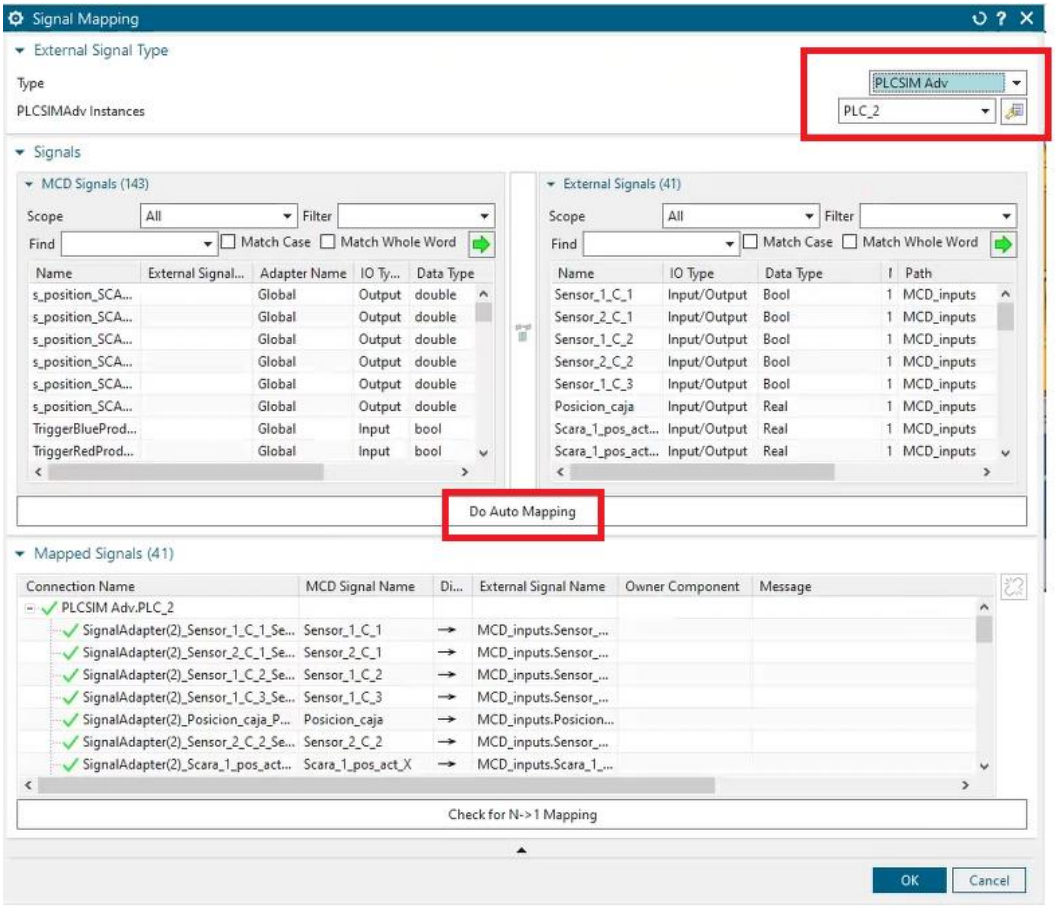


Figura 9-10 Mapeo de señales del exterior con NX MCD

Una vez hemos mapeado las señales estamos listos para arrancar la aplicación y simular el funcionamiento de nuestra línea de clasificación.

Figura 9-11 Play de la simulación NX MCD

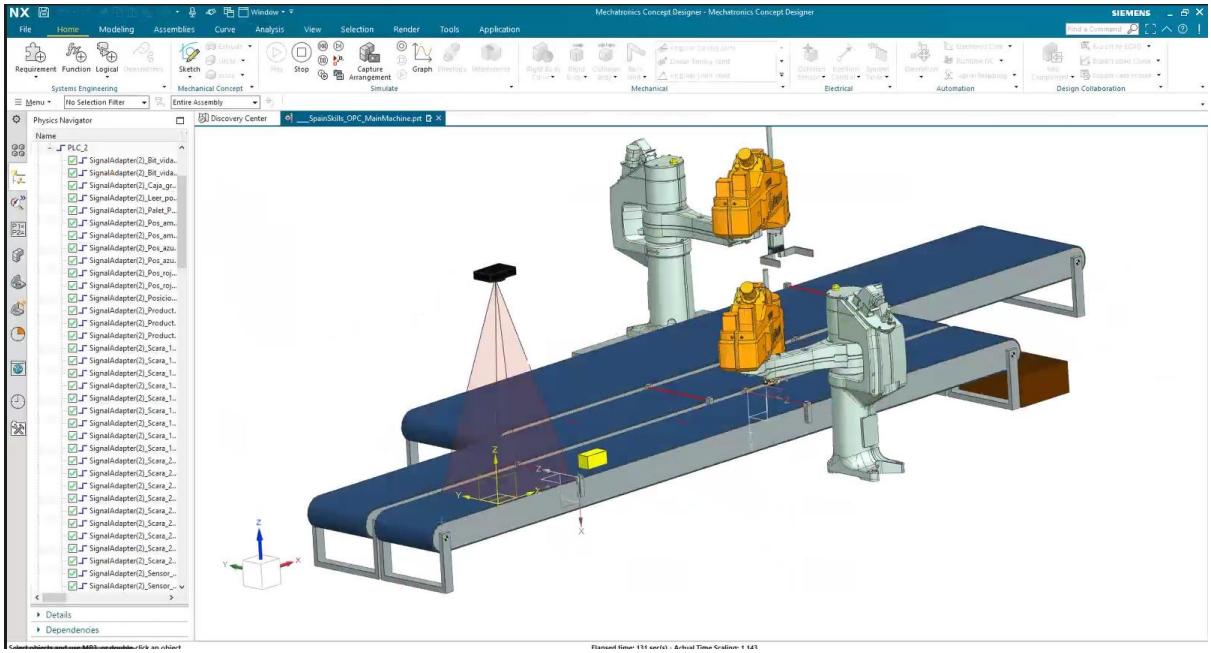


Figura 9-12 Simulación en marcha

10. Conclusiones y trabajo futuro

10.1 Conclusiones

La motivación principal de este TFG ha sido desarrollar una guía práctica para que cualquier interesado en empezar a trabajar con tecnologías Edge Computing orientadas a entornos industriales pueda desarrollar sus propias apps para ejecutarse en el entorno Siemens Industrial Edge.

Como hilo conductor y caso práctico de esta guía se ha utilizado una aplicación para Visión Artificial ya que es una de las áreas donde la tecnología Edge puede ser de más aplicación en el futuro cercano.

Se ha podido comprobar que los dispositivos Edge computing brindan una amplia gama de funcionalidades como son:

1. Sustitución de aplicaciones basadas en PC de difícil mantenimiento y despliegue

2. Adquisición de datos de producción para su procesado y análisis local o en la nube

3. Conectividad de aplicaciones haciendo de pasarela entre las redes de operación OT (Operational Technology) y las redes IT (Information Technology) para dar cobertura a las necesidades crecientes de comunicar un volumen cada vez mayor de datos.

4. Control de procesos en lazo cerrado basados en algoritmos implementados en lenguajes de alto nivel y/o con inteligencia artificial.

Por otro lado, con el gestor Industrial Edge Management se cubren funcionalidades como la implementación centralizada, la actualización y el control de versiones de aplicaciones, así como la gestión de dispositivos.

Entre los beneficios de trabajar con Siemens Industrial Edge Computing se cuentan:

1. Hardware dedicado: El hardware para correr la runtime de tiempo de ejecución de Siemens Industrial Edge incorpora preinstalado y configurada la misma runtime para un despliegue rápido de soluciones Edge en planta. Con mecanismos de backup y restore bien establecidos. Por otra parte, si se requiere correr la rumtime en un PC no siemens está previsto el lanzamiento en unos meses de la “Runtime Virtual” para habilitar esa posibilidad.

2. Conexión a Internet: Siemens Industrial Edge puede trabajar en modo standalone sin necesidad de conexión a internet de los dispositivos Edge pudiendo ser estos configurados y mantenidos desde un Industrial Edge Management (IEM) instalado en un PC de la red de planta que puede estar instalado en una DMZ para asegurar la instalación contra accesos no autorizados.

3. Usabilidad: Interfaz de gestión sencilla tanto en los dispositivos Edge locales como en el gestor que permite a usuarios sin conocimiento avanzados IT desplegar soluciones basadas en Industrial Edge.

4. Seguridad incluida por defecto por Siemens

5. El formato de contenedor Docker permite independencia y escalabilidad del hardware

6. Posibilidad de elegir cualquier lenguaje de programación disponible en un contenedor Docker

10.2 Trabajo futuro

El hecho de haber desarrollado una aplicación que se conecta con una red neuronal para visión artificial alojada en la nube de Azure supone unos tiempos de respuestas que se ven influenciados por múltiples factores como la latencia de la red o el TPS del algoritmo de Azure (dos transacciones por segundo como máximo en el modo gratuito).
Esto provoca que para según qué tipo de aplicación los tiempos de respuesta para la detección y clasificación de los objetos estén fuera del rango exigible.

Para abordar esta problemática en trabajos futuros se pueden seguir dos líneas de actuación:

1. Descargar la red neuronal entrenada desde Azure Cutom Vision como una imagen de Docker y ejecutarla en un dispositivo con el motor de tiempo de ejecución de Azure IoT Hub instalado.[63]

[63] https://docs.microsoft.com/en-us/azure/iot-edge/tutorial-deploy-custom-vision?view=iotedge-2020-11

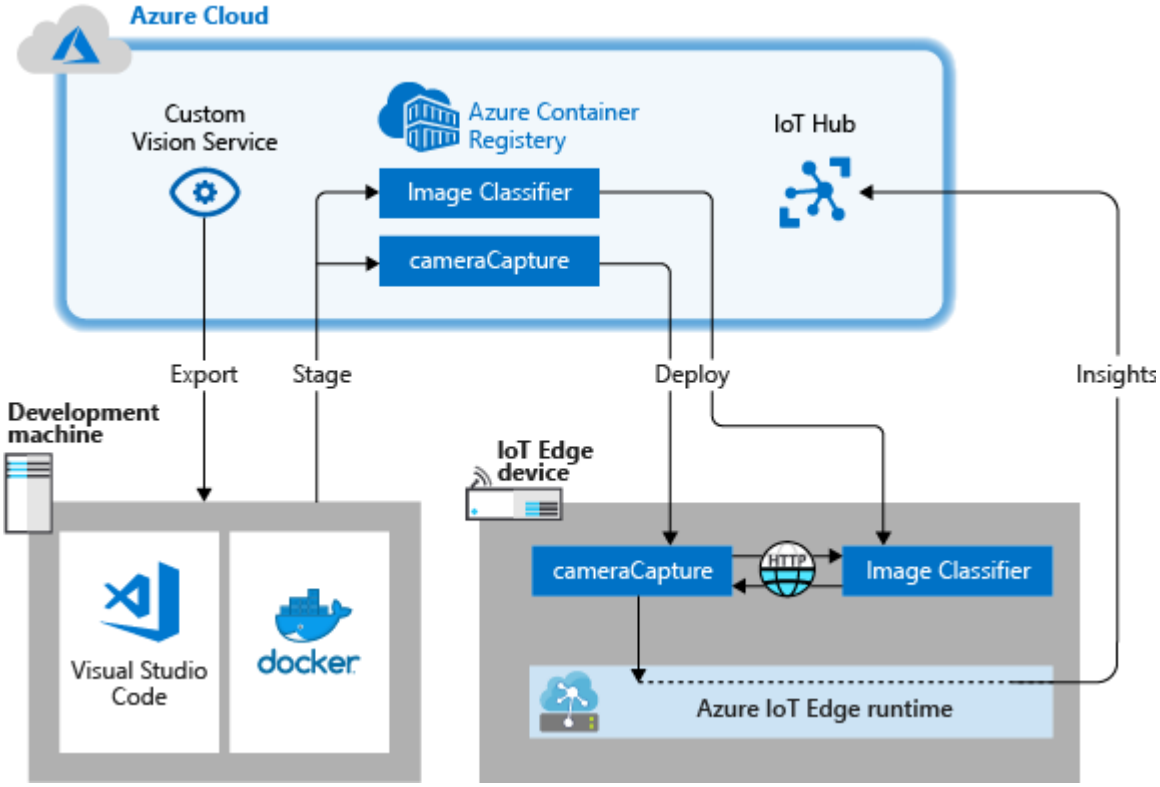


Figura 10-1 Crear un clasificador de imágenes con Custom Vision. Fuente Azure.

2. Utilizar un modelo de detección de objetos basado en DL compacto como YOLO[64] y dockerizarlo[65] para correr directamente en un dispositivo Siemens Industrial Edge sin necesidad de conexión exterior.

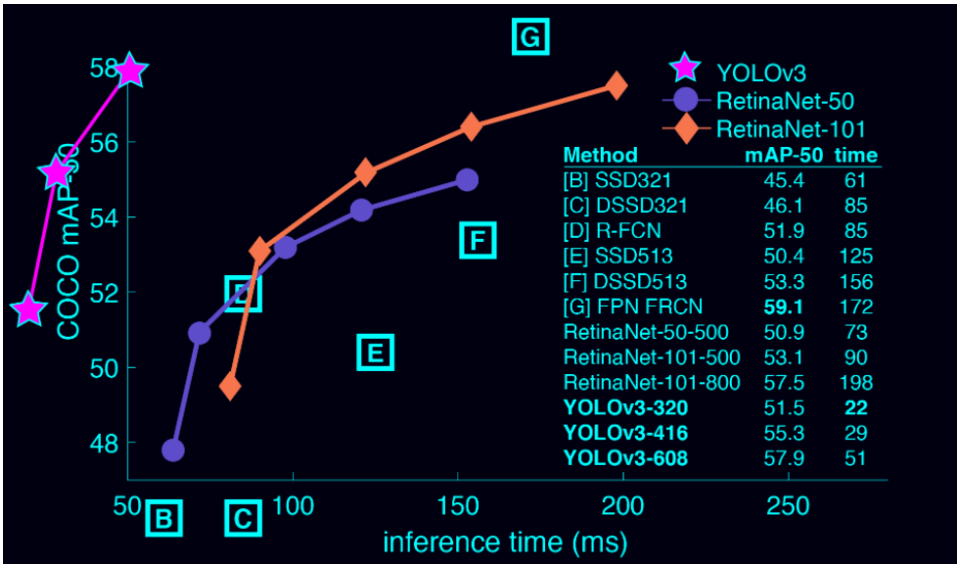


Figura 10-2 Comparación de YOLO con otros detectores. Fuente YOLO Darknet

[64] https://pjreddie.com/darknet/yolo/

[65] https://github.com/tancnle/docker-darknet

Presupuesto

En la medida de lo posible para la realización del presente proyecto se ha optado por utilizar software de libre distribución (Visual Studio Code) o licencias Trial (TIA Portal V17) o de estudiante (Azure Custom Vision, Docker Compose), por lo que los costes reales en los que se ha incurrido están muy por debajo de los que se indican a continuación. Pero se ha creído conveniente reflejar el coste real que hubiera tenido en el caso de desarrollar un prototipo para producción. En el caso de Azure Custom Vision al tratarse de un coste recurrente y en función del volumen de trabajo del sistema se ha creído conveniente simplemente reflejar la tabla de precios de la herramienta.

Costes Azure Custom Vision:

Instancia	Transacciones por segundo (TPS)	Características	Precio
Gratis	2 TPS	Transacciones de carga, entrenamiento y predicción	5.000 imágenes entrenadas gratis por proyecto
		Hasta 2 proyectos	10.000 predicciones al mes
		Hasta 1 hora de entrenamiento al mes	
Estándar	10 TPS	Transacciones de predicción	**€1,802** por 1000 transacciones
		Hasta 100 proyectos	
		Formación	**€9,007** por hora de proceso
		Almacenamiento de imágenes	**€0,631** por 1.000 imágenes
		Hasta 6 MB cada unidad	

Presupuesto estación de ingeniería y gemelo digital:

Referencia	Descripción	Precio unitario
6ES7718-1CB10-0BC2	SIMATIC Field PG M6 Advanced "i7-8850H (2,6 hasta 4,3 GHz; 6 cores + Hyper-Threading; 9 MB smart cache); 15,6"" Display, Full HD (1920 x 1080); WLAN 802.11ac & Bluetooth v5.0; DVD +/-RW;UHD Graphics 630" 1x 32GB DDR4 SDRAM SO-DIMM 512 GB SSD SATA (2,5") Con interfase S5-online y S5-EPROMMER; incl. Licencia STEP 5, cable S5-AG y adaptador S5-EPROM Windows 10 Enterprise LTSC 2019 64-Bit; "STEP 7 & WinCC & Safety Combo: STEP 7 Prof. Combo (V16 & 2017 SR1), WinCC Adv. Combo (V16 & flex. 2008 SP8), Safety Adv. Combo (V16 & distrib. V5.4 SP5)" "V17.0.0.0: STEP 7 Prof. V17; WinCC Adv. V17; Safety Adv V17; STEP 7 Prof. 2017 SR1; WinCC flex. 2008 SP8; Distributed Safety V5.4 SP5; STEP 5 V7.23 HF2 (inkl. Graph 5/II V7.15)	5.974,12 €
6ES7823-1FA03-0YA5	SIMATIC S7-PLCSIM Advanced V4.0	2.805,00 €
PLM-NX-V12-MCD	Siemens PLM, Software de diseño CAD NX, versión 12 + opción Mechatronic Concept Design (MCD)	3.200,00 €
	Total:	**11.979,12 €**

Presupuesto hardware del sistema de control de la línea:

Referencia	Descripción	Precio unitario
6ES7 516-3FN02-0AB0	CPU 1516F-3PN/DP Fail-safe CPU with display; work memory 1.5 MB code and 5 MB data; can be used for safety applications; supports consistent safety upload; supports PROFIsafe V2; 10 ns bit operation time; 5-stage protection concept, technology functions: motion control, closed-loop control, counting and measuring; tracing; Runtime options; isochronous mode (central); for all PROFINET interfaces: transport protocol TCP/IP.	5.466,30 €
6AV2 128-3GB06-0AX1	MTP700 Unified Comfort Panel 7.0" TFT display, 800 x 480 pixels, 16M colors; Multi touch; 1 x 422/485, 1 x PROFINET/Industrial	1.674,40 €

	Ethernet interface with MRP (2 Ports); 1 x Ethernet (Gigabit); 2 x SD card slot; 4 x USB	
6AV2170-2BA00-0AA0	EDGE Runtime for Unified Comfort (Device-managed)	226,00 €
TP-Link82760	Tapo C100 TP-Link TAPO C110 - Cámara Wi-Fi Interior, Resolución 3MP, Visión Nocturna Avanzada, Audio bidireccional, notificaciones en Tiempo Real, Compatible con Alexa y Google Home	30,00 €
	Total:	**7.396,70 €**

Coste de personal

		Tiempo[h]/Tarea	Coste [€]/hora	Total
Especificación detallada	1	32	100	3200
Gestión de proyectos	1	8	100	800
Preparación estación de ingeniería	1	8	50	400
Desarrollo app vision	1	100	50	5000
Desarrollo programa HMI/PLC	1	32	50	1600
Implementación gemelo digital	1	16	50	800
Puesta en marcha	1	16	75	1200
Pruebas FAT	1	16	75	1200
Documentación	1	100	15	1500
			Total:	**15.700 €**

Coste total:

A continuación, se indica el presupuesto con el coste total que supondría el desarrollo del proyecto.

Para el cálculo del coste de las licencias de ingeniería aplicadas a este proyecto se ha estimado una amortización lineal del coste de dichas licencias en tres años.

A las distintas partidas consignadas se le incrementa el IVA correspondiente.

	Cantidad	Coste sin IVA	IVA (21%)	Total con IVA
Coste material hardware	1	7.396,70 €	1.553,307 €	8.950 €
Coste licencias de ingeniería	0,33	11.979 €	839 €	4.832 €
Coste de personal	1	15.700 €	3.297 €	18.997 €
			Total:	**32.779 €**

Bibliografía y referencias

Bibliografía consultada

- Mathew Salvaris, Danielle Dean, Wee Hyong Tok, "**Deep Learning with Azure**" Apress, 2018, doi: https://doi.org/10.1007/978-1-4842-3679-6

- Fadi Al-Turjman, "**Edge Computing**", Springer 2019, doi: https://doi.org/10.1007/978-3-319-99061-3

- Siemens Industry Online Support (ID de artículo 109795865, Fecha del artículo: 05/03/2021). **Industrial Edge - App Developer Guide V1.2.1** [Online]. Available: https://support.industry.siemens.com/cs/es/en/view/109795865

- Siemens Industry Online Support (ID de artículo 109804671, Fecha del artículo: 12/14/2021). **SIMATIC HMI Unified Comfort Panels Industrial Edge Device – Operation** [Online]. Available: https://support.industry.siemens.com/cs/es/en/view/109804671

- Siemens Industry Online Support (ID de artículo 109778823, Fecha del artículo: 03/09/2021). **Siemens Industrial Edge Publisher for Unified Comfort Panels** [Online]. Available: https://support.industry.siemens.com/cs/es/en/view/109778875

- Siemens Industry Online Support (ID de artículo 109778875, Fecha del artículo: 03/24/2020). **SIMATIC HMI WinCC Unified Open Pipe** [Online]. Available: https://support.industry.siemens.com/cs/es/en/view/109778823

- Siemens Industry Online Support (ID de artículo 109781136, Fecha del artículo: 30/10/2020). **Industrial Edge** [Online]. Available: https://support.industry.siemens.com/cs/es/es/view/109781136

- Docker docs. **Docker overview** [Online]. Available: https://docs.docker.com/get-started/overview/

- Microsoft Docs. **Entrenamiento distribuido de modelos de aprendizaje profundo en Azure.** [Online]. Available: https://docs.microsoft.com/es-es/azure/architecture/reference-architectures/ai/training-deep-learning

- Microsoft Docs. **Documentación sobre Custom Vision.** [Online]. Available: https://docs.microsoft.com/es-es/azure/cognitive-services/custom-vision-service/

- **Visual Studio Code Docs. Getting Started.** [Online]. Available: https://code.visualstudio.com/docs

- **Flask, User´s Guide.** [Online]. Available: https://flask.palletsprojects.com/en/2.0.x/

Referencias:

[1] **Edge computing.** Wikipedia [Online]. Available: https://es.wikipedia.org/wiki/Edge_computing

[2] **Computación distribuida.** Wikipedia [Online]. Available: https://es.wikipedia.org/wiki/Computaci%C3%B3n_distribuida

[3] GSMA Intelligence. "**Understanding 5G: Perspectives on future technological advancements in mobile"** (2014) [Online]. Available: https://www.gsma.com/futurenetworks/wp-content/uploads/2015/01/2014-12-08-c88a32b3c59a11944a9c4e544fee7770.pdf

[4] Docker (software). Wikipedia. [Online]. Available: https://es.wikipedia.org/wiki/Docker_(software)

[5] **Docker Partners**. Docker [Online]. Available: https://www.docker.com/partners

[6] Siemens Industry Image Database 4.11 [Online]. Available: https://www.automation.siemens.com/bilddb/index.aspx?lang=en

[7] **Docker overview** [Online]. Available: https://docs.docker.com/get-started/overview/

[8] Nilesh Jayanandana, "**Build a Docker Image just like how you would configure a VM**". Platformer. [Online]. Available: https://medium.com/platformer-blog/practical-guide-on-writing-a-dockerfile-for-your-application-89376f88b3b5

[9] **Comparing Containers and Virtual Machines** [Online]. Available: https://www.docker.com/resources/what-container/

[10] Infaimon, "**Enciclopedia de la Visión**". [Online]. Available: https://infaimon.com/enciclopedia-de-la-vision/

[11] Siemens Industry Online Support (ID de artículo 109766012, Fecha del artículo: 07/09/2019). **Procesamiento de imagen industrial en SIMATIC IPC** [Online]. Available: https://support.industry.siemens.com/cs/es/en/view/109781417/149110787083

[12] **Apuntes Visión Artificial**, Grado Ingeniería en Electrónica de Comunicaciones UAH. Tema 5, pág. 3.

[13] **"Azure Machine Learning documentation."** [Online]. Available: https://docs.microsoft.com/en-us/azure/machine-learning/algorithm-cheat-sheet

[14] **"Evaluate Model component"**. Azure Machine Learning Documentation. [Online]. Available:

https://docs.microsoft.com/en-us/azure/machine-learning/component-reference/evaluate-model

[15] **Apuntes Visión Artificial**, Grado Ingeniería en Electrónica de Comunicaciones UAH.

[16] **Apuntes Visión Artificial**, Grado Ingeniería en Electrónica de Comunicaciones UAH.

[17] "**Deep learning vs. machine learning in Azure Machine Learning**" [Online]. Available: https://docs.microsoft.com/en-us/azure/machine-learning/concept-deep-learning-vs-machine-learning

[18] "**Deep learning vs. machine learning in Azure Machine Learning**" [Online]. Available: https://docs.microsoft.com/en-us/azure/machine-learning/concept-deep-learning-vs-machine-learning

[19] **Visión Artificial**, Grado Ingeniería en Electrónica de Comunicaciones UAH.

[20] **Visión Artificial**, Grado Ingeniería en Electrónica de Comunicaciones UAH.

[21] "**Deep Learning Framework Examples**" [Online]. Available: http://bit.ly/DLComparisons

[22] "**Deep Learning Framework Examples**" [Online]. Available: http://bit.ly/DLComparisons

[23] Orhan G. Yalçın, Towards Data Science. "**Top 5 Deep Learning Frameworks to Watch in 2021 and Why TensorFlow**" [Online]. Available: https://towardsdatascience.com/top-5-deep-learning-frameworks-to-watch-in-2021-and-why-tensorflow-98d8d6667351

[24] **Características de AWS IoT Greengrass** [Online]. Available: https://aws.amazon.com/es/greengrass/features/?pg=ln&sec=hs

[25] **Precios de AWS IoT Greengrass** [Online]. Available: https://aws.amazon.com/es/greengrass/pricing/

[26] **Azure Marketplace** [Online]. Available: https://azure.microsoft.com/es-es/marketplace/

[27] **Azure Certified Device catalog** [Online]. Available: https://devicecatalog.azure.com/

[28] **Azure IoT Edge** [Online]. Available: https://github.com/Azure/iotedge

[29] **Azure IoT Edge** Github [Online]. Available: https://github.com/Azure/iotedge

[30] **Precios de Azure IoT Edge** [Online]. Available: https://azure.microsoft.com/es-es/pricing/details/iot-edge/

[31] **Precios de Azure IoT Edge** [Online]. Available: https://azure.microsoft.com/es-es/pricing/details/iot-edge/

[32] **Operación extendida sin conexión con Azure IoT Edge** [Online]. Available: https://azure.microsoft.com/es-es/blog/extended-offline-operation-with-azure-iot-edge/

[33] **Uso de funcionalidades sin conexión ampliadas en dispositivos, módulos y dispositivos secundarios IoT Edge** [Online]. Available: https://docs.microsoft.com/es-es/azure/iot-edge/offline-capabilities?view=iotedge-2020-11

[34] Siemens Industry Online Support (ID de artículo 109781417, Fecha del artículo: 09/28/2021) [Online]. Available: **Edge app Data Service for Industrial Edge** V1.3 https://support.industry.siemens.com/cs/es/en/view/109781417/149110787083

[35] Siemens Industry Online Support (ID de artículo 109795865, Fecha del artículo: 05/03/2021). **Industrial Edge - App Developer Guide V1.2.1** [Online]. Available: https://support.industry.siemens.com/cs/es/en/view/109795865

[36] **Redes en Docker: exposición de puertos.** Colaboratorio.[Online]. Available: https://colaboratorio.net/davidochobits/sysadmin/2017/redes-en-docker-exposicion-de-puertos/

[37] **Python Tutorial for Beginners: Learn Python Quickly.** PythonLand.[Online]. Available: https://python.land/python-tutorial/what-is-python

[38] **Jupyter.** [Online]. Available: https://jupyter.org/

[39] **Our Company.** Docker. [Online]. Available: https://www.docker.com/company/

[40] **Express.** [Online]. Available: https://expressjs.com/

[41] **FeathersJS** [Online]. Available: https://docs.feathersjs.com/

[42] **Fastify** [Online]. Available: https://www.fastify.io/

[43] **Koa.** [Online]. Available: https://koajs.com/#introduction

[44] **About Django.** [Online]. Available: https://www.djangoproject.com/

[45] **What is Angular?** [Online]. Available: https://angular.io/guide/what-is-angular

[46] **Armin Ronacher.** Wikipedia. [Online]. Available: https://es.wikipedia.org/wiki/Armin_Ronacher

[47] **Python Software Foundation Supporting Member** [Online]. Available: https://psfmember.org/

[48] **Werkzeug Getting Started** [Online]. Available: https://werkzeug.palletsprojects.com/en/2.0.x/

[49] **Jinja.** Pallets projects [Online]. Available: https://palletsprojects.com/p/jinja/

[50] **Welcome to Click.** Pallets projects [Online]. Available: https://click.palletsprojects.com/en/8.0.x/

[51] Manual Docker Engine. "**Install Docker Engine on Ubuntu".** [Online]. Available: https://docs.docker.com/engine/install/ubuntu/

[52] Ignacio Van Droogenbroeck, "**Pique #19: Cómo exponer la API de Docker". cduser.com** [Online]. Available: https://cduser.com/pique-19-como-exponer-la-api-de-docker/

[53] **Dockerfile reference** [Online]. Available: https://docs.docker.com/engine/reference/builder/

[54] **Node.js.** Docker, Official Images [Online]. Available: https://hub.docker.com/_/node/

[55] **No se pueden instalar paquetes pip dentro de un contenedor docker con Ubuntu**. QA Stack. [Online]. Available: https://qastack.mx/programming/28668180/cant-install-pip-packages-inside-a-docker-container-with-ubuntu

[56] **Best practices for writing Dockerfiles.** Develop with Docker. [Online]. Available: https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#use-multi-stage-builds

[57] Chris Richardson, **Pattern: Microservice Architecture**. Microservice Architecture [Online]. Available: https://microservices.io/patterns/microservices.html

[58] Siemens Industry Online Support (ID de artículo 109778823, Fecha del artículo: 24/03/2020)
[Online]. Available: **SIMATIC HMI WinCC Unified Open Pipe**
https://support.industry.siemens.com/cs/es/es/view/109778823

[59] Siemens Industry Online Support (ID de artículo 109792717, Fecha del artículo: 08/11/2021). **Kit de desarrollo Data Service para Industrial Edge**
[Online]. Available:
https://support.industry.siemens.com/cs/es/es/view/109792717

[60] **Supported logging drivers.** Docker docs. [Online]. Available: https://docs.docker.com/config/containers/logging/configure/#supported-logging-drivers

[61] Santi Macias, "**Aplicaciones inteligentes con Microsoft Cognitive Services y Azure. Enmilocalfunciona**".
[Online]. Available: https://enmilocalfunciona.io/aplicaciones-inteligentes-con-microsoft-cognitive-services/

[62] Siemens Industry Online Support (ID de artículo 109758943, Fecha del artículo: 06/10/2020). **SIMATIC Machine Simulator Virtual commissioning of machines Getting Started**
[Online]. Available:
https://support.industry.siemens.com/cs/es/en/view/109758943

[63] **Tutorial: Perform image classification at the edge with Custom Vision Service**
[Online]. Available: https://docs.microsoft.com/en-us/azure/iot-edge/tutorial-deploy-custom-vision?view=iotedge-2020-11

[64] **YOLO: Real-Time Object Detection** [Online]. Available:
https://pjreddie.com/darknet/yolo/

[65] **YOLO Object Detection with Darknet** [Online]. Available:
https://github.com/tancnle/docker-darknet

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá